UTN FACULTAD REGIONAL MENDOZA
UNIVERSIDAD TECNOLÓGICA NACIONAL

Proyecto Final Junkode - Documentador Automático de Código Fuente

Universidad Tecnológica Nacional.
Facultad Regional Mendoza.
Ingeniería en Sistemas de Información.

Autores:

- Céspedes Ortega, Rodrigo Gabriel.
  DNI: 42082114. Legajo: 45185.
- Fernández Quatrini, Renzo Abel.
  DNI: 42669442. Legajo: 45352.
- Flores, Sebastián Andrés.
  DNI: 42478387. Legajo: 45241.
- Girala, Ramsés.
  DNI: 42167630. Legajo: 45106.
- Groisman, David Nathaniel.
  DNI: 36859670. Legajo: 38492.

Directores del Trabajo:
Vazquez, Alejandro.
Moralejo, Raul.
Manino, Gustavo.
Lemos, Gustavo.
Casas, Malena.

Año Académico 2022

Mendoza, 22 de Noviembre de 2022.

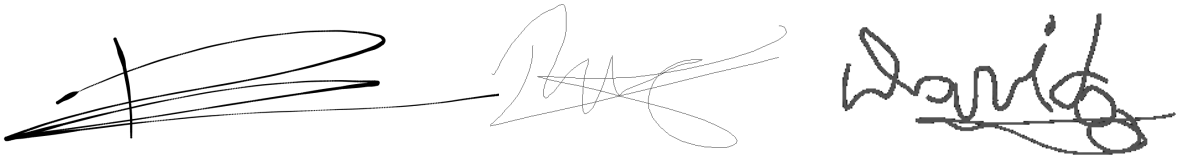
Biblioteca del Departamento de Ingeniería en Sistemas de Información.
Universidad Tecnológica Nacional.
Facultad Regional Mendoza.
S_____/_____D

Nos dirigimos con el fin de realizar la entrega del trabajo final de la carrera Ingeniería en Sistemas de Información.

El proyecto llevado a cabo como trabajo final de la carrera es "Junkode - Documentador Automático de Código Fuente".

La documentación presentada es la carpeta completa que incluye: Desarrollo de un sistema de información real, Diagrama GANTT e Histogramas de Recursos del Proyecto, Registro de reuniones modelo del equipo, Tabla de riesgos, Diagrama de clases del negocio de Junkode, Modelo de datos de junkode, Diagramas BPMN, Pantallas y reportes de Junkode, Trabajo Práctico Integrador "Dirección de Proyectos de Sistemas", Trabajo Práctico Integrador "Gerenciamiento de Sistemas".

Sin otro particular, nos despedimos cordialmente.



Céspedes Ortega, Rodrigo Gabriel. (45185)

Fernández Quatrini, Renzo Abel. (45352)

Groisman, David Nathaniel. (38492)

Girala, Ramsés. (45106)

Flores, Sebastián Andrés. (45241)

Resumen:

En la presente carpeta, se expone el proyecto desarrollado en concepto de Proyecto Final de la carrera de Ingeniería en Sistemas de Información, que consiste en una herramienta open-source que puede realizar análisis de código Java 8 en adelante para obtener reportes con documentación y métricas del mismo código de manera automática. Adicionalmente, puede vincularse directamente con una cuenta de usuario de GitHub para acceder a sus repositorios y llevar un historial con el avance de la documentación y métricas de un proyecto a lo largo de la vida del mismo. Desarrollado con ANTLRv4 en .NET, (C#) y desplegado en la nube en Kubernetes, Junkode es una herramienta perfecta para la customización y análisis de código fuente por parte de cualquier desarrollador interesado. Este es el motivo que llevó a que se tomara la decisión de desarrollar Junkode como una herramienta libre y abierta que pueda aportar una disminución notoria del tiempo implicado en el desarrollo de documentación de código fuente y, también, una oferta para el análisis de calidad y control evolutivo del mismo.

Palabras Clave:

- Métricas orientadas a objetos.
- Documentación de código.
- Seguimiento de calidad de software.
- Open-source.
- Mantenimiento de software.

Índice:

DESARROLLO DE UN SISTEMA DE INFORMACIÓN REAL

Definición de Requerimientos

1.Introducción al análisis de Requerimientos.

En el siguiente análisis de requerimientos, se presentan 5 herramientas de Análisis Estático de Código, en las cuales se basa el proyecto Junkode. Estas herramientas son: nDepends, Kiuwan, SonarQube, DeepScan y Veracode, que a continuación se describen en forma general y detallada.

Todas las herramientas a continuación realmente realizan un trabajo común que es el análisis de código estático, por lo que antes de comenzar a detallar estas herramientas primero se debe saber qué es un analizador estático de código:

Un analizador estático de código comprueba el código fuente para determinadas propiedades tales como la conformidad con estándares de codificación, métricas de calidad, seguridad o anomalías en el flujo de datos.

Con lo último definido, se explican en forma general las herramientas mencionadas.

2.Marco Teórico.

La funcionalidad de generación de documentación de código fuente que ofrece Junkode tiene como objetivo facilitar la tarea de mantenimiento de software por medio del análisis de código con ingeniería inversa.

Este punto se centra en facilitar el trabajo de quien realice el mantenimiento del código, puesto que en segundos podrá tener acceso a documentación certera en un formato amigable para comprender el código que debe mantener. Ya no solo hablando del diagrama de clases en sí, sino de que patrones se utilizan, los datos de las bases de datos, servicios externos, contenedores y más información, que ahorra, en función del tamaño del proyecto, días o semanas de cansador análisis de código fuente.

Esta funcionalidad que ofrece Junkode es especialmente útil en los momentos actuales y (también de cara al panorama futuro). Con cada vez más proyectos siendo desarrollados con metodologías AGILES, donde se focaliza la atención en codificar los entregables y no en su documentación, una herramienta como Junkode puede aportar un valor increíblemente significativo para estos proyectos, ahora podrán los desarrolladores centrarse en las entregas y dejar que la documentación la genere Junkode cuando sea necesario.

Por otro lado, es correcto pensar que la medición dentro de la ingeniería de software, es un tanto joven. La métrica más utilizada y relevante es la llamada Complejidad Ciclomática, esta métrica define el número de caminos independientes dentro de un fragmento de código

para proporcionar un indicador cuantitativo de la complejidad lógica de un programa. En general, si es menor o igual a 2 la complejidad lógica es correcta, si es mayor a 2 y menor o igual a 4 es regular, si es mayor a 4 y menor o igual a 7 es malo, y si es mayor a 7 es inaceptable.

Las métricas para java, al estar basado este lenguaje en la programación orientada a objetos, hacen hincapié en el encapsulamiento, la herencia y el polimorfismo, ya que son estos conceptos los que distinguen a un programa orientado a objetos de cualquier otro. Por lo tanto, las métricas orientadas a objetos se centran en métricas que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a una clase concreta.

La segunda funcionalidad de la herramienta se encarga de analizar esta información para cada clase del código, para así poder evaluar la calidad del mismo.

3.nDepends.

CONTROLA LA CALIDAD DE TU CÓDIGO CON NDEPEND. (s.f.). Obtenido de https://albertcapdevila.net/ndepend/.

3.1.Relevamiento General.

3.1.1.De la Organización.

NDepend es un analizador de código que genera informes navegables con métricas objetivas sobre la calidad del código analizado. Cada métrica está documentada y es monitoreable y editable. Además, es capaz de evaluar la deuda técnica del proyecto en días y horas: se puede crear un histórico de métricas y evaluar a medida que se avanza en el proyecto, si el código añadido ha generado o no más deuda técnica.

3.1.2.Funciones detectadas a nivel general y relaciones con otros Sistemas y Entidades.

- ➢ Análisis de Código:

Esta herramienta tiene una variedad de tipos de análisis de código, como los code smells, code legacy, entre otras; que brinda una estimación objetiva de la deuda técnica y los errores o “warnings” del código programado.

- ➢ Log de Resultados de Análisis:

Posee gráficas visuales de los análisis realizados. Estas gráficas tienen distintos usos, y son bastante importantes cuando se habla de un proyecto a gran escala. Entre los gráficos se puede resaltar el Dashboard, que muestra una información general del código analizado con detalles específicos.

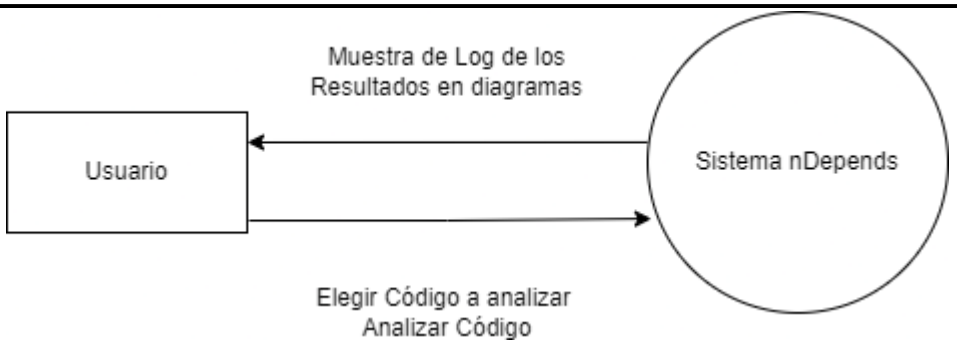


Diagrama 3.1.2.a: Diagrama de Interfaz de nDepends.

3.1.3.Tecnología de información.

nDepends se utiliza de 2 formas. Se puede instalar en el ordenador, donde tiene soporte para Windows, Linux y Mac. Esto permite poseer la aplicación corriendo en el sistema y poder elegir el código. La otra forma que posee, es la instalación de un plugin el Visual Code Studio, donde una vez instalado, se puede obtener un resultado rápido del test del código. Posee una integración con otras herramientas como Azure DevOps, herramientas de integración continua, como AppVeyor, entre otras. nDepends realiza un análisis de código para el framework .NET. La aplicación está desarrollada en C#.

3.2.Relevamiento detallado y análisis del Sistema.

3.2.1.Detalle, explicación y documentación detallada de todas las funciones seleccionadas.

- ➢ Análisis de Código:

Esta herramienta, como se explicó anteriormente, se utiliza igual de ambas formas. No se carga el código sobre la aplicación, si no que la aplicación corre sobre el código y a partir de esa instancia saca la información sobre el análisis.
A continuación, se observa cómo, desde una integración en Visual Studio Code, se instala. Se puede ver que en el menú en la parte superior figura un nDepends donde se puede realizar el análisis.

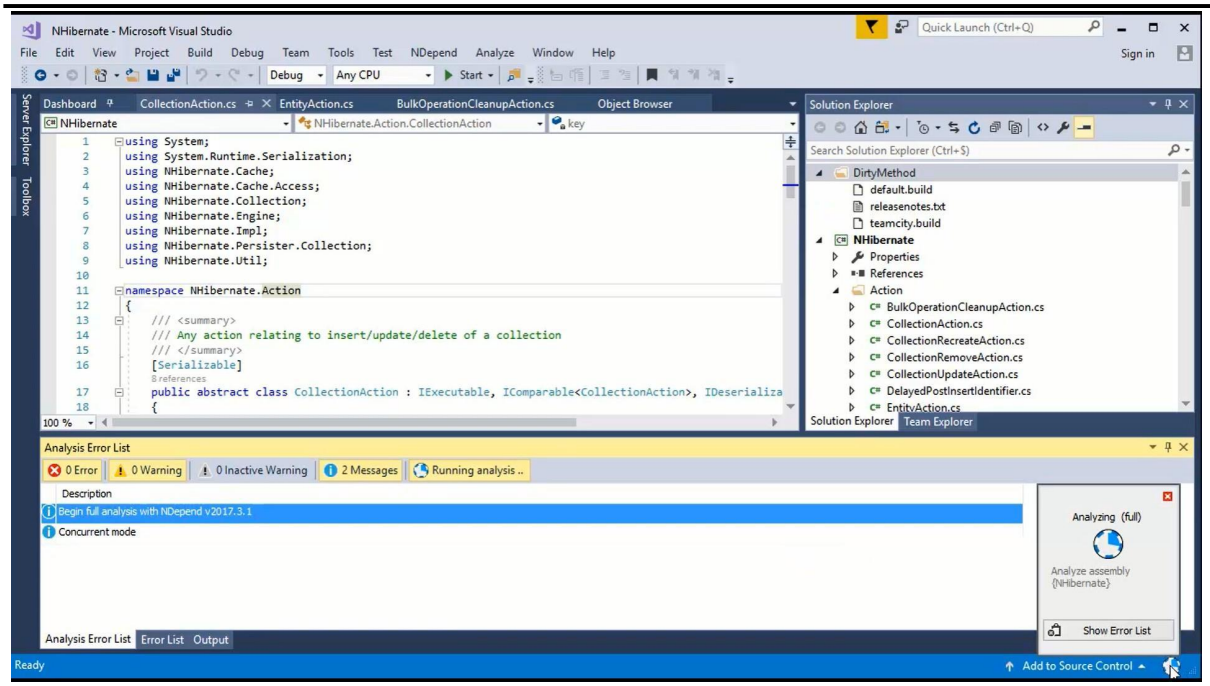


Figura 3.2.1.a: Captura de Análisis de nDepends.

➢ Log de Resultados de Análisis:

Una vez finalizado el análisis anterior, automáticamente se abre una pestaña de navegador donde se muestran los resultados del análisis. Aquí se puede observar distintas gráficas a seleccionar y un dashboard con la información más general del sistema.

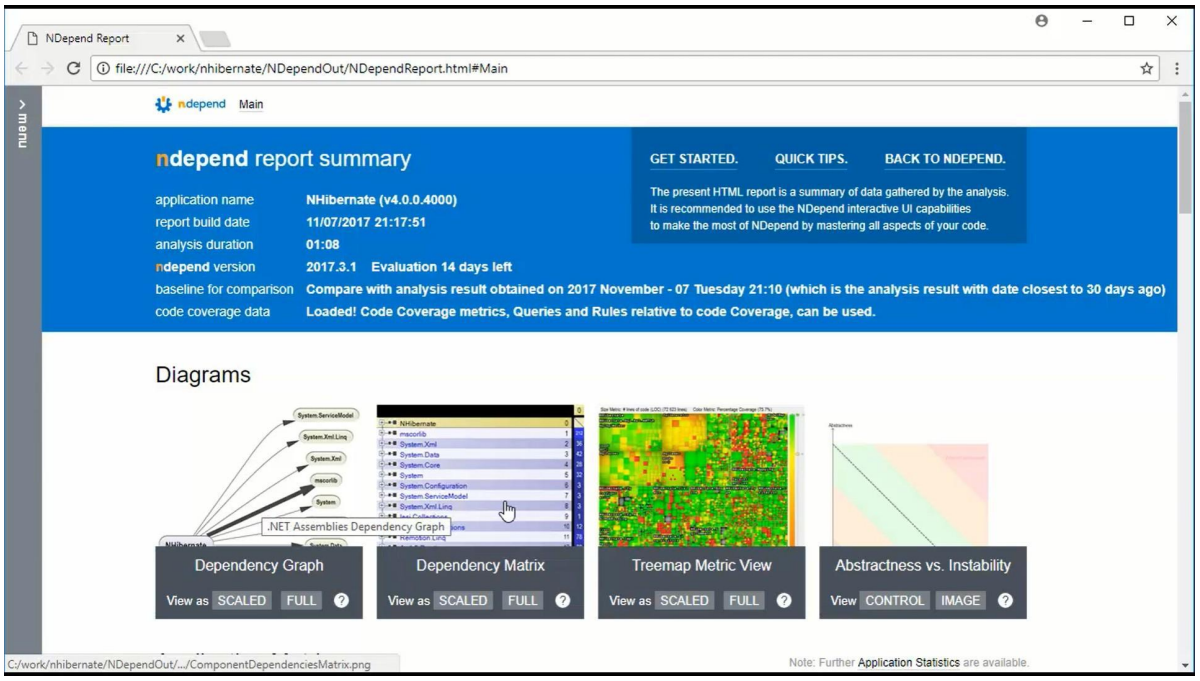


Figura 3.2.1.b: Captura de log de Resultado de Análisis nDepends.

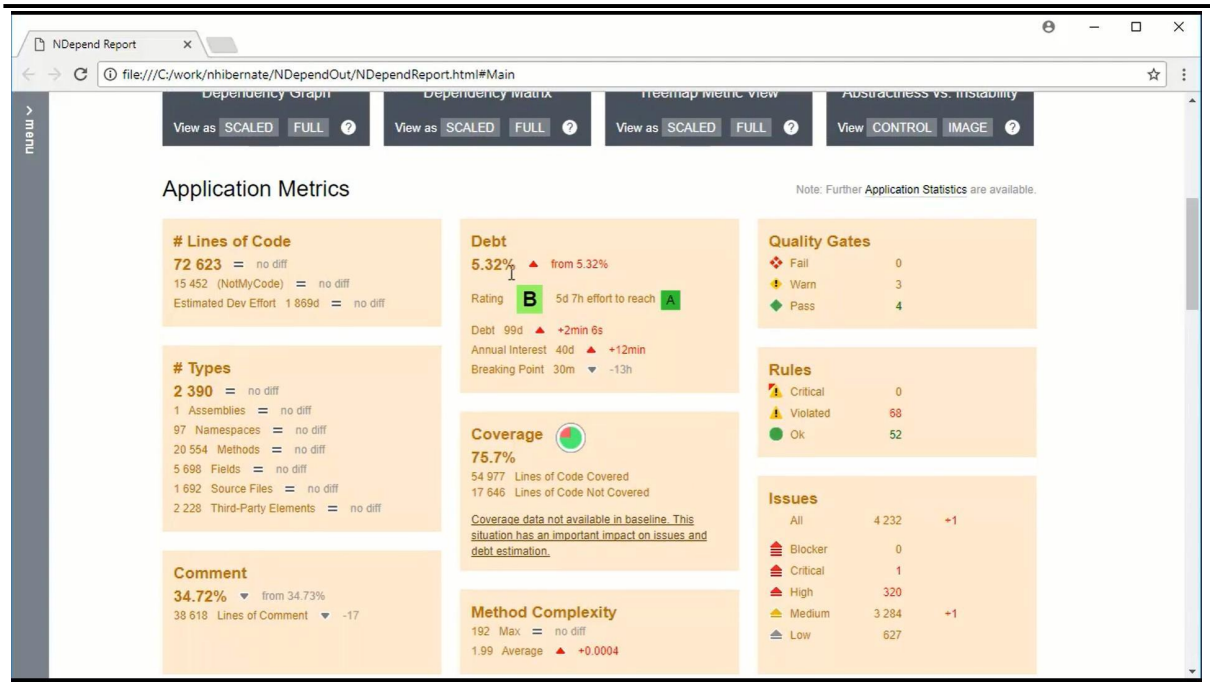


Figura 3.2.1.c: Dashboard de nDepends.

El dashboard (Figura 3.2.1.c) informa especificaciones generales del código analizado. Se destaca de las especificaciones la cantidad de líneas de código, el coverage, la deuda técnica, los comentarios, los errores graves y “warnings”, entre otras.

El programa brinda muchas gráficas que son de utilidad en proyectos de gran tamaño. Como, por ejemplo, la gráfica de dependencias, la matriz de dependencias, la vista de métricas, entre otras. A continuación, se explican las gráficas mencionadas.

En la vista de métricas, el código base se representa a través de un diagrama de árbol. Treemapping es un algoritmo de visualización para mostrar datos con estructura de árbol mediante el uso de una jerarquía de rectángulos anidados. La estructura de árbol utilizada en NDepend treemap es la jerarquía de código habitual. Los rectángulos de diagrama de árbol representan elementos de código. El nivel de opción determina el tipo de elemento de código representado por los rectángulos unitarios. El nivel de opción puede tomar los 5 valores: ensamblaje, espacio de nombres, tipo, método y campo. Las dos capturas de pantalla a continuación muestran la misma base de código que se muestra por nivel de método (a la izquierda) y nivel de espacio de nombres (a la derecha). Hay otros modos de visualizar el código, con colores para un mejor entendimiento.

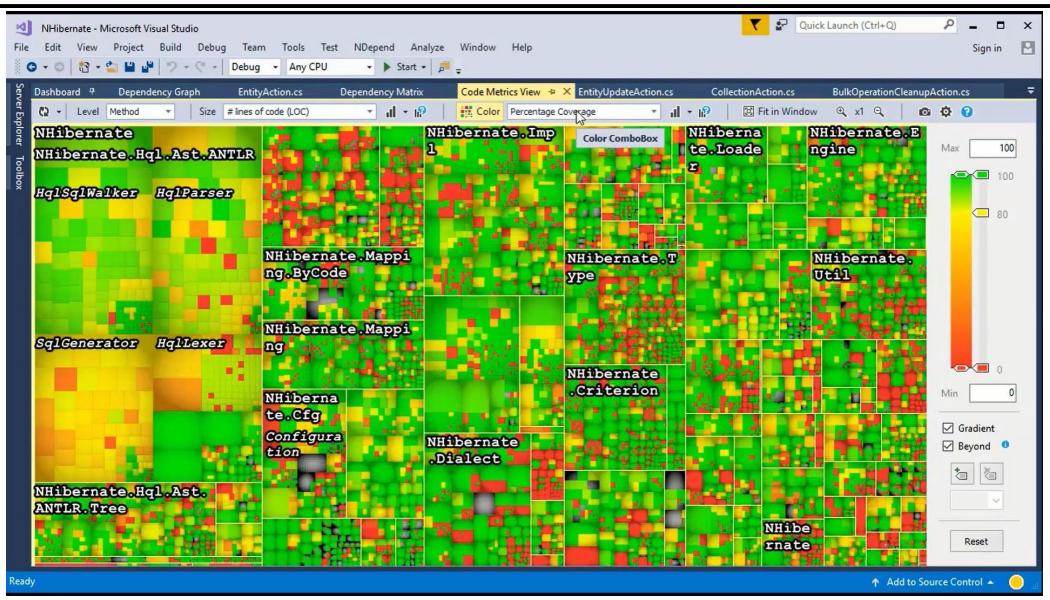


Figura 3.2.1.d: Vista de Métricas de nDepends.

El gráfico de dependencias visualiza la arquitectura de la aplicación y las dependencias entre sus componentes. Por lo que si del componente A sale una flecha que apunta al componente B, esto indica que el componente A tiene una dependencia con el componente B.

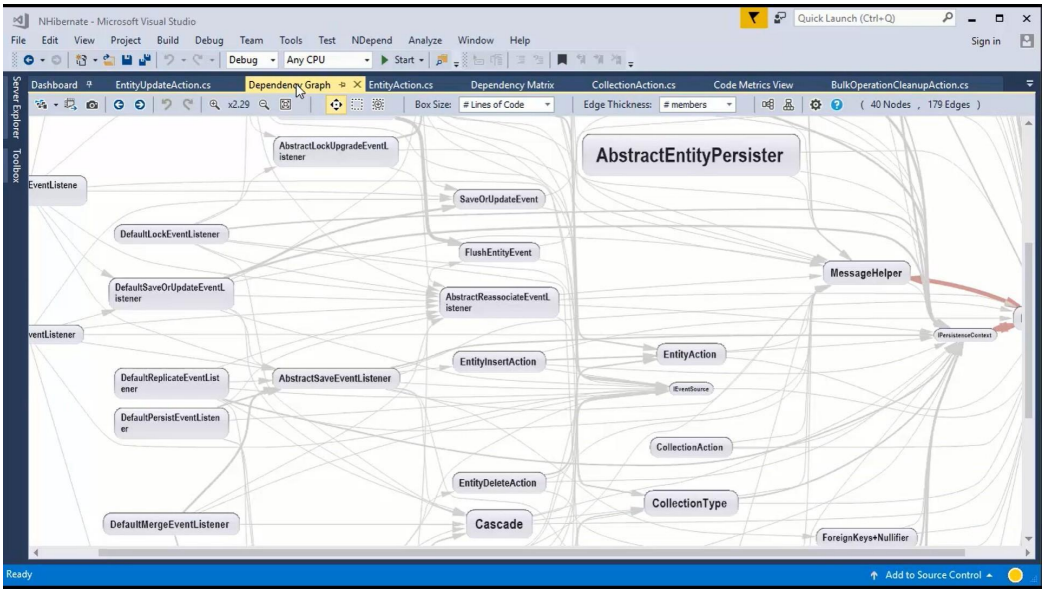


Figura 3.2.1.e: Gráfico de dependencias de nDepends.

La matriz de dependencia tiene el mismo objetivo que la gráfica de dependencias, pero tiene un mejor uso que el anterior en proyectos grandes ya que cuando el código es muy grande, el gráfico de dependencias deja de ser legible. En filas y columnas aparecen los mismos namespace. Las celdas no vacías representan las dependencias. Cuando se apoya el cursor sobre una celda de color azul, aparece una flecha señalando la fila la cual indica que el namespace de la columna utiliza el namespace de la fila.Si lo apoyamos sobre una celda

verde ocurre lo mismo, pero esta vez indicando que el namespace de la fila utiliza el namespace de la columna.El numero en las celdas son la cantidad de metodos y campos usados en el ensamblaje de los dos namespace.

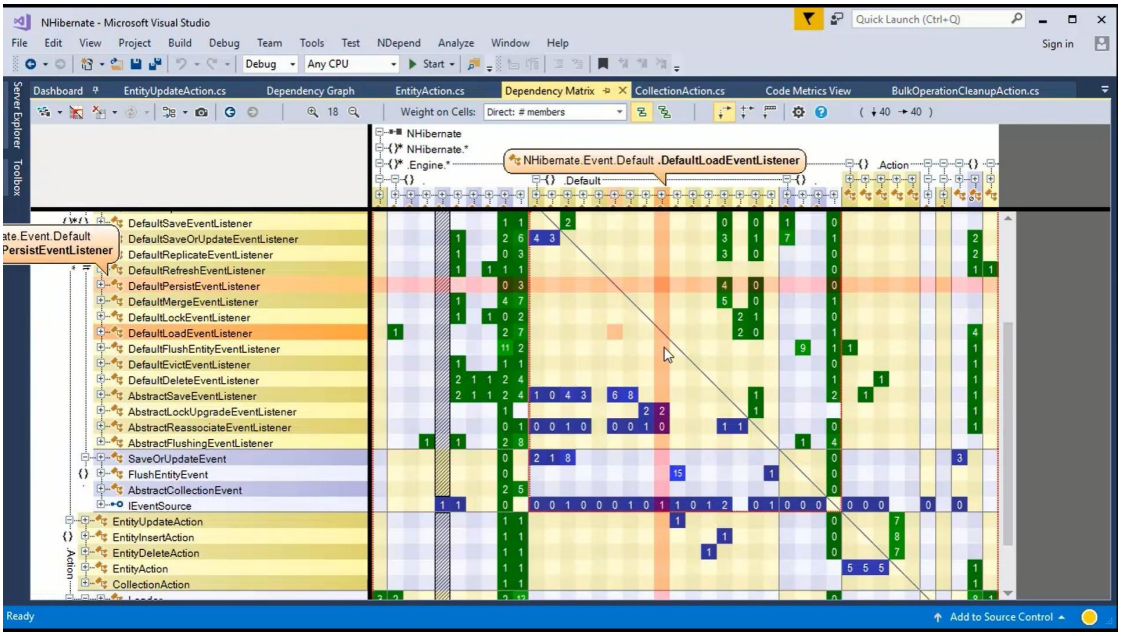


Figura 3.2.1.f: Matriz de dependencias de nDepends.

➢ Modelo lógico del sistema actual:

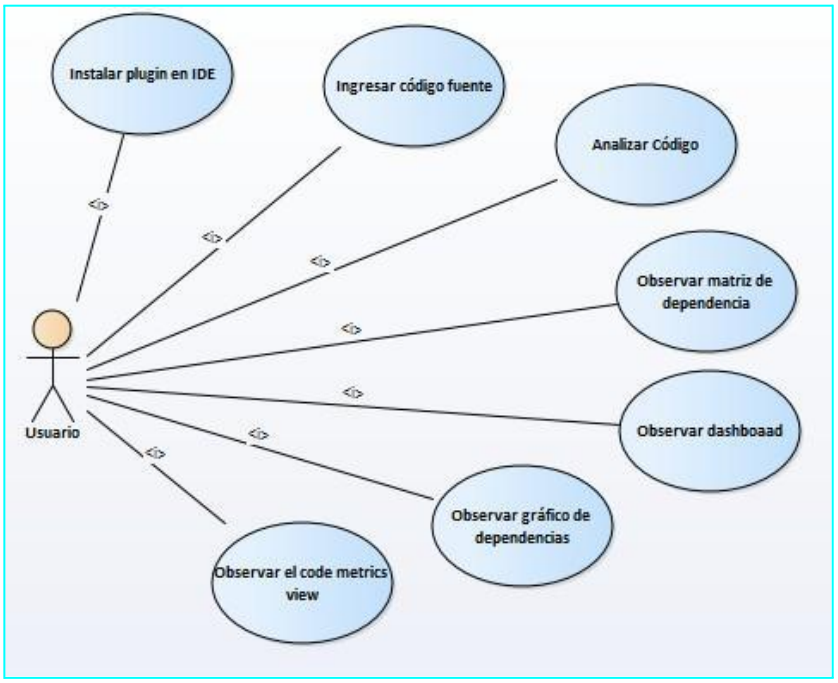


Diagrama 3.2.1.g: Modelo de casos de Uso de nDepends.

Problemas y necesidades detectados en las funciones relevadas en detalle y en su entorno organizacional

3.2.2.Problemas y Necesidades de nDepends:

- **Problema en el análisis de código:** Solamente analiza proyectos desarrollados en un solo framework, específicamente .NET.

- **Problema en el log de resultados del análisis:** Los diagramas son muy engorrosos a la vista, lo que dificulta el entendimiento de los mismos.

- **Necesidad en el análisis de código:** El analizador está orientado a proyectos de una gran escala, es decir que, para un proyecto de menor magnitud, no es una buena inversión y se pierde mucha utilidad en el producto en sí.

- **Necesidad en el log de resultados del análisis:** No hace ningún tipo de documentación del proyecto.

4.Kiuwan

¿Qué es Kiuwan? Analiza la seguridad y calidad de tu software. (s.f.). Obtenido de https://sentrio.io/blog/que-es-kiuwan/.
Kiuwan An Idea Inc. Company. (s.f.). Obtenido de https://www.kiuwan.com/

4.1.Relevamiento General.

4.1.1.De la Organización:

Kiuwan es una herramienta que se utiliza para analizar la seguridad de un código, de múltiples lenguajes. La misma se basa en los estándares NIST, MISRA, CWE, OWASP, que son estándares de ciberseguridad.

Una herramienta integral de seguridad de aplicaciones y aplicaciones web que puede ser utilizada por todas las partes interesadas en el proceso de desarrollo de software para potenciar la velocidad, minimizar la asignación de recursos y mitigar los compromisos de tiempo de desarrollo.

4.1.2.Funciones detectadas a nivel general y relaciones con otros Sistemas y Entidades.

- Login de Usuario.

Esta aplicación gestiona cuentas para poder utilizarse. Las 2 formas de utilizar el programa, pide una serie de datos personales para el uso de la herramienta.

- Load de código.

Para la carga de código, cuando se está trabajando en la nube, se tiene que subir el código a analizar en un archivo. Se tiene que descargar la aplicación al ordenador, y permite la carga del código. Este load de código, guarda los código por proyectos, y quedan en el

navegador.

➢ Análisis de código de vulnerabilidades.

Kiuwan Code Security, su solución SAST para analizar el código en busca de vulnerabilidades de seguridad

➢ Análisis de código de riesgos de componentes de terceros.

Insights Open Source, su solución SCA para controlar el riesgo de componentes de terceros y software de código abierto.

➢ Code Analysis (QA).

Code Analysis es la herramienta de testeo de la calidad del código de Kiuwan diseñada para desarrolladores e ingenieros QA.

➢ Governance y LifeCycle.

Kiuwan presenta también un módulo dedicado a la gobernanza y la gestión de la calidad a lo largo del ciclo de vida de la aplicación.

➢ Log de resultado del análisis.

Kiuwan muestra los resultados en el navegador del ordenador, con un dashboard amigable a la vista. Tiene varias tablas en para poder analizar donde lista las vulnerabilidades del código, y se pueden mostrar más detalladamente.

➢ Ver historial de análisis.

Posee una pequeña base de datos en la nube, donde de forma muy simple permite quedar un historial de proyectos con una cantidad limitada.

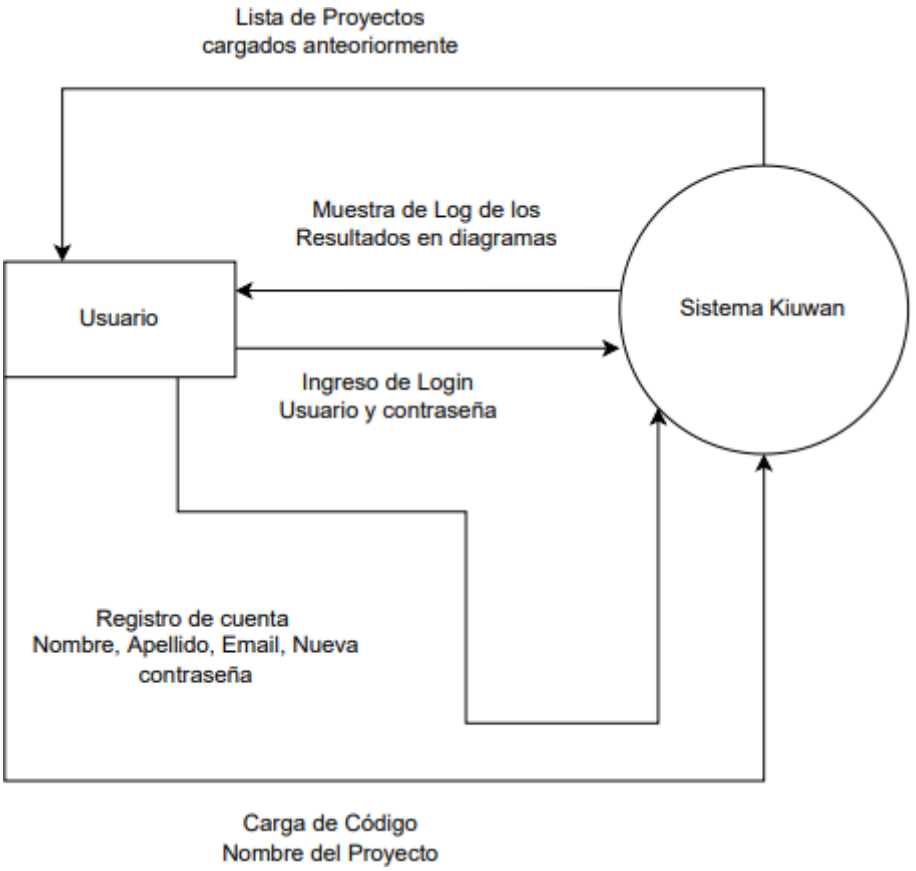


Diagrama 4.1.2.a: Diagrama de interfaces de Kiuwan.

4.1.3.Tecnología de información.

La misma posee dos formas de uso, una desde manera local mediante una pequeña aplicación descargable para Windows y Mac, que permite especificar la ruta del código a analizar. La otra manera es accediendo a una plataforma en la nube, la cual permite poder subir a internet el código que queremos analizar. Como se mencionó anteriormente, esta herramienta tiene 2 “productos” que son el Kiuwan Code Security y el Insights Open Source. Está desarrollada en Java.

4.2.Relevamiento detallado y análisis del sistema.

4.2.1.Detalle, explicación y documentación detallada de todas las funciones seleccionadas.

- ➢ Login de Usuario.

Al momento de utilizar la herramienta se pide una cuenta para iniciar sesión. Se solicita un usuario y contraseña. Luego se accede al sistema completo desde el navegador.

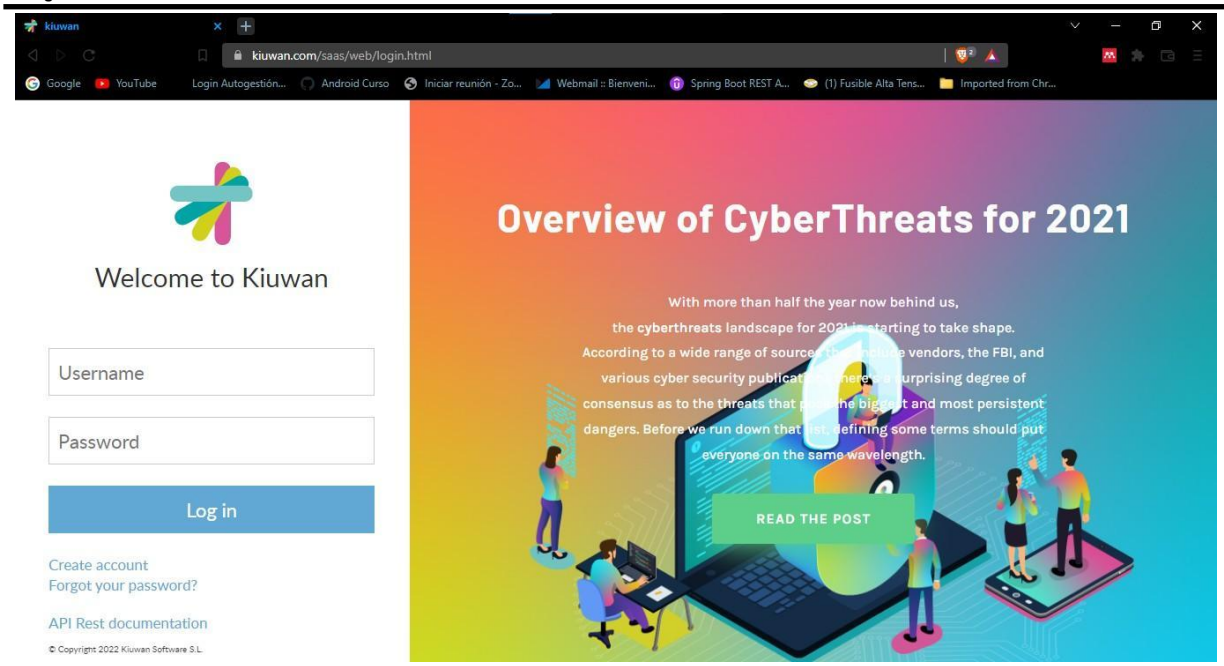


Figura 4-2.1.a: Captura de web de Inicio de Sesión de Kiuwan.

Para poder registrarse, pide nombre completo, correo electrónico, país de residencia, compañía y qué aplicación se desea comprar o testear con la demo.

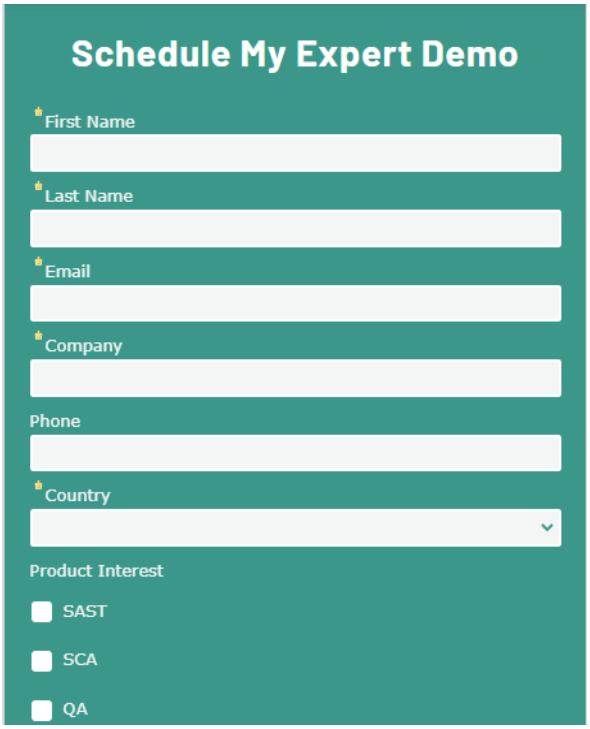


Figura 4.2.1.b: Formulario de solicitud de productos de Kiuwan.

➢ Load de código.

Para la carga de código se tiene que descargar una pequeña aplicación que realiza la subida del proyecto a la nube de Kiuwan. Esta permite una carga del proyecto indicando la ruta del proyecto en nuestro ordenador y se le coloca un nombre al proyecto para que se pueda

buscar en el navegador.

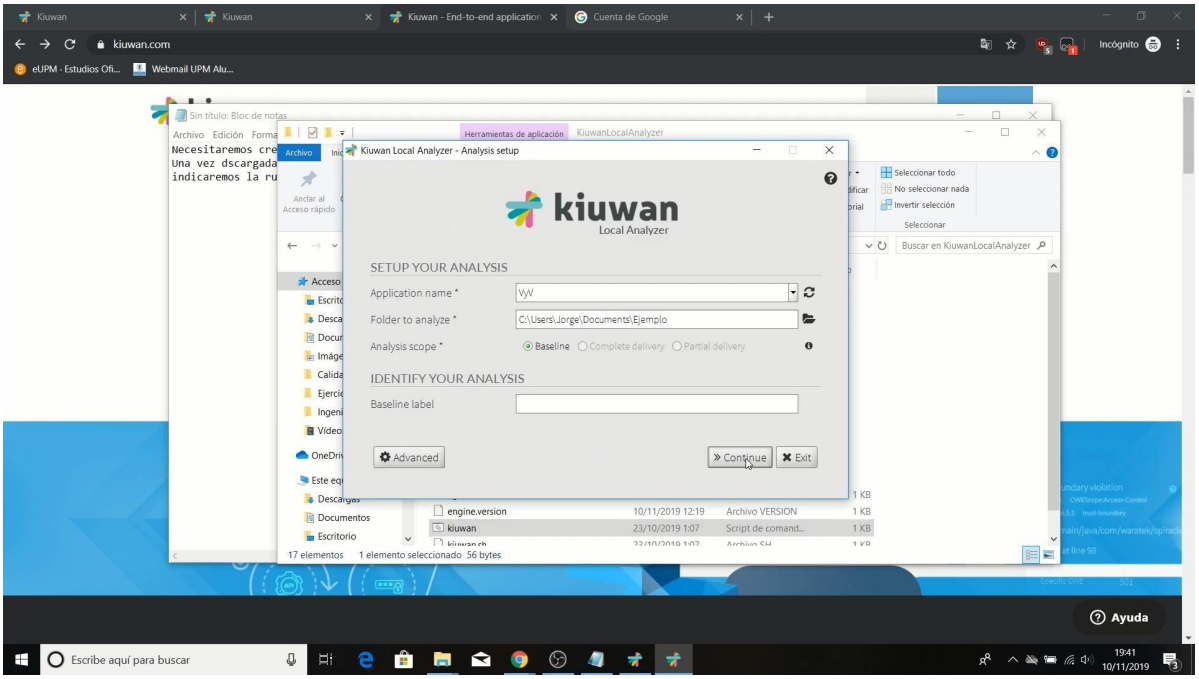


Figura 4.2.1.c: Captura de la aplicación para la carga de un proyecto de Kiuwan.

➢ Análisis de vulnerabilidades del Código.

Dependiendo del producto que se adquiera, se realizará el análisis. Esta herramienta se caracteriza por la rapidez en la que ejecuta este análisis de código.

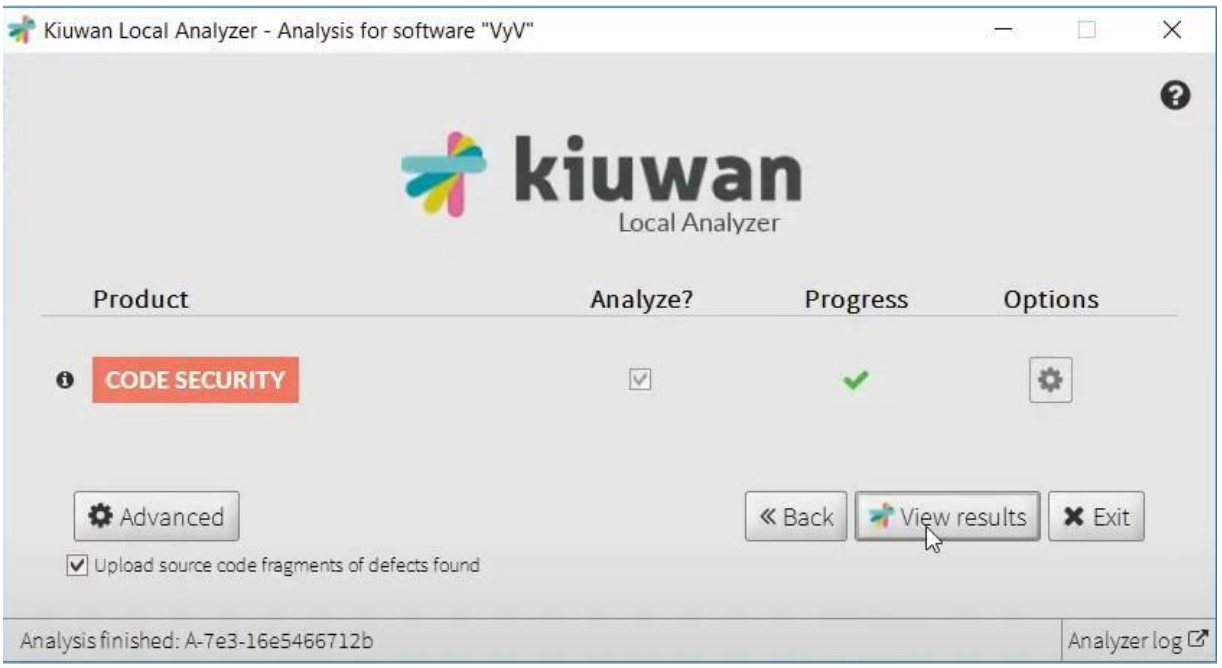


Figura 4.2.1.d: Captura a la aplicación de finalizado de análisis de Kiuwan.

La herramienta Kiuwan Code Security es una solución SAST (Static Application Security

Testing) que escanea automáticamente el código, posibilitando a los usuarios detectar y corregir vulnerabilidades de seguridad rápidamente.
Permite comprender el nivel de criticidad del software a partir de informes personalizados que utilizan calificaciones de seguridad estándares en la industria.También simula escenarios de mejora de la aplicación y proporciona planes de acción para resolver vulnerabilidades y gestionar la deuda técnica.

- ➢ Análisis de código de riesgos de componentes de terceros.

La solución SCA (Software Composition Analysis) de Kiuwan se encarga de descubrir vulnerabilidades causadas por soluciones de terceros y software Open Source. Aborda tanto debilidades de seguridad como cuestiones legales como el cumplimiento de licencias. Insights Open Source presenta un inventario completo de todos los componentes de código abierto y de terceros utilizados en la aplicación, localiza amenazas de seguridad y permite crear alertas automáticas de obsolescencia.

- ➢ Code Analysis (QA).

Ofrece una visión global de la calidad del código a partir de los cinco factores fundamentales del estándar ISO 25000: la mantenibilidad, la portabilidad, la eficiencia, la fiabilidad y la seguridad.
Kiuwan muestra el índice de riesgo de nuestra aplicación, el riesgo potencial que se está asumiendo por no prestar la atención suficiente a la calidad del código. Así como el esfuerzo requerido para alcanzar la calidad objetivo del proyecto.

Funciona de forma similar a la solución SAST. Ofrece un listado detallado de los defectos de calidad detectados (en lugar de vulnerabilidades de seguridad) y planes de acción con un set más amplio de métricas a tener en cuenta para su creación.
Al igual que en el módulo de seguridad, en el análisis de calidad de Kiuwan se puede mutear defectos que no consideremos relevantes y podemos revisar el código desde el problema detectado, los ficheros y hasta la línea de código concreta que presenta defectos. Igualmente, aporta información detallada sobre el problema localizado con ideas de cómo resolverlo.

- ➢ Governance y LifeCycle.

El apartado Governance ofrece una visión global de la calidad del código de un pipeline o una base de código. En este módulo las personas encargadas de tomar decisiones pueden monitorizar tendencias y valores de sus aplicaciones, como la calidad, el riesgo, la seguridad, la deuda técnica, etc.
Por su parte, el apartado de LifeCycle permite controlar la calidad del software que el equipo va incorporando a un repositorio. Y lo hace conectando los planes de acción con métricas de control de calidad. De esta manera, Kiuwan automáticamente indica al sistema de integración continua que un pull request tiene un rendimiento menor al deseado o que está listo para añadirse al repositorio.

- ➢ Log de resultado de análisis.

Cuando termina el análisis, se despliega en el navegador predeterminado por el computador, los resultados de los mismos. Estos tienen un dashboard muy amigable a la vista y dependiendo del producto, es como se muestran los resultados.Podemos ver la cantidad de

líneas de código,cantidad de líneas de código útiles,cantidad de archivos,cantidad de vulnerabilidades y a qué clase pertenecen. También podemos ver una puntuación de seguridad y el esfuerzo que nos llevará en tiempo mejorar esa puntuación.

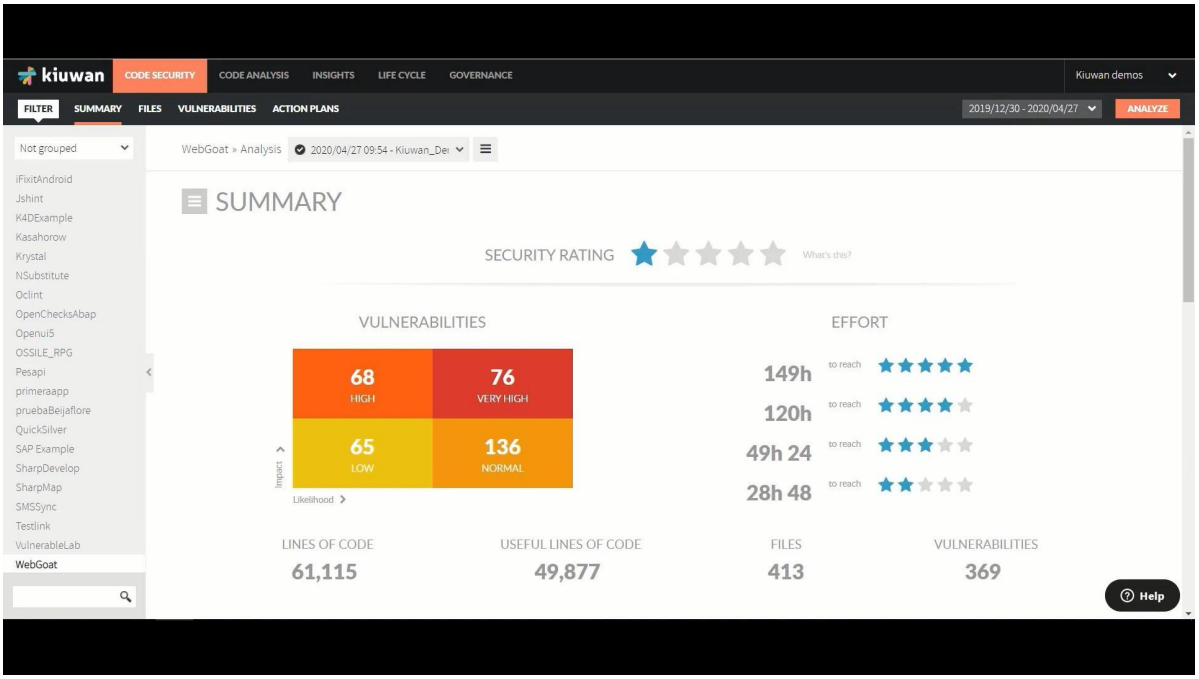


Figura 4.2.1.e: Dashboard de análisis SAST de Kiuwan.

En el apartado Files podemos ver todos los archivos del proyecto, junto con su puntuación de seguridad ,cantidad de líneas de código,cantidad de vulnerabilidades ,la prioridad y el esfuerzo que tomará arreglarlos.

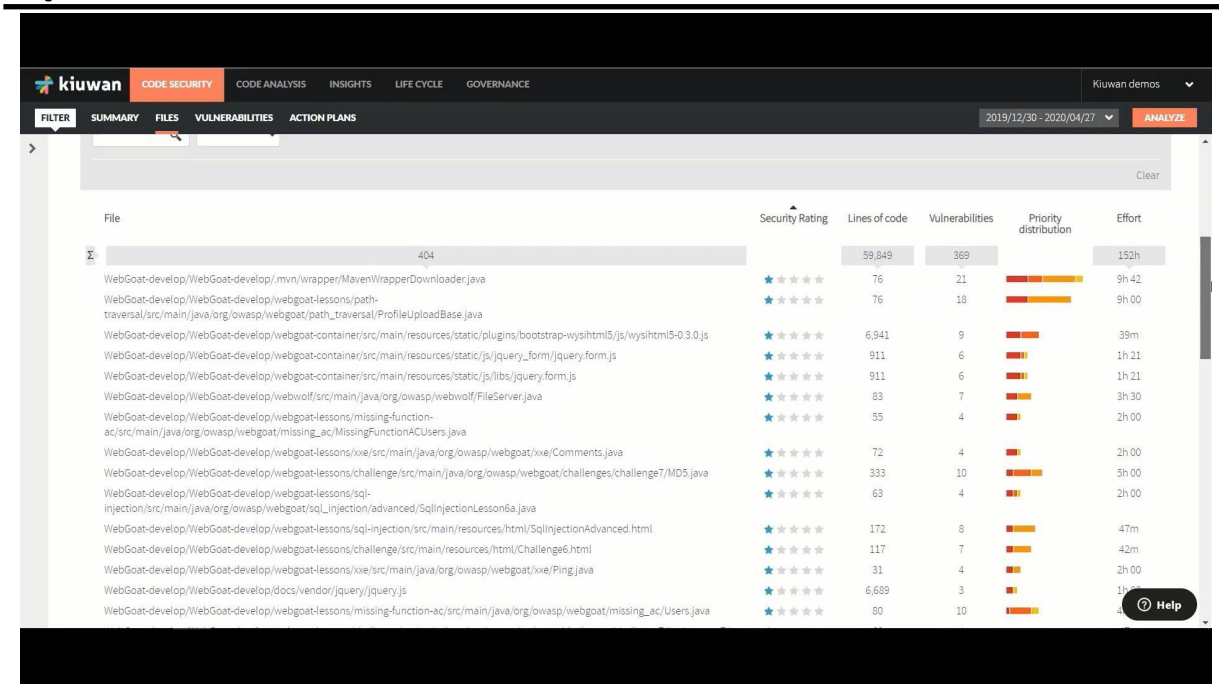


Figura 4.2.1.f: Apartado de archivos de Kiuwan.

En el apartado de vulnerabilities podemos ver 3 gráficos de vulnerabilidad organizados por tipo, lenguaje y prioridad. Además veremos la cantidad de reglas violadas, cantidad de vulnerabilidades , cantidad de vulnerabilidades de alto riesgo y la puntuación de seguridad.

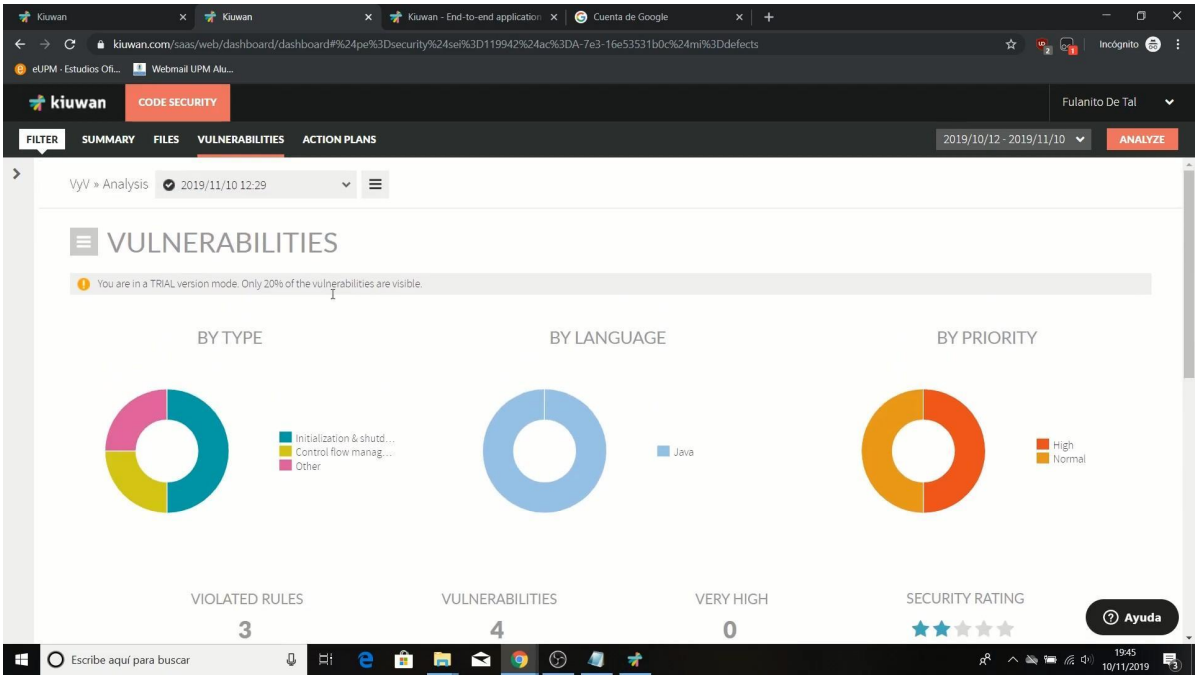


Figura 4.2.1.g: Captura de gráficos de vulnerabilidades de Kiuwan.

Si bajamos más en el mismo apartado, veremos cuál ha sido el problema, en que clase está,cuál es su prioridad y el tiempo que nos tomará arreglarlo.

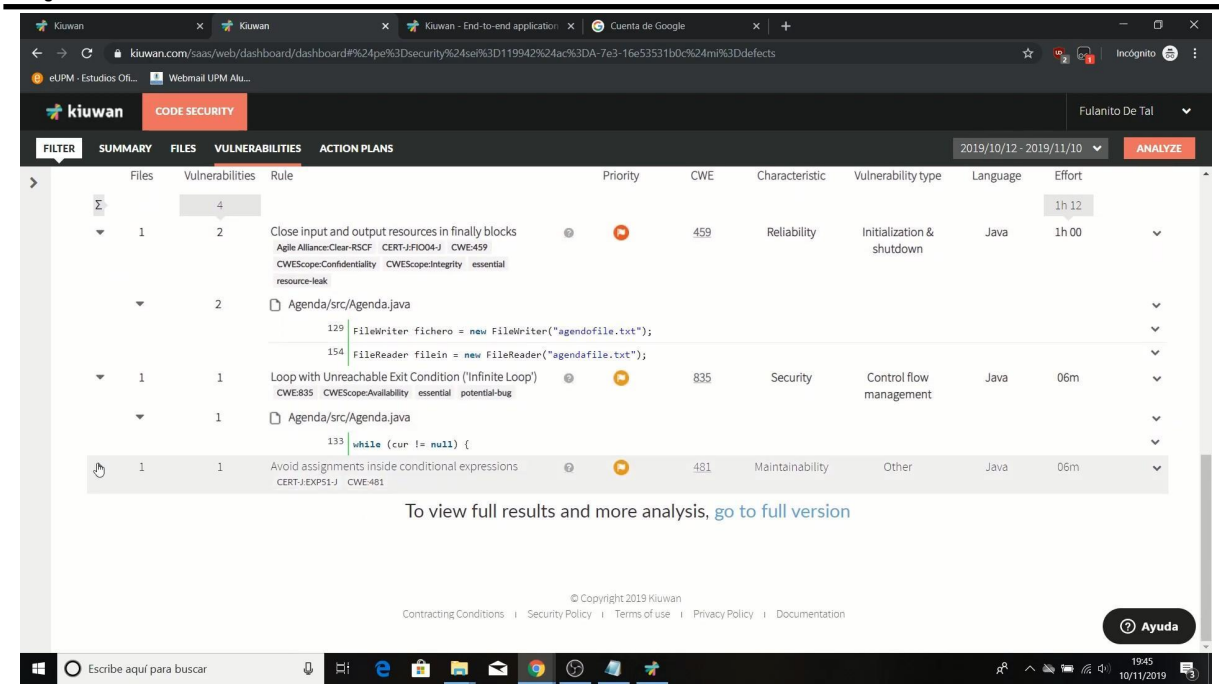


Figura 4.2.1.h: Tabla de vulnerabilidades de Kiuwan.

En la sección Action Plans podremos ver como quedaría nuestro proyecto si invertimos determinadas cantidad de horas o la puntuación que queremos conseguir.Esta pantalla nos hace una simulación y nos muestre el antes y el después de nuestro proyecto.

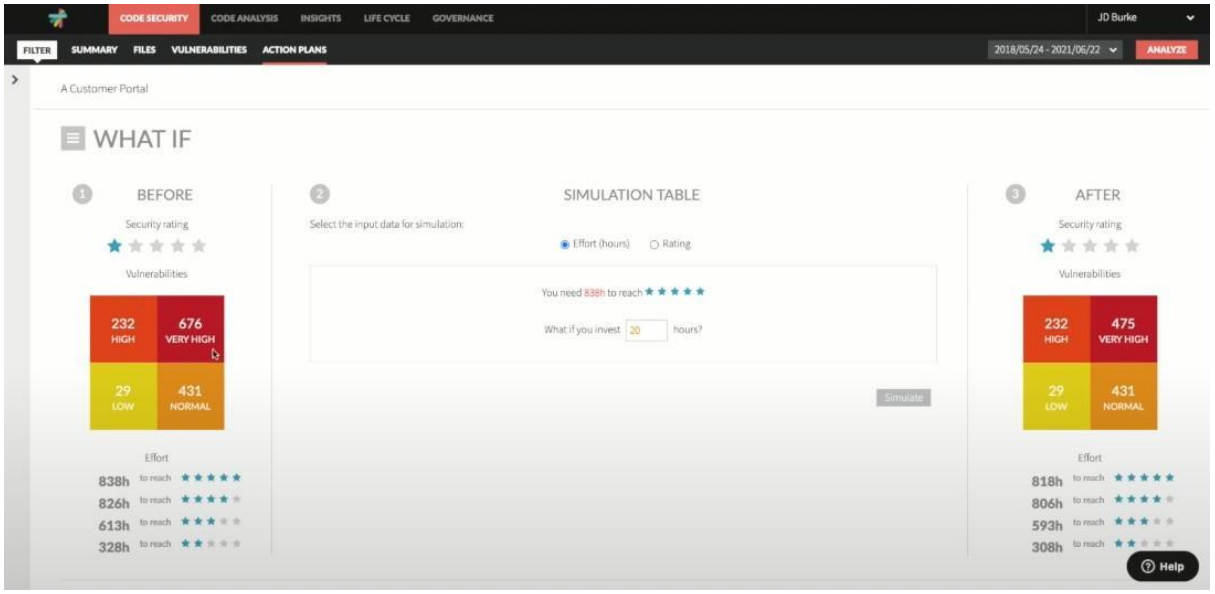


Figura 4.2.1.i: Action plan de Kiuwan.

Si bajamos en la pantalla, podemos ver en que se usarían esa cantidad horas,es decir,que vulnerabilidades se solucionarían.

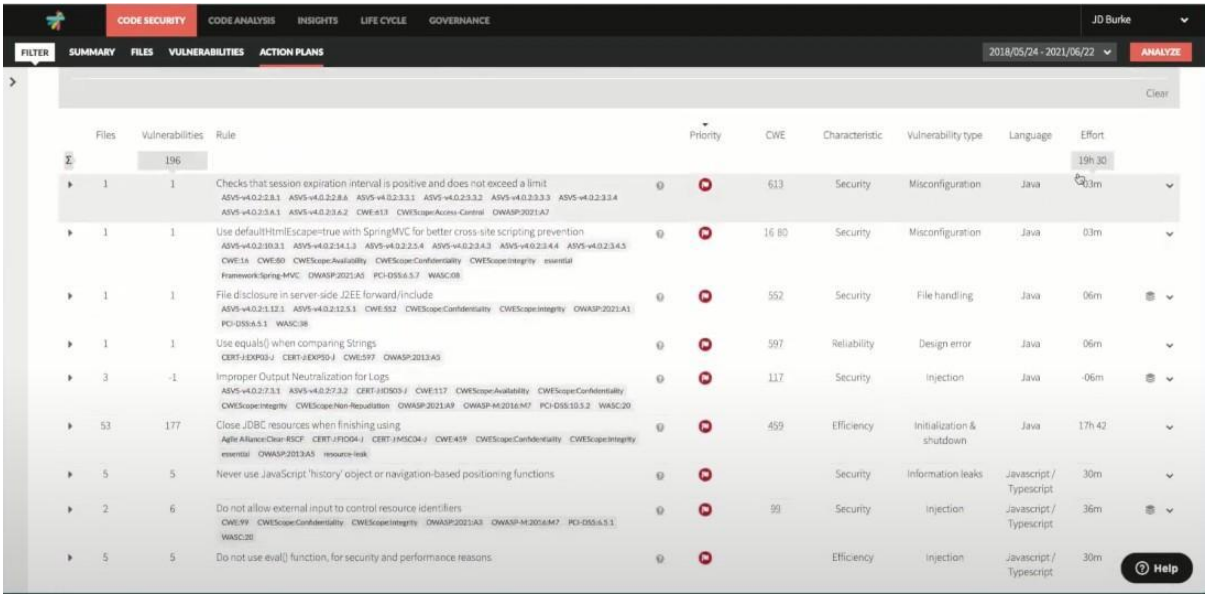


Figura 4.2.1.j: Vulnerabilidades a resolver de Kiuwan.

➢ Ver historial de análisis.

Kiuwan ofrece un base de datos en la nube donde quedan un historial de los análisis de proyectos anteriores. Los muestra como un listado en el lado izquierdo del dashboard, y muestra los nombre de los proyectos que se colocan en la carga.

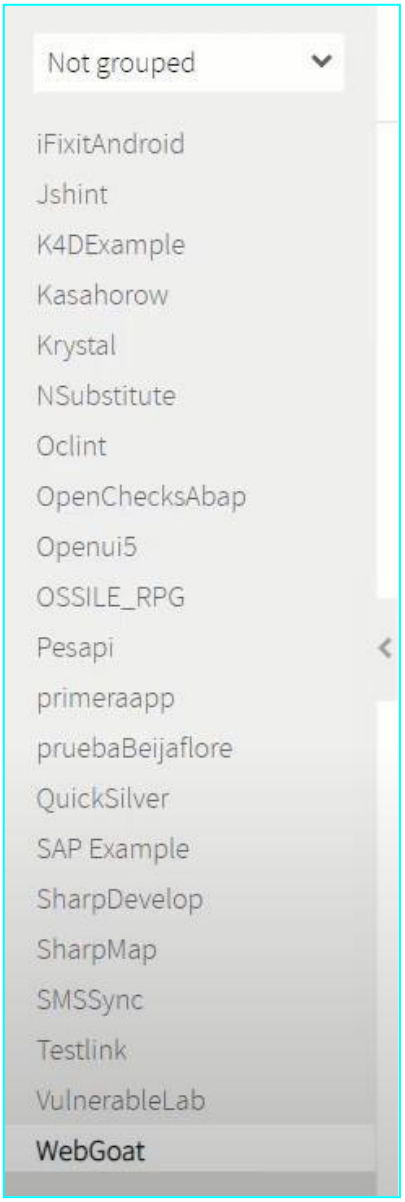


Figura 4.2.1.k: Historial de análisis de Kiuwan.

➢ Modelo lógico del sistema actual:

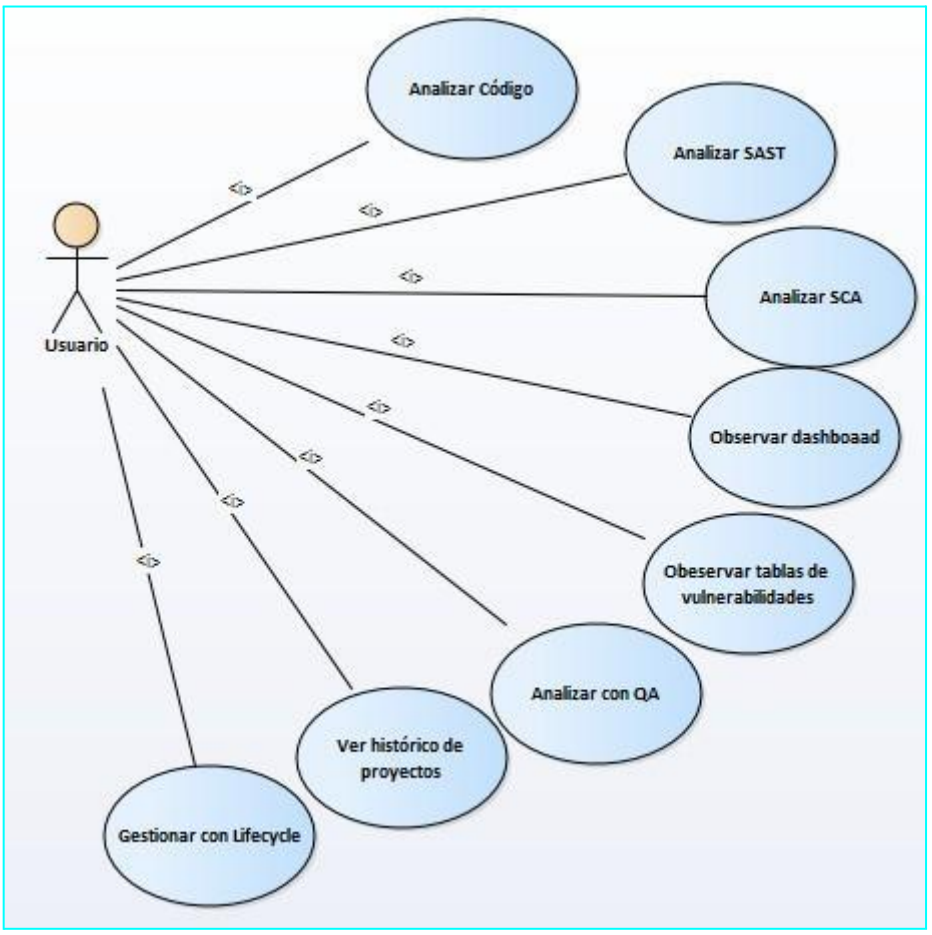


Diagrama 4.2.1.I: Modelo de casos de usos de Kiuwan.

4.2.2.Problemas y necesidades detectados en las funciones relevadas en detalle y en su entorno organizacional.

- **Problema en el login de usuario:** Para poder registrarse, el usuario debe esperar una autorización para adquirir cualquier tipo de producto, sea la demo o un producto comprado. Cuando uno realiza la petición, es donde van incluido los datos personales y con esos mismos datos se rigen para crear tu cuenta.

- **Problema en la carga de código:** Es obligatorio descargarse una pequeña aplicación. No es muy eficiente con respecto a la carga ya que se puede utilizar otras herramientas mucho más factibles.

- **Necesidad en login de usuario:** La posibilidad de crear un usuario, al intentar descargar la demo, mínimamente, y no estar esperando una autorización. Estas autorizaciones, por más de que son rápidas de responder y son por seguridad e integridad, es conveniente darle una pequeña libertad al usuario de poder tener la demo con mucha mayor facilidad.

- **Necesidad en el load de código:** Permitir subir el código más fácilmente, sin depender de

una aplicación.

- **Necesidad en el análisis de código (SATS, SCA y QA):** Poder hacer un análisis más extensivo, para que se pueda revisar la calidad del código. Ya que Kiuwan sólo analiza la seguridad.

- **Necesidad en el log de resultados del análisis:** Como se explica en la función anterior, solo se tiene un log de resultados de análisis de seguridad, esto desemboca en que la necesidad de esta función es mostrar un log más orientado a calidad y documentación.

5.SonarQube.

Qué es SonarQube: Verifica y analiza la calidad de tu código. (s.f.). Obtenido de https://sentrio.io/blog/que-es-sonarqube/.
Code Quality and Code Security | SonarQube. (s.f.). Obtenido de https://www.sonarqube.org/

5.1.Relevamiento General.

5.1.1.De la Organización.

SonarQube es una herramienta de revisión automática de código para detectar bugs, vulnerabilidades y malos olores de código. Analiza el código fuente de más de 20 lenguajes de programación. Es software libre y usa diversas herramientas de análisis estático de código fuente como Checkstyle, PMD o FindBugs para obtener métricas que ayudan a mejorar la calidad del código de un programa.

5.1.2.Funciones detectadas a nivel general y relaciones con otros Sistemas y Entidades.

- ➢ Login de Usuario.

La herramienta tiene una funcionalidad que brinda al usuario una cuenta permitiendo modificar e iniciar sesión en la misma.

- ➢ Análisis de Código.

SonarQube realiza un análisis de métricas para mantener un código limpio y prolijo. Separa entre faltas graves y leves, brinda de esta manera la posibilidad de poder centrarse en los errores más importantes al momento de corregirlos.

- ➢ Log de resultado de análisis

Esta función permite ver para cada archivo de nuestro código, cuáles son los problemas en específico que tiene. Permitiendo navegar entre los distintos archivos del proyecto.

- ➢ Historial de ejecuciones

Posee un pequeño historial de ejecuciones y sus resultados, mostrando comparativas con los análisis pasados.

- ➢ Quality profiles

Son colecciones de reglas que se revisan durante un análisis. Para cada tecnología hay un Quality Profile predeterminado, pero podemos tener varios para un mismo lenguaje.
En caso de que un proyecto analizado no tenga asignado explícitamente uno, se analizará con el perfil configurado como predeterminado para dicho lenguaje.

- Quality gate

Es el conjunto de condiciones que el proyecto debe cumplir antes de que pueda ser lanzado a producción. Se usa para hacer cumplir una política preestablecida.

Diagrama 5.1.2.a: Diagrama de interfaces de SonarQube.

5.1.3.Tecnología de información.

Es una herramienta que trabaja de manera local, para hacer uso de la misma se descarga una aplicación que se encuentra en la página oficial. La misma corre sobre el puerto 9000 y está programada en Java. Se requiere que se tenga descargado el jdk 11 en la computadora. La misma posee soporte para los sistemas operativos Windows, Linux y Mac.

5.2.Relevamiento detallado y análisis del Sistema.

5.2.1.Detalle, explicación y documentación detallada de todas las funciones seleccionadas.

- Login de Usuario

Para poder loguearse, se entra al puerto local 9000, y se muestra esta pantalla. Se debe ingresar tanto el el campo login como en el campo password, admin, debido a que es el usuario que la aplicación crea por defecto.

Figura 5.2.1.a: Captura de inicio de sesión de SonarQube.

➢ Análisis de Código.

El primer paso para poder analizar el código es agregar un archivo con la extensión properties, el cual contiene la información que necesita la aplicación para poder encontrar el código, debido a que esta herramienta no permite cargar un código, sino que lo busca en la computadora en base a la ruta que se especifique.

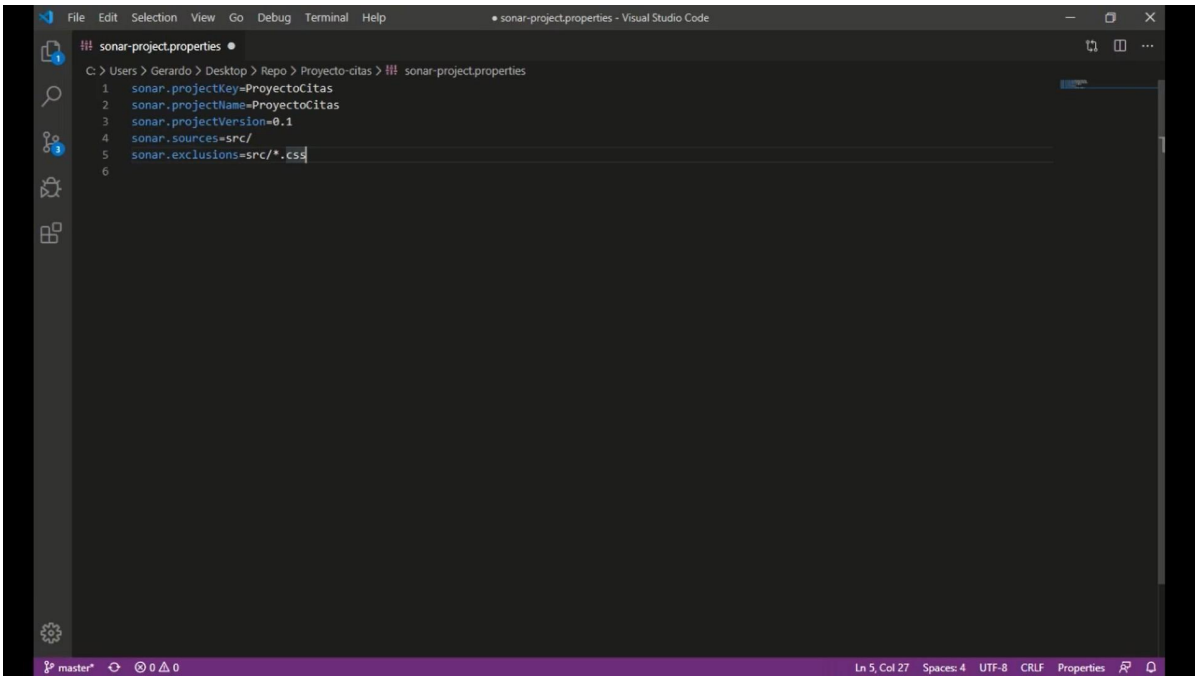


Figura 5.2.1.b: Captura del archivo properties de SonarQube.

Luego, para poder iniciar el análisis, se ingresa a la consola de comandos, ubicado en la ruta de la carpeta donde está almacenado el código y se ejecuta el comando sonar-scanner.

Figura 5.2.1.c: Consola de comandos para iniciar el análisis de SonarQube.

Para el análisis, el SonarQube detecta y clasifica 3 errores:

- Bug: un punto de fallo real o potencial en su software.
- Vulnerability: Un agujero de seguridad que puede usarse para atacar sus software.
- Code Smells: Un problema relacionado con la mantenibilidad en el código.

➢ Log de resultado de análisis.

Se muestra una vista general del proyecto en un dashboard, donde se puede ver la cantidad de bugs, de vulnerabilidades, de Hotspots, de Code Smells y de duplicaciones de código.

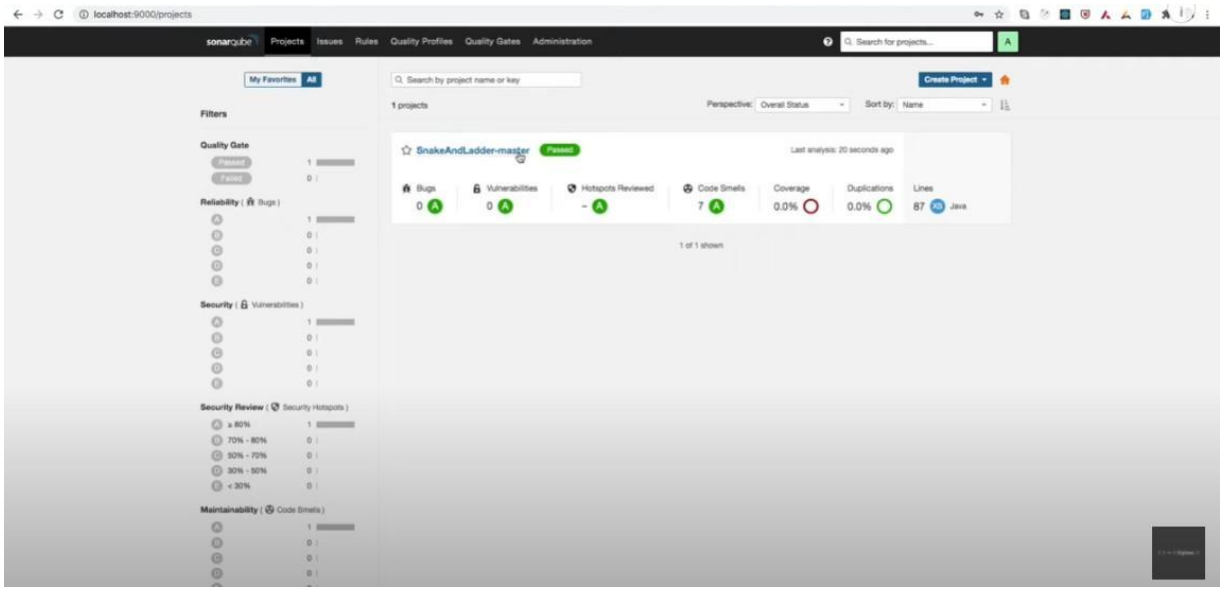


Figura 5.2.1.d: Dashboard de SonarQube.

También se puede acceder a esta vista, en la cual se ve cuáles de los problemas son graves y cuáles menores, e indicando a qué clase pertenece.

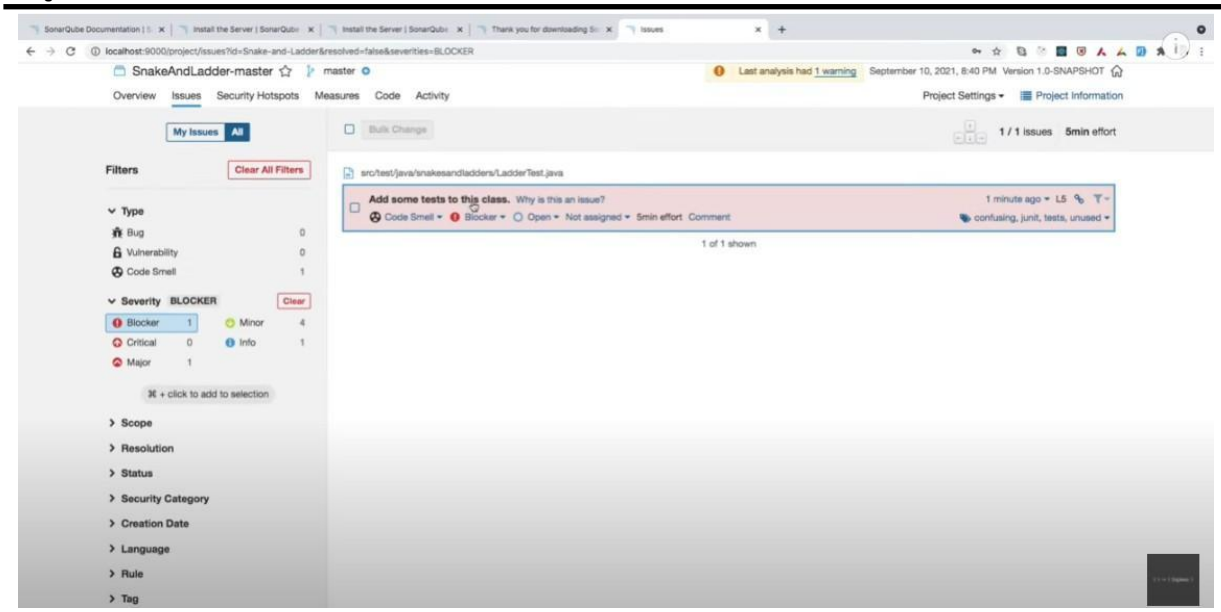


Figura 5.2.1.e: Captura de vistas de errores de SonarQube.

Y dentro de esa misma vista, se ve, para cada problema, con la opción “Why is this an issue” precisamente el motivo del problema. Se muestra en la parte inferior de la pantalla la descripción completa.

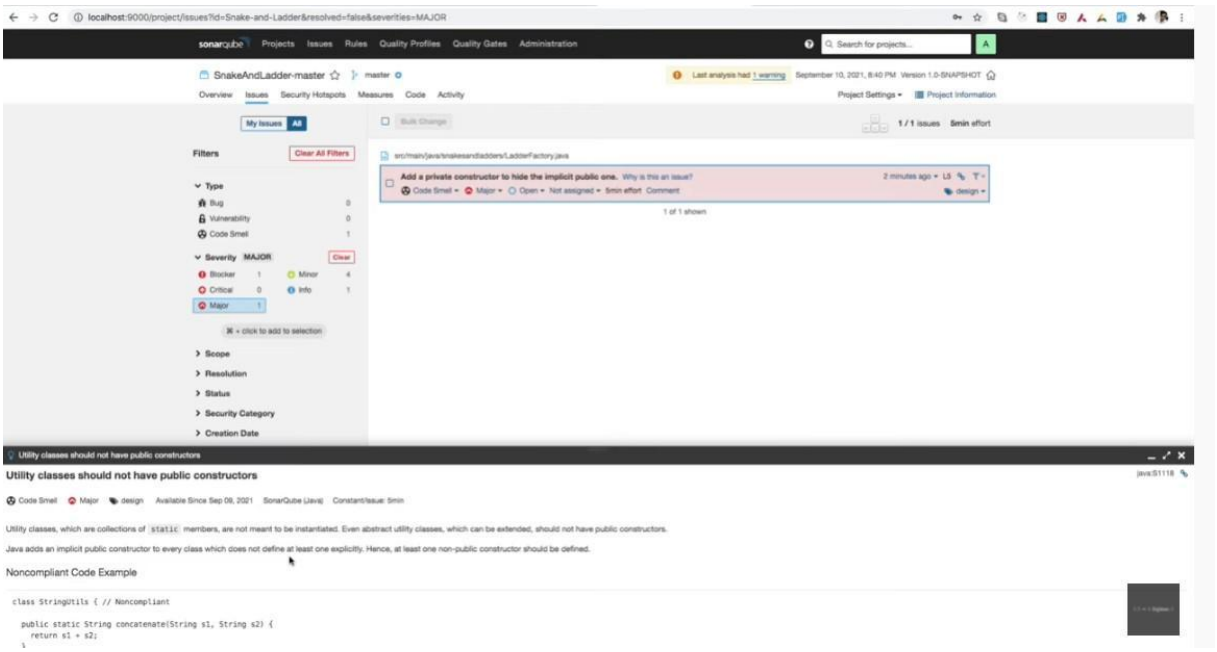


Figura 5.2.1.f: Captura del “Why is this an issue” de SonarQube.

➢ Historial de ejecuciones.

Esta funcionalidad muestra un historial de ejecuciones y compara todos los resultados de todas las pruebas pasadas. Se puede observar cómo van mejorando ciertos aspectos como: número de issues, cobertura de código, código duplicado, entre otros. Esta característica permite una integración continua del proyecto al que se le quiere hacer el análisis y mantenimiento de código.

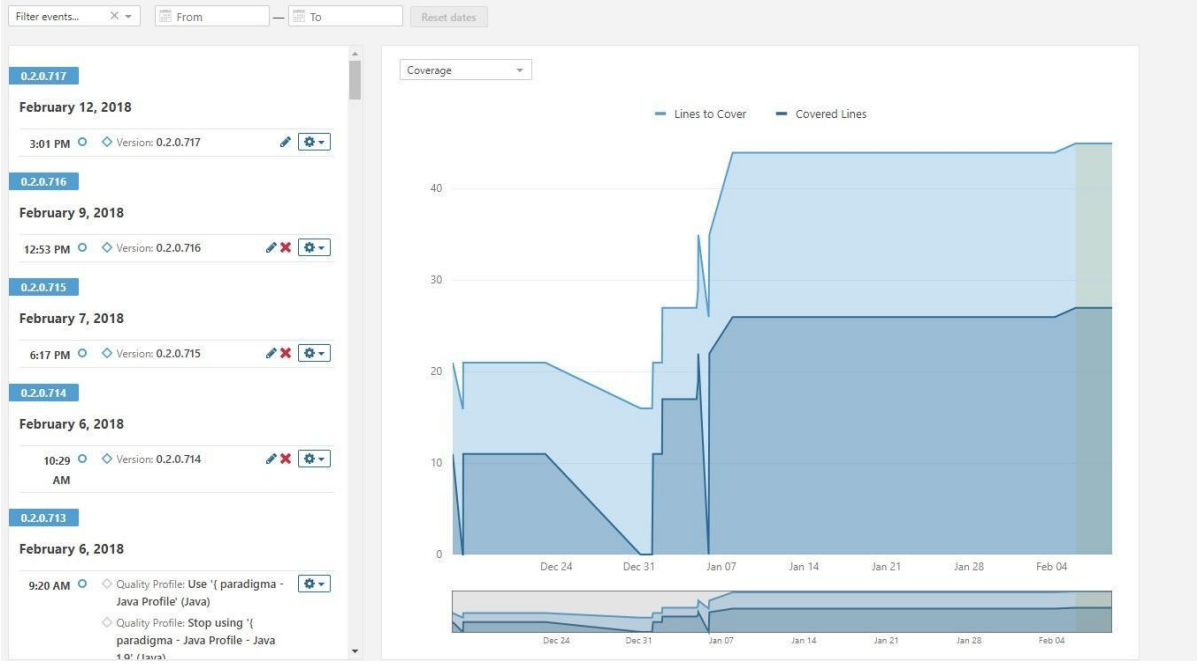


Figura 5.2.1.f: Historial de Ejecuciones.

➢ Quality profiles.

Para cada tecnología hay un quality profile, es decir, un conjunto de reglas predeterminado para esa tecnología. En caso de querer hacer algo más especializado, se puede crear un nuevo perfil con las reglas que se crean apropiadas. Una tecnología puede tener más de un perfil asignado.

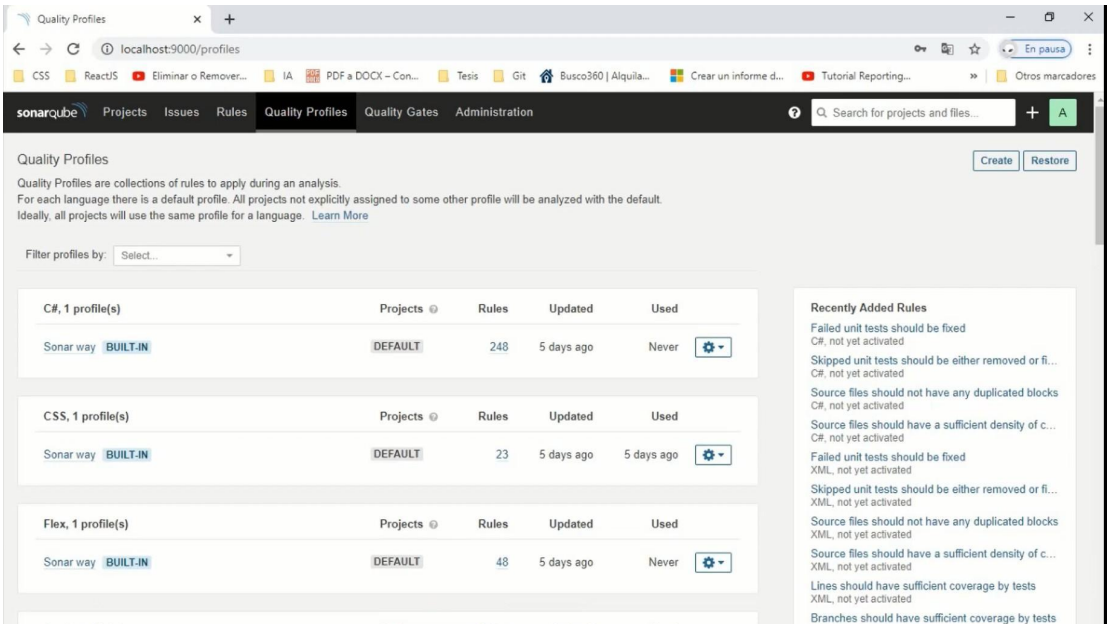


Figura 5.2.1.g: Configuración de Quality Profiles.

➢ Quality gate.

Con esta función se puede definir la política que se necesita cumplir para que un proyecto sea aprobado, es decir, que pasen una serie de condiciones booleanas. Para esto se seleccionan las métricas que se precisen junto con su operador y valor apropiados.

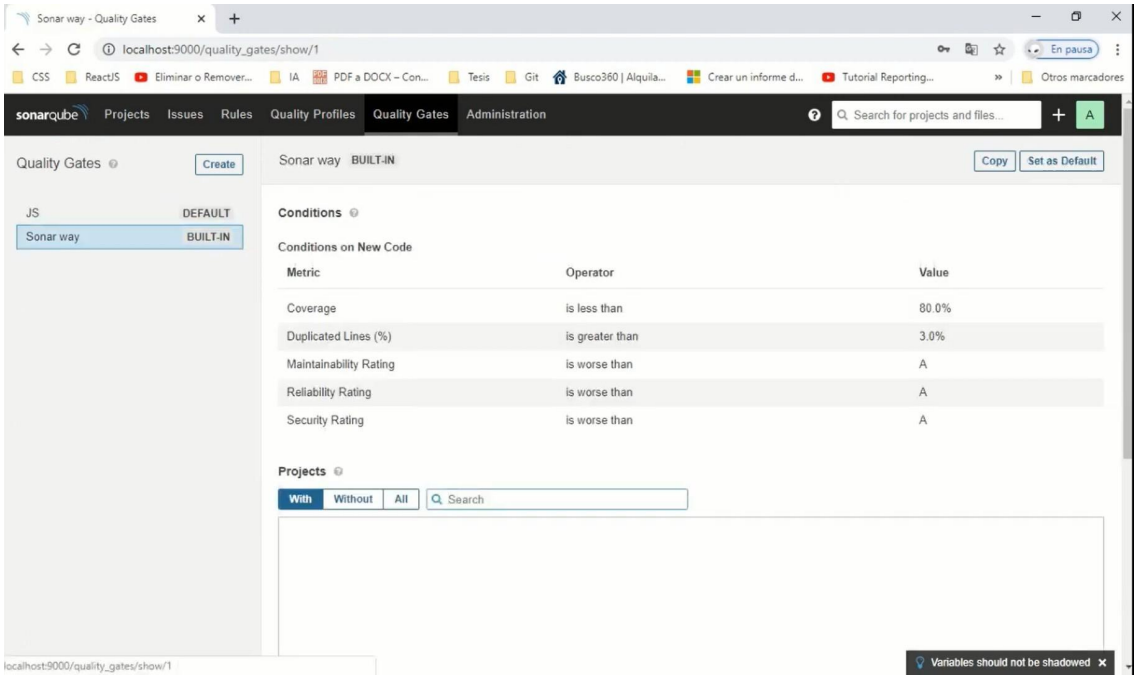


Figura 5.2.1.h: Configuración de Quality Gates.

➢ Modelo lógico del sistema actual.

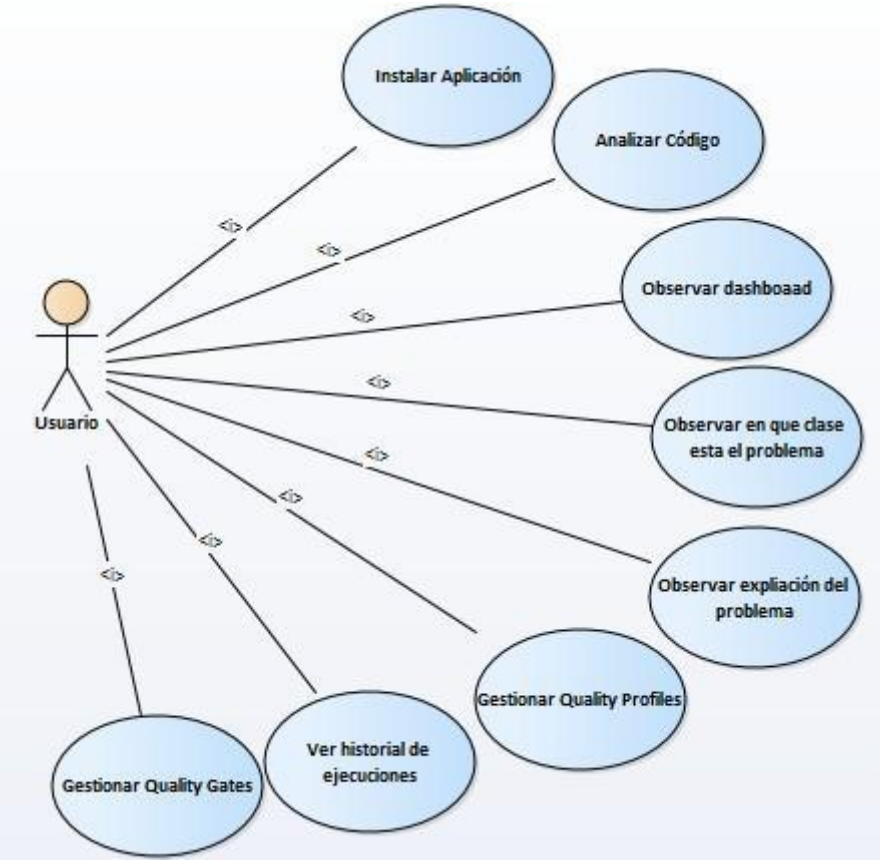


Diagrama 5.2.1.i: Modelo de caso de usos de SonarQube.

5.2.2.Problemas y necesidades detectados en las funciones relevadas en detalle y en su entorno organizacional.

- **Problema en el análisis de código:** Es muy tedioso el proceso para poder iniciar el análisis, debido a que se necesita tener un archivo que se debe mantener actualizada información hardcodeada, además de mantener siempre en la ubicación del proyecto los archivos ".bat".

- **Necesidad en el login de usuario:** Para poder ingresar al servicio se necesita un usuario y una contraseña, cuando hay posibilidades de poder ingresar más fácil con la integración de GitHub.

- **Necesidad en el análisis de código:** Realizar un análisis más enfocado a la documentación.

6.DeepScan.

Oficial, S. (s.f.). DeepScan. Obtenido de https://deepscan.io/

6.1.Relevamiento General.

6.1.1.De la Organización.

DeepScan es una herramienta de análisis estático de última generación para código JavaScript. Al seguir la ejecución y el flujo de datos del programa con mayor profundidad, puede encontrar problemas que los linters basados en sintaxis no pueden. Al clasificar los problemas como impactos de varios niveles y suprimir los ruidosos, DeepScan lo ayuda a concentrarse en los problemas impactantes. Además, una guía detallada le permite saber simplemente cuál y dónde está el problema.

6.1.2.Funciones detectadas a nivel general y relaciones con otros Sistemas y Entidades.

- ➢ Login de usuario.

Para acceder a la herramienta es necesario tener una cuenta en GitHub la cual se usa para loguearse a DeepScan.

- ➢ Load de código.

Cómo se accede a la herramienta desde una cuenta Github, se debe tener el código a analizar en un repositorio. Luego desde la herramienta, solo queda seleccionar el repositorio a analizar.

- ➢ Análisis de código.

DeepScan al analizar encuentra errores prácticos y smells de código mediante el análisis de flujo de datos. Clasifica los errores según su gravedad. Además, se enfoca en encontrar errores de tiempo de ejecución y problemas de calidad.

- ➢ Log de resultado del análisis.

Se muestra un resumen del proyecto (grado de la rama analizada, cantidad de líneas, cantidad de archivos, cantidad de problemas encontrados) como también los problemas encontrados en detalle. También permite ver un gráfico de tendencias para ver cómo cambió el proyecto con el paso del tiempo.

- ➢ Histórico de errores no resueltos en el tiempo.

Muestra un gráfico con los errores no resueltos a lo largo del tiempo. Estos errores están diferenciados en tres colores, verde, amarillo y rojo, los cuales significan el impacto de esos errores, en este caso, bajo impacto, impacto medio y alto impacto respectivamente.

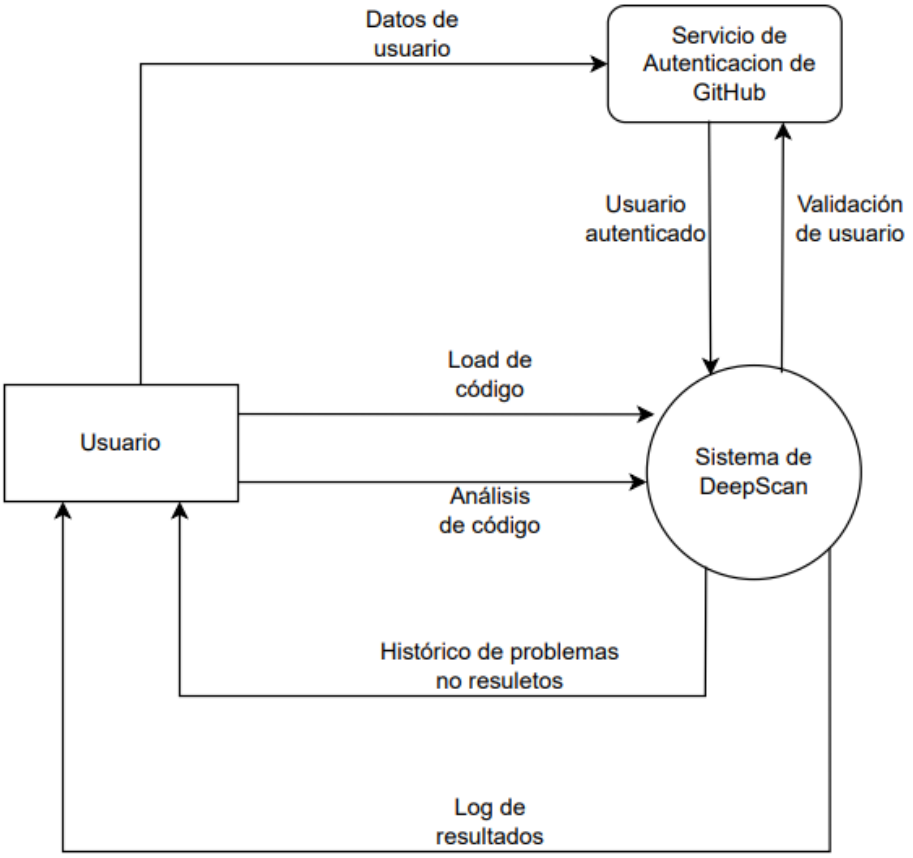


Diagrama 6.1.2.a: Diagrama de interfaces de Deepscan.

6.1.3.Tecnología de información.

DeepScan es una herramienta web pero también ofrece un paquete para atom y una extensión a Visual Studio Code para que realicen el análisis de JavaScript sobre la marcha. La misma está programada en Js y en Java. Y cuenta con soporte para múltiples navegadores, entre ellos se encuentran Google Chrome, Brave, Microsoft Edge, Safari.

6.2.Relevamiento detallado y análisis del Sistema.

6.2.1.Detalle, explicación y documentación detallada de todas las funciones seleccionadas.

- ➢ Login de usuario.

Para poder loguearse a DeepScan se necesita una cuenta de Github para posteriormente acceder con dicha cuenta.

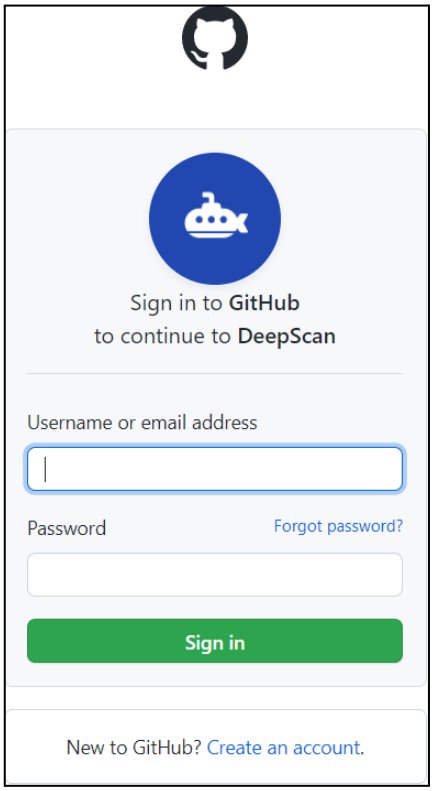


Figura 6.2.1.a: Inicio de sesión con GitHub.

➢ Load de código.

Para cargar el código a analizar, se presiona el botón Add from Repositories. Se elige entre los repositorios y simplemente haciendo un click en Add Project empieza el análisis.

