Universidad Tecnológica Nacional

Facultad Regional Mendoza

Ingeniería en Sistemas

Proyecto Final

Sistema de Generación de Código en Base a Metamodelos

Metacod

Cuerpo docente
Vazquez, Alejandro
Moralejo, Raul Omar
Manino, Gustavo

Grupo 12	
Alumno	Legajo
Vasquez, Alesis	31.362
Cruz, Leandro	34.772
de Sautu Riestra, Agustín	43.628
Perez, Angel Santiago	40.023
Manfredi, Gustavo	40.389

Año 2021

RESUMEN TÉCNICO

Breve Objetivo Del Proyecto

El proyecto busca ser un híbrido entre un low-code y un generador de código. Donde se facilita al usuario la creación de una aplicación web moderna mediante el diseño de un diagrama de dominio con un UML con restricciones del paradigma de orientación a objetos.

La aplicación busca ayudar al usuario en la tarea de la creación de una aplicación web, facilitando una solución inicial para el Frontend y para el Backend según los lenguajes, frameworks, librerías seleccionadas y el Modelo de dominio hecho en nuestro modelador de diagramas con una arquitectura limpia y buenas prácticas.

Breve justificación del trabajo realizado

El proyecto nace de la complejidad del desarrollo de las aplicaciones web modernas de hoy en día. Para un desarrollo de una aplicación web moderna es necesario tener conocimiento de muchas tecnologías diferentes como:

FRONTEND		BACKEND	
	SEO		C#
	Javascript		Java
	HTML5		Docker
	CSS3		Express JS
	React		Python
	Angular		Node js
	Vuejs		Swagger

Figura 1. Tecnologías para el desarrollo de una aplicación.

Aquí no se tienen en cuenta los controladores de versiones, bases de datos, y herramientas de DevOps.

Palabras claves

Sistema Generador de Codigo
Generador de Plataformas Web
Modelador de UML
Generador de ABMs
Generador de APIs REST
Lenguaje Orientados a Objetos

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Trabajo Práctico Anual Nº1

DEFINICIÓN DE REQUERIMIENTOS

Investigación preliminar y relevamiento general

1. Antecedentes y Sistemas similares

1.1. CUBA

1.1.1. Relevamiento general

1.1.1.1. Sistema Relevado

La plataforma CUBA es un marco de código abierto destinado a agilizar el proceso de desarrollo de aplicaciones comerciales. Combina una arquitectura probada, componentes de nivel empresarial listos para usar y herramientas productivas, para que pueda entregar aplicaciones web modernas más rápido que nunca. (CUBA. Studio User Guide)

https://www.cuba-platform.com/documentation/

1.1.1.2. Funciones detectadas e interfaces

A continuación, se presenta una lista de las funcionalidades observadas en este sistema y se indica con qué usuario interactúa cada funcionalidad.

- **Gestión de proyecto:** Permite la creación de un nuevo proyecto y establece sus configuraciones iniciales. También permite la creación de atributos y entidades.
- **Creación y eliminación de pantallas:** Es donde se administra toda la configuración de las pantallas CRUDs de cada entidad generada
- **Creación de base de datos:** Genera scripts de base de datos para crear SQL para la creación de la base de datos.
- **Utilización de Screen API:** Permite toda la intervención con la pantalla como el manejo de eventos de ratón y teclado.
- **Agregar marca:** Establece la configuración para agregar la marca a todas las interfaces.

Interfaces:

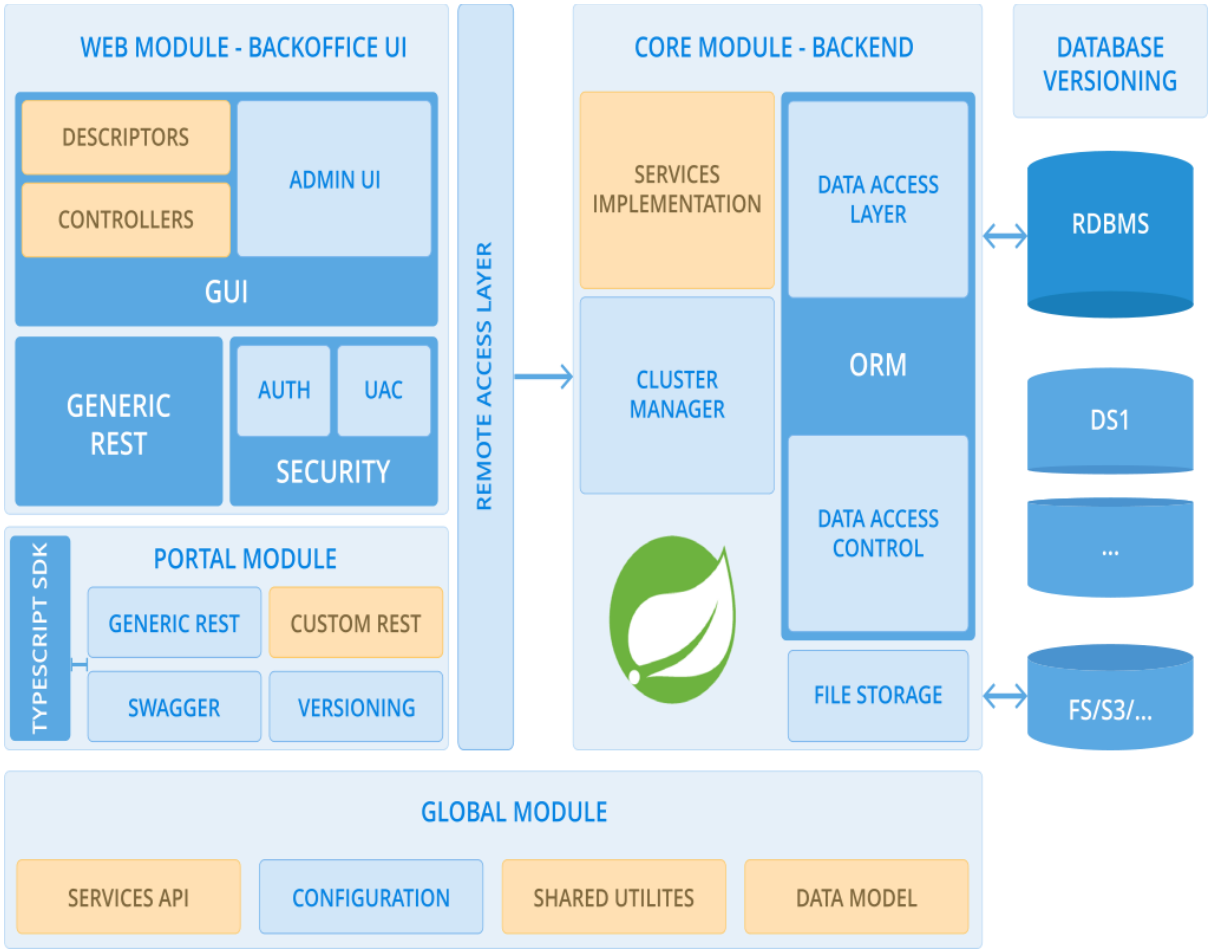


Figura 2. Interface de CUBA platform.

1.1.1.3. Tecnología de Información.

MODULAR
La arquitectura única permite que cualquier aplicación CUBA se incruste en otra aplicación CUBA. Esto hace que la modularización sea prácticamente trivial: divida su tarea en varias partes sueltas, desarrolle por separado y ensamble en un sistema completo.

ESCALABLE
El marco está diseñado para ser escalable tanto vertical como horizontalmente. Proporciona múltiples opciones de implementación según la carga de aplicación planificada y el tiempo de inactividad permitido.

COMPATIBLE
Las aplicaciones de CUBA son compatibles con los RDBMS más populares y se ejecutan en cualquier contenedor de servlets de Java. Pueden distribuirse como WAR, imagen de Docker, Jar o implementarse en las nubes.

1.1.2. Relevamiento detallado y análisis del sistema

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

Funcionalidad: Gestión de proyecto

Pasos

- En la ventana **Bienvenido a CUBA Studio**, haga clic en **Crear nuevo proyecto**, o use Menú principal → **Archivo** → **Nuevo** → **Proyecto**.
- Seleccione **Proyecto CUBA**.
- Especificar el **espacio de nombres del proyecto**: el espacio de nombres que se utiliza como prefijo para los nombres de las entidades y las tablas de la base de datos. El espacio de nombres puede constar únicamente de letras latinas y debe ser lo más corto posible. **Piense detenidamente en el nombre en esta etapa**, ya que cambiarlo más adelante requerirá una compleja intervención manual.
- Cambie el **paquete raíz** si es necesario. Es el paquete raíz (o base) para las clases de Java. Se puede ajustar más tarde, pero las clases generadas en la creación del proyecto no se moverán.
- Cambie el **prefijo del módulo** si es necesario. Los nombres de los módulos del proyecto CUBA van precedidos de este prefijo. El prefijo del módulo se puede ajustar más tarde.
- En el campo **SDK** del **proyecto**, seleccione un JDK correspondiente al JAVA_HOME que está configurado en su entorno. Si ve el <No SDK>valor, haga clic en **Nuevo** y seleccione la carpeta donde está instalado el JDK, por ejemplo, C:\Java\jdk8u202-b08en Windows o /Library/Java/JavaVirtualMachines/jdk8u202-b08/Contents/Home en macOS.
- Acepte la configuración de repositorios predeterminada o personalícela para el proyecto.
- Seleccione la **versión de la plataforma** que se utiliza en el proyecto. Si no tiene requisitos específicos, utilice la última versión publicada.
- Para realizar pruebas beta o acceder anticipadamente a las nuevas funciones de la plataforma CUBA, es posible que desee utilizar una de las versiones inestables de la plataforma CUBA, las que terminan con los sufijos BETA o SNAPSHOT. Para verlos en la lista desplegable **Versión de** la **plataforma**, debe seleccionar la casilla de verificación **Mostrar versiones inestables**. También tenga en cuenta que las versiones SNAPSHOT se publican solo en el repo.cuba-platform.comrepositorio de artefactos.
- Seleccione un valor en el campo desplegable **Soporte de idiomas** para cambiar el conjunto de lenguajes de programación utilizados en el proyecto, o deje el valor de **Java** recomendado.
- Usar el campo de las opciones **locales disponibles** para abrir la **Configuración regional** de diálogo del editor y añadir lugares más compatibles con el proyecto. Podrá cambiarlos más tarde si es necesario.
- Haga clic en **Siguiente**.
- En el segundo paso del asistente, puede configurar las propiedades del almacén de datos principal, por ejemplo, seleccionar la base de datos PostgreSQL local para el nuevo proyecto. Estas propiedades se pueden cambiar más tarde.
- Haga clic en **Siguiente**.
- Cambie el valor del campo **Nombre** del **proyecto**, si es necesario. El nombre debe contener solo letras latinas, números y guiones bajos.

- **La ubicación del proyecto** es la ruta al directorio del nuevo proyecto. Puede seleccionar otro directorio escribiéndose en el campo o haciendo clic en el botón de puntos suspensivos junto al campo.
- Haga clic en **Finalizar**. El proyecto vacío se crea en el directorio especificado y Studio comenzará a generar la información del proyecto a partir de archivos Gradle y a indexar el proyecto.
- Cuando se complete el proceso de sincronización e indexación del proyecto, verá el árbol del proyecto CUBA en la ventana de la herramienta **Proyecto**.
- Abra la ventana de la herramienta **Gradle** que está acoplada en el lado derecho de forma predeterminada. Haga clic en el icono de "llave inglesa" (**Configuración de Gradle**) y seleccione Project SDK en el campo **Gradle JVM**. Haga clic en **Aceptar**.
- Ahora puede empezar a trabajar con el proyecto.

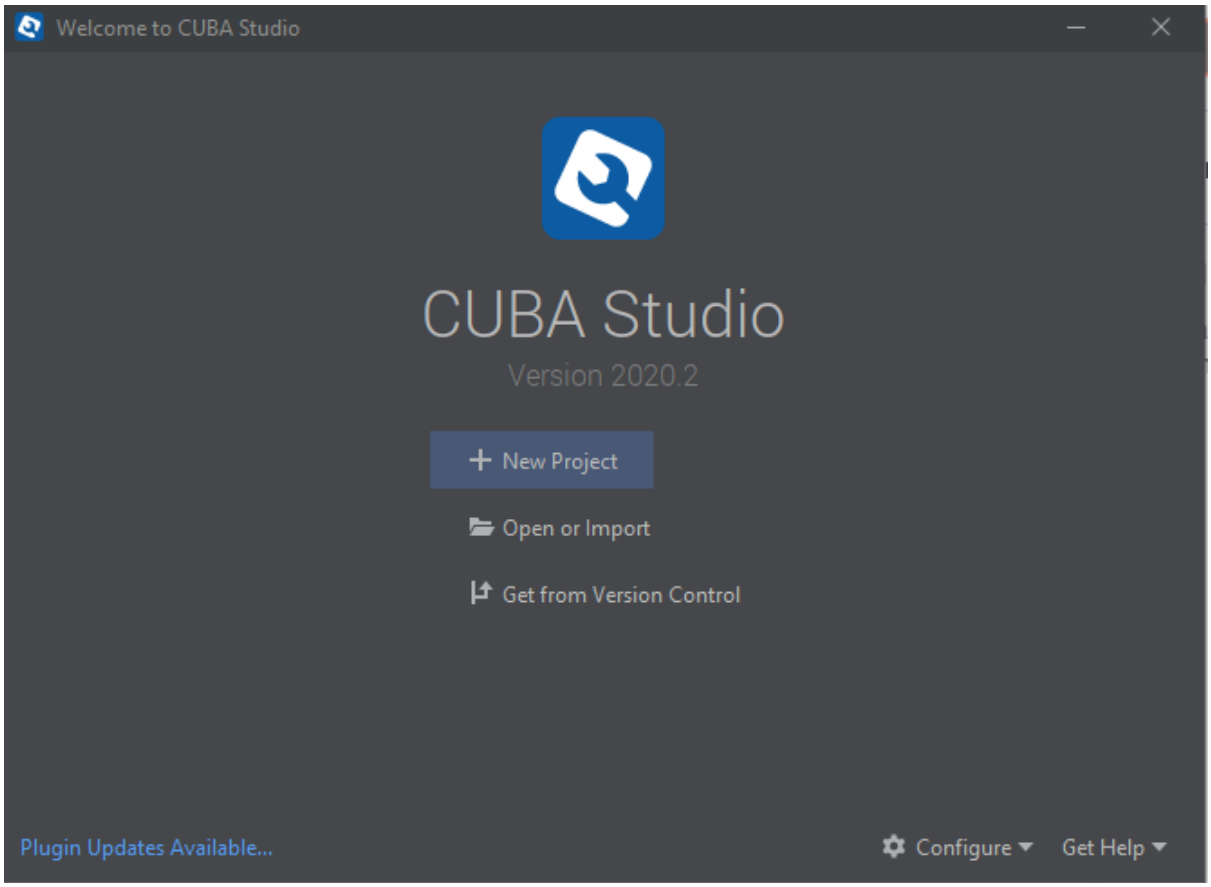


Figura 3. Funcionalidad: Gestión de un proyecto CUBA.

Funcionalidad: Creando nueva entidad

Pasos

- Seleccione la sección **Modelo de datos** o un paquete debajo en el árbol del proyecto y elija **Nuevo> Entidad** en el menú contextual.
- La **Nueva Entidad** aparece en el diálogo. Introduzca el nombre de la clase de entidad en el campo **Nombre de la entidad**, seleccione el tipo de la entidad y su identificación.
- Studio crea la clase de entidad y la registra en persistence.xml o metadata.xml, según el tipo de entidad. La clase creada se abrirá en el editor de código fuente.

- Studio muestra cuatro pestañas en la parte inferior del editor de código fuente de las entidades. Juntos forman el diseñador visual de entidades:

  El texto contiene el código fuente.
  Designer muestra la estructura de la entidad donde puede configurar la entidad y sus atributos utilizando una interfaz gráfica en lugar de escribir código Java.
  Índices muestra y le permite crear nuevos índices para la entidad seleccionada.
  La vista previa de DDL muestra el código DDL de solo lectura de la tabla correspondiente y sus restricciones de referencia.

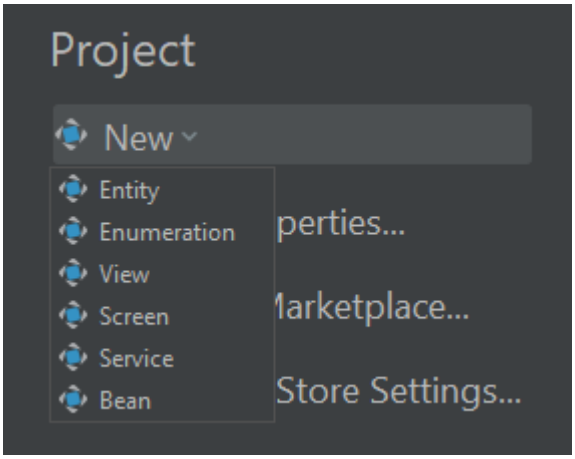


Figura 4. Opciones de creación CUBA.

Creación de una entidad donde permite seleccionar si la entidad persiste y el tipo de dato que se va a almacenar como id.

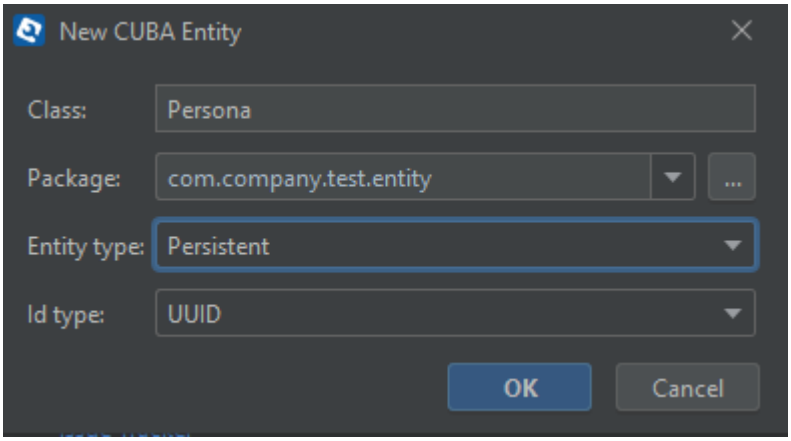


Figura 5. Creación de una entidad CUBA.

Código generado en java:

```java
package com.company.test.entity;

import ...

@Table(name = "TEST_PERSONA")
@Entity(name = "test_Persona")
public class Persona extends StandardEntity {
    private static final long serialVersionUID = -8985452843761194354L;
}
```

Figura 6. Código generado de una entidad CUBA.

Funcionalidad: Crear nuevo atributo

Hay varias formas de agregar un atributo a una entidad.

- Usando la interfaz gráfica del diseñador de entidades: cambie a la pestaña **Diseñador**, haga clic en **Nuevo** debajo de la tabla **Atributos** y complete los campos obligatorios en la ventana **Nuevo atributo**.
- El botón de globo en la parte derecha del campo **Nombre** le permite establecer inmediatamente un nombre fácil de usar para el atributo. Se almacena en el messages.propertiesarchivo y se utiliza de forma predeterminada en los componentes de la interfaz de usuario. Si ha definido varios idiomas para su aplicación, puede especificar el nombre localizado para todos los idiomas.

Figura 7. Opciones de un atributo CUBA

Funcionalidad: Creación y eliminación de pantallas

Para crear una nueva pantalla de IU genérica, seleccione **IU genérica** en el árbol del proyecto y haga clic en **Nuevo> Pantalla** en el menú contextual. Si desea crear una pantalla CRUD para una entidad, seleccione esta entidad en el árbol del proyecto y haga clic en **Nuevo> Pantalla** en el menú contextual:

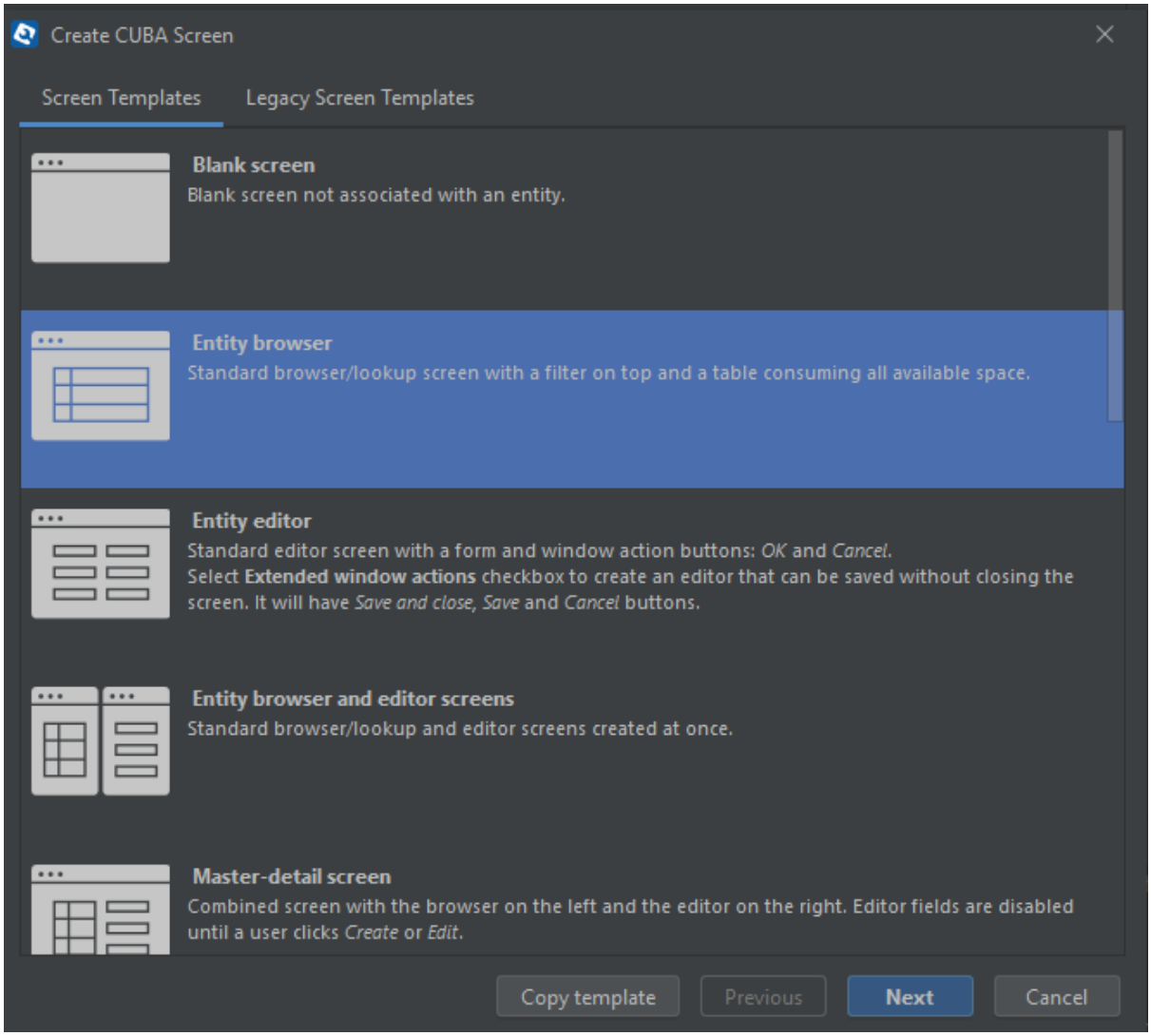


Figura 8. Opciones de pantallas para crear CUBA.

Si está creando la pantalla para una entidad seleccionada en el árbol del proyecto, el campo **Entidad** se completa automáticamente:

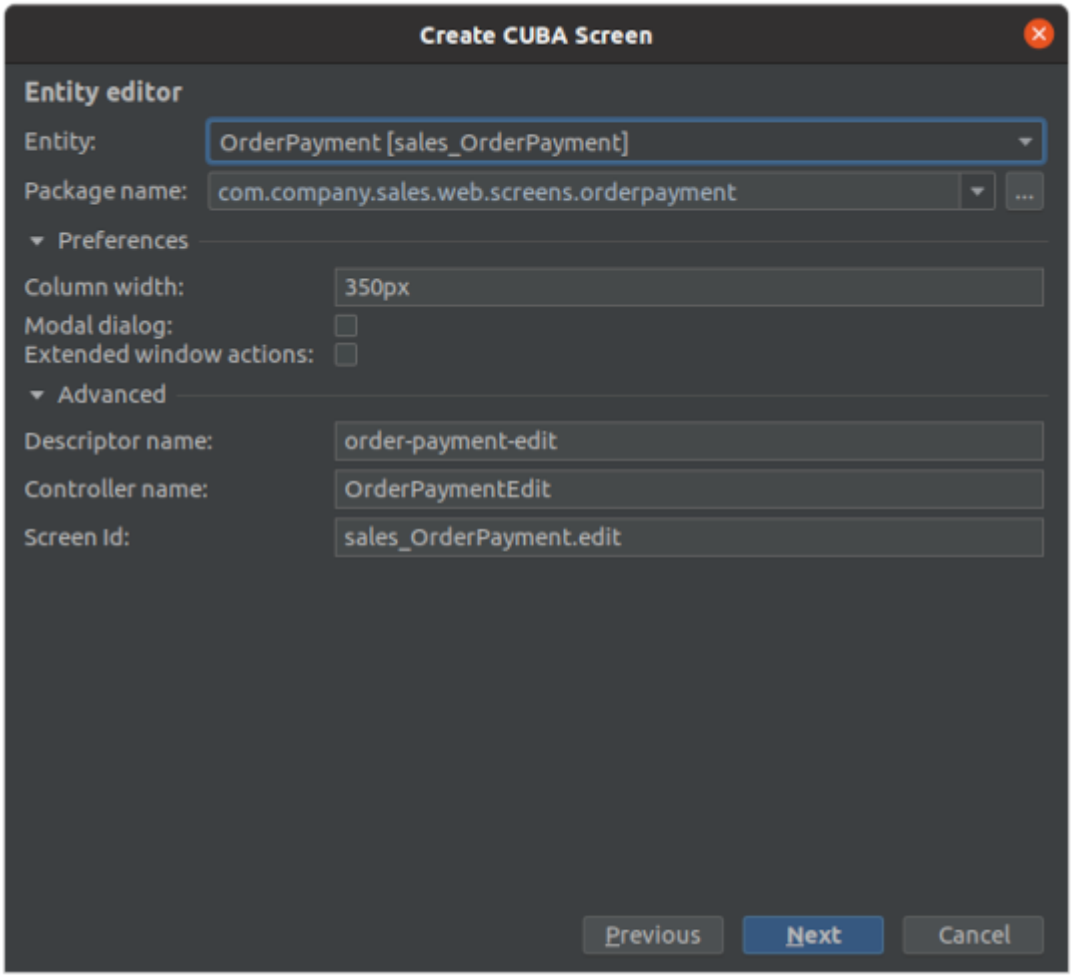


Figura 9. Editor de una pantalla CUBA.

La sección Avanzado le permite modificar el descriptor generado automáticamente y los nombres del controlador y la identificación de la pantalla. Es útil cuando tiene más de una pantalla para alguna entidad.

Los pasos del asistente Vista del navegador de entidades y Vista del editor de entidades le permiten seleccionar la cantidad de datos que muestra en la tabla del navegador o se edita en el formulario del editor de entidades. Las propiedades seleccionadas también determinan la vista de entidad utilizada para cargar entidades en la pantalla:

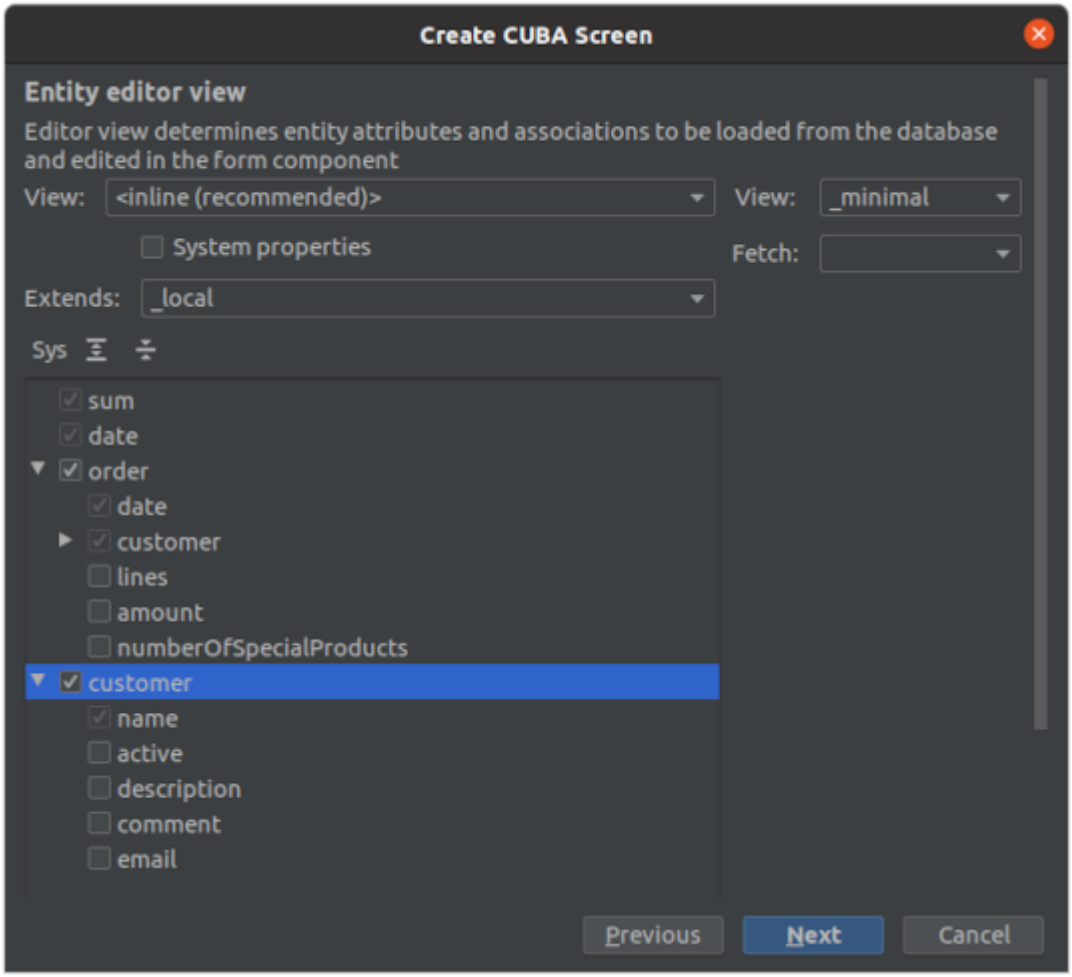


Figura 10. Editor de una pantalla CUBA.

Funcionalidad: Eliminar pantallas

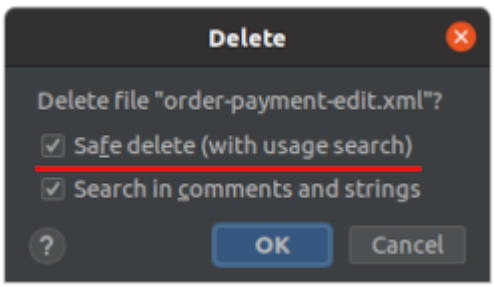


Figura 11. Cuadro de confirmación de eliminación de una pantalla CUBA.

Funcionalidad: Creación de base de datos

Seleccione el menú CUBA -> Generar scripts de base de datos para crear SQL para la creación de la base de datos. Puede revisar los scripts generados en la ventana emergente antes de guardarlos como archivos de proyecto. Tenga en cuenta que esos scripts son parte del proyecto, puede encontrarlos en el nodo Main Data Store en el árbol del proyecto.

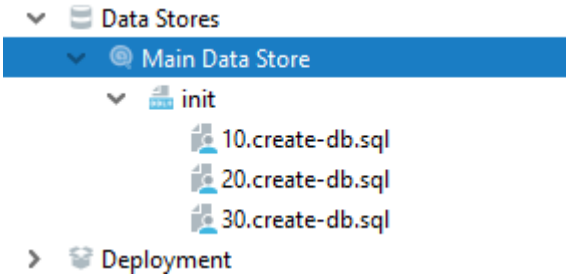


Figura 12. Estructura de las bases de datos CUBA.

Haga clic en el botón Crear base de datos para aplicar esos scripts y crear la base de datos. Además de las tablas de aplicaciones, CUBA crea tablas del sistema donde almacenamos información sobre usuarios, roles, tareas, etc.

Ejecutando la aplicación en modo de desarrollo
Seleccione CUBA -> Iniciar servidor de aplicaciones en el menú principal.