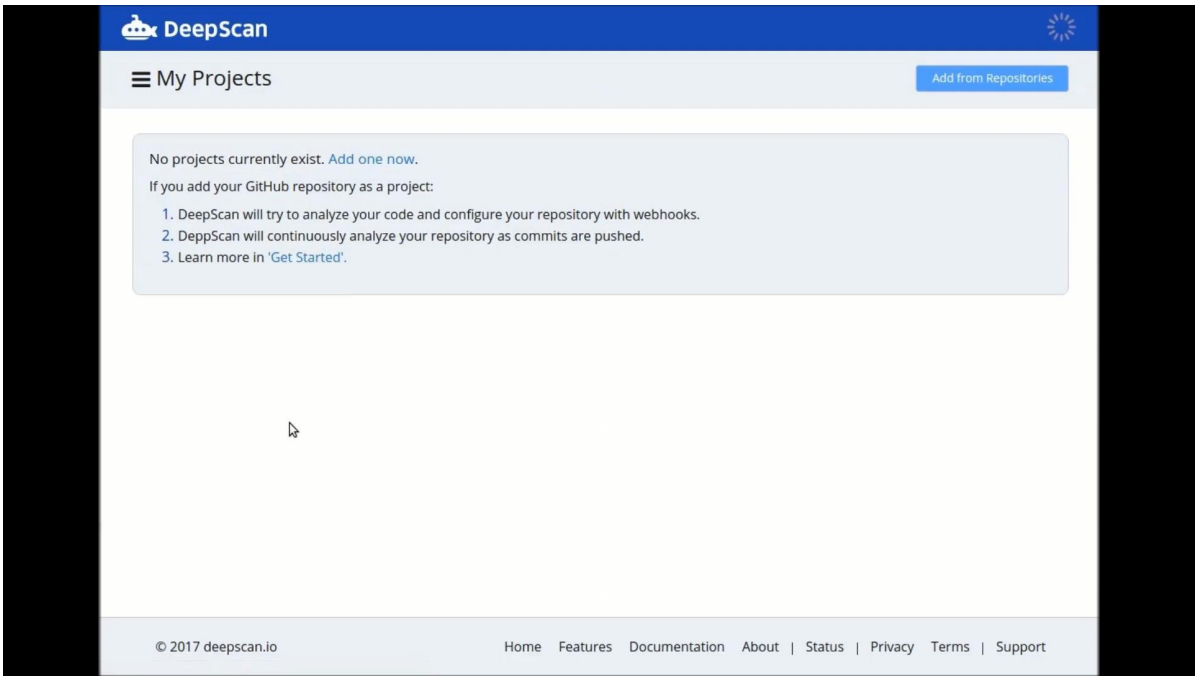


Figura 6.2.1.b: Captura de la carga del código de Deepscan.

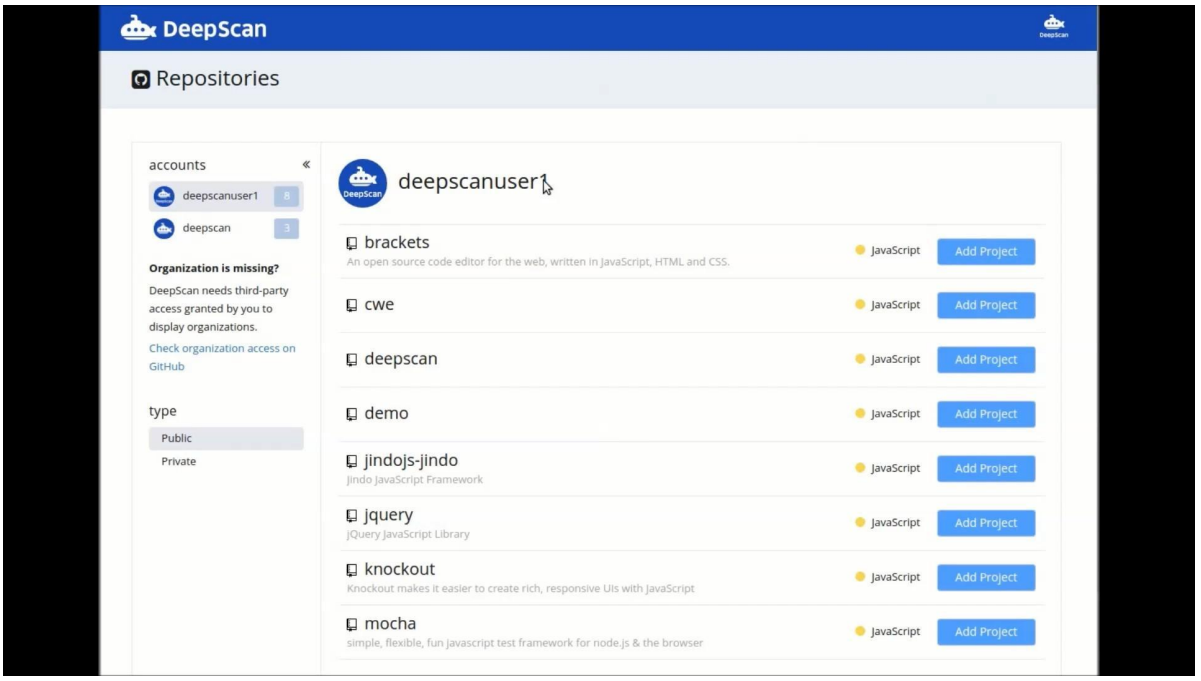


Figura 6.2.1.c: Selección de repositorio para la carga en Deepscan.

➢ Análisis de código.

En esta pantalla se puede ver cómo se lleva a cabo el análisis. El cúal separa los errores en 3 tipos de gravedad baja, media y alta.También se muestra la dirección del archivo donde se encontraron los problemas, con una descripción de como arreglar el error.

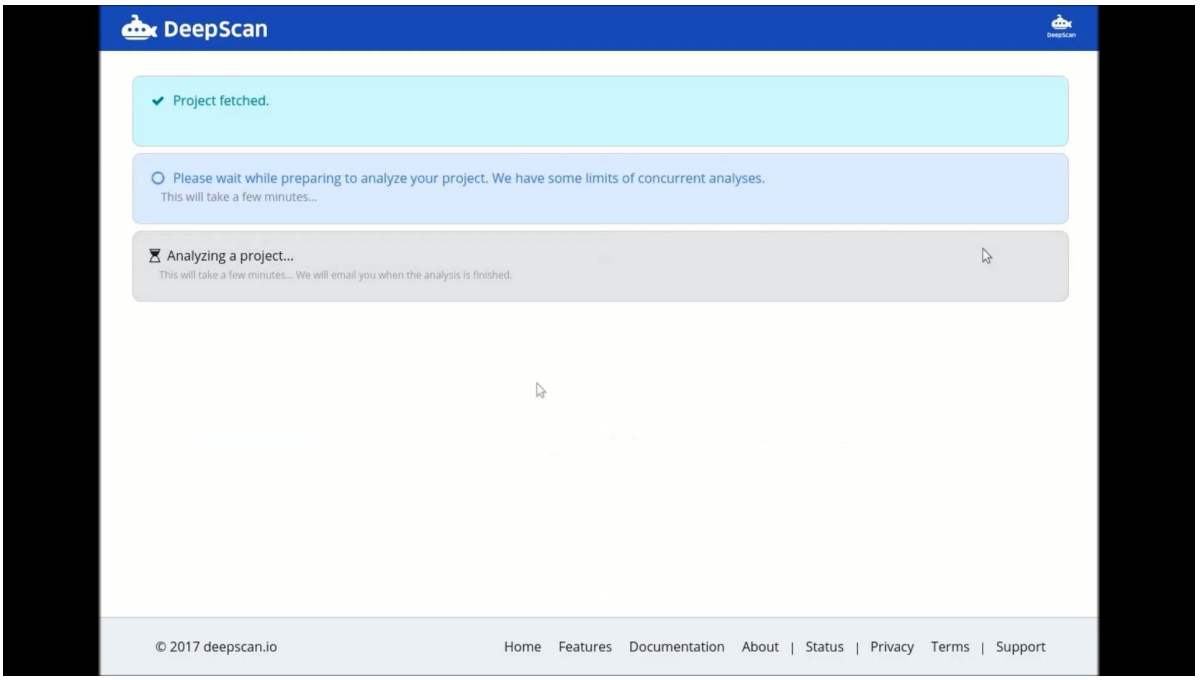


Figura 6.2.1.d: Captura del avance del análisis de Deepscan.

Cabe aclarar que desde un apartado llamado settings se puede ver reglas que vienen por

defecto y elegir con qué reglas analizar el proyecto.

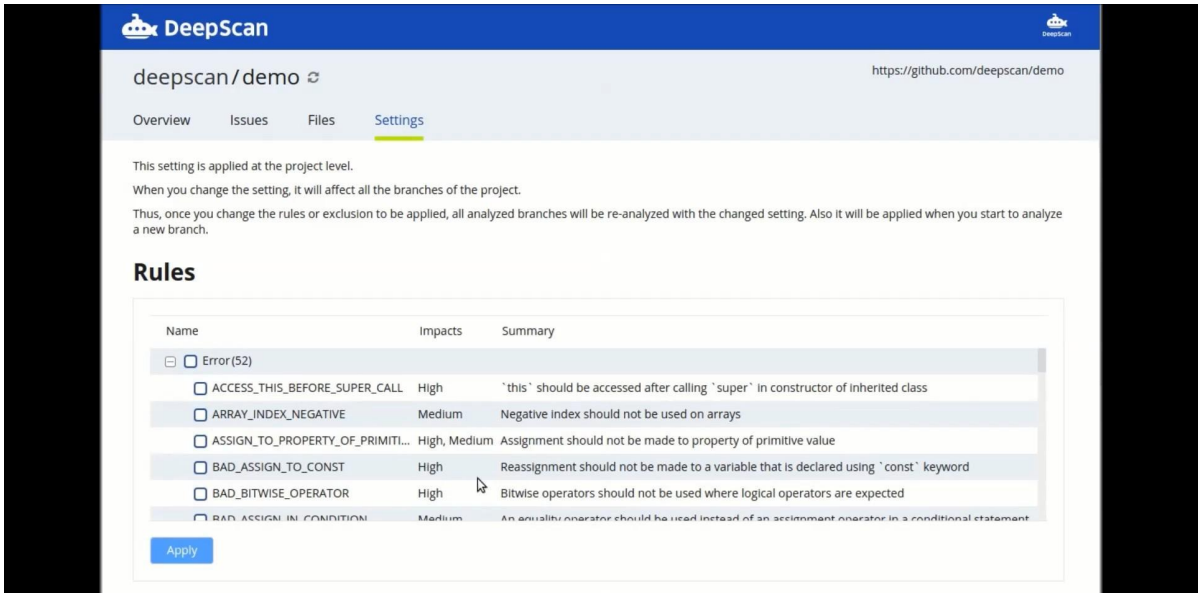


Figura 6.2.1.e: Captura de las reglas de análisis de Deepscan.

➢ Log de resultado del análisis.

Una vez finalizado el análisis, se muestra un dashboard principal en cual se puede ver si la prueba se realiza con éxito, cantidad de líneas analizadas, cantidad de archivos analizados, total de problemas encontrados, nuevos problemas encontrados, problemas arreglados, la tendencia del proyecto conforme pasa el tiempo y un grado de la rama analizada el cual se calcula en base a la densidad de problemas encontrados.

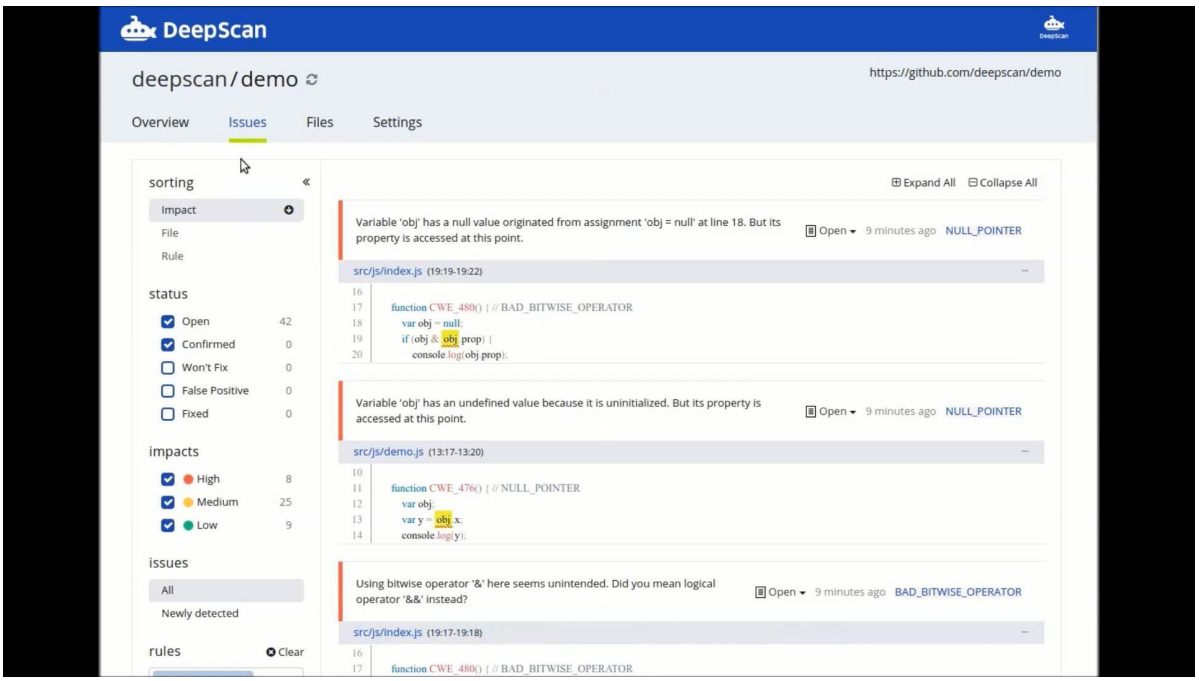


Figura 6.2.1.f: Dashboard con los resultados de los análisis de Deepscan.

En el apartado Issues se puede ver los problemas en detalles, clasificándolos en impacto y estado. Los errores estan resaltados de amarillo y tambien se indica cual es el problema y el nombre de la regla que se violo.

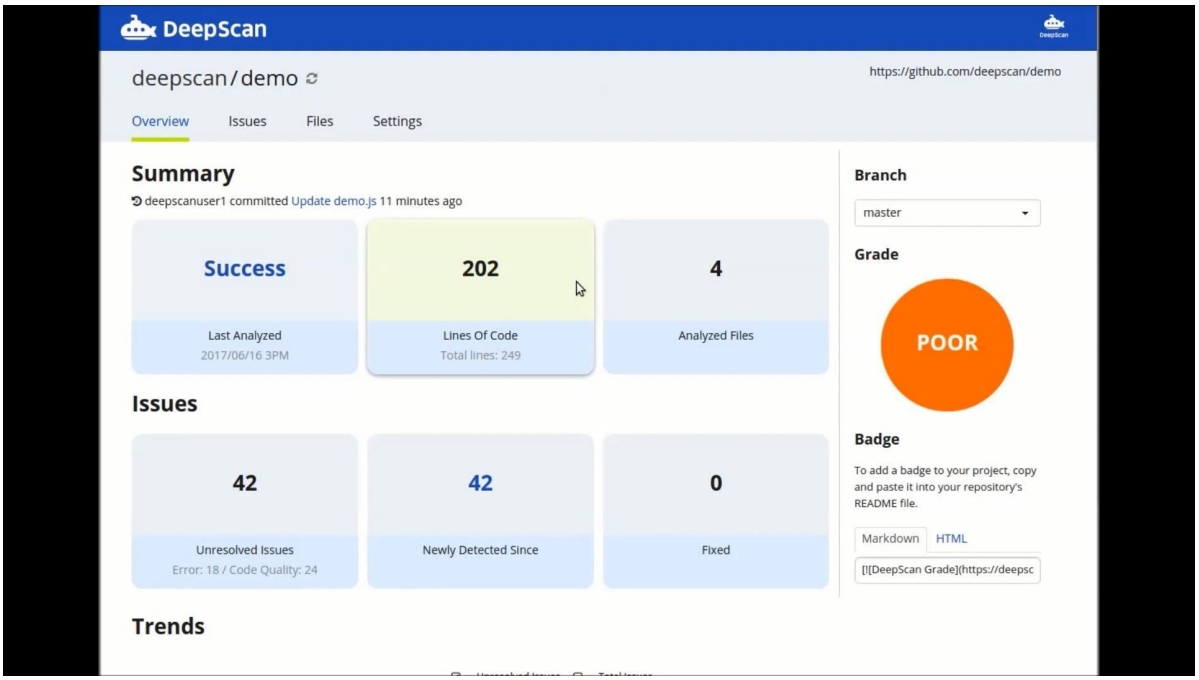


Figura 6.2.1.g: Captura del apartado de Issues de Deepscan.

➢ Histórico de errores no resueltos en el tiempo.

Se puede observar el gráfico de los errores no resueltos a lo largo del tiempo. A medida que los errores se van corrigiendo las líneas comienzan a bajar, lo cual indica que quedan menos errores sin resolver.

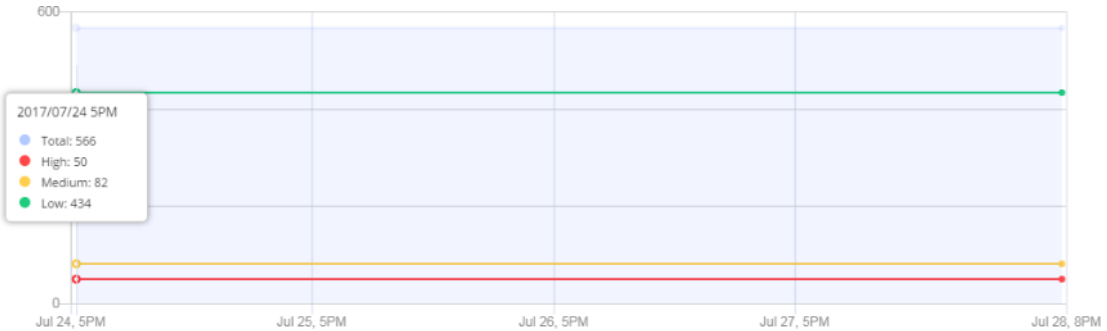


Figura 6.2.1.h: Histórico de No resueltos.

➢ Modelo lógico del sistema actual.

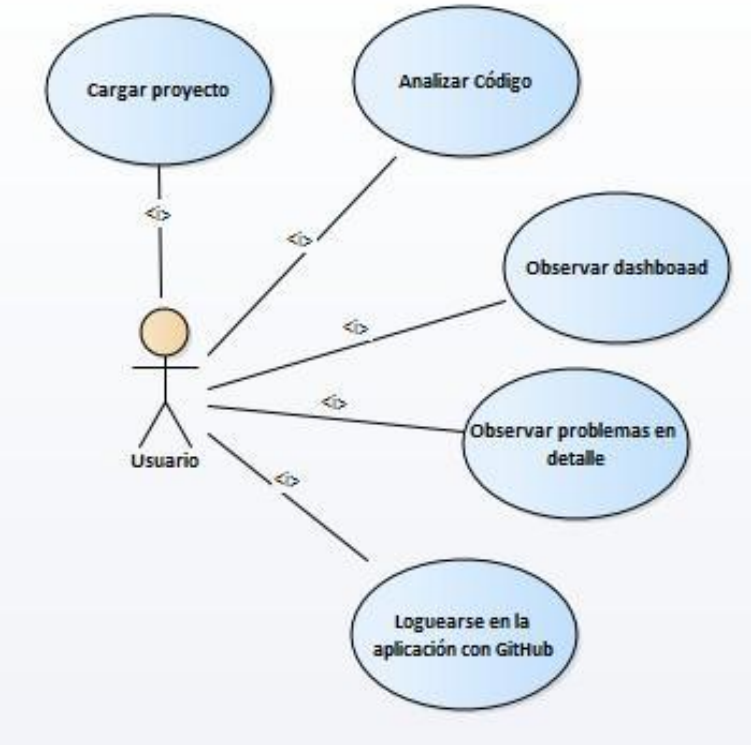


Diagrama 6.2.1.h: Modelo de caso de usos de Deepscan.

6.2.2.Problemas y necesidades detectados en las funciones relevadas en detalle y en su entorno organizacional.

- **Necesidad en el análisis de código:** Hacer un análisis más extensivo para no solo encontrar errores de calidad de código, sino también para que dicho análisis sirva para poder documentar el mismo.

- **Necesidad en el log de resultado del análisis:** No presenta diagramas ni gráficos de los análisis.

7.Veracode.

- Descubre las vulnerabilidades de tus aplicaciones con VERACODE. (s.f.). Obtenido de https://acktib.com/veracode/.

- Juarez, C. L.-G. (s.f.). Cómo realizar análisis de código con Veracode e integrarlo con IntelliJ. Obtenido de https://enmilocalfunciona.io/como-realizar-analisis-de-codigo-con-veracode-e-integrarlo-con-intellij/

7.1.Relevamiento General.

7.1.1.De la Organización.

Veracode proporciona múltiples tecnologías de análisis de seguridad de software en una sola

plataforma SaaS, incluido análisis estático (o prueba de caja blanca) todo lo cual sirve para prevenir vulnerabilidades de software como secuencias de comandos entre sitios e inyección SQL .

7.1.2.Funciones detectadas a nivel general y relaciones con otros Sistemas y Entidades.

- ➢ Login de usuario.

Para poder acceder a la herramienta Veracode se debe tener una cuenta para poder hacer uso de la herramienta. Esta tiene que ser autorizada por los administradores de Veracode.

- ➢ Load de código.

Para subir el código, se realiza a través de la página web de Veracode. Se le puede agregar algunas características como nombre y descripción del mismo.

- ➢ Análisis de código.

Esta herramienta posee un plus en el análisis ya que permite el análisis tanto estático como dinámico. Además de poder realizar un análisis de la página web, se puede instalar un plugin en Visual Code Studio para realizar un análisis en el IDE.

- ➢ Log de resultados del análisis.

Dependiendo de qué forma estemos utilizando (web o plugin), se muestran logs o interfaces distintas, pero ambas describen las mismas especificaciones.

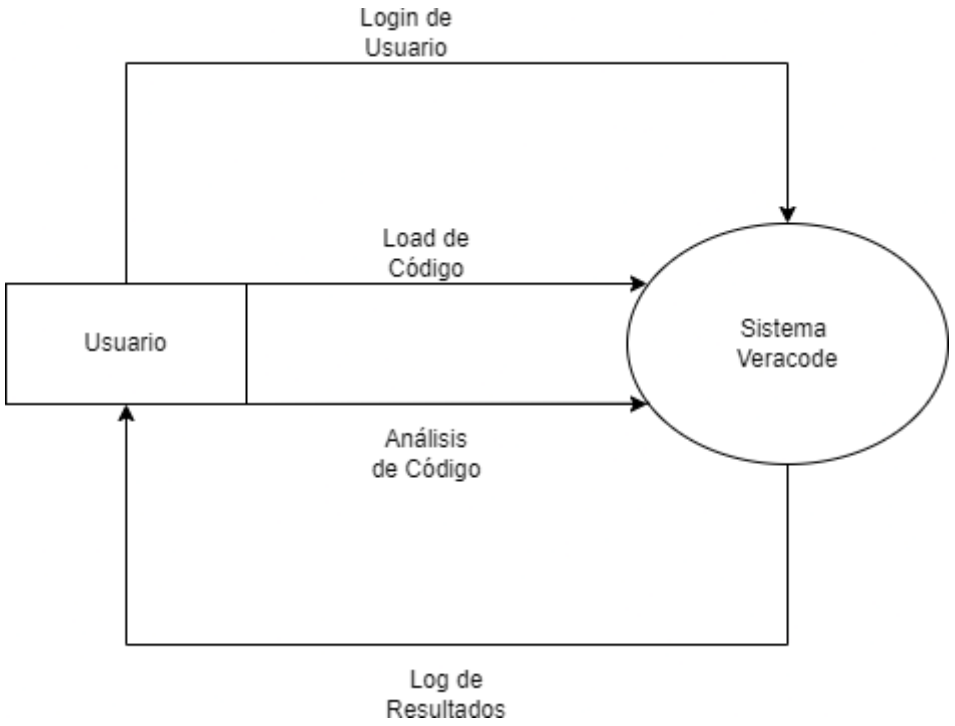


Diagrama 7.1.2.a: Diagrama de interfaces de Veracode.

7.1.3.Tecnología de información.

Herramienta de entorno web de esquema SaaS montado sobre una API**.** Tiene un plugin para poder instalarse en Visual Code para poder realizar el análisis. Como es una herramienta de entorno web, se puede acceder mediante Brave, Google Chrome, Microsoft Edge, Safari, entre otros exploradores. La misma está programada en Js y C#. (Code Quality and Code Security | SonarQube, s.f.).

7.2.Relevamiento detallado y análisis del sistema.

7.2.1.Detalle, explicación y documentación detallada de todas las funciones seleccionadas.

➢ Login de usuario.

Para registrar un usuario, es obligatorio ingresar un nombre, un apellido, un email, un nombre de usuario. De manera opcional, también es posible ingresar el teléfono. Ese usuario una vez que está registrado, se utiliza para iniciar sesión.

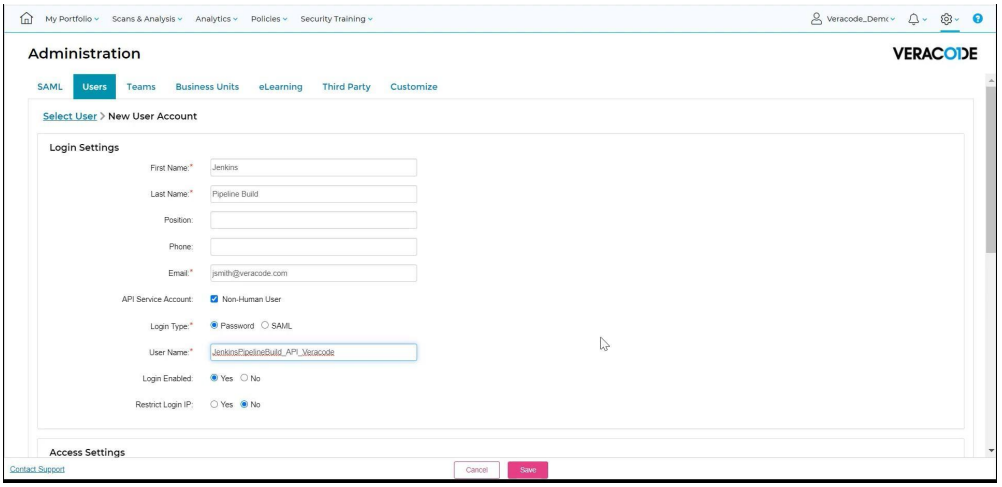


Figura 7.2.1.a: Captura de la creación de usuarios de Veracode.

Si bien no es una función para el login, cabe mencionar que permite utilizar roles, tales como Administrador, Creador, Administrador de Políticas, cada uno con distintos permisos y funciones.

➢ Load de código.

Para la carga de código, Veracode brinda un servicio web donde se pueden cargar nuestro código. En la carga del código, se tiene que colocar un nombre al proyecto, y opcionalmente, una pequeña descripción y/o tags.
Además, esta herramienta permite indicar el nivel de criticidad que tiene el código que vayamos a analizar. Estas policy control son totalmente parametrizables.

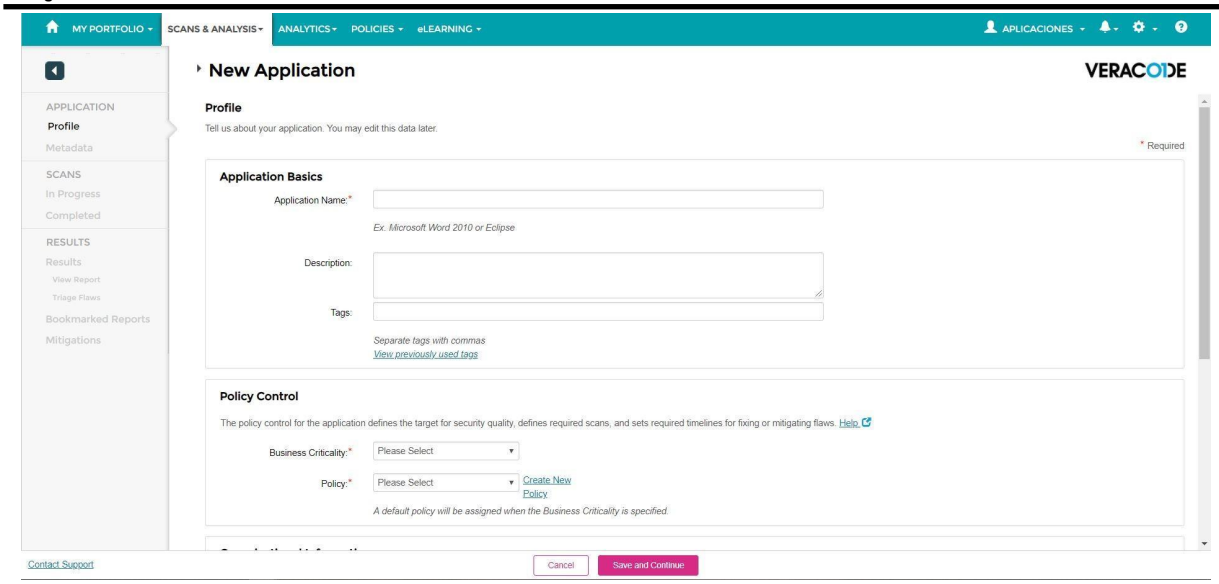


Figura 7.2.1.b: Captura de carga de código de Veracode.

➢ Análisis de Código.

Para poder analizar código, Veracode permite 2 formas de utilizarse, por un plugin en el IDE de Visual Studio Code o por la página web de Veracode.
En el primer caso, se debe instalar el plugin en el IDE Visual Studio Code. Con el código en el IDE, se debe realizar un scan with greenlight, que es una función de integración que posee Veracode con Visual Studio Code.

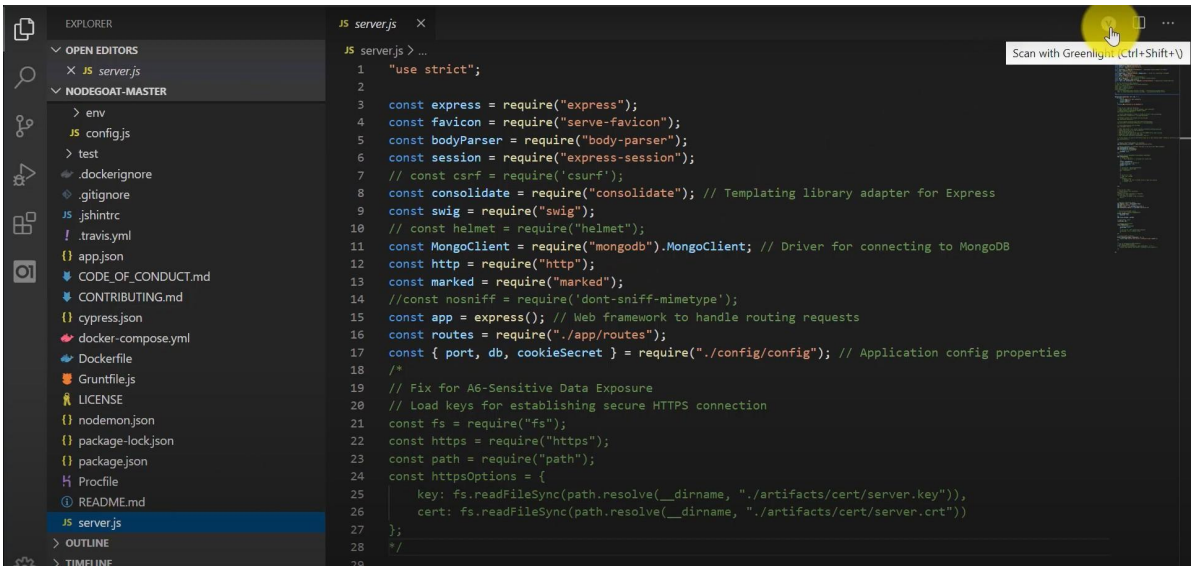


Figura 7.2.1.c: Captura de código de Visual Studio Code.

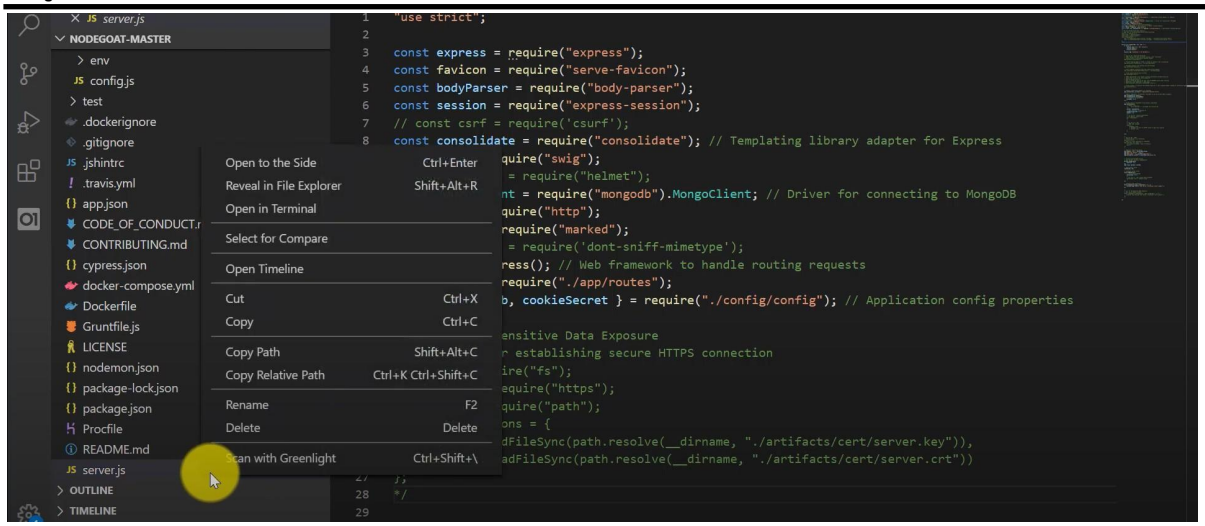


Figura 7.2.1.d: Comienzo de análisis de código con el plugin de Veracode.

En la página web, se observa un estilo de “formulario” donde se indicar, opcionalmente, información adicional a la aplicación a escanear, en relación al tipo de aplicación (terceros, con licencia, open source, etc), tipo de industria (agricultura, educación, etc), propósito de la aplicación (CRM, testing, seguridad, etc), tipo de aplicación (web, mobile, etc) y Custom Fields.

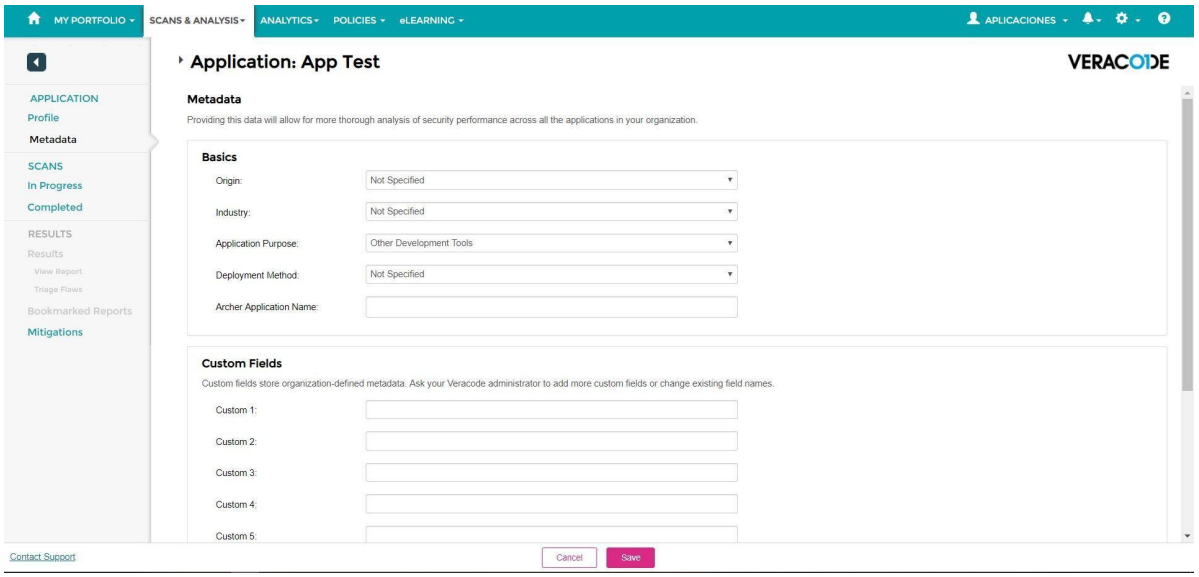


Figura 7.2.1.e: Captura de Inicio de análisis en web de Veracode.

➢ Log de resultado del análisis.

Si se accede mediante la aplicación web, la herramienta provee un dashboard, para ver el porcentaje de clases que pasan las políticas, cuales las pasan condicionalmente y cuáles no las pasan. También muestra en función del tiempo como ha ido evolucionando la seguridad del proyecto. Y, por último, el dashboard, permite ver los resultados para cada equipo.

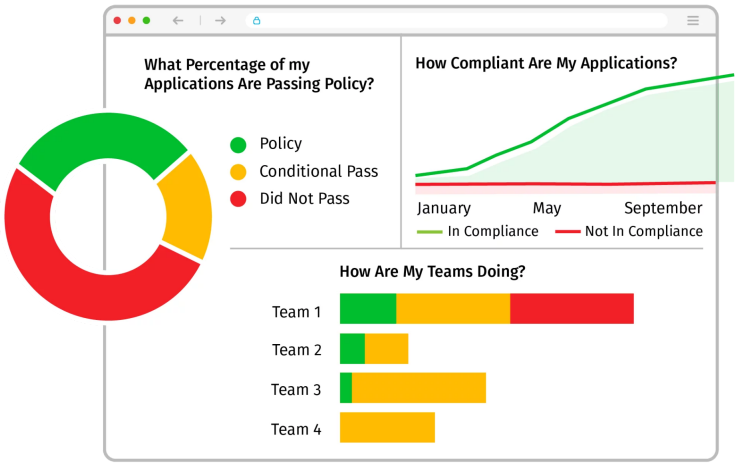


Figura 7.2.1.f: Avance de proyecto por equipo de Veracode.

También se puede observar los errores clasificados por importancia, y se indica en qué clase del proyecto se encuentra.

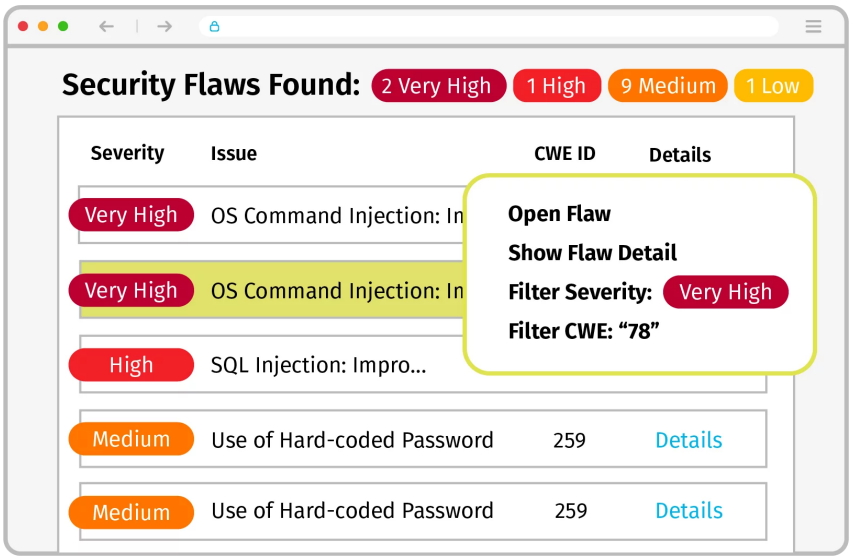


Figura 7.2.1.g: Imagen de errores clasificados por importancia de Veracode.

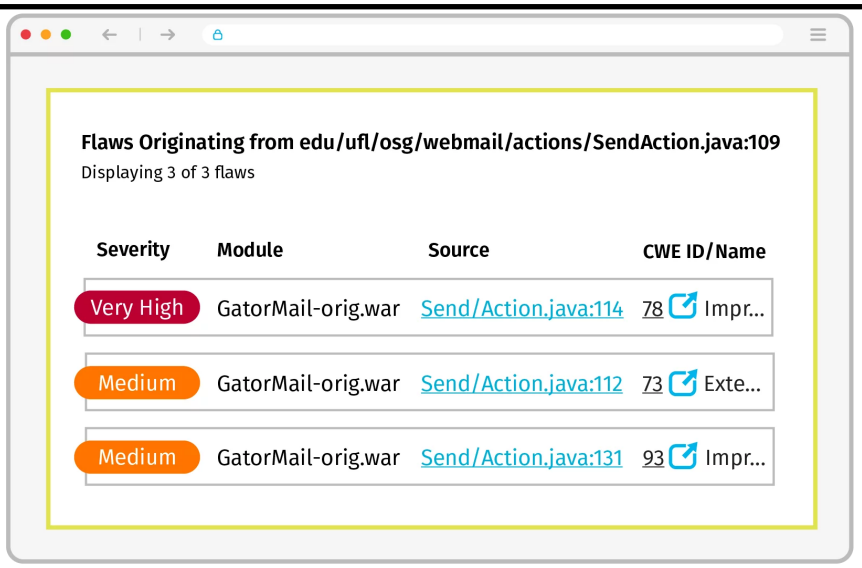


Figura 7.2.1.h: Imagen de errores por importancia de Veracode.

Si se usa mediante el plugin, este es el reporte. No tiene un dashboard, pero de la misma manera que en la aplicación web, se puede observar el conjunto de errores clasificados por importancia, y también se informa en qué clase del proyecto se encuentra.

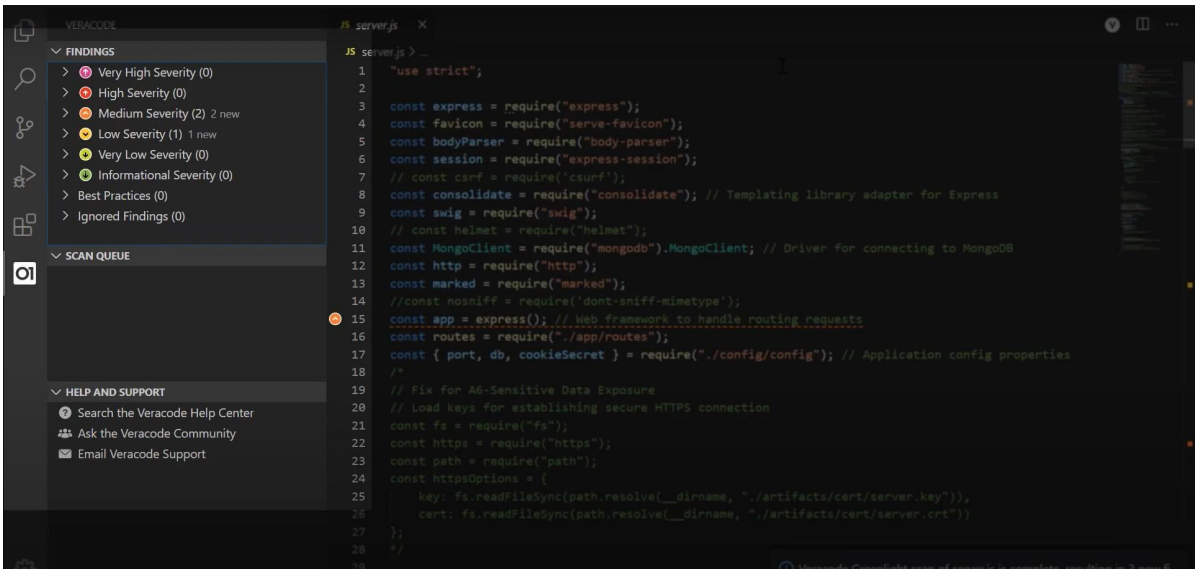


Figura 7.2.1.i: Muestra de resultados con el plugin de Veracode.

➢ Modelo lógico del sistema actual.

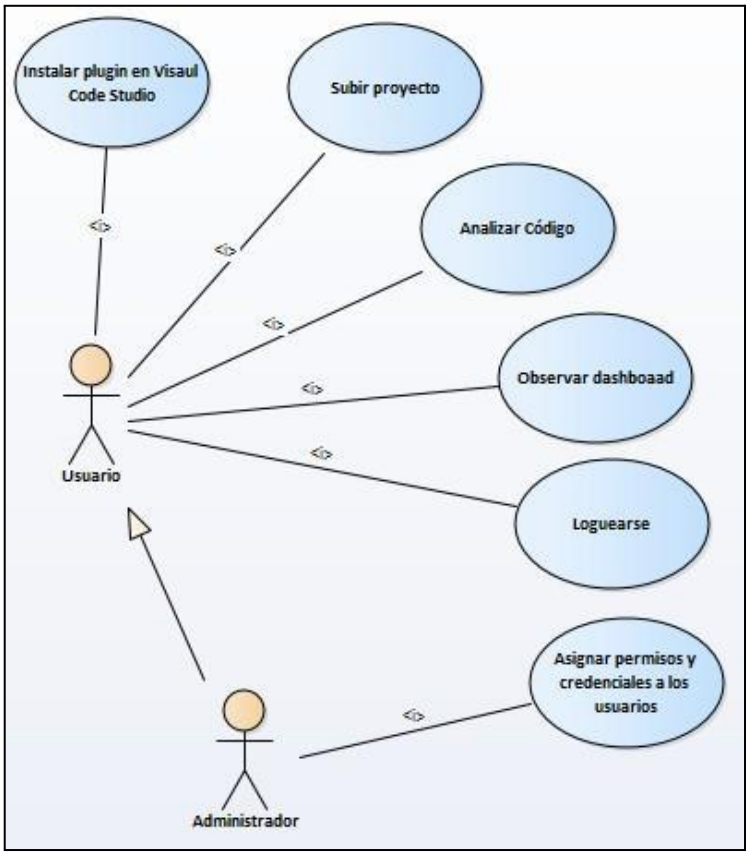


Diagrama 7.2.1.j: Diagrama de caso de usos de Veracode.

7.2.2.Problemas y necesidades detectados en las funciones relevadas en detalle y en su entorno organizacional.

- **Problema en el login de usuario:** Para poder crear el primer usuario de Administrador, se tiene que solicitar la autorización de parte de Veracode. Esto se realiza tanto para la demo como para el producto en sí.

- **Problema en el log de resultados de análisis:** Cuando se usa mediante el plugin, no se puede visualizar un dashboard.

- **Necesidad en el login de usuario:** Para poder ingresar al servicio se necesita un Usuario y una contraseña, cuando hay posibilidades de poder ingresar más fácil con la integración de GitHub.

- **Necesidad en el análisis de código:** Debido a que solamente realiza un análisis de seguridad, necesita un análisis más completo a la par que complejo para poder brindar información respecto de las métricas del código y documentación del mismo.

8.Comparativa Final.

	NDepend	Kiuwan	SonarQube	Deepscan	Veracode

Login de Usuario	--------	Cuenta de Kiuwan	Cuenta de SonarQube	Cuenta en Github	Cuenta en Veracode
Load de Código	--------	Con una aplicación	Con un archivo específico	Repositorio de GitHub	Una carga al servidor en la nube
Análisis de Código	Plugin	En la web	En la web	En la web	Plugin y en la Web
Log de Resultados de Análisis	Web	Web	Web	Web	Web y el IDE
Historial de Análisis	---------	Listado de análisis	Historial de Ejecuciones	----------	----------
Histórico de errores no resueltos en el tiempo	---------	---------	---------	Gráfica en web	---------
Análisis de Vulnerabilidades	---------	Análisis SAST	---------	---------	---------
Análisis de Software de terceros	---------	Análisis SCA	---------	---------	---------
Code Analysis	---------	Analizador de calidad de QA	---------	---------	---------
Governance y LifeCycle	---------	Gestión Continua	---------	---------	---------
Quality Profile	---------	---------	Gestión de reglas	---------	---------
Quality Gate	---------	---------	Gestión de deploy	---------	---------

9.Objetivos y alcances preliminares del nuevo Sistema.

Objetivos:

Diseñar una herramienta que analise código fuente para:

❖ Estudiar la calidad del código por medio de métricas y generar documentación de diseño y arquitectura del software.

❖ Generar un historial de reportes que permita el seguimiento de la evolución de la calidad del software a lo largo de su ciclo de vida.

Alcances:

El usuario utiliza la herramienta mediante una aplicación hosteada en la web, en la cual se puede registrar para poder tener un usuario y de esta manera tener acceso a un historial de reportes, sobre su código, que haya realizado a lo largo del tiempo. En la misma aplicación web se debe subir el código. A posteriori se podrá iniciar el análisis para visualizar el informe, en el cuál se muestran los errores categorizados según su gravedad y con la posibilidad de navegar entre estos para ver en qué clase se encuentran. También en el reporte, se puede ver el diagrama de clases del modelo de negocio y otros elementos de la infraestructura del proyecto de software, para poder tener está información actualizada siempre con la última versión.

10.Módulos del sistema:

Modulo de Usuarios: En el módulo de usuarios se diseñaron 3 tipo de usuarios para poder utilizar el sistema. Los usuarios administradores se ocupan del mantenimiento del sistema. Los usuarios registrados que podrán tener el historial a mano de los diferentes reportes para ver el progreso del proyecto y los usuarios invitados donde solo podrán tendrán accesos a algunas funciones del sistema.

Módulo Base de Análisis de Métricas: En este módulo se establece un análisis del código detallando en base a la metodología de UML para obtener reportes relevantes de la calidad de la misma.

Las métricas que se evalúan son:

- Cohesión.
- Acoplamiento.
- Profundidad en árbol de herencia.
- Cantidad de subclases de una clase.
- Complejidad ciclomática.
- Tamaño de métodos.
- Tamaño de clases.
- Longitud de nombre de atributos.

Módulo Base de Análisis de Documentación: Se analiza el código del proyecto para obtener una documentación del sistema para luego enviarlo al módulo de reportes. Se establece el análisis respecto de la coherencia y detalle de la calidad del código con un análisis sintáctico.

Módulo de Reportes: Es el módulo que se encarga de realizar el reporte de la información con los datos que reciba de los 2 módulos de análisis. El reporte contiene el diagrama de clases y contiene los resultados de la evaluación de cada una de las métricas mencionadas en el Módulo de Análisis de Métricas.
El módulo consiste en el desarrollo de las interfaces web y las conexiones del front-end en general, incluyendo la interfaz web de la página principal de Junkode, la de login, la de historiales, la documentación y control para administradores. También incluye la generación de documentación en formato PDF.

Módulo de Donaciones: Este módulo consiste en la integración de un sistema para la gestión de campañas de donación. Las donaciones realizadas por los usuarios tendrán como propósito ayudar a desarrollar el sistema.

Módulo de mensajería: Este módulo sirve para que el usuario pueda comunicarse con el administrador con los siguientes motivos: bug report, Dev quality, other. Facilitando así un mejor mantenimiento del sistema. El administrador que puede leer los mensajes de los usuarios, podrán filtrarlos por asuntos preestablecidos y administrar una blacklist para bloquear que los usuarios puedan enviar mensajes a los administradores.

Módulo de Información para administradores: Este consiste en una interfaz donde los administradores del sistema pueden consultar la información referida al uso de la herramienta por medio de Dashboards e indicadores. Los administradores tendrán la funcionalidad de poder poner a los usuarios con mensajes maliciosos en una blacklist.

Diseño

11.Objetivos y Alcances definitivos del Nuevo Sistema.

1.1.Objetivos:

Construir un software que se encargue de analizar código fuente con el fin de:

- ➢ Estudiar la calidad del código por medio de métricas y generar documentación de diseño y arquitectura del software.
- ➢ Generar un historial de reportes que permita el seguimiento de la evolución de la calidad del software a lo largo de su ciclo de vida.

11.2.Alcances:

El usuario utiliza la herramienta mediante una aplicación web, en la cual se puede registrar para tener una cuenta asociada a un usuario de GitHub y de esta manera tener acceso a un historial de los reportes sobre los resultados de los análisis del código de los repositorios que haya realizado con la herramienta. Ya registrado, puede seleccionar el repositorio y rama que contiene sus proyectos de software. A continuación, inicia el análisis del proyecto para obtener el informe generado, en el cuál se muestran los resultados categorizados en función de a qué elemento de la documentación pertenece. El usuario debe tener la capacidad de navegar entre estos resultados en busca de Información, smells y recomendaciones. Por otro lado, también existe la posibilidad de usar la herramienta como invitado, o sea sin tener una cuenta de GitHub, pero los usuarios invitados están ligados a contar con sus proyectos de software en su almacenamiento local y a no poder almacenar los resultados de sus análisis en el historial de Junkode.

11.3.Módulos del sistema Junkode:

Módulo de Usuarios: Hay solo dos aspectos sobre los que desarrollar seguridad: los usuarios y sus proyectos y reportes.

Para los usuarios se desarrollan los roles, junto con sus respectivos niveles de acceso.

Las credenciales de los usuarios no administradores son autenticadas mediante un servicio llamado OAuth2, que maneja las credenciales de GitHub de los usuarios. Configurarlo es la principal tarea de este punto en este módulo.

Por otro lado, la seguridad de los proyectos y sus respectivos reportes generados, se configuran credenciales y autenticaciones para las bases de datos en la nube y sus respaldos.

Módulo Base de Análisis de Métricas: Es el que utiliza los visitor de ANTLR para desglosar y analizar la información del código necesaria para obtener el informe de métricas, para luego enviarlo al Módulo de Reportes.

Las métricas que se evalúan son:
- ➢ Cohesión.
- ➢ Acoplamiento.
- ➢ Profundidad en árbol de herencia.
- ➢ Cantidad de subclases de una clase.
- ➢ Complejidad ciclomática.
- ➢ Tamaño de métodos.
- ➢ Tamaño de clases.
- ➢ Longitud de nombre de atributos.

Módulo Base de Análisis de Documentación: Es el que utiliza los visitor de ANTLR para destripar y analizar la información del código necesaria para obtener la documentación del sistema para luego enviarlo al Módulo de Reportes.

Luego, en función de cuáles archivos se encuentren en el proyecto o con qué tecnologías cuente, también pueden documentarse:
- ➢ Diagrama de entidades (Si el proyecto persiste entidades y emplea los frameworks Spring Boot o Micronaut).
- ➢ Información de Propiedades del sistema (Si el proyecto cuenta con un archivo “application.properties” o un archivo en formato YAML con el nombre “application” o “bootstrap”).
- ➢ Información de Dependencias (Si el proyecto gestiona sus dependencias con Maven o Gradle).
- ➢ Información sobre contenedores (Si el proyecto contiene un archivo Dockerfile).

Módulo de Reportes: El módulo consiste en el desarrollo de las interfaces web y las conexiones del front-end en general, incluyendo la interfaz web de la página principal de Junkode, la de login, historiales, documentación, guía de inicio rápido, mensajería y control para administradores (donde los administradores pueden consultar la información referida al sistema de la herramienta en forma de dashboards, responder la mensajería, gestionar la

blacklist y tener acceso a servicios especializados para gestionar las bases de datos, backups y donaciones).

Este módulo también incluye la generación de archivos en formato .pdf para los resultados de los análisis. Estos documentos son estáticos y no reactivos (como la página web), por lo que deben adaptar la presentación de la información para ofrecer una mayor comodidad y accesibilidad a quienes utilicen este formato.

Módulo de Donaciones: Este módulo consiste en la integración de las funcionalidades que ofrece Donorbox para la gestión de campañas de donación. Las tareas del módulo constan de la creación y configuración de la campaña de Donorbox y la integración de los artefactos que esta aporta al sistema (menú de donaciones y muro de donantes y comentarios) al diseño de la interfaz de donaciones de Junkode.

Las donaciones se realizan de forma única o periódica, y no tendrán limitaciones respecto de un mínimo, máximo, ni tipo de moneda.

Módulo de Mensajería: Este módulo consiste de dos partes: La primera es la del usuario registrado, el cual puede enviar mensajes a los administradores de Junkode. La segunda es la del Administrador, que puede leer los mensajes de los usuarios, filtrarlos por asuntos preestablecidos y administrar una blacklist para bloquear la funcionalidad de enviar mensajes a usuarios de la herramienta.

Módulo de Administradores: Aquí se desarrollan todas las funcionalidades previamente mencionadas a las que el administrador tendrá acceso. Consulta de métricas del desempeño del sistema, generación de backups, gestión de la mensajería y la blacklist y el manejo de los ABMs que componen el sistema.

Módulos incrementales de documentación: Como el nombre lo dice, este no es un único módulo, sino múltiples módulos, en los cuales se añaden nuevas funcionalidades y visitors a los módulos base para analizar nuevos elementos que puedan servir para analizar la documentación del proyecto. Estos serían incrementales y seguirán desarrollándose nuevos luego de la release de Junkode.

Ejemplos de estos módulos serían: un módulo que analice los archivos referentes a CD/CI, un módulo que agregue nuevas dependencias analizables (como AOP), o un módulo que analice servicios específicos (como Eureka o Spring Cloud Gateway) que se encuentren el proyecto.

Para la primera release del proyecto, se desarrollaron son siguientes Módulos incrementales de documentación:

+ Primer Módulo Incremental, Diagrama de Propiedades: Este módulo consiste en la incorporación de un nuevo diagrama a los resultados de un análisis. El diagrama contiene toda la información de las propiedades de un proyecto Java y se genera cuando se

encuentra un archivo “application.properties” o un archivo en formato YAML con las propiedades del proyecto.

Este módulo utiliza la misma API que se utilizó en el Módulo Base de Análisis de Documentación para generar el diagrama de clases, pero esta vez se le envía la información en formato YAML. También se necesita de una API nueva, llamada “app2yaml” la cual recibe un objeto JSON con el contenido de un archivo “application.properties”, y retorna otro objeto JSON con su equivalente en formato YAML.

+ Segundo Módulo Incremental, Listado de Constantes del sistema: En este módulo se incorporan algunas funciones “visit” a los visitors desarrollados en el Módulo Base de Análisis de Documentación para capturar la información referida a las constantes del sistema. Esta información es luego plasmada en una tabla que contiene los datos de declaración de la constante y la dirección de la clase donde se encuentra declarada.

+ Tercer Módulo Incremental, Observaciones de Buenas Prácticas UML: En este módulo se añaden funcionalidades a los visitors para registrar factores de la declaración de variables y clasificadores como su visibilidad y su capitalización. Esta información luego se plasma a modo de observaciones de buenas prácticas dentro del análisis de métricas de cada elemento del sistema.

Ejemplos de estas observaciones serían: Si se encuentra un atributo final y su nombre no está en mayúsculas, se debe mostrar una observación que indique “Se recomienda declara a los atributos constantes con nombres en mayúscula”, o si se encuentra un atributo con visibilidad “package” se muestre la observación “Se recomienda que los atributos tengan visibilidad private o protected”. Si a un mismo elemento le corresponde más de una observación, estas se concatenan para mostrarse.

12.Modelo Funcional.

El modelo funcional se realizó en un diagrama de clases en la herramienta Enterprise Architect. En este se confeccionó por completo el “negocio” del sistema y se incluyeron también los elementos propios del diseño que son relevantes para las entidades del negocio.

12.1.Diagrama de clases del negocio de Junkode.

Para ver el modelo funcional consulte el Anexo n°4 “Diagrama de clases del negocio de Junkode”.

12.2.Historias de Usuario.

Las siguientes historias de usuario contienen los requerimientos necesarios para el desarrollo de la herramienta Junkode. A continuación se encuentra un detalle de la nomenclatura y valores posibles de algunos atributos referidos en estas.

Usuario:

Se definieron 3 tipos de usuarios con necesidades diferentes para el sistema:

- Usuario Tipo 1: Usuario logeado por Github.
- Usuario Tipo 2: Usuario no logueado.
- Usuario Tipo 3: Usuario Admin de Junkode.

Prioridad:

- Alta: Es indispensable para que el sistema cumpla con su función fundamental.
- Media: Es una característica accesoria al sistema principal pero que resulta en una mejor experiencia para el usuario.
- Baja: Es una funcionalidad de apoyo de la cual el sistema podría prescindir, sin embargo su presencia aún aporta valor a la herramienta.

Estimación:
Para este ítem se plantearon Story Points siguiendo la escala que siguen las tallas de camisetas.

- XS: 1 día.
- S: 3 días.
- M: 7 días.
- L: 14 días.
- XL: 21 días.

Pantallas y Reportes:
En este ítem encontrará la pantalla relacionada con la historia de usuario. La forma de relacionarse es buscando la página que se menciona en la historia de usuario en el documento Anexo N°2: Pantallas y Reportes.

Criterios de aceptación:
Estas son las características que deben ser pasadas en las pruebas para comprobar que las funcionalidades especificadas en la historia de usuario hayan sido correctamente implementadas.

US-001			
Título	Log in GitHub		
Usuario	Como Usuario Tipo 1		
Función	Quiero ingresar con mi cuenta de GitHub		
Objetivo	Para tener acceso a mi información de GitHub desde Junkode		
Prioridad	Alta		
Estimación	M		
Pantallas y Reportes	Pantalla N°6		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Aceptación de acceso	Logueandose por GitHub	Cuando ingrese mis	Espero tener acceso

		credenciales correctamente	repositorios de mi cuenta de GitHub
Negación de acceso	Logueandose por GitHub	Cuando ingrese mal mis credenciales	Espero que no se pueda acceder a mis repositorios y se me indique que ingrese las credenciales nuevamente

US-002			
Título	Carga de Repositorios		
Usuario	Como Usuario Tipo 1		
Función	Quiero acceder a mis repositorios de GitHub		
Objetivo	Para ejecutar análisis sobre ellos		
Prioridad	Alta		
Estimación	M		
Pantallas y Reportes	Pantalla N°8		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Acceso a Repositorios	Carga de repositorios de la cuenta del usuario	Cuando quiero utilizar los repositorios	Espero tener acceso tanto los repositorios creados por mi, como los creados por terceros donde sea colaborador

US-003			
Título	Register con GitHub		
Usuario	Como Usuario Tipo 1		
Función	Quiero Crear una cuenta en Junkode con mi usuario de GitHub		
Objetivo	Para acceder a los repositorios de GitHub desde Junkode		
Prioridad	Alta		
Estimación	XL		
Pantallas y Reportes	Pantalla N°6		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Aceptación de credenciales	Registrándose a Junkode mediante usuario de GitHub	Cuando ingrese mis credenciales correctamente	Espero tener acceso a las credenciales de mi cuenta de GitHub

Negación de credenciales	Registrándose a Junkode mediante usuario de GitHub	Cuando ingrese mal mis credenciales	Espero que no se pueda acceder a mis credenciales y se me indique que las ingrese nuevamente
Usuario de Junkode	Registrándose a Junkode mediante usuario de GitHub	Cuando mis credenciales de GitHub sean aceptadas	Espero que se cree un nuevo usuario de Junkode con el usuario de GitHub para así no tener que registrarme nuevamente

US-004			
Título	Subida Proyecto		
Usuario	Como Usuario Tipo 1 o 2		
Función	Quiero subir mi proyecto a Junkode		
Objetivo	Para ejecutar el análisis sobre el proyecto		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°5 o Pantalla N°8		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Carga exitosa del archivo.	Carga del proyecto en la página de Junkode.	Cuando seleccione el directorio con el proyecto Java.	Espero que se me informe que el archivo se subió correctamente y que está listo para analizar.
Error a cargar el archivo.	Carga del proyecto en la página de Junkode.	Cuando seleccione un directorio con un formato no compatible con FAT32.	Espero que se me niegue la carga, se me indique que hubo un error en el formato y que vuelva a cargar
Carga mediante gestor de archivos.	Carga del proyecto en la página de Junkode.	Cuando se me pida que seleccione el directorio con el proyecto.	Espero que aparezca un gestor de archivos pop up donde pueda seleccionar el directorio con el proyecto a cargar.

US-005	
Título	Visualización general de Resultados
Usuario	Como Usuario Tipo 1 o 2

Función	Quiero ver los resultados del análisis realizado		
Objetivo	Para verificar que este todo correcto en en el código		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°11-15		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Resultados específicos del Análisis de Métricas	Se generó un reporte de un análisis de métricas	Cuando se genere el reporte.	Espero ver: Calificación particular de cada elemento del proyecto para cada métrica disponible y una lista de las calificaciones posibles
Resultados específicos del Análisis de Documentación	Se generó un reporte de un análisis de documentación	Cuando se genere el reporte	Espero ver: Diagrama de entidades del proyecto, información sobre contenedores e información de propiedades del proyecto.

US-006			
Título	Log in de Administrador		
Usuario	Como Usuario de tipo 3		
Función	Quiero ingresar a mi cuenta de administrador en la herramienta		
Objetivo	Para loguearme en la página		
Prioridad	Alta		
Estimación	M		
Pantallas y Reportes	Pantalla N°6		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Ingreso a mi cuenta en la página.	Loguearse como administrador.	Cuando Ingreso mi usuario y contraseña de mi cuenta (de GitHub) de administrador en la herramienta.	Espero que se valide las credenciales y me redireccione a mi cuenta
Negación de acceso a la cuenta.	Loguearse como administrador.	Cuando ingreso erróneamente mi usuario	Espero que no se validen las credenciales y me

		y contraseña de mi cuenta (de GitHub) de administrador en la herramienta.	redireccione a la página principal. Que me indique que se ingresaron mal usuario y contraseña.
Pantalla de Administrador.	Loguearse como administrador.	Cuando haya ingresado como administrador.	Espero que se me redirija a la interfaz principal de usuario administrador.

US-007			
Título	Creación de proyecto analizable		
Usuario	Como usuario de tipo 1		
Función	Quiero crear un nuevo proyecto		
Objetivo	Para que quede registrado en la herramienta		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°8		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Creacion solo para usuarios con cuenta	Creando un nuevo proyecto para usuarios registrados con GitHub	Cuando elijo crear un nuevo proyecto	Espero que se me pida poner un nombre al proyecto y asociarlo a un repositorio.
Actualización de lista lateral de proyectos	Tras haber creado el nuevo proyecto	Cuando vea la barra lateral de proyectos	Espero que aparezca primero en la lista el nuevo proyecto

US-008			
Título	Historial del proyecto		
Usuario	Como usuario de tipo 1		
Función	Quiero ver el historial de análisis del proyecto		
Objetivo	Para ver los resultados de los análisis realizados anteriormente		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°9		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia

Nomenclatura.	Visualización del historial de los análisis.	Cuando visualice la lista de resultados del historial.	Espero que sus nombres estén compuestos por el nombre de la rama analizada y la fecha y hora en la que se realizó el análisis.
Historial de resultados.	Visualización del historial de los análisis.	Cuando elijo ver historial de proyecto.	Espero ver todos los resultados obtenidos ordenados en orden cronológico.

US-009			
Título	Baja de proyecto analizable		
Usuario	Como usuario de tipo 1		
Función	Quiero eliminar el proyecto en la herramienta		
Objetivo	Para dar de baja el proyecto del sistema		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°17		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Eliminación del proyecto.	Dar de baja el proyecto de la herramienta.	Cuando seleccione eliminar proyecto.	Espero que me elimine el proyecto de mi cuenta en Junkode junto con los historiales de resultados.
Actualización de lista lateral de proyectos.	Tras haber eliminado el nuevo proyecto.	Cuando vea la barra lateral de proyectos.	Espero que ya no figure el proyecto que fue eliminado.

US-010	
Título	Cambio de nombre del proyecto
Usuario	Como usuario de tipo 1
Función	Quiero cambiar el nombre del proyecto
Objetivo	Para identificarlo de otra manera
Prioridad	Media
Estimación	M
Pantallas y Reportes	Pantalla N°18

Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Modificacion nombre	Modificación del nombre del proyecto	Cuando seleccione cambiar título del proyecto	Espero que me cambie el nombre de mi proyecto
Set de caracteres admisibles en el nombre del proyecto	Modificación del nombre del proyecto	Cuando ingrese un caracter distinto de los caracteres ASCII [32-126] o una palabra reservada que haga alusión a instrucciones SQL o entidades del sistema	Espero que la modificación no sea persistida y el proyecto figure con el nombre que tenía previo a este cambio
Actualización de lista lateral de proyectos	Tras haber cambiado el nombre del proyecto	Cuando vea la barra lateral de proyectos	Espero que el proyecto figure con su nuevo nombre

US-011			
Título	Generación de documentos		
Usuario	Como usuario Tipo 1		
Función	Quiero obtener un documento con los resultados del análisis del proyecto		
Objetivo	Para poder visualizar los resultados del análisis en un documento		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°16		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Generar el documento	Generar el pdf con el resultado del análisis	Cuando hago click en generar documentación	Espero que me cree un pdf con los resultados del análisis
Disponibilidad usuario tipo 1	Tras haber realizado el análisis	Cuando sea un usuario de tipo 1 y el análisis tenga un proyecto creado	Espero poder generar la documentación después de haber realizado el análisis o desde la interfaz del historial de análisis del proyecto

US-012	
Título	Análisis de métricas
Usuario	Como Usuario Tipo 1 o 2
Función	Quiero realizar el análisis sobre la calidad del código

Objetivo	Para analizar las métricas del código		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°10		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Generación de análisis de métricas.	Realizando el análisis de métricas.	Cuando se encuentren archivos con sintaxis java en mi proyecto.	Espero que se analicen las métricas y prácticas de los elementos de mi proyecto de forma objetiva.

US-013			
Título	Análisis para documentación		
Usuario	Como Usuario Tipo 1 o 2		
Función	Quiero realizar el análisis de generación de documentación		
Objetivo	Para obtener la documentación del código		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°10		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Generación de diagrama de clases.	Realizando el análisis de la documentación.	Cuando encuentren archivos java con anotaciones de entidades de Spring Boot o Micronaut en mi proyecto.	Espero que se genere un diagrama de clases de mi proyecto.

US-014			
Título	Descarga de documentos		
Usuario	Como Usuario Tipo 1		
Función	Quiero descargar la documentación		
Objetivo	Para tener localmente los resultados del análisis del proyecto		
Prioridad	Media		
Estimación	S		
	Pantalla N°16		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia

PDF del análisis	Descargando los resultados	Cuando seleccione ambos documentos	Espero la descarga de un archivo PDF con la documentación generada del análisis combinada con las métricas generadas por el análisis

US-015			
Título	Enlaces a la documentación de la herramienta		
Usuario	Como Usuario Tipo 1 o 2		
Función	Quiero acceder a la documentación de la herramienta		
Objetivo	Para ver los manuales de usuarios		
Prioridad	Media		
Estimación	XS		
Pantallas y Reportes	Pantalla N°2		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Manuales de Usuarios	Desde la página Documentation de la herramienta	Cuando seleccione el enlace del manual	Espero que me redireccione al manual de usuario
Repositorio de GitHub	Desde la página Documentation de la herramienta	Cuando seleccione el enlace del repositorio de GitHub	Espero que me redireccione al repositorio con el código fuente de Junkode
Canal de YouTube	Desde la página Documentation de la herramienta	Cuando seleccione el enlace del canal de YouTube de Junkode	Espero que me redireccione a la página principal del canal de Youtube de Junkode

US-016	
Título	Log out
Usuario	Como Usuario Tipo 1 o 3
Función	Quiero desloguearme de la página
Objetivo	Para salir de mi cuenta
Prioridad	Alta
Estimación	M
Pantallas y Reportes	Pantalla N°7
Criterios de aceptación	

Título	Contexto	Trigger	Consecuencia
Salida de la herramienta	Deslogueandose de la página	Cuando seleccione log out	Espero desloguearme y que se redireccione a la página para poder iniciar sesión

US-017			
Título	Interfaz Web		
Usuario	Como Usuario Tipo 1 o 2 o 3		
Función	Quiero acceder desde una interfaz web		
Objetivo	Para poder acceder desde cualquier navegador		
Prioridad	Media		
Estimación	M		
Pantallas y Reportes	-		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Responsive	Utilizando la herramienta	Cuando utilice la herramienta	Espero una interfaz que se adapte a dispositivos móviles y tabletas
Usabilidad	Utilizando la herramienta	Cuando utilice la herramienta	Espero una interfaz que sea amigable y de fácil uso

US-018			
Título	Visualización de Dashboard de estadísticas		
Usuario	Como Usuario Tipo 3		
Función	Quiero ver un Dashboard con estadísticas		
Objetivo	Para analizar las estadísticas del uso de la herramienta		
Prioridad	Alta		
Estimación	L		
Pantallas y Reportes	Pantalla N°23		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Visualización de Dashboard	Viendo la estadísticas	Cuando acceda a las estadísticas	Espero ver un Dashboard de estadísticas resumiendo la información más importante en el mismo

Lista de Indicadores	Viendo la estadísticas	Cuando acceda a las estadísticas	Espero ver específicamente: Gráfico de proporción de análisis con diagramas de clases, gráfico de proporción de análisis con diagrama de configuración,gráfico de línea de la cantidad de análisis realizados por mes y gráfico de línea de la cantidad de mensajes enviados por dia

US-019			
Título	Compatibilidad con análisis de contenedores		
Usuario	Como Usuario Tipo 1 o 2		
Función	Quiero que se analicen los datos de los contenedores presentes en mi proyecto		
Objetivo	Para poder visualizarlos en los reportes generados		
Prioridad	Media		
Estimación	M		
Pantallas y Reportes	Pantalla N°11-15		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Dockerfile	Análisis de documentación del proyecto	Cuando se encuentre un archivo dockerfile en el proyecto	Espero obtener información acerca del contendor (Imagen sobre la que se construye y versión y link de la imagen actual del proyecto)

US-020	
Título	Compatibilidad con análisis de Bases de datos
Usuario	Como Usuario Tipo 1 o 2
Función	Quiero que se analicen los metadatos de las conexiones a bases de datos presentes en mi proyecto
Objetivo	Para poder visualizarlos en los reportes generados
Prioridad	Alta

Estimación	L		
Pantallas y Reportes	Pantalla N°11-15		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Conexiones a bases de datos	Análisis de documentación del proyecto	Cuando se encuentre una conexión a una base de datos en las dependencias o propiedades del proyecto	Espero obtener información acerca de la base de datos (nombre, motor, puertos, credenciales y demás valores específicos para cada motor)
Diagrama relacional	Análisis de documentación del proyecto	Cuando se encuentre una conexión a una base de datos relacional en las dependencias o propiedades del proyecto	Espero obtener el diagrama relacional con las tablas que integran la base de datos

US-021			
Título	Auditoría		
Usuario	Como Usuario Tipo 3		
Función	Quiero que las modificaciones de los proyectos queden auditadas		
Objetivo	Para facilitar la recuperación en caso de que la tabla de proyectos se corrompa y dar en un futuro la posibilidad de restablecer esos datos		
Prioridad	Media		
Estimación	S		
Pantallas y Reportes	Pantalla N°24		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Auditoría de la creación de proyectos.	Tras la creación de un proyecto.	Cuando se cree un nuevo proyecto.	Espero que se genere una nueva entrada en la tabla de auditoría de proyecto informando la creación de esta nueva instancia.
Auditoría de la modificación de proyectos.	Tras la modificación de un proyecto.	Cuando se modifique un nuevo proyecto.	Espero que se genere una nueva entrada en la tabla de auditoría de

			proyecto informando la modificación de esta instancia. Esta no debe limitarse a la modificación del nombre, sino a la modificación de cualquier atributo no relacional de esta.
Auditoría de la baja de proyectos.	Tras la baja de un proyecto.	Cuando se elimine un nuevo proyecto.	Espero que se genere una nueva entrada en la tabla de auditoría de proyecto informando la baja de esta instancia.

US-022			
Título	Donaciones		
Usuario	Como Usuario Tipo 1 o 2		
Función	Quiero hacer donaciones a Junkode		
Objetivo	Para contribuir con el proyecto		
Prioridad	Baja		
Estimación	M		
Pantallas y Reportes	Pantalla N°4		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Interfaz de Donación	En la página Home de Junkode	Cuando seleccione la opción de donación	Espero que se me redirija a una interfaz para hacer la donación
Lista de donadores	En la interfaz de donación	Cuando entre la interfaz	Espero ver una lista con los nombres y imagenes de los usuarios que recientemente han contribuido con donaciones para Junkode

US-023	
Título	Backups
Usuario	Como Usuario Tipo 3

Función	Quiero gestionar copias de seguridad de las bases de datos		
Objetivo	Para tener respaldo de información en caso de que haya que hacer una recuperación de datos		
Prioridad	Media		
Estimación	L		
Pantallas y Reportes	Pantalla N°25,26		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Frecuencia de los Backups	Realizando la configuración del Backup	Cuando se implementen los backups	Espero que se guarden copias una vez cada dos semanas de las bases de datos
Ubicación de los Backups	Realizando la configuración del Backup	Cuando se implementen los backups	Espero que se guarden las copias en una drive en la nube, de forma que puedan ser consultadas por los administradores en cualquier momento y desde cualquier dispositivo
Borrado de los Backups	Realizando la configuración del Backup	Cuando se implementen los backups	Espero que las copias con más de dos meses de antigüedad se eliminen automáticamente
Formato de los Backups	Realizando la configuración del Backup	Cuando se implementen los backups	Espero que las copias reciban la nomenclatura "Año-Mes-Dia-Base_de_Datos" y se compriman en formato .zip antes para guardarlas de forma ordenada y que precise de un menor espacio almacenamiento.
Enlace a SQL Bak	En la interfaz de Administrador	Cuando seleccione el enlace a SQL bak	Espero que me dirija al dashboard de jobs de SQL Bak

US-024			
Título	Mensaje Usuario		
Usuario	Como Usuario Tipo 1		
Función	Quiero enviar mensajes a los administradores		
Objetivo	Para consultar algún problema.		
Prioridad	Baja		
Estimación	M		
Pantallas y Reportes	Pantalla N°20		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Mensajería	Luego de estar logueado como usuario	Cuando seleccione la opción de Message	Espero que se me redirija a una interfaz para enviar un nuevo mensaje
Nuevo mensaje	En la interfaz de mensajería	Cuando entre la interfaz	Espero poder crear un nuevo mensaje eligiendo la razón y con el contenido del mismo

US-025			
Título	Mensaje Usuario		
Usuario	Como Usuario Tipo 3		
Función	Quiero meter en la blacklist a los usuarios.		
Objetivo	Para que el usuario que esté dentro no pueda enviar más mensajes		
Prioridad	Baja		
Estimación	M		
Pantallas y Reportes	Pantalla N°22		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Blacklist	Luego de estar logueado como administrador	Cuando seleccione la opción de Blacklist	Espero que se me redirija a una interfaz para meter en la blacklist a un usuario
Nuevo integrante de la Blacklist	En la interfaz de Blacklist	Cuando entre la interfaz	Espero poder elegir al usuario que quiera y meterlo dentro de la BlackList

US-026			
Título	Mensaje Usuario		
Usuario	Como Usuario Tipo 3		
Función	Quiero agregar una nueva razón		
Objetivo	Para que el usuario de tipo 1 pueda elegirla al mandar un mensaje.		
Prioridad	Baja		
Estimación	M		
Pantallas y Reportes	Pantalla N°21		
Criterios de aceptación			
Título	Contexto	Trigger	Consecuencia
Reasons	Luego de estar logueado como administrador	Cuando seleccione la opción de Reasons	Espero que se me redirija a una interfaz para añadir una nueva razón
Nuevo razón	En la interfaz de Reasons	Cuando entre la interfaz	Espero poder crear una nueva razón

12.3.Diagramas BPMN.

Para ver el flujo de información en los procesos de Junkode en notación BPMN 2.0, consulte el Anexo n°6 “Diagramas BPMN”.

13.Pantallas y Reportes.

Se realizaron diseños de las pantallas de usuario utilizando la herramienta la librería de Front-End reactiva Vue.js. En estos se pueden ver plasmadas las opciones e información que el sistema provee a sus usuarios..

Para ver las pantallas consulte el Anexo n°7 “Pantallas y reportes de Junkode”.

14.Modelo De Datos.

El modelo de datos es un modelo relacional que representa las tablas que se almacenarán en las bases de datos una vez que ya esten implementadas, incluyendo entidades, auditoría y versionado.

Para el modelo de datos consulte el Anexo n°15 “Diagrama Relacional de Junkode”.

Desarrollo e Implementación

15.Programación y Documentación.

A continuación se presenta el código fuente desarrollado por Junkode para el Módulo Base de Análisis de Métricas, junto con la documentación de comentarios correspondiente.

Denominación del módulo.	Análisis de Métricas.
Descripción.	Utiliza los visitor de ANTLR para desglosar y analizar la información del código necesaria para obtener el informe de métricas, para luego enviarlo al Módulo de Reportes.
Nombre y apellido de programador.	Renzo Fernandez.
Nombre y apellido del usuario que aprobó.	Ramses Girala.
Fecha de inicio de la programación.	10/08/2022.
Fecha de finalización.	08/09/2022.
User Story asociada.	US-005 - Visualización general de Resultados, US-009 - Historial del proyecto, US-013 - Análisis de métricas, US-027 - Auditoría.
Clases involucradas.	metricResultado, proyectoResultado, Clase, Atributo, Método
Casos de pruebas utilizados.	Prueba n°04-Complejidad ciclomática. Prueba n°05 -Cantidad Hijos.
Resultados de las pruebas.	Prueba n°04-Fracaso. Prueba n°05 -Fracaso.
Tecnologías utilizadas.	ANTLR 4 ,C#,ASP.NET 6.

Tabla 15.a: Planilla de desarrollo del Módulo Base de Análisis de Métricas.

15.1.Recolección de la Información.

El trabajo del módulo comienza con los visitors de ANTLR. Estos, son clases extraen la información de relevancia para la herramienta de las clases del proyecto seleccionado por el usuario.

MainVisitor.cs* MainMetricas.cs*

Application test.Application.Analyzer.Visitors.Java.MainVisitor

```
1  using Antlr4.Runtime.Misc;
2  using test.Application.Analyzer.Visitors.Java.Code;
3  using test.Application.ANTLR;
4  using test.Application.Model.Code;
5  using static test.Application.ANTLR.JavaParser;
6
7  namespace test.Application.Analyzer.Visitors.Java;
8
9  //MainVisitor en donde se realizan las llamadas a los distintos visitors necesarios
   3 referencias
10 public class MainVisitor : JavaParserBaseVisitor<object>
11 {
       6 referencias
12     public Interface Interface { get; set; }
       6 referencias
13     public Class Class { get; set; }
       3 referencias
14     public string PackageName { get; set; }
15
16     private readonly PackageVisitor packageVisitor;
       2 referencias
17     public MainVisitor(Class Class, Interface Interface, PackageVisitor packageVisitor)
18     {
19         this.Interface = Interface;
20         this.Class = Class;
21         this.packageVisitor = packageVisitor;
22     }
23
24     //En este metodo se analiza si la clase o interface analizada tiene algun error sintactico
       4 referencias
25     public override object VisitCompilationUnit(CompilationUnitContext context)
26     {
27         if (context.exception != null) {
28             this.Class.ErrorInGrammar = true;
29             this.Interface.ErrorInGrammar = true;
30         }
31
32         return base.VisitCompilationUnit(context);
33     }
34     //En este metodo se obtiene el nombre del paquete de la clase o interface
```

Figura 15.1.a: Código fuente documentado de MainVisitor.cs parte uno.

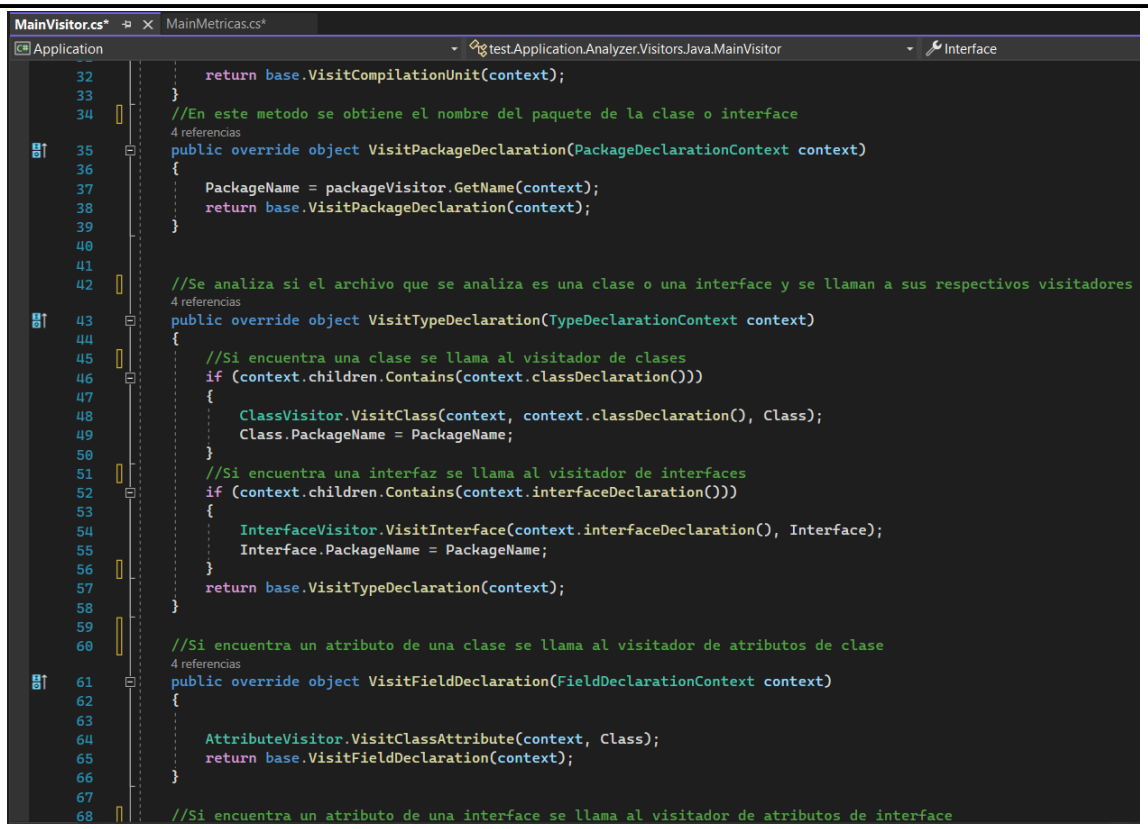


Figura 15.1.b: Código fuente documentado de MainVisitor.cs parte dos.

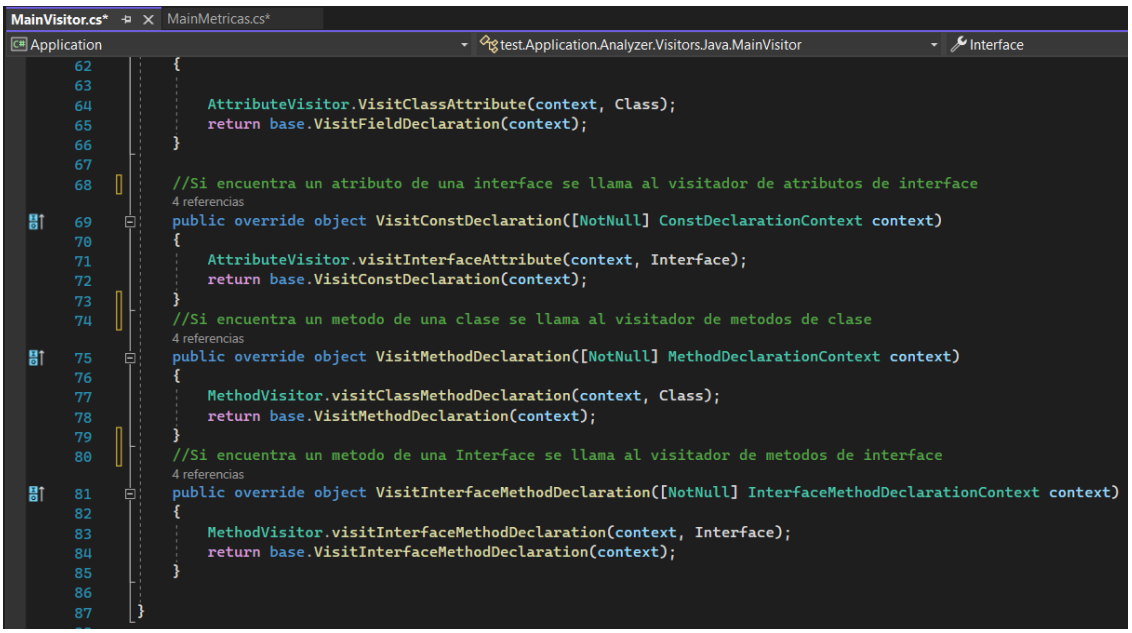


Figura 15.1.c: Código fuente documentado de MainVisitor.cs parte tres.

ClassVisitor.cs*

Application | test.Application.Analyzer.Visitors.Java.Code.ClassVisitor | VisitClass(TypeDeclarationC

```
1 referencia
public class ClassVisitor
{
    //Metodo que extrae la informacion mas importante de una clase, al cual se le pasa como parametros
    //dos contextos necesarios y una clase en la que se guarda la informacion
    1 referencia
    public static void VisitClass(TypeDeclarationContext contextT, ClassDeclarationContext context, Class clase)
    {
        //Se obtiene el nombre de la clase
        clase.Name = context.IDENTIFIER().GetText();
        //Condicional para saber si el nombre de la clase empieza con minuscula
        if (Char.IsLower(clase.Name[0])){
            clase.StartWithLower = true;
        }
        //Bucle for para saber si la clase hereda de alguna otra y guardar el nombre de la clase padre
        for (int i = 0; i < context.ChildCount; i++)
        {
            if (context.GetChild(i).GetText() == "extends")
            {
                clase.NameFather = context.typeType().classOrInterfaceType().GetText();
            }
        }
        //Bucle para saber si la clase es una entidad , es publica y es abstracta
        foreach (ClassOrInterfaceModifierContext x in contextT.classOrInterfaceModifier())
        {
            if (x.GetText() == "@Entity")
            {
                clase.IsEntity = true;
            }
            if (x.GetText() == "public")
            {
                clase.Visibility = "public";
            }
            if (x.GetText() == "abstract")
            {
                clase.isAbstract = true;
            }
        }
    }
}
```

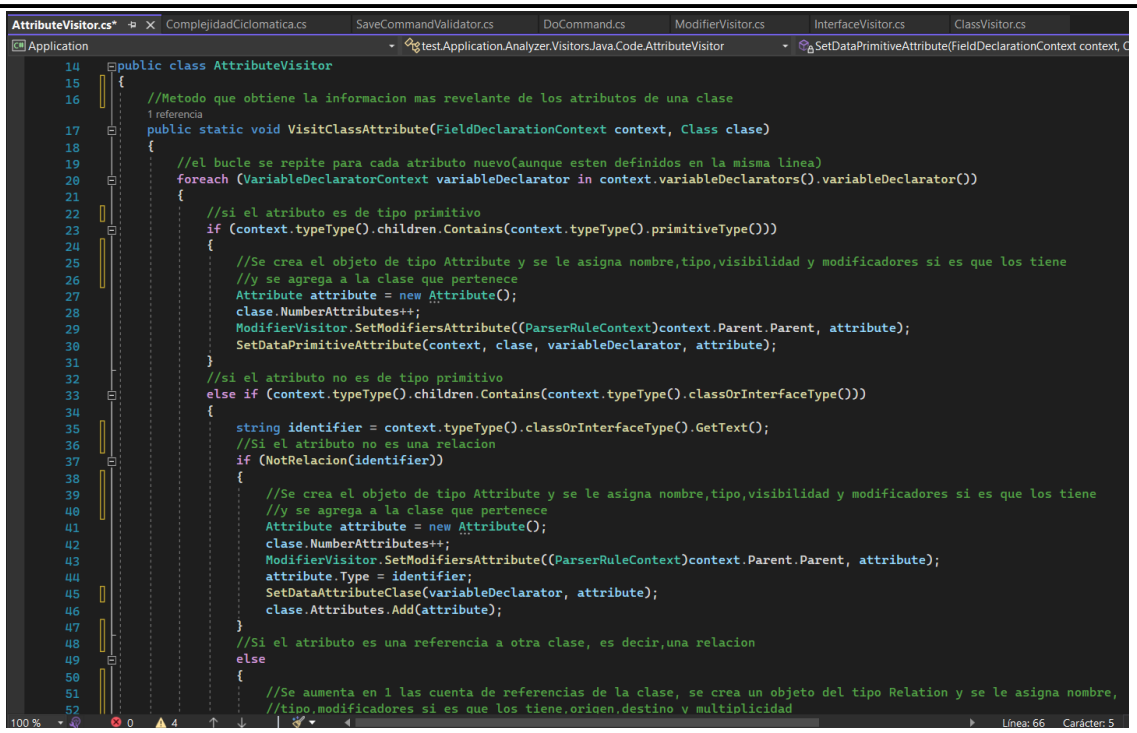
Figura 15.1.d: Código fuente documentado de ClassVisitor.cs.

InterfaceVisitor.cs* | ClassVisitor.cs*

Application | test.Application.Analyzer.Visitors.Java.Code.InterfaceVisitor | VisitInterface(Inter

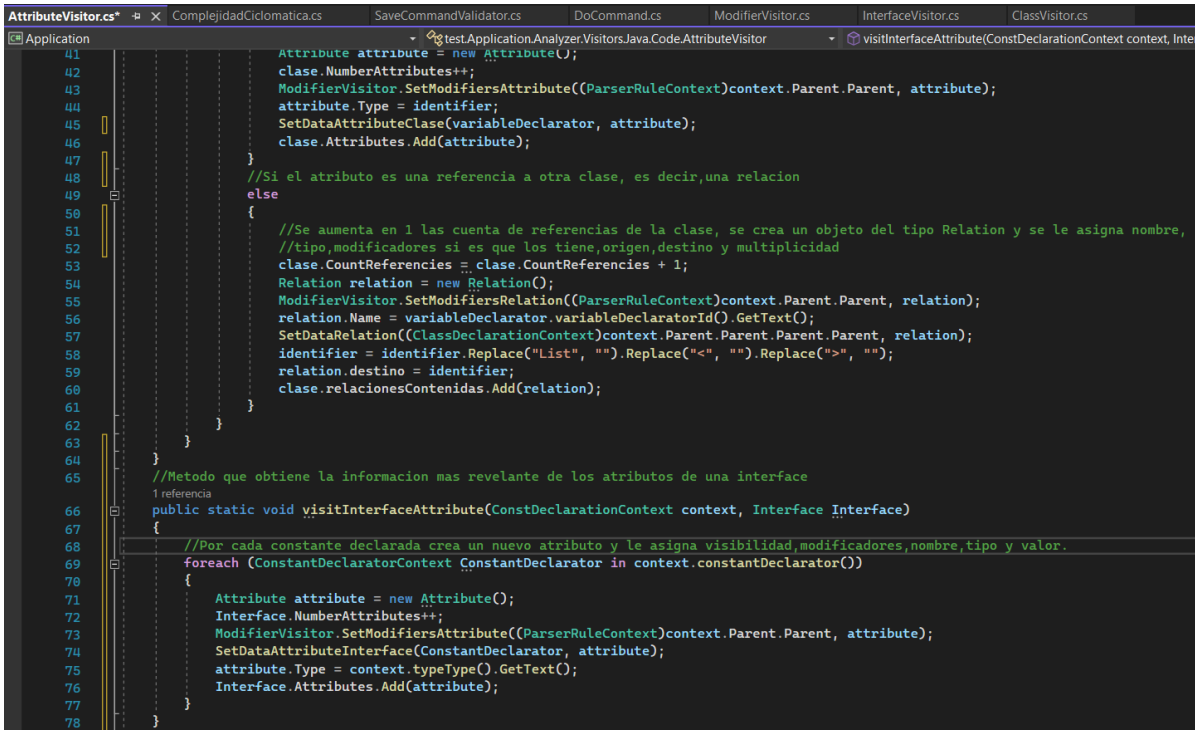
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using test.Application.Model.Code;
using static test.Application.ANTLR.JavaParser;

namespace test.Application.Analyzer.Visitors.Java.Code;
1 referencia
public class InterfaceVisitor
{
    //Metodo que extrae la informacion mas importante de una interface, al cual se le pasa como parametros
    //el contextos necesario y una interface en la que se guarda la informacion
    1 referencia
    public static void VisitInterface(InterfaceDeclarationContext context, Interface interfaz)
    {
        //Se guarda el nombre de la interface
        interfaz.Name = context.IDENTIFIER().GetText();
        //Bucle para saber cuantas lineas tiene
        foreach (InterfaceBodyDeclarationContext x in context.interfaceBody().interfaceBodyDeclaration())
        {
            interfaz.NumberLinesFile = interfaz.NumberLinesFile + 1;
        }
        //Se guardan todas las herencias que tenga la interface
        if (context.GetText().Contains("extends"))
        {
            foreach (TypeTypeContext x in context.typeList().typeType())
            {
                interfaz.NameFathers.Add(x.GetText());
            }

        }
    }
}
```

Figura 15.1.e: Código fuente documentado de InterfaceVisitor.cs.



```
public class AttributeVisitor
{
    //Metodo que obtiene la informacion mas revelante de los atributos de una clase
    1 referencia
    public static void VisitClassAttribute(FieldDeclarationContext context, Class clase)
    {
        //el bucle se repite para cada atributo nuevo(aunque esten definidos en la misma linea)
        foreach (VariableDeclaratorContext variableDeclarator in context.variableDeclarators().variableDeclarator())
        {
            //si el atributo es de tipo primitivo
            if (context.typeType().children.Contains(context.typeType().primitiveType()))
            {
                //Se crea el objeto de tipo Attribute y se le asigna nombre,tipo,visibilidad y modificadores si es que los tiene
                //y se agrega a la clase que pertenece
                Attribute attribute = new Attribute();
                clase.NumberAttributes++;
                ModifierVisitor.SetModifiersAttribute((ParserRuleContext)context.Parent.Parent, attribute);
                SetDataPrimitiveAttribute(context, clase, variableDeclarator, attribute);
            }
            //si el atributo no es de tipo primitivo
            else if (context.typeType().children.Contains(context.typeType().classOrInterfaceType()))
            {
                string identifier = context.typeType().classOrInterfaceType().GetText();
                //Si el atributo no es una relacion
                if (NotRelacion(identifier))
                {
                    //Se crea el objeto de tipo Attribute y se le asigna nombre,tipo,visibilidad y modificadores si es que los tiene
                    //y se agrega a la clase que pertenece
                    Attribute attribute = new Attribute();
                    clase.NumberAttributes++;
                    ModifierVisitor.SetModifiersAttribute((ParserRuleContext)context.Parent.Parent, attribute);
                    attribute.Type = identifier;
                    SetDataAttributeClase(variableDeclarator, attribute);
                    clase.Attributes.Add(attribute);
                }
                //Si el atributo es una referencia a otra clase, es decir,una relacion
                else
                {
                    //Se aumenta en 1 las cuenta de referencias de la clase, se crea un objeto del tipo Relation y se le asigna nombre,
                    //tipo,modificadores si es que los tiene,origen,destino y multiplicidad
```

Figura 15.1.f: Código fuente documentado de AttributeVisitor.cs parte uno.



```
                    Attribute attribute = new Attribute();
                    clase.NumberAttributes++;
                    ModifierVisitor.SetModifiersAttribute((ParserRuleContext)context.Parent.Parent, attribute);
                    attribute.Type = identifier;
                    SetDataAttributeClase(variableDeclarator, attribute);
                    clase.Attributes.Add(attribute);
                }
                //Si el atributo es una referencia a otra clase, es decir,una relacion
                else
                {
                    //Se aumenta en 1 las cuenta de referencias de la clase, se crea un objeto del tipo Relation y se le asigna nombre,
                    //tipo,modificadores si es que los tiene,origen,destino y multiplicidad
                    clase.CountReferencies = clase.CountReferencies + 1;
                    Relation relation = new Relation();
                    ModifierVisitor.SetModifiersRelation((ParserRuleContext)context.Parent.Parent, relation);
                    relation.Name = variableDeclarator.variableDeclaratorId().GetText();
                    SetDataRelation((ClassDeclarationContext)context.Parent.Parent.Parent.Parent, relation);
                    identifier = identifier.Replace("List", "").Replace("<", "").Replace(">", "");
                    relation.destino = identifier;
                    clase.relacionesContenidas.Add(relation);
                }
            }
        }
    }
    //Metodo que obtiene la informacion mas revelante de los atributos de una interface
    1 referencia
    public static void visitInterfaceAttribute(ConstDeclarationContext context, Interface Interface)
    {
        //Por cada constante declarada crea un nuevo atributo y le asigna visibilidad,modificadores,nombre,tipo y valor.
        foreach (ConstantDeclaratorContext ConstantDeclarator in context.constantDeclarator())
        {
            Attribute attribute = new Attribute();
            Interface.NumberAttributes++;
            ModifierVisitor.SetModifiersAttribute((ParserRuleContext)context.Parent.Parent, attribute);
            SetDataAttributeInterface(ConstantDeclarator, attribute);
            attribute.Type = context.typeType().GetText();
            Interface.Attributes.Add(attribute);
        }
    }
```

Figura 15.1.g: Código fuente documentado de AttributeVisitor.cs parte dos.

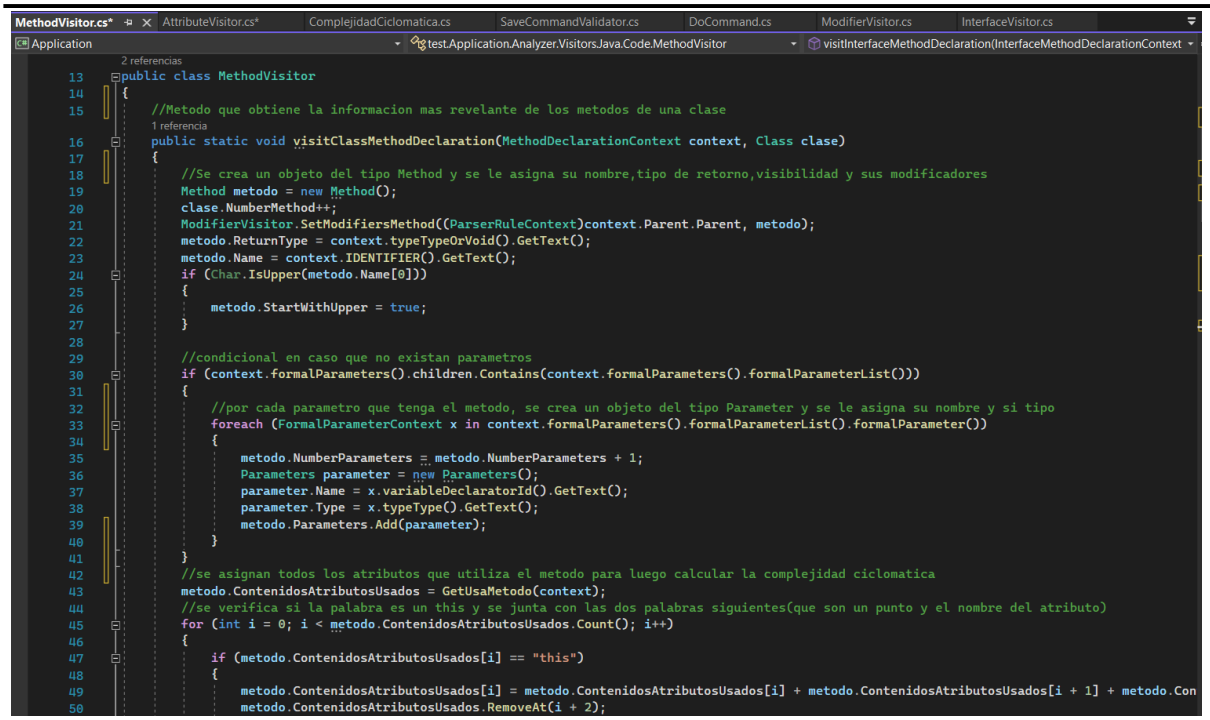


Figura 15.1.h: Código fuente documentado de MethodVisitor.cs parte uno.

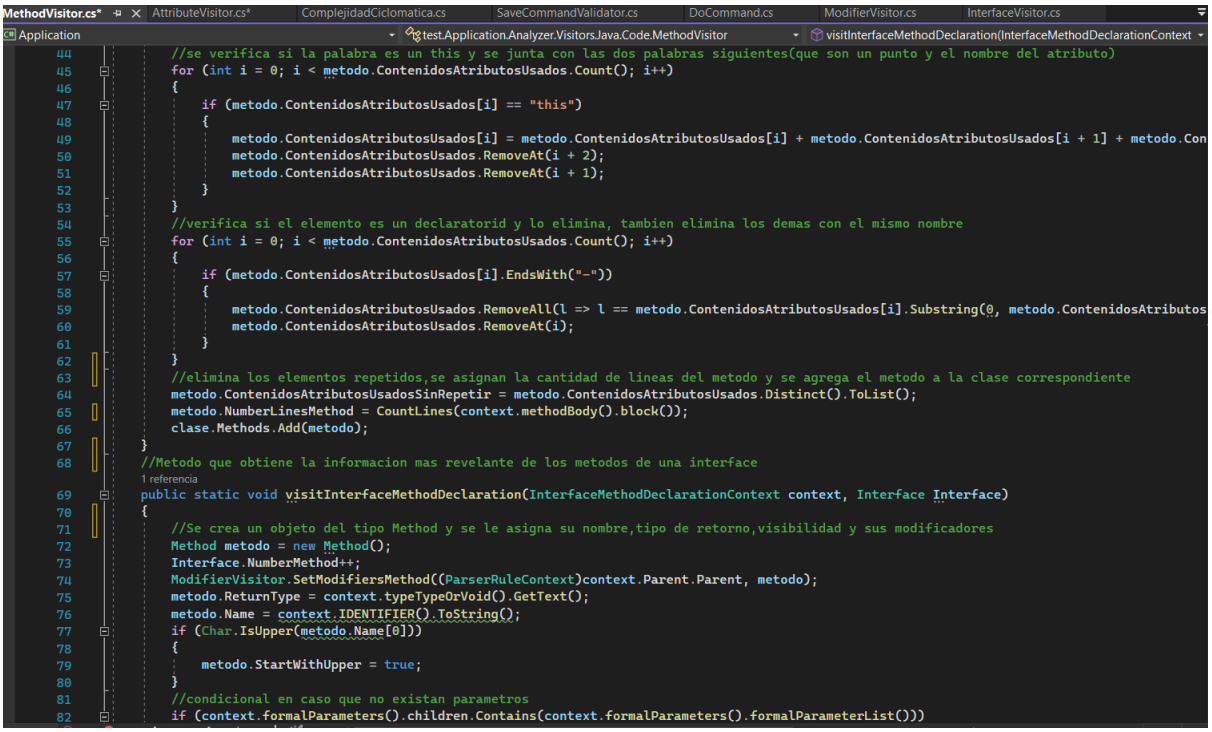


Figura 15.1.i: Código fuente documentado de MethodVisitor.cs parte dos.

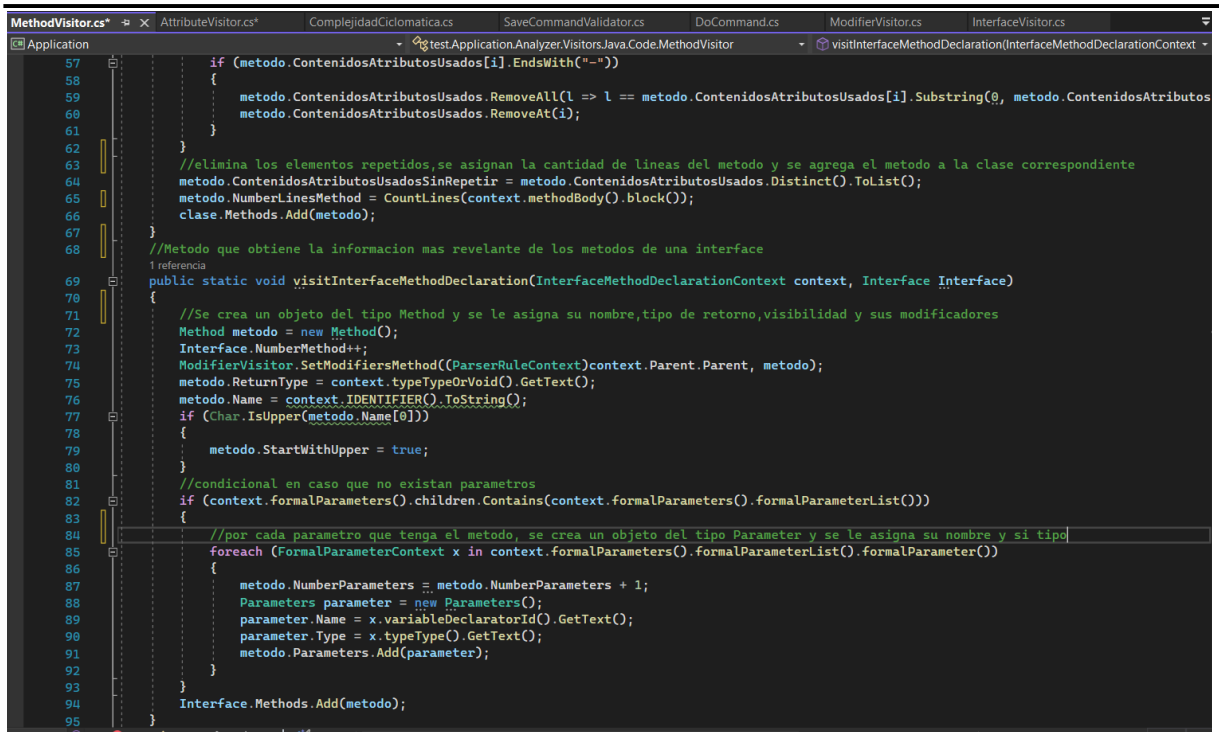


Figura 15.1.j: Código fuente documentado de MethodVisitor.cs parte tres.

15.2.Procesamiento de la información y Cálculo de las métricas.

Una vez que se obtiene toda la información de mayor importancia, se procede a analizarla para así calcular las métricas.

MainMetricas.cs* | Cohesion.cs* | CantidadHijos.cs* | MethodVisitor.cs | AttributeVisitor.cs

Application — test.Application.Analyzer.Metricas.MainMetricas

```
using test.Application.Model.Code;

namespace test.Application.Analyzer.Metricas;
2 referencias
public class MainMetricas
{
    2 referencias
    public void GetMetricas(List<Classifier> classifiers)
    {
        //FOREACH PARA RECORRER TODOS LOS CLASIFICADORES
        //LLAMADO A TODOS LOS METODOS QUE CALCULAN METRICAS DE CLASES
        foreach (var clasificador in classifiers)
        {
            Acoplamiento.TieneAcoplamiento(clasificador);
            TamañoClaseInterface.TamañoArchivo(clasificador);
            ProfundidadHerencia.profundidadHerencia(classifiers, clasificador);
            CantidadHijos.cantidadHijos(classifiers, clasificador);
            Cohesion.cohesion(clasificador);
            CantidadAtributos.cantidadAtributos(clasificador);
            CantidadMetodos.cantidadMetodos(clasificador);

            //FOREACH PARA RECORRER TODOS LOS METODOS DE UNA CLASE
            //LLAMADO A TODOS LOS METODOS QUE CALCULAN METRICAS DE METODOS
            foreach (var metodo in clasificador.Methods)
            {
                TamañoMetodos.TamañoMetodo(metodo);
                ComplejidadCiclomatica.complejidadCiclomatica(metodo);
            }

            //FOREACH PARA RECORRER TODOS LOS ATRIBUTOS DE UNA CLASE
            //LLAMADO A TODOS LOS METODOS QUE CALCULAN METRICAS DE ATRIBUTOS
            foreach (var atributos in clasificador.Attributes)
            {
                LongitudAtributo.longitudAtributo(atributos);

            }

        }
    }
}
```

Figura 15.2.a: Código fuente documentado de MainMetrics.cs.

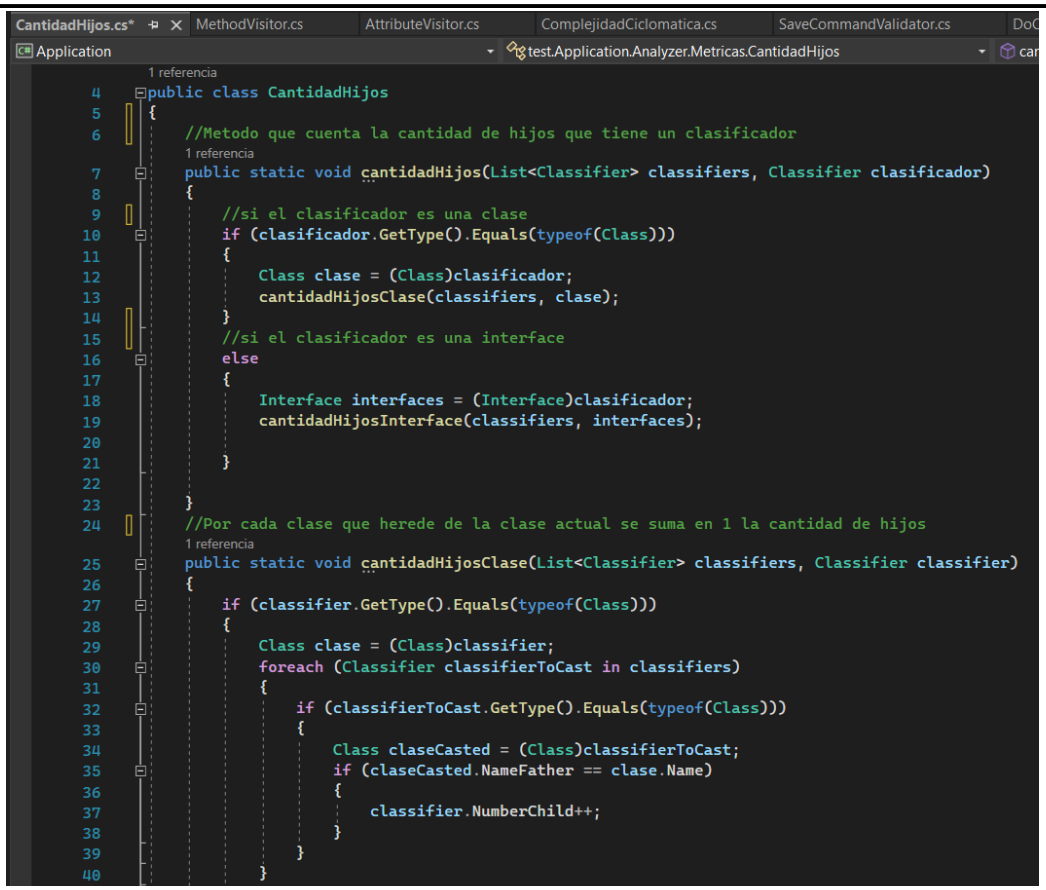


Figura 15.2.b: Código fuente documentado de CantidadHijos.cs parte uno.

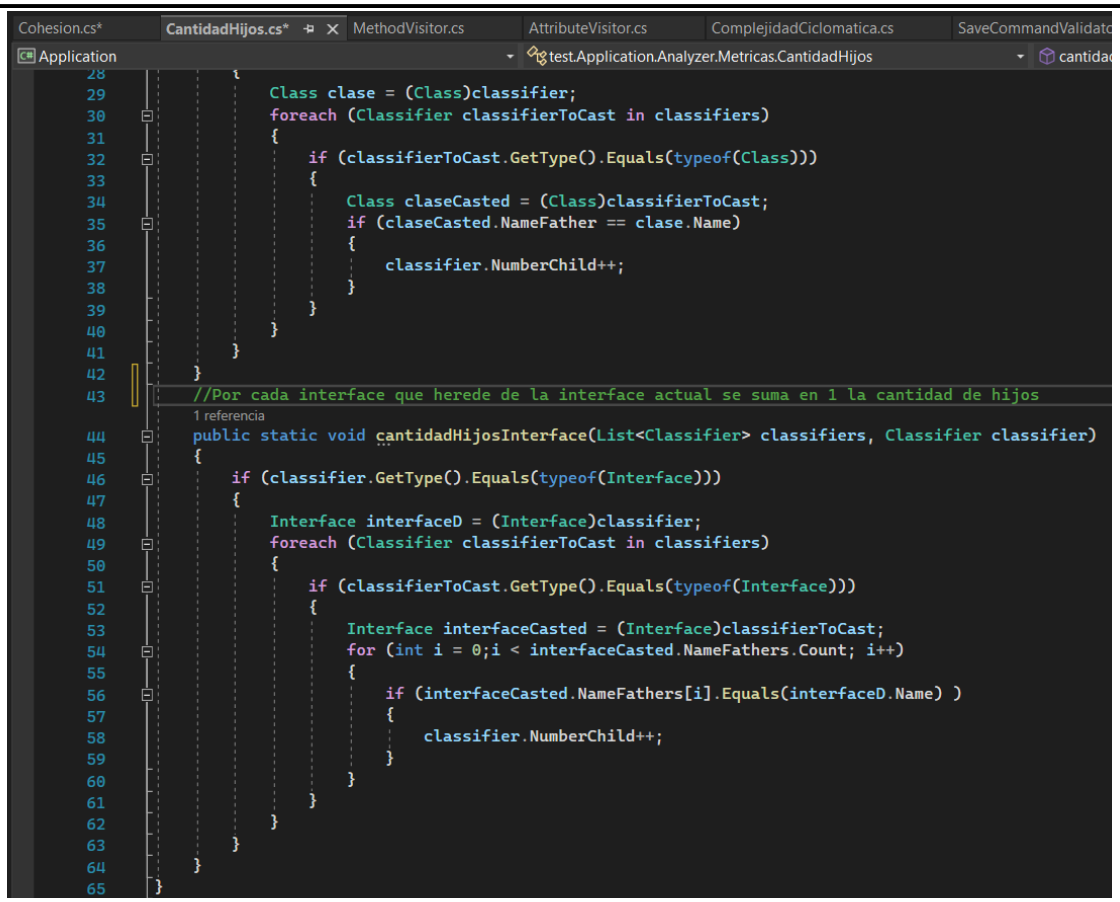


Figura 15.2.c: Código fuente documentado de CantidadHijos.cs parte dos.

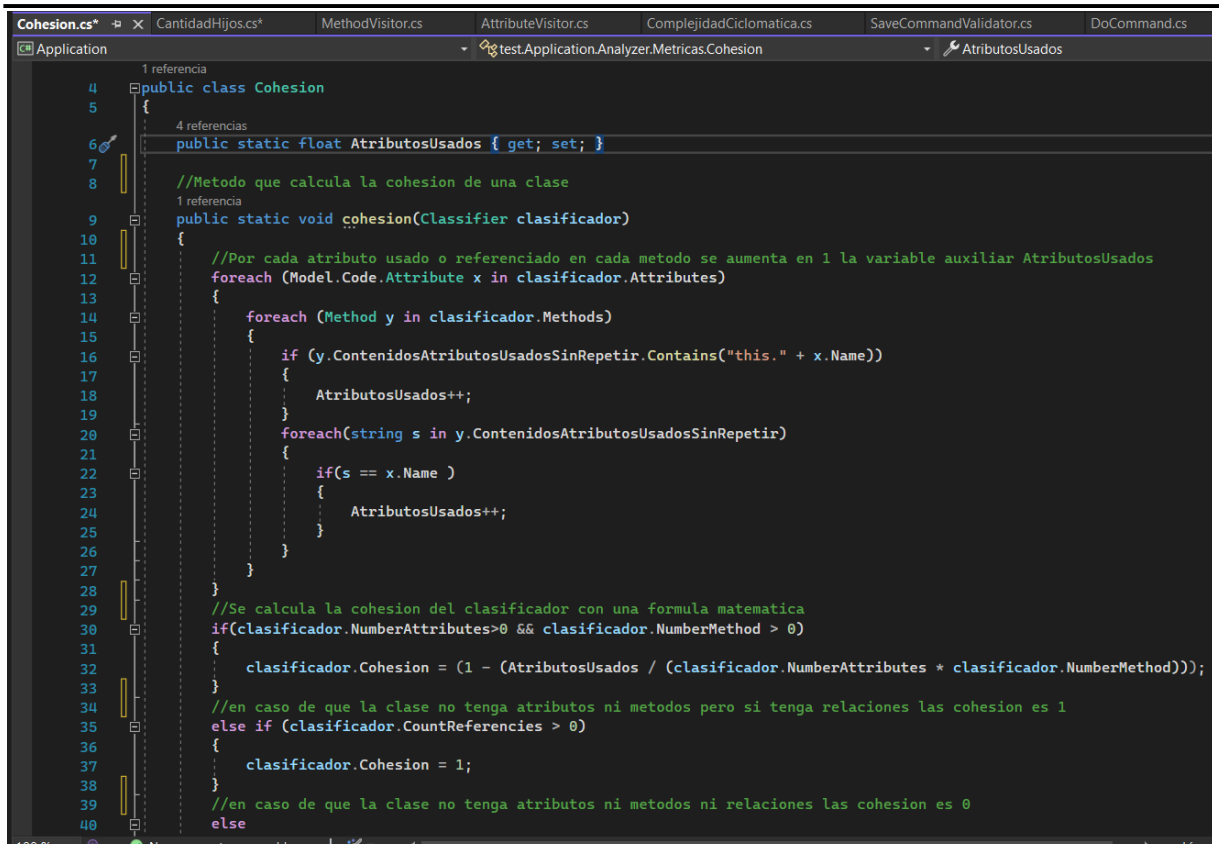


Figura 15.2.d: Código fuente documentado de Cohesion.cs parte uno.

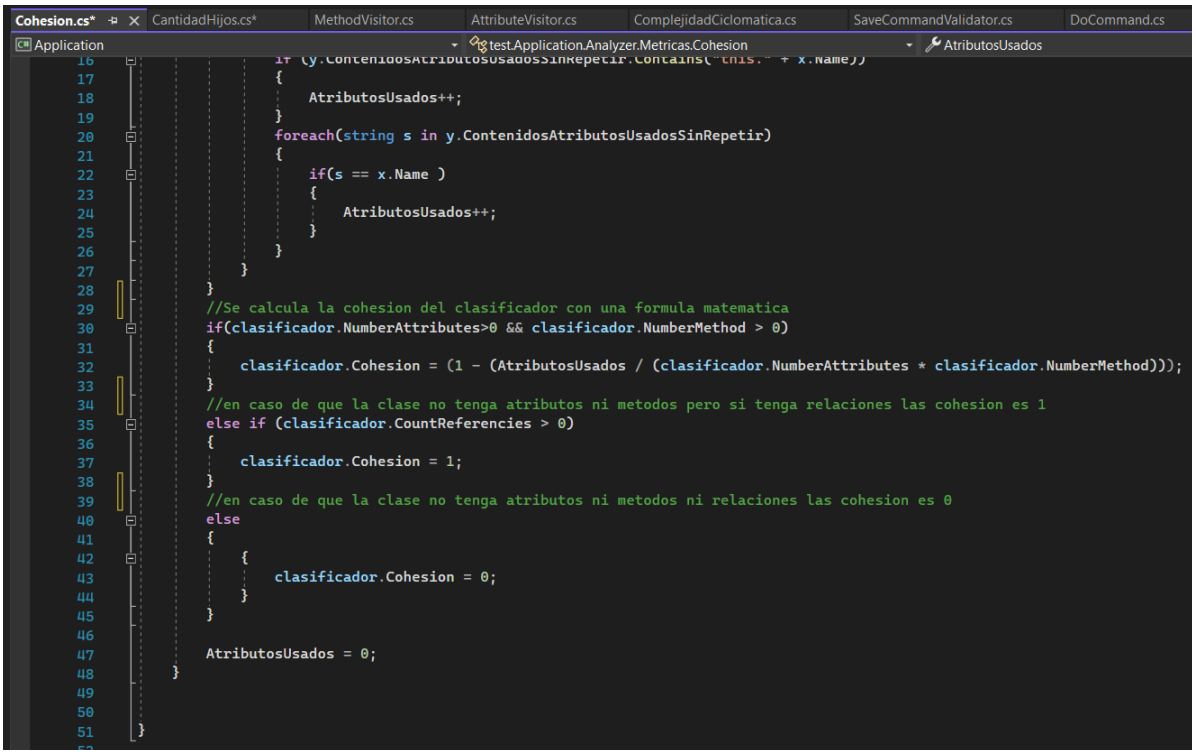


Figura 15.2.e: Código fuente documentado de Cohesion.cs parte dos.

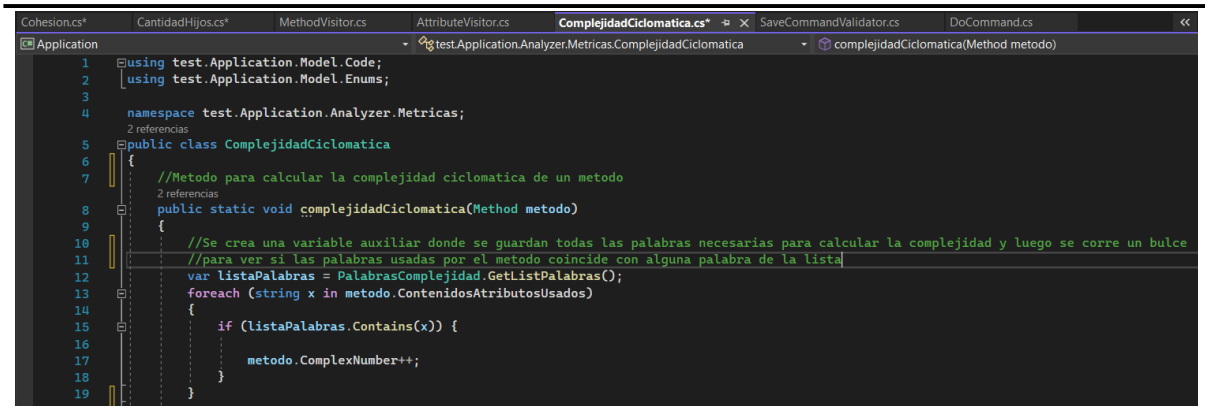


Figura 15.2.f: Código fuente documentado de ComplejidadCiclomatica.cs.

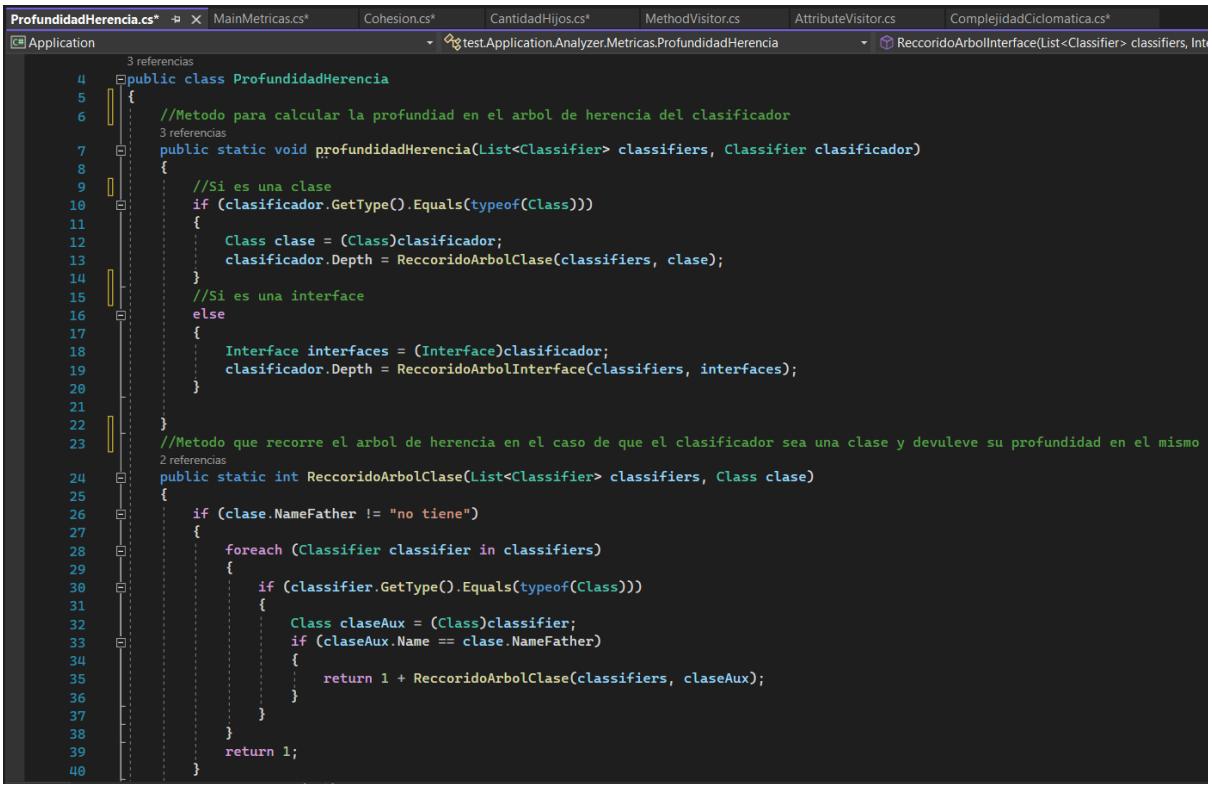


Figura 15.2.g: Código fuente documentado de ProfundidadHerencia.cs.

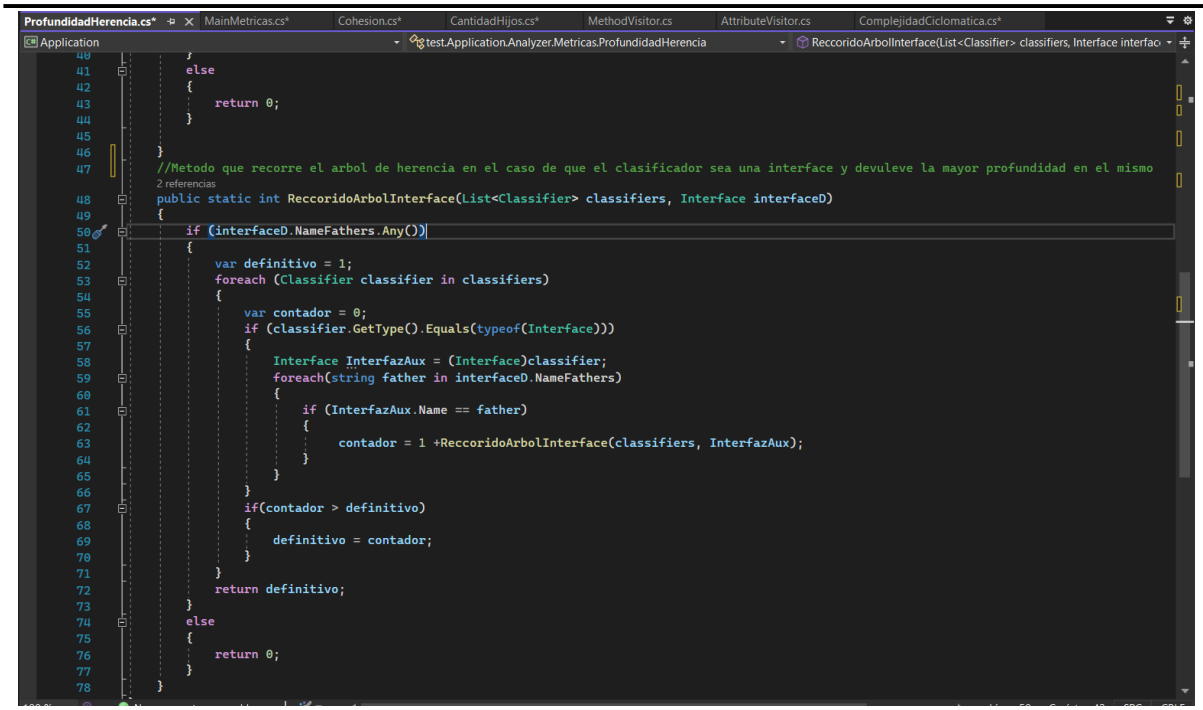


Figura 15.2.h: Código fuente documentado de ProfundidadHerencia.cs.

15.3.Persistencia en la base de datos.

Una vez que se han calculado todas las métricas de cada clase, atributo y método, se las almacena en la base de datos.

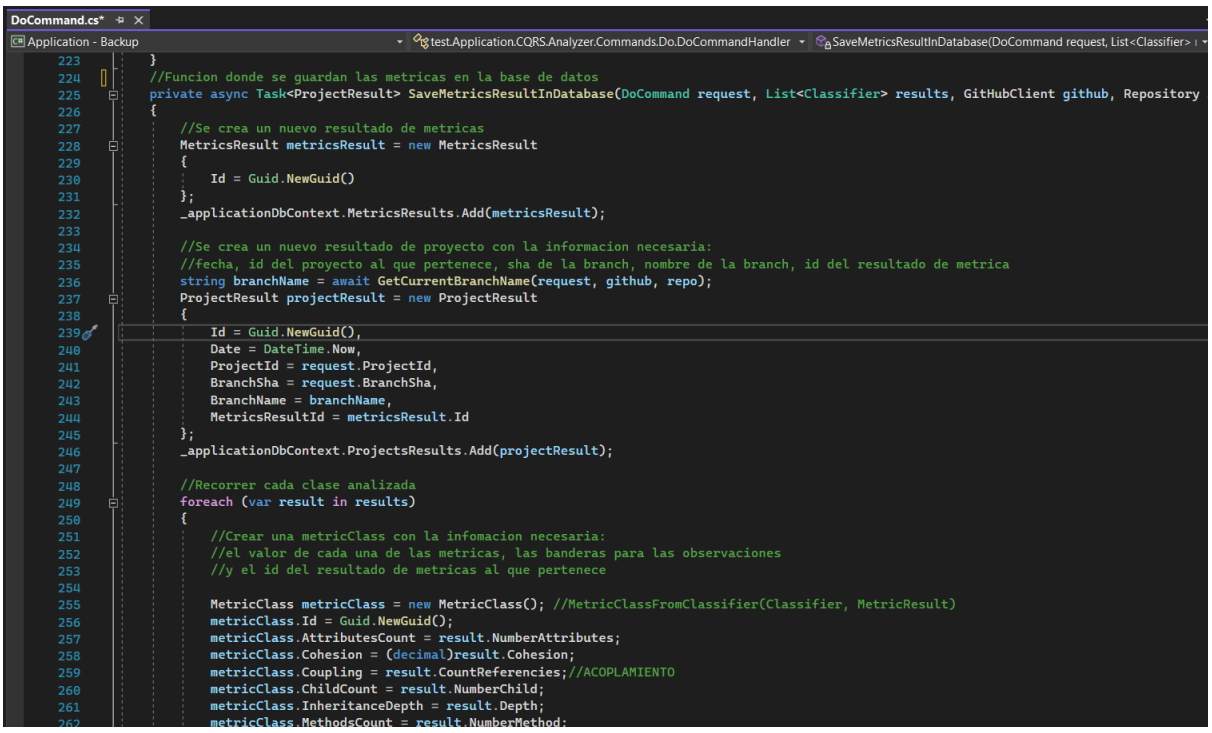


Figura 15.3.a: Código fuente documentado de DoCommand.cs parte uno.

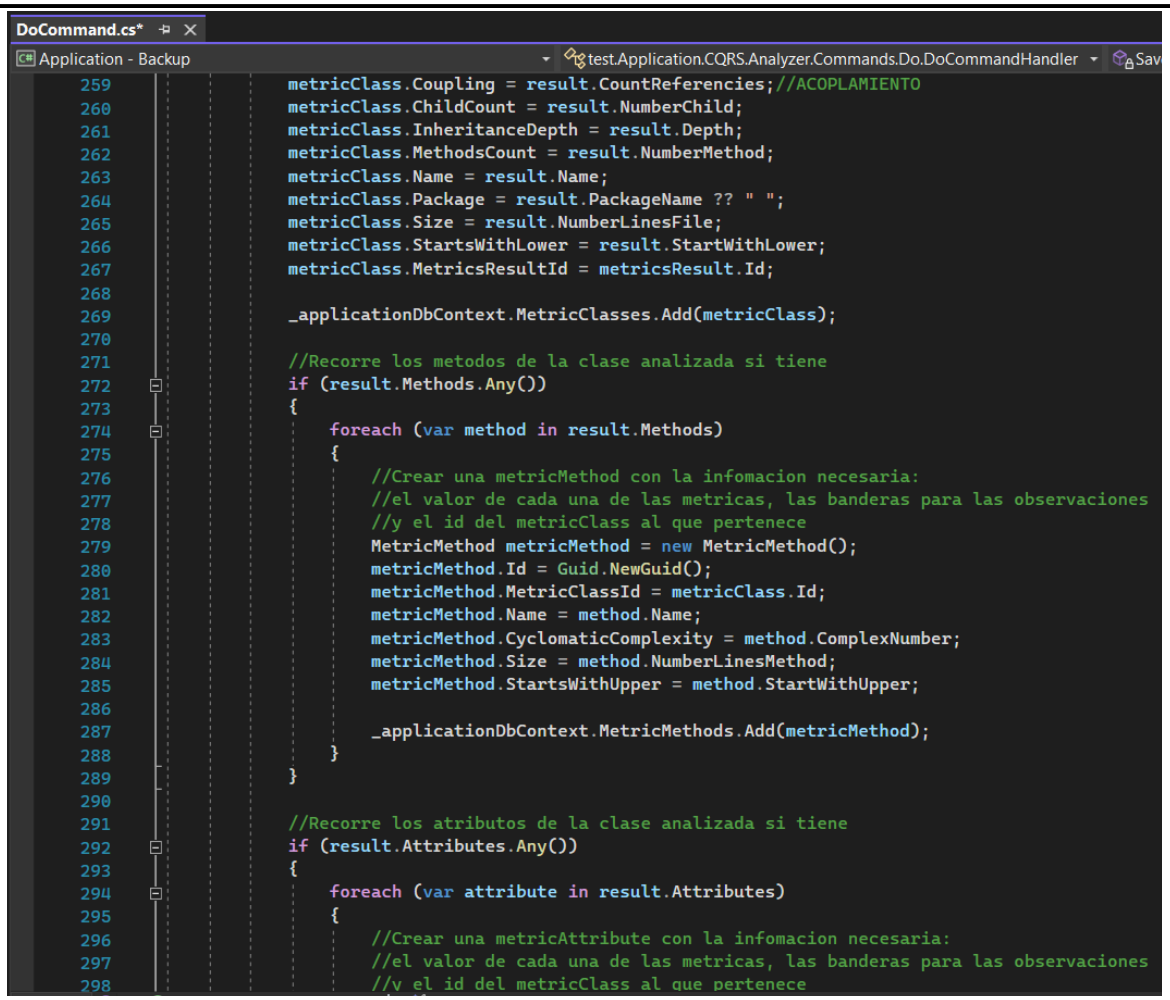


Figura 15.3.b: Código fuente documentado de DoCommand.cs parte dos.

DoCommand.cs* — Application - Backup — test.Application.CQRS.Analyzer.Commands.Do.DoCommandHandler

```
277                     //el valor de cada una de las metricas, las banderas para las observaciones
278                     //y el id del metricClass al que pertenece
279                     MetricMethod metricMethod = new MetricMethod();
280                     metricMethod.Id = Guid.NewGuid();
281                     metricMethod.MetricClassId = metricClass.Id;
282                     metricMethod.Name = method.Name;
283                     metricMethod.CyclomaticComplexity = method.ComplexNumber;
284                     metricMethod.Size = method.NumberLinesMethod;
285                     metricMethod.StartsWithUpper = method.StartWithUpper;
286
287                     _applicationDbContext.MetricMethods.Add(metricMethod);
288                 }
289             }
290
291             //Recorre los atributos de la clase analizada si tiene
292             if (result.Attributes.Any())
293             {
294                 foreach (var attribute in result.Attributes)
295                 {
296                     //Crear una metricAttribute con la infomacion necesaria:
297                     //el valor de cada una de las metricas, las banderas para las observaciones
298                     //y el id del metricClass al que pertenece
299                     MetricAttribute metricAttribute = new MetricAttribute();
300                     metricAttribute.Id = Guid.NewGuid();
301                     metricAttribute.Name = attribute.Name;
302                     metricAttribute.Length = attribute.Length;
303                     metricAttribute.StartsWithUpper = attribute.StartWithUpper;
304                     metricAttribute.ConstantWithLower = attribute.NoUpper;
305                     metricAttribute.HasPackageVisibility = attribute.IsPackage;
306                     metricAttribute.MetricClassId = metricClass.Id;
307
308
309                     _applicationDbContext.MetricAttributes.Add(metricAttribute);
310                 }
311             }
312
313         }
314         return projectResult;
315     }
```

Figura 15.3.c: Código fuente documentado de DoCommand.cs parte tres.

16.Planificación de Capacitación.

El plan de capacitación de Junkode fue ideado para cumplir la siguiente lista de características:

➢ Elementos de capacitación disponibles en todo momento: Para que el usuario pueda consultarlos y aprender a utilizar la herramienta sin la necesidad de atarse a reuniones con horarios fijos.

➢ Segmentos de capacitación independientes: El programa debe poder dividirse en segmentos de capacitación, cada uno con un su propio objetivo específico, y estos deben poder ser consultados en cualquier orden y sin la necesidad de haber consultado otro previamente.

➢ Inicio rápido: La duración de los segmentos debe ser corta, con el fin de permitir un arranque rápido de los usuarios dentro de las funcionalidades de la herramienta.

A continuación puede encontrarse el Plan de Capacitación, pero los elementos a los que se hacer referencia en este pueden ser encontrados en:

➢ Anexo N°8: Guía de Inicio Rápido de Junkode.

➢ Anexo N°9: Manual de Usuario de Junkode.

➢ Anexo N°10: Manual de Administradores de Junkode.
➢ Videos de la Guía de Inicio Rápido:
○ 01: Carga de Proyecto Local (https://youtu.be/44t__8XQp5o).
○ 02: Carga de Proyecto desde GitHub (https://youtu.be/bXdzduTu3Z4).
○ 03: Resultados del Análisis (https://youtu.be/xPfwII2wPu8).

16.1.Programa de Capacitación.

Objetivos generales:
➢ Presentar instrucciones para facilitar el uso de la herramienta.
➢ Entrenar a los usuarios en los usos de la herramienta para obtener documentación técnica especializada para los proyectos particulares.
➢ Propiciar una mejor comprensión de la documentación resultante de los análisis de proyectos.

Segmento de capacitación:
Nombre: Sesión.
Objetivos Temáticos:
➢ Explicar los alcances de las funcionalidades del sistema para usuarios registrados y no registrados.
➢ Mostrar la relación entre sesión de Github y la sesión de Junkcode.
Material: Un apartado en la guía de inicio rápido e información detallada dentro del manual de usuario.
Modalidad: Virtual y de acceso público.
Destinatarios: Usuario registrado por Github y no registrado.
Duración estimada: 10 minutos.

Segmento de capacitación:
Nombre: Carga de código.
Objetivos Temáticos:
➢ Enseñar las opciones de carga de código disponibles para cada tipo de usuario.
➢ Ofrecer un ejemplo para cada tipo de carga de proyectos.
Material: Un apartado en la guía de inicio rápido, dos vídeos tutoriales e información detallada dentro del manual de usuario.
Modalidad: Virtual y de acceso público.
Destinatarios: Usuario registrado por Github y no registrado
Duración estimada: 25 minutos.

Segmento de capacitación:
Nombre: Análisis y resultados.
Objetivos Temáticos:
➢ Presentar los distintos elementos de documentación generados por Junkode.
➢ Describir el uso y las condiciones de generación de los resultados de un análisis.
Material: Un apartado en la guía de inicio rápido, un video tutorial e información detallada dentro del manual de usuario.
Modalidad: Virtual y de acceso público.

Destinatarios: Usuario registrado por Github y no registrado.
Duración estimada: 25 minutos.

Segmento de capacitación:
Nombre: Historial de resultados.
Objetivos Temáticos:
➢ Mostrar cómo se almacenan los resultados de los análisis en función de su fecha de creación.
➢ Enseñar a hacer consultas sobre resultados anteriores de un proyecto.
Material: Un apartado en la guía de inicio rápido e información detallada dentro del manual de usuario.
Modalidad: Virtual y de acceso público.
Destinatarios: Usuario registrado por Github.
Duración estimada: 10 minutos.

Segmento de capacitación:
Nombre: Administradores.
Objetivos Temáticos:
➢ Interpretar las métricas que figuran en la interfaz de administradores.
➢ Mostrar mecánicas internas del sistema (Servicios internos y externos).
➢ Enseñar las funcionalidades exclusivas de las que se dispone como administrador.
Material: Manual de Administradores de Junkode.
Modalidad: Virtual y privada.
Destinatarios: Usuario registrado por Github.
Duración estimada: 30 minutos.

17.Planificación, Ejecución y Documentación de Pruebas.

A continuación se listan las pruebas realizadas durante el desarrollo de Junkode separadas en cinco categorías:
➢ Pruebas de validación de ingreso de datos.
➢ Pruebas de lógica de los módulos principales.
➢ Pruebas de integración entre módulos.
➢ Pruebas de carga.
➢ Pruebas de seguridad por niveles de usuarios.

17.1.Pruebas de validación de ingreso de datos.

Prueba de sistema de validación de ingreso de datos	
Número	01
Objetivo	Comprobar que la rama seleccionada por el usuario para realizar el análisis contenga al menos un archivo con extensión “.java”.

<table>
<tr><td>Requerimientos</td><td colspan="3">-Usuario logueado.
-Proyecto creado en el sistema por el usuario.</td></tr>
<tr><td>Resultado esperado</td><td colspan="3">Warning que indica que no hay archivos java en la rama seleccionada.</td></tr>
<tr><td>Lote de prueba con atributos y valores a utilizar</td><td colspan="3">Usuario logueado con:
-username:renzo_fer00@hotmail.com.ar
-password:contraseña1234 .

Proyecto creado con nombre “Sin archivos java” asociado al repositorio “RepositorioSinJava” el cual no contiene archivos con extension “.java”.

Rama “Main” elegida.</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Warning que indica que no hay archivos java en la rama seleccionada.</td></tr>
<tr><td rowspan="6">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario</td><td>Sistema</td></tr>
<tr><td>01</td><td>Hacer click en el botón “View” del proyecto “Sin archivos java”.</td><td></td></tr>
<tr><td>02</td><td>Hacer click en “New Result”.</td><td></td></tr>
<tr><td>03</td><td>Elegir la rama “Main”.</td><td></td></tr>
<tr><td>04</td><td>Hacer click en “DoAnalyze”.</td><td></td></tr>
<tr><td>05</td><td></td><td>Warning que indica que no hay archivos java en la rama seleccionada.</td></tr>
</table>

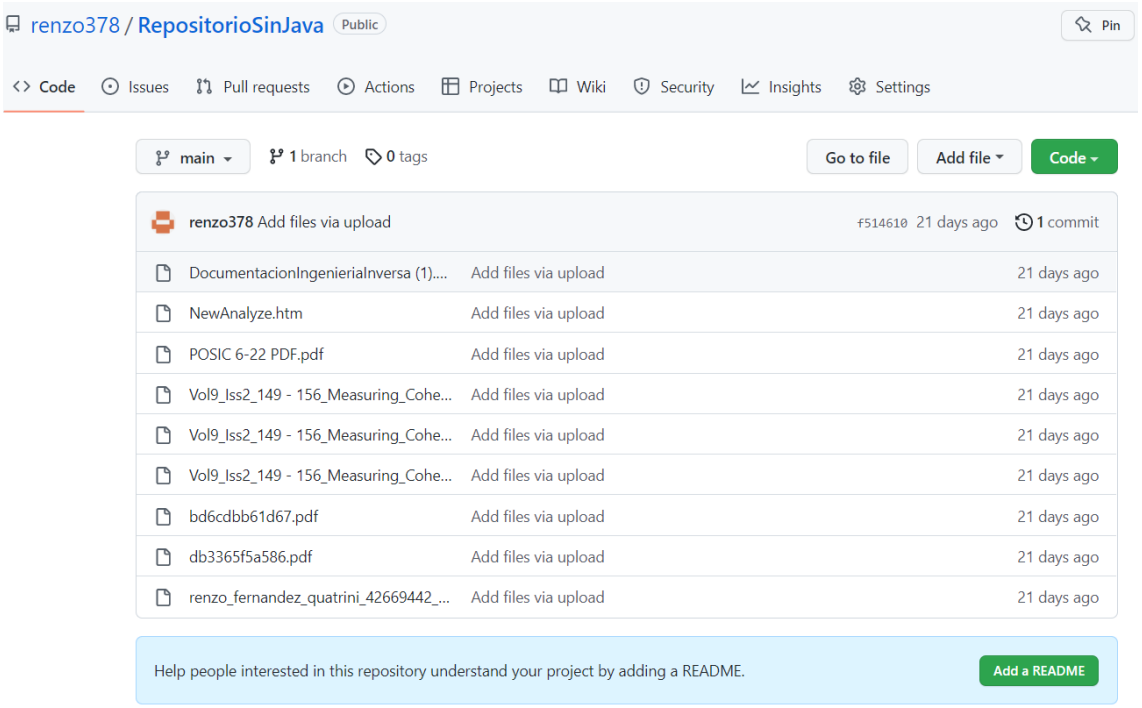


Figura 17.1.a: Contenido de la rama “main” del repositorio de prueba “RepositorioSinJava”.

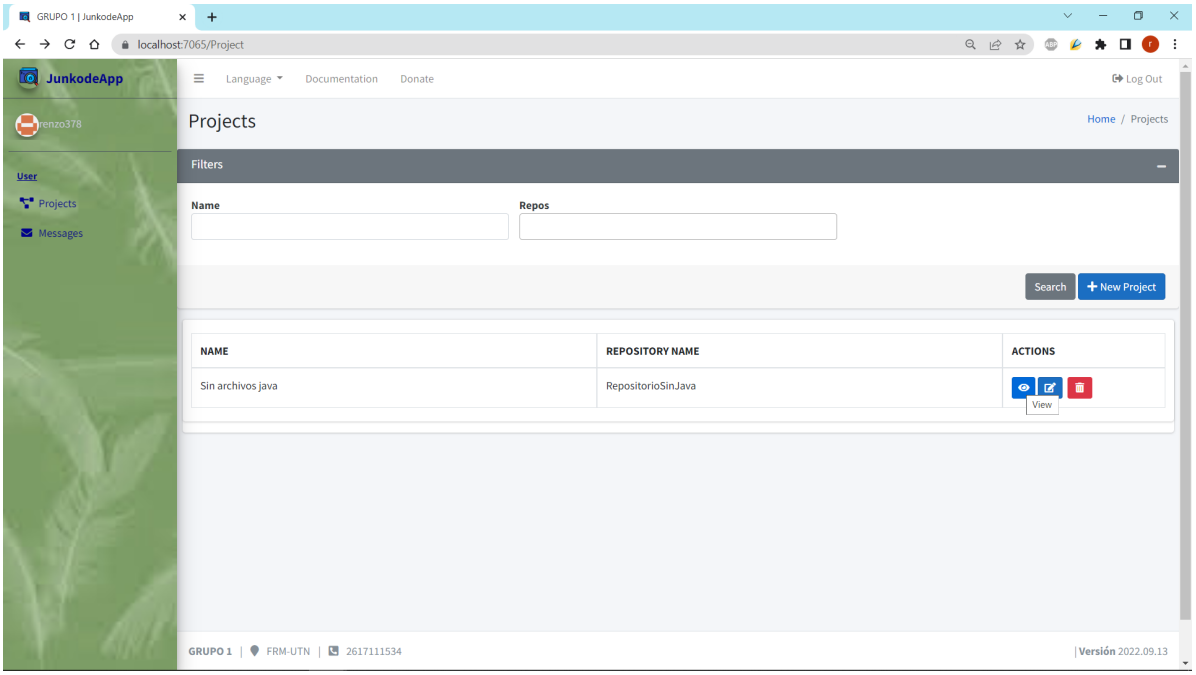


Figura 17.1.b: Pantalla correspondiente al Paso 01 de la Prueba número 01.

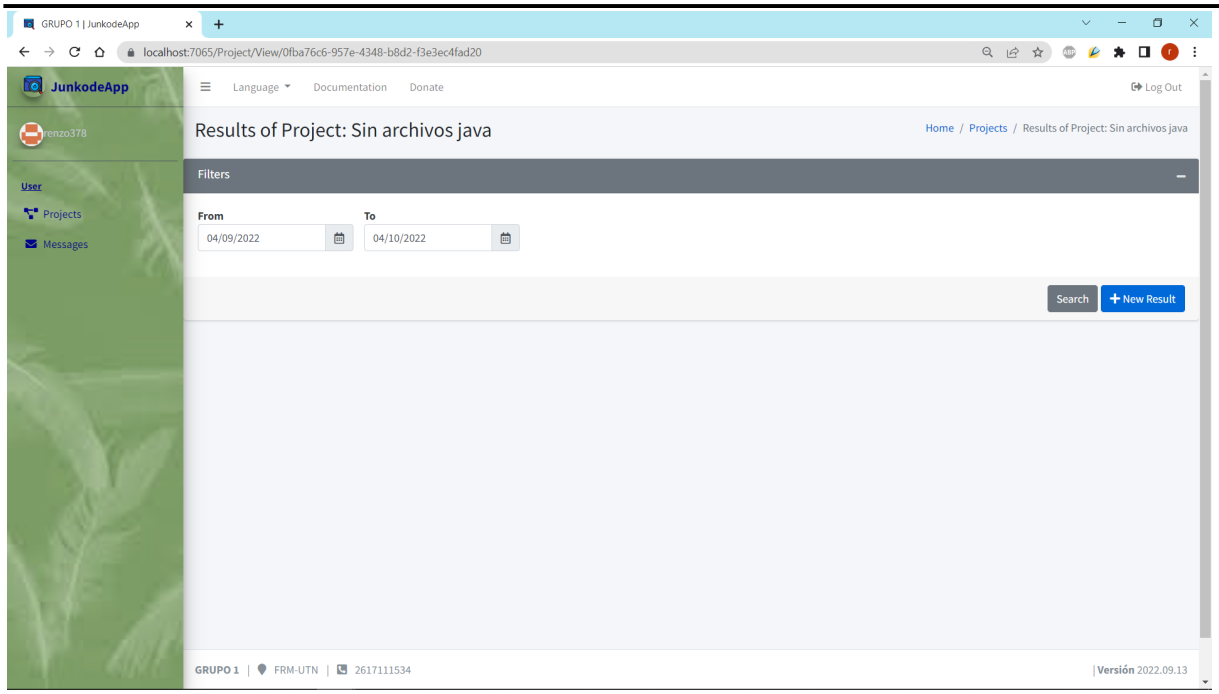


Figura 17.1.c: Pantalla correspondiente al Paso 02 de la Prueba número 01.

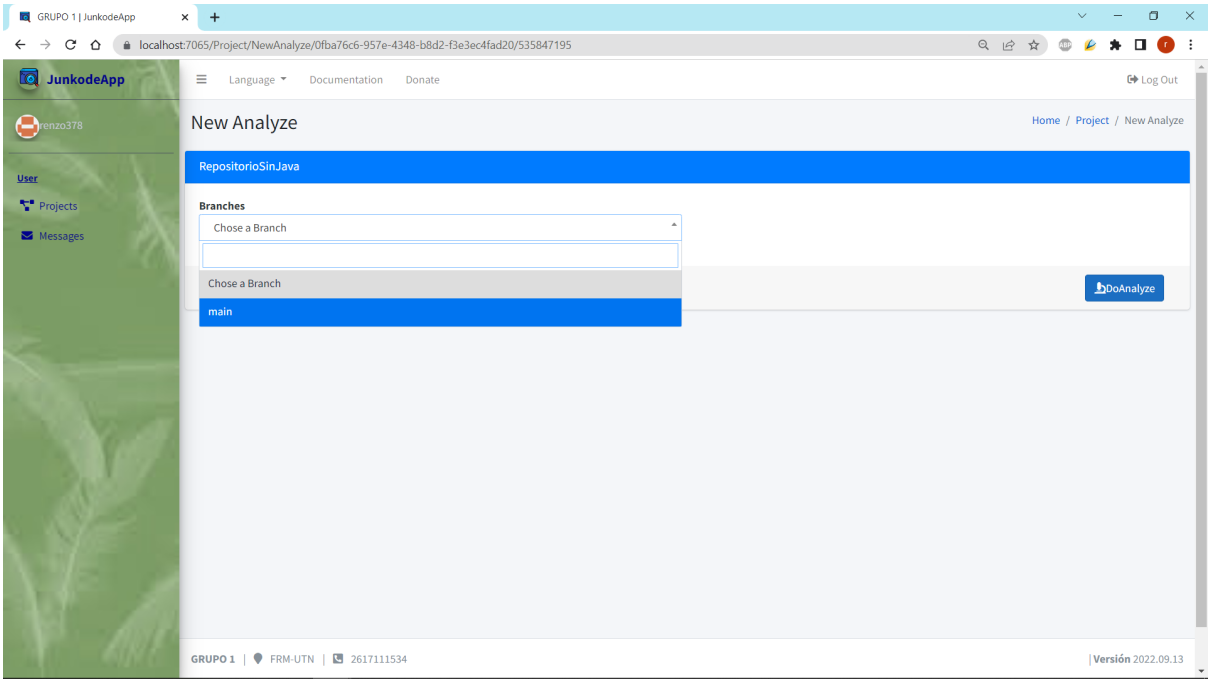


Figura 17.1.d: Pantalla correspondiente al Paso 03 de la Prueba número 01.

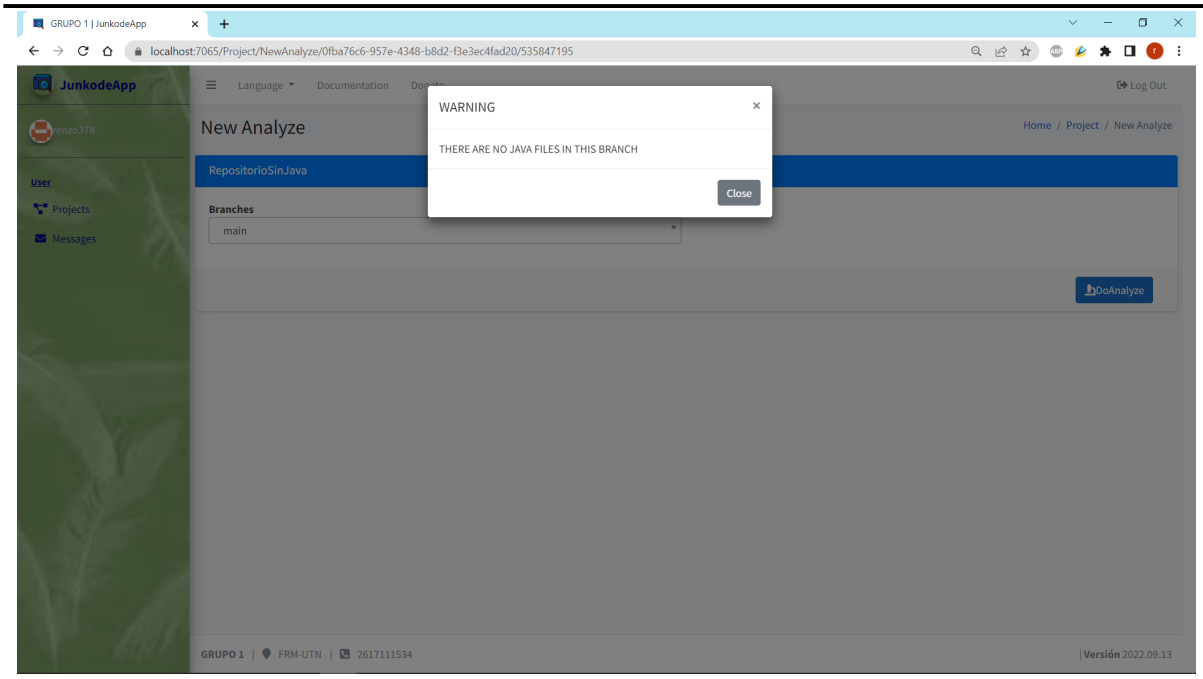


Figura 17.1.e: Pantalla correspondiente al éxito del Paso 05 de la Prueba número 01.

Prueba de sistema de validación de ingreso de datos			
Número	02		
Objetivo	Comprobar que el usuario no cree un proyecto sin nombre.		
Requerimientos	-Usuario logueado y con repositorios en su cuenta de github.		
Resultado esperado	Warning que indica que el nombre del proyecto no puede ser vacío.		
Lote de prueba con atributos y valores a utilizar	Usuario logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 . Proyecto sin nombre asociado al repositorio “Escuela”.		
Resultado obtenido	Creación del proyecto sin nombre.		
Pasos lógicos de	N°	Usuario	Sistema
	01	Hacer click en “New Project”.	
	02	Elegir el repositorio “Escuela”.	
	03	Hacer click en “Save”.	

la prueba	04		Mensaje que indica que el proyecto fue creado correctamente.
Acciones correctivas	Corregir función “SaveCommandValidator” la cual verifica que un proyecto no puede tener nombres vacíos.		

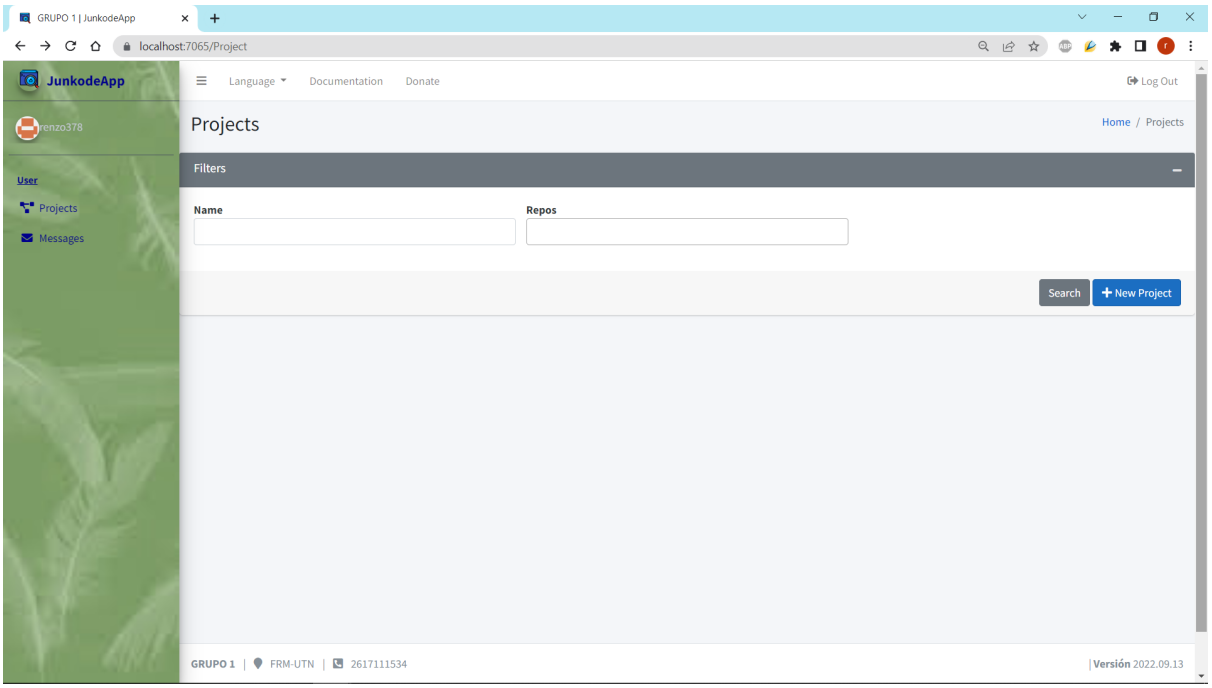


Figura 17.1.f: Pantalla correspondiente al del Paso 01 de la Prueba número 02.

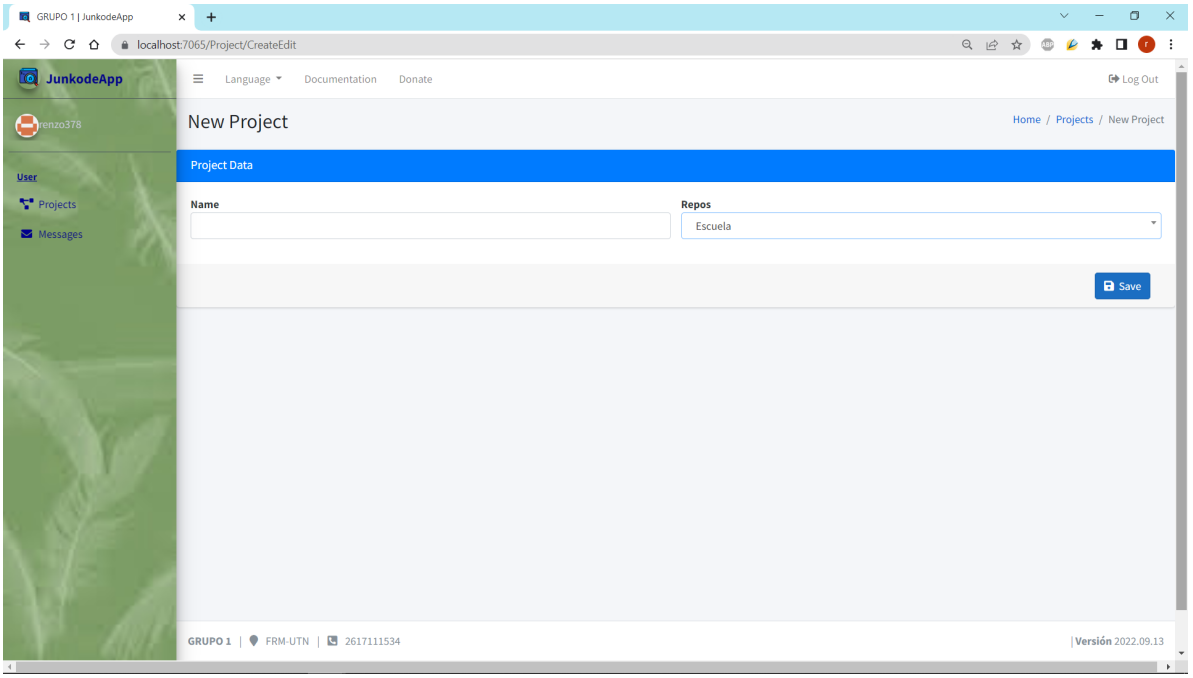


Figura 17.1.g: Pantalla correspondiente al Paso 02 de la Prueba número 02.

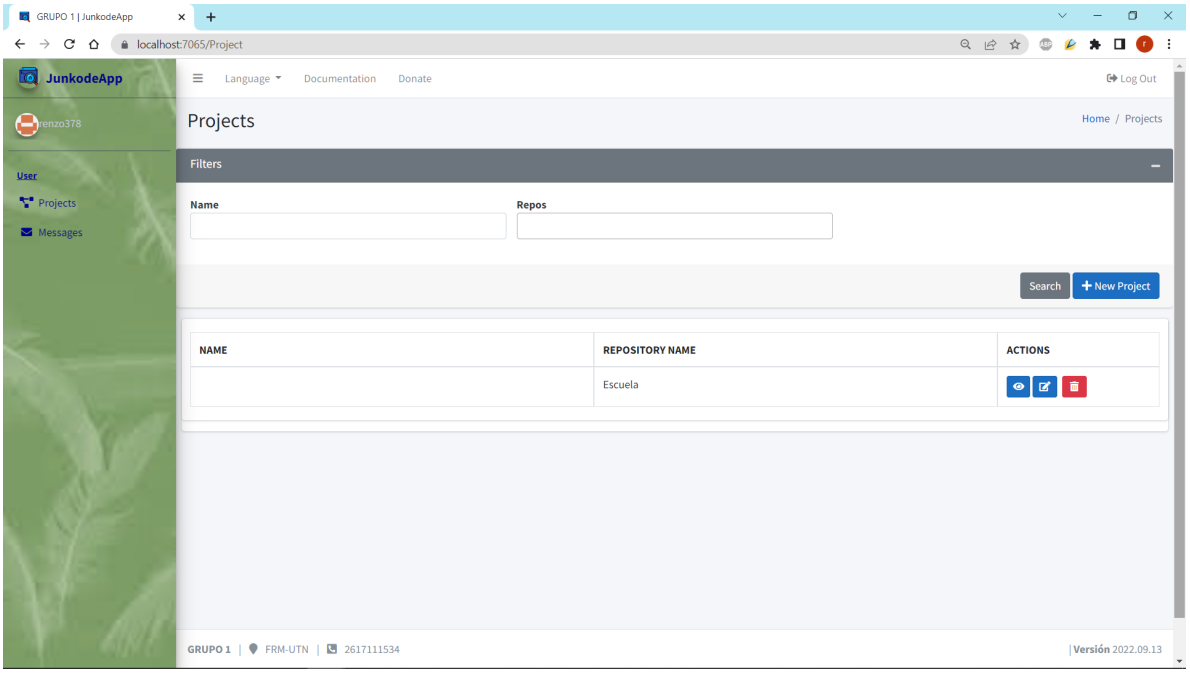


Figura 17.1.h: Pantalla correspondiente al Paso 04 fallido de la Prueba número 02.

Prueba de sistema de validación de ingreso de datos			
Número	03		
Objetivo	Comprobar que el usuario no cree un proyecto con sentencias SQL en el nombre.		
Requerimientos	-Usuario logueado en el sistema y con repositorios en su cuenta de github.		
Resultado esperado	Warning que indica que el nombre del proyecto no puede contener sentencias SQL en su nombre.		
Lote de prueba con atributos y valores a utilizar	Usuario logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 . Proyecto con nombre “Select” asociado al repositorio “Escuela”.		
Resultado obtenido	Proyecto creado con sentencias SQL en su nombre.		
	N°	Usuario	Sistema
	01	Hacer click en “New Project”.	

Pasos lógicos de la prueba	02	Elegir repositorio “Escuela ”y escribir como nombre de proyecto “Select”.	
	03	Hacer click en “Save”.	
	04		Mensaje que indica que el proyecto fue creado correctamente.
Acciones correctivas	Añadir a la función “SaveCommandValidator” que verifique que un proyecto no pueda incluir sentencias SQL en su nombre.		

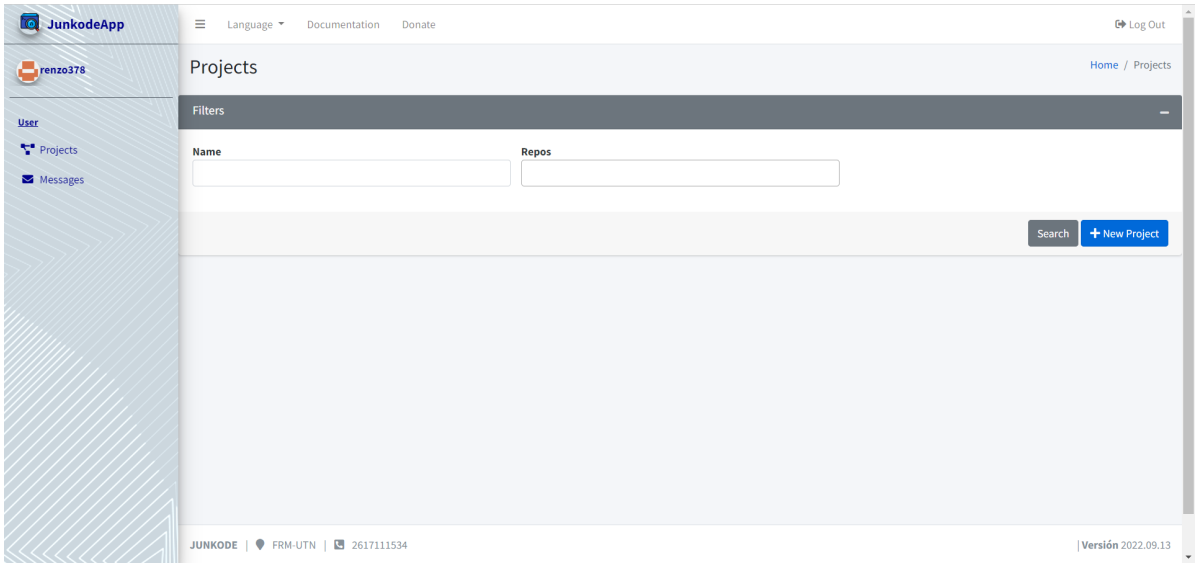


Figura 17.1.i: Pantalla correspondiente al Paso 01 de la Prueba número 03.

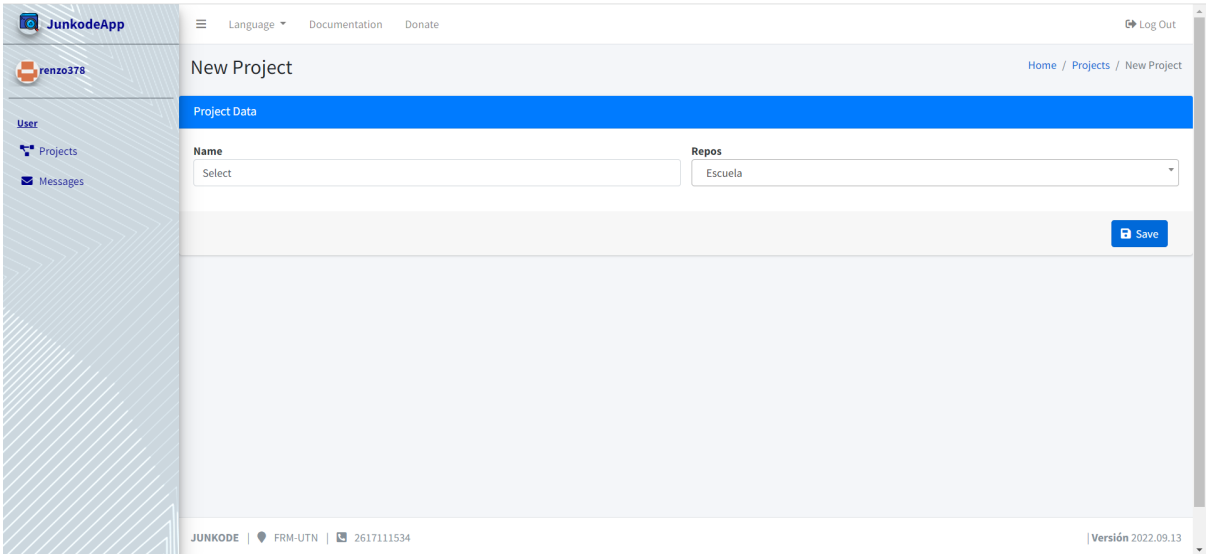


Figura 17.1.j: Pantalla correspondiente al Paso 02 de la Prueba número 03.

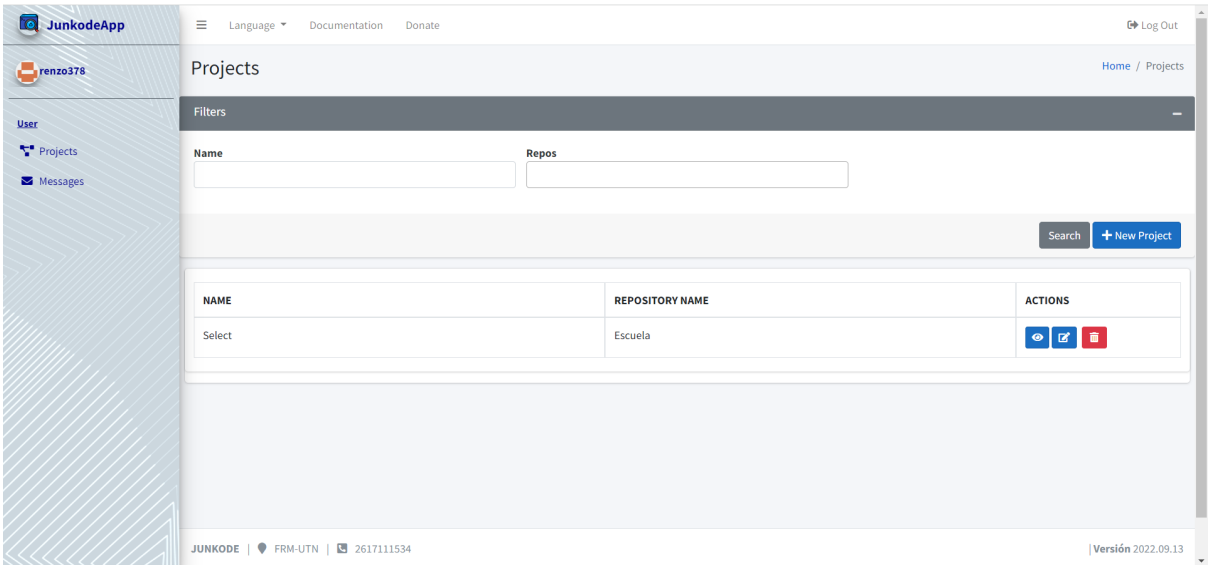


Figura 17.1.k: Pantalla correspondiente al Paso 04 fallido de la Prueba número 03.

17.2.Pruebas de lógica de los módulos principales.

Prueba de sistema de lógica de los módulos principales	
Número	04
Objetivo	Comprobar que la métrica “Complejidad ciclomática” es calculada correctamente.
Requerimientos	-Usuario logueado. -Proyecto creado. -Rama a analizar elegida.
Resultado esperado	Complejidad ciclomática de método “metodoPrueba” igual a 5.
Lote de prueba con atributos y valores a utilizar	Usuario logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 . Proyecto con nombre “PruebaComplejidad” y repositorio “Prueba” asociado. Rama “Main” elegida. Archivo “ComplejidadCiclomatica.java” en la rama Main el cual contiene el método “metodoPrueba”.

<table>
<tr><td>Resultado obtenido</td><td colspan="3">Complejidad ciclomática de método “metodoPrueba” igual a 1.</td></tr>
<tr><td rowspan="5">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario</td><td>Sistema</td></tr>
<tr><td>01</td><td>Una vez seleccionada la rama “Main” hacer click en “DoAnalyze”.</td><td></td></tr>
<tr><td>02</td><td>Seleccionar la clase “ComplejidadCiclomatica”.</td><td></td></tr>
<tr><td>03</td><td>Seleccionar el método “metodoPrueba”.</td><td></td></tr>
<tr><td>04</td><td></td><td>Se muestran los datos analizados del método seleccionado.</td></tr>
<tr><td>Acciones correctivas</td><td colspan="3">Corregir donde es calculada la complejidad ciclomática ya que solo se está teniendo en cuenta el valor constante que es 1 y no se está calculando la complejidad del método.</td></tr>
</table>

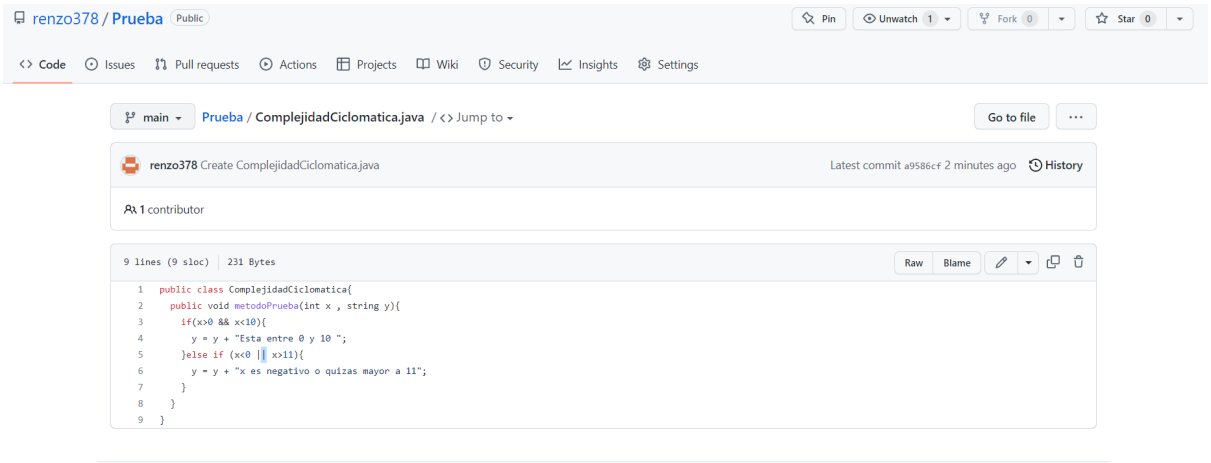


Figura 17.2.a: Contenido del archivo “ComplejidadCiclomatica.java” la rama “main” del repositorio de prueba “Prueba”.

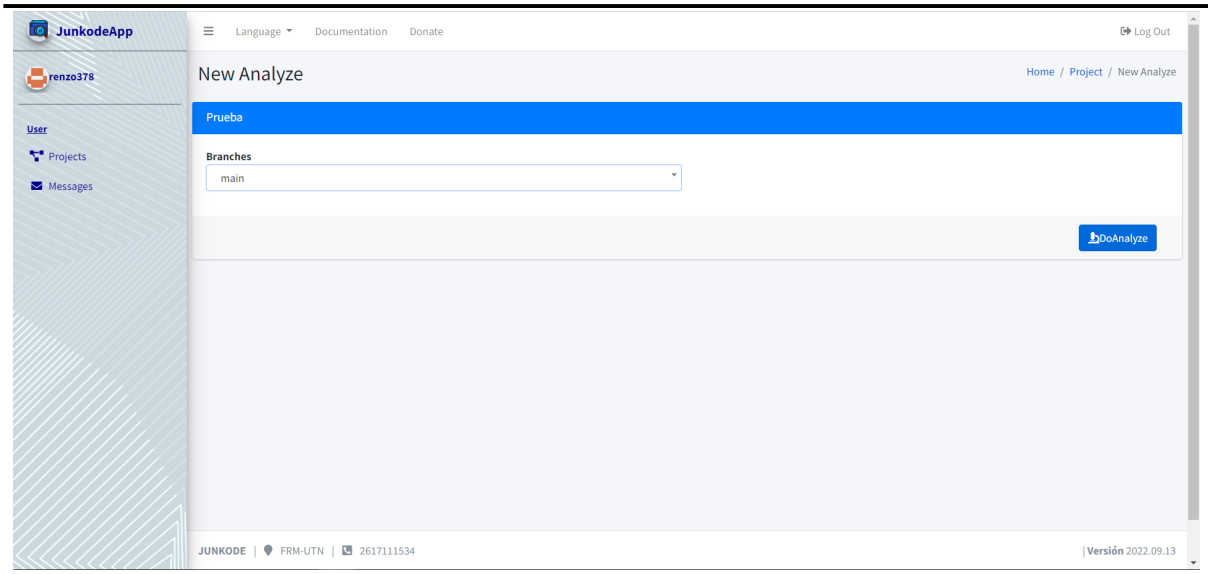


Figura 17.2.b: Pantalla correspondiente al Paso 01 de la Prueba número 04.

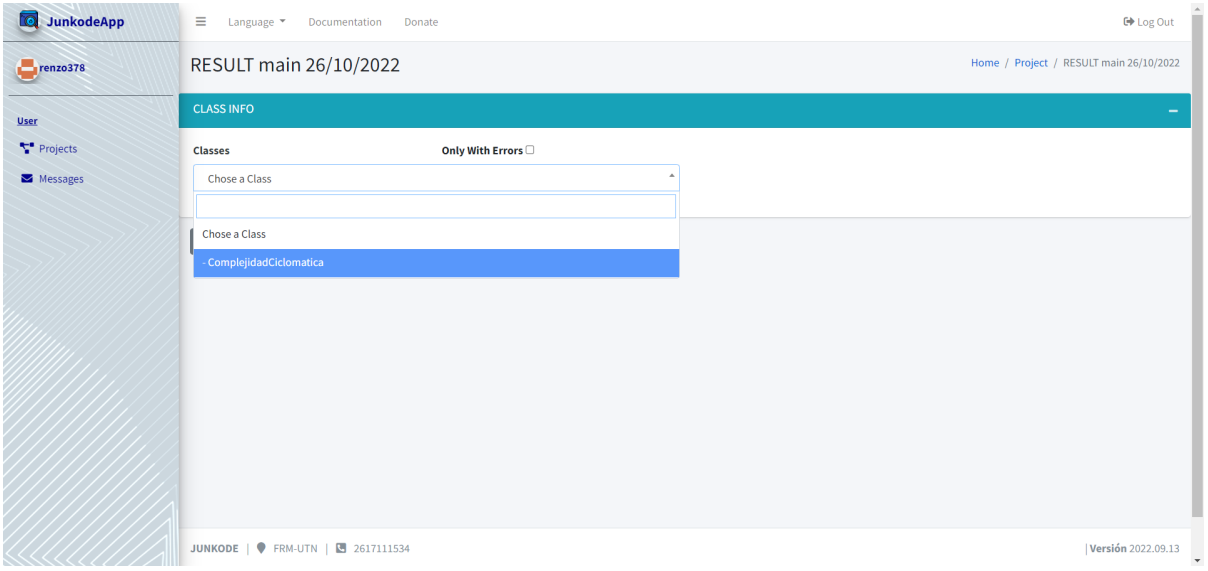


Figura 17.2.c: Pantalla correspondiente al Paso 02 de la Prueba número 04.

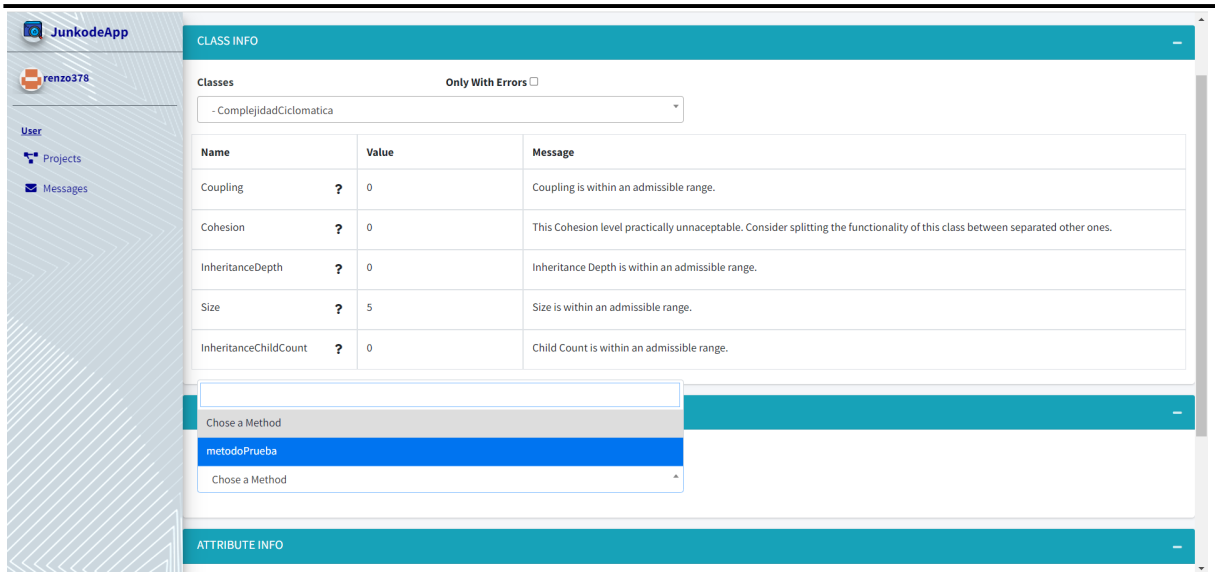


Figura 17.2.d: Pantalla correspondiente al Paso 03 de la Prueba número 04.

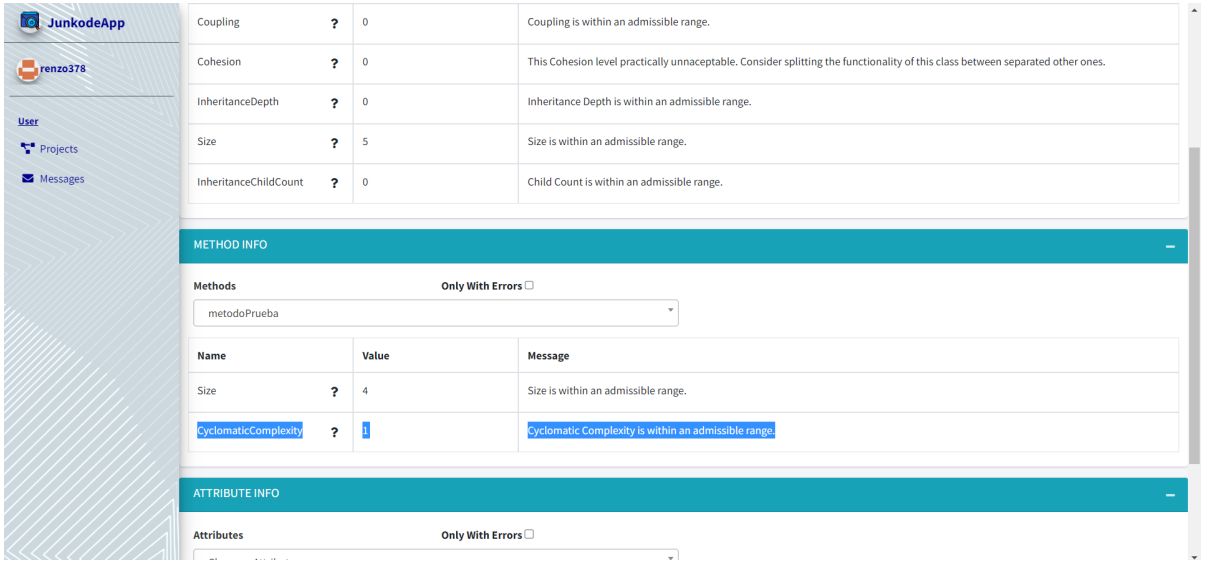


Figura 17.2.e: Pantalla correspondiente al Paso 04 fallido de la Prueba número 04.

Prueba de sistema de lógica de los módulos principales	
Número	05
Objetivo	Comprobar si la métrica “Cantidad Hijos” es calculada correctamente para la clase “Padre”.
Requerimientos	-Usuario logueado. -Proyecto creado. -Rama a analizar elegida.
Resultado	Cantidad de hijos de la clase “Padre” igual a 3.

<table>
<tr><td>esperado</td><td colspan="3"></td></tr>
<tr><td>Lote de prueba con atributos y valores a utilizar</td><td colspan="3">Usuario logueado con:
-username:renzo_fer00@hotmail.com.ar .
-password:contraseña1234 .

Proyecto con nombre “CantidadHijos” y repositorio “Prueba” asociado.

Rama “Main” elegida.

Repositorio “Prueba” el cual contiene 4 archivos con extensión “.java” .Un archivo es una clase de la cual heredan los otros 3 archivos restantes que también son clases.</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Cantidad de hijos de la clase “Padre” igual a 1.</td></tr>
<tr><td rowspan="4">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario</td><td>Sistema</td></tr>
<tr><td>01</td><td>Una vez seleccionada la rama “Main ”hacer click en “DoAnalyze”.</td><td></td></tr>
<tr><td>02</td><td>Seleccionar la clase “Padre”.</td><td></td></tr>
<tr><td>03</td><td></td><td>Se muestran los datos analizados de la clase seleccionada.</td></tr>
<tr><td>Acciones correctivas</td><td colspan="3">Corregir donde es calculada la métrica “Cantidad Hijos” ya que solo cuenta el primer hijo que tiene una clase y no la totalidad de clases que heredan de la misma.</td></tr>
</table>