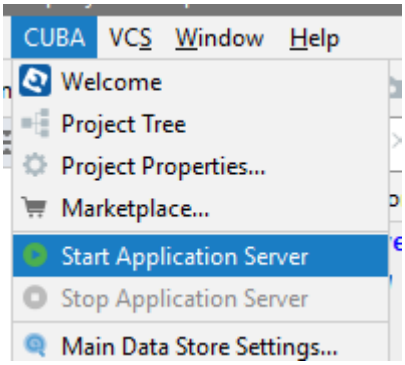


Figura 13. Opción de ejecutar la aplicación en modo desarrollo CUBA.

Después de un tiempo, puede acceder a la aplicación utilizando el navegador. La URL se muestra en la caja de herramientas Ejecutar en IDEA. En nuestro caso será http: // localhost: 8080 /app. Abra la URL en su navegador web e inicie sesión en la aplicación usando "admin" como nombre de usuario. La contraseña es "admin" de forma predeterminada. Puede encontrar pantallas para la manipulación de entidades en el menú Aplicación.

Luego agreguemos algunos datos a la base de datos: un par de oradores y dos sesiones para ellos programadas para el resto de la semana. Puede intentar ingresar un correo electrónico no válido para un orador y ver que el validador de correo electrónico funciona como se esperaba.

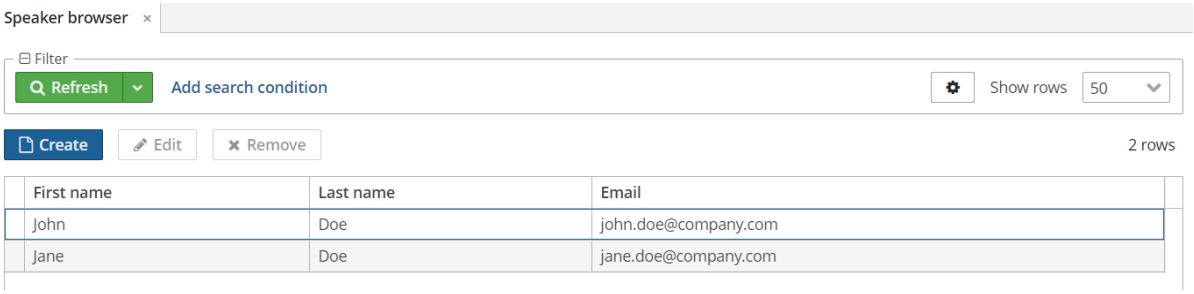


Figura 14. Datos de la base de datos CUBA.

Funcionalidad: Utilización de Screen API

En la session-browse.xmlselección sessionsCalendary vaya a la pestaña Controladores en la ventana Inspector de componentes. Seleccione CalendarEventClickEventy haga clic en la flecha para pasar al controlador.

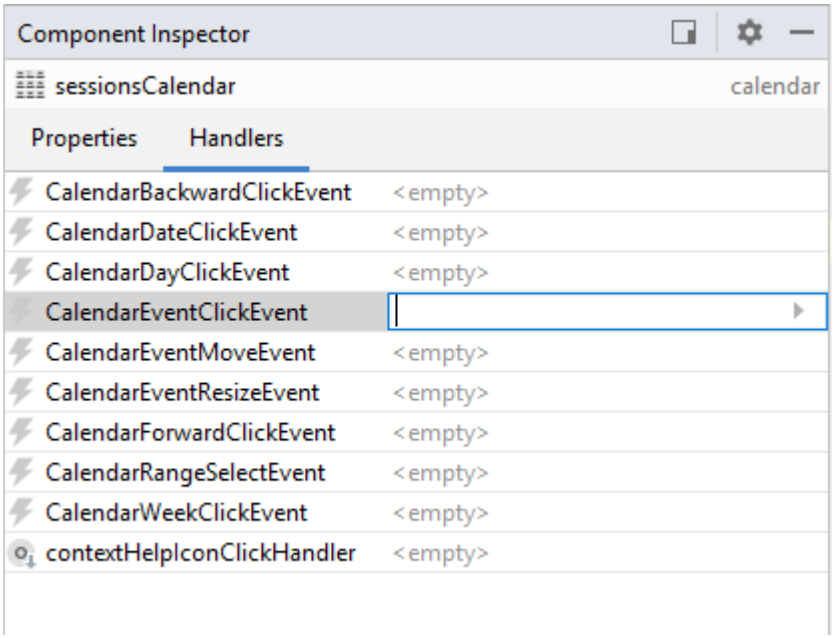


Figura 15. Controladores de eventos Screen API CUBA.

El método vacío se genera automáticamente.

```
@Subscribe("sessionsCalendar")
public void
onSessionsCalendarCalendarEventClick(Calendar.CalendarEventClickEvent<LocalDateTime> event) {
}
```

Funcionalidad: Adición de lógica empresarial

Haga clic con el botón derecho en el nodo de servicio en el árbol del proyecto de CUBA y seleccione Nuevo -> Servicio. Esto abrirá un cuadro de diálogo de creación de servicios.

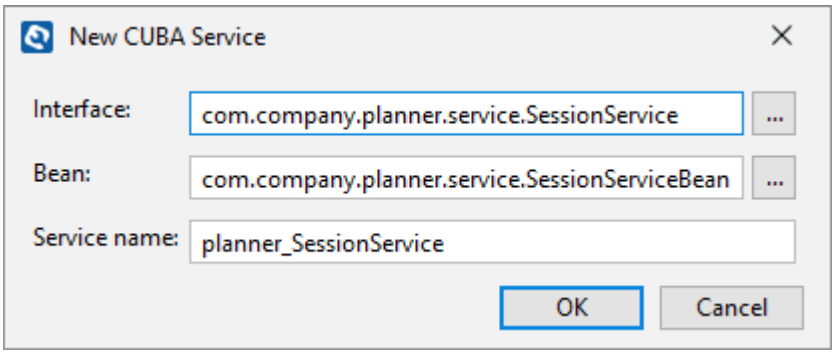


Figura 16. Creación de un nuevo servicio CUBA.

```
public interface SessionService {
  String NAME = "planner_SessionService";

  Session rescheduleSession(Session session, LocalDateTime newStartDate);
}
```

Funcionalidad: Agregar marca

Abra el archivo del paquete de mensajes principal, messages.properties que se encuentra en la interfaz de usuario genérica, el nodo Paquete de mensajes principal en el árbol del proyecto de CUBA.

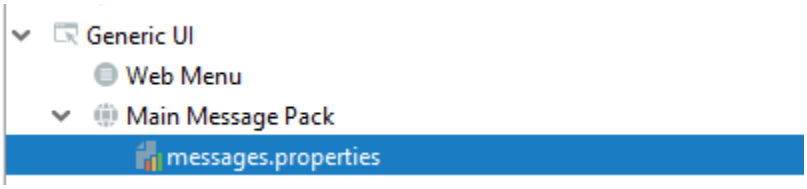


Figura 17. Localización del message.properties CUBA.

1.1.2.2. Modelo Lógico

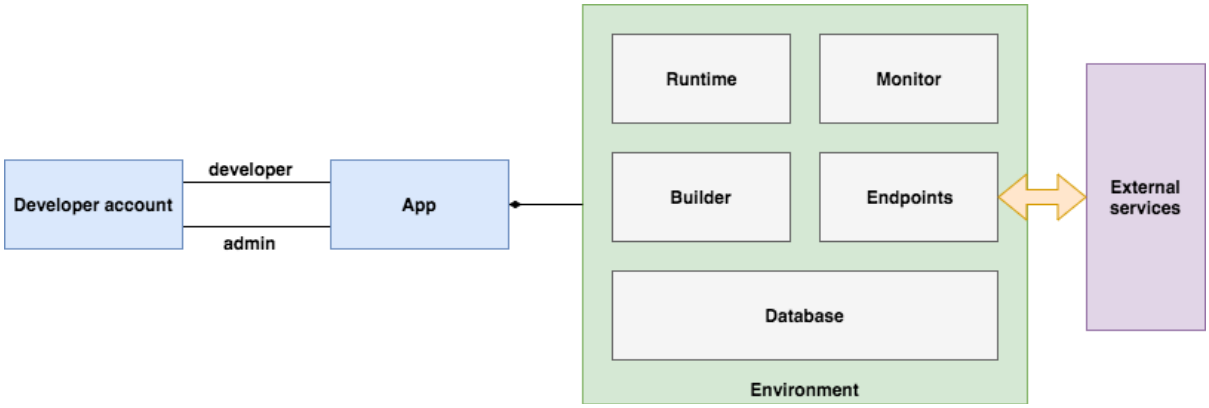


Figura 18. Modules (Modelo Lógico) CUBA.

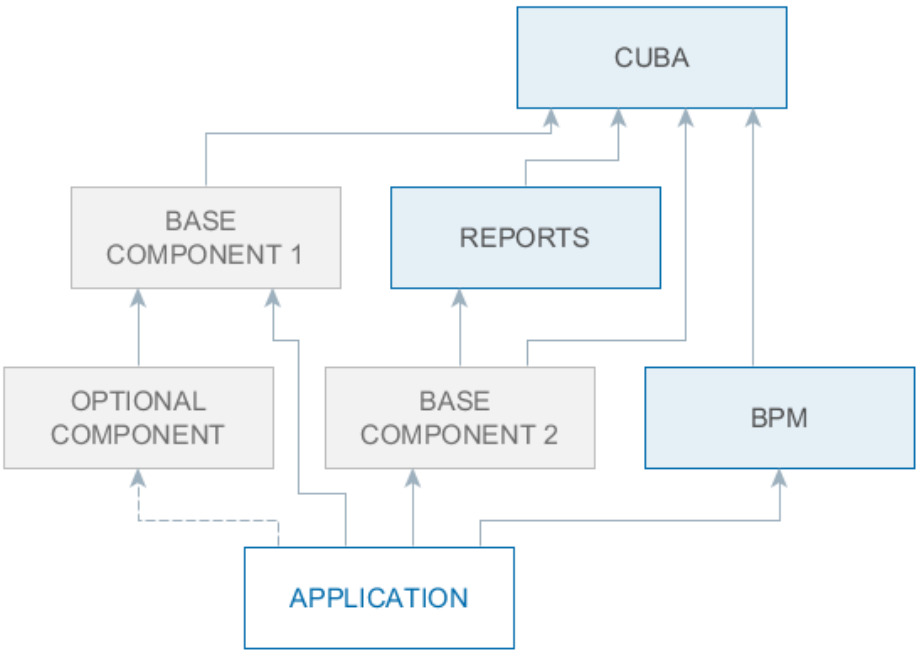


Figura 19. Developers Tools CUBA.

1.1.2.3. ***Problemas y necesidades detectados***

FUNCIONALIDAD	PROBLEMA/NECESIDAD
Generación de código	1- Solo genera código en JAVA, y no permite generar software en otros lenguajes, tecnologías y arquitecturas. 2- No es compatible con Kotlin. El código de Kotlin le permitiría utilizar declaraciones más breves y concisas, para que pueda crear aplicaciones incluso más rápido que antes
Adición de la lógica empresarial	No se encuentran problemáticas con esta función.
Agregar marca	Es una funcionalidad sencilla en la que no se encontraron problemas en versiones recientes
Acceso a la herramienta	Solo versión de escritorio, lo que no posibilita la gestión web y colaboración entre distintos usuarios.
Gestión de proyecto	La plataforma CUBA dice que tiene un IDE, que puede simplificar la creación y gestión de proyectos. Pero a veces cambiar entre Studio e IDE puede ser un poco molesto. La plataforma CUBA se está volviendo a desarrollar, por lo que Studio pronto se convertirá en un complemento para IDEA.
Utilización de Screen API	Es una funcionalidad sencilla en la que no se encontraron problemas en versiones recientes
Gestion de Usuarios	No se puede definir reglas de acceso en código Java, por lo tanto, se necesita exportar definiciones de roles desde su entorno de prueba e importarlas a la versión de producción. Podríamos definir las reglas de acceso a regla para entidades, atributos y pantallas

Creación de Base de datos	1- Al utilizar una base de datos relacional, el uso de CUBA logra la mejor eficiencia de desarrollo. Y si quiere usar NoSQL, CUBA se comporta como Spring, y hay muchos lugares donde necesita escribir su propio código. 2- En un proyecto complejo, si el número de scripts de creación de base de datos supera los 100, la secuencia de ejecución del script puede ser incorrecta
Generación de pantallas crud	No se encuentran funcionalidades o problemáticas con esta función.
Creación del Modelo de datos	En CUBA, usamos más archivos de configuración XML que las aplicaciones típicas de Spring Boot, porque la plataforma proporciona más servicios y se necesita algo de XML como conexión entre varios módulos.

Tabla 1. Problemas y necesidades detectados CUBA

1.2. OPENXAVA

1.2.1. Relevamiento general

1.2.1.1. Sistema Relevado

Es una Plataforma Low-Code de código abierto para desarrollo rápido de aplicaciones empresariales. Escribe las clases del dominio con Java. Obtén una aplicación web lista para producción.
Alta productividad
Solamente escribes la lógica de negocio y la estructura de los datos. No escribes HTML, JavaScript, CSS, SQL, etc. La interfaz de usuario y la lógica de base de datos se proveen automáticamente. (OPENXAVA. Documentation, Reference Guide)

https://www.openxava.org/doc/

1.2.1.2. Funciones detectadas e Interfaces

- **Creación del proyecto:** Permite la creación de un proyecto, configuración de nombre, idioma entre otros.
- **Modelar con Java:** Permite la creación de una clase de java (nombres, modificadores, paquete etc.)
- **Pruebas automáticas:** Utiliza Junit como herramienta de testing, permite crear modelos extendiéndose de las clases y ejecutar las pruebas junto con un conjunto de operaciones ya integradas
- **Soporte de Herencia:** Habilita la herencia de entidades, es una herramienta muy útil para simplificar el modelo.
- **Validación avanzada:** Permite verificar y customizar comportamientos de los atributos y las entidades de manera avanzada.
- **Refinar el comportamiento predefinido:** Configuración avanzada del ruteo y los controladores.
- **Lógica de negocio básica y Comportamiento y lógica de negocio:** Creación de acciones personalizada (lógica) en los módulos y definir un controlador con esa acción.
- **Java Persistence API:** Configuración del ORM, relación y persistencias.

- **Anotaciones:** Son atributos que permiten dotar a los atributos y clases con ciertos comportamientos.

Interfaces

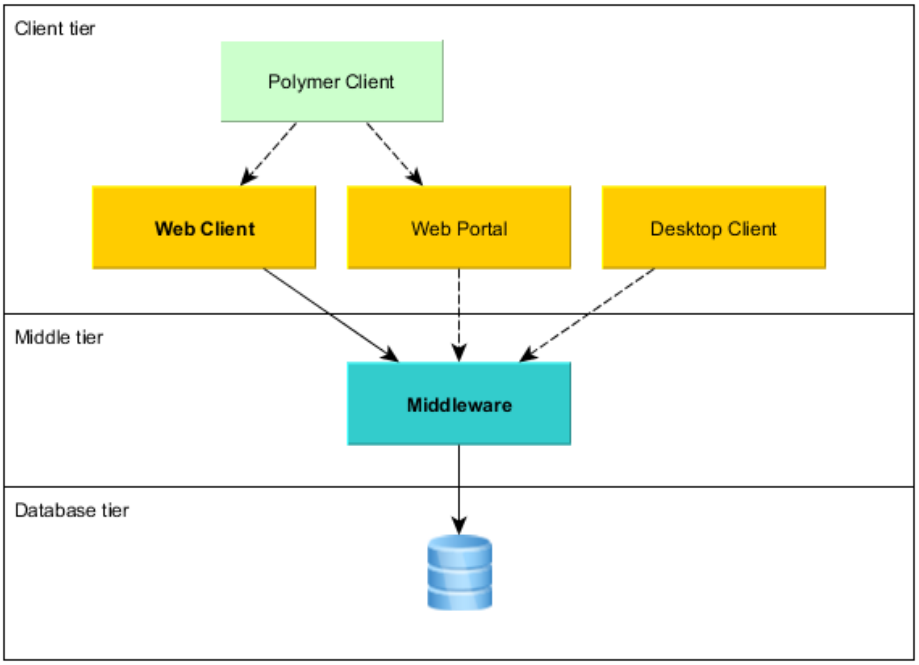


Figura 20. Interface de OpenXava.

1.2.1.3. Tecnología de la información

Aprender cómo escribir clases de Java simples es suficiente para escribir aplicaciones completas. La distribución de OpenXava viene preparada para un inicio rápido.

- Aplicaciones con mucha funcionalidad
- Interfaz de usuario AJAX sin recarga de página. Modo lista con paginación, ordenación, filtrado, añadir/quitar/mover columnas, informes PDF, exportación a Excel, formato tarjetas, gráficos, etc. Modo detalle con pestañas, marcos, diálogos, editores para referencias y colecciones, disposición adaptable, etc.
- Interfaz de usuario móvil
- Aparte de la interfaz de usuario web convencional puedes obtener una aplicación web para móvil a partir del mismo código.
- Código abierto
- Licencia LGPL que te permite desarrollar aplicaciones comerciales sin pagar nada.
- El marco de trabajo Java orientado al dominio más usado

1.2.2. Relevamiento detallado y análisis del sistema

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

Funcionalidad: Creación del proyecto y crear nueva entidad
En primer lugar, has de crear un proyecto OpenXava nuevo

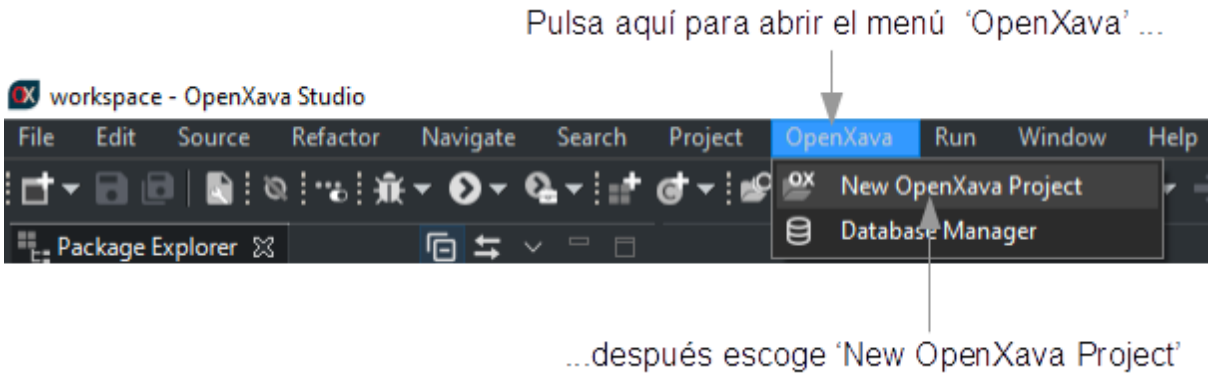


Figura 21. Menú de creación de un nuevo proyecto OPENXAVA

Entonces aparecerá un asistente. Teclea el nombre del proyecto, Facturación. Ten cuidado de poner la F de Facturación en mayúscula y no usar acento, para que los vínculos de abajo funcionen correctamente. Recuerda también escoger español como idioma para que el código de este tutorial funcione bien. Presiona en Finish:

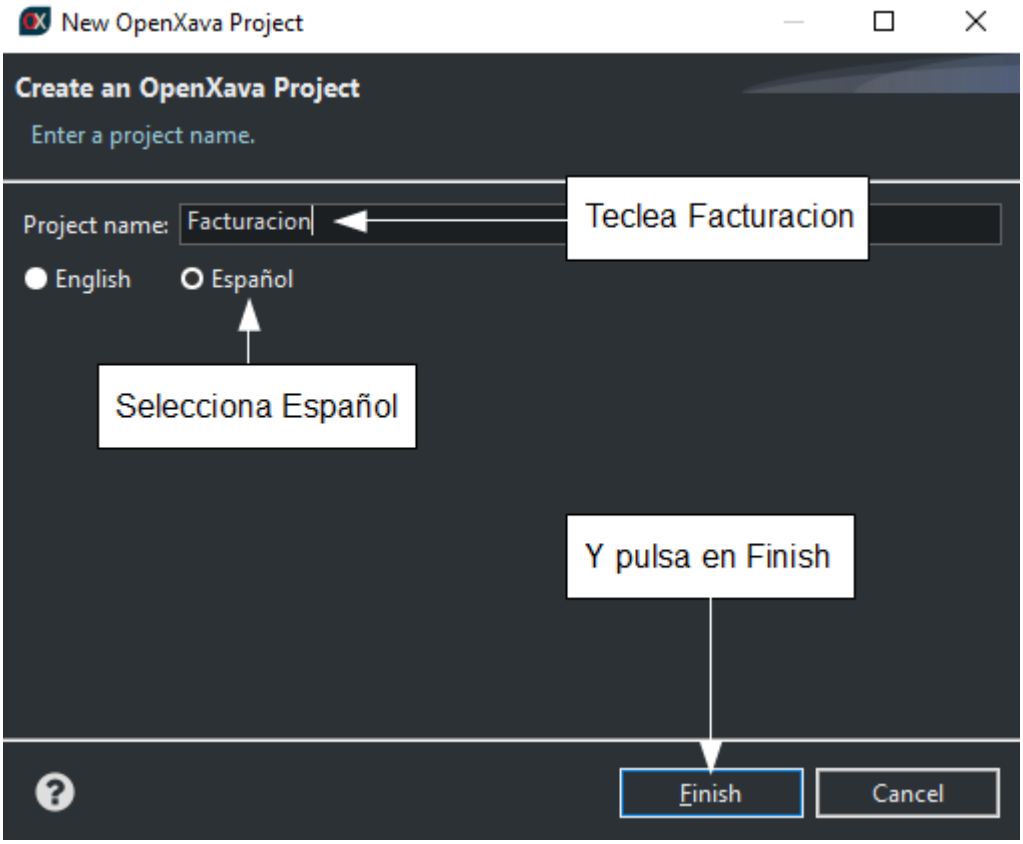


Figura 22. Opciones de creación de un nuevo proyecto OPENXAVA.

Ahora tu proyecto ya está listo para empezar a escribir código:

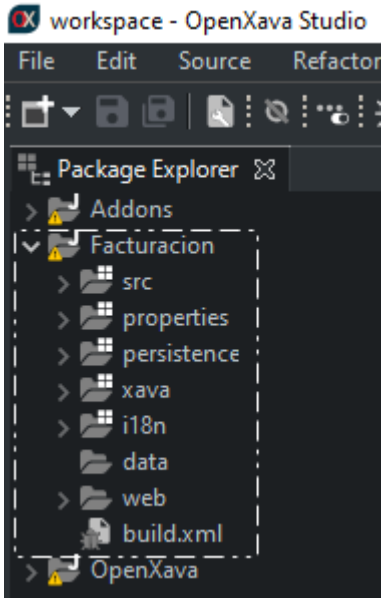


Figura 23. Estructura básica de un proyecto OPENXAVA.

Funcionalidad: Crear entidad

Desarrollar es muy fácil: solo has de añadir entidades para ir haciendo crecer tu aplicación. Empezaremos con una versión simplificada de Cliente con solo número y descripción.
Selecciona el paquete com.tuempresa.facturacion.modelo y pulsa el botón New Java Class

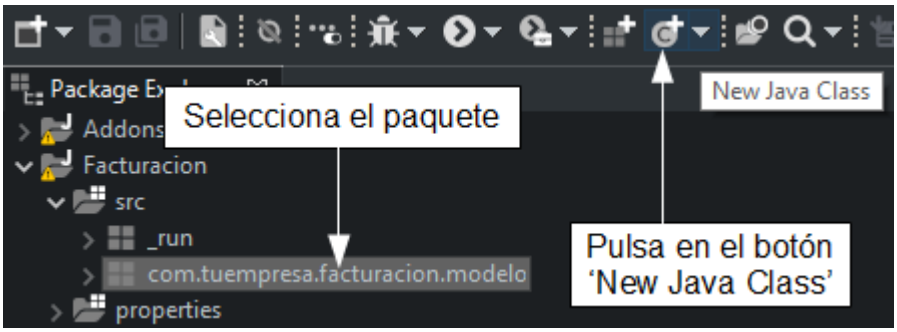


Figura 24. Menú de creación de una nueva entidad OPENXAVA.

Después teclea Cliente como nombre de clase y pulsa Finish.

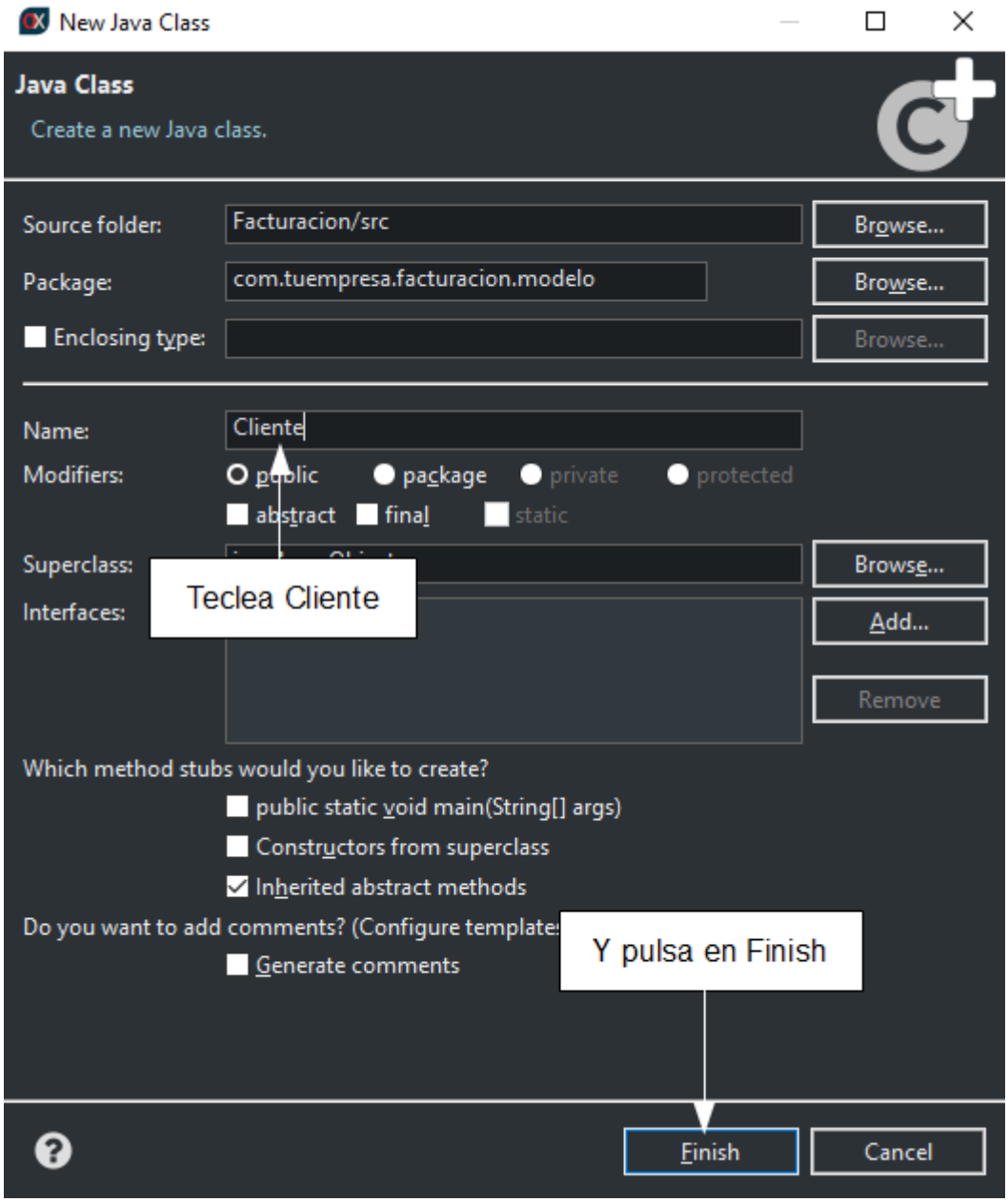


Figura 25. Opciones de creación de una nueva entidad OPENXAVA.

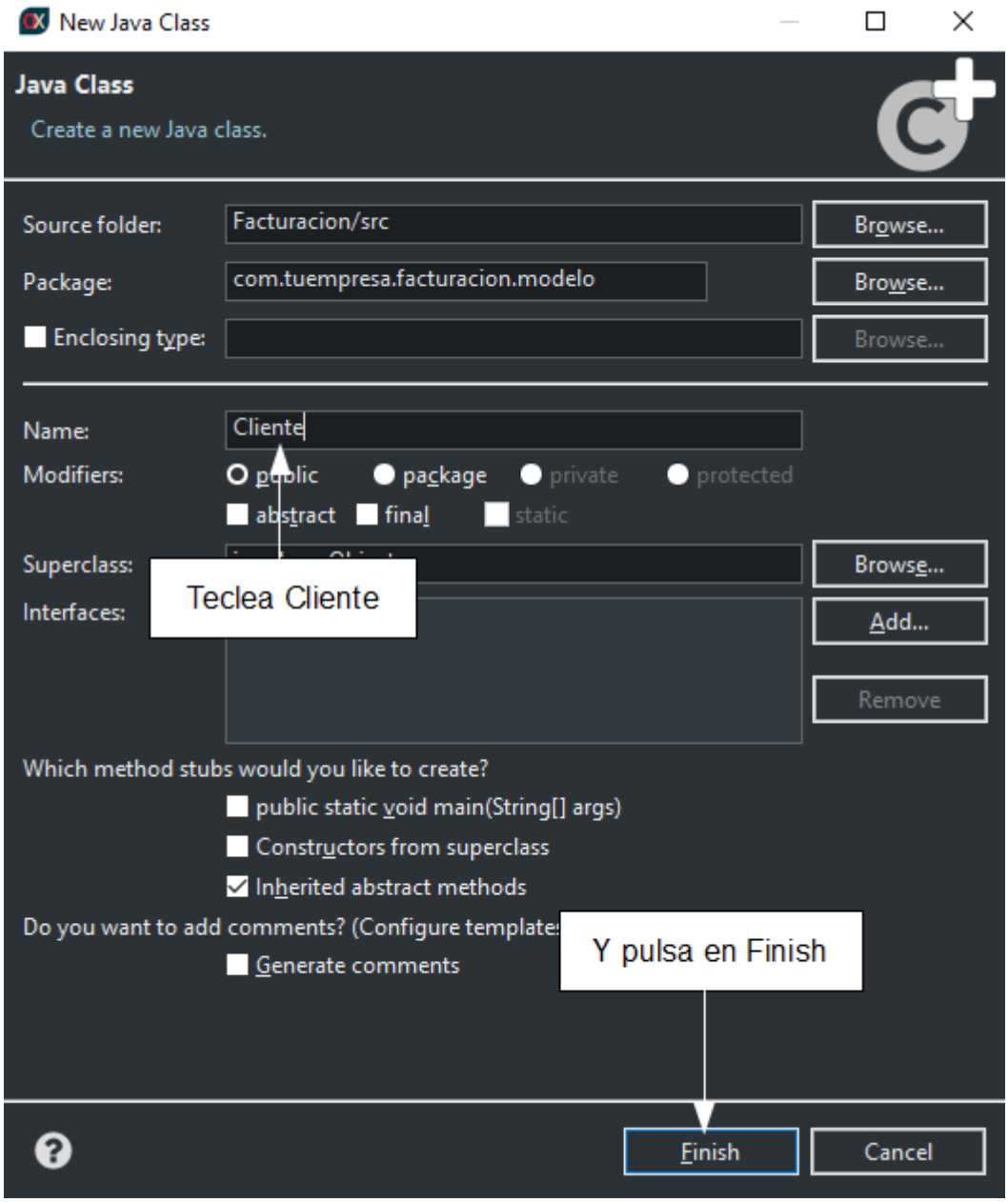


Figura 26. Opciones de creación de una nueva entidad 2 OPENXAVA.

Funcionalidad: Pruebas automáticas

Junit: Es una herramienta muy popular para hacer pruebas automáticas. Esta herramienta está integrada con OpenXava Studio, por tanto, no se necesita descargarla para poder usarse. OpenXava extiende las capacidades de JUnit para permitir probar un módulo de OpenXava exactamente de la misma forma que lo haría un usuario final. De hecho, OpenXava usa HtmlUnit, un software que simula un navegador real (incluyendo JavaScript) desde Java. Todo está disponible desde la clase de OpenXava ModuleTestBase, que te permite automatizar las pruebas que tú harías a mano usando un navegador de verdad de una forma simple.

La mejor manera de entender cómo funcionan las pruebas en OpenXava es verlo en acción.

ModuleTestBase: Para crear una prueba para un módulo de OpenXava extendemos de la clase ModuleTestBase del paquete org.openxava.tests. Esta clase te permite conectar con un módulo OpenXava como un navegador real, y tiene muchos métodos útiles para probar tu módulo. Creemos la prueba para tu módulo Cliente.

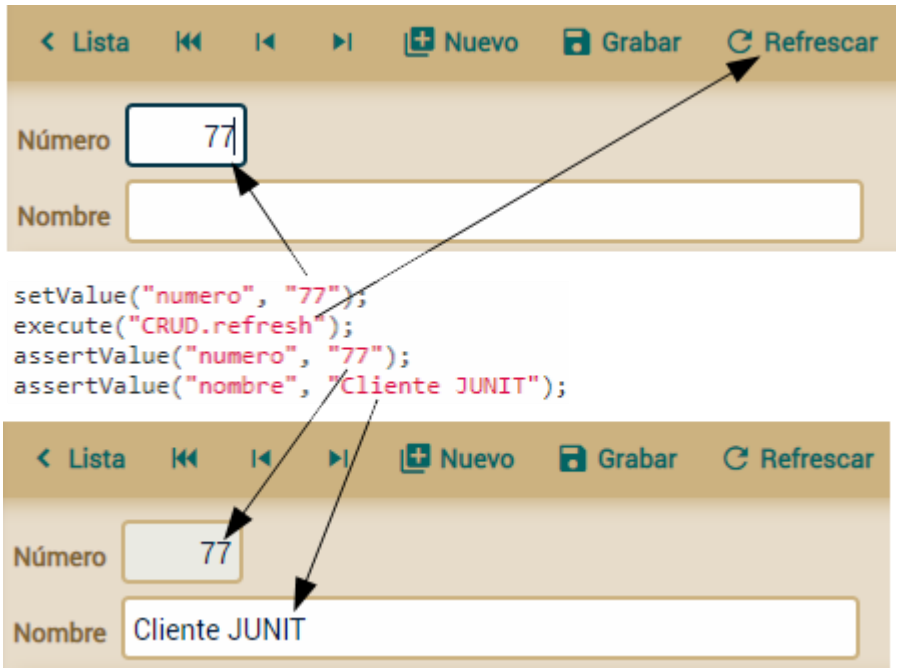


Figura 27. Opciones ModuleTestBase OPENXAVA.

En execute() tienes que especificar el nombre calificado de la acción, esto quiere decir NombreControlador.nombreAccion. ¿Cómo puedes saber el nombre de la acción? Pasea tu ratón sobre el vínculo de la acción y verás en la barra inferior de tu navegador un código JavaScript que incluye el nombre calificado de la acción:

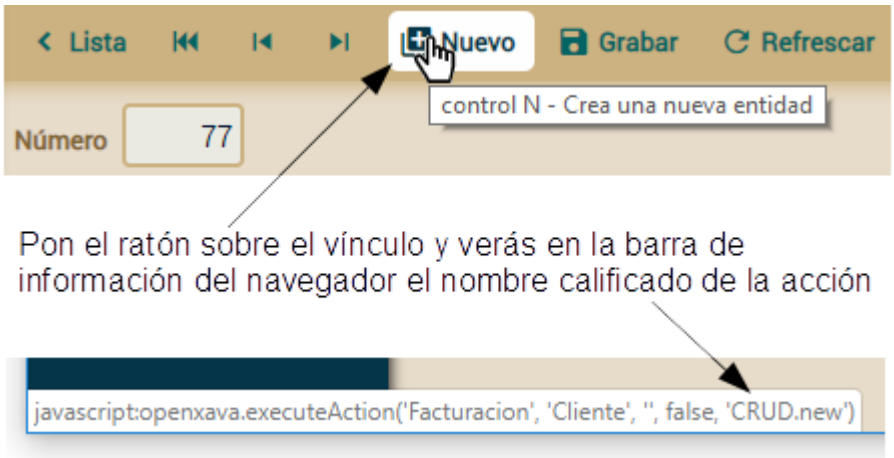


Figura 28. Nueva entidad en ModuleTestBase OPENXAVA.

Ejecutar las pruebas desde OpenXava Studio: JUnit está integrado dentro del OpenXava Studio, por eso ejecutar las pruebas es muy fácil. Para que la prueba funcione su aplicación tiene que estar

ejecutándose, si no es el caso se inicia antes. Para ejecutar la prueba pon el ratón sobre la clase de prueba, PruebaCliente, y con el botón derecho escoge Run As > JUnit Test:

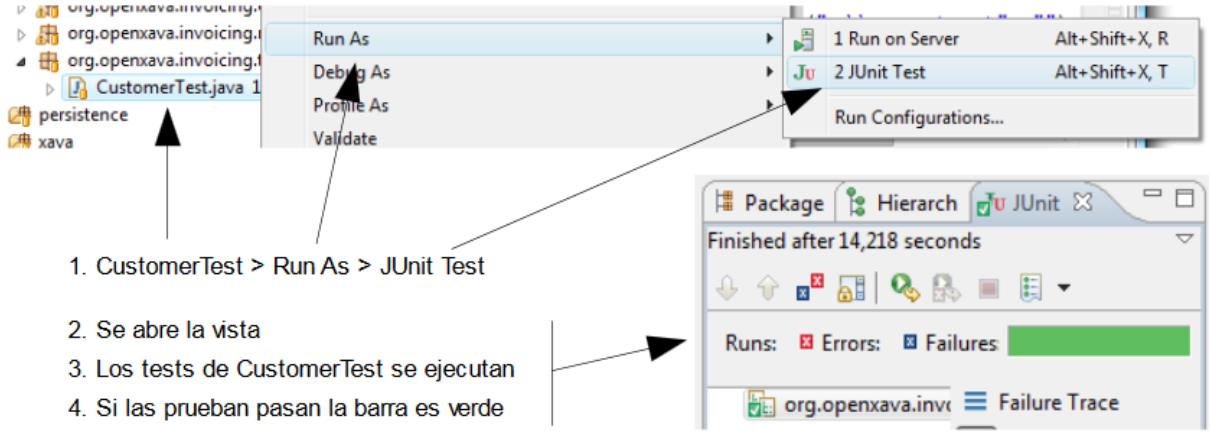


Figura 29. Ejecución de pruebas en JUnit OPENXAVA.

Funcionalidad: Soporte de Herencia

Heredar de una superclase mapeada: Las clases Autor, Categoría y Factura tienen algo de código en común. Este código es la definición del campo oid:

```
@Id @GeneratedValue(generator="system-uuid") @Hidden
@GenericGenerator(name="system-uuid", strategy="uuid")
@Column(length=32)
String oid;
```

Este código es exactamente el mismo para todas estas clases.
Primero, movemos el código común a una clase marcada como @MappedSuperclass. La llamamos Identificable:

```
@MappedSuperclass // Marcada como una superclase mapeada en vez de como una entidad
@Getter @Setter
public class Identificable {

   @Id @GeneratedValue(generator="system-uuid") @Hidden
   @GenericGenerator(name="system-uuid", strategy="uuid")
   @Column(length=32)
   String oid; // La definición de propiedad incluye anotaciones de OpenXava y JPA
```

Ahora puedes definir las entidades Autor, Categoría y Factura de una manera más sucinta. Para ver un ejemplo aquí tienes el nuevo código para Categoría:

```
@Entity @Getter @Setter
public class Categoria extends Identificable { // Extiende de Identificable
                // por tanto no necesita tener una propiedad id
   @Column(length=50)
   String descripcion;
```

```
}
```

Herencia de entidades**:** Una entidad puede heredar de otra entidad. Esta herencia de entidades es una herramienta muy útil para simplificar tu modelo. Vamos a usarla para añadir una nueva entidad, Pedido, a tu aplicación Facturación.

Se quiere añadir un nuevo concepto a la aplicación Facturación: pedido. Mientras que una factura es algo que quieres cobrar a tu cliente, un pedido es algo que tu cliente te ha solicitado. Estos dos conceptos están fuertemente unidos, porque cobrarás por las cosas que tu cliente te ha pedido y tú le has servido.
Sería interesante poder tratar pedidos en tu aplicación y asociar estos pedidos con sus correspondientes facturas. Tal como muestra el siguiente diagrama UML:

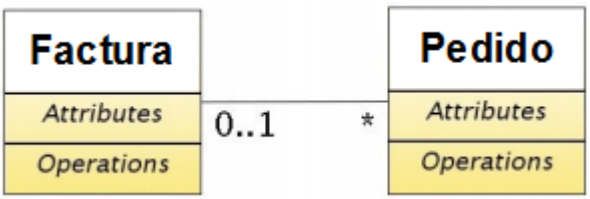


Figura 30. Relación Factura-Pedido OPENXAVA.

Y con código Java:

```
@Entity
public class Factura {

   @OneToMany(mappedBy="factura")
   Collection<Pedido> pedidos;
   ...
}

@Entity
public class Pedido {

   @ManyToOne // Sin carga vaga (1)
   Factura factura;
   ...
}
```

Propiedad calculada simple: El primer paso será añadir una propiedad de importe a Detalle. Lo que queremos es que cuando el usuario elija un producto y teclee la cantidad el importe de la línea sea recalculado y mostrado al usuario:

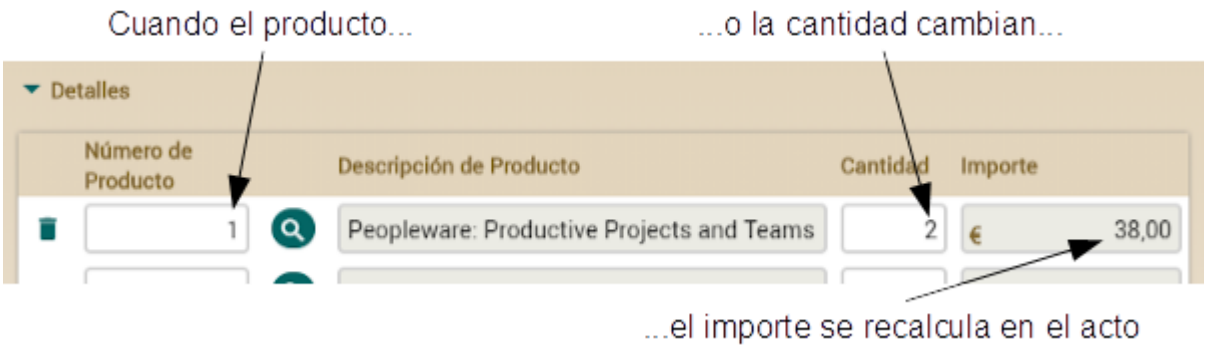


Figura 31. Ejemplo de propiedades calculadas OPENXAVA.

Añadir esta funcionalidad a tu actual código es prácticamente añadir una propiedad calculada a Detalle. Simplemente añade el código siguiente a la clase Detalle:

```
// Propiedad persistente
@Stereotype("DINERO")
@Depends("producto.numero, cantidad") // Cuando usuario cambie producto o cantidad
public BigDecimal getImporte() { // esta propiedad se recalculará y se redibujará
   if (producto == null || producto.getPrecio() == null) return BigDecimal.ZERO;
   return new BigDecimal(cantidad).multiply(producto.getPrecio());
}
```

```
@ElementCollection
@ListProperties("producto.numero, producto.descripcion, cantidad, importe") // importe añadida
Collection<Detalle> detalles;
```

Usar @DefaultValueCalculator
La forma en que calculamos el importe de la línea de detalle no es la mejor. Tiene, al menos, dos inconvenientes. El primero es que el usuario puede querer tener la posibilidad de cambiar el precio unitario. Y segundo, si el precio de un producto cambia los importes de todas las facturas cambian también, y esto no es bueno.

Para evitar estos inconvenientes lo mejor es almacenar el precio de cada producto en cada línea de detalle. Añadamos pues una propiedad persistente precioPorUnidad a la clase Detalle y calculemos su valor desde precio de Producto usando un @DefaultValueCalculator. De tal forma que consigamos el efecto que puedes ver en la siguiente figura:

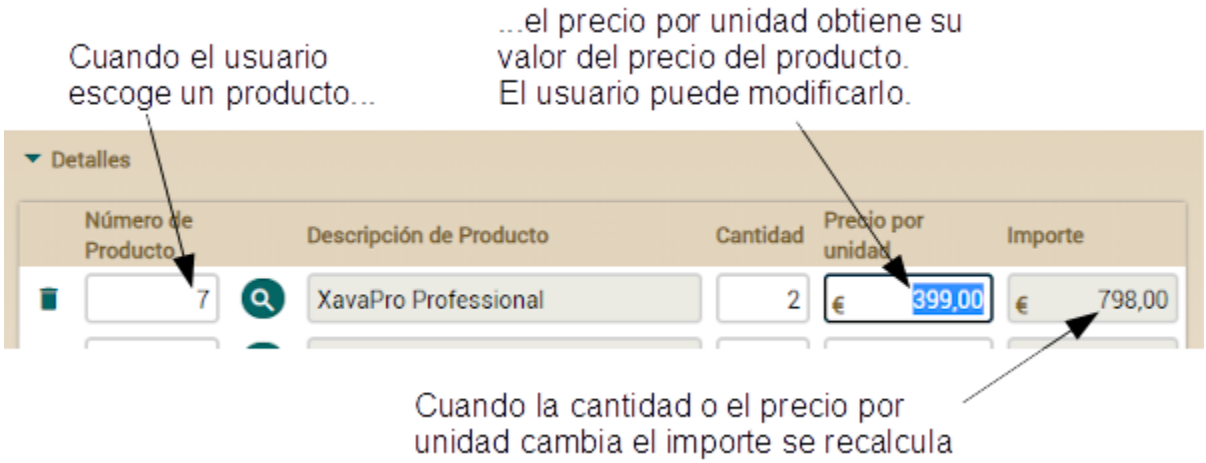


Figura 32. Ejemplo de propiedades calculadas 2 OPENXAVA.

Funcionalidad: Lógica de negocio básica

Propiedades persistentes con @Calculation

A veces las propiedades calculadas no son la mejor opción. Imagínate que tienes una propiedad calculada en Factura, digamos descuento:

```
// NO LO AÑADAS A TU CÓDIGO, ES SÓLO PARA ILUSTRAR
public BigDecimal getDescuento() {
   return getImporte().multiply(new BigDecimal("0.10"));
}
```

Si necesitas procesar todas las facturas cuyo descuento sea mayor de 1000, has de escribir un código como el siguiente:

```
// NO LO AÑADAS A TU CÓDIGO, ES SÓLO PARA ILUSTRAR
Query query = getManager().createQuery("from Factura"); // Sin condición en la consulta
for (Object o: query.getResultList()) { // Itera por todos los objetos
   Factura f = (Factura) o;
   if (f.getDescuento() // Pregunta a cada objeto
      .compareTo(new BigDecimal("1000")) > 0) {
         f.hacerAlgo();
   }
}
```

De esta manera ponemos el peso de seleccionar los registros en el servidor de la base de datos y no en el servidor Java. Además, los descuentos no se recalculan cada vez, sino que ya está calculados y grabados.

Este hecho tiene también efecto en el modo lista, porque el usuario no puede filtrar ni ordenar por las propiedades calculadas, pero sí que lo puede hacer usando propiedades persistentes con @Calculation:

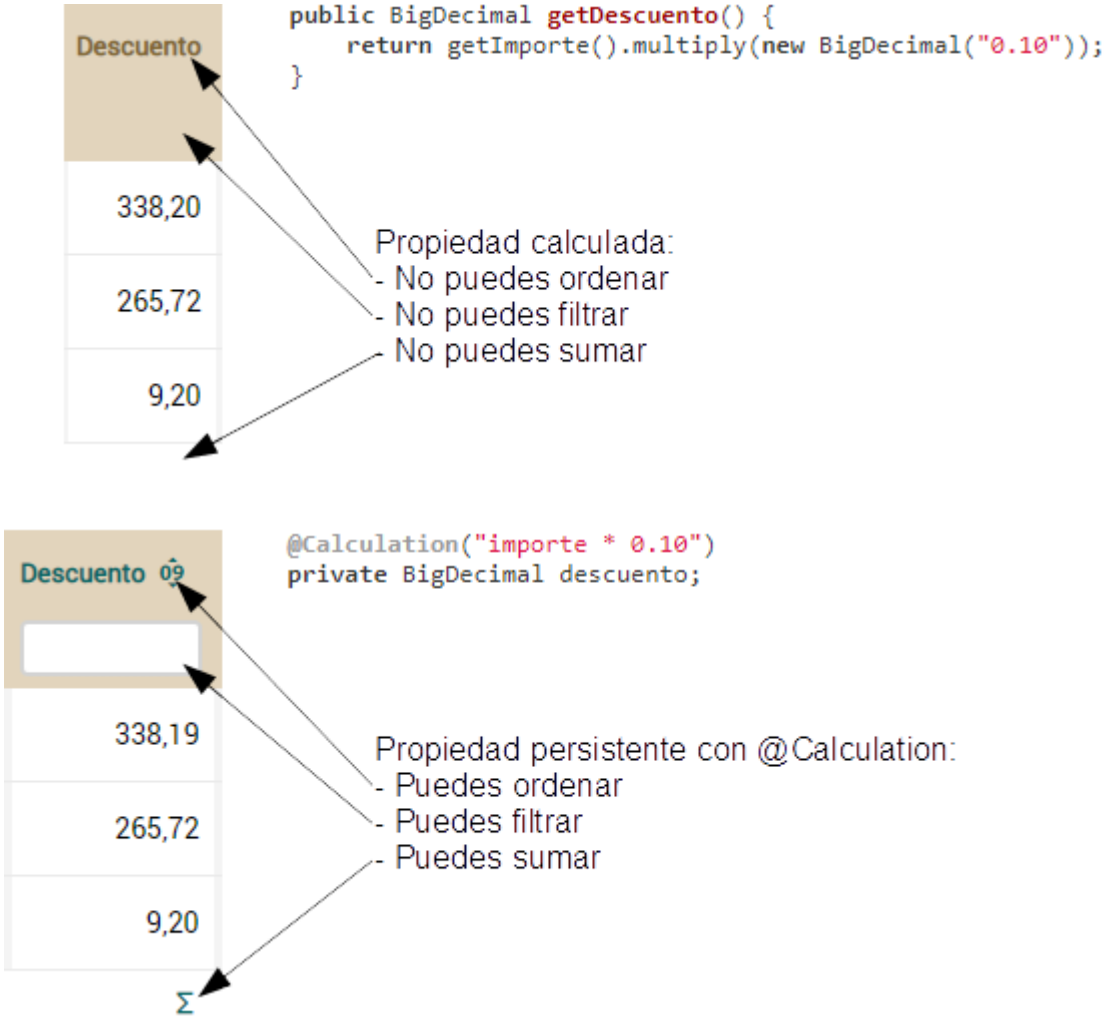


Figura 33. Uso de @Calculation OPENXAVA

Propiedades de total de una colección

También queremos añadir importes a Pedido y Factura. Tener IVA, importe base e importe total es indispensable. Para hacerlo sólo necesitas añadir unas pocas propiedades a la clase DocumentoComercial. La siguiente figura muestra la interfaz de usuario para estas propiedades:

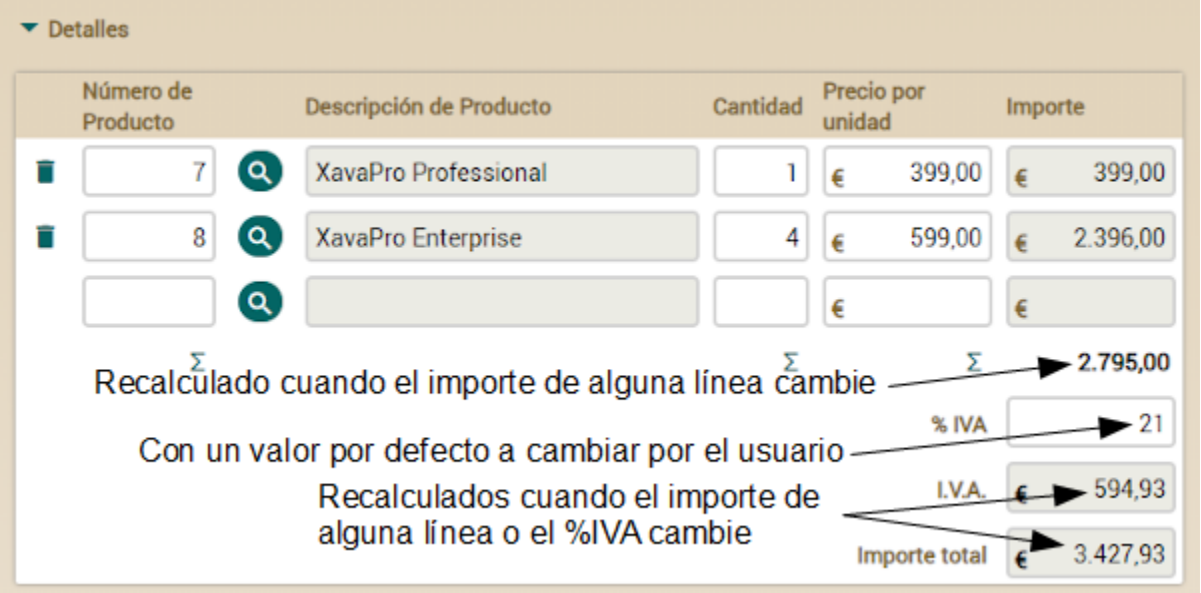


Figura 34. Múltiples propiedades calculadas OPENXAVA

Funcionalidad: Validación avanzada

Añadir la propiedad entregado a Pedido: Para hacer esto, lo primero es añadir una nueva propiedad a la entidad Pedido. La propiedad "entregado":

```
@Column(columnDefinition="BOOLEAN DEFAULT FALSE")
boolean entregado;
```

Además, es necesario añadir la propiedad entregado a la vista. Modifica la vista Pedido como muestra el siguiente código:

```
@View(extendsView="super.DEFAULT",
   members=
      "diasEntregaEstimados, entregado, " + // Añade entregado
      "factura { factura }"
)
...
public class Pedido extends DocumentoComercial {
```

Ahora podemos intentar añadir pedidos a una factura con la aplicación, verás como los pedidos no entregados son rechazados. Ve al módulo Facturas, selecciona la pestaña PEDIDOS de una factura y desde ahí pulsa en el botón Añadir:

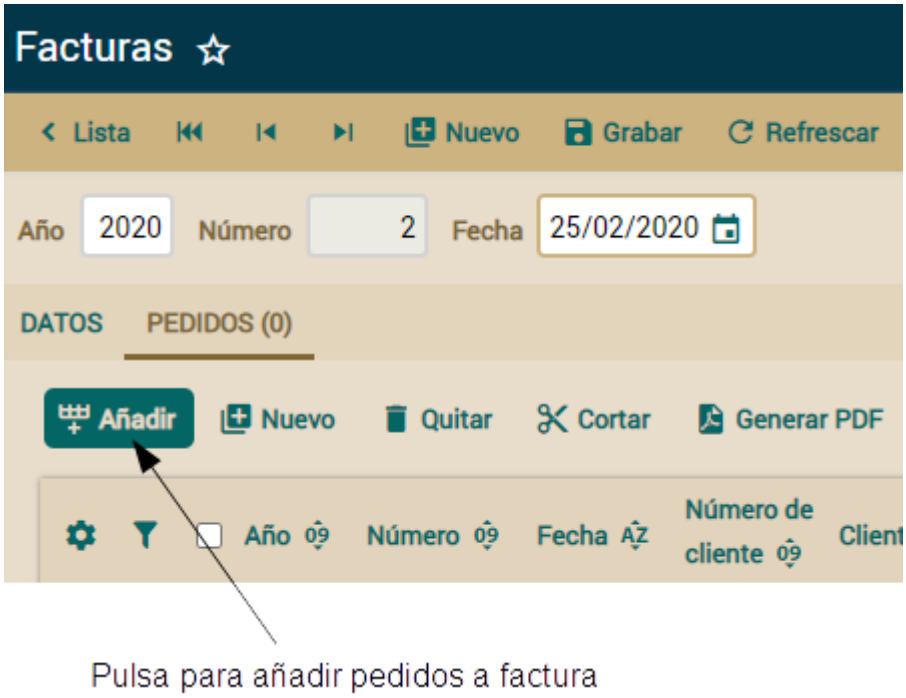


Figura 35. Añadir un pedido a una factura OPENXAVA.

Se muestra un diálogo con una lista de pedidos para escoger. Selecciona dos, uno de ellos no entregado todavía y pulsa en AÑADIR:

Figura 36. Añadir un pedido a una factura 2 OPENXAVA.

Entonces el pedido entregado se añadirá mientras que el otro es rechazado, generando los siguientes mensajes:

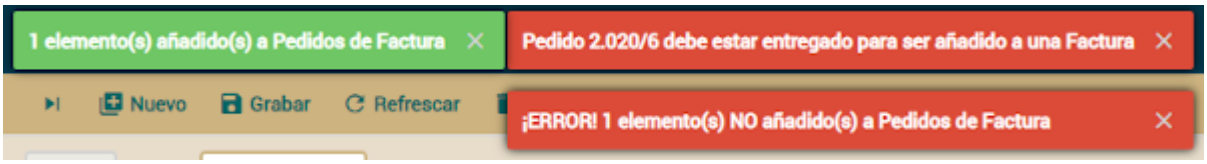


Figura 37. Error al añadir un pedido OPENXAVA.

Definir tu propia anotación ISBN
Creemos la anotación @ISBN. Primero, crea un paquete en tu proyecto llamado com.tuempresa.facturacion.anotaciones. Pulsa en él con el botón derecho del ratón y escoge New > Annotation, como sigue

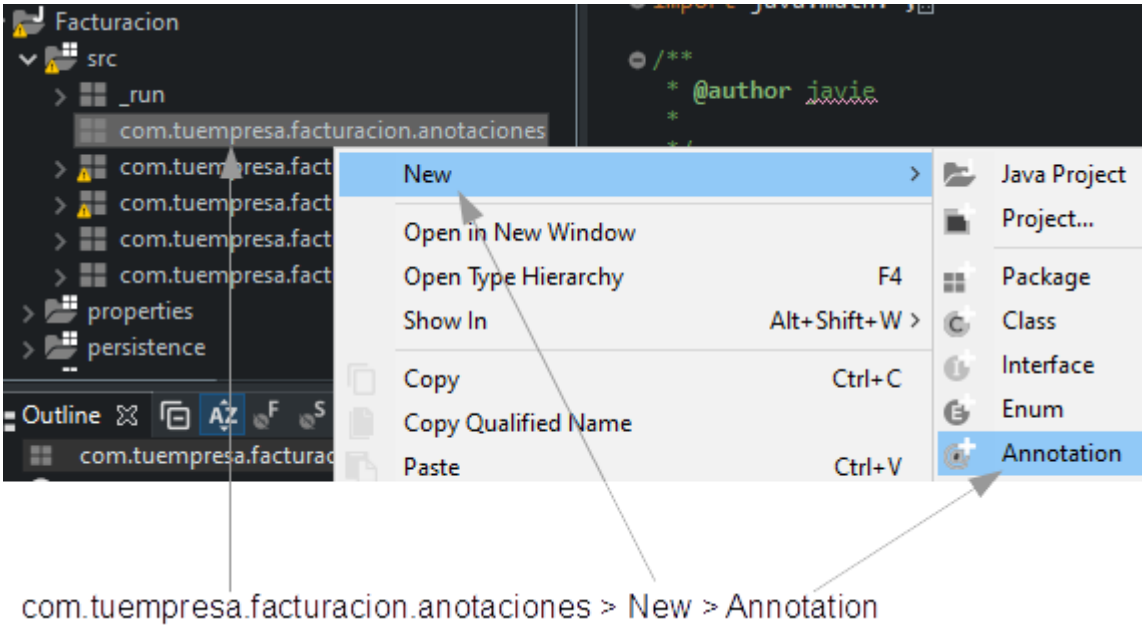


Figura 38. Nueva anotación OPENXAVA.

Se muestra un diálogo, teclea ISBN y pulsa en Finish:

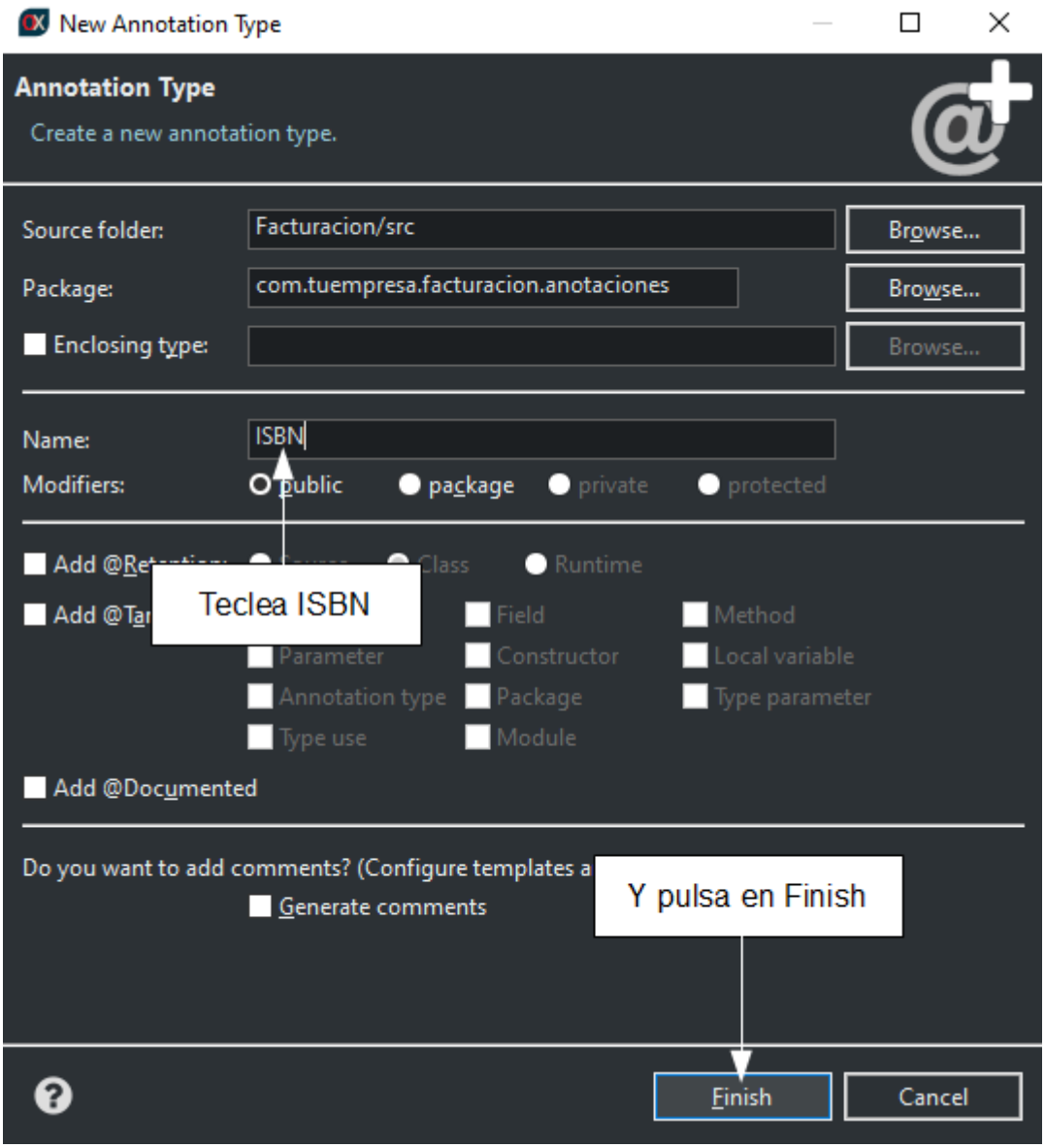


Figura 39. Opciones de la nueva anotación OPENXAVA.

Funcionalidad: Refinar el comportamiento predefinido con "acciones personalizadas"
Controlador Typical
Por defecto el módulo Factura usa las acciones del controlador Typical. El controlador Typical está definido en default-controllers.xml que se encuentra en la carpeta OpenXava/xava de tu workspace. Una definición de controlador es un fragmento de XML con una lista de acciones. OpenXava aplica por defecto el controlador Typical a todos los módulos. Puedes ver su definición:

```
<controller name="Typical"> <!-- 'Typical' hereda sus acciones de los controladores -->
   <extends controller="Navigation"/> <!-- 'Navigation', -->
   <extends controller="CRUD"/> <!-- 'CRUD' -->
   <extends controller="Print"/> <!-- 'Print' -->
   <extends controller="ImportData"/> <!-- e 'ImportData' -->
</controller>
```

Refinar el controlador para un módulo
Empezaremos refinando la acción para borrar del módulo Factura. Nuestro objetivo es que cuando el usuario pulse en el botón de borrar, la factura no sea borrada de la base de datos, sino que simplemente se marque como borrada. De esta forma, podemos recuperar las facturas borradas si fuese necesario.