```
1   public class Padre{
2
3
4   }
```

```
1   public class Hijo1 extends Padre{
2   }
```

```
1   public class Hijo2 extends Padre{
2   }
```

```
1   public class Hijo3 extends Padre{
2   }
```

Figuras 17.2.f: Contenido de las clases de prueba de la rama “main” del repositorio de prueba “Prueba”.

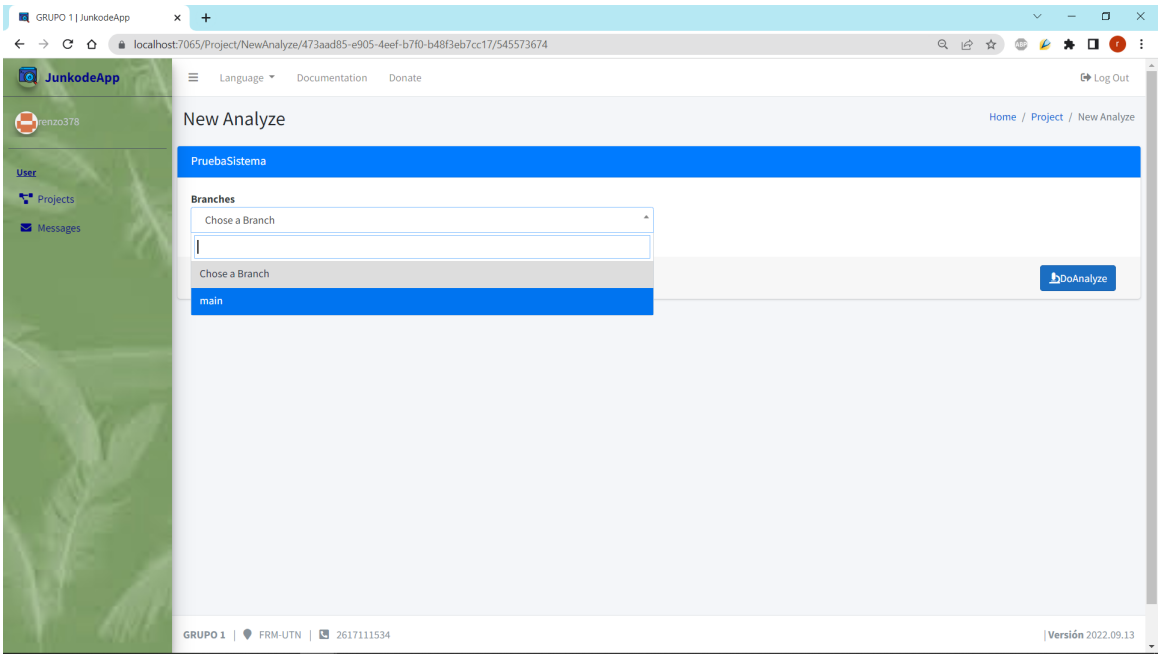


Figura 17.2.g: Pantalla correspondiente al Paso 01 de la Prueba número 05.

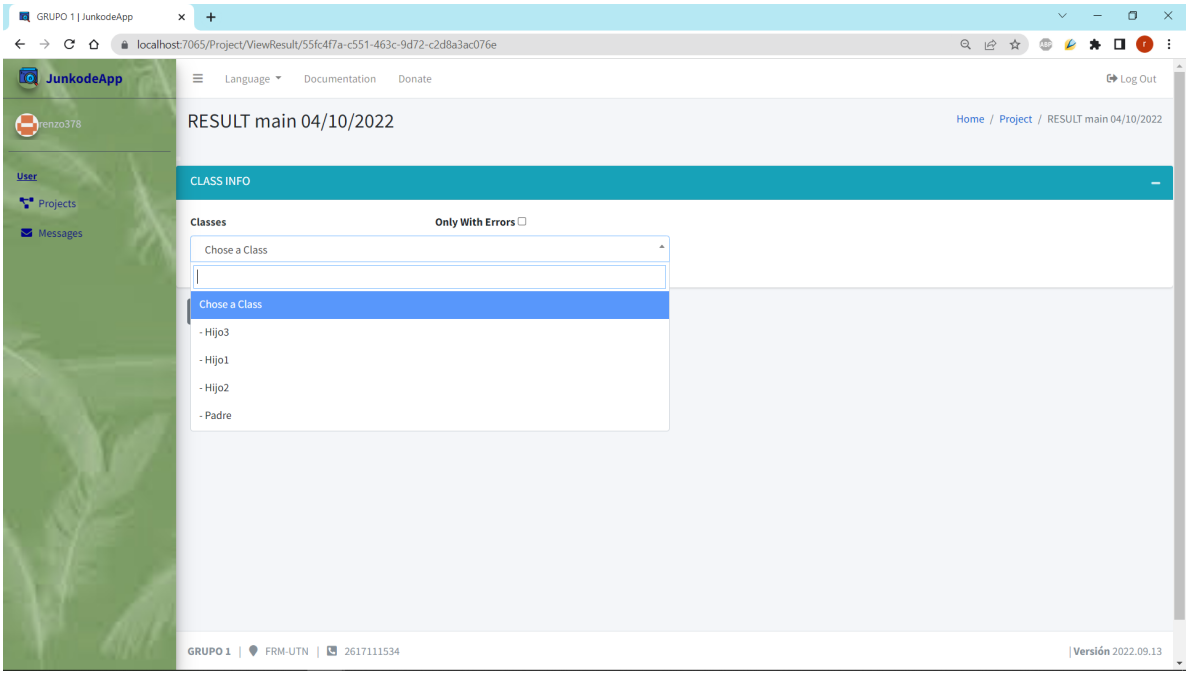


Figura 17.2.h: Pantalla correspondiente al Paso 02 de la Prueba número 05.

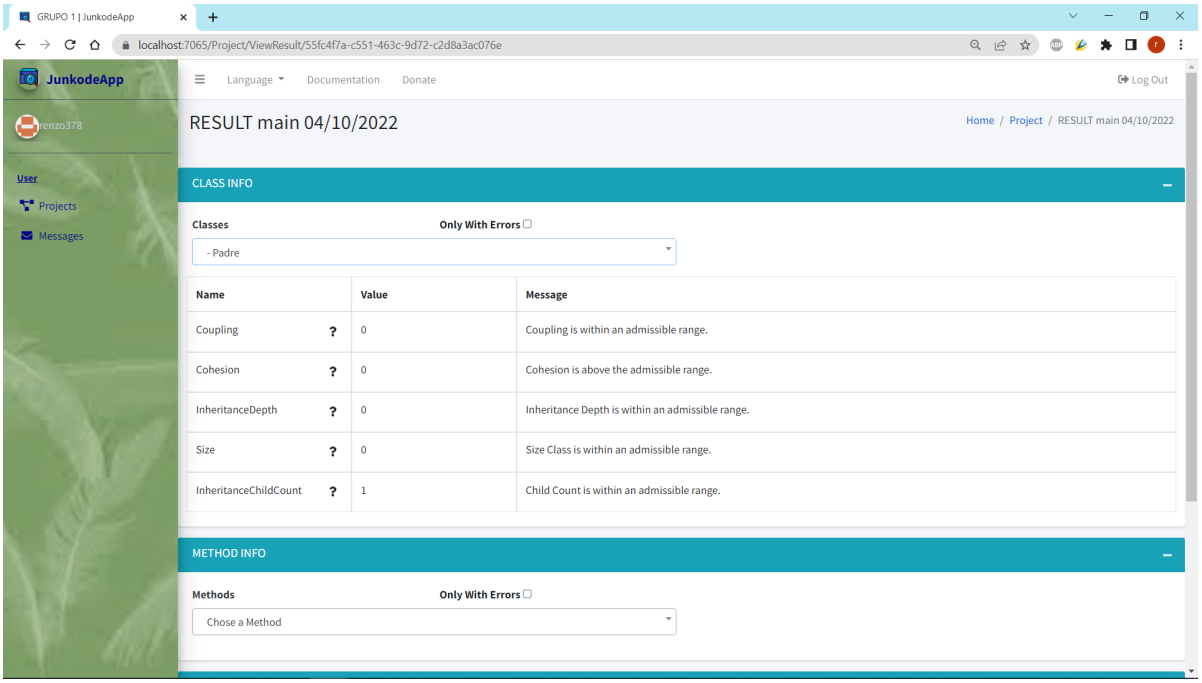


Figura 17.2.i: Pantalla correspondiente al Paso 03 fallido de la Prueba número 05.

Prueba de sistema de lógica de los módulos principales	
Número	06
Objetivo	Comprobar que una entidad es graficada correctamente teniendo su nombre, sus métodos y sus atributos.
Requerimientos	-Usuario logueado. -Proyecto creado. -Rama a analizar elegida.
Resultado esperado	Clase “Persona” graficada con los atributos,”nombre” y “apellido”, y el método “nombreCompleto” graficados también de forma completa(visibilidad,tipo,retorno,nombre) dentro del package “pruebaSistema”.
Lote de prueba con atributos y valores a utilizar	Usuario logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 . Proyecto con nombre “Graficador” y repositorio “Prueba” asociado. Rama “Main” elegida.

<table>
<tr><td></td><td colspan="3">Repositorio “Prueba” el cual contiene 1 archivos con extensión “.java”.El mismo es una clase llamada “Persona” la cual tiene una anotación “@Entity”, dos atributos, “nombre” y “apellido” ambos con visibilidad private y tipo de dato String y un método llamado “nombreCompleto” el cual tiene visibilidad public,tipo de retorno String y forma parte del package “pruebaSistema”.</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Clase “Persona” graficada con los atributos,”nombre” y “apellido”, y el método “nombreCompleto” pero sin su visibilidad y tipo de dato o retorno.</td></tr>
<tr><td rowspan="3">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario</td><td>Sistema</td></tr>
<tr><td>01</td><td>Seleccionar rama “Main” y hacer click en “DoAnalyze”.</td><td></td></tr>
<tr><td>02</td><td></td><td>Se muestran los datos analizados de la clase seleccionada.</td></tr>
<tr><td>Acciones correctivas</td><td colspan="3">Corregir el body que se pasa a la petición POST a la API ”graficadora” ya que no está graficando la visibilidad de los atributos y métodos.</td></tr>
</table>

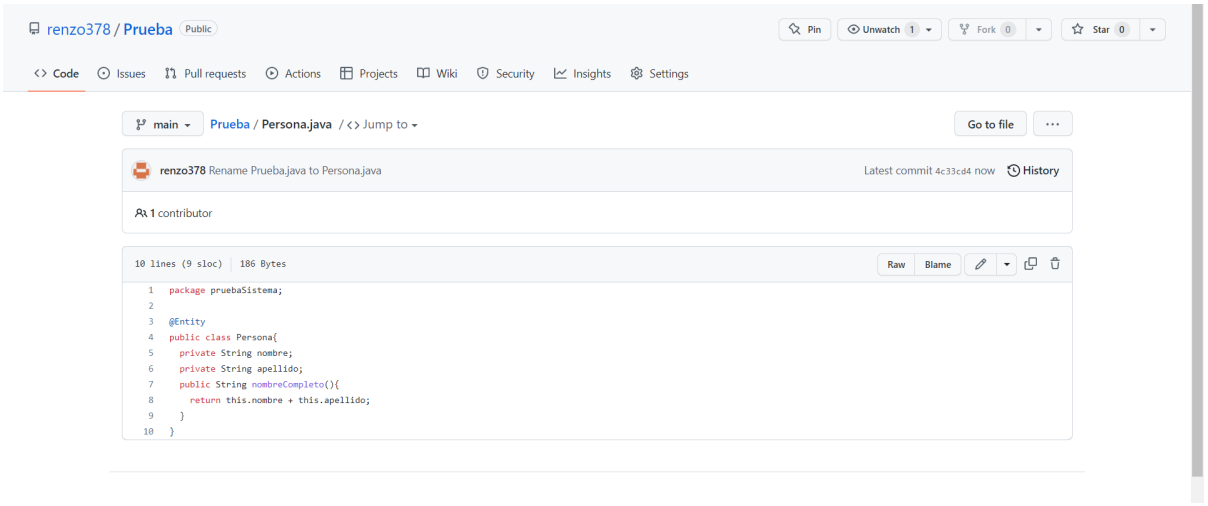


Figura 17.2.j: Contenido del archivo “Persona.java” la rama “main” del repositorio de prueba “Prueba”.

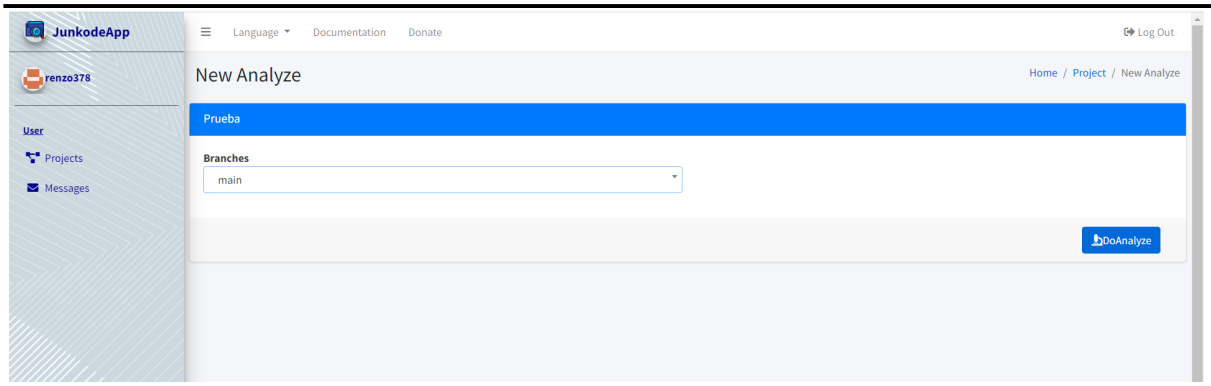


Figura 17.2.k: Pantalla correspondiente al Paso 01 de la Prueba número 06.

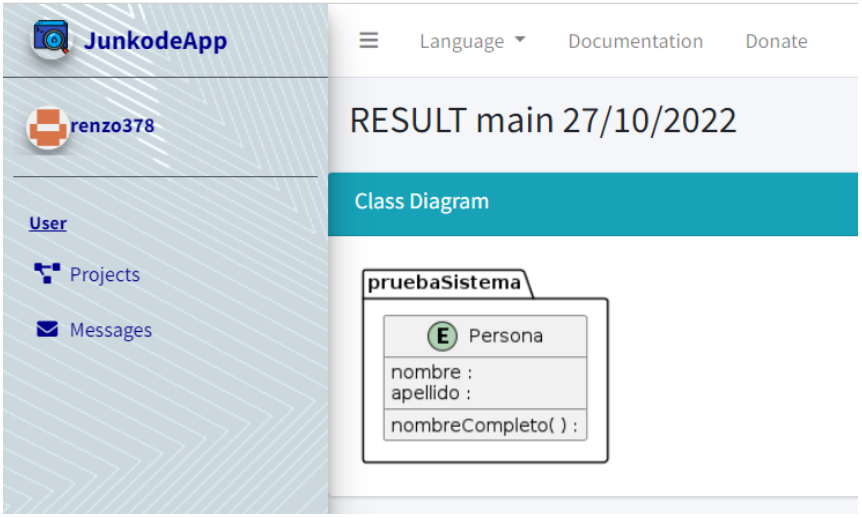


Figura 17.2.l: Pantalla correspondiente al Paso 02 fallido de la Prueba número 06.

17.3.Pruebas de integración entre módulos.

Prueba de sistema de integración entre módulos	
Número	07
Objetivo	Comprobar que se puedan observar correctamente todas las métricas analizadas en pantalla.
Módulos Integrados	Módulo Base de Análisis de Métricas. Módulo de Reportes.
Requerimientos	-Usuario logueado. -Proyecto creado. -Rama a analizar elegida.
Resultado esperado	Se muestran las métricas de la clase,de sus atributos y de su método.

<table>
<tr><td>Lote de prueba con atributos y valores a utilizar</td><td colspan="3">Usuario logueado con:
-username:renzo_fer00@hotmail.com.ar .
-password:contraseña1234 .

Proyecto con nombre “MetricasReportes” y repositorio “Prueba” asociado.

Rama “Main” elegida.

Repositorio “Prueba” el cual contiene 1 archivos con extensión “.java”.El mismo es una clase llamada “Persona” la cual tiene una dos atributos, “nombre” y “apellido” y un método llamado “nombreCompleto”.</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Se muestran las métricas de la clase sin mostrar las métricas de los atributos ni el método.</td></tr>
<tr><td rowspan="3">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario</td><td>Sistema</td></tr>
<tr><td>01</td><td>Seleccionar rama “Main”y hacer click en “DoAnalyze”.</td><td></td></tr>
<tr><td>02</td><td></td><td>Se muestra un reporte de la clase con sus métricas analizadas.</td></tr>
<tr><td>Acciones correctivas</td><td colspan="3">Corregir la clase “VieResult” ya que es donde se cargan los elementos que se deben mostrar en pantalla. Solo se están viendo las métricas de la clase pero no la de los atributos y métodos.</td></tr>
</table>

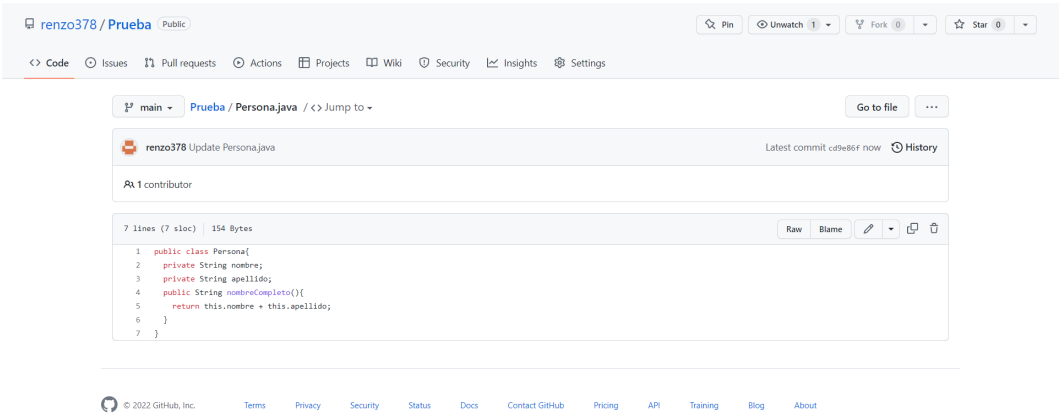


Figura 17.3.a: Contenido del archivo “Persona.java” la rama “main” del repositorio de prueba “Prueba”.

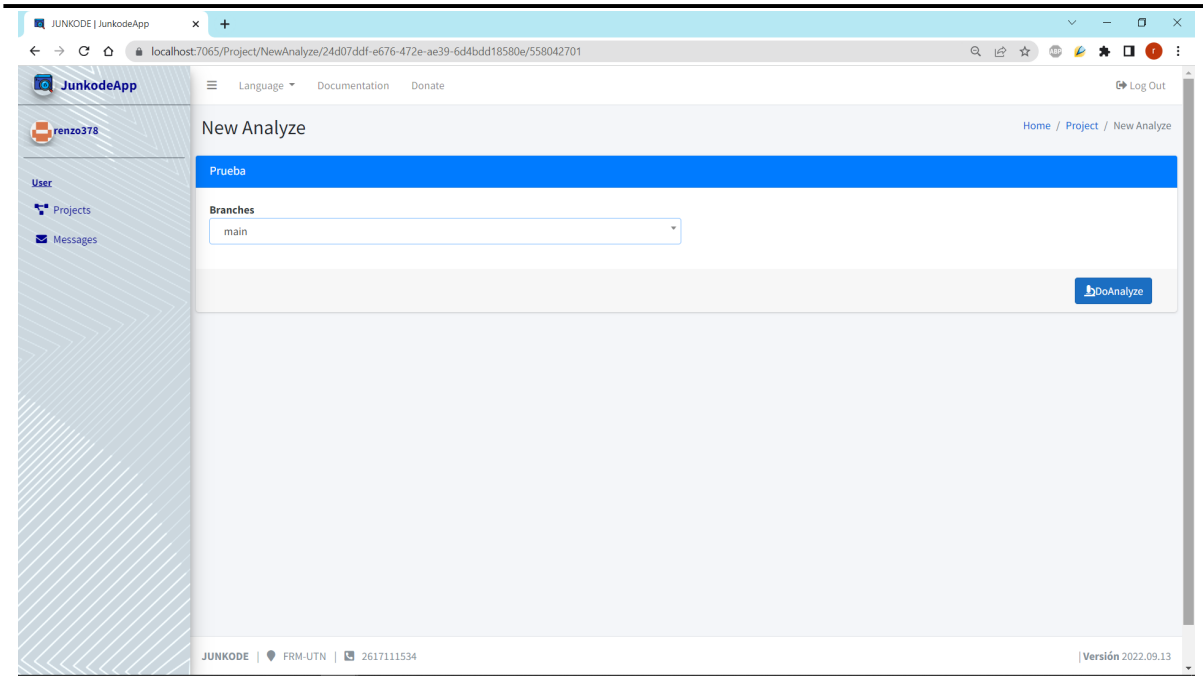


Figura 17.3.b: Pantalla correspondiente al Paso 01 de la Prueba número 07.

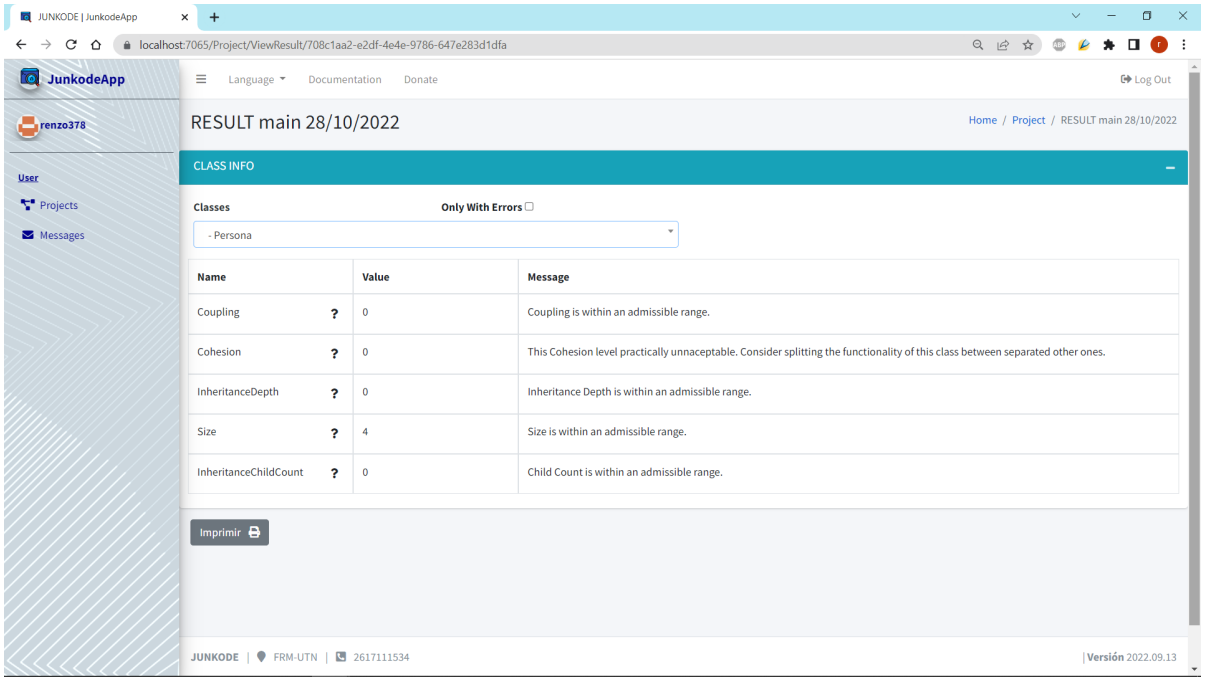


Figura 17.3.c: Pantalla correspondiente al Paso 02 fallido de la Prueba número 07.

Prueba de sistema de integración entre módulos	
Número	08
Objetivo	Comprobar que se puedan observar correctamente toda la documentación del proyecto analizado en el documento pdf.

Módulos Integrados	Módulo Base de Análisis de Documentación. Módulo de Reportes.		
Requerimientos	-Usuario logueado. -Proyecto creado. -Rama a analizar elegida.		
Resultado esperado	Se crea un pdf en el que se puede ver la documentación del proyecto analizado.		
Lote de prueba con atributos y valores a utilizar	Usuario logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 . Proyecto con nombre “Graficador” y repositorio “Prueba” asociado. Rama “Main” elegida. Repositorio “Prueba” el cual contiene 1 archivos con extensión “.java”.El mismo es una clase llamada “Persona” la cual tiene una anotación “@Entity”, dos atributos, “nombre” y “apellido” ambos con visibilidad “private” y tipo de dato “String” y un método llamado “nombreCompleto” el cual tiene visibilidad “public”, tipo de retorno “String” y forma parte del package “pruebaSistema”.		
Resultado obtenido	Se crea un pdf en blanco.		
Pasos lógicos de la prueba	N°	Usuario	Sistema
	01	Seleccionar rama “Main”y hacer click en “DoAnalyze”.	
	02		Se muestran los resultados del análisis.
	03	Bajar hasta el final de la pestaña y hacer click en “Imprimir”.	
	04		Se crea pdf en blanco.
Acciones correctivas	Corregir la función que inserta los resultados en el pdf ya que el pdf se crea correctamente pero no se incluyen los resultados por lo que el pdf queda en blanco.		

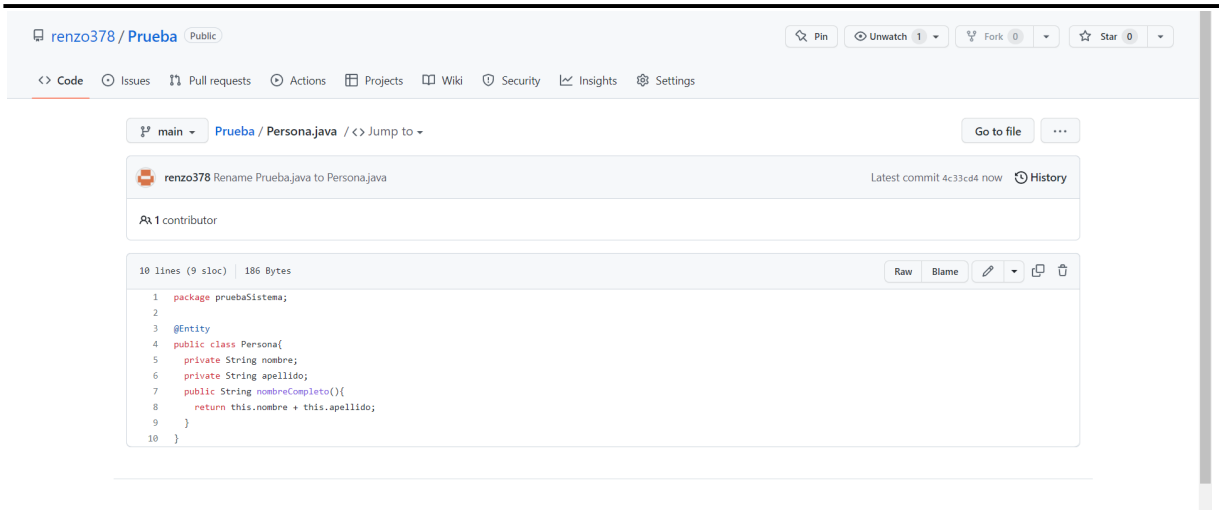


Figura 17.3.d: Contenido del archivo “Persona.java” la rama “main” del repositorio de prueba “Prueba”.

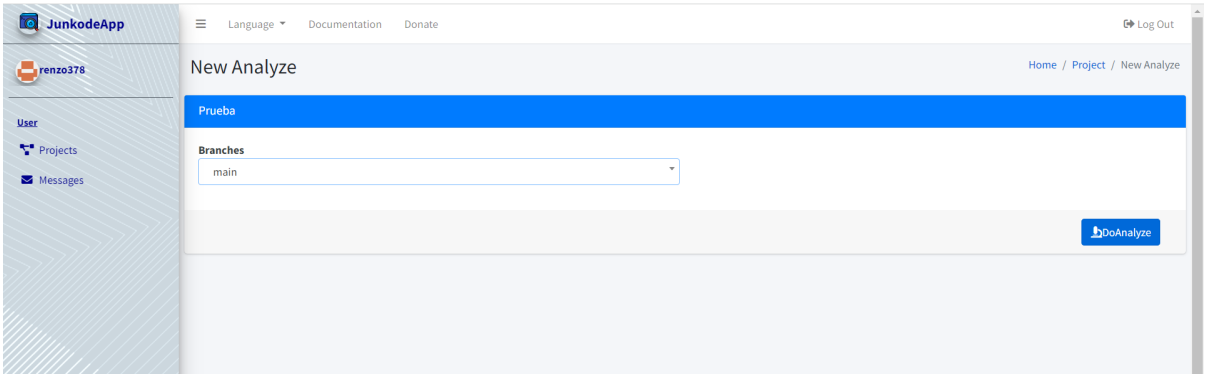


Figura 17.3.e: Pantalla correspondiente al Paso 01 de la Prueba número 08.

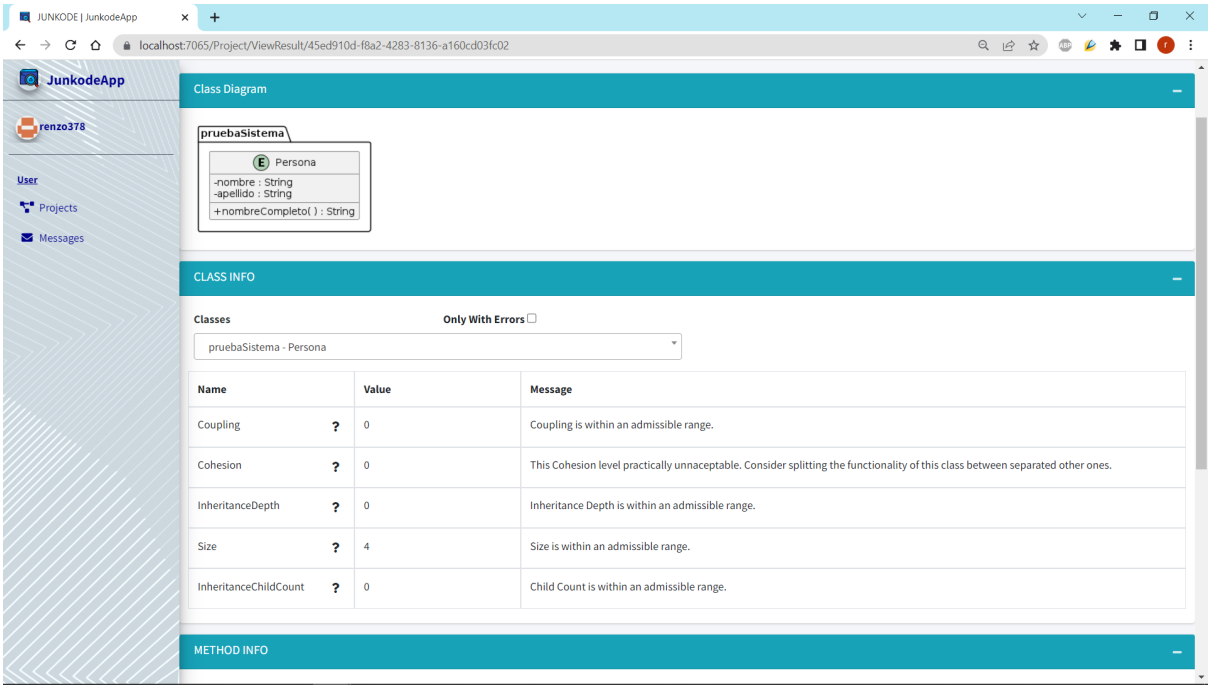


Figura 17.3.f: Pantalla correspondiente al Paso 02 de la Prueba número 08.

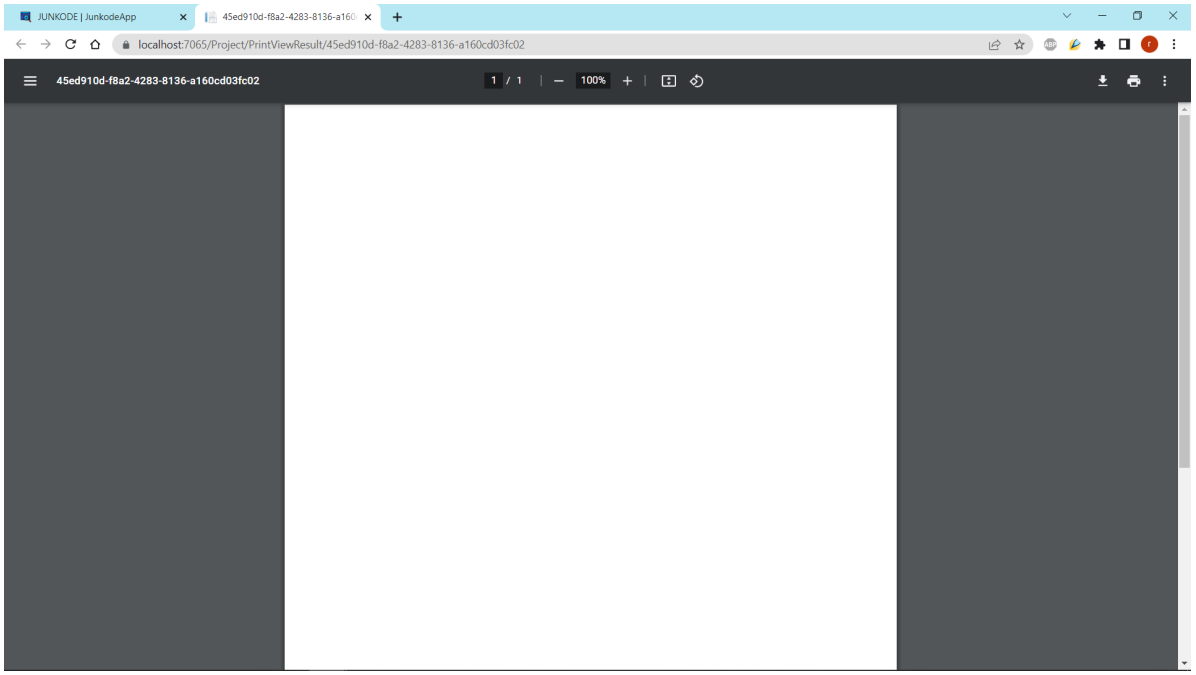


Figura 17.3.g: Pantalla correspondiente al Paso 04 fallido de la Prueba número 08.

Prueba de sistema de integración entre módulos	
Número	09
Objetivo	Un administrador pueda crear una nueva razón para el envío de mensajes y un usuario logueado con el rol de usuario pueda mandar un mensaje bajo esa razón.
Módulos Integrados	Módulo de Reportes. Módulo de Mensajería. Módulo de Administradores.
Requerimientos	-Usuario logueado con el rol de usuario. -Usuario logueado con el rol de administrador.
Resultado esperado	Nueva razón creada. Mensaje creado bajo esa razón.
Lote de prueba con atributos y valores a utilizar	Usuario(rol de administrador) logueado con: -username: ramsesgiralald@gmail.com . -password:ramseselmejor8 . Usuario(rol de usuario) logueado con: -username:renzo_fer00@hotmail.com.ar . -password:contraseña1234 .

<table>
<tr><td></td><td colspan="3">Nueva razón creada con el nombre “Help”.
Nuevo mensaje creado con la razón “Help” y el mensaje “Necesito ayuda. No puedo analizar mi proyecto”.</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Nueva razón creada.
Mensaje creado bajo esa razón.</td></tr>
<tr><td rowspan="11">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario (administrador)</td><td>Sistema</td></tr>
<tr><td>01</td><td>Hacer click en la opción “Reasons” de la barra lateral.</td><td></td></tr>
<tr><td>02</td><td>Hacer click en “New”.</td><td></td></tr>
<tr><td>03</td><td>En el campo Name completar con “Help” y hacer click en “Save”.</td><td></td></tr>
<tr><td>04</td><td></td><td>Nueva razón creada.</td></tr>
<tr><td>N°</td><td>Usuario (usuario)</td><td>Sistema</td></tr>
<tr><td>05</td><td>Hacer click en la opción “Messages” de la barra lateral.</td><td></td></tr>
<tr><td>06</td><td>Hacer click en “New Message”.</td><td></td></tr>
<tr><td>07</td><td>Elegir la razón “Help” y completar el campo Message con “Necesito ayuda. No puedo analizar mi proyecto.”.</td><td></td></tr>
<tr><td>08</td><td></td><td>Nuevo mensaje creado bajo el asunto “Help”.</td></tr>
</table>

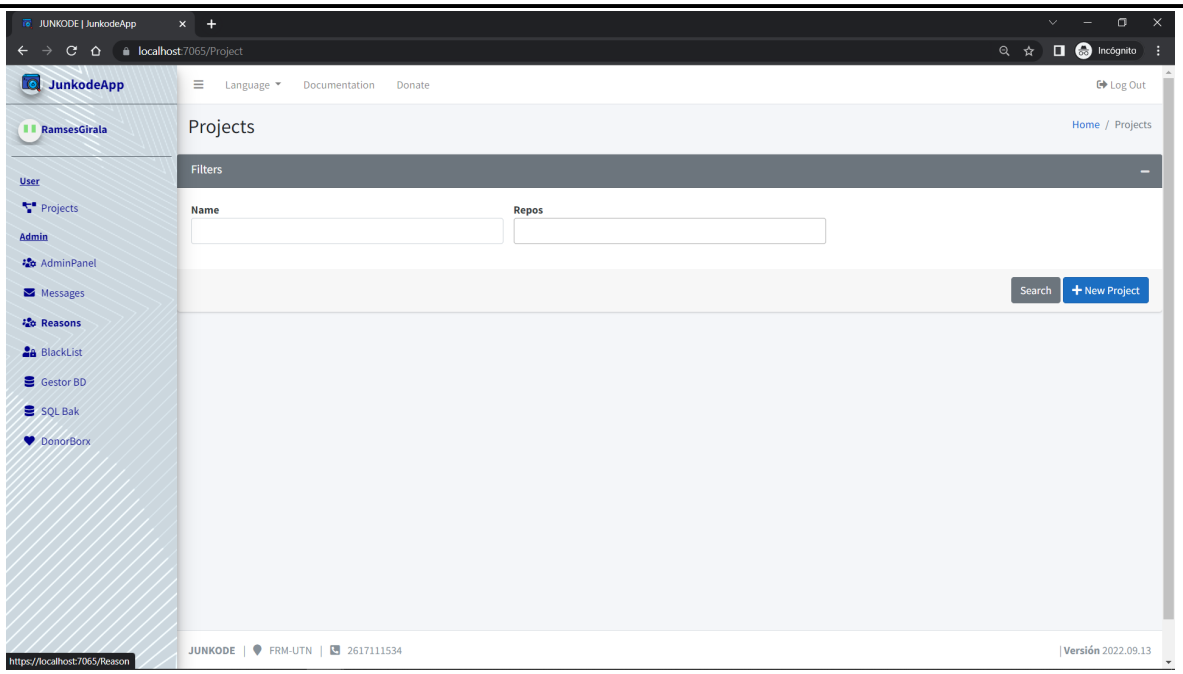


Figura 17.3.h: Pantalla correspondiente al Paso 01 de la Prueba número 09.

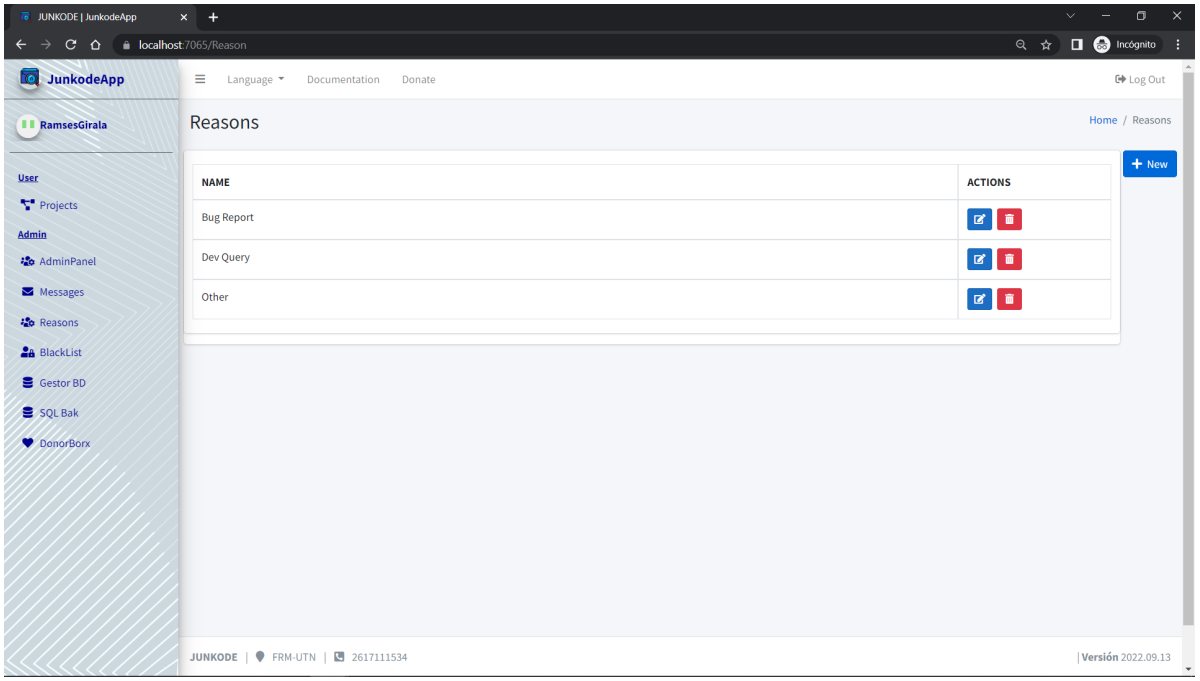


Figura 17.3.i: Pantalla correspondiente al Paso 02 de la Prueba número 09.

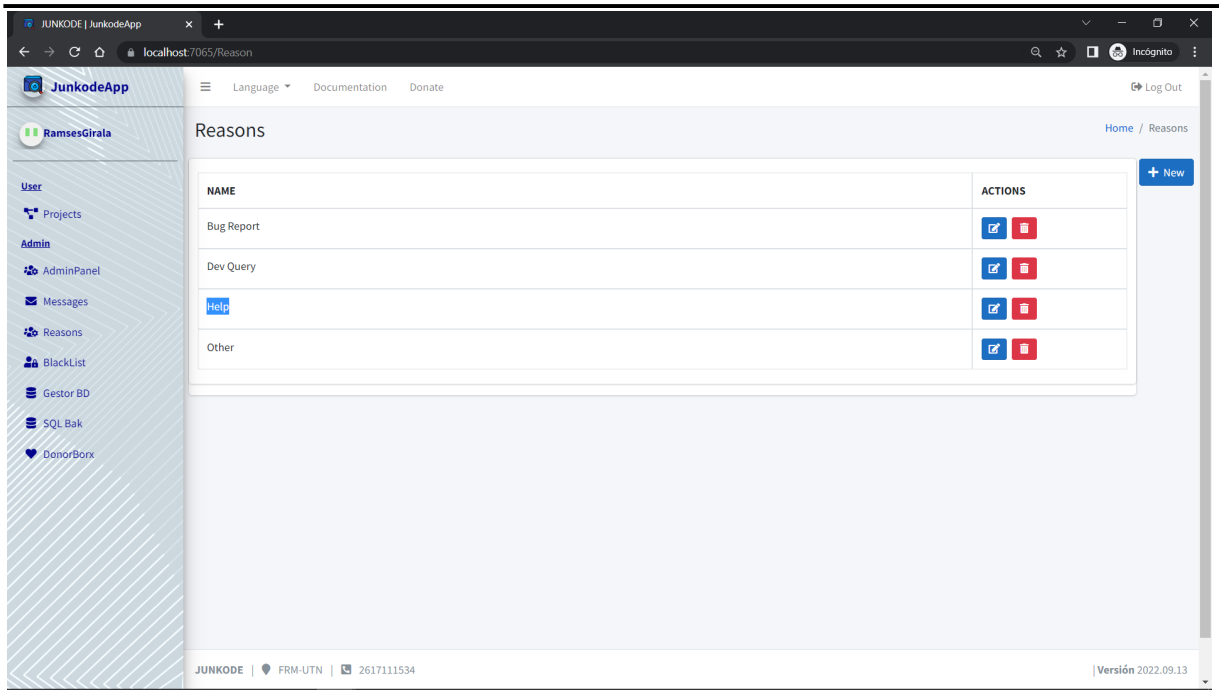


Figura 17.3.k: Pantalla correspondiente al Paso 03 de la Prueba número 09.

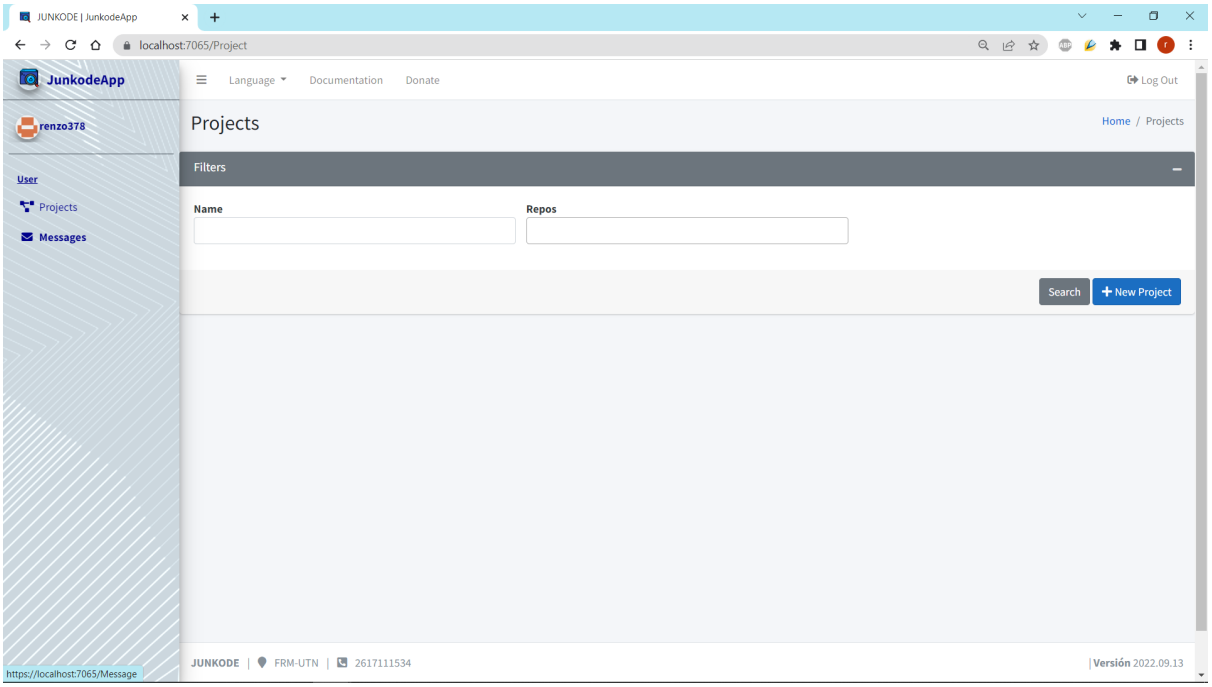


Figura 17.3.l: Pantalla correspondiente al Paso 05 de la Prueba número 09.

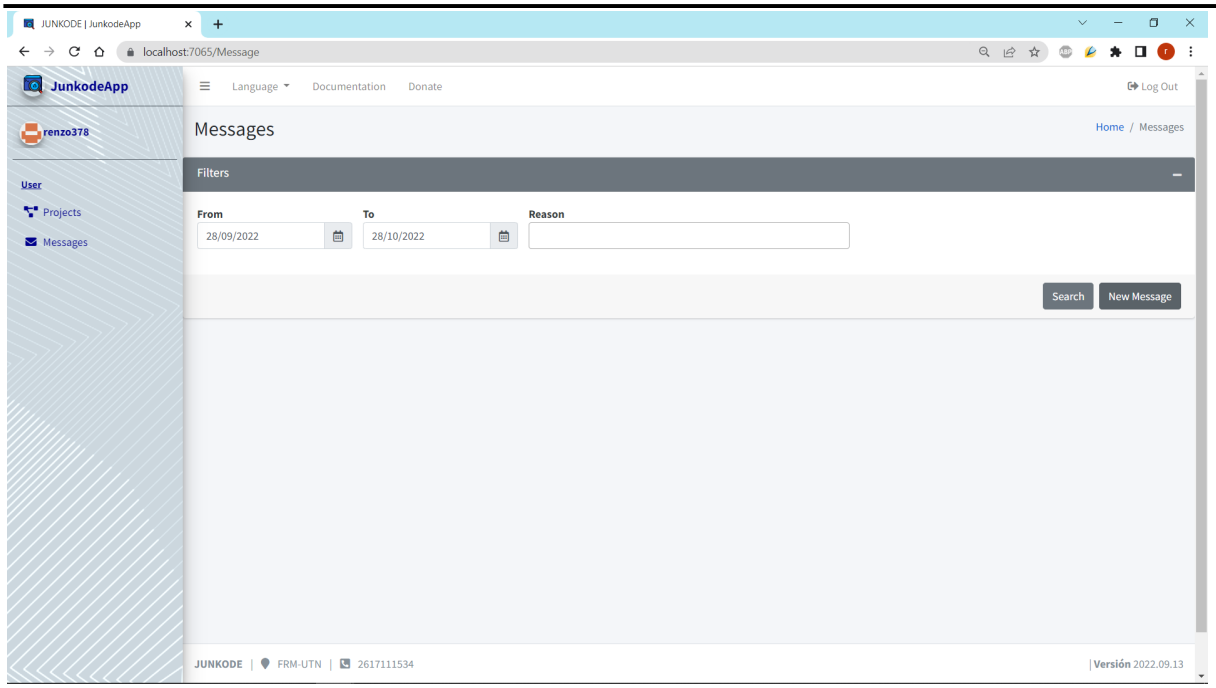


Figura 17.3.m: Pantalla correspondiente al Paso 06 de la Prueba número 09.

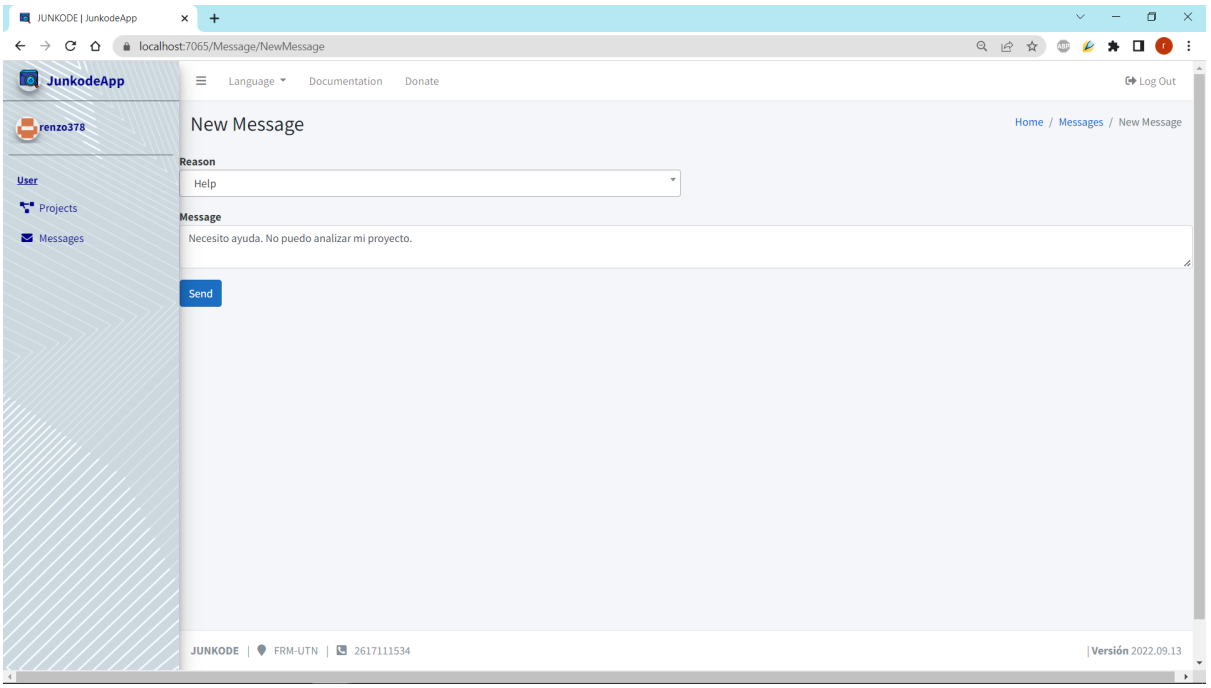


Figura 17.3.n: Pantalla correspondiente al Paso 07 de la Prueba número 09.

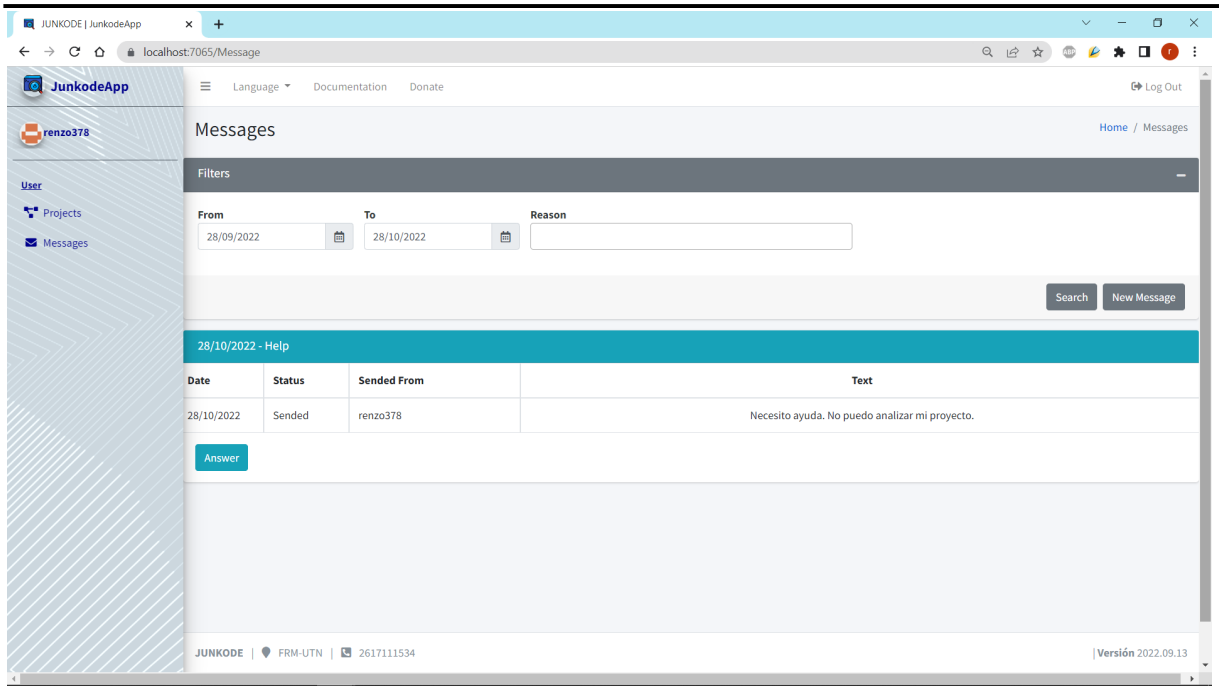


Figura 17.3.o: Pantalla correspondiente al Paso 08 exitoso de la Prueba número 09.

17.4.Pruebas de carga.

Prueba de carga	
Número	10
Objetivo	Comprobar que la API Graficadora soporte un mínimo de 50 consultas concurrentes (en un segundo).
Herramienta utilizada	Apache JMeter.
Resultado esperado	Éxito.
Resultado obtenido	Éxito.

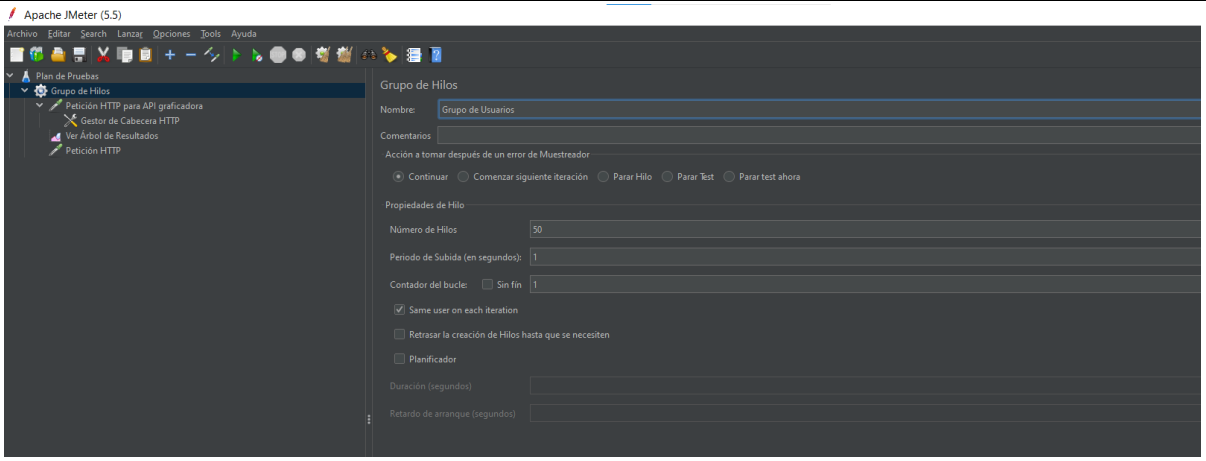


Figura 17.4.a: Pantalla con los parámetros de la Prueba número 10.

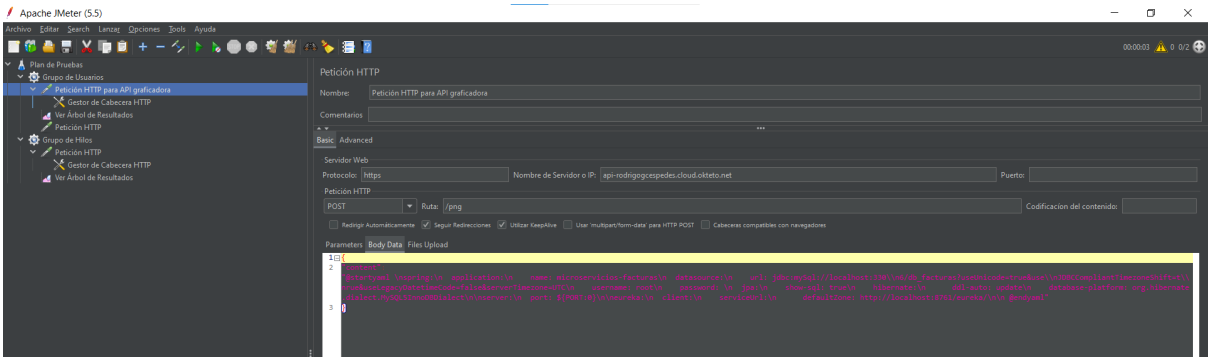


Figura 17.4.b: Pantalla con la ejecución de la carga de la Prueba número 10.

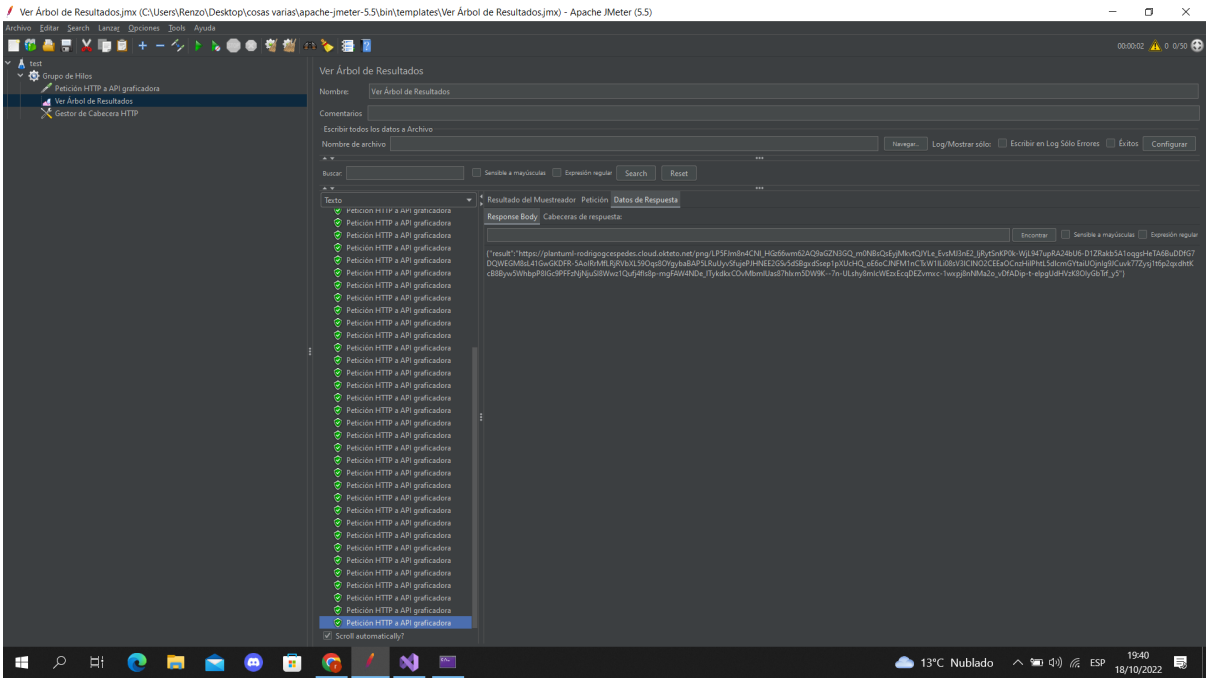


Figura 17.4.c: Pantalla con los resultados de las consultas de la Prueba número 10.

Prueba de carga	
Número	11
Objetivo	Comprobar que la API “app2yaml” responda con éxito al menos el 98% de las peticiones cuando recibe más de 100 peticiones en 5 segundos.
Herramienta utilizada	Apache JMeter.
Resultado esperado	Éxito.
Resultado obtenido	Éxito.

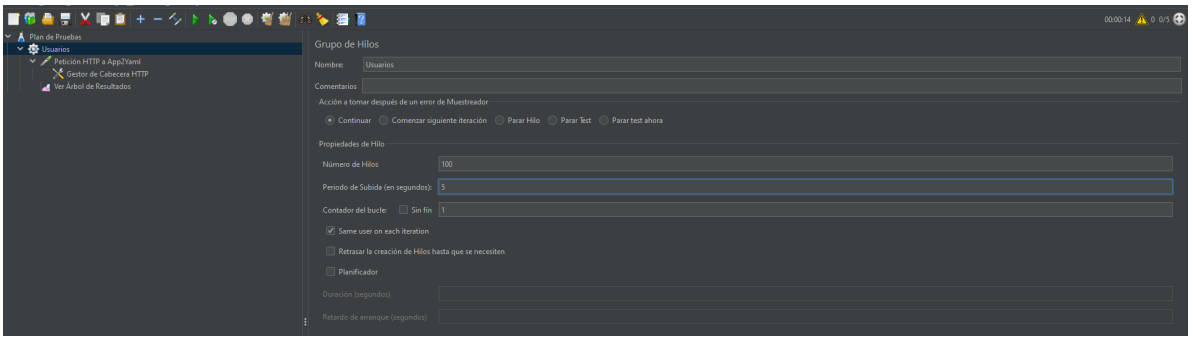


Figura 17.4.d: Pantalla con los parámetros de la Prueba número 11.

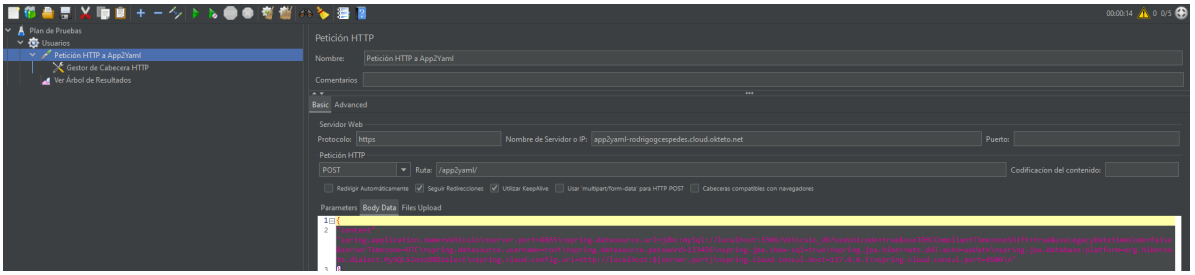


Figura 17.4.e: Pantalla con la ejecución de la carga de la Prueba número 11.

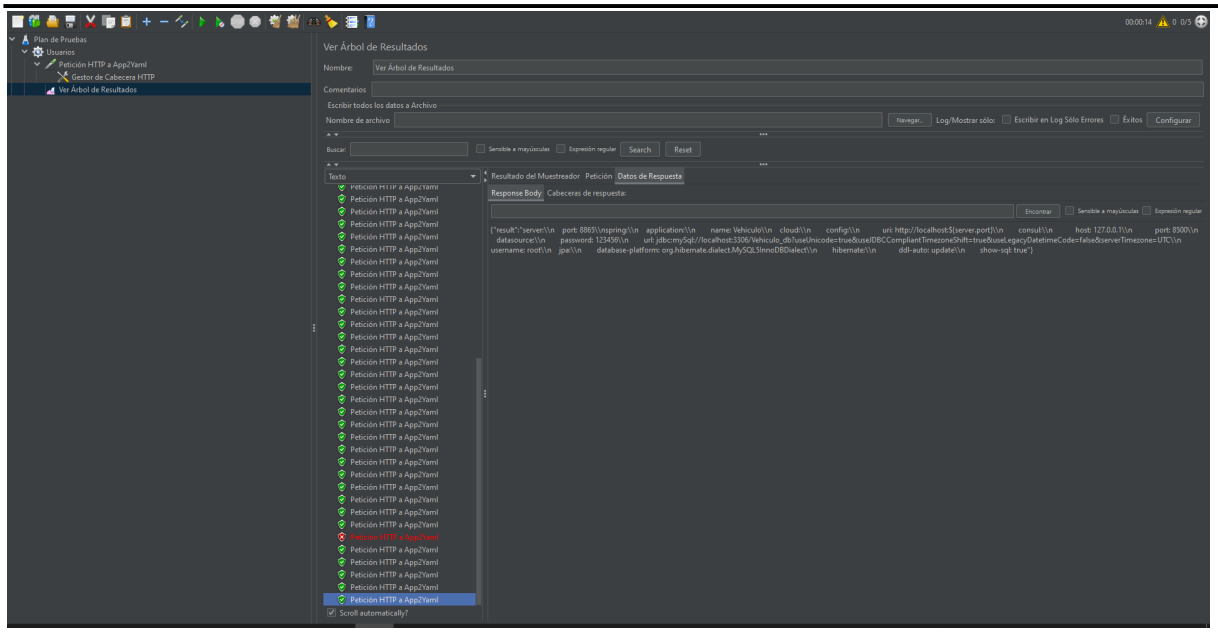


Figura 17.4.f: Pantalla con los resultados de las consultas de la Prueba número 11.

Prueba de carga	
Número	12
Objetivo	Comprobar que el servicio propio de PlantUML soporte un mínimo de 100 consultas concurrentes (en 5 segundos).
Herramienta utilizada	Apache JMeter.
Resultado esperado	Éxito.
Resultado obtenido	Éxito.

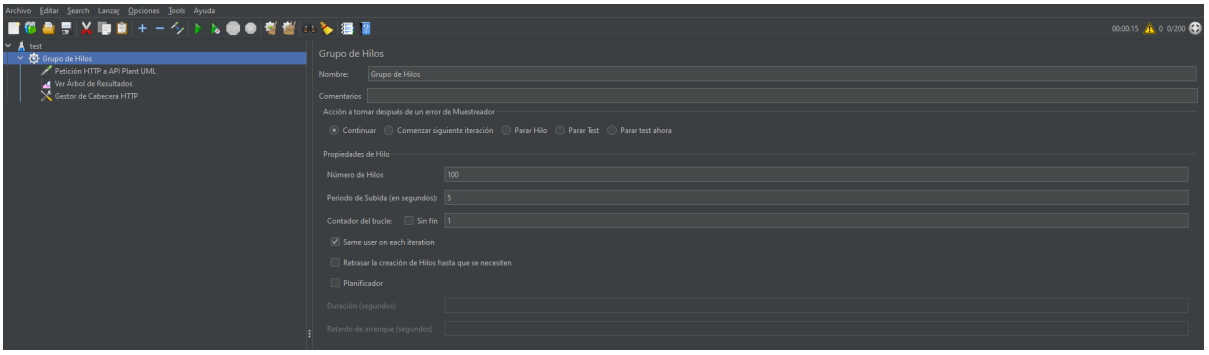


Figura 17.4.g: Pantalla con los parámetros de la Prueba número 12.

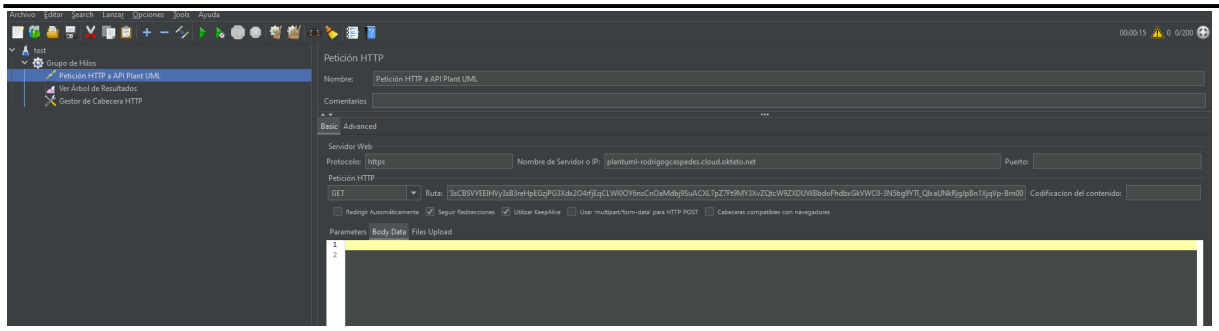


Figura 17.4.h: Pantalla con la ejecución de la carga de la Prueba número 12.

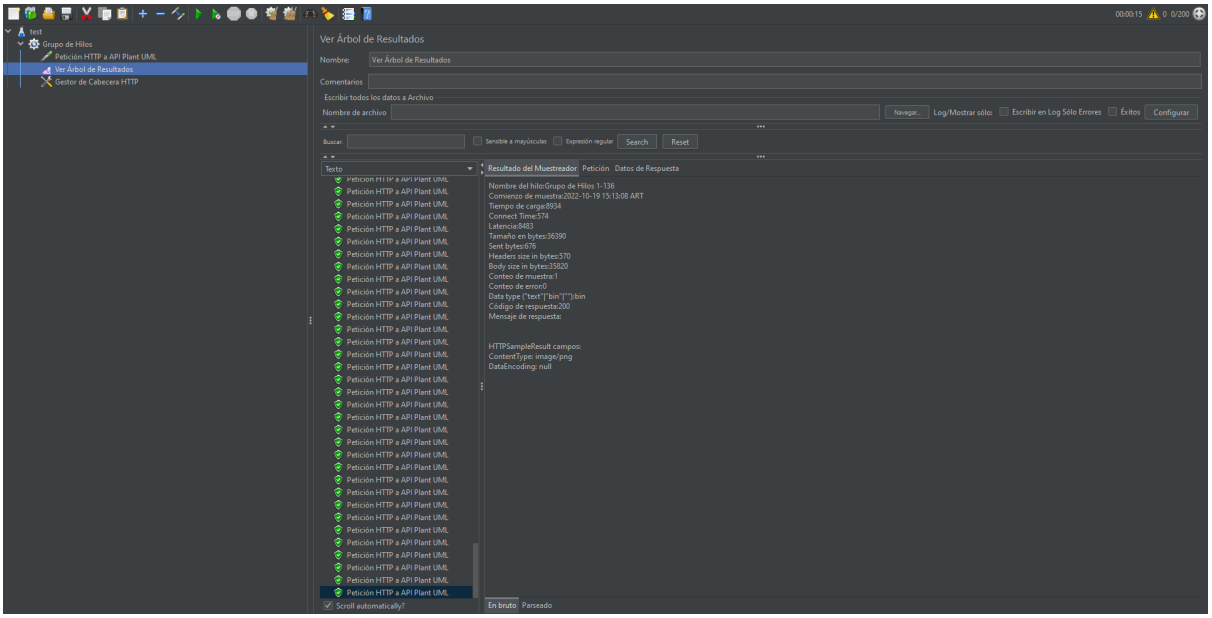


Figura 17.4.i: Pantalla con los resultados de la Prueba número 12.

17.5.Pruebas de seguridad por niveles de usuarios.

Prueba de seguridad por niveles de usuarios	
Número	13
Objetivo	Comprobar que un usuario no logueado no pueda acceder al menú de los usuarios logueados sin antes haberse logueado.
Requerimientos	-Usuario no logueado. -Estar situado en la landing Page.
Resultado esperado	Redirigir al usuario a la vista de logueo.
Lote de prueba con atributos y	Como no hace falta que el usuario esté logueado, no son necesarias credenciales ni valores específicos.

valores a utilizar			
Resultado obtenido	Redirigir al usuario a la vista de logueo.		
Pasos lógicos de la prueba	N°	Usuario	Sistema
	01	Se intenta acceder a la url https://localhost:7065/Project (la cual dirige al endpoint que contiene el listado de proyectos Junkode de un usuario).	
	02		Se redirige el usuario a una pestaña para que se loguee.

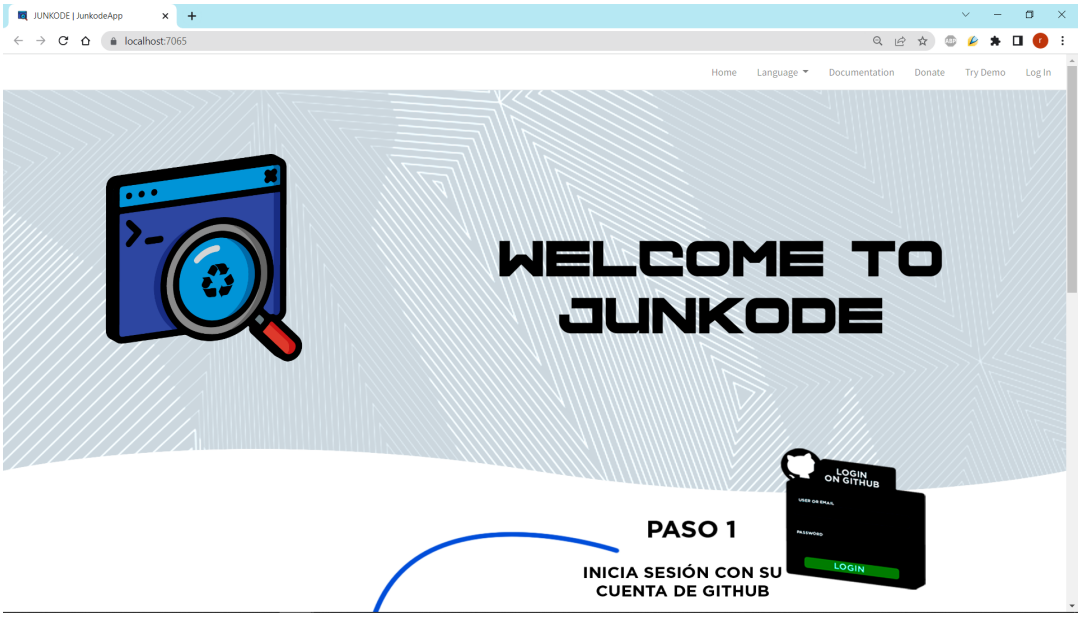


Figura 17.5.a: Pantalla home de Junkode.

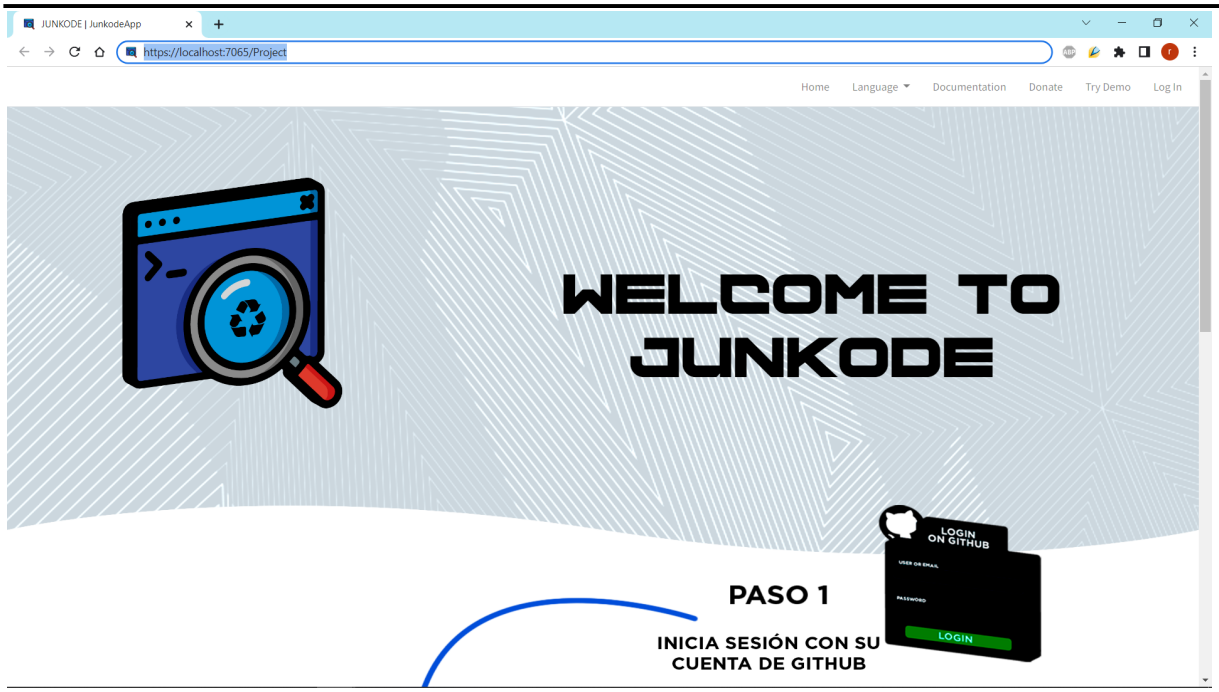


Figura 17.5.b: Pantalla correspondiente al Paso 01 de la Prueba número 13.

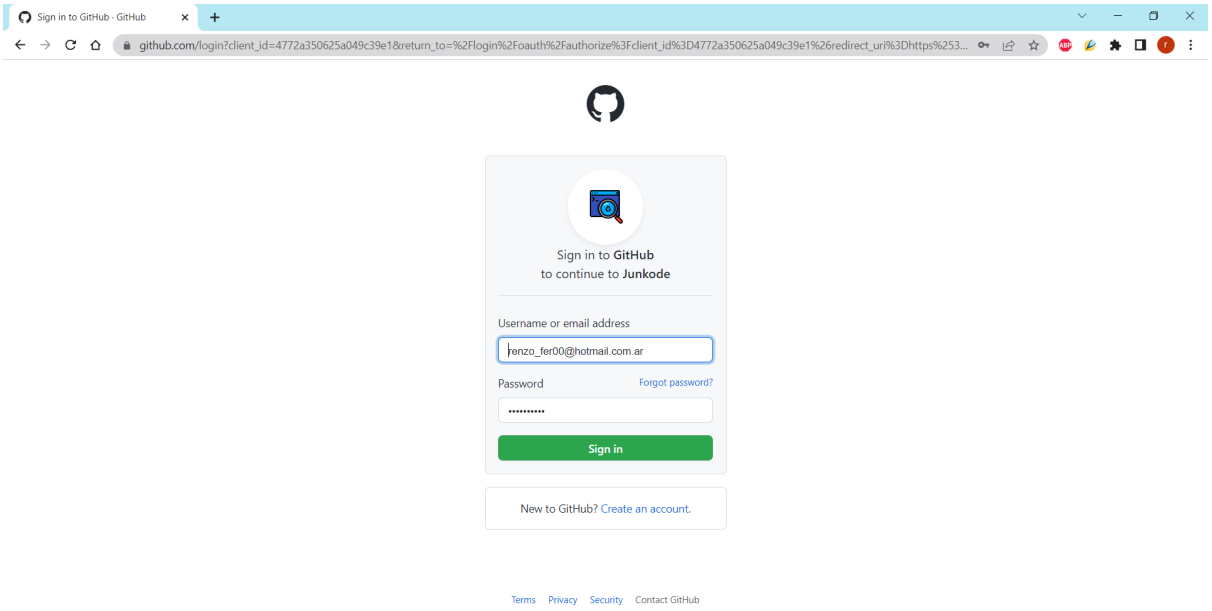


Figura 17.5.c: Pantalla correspondiente al Paso 02 exitoso de la Prueba número 13.

Prueba de seguridad por niveles de usuarios	
Número	14
Objetivo	Comprobar que un usuario logueado con el rol de usuario no pueda acceder a las funciones de los usuarios administradores.

Requerimientos	-Usuario logueado con el rol de usuario.		
Resultado esperado	Se redirige al usuario a una pestaña que indica que el acceso está prohibido para el usuario.		
Lote de prueba con atributos y valores a utilizar	Usuario(rol de usuario) logueado con: -username:renzo_fer00@hotmail.com.ar -password:contraseña1234		
Resultado obtenido	Usuario accede al área donde solo acceden los administradores.		
Pasos lógicos de la prueba	N°	Usuario	Sistema
	01	Se intenta acceder a la url https://localhost:7065/Admin (la cual dirige a un endpoint que contiene los dashboards de control de los administradores).	
	02		Se accede al área de administradores con todas sus funciones.
Acciones correctivas	Agregar en aquellas zonas a las que solo puedan acceder usuarios logueados con el rol de administrador políticas de seguridad.		

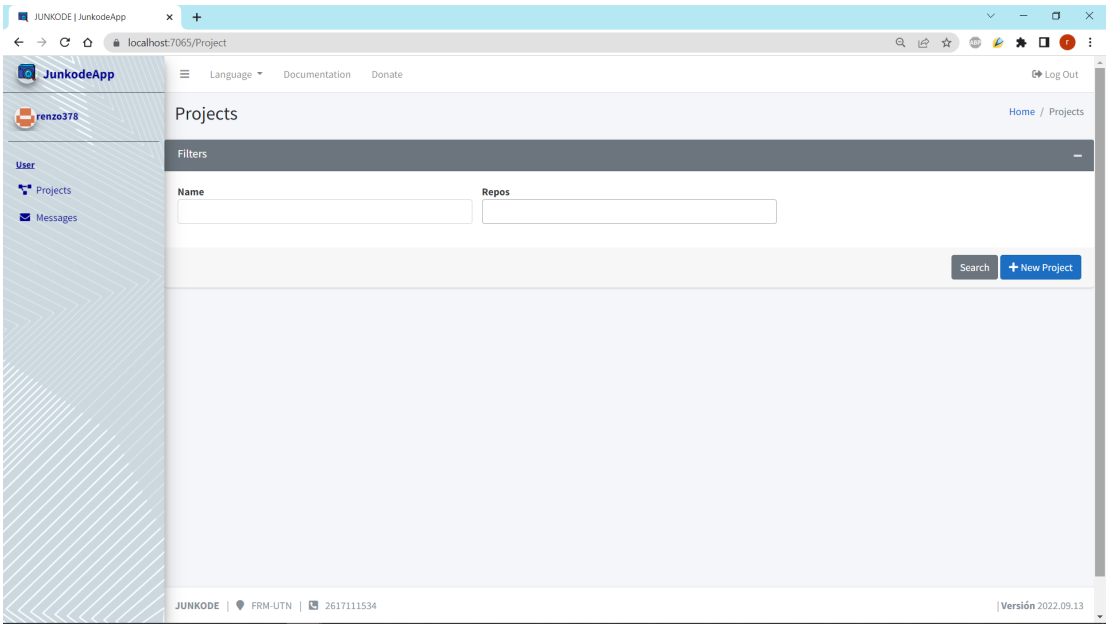


Figura 17.5.d: Pantalla Principal de la interfaz de Usuario Registrado con GitHub.

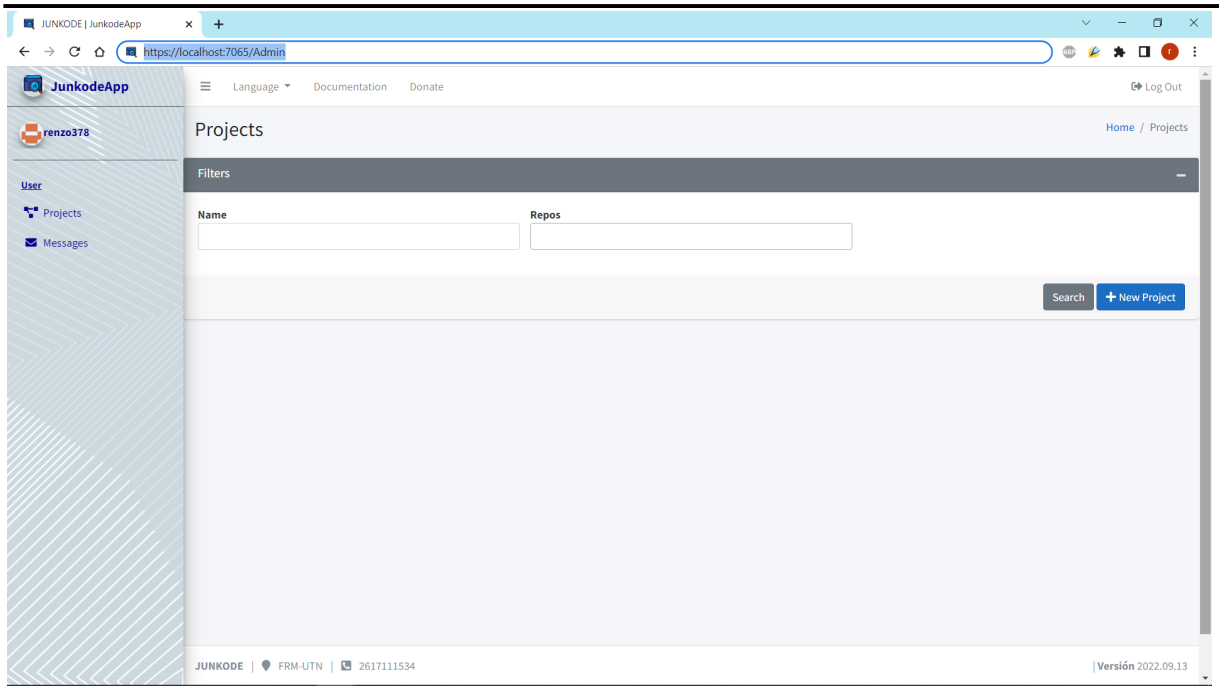


Figura 17.5.e: Pantalla correspondiente al Paso 01 de la Prueba número 14.

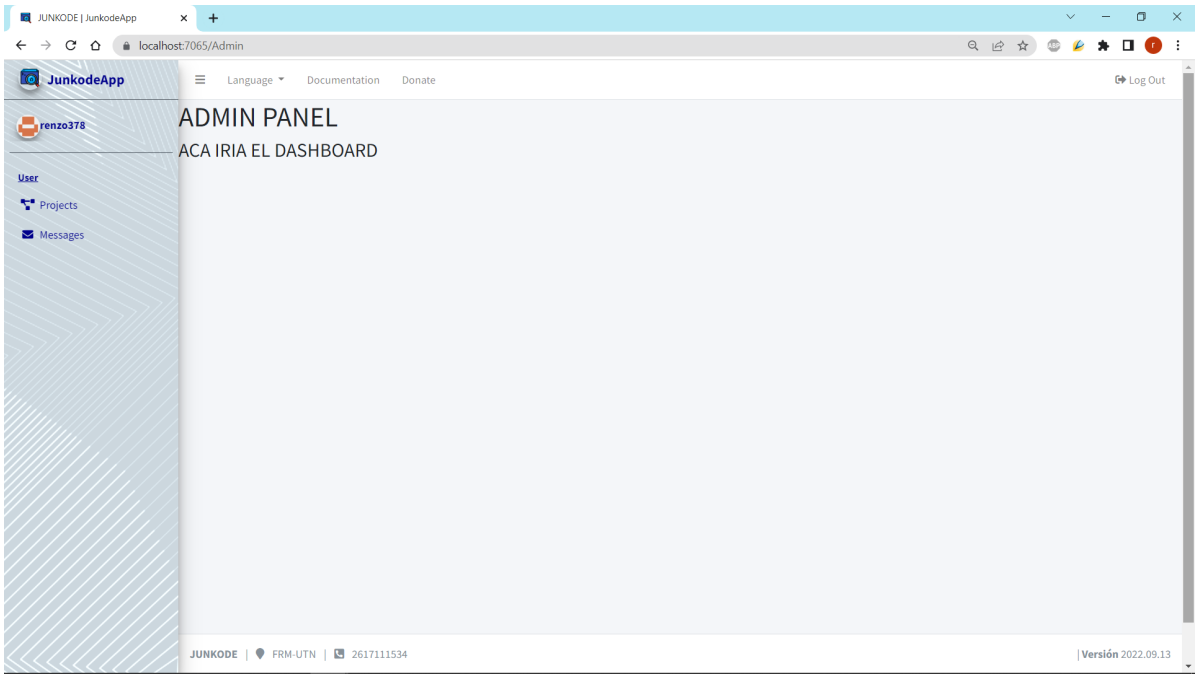


Figura 17.5.f: Pantalla correspondiente al Paso 02 fallido de la Prueba número 14.

Prueba de seguridad por niveles de usuarios	
Número	15
Objetivo	Comprobar que un usuario logueado como administrador tenga acceso a la blacklist y pueda ingresar a un usuario en ella.

<table>
<tr><td>Requerimientos</td><td colspan="3">-Usuario logueado como administrador.
-Usuario con rol de usuario existente en base de datos.</td></tr>
<tr><td>Resultado esperado</td><td colspan="3">Usuario “renzo378” añadido a la blacklist.</td></tr>
<tr><td>Lote de prueba con atributos y valores a utilizar</td><td colspan="3">Usuario(rol de administrador) logueado con:
-username: ramsesgiralald@gmail.com .
-password:ramseselmejor8 .

Usuario(rol de usuario) logueado con:
-username:renzo_fer00@hotmail.com.ar .
-password:contraseña1234 .</td></tr>
<tr><td>Resultado obtenido</td><td colspan="3">Usuario “renzo378” añadido a la blacklist.</td></tr>
<tr><td rowspan="4">Pasos lógicos de la prueba</td><td>N°</td><td>Usuario(administrador)</td><td>Sistema</td></tr>
<tr><td>01</td><td>Hacer click en la opción “Blacklist” de la barra lateral.</td><td></td></tr>
<tr><td>02</td><td>Seleccionar al usuario “renzo378” y hacer click en el botón “Add”.</td><td></td></tr>
<tr><td>03</td><td></td><td>Muestra una tabla que contiene a los usuario que se encuentran en la blacklist.</td></tr>
</table>

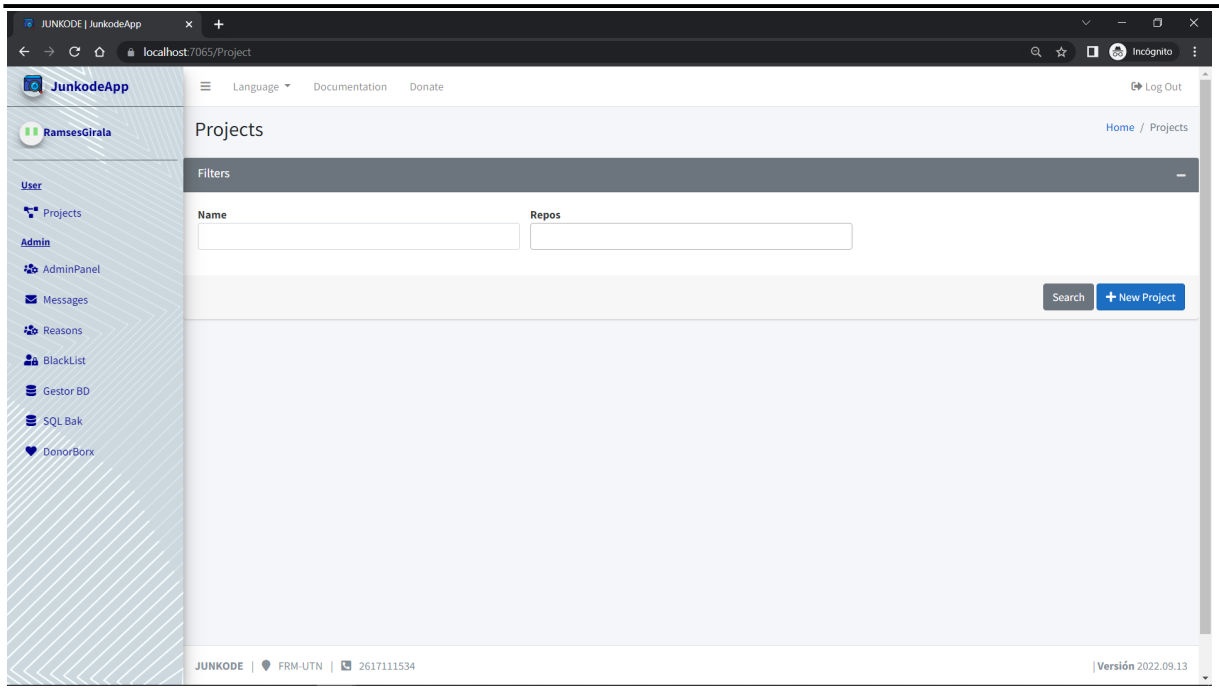


Figura 17.5.g: Pantalla correspondiente al Paso 01 de la Prueba número 15.

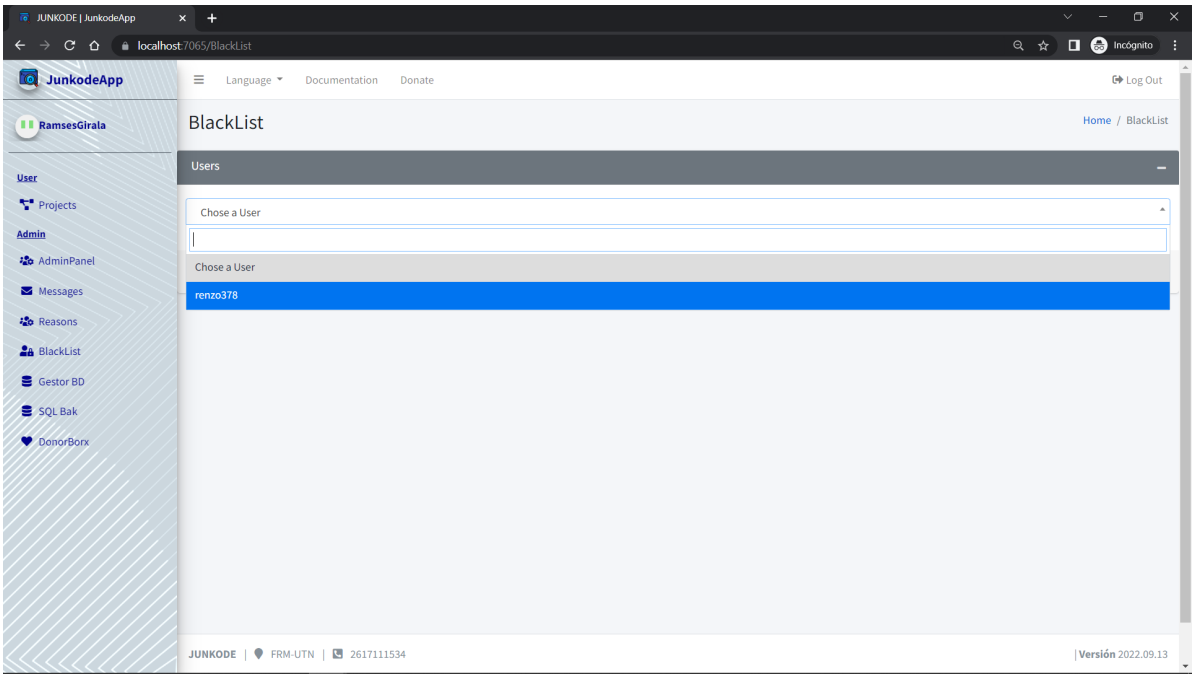


Figura 17.5.h: Pantalla correspondiente al Paso 02 de la Prueba número 15.

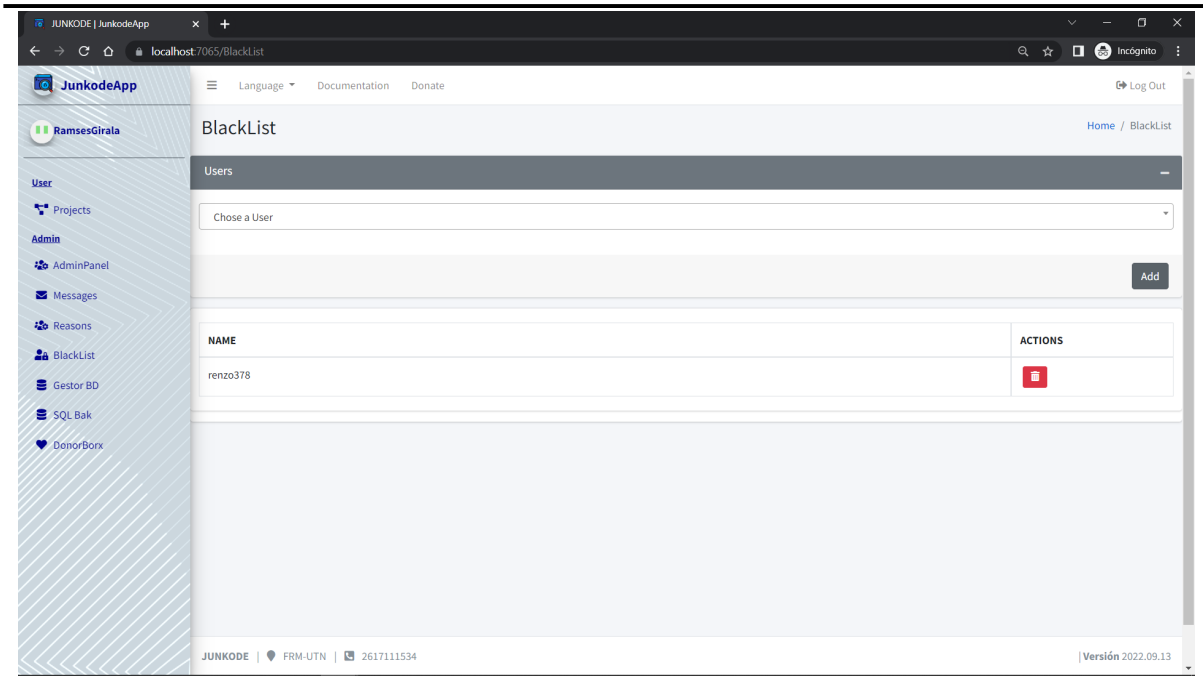


Figura 17.5.i: Pantalla correspondiente al Paso 03 exitoso de la Prueba número 15.

18.Manuales de Usuario.

Los manuales de usuario son una guía detallada de las funcionalidades que Junkode presenta.

Estas funcionalidades se encuentran separadas en dos manuales, el Manual de Usuario y el Manual de Administradores. Solamente el Manual de Usuario está abierto al público, puesto que el de Administradores se encuentra reservado solo para la consulta de estos.

Los manuales pueden ser encontrados en:

- ➢ Anexo N°9: Manual de Usuario de Junkode.
- ➢ Anexo N°10: Manual de Administradores de Junkode.

19.Planificación de la Implementación del sistema.

Si bien Junkode nació con la idea de ser un proyecto monolítico, con el avance del desarrollo se optó por convertirlo en un proyecto de microservicios (con un microservicio principal muy grande).

En vista de que el Junkode necesitaría del despliegue e implementación de múltiples servicios desarrollados por el equipo de trabajo y la conexión con muchos otros servicios externos, se tomó la decisión de realizar una implementación por partes, en la que cada servicio pudiera desarrollarse y posteriormente desplegarse por separado, independientemente del estado del desarrollo o implementación de los demás servicios.

19.1.Programa de la Implementación de Junkode.

19.1.1.Objetivos de la implementación:

➢ Desplegar y conectar correctamente todos los servicios nativos que conforman a Junkode.
➢ Conectar correctamente los servicios externos al servicio principal de Junkode una vez que este ya se encuentre desplegado.
➢ Desplegar el sistema dentro de los requerimientos máximos de Okteto.
➢ Proveer al sistema con un dominio propio y contrato SSL gratuitos.

19.1.2.Recursos involucrados:

Como el proyecto en sí no requiere de recursos financieros ni materiales para su funcionamiento (por estar hecho con tecnologías de software libre y tecnologías en la nube, respectivamente), los únicos recursos de los que se precisan son los recursos humanos y los servicios.

➢ Recursos Humanos:
○ Las tareas de implementación solo necesitan de la completa atención de un solo desarrollador del equipo de Junkode. Sin embargo, debido a que los servicios que de desplegarán son diversos y fueron desarrollados por distintos desarrolladores, es probable que se deba acudir a estos en busca de asesoramiento o incluso ayuda en momentos específicos del proceso de puesta en marcha (Es importante resaltar de todas formas que solo habrá un desarrollador dedicado a estas tareas).

➢ Servicios:
○ Servicio de Despliegue de servicios (Okteto).
○ Servicio de Motor de Bases de Datos en la nube (MySQL hosteado en remotemysql.com).
○ Servicio de Graficador (Imagen de Docker de PlantUML).
○ Servicio de API Graficadora (Imagen Docker de una API en Python).
○ Servicio remoto de conversión app2yaml (JavaInUse).
○ Servicio de API app2yaml (Imagen Docker de una API en Python).
○ Servicio de Autenticación de usuarios (OAuth 2.0 y GitHub).
○ Servicio remoto de donaciones (Donorbox).
○ Servicio remoto de gestión de Backups (SQLBak).
○ Servicio de Dominios gratuitos web (freenom y CloudDNS).
○ Servicio de SSL (SSL For Free).

19.1.3.Pertinencia de las actividades:

Si bien el sistema está formado por microservicios, estos no tienen utilidad por separado, ya sea porque dependen de otros para funcionar o porque su funcionalidad es tan específica y centrada en un agregado que no tendría sentido tomarlo en consideración como un servicio aislado.

Es por esto que se optó por aplicar un método de implementación directo, en el que los servicios se irán desplegando y levantando uno por uno, para que puedan ser consultados por los servicios que requieran que estén activos, pero que el objetivo final del plan de implementación sea el despliegue de Junkode como sistema único e íntegro.

Algo importante a resaltar es que en el programa solo figuran tareas referidas al despliegue de servicios nativos y no de los externos. Esto se debe a que los servicios externos ya se encuentran conectados al servicio principal de Junkode y ya fueron testeados durante su desarrollo, por lo que ya no es necesario hacerlo nuevamente aquí.

Luego que los servicios se hayan desplegado y conectado, se realizará un último test, el cual consiste en una prueba de que el sistema ya integrado se encuentre funcionando correctamente y, finalmente, es necesaria una evaluación del deployment ya realizado, comparando los objetivos planteados inicialmente con los resultados finalmente obtenidos.

19.2.Listado de tareas para la implementación:

Dado que todas estas tareas son realizadas por la misma persona y que los recursos no son insumos, sino información que puede ser utilizada de manera simultánea para diversas tareas, todas las tareas a continuación tienen un orden secuencial de consecuencia, o sea, una tarea comienza cuando termina su inmediata anterior. La duración se encuentra indicada en días hábiles de 8 horas cada uno. También puede consultarse el diagrama GANTT del proyecto completo en el Anexo N°4: Diagrama de GANTT de Junkode, el cual incluye estas actividades.

1. Despliegue de la Base de Datos en la nube:
   - Descripción: Esta tarea consiste en la creación de una base de datos en el servidor remotemysql.com. Esta debe crearse atándose a los alcances máximos que ofrece el servidor en la nube para despliegues gratuitos.
   - Duración: Hito.
   - Predecesor: Integración de las herramientas.

2. Configuración del servicio de Backup:
   - Descripción: Esta tarea consiste en la configuración de Job automático de generación de Backups en el servicio Web SQLBak. En este se deben almacenar las credenciales de acceso a la Base de Datos, elegir la frecuencia con que se realizarán los Backups (quincenalmente), ingresar las credenciales del repositorio donde se almacenarán (Google Drive), seleccionar las opciones de encriptado pertinentes y definir la frecuencia con la que los Backups serán eliminados automáticamente.
   - Duración: Hito.
   - Predecesor: Despliegue de la Base de Datos en la nube.

3. Creación de las tablas de la Base de Datos:
   - Descripción: Se ejecuta un script SQL que genera todas las tablas de la base de datos, incluyendo entidades, auditorías y usuarios. Este script ya fue generado automáticamente antes durante el desarrollo de la base de datos.

- Duración: Hito.
- Predecesor: Conexión del Gestor a la Base de Datos en la nube.

4. Creación de imagen de Graficador:
- Descripción: Durante esta tarea se desarrolla el servicio de API graficadora, el cual debe recibir un objeto JSON que incluya la sintaxis de PlantUML y entregarselo a un servicio graficador de PlantUML para que lo procese y retorne una imagen con el modelo de clases[1]. Para desarrollar esta tarea se necesita una imagen de docker que consiste en un servidor Tomcat que tiene corriendo el servicio graficador de PlantUML. Esta imagen es provista por los desarrolladores de PlantUML en su Docker Hub.
- Duración: 4 días.
- Predecesor: Creación de las tablas de la Base de Datos.

5. Creación de imagen de API app2yaml:
- Descripción: Similar a la tarea anterior, esta tarea consiste en desarrollar una API en Python que reciba un objeto JSON con texto en formato de application.properties y lo redirija al servicio conversor “app2yaml” del sitio web JavaInUse para que lo procese y retorne otro objeto JSON, pero esta vez con texto en formato yaml. En este caso, se consume directamente el servicio web de JavaInUse, por lo que no es necesario desplegar este servicio en un cluster propio, como si fue necesario para el servicio de PlantUML de la tarea anterior.
- Duración: 4 días.
- Predecesor: Creación de las tablas de la Base de Datos.

6. Despliegue del servicio PlantUML, API Graficadora y API app2yaml:
- Descripción: El servicio graficador (API graficadora y servicio de PlantUML) se despliega en su propio cluster de Okteto. Cada uno de estos servicios se despliega junto con sus respectivos “Services” y “ReplicaSets” de Kubernetes (resaltando que de cada uno de estos servicios se pueden crear hasta 3 réplicas) y se expone el puerto público al que este puede ser consultado. No hay necesidad de configurar un Gateway, Discovery Service ni Load Balancer porque se utiliza el Ingress Controller nativo de Okteto (que es básicamente un NGINX Gateway) para controlar los ingresos y sesiones de las réplicas de los servicios que se crearán cuando sea necesario.
- Duración: Hito.
- Predecesor: Creación de imagen de Graficador y Creación de imagen de API app2yaml.

7. Despliegue del servicio Junkode:
- Descripción: El servicio principal de Junkode es el que contiene todos los servicios no mencionados los anteriores pasos del deployment (entiendase, análisis de código, ABMs de administrador, módulos del front-end, conexiones a los demás servicios y documentación). Este servicio se despliega en un cluster distinto al de los servicios desplegados anteriormente. Este servicio se despliega en conjunto con sus respectivos “Services” y “ReplicaSets” de Kubernetes y se expone el puerto público al que este puede

[1] Esta misma API también es utilizada para la generación de los diagramas de propiedades que fueron agregados al sistema durante el segundo módulo incremental.

ser consultado. Este servicio cuenta con hasta diez réplicas para ser creadas y orquestadas por el Ingress Controller nativo de Okteto.
- Duración: Hito.
- Predecesor: Despliegue del servicio PlantUML, API Graficadora y API app2yaml.

8. Testing del Despliegue:
- Descripción: Se realiza un prueba rápida que consiste en entrar a todos los endpoints públicos generados y corroborar que todos se hayan conectado correctamente. Esta prueba se realiza por medio del mismo navegador, por MySQL-Workbench y Postman.
- Duración: 2 días.
- Predecesor: Despliegue del servicio Junkode.

9. Configurar dominio de red:
- Descripción: Por medio del portal CloudDNS se le asigna un nombre distintivo del sistema a los endpoints que deben ser accedidos por el público.
- Duración: Hito.
- Predecesor: Testing del Despliegue.

10. Implementar certificación SSL:
- Descripción: Por medio del portal SSL for Free se obtiene un contrato de SSL para que el sistema se exponga bajo el protocolo https. La key de este contrato debe ser dockerizada y deployada como un elemento “Secret” de Kubernetes en el cluster donde se encuentra la base de datos.
- Duración: Hito.
- Predecesor: Cambiar nombre de dominio.

11. Evaluación de la implementación resultante:
- Descripción: Una vez finalizado el proceso, se comparan los resultados obtenidos con los objetivos planteados inicialmente y se proponen tareas de rectificación y/o control que puedan ser necesarias.
- Duración: 1 día.
- Predecesor: Implementar un certificado SSL como Secret.

PLANIFICACIÓN DE PROYECTOS DE SISTEMAS

20.Actividades.

20.1.Definición y descripción de actividades.

20.1.1.Etapa de Investigación:

- **Investigación de herramientas similares:** Este paso consiste en averiguar si hay herramientas similares actualmente en existencia. Esto se hace con el fin de saber en qué enfocarse y también de abrir la vista a opciones ya implementadas.
- **Investigación de herramientas de graficación:** Básicamente es necesaria una herramienta que reciba texto y con eso pueda graficar los diagramas. Fue utilizada una anteriormente pero ante la dificultad de implementarla para hacer cosas automáticamente surgió la necesidad de buscar una nueva.
- **Investigación de herramientas de análisis de texto:** Esta tarea afortunadamente ya había sido satisfecha para el momento de hacer esta lista, sin embargo permanece en la lista porque sería algo que naturalmente tendría que planificarse, por eso se le asignó una duración tan corta y al principio del proyecto.
- **Investigación de buenas prácticas de codificación:** Es importante conocer las normas y estándares de programación, pero también las buenas prácticas de programación, las cuales muchas veces se contradicen con los estándares.
- **Investigación de herramientas para deployment del sistema:** Esta tarea puede tomar mucho más tiempo que las demás, porque investigación no solo implica averiguar que la herramienta existe y poder instalarla, sino también poder hacer que funcione, y como esta tarea (como muchas de las siguientes) depende de tecnologias que aun no decidimos, puede tomar mucho tiempo.
- **Investigación de frameworks (spring, micronaut, quarkus):** Como los analizadores de texto necesitan conocer dependencias para ubicarse y saber la arquitectura de programa que leen, es necesario conocer cada uno de estos frameworks de java, puesto que todos tienen anotaciones distintas. Esta tarea implica crear un proyecto completamente funcional (con controladores, repositorios, bases de datos, etc.) con cada uno de estos frameworks para saber después cómo analizarlos genéricamente.
- **Investigación de bases de datos y buckets:** Esta tarea existe principalmente por los buckets, que son repositorios donde se guardan archivos binarios, como los pdfs que el sistema genera. La dificultad de esta investigación se encuentra en vincular la base de datos con el bucket gratis que se encuentre disponible. También es en esta tarea donde se investigan las formas de realizar almacenar copias de respaldo.
- **Investigación del lenguaje Java:** Esta es otra tarea que también ya estaba hecha y era prácticamente para el análisis de métricas, puesto que con esta se puede conocer las limitaciones que tiene el lenguaje frente a distintos inputs. Esta tarea consiste básicamente en leer documentación de java.
- **Investigación de generador automático de texto:** Está también ya estaba hecha y el motivo de que exista es básicamente porque es más fácil generar texto a partir del código después de ese texto generar el diagrama, que generar el diagrama desde el inicio. La herramienta implementada se llama StringTemplate y es realmente bastante simple de usar.
- **Investigación de herramientas de altos recursos:** Dado que todas las investigaciones anteriores estaban enfocadas principalmente en la búsqueda de soluciones libres de pago, todas cuentan con recursos limitados. Por lo que el objetivo de esta actividad es investigar herramientas que ofrecen mayores capacidades y recursos donde Junkode pueda funcionar si llegado el momento, las herramientas gratuitas le son insuficientes.

20.1.2.Etapa de Capacitación:

- **Capacitación uso ANTLR4:** Todas las tareas de capacitación consisten en hacer que quienes hasta ahora nunca han utilizado las herramientas se familiaricen con ellas. Siempre teniendo en cuenta que se ocupa tanto alguien que domina la herramienta como quienes aprenden. Específicamente en este caso no se ocupa a alguien para enseñar, puesto que Ramses ya había grabado unos videos de esta herramienta el año pasado, y quienes deban aprender solo deben verlos.
- **Capacitación en los frameworks previamente investigados:** Esta no es una tarea muy larga puesto que una vez que todo está funcionando es sencillo explicar los frameworks. Esto se debe a que los tres son muy similares.
- **Capacitación en contenedores:** El deployment del sistema será en contenedores, por lo que es fundamental que todos al menos conozcan los fundamentos de esta tecnología. La intención aquí no es hacer un tutorial, sino dar una explicación y un modelo simple, por eso dura tan poco.
- **Capacitación uso GitHub:** Para el trabajo utiliza un repositorio de GitHub y es importante que todos quienes vayan a utilizarlo sepan cómo hacerlo y así no se desate un caos como en muchos otros proyectos open source.
- **Capacitación en árboles semánticos y analizadores sintácticos:** Esto es básicamente una capacitación sobre la sintaxis de java, y es para poder armar analizadores de métricas más eficientes que los que ya se habían desarrollado en años anteriores.
- **Capacitación de buenas prácticas:** Esta no solo es una capacitación, sino que es incluso un debate sobre cuáles de las normativas investigadas serán las tenidas en cuenta.

20.1.3.Etapa de Diseño:

- **Diseño diagrama de clases:** Esta tarea tiene como objetivo definir cómo almacenaremos la metadata circundante al documento que el sistema genera.
- **Diseño modelo relacional:** Esta tarea junto a la anterior corresponden a la etapa de diseño y en ella deben respetarse todas las buenas prácticas propuestas en por la herramienta que se está desarrollando.
- **Diseño interfaces del sistema:** Estos serán desarrollados en VUE, o sea que serán interfaces web. A pesar de que generalmente esto lleva mucho tiempo (y en especial para el equipo dado que ninguno se especializa en front-end) las interfaces que el proyecto requiere son muy simples y no hay mucho que debatir respecto a ellas.
- **Definición de el BackBone:** Esta es la etapa del análisis en la que se deben definir las historias de usuario a realizar. Este es un trabajo que es más tedioso que complicado, por eso se le ha dedicado una semana entera.

20.1.4.Etapa de Desarrollo:

- **Codificación módulo de Usuario:** Se debe crear el módulo que gestione tanto la autenticación como los niveles de acceso a las distintas funcionalidades de los distintos usuarios.
- **Prueba del módulo Seguridad:** Se prueba intentando vulnerar la seguridad de SpringSecurity y se comprueban las funciones de seguridad ya implementadas tanto por Oauth como por GitHub.
- **Codificación módulo de análisis de métricas:** Si bien ya se contaba con algo de esto programado, hace falta mejorarlo, dándole ponderación a las métricas y agregando las nuevas normas que se definieron en las etapas anteriores.
- **Prueba módulo de análisis de métricas:** Se prueba ingresando múltiples proyectos de múltiples versiones de Java y con líneas de código específicas para probar las funcionalidades (condicionales anidados, herencias profundas, nombres no válidos, etc.).
- **Codificación módulo de análisis de documentación:** Si bien este módulo fue enseñado en clase con anterioridad, la verdad es que hay que cambiar muchas cosas. Para empezar, hay que generar texto de nuevo desde cero para el graficador, e incluir los nuevos frameworks y nuevas funcionalidades. Sin embargo, no deja de ser algo que ya se hizo una vez, y por eso solo se le dió 5 días para hacerlo.
- **Prueba del módulo de análisis de documentación:** Se prueba ingresando múltiples documentos con la mayor heterogeneidad posible tanto en frameworks, bases de datos, gestores de dependencias, servicios de infraestructura, etc.
- **Codificación módulo de Reportes:** Se debe crear el módulo que se encargará de juntar y enviar en el formato correspondiente al módulo web la información que consiguieron los módulos de análisis de métricas y de análisis de documentación.
- **Prueba módulo de Reportes:** Se prueba visualizar y crear los reportes, tanto a partir de un análisis como de metadata.
- **Codificación módulo Web:** Se debe crear el módulo que se encarga de gestionar las interfaces visuales, y el que se encarga de permitir que se suba código.
- **Prueba Módulo Web:** Se prueba que sean responsivas y no agresivas al usuario. También se testean los hipervínculos y los tokens que estos utilizan.
- **Configuración de los contenedores:** Definir como correrán, cuantas instancias, y cómo se crearán. Esto también se relaciona directamente con el deployment que se implemente.
- **Prueba de los contenedores:** Prueba del cluster conjunto, si los endpoints funcionan y si los pods se comunican correctamente.
- **Configuración base de datos:** Esto implica la creación de tablas y colecciones necesarias para el almacenamiento de los registros, junto con los triggers para hacer posible su uso. También incluye la implementación de copias de respaldo y su almacenamiento en la nube.
- **Prueba de la base de datos:** Más que probar la base de datos, se realizan pruebas a sus conexiones en esta tarea. En cuanto a los backups se prueba que la creación, almacenamiento y borrado de copias se haga de manera automática y correcta.
- **Integración de las herramientas:** Finalmente, hay que unir todo y asegurarse de que todo funcione. Puede notarse lo largo que puede ser este trabajo en el tiempo que se dispuso en el diagrama de gantt para realizar esta tarea.

20.1.5.Etapa de Testing:

- **Planificación del test de integración:** Definir el entorno de prueba y el orden en el que se probarán las conexiones entre las herramientas.
- **Testing de integración:** Hacer pruebas para asegurarse de que todo junto funcione correctamente.
- **Planificación del test de performance:** Se definen los aspectos a evaluar del desempeño del sistema (estrés, servicios caídos, recuperación, resiliencia, etc).
- **Test de performance:** Analizar rendimiento y estrés del sistema, esto es necesario debido a que se utilizarán herramientas gratis y estas generalmente no brindan muchos recursos para el sistema.

20.1.6.Etapa de Documentación:

- **Redacción de Manuales de Usuario:** Esto incluye redacción y exportación como documentación web.
- **Producción de Video-Tutoriales:** Las herramientas para desarrolladores a menudo tienen canales de youtube oficiales, por los que es más fácil transmitir la información por medio de videotutoriales. Estos sin lugar a duda toman más tiempo para hacer que los manuales.
- **Redacción de Guía de Inicio Rápido:** Esto incluye redacción y exportación como documentación web, incluyendo los videos-tutoriales.

20.1.7.Etapa de Despliegue:

- **Deployment en okteto:** Finalmente, la opción gratuita más útil de la que se dispone es deployar en okteto, el cual permite correr kubernetes de forma gratuita y otorga ips públicas donde publicar los endpoints del sistema.
- **Deployment de la base de datos:** Esta tarea consiste en la creación de una base de datos en el servidor remotemysql.com. Esta debe crearse atándose a los alcances máximos que ofrece el servidor en la nube para despliegues gratuitos y a las limitaciones de privilegios que el proveedor estipula.
- **Configuración del servicio de backups:** Esta tarea consiste en la configuración de Job automático de generación de Backups en el servicio Web SQLBak. En este se deben almacenar las credenciales de acceso a la Base de Datos, elegir la frecuencia con que se realizarán los Backups (quincenalmente), ingresar las credenciales del repositorio donde se almacenarán (Google Drive), seleccionar las opciones de encriptado pertinentes y definir la frecuencia con la que los Backups serán eliminados automáticamente.
- **Creación de las tablas de la Base de Datos:** Se ejecuta un script SQL que genera todas las tablas de la base de datos, incluyendo entidades, auditorías y usuarios. Este script ya fue generado automáticamente antes durante el desarrollo de la base de datos.
- **Creación de imagen de Graficador:** Durante esta tarea se desarrolla el servicio de API graficadora, el cual debe recibir un objeto JSON que incluya la sintaxis de PlantUML y entregárselo a un servicio graficador de PlantUML para que lo procese y

retorne una imagen con el modelo de clases. Para desarrollar esta tarea se necesita una imagen de docker que consiste en un servidor Tomcat que tiene corriendo el servicio graficador de PlantUML. Esta imagen es provista por los desarrolladores de PlantUML en su Docker Hub.

- **Creación de imagen de API app2yaml:** Similar a la tarea anterior, esta tarea consiste en desarrollar una API en Python que reciba un objeto JSON con texto en formato de application.properties y lo redirija al servicio conversor “app2yaml” del sitio web JavaInUse para que lo procese y retorne otro objeto JSON, pero esta vez con texto en formato yaml. En este caso, se consume directamente el servicio web de JavaInUse, por lo que no es necesario desplegar este servicio en un cluster propio, como si fue necesario para el servicio de PlantUML de la tarea anterior.
- **Despliegue del servicio PlantUML, API Graficadora y API app2yaml:** El servicio graficador (API graficadora y servicio de PlantUML) se despliega en su propio cluster de Okteto. Cada uno de estos servicios se despliega junto con sus respectivos “Services” y “ReplicaSets” de Kubernetes (resaltando que de cada uno de estos servicios se pueden crear hasta 3 réplicas) y se expone el puerto público al que este puede ser consultado. No hay necesidad de configurar un Gateway, Discovery Service ni Load Balancer porque se utiliza el Ingress Controller nativo de Okteto (que es básicamente un NGINX Gateway) para controlar los ingresos y sesiones de las réplicas de los servicios que se crearán cuando sea necesario.
- **Despliegue del servicio Junkode:** El servicio principal de Junkode es el que contiene todos los servicios no mencionados los anteriores pasos del deployment (entiéndase, análisis de código, ABMs de administrador, módulos del front-end, conexiones a los demás servicios y documentación). Este servicio se despliega en un cluster distinto al de los servicios desplegados anteriormente. Este servicio se despliega en conjunto con sus respectivos “Services” y “ReplicaSets” de Kubernetes y se expone el puerto público al que este puede ser consultado. Este servicio cuenta con hasta diez réplicas para ser creadas y orquestadas por el Ingress Controller nativo de Okteto.
- **Testing del Despliegue:** Se realiza un prueba rápida que consiste en entrar a todos los endpoints públicos generados y corroborar que todos se hayan conectado correctamente. Esta prueba se realiza por medio del mismo navegador, por MySQL-Workbench y Postman.
- **Configuración del dominio de red:** Por medio del portal CloudDNS se le asigna un nombre distintivo del sistema a los endpoints que deben ser accedidos por el público.
- **Implementación del certificado SSL:** Por medio del portal SSL for Free se obtiene un contrato de SSL para que el sistema se exponga bajo el protocolo https. La key de este contrato debe ser dockerizada y deployada como un elemento “Secret” de Kubernetes en el cluster donde se encuentra la base de datos.
- **Evaluación de la implementación resultante:** Una vez finalizado el proceso, se comparan los resultados obtenidos con los objetivos planteados inicialmente y se proponen tareas de rectificación y/o control que puedan ser necesarias.

20.1.8.Preparación:

- **Selección de sistema:** Primera tarea del proyecto, consiste en definir el sistema, empresa, idea o solución a desarrollar durante el proyecto.
- **Desarrollo de Trabajo Práctico Integrador "DIRECCIÓN DE PROYECTOS DE SISTEMAS":** La tarea consiste en responder las consignas dadas por la cátedra en un informe para que este sea entregado y evaluado.
- **Desarrollo de Trabajo Práctico Integrador "GERENCIAMIENTO DE SISTEMAS":** La tarea consiste en responder las consignas dadas por la cátedra en un informe para que este sea entregado y evaluado.
- **Preparación segundo demo:** Elaborar una presentación y una muestra en marcha del sistema desarrollado hasta el módulo de autenticación de usuarios.
- **Preparación tercer demo:** Elaborar una presentación y una muestra en marcha de la primera pre-release del sistema.
- **Hacer boceto de póster:** Diseñar un boceto sencillo del póster.
- **Diseñar poster:** Confeccionar el diseño definitivo que tendrá el póster del proyecto

20.1.9.Hitos:

- **Fecha límite para la confirmación o modificación de las guías de trabajo y selección de la Organización o Empresa y Sistema:** Elegir proyecto a realizar por el equipo de trabajo con fecha 22/03/2022.
- **Exposición de proyecto de Sistemas a desarrollar:** Exponer a los demás equipos de trabajo el proyecto a desarrollar con fecha 29/03/2022.
- **Etapa de Definición de Requerimientos:** Presentación de la etapa de requerimiento del sistema con fecha 26/04/2022.
- **Etapa de Diseño:** Presentación de la etapa del diseño del sistema con fecha 14/06/2022.
- **Inicio de diseño de papers para Congreso CONAIISI:** Desarrollo de papers para el Congreso CONAIISI con fecha 14/06/2022.
- **Demo de cada Sistema en aula para todo el curso:** Entrega de una demo para todo el curso del sistema con fecha 13/09/2022.
- **Primera revisión de cada póster para exposición:** Revisión de póster para exposición con fecha 27/09/2022.
- **Segunda revisión de cada póster para exposición:** Revisión de póster para exposición con fecha 11/10/2022.
- **Demo de cada Sistema y poster para exposición:** Entrega de una demo y poster para exposición del sistema con fecha 11/10/2022.
- **Etapa de Desarrollo e Implementación:** Presentación de la etapa de desarrollo e implementación del sistema con fecha 01/11/2022.
- **Demo de cada Sistema y ensayo de exposición:** Entrega de una demo y práctica de la exposición con fecha 08/11/2022.
- **16ª Exposición Anual de Proyectos de Sistemas:** Exposición anual de Proyectos de Sistemas con fecha 15/11/2022.
- **Trabajo Práctico Integrador "DIRECCIÓN DE PROYECTOS DE SISTEMAS":** Entrega del Trabajo Práctico Integrador N°1 con fecha 17/05/2022.
- **Trabajo Práctico Integrador "GERENCIAMIENTO DE SISTEMAS":** Entrega del Trabajo Práctico Integrador N°2 con fecha 16/08/2022.

20.2.Diagrama de tiempos.

Para ver el diagrama de tiempos del proyecto, consulte el Anexo n°1 “Diagrama de Gantt e Histogramas de Recursos”.

21.Organización para la ejecución del proyecto.

21.1.Equipo de trabajo (estructura, puestos, perfiles, cantidades).

El equipo de trabajo está estructurado de 8 puestos, los cuales son Coordinador, Desarrollador, Analista, Tester, Encargado de Documentación, Investigador, Quality Assurance y Capacitador. A continuación se describen los mismos:

Coordinador:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completado. Por lo menos 1 curso realizado en planificación de proyectos. 2 años de experiencia comprobable en algún puesto similar. Habilidades de liderazgo, motivación y comunicación. Ser organizado y responsable.
- Cantidad: Uno.

Desarrollador:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo. Conocimientos en los siguientes campos: paradigma orientado a objetos, lenguaje C#, Antlr4, programación reactiva. Nivel de Inglés intermedio. Disponibilidad para trabajos fuera de horario.
- Cantidad: Tres[2].

Analista:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo. Por lo menos 1 curso realizado en análisis de sistemas. Experiencia comprobable en análisis de sistemas. Conocimientos en metodología de desarrollo tradicional.
- Cantidad: Dos.

Tester:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo. Por lo menos 1 curso realizado de Testing. Experiencia con Test Unitarios de JUnit y Apache Jmeter. Nivel de Inglés intermedio.
- Cantidad: Uno.

[2] En las cantidades se cuenta tanto a quienes están encargados de esta actividad como quienes hacen tareas de soporte para la misma.

Encargado de la documentación:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo.Nivel de Inglés avanzado. Experiencia comprobable en el uso de herramientas de edición audiovisual. Habilidades de redacción para la documentación escrita, de edición y en hacer material audiovisual.
- Cantidad: Dos.

Investigador:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo.Nivel de Inglés avanzado Habilidades en búsqueda y documentación de información. Ser indagador y dedicado.
- Cantidad: Cinco.

Quality Assurance:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo. Nivel de Inglés avanzado. Habilidades para el diseño de presentación y entregables. Experiencia comprobable en el puesto de al menos 1(un) año. Debe ser prolijo y meticuloso.
- Cantidad: Uno.

Capacitador:

- Perfil: Trabajo remoto. Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio. Secundario completo.Nivel de inglés intermedio.Disponibilidad para trabajar fuera de horario. Conocimiento en paradigmas orientados a objetos, Java, Kubernetes, Github , SpringBoot, Vue y ANTLR. Habilidades para la comunicación y la transmisión de conocimiento.
- Cantidad: Dos.

21.2.Funciones principales de los miembros del equipo de trabajo.

Coordinador:

- Tareas: Será quien se encargue de la organización y gestión de los recursos del equipo. Entre sus tareas se encuentran: pautar reuniones, dividir trabajos y controlar entregables.
- Recursos: Ramsés, Girala.

Desarrollador:

- Tareas: Proponer y desarrollar soluciones de software para satisfacer las necesidades del proyecto. Básicamente se encarga implementar los módulos programables del sistema (los cuales en este caso específico componen la integridad del sistema) y luego coordinarlos de forma que ya no funcionen como módulos separados, sino como un sistema único.

- Recursos: Ramsés, Girala - Renzo, Fernandez.
- Soporte de: Rodrigo, Céspedes.

Analista:

- Tareas: Estudiar y describir por medio de modelos las funciones que debe incluir el sistema.
- Recursos: Rodrigo, Céspedes.
- Soporte de: David, Groisman.

Tester:

- Tareas: Realizar tests unitarios, de integración y de performance a los módulos y al proyecto ya integrado para asegurar el correcto funcionamiento del sistema.
- Recursos: Renzo, Fernandez.

Encargado de la documentación:

- Tareas: Con “documentación” se hace referencia a la documentación para los usuarios de la herramienta, y esta documentación estará dispuesta en forma escrita en manuales de usuario y en forma de videotutoriales, que son la forma más utilizada actualmente por las empresas desarrolladoras de software para desarrolladores de software. Además, se encargan de la traducción de la documentación del software.
- Recursos: Rodrigo, Céspedes.
- Soporte de: David, Groisman.

Investigador:

- Tareas: Hacer investigaciones de las herramientas, metodologías, procesos y herramientas necesarias para la realización del proyecto, y a su vez resumir, depurar y documentar esta información para que pueda ser dada a conocer a los demás integrantes del equipo.
- Recursos: Rodrigo, Céspedes - Sebastián, Flores - Renzo, Fernandez - David, Groisman - Ramsés, Girala.

Quality Assurance:

- Tareas: El control de calidad en este proyecto no se limita al código, sino que también incluye la documentación, investigación, presentaciones y entregables para la cátedra.
- Recursos: Sebastián, Flores.

Capacitador:

- Tareas: Compartir sus conocimientos de forma síncrona o asíncrona (por medio de videos pregrabados) con los demás miembros del equipo respecto de herramientas, metodologías o tecnologías.
- Recursos: Ramsés, Girala.
- Soporte de: Rodrigo, Céspedes.

21.3.Métodos de comunicación formal, control de avance, retroalimentación, decisiones.

21.3.1.Métodos de comunicación formal:

Para comunicar sobre reuniones, hacer consultas y organizar tareas de forma asincrónica se emplea Whatsapp; Para reuniones sincrónicas se utiliza un servidor de Discord (el cual es una herramienta de videollamadas con canales de texto, similar a Microsoft Teams), estas reuniones son para trabajar, aclarar dudas o hacer consensos; Y finalmente, se realizan reuniones presenciales en la universidad durante el horario de la materia “Proyecto Final” en la que se realizan las mismas actividades que en las reuniones de Discord y además se les realizan consultas a los profesores.

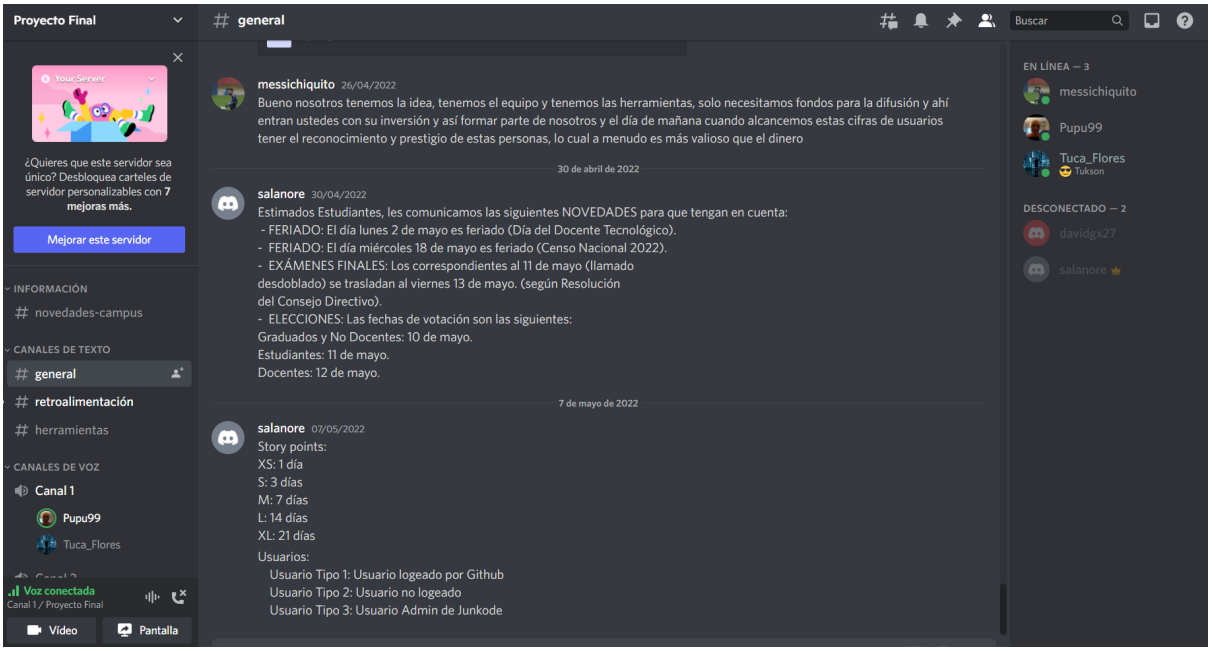


Figura 21.3.1.a: Captura del servidor de Discord.

En Discord, hay tres canales de voz para poder dividirnos tareas y poder cada uno trabajar en su propio canal y en caso de ser solicitado, se lo llama entrando a su canal. Además, tenemos un canal de texto general para poder enviar fotos, archivos, links, videos, etc. Otro canal de texto que se llama retroalimentación para correcciones o visado de tareas, y también en el canal herramientas para poder enviar información para el sistema en sí, como links de diseños de Front-End, las herramientas del requerimiento, etc. Y por último el canal de novedades, para dar noticias al equipo completo.

Figura 21.3.1.b: Captura del Ícono del Grupo de WhatsApp.

En el caso de WhatsApp, tenemos un grupo con los integrantes del equipo de trabajo. Esta herramienta la usamos para informar sobre reuniones, avisos al mismo equipo, notificación general por parte del coordinador, división de tareas, entre otras cosas.

21.3.2.Métodos de control de avance:

Para la organización de tareas se utiliza Trello, el cual es una herramienta en la que se ponen notas de tipo “post its”. Hay 3 tarjetas que son: tareas a realizar, tareas finalizadas y tareas en revisión. Y además, cada miembro del equipo tiene su propia tarjeta con las tareas designadas. Cada tarea tiene una etiqueta con el nombre de la etapa a la que hace referencia.

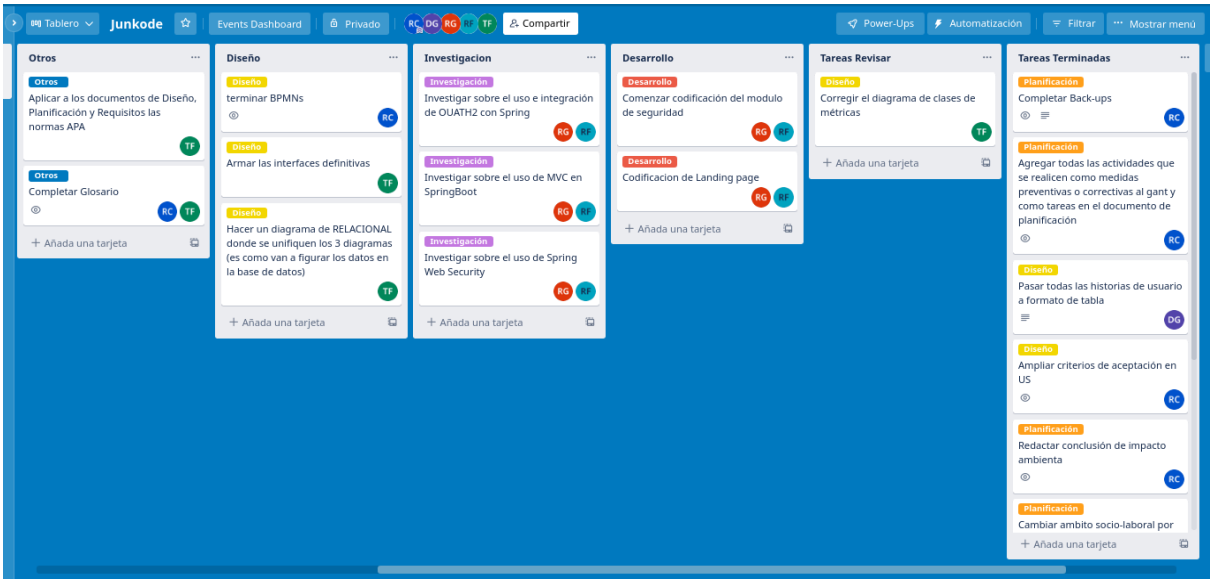


Figura 21.3.2.a: Trabajos asignados en Trello.

Cuando alguien termina una tarea, simplemente la quita de su pila y la pone en “revisar”, esto debe ser luego avisado por alguno los medios de comunicación y así podrá revisarse el trabajo realizado, y luego moverlo a “terminado”.