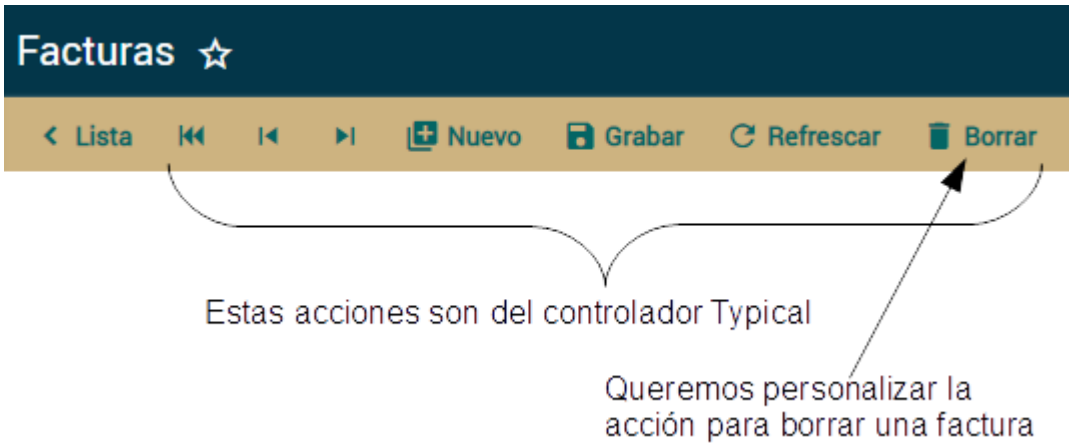


Figura 40. Acciones de un típico controlador OPENXAVA.

La figura anterior muestra las acciones de Typical; queremos todas estas acciones en nuestro módulo Factura, con la excepción de que vamos a escribir nuestra propia lógica para la acción de borrar.
Define tu propio controlador para Factura añadiéndolo al archivo controladores.xml de la carpeta xava de tu proyecto

Escribir tu propia acción
Primero crea un nuevo paquete llamado com.tuempresa.facturacion.acciones. Después añádele una clase EliminarFactura, con este código:

```
public class EliminarFactura
   extends ViewBaseAction { // ViewBaseAction tiene getView(), addMessage(), etc

   public void execute() throws Exception { // La lógica de la acción
      addMessage( // Añade un mensaje para mostrar al usuario
         "¡No te preocupes! Sólo he borrado la pantalla");
      getView().clear(); // getView() devuelve el objeto xava_view
         // clear() borra los datos en la interfaz de usuario
   }
}
```

Una acción es una clase simple. Tiene un método execute() con la lógica a hacer cuando el usuario pulse en el botón o vínculo correspondiente. Una acción ha de implementar la interfaz org.openxava.actions.IAction, aunque normalmente es más práctico extender de BaseAction, ViewBaseAction o cualquier otra acción base del paquete org.openxava.actions.
ViewBaseAction tiene una propiedad view que puedes usar desde dentro de execute() mediante getView(). Este objeto del tipo org.openxava.view.View permite manejar la interfaz de usuario, en este caso borramos los datos visualizados usando getView().clear().

También usamos addMessage(). Todos los mensajes añadidos con addMessage() se muestra al usuario al final de la ejecución de la acción. Puedes, bien añadir el mensaje a mostrar, o bien un id de una entrada en i18n/MensajesFacturacion_es.properties.
La siguiente imagen muestra el comportamiento del módulo Factura después de añadir la acción de borrar personalizada:

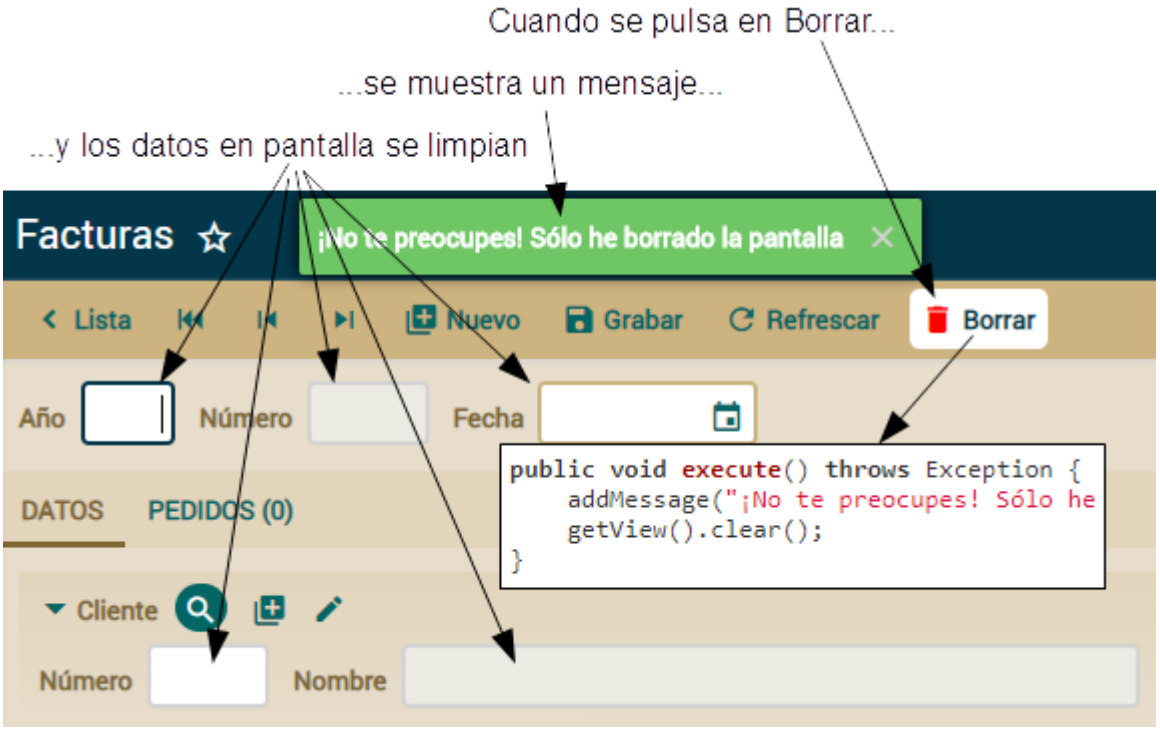


Figura 41. Mensaje al intentar borrar una factura OPENXAVA.

Acciones genéricas
El código actual de EliminarFactura refleja la forma típica de escribir acciones. Es un código concreto que accede directamente a entidades concretas para manipularlas.
Pero a veces puedes encontrarte alguna lógica en tu acción susceptible de ser usada y reusada por toda tu aplicación, incluso en todas tus aplicaciones. En este caso, puedes utilizar algunas técnicas para crear código más reutilizable y así convertir tus acciones en acciones genéricas.
Modificar y gestionar el Comportamiento y lógica de negocio aplicados al proyecto

Funcionalidad: Lógica de negocio desde el modo detalle
Empezaremos con el caso más simple: un botón en modo detalle para ejecutar cierta lógica. En este caso para crear la factura desde un pedido:

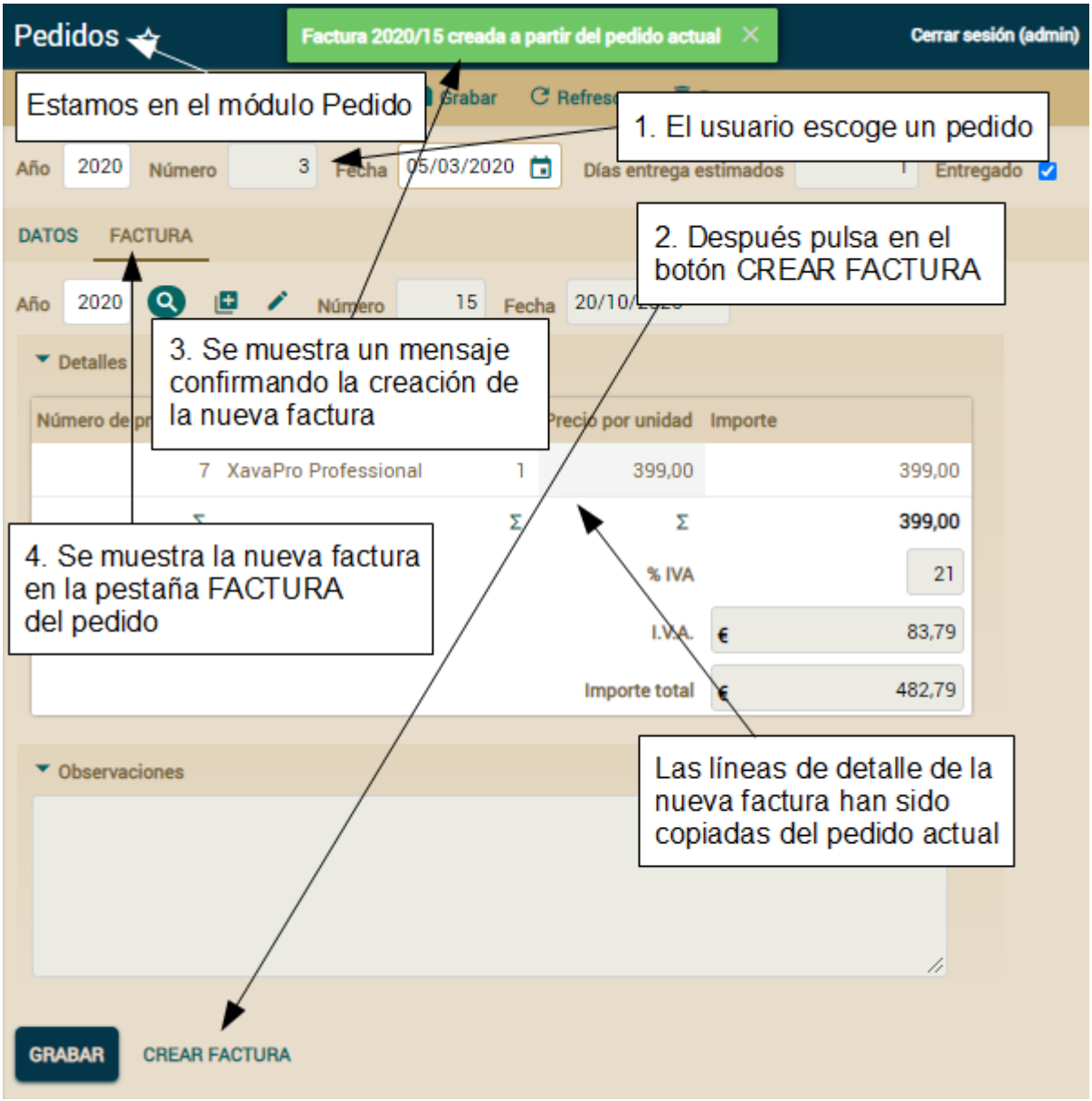


Figura 42. Lógica de negocio desde el modo detalle OPENXAVA.

Crear una acción para ejecutar lógica personalizada

Como ya sabes el primer paso para tener una acción personalizada en tu módulo es definir un controlador con esa acción. Por tanto, editemos controladores.xml y añadamos un nuevo controlador. El siguiente código muestra el controlador Pedido:

```
<controlador nombre="Pedido">
  <hereda-de controlador="Facturacion"/> <!-- Para tener las acciones estándar -->

  <accion nombre="crearFactura" modo="detail"
    clase="com.tuempresa.facturacion.acciones.CrearFacturaDesdePedido"/>
  <!-- modo="detail" : Sólo en modo detalle -->

</controlador>
```

Escribiendo la lógica de negocio real en la entidad
La lógica de negocio para crear una nueva Factura está en la entidad Pedido, no en la acción. Poner la lógica para crear una Factura dentro de Pedido es un enfoque pragmático, porque de esta forma podemos usar esta lógica desde otras acciones, procesos masivos, servicios web, etc.

Veamos el código del método crearFactura() de la clase Pedido, no olvides añadir los imports indicados:

```
public class Pedido extends DocumentoComercial {

   ...

   public void crearFactura() throws Exception { // throws Exception para tener
                                                 // un código más simple, de momento
      Factura factura = new Factura(); // Instancia una factura
      BeanUtils.copyProperties(factura, this); // y copia el estado del pedido actual
      factura.setOid(null); // Para que JPA sepa que esta entidad todavía no existe
      factura.setFecha(LocalDate.now()); // La fecha para la nueva factura es hoy
      factura.setDetalles(new ArrayList<>(getDetalles())); // Clona la colección detalles
      XPersistence.getManager().persist(factura);
      this.factura = factura; // Siempre después de persist()
   }
}
```

Lógica de negocio desde el modo lista
Las acciones de lista son una herramienta utilísima para dar al usuario la posibilidad de aplicar lógica a varios objetos a la vez. En nuestro caso, podemos añadir una acción en el modo lista para crear una nueva factura automáticamente a partir de varios pedidos seleccionados en la lista, de esta manera:

Figura 43. Lógica de negocio desde el modo lista OPENXAVA.

Aquí se muestra cómo esta acción de lista agarra los pedidos seleccionados y crea una factura a partir de ellos. Simplemente copia los datos del pedido en la nueva factura, añadiendo las líneas de detalle de todos los pedidos en una única factura. También se muestra un mensaje. Veamos cómo codificar este comportamiento.

Acción de lista con lógica propia

Como ya sabes, el primer paso para tener una acción propia en tu módulo es añadirla a un controlador. Por tanto, editemos controladores.xml añadiendo una nueva acción al controlador Pedido:

```
<controlador nombre="Pedido">

  ...

  <!-- La nueva acción -->
  <accion nombre="crearFacturaDesdePedidosSeleccionados"
    modo="list"
```

```
    clase="com.tuempresa.facturacion.acciones.CrearFacturaDesdePedidosSeleccionados"/>
  <!-- modo="list": Sólo se muestra en modo lista -->

</controlador>
```

Funcionalidad: Java Persistence API
Entidad: Una entidad JPA se define así:

```
@Entity  // Para definir esta clase como persistente
@Table(name="GSTCST")  // Para indicar la tabla de la base de datos (opcional)
public class Cliente {
```

Como puedes ver solo has de marcar tu clase con la anotación @Entity y opcionalmente también con la anotación @Table, en este caso estamos diciendo que la entidad Cliente se graba en la tabla GSTCST de la base de datos. De ahora en adelante, JPA se guarda y recupera la información entre los objetos Cliente en la aplicación y la tabla GSTCST en la base de datos, como se muestra aquí:

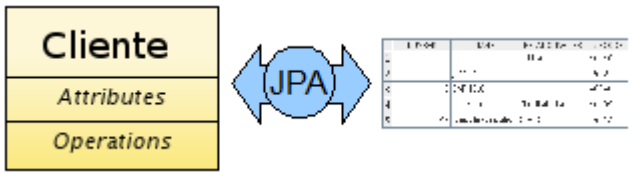


Figura 44. Relación entidad JPA OPENXAVA.

Propiedades
El estado básico de una entidad se representa mediante propiedades. Las propiedades de una entidad son propiedades Java convencionales, con getters y setters:

```
private String nombre;

public String getNombre() {
   return nombre;
}

public void setNombre(String nombre) {
   this.nombre = nombre;
}
```

Por defecto las propiedades son persistentes, es decir, JPA asume que la propiedad nombre se almacena en la columna llamada 'nombre' de la tabla en la base de datos. Si quieres que una determinada propiedad no se guarde en la base de datos has de marcarla como @Transient:

```
@Transient  // Marcada como transitoria, no se almacena en la base de datos
private String nombre;

public String getNombre() {
   return nombre;
}
```

```
public void setNombre(String nombre) {
   this.nombre = nombre;
}
```

Es obligatorio que al menos una propiedad sea clave (1). Has de marcar la propiedad clave con @Id y normalmente se mapea contra la columna clave de la tabla. @Column puede usarse para indicar la longitud sin el nombre de columna (2). La longitud es usada por el motor JPA para la generación de esquema, pero también es usada por OpenXava para conocer el tamaño del editor en la interfaz de usuario. A partir del código de la entidad Cliente OpenXava genera la siguiente interfaz de usuario:

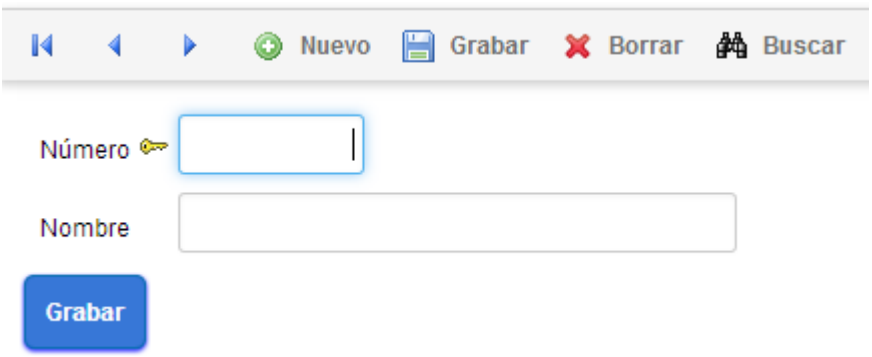


Figura 45. Propiedad clave OPENXAVA

Referencias
Una entidad puede hacer referencia a otra entidad. Únicamente has de definir una referencia Java convencional anotada con la anotación JPA @ManyToOne:

```
@Entity
public class Factura {

   @ManyToOne(  // La referencia se almacena como una relación a nivel de base de datos (1)
      fetch=FetchType.LAZY,  // La referencia se cargará bajo demanda (2)
      optional=false)  // La referencia tiene que tener valor siempre
   @JoinColumn(name="INVCST")  // INVCST es la columna para la clave foranea (3)
   private Cliente cliente;  // Una referencia Java convencional (4)

   // Getter y setter para cliente
```

Figura 46. Relación factura cliente OPENXAVA.

Esta es la interfaz de usuario que OpenXava genera automáticamente para una referencia:

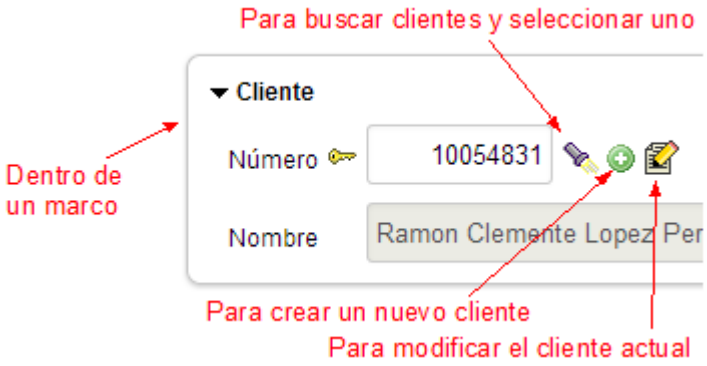


Figura 47. Opciones de la clave primaria OPENXAVA.

Funcionalidad: Anotaciones

Validación declarativa

La forma preferida de hacer validación en OpenXava es mediante anotaciones, es decir, de manera declarativa. Por ejemplo, sólo has de marcar una propiedad como @Required:

```
@Required  // Esto fuerza a validar esta propiedad como requerida al grabar
private String nombre;
```

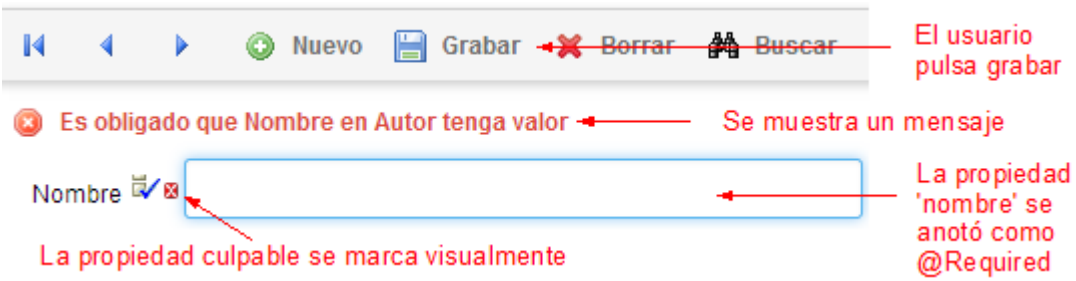


Figura 48. Error de campo @Required OPENXAVA.

1.2.2.2. Modelo Lógico

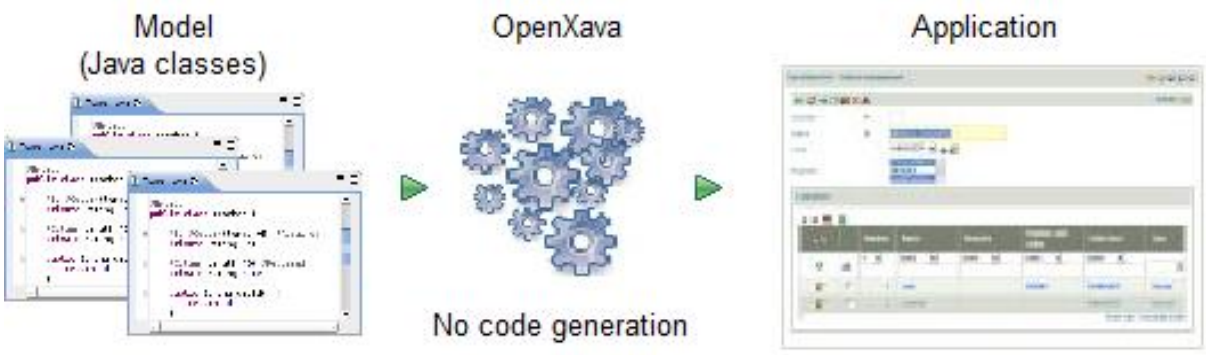


Figura 49. Diagrama funcional (Modelo lógico) OPENXAVA.

1.2.2.3. Problemas y necesidades

Funcionalidad	Problema/Necesidad
Creación del proyecto	En esta funcionalidad toda la creación de proyecto y sus archivos de configuración son definidos en base a plantillas, las cuales están hardcodeadas y no se pueden modificar en base a mayores parámetros, por lo que esta función no se adapta a las distintas empresas que pueden hacer uso de estas funciones.
Modelar con Java	No tiene ninguna funcionalidad de implementación y generación automática de diagramas no se representan las clases, interfaces, etc. en modelos de dominio y no tiene la posibilidad con funciones de implementar funciones orientadas al diseño que incluyen generación de métodos y funciones automáticamente. No implementa ni UML ni ninguna forma de modelado.
Pruebas automáticas	No se encuentran funcionalidades o problemas, en esta función. Se encuentra una posible mejora utilizando otros módulos de testing además de las básicas en Java, o en lenguajes también orientados al Frontend.
Soporte de Herencia	No tiene la posibilidad de añadir descriptores de parametrización específicos que permite el lenguaje java.
Validación avanzada y Anotaciones	Se pueden realizar múltiples validaciones y agregación de etiquetas y Anotaciones, pero no tiene posibilidad de crear conjuntos de estos para ser reutilizados con frameworks ya que cada proyecto debe crear íntegramente sus validaciones ya que deben tener compatibilidad con este.
Refinar el comportamiento predefinido	No se encuentran funcionalidades o problemáticas con esta función.
Lógica de negocio básica y Comportamiento	Si bien se pueden crear lógicas de negocio y aplicar reglas de negocio muy detalladamente, todas deben estar definidas en un paradigma orientado a objetos, al ser solo un sistema basado en Java, tiene que ajustarse a las reglas de negocio con las desventajas que este tiene.
Java Persistence API	Es el único framework predefinido que tiene incluido, es poco óptimo para agregar otro tipo de framework o método de persistencia.

Tabla 2. Problemas y necesidades detectados OPENXAVA.

1.3. JHIPSTER

1.3.1. Relevamiento general

1.3.1.1. Sistema Relevado

En colaboración con otros desarrolladores, Julien Dubois publicó en octubre de 2013 el proyecto JHipster (abreviatura de “Java Hipster”). El nombre se inspira principalmente en el concepto mismo del generador de aplicaciones web, y es que JHipster pretendía unir Java con las herramientas de desarrollo web más populares de ese momento. Hoy en día, esta colección de frameworks tiene como mercado principal al sector empresarial y su objetivo es brindar a las empresas un alto grado de productividad durante el desarrollo, al tiempo que se cuida en extremo la calidad de la aplicación final desarrollada. El desarrollo continuo de este proyecto está en manos de un equipo formado por más

de 15 desarrolladores principales y cientos de colaboradores. El código de JHipster, que se distribuye bajo una licencia Apache 2.0, está disponible de manera gratuita para todo el mundo en GitHub. (JHipster, Dubois, Sasidharan, Pascal, Release notes and Documentation)

https://www.jhipster.tech/releases/+

1.3.1.2. Funciones detectadas e interfaces

A continuación, se presenta una lista de las funcionalidades observadas en este sistema y se indica con qué usuario interactúa cada funcionalidad.

- **Gestión usuario:** Creación de los usuarios y sus permisos**.**
- **Gestión de métricas:** Muestra métricas referidas a la aplicación que se está creando.
- **Funcionalidad Health:** Muestra información sobre el estado de los discos.
- **Funcionalidad Creación de Base de datos:** Gestiona la base de datos que utiliza la aplicación a generar.

Interface

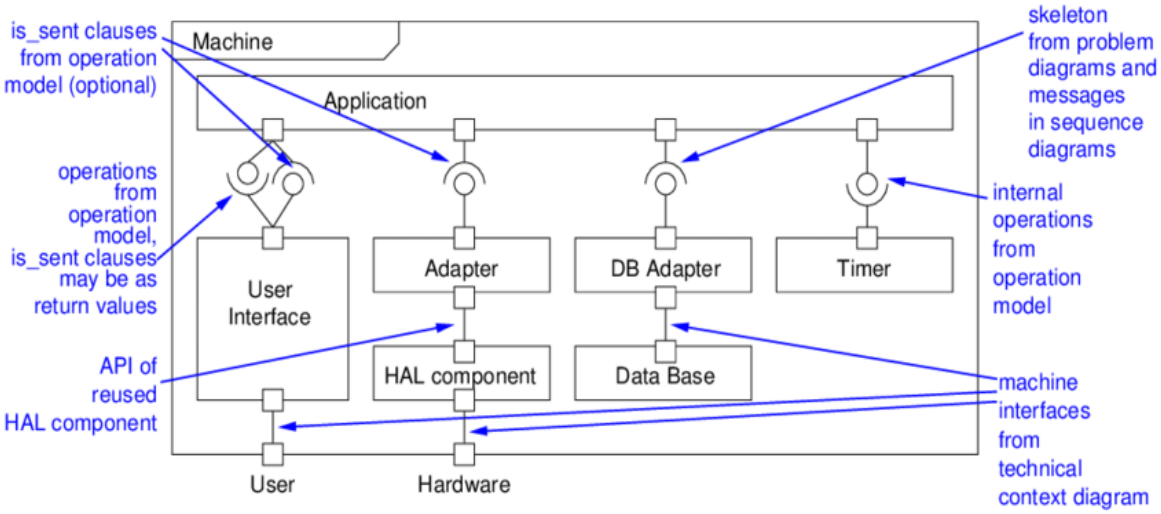


Figura 50. Arquitectura JHipster.

1.3.1.3. Tecnología de Información

JHipster nos ofrece un conjunto de herramientas para desarrollar y diseñar los elementos correspondientes al Frontend y también al Backend de un proyecto de desarrollo. Por ejemplo, el framework Spring Boot es la base más adecuada para obtener un robusto stack de Java del lado del servidor que nos permita conectarnos a varias bases de datos, motores de virtualización y herramientas de seguimiento. Para conectarse al Frontend, utiliza una interfaz REST. A continuación, puedes encontrar algunas de las opciones disponibles del lado del servidor en JHipster:

- **Bases de datos:** MariaDB, PostgreSQL, Oracle, MySQL, MongoDB
- **Virtualización:** Docker, Kubernetes, AWS
- **Entornos de ejecución de pruebas:** Karma, Cucumber
- **Indexación:** ElasticSearch
- **Cachés:** Ehcache, Infinispan

- **Sistemas de monitorización:** Prometheus

En cuanto al desarrollo de Frontend, JHipster se basa principalmente en el framework JavaScript Angular.js y en la biblioteca JavaScript React. Estos dos componentes pueden combinarse con Twitter Bootstrap, el proyecto pionero de frameworks web, y con la plantilla web alternativa HTML5 Boilerplate. También existe la opción de ampliar JHipster utilizando el lenguaje de hoja de estilos SASS, lo que nos sirve para simplificar el diseño mediante CSS3.
A la hora de utilizar todas estas soluciones innovadoras y que el flujo de trabajo sea óptimo, contamos con herramientas como el generador de código Yeoman, el module bundler de JavaScript Webpack o el gestor de tareas Gulp (para JavaScript) así como Maven y Gradle (para Java).

1.3.2. Relevamiento detallado y análisis del sistema

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

Funcionalidad: Gestión de usuarios
Utiliza Spring Security para autenticar y asignar roles. Tiene las funciones de crear, modificar y eliminar usuarios. Los componentes pueden ser reutilizados.

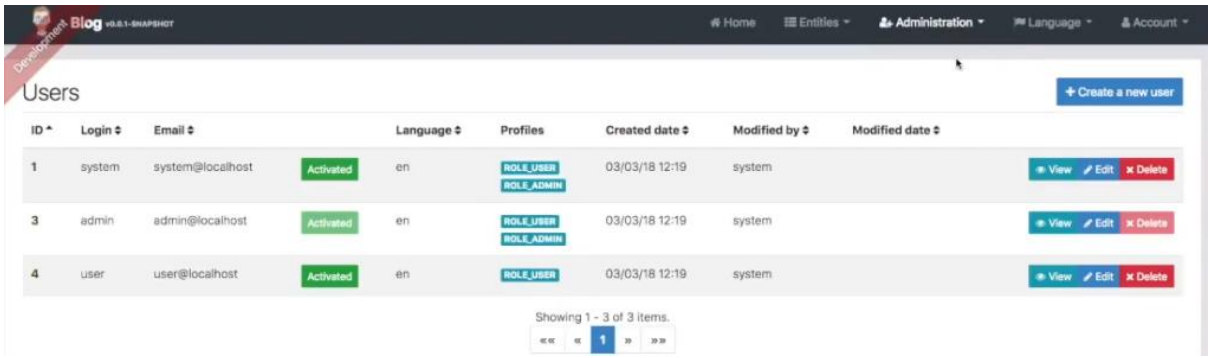


Figura 51. Funcionalidad: Gestión de usuarios JHipster

Funcionalidad: Gestión de métricas
Se muestra la memoria que utiliza la aplicación, los hilos de ejecución, la lista de http request que se están produciendo, los códigos que se están generando.

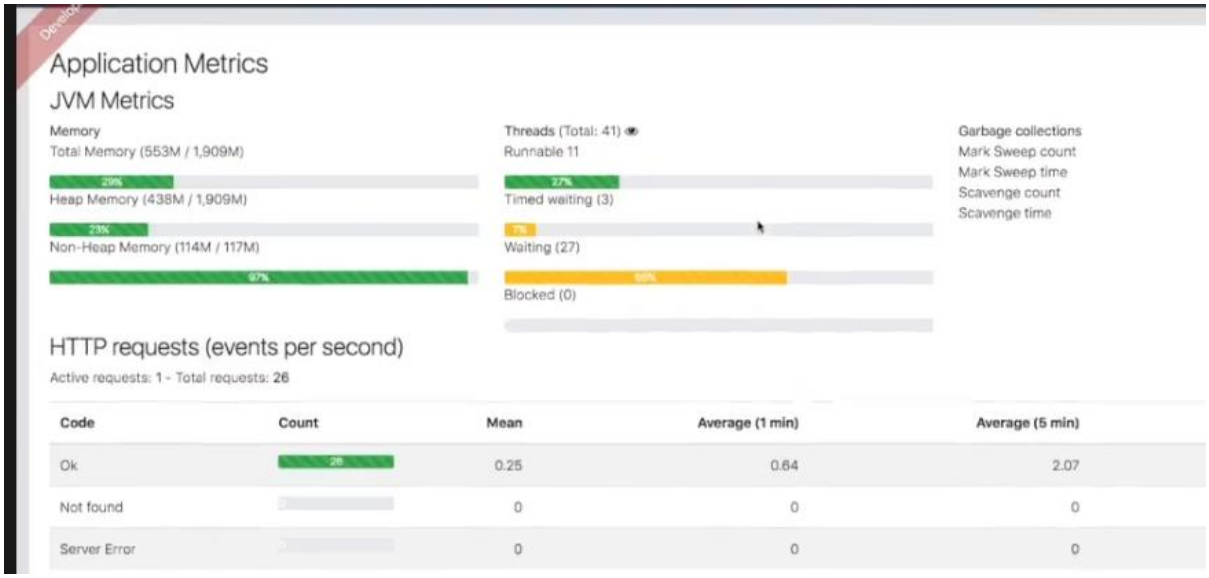


Figura 52. Funcionalidad: Gestión de métricas JHipster.

Funcionalidad: Health

Podemos ver espacio en disco utilizado, la base de datos seleccionada.