Mientras que se usa Trello para el control del avance de las actividades en el proyecto, se usa GITHUB como controlador de versionado y acceso para el desarrollo de software. Se detalla más información del repositorio de GITHUB en el siguiente punto.

21.3.3.Métodos de retroalimentación:

Respecto a la retroalimentación interna (o sea el feedback de un miembro del equipo a otro) se ejerce por medio del método de comunicación “Discord” donde se hace una reunión de 15 minutos , una por semana, en la cual se felicitan las buenas acciones y se corrigen aquellas donde no hubo un buen rendimiento. En cuanto a la retroalimentación externa (de usuarios no desarrolladores y docentes), es directa. Los docentes las dan durante las reuniones presenciales en la Universidad y los usuarios del sistema pueden darla a través de contactos que se encuentran a su disposición en el repositorio de GITHUB, e incluso pueden hacer forks del proyecto, hacer sus propios cambios y proponer sus propias pull requests.

21.3.4.Métodos de toma de decisiones:

Las decisiones son tomadas en una minuta, en la que se cita a todos los miembros del equipo y se debate. De forma en la que todos tomamos una decisión. Pero en casos particulares, no se abrirá la posibilidad de debate y el coordinador del grupo tomará las decisiones de manera autocrática. Ver Anexo N°2 “Registro de reuniones del Equipo”.

21.4. Gestión de Configuración del Software: Método de gestión de versionado durante todo el proyecto.

Se usa GITHUB como controlador de versionado y acceso para el desarrollo de software. Se cuenta con un repositorio público de código (aún vacío) con una rama propia para cada integrante del grupo y una rama principal donde se manejan las releases del proyecto. Cada miembro tiene acceso a todas las funciones que GITHUB otorga. Y es un repositorio público debido a la intención primera de que el proyecto se desenvuelva en un software open source.

Para nombrar las distintas versiones, se diferencia por un lado una versión que se lanza para el uso y por otro lado lo que es una versión de desarrollo que todavía no se lanza para el uso del público. Las versiones finales se nombran con un formato XX.YY donde XX es un número entero que indica el número de versión y YY es un número entero que indica el número de subversión(por ejemplo 1.3). Y para las versiones de desarrollo se adopta el formato XXmYYdZZ, donde XX representa el año que se lanzó, YY el mes y ZZ el día(por ejemplo 2022m5d14 sería la versión que se subió el 14 de Mayo del 2022).

Las versiones de los distintos archivos de documentación se van a nombrar como aa_V_bb, donde aa representa el nombre del archivo y bb el número de versión(Un ejemplo es Planificación_V_4). Con la excepción de las presentaciones, las cuales se guardan como cc_dd/ee/ff, siendo cc el nombre del día de la semana de la fecha de la presentación, dd ee

y ff los números del día,mes y año de esa fecha respectivamente(Por ejemplo Martes_26/04/2022).

Es importante diferenciar también entre el código fuente y la configuración como tal del software. Ambos son guardados en un repositorio público de GITHUB, pero la configuración se guarda en una rama privada. De igual manera, al momento de hacer el deploy, la configuración estará dentro de un config server respaldado por secrets de kubernetes.

22.Factibilidad.

22.1.Diagrama de recursos.

Para ver el diagrama de recursos del proyecto, consulte el Anexo n°1 “Diagrama de Gantt e Histogramas de Recursos”.

22.2.Análisis de factibilidad.

22.2.1.Factibilidad Técnica:

Lista de Necesidades:

- Volumen de Datos:

El servicio remotemysql.com ofrece hasta 100MB de almacenamiento gratis. Para las primeras releases del proyecto esto es más que suficiente, ya que solamente se guarda la metadata necesaria para generar el reporte.

De igual manera, en caso de superar ese límite, la información se almacenará en DynamoDB que es un servicio de AWS para bases de datos no relacionales, y se haría la muda de datos con Spoon. Sin embargo, incluso existe una posibilidad de correr un contenedor con una imagen de DynamoDB en Okteto, que sería una solución gratuita y permitiría hasta 5GB de almacenamiento.

Lo anterior fue respecto al volumen de datos máximo, y en cuanto al mínimo respecta, Junkode no necesita de información para ser factible, por lo que no existe un mínimo real de usuarios, y por ende tampoco un volumen de datos mínimo.

- Tipo de procesamiento de datos:

Se realizan tareas de carga, análisis, transmisión y persistencia de datos. Todas estas actividades son factibles por medio de: Librerías de integración de Front-end con Back-end para la carga, servicios con parsers y lexers para el análisis, librerías de manejo en formato JSON para la comunicación de los servicios y conexiones a bases de datos en la nube con para la persistencia de datos.

- Backups:

Se realizan backups automáticos cada cuatro semanas de toda la metadata de la base de datos (incluyendo la auditoría) por medio del servicio web SqlBak, el cual es gratuito si se trabaja con un solo Job (el cual es el caso de Junkode). La metadata se encripta, se

comprime, se pone como nombre del respaldo la fecha en la que se hizo seguida por el nombre de la base de datos y se guarda en Google Drive, en la nube. Cuando estos respaldos tienen más de dos meses de antigüedad, se eliminan. Todo este proceso ocurre de forma automática, sin embargo los administradores disponen también de la opción de hacerlo manualmente cuando así lo requieran desde su endpoint.

- Recuperación:

Esto se maneja desde el mismo SqlBak, desde este se pueden restaurar cualquiera de los respaldos realizados previamente. El proceso se hace a través de una interfaz de usuario amigable y que facilita la tarea de la importación de los datos.

- Integración con otros Sistemas

La principal integración que debe hacer el sistema es con GitHub, puesto que de ahí obtendrá sus usuarios, junto con sus repositorios. Esto se logra por medio del servicio Oauth, que permite que los usuarios ingresen a Junkode con las credenciales de GitHub.

Más allá de eso, todas las integraciones son sencillas, puesto que la mayoría de los servicios corren en un cluster de contenedores gratuito llamado Okteto. En caso de que surjan limitaciones por la cantidad límite de pods en Okteto, pueden desplegarse otros clusters, o hacerse un despliegue en Heroku o pagar por el servicio EKS de Amazon para gestionar contenedores sin limitaciones fijas. De todas las soluciones, esta última es la única que se paga.

- Crecimiento funcional y de TI para el Sistema

Al ser Open Source, el incremento de usuarios sería algo bueno para la herramienta más que algo malo, puesto que ayudarían a hacer crecer la comunidad y darían soporte a los problemas. También ayudan a la subsistencia de la herramienta, no solo con su uso, sino también con sus donaciones. En base a proyectos similares (Manjaro), se pudo estudiar qué aproximadamente 1 de cada 1000 usuarios dona mensualmente para mantener la herramienta, mientras que el dinero donado tiene un promedio de 6 USD y un techo de 100 USD. Esto indica que el proyecto no tendrá pérdidas monetarias en caso de que tengas que expandirse a herramientas de pago.

Respecto a la expansión en tecnologías, cómo se implementa el patrón de genéricos sobre las clases de ANTLRv4, no hay inconvenientes en agregar otros frameworks o herramientas para analizar. De igual forma, las comunidades en internet de ANTLRv4 facilitan sintaxis de otros lenguajes en sus repositorios, lo que facilita una posible expansión a otros lenguajes.

22.2.2.Factibilidad Operativa:

Lista de Necesidades:

- Personas:

Una particularidad de Junkode es que, por analizar lenguajes de programación y sus librerías, el análisis o la búsqueda de tokens puede quedar obsoleta muy rápidamente. Hay 2 tipos de personas que pueden evitar que esto ocurra: el equipo de desarrollo del proyecto

(dándole mantenimiento organizado en forma de parches) y la comunidad participando en foros, proponiendo soluciones y haciendo pull requests al repositorio de Junkode para actualizar funciones o solucionar bugs. De esta manera se puede apreciar que siempre deberá haber gente atenta al proyecto y que la formación de una comunidad de usuarios que se preocupe y ocupe del código sería enormemente beneficioso, pero no estrictamente necesario.

- Testing con usuarios

Los usuarios no solamente podrán incluir y proponer funcionalidades para el sistema, sino también probarlas. Esto se hace por el mismo medio, GitHub. Este permite tener ramas con distintas versiones clonables del proyecto que pueden usarse para llevar distintas versiones beta para usuarios. De esta forma, los usuarios pueden tener feedback directo no solo con los administradores de Junkode, sino con su equipo de desarrollo y código fuente.

- Resolución de Conflictos

En un sistema que se basa en sus usuarios para su mejora como Junkode, naturalmente surgirán conflictos relacionados con los merges que los usuarios realicen al repositorio. Si bien la idea es evitarlos limitando las capacidades que los usuarios colaboradores de GitHub tengan en el repositorio, si un conflicto debido a cambios generados por usuarios llegase a ocurrir en alguna realice de la rama principal, será el equipo de desarrollo el encargado de analizar el problema, proponer una solución y efectuarla (Pudiendo esta solución incluso ser un Rollback de la rama main).

- Diseño de campaña para involucrar a los usuarios.

Como se expresó en los puntos anteriores, el proyecto saca provecho de la comunidad activa que en él participe. Para motivar esta participación se planea tomar dos medidas: la primera es abrir un canal de YouTube de la herramienta, con tutoriales, ejemplos y noticias de lo que ocurra con esta, teniendo en cuenta que los videos deben existir en la mayor cantidad de idiomas posibles. La segunda medida sería motivar a la interacción de la comunidad en sí, creando foros para dudas y consultas sobre la herramienta, y participando activamente como desarrolladores en estos.

En lo que respecta al futuro próximo del proyecto se tratarán solo las dos medidas nombradas creación e interacción de la comunidad. Sin embargo, en un futuro más lejano es posible incluir una nueva medida, la fidelización. Esto sería copiando la estrategia de Kubernetes, creando “embajadores” voluntarios de la herramienta en distintas ciudades para que den a conocer las herramientas. A estos embajadores se les otorgarían claves de cursos gratis para que puedan repartirlas y también merchandising de la herramienta (Por más que parezca una estrategia rara, Kubernetes tiene hasta ropa de bebés con su logo en sus tiendas oficiales, así que la idea de simples remeras, stickers o camperas no es para nada extraordinaria).

22.2.3.Factibilidad Legal:

Hay dos aspectos importantes a resaltar en este punto:

El primero es la persistencia de la información de los usuarios registrados en Junkode. Los usuarios dan su consentimiento para que la información de sus proyectos sea almacenada al momento de ingresar con un usuario de GitHub por primera vez a Junkode.

El siguiente es un extracto del manual de usuario de Junkode que hace alusión a este consentimiento:

“Junkode tendrá acceso a recolectar y almacenar información sobre el código fuente de los proyectos de GitHub que el usuario persista en Junkode. Esto incluye, no exclusivamente, metadata de generación de diagramas, métricas del código, versiones y disposición de ramas de un repositorio de GitHub.

Junkode podrá continuar almacenando información sobre un repositorio de GitHub incluso si el repositorio o la cuenta dueña de este son eliminados. Esta información podrá de igual manera ser consultada y/o eliminada por el usuario de Junkode que la generó cuando así lo disponga.

Los datos serán almacenados sólo luego de un proceso de serialización y encriptación para proteger la seguridad de los mismos.”

El segundo aspecto legal a mencionar es que el consentimiento del usuario para que Junkode acceda a sus repositorios ya se encuentra estipulado por GitHub en el momento de ingresar al sistema desde Oauth2. La interfaz que GitHub provee en ese momento contiene sus propios términos y condiciones.

22.2.4.Factibilidad Económica:

Los costos que tiene este proyecto son de horas humanas, es decir que la inversión que se realiza es únicamente de tiempo. Y los beneficios no son monetarios, son de reconocimiento y satisfacción para el equipo, y de automatización y ahorro de tiempo para los usuarios.

Si se tratara de un equipo pago de 5 desarrolladores (basados en salarios de glassdoor) los costos del proyecto serían de la forma:

Factibilidad Económica (suponiendo que se le debe pagar a los desarrolladores)								
	Jun	Jul	Ago	Sep	Oct	Nov	Dec	**TOTAL**
Costos (Pesos):								
Sueldos	-450.000	-450.000	-450.000	-450.000	-450.000	-450.000	-450.000	-3.150.000
Beneficios (Pesos):								
	----	----	----	----	----	----	----	----
BALANCE TOTAL								-3.150.000

Tabla 22.2.4.a: Tabla de Factibilidad Económica.

O sea, si el proyecto no contase con mano de obra gratuita sería rotundamente no factible económicamente.

Como se verá a continuación, el proyecto es susceptible de que surjan ingresos y egresos, sin embargo, dado que lucrar no es el propósito del proyecto, y que los egresos son potencialmente nulos, es que en el aspecto económico son considerados insignificantes. Por lo tanto, el proyecto es factible del punto de vista económico.

22.2.5.Factibilidad Financiera:

Junkode es un proyecto de software libre y open-source, por lo que no busca realizar ninguna actividad con propósitos lucrativos, solo acepta donaciones. Los usuarios que donan son (en base a comparaciones con otros softwares libres) 1 de cada 1000, con una media de donaciones de 6 dólares y un techo 100.

En cuanto a los egresos respecta, como ya se ha dicho múltiples veces, el sistema utiliza únicamente herramientas de uso gratuito a no ser que haya una demanda excesiva de los servicios que obligue a Junkode a contratar servicios en la nube.

De esta manera, así quedaría el balance de ingresos y egresos si se superan las condiciones de umbral (Detalles en el siguiente apartado).

Factibilidad Financiera (suponiendo que se superaron todas las condiciones de umbral)													
Meses	1	2	3	4	5	6	7	8	9	10	11	12	**TOTAL**
Ingresos (USD):													
Donaciones	0	1	0	0	0	0	0	0	5	0	0	0	6
Egresos (USD):													
Costos de AWS	-1,5	0	0	-1,5	0	0	-1,5	0	-1,5	0	0	-1,5	-7,5
BALANCE SUBTOTAL													-1,5

Tabla 22.2.5.a: Tabla de Factibilidad Financiera.

Estos valores fueron calculados aleatoriamente en base al método Montecarlo y como se puede apreciar, en un año habría una pérdida de 1,50 dólares. Teniendo en cuenta que la cantidad de consultas de la herramienta fue estimada apuntando a un caso de sobrecarga (cada vez que se cobran 1,5 dólares es que hubo un millón de lecturas y escrituras a la base de datos) y que perder una cantidad menor a 10 dólares en un año es algo que el equipo está dispuesto a aceptar, entonces se concluye que el proyecto es financieramente factible.

22.3.Costos desagregados por recursos (personal, tecnología) con periodicidad mensual.

Es difícil hablar de costos en este proyecto debido a que solo existen costos potenciales. O sea, que solo sería necesario que el proyecto incurra en costos si su demanda fuese demasiado grande y/o fluctuante (o sea, con picos y valles de demanda de servicios).

En estos casos habría que contratar servicios on-Cloud para dar soporte a las funcionalidades demandadas (Afortunadamente el proyecto está pensado para correr sobre entornos de Kubernetes, por lo que hacer el cambio de que el sistema pase de correr sobre servicios locales a correr en servidores en la nube no implicaría mayores complicaciones).

Se ha elegido a AWS (Amazon Web Service) como proveedor de servicios on-Cloud. Esto se decidió fundamentalmente en base a los precios que ofrece frente a la competencia (Google Cloud y Microsoft Azure), debido que tanto los servicios ofrecidos como su calidad eran prácticamente iguales.

El primer servicio que sería necesario en caso de una sobrecarga de demandas serían los Buckets, donde se almacenan los pdfs con los reportes.

El precio de estos se mide por Gigabytes ocupados por mes, y para valores que estén dentro de los 50TB, cada gigabyte cuesta 0,023 USD al mes.

Teniendo en cuenta que el límite a ser superado para contratar este servicio sería la generación de 10 pdfs diarios, y que los pdfs generados son de aproximadamente 40MB, la cuenta sería:

10 pdfs al día x 30 días x 0,04 GB = 12 GB nuevos por mes.

12 GB nuevos por mes x 0,023 USD = 0,276 USD por mes.

Esto significa que cada mes (si se mantiene ese ritmo) habría que pagar 0,276 USD más de almacenamiento, y tomaría más de 347 años superar los 50TB, por lo que nunca se entraría dentro de otro plan.

El segundo servicio de AWS que se debería contratar es AWS EKS (Elastic Kubernetes Service) que es lo que reemplazaría a Okteto y pasaría a gestionar el cluster de Kubernetes de Junkode. El precio de este servicio es de 0,10 USD por hora por cluster.

En el caso de Junkode sólo sería necesario un cluster, por lo que el precio mensual ascendería a:

0,10 USD cada hora x 24 hs x 30 días = 72 USD por mes.

Finalmente se encuentra el servicio de DynamoDB, la base de datos no relacional que sería utilizada para la metadata y sustituiría a Mongo Atlas.

El simple estimar el precio de este servicio, puesto que si bien cobra tanto por almacenamiento como por cantidad de consultas, las consultas se cuentan por millón (es

decir, se paga un precio fijo por cada millón de consultas que reciba la base de datos) y se diferencian entre consultas de lectura o escritura (un millón de consultas de lectura vale 0,25 USD y un millón de consultas de escritura vale 1,25 USD), mientras que el almacenamiento es gratis hasta los 25GB.

La condición para contratar este servicio sería que haya más de 5GB de metadata almacenados y ya no se pueda utilizar Mongo Atlas de forma gratuita (Es importante aclarar que pagar por utilizar Mongo Atlas sería más costoso que cambiarse a AWS DynamoDB, la mudanza de la base de datos se haría con Spoon).

De esta forma el costo mensual (como no se superarían el millón de lecturas ni el millón de lecturas en un mes) sería de solo 1,50 USD, y podría ser aún menos, puesto que solo habrá otro recargo cuando se superen el millón de consultas por separado, o sea si con las consultas del siguiente mes tampoco se llega al millón de consultas, este no se paga y prácticamente se pagarían 1,50 USD por dos meses, y así el tercero y sucesivamente. Sin embargo se dejará como necesario el pago de 1,50 USD mensuales para el cálculo.

También es importante aclarar en este punto que no se tiene en cuenta el almacenamiento que ocupan las copias de respaldo debido a que estas se almacenan en cuentas de google drive que tienen hasta el triple de capacidad gratuita de almacenamiento, por lo que sí las bases de datos alcanzarán su máximo de 5GB gratuitos, el backup aun podría guardar hasta 3 copias de respaldo de la base de datos completa.

En resumen los costos que se plantearían mensualmente serían:

Insumo	Precio Mensual (USD)	Disp.
AWS Bucket s3 (EEUU Ohio) 50TB	0,276 x cantidad de meses en uso	&
AWS EKS (un solo cluster)	72	&
AWS DynamoDB service On-Demand	1,5	&
TOTAL	73,776	

Tabla 22.3.a: Resumen de los costos que se plantearían mensualmente.

22.4.Análisis de riesgos.

Para estudiar los riesgos del sistema, primero es necesario conocer sus potenciales amenazas, puesto que son estas de las que pueden generar los riesgos para este.

Para estudiar las amenazas se tuvieron en cuenta dos dimensiones de estas: su probabilidad de ocurrencia y el impacto que podría tener sobre el sistema (ambas dimensiones medidas en escala de 1 a 5, donde 5 implica que la amenaza es más probable o de mayor impacto). El producto de estos dos indicadores da como resultado el riesgo de la amenaza.

Luego para cada amenaza se plantearon medidas para reducir estos factores, cada medida puede reducir la probabilidad de ocurrencia, el impacto, o ambos, y con los nuevos valores de los indicadores, se vuelve a calcular el riesgo de la misma forma, dando como resultado un número menor o igual al riesgo antes de las medidas.

La clasificación del riesgo sigue un código de color, si es menor a 7 le corresponde el color verde, si es mayor o igual a 7 pero menor de 15, amarillo y si es mayor o igual a 15, rojo.

El resultado del estudio de riesgo puede apreciarse en la Tabla de riesgos.

Para ver la tabla de riesgos del proyecto, consulte el Anexo n°3: Tabla de riesgos.

22.5.Análisis de impacto ambiental.

Cada ámbito susceptible de ser afectado por Junkode se encuentra nombrado a continuación, y el impacto que se proyecta en ellos se encuentra clasificado de acuerdo a las siguientes categorías.

- Signo (+, - o neutro): Determina si mejora, degrada o es inocuo para el ambiente.
- Magnitud (Alto, Medio o Baja): La magnitud del impacto califica la dimensión o tamaño del cambio producido en el ambiente.
- Alcance (Global, Local o Restringido): Si el sistema afectará a todos los elementos del ambiente, a unos particulares o si será cerrado para una minoría.
- Persistencia (Alta, Media o Baja): Baja si dura menos de 1 año; media si dura de 1 a 3 años y alta si dura de 4 a diez años.

Ámbito Directivo
En este ambiente la herramienta permite a los directivos a cargo de los proyectos controlar las métricas que manejan sus desarrolladores subordinados, para controlar así que estos no dañen el diseño de un sistema. Por otro lado, este nivel empresarial también puede verse beneficiado por el ahorro de tiempo que la herramienta supone.

- Signo: Es positivo ya que permite un mejor desempeño en la supervisión de los subordinados al estar controlando cómo se va desempeñando el desarrollo del proyecto.
- Magnitud: Posee una magnitud alta al momento de evaluar la consistencia del sistema
- Alcance: El alcance es local ya que se establece sobre los directivos que trabajan en el mismo proyecto.
- Persistencia: Alta ya que si el uso de la herramienta comienza en las primeras instancias del proyecto la duración del impacto prevalece toda la vida del proyecto.

Ámbito Operacional
En este ámbito la herramienta permite la generación de documentación y provocará un ahorro de tiempo que, en función del tamaño del proyecto, puede ser de semanas para el mantenimiento de proyectos. Esta funcionalidad no se ha desarrollado hasta el momento,

por lo cual ayudará a la hora de la producción de informes y el modelado del diagrama de clases del proyecto.

A diferencia de lo que se podría inferir, Junkode no tiene un impacto alternativo como sustitución de trabajadores destinados a tareas de mantenimiento, por el contrario, seguirá siendo necesaria gente que de mantenimiento a software y Junkode es solo una herramienta para facilitar su trabajo.

- Signo: Es positiva ya que simplifica el trabajo para los desarrolladores que mantienen el código.
- Magnitud: La magnitud es alta puesto que permite el desarrollo con metodologías ágiles solucionando la generación de documentación.
- Alcance: El alcance es local ya que la generación de documentación es para un proyecto.
- Persistencia: Baja, debido a que la funcionalidad sirve para generar documentación rápida cuando no la hay. Para periodos largos de mantenimiento es más conveniente hacer ingeniería inversa a fondo sobre el proyecto y que el equipo de desarrolladores genere su propia documentación.

Ámbito Social

El impacto de la herramienta es muy amplio, ya que se encuentra libre para el público. Es decir que toda persona o empresa puede utilizarla de forma gratuita y también acceder al código fuente para revisarlo y bifurcarse para su conveniencia. Cualquier persona puede mejorar las funcionalidades y compartirlas con el resto de personas que utilicen esta herramienta.

- Signo: Es positivo puesto que se brinda una nueva herramienta para ayudar a los desarrolladores de forma gratuita.
- Magnitud: La magnitud es alta porque es una funcionalidad que hasta el momento no existe como servicio.
- Alcance: El alcance es global, ya que el uso de la herramienta no está atado a ninguna restricción para su uso.
- Persistencia: Es alta puesto que no hay planes de dejar de brindar servicios.

Ámbito Ambiental

Junkode no tiene un impacto negativo en medio ambiente porque los recursos que de los que precisa son muy austeros en cuanto a consumo, capacidad de procesamiento, y almacenamiento. Tampoco tendría un impacto positivo, puesto que el proceso que se mejora tampoco tenía un impacto negativo notable sobre el medio ambiente.

- Signo: Neutro.
- Magnitud: Bajo no producirá ningún cambio en el ambiente.
- Alcance: Restringido. Solo está limitada para el uso en pc lo cual no afecta el ambiente.
- Persistencia: Ninguno.

Factibilidad Ambiental

Ambiente	Signo	Magnitud	Alcance	Persistencia
Ámbito Directivo	+	Alta	Local	Alta
Ámbito Operacional	+	Media	Local	Baja
Ámbito Social	+	Alta	Global	Alta
Ámbito Ambiental	Neutro	Baja	Restringida	Ninguna

Tabla 22.5.a: Tabla de Factibilidad Ambiental.

De esta forma, se puede concluir que la etapa del ciclo de vida del proyecto donde Junkode puede brindar un mayor valor agregado es en la durante la implementación y el mantenimiento.

Sin embargo, y como puede apreciarse en la tabla, el potencial de Junkode no está en ser una herramienta de uso empresarial, sino en su uso por la comunidad mundial de desarrolladores que pueden requerir de su servicio. Puesto que es en este ámbito donde se ve mayor magnitud, alcance y persistencia juntos.

Conclusiones:

La propuesta de Junkode resulta interesante cuanto menos. A través de lo expuesto a lo largo de este documento puede apreciarse no solo la importancia y el impacto con el que este proyecto podría repercutir, sino también sobre su robustez, flexibilidad y orientación hacia a la comunidad.

Junkode aborda la problemática actual de la escasez de documentación técnica de proyectos de software con una solución tan asertiva como práctica. La generación automática de documentación le permite a los desarrolladores Java seguir practicando sus metodologías ágiles sin el temor de que podrían encontrarse sin documentos técnicos actualizados en el futuro.

El resultado superó las expectativas inicialmente planteadas y consecuencia de esto es que Junkode se encuentre ya desplegado con tres funcionalidades incrementales ya incorporadas. Junkode fue desarrollado y desplegado teniendo en cuenta la heterogeneidad dentro del desarrollo de código Java y los niveles de cambio constantes que manejan las librerías y dependencias actuales. Esta es una de las principales causas de los resultados satisfactorios que tienen los usuarios de la herramienta.

Pues todo esto, a su vez, resulta en una posible mejora de la calidad de desarrollo y mantenimiento de código fuente Java y en una ayuda a la comunidad global de desarrolladores de software (posibilidad ligada al uso y asimilación de la herramienta dentro del “mercado de software de desarrollo”). Sin lugar a duda, la utilidad esperada de la realización de este proyecto fue alcanzada en términos de disponibilidad y usabilidad, y superada en términos de funcionalidades ofrecidas a los usuarios y potencial de impacto.

Por otro lado, del enfoque de utilizar únicamente herramientas Open-Source se vio beneficiado el equipo del Proyecto (cuyos integrantes no tuvieron que pagar nada para desplegar una aplicación de altos estándares), pero mucho mas beneficiados se vieron los desarrolladores, que a lo largo de este trabajo aprendieron a utilizar más de una veintena de herramientas que, antes de realizar el proyecto, ni siquiera concebían que pudieran ser accedidas de manera gratuita. El conocimiento de estas herramientas y tecnologías, y el know-how de cómo utilizarlas y aprovecharlas, es probablemente el resultado más valioso que se llevan los desarrolladores del Proyecto Final.

Junkode es un proyecto que no cierra las puertas a ninguna idea, y con ese enfoque fue diseñado, con una flexibilidad tanto operativa como estructural que le permite no solo adaptarse a los cambios del entorno, sino que sus usuarios puedan adaptarse cómodamente a él.

Con esto último, Junkode no plantea revolucionar los procesos de desarrollo de software, pero no porque sus fundamentos carezcan de peso, sino porque simplemente no es su objetivo. Su objetivo es ofrecer un conjunto de utilidades a los desarrolladores para facilitar su trabajo y de esta manera, eventualmente sentar un precedente como herramienta de documentación de código Open-Source, libre y gratuita. Este último, si bien ese no es un objetivo explícito del Proyecto, si es una aspiración a futuro y una contribución a la comunidad de desarrolladores Open-Source, que tanta ayuda le brindó al Equipo del Proyecto Junkode.

Finalmente, Junkode es un proyecto que presenta todo lo necesario para ser mantenible en el tiempo, puesto que el principal factor que esto requiere, es el factor humano, y los desarrolladores de Junkode se demuestran a sí mismos, a través del sistema funcionando y de las páginas de este documento, como un equipo por sobre todas las cosas, profesional.

Retomando un párrafo anterior, se resalta que el proyecto no solo alcanzó los objetivos inicialmente planteados, sino que también se excedió a estos tanto en el trabajo finalmente realizado (con el desarrollo de tres módulos incrementales), como en su previsión a futuro. Se concluye de esta manera que Junkode resultó ser: un éxito como proyecto, un precedente como herramienta de documentación, y un orgullo para el equipo de estudiantes que lo desarrolló.

TRABAJOS PRÁCTICOS INTEGRADORES

A lo largo del desarrollo del Proyecto Final, tuvieron lugar dos Trabajos Prácticos Integradores que debían ser resueltos de manera grupal por el Equipo del Proyecto. Consulte el Anexo n°11 TRABAJO PRÁCTICO INTEGRADOR - “DIRECCIÓN DE PROYECTOS DE SISTEMAS ” y el Anexo n°12 TRABAJO PRÁCTICO INTEGRADOR - “GERENCIAMIENTO DE SISTEMAS” para encontrar los Trabajos Prácticos Integradores.

Glosario:

.

.NET: Framework de C# que permite la creación y ejecución de servicios web y aplicaciones de Internet.

.xml: (eXtensible Markup Language) Metalenguaje que permite definir el desarrollo de paginas web.

A

Acoplamiento: Grado de dependencia entre los componentes o clases entre sí. El acoplamiento es la medida que define qué tanto un componente o clase dependen de otro, generando cambios externos o alterando la funcionalidad del mismo. Se habla de acoplamiento bajo cuando existe una independencia entre los componentes entre sí, por el contrario un alto acoplamiento es cuando se tiene varias dependencias relacionadas a un solo componente.

ANTLRv4: (ANother Tool for Language Recognition; en español "otra herramienta para reconocimiento de lenguajes") es una herramienta que opera sobre lenguajes, proporcionando un marco para construir reconocedores (parsers), intérpretes, compiladores y traductores de lenguajes a partir de las descripciones gramaticales de los mismos.

API: Application Programming Interface (Interfaz de Programación de Aplicaciones). Es un programa que consiste en únicamente la comunicación en un formato de dto (JSON, XML, etc).

Árbol de herencia: Diagrama que plasma la relación generalidad y especificidad de las clases de un sistema, siendo la superclase la raíz del árbol y las subclases, las hojas.

AOP: Aspect-Priented Programming (Programación orientada a aspectos). Paradigma de programación que permite una adecuada modularización de las aplicaciones y posibilita una mejor separación de responsabilidades.

B

Backup: Copia de respaldo donde se almacena una imagen actual del estado de un sistema. En el caso de Junkode, de las tablas de la base de datos.

Back-End: Back-End es la parte o rama del desarrollo web encargada de que toda la lógica de una página funcione. Consiste en el conjunto de acciones que pasan dentro de una web, pero que no se puede ver por el usuario.

Bucket: Repositorio de archivos binarios (imágenes, documentos, archivos de audio, etc.) etiquetados. En el caso de que Junkode escale, estos podrían utilizarse para

almacenar las imágenes de los diagramas y/o los documentos en formato .pdf de la documentación.

Burndown charts: Herramienta utilizada en el desarrollo ágil de software, que relaciona el trabajo pendiente con un tiempo determinado. Se utilizan para estimar la probabilidad de que su equipo complete su trabajo en el tiempo disponible.

Blacklist: Lista de usuarios que se encuentran con acceso denegado a un sistema. En el caso de Junkode, son los usuarios privados de la funcionalidad del módulo de mensajería interna para prevenir spam y ataques de estrés de consultas.

BPMN: Business Process Model and Notation (Modelo y Notación de Procesos de Negocio). Es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo. Junkode cuenta con diagramas de sus procesos estandarizados en notación BPMN 2.0.

C

C#: Lenguaje de programación, basado en objetos y fuertemente tipado sobre el que está escrito el código fuente de Junkode.

CD: Despliegue continuo. Es una metodología de deployment de software que utiliza pruebas automatizadas para validar que todos los cambios en una base de código sean precisos y estén listos para implementarse de forma autónoma en un entorno de producción. En este proyecto se trabaja junto con CI para hacer análisis de ellos.

CI: Integración contínua. Es una metodología de trabajo muy utilizada en AGILE en la que se monitorea un repositorio de código a la espera de cambios para implementarlos de manera automática. En este proyecto se lo trabaja junto con CD para hacer análisis de ellos.

Cohesión: Es el grado en el que una función, componente o clase realiza únicamente la tarea para la cual fue diseñada. Se habla de cohesión alta cuando la relación la relación entre una clase y una funcionalidad es unívoca. Por el contrario, se habla de una cohesión baja cuando existe relación con múltiples funcionalidades del sistema.

Complejidad ciclomática: Métrica que define el número de caminos independientes dentro de un fragmento de código para proporcionar una medición cuantitativa de la complejidad lógica de un programa.

Contenedor: Entorno de virtualización similar a una máquina virtual pero completamente simplificado. En este proyecto la palabra “contenedor” y la palabra “docker” se utilizan indistintamente.

D

Dashboard: Tablero con un conjunto de métricas e indicadores que facilitan la aprehensión de los datos a quién los observa.

Deploy: Conjunto de actividades de despliegue que permiten que un sistema de software esté disponible para su uso.

Discovery: Servicio de infraestructura utilizado para encontrar instancias de servicios en otros sockets. Junkode no implementa un servicio de discovery, pero si los analiza.

Docker: Herramienta para crear y gestionar contenedores. En este proyecto la palabra “contenedor” y la palabra “docker” se utilizan indistintamente.

Dominio Web: Nombre exclusivo y único que se le asigna a un socket para hacerlo más fácil de memorizar y escribir con la finalidad de que cualquier usuario de internet pueda identificarlo y consultarlo.

Donorbox: Plataforma de financiación online que permite la creación de campañas de recaudación con total transparencia y brinda las herramientas necesarias para la integración a las páginas web.

DynamoDB: Es una base de datos NoSQL de clave-valor sin servidor y completamente administrada que está diseñada para ejecutar aplicaciones de alto rendimiento a cualquier escala. Su relevancia para el proyecto es la velocidad de respuesta para consultas a tablas no normalizadas, lo que puede ser de provecho para Junkode en futuras expansiones.

E

Endpoint: URL al que se le pueden realizar consultas. Estos son documentados en Junkode, para saber los identificadores con los que cuentan los controladores del proyecto analizado.

Enterprise Architect: Herramienta de modelado de diagramas UML, BPM, GANTT, etc. Utilizada para la representación de los diagramas de clases y entidades en la documentación de Junkode.

F

Fallback: El término "fallback" hace alusión a una opción de contingencia por la cual optar si la opción primaria no está disponible. Es decir, es una opción alternativa que se seguirá solo si la opción principal falla.

Framework: Entorno de trabajo que facilita librerías y dependencias para un proyecto de programación

Front-end: Componentes y elementos de un sistema que son visibles y accesibles para los usuarios.

G

Gateway: Servicio de infraestructura utilizado para dirigir peticiones entre distintos endpoints y comunicar servicios corriendo en distintos sockets.

Grafana: Herramienta utilizada para generar tableros de comando y dashboards de la información almacenada en bases de datos.

J

Java 8: Lenguaje de programación, basado en objetos y fuertemente tipado.

Jenkins: Herramienta de integración continua que permite probar updates de proyectos automáticamente para asegurar una correcta integración. Esta es estudiada por el analizador de documentación.

K

Kubernetes: Herramienta para gestionar conjuntos (clusters) de contenedores. En estos está basado el despliegue de Junkode, específicamente en el cluster online gratuito “Okteto”.

M

Metadata: Conjunto de datos que describen el contenido informativo de un recurso o de información de los mismos.

Metodologías ágiles: Conjunto de métodos, técnicas y herramientas de gestión de proyectos de software focalizada en la flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias volátiles del entorno.

Métodos: Porción de código que realiza tareas asociadas a un objeto.

Métrica: Indicador que se utiliza para determinar el estado, relación o condición de un hecho. Utilizado en este proyecto tanto para medir calidad de código como el desempeño del sistema.

Micronaut: Framework de Java estudiado por el analizador de documentación, y que está enfocado en evitar la reflexión, reduciendo así el consumo de memoria y mejorando los tiempos de inicio.

Mongo Atlas: Base de datos basada en la nube y completamente administrada de MongoDB que combina modelos de datos similares a JSON, indexación y búsqueda avanzadas, y escalabilidad elástica.

N

Namespace: un namespace o espacio de nombres es un medio para organizar clases dentro de un entorno, agrupándolas de un modo más lógico y jerárquico.

O

Oauth: Servicio de autenticación para usuarios de servicios terceros. Este es el servicio que permite que a los usuarios de Junkode poder acceder a sus repositorios.

Okteto: Herramienta gratuita para desplegar clusters (entornos) de Kubernetes con acceso público global.

P

Parser: Programa que analiza sintácticamente una cadena de caracteres según las reglas preestablecidas de una gramática formal.

Patrón visitor: Patrón del lenguaje UML que permite ejecutar operaciones sobre un grupo con diferentes clases, haciendo que un objeto visitante implemente distintas variantes de la misma operación que correspondan a todas las clases de objetivo.

PlantUML: Herramienta de código abierto que permite a los usuarios crear diagramas de UML a partir de texto. Esta es utilizada por Junkode para la generación de los diagramas de entidades y de propiedades.

Proceso unificado: Metodología de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y consiste en la transformación lineal de requisitos de los usuarios en unidades de valor de software.

R

Repositorio: Almacén remoto de código fuente. Para el proyecto, es contenedor del código java del usuario para ser analizado. Específicamente, el repositorio remoto de código que Junkode consulta es GitHub.

S

Spammers: Persona o programa que envía mucha correspondencia electrónica no solicitada.

Spring Boot: Framework de Java estudiado por el analizador de documentación.

SSL: Certificado de página segura. Es un protocolo de seguridad que da credibilidad a un sitio web, certificando que realmente lo ha gestionado el dueño del dominio.

U

UML: (Lenguaje de Modelado Unificado) Conjunto de reglas y estándares para representar la arquitectura física de un sistema informático.

US: User Story o Historia de Usuario es una unidad mínima de funcionalidad para el cliente, y son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos.

V

VUE.js: Es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página.

Y

YAML: (o YML) Es un lenguaje de declaración de datos que facilita la legibilidad y la capacidad de escritura del usuario. Este formato de serialización de datos se utiliza para redactar archivos de configuración y se puede usar junto con todos los lenguajes de programación.

Índice de Pantallas y Figuras.

Bibliografía y Referencias Bibliográficas.

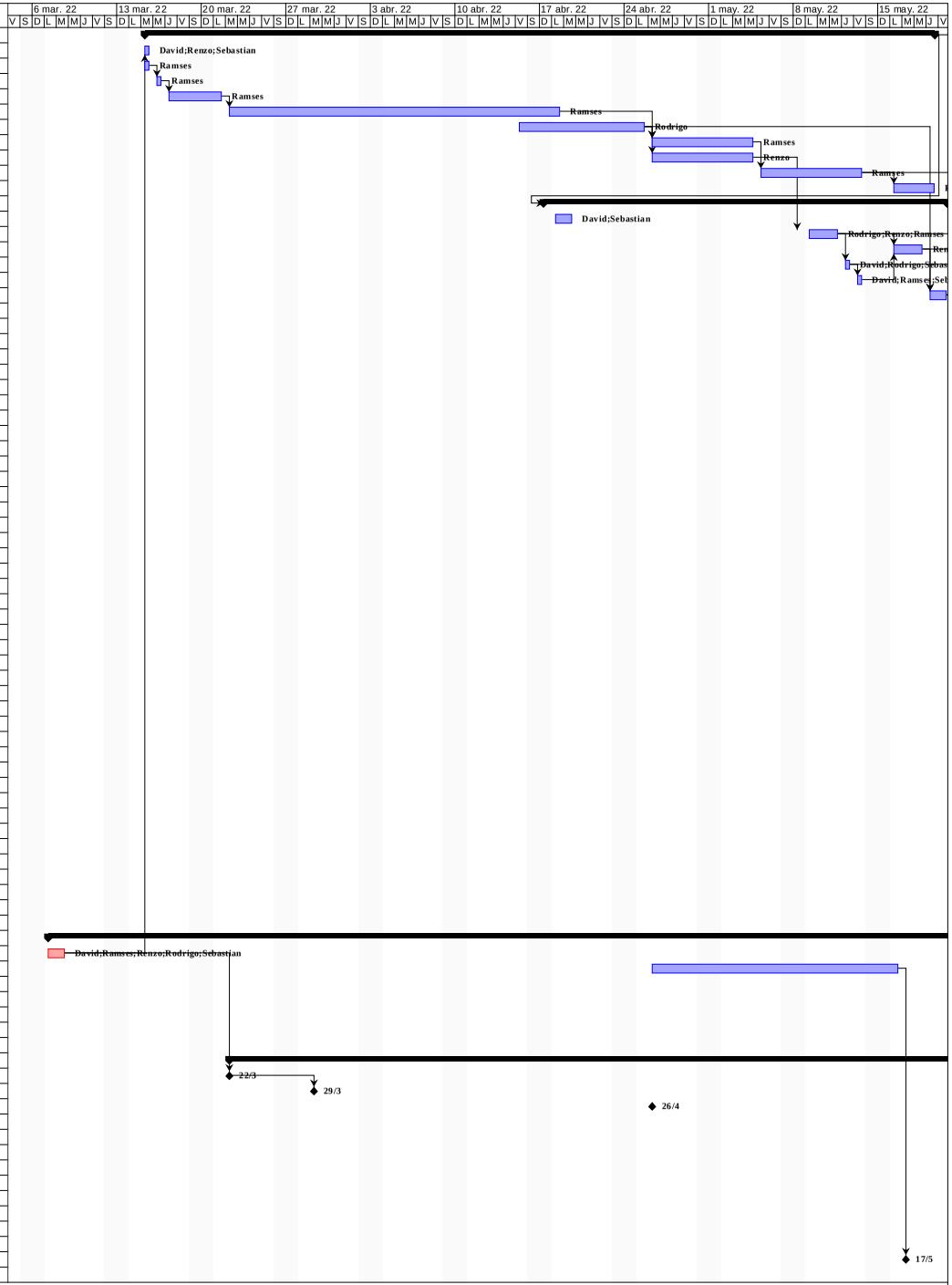
¿Qué es Kiuwan? Analiza la seguridad y calidad de tu software. (s.f.). Obtenido de https://sentrio.io/blog/que-es-kiuwan/

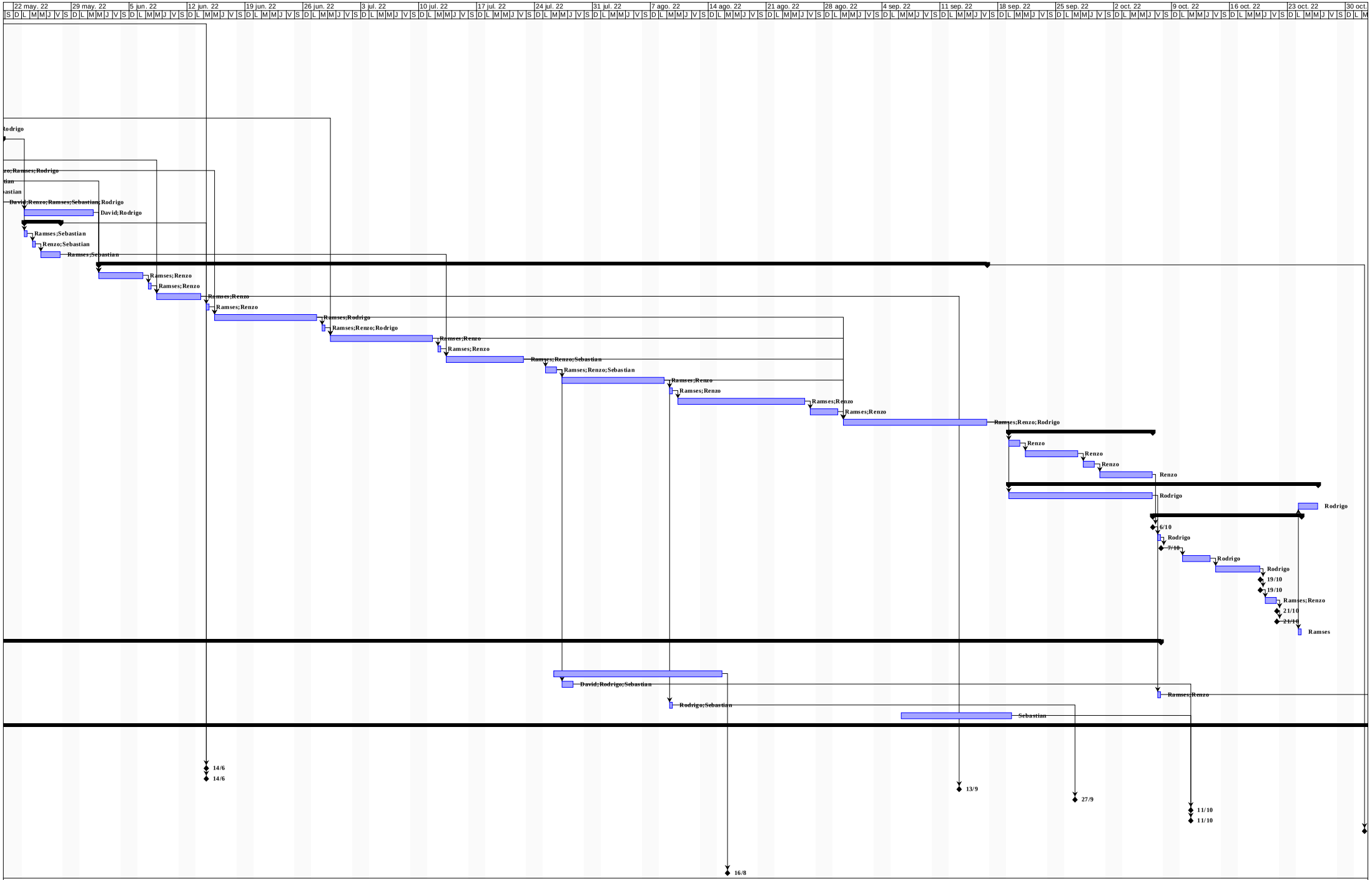
Code Quality and Code Security | SonarQube. (s.f.). Obtenido de https://www.sonarqube.org/
CONTROLA LA CALIDAD DE TU CÓDIGO CON NDEPEND. (s.f.). Obtenido de https://albertcapdevila.net/ndepend/
Descubre las vulnerabilidades de tus aplicaciones con VERACODE. (s.f.). Obtenido de https://acktib.com/veracode/
uarez, C. L.-G. (s.f.). Cómo realizar análisis de código con Veracode e integrarlo con IntelliJ. Obtenido de https://enmilocalfunciona.io/como-realizar-analisis-de-codigo-con-veracode-e-integrarlo-con-intellij/
Kiuwan An Idea Inc. Company. (s.f.). Obtenido de https://www.kiuwan.com/
Oficial, S. (s.f.). DeepScan. Obtenido de https://deepscan.io
Qué es SonarQube: Verifica y analiza la calidad de tu código. (s.f.). Obtenido de https://sentrio.io/blog/que-es-sonarqube/
Scanning Source Code Using Veracode for VS Code. (s.f.). Obtenido de https://www.youtube.com/watch?v=halOpSJa5kM

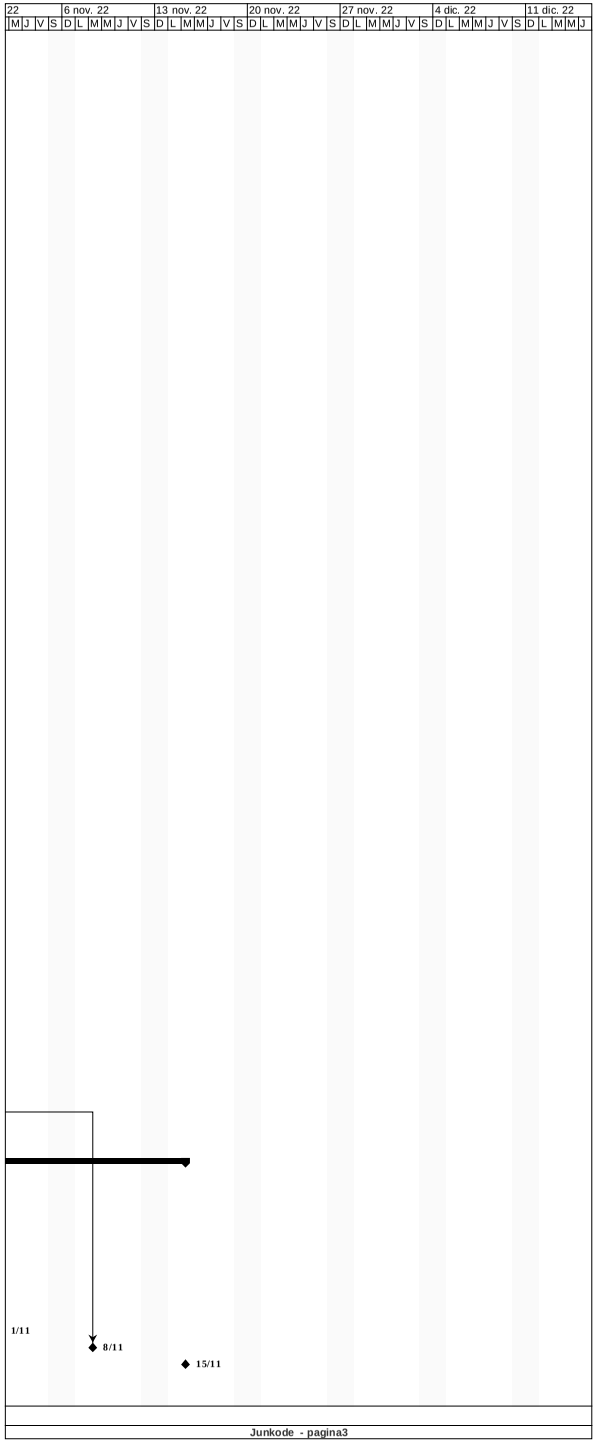
ANEXO N° 1 - DIAGRAMA GANTT E HISTOGRAMAS DE RECURSOS DEL PROYECTO

JUNKODE - Documentador Automático de Código Fuente

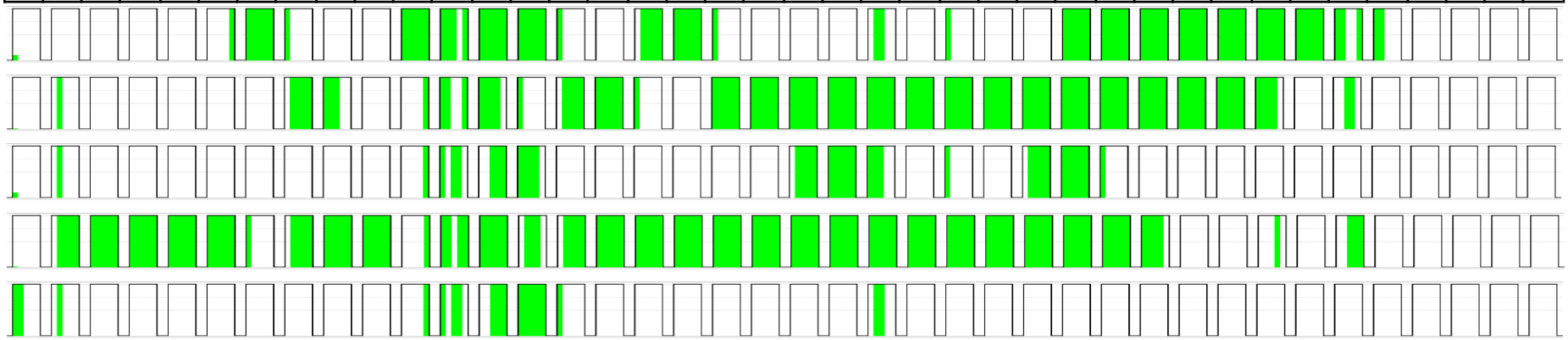
		Nombre	Duracion	Inicio	Terminado	Predecesores	Nombres del Recurso
1		**Investigación**	**48 days**	**15/3/22 08:00**	**19/5/22 17:00**		
2		Investigación de herramientas similares	1 day	15/3/22 08:00	15/3/22 17:00	61	David;Renzo;Sebastian
3		Investigación de herramientas de análisis de texto	1 day	15/3/22 08:00	15/3/22 17:00		Ramses
4		Investigación de generador automático de texto	1 day	16/3/22 08:00	16/3/22 17:00	3	Ramses
5		Investigación del lenguaje Java	3 days	17/3/22 08:00	21/3/22 17:00	4	Ramses
6		Investigación de herramientas de graficación	20 days	22/3/22 08:00	18/4/22 17:00	5	Ramses
7		Investigación de buenas prácticas de codificación	7 days	15/4/22 08:00	25/4/22 17:00		Rodrigo
8		Investigación de herramientas para deployment del sistema	7 days	26/4/22 08:00	4/5/22 17:00	6;7	Ramses
9		Investigación de frameworks (spring, micronaut, quarkus)	7 days	26/4/22 08:00	4/5/22 17:00	7	Renzo
10		Investigación de bases de datos y buckets	7 days	5/5/22 08:00	13/5/22 17:00	8	Ramses
11		Investigación de herramientas de altos recursos	4 days	16/5/22 08:00	19/5/22 17:00	10	Rodrigo
12		**Capacitación**	**25 days**	**17/4/22 08:00**	**20/5/22 17:00**	**1**	
13		Capacitacion uso ANTLR4	2 days	17/4/22 08:00	19/4/22 17:00		David;Sebastian
14		Capacitación en los frameworks previamente investigados	3 days	8/5/22 08:00	11/5/22 17:00	9	Rodrigo;Renzo;Ramses
15		Capacitación en contenedores	3 days	16/5/22 08:00	18/5/22 17:00	14;17	Renzo;Ramses;Rodrigo
16		Capacitación uso GITHUB	1 day	12/5/22 08:00	12/5/22 17:00	14	David;Rodrigo;Sebastian
17		Capacitación en árboles semánticos y analizadores sintácticos	1 day	13/5/22 08:00	13/5/22 17:00	16	David;Ramses;Sebastian
18		Capacitación de buenas prácticas	2 days	19/5/22 08:00	20/5/22 17:00	7;15	David;Renzo;Ramses;Sebastian;Rodrigo
19		Definición de el BackBone	7 days	23/5/22 08:00	31/5/22 17:00	12	David;Rodrigo
20		**Diseño**	**5 days**	**23/5/22 08:00**	**27/5/22 17:00**		
21		Diseño diagrama de clases	1 day	23/5/22 08:00	23/5/22 17:00	18	Ramses;Sebastian
22		Diseño modelo relacional	1 day	24/5/22 08:00	24/5/22 17:00	21	Renzo;Sebastian
23		Diseño interfaces del sistema	3 days	25/5/22 08:00	27/5/22 17:00	22	Ramses;Sebastian
24		**Implementación**	**78 days**	**1/6/22 08:00**	**16/9/22 17:00**		
25		Codificación módulo analizador de métricas	4 days	1/6/22 08:00	6/6/22 17:00	16;18;19	Ramses;Renzo
26		Prueba del módulo analizador de métricas	1 day	7/6/22 08:00	7/6/22 17:00	25	Ramses;Renzo
27		Codificación módulo documentador de código	4 days	8/6/22 08:00	13/6/22 17:00	14;26	Ramses;Renzo
28		Prueba del módulo documentador de código	1 day	14/6/22 08:00	14/6/22 17:00	27	Ramses;Renzo
29		Configuración de los contenedores	9 days	15/6/22 08:00	27/6/22 17:00	15;28	Ramses;Rodrigo
30		Prueba de los contenedores	1 day	28/6/22 08:00	28/6/22 17:00	29	Ramses;Renzo;Rodrigo
31		Configuración base de datos	9 days	29/6/22 08:00	11/7/22 17:00	10;30	Ramses;Renzo
32		Prueba de la base de datos	1 day	12/7/22 08:00	12/7/22 17:00	31	Ramses;Renzo
33		Codificación módulos interfaces Web	8 days	13/7/22 08:00	22/7/22 17:00	23;32	Ramses;Renzo;Sebastian
34		Prueba módulos interfaces Web	2 days	25/7/22 08:00	26/7/22 17:00	33	Ramses;Renzo;Sebastian
35		Codificación módulo de usuarios	9 days	27/7/22 08:00	8/8/22 17:00	34	Ramses;Renzo
36		Prueba del módulo de usuarios	1 day	9/8/22 08:00	9/8/22 17:00	35	Ramses;Renzo
37		Codificación módulo de historiales y estadísticas	12 days	10/8/22 08:00	25/8/22 17:00	36	Ramses;Renzo
38		Prueba del módulo de historiales y estadísticas	2 days	26/8/22 08:00	29/8/22 17:00	37	Ramses;Renzo
39		Integración de las herramientas	14 days	30/8/22 08:00	16/9/22 17:00	29;31;33;35;38	Ramses;Renzo;Rodrigo
40		**Testing**	**14 days**	**19/9/22 08:00**	**6/10/22 17:00**		
41		Planificación del test de integración	2 days	19/9/22 08:00	20/9/22 17:00	39	Renzo
42		Testing de integración	5 days	21/9/22 08:00	27/9/22 17:00	41	Renzo
43		Planificación del test de performance	2 days	28/9/22 08:00	29/9/22 17:00	42	Renzo
44		Test de performance	5 days	30/9/22 08:00	6/10/22 17:00	43	Renzo
45		**Documentación**	**28 days**	**19/9/22 08:00**	**26/10/22 17:00**		
46		Redacción de Manuales de Usuario	14 days	19/9/22 08:00	6/10/22 17:00	39	Rodrigo
47		Producción de las Guías de Inicio Rápido y Video-Tutoriales	3 days	24/10/22 08:00	26/10/22 17:00	58	Rodrigo
48		**Despliegue**	**12 days**	**6/10/22 17:00**	**24/10/22 17:00**		
49		Despliegue de la base de datos en la nube	0 days	6/10/22 17:00	6/10/22 17:00	44	Rodrigo
50		Configuración del servicio de BackUps	1 day	7/10/22 08:00	7/10/22 17:00	49	Rodrigo
51		Creación de Las tablas de la Base de Datos	0 days	7/10/22 17:00	7/10/22 17:00	50	Rodrigo
52		Creación de imagen de Graficador	4 days	10/10/22 08:00	13/10/22 17:00	51	Rodrigo
53		Creacion de imagen de API app2yaml	4 days	14/10/22 08:00	19/10/22 17:00	52	Rodrigo
54		Despliegue del servicio PlantUML, API Graficadora y API app2yaml	0 days	19/10/22 17:00	19/10/22 17:00	53	Rodrigo
55		Despliegue del servicio Junkode	0 days	19/10/22 17:00	19/10/22 17:00	54	Ramses
56		Testing del Despliegue	2 days	20/10/22 08:00	21/10/22 17:00	55	Ramses;Renzo
57		Configurar dominio de red	0 days	21/10/22 17:00	21/10/22 17:00	56	Rodrigo
58		Implementar certificación SSL	0 days	21/10/22 17:00	21/10/22 17:00	57	Ramses;Rodrigo
59		Evaluación de la implementación resultante	1 day	24/10/22 08:00	24/10/22 17:00	58	Ramses
60		**Preparación**	**155 days**	**7/3/22 08:00**	**7/10/22 17:00**		
61		Selección de sistema	2 days	7/3/22 08:00	8/3/22 17:00		David;Ramses;Renzo;Rodrigo;Sebastian
62		Desarrollo de Trabajo Práctico Integrador "DIRECCIÓN DE PROYECTOS DE SISTEMAS"	15 days	26/4/22 08:00	16/5/22 17:00		
63		Desarrollo de Trabajo Práctico Integrador "GERENCIAMIENTO DE SISTEMAS"	15 days	26/7/22 08:00	15/8/22 17:00		
64		Preparación segundo demo	2 days	27/7/22 08:00	28/7/22 17:00	34	David;Rodrigo;Sebastian
65		Preparación tercer demo	1 day	7/10/22 08:00	7/10/22 17:00	46	Ramses;Renzo
66		Hacer boceto de poster	1 day	9/8/22 08:00	9/8/22 17:00	35	Rodrigo;Sebastian
67		Diseñar poster	10 days	6/9/22 08:00	19/9/22 17:00		Sebastian
68		**Hitos**	**170 days**	**22/3/22 08:00**	**15/11/22 08:00**		
69		Fecha límite para la selección de la Organización o Empresa y Sistema	0 days	22/3/22 08:00	22/3/22 08:00	61	
70		Exposición de proyecto de Sistemas a desarrollar	0 days	29/3/22 08:00	29/3/22 08:00	69	
71		Inicio de etapa de Definición de Requerimientos	0 days	26/4/22 08:00	26/4/22 08:00		
72		Etapa de Diseño	0 days	14/6/22 08:00	14/6/22 08:00	20	
73		Inicio de diseño de papers para Congreso CONAIISI	0 days	14/6/22 08:00	14/6/22 08:00	1	
74		Demo de cada Sistema en aula para todo el curso	0 days	13/9/22 08:00	13/9/22 08:00	27	
75		Primera revisión de cada póster para exposición	0 days	27/9/22 08:00	27/9/22 08:00	66	
76		Segunda revisión de cada póster para exposición	0 days	11/10/22 08:00	11/10/22 08:00	67	
77		Demo de cada Sistema y poster para exposición	0 days	11/10/22 08:00	11/10/22 08:00	64	
78		Etapa de Desarrollo e Implementación	0 days	1/11/22 08:00	1/11/22 08:00	24	
79		Demo de cada Sistema y ensayo de exposición	0 days	8/11/22 08:00	8/11/22 08:00	65	
80		16ª Exposición Anual de Proyectos de Sistemas	0 days	15/11/22 08:00	15/11/22 08:00		
81		Trabajo Práctico Integrador "DIRECCIÓN DE PROYECTOS DE SISTEMAS"	0 days	17/5/22 08:00	17/5/22 08:00	62	
82		Trabajo Práctico Integrador "GERENCIAMIENTO DE SISTEMAS"	0 days	16/8/22 08:00	16/8/22 08:00	63	




22 may. 22
29 may. 22
5 jun. 22
12 jun. 22
19 jun. 22
26 jun. 22
3 jul. 22
10 jul. 22
17 jul. 22
24 jul. 22
31 jul. 22
7 ago. 22
14 ago. 22
21 ago. 22
28 ago. 22
4 sep. 22
11 sep. 22
18 sep. 22
25 sep. 22
2 oct. 22
9 oct. 22
16 oct. 22
23 oct. 22
30 oct.
Rodrigo
David;Renzo;Ramses;Sebastian;Rodrigo
David;Rodrigo
Ramses;Sebastian
Renzo;Sebastian
Ramses;Sebastian
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo
Ramses;Rodrigo
Ramses;Renzo;Rodrigo
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo;Sebastian
Ramses;Renzo;Sebastian
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo
Ramses;Renzo;Rodrigo
Renzo
Renzo
Renzo
Renzo
Rodrigo
Rodrigo
6/10
Rodrigo
7/10
Rodrigo
Rodrigo
19/10
19/10
Ramses;Renzo
21/10
21/10
Ramses
David;Rodrigo;Sebastian
Ramses;Renzo
Rodrigo;Sebastian
Sebastian
14/6
14/6
13/9
27/9
11/10
11/10
16/8


22
6 nov. 22
13 nov. 22
20 nov. 22
27 nov. 22
4 dic. 22
11 dic. 22
M J V S D L M M J V S D L M M J V S D L M M J V S D L M M J V S D L M M J
1/11
8/11
15/11
Junkode - pagina3

6 3 22	13 3 22	20 3 22	27 3 22	3 4 22	10 4 22	17 4 22	24 4 22	1 5 22	8 5 22	15 5 22	22 5 22	29 5 22	5 6 22	12 6 22	19 6 22	26 6 22	3 7 22	10 7 22	17 7 22	24 7 22	31 7 22	7 8 22	14 8 22	21 8 22	28 8 22	4 9 22	11 9 22	18 9 22	25 9 22	2 10 22	9 10 22	16 10 22	23 10 22	30 10 22	6 11 22	13 11 22	20 11 22	27 11 22	4 12 22



ANEXO N° 2 - REGISTRO DE REUNIONES MODELO DEL EQUIPO

JUNKODE - Documentador Automático de Código Fuente

Fecha	Tiempo Transcurrido	Asistentes	Objetivos	Tareas Realizadas	Tareas a Realizar	Documentación	Responsable	Personal o Remota
20/6/2022	3 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián Groisman,David	Realizar las correcciones de requisitos	Correcciones Anteriores	Agregar Anexos a informe de Requerimientos	Requerimentos_V_4	Girala, Ramsés	Remota
21/5/2022	1 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián Groisman,David	Avanzar informe de diseño	Diagrama de clase de graficador y de analizador de métricas	Seguir desarrollando las historias de usuario	Diseño_V_1 Diagrama_Clase Historias_Usuario	Girala, Ramsés	Remota
23/5/2022	2 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián Groisman,David	Realizar las correcciones de planificación	Correcciones anteriores	Agregar anexos a la planificación	Planificación_V_3	Girala, Ramsés	Remota
9/7/2022	2 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián	Analizar retrospectivamente los Resultados del Primer Semestre	Resaltado de los puntos mas importantes de cada etapa del semestre	Actualizar tareas de Trello con las actividades del receso invernal	Planificación_V_5	Cespedes Rodrigo	Remota
12/11/2022	2 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián Groisman,David	Ensayo para la Exposición de Sistemas	Ensayar, cronometrar duraciones y corregir el guión	Prolongar las explicaciones de desarrollo para llenar el tiempo	Carpeta Final del Proyecto	Cespedes Rodrigo	Remota
13/11/2022	2 hs	Girala, Ramsés Céspedes, Rodrigo Fernández, Renzo Flores, Sebastián Groisman,David	Ensayo para la Exposición de Sistemas	Ensayar, cronometrar duraciones y corregir el guión	Añadir nuevo tema de UML a la presentación para prolongar su duración	Carpeta Final del Proyecto	Cespedes Rodrigo	Remota

ANEXO N° 3 - TABLA DE RIESGOS

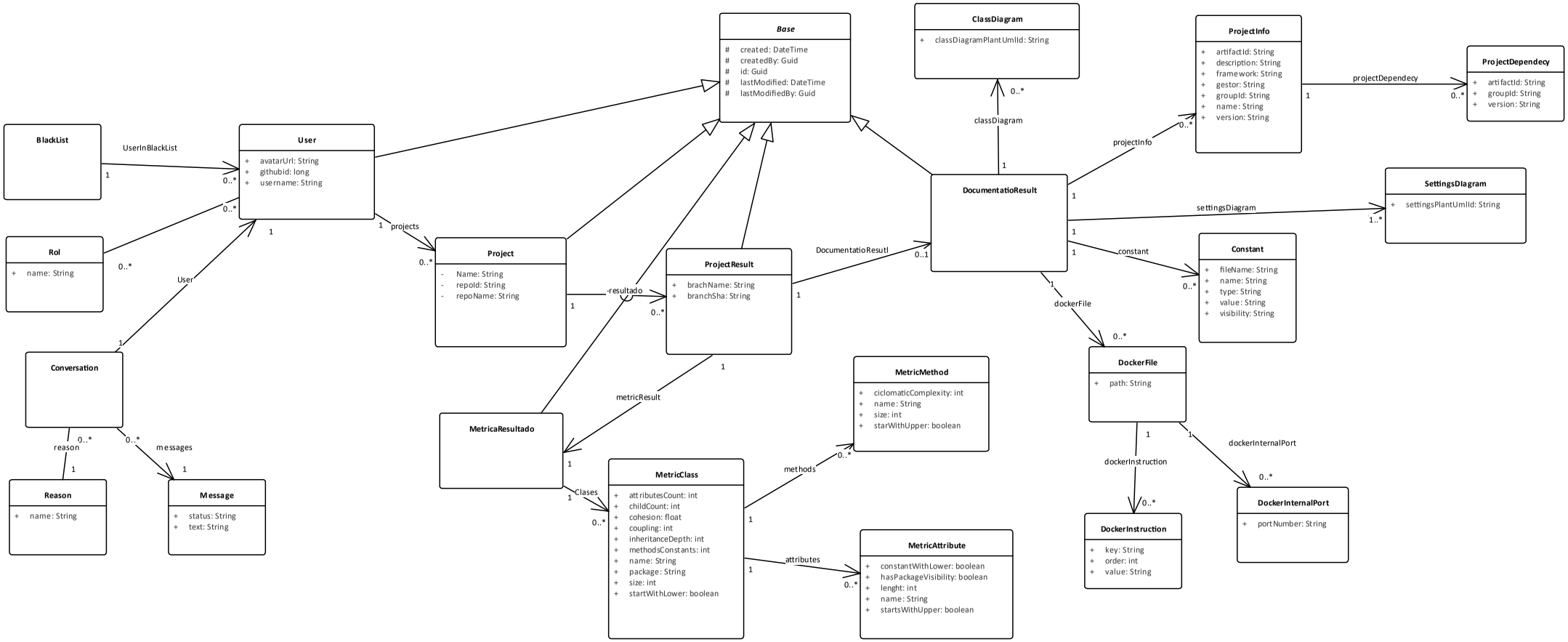
JUNKODE - Documentador Automático de Código Fuente

Anexo n°3 Tabla de Riesgos

Amenaza	Descripción	Probabilidad antes de la medida	Impacto antes de la medida	Riesgo antes de la medida	Medidas Preventivas	Medidas correctivas	Probabilidad después de la medida	Impacto despues de la medida	Riesgo después de la medida
Caida de servicio de deploy	El servicio de deploy del sistema, puede ser que se caiga no permita acceder al sistema	1	4	4		Se tendrá un plan de contingencia en caso de que se caiga, deployándose en otro servicio de deploy (disminuye el impacto)	1	2	2
Caída de servicio auxiliares	Los servicios que utiliza el sistema puede que se caigan por algún motivo ajeno al sistema, dejandolo fuera de servicio	1	3	3			1	3	3
Malintención de la comunidad	Los proyectos open-source comunmente reciben contribuciones de la comunidad de desarrolladores que los utilizan donde resuelven bugs que hayan encontrado. Pero esto puede ser un arma de doble filo, puesto que desarrolladores malintencionados inserten codigo perjudicial al repositorio del proyecto	2	4	8	Control de los pull-request antes de incorporarlos al repositorio (disminuye la probabilidad)	Dedicar autoridades sobre las ramas del proyecto y no sobre el proyecto en si (disminuye el impacto)	1	3	3
Deserción de integrantes del proyecto	Un integrante del grupo , abandona el proyecto,lo cual provoca que retrasen los tiempos y ocurran diversos problemas	1	3	3			1	3	3
Obsolescencia de los frameworks de Spring, Micronaut, Quarkus	Los frameworks estudiados por el sistema y sobre los que puede tomar decisiones e inferir resultados entán en constante cambio, con anotaciones que puede modificarse y/o quedarse obsoletas. La obsolesencia de estas herramientas implicaría la obsolecensia de Junkode	3	3	9	Estar al dia en cuanto al uso de frameworks y estar siempre un paso adelantado a los cambios (disminuye la probabilidad)		1	3	3
Inactividad de la comunidad	Este amenza trata de que la comunidad deja de usar el sistema, produciendo que dejen de haber foros , es decir, "deja de haber soporte tecnico"	3	2	6		Tener documentacion sumamente completa y detallada (disminuye el impacto)	3	1	3
Ciberataques	Los ciberataques pueden dar de baja el servicio inundandolo de peticiones, o incluso encontrando vulnerabilidades en los sistemas de datos	2	2	4			2	2	4
Cambios drásticos en la sintaxis Java	Puede ocurrir que salga una nueva versión de java con grandes cambios estructurales, dejando obsoletas a las anotaciones, lo cual desemboca en grandes complicaciones en el analizador	3	4	12	Estar al dia en cuanto a los cambios de las actualizaciones de Java y estar siempre un paso adelantado a los cambios (disminuye la probabilidad)		1	4	4
Limitación debido a los recursos	Al usar solo herramientas open source, provoca una limitacion de los programas y funcionalidades que le podemos dar a esa herramienta	2	3	6			2	3	6
Actualizacion de las herramientas de desarrollo	Se actualiza la version de una herramienta, lo cual vuelve incompatibles a las otros programas usados por esa herramienta	3	4	12	Correr las herramientas en entornos de contenedores como docker (disminuye la probabilidad)		1	4	4
Cambios en las herramientas de desarrollo	Este riesgo es distinto al anterior, ya que este trata sobre cambios en la herramientas de desarrollo. Dichos cambios pueden ser que las herramientas dejen de ser gratuitas y pasen a ser pagas o que dejen de tener soporte.	3	2	6		Estar al dia en cuanto a los cambios de las herramientas y asi en caso de que ocurra algun cambio en las mismas que nos afecten, buscar alguna otra herramienta para el reemplazo de esta (disminuye el impacto)	3	1	3
Incremento drástico de consultas	El sistema está implementado sobre servicios libres, por lo que los recursos con los que cuenta son límitados y la concorrencia de muchos usuarios consultando al sistema podría inhabilitar el servicio, ya sea solicitando mas instancias de servicios que los nodos de los que dispone okteto o llenando las bases de datos o los buckets	4	4	16	Optimizar al máximo los recursos en Okteto con multiples cuentas y almacenar mas metadata que documentos por proyecto (disminuye la probabilidad)	Planificar y disponer de alternativas con mayores rescursos listas para el despliegue del sistema (disminuye el impacto)	2	3	6

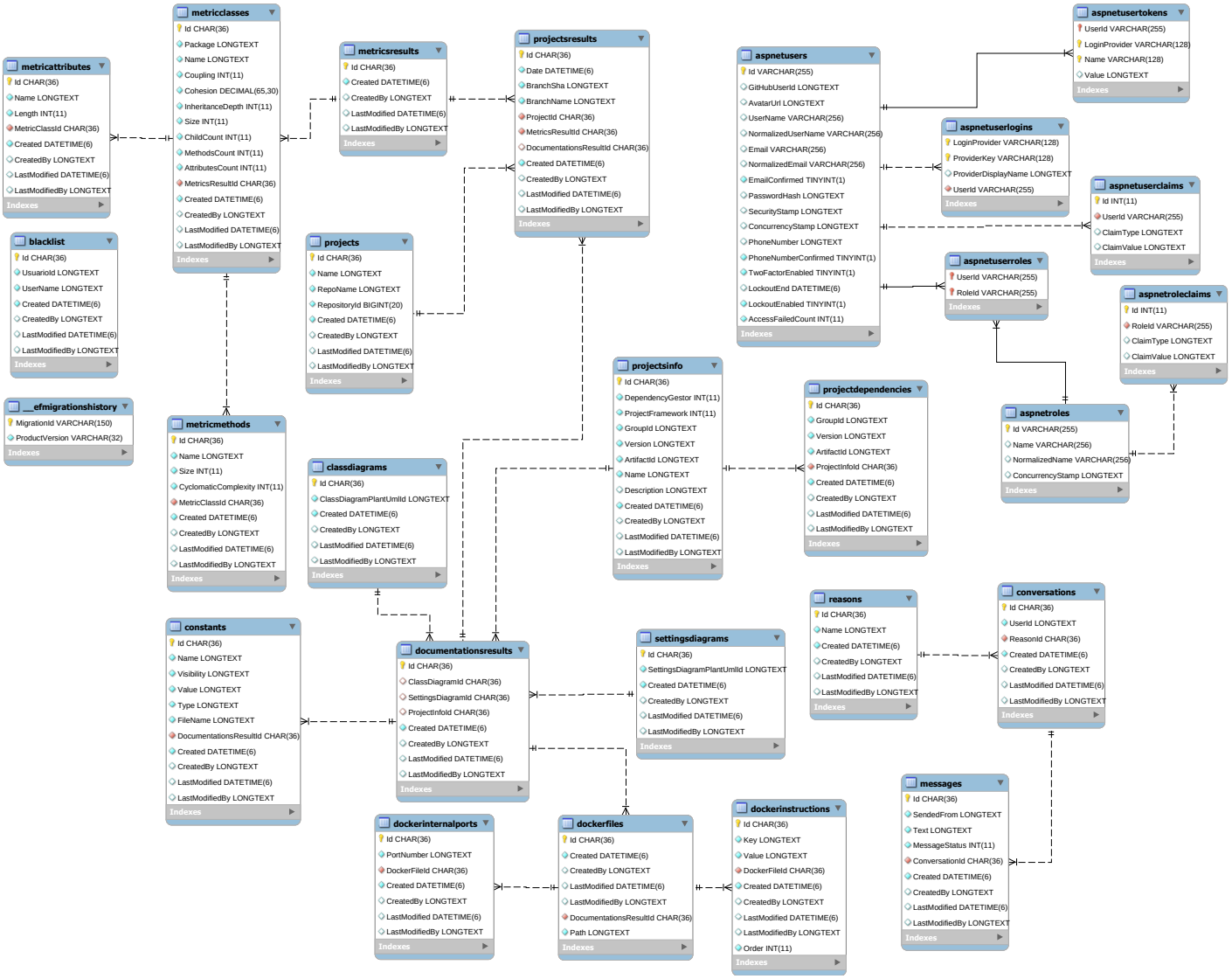
ANEXO N° 4 - DIAGRAMA DE CLASES DEL NEGOCIO DE JUNKODE

JUNKODE - Documentador Automático de Código Fuente


Base
created: DateTime
createdBy: Guid
id: Guid
lastModified: DateTime
lastModifiedBy: Guid
ClassDiagram
+ classDiagramPlantUmlId: String
ProjectInfo
+ artifactId: String
+ description: String
+ framework: String
+ gestor: String
+ groupId: String
+ name: String
+ version: String
ProjectDependecy
+ artifactId: String
+ groupId: String
+ version: String
projectDependecy
BlackList
UserInBlackList
User
+ avatarUrl: String
+ githubid: long
+ username: String
projects
Project
- Name: String
- repoId: String
- repoName: String
resultado
ProjectResult
+ brachName: String
+ branchSha: String
DocumentatioResutl
DocumentatioResult
classDiagram
projectInfo
settingsDiagram
SettingsDiagram
+ settingsPlantUmlId: String
constant
Constant
+ fileName: String
+ name: String
+ type: String
+ value: String
+ visibility: String
dockerFile
DockerFile
+ path: String
dockerInstruction
DockerInstruction
+ key: String
+ order: int
+ value: String
dockerInternalPort
DockerInternalPort
+ portNumber: String
Rol
+ name: String
User
Conversation
reason
messages
Reason
+ name: String
Message
+ status: String
+ text: String
MetricaResultado
metricResult
MetricClass
+ attributesCount: int
+ childCount: int
+ cohesion: float
+ coupling: int
+ inheritanceDepth: int
+ methodsConstants: int
+ name: String
+ package: String
+ size: int
+ startWithLower: boolean
clases
methods
MetricMethod
+ ciclomaticComplexity: int
+ name: String
+ size: int
+ starWithUpper: boolean
attributes
MetricAttribute
+ constantWithLower: boolean
+ hasPackageVisibility: boolean
+ lenght: int
+ name: String
+ startsWithUpper: boolean

ANEXO N° 5 - MODELO DE DATOS DE JUNKODE

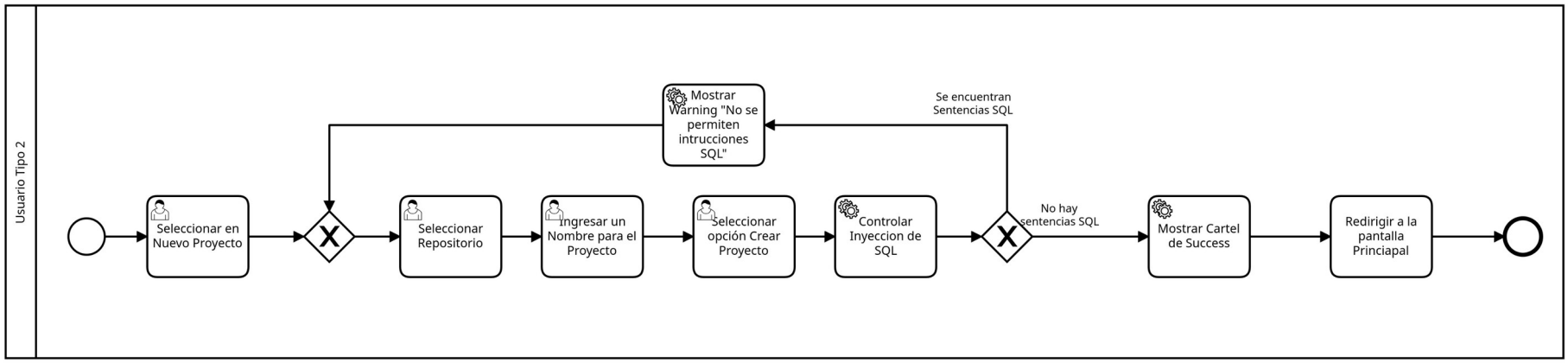
JUNKODE - Documentador Automático de Código Fuente


metricattributes
Id CHAR(36)
Name LONGTEXT
Length INT(11)
MetricClassId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
metricclasses
Id CHAR(36)
Package LONGTEXT
Name LONGTEXT
Coupling INT(11)
Cohesion DECIMAL(65,30)
InheritanceDepth INT(11)
Size INT(11)
ChildCount INT(11)
MethodsCount INT(11)
AttributesCount INT(11)
MetricsResultId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
metricsresults
Id CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
projectsresults
Id CHAR(36)
Date DATETIME(6)
BranchSha LONGTEXT
BranchName LONGTEXT
ProjectId CHAR(36)
MetricsResultId CHAR(36)
DocumentationsResultId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
aspnetusers
Id VARCHAR(255)
GitHubUserId LONGTEXT
AvatarUrl LONGTEXT
UserName VARCHAR(256)
NormalizedUserName VARCHAR(256)
Email VARCHAR(256)
NormalizedEmail VARCHAR(256)
EmailConfirmed TINYINT(1)
PasswordHash LONGTEXT
SecurityStamp LONGTEXT
ConcurrencyStamp LONGTEXT
PhoneNumber LONGTEXT
PhoneNumberConfirmed TINYINT(1)
TwoFactorEnabled TINYINT(1)
LockoutEnd DATETIME(6)
LockoutEnabled TINYINT(1)
AccessFailedCount INT(11)
Indexes
aspnetusertokens
UserId VARCHAR(255)
LoginProvider VARCHAR(128)
Name VARCHAR(128)
Value LONGTEXT
Indexes
aspnetuserlogins
LoginProvider VARCHAR(128)
ProviderKey VARCHAR(128)
ProviderDisplayName LONGTEXT
UserId VARCHAR(255)
Indexes
aspnetuserclaims
Id INT(11)
UserId VARCHAR(255)
ClaimType LONGTEXT
ClaimValue LONGTEXT
Indexes
aspnetuserroles
UserId VARCHAR(255)
RoleId VARCHAR(255)
Indexes
aspnetroleclaims
Id INT(11)
RoleId VARCHAR(255)
ClaimType LONGTEXT
ClaimValue LONGTEXT
Indexes
aspnetroles
Id VARCHAR(255)
Name VARCHAR(256)
NormalizedName VARCHAR(256)
ConcurrencyStamp LONGTEXT
Indexes
blacklist
Id CHAR(36)
UsuarioId LONGTEXT
UserName LONGTEXT
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
projects
Id CHAR(36)
Name LONGTEXT
RepoName LONGTEXT
RepositoryId BIGINT(20)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
__efmigrationshistory
MigrationId VARCHAR(150)
ProductVersion VARCHAR(32)
Indexes
metricmethods
Id CHAR(36)
Name LONGTEXT
Size INT(11)
CyclomaticComplexity INT(11)
MetricClassId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
classdiagrams
Id CHAR(36)
ClassDiagramPlantUmlId LONGTEXT
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
projectsinfo
Id CHAR(36)
DependencyGestor INT(11)
ProjectFramework INT(11)
GroupId LONGTEXT
Version LONGTEXT
ArtifactId LONGTEXT
Name LONGTEXT
Description LONGTEXT
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
projectdependencies
Id CHAR(36)
GroupId LONGTEXT
Version LONGTEXT
ArtifactId LONGTEXT
ProjectInfoId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
constants
Id CHAR(36)
Name LONGTEXT
Visibility LONGTEXT
Value LONGTEXT
Type LONGTEXT
FileName LONGTEXT
DocumentationsResultId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
documentationsresults
Id CHAR(36)
ClassDiagramId CHAR(36)
SettingsDiagramId CHAR(36)
ProjectInfoId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
settingsdiagrams
Id CHAR(36)
SettingsDiagramPlantUmlId LONGTEXT
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
reasons
Id CHAR(36)
Name LONGTEXT
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
conversations
Id CHAR(36)
UserId LONGTEXT
ReasonId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
messages
Id CHAR(36)
SendedFrom LONGTEXT
Text LONGTEXT
MessageStatus INT(11)
ConversationId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
dockerinternalports
Id CHAR(36)
PortNumber LONGTEXT
DockerFileId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Indexes
dockerfiles
Id CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
DocumentationsResultId CHAR(36)
Path LONGTEXT
Indexes
dockerinstructions
Id CHAR(36)
Key LONGTEXT
Value LONGTEXT
DockerFileId CHAR(36)
Created DATETIME(6)
CreatedBy LONGTEXT
LastModified DATETIME(6)
LastModifiedBy LONGTEXT
Order INT(11)
Indexes

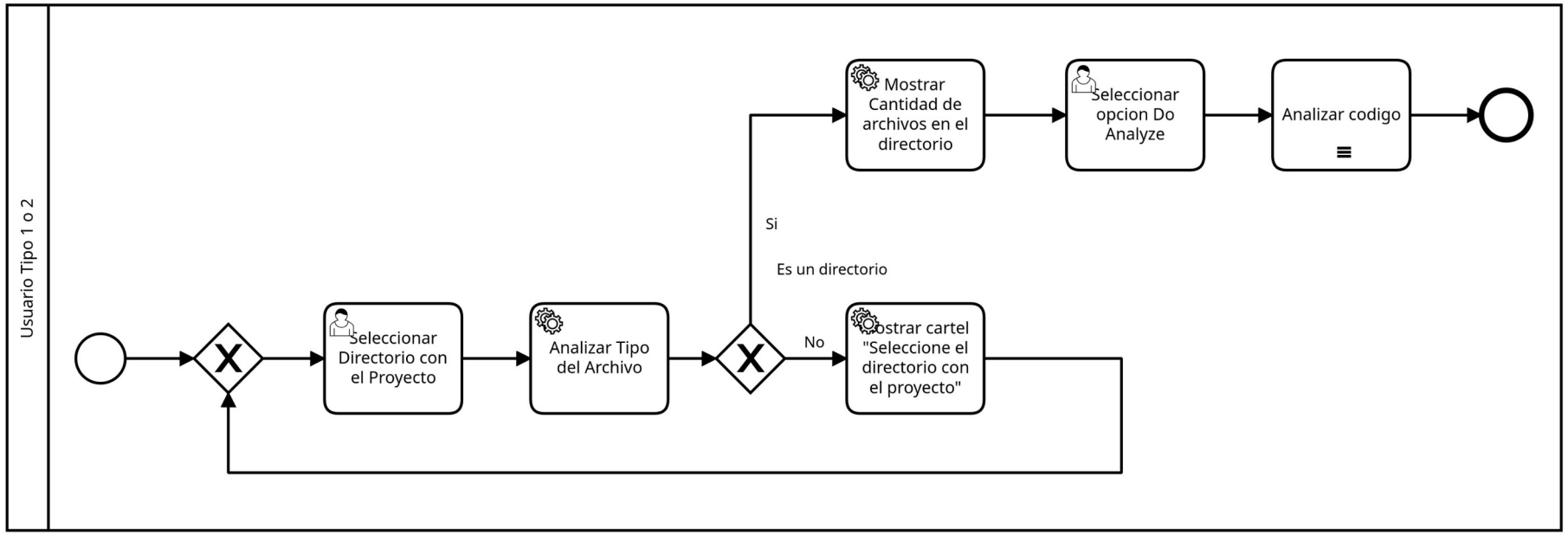
ANEXO N° 6 - DIAGRAMAS BPMN

JUNKODE - Documentador Automático de Código Fuente

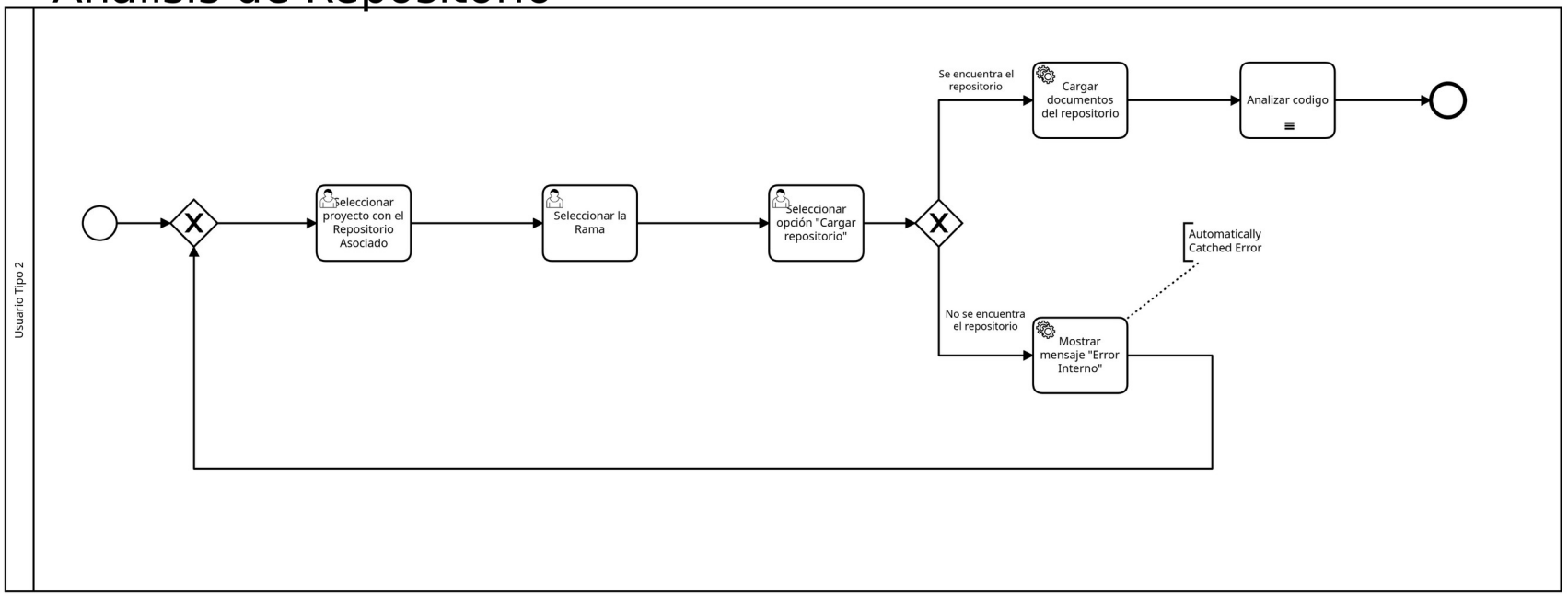
Crear Proyecto



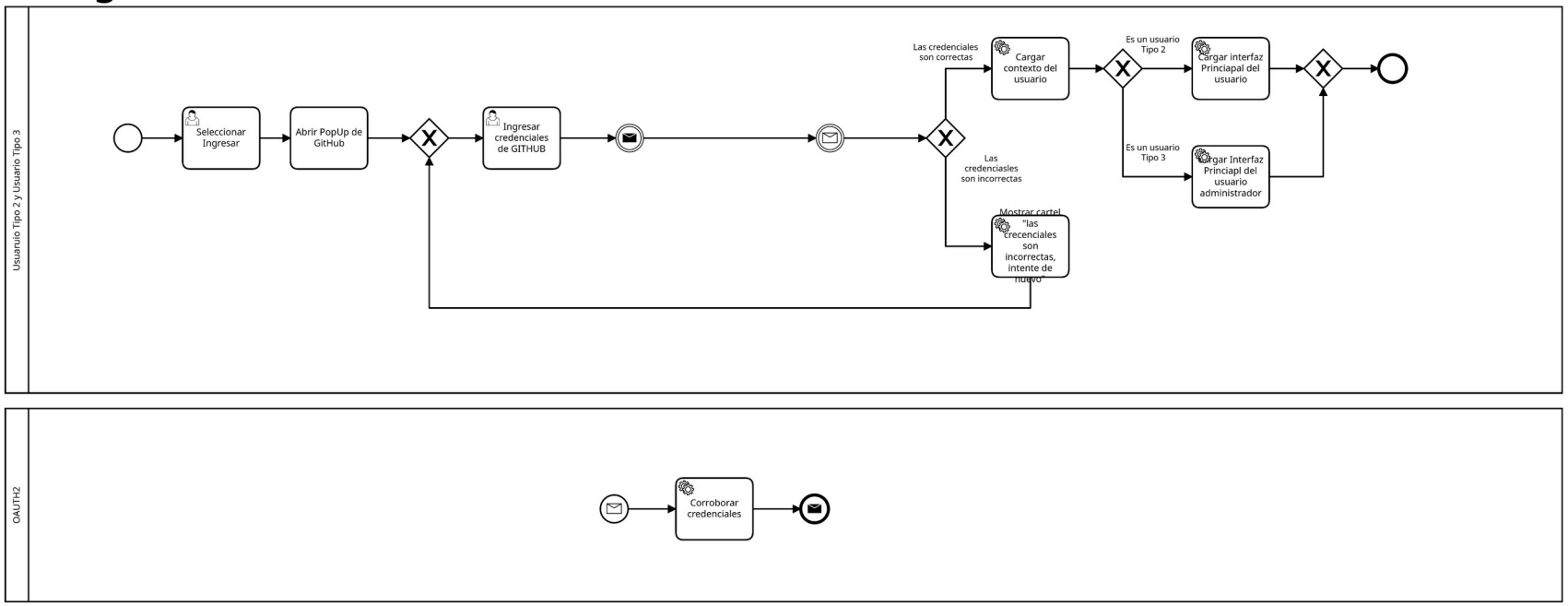
Análisis de Proyecto Local



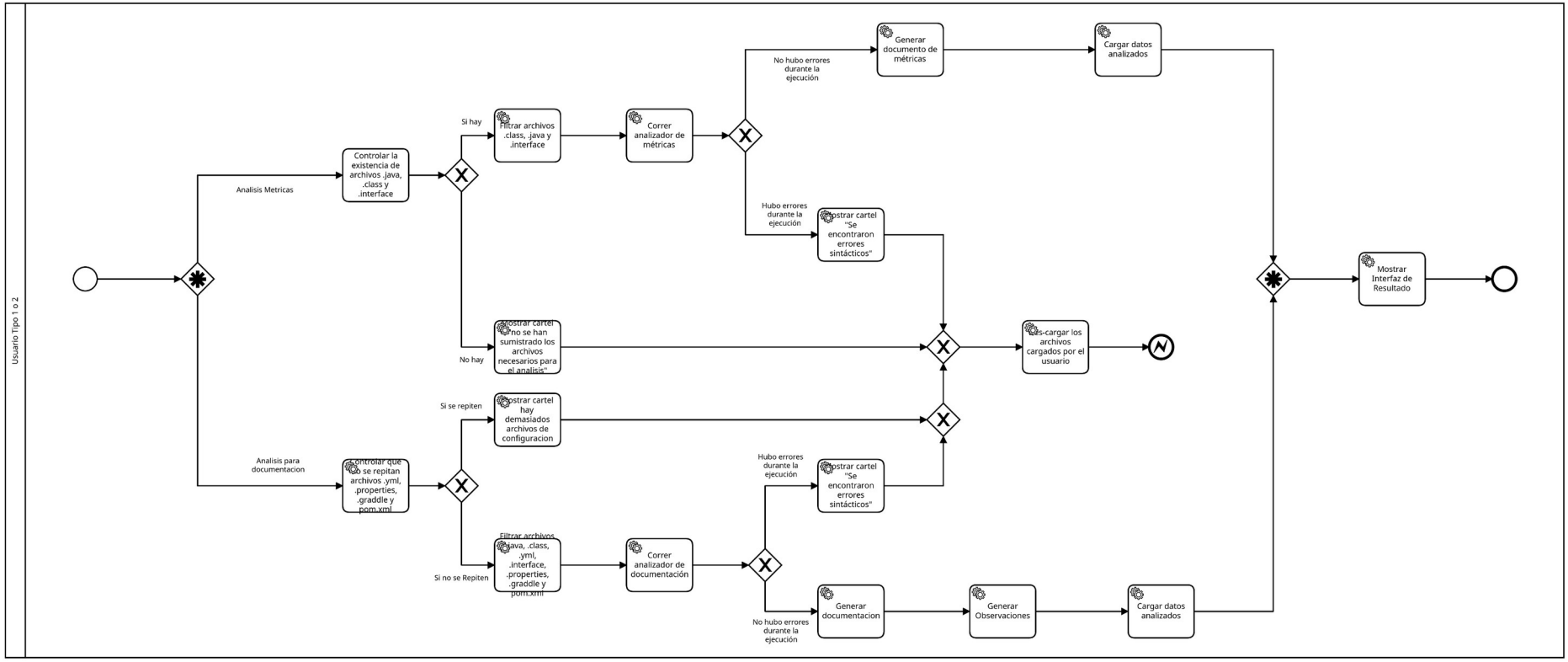
Análisis de Repositorio



Ingresar Usuario



Análisis de Proyecto



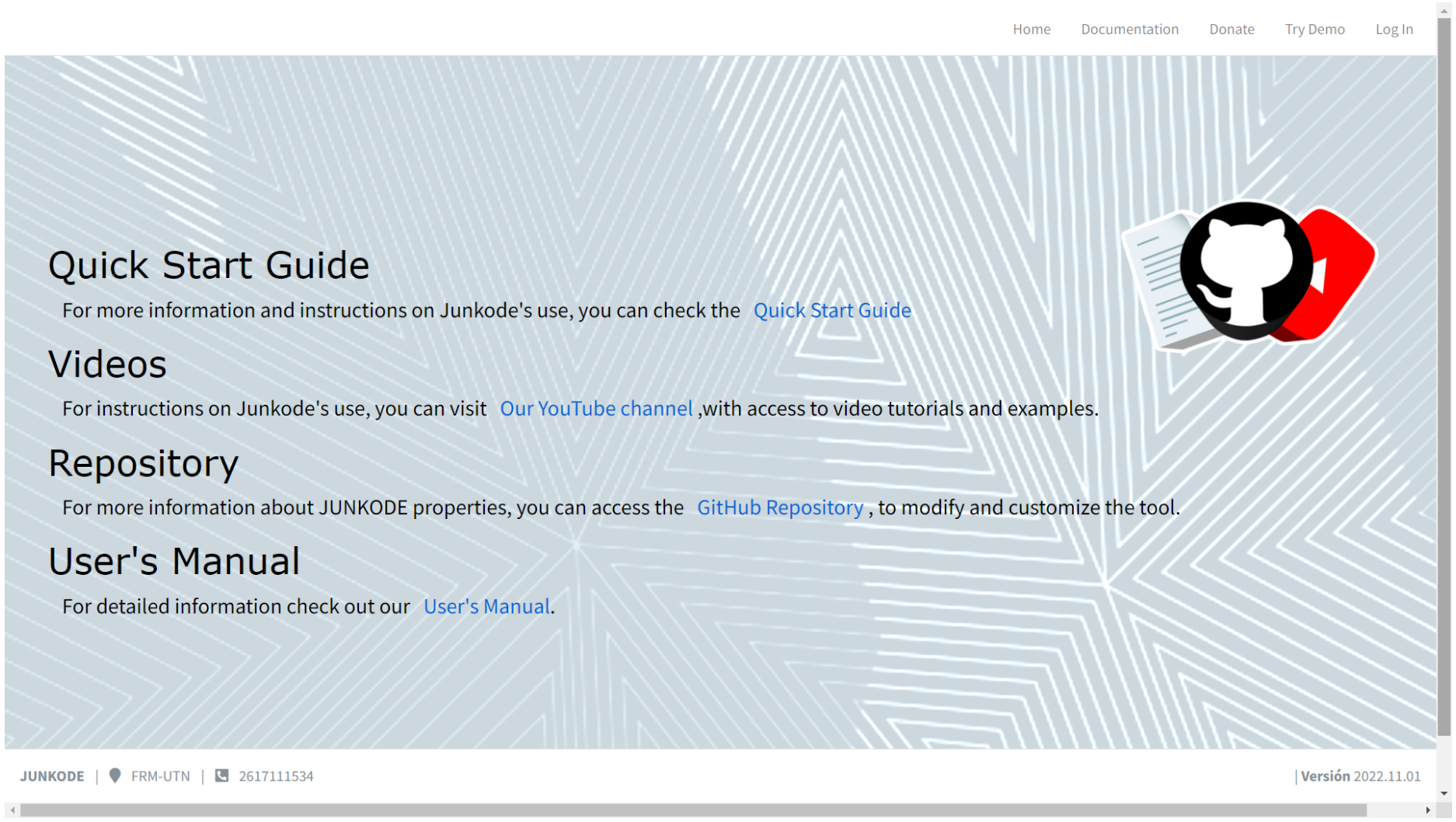
ANEXO N° 7 - PANTALLAS Y REPORTES DE JUNKODE

JUNKODE - Documentador Automático de Código Fuente

Pantalla N°1.


Home
Documentation
Donate
Try Demo
Log In
WELCOME TO
JUNKODE

Pantalla N°2.


Home
Documentation
Donate
Try Demo
Log In
Quick Start Guide
For more information and instructions on Junkode's use, you can check the Quick Start Guide
Videos
For instructions on Junkode's use, you can visit Our YouTube channel ,with access to video tutorials and examples.
Repository
For more information about JUNKODE properties, you can access the GitHub Repository , to modify and customize the tool.
User's Manual
For detailed information check out our User's Manual.
JUNKODE | FRM-UTN | 2617111534
| Versión 2022.11.01

Pantalla N°3.

Home Documentation Donate Try Demo Log In

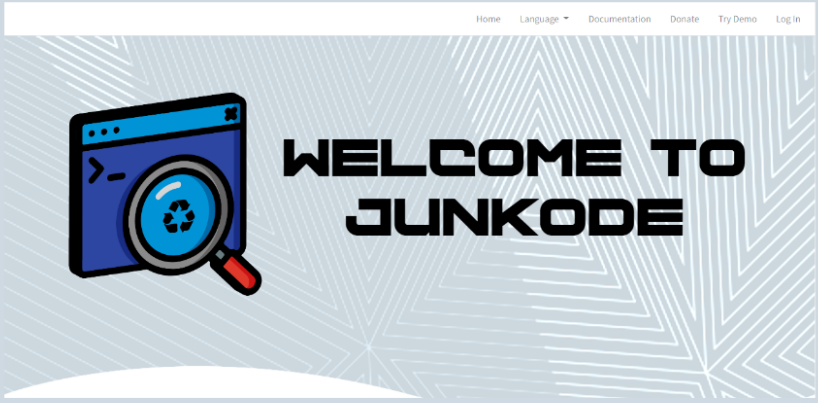
Guía de inicio de Junkode

ÍNDICE

El objetivo de esta guía es poder aplicar y entender el funcionamiento de JUNKODE

Sesión

Junkode permite a sus usuarios acceder a un set de funcionalidades limitadas aún si no ha iniciado sesión con una cuenta (carga de proyecto local, análisis y descarga de resultados).
Para poder utilizar todas las funcionalidades de Junkode, se debe disponer de una cuenta de Github, para poder iniciar sesión.

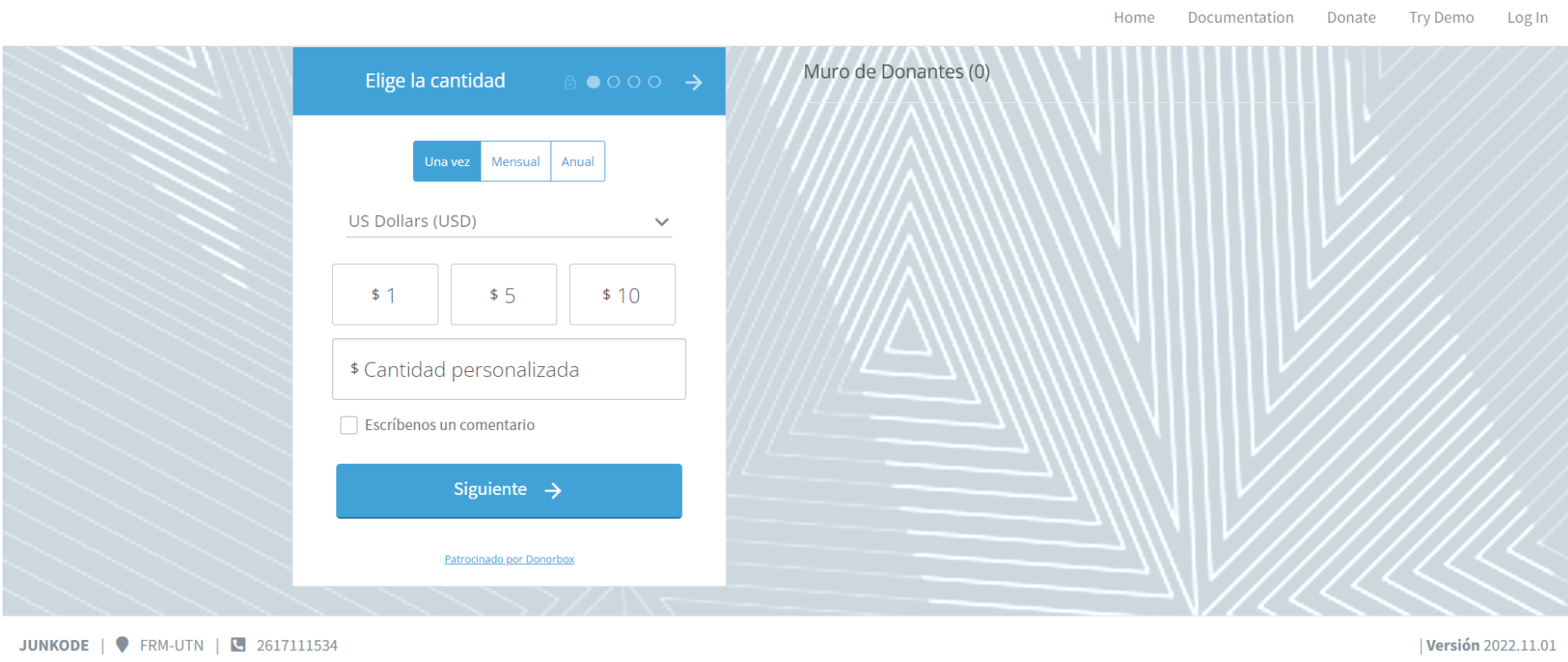


Las cuentas de Junkode están vinculadas con los Usuarios de GitHub, esto permite que los repositorios puedan ser accedidos de manera confiable y segura.
La sesión puede iniciarse desde el botón “Log In” de la página principal de Junkode, o desde la carga del código

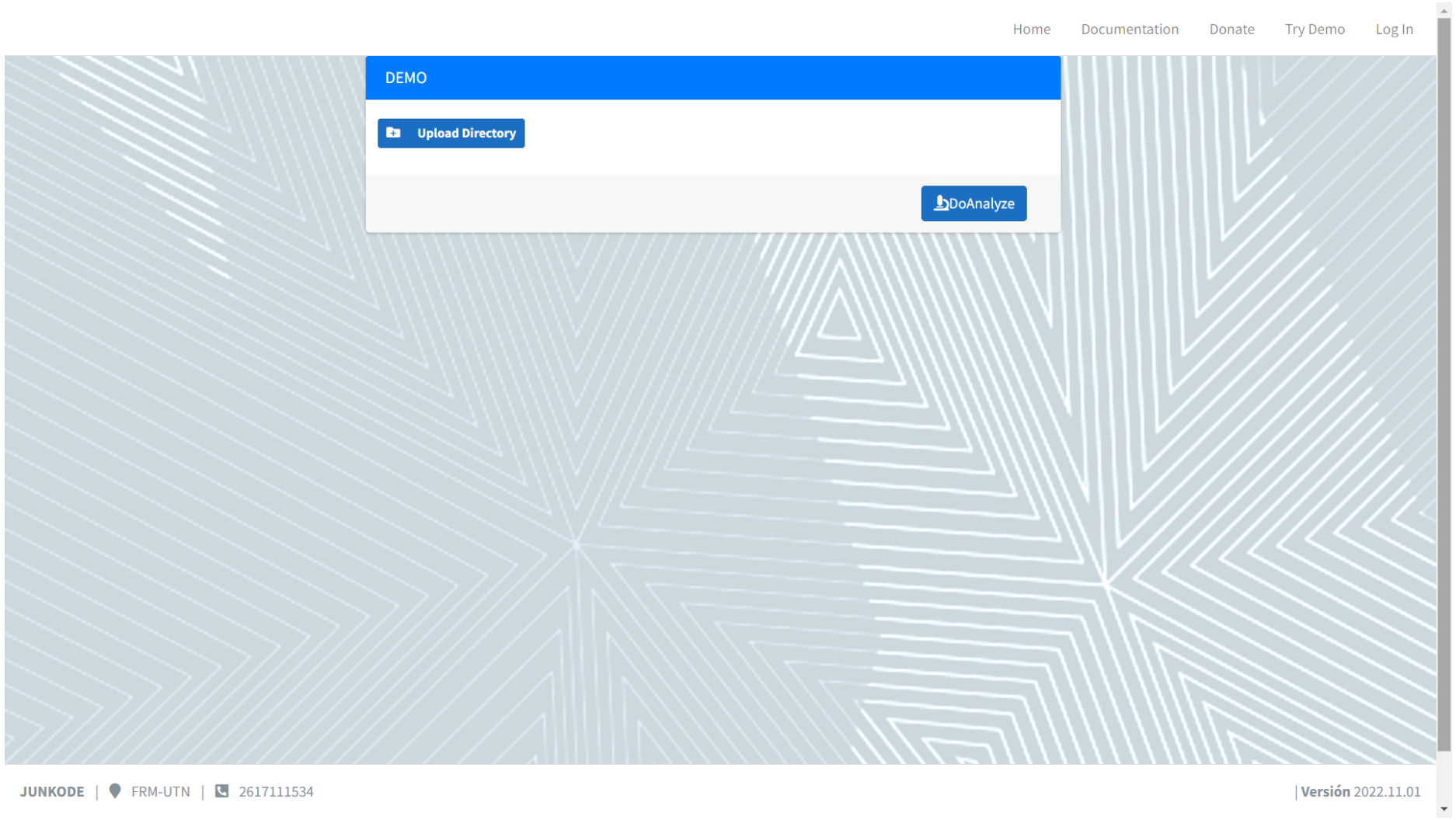
Carga del Código

La carga del proyecto Java se puede realizar de dos formas distintas

Pantalla N°4.


Home
Documentation
Donate
Try Demo
Log In
Elige la cantidad
Una vez
Mensual
Anual
US Dollars (USD)
$ 1
$ 5
$ 10
$ Cantidad personalizada
Escríbenos un comentario
Siguiente
Patrocinado por Donorbox
Muro de Donantes (0)
JUNKODE | FRM-UTN | 2617111534
| Versión 2022.11.01

Pantalla N°5.


Home
Documentation
Donate
Try Demo
Log In
DEMO
Upload Directory
DoAnalyze
JUNKODE | FRM-UTN | 2617111534
| Versión 2022.11.01

Pantalla N°6.

Sign in to **GitHub**
to continue to **Junkode**

Username or email address

ramsesgiralald@gmail.com

Password Forgot password?

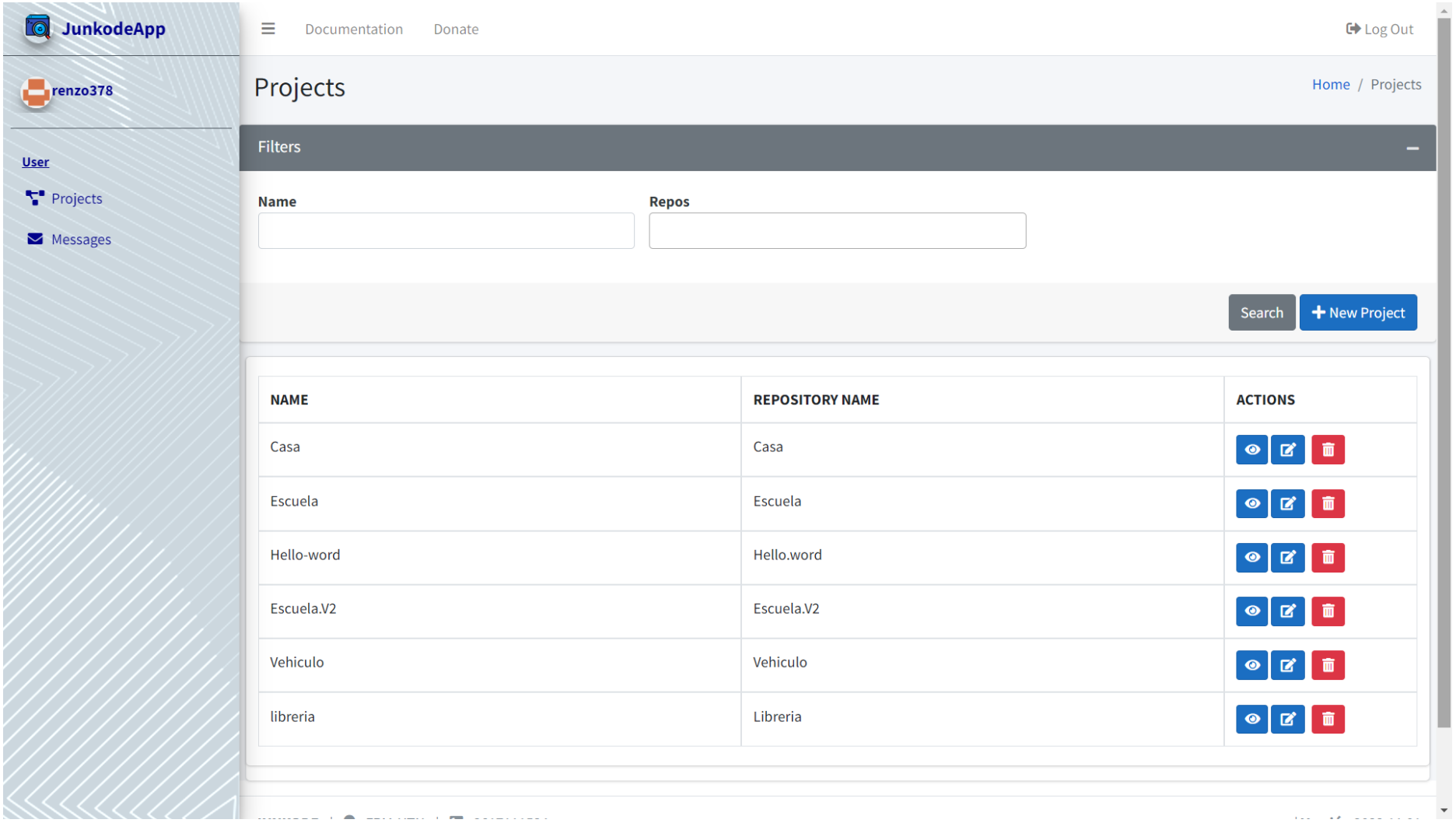
••••••••••••

Sign in

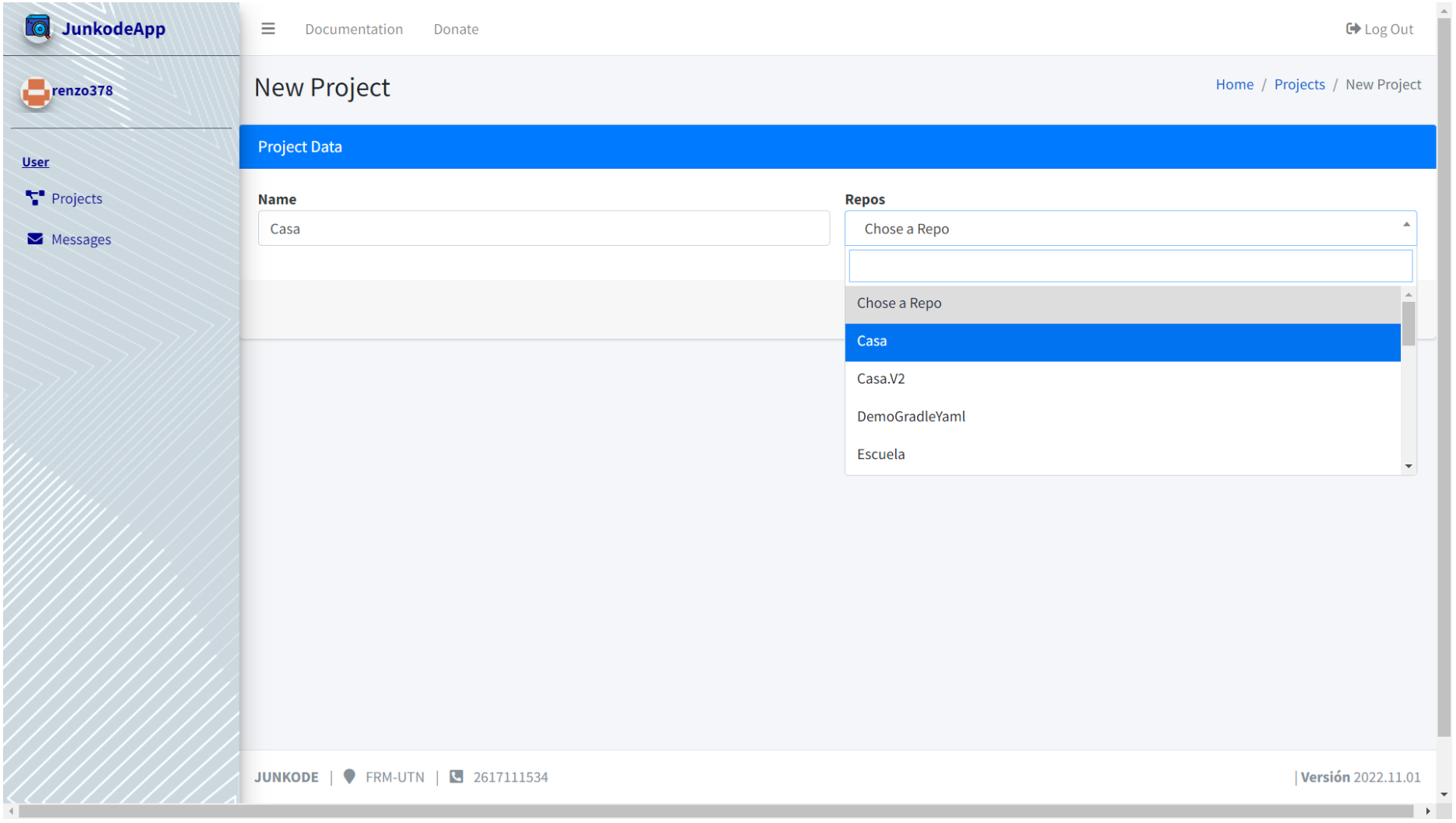
New to GitHub? Create an account.

Terms Privacy Security Contact GitHub

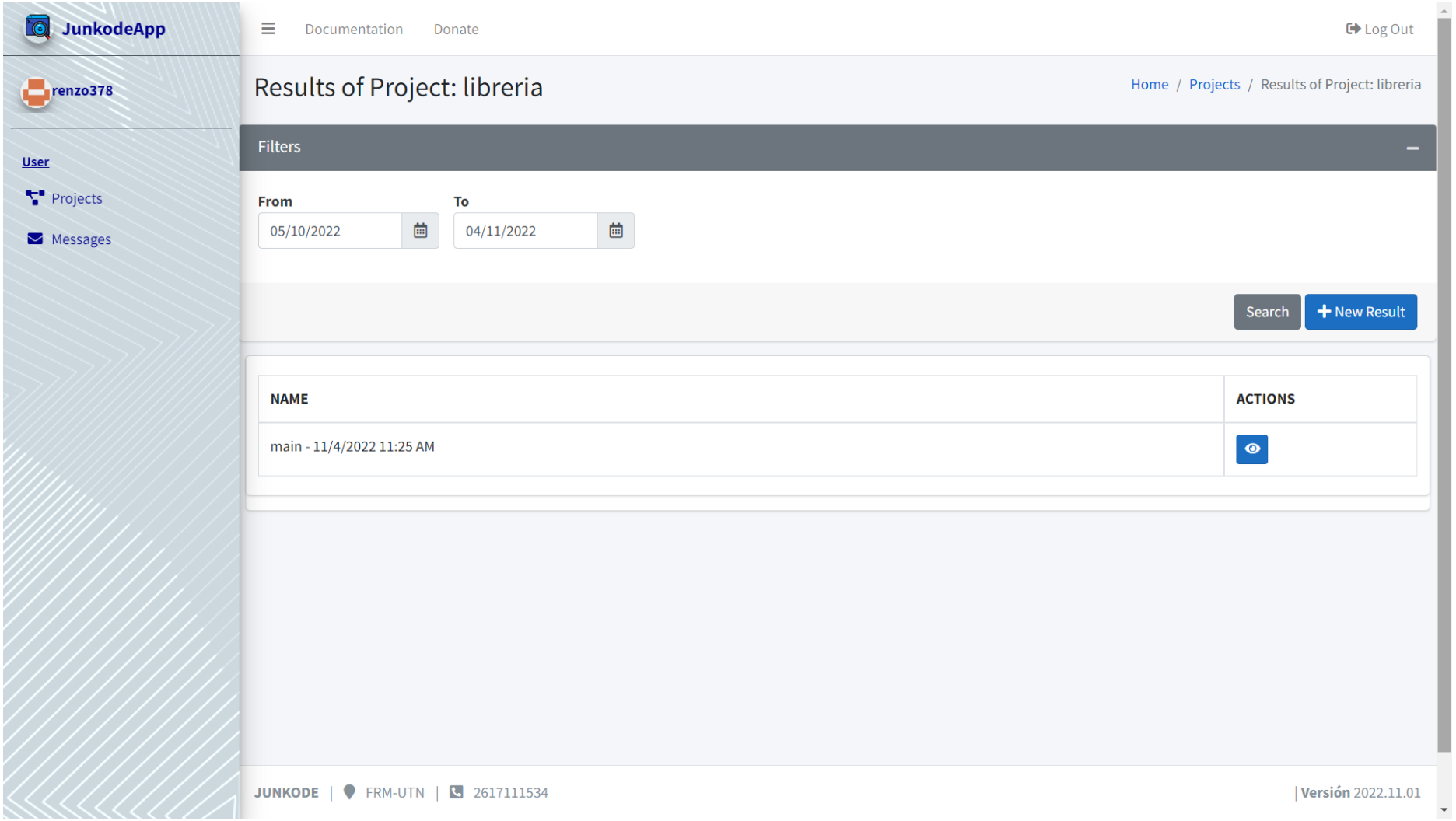
Pantalla N°7.


JunkodeApp
Documentation
Donate
Log Out
renzo378
Projects
Home / Projects
User
Projects
Messages
Filters
Name
Repos
Search
New Project
NAME
REPOSITORY NAME
ACTIONS
Casa
Casa
Escuela
Escuela
Hello-word
Hello.word
Escuela.V2
Escuela.V2
Vehiculo
Vehiculo
libreria
Libreria

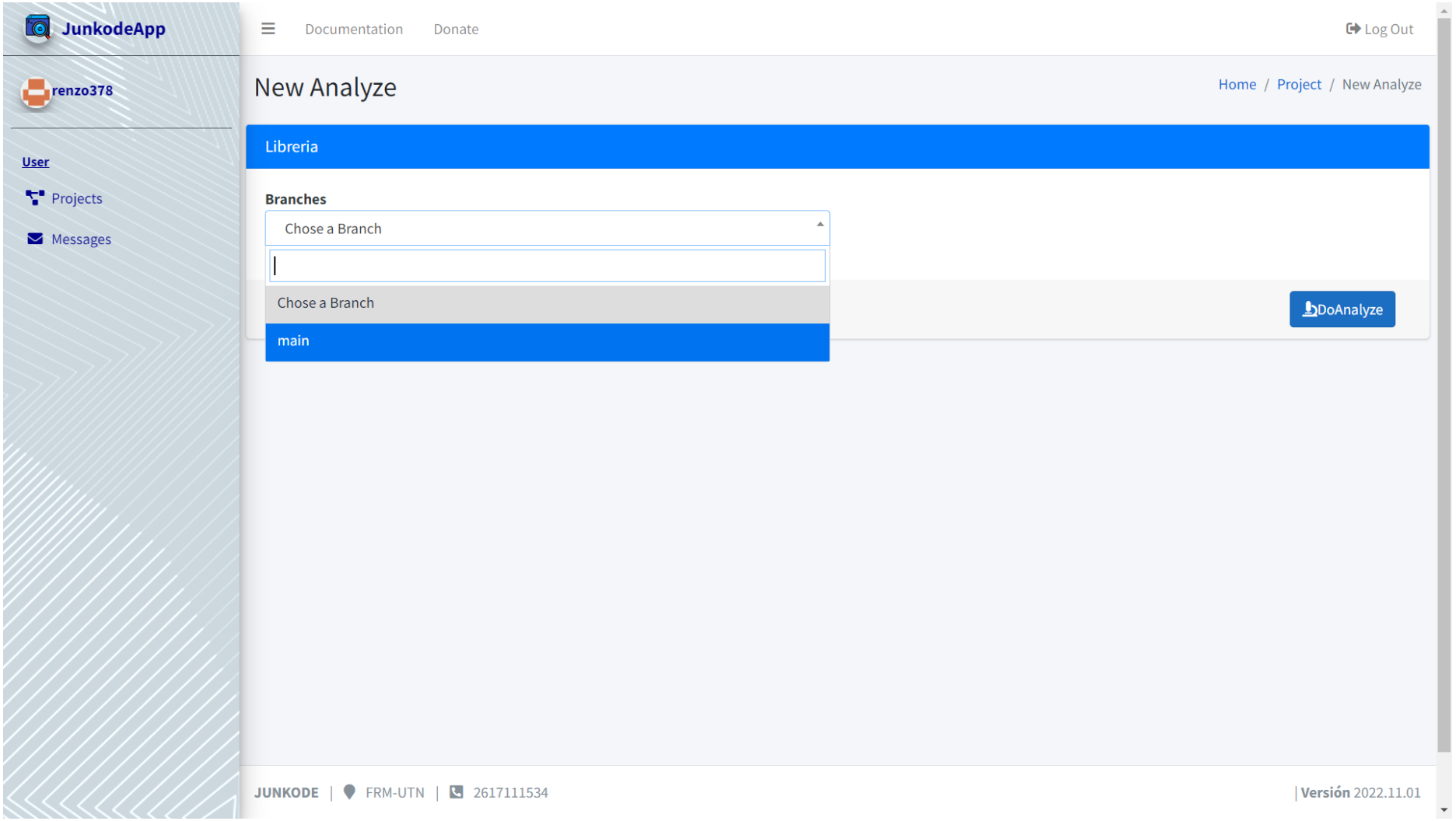
Pantalla N°8.


JunkodeApp
Documentation
Donate
Log Out
renzo378
New Project
Home / Projects / New Project
User
Projects
Messages
Project Data
Name
Casa
Repos
Chose a Repo
Chose a Repo
Casa
Casa.V2
DemoGradleYaml
Escuela
JUNKODE | FRM-UTN | 2617111534
| Versión 2022.11.01

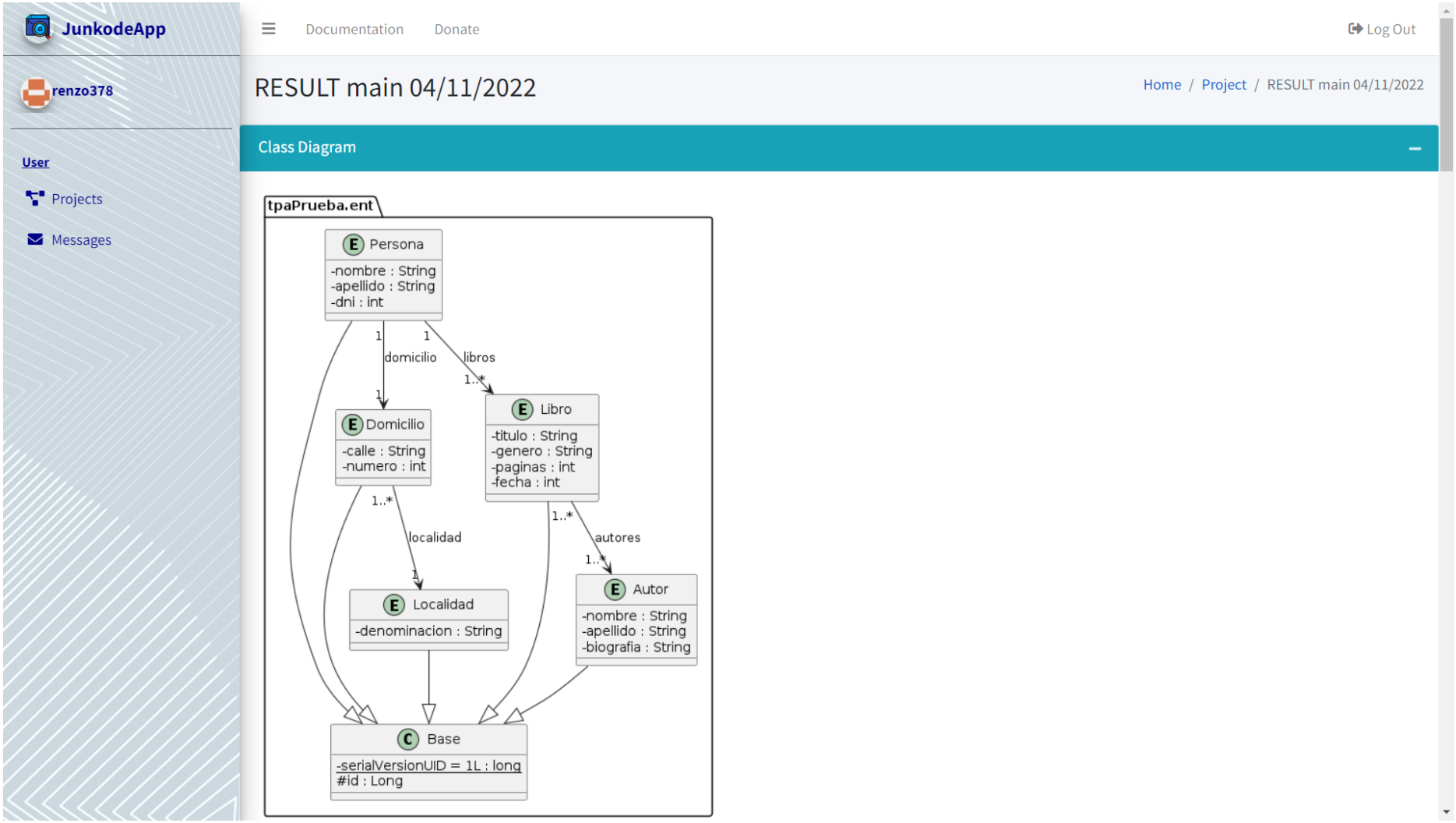
Pantalla N°9.


JunkodeApp
Documentation
Donate
Log Out
renzo378
Results of Project: libreria
Home / Projects / Results of Project: libreria
User
Projects
Messages
Filters
From
05/10/2022
To
04/11/2022
Search
New Result
NAME
ACTIONS
main - 11/4/2022 11:25 AM
JUNKODE | FRM-UTN | 2617111534
Versión 2022.11.01

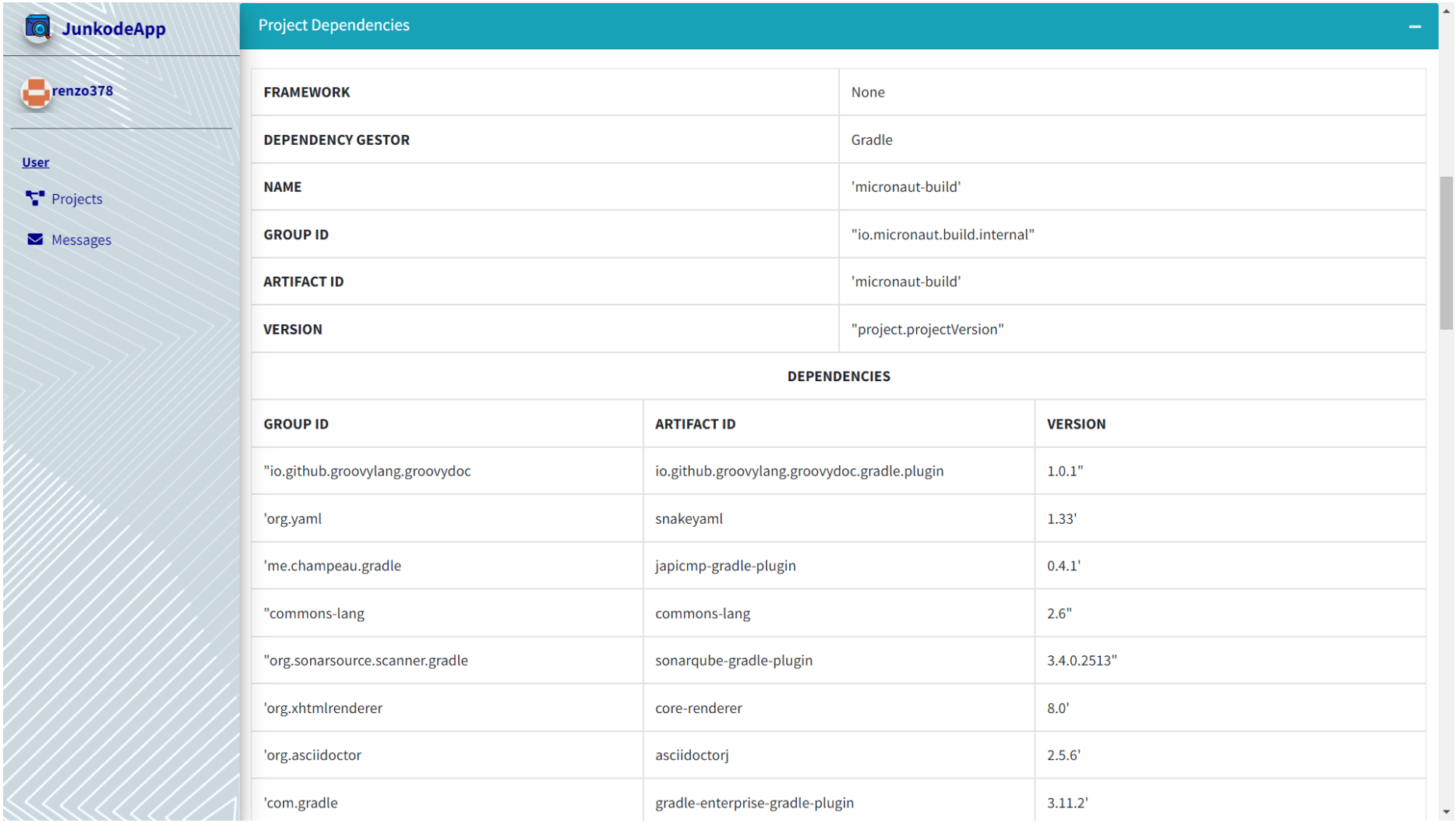
Pantalla N°10.



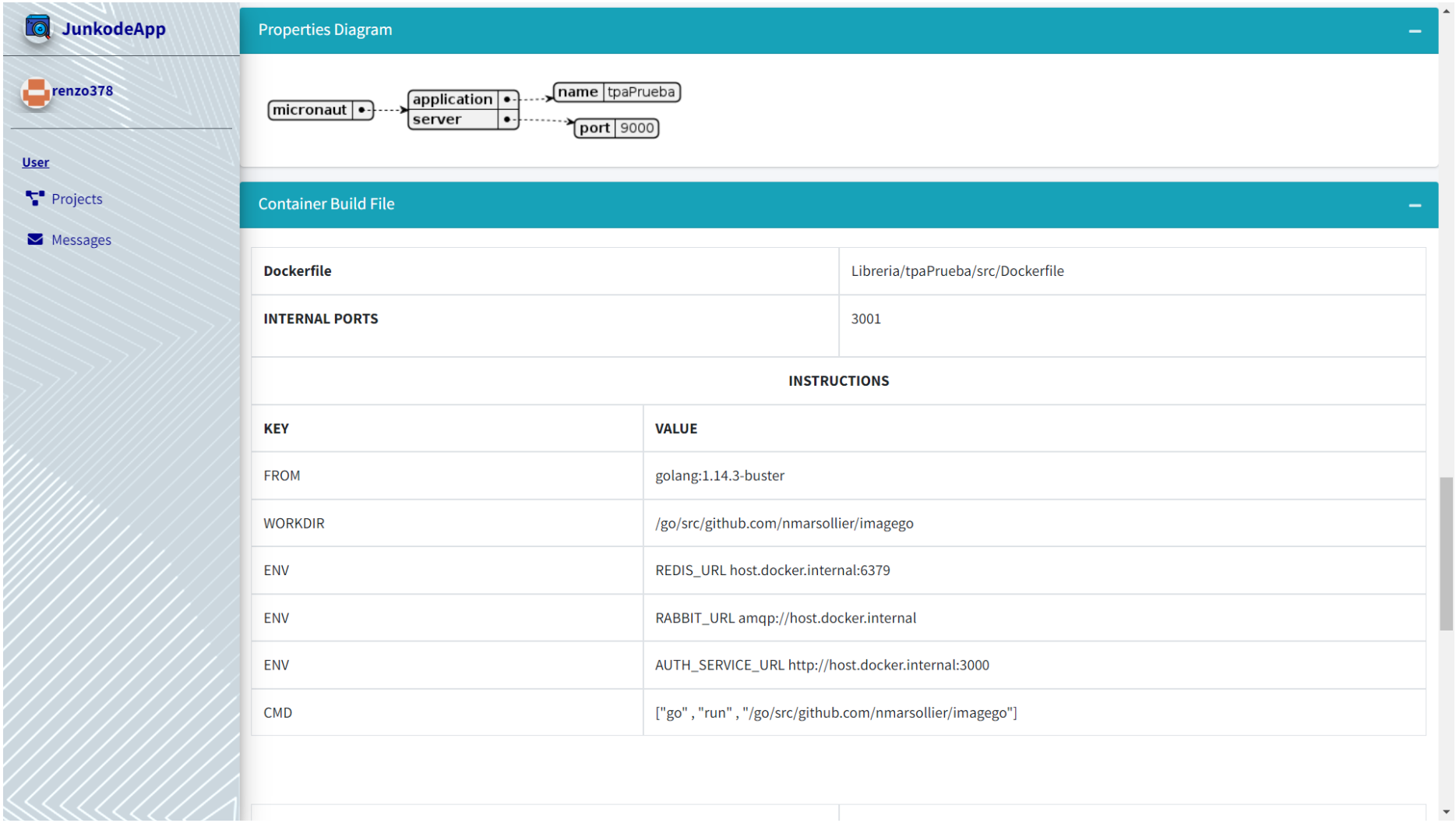
Pantalla N°11.



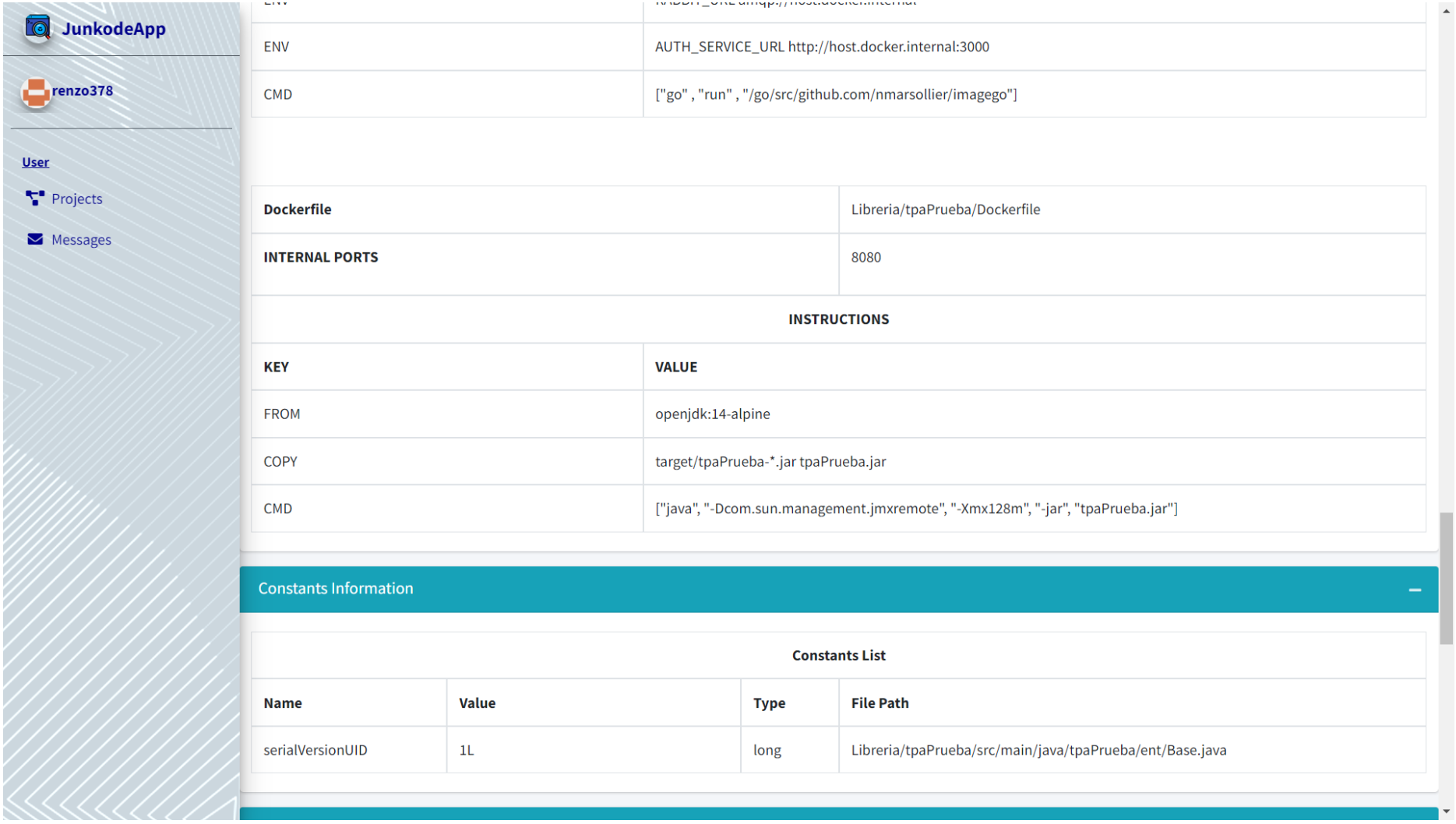
Pantalla N°12.



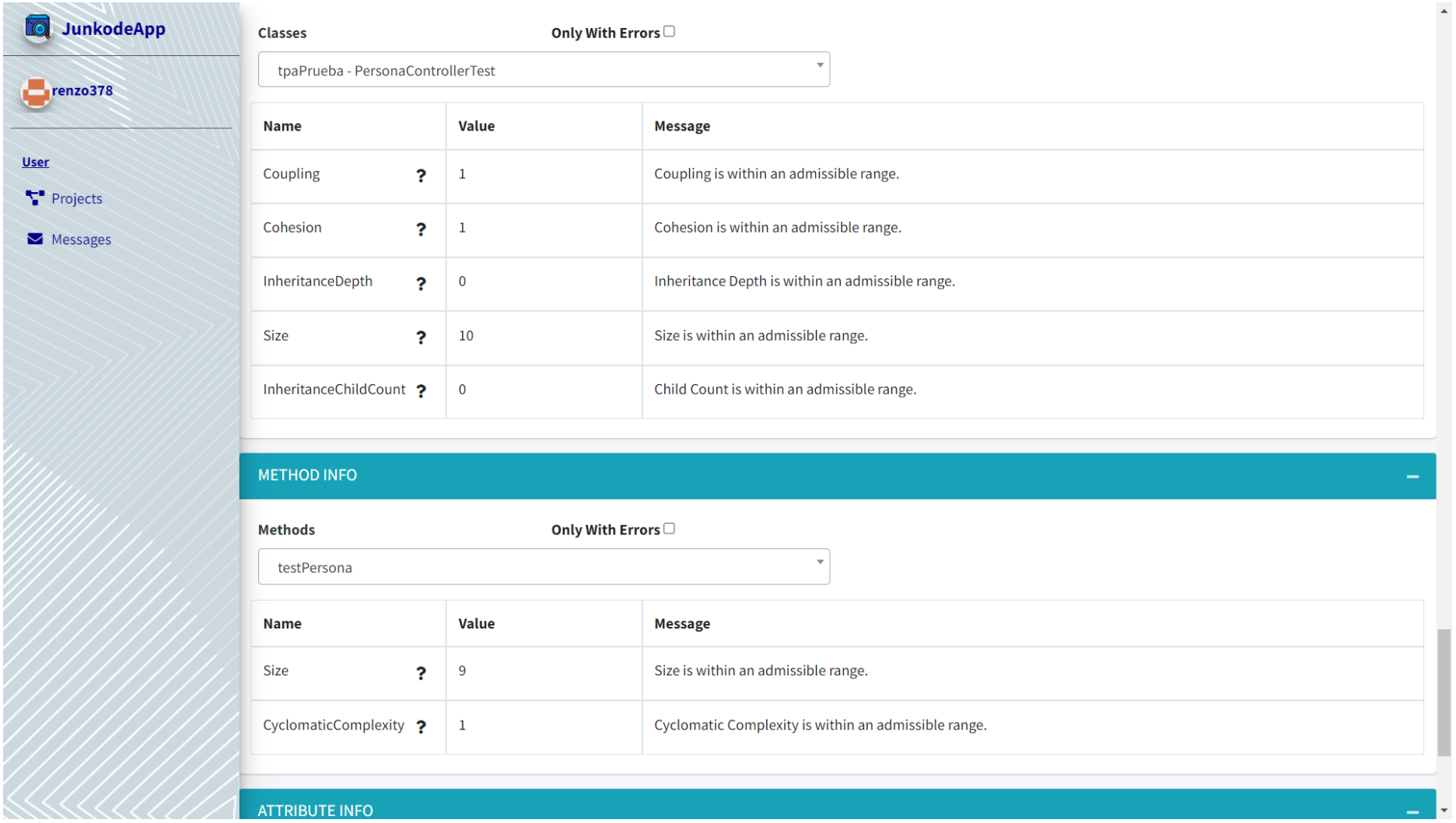
Pantalla N°13.



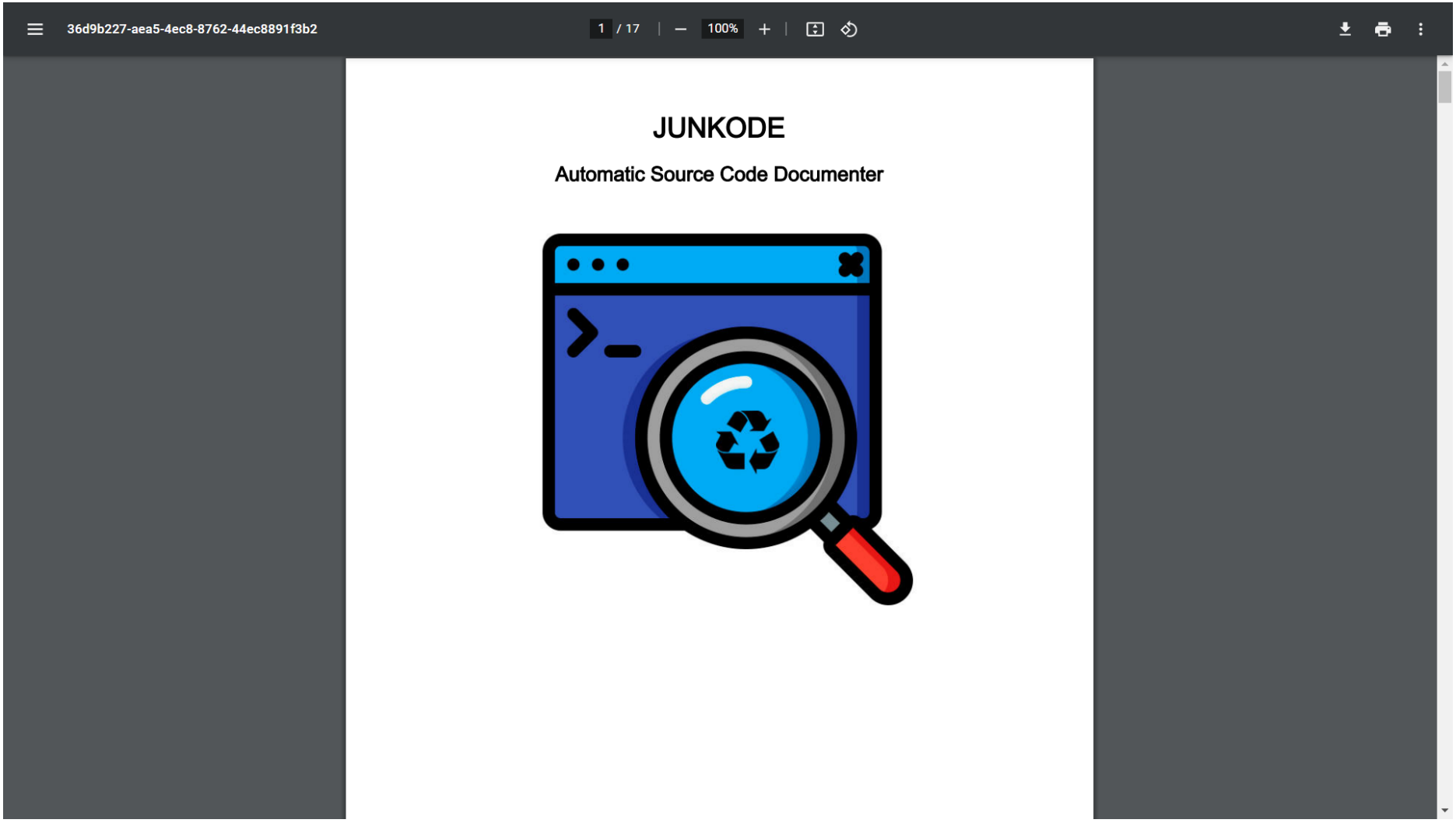
Pantalla N°14.



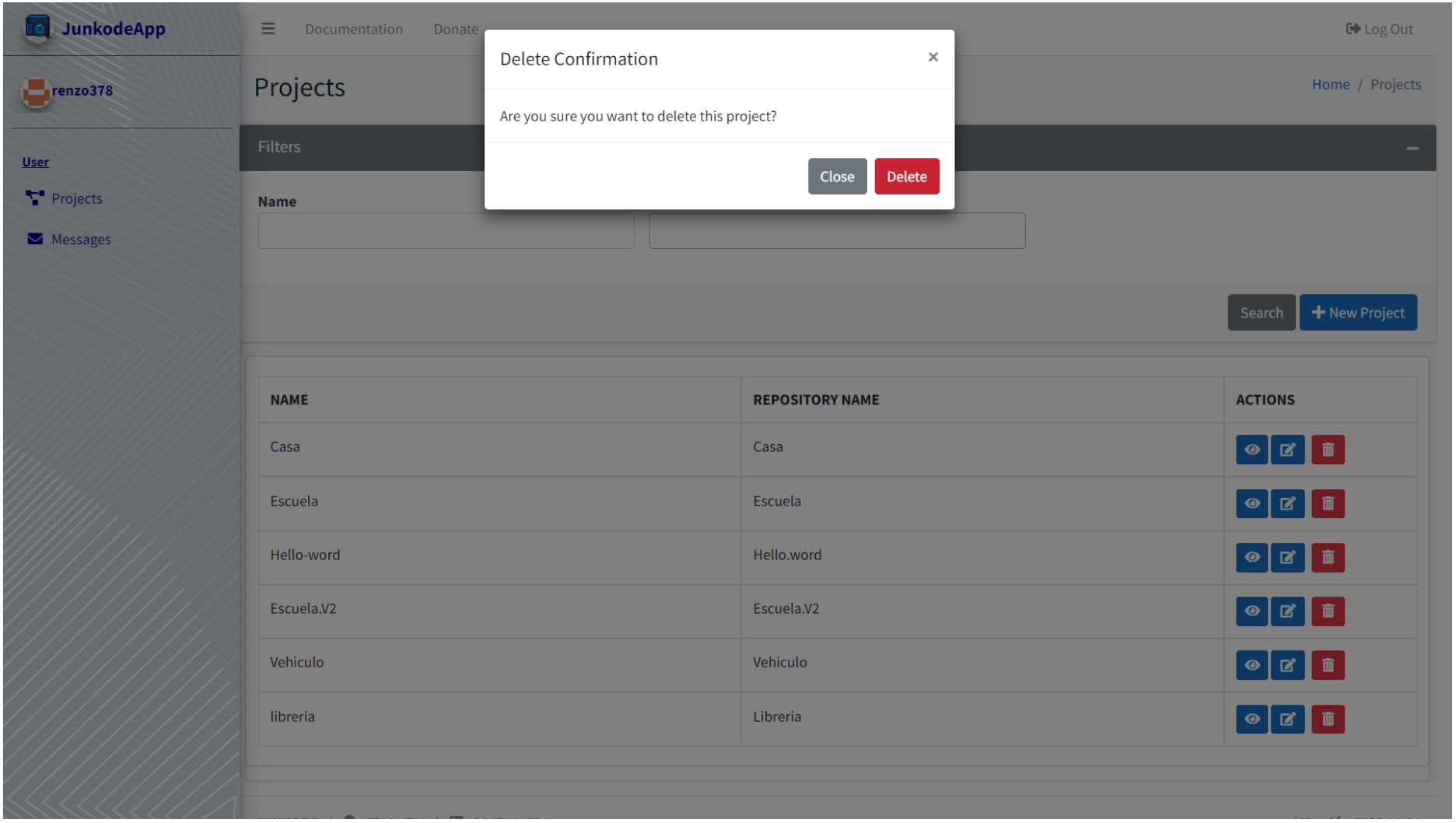
Pantalla N°15.



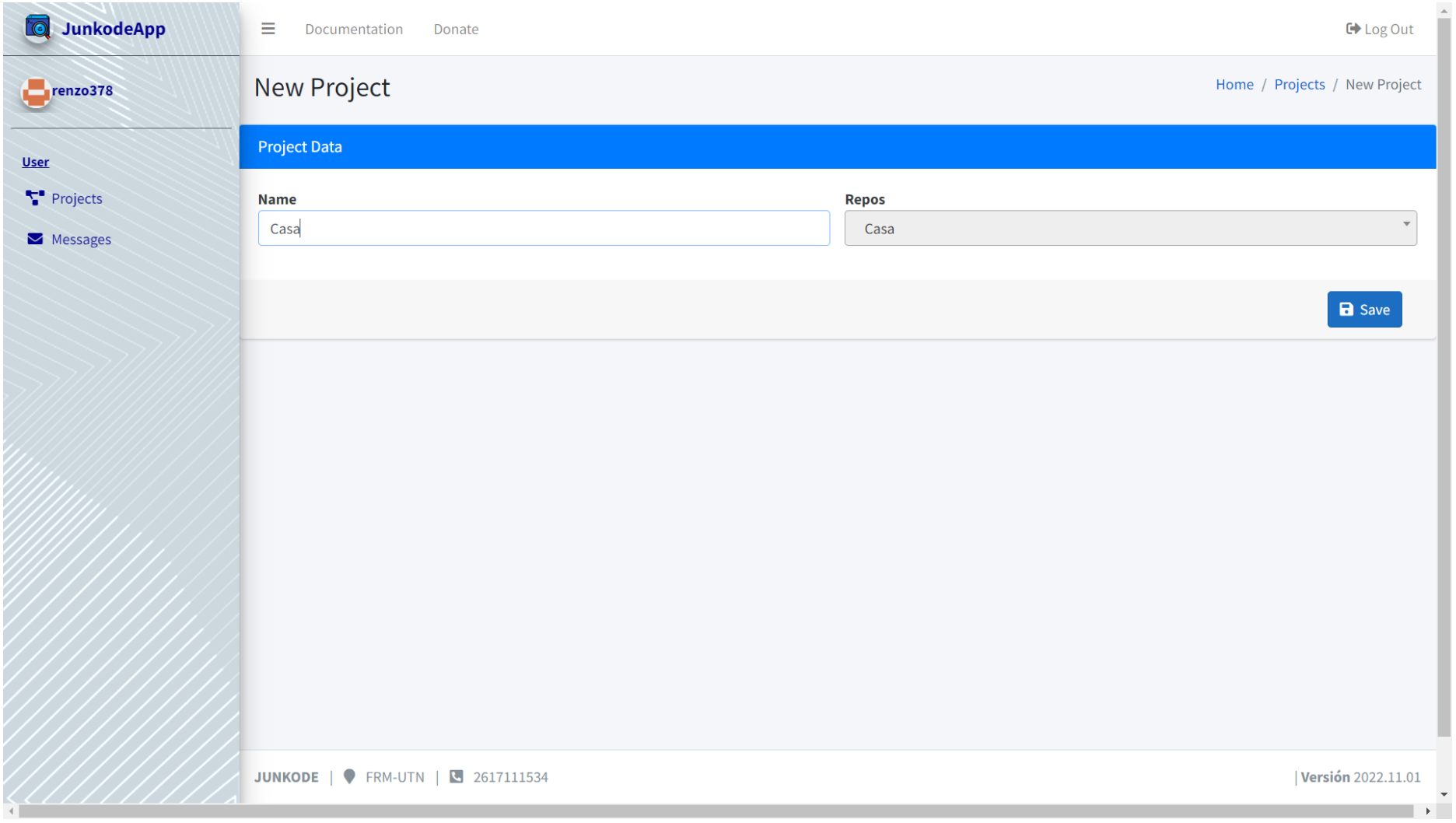
Pantalla N°16.



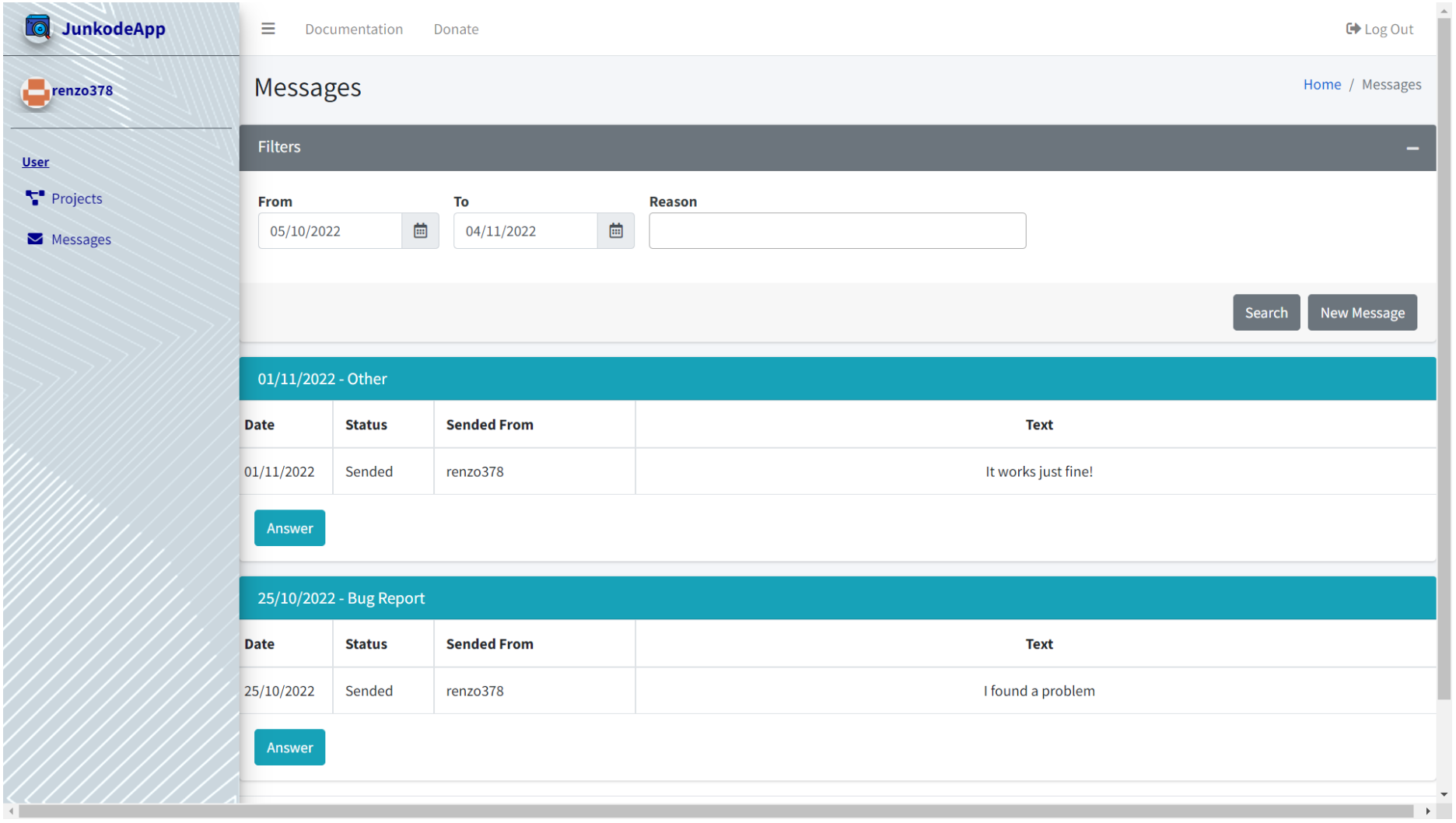
Pantalla N°17.



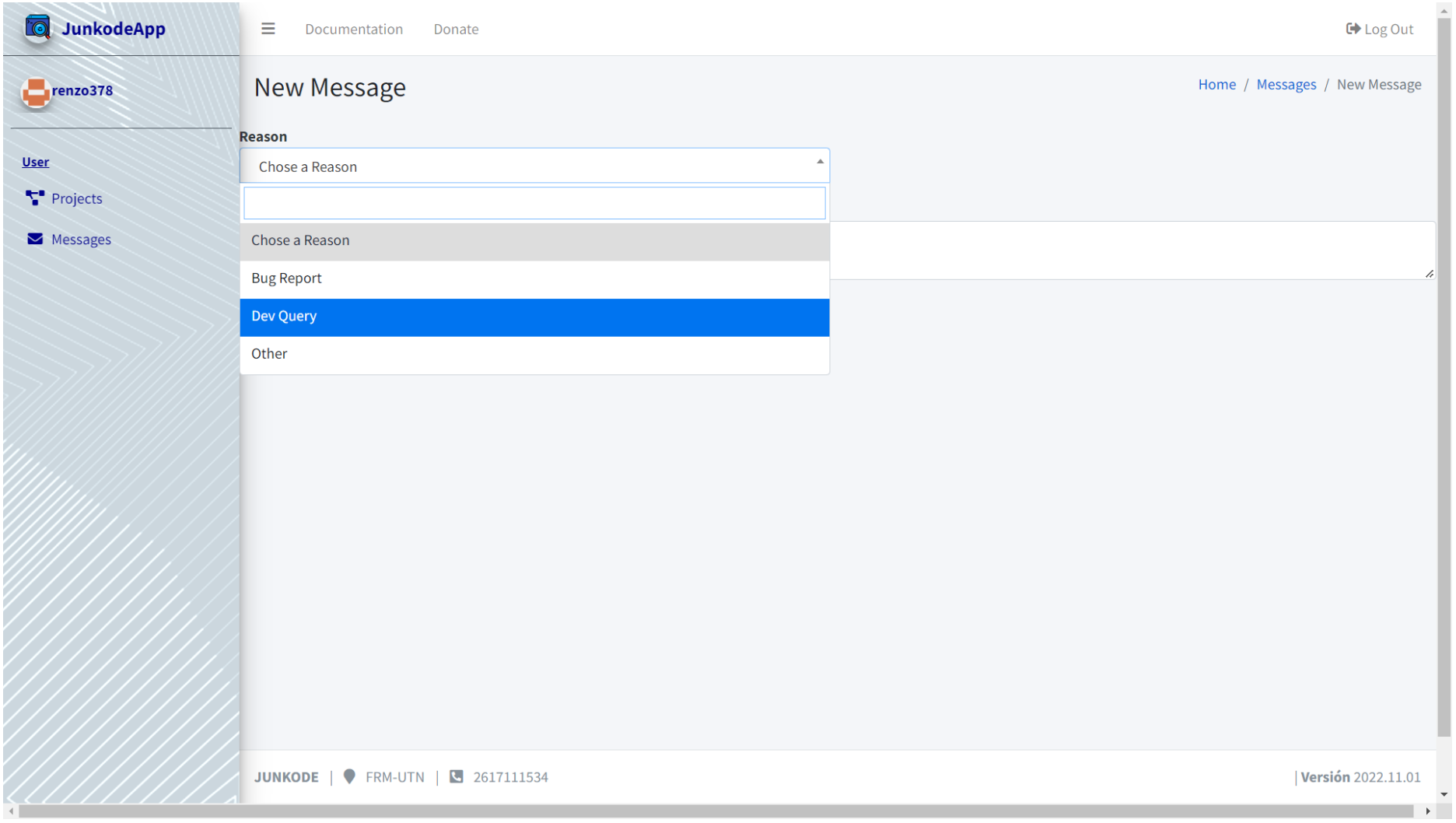
Pantalla N°18.



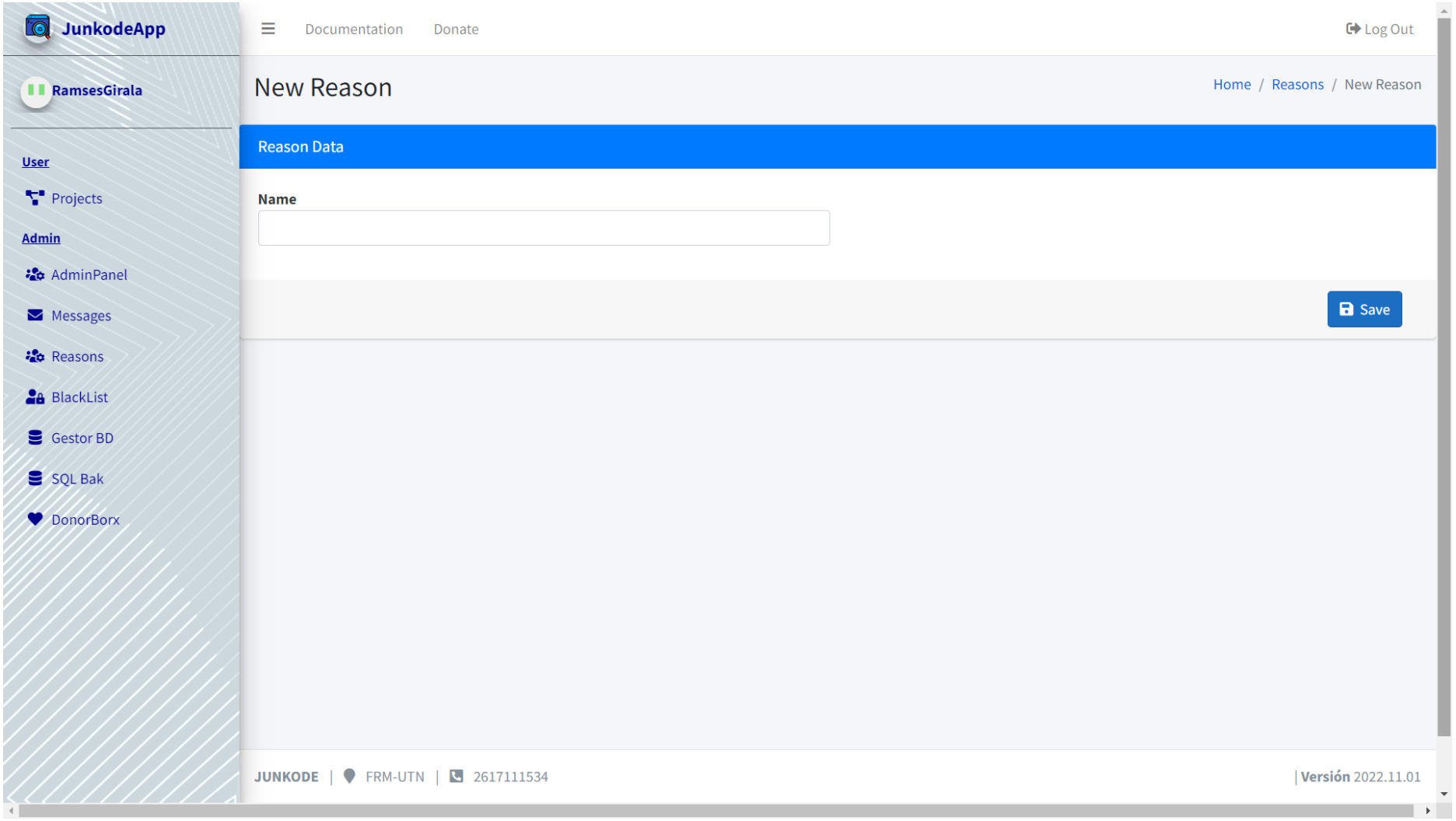
Pantalla N°19.



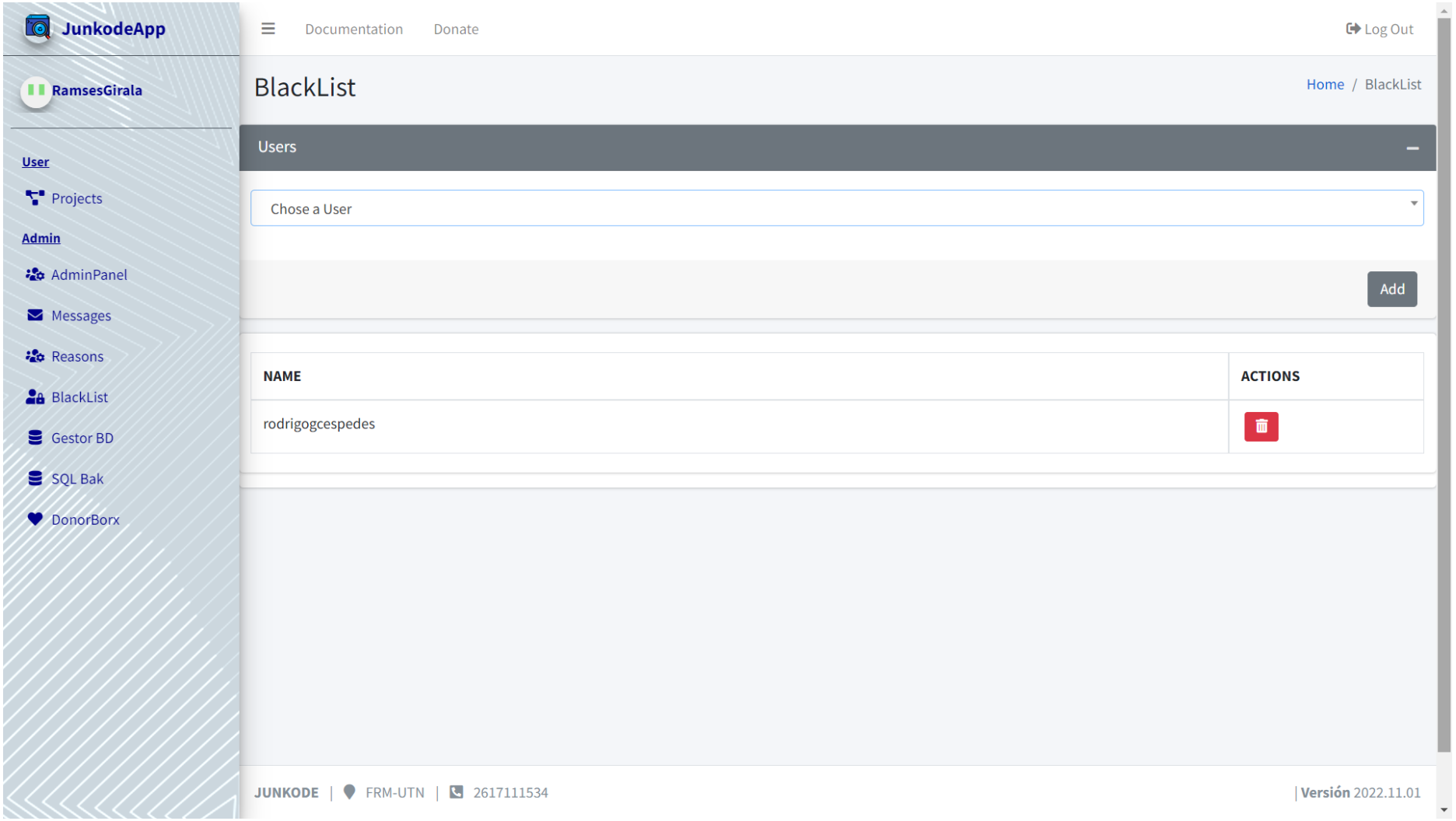
Pantalla N°20.



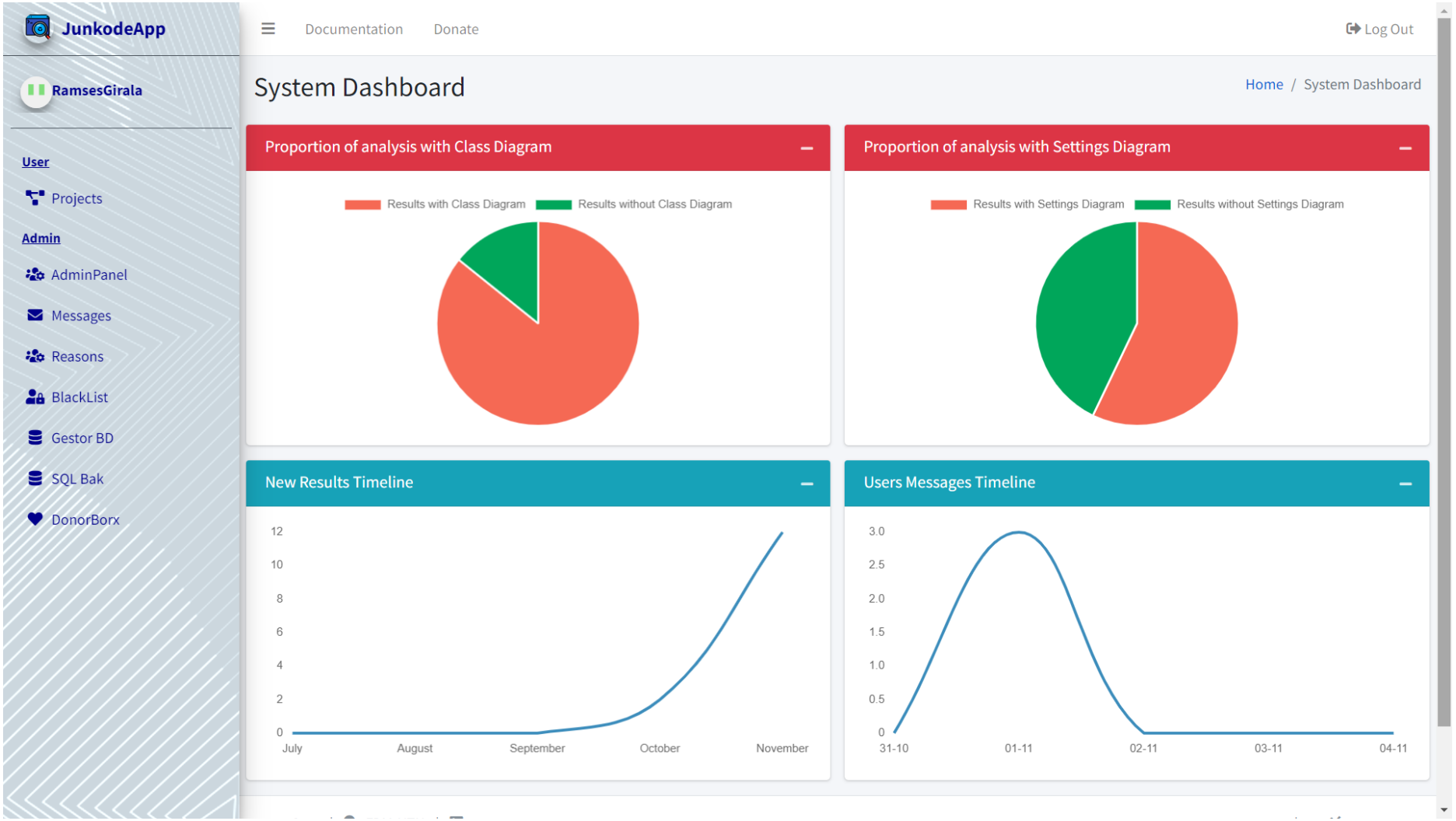
Pantalla N°21.



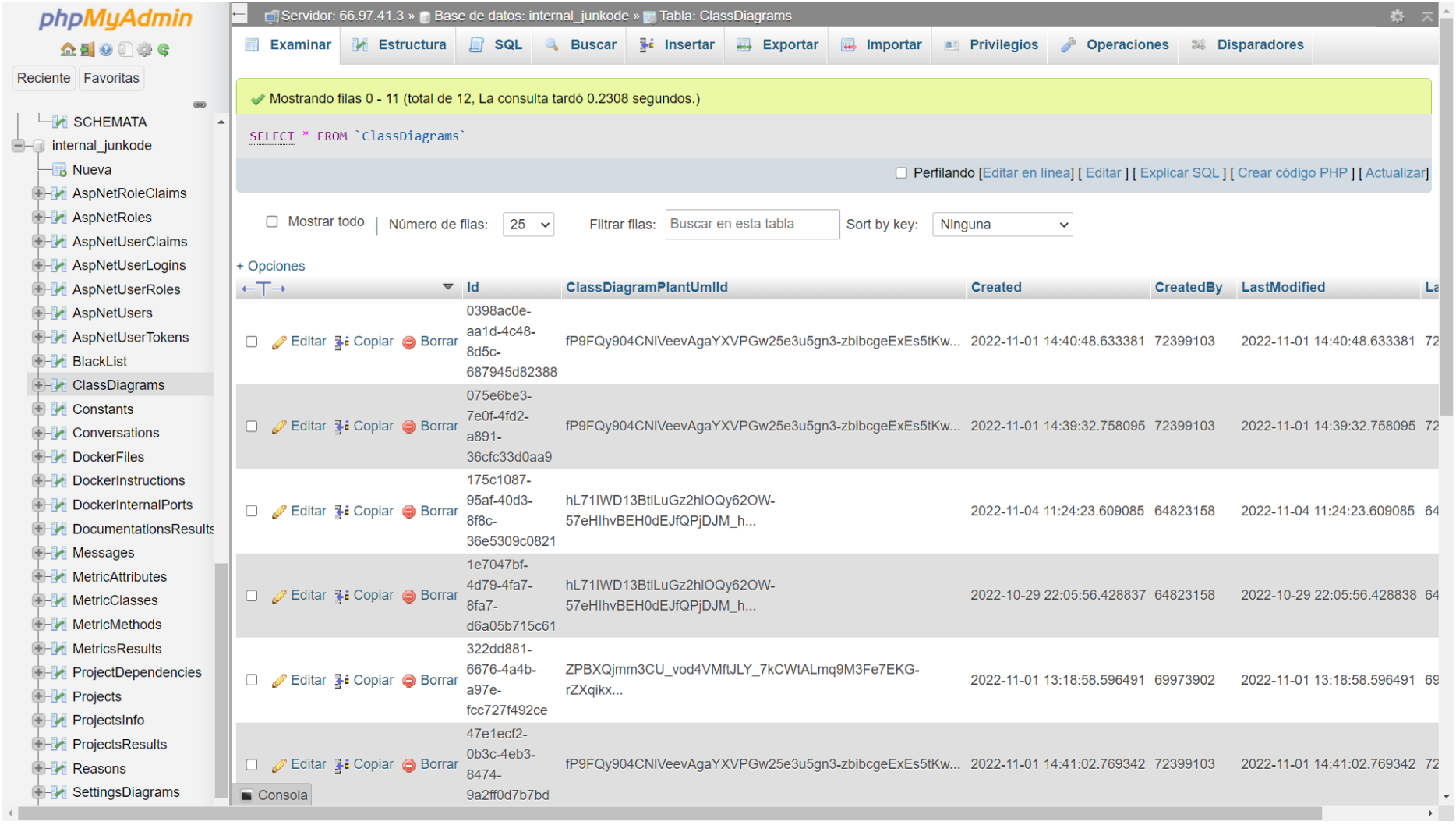
Pantalla N°22.



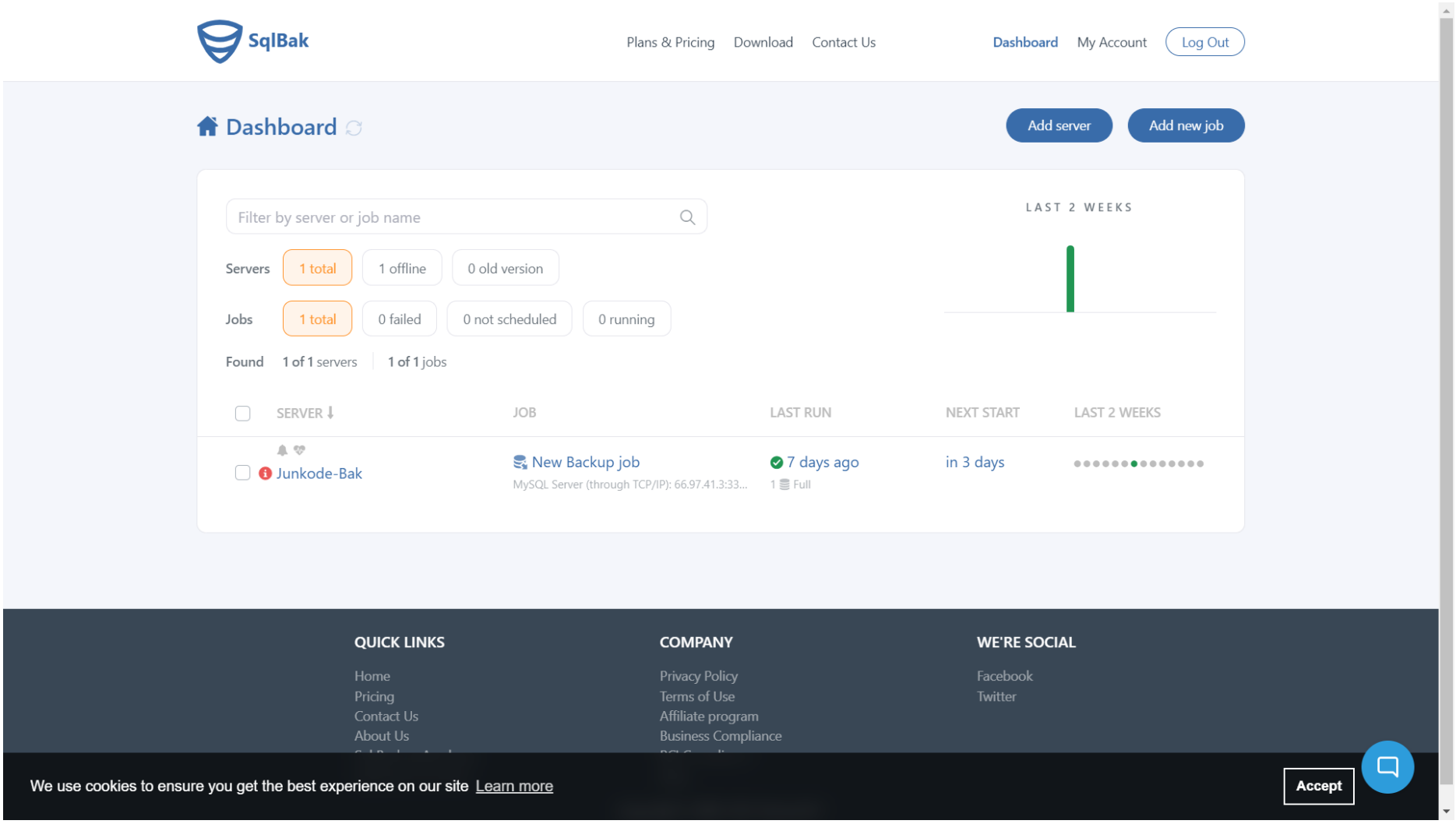
Pantalla N°23.



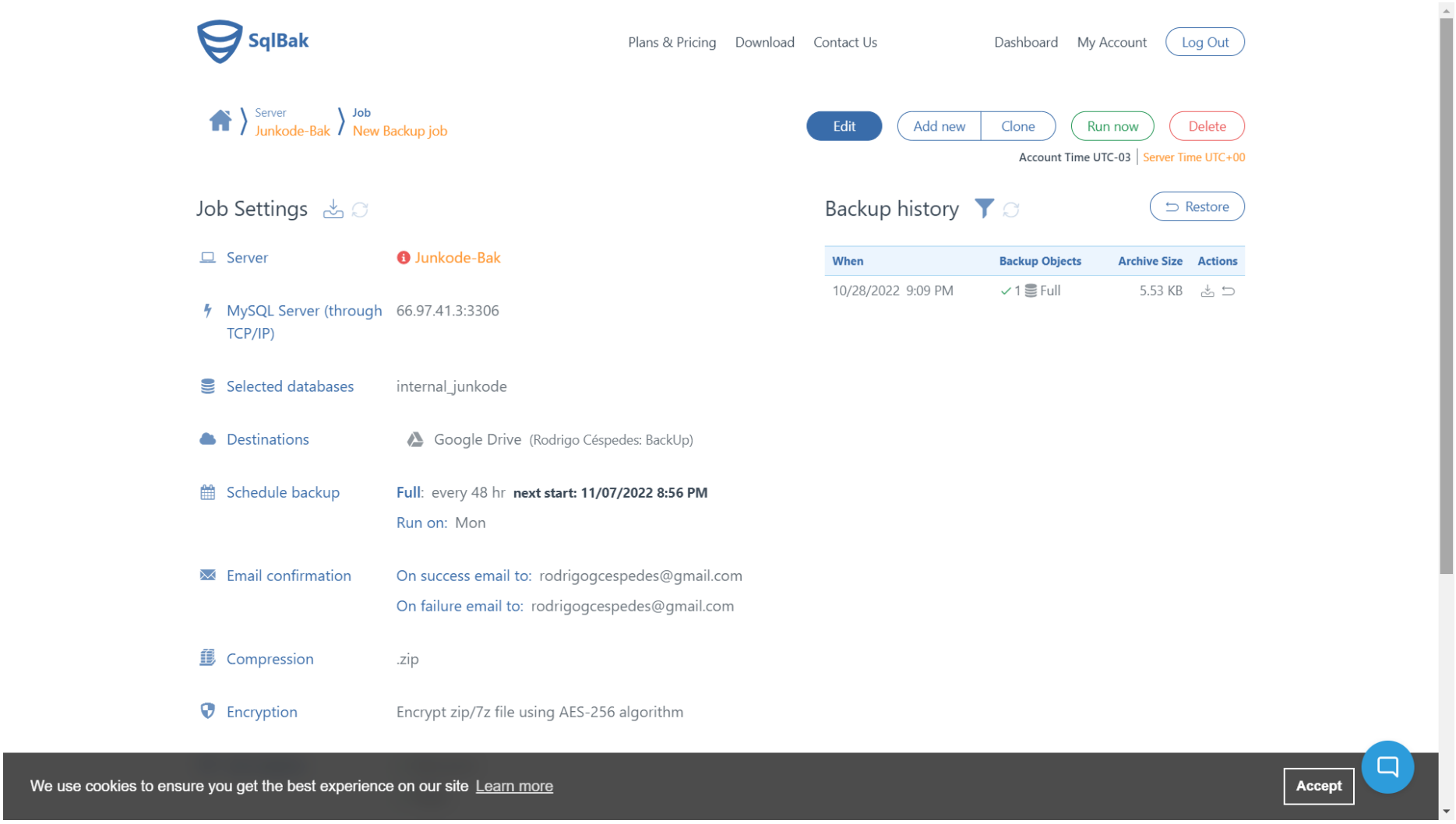
Pantalla N°24.



Pantalla N°25.



Pantalla N°26.



ANEXO N° 8 - GUÍA DE INICIO RÁPIDO

JUNKODE - Documentador Automático de Código Fuente

En esta Guía de Inicio Rápida se encuentran las funcionalidades que ofrece Junkode. Esta guía también puede ser encontrada en www.junkode.ga/Home/Guia .

1-Sesión:

Junkode permite a sus usuarios acceder a un set de funcionalidades limitadas aún si no ha iniciado sesión con una cuenta (carga de proyecto local, análisis y consulta de resultados).

Para poder utilizar todas las funcionalidades de Junkode, se debe disponer de una cuenta de Github, para poder iniciar sesión.

Las cuentas de Junkode están vinculadas con los Usuarios de GitHub, esto permite que los repositorios puedan ser accedidos de manera confiable y segura.

La sesión puede iniciarse desde el botón “Log In” de la página principal de Junkode, o desde

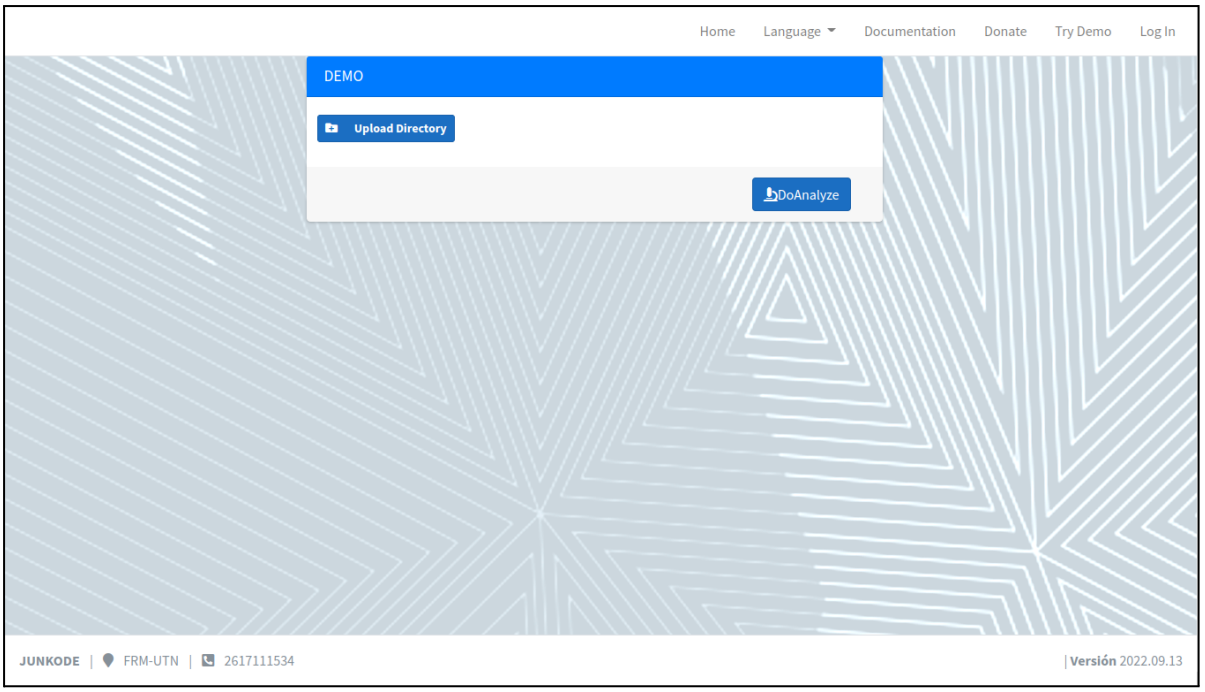
2-Carga del código.

La carga del proyecto Java se puede realizar de dos formas distintas.

Carga de proyecto local.

Desde la página Home de Junkode se debe seleccionar la opción “Try Demo” de la barra superior. Luego se debe seleccionar “Upload Directory” y seleccionar el directorio que contenga el proyecto Java a ser analizado.

Es importante resaltar que el directorio a ser cargado debe contener el proyecto Java completo, y no un subdirectorio de este. Esto con el fin de obtener los resultados completos del proyecto y no solo de un paquete, o no incluir los archivos de propiedades o gestión de dependencias, por ejemplo.

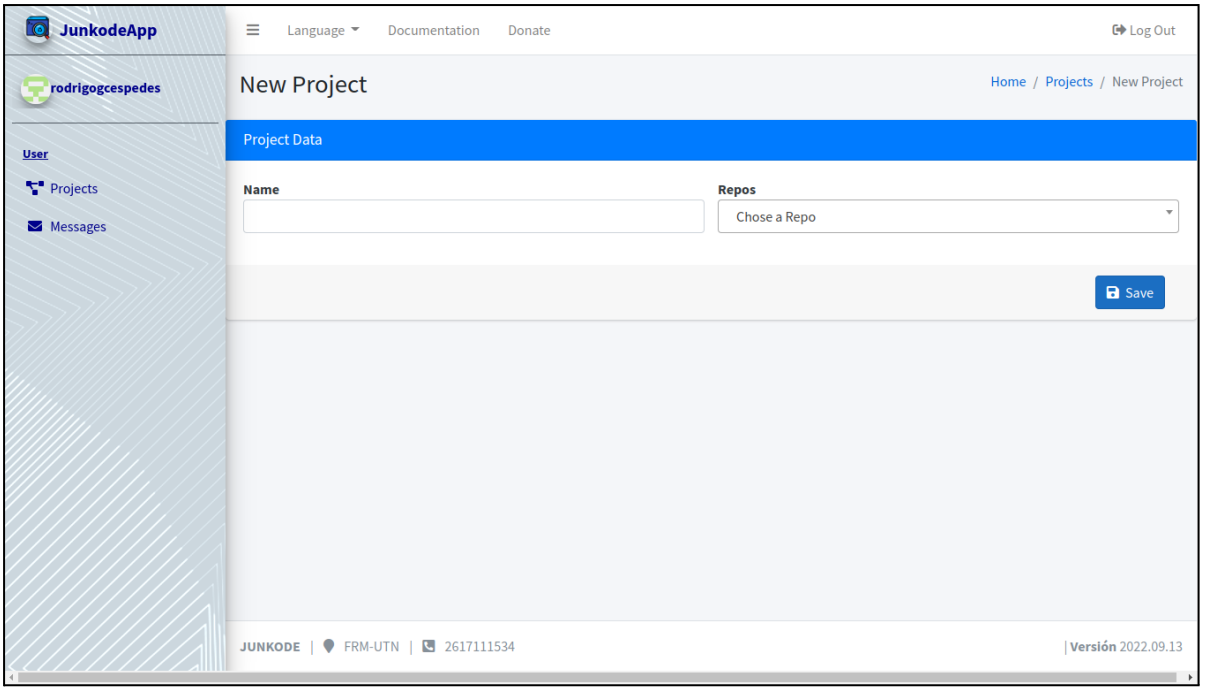


Una vez que el proyecto haya sido cargado, se debe seleccionar “DoAnalyze” y esperar unos segundos para ver los resultados.

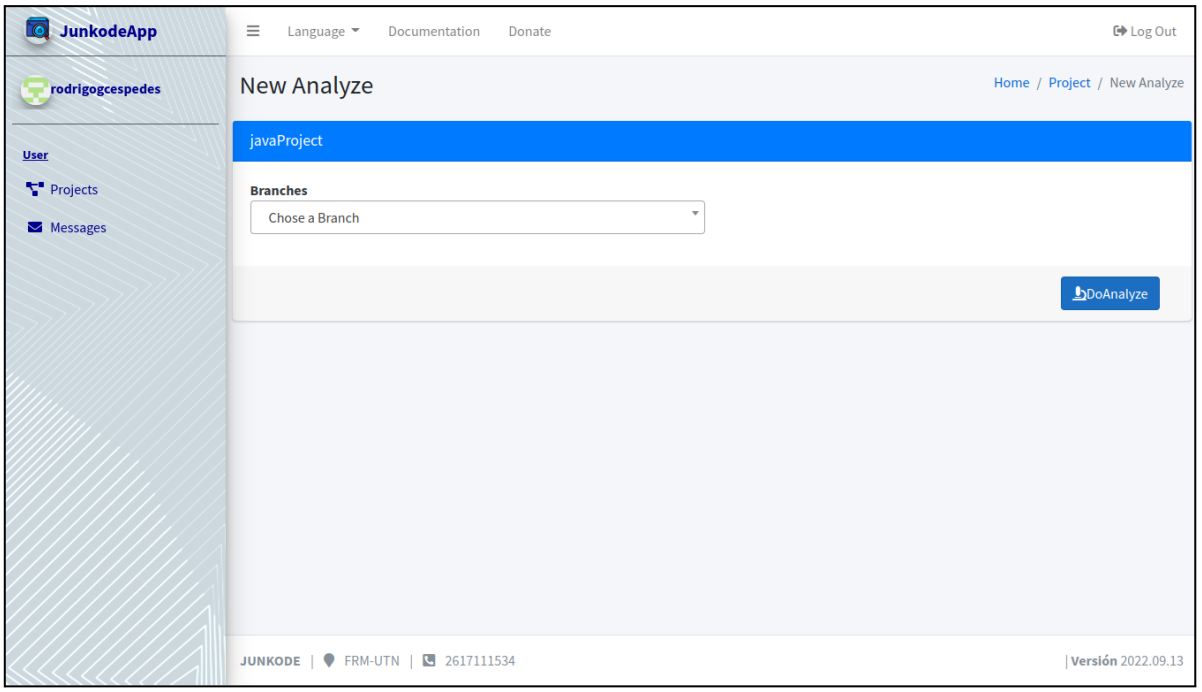
Carga de proyecto desde Github.

Para esto es necesario que el proyecto se encuentre en un repositorio de Github al que el usuario tenga acceso. Estando logueado con esa cuenta, se debe seleccionar “New Project” en la pantalla principal de proyectos.

Luego se debe ingresar el nombre del proyecto, seleccionar el repositorio y presionar en “Save”.



Luego se debe seleccionar la opción de visualizar del proyecto recién creado para ver su historial de resultados y desde ahí hacer click en “New Result”. Finalmente, se selecciona una rama a analizar, se presiona en “DoAnalyze” y se espera unos segundos para ver los resultados.



3- Análisis y resultados.

En el análisis de código que ofrece Junkode pueden distinguirse dos categorías, un análisis de métricas y un análisis de documentación.

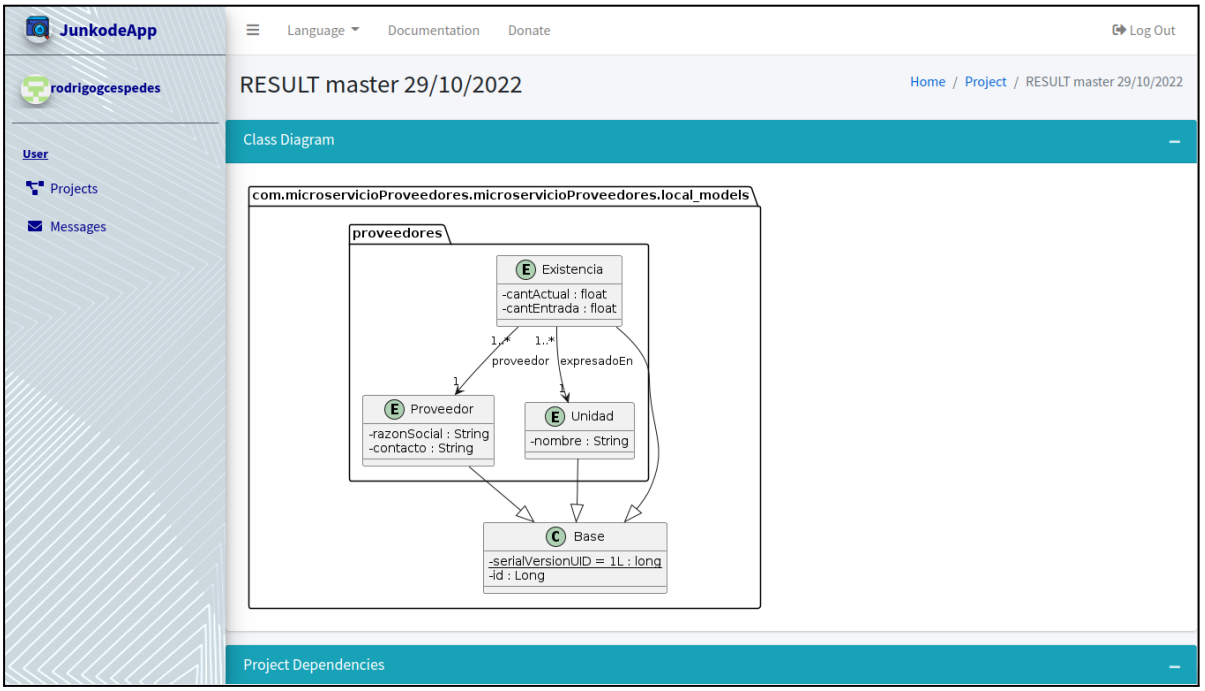
Es importante aclarar que ambos tipos de análisis tienen elementos que sólo se generan si ciertos archivos están presentes en el proyecto analizado.

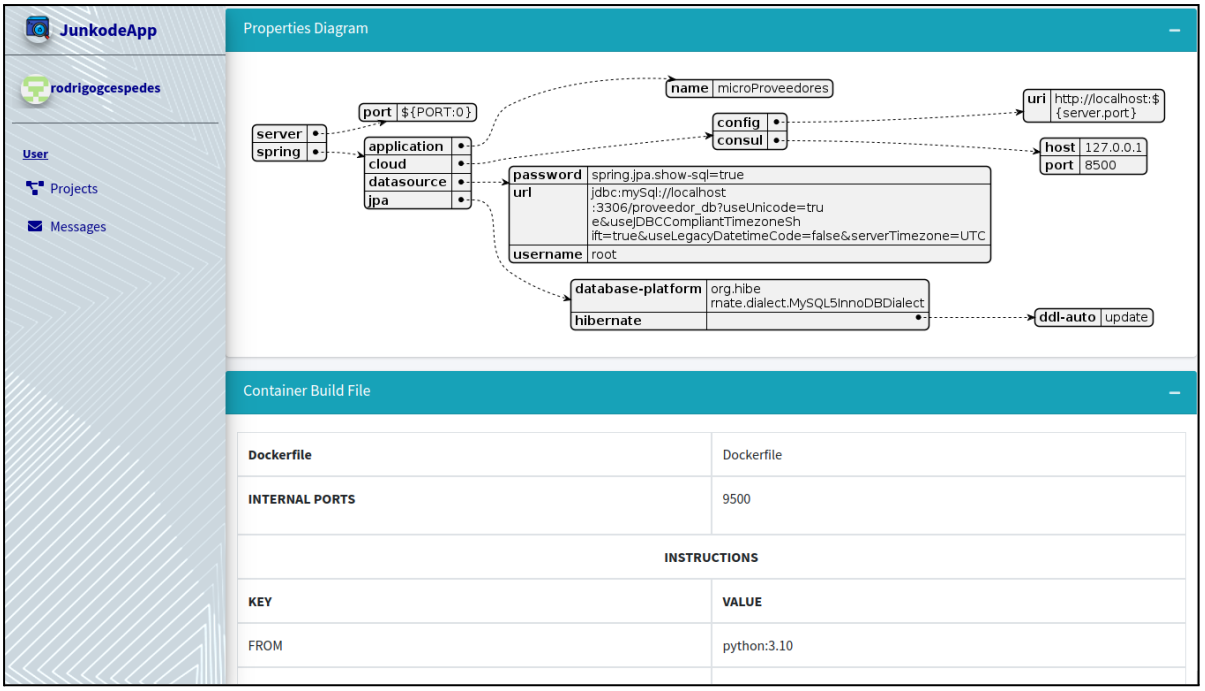
Al finalizar un análisis, se pueden obtener los siguientes resultados.

Análisis de métricas:

- Cohesión
- Acoplamiento.
- Profundidad en árbol de herencia.
- Número de hijos.
- Complejidad ciclomática.
- Tamaño de métodos.
- Tamaño de clases.
- Longitud de nombre de atributos.

Muchas de estas métricas son para elementos distintos (Clases, métodos o atributos), desde la interfaz se puede navegar entre las distintas clases para ver los resultados específicos de cada uno de estos elementos.





Análisis de documentación:

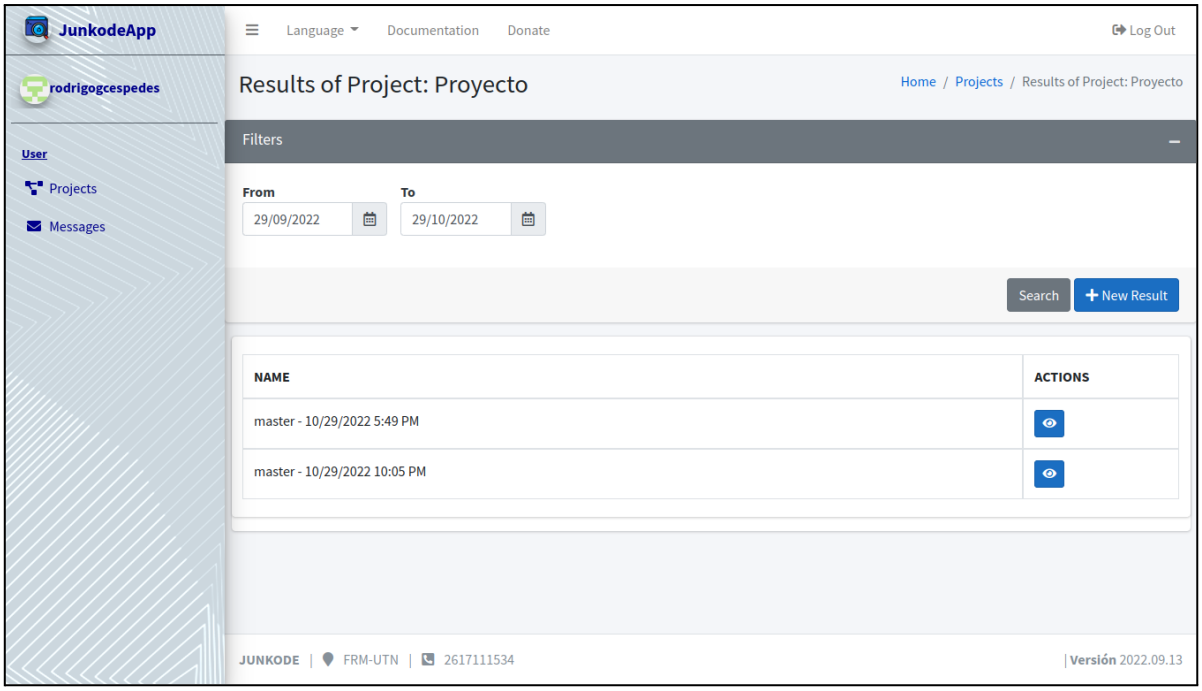
- ➢ Diagrama de clases.
- ➢ Diagrama de propiedades.
- ➢ Listado de constantes del sistema.
- ➢ Información sobre contenedores.
- ➢ Información sobre los gestores de dependencias.

Los resultados generados por los análisis se podrán descargar como archivos .pdf.

4- Historial de los análisis realizados:

Al realizar el análisis de un repositorio, se guarda el resultado del mismo con información de la fecha, hora y rama a la que corresponden.

Cualquier nuevo análisis del mismo proyecto se guardará automáticamente también.



Al seleccionar un proyecto, aparecerán todos los resultados (ordenados por fecha de creación) de los que ese proyecto dispone.

Allí se puede seleccionar la opción de visualizar cada uno de esos resultados, y desde ellos también se puede descargar su archivo .pdf.

ANEXO N° 9 - MANUAL DE USUARIO DE JUNKODE

JUNKODE - Documentador Automático de Código Fuente

JUNKODE

Documentador Automático de Código Fuente

Manual de Usuario
Nov. 2022

Manual de Usuario

En el siguiente manual se detallan las funcionalidades que ofrece al público la herramienta Junkode.

1.Introducción:

Junkode es una herramienta open-source que puede realizar análisis de código Java 8 en adelante para obtener reportes con documentación y métricas del mismo código de manera automática.

Esta se encuentra desarrollado con ANTLRv4 en .NET, (C#) y desplegado en la nube en Kubernetes, Junkode es una herramienta perfecta para la customización y análisis de código fuente por parte de cualquier desarrollador interesado. Este es el motivo que llevó a que se tomara la decisión de desarrollar Junkode como una herramienta libre y abierta que pueda aportar una disminución notoria del tiempo implicado en el desarrollo de documentación de código fuente y, también, una oferta para el análisis de calidad y control evolutivo del mismo.

2.Funcionalidades de análisis:

Junkode ofrece variedad de artefactos de documentación como listas, tablas, diagramas de clases y diagramas de propiedades. Estos pueden discriminarse en dos grupos en base al aspecto del proyecto en el que se enfocan a analizar, estos aspectos son los referidos al Análisis de Documentación y al Análisis de Métricas.

2.1.Análisis de Documentación:

Junkode genera documentación automáticamente de proyectos Java 1.8. A continuación se enuncian y detallan los artefactos generados:

- Diagrama de clases: (Fig. 2.1.a) Este diagrama muestra la relaciones entre todas las entidades persistentes del proyecto analizado. También muestra las clases no persistentes (que no son entidades) con las que las entidades se relacionan. Esto se hace por medio del análisis de las anotaciones de las clases. Este diagrama solo se genera si el proyecto implementa el framework Spring Boot o Micronaut.

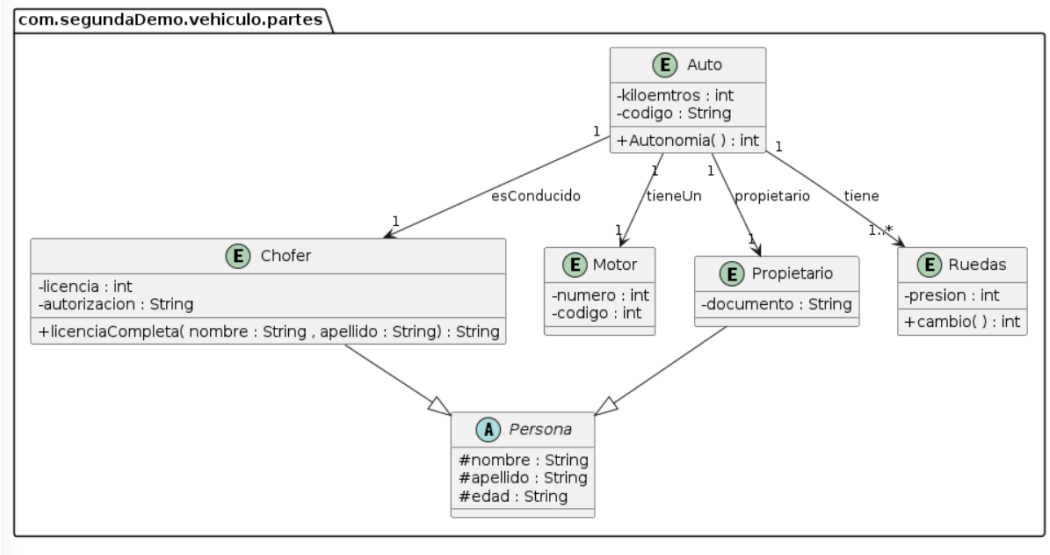


Figura 2.1.a: Ejemplo de diagrama de Clases generado por Junkode.

➢ Diagrama de propiedades: (Fig. 2.1.b) Este diagrama es la representación gráfica del archivo de propiedades del proyecto. Permite apreciar mejor la información referida a las bases de datos del proyecto e información de las conexiones de este con otros servicios de infraestructura como los pueden ser discoverys, config servers o gateways.

Este se genera si el proyecto cuenta con un archivo:

- application.properties
- application.yaml
- application.yml
- bootstrap.yaml
- bootstrap.yml

Si se encuentra más de uno de estos archivos, ocurrirá una excepción que indicará que solo se pueden analizar proyectos cuando tengan un solo archivo de propiedades. Esto impide que los usuarios puedan analizar un repositorio con las propiedades de múltiples microservicios, como el caso de un config server.

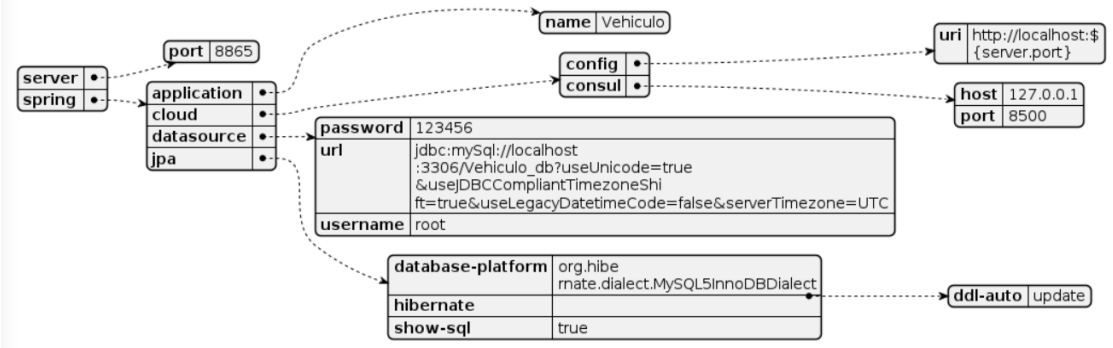


Figura 2.1.b: Ejemplo de diagrama de Propiedades generado por Junkode.

➢ Listado de dependencias: (Fig. 2.1.c) Este es un listado con todas las dependencias del proyecto con sus respectivas versiones. Funciona con proyectos que utilicen Maven o Gradle para gestionar sus dependencias.

Este se genera si el proyecto cuenta con un archivo:

- pom.xml
- build.gradle
- settings.gradle

Si se encuentra más de uno de estos archivos, ocurrirá una excepción que indicará que solo se pueden analizar proyectos cuando cuenten con un solo gestor de dependencias (Para los proyectos Gradle, es opcional que exista el archivo settings.gradle).

Adicionalmente, también se listarán los siguientes datos relativos al proyecto cuando se los encuentre en los archivos previamente mencionados: Project Name, Group Id, Artifact Id, Version y Description.

Framework	SpringBoot	
Dependency Gestor	Gradle	
Name	'microservicios-facturas'	
Group Id	'com.example.facturas'	
Artifact Id	'microservicios-facturas'	
Version	'0.0.1-SNAPSHOT'	
Dependencies		
Group Id	Artifact Id	Version
'org.springframework.cloud	spring-cloud-starter-netflix-eureka-client'	no especificada
'org.projectlombok	lombok'	no especificada
'mysql	mysql-connector-java'	no especificada
'com.google.code.gson	gson'	no especificada
'org.springframework.boot	spring-boot-starter-web'	no especificada
'org.springframework.cloud	spring-cloud-starter-openfeign	no especificada
'org.springframework.boot	spring-boot-starter-test'	no especificada
'org.springframework.boot	spring-boot-starter-data-jpa'	no especificada
'io.springfox	springfox-swagger2	2.9.2'
'io.springfox	springfox-swagger-ui	2.9.2'

Figura 2.1.c: Ejemplo de listado de dependencias generado por Junkode.

➢ Información de Docker build: (Fig. 2.1.d) Este artefacto es una tabla de dos columnas que se conforma por la instrucción primitiva de docker (en la primera columna) y el “valor” que se le aporta (en la segunda columna).

Este artefacto solo se genera si se encuentra un archivo Dockerfile.
Por motivos de comodidad, en la cabecera de la tabla también se enuncian los puertos que la imagen expone y la imagen madre de la que parte (Ambos elementos sólo figuran en la tabla en el caso de que el archivo Dockerfile cuente con ellos).

BASE IMAGE	microsoft/nanoserver
INTERNAL PORTS	80
INSTRUCTIONS	
KEY	**VALUE**
SHELL	["powershell","-command"]
RUN	New-Item -ItemType Directory C:\Example
ADD	Execute-MyCmdlet.ps1 c:\example\
RUN	c:\example\Execute-MyCmdlet -sample 'hello world'

Figura 2.1.d: Ejemplo de información de Docker build file capturada por Junkode.

2.2.Análisis de Métricas:

Para el análisis de métricas Junkode utiliza criterios de evaluación propios de UML. Esto implica que los resultados de los análisis tendrán mayor importancia para el proyecto si este sigue una metodología de desarrollo tradicional y se apega al paradigma de programación orientada a objetos más que a otros paradigmas. Este es el motivo por el que Junkode generalmente brinda resultados de métricas más certeros con proyectos de Java 1.8.

Junkode dispuso una tabla de 4 niveles en los que se clasificará el desempeño del código fuente para cada métrica:

1. Correcto: Representado con el color blanco.
2. Regular: Representado con el color amarillo.
3. Malo: Representado con el color naranja.
4. Inaceptable: Representado con el color rojo.

Las métricas que Junkode evalúa son las siguientes:

2.2.1.Métricas para las Clases:

➢ Cohesión: Es el grado en el que una función, componente o clase realiza únicamente la tarea para la cual fue diseñada. Se habla de cohesión alta cuando la relación la relación entre una clase y una funcionalidad es unívoca. Por el contrario, se habla de una cohesión baja cuando existe relación con múltiples funcionalidades del sistema.
- Correcto: Si es mayor o igual a 0,7.
- Regular: Si es menor a 0,7 y mayor o igual a 0,5.
- Malo: Si es menor a 0,5 y mayor o igual a 0,3.
- Inaceptable: Si es menor a 0,3.

➢ Acoplamiento: Grado de dependencia entre los componentes o clases entre sí. El acoplamiento es la medida que define qué tanto un componente o clase dependen de otro, generando cambios externos o alterando la funcionalidad del mismo. Se habla de acoplamiento bajo cuando existe una independencia entre los componentes entre sí, por el contrario un alto acoplamiento es cuando se tiene varias dependencias relacionadas a un solo componente.

- Correcto: Si es menor o igual a 3.
- Regular: Si es mayor a 3 y menor o igual a 6.
- Inaceptable: Si es mayor a 6.

➢ Profundidad en árbol de herencia: Es el número que indica la cantidad de clases de la que una clase hereda por recursión. En java, una clase solo puede ser subclase directa de una superclase, pero esta superclase podría ser subclase de otra, y por lo tanto, la primera subclase ya estaría heredando de dos superclases y le corespondería estar en la tercera generación del árbol de profundidad.
- Correcto: Si es menor o igual a 2.
- Regular: Si es mayor a 2 y menor o igual a 3.
- Malo: Si es mayor a 3 y menor o igual a 4.
- Inaceptable: Si es mayor a 4.

➢ Número de hijos: Es la cantidad de subclases que heredan de la clase. Es conveniente que si este número es mayor a dos, la superclase sea abstracta.
- Correcto: Si es menor o igual a 3.
- Regular: Si es mayor a 3 y menor o igual a 4.
- Malo: Si es mayor a 4 y menor o igual a 6.
- Inaceptable: Si es mayor a 6.

➢ Tamaño de clases: Cantidad de líneas netas (sin líneas vacías ni comentarios) del archivo .class analizado. Es recomendable mantener las clases lo más pequeñas posibles para facilitar su comprensión y mantenimiento.
- Correcto: Si es menor o igual a 60.
- Regular: Si es mayor a 60 y menor o igual a 90.
- Malo: Si es mayor a 90 y menor o igual a 150.
- Inaceptable: Si es mayor a 150.

2.2.2.Métricas para los Atributos:

➢ Longitud de nombre de atributos: La longitud de los nombre de los atributos es algo fundamental, puesto que estos pueden ser demasiado cortos (impidiendo que se los identifique adecuadamente y dificultado su interpretación durante el mantenimiento) o demasiado largos (por motivos prácticos, nombres largos no son solo engorrosos, sino que también puede dar problemas para ser interpretados por los compiladores).
- Correcto: Si es menor o igual a 13 y mayor o igual a 5.
- Regular: Si es menor o igual a 20 y mayor a 13. O si es menor a 5 y mayor o igual a 3.
- Inaceptable: Si es menor a 3 y mayor a 20.

2.2.3.Métricas para los Métodos:

➢ Complejidad ciclomática: Esta métrica define el número de caminos independientes dentro de un fragmento de código para proporcionar una medición cuantitativa de la complejidad lógica de un programa.
- Correcto: Si es menor o igual a 2.
- Regular: Si es mayor a 2 y menor o igual a 4.
- Malo: Si es mayor a 4 y menor o igual a 7.
- Inaceptable: Si es mayor a 7.

- Tamaño de métodos: Cantidad de líneas netas (sin líneas vacías ni comentarios) del método analizado. Es recomendable mantener los métodos lo más específicos posibles para facilitar su comprensión y mantenimiento.
  - Correcto: Si es menor o igual a 10.
  - Regular: Si es mayor a 10 y menor o igual a 20.
  - Malo: Si es mayor a 20 y menor o igual a 30.
  - Inaceptable: Si es mayor a 30.

3.Formas de uso:

3.1.Usuario de GitHub:

Junkode permite a los usuarios conectarse con sus cuentas de GitHub, para así poder acceder y manipular con mayor facilidad sus repositorios.

El usuario puede registrarse desde la página Home, donde un aviso Pop Up se abrirá y solicitará las credenciales de la cuenta de GitHub (Así como también los términos y condiciones de uso del Autenticador, que se aceptan al ingresar las credenciales). Una vez ingresada la información para el log in, el usuario podrá acceder a su cuenta de Junkode, directamente conectada a la sesión de GitHub.

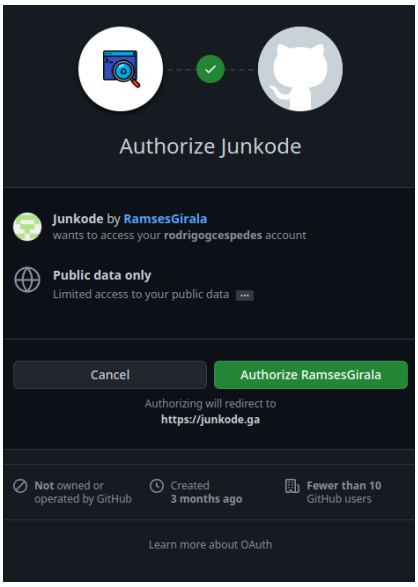


Figura 3.1.a: Interfaz de log in de Oauth provista por GitHub.

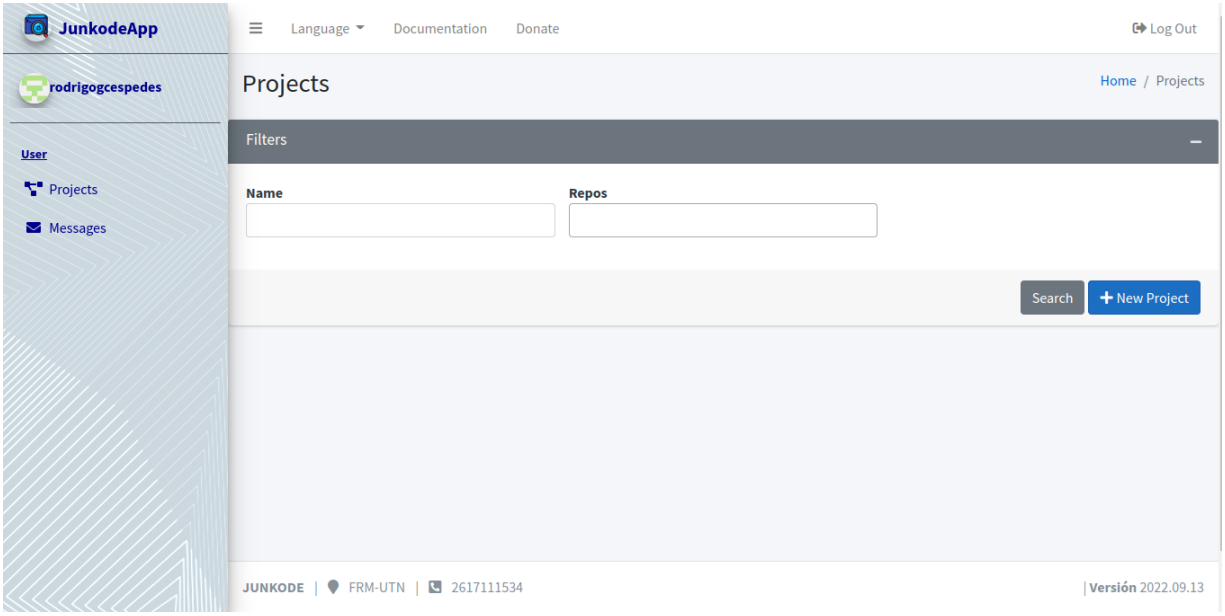


Figura 3.1.b: Pantalla principal de usuario logueado.

Una vez con su sesión iniciada, el usuario puede crear nuevos proyectos de Junkode y vincularlos directamente a un repositorio al que él tenga acceso.

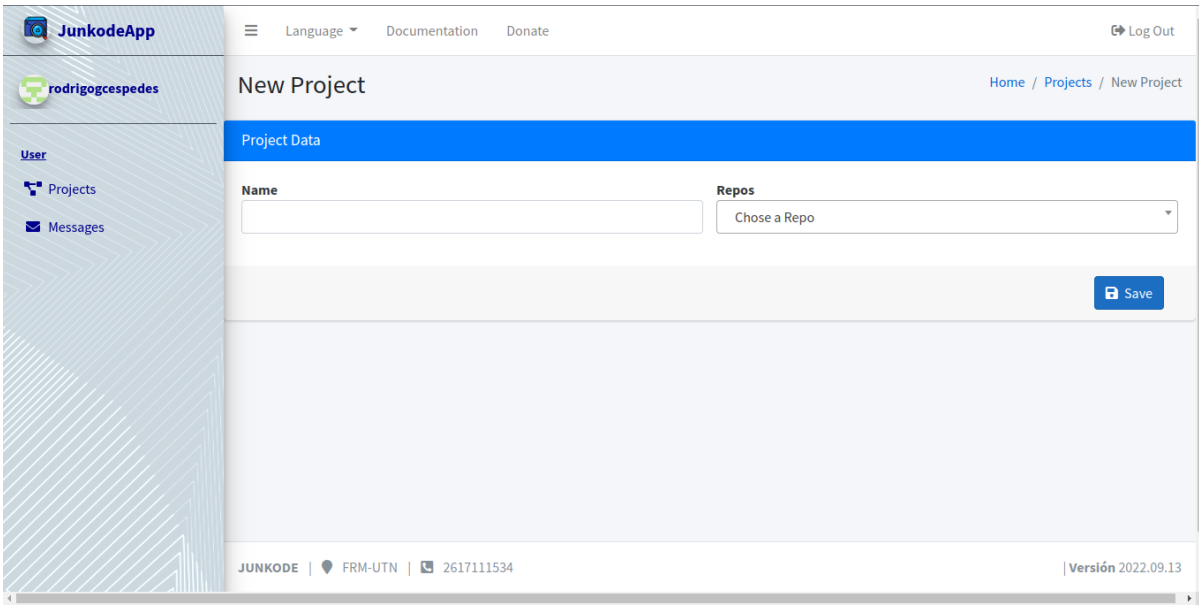


Figura 3.1.c: Pantalla de creación de un proyecto Junkode.

Para hacer un análisis, luego se debe seleccionar ese proyecto creado y seleccionar la rama que será analizada. Esto generará automáticamente los resultados del análisis. (Esta tarea puede tomar hasta unos minutos de tiempo para concluirse).

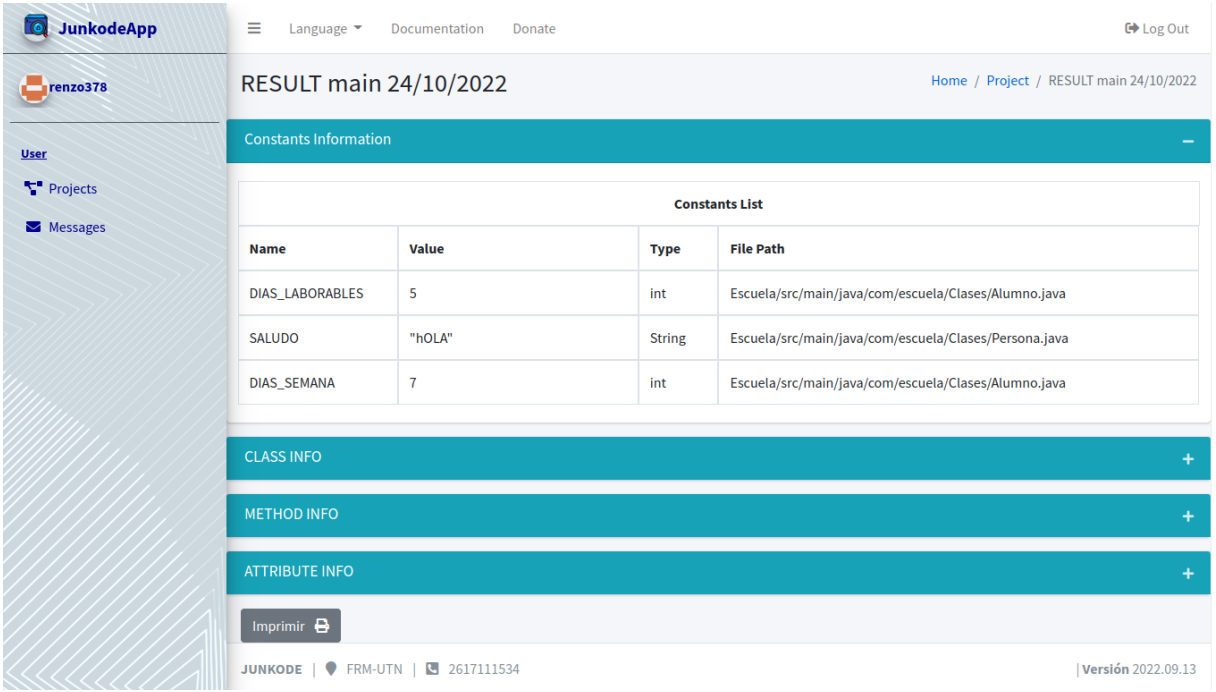


Figura 3.1.d:Pantalla de resultados del análisis.

Para acceder al historial de resultados de un proyecto, el usuario solo debe seleccionar ese proyecto. De esta manera puede acceder un listado de resultados de análisis de ese proyecto (o sea, del repositorio vinculado a esta) ordenados cronológicamente. Desde aquí puede filtrar estos resultados por ramas y fechas de generación. También puede consultarlos o eliminarlos.

Como detalle técnico, es necesario aclarar que Junkode se vale de la librería Octokit para el manejo de información de los repositorios y de Oauth para la autenticación y gestión de tokens de sesión.

También es importante aclarar que Junkode tendrá acceso a recolectar y almacenar información sobre el código fuente de los proyectos de GitHub que el usuario persista en Junkode. Esto incluye, no exclusivamente, metadata de generación de diagramas, métricas del código, versiones y disposición de ramas de un repositorio de GitHub.

Junkode podrá continuar almacenando información sobre un repositorio de GitHub incluso si el repositorio o la cuenta dueña de este son eliminados. Esta información podrá de igual manera ser consultada y/o eliminada por el usuario de Junkode que la generó cuando así lo disponga.

Los datos serán almacenados sólo luego de un proceso de serialización y encriptación para proteger la seguridad de los mismos.

3.1.1.Ventajas del usuario de GitHub:

- ➢ Se pueden almacenar los resultados de un análisis.
- ➢ Puede hacerse un seguimiento de la evolución del proyecto por un historial.
- ➢ Los resultados de los análisis pueden descargarse en formato .pdf.

- ➢ Se tiene acceso directo tanto a los repositorios directos del usuario de GitHub, como a los repositorios de terceros donde el usuario sea colaborador.

3.2.Usuario invitado:

El usuario invitado es el que no se registra para utilizar la herramienta. Junkode ofrece las funcionalidades completas de análisis para usuarios no registrados.

Para acceder a estas, el usuario debe contar con el proyecto Java a analizar en su dispositivo local.

Para acceder a la pantalla de análisis de invitado puede seleccionarse cualquiera de los enlaces que lo referencian en la página home.

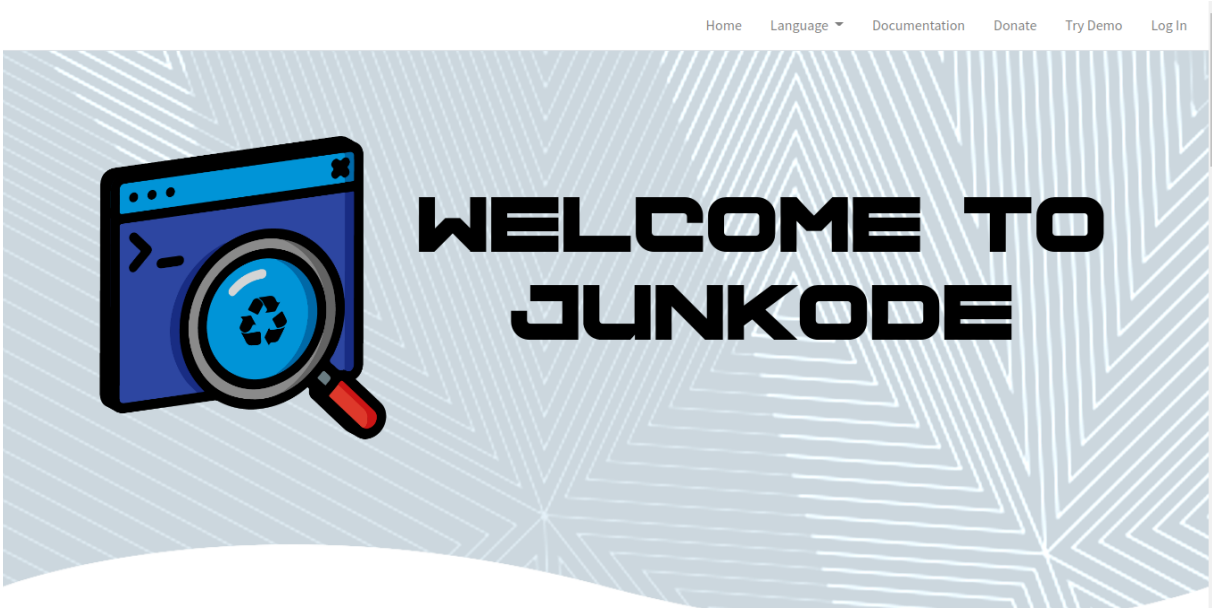


Figura 3.2.a: Pantalla Home de Junkode.

El usuario aquí debe seleccionar el directorio que contenga el proyecto Java completo para ejecutar el análisis. La carga y análisis del proyecto para este usuario es mucho más rápida, debido a que no hace falta buscar los archivos del proyecto a un repositorio de GitHub ni ejecutar instrucciones a una base de datos en la nube.

Una vez finalizado el análisis, se mostrarán las tarjetas con los resultados obtenidos. Sin embargo, el usuario invitado no tiene acceso a la funcionalidad de descarga de un archivo .pdf con los resultados del análisis, ni el acceso a que este se archivado en un historial.

3.1.1.Ventajas del usuario invitado:

- ➢ La carga de proyectos es considerablemente más rápida que la de los usuarios de GitHub.
- ➢ Pueden analizarse proyectos almacenados localmente, lo que permite a los usuarios analizar ramas locales de un repositorio, proyectos que se trabajen con gestores de

dependencias remotos distintos de GitHub y proyectos que no implementen gestión de versionado.

4.Historial de proyecto:

Los usuarios que se registren por GitHub pueden almacenar, en sus cuentas de Junkode, un historial con los resultados de los análisis de cada uno de sus proyectos ordenados de manera cronológica.

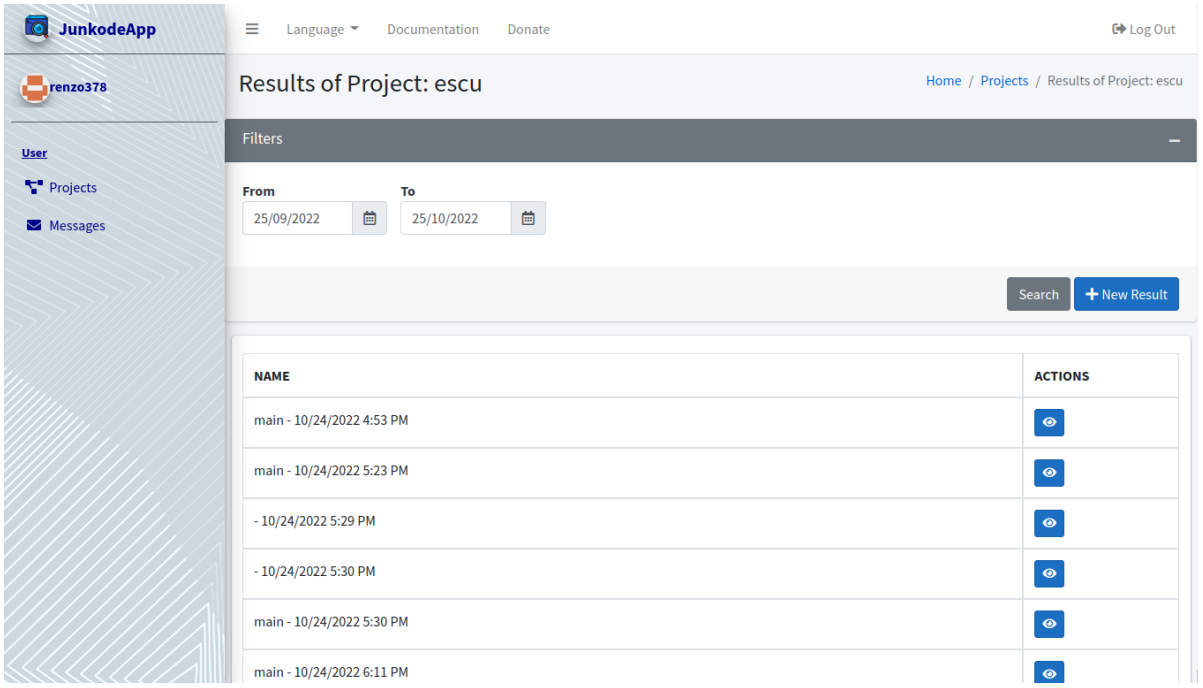


Figura 4.a: Historial de un proyecto de Junkode.

Esta es una funcionalidad en extremo útil, puesto que permite a los desarrolladores llevar un control sobre la evolución de las métricas y documentación de su código. Ejemplos de estos controles podrían ser comparar la cantidad de métodos con una complejidad ciclomática no admisible a lo largo del tiempo, o comparar listas de dependencias para analizar cambios de versiones o incorporación de nuevas librerías.

Momentáneamente, se conserva la funcionalidad de que los resultados de un historial puedan ser borrados (esto para ordenar de mejor manera los resultados y así contar con una vista más limpia y menos sobrecargada de los historiales), sin embargo no se confirma que esta funcionalidad continúe siendo accesible en futuras releases.

5.Reportes de análisis:

Los usuarios de Junkode que se conecten por medio de una cuenta de GitHub tendrán acceso a la funcionalidad de exportar los resultados de su análisis en un documento pdf.

Este documento se encontrará etiquetado con el nombre del proyecto de Junkode, la rama analizada y la fecha y hora del análisis.

Para facilitar la navegabilidad, se dispone de tres comodidades implementadas:

La primera es la incorporación de un índice con hipervínculos para que el usuario pueda conocer de antemano cuales son los elementos de documentación que fueron analizados en su proyecto (recordando que muchos elementos de documentación están condicionados a la existencia de ciertos elementos en el proyecto Java), y para que pueda dirigirse a ellos de manera rápida y sin necesidad de una búsqueda.

La segunda comodidad incorporada en los documentos generados son los hipervínculos de las tablas. Las tablas de análisis de métricas exponen los resultados para cada una de las clases del proyecto, pero también para cada uno de los métodos y atributos de estas. Por lo tanto, las tablas de clases fueron diseñadas para funcionar como una lista de redireccionamiento, donde se puede seleccionar cada clase de la lista para ser redireccionado directamente a las tablas de análisis de sus métodos y atributos.

Finalmente, la tercera comodidad incorporada es la forma tabular que se le dio a los resultados de análisis de métricas.

Ya se habló en el anterior punto de cómo esta forma tabular sirve para ordenar las tablas en una lista de redireccionamiento cómoda para los usuarios, pero el verdadero propósito de esta es permitir una mayor concentración de resultados por página del documento.

Junkode solo enseña en las tablas los valores numéricos del desempeño de cada clase, método y atributo para con cada métrica, junto con un código de color que permite vislumbrar rápidamente el estado de los resultados.

Esta simplificación es posible debido a que la explicación detallada de las métricas, junto con los significados de los colores se encuentra en una página previa, donde se aclaran los rangos de aceptación de cada métrica.

Esta configuración tabular de los resultados, permite que Junkode pueda generar análisis de proyectos monolíticos con cientos de clases, con cientos de métodos cada una, y aún así no generar documentos de más de 25 páginas.

UML Metrics Analysis

Nro	Package - Class	Cohesion	Coupling	InheritanceDepth	Size	InheritanceChildCount
1	com.escuela.Clases - Director	0.00	0	1	1	0
2	com.escuela.Clases - Profesor	1.00	1	1	4	0
3	com.escuela.Clases - Persona	0.50	0	0	6	3
4	com.escuela.Clases - alumno	1.00	4	1	17	0
5	com.escuela.demo - EscuelaApplication	0.00	0	0	2	0
6	com.escuela.Clases - materia	0.75	0	0	8	0
7	com.escuela.funciones - Complejidad	1.00	0	0	7	0
8	com.escuela.demo - EscuelaApplicationTests	0.00	0	0	1	0

com.escuela.Clases - Director

Nro	Attribute Name	Length
1	Antiguedad	10

Figura 5.a: Tabla de métricas de documento pdf de resultados de Junkode.

6.Servicios externos:

Junkode se vale de múltiples servicios externos para generar sus análisis. Es importante distinguirlos y conocerlos, para así poder saber cuando una falla puede ser propia del servicio de Junkode en sí, o de los servicios de los que este depende.

6.1.API Graficadora:

Esta es una API consta de dos partes (ambas desplegadas y hosteadas por el equipo de Junkode).

La primera es la API propiamente dicha, que recibe un objeto JSON con el texto en sintaxis PlantUML para graficar, y retorna otro objeto JSON con un enlace a la imagen cruda generada por el servicio graficador.

Esta API puede ser consultada por una imagen en formato PNG, SVG o texto ASCII, solicitando a las URIS “/png”, “/svg” y “/txt” respectivamente.

La segunda parte es el servicio graficador, el cual es un servicio dockerizado de PlantUML (una herramienta que convierte texto en diagramas). Este servicio está hosteado por Junkode, pero la imagen de docker es la de un servidor Tomcat provisto por PlantUML.

El servicio graficador recibe el texto de con la sintaxis del diagrama que se pretende graficar y devuelve una semilla de graficación, la cual siempre genera el mismo gráfico. Luego esta semilla es anexada al final de una URI para ser devuelta por la API graficadora.

Es importante resaltar que el usuario nunca tendrá interacción con el servicio graficador en sí, sino que desde el servicio principal de Junkode se enviará una solicitud a la API graficadora y esta es la que se comunicará con el servicio graficador.

Esta API es utilizada tanto para la generación de los diagramas de clases UML como para los diagramas de propiedades YAML.

6.2.API Conversora “app2yaml”:

Esta API es empleada para convertir las propiedades escritas en formato “application.properties” a formato YAML. Esto es necesario puesto que el servicio graficador solo genera el diagrama de propiedades si la entrada se encuentra en formato YAML.

Nuevamente, esta también consta de dos partes:

La primera (muy similar a la anterior) es la API propiamente dicha. Recibe un objeto JSON con el cuerpo del archivo “application.properties” y devuelve otro objeto JSON con las propiedades equivalentes escritas en formato YAML. Esta API se encuentra hosteada por Junkode.

La segunda parte es el servicio conversor en sí. Este es el único servicio de todo el sistema que no fue desplegado por el equipo de Junkode, sino que solamente es consumido. Se trata del servicio web JavaInUse (www.javainuse.com), el cual realiza la conversión de texto en el formato de propiedades al formato YAML.

La API conversora “app2yaml” solo es consultada cuando se encuentra un proyecto con un archivo “application.properties”. O sea, si el proyecto java a analizar maneja sus propiedades con un archivo en formato YAML (cualquiera sea el archivo) o si el proyecto no cuenta con un archivo de propiedades, este servicio no es requerido y, por lo tanto, no se lo consume.

7.Excepciones y advertencias:

Junkode maneja múltiples excepciones que pueden llegar a ocurrir durante la carga de datos o ejecución del análisis. Estas pueden ser de naturaleza preventiva (o sea, que realizan un preprocesamiento del código en busca de algo que pueda generar un error), o alternativa (o sea que el análisis se encuentra con un error durante su ejecución y toma medidas para tratarlo). En ambos casos, estas excepciones se manifiestan a través de advertencias para la interpretación del usuario.

A continuación se listan las excepciones del sistema, junto con sus respectivas advertencias:

1. Excepción por ausencia de código Java: Esta es una excepción que ocurre cuando se intenta cargar un repositorio o directorio que no contiene ningún archivo con sintaxis Java. Es una excepción de naturaleza preventiva que detiene la carga de datos mostrando la siguiente advertencia al usuario.

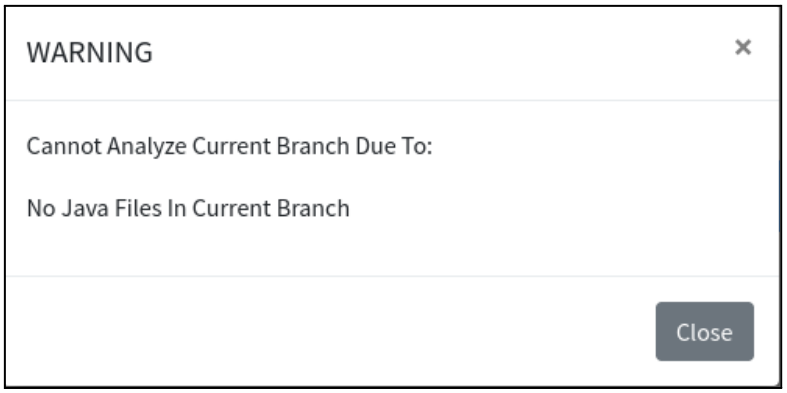


Figura 7.a: Advertencia por ausencia de código Java.

2. Excepción por múltiples gestores de dependencias: Esta es una excepción que se activa cuando se encuentran múltiples archivos de gestores de dependencias en una sola rama o directorio. Un proyecto solo puede contar con uno de los siguientes archivos simultáneamente, “pom.xml” y “build.gradle” (opcionalmente también puede contar con un archivo “settings.gradle”, o con ninguno de estos). Si más de uno de estos archivos es encontrado durante la carga de archivos del proyecto, esta se detiene y el usuario obtiene un cartel de advertencia.

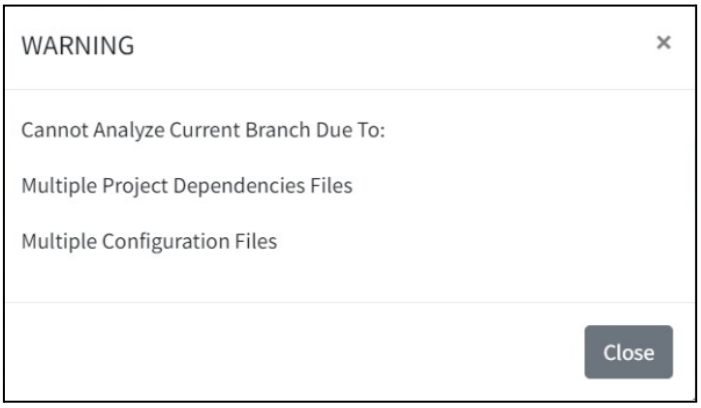


Figura 7.b: Advertencia por múltiples gestores de dependencias.

3. Excepción por múltiples gestores de propiedades: De la misma forma que la advertencia anterior, la carga de datos se detiene si se encuentra más de uno de los siguientes archivos (ya sea archivos iguales o distintos): “application.properties”, “application.yaml”, “application.yml”, “bootstrap.yaml” o “bootstrap.yml”. Es importante aclarar que los demás archivos YAML que el proyecto pueda contener serán ignorados debido a que este es un formato común para muchas herramientas.

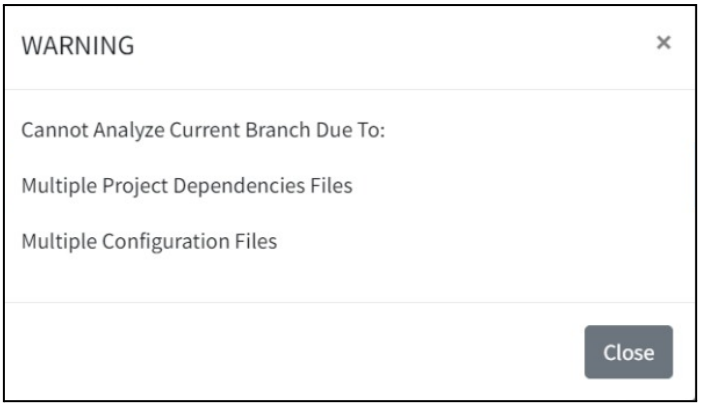


Figura 7.c: Advertencia por múltiples gestores de propiedades.

4. Excepción por error gramatical: Junkode funciona implementando analizadores de gramáticas regulares, por lo que es sensible a errores sintácticos. Durante el análisis del código, puede activarse esta excepción alternativa, la cual detiene el proceso de análisis y muestra un mensaje de advertencia al usuario, alertandolo de los errores sintácticos presentes en su proyecto. Esto no ocurre solo para los archivos con sintaxis Java, sino para todos los archivos con sintaxis susceptibles de ser analizadas por Junkode.

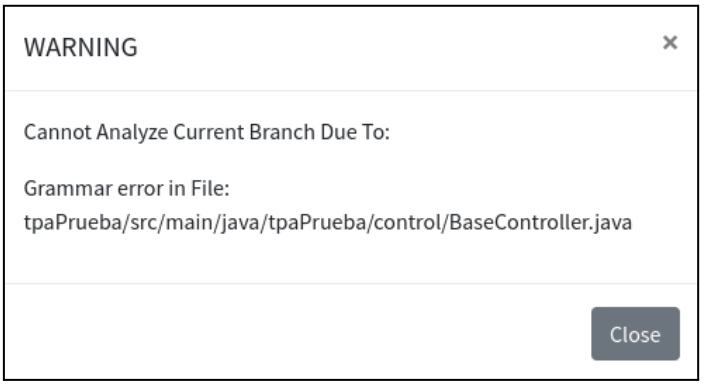


Figura 7.d: Advertencia por error gramatical.

5. Excepción por instrucción SQL: Hay muy pocos momentos en los que un usuario puede ingresar texto directamente a Junkode. Específicamente, solo puede hacerlo al nombrar y renombrar un proyecto y al escribir un mensaje por el sistema de mensajería interna. En cualquiera de esos casos, hay una acepción que se acciona cuando se encuentra algún elemento primitivo de instrucción SQL en el texto ingresado por el usuario (o sea, texto como “select”, “user”, “while”, “=”, etc.). Esta excepción tiene la finalidad de evitar que los usuarios puedan hacer inyección de SQL para acceder a información sensible del sistema o, aún peor, modificarla. La excepción se manifiesta en forma de cartel de advertencia, indicando que no se pueden incluir instrucciones SQL en el texto perceptible por el sistema.

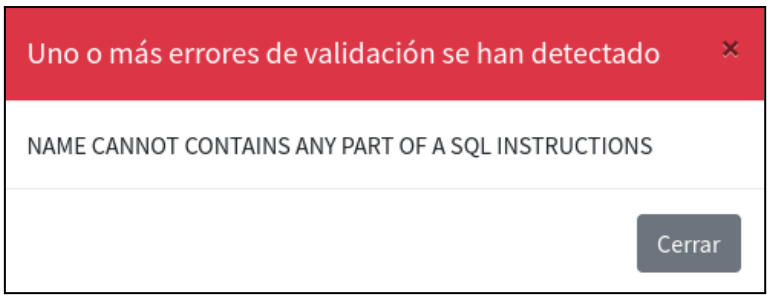


Figura 7.e: Advertencia por instrucción SQL.

6. Excepción por error sintáctico para el gráfico de propiedades: El analizador sintáctico de Junkode reconoce la sintaxis YAML y reporta los errores gramaticales en estos archivos según la excepción anterior. Sin embargo, PlantUML tiene su propio trato para los archivos YAML y tiene ciertas restricciones que limitan el prácticamente infinito conjunto de configuraciones de un archivo YAML. Dado que este análisis depende de PlantUML y no del equipo de Junkode, no se detiene el análisis, sino que ocurre una excepción alternativa que tiene como resultado que el diagrama de propiedades no se genere, sino que en su lugar se genere una imagen de advertencia para el usuario, indicando que la sintaxis de su archivo YAML no es correctamente reconocida por el servicio graficador de PlantUML.

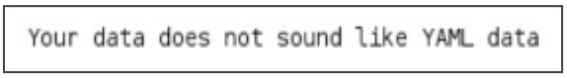


Figura 7.f: Advertencia por error sintáctico para el gráfico de propiedades.

8.Otras fuentes de información:

Para más información sobre el uso de la herramienta, puede consultar las guías de inicio rápido de la barra de navegación de la página Home de Junkode. En esta guía puede encontrar información paso a paso y videos instructivos para comenzar a utilizar Junkode rápidamente.

Para consultas sobre la terminología y vocabulario empleados en el manual, se encuentra a disposición un glosario con las definiciones de múltiples palabras técnicas para su mejor comprensión.

También puede contactar con el equipo de Junkode por medio del sistema de mensajería interno de la herramienta (se solicita estar registrado para esta funcionalidad), o al email organizacional de Junkode: junkode.team@gmail.com .

Finalmente, se puede consultar el canal de youtube oficial de Junkode (@Junkode) y su repositorio de GitHub (github.com/RamsesGirala/Junkode). En este último puede encontrarse todo el código fuente que le permite a Junkode brindar sus servicios a los usuarios. Como usuario, puede sentirse libre de hacer un fork del repositorio para experimentar con él y adaptar sus funcionalidades a su gusto.

Se ruega que se esté al pendiente de las nuevas actualizaciones que incrementen las funcionalidades brindadas por Junkode en el futuro, y a las actualizaciones de la documentación oficial de la herramienta.

Muchas gracias por utilizar Junkode.

ANEXO N° 10 - MANUAL DE ADMINISTRADORES DE JUNKODE

JUNKODE - Documentador Automático de Código Fuente

JUNKODE

Documentador Automático de Código Fuente

Manual de Administrador
Nov. 2022

Manual de Administrador

Este manual está destinado a los usuarios que cumplan el rol de administradores Junkode y tiene la finalidad de introducirlos a las funcionalidades específicas presentes para este rol.

Un usuario administrador de Junkode debe ingresar al sistema de la misma manera que lo hace un usuario común y también tiene acceso a las mismas funcionalidades a las que puede acceder un usuario común.

Además de las funcionalidades generales, el administrador tiene acceso a los implementos del sistema que figuran en su barra lateral y que serán descriptos a continuación por este manual de uso.

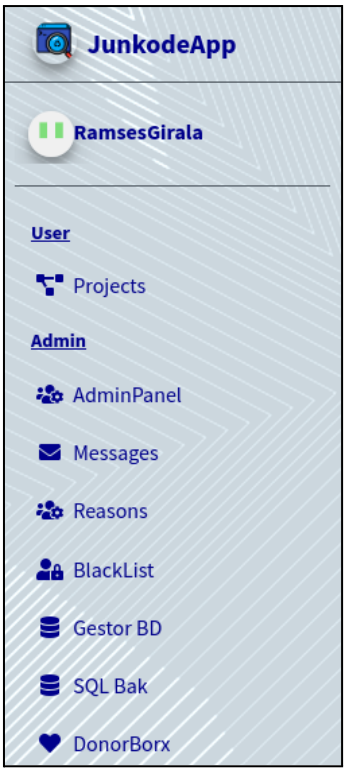


Figura a: Barra lateral de navegación del usuario administrador.

1.Dashboards del sistema.

Junkode ofrece a sus administradores un conjunto de gráficos con información útil del uso de la herramienta por los usuarios.

Dado que siempre puede ser conveniente incorporar nuevos indicadores, los gráficos se presentan en forma de menúes plegables. De esta forma, este diseño está pensado para ser más útil en el futuro, cuando aparezcan más métricas y gráficos en la interfaz de dashboards, el administrador pueda observar sólo los títulos de manera rápida para ubicar con mayor facilidad el indicador que busca.

A continuación se enumeran los indicadores con los que cuenta el sistema:

1.1.Proporción de análisis con Diagramas de Clases.

Este indicador señala el porcentaje de todos los resultados de análisis (y de todos los proyectos de Junkode) que generaron un diagrama de clases.

Este indicador se vale de la cantidad de resultados que tienen almacenados resultados de diagrama de clases y de los que no.

Contar con la proporción tiene dos funcionalidades: A corto plazo permite distinguir qué porcentaje de los proyectos están utilizando Spring Boot o Micronaut durante su desarrollo. Con esta información se puede inferir en sí la importancia que tiene la generación de este elemento de documentación en sí.

La segunda funcionalidad es a largo plazo, puesto que es necesario que haya suficientes resultados para que la proporción ya no sea tan sensible a los cambios. A largo plazo podría servir a los administradores para ver si hay una disminución sostenida pero no necesariamente abrupta de la generación de diagramas de clases. Esto podría indicar que ha habido una actualización en los frameworks Micronaut o Spring Boot y que las anotaciones utilizadas para generar los diagramas ya no están funcionando. De esta forma, la proporción de resultados con diagrama de clases sería útil para detectar cuando los visitors de entidades necesitan mantenimiento.

El indicador se refleja en un gráfico circular como se muestra a continuación.

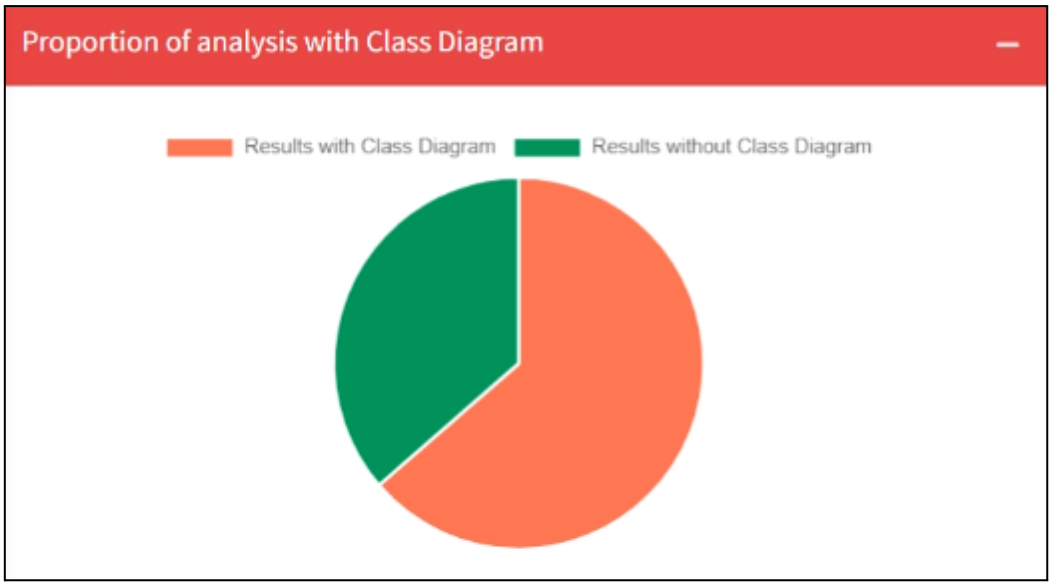


Figura 1.1: Gráfico circular de la Proporción de Análisis con Diagramas de Clases.

1.2.Proporción de análisis con Diagramas de Propiedades.

De manera similar al anterior, este indicador muestra la relación entre los resultados que cuentan con un diagrama de propiedades y los que no. Se basa también en la cantidad de resultados almacenados que tienen un diagrama de propiedades asociado.

La funcionalidad de este indicador es también similar a la funcionalidad a largo plazo del anterior. Permite notar cuándo el análisis de propiedades se encuentra obsoleto, en este caso debido a la ausencia de los archivos utilizados para generar los diagramas, o sea archivos “application.properties” y archivos en formato YAML con nombres específicos.

La diferencia que este indicador tiene con el anterior, es que si hay un cambio en la forma de almacenar las propiedades de un proyecto Java, la proporción de resultados con diagramas de propiedades caería con celeridad y de forma más abrupta. Esto permitiría al administrador advertir rápidamente este cambio para hacer el mantenimiento pertinente en el proyecto de Junkode.

Este indicador se representa con un diagrama circular de la siguiente manera:

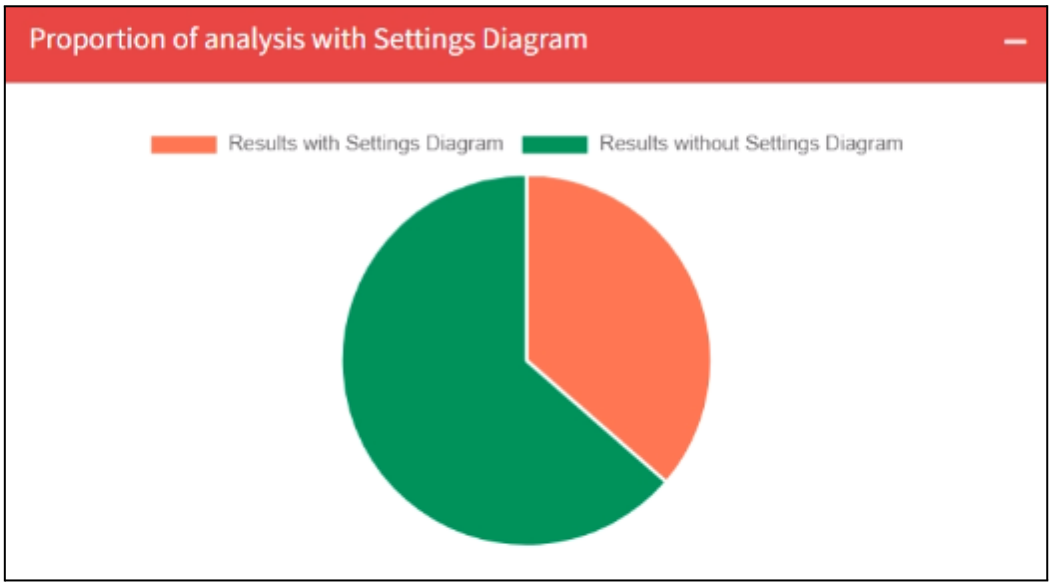


Figura 1.2: Gráfico circular de la Proporción de Análisis con Diagramas de Propiedades.

1.3.Serie de tiempo de resultados.

Este indicador muestra la cantidad de resultados que son generados por la herramienta en relación a días anteriores. De esta manera, permite al administrador ver el flujo cambiante de usuarios que utilizan Junkode.

El indicador le permite al administrador advertir picos de uso y esto es fundamental para Junkode puesto que este es un proyecto que corre sobre herramientas gratuitas, que no disponen de toda la capacidad que puede ser necesaria para soportar picos de concurrencia. Esto significa que con la serie de tiempo se puede estimar el crecimiento de la comunidad de usuarios de Junkode y, de esta forma, prever cuándo será conveniente comenzar a implementar servicios pagos para mantener la calidad de respuesta del sistema.

Este indicador se representa en un diagrama de series de tiempo como se muestra a continuación:

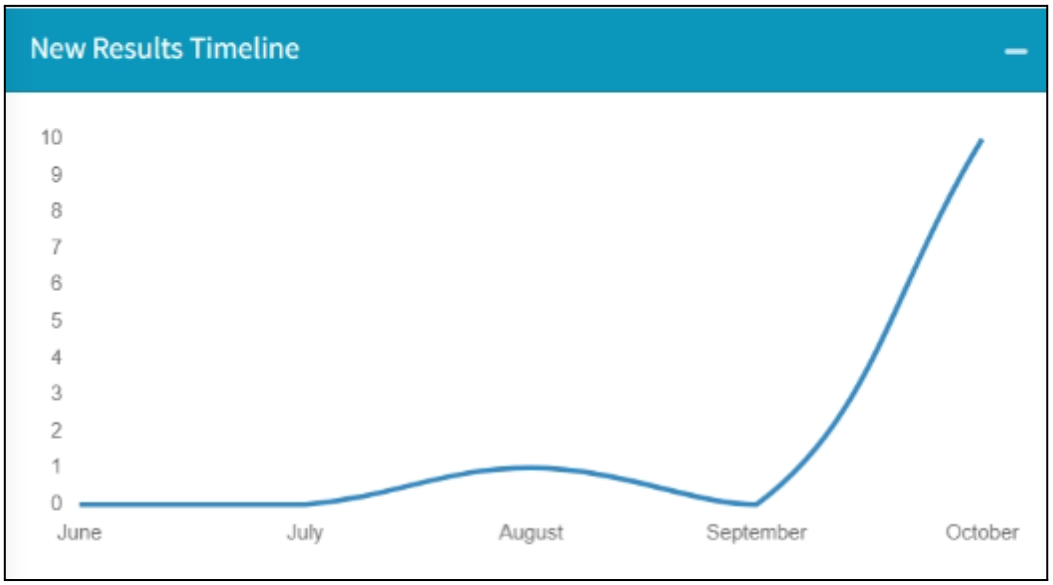


Figura 1.3: Gráfico de Serie de Tiempo de Resultados.

1.4.Serie de tiempo de Mensajes de los usuarios.

Este indicador muestra la evolución en el tiempo que tiene el uso del sistema de mensajería interna por los usuarios.