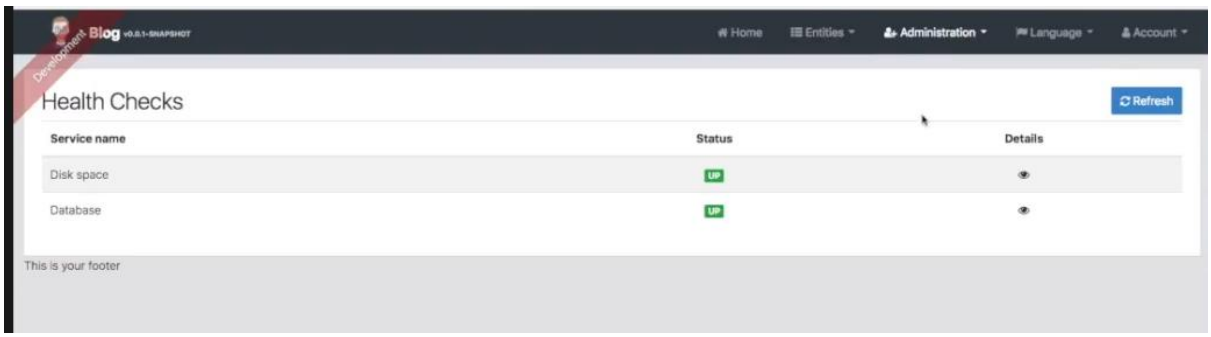


Figura 53. Funcionalidad Healt JHipster.

Funcionalidad: Creación de base de datos

Generar scripts de base de datos para crear SQL para la creación de la base de datos. Puede revisar los scripts generados en la ventana emergente antes de guardarlos como archivos de proyecto. Tenga en cuenta que esos scripts son parte del proyecto, puede encontrarlos en el nodo Main Data Store en el árbol del proyecto.

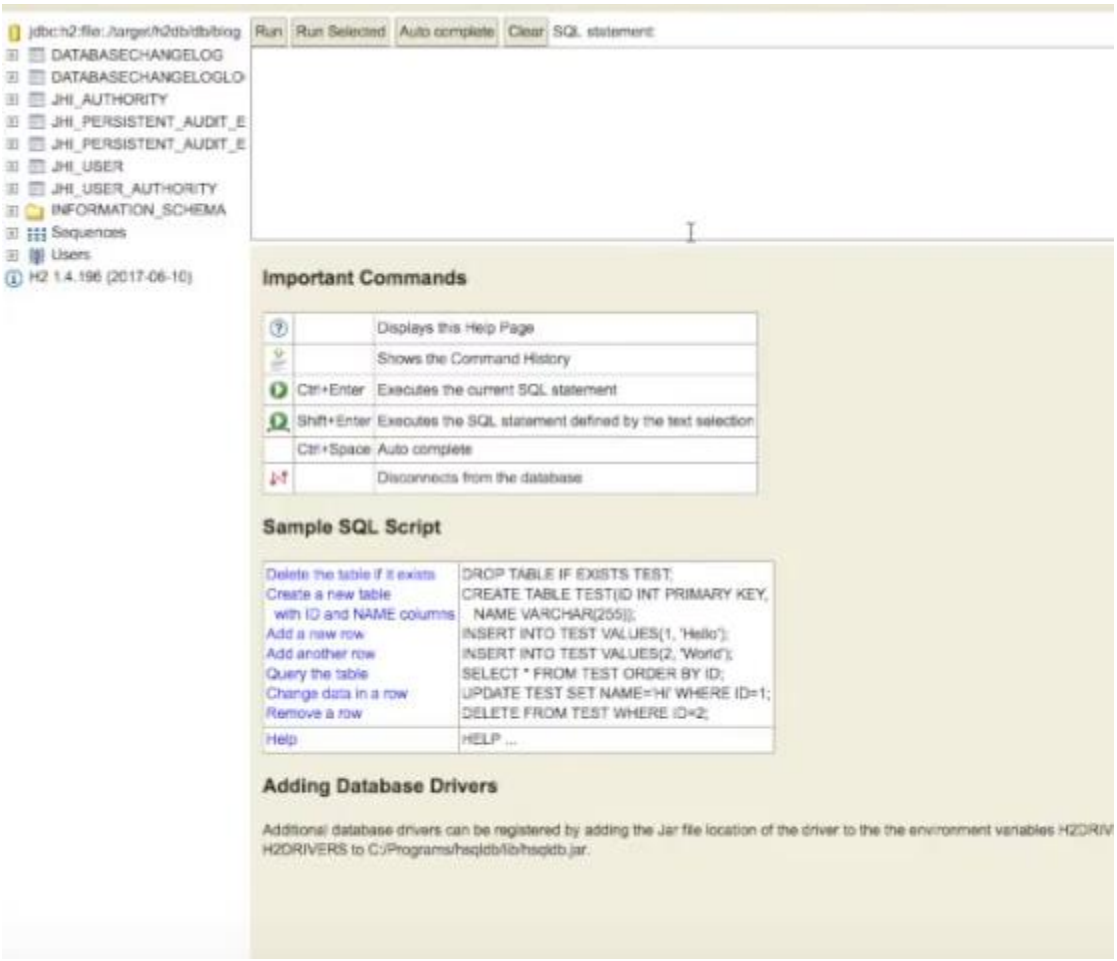


Figura 54. Funcionalidad: Creación de base de datos JHipster.

1.3.2.2. Modelo Lógico

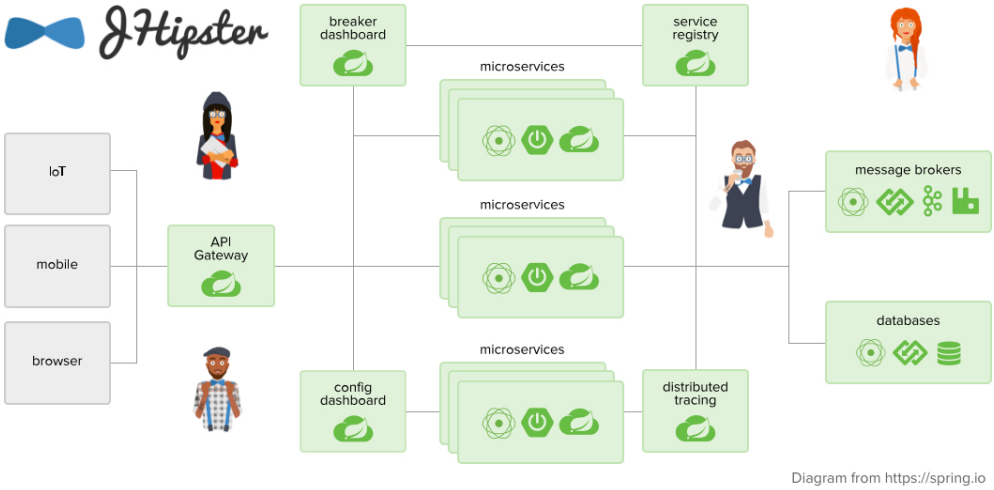


Figura 55. Modelo lógico de JHipster.

1.3.2.3. Problemas y necesidades detectados.

FUNCIONALIDAD	PROBLEMA/NECESIDAD
Definición del dominio	No soporta la definición de entidades y las relaciones entre ellas mediante la práctica manipulación de elementos visuales. En lugar de ello, se definen mediante JDL (JHipster Domain Language), un lenguaje de dominio específico de JHipster. Sí es posible visualizar el dominio de forma gráfica, luego de haberlo definido mediante sintaxis JDL, utilizando JDL-Studio o alguno de los plugins para IDE.
Generación de código fuente para el backend	Se basa en Spring Boot 2.1 para generar código Java 11+, por lo que no da soporte a otros lenguajes de programación.
Regeneración de código	Usar JHipster luego del proyecto inicial genera conflictos. Las clases y los scripts Liquibase se sobrescriben y se debe ser muy cuidadoso al cambiar el modelo JDL inicial.
Acceso a la REST API de la aplicación generada	Las respuestas REST devueltas desde los endpoints no siempre se corresponden con los requerimientos de negocios, usualmente se deberán modificar las respuestas JHipster REST iniciales.

Tabla 3.Problemas y necesidades detectados JHipster.

1.4. ALTOVA UMODEL

1.4.1. Relevamiento general

1.4.1.1. Acerca de la organización

Altova UModel es una potente y sencilla herramienta UML para diseñar software de forma visual. Diseñe modelos de aplicaciones con UML de forma visual y genere código Java, C++, C# o Visual Basic .NET, así como documentación del proyecto. Convierte programas en diagramas UML mediante ingeniería inversa, mejorarlos y terminar el proceso regenerando el código de programa. (Altova. Herramienta de modelado de software UML).

https://www.altova.com/es/umodel

1.4.1.2. Funciones detectadas e Interfaces

- **Generar código fuente a partir de modelos UML**
- **Generar clases del código fuente a partir de diagramas de clases:** UModel crea código Java, C++, C# o Visual Basic .NET a partir de las clases de su modelo UML.
- **Código fuente a partir de diagramas de secuencia:** Con UModel los desarrolladores pueden generar código a partir de diagramas de secuencia para métodos que describen operaciones de clase.
- **Creación de operaciones en clases con referencia:** Y la creación automática de operaciones con la que puede crear una operación nueva en la clase de destino con solo teclear el nombre del mensaje nuevo que se desea.
- **Posibilidad de generar código a partir de diagramas de máquina de estados:** UModel permite generar código totalmente ejecutable a partir de diagramas de máquina de estados para que pueda empezar a probar la lógica reflejada en su diagrama de máquina de estados.
- **Transiciones y operaciones de clases:** Con la opción “Creación automática de operaciones” podrá crear una operación nueva en la clase de destino con solo teclear el nombre del mensaje nuevo.
- **Ingeniería inversa de código fuente:** UModel puede importar archivos de código fuente Java desde proyectos de JBuilder, Eclipse y NetBeans, código fuente C++ de Microsoft Visual Studio, código fuente C# a partir de Visual Studio y Borland C# y archivos de proyecto de Visual Basic .NET.
- **Ingeniería inversa de archivos binarios:** UModel puede importar archivos binarios de Java, C++, C# y Visual Basic .NET.
- **Sincronización automática de modelo y código:** La integración de UModel en entornos IDE permite un proceso de ingeniería de ida y vuelta aún más sofisticado.
- **Esquemas XML en UML:** UModel incluye un tipo especial de diagrama y funciones de ingeniería de código para esquemas XML.
- **Diagramas de bases de datos UML:** Con UModel puede importar tablas de BD relacionales para crear diagramas de base de datos UML, modificar diagramas de tablas de datos ya existentes y generar scripts de cambios SQL para sincronizar la base de datos.
- **Documentación de proyectos en UModel:** La documentación es un factor importante del desarrollo de software. UModel acelera las tareas de documentación generando automáticamente archivos de documentación de proyecto en formato HTML, Microsoft Word o RTF.

1.4.1.3. Tecnología de la información

Altova UModel soporta distintas tecnologías e infraestructuras, en las que están:

- **Bases de datos:** Oracle, MySQL.
- **Lenguajes:** C++, C#, Java, Visual Basic .NET, HTML.

1.4.2. Relevamiento detallado y análisis del sistema

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

Funcionalidad: Generar clases del código fuente a partir de diagramas de clases
UModel crea código Java, C++, C# o Visual Basic .NET a partir de las clases de su modelo UML. Gracias a ello se prescinde de escribir código de infraestructura de nivel inferior y se puede concentrar en la lógica de negocio y en la arquitectura global de su proyecto.

Con las funciones de generación de código fuente de UModel se puede acelerar la fase de implementación de un proyecto y evitar errores no deseados, que suelen producirse al escribir código a mano y que quitan mucho tiempo a la hora de depurarlos.

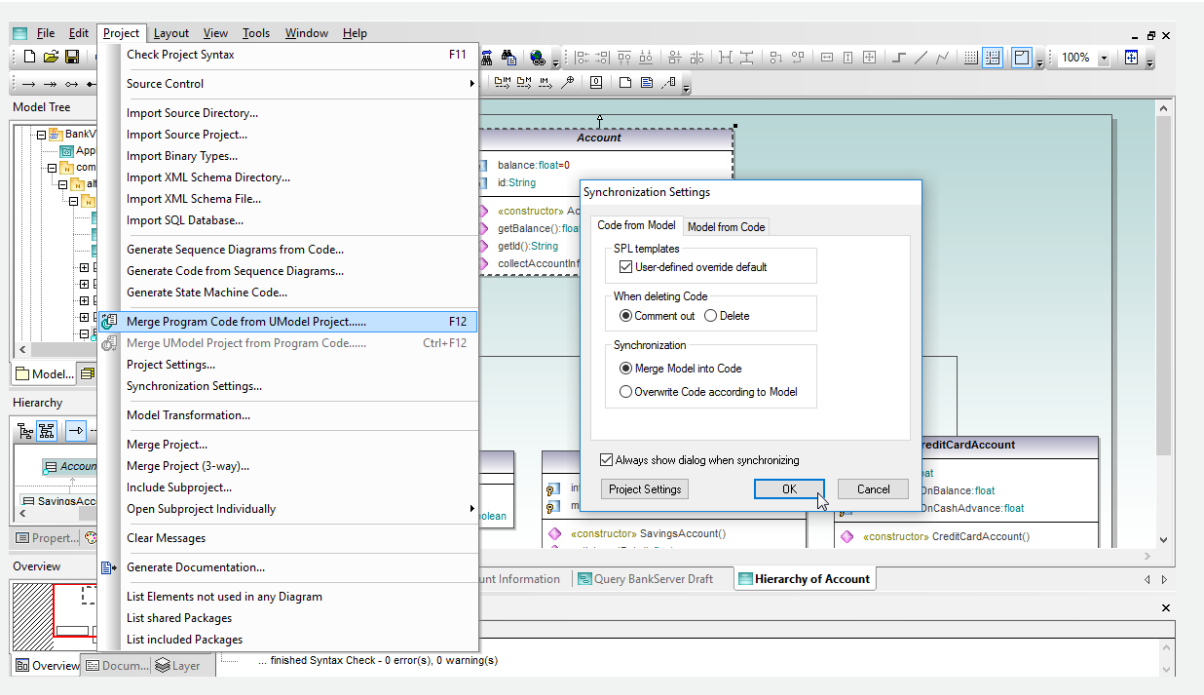


Figura 56. Funcionalidad: Generar clases del código fuente a partir de diagramas de clases Altova.

Funcionalidad: Código fuente a partir de diagramas de secuencia

Con UModel los desarrolladores pueden generar código a partir de diagramas de secuencia para métodos que describen operaciones de clase. El usuario puede insertar cuerpos de código completos

en diagramas de secuencia y crear una aplicación ejecutable entera sin tener que editar el código a mano.

Puede generar código fuente a partir de diagramas de secuencia UML nuevos al aplicar ingeniería directa a un diseño nuevo; puede actualizar código existente al revisar diagramas de secuencia a los que se aplicó ingeniería inversa; y puede aplicar ingeniería de ida y vuelta para sincronizar cambios posteriores en el código fuente o en los diagramas de secuencia de su modelo UML.

UModel permite generar código compatible con los lenguajes Java, C++, C# y Visual Basic a partir de diagramas de secuencia.

Funcionalidad: Creación de operaciones en clases con referencia

Al añadir un mensaje nuevo a una línea de vida que representa una clase, puede asignarle un nombre o elegir una operación de la clase de destino en la ventana Propiedades.

Con “Creación automática de operaciones” puede crear una operación nueva en la clase de destino con solo teclear el nombre del mensaje nuevo

Funcionalidad: Posibilidad de generar código a partir de diagramas de máquina de estados

UModel permite generar código totalmente ejecutable a partir de diagramas de máquina de estados para que pueda empezar a probar la lógica reflejada en su diagrama de máquina de estados.

Puede generar código siguiendo el proceso normal de generación de código de proyecto, que se puede iniciar desde el menú Proyecto o directamente desde el menú contextual del diagrama de máquina de estados.

El menú contextual de UModel incluye también una opción para revisar la sintaxis del diagrama de la máquina de estados y así asegurarse de que el código generado es válido.

En el cuadro de diálogo "Generar código de la máquina de estados" podrá configurar la generación de código e incluso especificar si el código se debe volver a generar automáticamente cuando se genere código de proyecto.

Funcionalidad: Transiciones y operaciones de clases

Cuando añada una transición nueva a un diagrama que esté dentro de una clase o interfaz, puede usar la lista desplegable de la ventana Propiedades para asignar una operación de la clase de destino.

Con la opción “Creación automática de operaciones” podrá crear una operación nueva en la clase de destino con solo teclear el nombre del mensaje nuevo.

Funcionalidad: Ingeniería inversa de código fuente

UModel puede importar archivos de código fuente Java desde proyectos de JBuilder, Eclipse y NetBeans, código fuente C++ de Microsoft Visual Studio, código fuente C# a partir de Visual Studio y Borland C# y archivos de proyecto de Visual Basic .NET.

Puede importar un solo directorio, un árbol de directorio o un proyecto entero. Asimismo, puede elegir entre combinar el código importado con el proyecto de UModel o crear un proyecto nuevo.

Si importa código fuente Java que venga acompañado de JavaDocs, la ventana de documentación de UModel puede rellenarse para cada diagrama UML. También puede importar DocComments de C# y Visual Basic .NET como documentación para su proyecto de modelado.

Las funciones de ingeniería inversa de código Visual Basic .NET de UModel son orientadas a líneas y no hacen distinción entre mayúsculas y minúsculas (es decir no diferencia entre Class1, CLASS1, class1 y ClAsS1), de acuerdo con los requisitos de denominación de Visual Basic .NET, que no son tan rigurosos.

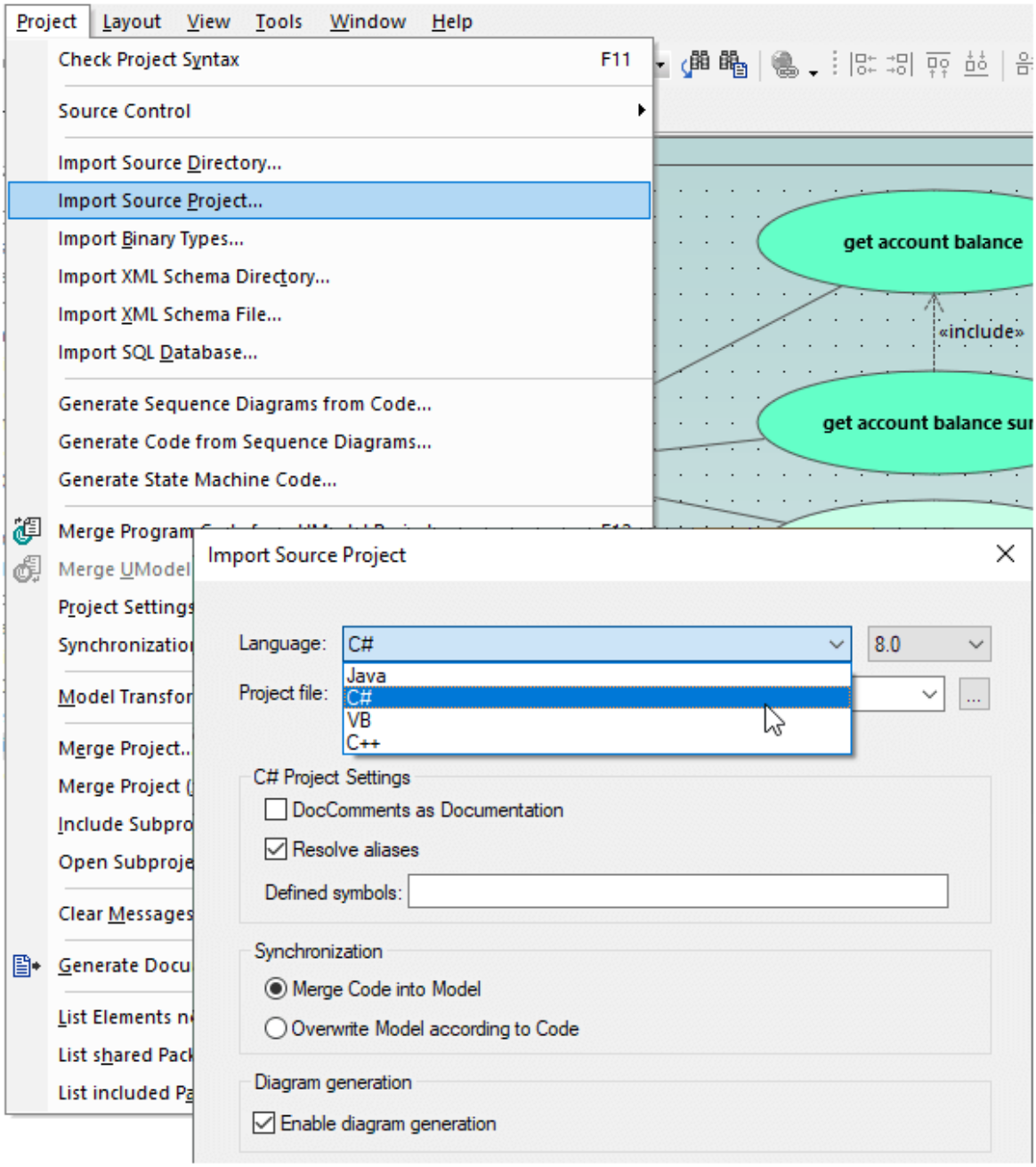


Figura 57. Funcionalidad: Ingeniería inversa de código fuente Altova.

Funcionalidad: Ingeniería inversa de archivos binarios

UModel puede importar archivos binarios de Java, C++, C# y Visual Basic .NET. Para Java, la importación de tipos es compatible con todos los archivos de clases que se ajusten a la especificación de la máquina virtual Java.

Para C#, la importación de tipos es compatible con los ensamblados destinados a .NET Framework, .NET Core y .NET Compact Framework para PocketPC, Smartphone y WindowsCE. Para Visual Basic .NET, se pueden importar archivos DLL y EXE del sistema de archivos o ensamblados del caché global de ensamblados (GAC) o desde una referencia MSVS.NET.

Dependiendo de los requisitos de los binarios que seleccione el cuadro de diálogo ofrece más o menos opciones de configuración. La ayuda integrada de UModel guía en cada parte el proceso de importación de archivos binarios.

Funcionalidad: Sincronización automática de modelo y código

La integración de UModel en entornos IDE permite un proceso de ingeniería de ida y vuelta aún más sofisticado. Cuando se utiliza UModel Enterprise Edition dentro de Microsoft® Visual Studio .NET® o de Eclipse, se puede abrir el proyecto de UModel en una ventana y el código de aplicación asociado en otra ventana en un editor de código fuente.

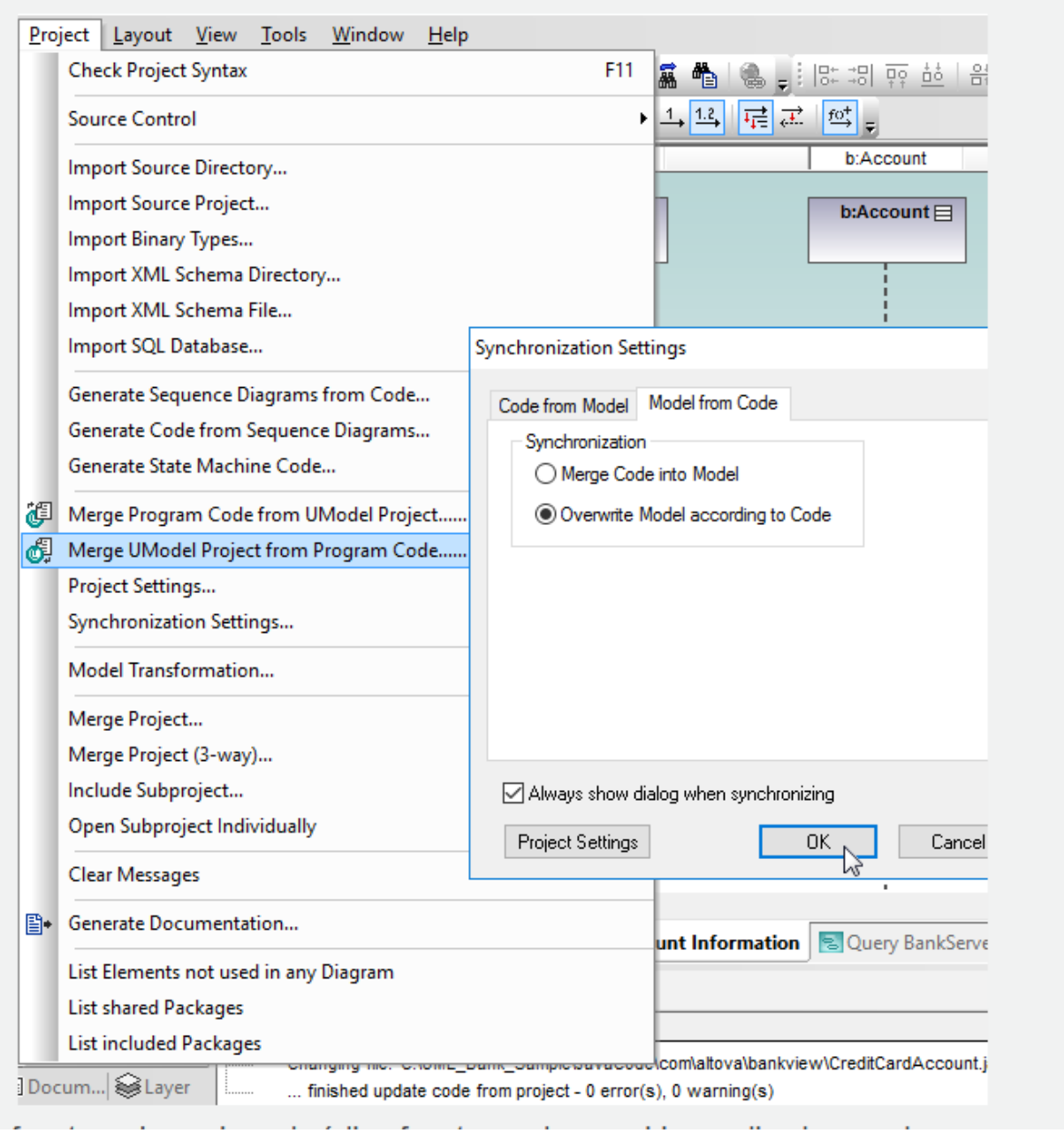


Figura 58. Funcionalidad: Sincronización automática de modelo y código Altova.

Se puede sincronizar el modelo UML con los cambios realizados en el código fuente o sincronizar el código fuente con los cambios realizados en el modelo UML. Gracias a ello se puede comprobar inmediatamente el impacto de las correcciones realizadas, ya sea en el modelo o directamente en el código fuente.

Funcionalidad: Esquemas XML en UML

UModel incluye un tipo especial de diagrama y funciones de ingeniería de código para esquemas XML. El diagrama de esquema XML de UModel presenta esquemas XML en un formato similar al de los diagramas de clases UML, mostrando los elementos globales (elementos element, simpleType y complexType) como clases y los atributos de los elementos en un compartimento de atributos.

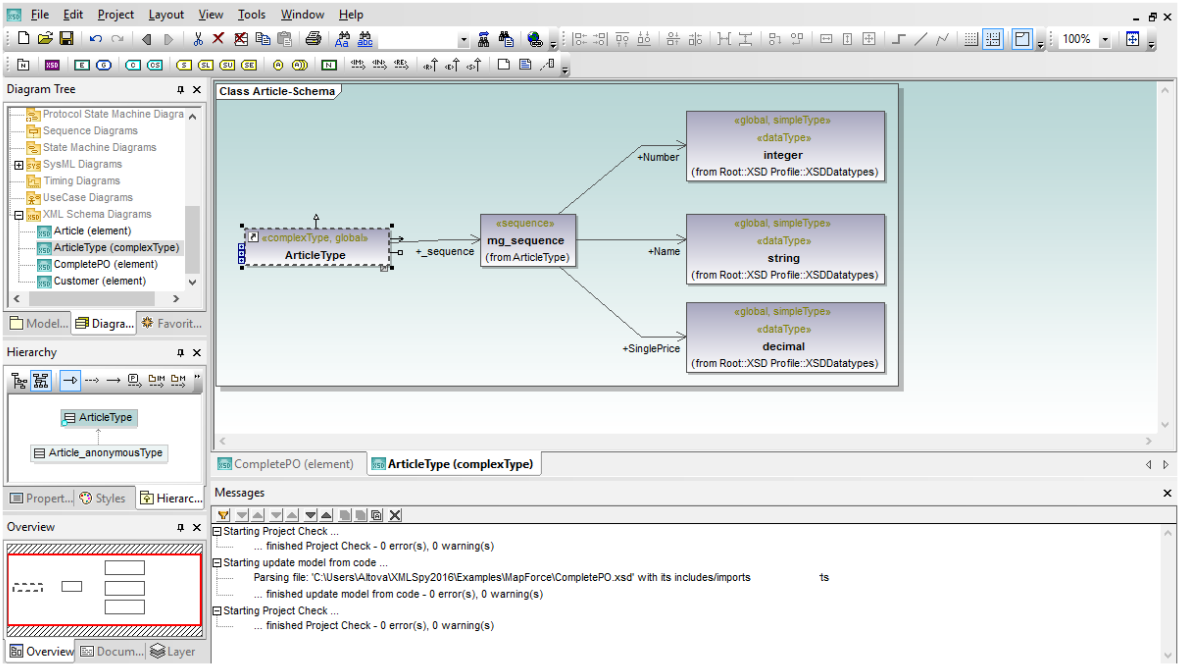


Figura 59. Funcionalidad: Esquemas XML en UML Altova.

Para mostrar los detalles del esquema UModel se sirve de notas UML. El esquema XML se trata como si fuera código fuente de aplicación y se le aplica ingeniería inversa para crear un proyecto de UModel. El archivo de proyecto y los diagramas que contiene son un modelo del esquema XML y no el esquema propiamente dicho.

Puesto que el proyecto de UModel y el esquema XML propiamente dicho son archivos distintos, el modelo UML del esquema constituye un nivel de abstracción entre el diseño del esquema y el archivo XSD. Esto permite a los desarrolladores trabajar en equipos y mejorar los esquemas en equipo con solo modificar el proyecto de UModel, tratando el modelo UML como prototipo. Si se realizan cambios en el modelo del esquema XML, estos mismos cambios se escriben en el archivo de esquema XML (*.xsd) durante la generación de código o sincronización de proyecto.

En UModel también puede aplicar ingeniería de ida y vuelta a los archivos de esquema XML. Si el esquema se modifica fuera de UModel, puede sincronizar el proyecto de UModel y el diagrama XML con los cambios realizados.

Funcionalidad: Diagramas de bases de datos UML

Dado que las aplicaciones de software interactúan con cantidades de datos cada vez mayores, las estructuras y el diseño de bases de datos son una parte fundamental de los proyectos de desarrollo de software. Con UModel puede importar tablas de BD relacionales para crear diagramas de base de datos UML, modificar diagramas de tablas de datos ya existentes y generar scripts de cambios SQL para sincronizar la base de datos. También puede diseñar tablas y relaciones de BD nuevas desde cero y generar scripts CREATE SQL.

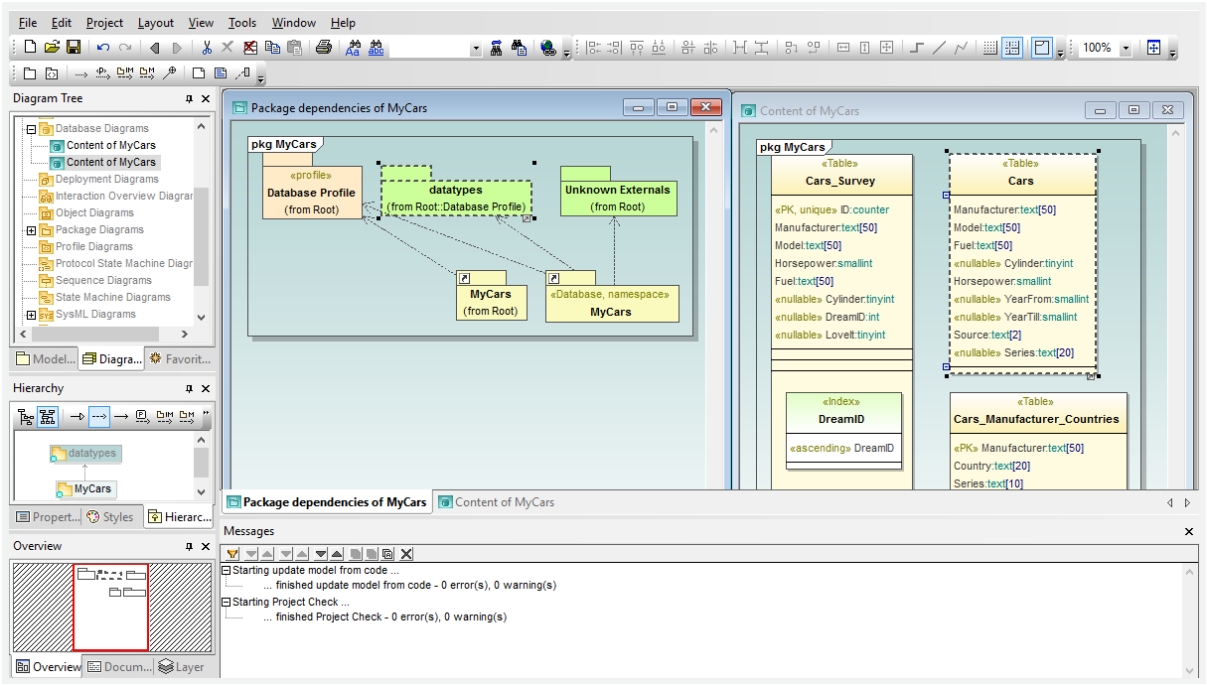


Figura 60. Funcionalidad: Diagramas de bases de datos UML Altova.

También puede importar una BD relacional entera en un solo paso o seleccionar solo las tablas de la BD que más le interesen.

Los diagramas de base de datos de UModel son compatibles con varias bases de datos y son capaces de acomodar dialectos SQL, tipos de datos y otras características propias de cada tipo de base de datos. UModel admite estos elementos de base de datos: esquemas, tablas, vistas, restricciones de comprobación, claves primarias, foráneas y únicas, índices, procedimientos almacenados, funciones, desencadenadores, asociaciones de relación entre BD y relaciones entre BD con atributos.

Funcionalidad: Documentación de proyectos en UModel

La documentación es un factor importante del desarrollo de software. UModel acelera las tareas de documentación generando automáticamente archivos de documentación de proyecto en formato HTML, Microsoft Word o RTF.

Gracias al diseño integrado de documentación de proyectos de UModel puede personalizar la documentación. Seleccione el formato de salida y las opciones de procesamiento de imagen y personalice la documentación según sus requisitos. También puede seleccionar en qué detalle se documenta cada elemento. Por ejemplo, puede incluir diagramas de jerarquías para ilustrar las relaciones entre las clases.

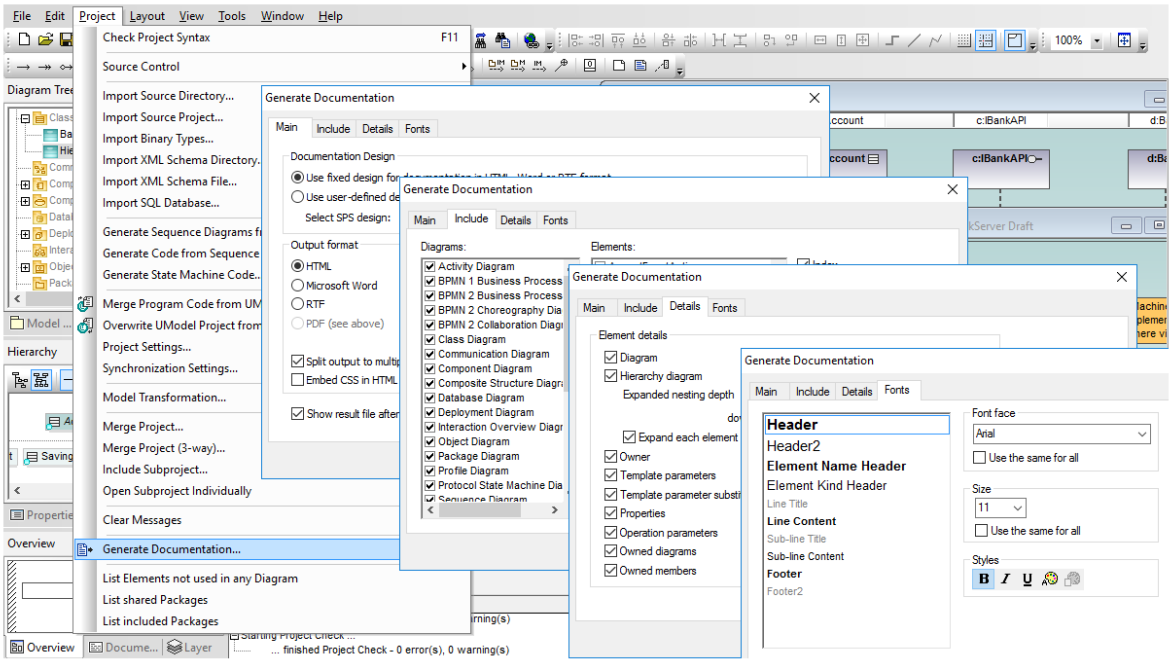


Figura 61. Funcionalidad: Documentación de proyectos en UModel Altova.

UModel añade hipervínculos dentro de la documentación para que sea más fácil navegar por ella. También se añaden automáticamente hipervínculos a los archivos de código fuente generados. Gracias al diseño integrado de documentación de UModel también podrá controlar el aspecto de la documentación del proyecto, especificando el estilo y tamaño de la fuente de cada bloque de texto.

Si necesita personalizar aún más la documentación, puede usar una hoja de estilo SPS y Altova StyleVision creará una copia totalmente personalizada de la documentación de su proyecto UML.

UModel viene con una hoja de estilos de muestra para la documentación. Puede usar esta hoja de estilos tal y como está o editarla en StyleVision de acuerdo con sus requisitos. La hoja de estilos permite incrustar imágenes en su documentación de proyecto UML (como el logotipo de la compañía), encabezados y pies de página y bloques de texto estándar. La documentación basada en hojas de estilos también se puede generar en PDF.

Resumiendo, puede seleccionar una hoja de estilos en UModel y enviar instrucciones de ejecución directamente a StyleVision para crear su propio diseño de forma automatizada.

1.4.2.2. Modelo Lógico

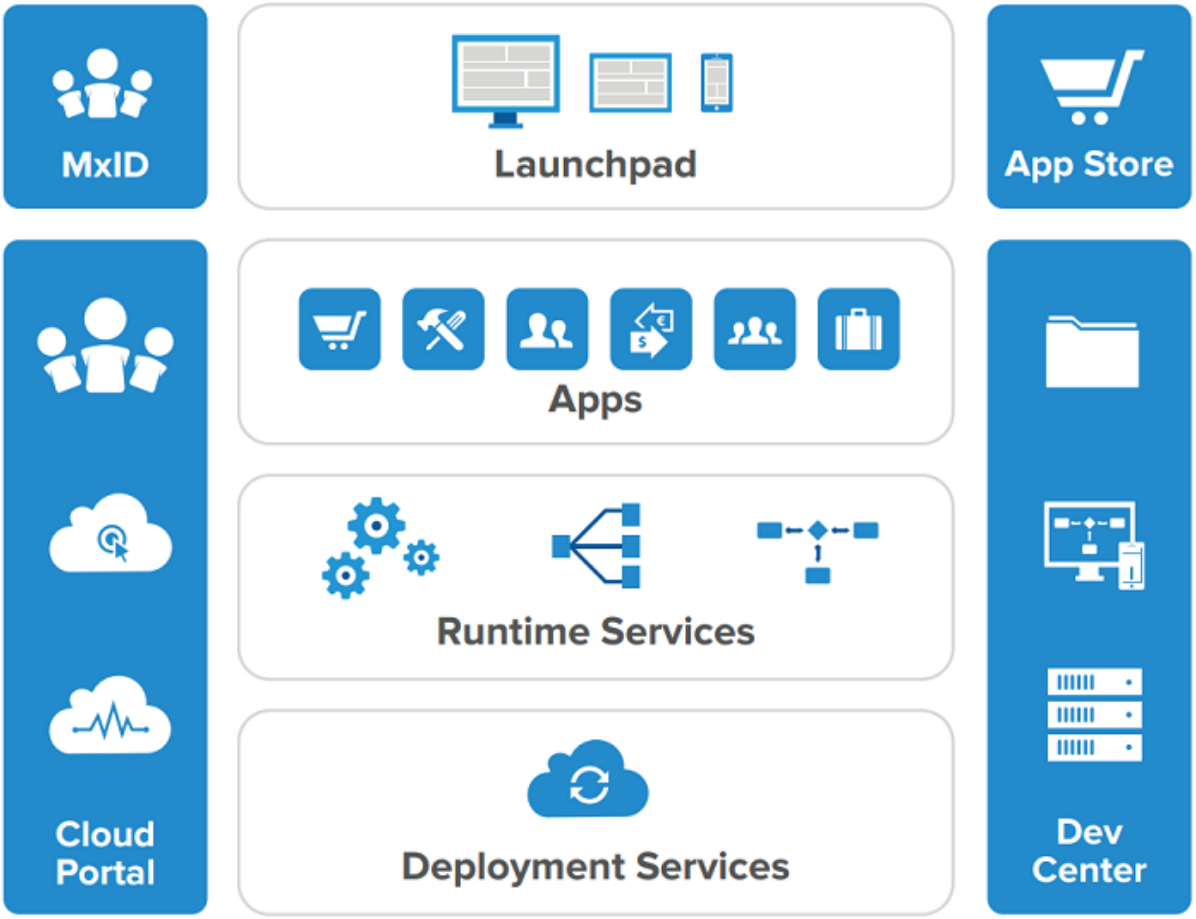


Figura 62. Modelo Lógico en UModel Altova.

1.4.2.3. Problemas y necesidades

En su versión más reciente Enterprise Edition v2010, presenta un entorno de trabajo agradable, sencillo a la vista y muy intuitivo a la hora de realizar los diferentes modelos. Como en la mayoría de las herramientas UML, en este entorno podemos trabajar de dos formas de manera simultánea, la vista de árbol de proyecto o gráficamente en el modelo. Tiene la capacidad de crear todos los diagramas UML descritos, así como también diagramas SysML, o proyectos orientados a la generación de Schemas XML. Además de todos estos puntos positivos, podemos añadir que es una herramienta rápida y eficaz. No es nada pesada para el sistema y solventa los problemas velozmente. El único inconveniente de UModel es que, en su versión actual, no permite definir asociaciones entre estereotipos en un perfil. Estas asociaciones son básicas en la creación de perfiles y, por tanto, esta restricción constituye un impedimento fundamental para su utilización.

FUNCIONALIDAD	PROBLEMA/NECESIDAD
Generar código fuente a partir de modelos UML	Solo genera código fuente (Java, C++, C# o .NET) para el backend y no para el frontend, además está limitado a los lenguajes mencionados.
Generar clases del código fuente a partir de diagramas de clases.	Solo genera clases de los lenguajes (Java, C++, C#) para el backend y no para el frontend, además está limitado a los lenguajes mencionados.
Código fuente a partir de diagramas de secuencia	No presenta problemas
Posibilidad de generar código a partir de diagramas de máquina de estados	No presenta problemas
Creación de operaciones en clases con referencia	No presenta problemas
Transiciones y operaciones de clases	No presenta problemas
Ingeniería inversa de código fuente	Solo puede importar archivos de código fuente Java desde proyectos de JBuilder, Eclipse y NetBeans, código fuente C++ de Microsoft Visual Studio, código fuente C# a partir de Visual Studio y Borland C# y archivos de proyecto de Visual Basic .NET.
Ingeniería inversa de archivos binarios	Solo puede importar archivos binarios de Java, C++, C# y Visual Basic .NET.
Sincronización automática de modelo y código	No presenta problemas
Esquemas XML en UML	No presenta problemas
Diagramas de bases de datos UML	Solo trabaja con base de datos relacionales
Documentación de proyectos en UModel	No presenta problemas

Tabla 4. Problemas y necesidades detectados ALTOVA UMODEL.

1.5. SLINGR

1.5.1. Relevamiento general

1.5.1.1. Sistema Relevado

SLINGR es una plataforma para crear aplicaciones en la nube que pueden integrarse fácilmente con otras soluciones SaaS que existen.
Por otro lado, sabemos que hay muchas soluciones SaaS excelentes que se crearon con un propósito específico en mente, lo que las hace muy eficientes para el problema que resuelven. Intentar reproducir eso en SLINGR no tiene sentido, por lo que nuestro enfoque es proporcionar herramientas para que pueda integrar fácilmente esas aplicaciones en la nube existentes y dedicar sus esfuerzos a crear esos flujos de trabajo personalizados que su empresa necesita.

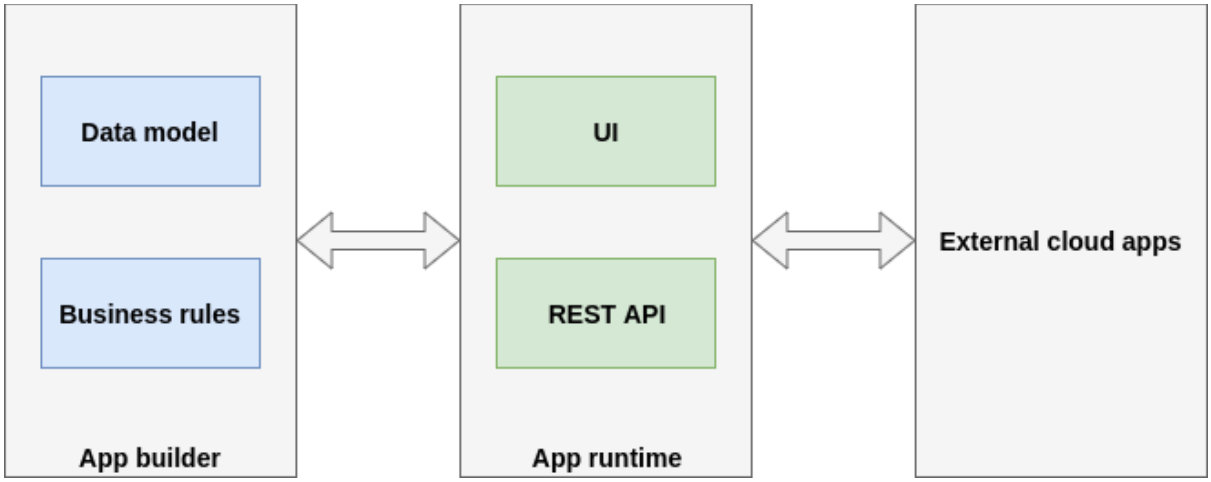


Figura 63. Arquitectura de SLINGR.

https://slingr-stack.github.io/platform/

1.5.1.2. Funciones detectadas e interfaces

- **Crea la aplicación**: Permite crear una aplicación, seleccionar el nombre, el tipo de plan etc.
- **Crear entidad de tareas**: Permite la creación y edición de las entidades de tareas junto a sus etiquetas número, título, estado, descripción y sus respectivos campos.
- **Crea una vista básica**: Configurar de la vista de las entidades donde estarán incluidas las operaciones CRUS y los listados.
- **Agregar acciones y definir el flujo de trabajo**: Permite la creación de procesos o flujos de trabajo, los cuales son un conjunto de acciones encadenadas donde podemos especificar su secuencia.
- **Agregar vista de flujo de trabajo**: Está relacionada con la funcionalidad anterior. Son las vistas de los flujos de trabajo.
- **Prueba de aplicación**: "Pushear la aplicación" indica que subimos a la nube para poder verla y utilizarla.
- **Permisos**: Configuración de los permisos de cada usuario para cada vista como la edición, lectura y escritura de cada entidad.
- **Integraciones**: Permite la integración con aplicaciones exteriores.
- **Supervise su aplicación**: Tablero de control con el historial de logs y operaciones cursadas.

Interfaces:

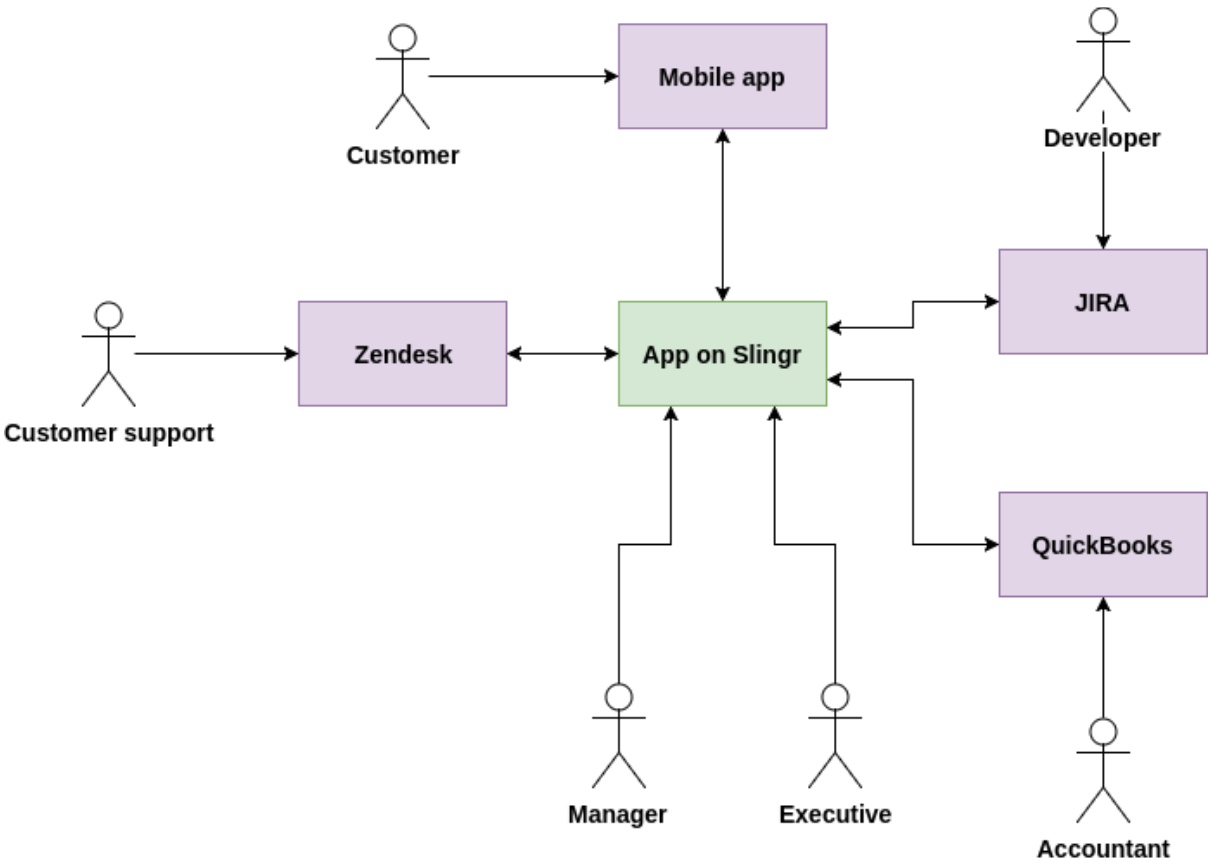


Figura 64. Modelo lógico de SLINGR.

1.5.1.3. Tecnología de Información

El enfoque seguido por SLINGR es que el usuario describe su modelo de datos y reglas comerciales, y esa información se utiliza para crear la aplicación. La idea detrás de esto es que puede concentrarse en resolver problemas comerciales en lugar de problemas técnicos, lo que generalmente es lo que termina haciendo al crear aplicaciones comerciales utilizando herramientas de nivel más bajo como Java o .Net. Nuestro objetivo es aumentar increíblemente la productividad al crear aplicaciones.

1.5.2. Relevamiento detallado y análisis del sistema

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

Funcionalidad: Crear la aplicación
Suponemos que ya tiene una cuenta de desarrollador en slingr.io . Si no es así, crea uno.
Una vez que haya iniciado sesión en SLINGR, debe crear una nueva aplicación:

1. Haga clic en el `Create` botón en la parte superior derecha de la `Apps` sección.
2. Complete la etiqueta y el nombre de la aplicación. El nombre debe ser único en la plataforma.
3. Seleccione `SLINGR Free` como plan para su aplicación.

4. Seleccione la opción para empezar desde cero.
5. Espere hasta que se cree la aplicación. Esto tarda un minuto en completarse.

Una vez que se crea su aplicación, verá sus detalles. De forma predeterminada, usted será el propietario de la aplicación y tendrá permisos de desarrollador para el entorno de desarrollo, que se crea de forma predeterminada.
En los detalles del entorno podrás navegar a los diferentes componentes de la aplicación:

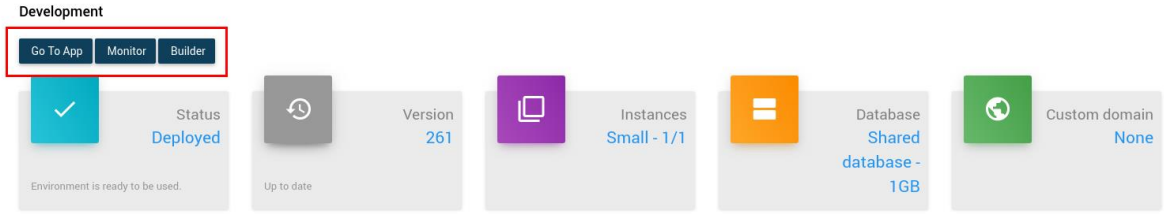


Figura 65. Componentes de la aplicación generada por SLINGR.

- Go To App: lo llevará a la aplicación en ejecución. En este momento no tiene mucho sentido porque aún no hemos hecho nada, así que está vacío.
- Monitor: desde allí podemos ver el estado de la aplicación, trabajos en segundo plano, registros, etc. Si necesitas saber qué está pasando, este es el lugar para mirar.
- Builder: aquí es donde desarrollaremos nuestra aplicación, el creador de aplicaciones.

Como nuestra nueva aplicación está vacía, lo primero que debemos hacer es ir al creador de aplicaciones haciendo clic en el Builder Botón. Eso abrirá automáticamente el constructor en una nueva pestaña para que podamos seguir con el tutorial.

Funcionalidad: Crear entidad de tareas

Una vez que ingrese al creador de aplicaciones, es hora de comenzar a desarrollar la aplicación. Lo primero que haremos será crear la tasks entidad, que contendrá la definición de tareas en nuestra aplicación:

1. En el menú principal, vaya a Model > Entities.
2. Haga clic en el Create Botón en la parte superior derecha de la lista.
3. Complete la etiqueta con Tasks y el nombre se completará automáticamente con tasks cuando se mueva a otro campo.

4. Haga clic en Create. Esto creará una entidad vacía y debería ver esto en su menú principal:

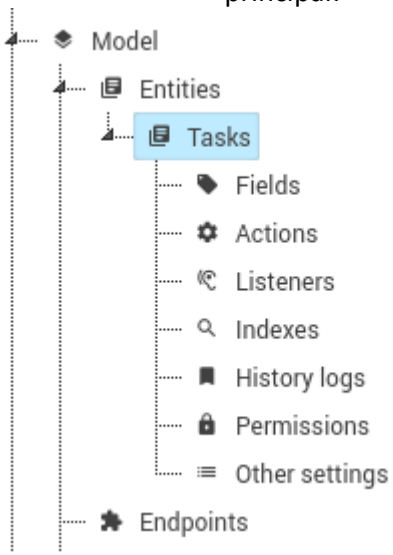


Figura 66. Creación de entidad Tasks en el menú principal del modelo de SLINGR.

Verá que casi todos los elementos requerirán una etiqueta y un nombre. La etiqueta es un nombre amigable para los humanos, mientras que el nombre es un identificador interno que probablemente usará en sus scripts para hacer referencia al elemento. Por lo general, el nombre no permite espacios ni caracteres especiales y debe ser único en su contexto. Sugerimos mantener los nombres lo más estables posible para evitar tener que hacer cambios en sus scripts. Ahora es el momento de crear los campos dentro de la tasks entidad. Crearemos los siguientes campos:

Etiqueta	Nombre	Tipo	Requerido	Valor por defecto	Notas
Número	número	Autoincremento	-	-	Este es un número que se incrementará con cada nueva tarea creada.
Título	título	Texto	sí	-	El título de la tarea.
Estado	estado	Elección	sí	Que hacer	Este es el estado de la tarea. Los valores posibles serán To do, In progress, Done Archived

Descripción	descripción	HTML	No	-	Una descripción más larga de la tarea.

Tabla 5. Campos Taks Entidad.

Para crear estos campos, haga lo siguiente:

1. Seleccione `Model > Entities > Tasks > Fields` en el árbol de la izquierda.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista.
3. Allí deberás completar la información básica: etiqueta, nombre, tipo y multiplicidad. Dependiendo del tipo se pueden ver algunas opciones adicionales en tiempo de creación, que es el caso para el `Status` campo en el que tendrá que añadir los tres estados posibles: `To do`, `In progress`, `Done`y `Archived`. Deje el nombre sugerido para todos estos valores posibles:

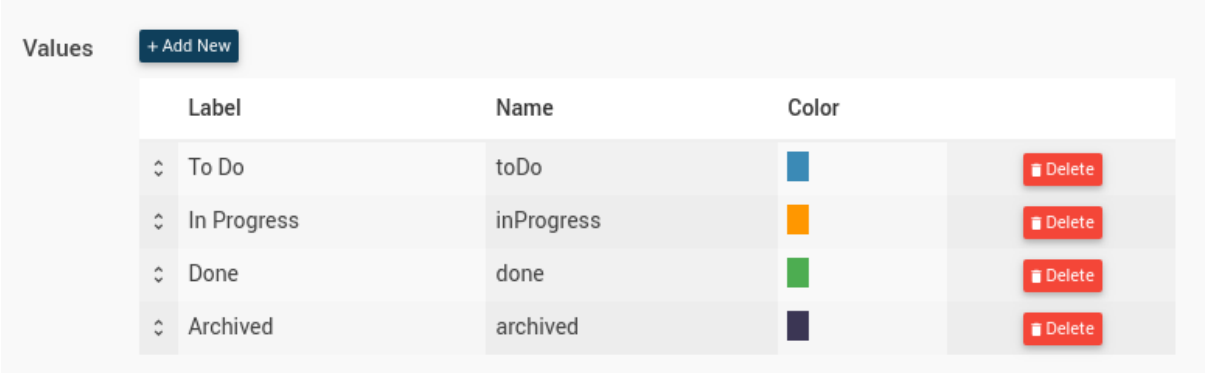


Figura 67. Creación de campos dentro de la entidad Tasks en SLINGR.

4. Al hacer clic en `Save and edit` se creará el campo y se abrirá la vista de detalles para ese campo, donde podrá cambiar otras configuraciones, como el valor predeterminado y la marca requerida. En nuestra aplicación de muestra, debemos ocuparnos de las siguientes cosas para estos campos:
   - `Title`: el ajuste `Required` debería ser `Always`.
   - `Choice`: el ajuste `Required` debe ser `Always`. Además, `Default value` debe poner `Value` y luego seleccionar el valor `To do` del menú desplegable.
5. Una vez que termine de editar todas las configuraciones (como el valor predeterminado o la bandera requerida), haga clic en `Save` y volverá a la lista para agregar más campos repitiendo los pasos anteriores.

   Deberías terminar con una estructura como esta:

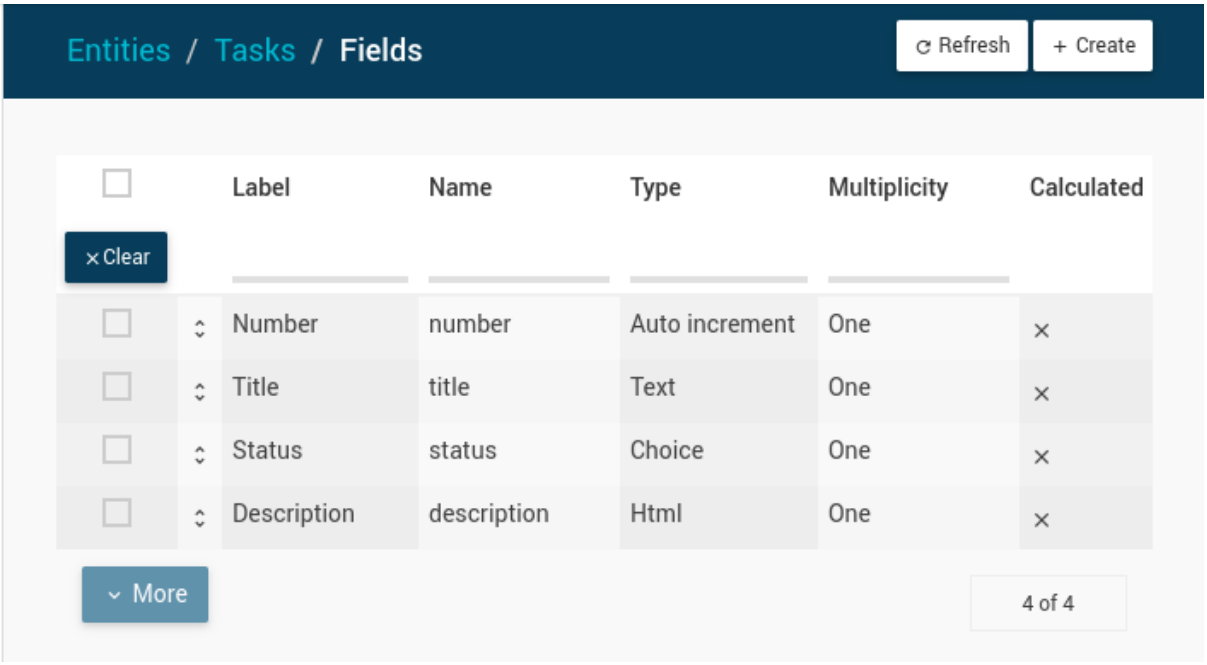


Figura 68. Estructura de la configuración inicial de la entidad Tasks en SLINGR.

El paso final para esta configuración inicial de la entidad es definir la etiqueta de registro. La etiqueta de registro se calcula para cada registro y es algo que identificará su registro. No es necesario que sea único, pero debe ser amigable para los humanos.

Para configurar la etiqueta de registro, vuelva al `Model > Entities > Tasks` menú principal y haga lo siguiente:

1. Configure la opción `Script` para `Record label`.
2. En el editor de código a continuación, coloque este script: `return record.field('number').val() + '. ' + record.field('title').val();`
3. Debería ver algo como esto:

Figura 69. Configuración de la etiqueta de registro en SLINGR.

4. Haga clic en `Apply` para guardar los cambios.

Funcionalidad: Crear una vista básica

Ahora que tenemos la entidad, el siguiente paso es crear una vista para que podamos ver algo en nuestra aplicación. Para eso crearemos una vista de cuadrícula simple que permitirá listar, crear, editar y eliminar tareas.
Para crear la vista de cuadrícula, siga estos pasos:

1. En el creador de aplicaciones, en el menú principal, seleccione `Model > Entities > Tasks > Views`.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista y seleccione la opción `Add grid view`.
3. Complete el formulario con los siguientes valores:
    - `Label`: Todas las tareas
    - `Name`: todas las tareas
    - `Columns`: Número, Título, Estado
4. Haga clic en `Create` para completar la creación de la vista de cuadrícula.

Una vez que tengamos la vista debemos indicar cómo navegar hacia ella. Usaremos el menú principal de la aplicación para eso, así que en el menú principal del constructor ve `User interface > Navigation > Main menu` y sigue estos pasos:

1. Haga clic en `Add new menu entry` en la parte superior derecha de la lista y luego haga clic en la opción `Add view entry`.
2. Complete el formulario con los siguientes valores:
    - `View`: Todas las tareas
    - `Label`: Todas las tareas
    - `Icon`: ícono de lista de vista (puede filtrar íconos escribiendo el nombre)
3. Finalmente haga clic en `Create`.

Funcionalidad: Pruebe su aplicación

¡Ahora tenemos lo básico para ver algo en nuestra aplicación! Es hora de trasladar nuestro trabajo del creador de aplicaciones al tiempo de ejecución de la aplicación. Este proceso se llama `Push changes` y se puede acceder desde el menú secundario en el generador de aplicaciones:

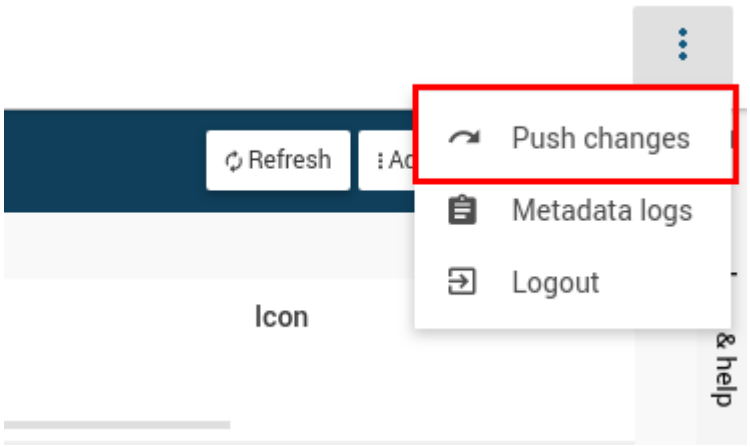


Figura 70. Realizar prueba de ejecución de la aplicación generada en SLINGR.