Para obtenerlos, se procesan los datos de auditoría de las entidades de mensajes persistentes. Cuando se encuentra un mensaje con una fecha de creación específica, aumenta un contador que lleva la cuenta de los mensajes enviados en ese día. Esto tiene en cuenta únicamente los mensajes enviados por los usuarios (ya sea en conversaciones creadas por ellos o en respuestas de mensajes de administradores). O sea, no se tienen en cuenta los mensajes enviados por los administradores.

La funcionalidad de este indicador es puntualmente advertir un pico de mensajes internos. Esto se realiza con la finalidad de encontrar rápidamente a los usuarios autores de estos picos, analizar sus mensajes para identificar si se trata de spammers y así poder incluirlos en la lista negra del sistema de mensajería interna (ver sección 2.2.Blacklist).

Este indicador se representa en un diagrama de series de tiempo como se muestra a continuación:

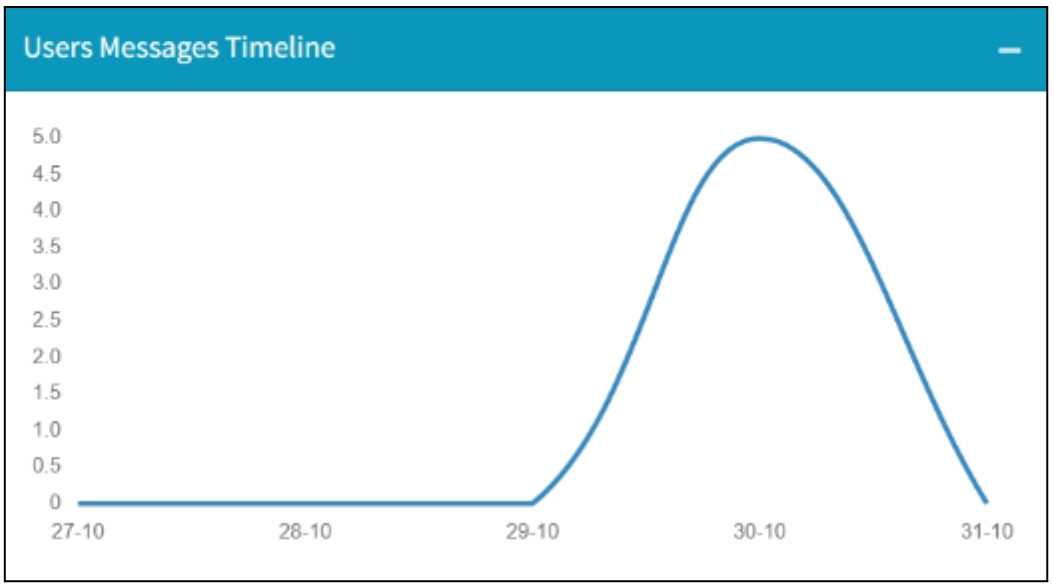


Figura 1.4: Gráfico de Serie de Tiempo de Mensajes de los usuarios.

2.Mensajería Interna.

Junkode ofrece un sistema de mensajería interna para sus usuarios registrados. En este los usuarios pueden escribir mensajes a los administradores de Junkode con un motivo predefinido, para que estos últimos les den respuesta.

Desde la interfaz de administrador se puede acceder a los mensajes recibidos y desde allí filtrarlos, por fecha y motivo, y contestarlos.

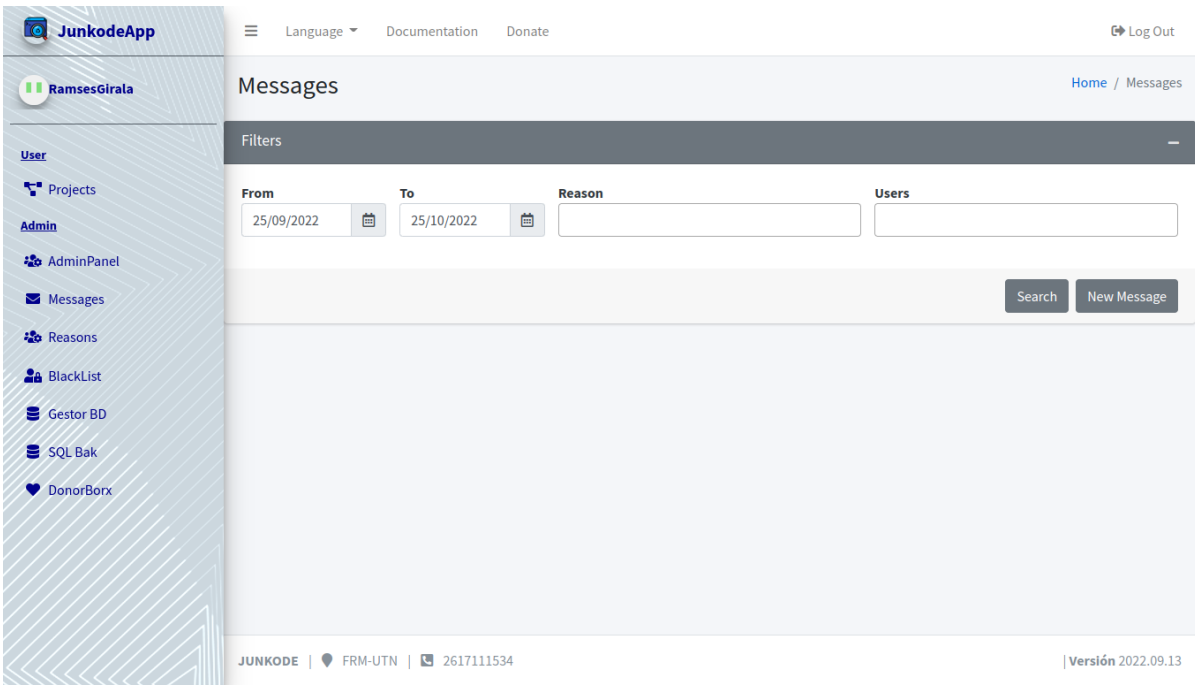


Figura 2: Pantalla de bandeja de entrada del administrador.

Lo que cada mensaje hace es crear una nueva conversación, en la que se guarda una lista con cada uno de los mensajes enviados, junto con los datos de su emisor y el timestamp de la fecha y hora de creación.

Es importante resaltar que solamente los usuarios registrados pueden iniciar una conversación, o sea, una conversación no puede ser iniciada por un usuario invitado ni por un usuario administrador.

2.1.Motivos.

Llamados internamente “Reasons”, los motivos son las categorías entre las cuales se dividen los mensajes de Junkode y tienen como funcionalidad filtrar los mensajes según su propósito.

El sistema inicialmente cuenta con los motivos “Consulta”, “Bug-Report” y “Otro”, sin embargo, se cuenta con un ABM en la vista de administrador para crear un nuevo motivo cuando sea necesario (así como también modificar o eliminar los ya existentes).

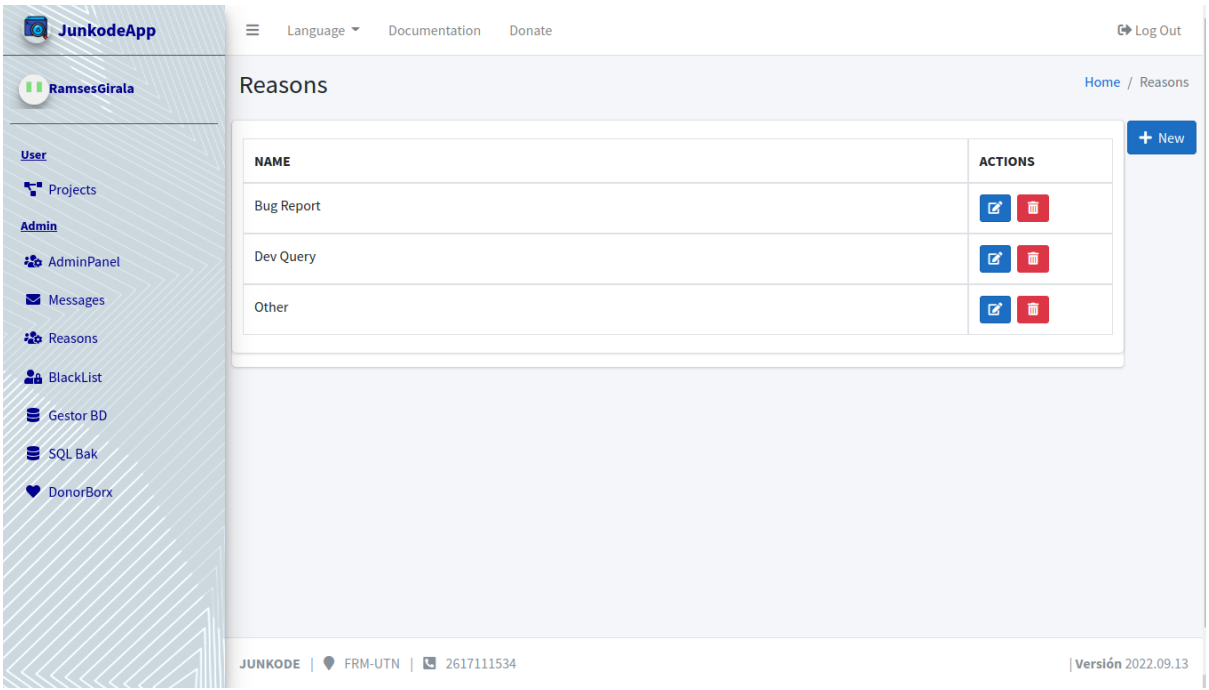


Figura 2.1: ABM de Motivos.

2.2.Lista Negra.

Llamada internamente “Blacklist”, esta es una herramienta que le permite a los administradores prohibirle el acceso a la mensajería interna a usuarios puntuales. Esta funcionalidad es imprescindible al contar con un sistema de mensajería interna, puesto que estos sistemas se prestan a la aparición de spammers y hackers que puedan intentar ejecutar inyecciones de SQL, y la funcionalidad de lista negra permite que los administradores puedan controlar esta situación bloqueando a dichos usuarios.

Un usuario puede ser añadido a la lista negra por su nombre de usuario (o sea, su nombre de usuario en GitHub) y de igual forma puede ser eliminado de la lista negra si el administrador lo ve conveniente.

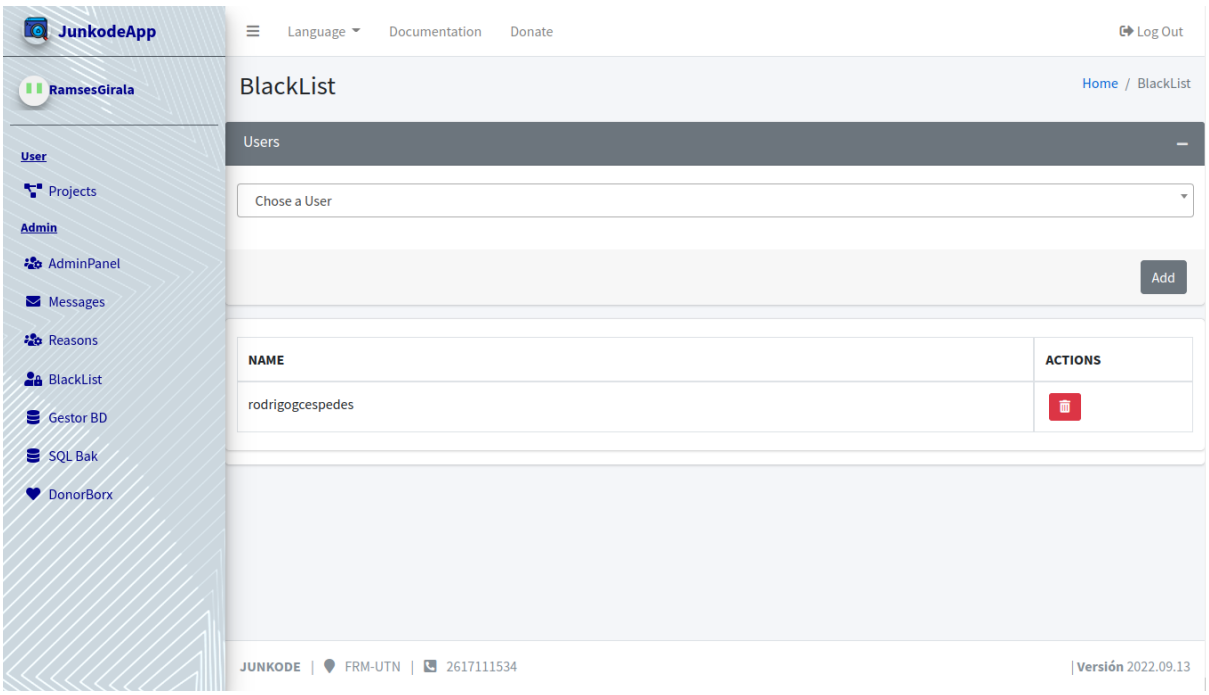


Figura 2.2:Lista negra de Junkode.

3.Administración de Bases de datos.

Como administrador, se tiene acceso a la gestión de la base de datos de Junkode, la cual se da a través de servicios web especializados para la gestión y administración de las bases de datos y sus respectivos back-ups.

El administrador puede dirigirse hacia esos servicios web desde los hipervínculos ubicados en la barra lateral de su vista.

3.1.Gestor de Bases de datos.

La gestión de la base de datos se hace por medio del gestor PHPMyAdmin, el cual es una herramienta web que permite un fácil acceso a la gestión de las bases de datos de Junkode.

Para ingresar a esta, puede dirigirse al enlace de PHPMyAdmin que se encuentra en la barra lateral de la interfaz de administrador, y luego iniciar las credenciales necesarias para acceder como administrador al manejo de la base de datos. La información solicitada es el Host de la base de datos, el nombre de usuario y su contraseña (por razones de seguridad, estos datos no se encuentran incluidos en este manual).

La base de datos se encuentra hosteada en un servidor gratuito en la nube, lo que puede implicar que el gestor de la base de datos demore en procesar las instrucciones que se le provean.

Puede accederse a PHPMyAdmin desde el enlace en la barra lateral de las interfaces del administrador, sin embargo se requieren las credenciales (servidor, usuario y contraseña) para acceder a la base de datos de Junkode.

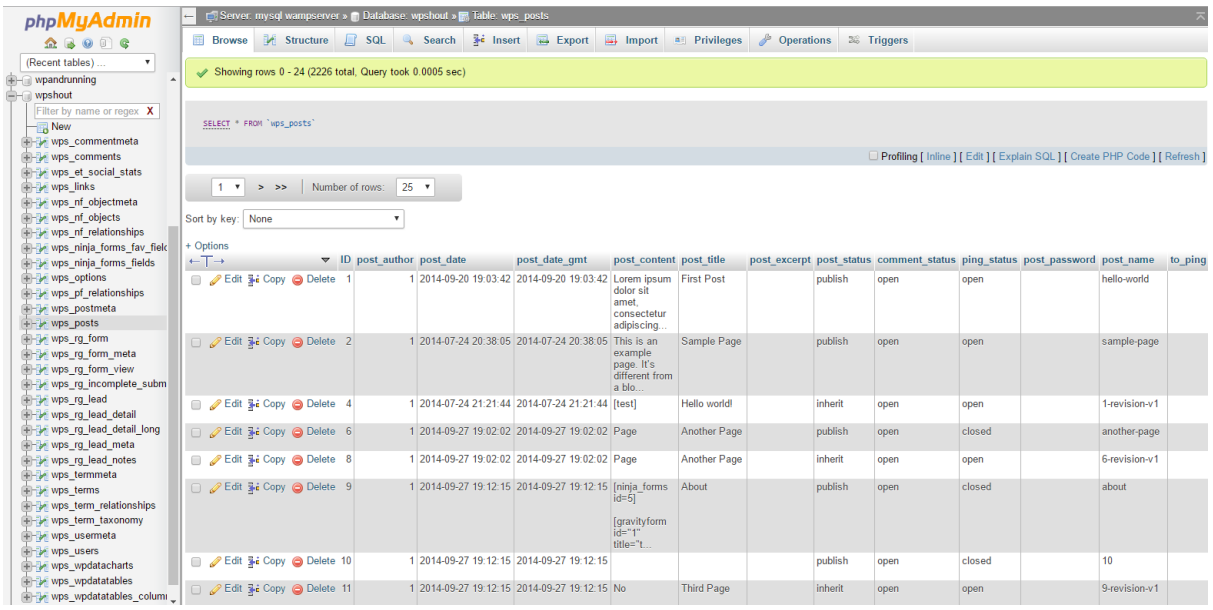


Figura 3.1: Gestor de Base de Datos PHPMyAdmin.

Desde el gestor de bases de datos se tiene completa libertad para trabajar sobre las bases de datos. Se puede, entre otras cosas, crear nuevas tablas, consultar auditorías, eliminar resultados, hacer consultas SQL personalizadas, hacer migraciones de datos, ejecutar scripts SQL e incluso hacer backups locales (esta última funcionalidad no es recomendable, ver el apartado 3.2 Gestión de Backups).

Están fuera de las funcionalidades del gestor: Bloquear y desbloquear tablas, crear nuevos esquemas, crear usuarios y revocar privilegios a usuarios ya existentes.

3.2.Gestor de Backups.

Los back ups son gestionados por SqlBak, el cual es un servicio que permite la automatización de backups del sistema.

Cada cuatro semanas esta herramienta genera un back up automático de toda la base de datos. La información SQL es comprimida, cifrada con una clave de seguridad (que no será expuesta en este manual) y archivada en un almacén de Google Drive.

La herramienta también se encarga de la eliminación automática de esa información, cada 3 meses se elimina el back up más antiguo que se encuentre aún archivado, conservando siempre dos snapshots del sistema almacenadas.

Desde esta misma herramienta también se pueden realizar Back Ups bajo demanda, o sea, que el administrador puede hacer un backup de manera manual cuando lo vea conveniente.

Puede accederse al proyecto Junkode-Bak desde la barra lateral de las interfaces de usuario, sin embargo es requerido el ingreso de credenciales para permitir la visualización gestión de los Jobs de Back Up.

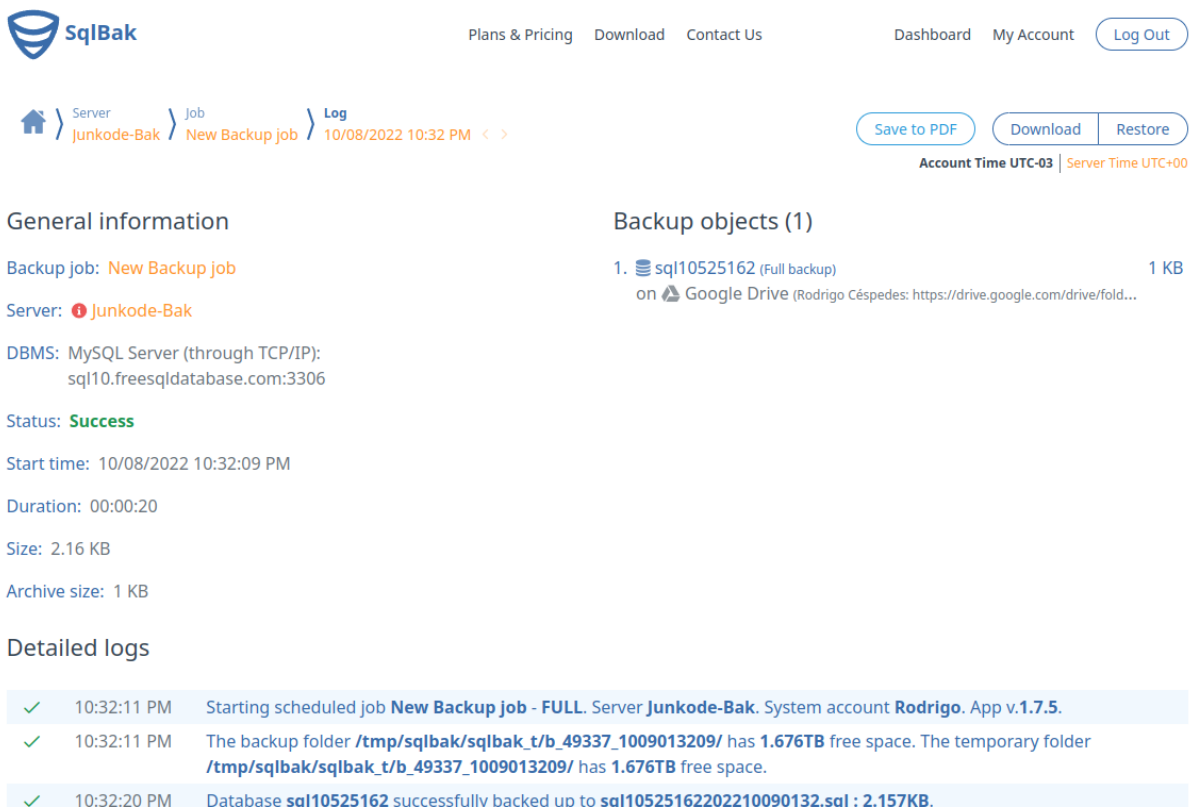


Figura 3.2.1: Dashboard de SqlBack.

Desde esta herramienta también se pueden hacer Backups bajo demanda, o sea, en el momento en que el administrador lo requiera, solo es necesario dirigirse al Job desde la pantalla del Dashboard de SqlBak, y seleccionar el botón “run now” de la derecha de la pantalla.

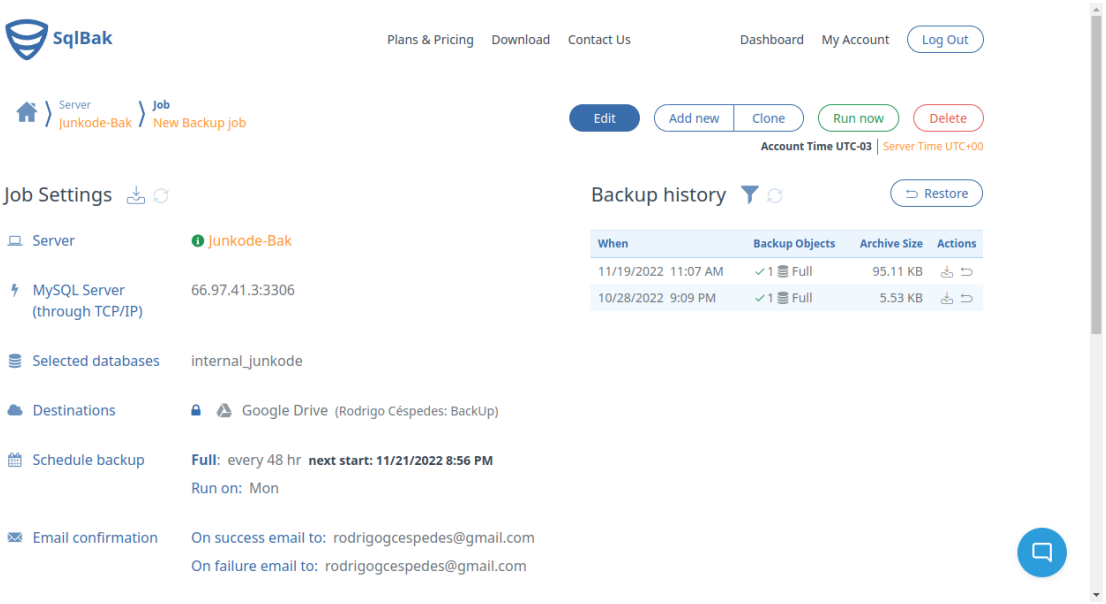


Figura 3.2.1: Job de SqlBack.

Para poder hacer esto es necesario que esté corriendo la instancia del servicio de Docker que contiene la acreditación desplegada de SqlBak, el cual requiere de un hash que no será publicado en este manual.

3.3.Recuperación de la Base de datos.

Como se especificó en el ítem anterior, las copias de respaldo solo deberían ser gestionadas por SQLBak, por lo que también lo deben ser las recuperaciones de la base de datos.

En la barra lateral de la vista del administrador puede encontrarse un enlace con el nombre “Recovery”, el cual redirige al administrador a la interfaz de gestión de recoveries del Job de SQLBack. En esta pantalla, pueden encontrarse todos los respaldos que se conservan de la base de datos, identificados con la fecha y hora de su creación.

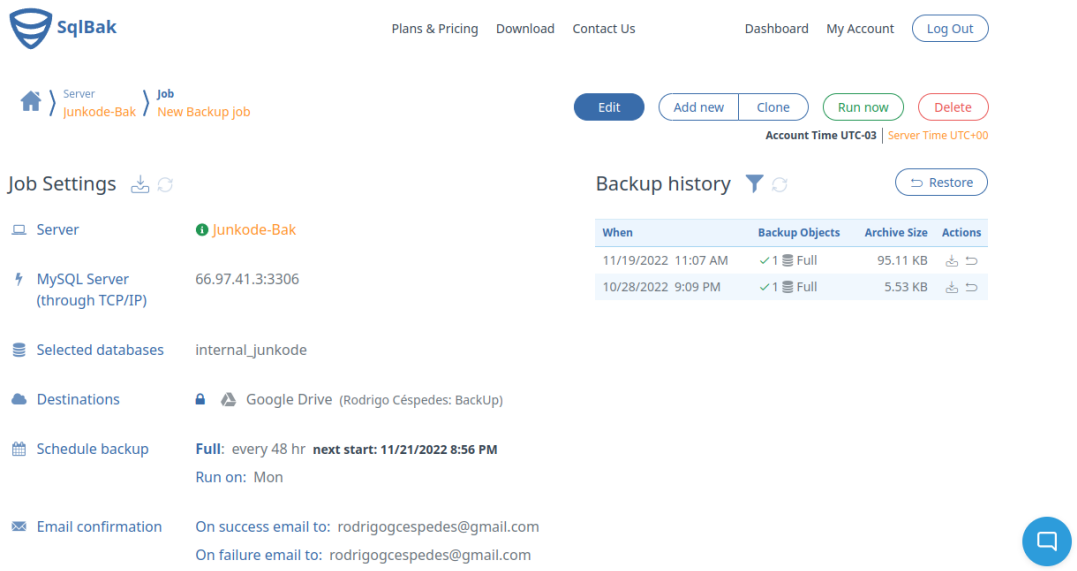


Figura 3.3.1: Pantalla con el historial de BackUps de SqlBak.

Para restaurar uno de estos respaldos solo se debe seleccionar la flecha que se encuentra sobre estos y seleccionar la base de datos a recuperar (por el momento, Junkode cuenta con una sola base de datos, pero desde aquí podrían recuperarse múltiples bases de datos simultáneamente).

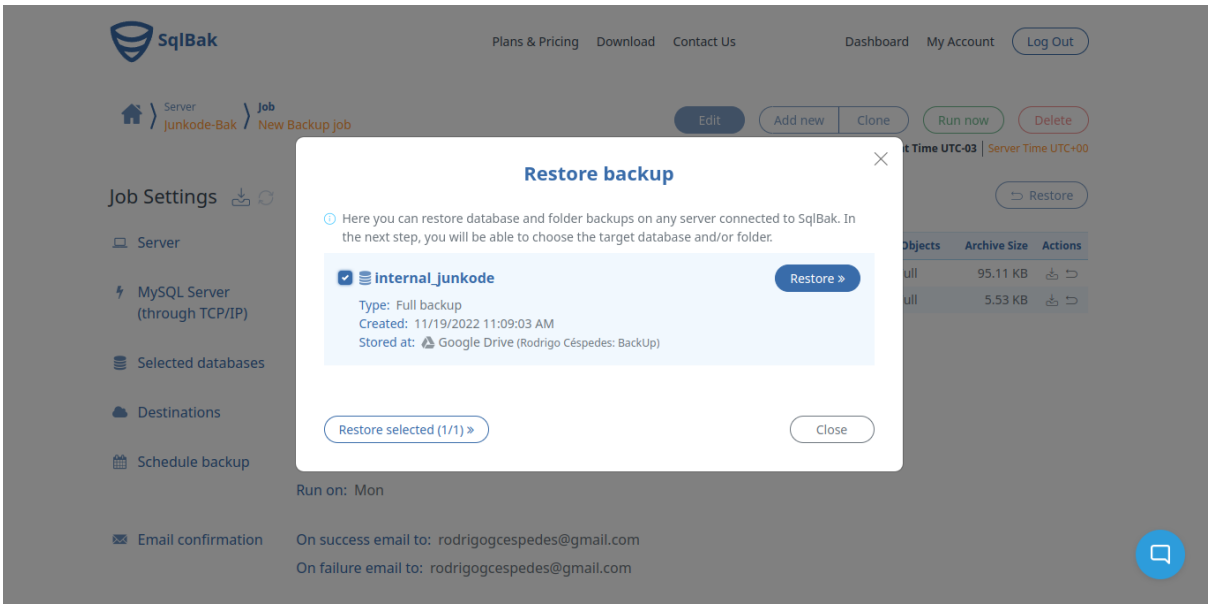


Figura 3.3.2: Pantalla de selección de bases de datos a restaurar.

Para hacer una recuperación hay tres niveles de seguridad a pasar, uno más que para el BackUp. Los dos primeros son compartidos con la funcionalidad de administrar SqlBak, que son: Ser un usuario administrador (para acceder al enlace del dashboard o del job de SqlBak desde Junkode) y poseer las credenciales de la cuenta SqlBak de Junkode.

La nueva restricción de seguridad que es añadida para esta funcionalidad es la de conocer la contraseña para desencriptar los archivos comprimidos que contienen los respaldos de

los datos (es importante recordar que los respaldos deben estar siempre encriptados para que nadie pueda acceder a su información). Esta restricción implica un nuevo “nivel de permisos” para los administradores, puesto que aunque a un administrador se le provean las credenciales de SqlBak para gestionar los backups, no va a poder restaurar versiones anteriores si no se le provee también esta segunda contraseña.

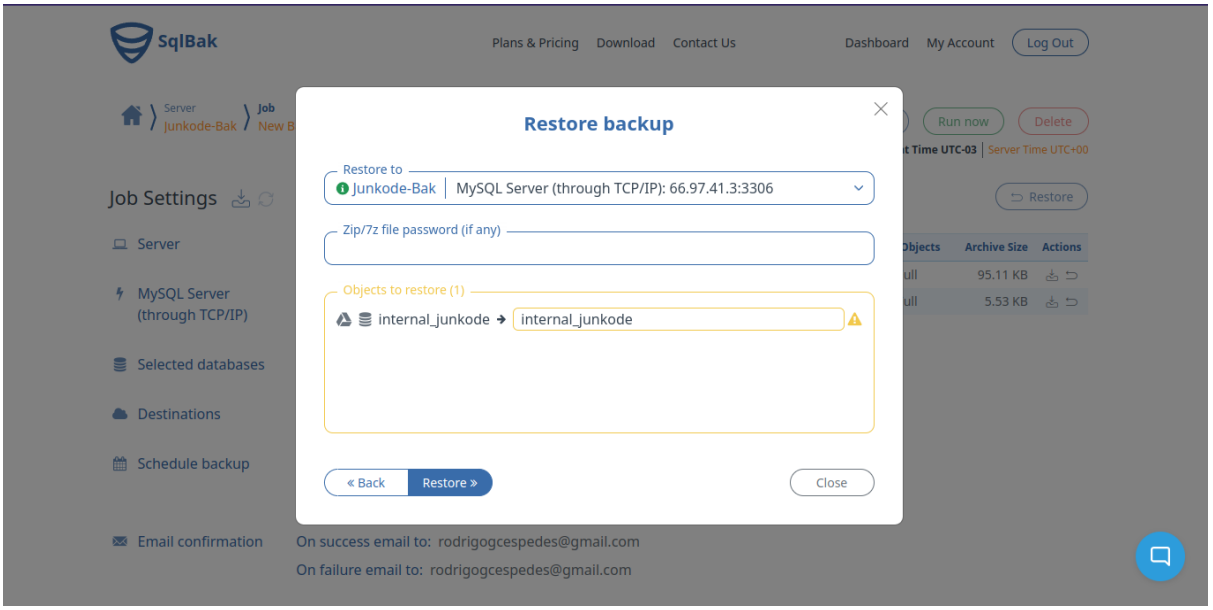


Figura 3.3.3: Pantalla de SqlBak donde se solicita la contraseña para desencriptar el respaldo.

Es importante recordar que los datos de la recuperación sobreescriben la base de datos actual. Si bien esto puede hacerse en producción, puesto que se aplica el control de integridad de CopyOnWrite, es recomendable no hacerlo a menos de que sea completamente meritorio (es preferible que se restauren los objetos necesarios desde el gestor de bases de datos, antes de que se haga una restauración completa de la base de datos). Para esto, SqlBak muestra un último cartel de advertencia, en el cual se pide por favor que el administrador evalúe nuevamente si hacer una recuperación es la mejor solución a los problemas.

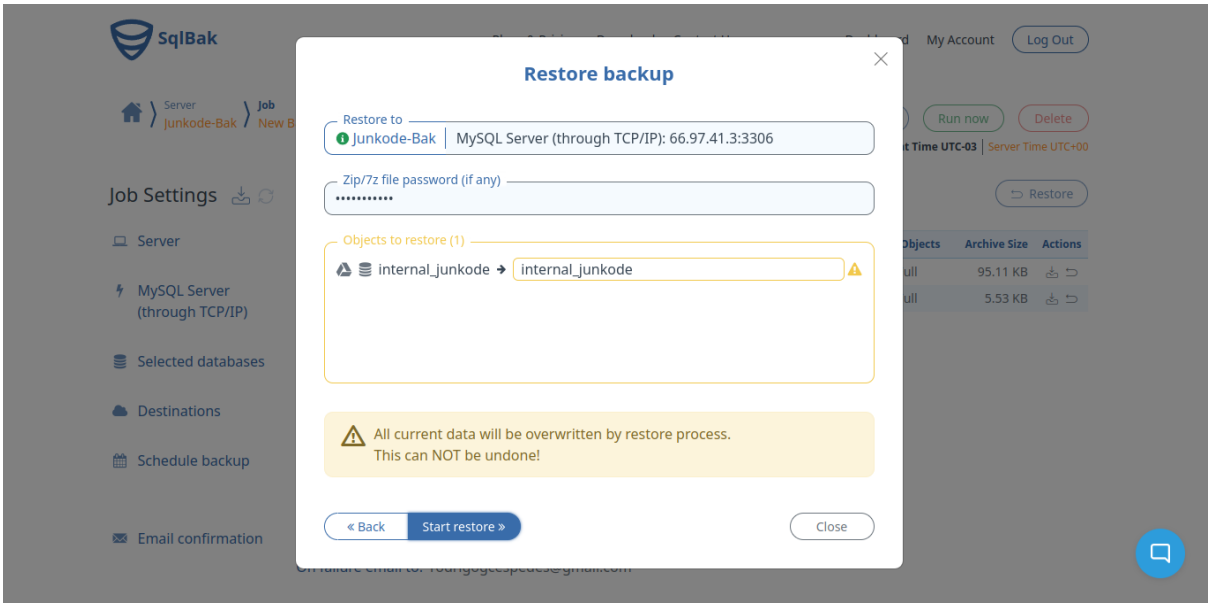


Figura 3.3.4: Pantalla de advertencia de sobreescritura de datos del respaldo.

4.Campaña de donación.

La campaña de donación de Junkode puede ser gestionada directamente desde el portal de Donorbox. Este provee herramientas de análisis de las donaciones, así como también las herramientas de customización de los elementos de la campaña (divisas aceptadas, medios de pago aceptados, cartera donde se deposita, tiempos de comisión, etc.) y los generadores de artefactos embebidos para ser añadidos o modificados en la página de donaciones de Junkode (también se cuenta con un endpoint de donación propio de Donorbox para la campaña de Junkode, pero este no es utilizado).

El enlace a la campaña de donación de Junkode en el portal de Donorbox se encuentra en la barra lateral de las interfaces de administrador. Serán solicitadas las credenciales de Donorbox al momento de intentar ingresar.

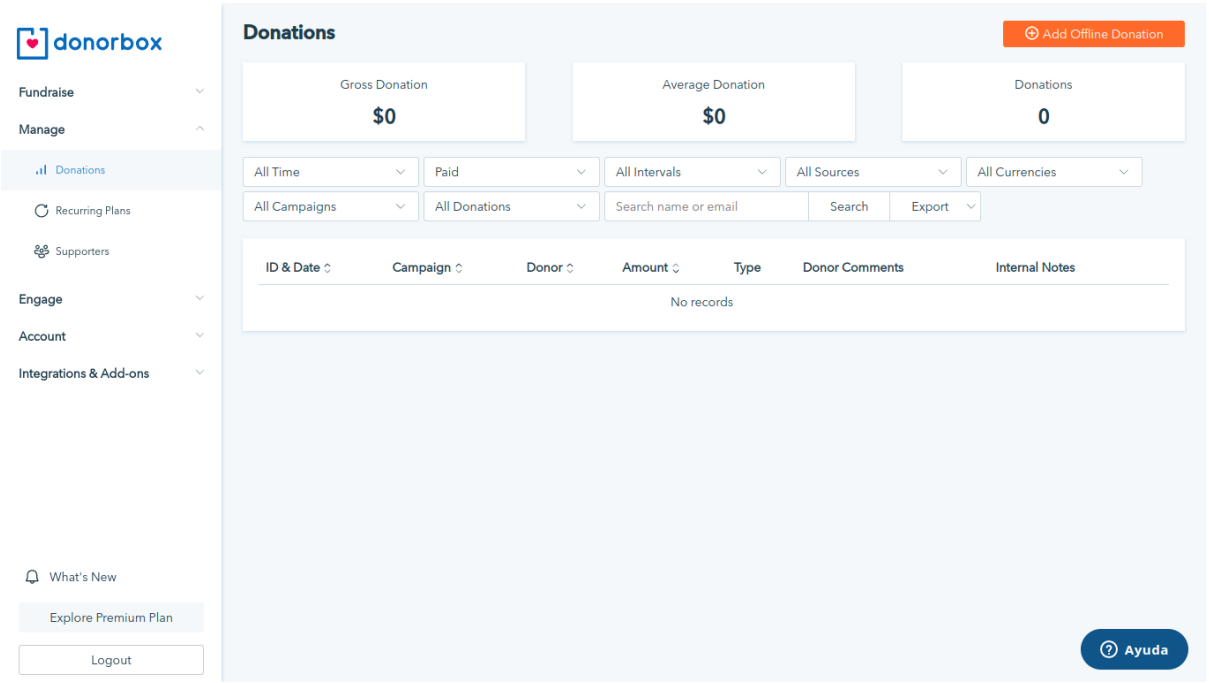


Figura 4: Panel principal del gestor de donaciones de Donorbox.

TRABAJOS PRÁCTICOS INTEGRADORES

JUNKODE - Documentador Automático de Código Fuente

ANEXO N° 11 - TRABAJO PRÁCTICO INTEGRADOR “DIRECCIÓN DE PROYECTOS DE SISTEMAS”

JUNKODE - Documentador Automático de Código Fuente

PROYECTO FINAL

TRABAJO PRÁCTICO INTEGRADOR
DIRECCIÓN DE PROYECTOS DE SISTEMAS

Junkode

2022

Integrantes:

(45185) Céspedes Ortega, Rodrigo Gabriel
(45352) Fernández Quatrini, Renzo Abel
(45241) Flores, Sebastián Andrés
(45106) **Girala, Ramses | Coordinador**
(38492) Groisman, David Nathaniel

Índice

1. Ordenar del 1 al 15 según la importancia (en el puesto N°1 la de mayor importancia) que le otorga a cada una de las funciones que deberías realizar como Jefe (o Director) de Proyecto, con una breve explicación de cada una.

El siguiente punto fue realizado en base a la lista de tareas de un líder de proyecto que se encuentra en la documentación de la cátedra. A continuación se hallan las 15 principales tareas seleccionadas, y luego de estas, las tareas que figuraban en la documentación pero no fueron incluidas de la lista junto con la razón de su exclusión.

1. **Liderar y ejercer diferentes estilos de liderazgo:** Esta es la principal actividad que debe realizar un jefe de proyecto, puesto que es la característica principal que identifica a este rol. Esta actividad resume todas las actividades sociales siguientes en la lista. Incluye motivación, delegación de tareas, toma de decisiones, etc.

2. **Tomar decisiones:** Es imperativo que un jefe de proyecto sea capaz de tomar decisiones, porque si bien el proyecto puede llevarse de forma democrática y que la toma de decisiones se preste a la participación de todo el equipo por una delegación de autoridad, es muy probable que llegue un momento en el que la democracia sea viable y se deba recurrir a la autocracia. Ahí es donde se ve que la responsabilidad de las decisiones recae únicamente sobre el líder, y debe estar preparado para ese momento.

3. **Aplicar diferentes estilos y técnicas de comunicación interpersonal:** Esta actividad consiste en la labor del jefe que resulta más importante para los demás miembros del equipo, puesto que para lograr la actividad que se encuentra primera en la lista, es necesario ser capaz de de dominar las distintas técnicas de comunicación que pueda requerir cada integrante del equipo. Esta actividad está debajo de “Liderazgo y ejercicio de diferentes estilos de liderazgo” porque si bien es necesaria para que esta se pueda realizar, no es suficiente.

4. **Planificar y gestionar la planificación:** Determinar cómo se llevarán a cabo las primeras instancias del proyecto es fundamental, y es donde se aprecia directamente el rol del jefe. Esta etapa de la vida del proyecto resalta tanto para el jefe debido a que es cuando le corresponde a él hacer más trabajo que a los demás miembros del equipo. Por ejemplo, durante la implementación del proyecto, el jefe toma decisiones y se involucra con las tareas de desarrollo, pero quienes realmente están trabajando son los demás miembros del equipo, y el jefe trabaja como uno más. Pero durante la planificación es distinto, el jefe trabaja mucho más que los demás miembros, y esto se debe a que es su deber establecer las bases de lo que será luego el proyecto en el que sí trabajarán mucho más activamente los miembros de su equipo.

5. **Diseñar y ejecutar acciones para el logro de equipos equilibrados y efectivos:** El líder no es nada sin un equipo que lo acompañe, de hecho eso sería un absurdo.

Y si bien no está contemplado dentro de las facultades del jefe de proyecto hacer que todos los miembros de su equipo se agraden y sinergicen, si es una responsabilidad de este hacer todo lo que esté en sus manos para que a ellos les sea más fácil lograr eso. Para lograr esto, es fundamental que el líder comprenda las diferencias entre las funciones de cada miembro del equipo, del rol de cada miembro del equipo, puesto que este segundo hace referencia a lo que ese miembro puede aportar para el proyecto, y no tanto las actividades que va a realizar. Cuando se habla de tener un equipo balanceado, es a este aspecto al que se hace referencia, y para que el jefe del proyecto pueda alcanzar este equilibrio es necesario que diseñe y ejecute acciones (que pueden ser desde simples charlas hasta arreglos de horarios y tareas) para que los miembros del equipo puedan trabajar de la forma en la que más integrados se encuentren sus roles y más favorecidos se vean ellos en conjunto.

6. **Asignar tareas y recursos:** El líder es responsable por el aprovechamiento máximo de los recursos que dispone, por lo que para poder llevar a cabo esta tarea, debe tener un entendimiento extensivo de los mismos para, de esta manera, ser capaz de asignarlos de la manera más eficiente. De la misma manera debe actuar frente a la asignación de tareas, ya que el rendimiento de las mismas dependerá de la persona que la realice, y lo que se espera es un óptimo rendimiento de la misma.

7. **Analizar a las personas, diseñar y aplicar técnicas de motivación individual:** Las personas mejoran en gran medida su rendimiento y productividad cuando están motivadas. Para poder conseguir esto, el líder debe realizar la labor de analizar lo que más pueda a las personas que tiene a cargo. Esto es necesario para poder diseñar de manera correcta las distintas técnicas de motivación individual que posteriormente se aplicarán. La importancia de un buen análisis recae en que debido a que todas las personas son distintas, no todas necesitan las mismas técnicas para motivarse.

8. **Administrar eficientemente los recursos y gestionar presupuestos:** Para que el proyecto pueda llevarse a cabo con éxito, es necesario que el presupuesto del mismo sea bien utilizado. Por lo que el líder debe asegurarse de que no le falte dinero a ningún área, porque en caso que esto suceda, dicha área no será capaz de finalizar la tarea que se le fue asignada, poniendo en peligro el desarrollo exitoso del proyecto.

9. **Emitir órdenes e instrucciones:** El líder de proyecto es quien carga con la responsabilidad del cumplimiento del mismo, por lo que para asegurarse de eso, debe encargarse de emitir órdenes e instrucciones específicas en los momentos

determinados que él considere necesario, para asegurar el cumplimiento de las tareas.

10. **Aplicar retroalimentación y resolución de conflictos:** A medida que el proyecto avance, las relaciones interpersonales pueden ir empeorando con el tiempo, dando lugar al surgimiento de conflictos entre el personal. En estos momentos es donde el líder debe mostrarse firme y buscar la manera de llegar a un consenso entre las partes, consiguiendo resolver el conflicto, ya que si no lo consigue, nuevamente se pone en riesgo el cumplimiento del proyecto.

11. **Supervisar y controlar cumplimiento:** Son todas las actividades que se realizan para controlar el esfuerzo de los empleados, con el objetivo de que éstos cumplan con las funciones o tareas que le han sido asignadas. En caso de que los resultados no sean los esperados, el líder debe averiguar el porqué y encontrar una solución para lograr los resultados esperados.

12. **Gestionar riesgos:** A pesar de que el jefe de proyecto delegue tareas a los miembros de su equipo, debe recordar que él sigue siendo el responsable del proyecto y, por tanto, de cada una de esas tareas, por lo que es necesario que tenga planes de prevención y contingencia para cada posible amenaza que pueda presentarse para el proyecto. Esta tarea es llamativa porque está presente desde la planificación del proyecto hasta que se este se descontinúa, o sea, gestión de riesgos es una función del jefe, intrínseca al proyecto.

13. **Aplicar técnicas y métricas de evaluación de ejecución y finalización del Proyecto:** El jefe del proyecto realiza esta actividad en busca de conocer qué tanto el proyecto ha logrado cumplir sus objetivos. Para lograrla es necesario que antes este haya producido la información para realizar esta evaluación. Esta información proviene de la definición de métricas e indicadores que no solo permiten comprender qué datos son los relevantes a relevar, sinó que también los presentan de una forma cómoda y rápida para interpretar a la hora de hacer la evaluación.

14. **Generar informes iniciales, parciales y finales:** El jefe de proyecto debe generar informes a lo largo de todo el ciclo de vida de este. Los mismos tienen como fin documentar información recogida y previamente analizada para que así luego el jefe pueda apoyarse en estos a la hora hacer una evaluación del proyecto y a la hora de la toma de decisiones.

15. **Formular el proyecto:** Es el procedimiento general para el planteamiento de un proyecto. Para el mismo, el jefe debe identificar la necesidad a ser satisfecha,

describirla, proponer objetivos o beneficios, identificar y cuantificar recursos, listar las tareas necesarias, estimar tiempos y definir métricas de evaluación. Como se dijo en el ítem número cuatro, la etapa de planificación es la etapa donde más se nota la participación del líder y esto se debe principalmente al desarrollo de esta actividad.

Las actividades del jefe de proyecto no incluidas en la lista son:

-**Liderar el diseño de planes de testing, capacitación, implementación, manuales, documentación técnica, de operación, específicas:** No se incluyó esta actividad en la lista porque ya se encuentra presente repartida entre las actividades “planificación y gestionar la planificación” y “Generar informes iniciales, parciales y finales”.

-Ejercer diferentes tipos de autoridad: Esta actividad no se incluyó porque está dentro de la actividad “aplicar diferentes tipos de liderazgo para el proyecto”. Puesto que para poder aplicar distintos tipos de liderazgo, es generalmente necesario recurrir a distintos tipos de autoridad para con cada miembro del equipo, en pos de la forma de liderazgo que se desee emplear.

-Ejercer el coaching: Coaching es la disciplina de hacer preguntas para ayudar a otras personas, a través del aprendizaje, en la exploración y el descubrimiento de nuevas creencias que tienen como resultado el logro de sus objetivos. Dada esta definición de coaching como la convenida, no se incluye en la lista puesto que está incluida ya dentro de “analizar a las personas, diseñar y aplicar técnicas de motivación individual”.

-**Verificar entregables:** No se incluye esta actividad porque ya está incluida dentro de la actividad de “Supervisión y control de cumplimiento” y en parte también dentro de “Aplicar técnicas y métricas de evaluación de ejecución y finalización del Proyecto”.

-**Efectuar tareas técnicas de análisis, diseño, desarrollo, testing, implementación:** Esta actividad, si bien es sumamente importante, no se encuentra dentro de las primeras 15 más importantes, puesto que es algo no intrínseco al líder, sino a cualquier miembro del equipo. En la documentación brindada por la cátedra ya se resaltaba que esta actividad solo compone el 20% de las tareas de un líder, y esto es fundamentalmente porque ya hay otras personas para hacerse cargo de esas tareas, de modo que la responsabilidad del líder no deben ser esas tareas, sino las personas que las realizan.

-**Aplicar técnicas y métricas de estimación de tiempo y esfuerzo y evaluación inicial del Proyecto:** Esta es una tarea que queda englobada dentro de la actividad “Formular el proyecto” y por dicho motivo no se tiene en cuenta en esta lista.

2. Cuáles son las 5 principales funciones que cumplirá durante la fase anterior a la ejecución del proyecto, el "Jefe (o Director) de Proyecto" (pueden repetirse con las del punto anterior).

1. Planificar y gestionar la planificación.
2. Diseñar y ejecutar acciones para el logro de equipos equilibrados y efectivos.
3. Formular el proyecto.
4. Asignar tareas y recursos.
5. Desarrollar análisis de factibilidad y riesgos.

3. Cuáles son las 5 principales funciones que cumplirá durante la fase de ejecución del proyecto, el "Jefe (o Director) de Proyecto" (pueden repetirse con las del punto 1).

1. Liderar y ejercer diferentes estilos de liderazgo.
2. Analizar a las personas, diseñar y aplicar técnicas de motivación individual.
3. Aplicar retroalimentación y resolución de conflictos.
4. Realizar, mediante diferentes técnicas, el control y la evaluación del proyecto.
5. Tomar decisiones.

4. Si los obligarán a incorporar al equipo del Proyecto a 2 personas, en qué momento los incorporaría, en cuál puesto y perfil y qué actividades les asignaría.

Al primero lo incorporaremos en etapa de desarrollo del proyecto, ocupando el puesto de desarrollador.A esta persona se le asignará las actividades de desarrollo de login de usuario,conexión de usuarios a la base de datos y dar apoyo a los demás desarrolladores en sus actividades. El perfil del puesto es el siguiente:

- Secundario completo.
- Disponibilidad para realizar trabajo remoto y bajo concepto de "guardia".
- Disponibilidad de teléfono móvil propio, correo electrónico y conexión a internet en domicilio.
- Conocimientos en los siguientes campos: paradigma orientado a objetos, lenguaje java, spring boot, spring security y oauth2.0.
- Nivel de Inglés intermedio.
- Experiencia demostrable en proyectos anteriores.
- Proactividad.
- Capacidad de trabajo en equipo.
- Ávido de conocimientos nuevos.

A la segunda persona la incorporaremos una vez el proyecto esté finalizado y se comience con la difusión del mismo, asignándole el puesto de encargado de marketing digital.A esta persona se le asignará las actividades relacionadas a la comunicación del proyecto al público,análisis de mercado, participará en los congresos, trabajará en el marketing y publicidad del software.El perfil del puesto es el siguiente:

- Secundario completo.
- Disponibilidad para realizar trabajo remoto y bajo concepto de “guardia”.
- Disponibilidad de teléfono móvil propio,correo electrónico y conexión a internet en domicilio.
- Conocimientos en: anuncios en línea y herramientas de analitica web.
- Nivel de Inglés avanzado.
- Al menos 2 cursos realizados en marketing digital.
- Excelentes habilidades de comunicación.
- Creatividad.
- Gran capacidad analítica.

5. **Decidir qué estilo de liderazgo se deberá utilizar durante la ejecución del Proyecto, con la fundamentación correspondiente. Recordamos que los estilos de liderazgo pueden ser:**

   - **LIBRE: Cuando se dispone de personas en el equipo de trabajo que tienen alto grado de preparación, capacidad y responsabilidad.**
   - **DEMOCRÁTICA: Cuando se intenta lograr el tratamiento participativo de todos los temas, situaciones y llegar a decisiones por consenso.**
   - **AUTOCRÁTICA: Cuando por diferentes motivos, no se puede aplicar ninguna de las anteriores y se necesitan tomar y ejecutar decisiones rápidas.**

Hay factores que determinan qué estilo de liderazgo es más adecuado según el proyecto a desarrollar, estos son:

- Tiempo para poder desarrollar el proyecto: El factor tiempo es clave para poder tomar las decisiones de manera precisa. Si se cuenta con una plenitud de tiempo se puede realizar un buen análisis para las decisiones en el desarrollo de un proyecto, permitiendo un enfoque democrático. Pero en cambio si no se tiene el suficiente tiempo se deben realizar decisiones rápidas con un estilo más autocrático. Dependiendo de la situación y de los conocimientos de los integrantes también se podría desarrollar un estilo de liderazgo libre, esto solo si tienen un alto grado de conocimientos y experiencias.
- Nivel de conocimiento de los integrantes: cuando se disponen de personas especializadas y capacitadas en las áreas a desarrollar del proyecto, se puede obtener diferentes puntos de vista para este, lo cual ocasiona que se pueda desarrollar de manera eficiente y con una buena comunicación. Se puede producir un buen respeto a la hora de discutir diferentes ideas y de como poder llevarlas adelante.

    - Desarrollar un proyecto con personas donde la mayoría de los integrantes tienen experiencia y capacitación resulta muy efectivo para el estilo de liderazgo democratico. Si todos los integrantes tienen los conocimientos y experiencias requeridas para desarrollar este proyecto algunas veces es bueno tener una combinación de liderazgo de democratico con libre, ya que las personas tienen experiencia en sus tareas y les resulta más fácil poder implementarlas de forma individual.
    - Al estar con integrantes que se tienen poca experiencia es ideal poder guiarlos de formas precisas o poder ayudarlos a desarrollar habilidades para las diferentes tareas. Es importante que los que tienen conocimientos y experiencias desempeñen el papel de guía para los que tienen falta de conocimientos. Como guiar a una persona, o proporcionar capacitación adecuada, requiere de tiempo, algunas veces es necesario realizar un liderazgo del tipo autocrático, ya que es necesario a veces tomar las decisiones rápidamente para no perder demasiado el tiempo en las capacitaciones de los integrantes.
- Química entre los integrantes o nivel de compañerismo: Este factor es importante a la hora de desarrollar un proyecto ya que puede provocar conflictos entre los integrantes. En situaciones donde los integrantes tengan alto grado de conocimientos en las áreas de conocimientos puede producir un conflicto por los distintos puntos de vista y ninguno quiere ceder en su postura. En estos casos conviene usar un estilo de liderazgo autocrático para poder avanzar con el desarrollo. También se podría usar un estilo democrativo para poder decidir por mayoría para saber elegir el mejor punto de vista. No sería recomendable en estos casos usar el liderazgo libre porque puede que los miembros del equipo no se puedan coordinar bien para el desarrollo del proyecto. El cómo se va a desempeñar un miembro del equipo con los demás integrantes es muy importante para elegir el estilo de liderazgo adecuado.

Para el equipo de trabajo de Junkode, teniendo en cuenta lo explicado, se llega a la siguiente conclusión sobre cuál es el tipo de liderazgo a adoptar:

Normalmente, se emplea un liderazgo democrático, puesto que todos los miembros del equipo tienen la capacidad suficiente para participar y hacer su aporte para la toma de decisiones, y sólo en casos de urgencia, se recurrirá a la autocracia para la toma de decisiones, teniendo la voz el Coordinador del grupo (cabe aclarar, que la existencia de esta autocracia se decidió democráticamente).

Como se puede entender, el único factor que puede alterar el estilo de liderazgo que se aplica sobre el equipo de Junkode es el tiempo, puesto que todos los integrantes del equipo se encuentran capacitados y existe un excelente ambiente de trabajo entre los miembros del equipo.

6. **Detallar los principales 10 riesgos que pueden aparecer en el proyecto, cuáles serían sus consecuencias y qué impacto tendrían esas consecuencias. Además, detallar cuáles son las medidas preventivas para cada uno de los riesgos. Recordamos que las medidas preventivas tienen como objetivo reducir la probabilidad de ocurrencia de cada riesgo o reducir el impacto que produciría cada riesgo.**

   Para estudiar los riesgos del sistema, primero es necesario conocer sus potenciales amenazas, puesto que son estas de las que pueden generar los riesgos para este.

   Para estudiar las amenazas se tuvieron en cuenta dos dimensiones de estas: su probabilidad de ocurrencia y el impacto que podría tener sobre el sistema (ambas dimensiones medidas en escala de 1 a 5, donde 5 implica que la amenaza es más probable o de mayor impacto). El producto de estos dos indicadores da como resultado el riesgo de la amenaza.

   Luego para cada amenaza se plantearon medidas para reducir estos factores, cada medida puede reducir la probabilidad de ocurrencia, el impacto, o ambos, y con los nuevos valores de los indicadores, se vuelve a calcular el riesgo de la misma forma, dando como resultado un número menor o igual al riesgo antes de las medidas.

   La clasificación del riesgo sigue un código de color, si es menor a 7 le corresponde el color verde, si es mayor o igual a 7 pero menor de 15, amarillo y si es mayor o igual a 15, rojo.

   El resultado del estudio puede encontrarse al final de este informe, bajo el título Anexo n°1 “Tabla de riesgos”.

7. **Decidir cuál enfoque de resolución de conflictos aplicará en supuestas situaciones (que también detallará) que se le puedan presentar durante el proyecto. Si tuviera que aplicar los conceptos de negociación, cuáles aspectos consideraría.**

   Se toman diferentes tipos de enfoques que dependen de cuáles sean los conflictos, a continuación se explica qué enfoque se aplicará en qué situación:

   - **Enfoques de Colaboración:**

     Se aplica cuando en el proyecto algún integrante tenga dudas en la manera de realizar una tarea, la manera de resolverlo es que algún miembro del equipo que tenga más conocimientos en la resolución de esa tarea lo va a ayudar. Esto se hace porque el objetivo es aprender, entonces, de esta manera, se soluciona el conflicto a la par que se capacita a un miembro del equipo.

➢ **Enfoques Agresivos:**

En situaciones críticas en las cuales sea vital tomar acciones rápidas y decisivas se aplica este enfoque. Estas situaciones están previstas que ocurran cerca de los Hitos del proyecto, debido a que se pueden presentar conflictos varios conflictos con los entregables y al estar tan cerca de la fecha de las entregas se justifica tomar acciones más asertivas, algunos son:

- Algún integrante no ha realizado su parte: Cuando esto suceda lo que se realiza es asignarle la tarea a otro miembro del equipo.
- Dos o más integrantes no se pueden poner de acuerdo en la realización de una tarea: Se resuelve haciendo que el coordinador del equipo elija cuál de las realizaciones es la más correcta.

➢ **Enfoques de Arreglo:**

Este enfoque se aplica en el caso de la deserción de un integrante del equipo, ya que se busca llegar a soluciones rápidas. Si se presenta la situación de que un integrante abandona el equipo, el desarrollo del proyecto se ve comprometido, entonces se busca que el proyecto se vea lo menos afectado posible por su deserción. Se reparten las tareas del miembro desertor entre el resto de los integrantes del equipo y se revisa para planificar capacitaciones en caso de que sea necesario.

8. Detallar al menos 5 técnicas de motivación que utilizará durante el proyecto (indicando si se trata de técnicas de motivación positiva o negativa), y detallar en qué tipos de situaciones sería necesario aplicar cada una.

Mayor participación en la toma de decisiones en el desarrollo del proyecto (positiva): Al miembro del equipo que se muestra que tiene un interés mayor en las actividades designadas, se lo empieza a tener en cuenta en decisiones más importantes. Estas decisiones están basadas en el contexto de las actividades que realiza dicho miembro. Este incentivo produce una mayor productividad y genera una sensación de mayor “seguridad de empleo” (en este caso, quiere decir que el miembro continuará en el equipo de trabajo y en el proyecto).

Incentivos monetarios (negativa): Cuando un miembro del equipo del equipo empiece a demostrar una baja productividad y no se lo vea motivado, se le da pequeños incentivos monetarios. Esto se aplica con una metodología llamada “premio a futuros aumentos”, la cual es dar un premio monetario, y en caso de que denote una mejora en el desarrollo de sus actividades, pasa de ser un premio a ser algo fijo en el sueldo.

Si bien este método de motivación no es aplicable al proyecto en este momento (debido a que actualmente no existe remuneración alguna para los miembros del equipo del proyecto), si es considerada para ser implementada en un futuro, puesto que estos incentivos reflejarían el aumento de donaciones en un mes, lo que permitiría su entrega a modo de premio y, si se mantiene el nivel de calidad ofrecida, también lo harán las donaciones, permitiendo que se pueda aplicar la metodología “premio a futuros aumentos” en Junkode.

Aumento de las “responsabilidades” (positiva): Cuando se vea que un miembro del equipo está realizando su trabajo de manera excepcional, se lo promoverá aumentando sus “responsabilidades”. Responsabilidades está entre comillas puesto que estas no pueden asignarse ni transferirse, la responsabilidad del proyecto siempre va a ser del líder del proyecto. Entonces, lo que realmente se aumenta para el miembro del equipo es la autoridad, puesto que ahora se le asignan nuevas tareas que antes no se le asignaban, ayudándolo moralmente y proveyendole la motivación que le corresponde.

Mejora espacio de trabajo (positivo): Según la teoría de la motivación de Herzberg, este punto no está tan referido a una técnica de motivación, sino a una técnica de conservación de Sanidad. Cuando un miembro del equipo del proyecto presente dificultades con el manejo de programas requeridos para el desarrollo de la herramienta (ya sea por problemas de su computadora, versiones de programas, incompatibilidad de dependencias, etc.), el jefe del proyecto reaccionará brindándole soporte, ya sea ayudándole con los programas de desarrollo, brindando ayuda indirecta o incluso reasignando las tareas, con el fin de que el miembro del equipo se sienta cómodo con el trabajo que realiza y con los medios con los que lo realiza.

Reconocimiento grupal por aporte realizado (positiva): Si se realiza reuniones con los demás miembros del equipo, se le realiza un reconocimiento frente a todo el equipo. Este reconocimiento hace que la persona se sienta importante y esencial dentro del equipo. Además, este reconocimiento se puede amplificar en el caso de darle algún curso o capacitación ayudándole a su formación personal.

9. Describir el método de conversión del Sistema (para pasar del sistema actual al nuevo, por ej. directo, paralelo, por etapas, piloto o alguna combinación de ellos), con todas las actividades a realizar. Se debe registrar en este punto no sólo el método y las actividades sino también la justificación correspondiente al máximo nivel de detalle.

El método de conversión del sistema que se va a utilizar en la implementación es una combinación del paralelo con el piloto. Esto se debe a que la herramienta está principalmente orientada a software factories, donde se desarrollan múltiples proyectos en paralelo, y donde la documentación de los sistemas se queda obsoleta rápidamente debido a que no se actualiza.

El motivo de hacerlo en paralelo, es para que la empresa que utilice la herramienta pueda evaluar cómo evolucionan por separado los dos proyectos, uno donde se documente manualmente como se realiza hoy en día y otro donde la documentación se realice automáticamente con la herramienta Junkode, usando una prueba piloto, y que en base a esa evaluación pueda tomar la decisión de si es conveniente o no la implementación total de la herramienta. Este método de conversión del sistema está estipulado para que dure hasta 4 iteraciones de los proyectos que se realicen en paralelo, de esta manera lo que se consigue es poder observar luego de cambios de versiones de los sistemas, cómo ha evolucionado la documentación en uno y no en otro. Se estima que este tiempo puede durar aproximadamente 2 meses.

Los pasos para llevar esta conversión a cabo son:

1. Selección de funcionalidades del piloto: Es importante recordar que lo que se pondrá a prueba es un piloto y no la herramienta terminada, por lo que es necesario definir cuáles serán las características en las que el piloto deberá enfocarse y a cuales no se les dará tanta importancia en el desarrollo de la prueba. En este caso se probará el módulo de generación automática de documentación.
2. Definición de las métricas que se evaluarán de dichas funcionalidades: Se estipulan las mediciones que se pretenden estudiar con la implementación de este piloto, junto con los valores y resultados que se espera obtener en estas cuando se realice la evaluación. Los indicadores en este caso serán: Cantidad de diagramas de clases producidos, parecido entre la planificación del arquitecto de software con el resultado final, Claridad de la documentación generada (medida en base a la similitud de esta con el entregable y la cantidad de documentación respecto al mismo).
3. Desarrollo y entrega del piloto: Deben desarrollarse las funcionalidades definidas en el punto anterior y generarse un entregable que sea presentado a quien tenga la autoridad correspondiente para ponerlo en implementación. Esto incluye tanto planificación y desarrollo de los módulos necesarios para la prueba. No incluye el análisis ni el diseño porque, en gran medida, lo definido en esas etapas no es intrínseco a la prueba, sino al proyecto en sí, por lo que ya deberían estar definidos los modelos como diagramas de clases o la representación de la información en la bases de datos, y en este punto sólo deberían tomarse y limitarse para satisfacer los requerimientos a evaluar del proceso de conversión.
4. Despliegue del piloto: Como el despliegue del producto terminado no tiene coste, se despliega entonces de la misma manera el piloto. Se crean los clusters de Kubernetes en Okteto con las réplicas del piloto y opcionalmente también la base de datos, el generador de diagramas y conectores a otras APIs web que el piloto necesite en caso de que se haya establecido una métrica que los requiera durante el paso número 2.
5. Definición de los criterios de selección para elegir los proyectos a comparar en la prueba: Es importante que los proyectos cuyo desarrollo será comparado muestren homogeneidad y a su vez presenten factores que permitan que el sistema enseñe sus capacidades y limitaciones. En este caso los proyectos deben ser ambos

desarrollados en microservicios, implementar el mismo framework y utilizar al menos 2 servicios de infraestructura.

6. Selección de los proyectos: En base a los criterios previamente definidos, se seleccionan dos proyectos para ser evaluados. En esta etapa también se determina el equipo de desarrolladores que trabajará sobre cada uno y se planifican los entregables que deben entregar.
7. Capacitación sobre el uso del piloto: Se le da una capacitación breve al equipo que desarrollará y documentará su proyecto con Junkode, haciendo especial énfasis en la documentación existente en forma de manuales, videos y guías de inicio rápido.
8. Desarrollo de los proyectos: Se comienzan a desarrollar los proyectos y se evalúan los resultados con cada entregable, en forma de opiniones de los desarrolladores, del líder del proyecto y del arquitecto de software del proyecto. Esto se hace con cada uno de los cuatro entregables planteados para la prueba.
9. Evaluación de resultados: Una vez terminado el cuarto entregable, se evalúan los resultados, o sea la documentación del código (y en menor medida el código en sí), junto con una recapitulación de las opiniones de los involucrados en el desarrollo de la prueba para poder llegar a conclusiones que permitan determinar si se debe seguir perfeccionando la herramienta y probar otro piloto antes de pasar a producción, o si la herramienta puede adoptarse de manera definitiva.

Anexo n°1: Análisis de Riesgos_V_2

Amenaza	Descripción	Probabilidad antes de la medida	Impacto antes de la medida	Riesgo antes de la medida	Medidas Preventivas	Medidas correctivas	Probabilidad después de la medida	Impacto despues de la medida	Riesgo después de la medida
Caida de servicio de despliegue	Okteto puede quedar fuera de srvicio por muchos motivos, y esto afectaría directamente al proyecto, no solo porque habría que buscar otra alternativa gratuita para su despliegue.	1	4	4		Se tendrá un plan de contingencia en caso de que se caiga, deployándose en otro servicio de deploy (disminuye el impacto).	1	2	2
Caída de servicios auxiliares	Hay múltiples herramientas de las que depende el proyecto que, de caerse, dejarían el proyecto sin funcinonar. Esto imposibilitaría poder siquiera hacer pruebas en el proyecto, y tener que buscar nuevas alternativas puede afectar el análisis de factibilidad.	1	3	3			1	3	3
Herramientas de uso propietario	Junkode es factible actualmente porque ninguna de las herramientas que utiliza son de naturaleza propietaria, por lo que si durante el desarrollo, estas herramientas cambian sus políticas de acceso, Junkode se verá directamente afectado y posiblemente obligado a cambiar de herramienta.	1	3	3			1	3	3
Cambios de la Planificación	Ya sean por cambios impuestos por la cátedra, por entes de fuerza mayor o por el mismo equipo, tener que hacer cambios sobre el conograma de trabajo planteado a principio de año puede afectar los tiempos del proyecto.	1	3	3	El proyecto fue planificado con la herramienta ProjectLibre la cual provee la funcionalidad de acomodar las tareas en base a fechas limite y no a la duración de la misma. Utilizandola, automaticamente se armó el primer conograma de actividades, dejando tiempos de holgura para el desarrollo de muchas de las tareas que lo componían.		1	2	2
Incrementos inesperados de complejidad del proyecto	Estos incrementos pueden ser repentinos y, como el nombre de la amenaza lo indica, inesperados. Estos pueden ocurrir curante las etapas de programación y despliegue, y pueden acarrear grandes demoras al proyecto. Ejemplos podrían ser: Problemas de compatibilidad por versionado de las herramientas, aparecición de nuevas limitantes al momento de realizar la integración de tecnologías o incluso simples errores de las herramientas empleadas que no cuenten con documentación para ser resueltos.	4	3	12		Se han elegido herramientas muy frecuentadas por desarrolladores, por lo que se cuenta con una gran comunidad de personas en foros para solventar los errores en caso de que ocurran. De igual forma se cuentará irectamente con el asesoraciento necesario de expertos en las tecnologías que puedan originar estos problemas. Se cuenta actualmente con contactos expertos en Java, C#, .NET, Docker, Kubernetes y Okteto.	4	1	4
Nuevas tecnologías de desarrollo	Normalmente la aparición de nuevas tecnologías que mejoran las funcionalidades de las que se tienen es algo bueno, pero esto puede ser altamente perjudicial si ocurre durante el desarrollo del proyecto. Un año de desarrollo es mucho tiempo para que muchas tecnologías aparezcan y otras se queden obsoletas. Hacer un cambio una herramienta como el lenguaje de programación, framework o base de datos cuando el proyecto ya está en marcha, puede traer problemas para el proyecto. No solo pueden afectar su planificación, sino también la entrega del proyecto en sí. Esto sin embargo no tiene tanto impacto si se tiene en cuenta que casi siempre es posible dar marcha atras y volver a las herramientas anteriores.	2	3	6			2	3	6
Deserción de integrantes del proyecto	Un integrante del grupo abandona el proyecto, lo cual provoca que retrasen los tiempos y puede dar lugar a que ocurran conflictos.	1	3	3			1	3	3
Ciberataques	Los ciberataques pueden dar de baja el servicio inundandolo de peticiones o incluso encontrando vulnerabilidades en los sistemas de datos. Esto es preocupante porque los servicios de bases de datos, aunque vacios, se encuentran ya desplegados y accesibles mucho antes del despliegue del proyecto, por lo que pueden ser tomado desde entonces y hasta pueden ser puestos bajo ransom.	2	2	4			2	2	4
Limitación debido a los recursos	Al usar solo herramientas open source, ocurre una limitacion de los programas y funcionalidades a los que se pueden acceder.	2	3	6			2	3	6
Incremento drástico de consultas	El sistema está implementado sobre servicios libres, por lo que los recursos con los que cuenta son limitados y la concorrencia de muchos usuarios consultando al sistema podría inhabilitar el servicio, ya sea solicitando mas instancias de servicios que los nodos de los que dispone okteto o llenando las bases de datos o los buckets	4	4	16	Optimizar al máximo los recursos en Okteto con multiples cuentas y almacenar mas metadata que documentos por proyecto (disminuye la probabilidad).	Planificar y disponer de alternativas con mayores rescursos listas para el despliegue del sistema (disminuye el impacto).	2	3	6

ANEXO N° 12 - TRABAJO PRÁCTICO INTEGRADOR “GERENCIAMIENTO DE SISTEMAS”

JUNKODE - Documentador Automático de Código Fuente

PROYECTO FINAL

TRABAJO PRÁCTICO INTEGRADOR
GERENCIAMIENTO DE SISTEMAS

Junkode

2022

Integrantes:

(45185) Céspedes Ortega, Rodrigo Gabriel
(45352) Fernández Quatrini, Renzo Abel
(45241) Flores, Sebastián Andrés
(45106) **Girala, Ramses | Coordinador**
(38492) Groisman, David Nathaniel

En este trabajo se trabajó sobre un Departamento de Sistemas de una empresa virtual brindada por la cátedra con la siguiente estructura organizativa.

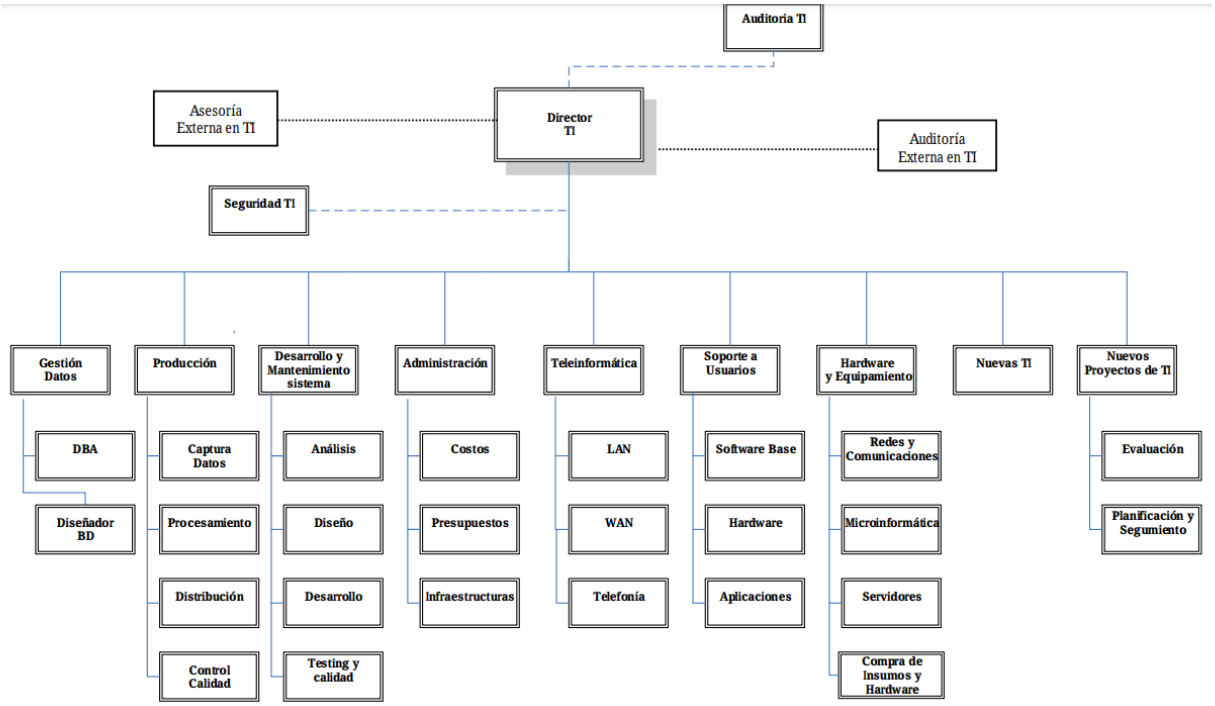


Figura 1 - Organigrama del Departamento de Sistemas brindado por la Cátedra.

El Área correspondiente al Grupo 1 para las consignas a las que les corresponde un área específica del Departamento de Sistemas, es el Área de Gestión de Datos. Esta Área se presupone compuesta por dos Administradores de Bases de Datos y dos Diseñadores de Bases de Datos, de la siguiente manera:

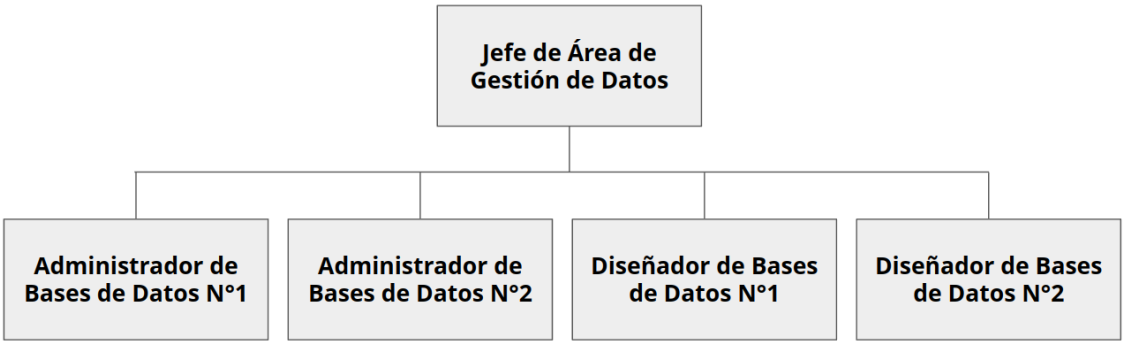


Figura 2 - Organigrama del Área de Gestión de Datos.

1. **La empresa está por construir un edificio nuevo de Data Center. Para ello está nivelando el terreno donde construirá el edificio, en una sola planta, de 500 m2. Detallar principales recomendaciones generales "técnicas y de seguridad física" para el Data Center, tanto para la fase de construcción del edificio como para toda la infraestructura, amoblamientos e instalaciones que sean necesarias.**
   Para la construcción de un Data Center se distinguieron las recomendaciones en 2 aspectos: recomendaciones para la construcción externa y medidas de seguridad del lugar físico del Data Center, y recomendaciones para construcción y distribución del Data Center en sí.

   En los Data Center hay muchos factores que pueden ser determinantes para su construcción. Uno de los factores que hay que tener en cuenta es la ubicación del edificio. Esto se debe a que si está en una zona sísmica, se recomienda que todo el edificio tenga características sismo-resistentes (por lo que también los racks donde van a estar los equipos tiene que estar anclados a la estructura para rigidizar y estabilizar la estructura). En general hay que tener en cuenta que no existan riesgos naturales que puedan afectar al Data Center y también los servicios de proveedores de Internet, emergencias disponibles (como bomberos y policía). También, se recomienda poner los Data Center en un primer piso. Esto se debe a que los cables de fibra óptica se manejan a esas alturas, por lo que es mejor a la hora de la instalación. Por otro lado, hay medidas de seguridad que tienen que estar presentes a toda hora. En la siguiente lista se detalla medidas de seguridad a tener en cuenta:
   - **Detectores de Humo:** Se recomienda que tenga estos dispositivos dentro del edificio y del Data Center para poder detectar posibles incendios.
   - **Matafuegos:** Se aconseja que el matafuego que se encuentre próximo al Data Center sea de HCFC (hidroclorofluorocarbono), que es específicamente para componentes electrónicos, ya que cualquier otro tipo de matafuego puede hacerle daño a los equipos del Data Center.
   - **Tablero Eléctrico:** Es muy recomendable que el Data Center tenga su propio tablero eléctrico con sus propias térmicas y conexiones. Esto se debe a que consume mucha energía, por lo que cualquier suba o baja de tensión, puede llegar a afectar a otros equipos que estén conectados a ese tablero.
   - **Salidas de Emergencias:** El Data Center debe contar con una salida de emergencia, además de poseer la señalización necesaria indicando la salida de emergencia.
   - **Accesibilidad:** Los Data Center no pueden ser accedidos por gente que no esté autorizada, por lo que se debe poner de forma "aislada" al contacto de la gente que no puede acceder. Además, se recomienda incluir seguridad especial en sus entradas como puerta con lectores de huellas o códigos de seguridad para su apertura.

- **Estabilizadores:** Los protectores de voltaje se aconsejan para proteger a los equipos de subidas de tensión. Además, se pueden usar en sinergia con sistemas UPS y grupos electrógenos, en varios tipos de configuraciones (paralelo, en serie o combinación).
- **Protectores de Red:** Estos protectores se recomiendan porque raramente los servicios de Internet, por los cables de red, generan una alta tensión y puede quemar los equipos de red del Data Center.
- **Cañerías de Servicios:** No debería haber instalaciones de agua ni gas dentro ni cerca del Data Center. Ya que cualquier problema de estos servicios, puede dañarlo.

Para la construcción interna del Data Center se tiene que saber que tipo de Data Center se necesita, ya que el tamaño de estos influyen mucho en los elementos y características del mismo. Partiendo por el suelo y el techo, se recomienda que sea flotante, por 2 razones. Primero, porque a la hora de tener que realizar el cableado estructurado, se puede plantear por debajo o por encima del mismo. Segundo, para la refrigeración de los equipos del Data Center, porque una característica importante es la temperatura que pueden llegar a generar. Siguiendo con lo último, es recomendable tener una distribución de pasillos fríos y pasillos calientes. Lo que se busca con esta organización es que los equipos tomen aire frío por la parte frontal y que expulsen aire caliente por la parte trasera. Además, si se utiliza este método, se recomienda realizar el cerramiento de los pasillos fríos. Esto logra que haya una temperatura homogénea entre el piso y el techo (hasta 10°C).
De la misma forma que el anterior, se listará algunos elementos primordiales para un Data Center:

- **Aires Acondicionados:** Se recomienda poseer mínimo 2 aires acondicionados para el Data Center, de esa forma se puede turnar el uso de los aires, con algún dispositivo (ej.: Arduino que soporte protocolo MQTT), ahorrando el consumo eléctrico, y haciendo que descansen los aires.
- **UPS:** Este dispositivo es muy aconsejable ya que mediante un apagón no programado, los dispositivos eléctricos continúan trabajando. Además, algunos los UPS tienen otra función como la corrección de ciertos fallos del suministro eléctrico (sobretensión, bajas de tensión, inestabilidad de frecuencias, entre otras).
  Estos deben contar con una autonomía suficiente para mantener alimentados los dispositivos mientras se apagan los servicios y se hacen checkpoints de las bases de datos. El tiempo necesario para esto dependerá de si estas tareas son manuales o si están automatizadas. En el caso propuesto, es manual, pero con la posibilidad de hacerlo remoto, por lo que 10 minutos de autonomía sería un buen tiempo objetivo.
- **Equipamiento Blanco e Iluminación:** Es recomendable que los racks y los equipos que estén a disposición sean de color blanco, ya que según un estudio de Graeme Shaw, se obtiene un ahorro energético al usar muy poca iluminación. Esto se debe a

que el color blanco es un 89% más reflectario que el color negro (4%), por lo que con menos iluminación, se logra visualizar todo el Data Center, al igual que un ahorro energético. Además, siempre es recomendable la utilización de focos o tubos led que tienen un menor uso energético.

- **Racks:** Como se dijo al principio, se tiene que saber que tipo de Data Center es el que se quiere construir. Para ello vamos a dar características generales que se tendría que tener en cuenta:
    - Tamaño: Se recomienda utilizar uno de 42U (1,86m altura) para los equipos del Data Center, y otros más pequeños 22U para cada uno de los pisos sobre los que se extenderá el cableado estructurado.
    - Ventiladores: Es aconsejable que se coloquen los rack que más temperatura adquiere, ventiladores que se colocan como bandejas en la parte superior del rack ayudando a la refrigeración
    - Ordenadores de cables: Los Data Center se caracterizan por tener muchos cables, por eso es recomendable que tengan pasacables con cepillo y guías de cables, para poder identificar cada cable. Además, se pueden incorporar etiquetas para facilitar esa identificación.
    - Regletas de alimentación (Zapatillas): Cada rack debería tener uno ya que es la alimentación de corriente a los equipos dentro.
    - Patch panel: Es uno de los accesorios más necesarios de los racks, ya que se encarga de organizar los cables dentro del rack. Los cables externos se conectan directamente al dispositivo, y en el interior se redirigen al dispositivo en cables más cortos y organizados.
    - Cajas ignífugas: Es muy aconsejable que los racks sean en cajas ignífugas, para la protección de los equipos en caso de incendio.
    - Cajas con aislamiento acústico: En caso de que se tenga que trabajar en el Data Center, se recomienda el uso de racks insonorizados para eliminar el ruido que genera los equipos trabajando dentro del mismo.
- **Back-Ups:** Para este servicio se recomienda cintas magnéticas. Estos son rollos con gran capacidad de almacenamiento y que permiten un fácil transporte. Para ello también se debería poseer un equipo de grabación de cintas.
- **Seguridad de Rack:** Se recomienda que cada rack posea una seguridad electrónica para su apertura (esto puede ser con seguridad biométrica) y videocámaras IP con un control 24 horas.
- **Cableado:** El cableado que se recomienda son de dos tipos:
    - Fibra óptica: Para el backbone, o la comunicación entre el Data Center y los racks de los diferentes pisos, se recomienda el uso de la fibra óptica de tipo multimodo (62,5/125 um). Además, de parte de los proveedores de Internet, también es recomendable que la conexión vaya por este medio.

    - Par trenzado: Por otro lado, la comunicación entre los equipos dentro del Data Center, es conveniente el uso del cable UTP Categoría 6, para una mayor inmunidad al ruido. Para los equipos de trabajo (computadoras de escritorio o notebooks), se recomienda el uso de cable UTP Categoría 6a, puesto que ofrece un mejor aislamiento al ruido que es apreciable para extensiones largas de cableado y es mucho mas barato que el cable UTP categoría 7.
- **Seguridad CCTV:** Se recomienda poseer una sala de control de cámaras y tener videocámaras con IP vigilando y controlando el acceso las 24 horas. En caso de que el edificio no tenga el espacio físico, se podría tercerizar contratando un agente externo.
- **Sensores:** Es aconsejable que se coloquen sensores de todo tipo dentro de un Data Center. Para ello se lista los sensores que son recomendados:
    - Temperatura: Para todos los racks y para el ambiente, tiene que haber sensores de temperatura para saber si el sistema de refrigeración está funcionando correctamente.
    - Humedad: Estos sensores se colocarían cerca de alguna posible cañería que rodee externamente por el Data Center. Se recomienda ya que la humedad puede dañar a los equipos.
    - Apertura de puertas: La apertura de puerta se recomienda controlar ya que puede haber alguna infiltración del personal, irrumpiendo el acceso al Data Center y violando la seguridad de los datos.
    - Humo: En caso de que algún equipo se incendie, estos sensores se activarían indicando lo sucedido. Por esta misma razón son recomendables.
    - Eléctricos: Las subidas y bajadas de tensión son muy comunes, por lo que se recomienda tener estos sensores, donde advierten cuando se producen.
- **Proveedores de Internet:** Es recomendable poseer 2 proveedores de Internet mínimo. Esto se debe a que las compañías que brindan este tipo de servicio, pueden sufrir algún problema y deja de funcionar correctamente, por lo que hay que tener otro proveedor como plan de contingencia.
- **Routers y Switchers:** Se recomienda poseer router y switchers distintos de los que te provee la empresa del servicio al Internet. Este equipamiento sirve para poder armar toda la topología de la red en la empresa.
    - Capacidad: Su selección debe prever siempre al menos un 10% de puertos libres, para permitir una escalabilidad rápida.
    - Velocidad: En cuanto a la velocidad de transmisión, es importante que tengan una capacidad de permutación superior a la de los servidores y demás equipos del Data Center, pero no necesariamente superior a la del cableado (O sea, si llega a producirse un cuello de botella en el futuro, es preferible que ocurra en el router o en un switcher a que ocurra en el cableado).

    - Servicios y garantía: Finalmente, otro punto a tener en cuenta son los servicios de soporte con los que cuentan. Generalmente los dispositivos de marcas CISCO, Huawei o Ubiquiti son administrados y cuentan con servicio, garantía y profesionales especializados disponibles, mientras que otras alternativas, no las ofrecen.
    - Alimentación: Se recomienda la utilización de dispositivos que soporten PoE, lo que quiere decir que el router y el switcher deben ser capaces de transmitir energía para los demás dispositivos (Esto no es fundamental).
    - Características adicionales: Para los switchers es importante corroborar si soportan VLANS o cuánta ram interna tienen. Y para los routers es importante corroborar bien cada uno de sus puertos, puesto que es común que algunos sean in/out y los cuenten por separado y que estos tengan velocidades de permutación distintas.
- **Wi-Fi:** No se recomienda que haya Access Points en el Data Center, puesto que habría pocos motivos para necesitar de dispositivos inalámbricos en él. Incluso es una medida de seguridad.
- **Zona desmilitarizada:** Es recomendable contar con una DMZ o zona desmilitarizada para el manejo de los administradores del Data Center y mantener un tránsito ágil dentro del mismo. Es importante mencionar que la infraestructura del cableado estructurado debe cuidar meticulosamente que no puedan existir formas de acceso a esta fuera de las controladas por la empresa. Por ejemplo, un error muy común sería colocar los access points del edificio en puertos de switchers dentro de la DMZ, de la misma manera que las estaciones de trabajo. Esto dejaría una vulnerabilidad expuesta a cualquier dispositivo móvil capaz de ingresar a la empresa.
  Los bastidores pueden ser de hardware, en forma de router, para lo cual solo se necesitaría uno, o en forma de software, en forma de computadoras pequeñas con una gran capacidad de permutación en este caso harían falta 2 (también pueden combinarse ambas técnicas para tener una intranet más segura).

Por otro lado, se recomiendan algunos servicios externos a tener en cuenta. Primero, el servicio de limpieza tendría que ser rápido y con productos que no dañen a los equipos. El personal de limpieza tendría que estar vigilado por las cámaras de seguridad y acompañado por personal del edificio. Segundo, se tendría que contratar un seguro de daños (terceros o fallo del equipo) y robo de equipos en el Data Center. Este seguro es para que se pueda recuperar parte de la inversión en el equipo y que se pueda restaurar lo más pronto posible.

2. **Si consideramos que trabajan, como mínimo, dos personas en cada una de las áreas detalladas, cuál es el tipo de estructura organizativa mostrada en el organigrama. Además, podría explicar cuáles otros tipos de estructuras organizativas podrían utilizarse.**

   El organigrama presenta una estructura organizativa por funciones. Este tipo de departamentalización se caracteriza por agrupar las tareas y actividades de una empresa de acuerdo con las principales funciones que se desarrollan.

   Esta estructura organizativa también facilita la formación del personal, fomenta la especialización del trabajo y facilita el proceso de control.

   Algunos criterios alternativos que se podrían aplicar son:

   - Estructura organizativa por procesos: Está asociada a las actividades con base a las etapas o fases en las que efectúa un proceso productivo. Al especializarse en cada fase del proceso se logra el ahorro del tiempo y el incremento de la eficiencia. Al aplicar este tipo de estructura, cada proceso va a pertenecer a un área específica para su desarrollo. Ejemplo para un proceso de respaldo y recuperación de datos, habrá un encargado de respaldo y recuperación de datos.

   - Estructura organizativa por cliente: Se utiliza cuando cada una de las actividades de la empresa son puestas bajo la responsabilidad de un jefe en función de sus clientes. El caso en el que esta estrategia de departamentalización podría tener resultar efectiva, sería si la empresa fuera una software factory que almacenará los datos de manera propia y on-premise. De esta forma, los trabajadores del Área de Gestión de Datos podrían abocarse a las bases de datos de un cliente en particular, lo cual resultaría efectivo, porque no se ocasiona ningún perjuicio al ignorar las demás bases de datos. Por otro lado, y justamente por este último motivo, es que no sería conveniente aplicar esta estrategia de departamentalización orientada a clientes internos de la empresa, puesto que no se podría alcanzar el grado de libertad necesario entre las bases de datos, y por lo tanto, tampoco entre sus administradores.

   - Estructura organizativa geográfica: Es común para empresas que operan en distintas regiones geográficas. Favorece a la formación del personal y permite una mejor coordinación. Esta forma de departamentalización podría aplicarse si el hardware que soporta a las bases de datos se encuentra distribuido en distintos datacenters, y que deba haber DBAs específicos para cada uno.

- Estructura organizativa Matricial: En esta se realiza una combinación de la departamentalización funcional y de producto o de proyecto. Esto genera una organización mixta, donde se combina la forma vertical de la forma de organización funcional y la horizontal de la organización de producto o de proyecto. Podría ser conveniente aplicarla en el Área de Gestión de Datos si se contase con muchas mas bases de datos que diseñadores DBAs tenga la empresa. De esta forma, los trabajadores podrían cambiar de proyecto en el que están trabajando rápidamente (aunque con las consecuencias que implica cambiar los objetivos y tecnologías de trabajo con regularidad).

- Estructura organizativa por productos: En este tipo de departamentalización se agrupan actividades de acuerdo a los resultados de la organización es decir por cada producto o servicio. Optimiza la utilización de tecnología, equipos y conocimientos en la generación de cada producto o servicio específico. En el caso de la gestión de datos, esta departamentalización puede ser conveniente cuando coexistan múltiples motores de bases de datos en una empresa, puesto que cada una requiere conocimientos y tecnologías diferentes y con ellas también varía la representación de la información y las funcionalidades a las que se tiene acceso.

3. **Detallar y explicar como mínimo seis servicios que brinde el área seleccionada (sea interna o externa a la empresa).**

- **Administración de las Base de Datos:** En este servicio los DBAs realizan todas las tareas referentes a las bases de datos en sí. Esto incluye creación, monitoreo, escalado, migración, recuperación, roll-backs, etc.

- **Seguridad:** Si bien las medidas de seguridad dentro de la empresa son tomadas por el staff de Seguridad-TI, estas medidas no pueden ser aplicadas sobre los equipos y metodologías del Área de Gestión de Datos sin que el Jefe de Área las autorice primero. Ejemplo de esto sería el encriptado de los datos en las bases de datos o la definición y gestión de roles y usuarios de las mismas. En ambos casos el staff de Seguridad-TI es quien determina las medidas a tomar, pero es el Área de Gestión de Datos la que es responsable de aceptar esas medidas, reglamentarias e implementarlas.

- **Accesos a la Base de Datos:** En todo momento se tiene que saber quién, cuándo y cómo se acceden a los datos. Es decir, que todos los registros tienen que tener auditoría. En caso de que se note algo inusual se puede generar una alerta sobre dicha actividad.

- **Back-Ups:** El servicio de backups se encarga de la copia de seguridad de la base de datos. Dependiendo del negocio de la empresa, se respaldan los datos más importantes en caso de algún ataque, corrupción de datos, falla de hardware o software, entre otros posibles eventos. Normalmente se guardan en discos HDD o en cintas magnéticas, y se realizan en intervalos de tiempo, es decir, semanalmente o mensualmente.

- **Archivos internos de las bases de datos:** Si bien hoy en día, con el auge de los microservicios, muchos desarrolladores gestionan muchos de los parámetros de sus bases de datos, es importante que siempre sea personal del Área de Gestión de Datos quien maneje los archivos más complejos de estas (controlfiles, datafiles, redologs, etc.).

  Desde estos, por ejemplo, es posible determinar de qué tamaño van a ser los archivos de datos de las bases de datos para así optimizar su almacenamiento físico (esto es fundamental cuando se usan dispositivos como cintas magnéticas), es posible también gestionar la cantidad de réplicas que puede tener un log y su distribución entre los medios físicos (por razones de resiliencia) o gestionar grupos de logfiles (para optimizar la frecuencia de los checkpoints y reducir el tiempo en estado idle del motor de la base de datos).

- **Resiliencia de las bases de datos:** La administración de bases de datos no se limita al software en sí, sino que también incluye la gestión del hardware que lo soporta y la de los propios datos en sí. Esto ofrece capacidad de resiliencia de las bases de datos frente a accidentes o ataques.

  Para el primer caso, esto implica seleccionar los servidores donde se encontrarán las raids, seleccionar la tecnología de la RAID, seleccionar las bases de datos que pertenecerán a cada una y encargarse del mantenimiento y escalado de estas estructuras.

  En el segundo caso implica la adopción de patrones de consulta de bases de datos que optimicen su integridad relacional. Ejemplo de esto sería promover el uso de patrones CQRS físicos, donde las operaciones de escritura de las bases de datos se realicen sobre elementos de almacenamiento con controles complejos y que reflejen imágenes de sus estados en dispositivos de almacenamiento más rápidos para las operaciones de lectura.

4. Con ejemplos del área seleccionada, explique las características de un equipo de trabajo efectivo y un equipo de trabajo equilibrado.

En el área de gestión de datos se desempeñan dos roles:

El Administrador de bases de datos (DBA), quién se encarga de:

- Asegurar el buen funcionamiento de las bases de datos.
- Evitar la pérdida de datos y reparar las que eventualmente ocurran.
- Gestionar los procesos de Back-up, restauración y migración de las bases de datos.
- Dar soporte y mantenimiento al software y servicios de bases de datos.

El Diseñador de bases de datos:

- Diseñar los modelos relacionales o de almacenamiento no normalizado de los modelos de datos provistos por los analistas de sistemas.
- Planificar las interacciones que las bases de datos tendrán entre sí.
- Buscar y fomentar la construcción de bases de datos consistentes, escalables y fácilmente mantenibles.

Equipo de trabajo efectivo: en este se entabla mucha comunicación entre los roles para poder diseñar las bases datos, con la cual hay retroalimentación y se establecen las metas claras y precisas.

Por definición, un equipo de trabajo equilibrado cuenta con las siguientes características:

1. Libre expresión de todos los miembros.
2. Principio del trabajo en conjunto, que se logra mediante una delegación eficaz del líder, generando sinergia entre los miembros del equipo de trabajo, cuando los resultados del trabajo en conjunto son mejores que los resultados del trabajo individual.
3. Todos están dispuestos a asumir riesgos, ya que hay una adecuada planificación y gestión de riesgos de parte del líder.
4. Existe espíritu de coaching entre todos los integrantes del equipo, mediante la aplicación de las principales actividades del coaching: Saber escuchar de distintas fuentes y estar atento a lo que le ocurre o piensa cada persona de su equipo, acompañar a cada uno en situaciones difíciles o que no se sabe cómo continuar, proveer los recursos necesarios, contener anímicamente y ayudar en todo lo que fuere necesario para cada persona.
5. Hay objetivos comunes y metas claras bien arraigados en todos los miembros.
6. Existen iniciativas, deseos y voluntad de participación, respeto por todos y siempre los miembros están dispuestos a colaborar.
7. Aceptación de decisiones por consenso general, aún cuando existan divergencias individuales.

8. Buena relación de los miembros con otros integrantes de otros proyectos y otras áreas, para aprovechar las experiencias ajenas y poner en valor las propias.
9. Retroalimentación de todos los integrantes del equipo de trabajo a los efectos de pensar y poner en práctica permanente acciones de mejora continua.

➢ Ejemplo: Si se dispone de dos DBAs y dos diseñadores y se debe trabajar en dos bases de datos distintas, se establecerá una división de trabajo donde el diseñador se comunicara con un DBA específico de la base de datos a trabajar (ambos consignados a un proyecto específico ya preestablecido del Área de Gestión de Datos). El diseñador decidirá cómo estructurar el modelo relacional pero si tiene problemas de cómo hacerlo será posible pedir la opinión a otro diseñador para tener otro punto de vista. Es imprescindible tener la mente abierta a escuchar opiniones de los demás miembros para lograr un trabajo efectivo.
El diseñador presenta el modelo relacional al DBA y este se encargará de trabajar sobre el modelo conceptual para que se pueda implementar en las tecnologías disponibles. Si no es posible implementarlo, se debe hablar con el diseñador para que reestructure el modelo relacional con las sugerencias que el DBA proporcione.
Entre los diseñadores también debe de establecerse una buena comunicación con el fin de hallar consenso sobre cuáles serán los límites de las bases de datos. Esto último es específicamente para no entorpecer el trabajo de los DBAs.

En el anterior ejemplo pueden encontrarse incluidas todas las características de Equipos de Trabajo Efectivos previamente mencionadas:

En las primeras dos oraciones del primer párrafo se puede observar que cada par “DBA y Diseñador” conforman un equipo efectivo particular donde se manifiestan las características n° 5 y 7 postuladas.

A continuación, en las últimas dos oraciones del primer párrafo se pueden encontrar las características n° 4, 6, 8 y 9, mostrando que no solo cada par de trabajadores del Área se comporta como un Equipo de Trabajo Efectivo, sino que el conjunto de todos los pares (o sea, el Área en toda su extensión) es un Equipo de trabajo Efectivo en sí.

En el segundo párrafo puede inferirse que las decisiones que toman los DBA y desarrolladores son acordes a la tercera característica y que se encuentran previstas dentro de los riesgos esperados de la planificación.

Finalmente, el último párrafo hace alusión directa las características n° 1 y 9, mientras que la característica n° 2 es un resultado que se alcanza como consecuencia del conjunto de

actividades realizadas en durante el desarrollo de la tarea el ejemplo y del cumplimiento de todas las demás características propias de un Equipo de Trabajo Efectivo durante este.

Equipo equilibrado: en este se establecen las tareas que se desempeñan sin establecer retroalimentación. Se dictan directamente a realizar las actividades sin opiniones o sentimientos personales que se involucren en el medio del desarrollo.

Por definición, un equipo de trabajo equilibrado cuenta con las siguientes características:

1. Cantidad de integrantes, de acuerdo con recomendaciones de alcance de control del líder.
2. Disponibilidad de tiempo.
3. Necesidades personales y fines propios.
4. Actitud (positiva, negativa, colaboración, egoísta, etc.)
5. Roles (orientado a la tarea, orientado a la relación, etc.)
6. Personalidad (introvertido, extrovertido, agresivo, sumiso, solitario, etc.)
7. Competencias técnicas y nivel de capacitación.
8. Ingenio, creatividad, generación de ideas, inquietudes, nuevos proyectos, etc.
9. Adaptabilidad al estrés.

    - ➢ Ejemplo: Si se desarrollan dos bases de datos, deberá definirse quienes trabajan en esas bases de datos y cómo se dividirán las gestión de cada uno. Los diseñadores tienen que realizar modelos relacionales que se adapten de forma adecuada al sistema y los administradores se enfocan en implementarlo de la mejor manera en el sistema. Cada persona se enfocará en la tarea que le fue dada sin intervenir con los demás miembros del equipo.

En la primera oración del ejemplo es donde se hacen presente las características n° 1, 4, 5, 6 y 7; las cuales son alusivas a la selección de los miembros de un equipo, en este caso, enfocado para la asignación de tareas.

En la segunda oración se aprecia la distinción de tareas entre los distintos roles definidos anteriormente. Esto facilita (aunque no siempre garantiza) el cumplimiento de la octava característica, fomentando justamente todo lo que en esta se postula.

Finalmente, en la última oración del ejemplo pueden inferirse los principios n° 2 y 3, puesto que, dado el grado de independencia que presenta la tarea, los integrantes del Área de Gestión de Datos pueden realizar sus actividades sin estar ligados estrictamente a otros miembros de la misma área.

5. **Analizar la aplicación del "Coaching Eficaz" en el área seleccionada. O sea, de qué forma revelaría la situación del personal y cuáles acciones realizaría Ud. como Jefe del área seleccionada para poder aplicar correctamente el coaching.**
   Como Jefe de área es fundamental tener en cuenta dos cosas antes de comenzar un proceso de coaching: La primera es que lo recomendable es que el proceso no sea organizado ni dirigido por uno mismo, sino que por un coach profesional (y preferentemente con experiencia con equipos de trabajo similares a los del departamento). Esto es importante porque el Jefe de Área también debe participar como coachee durante el coaching, por lo que no puede tomar el lugar de coach (De hecho, el Jefe de Área será el primero en someterse a un coaching, debido a que este proceso se realiza en orden jerárquico, en este caso va primero el Jefe del área de Gestión de Datos, luego DBAs y luego los Diseñadores de Bases de Datos).

   El segundo factor a tener en cuenta para la aplicación de un coaching eficaz, es que los coachees deben estar dispuestos a participar de la actividad de coaching. De nada servirán los procesos para ayudar a los empleados del área si estos no reconocen primero que deben ser ayudados.

   Estos dos puntos mencionados sirven para entender que la aplicación de un coaching eficaz contempla factores que están fuera del dominio del Jefe de Área (y que de hecho, él mismo participa del proceso de la misma forma que sus empleados). Con esto ya aclarado, se procede a enumerar los pasos de un proceso de coaching a modo de sugerencia del Jefe de Área:

   1. Identificación de necesidades: Se necesitan identificar en primer lugar las dificultades del coachee para esclarecer sus metas. Para esto es necesario llegar a un acuerdo entre el coach y el coachee y hacerle ver al último que estas acciones tendrán impacto en su vida diaria y en su desempeño (De esta forma se busca cumplir con lo planteado en los párrafos anteriores).
   2. Realizar un Feedback: Si bien esto depende casi completamente del Coach (tanto para la confección de los cuestionarios e implementación), si puede el Jefe de Área sugerir la metodología de Feedback a implementar. En este caso se recomienda Retroalimentación 360° por los motivos explicados en el siguiente punto del Trabajo Práctico Integrador.
   3. Plan de acción: Esta etapa está orientada a generar opciones, llevar a cabo o ejecutar las acciones necesarias y trabajar en la mejora de las competencias más relevantes para lograr los objetivos.
   4. Seguimiento: En esta etapa se ayuda a identificar los obstáculos, se establecen estrategias para superarlos y se refuerza cada uno de los logros que vaya alcanzando el coachee.

5. Evaluación: Esta se basará en el plan del coaching y en los objetivos que se plantearon. Generalmente se realizan diversos cuestionamientos para hacer una mejor valoración del proceso y se revisan los resultados donde se puedan verificar los progresos del coachee y los hábitos nuevos generados.

Una vez que se cuente con los resultados de la evaluación, el Jefe del área de Gestión de datos, junto con el coach determinarán cómo continuará el proceso de coaching, o sea, si será necesario continuarlo o no. Si no se alcanzaron las metas propuestas por los coachees, es recomendable continuar con el proceso de coaching sin interrupciones. Sin embargo, si no solo no se alcanzaron las metas, sino que tampoco se contemplan cambios de mejora de comportamiento de los empleados, puede que estos no estén convencidos realmente del programa de coaching. En este último caso, la única situación en la que el Jefe de Área puede hacer algo es cuando el problema de convicción de los empleados es el coach. No todos los empleados pueden desarrollar la afinidad necesaria con un coach para obtener los resultados de un coaching eficaz, por lo que una posible solución sería cambiar al coach por otro. Sin embargo esto solo serviría cuando se esté seguro de que la razón por la que el coaching no da resultados, es el coach.

Lo que se espera de la aplicación del Coaching es poder apreciar los beneficios propios de sus funciones una vez que se haya efectuado. En definitiva, se espera que las tareas de coaching:

- ➢ Permitan descubrir puntos ciegos en la forma de ver, actuar y conducirse de los coachees, sin limitarse estrictamente al ámbito laboral ni empresarial.
- ➢ Generen contexto para vislumbrar problemáticas actuales o potenciales.
- ➢ Propongan planes de acción (tanto particulares como grupales) que permitan el logro de objetivos. Estos planes pueden generarse en forma de estrategias o series de pasos, mientras tengan en común la determinación de un lapso fijo.
- ➢ Brinden herramientas para una mejor comunicación dentro del Área.
- ➢ Motiven y apoyen el compromiso de los coachees, tanto con ellos mismos como con sus acciones.
- ➢ Consoliden oportunidades de crecimiento o mejora.
- ➢ Contribuyan al desarrollo de mecanismos de toma de decisiones.

6. **Analizar la aplicación de “Retroalimentación a 360°” en el área seleccionada. O sea, cuáles serían todas fuentes de información y acciones que Ud. aplicaría como Jefe del área seleccionada para poder aplicar correctamente la retroalimentación a 360°, para mejorar su propia gestión a cargo del área.**

La idea de realizar una retroalimentación 360° sobre el Área de Gestión de Datos es contemplar puntos de vista distintos a los de los cargos jerárquicos superiores, como se

realizaría la retroalimentación tradicionalmente. Esto quiere decir que las partes que vayan a dar una retroalimentación no pueden ser solamente seleccionadas por el Jefe de área, puesto que esto también rompería con el propósito de la retroalimentación 360°.

Es por esto que lo primero que se debe hacer en el procedimiento de ejecutar una retroalimentación 360° adecuada, es contratar los servicios de un coach profesional para gestionarla.

Si bien será el profesional quien seleccione finalmente las partes que brindarán su opinión en el proceso de retroalimentación, el Jefe del área de Gestión de datos será quién deba aportarle la información necesaria sobre el negocio y los trabajadores. Dentro de esta información se encuentra una lista de retroalimentadores sugeridos:

- El mismo Jefe del área de Gestión de Datos.
- Los compañeros de trabajo de quién estará bajo estudio (si es un DBA, los demás DBAs).
- Jefe del área de Hardware y equipamiento junto con el Encargado de Compra de Insumos y Hardware (quienes pueden dar una referencia acerca de insumos que pueda haber solicitado el trabajador bajo estudio).
- Staff de Seguridad TI (puesto que al ser un ente ajeno a la organización pueden aportar información que no sea obtenible por otros medios).
- Área de Recursos Humanos (en caso de que la empresa cuente con una).
- Cualquier otra persona o conjunto de personas que el coach profesional considere pertinente para la aplicación de la retroalimentación 360°.

Finalmente, también se sugeriría el rango de tiempo sobre el que se generarán las devoluciones, estos pueden ser los últimos 6 meses, el último proyecto realizado, o desde la última retroalimentación 360° realizada. Nuevamente, esto es será solo una sugerencia y quedará en manos del profesional decidir cuál será el plazo de tiempo a evaluar.

7. Detallar las funciones que podría tener un Tablero de Comandos del área seleccionada y el diseño de la pantalla principal del mismo.

Se pueden clasificar los elementos presentes en un Tablero de Comandos del Área de Gestión de Datos en dos grandes grupos: Los Dashboards (los cuales contienen información estadística e indicadores propios de las bases de datos) y las herramientas de gestión de bases de datos (Programas especializados para la administración de bases de datos, backups, migraciones, auditoría, etc). A continuación se describirán los elementos principales con los que deberá contar cada uno de estos grupos (Es importante aclarar que para este ejemplo, se presupone un entorno de múltiples bases de datos que comparten el mismo paradigma de diseño, en este caso es un paradigma relacional SQL):

Dashboards:

Las principales entradas de información con las que debe contar un tablero de comandos de un Área de Gestión de Datos, serían provenientes de las propias bases de datos, o sea, serían mediciones cuantificables y representables a través de gráficos. Los gráficos que podrían encontrarse en estos tableros de comandos serían:

- **Capacidad de almacenamiento disponible:** Se puede representar en un gráfico de tipo pie chart que muestre el almacenamiento que posee el dispositivo físico (o servicio en la nube) que soporta las bases de datos. Esto se puede clasificar con una muestra dinámica observando el espacio libre, el ocupado y una predicción de cuánto se estima ocupar en un lapso de tiempo determinado, como 1 año.

- **Consultas a la Base de Datos:** Es un indicador que muestra las consultas que una base de datos específica tiene en un lapso de tiempo determinado. Esto se puede representar con un gráfico de línea (time series) que contenga la cantidad de consultas en un periodo de tiempo determinado. Estos intervalos de tiempo deben ser seleccionables por el usuario que los consulte

- **Población de tablas y colecciones:** Esta función muestra en un gráfico de barras la cantidad de registros que se crearon en cada tabla o colección de una base de datos específica. Para ellos se pueden filtrar estas consultas por periodicidad (año, semestre, trimestre, mes). En caso de que haya tablas con atributos que relacionan con otras tablas (Foreign Key), se podrá acceder a los datos de esa tabla relacionada, con el mismo gráfico y la misma periodicidad. Para poder navegar entre tablas y colecciones, se realiza con los botones al lado izquierdo del gráfico.

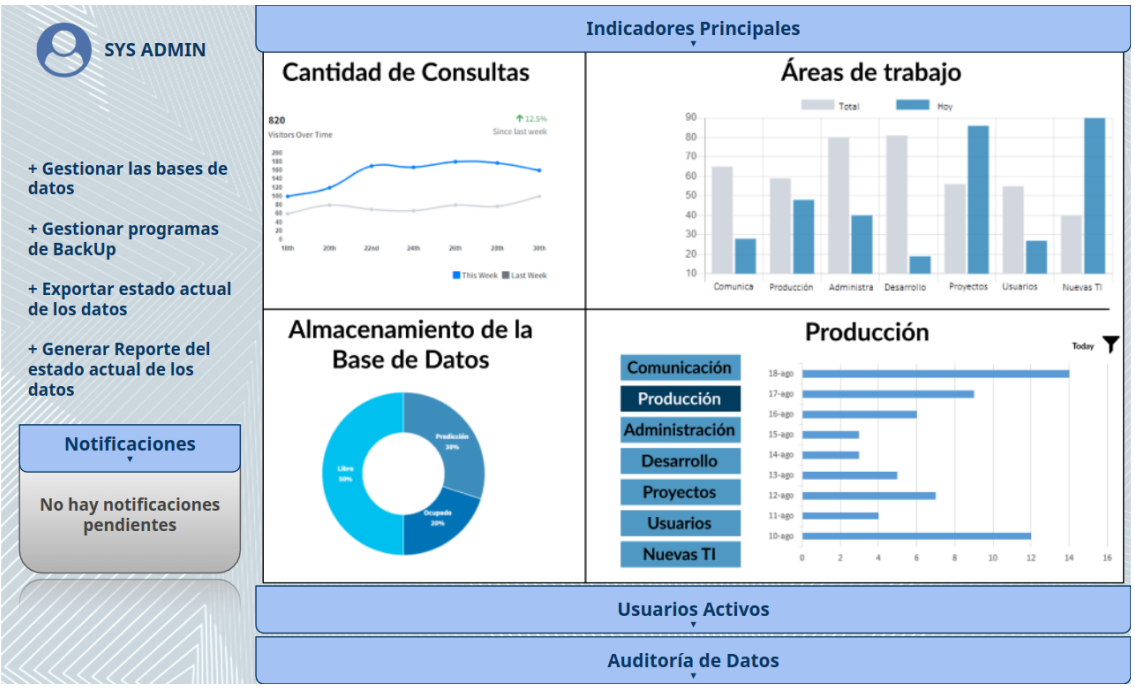


Figura 3 - Dashboards del tablero de comando del Área de Gestión de Datos.

Herramientas de Gestión:

Además de los dashboards, el Jefe del Área de Gestión de Datos debe tener acceso a información de control sobre otros aspectos de referidos a su área, como los siguientes:

- Lista de usuarios conectados, pudiendo filtrarlos por roles, con el fin de ver específicamente a los usuarios administradores conectados a la base de datos.
- Acceso a la auditoría de las tablas de la base de datos.
- Acceso a la información e historial de back ups creados y almacenados de cada base de datos.
- Herramientas que permitan definir programas de back up (como calendarios, elementos a ignorar, encriptación del respaldo, lugar de almacenamiento, periodos de conservación y eliminación automática, etc).
- Un elemento que permita la generación instantánea de un back up.
- Herramientas de migración de bases de datos.
- Herramientas de monitoreo del estado de los elementos de almacenamiento físico que permitan ver cuando una raid se encuentre inaccesible, o cuando un servidor NAS esté fuera de servicio.
- Un elemento que permita poner en modo Sleep a los dispositivos de almacenamiento.

- ➢ Alertas que llamen la atención de sus usuarios cuando sea necesario (Por ejemplo, cuando se registren conexiones de un usuario que no se supone que deba estar conectado).
- ➢ Elementos que capturen la información de los dashboards e indicadores y la desplieguen en forma de reportes para su consulta.

Es importante mencionar que no es imprescindible que todos estos elementos se encuentren en el tablero en sí, de hecho es recomendable que el tablero tenga enlaces a herramientas especializadas que controlan uno o múltiples elementos de la lista (como gestores de bases de datos o gestores de “Jobs” de Back Ups).

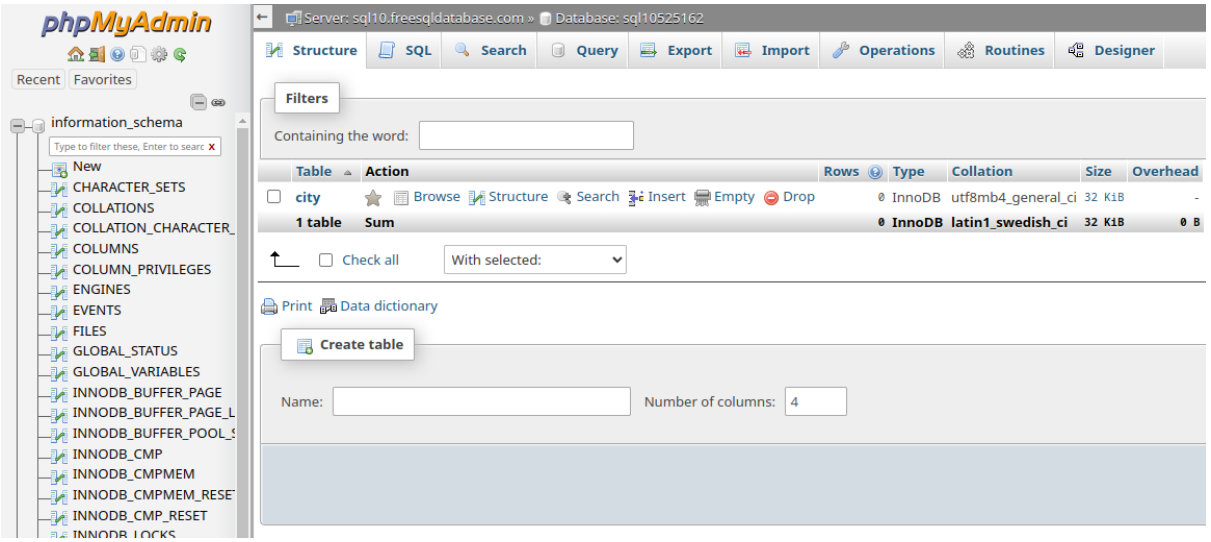


Figura 4 - phpMyAdmin: Gestor Web de bases de datos.

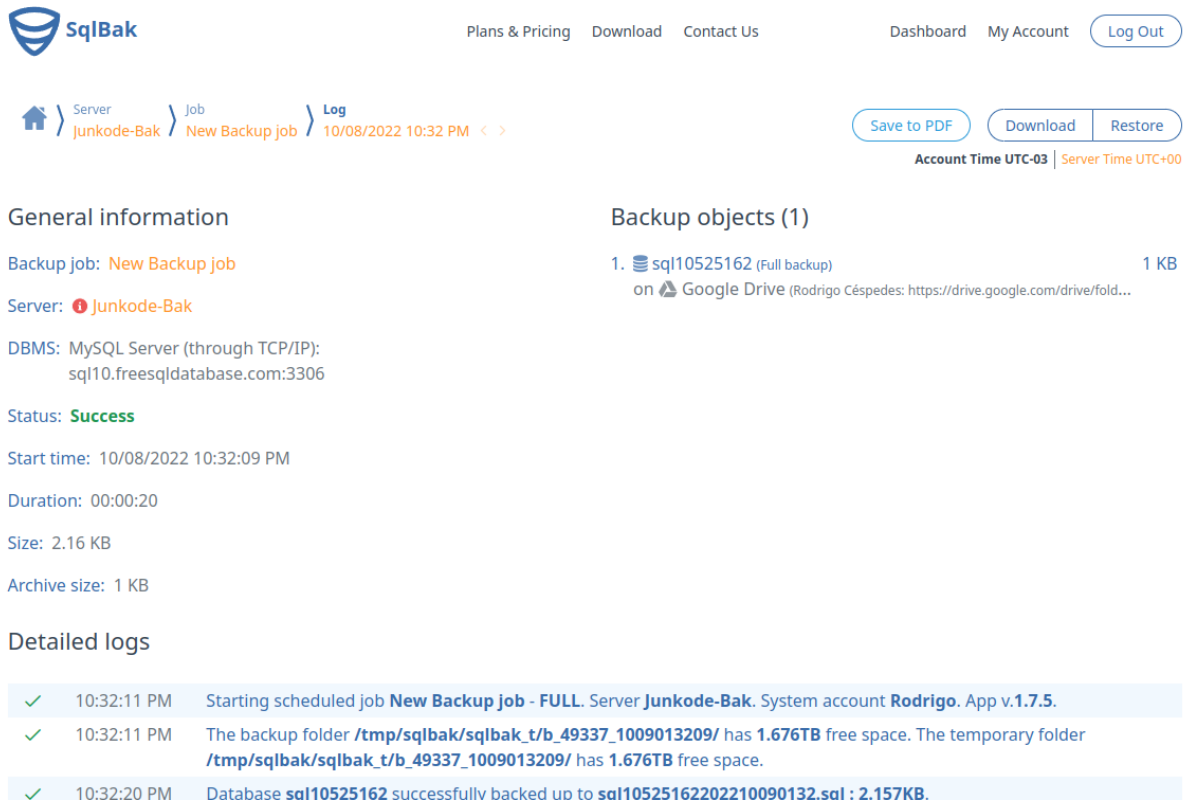


Figura 5 - SqlBak: Gestor de programas de exportación y respaldo de bases de datos.

8. **Elaborar una estrategia de mejora del área seleccionada, que contenga como mínimo 20 actividades a realizar en los próximos 2 años, distribuidas según el momento de ejecución (por ej. con cronograma mensual). La estrategia tiene que estar orientada a mejorar día a día la calidad en la gestión del área, por ej. mejorar el rendimiento del personal, mejorar los resultados, apoyar a los objetivos de la empresa u organización, tener una adecuada relación con otras áreas, eficiencia, generación proactiva, reducción de errores, mejoramiento de relaciones interpersonales, satisfacción continua de los Clientes internos y externos, potenciar fortalezas, aprovechar oportunidades, reducir debilidades y estar preparado para las amenazas, etc.**

   El proceso de mejora incremental planteado para el Área es el de optimizar la eficiencia de las tareas de esta por medio de adoptar metodologías de bases de datos por servicios, escalamiento horizontal y almacenamiento en la nube.

Para este punto se propone la confección de un mapa estratégico que puede encontrarse al final del listado de actividades, bajo el nombre de “Mapa estratégico del Programa de mejora incremental del Área de Gestión de Datos”. Dicho mapa estratégico consta de una duración de 24 meses y sus objetivos (enumerados pero no en un orden de precedencia) son:

1. Programa de mejora incremental del Área de Gestión de Datos:
   - ➢ Mes estimado para que se logre: Mes 24.
   - ➢ Descripción: El programa en sí consiste en una sucesión de decisiones y medidas directivas con el fin de aumentar la eficiencia de las tareas del área de Gestión de Datos de manera incremental a lo largo de 2 años.
   - ➢ Objetivos consecuentes:
     - ○ Probar de distintos motores de DB.
     - ○ Integrar herramientas de migración de Bases de Datos.
     - ○ Implementar nuevas tecnologías de acceso a las Bases de Datos.
     - ○ Brindar capacitación sobre tecnologías Cloud para la gestión de datos.
     - ○ Probar tecnologías de hardware alternativas.
     - ○ Brindar capacitación sobre nuevos medios de Back-Up y recuperación de datos.
     - ○ Desarrollar semilleros de datos parametrizables a partir de datos de producción.

2. Probar de distintos motores de DB:
   - ➢ Mes estimado para que se logre: Entre los meses 2 y 3.
   - ➢ Descripción: Cada vez es más frecuente que las empresas manejan sus servicios en con sus propias bases de datos y cada una con el motor de base de datos que les sea más eficiente. Para saber cuál es el motor más eficiente es necesario probarlos en paralelo. Esto es sencillo cuando se habla de microservicios, pero casi imposible con monolitos, por lo que el tiempo que puede tomar alcanzar este objetivo depende de eso. Esta estimación se hace asumiendo que la empresa desarrolla su software en forma de microservicios.
   - ➢ Objetivos consecuentes:
     - ○ Alcanzar una mayor desvinculación de las Bases de Datos y de los DBA.
     - ○ Reestructurar las Bases de Datos.
     - ○ Reorganizar a los DBA en base a las nuevas Bases de Datos.

3. Integrar herramientas de migración de Bases de Datos:
   - ➢ Mes estimado para que se logre: Mes 1.

- ➢ Descripción: Cuando se encuentra un motor de base de datos mejor para la gestión de datos de un servicio específico, es necesario migrar los datos de la base de datos actual a la nueva. Esto requiere de herramientas de migración genéricas (como spoon) o específicas (como Heidi para migrar de MySQL a MariaDB), al igual que instrucción para los DBAs para realizarla. La finalidad de este objetivo es que el Área de Gestión de Datos cuente con herramientas listas para realizar migraciones de datos cuando sea necesario.
- ➢ Objetivos consecuentes:
    - ○ Renovar los Pipelines de DWH.
    - ○ Reestructurar las Bases de Datos.

4. Implementar nuevas tecnologías de acceso a las Bases de Datos:
    - ➢ Mes estimado para que se logre: Entre los meses 2 y 3.
    - ➢ Descripción: Muchas veces se presentan problemas de acceso a las bases de datos cuando estas se encuentran montadas directamente sobre servidores (si el servidor se cae, las bases de datos de sus discos quedan inaccesibles para los servicios que no se han caído). La solución a esto es implementar servidores NAS o servidores específicos de almacenamiento de datos, para garantizar que estos puedan ser siempre accedidos y así conseguir resiliencia en las bases de datos frente a las fallas de los servicios.
    - ➢ Objetivos consecuentes:
        - ○ Reducir el costo de mantenimiento del Data Center.

5. Brindar capacitación sobre tecnologías Cloud para la gestión de datos:
    - ➢ Mes estimado para que se logre: Mes 2.
    - ➢ Descripción: Actualmente existen muchos servicios de almacenamiento bajo demanda brindados por diversos proveedores en la nube. Es crucial dominarlos debido a que es la solución más resiliente que existe. Este es el motivo por el cual este objetivo pretende que los DBAs aprendan sobre las tecnologías de gestión de datos en la nube (Motores de bases de datos, Buckets, Caches, etc.) para puedan conocer todas los opciones de las que se dispone para realizar gestión de datos, y no solo las on-premise.
    - ➢ Objetivos consecuentes:
        - ○ Reestructurar las Bases de Datos.
        - ○ Reorganizar a los DBA en base a las nuevas Bases de Datos.
        - ○ Contratar servicios en la nube para Buckets.
        - ○ Enfatizar la implementación de un enfoque de escalado horizontal para las Bases de Datos.

6. Probar tecnologías de hardware alternativas:

- ➢ Mes estimado para que se logre: Entre los meses 4 y 5.
- ➢ Descripción: Este objetivo se enfoca en aumentar la eficiencia y resiliencia de las bases de datos pero desde el hardware. Esto puede ir desde implementar distintos tipos de RAIDs (o combinaciones de ellos) hasta adquirir nuevos elementos para el data center (como servidores NAS para las bases de datos). Es importante resaltar que el objetivo sigue siendo la prueba de alternativas posibles para así tener un panorama mucho más amplio a la hora de seleccionar tecnologías de hardware (que son generalmente las más costosas) para la gestión de datos.
- ➢ Objetivos consecuentes:
    - Reducir el costo de mantenimiento del Data Center.

7. Brindar capacitación sobre nuevos medios de Back-Up y recuperación de datos:
   - ➢ Mes estimado para que se logre: Mes 4.
   - ➢ Descripción: Mientras que los motores de bases de datos y el hardware se actualizan de manera frecuente en la mayoría de los datacenters, generalmente no ocurre lo mismo con la tecnología de Back-ups. Este objetivo implica la capacitación sobre medios de Back-up que la empresa pueda usar para respaldar su información, para de esta forma saber cómo realizarlo correctamente. O sea, si debe realizarse automática o manualmente, o si es conveniente almacenarlo en la nube o on-premise.
   - ➢ Objetivos consecuentes:
       - Automatizar las tareas de respaldo, restauración y recuperación de Bases de Datos.

8. Desarrollar semilleros de datos parametrizables a partir de datos de producción:
   - ➢ Mes estimado para que se logre: Entre los meses 3 y 4.
   - ➢ Descripción: Este objetivo implica la creación de una solución de software automatizada para la obtención de datos para que los desarrolladores de otras áreas puedan utilizar en sus entornos de pruebas. Estos datos deben ser aleatorios, parametrizables y no contener ninguna información sensible (en caso de que los campos requeridos para la prueba sean sensibles, deberán generarse también aleatoriamente sobre los demás datos de producción).
   - ➢ Objetivos consecuentes:
       - Lograr una mayor personalización de los datos para entornos de prueba para el Área de Desarrollo y Mantenimiento.

9. Renovar los Pipelines de DWH:
   - ➢ Mes estimado para que se logre: Entre los meses 6 y 8.

- ➢ Descripción: Si bien este objetivo no se asimila a los demás, si resulta altamente relevante para aumentar la eficiencia del área. Debido a que cambiarán las bases de datos y que se implementarán herramientas para hacer migraciones entre estas, es un buen momento para rediseñar los pipelines que convierten los datos en vistas minables para los niveles gerenciales. Para esto es necesario que los DBAs de las bases de datos originales (haciendo uso de las consultas requeridas por los niveles estratégicos de la empresa) replanteen la manera en la que los datos serán extraídos y limpiados una vez que las bases de datos se hayan especializado, y luego de que replanteen estos trabajos, los desarrollen e implementen.
- ➢ Objetivos consecuentes:
    - ○ Aumentar la eficiencia de la Gestión de Datos de la empresa.

10. <u>Reorganizar a los DBA en base a las nuevas Bases de Datos:</u>
    - ➢ Mes estimado para que se logre: Mes 7.
    - ➢ Descripción: Al ahora implementar las bases de datos en distintos motores, deberá a cada una asignársele un DBA para que sea responsable de ella. Solo se necesita de uno, puesto que ahora las bases de datos serán más específicas y menos complejas, y de esta forma los DBAs podrán especializarse en motores de bases de datos específicos, lo que permitirá un uso más apropiado de las capacidades de cada motor.
    - ➢ Objetivos consecuentes:
        - ○ Profundizar la comunicación entre los Diseñadores de bases de datos y el Área de Desarrollo y Mantenimiento.

11. <u>Alcanzar una mayor desvinculación de las Bases de Datos y de los DBA:</u>
    - ➢ Mes estimado para que se logre: Mes 7.
    - ➢ Descripción: La idea de que toda la información sea almacenada en una sola base de datos es poco práctica en proyectos grandes debido a que impide un escalado sencillo de los sistemas. La idea aquí es desvincular las bases de Datos demasiado grandes en otras más específicas. Como esto implica que cada base de datos ahora podrá correr sobre el motor específico que más se adapte a la misma, también implica que los DBAs ya no deberán trabajar todos sobre bases de datos comunes, sino que a cada uno le corresponderán algunas en específico en base a las tecnologías que estas usen. Esto no solo le da más independencia a las Bases de Datos, sino que también a los DBAs.
    - ➢ Objetivos consecuentes:
        - ○ Profundizar la comunicación entre los Diseñadores de bases de datos y el Área de Desarrollo y Mantenimiento.

12. Reestructurar las Bases de Datos:
    - ➢ Mes estimado para que se logre: Entre los meses 7 y 8.
    - ➢ Descripción: Cuando ya se hayan definido las bases de datos específicas y se sepan las tecnologías óptimas para la implementación de cada una, se deben reestructurar las bases de datos en las nuevas. O sea, se deben crear y migrar los datos de una base a la otra. Para que este objetivo se logre es indispensable que se cuente con el conocimiento y las herramientas a las que hacían referencia los objetivos anteriores (acceso a bases de datos, migración de bases de datos y servicios de almacenamiento en la nube).
    - ➢ Objetivos consecuentes:
        - ○ Profundizar la comunicación entre los Diseñadores de bases de datos y el Área de Desarrollo y Mantenimiento.

13. Contratar servicios en la nube para Buckets:
    - ➢ Mes estimado para que se logre: Mes 8.
    - ➢ Descripción: Los archivos binarios son con frecuencia almacenados junto con la metadata que los representa, en las mismas bases de datos. Esto trae gigantescos problemas para el escalado, por lo que se ha resuelto solo almacenar la metadata en las bases de datos y subir los archivos binarios a la nube, puesto que estos son los que con mayor celeridad crecen en volumen y por tanto, los que mayor necesidad de escalado bajo demanda rápido tienen.
    - ➢ Objetivos consecuentes:
        - ○ Reducir el costo de escalado de las bases de datos.

14. Enfatizar la implementación de un enfoque de escalado horizontal para las Bases de Datos:
    - ➢ Mes estimado para que se logre: Entre los meses 8 y 12.
    - ➢ Descripción: Este es uno de los objetivos que más relevancia van a tener sobre las disminuciones de gastos más adelante. Implica promover la creación de nuevas instancias de servicios para gestión de datos bajo demanda como forma de enfrentar un aumento drástico o sostenido de consultas. Consiste específicamente en hacer correr más instancias de las bases de datos (conservando aún su integridad) para satisfacer estas consultas, en lugar de simplemente potenciar el hardware que las soporta.
    - ➢ Objetivos consecuentes:
        - ○ Reducir el costo de escalado de las bases de datos.

15. Automatizar las tareas de respaldo, restauración y recuperación de Bases de Datos:
    - ➢ Mes estimado para que se logre: Entre los meses 12 y 13.

- ➢ Descripción: La especialización de las bases de datos por servicios implica un aumento de la cantidad de bases de datos y también un aumento en su heterogeneidad. Esto implica que si los procesos de backup, restauración y recuperación de datos siguen realizándose de manera manual, se perderá mucho más tiempo. Este objetivo implica el desarrollo de scripts genéricos que automaticen estas tareas para cada motor de bases de datos que la empresa emplee. Deben ser genéricos para que luego puedan automatizarse estos procesos también en las nuevas bases de datos que el área cree.
- ➢ Objetivos consecuentes:
    - Reducir el costo de mantenimiento del Data Center.

16. Profundizar la comunicación entre los Diseñadores de bases de datos y el Área de Desarrollo y Mantenimiento:
    - ➢ Mes estimado para que se logre: Entre los meses 12 y 15.
    - ➢ Descripción: Los clientes del Área de Gestión de Datos, son otros empleados de la empresa. Los más estrechamente relacionados al Área son los de Desarrollo y Mantenimiento. Es imperativo que esta relación se refuerce para llevar a cabo la especialización de bases de datos propuesta, puesto que esta especialización se hará en base a los servicios que esta Área desarrolle. Por lo que cada Diseñador de Base de Datos deberá ser capacitado sobre el negocio que circunda a su base de datos por los miembros del Área de Desarrollo y Mantenimiento que realizaron el análisis del sistema.
    - ➢ Objetivos consecuentes:

17. Lograr una mayor personalización de los datos para entornos de prueba para el Área de Desarrollo y Mantenimiento:
    - ➢ Mes estimado para que se logre: Entre los meses 12 y 13..
    - ➢ Descripción: Los semilleros generados con anterioridad muestran su utilidad en este punto. La generación de conjuntos de datos parametrizables permite que los equipos de testing trabajen con valores verosímiles pero libres de información sensible y, mucho más importante, los mantiene alejados de los datos de las bases de datos en producción.
    - ➢ Objetivos consecuentes:
        - Aumentar la eficiencia de la Gestión de Datos de la empresa.

18. Reducir el costo de mantenimiento del Data Center:
    - ➢ Mes estimado para que se logre: Entre los meses 16 y 20.
    - ➢ Descripción: Este objetivo se cumple indirectamente si se cumplen los anteriores, o sea, si se despliegan bases de datos y buckets en la nube, se facilita el acceso físico a los datos separando los servidores con las RAIDs de los servidores con

servicios, se automatizan los procesos de backup y restauración, se generan pipelines para la la migración de las bases de datos y se suprime la necesidad de acceso de los empleados de otras áreas a los equipos de gestión de datos. Todo esto conlleva consigo un ahorro económico a la hora de mantener el data center.

- ➢ Objetivos consecuentes:
    - Aumentar la eficiencia de la Gestión de Datos de la empresa.

19. Reducir el costo de escalado de las bases de datos:
    - ➢ Mes estimado para que se logre: Entre los meses 16 y 22.
    - ➢ Descripción: Al tener bases de datos separadas por servicios, es mucho más sencillo escalar, puesto que solo se deben incrementar los recursos de una base de datos en específico, o sea que solo se debe pagar por una en específico. Por otro lado, el escalado horizontal permite ahorrar mucho en hardware y también permite que los servicios de acceso, consulta, modificación y replicación de bases de datos se realice de una manera mucho más rápida y sin perjuicios para el usuario de las bases de datos.
    - ➢ Objetivos consecuentes:
        - Aumentar la eficiencia de la Gestión de Datos de la empresa.

20. Aumentar la eficiencia de la Gestión de Datos de la empresa:
    - ➢ Mes estimado para que se logre: Entre los meses 16 y 24.
    - ➢ Descripción: El objetivo final de este plan estratégico es aumentar la eficiencia de la Gestión de Datos. A lo largo de los puntos anteriores pudo apreciarse cómo se aprovechaban recursos y reducían costos, de manera que en este punto, el Área de Gestión de Datos de la empresa cuenta con trabajadores capacitados en todos los ámbitos del espectro de la gestión de datos y con bases de datos especializadas que escalan rápidamente bajo demanda sin requerir recursos innecesarios, o sea, sin incurrir en costes innecesarios. Esto último es en sí la definición de eficiencia.

Anexo n°1: Mapa Estratégco del Programa de Mejora Incremental del Área de Gestión de Datos