Cuando haga clic en él, verá un modal que muestra los cambios que se enviarán al tiempo de ejecución y le permite escribir una descripción de los cambios que hizo, que es opcional.
Haga clic en el botón `Push changes`. Una vez que se hayan enviado todos los cambios, verá un mensaje de éxito y estará listo para probar su aplicación. Para eso, regrese a SLINGR y vea la configuración del entorno:

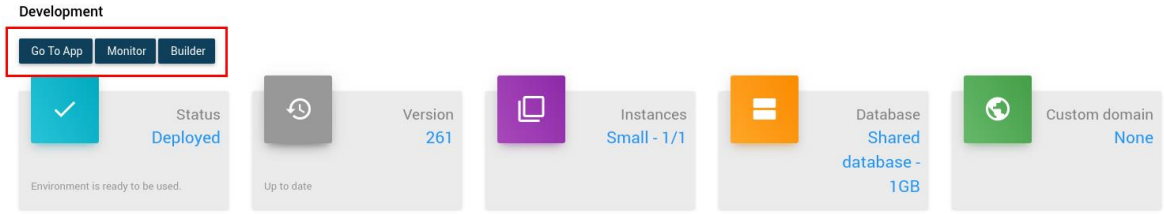


Figura 71. Componentes de la aplicación generada por SLINGR.

Allí haga clic en `Go To App`, que lo llevará a su aplicación. Debería verse similar a esto:

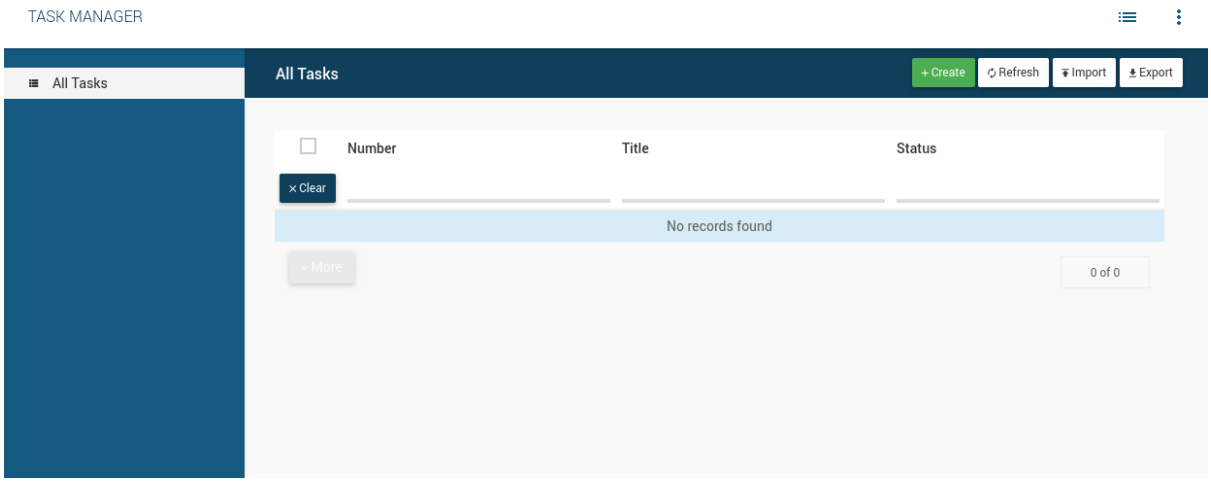


Figura 72. Interfaz de administración de tareas de la aplicación generada en SLINGR

Allí podrá crear nuevas tareas, ver y editar las existentes, así como eliminarlas. También puede filtrar tareas usando los encabezados de lista, exportar e importar registros. ¡Todo esto se ha creado automáticamente a partir de la definición de su aplicación!

Funcionalidad: Agregar acciones y definir el flujo de trabajo

Ya tienes las operaciones básicas para crear, editar y eliminar tareas. Sin embargo, sería mejor tener un flujo de trabajo personalizado que aplique algunas reglas. Por ejemplo, una tarea solo se puede mover `In progress` si está en el estado `To do`.
Para crear el flujo de trabajo haremos lo siguiente:

- Cree acciones para mover el problema a través de los diferentes estados.
- No permite modificar manualmente el estado.

Empecemos por la creación de las acciones. Para eso, deberá ir Model > Entities > Tasks > Actions al menú principal del creador de aplicaciones y hacer lo siguiente:

1. Haga clic en el Create botón en la parte superior derecha de la lista.
2. Para Label poner Start worky para Name dejar el valor predeterminado (startWork).
3. En Preconditions indicará en qué casos se puede ejecutar la acción. Para la acción, startWork la condición previa es que el campo Status debe ser To do. Esto se puede indicar con una expresión, así que seleccione la opción Expression para Preconditions. Luego, deberá configurar la siguiente expresión haciendo clic en Add new rule y configurarla de esta manera:

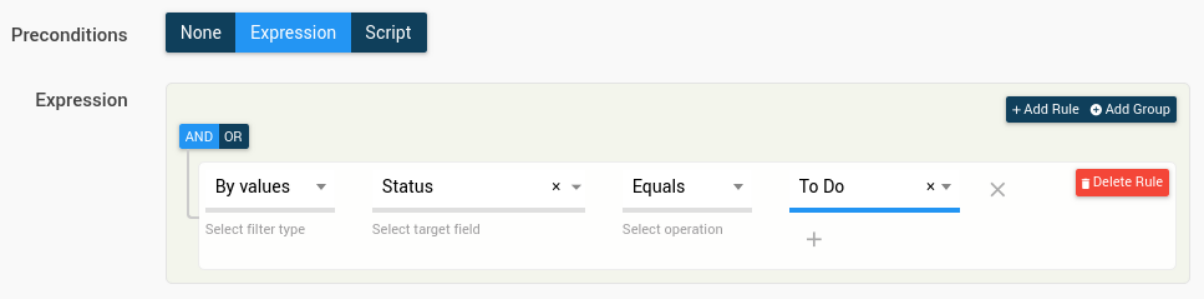


Figura 73. Definición de expresiones.

4. En el campo, Action script agregue el siguiente script al cuerpo de la función:
```
record.lock(function(record) {
        record.field('status').val('inProgress');
        sys.data.save(record);
});
```
5. Deje los otros campos como están y haga clic en Create.

Entonces necesitamos crear las siguientes acciones usando los mismos pasos que arriba:

Etiqueta	Nombre	Condición previa	Guión de acción

Completo	completo	Status igual a In progress	record.lock(function(record) {
			record.field('status').val('done');
			sys.data.save(record);
			});

Archivo	archivo	Status NO es igual a Archived	record.lock(function(record) {
			record.field('status').val('archived');
			sys.data.save(record);
			});

Detén el trabajo	Detén el trabajo	Status igual a In progress	record.lock(function(record) {
			record.field('status').val('toDo');
			sys.data.save(record);

			});

Reabrir	reabrir	Status es igual a Done o Archived	record.lock(**function**(record) {
			  record.field('status').val('toDo');
			  sys.data.save(record);
			});

Tabla 6. Acciones de Campos.

¡Bien, ahora tienes todas las acciones para administrar el flujo de trabajo! Sin embargo, hay un problema: cualquiera puede cambiar el campo `status` simplemente editando la tarea. Esto no es lo que queremos; en su lugar, queremos obligar a las personas a seguir el flujo de trabajo que definimos.
Para evitar que las personas cambien el estado de forma no válida, lo que haremos es que sea de solo lectura:

1. Vaya a `Model > Entities > Tasks > Fields > Status` en el menú principal.
2. Seleccione la pestaña `Display options`.
3. Para la opción `Read only` seleccione `Always`.
4. Guarde los cambios haciendo clic en `Apply`.

Para simplificarlo, acabamos de hacer que el campo sea de solo lectura. Sin embargo, la forma correcta de hacerlo es eliminar los permisos para cambiar ese campo, que se aplicará a nivel de API y no solo a la interfaz de usuario. Eso lo veremos más tarde.
Finalmente, para facilitar el acceso a las acciones desde la vista de cuadrícula, habilitaremos las acciones allí:

1. Vaya a `Model > Entities > Tasks > Views > All tasks` en el menú principal (también puede encontrar el mismo elemento debajo `UserRecordinterface > Grid views > All tasks`).
2. En el interior `List settings`, establezca la opción `Show actions` a `All`, y establece el indicador `Show actions column`.
3. Guarde los cambios haciendo clic en `Apply`.

¡Ahora estamos listos para pushear cambios y ver nuestro flujo de trabajo en su lugar! Así que continúe e introduzca los cambios utilizando la opción `Push changes` en el menú secundario del creador de aplicaciones.
Si ahora va a su aplicación, verá que las acciones están disponibles en la vista de cuadrícula, así como en la vista de solo lectura de las tareas:

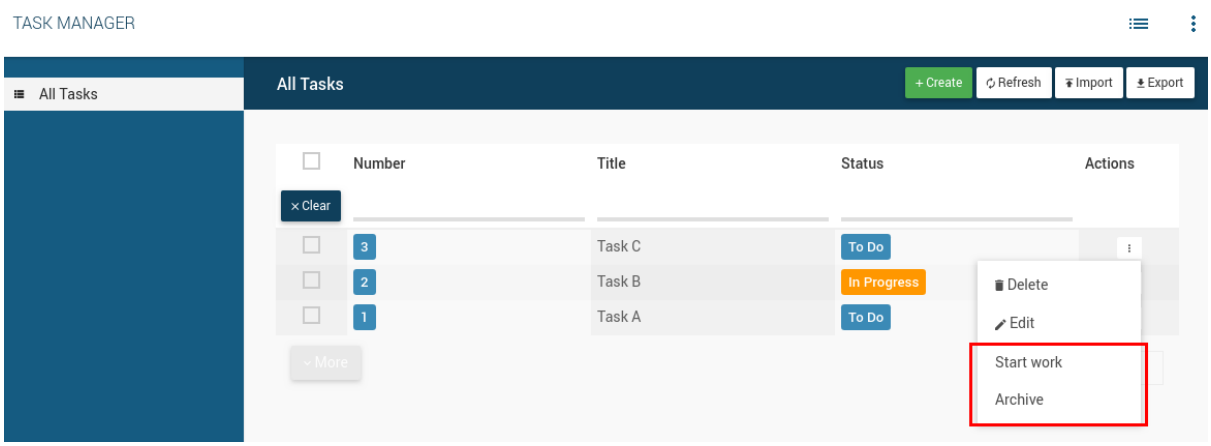


Figura 74. Acciones individuales disponibles en la aplicación generada por SLINGR.

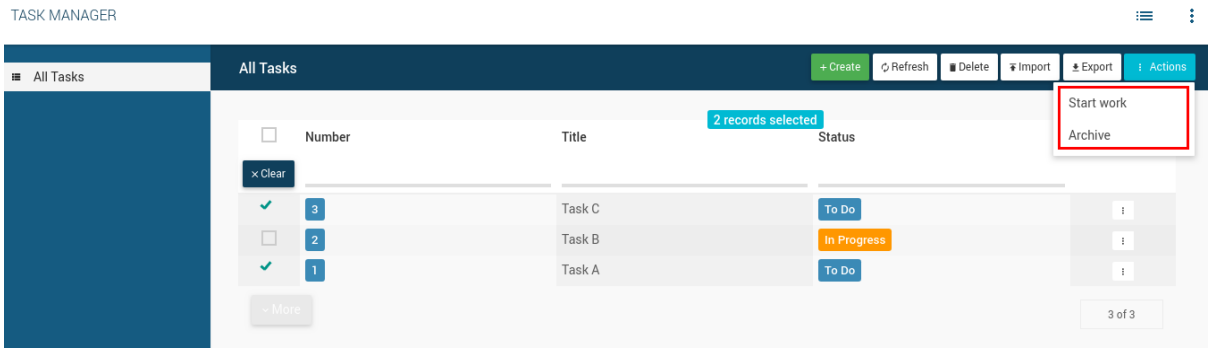


Figura 75. Acciones en lote disponibles en la aplicación generada por SLINGR.

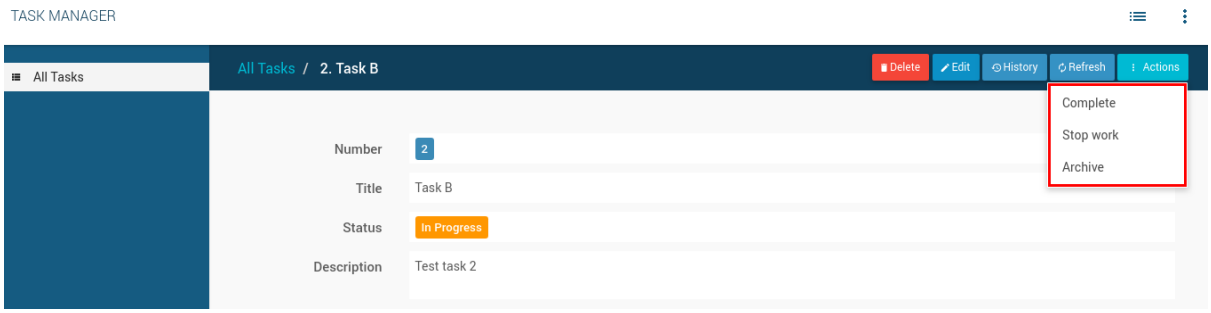


Figura 76. Edición de propiedades de tareas creadas en la aplicación generada por SLINGR.

Funcionalidad: Agregar vista de flujo de trabajo

Ya tenemos nuestro modelo de datos, así como nuestro flujo de trabajo definido. Sin embargo, podemos hacer que nuestra aplicación sea mucho más elegante utilizando vistas de flujo de trabajo. Para darte una idea de lo que estamos hablando, así es como se verá:

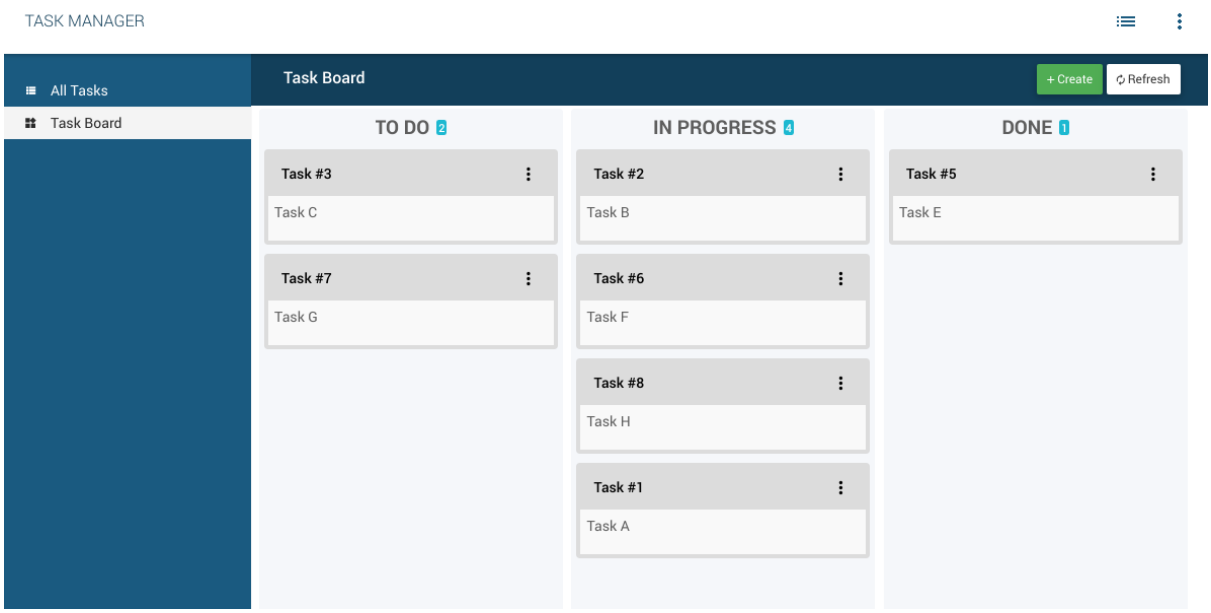


Figura 77. Vista del tablero de tareas en forma de kanban de la aplicación generada por SLINGR.

Ahora creemos la vista del flujo de trabajo:

1. Vaya a `Model > Entities > Tasks > Views` en el menú principal.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista y seleccione `Add workflow view`.
3. Complete el formulario con:
   - `Label`: Tablero de tareas
   - `Name`: tablero de tareas
4. Haga clic en `Create and edit`.

Una vez que haya creado la vista de flujo de trabajo, verá sus detalles de configuración. Lo primero que debe hacer es configurar la configuración de la tarjeta:

1. En la configuración, `Header` seleccione `Script` y coloque el siguiente código en el cuerpo de la función:
   ```
   return 'Task #' + record.field('number').val();
   ```
2. En el ajuste `Summary`dejar `Field`seleccionado y en `Summary field` seleccionar `Title`.
3. Guarde los cambios haciendo clic en `Apply`.

Ahora debemos definir las columnas, pero antes de hacer eso agregaremos un nuevo campo a la entidad que será necesario para permitir la clasificación de registros:

1. Vaya a `Model > Entities > Tasks > Fields` en el menú principal.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista.
3. Complete el formulario con:
   - `Label`: Rango
   - `Name`: rango
   - `Type`: Rango
4. Haga clic en `Create and edit` para guardarlo.

5. Vaya a la pestaña Display options.
6. Establezca la opción Visible en Never. Esto es para ocultar este campo ya que no queremos mostrárselo a nuestros usuarios, solo lo usaremos internamente para mantener el rango de tareas.
7. Haga clic en Save para guardar los cambios.

Ahora estamos listos para crear las columnas en la vista del flujo de trabajo:

1. Vaya a Model > Entities > Tasks > Views > Task board > Columns en el menú principal.
2. Haga clic en el Create botón en la parte superior derecha de la lista.
3. Complete el de con:
   - Label: Que hacer
   - Filters: Statusigual aTo do
   - Coloque la bandera Allow to rank records y seleccione el campo Rank en Rank field
4. Haga clic en Create para guardar la columna.

Repita el mismo proceso para crear estas dos columnas adicionales (¡no olvide configurar el rango de registros!):

Etiqueta	Filtros
En curso	Status igual a In progress
Hecho	Status igual a Done

Tabla 7: Tabla filtros, Slingr.

Una vez que tenga las columnas necesitamos definir las transiciones que permitirán mover una tarjeta de una columna a otra:

1. Vaya a Model > Entities > Tasks > Views > Task board > Transitions en el menú principal.
2. Haga clic en el Create botón en la parte superior derecha de la lista.
3. Complete el formulario con:
   - Label: Empezar a trabajar
   - Source column: Qué hacer
   - Target column: En curso
   - Action: Iniciar trabajo (tasksStartWork)
4. Haga clic en Create ara guardar la transición.

Repita el mismo proceso para crear estas otras transiciones:

Etiqueta	Columna fuente	Columna de destino	Acción
Completo	En curso	Hecho	Complete (tasksComplete)
Detén el trabajo	En curso	Qué hacer	Detener el trabajo (tasksStopWork)

Reabrir de hecho	Hecho	Qué hacer	Reabrir (tasksReopen)

Tabla 8. Creación de Transiciones.

¡Estamos casi allí! Haremos algunas mejoras:

1. Vaya a Model > Entities > Tasks > Views > Task board > CRUD actions > Tasks en el menú principal.
2. Para Create, Ready Update coloque la bandera Open in modal y haga clic en Apply. Esto permitirá abrir tareas en un modal en las vistas de este flujo de trabajo.
3. Ahora ve a Model > Entities > Tasks > Views > Task board en el menú principal.
4. En el interior Cards settings, en la configuración Show actions seleccione Some.
5. Luego Available actions haga clic en Addy seleccione la acción Archive (tasksArchive). Esto es para poder archivar tareas porque no creamos una columna para el Archived estado para mantener limpia esta vista.

Finalmente agreguemos la nueva vista a la navegación:

1. Ir a UserRecordinterface > Main menu.
2. Haga clic en el Add new menu entry botón en la parte superior derecha de la lista y seleccione Add view entry.
3. Complete el formulario con:
   - View: Tablero de tareas
   - Label: Tablero de tareas
   - Icon: ¡selecciona el que más te guste!
4. Haga clic en Create para guardar la entrada del menú.
5. Mueva la nueva entrada del menú a la parte superior. Puede hacerlo usando arrastrar y soltar con las flechas al lado de la casilla de verificación en la lista de entradas del menú:

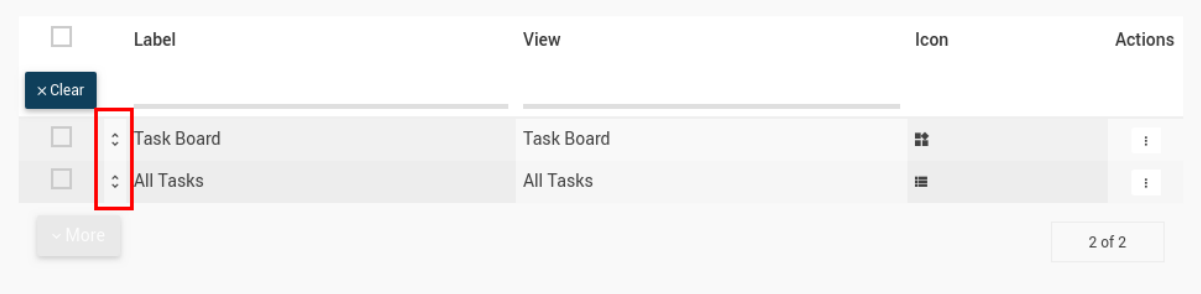


Figura 78. Agregar la nueva lista a la navegación.

¡Bien, hemos terminado! Ahora hagamos cambios para ver los resultados.

Funcionalidad: Probar la aplicación

Ahora tenemos la aplicación resultante:

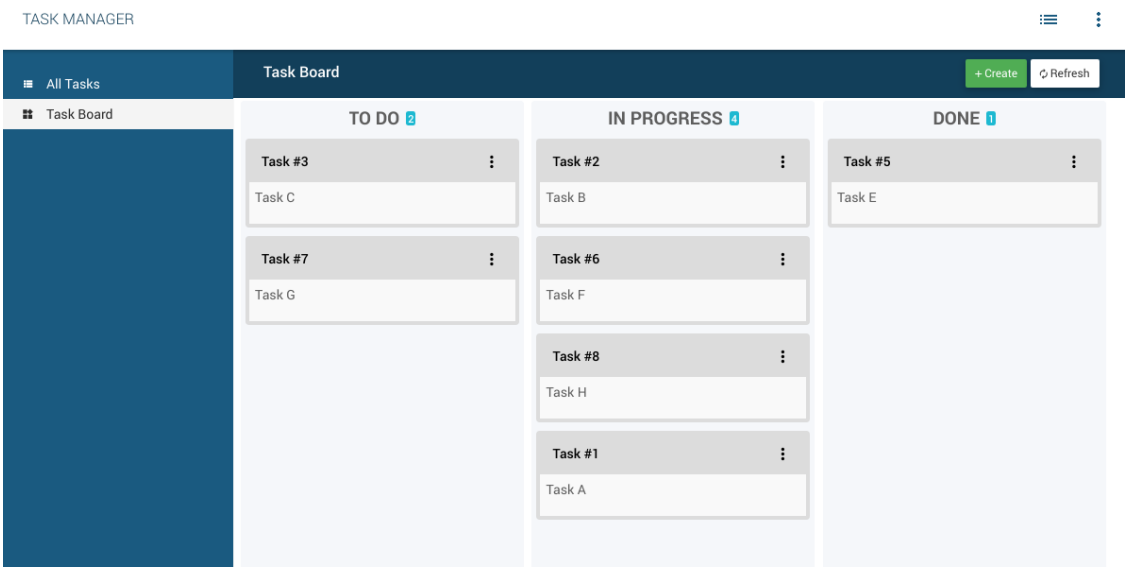


Figura 79. Probar la aplicación.

Estas son algunas de las cosas que puede hacer desde allí:

- Cree nuevas tareas en una ventana emergente:

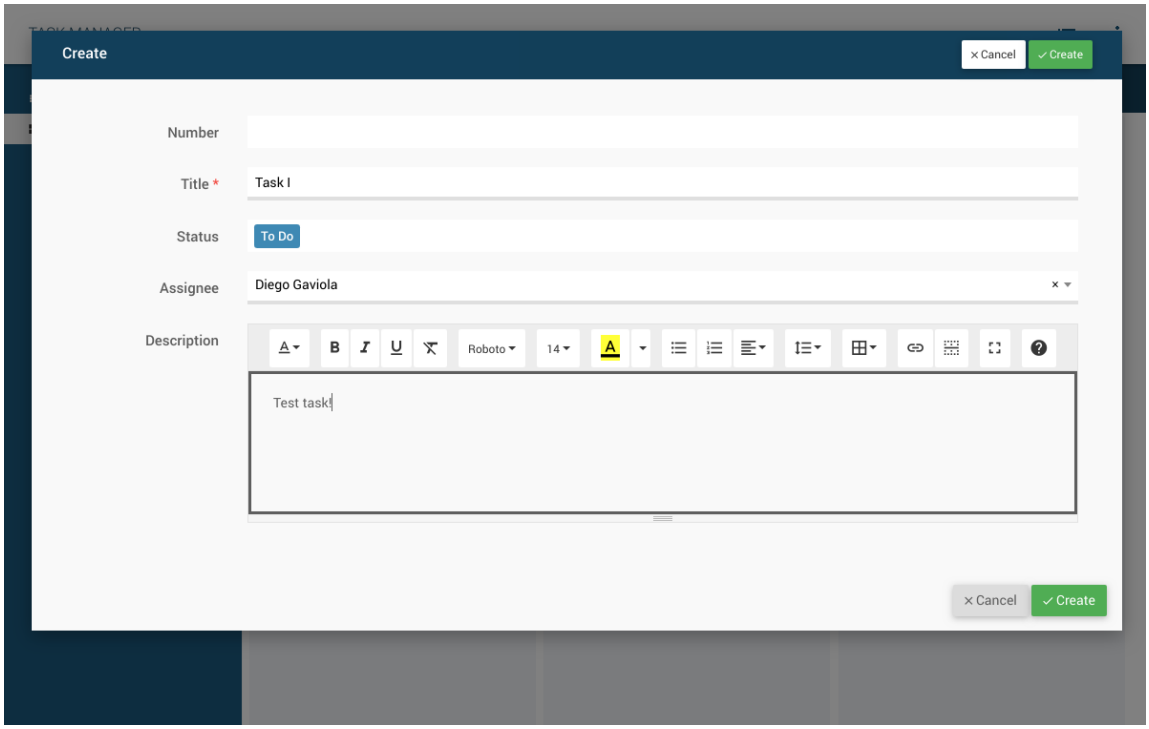


Figura 80. Crear nuevas tareas en una ventana emergente.

- Puede mover tareas de una columna a otra:

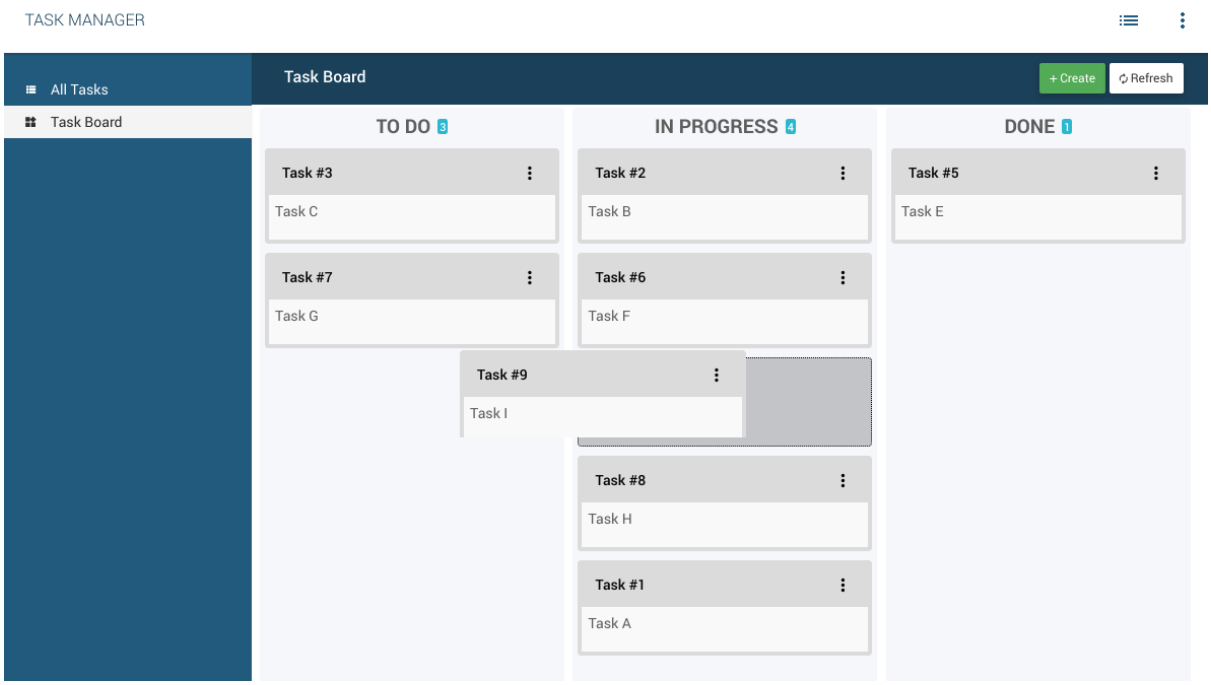


Figura 81. Mover tareas de una columna a otra.

- Puede archivar tareas y realizar las operaciones CRUD básicas:

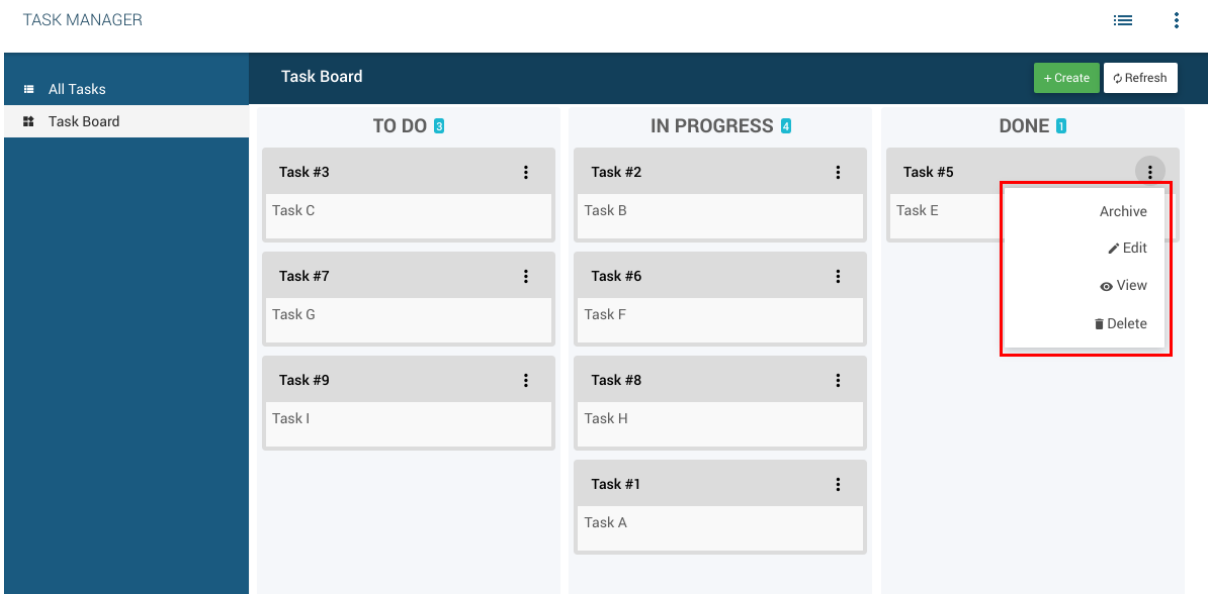


Figura 82. Archivar tareas y realizar las operaciones CRUD básicas.

- Puede clasificar las tareas (moverlas hacia arriba y hacia abajo):

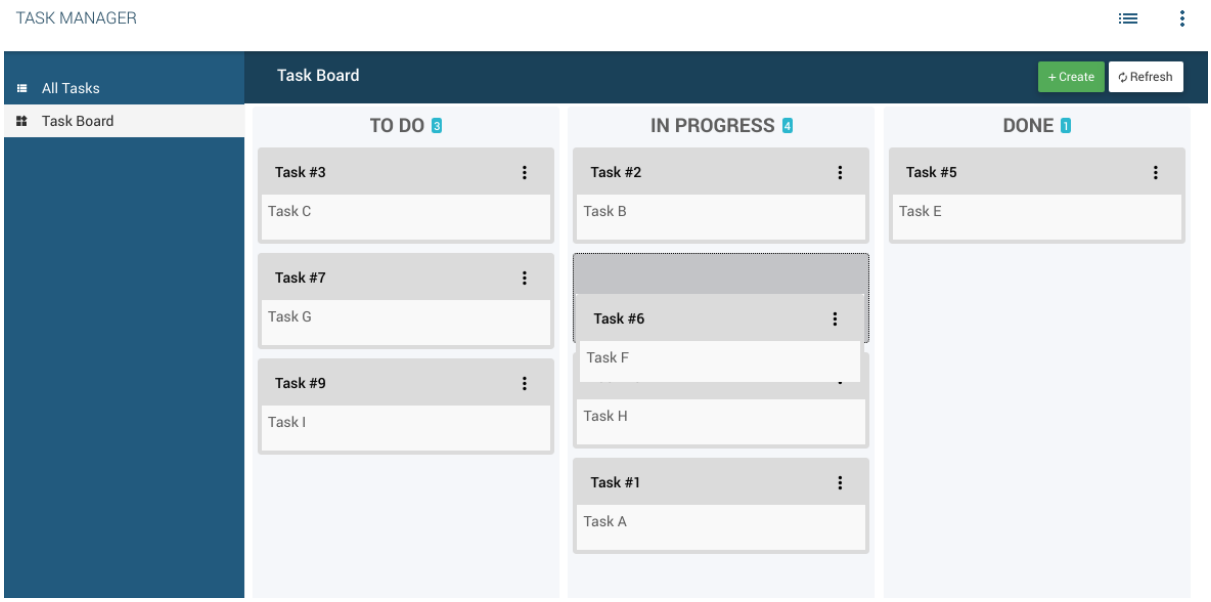


Figura 83. Clasificar las tareas (moverlas hacia arriba y hacia abajo).

Funcionalidad: Permisos

Ahora, agregaremos algunos permisos para que pueda ver cómo funcionan los grupos y los usuarios. Esto es lo que haremos:

- Agrega un nuevo campo `assignee` a la `tasks` entidad. Este será un campo de tipo `Relationship` que señalará al usuario asignado para completar la tarea. El `Related entity` debe ajustarse a `System > Users`.
- Crearemos dos grupos: `Manager` y `Support`. Los usuarios del primer grupo tendrán acceso a todas las tareas, mientras que los del segundo solo podrán ver las tareas asignadas y no podrán reasignar tareas o cambiar el título de una tarea.
- Ninguno de los grupos tendrá acceso para cambiar el campo `status` manualmente.
- Crearemos 3 usuarios: 1 administrador y 2 usuarios de soporte.
- Finalmente, probaremos que todo esté funcionando como se esperaba.

Así que vayamos ay `Model > Entities > Tasks > Fields` agreguemos un nuevo campo con etiqueta `Assignee`, nombre `assignee` y tipo `Relationship`. Asegúrese de establecer el valor predeterminado para usar un método `Script` como este:

```
return sys.context.getCurrentUserRecord();
```

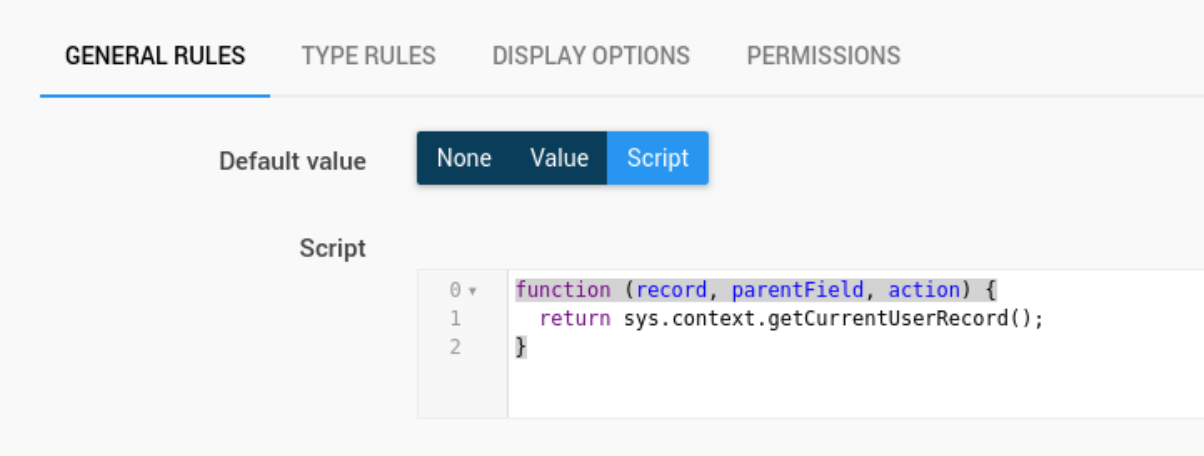


Figura 84. Agreguemos un nuevo campo con etiqueta.

Asegúrese de haber guardado sus cambios y mueva el campo para que esté arriba `Description`. Ahora agreguemos los dos nuevos grupos:

1. Vaya a `Security > Groups` en el menú principal.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista.
3. Establezca la etiqueta `Manager` y el nombre `manager` y haga clic en `Create and edit`.
4. Ahora necesitamos agregar permisos para acceder a entidades y vistas. Ir a `Security > Groups > Manager > Entity permissions`.
5. Seleccione la `Task`entidad y luego haga clic en `Apply permissions`:

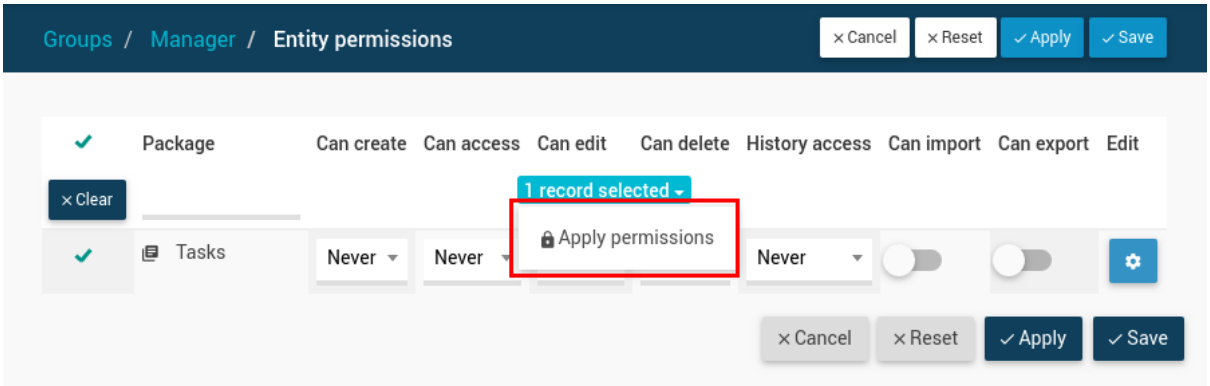


Figura 85. Agregar nuevos grupos de permisos.

6. Allí seleccione la opción `Read/write` y haga clic en `Apply`.
7. Ahora, seleccione la `System > Users` entidad y luego haga clic en `Apply permissions` y esta vez seleccione `Read-Only`.
8. Luego haga clic en el botón de configuración que está en la última columna (`Edit`) y configure los permisos de los campos para que los campos `Status` y `Rank` sean de solo lectura.

Figura 86. Configurar permisos de los campos.

Esta es la forma correcta de hacer cumplir los permisos en lugar de hacer que el campo sea de solo lectura en la interfaz de usuario.

9. Finalmente, haga clic en el botón `Apply` en la parte superior derecha de la lista para conservar los cambios.
10. Entonces ve a `Security > Groups > Manager > View permissions`.
11. Establezca la bandera en la columna `Permission` para ambas vistas:

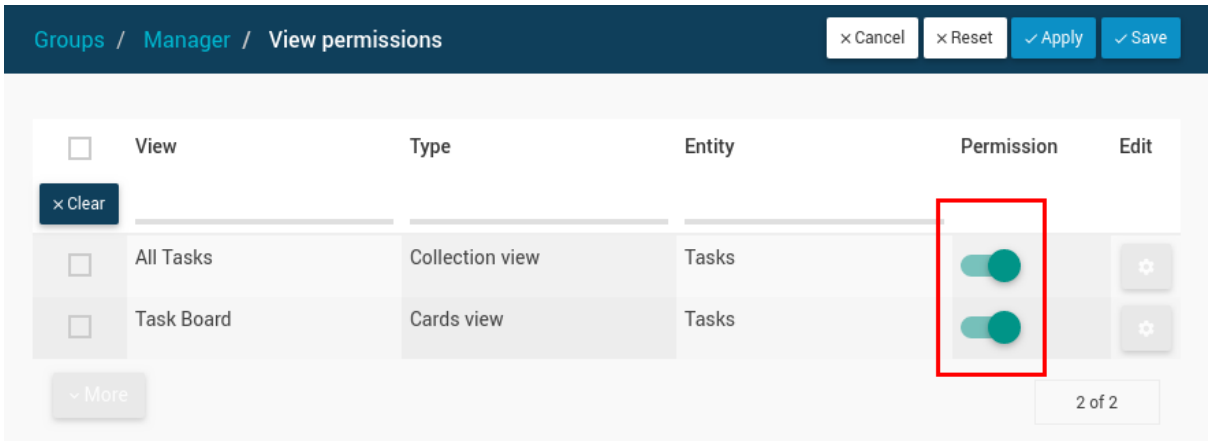


Figura 87. Establecer la bandera para las vistas.

12. Finalmente, haga clic en el `Apply` botón en la parte superior derecha de la lista para conservar los cambios.

Luego repita el mismo proceso para crear el `Support` grupo. Sin embargo, editaremos los permisos para la `Tasks` entidad:

1. Una vez que haya creado el `Support` grupo y haya agregado entidades y permisos de vistas, vaya a `Security > Groups > Support > Entity permissions` en el menú principal.
2. En la `Tasks` entidad, haga clic en el botón debajo de la `Edit` columna.
3. Los permisos de entidad deben configurarse así:

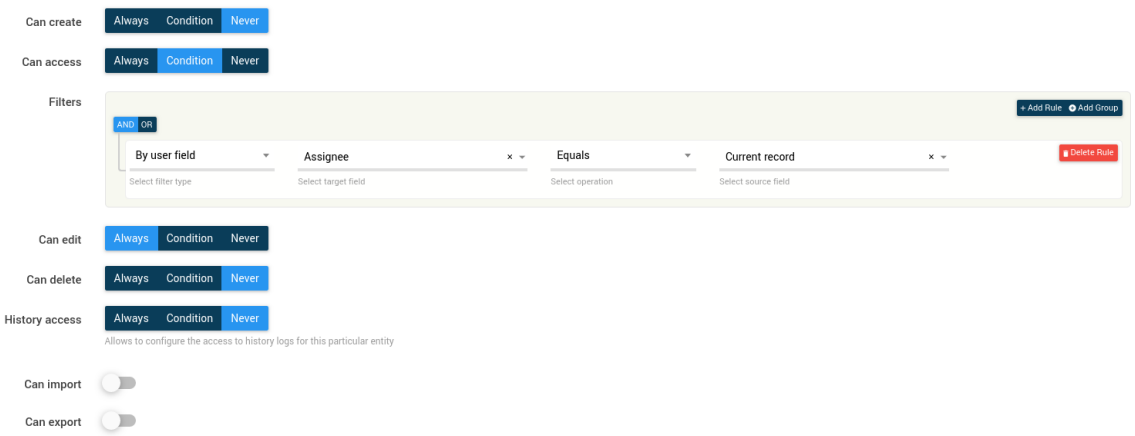


Figura 88. Editar los permisos para la entidad taks.

4. Esto es lo que está configurado:
   - Sin permisos de creación
   - El acceso solo se permite si el asignado es el mismo que el usuario actual, lo que significa que el usuario de soporte solo verá las tareas asignadas a él.
   - La edición siempre está permitida (siempre que tengan permiso de acceso)
   - Sin permisos de eliminación
5. Los permisos de campo deben configurarse así:

Figura 89. Configuración de Permisos de Campos.

Con esa configuración quitamos acceso de escritura `assignee`, `title`y `status`.

6. Los permisos de acción deben configurarse así:

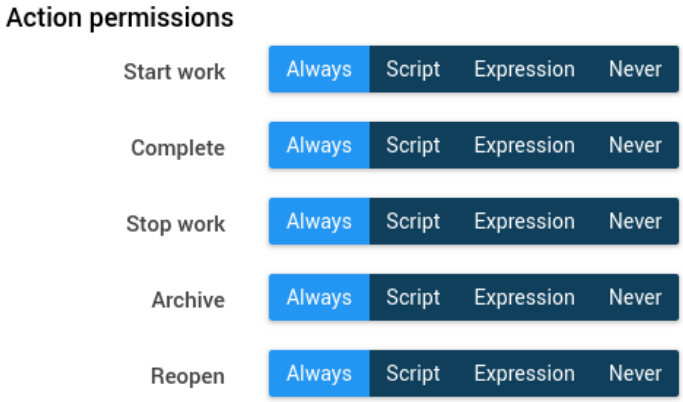


Figura 90. Configuración de Permisos de acción.

7. Ahora, seleccione la `System > Users` entidad y luego haga clic en `Apply permissions` y seleccione `Read-Only`.
8. Guarde los cambios haciendo clic en `Apply`.
9. Configure los permisos de visualización `Security > Groups > Support > View permissions` de la misma forma que para los administradores.

Ahora tenemos los dos grupos creados y se han definido los permisos. Empujemos cambios para que podamos usar estos grupos al crear usuarios.
Es hora de crear algunos usuarios (asegúrese de presionar los cambios antes):

1. Vaya a `Security > Users` en el menú principal.
2. Haga clic en el `Create` botón en la parte superior derecha de la lista.
3. Complete el de con:
    - `First name`: Gerente1
    - `Last name`: Prueba
    - `Email`: manager1@test.com
    - `Generate Password`: cierto
    - `Groups`: agrega el `Manager` grupo como `Primary`
4. Haga clic en `Create` para guardar el usuario.

Luego repita el mismo proceso para estos usuarios:

Nombre propio	Apellido	Correo electrónico	Grupo primario
Soporte1	Prueba	support1@test.com	Apoyo
Soporte2	Prueba	support2@test.com	Apoyo

Tabla 9. Usuarios para creación.

No necesita introducir cambios cuando trabaja con usuarios porque los usuarios se modifican directamente en el tiempo de ejecución, ya que no forman parte de los metadatos de la aplicación.
Ahora tenemos los permisos definidos y los usuarios para probar, así que vayamos al tiempo de ejecución de la aplicación:

- Si inició sesión desde su usuario desarrollador, podrá cambiar a cualquier otro usuario. Esto se puede hacer con la opción `Switch user` en el menú de usuario del tiempo de ejecución de la aplicación:

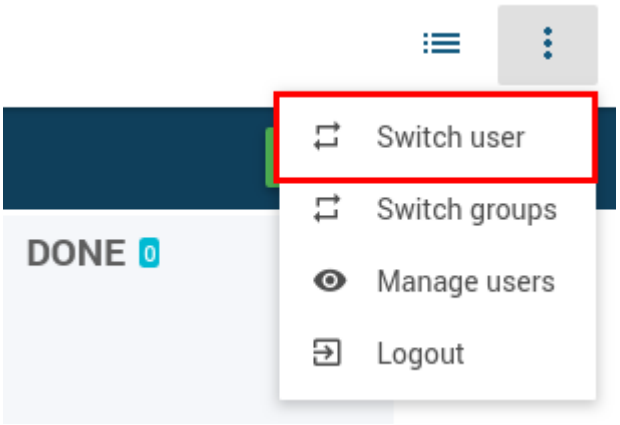


Figura 91. Configuración de Permisos de acción.

- Cambiar al usuario `Manager1 Test`.
- Debería poder ver todas las tareas.
- Cree dos tareas: asigne una `Support1 Test`y la otra a `Support2 Test`.
- Cambie a usuario `Support1 Test` usando la `Switch user` opción en el menú de usuario.
- Solo debería ver las tareas asignadas a `Support1 Test`.
- Cambie a usuario `Support2 Test` usando la `Switch user` opción en el menú de usuario.
- Solo debería ver las tareas asignadas a `Support2 Test`.
- Puede volver a su usuario desarrollador con la misma `Switch user` opción.

Eso es lo básico sobre los permisos, pero hay más funciones para permitir una administración de permisos más detallada, solo consulte la referencia del desarrollador para obtener más información.

Funcionalidad: Integraciones

En SLINGR, para integrarse con otros servicios, utilizará “endpoints”. Facilitan la conexión a otras aplicaciones como Google Calendar, Twilio, Chargify, QuickBooks, etc.
El plan SLINGR Free tiene un número limitado de endpoints disponibles (si necesitas más, ¡puedes actualizar a otro plan!), Donde podemos encontrar el endpoints HTTP que permite realizar solicitudes HTTP genéricas. Esto es perfecto porque muchos servicios tienen API REST a las que se puede acceder con solicitudes HTTP regulares.

Aunque puede realizar solicitudes a muchos servicios utilizando el punto final HTTP, es mejor utilizar endpoints específicos, ya que proporcionan muchas más funciones y simplifican la interoperabilidad con el otro servicio.
En este caso, usaremos el endpoints HTTP para publicar notificaciones sobre nuevas tareas en un canal de Slack. Esto significa que, para continuar, deberá tener una cuenta de Slack con acceso a un espacio de trabajo de Slack. Puede crear el suyo de forma gratuita en slack.com.
Una vez que tenga su espacio de trabajo de Slack, deberá crear una aplicación de Slack en api.slack.com/apps. Asegúrese de hacer lo siguiente:

- Configuras tu aplicación para tener un bot.
- Instalas tu aplicación en tu espacio de trabajo.

Estas son las secciones donde encontrará esas cosas:

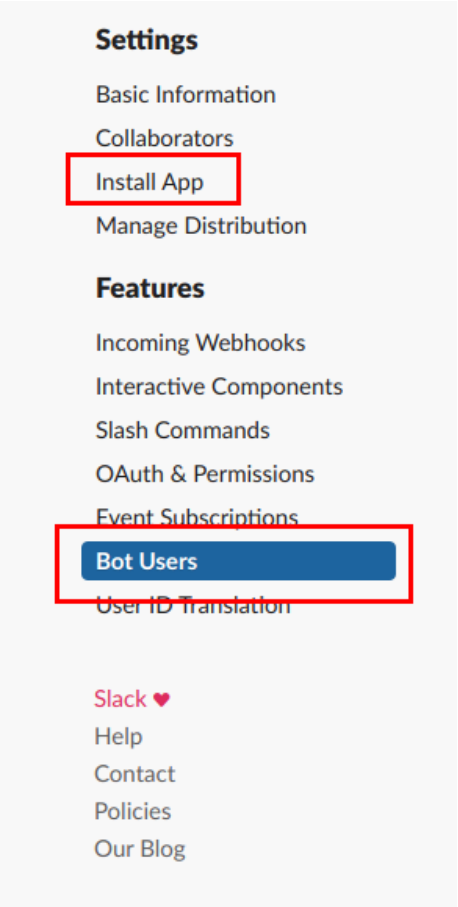


Figura 92. Configuración para obtención de un bot e instalar aplicación.

Una vez que instale la aplicación en su espacio de trabajo, encontrará el `Bot UserRecordOAuth Access Token`, que está en la sección `Install App`. Cópialo porque lo usaremos en nuestra aplicación. Sin embargo, en lugar de codificarlo en el código, haremos algo mucho mejor: usaremos variables de entorno.

Las variables de entorno permiten almacenar credenciales o configuraciones que deben ser seguras (están encriptadas) y que probablemente cambien del entorno (por ejemplo,

probablemente usará una cuenta de Slack diferente en dev que en prod). Así que agreguemos las variables de entorno que necesitaremos:

1. Ir al creador de aplicaciones
2. Ir a `Environment settings > Environment variables` en el menú principal
3. Haga clic en `Create` para crear una nueva variable de entorno
4. Pon `SLACK_TOKEN` el nombre de la variable y pega el token del bot que copiaste de tu aplicación Slack.
5. Haga clic en `Create` para guardar la variable
6. Haga clic de `Create`nuevo para agregar otra variable de entorno
7. Nómbrelo `SLACK_CHANNEL`y ponga el ID del canal donde desea publicar sus mensajes
8. Haga clic en `Create` para guardar la variable

Deberías terminar con algo como esto:

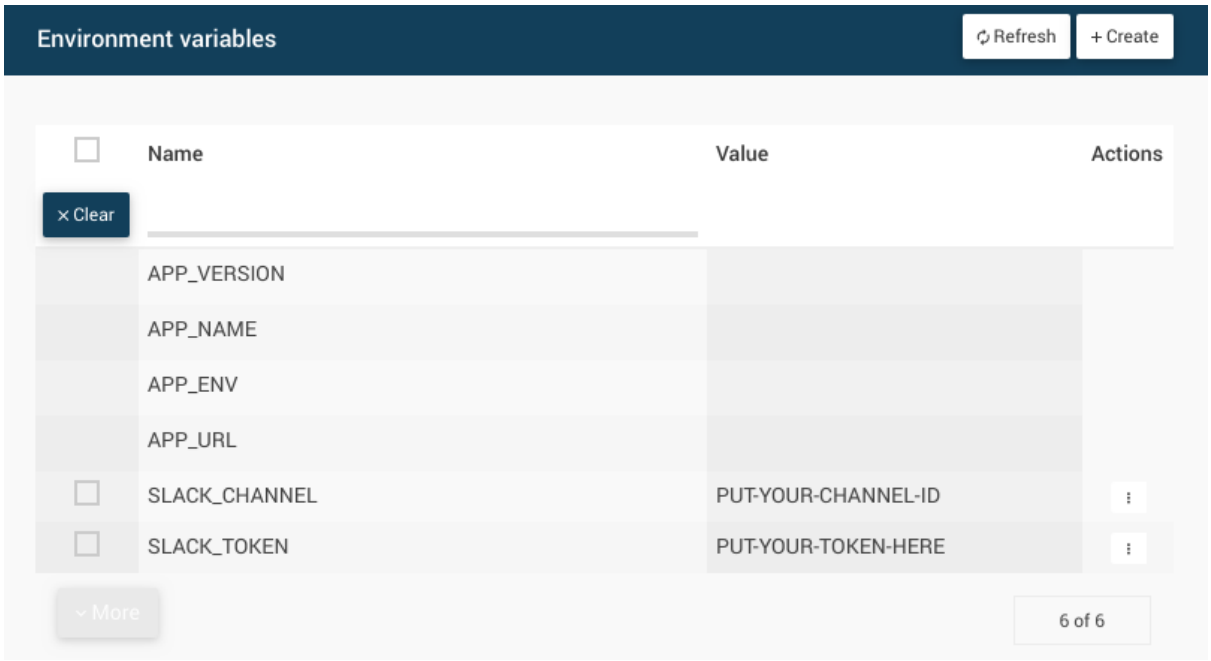


Figura 93. Resultado de agregar variables de entorno.

Bien, ahora necesitamos escribir el código que enviará la notificación. Como esta función podría reutilizarse en diferentes lugares de la aplicación, la colocaremos en una biblioteca. Las bibliotecas son excelentes para poner código que reutilizará en diferentes lugares de su aplicación. Vamos a hacerlo:

1. Ir a `Model > Libraries` en el menú principal.
2. Haga clic en `Create`, en la parte superior del panel que se encuentra en el medio.
3. Nombra la biblioteca `slack`y haz clic en `Save`.
4. Luego, en el editor de código de la derecha ingrese el siguiente código:

```
exports.sendMessage = function(message) {
        app.endpoints.http.post({
                path: 'https://slack.com/api/chat.postMessage',
                headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
```

```
                },
                body: {
                        token: '${SLACK_TOKEN}',
                        channel: '${SLACK_CHANNEL}',
                        text: message
                }
        });
};
```

5. Haga clic en Save para guardar la biblioteca.

Aquí hay algunas cosas a tener en cuenta:

- La forma de "exportar" símbolos en la biblioteca es agregándolos dentro del objeto exports. Es por eso, que, ponemos nuestra función sendMessage de exports, por lo que podemos llamar de otros lugares de nuestra aplicación.
- Para usar variables de entorno, debe rodearlas con ${}. Estos serán marcadores de posición que serán reemplazados por el valor real de la variable en ese entorno.

Entonces ya tenemos la aplicación de Slack, la configuración en variables de entorno y la librería con el código para enviar una notificación a un canal. Lo único que nos falta es agregar algo que active la notificación cuando se crea una nueva tarea. ¡Y aquí es donde los oyentes entran en acción!
Los oyentes permiten reaccionar a los eventos que suceden en su aplicación. El evento podría ser algo que suceda en otra aplicación conectada a través de un endpoint, podría ser un temporizador o cambios en los datos de tu aplicación, que es lo que estamos buscando. Creamos el oyente:

1. Ir a Model > Listeners en el menú principal (también hay un atajo dentro de la entidad).
2. Haga clic en Create para configurar un nuevo oyente.
3. Ponga Send notification to Slack como etiqueta y deje el nombre calculado.
4. Seleccionar Data como Type.
5. En Entity seleccionar la Tasks entidad.
6. En Rules haga clic en Add New.
7. En Source events seleccionar UserRecordEvents y Script Events.
8. En Event seleccionar On record created.
9. Dejar Condition a None.
10. Haga clic en Add para agregar la regla.
11. Pon la bandera Execute in the background.
12. Escribe este guión en Action:
    ```
    app.slack.sendMessage('Task created: '+record.label());
    ```
    Aquí puedes ver cómo llamamos a las bibliotecas, usando el prefijo app..
13. Finalmente haga clic en Create para guardar el oyente.

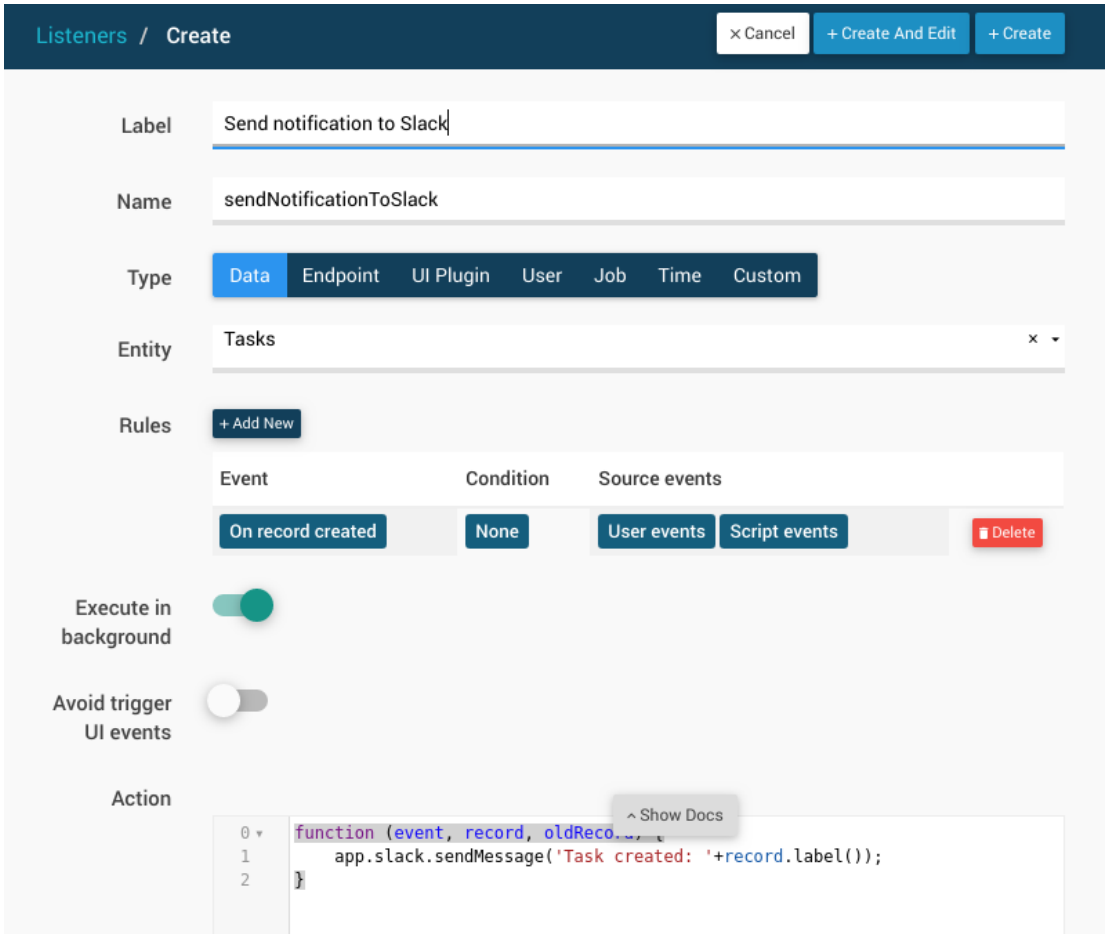


Figura 94. *Creación y guardado del oyente.*

Muy bien, ¡estamos listos para probar nuestra integración! Pusheemos los cambios.
Ahora, cuando crea una nueva tarea, debería ver la notificación en su canal de Slack. Si no lo ve, significa que hay un problema. Pero no se preocupe, siga leyendo para ver cómo puede averiguar qué está pasando, usando el monitor de la aplicación.

Funcionalidad: Supervisar la aplicación

Hemos terminado con el desarrollo de la aplicación, pero es bueno mencionar que puede ver lo que está sucediendo en su aplicación en el monitor de la aplicación.
Para acceder al monitor de la aplicación, debe hacer clic en el botón `Monitor` en la configuración del entorno:

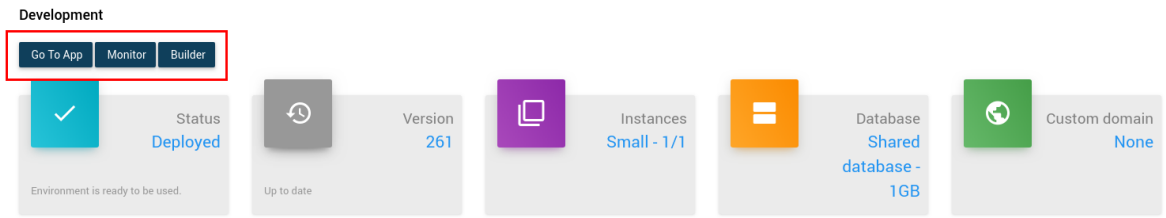


Figura 95. *Visualizar aplicación desde el monitor de la aplicación.*

Desde allí puede tener información de sus aplicaciones en ejecución. Por ejemplo, puede ver los registros de su aplicación en la Logs sección:

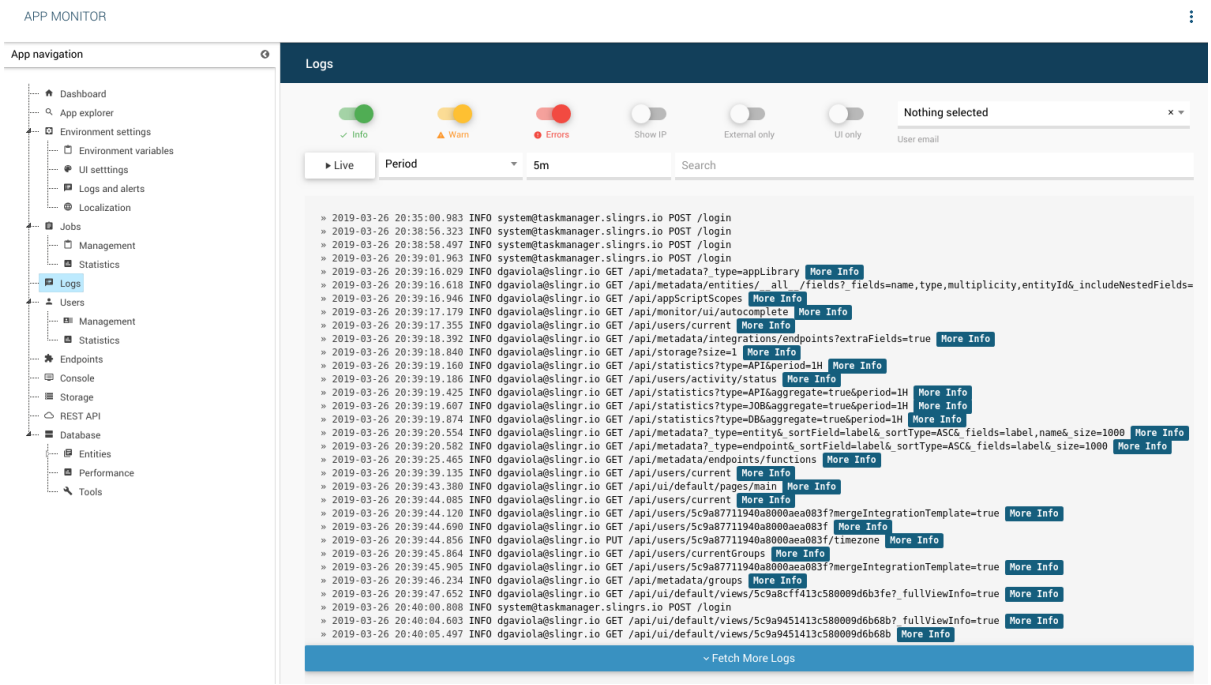


Figura 96. Visualizar información de aplicaciones en ejecución.

O ver el estado de los trabajos en segundo plano:

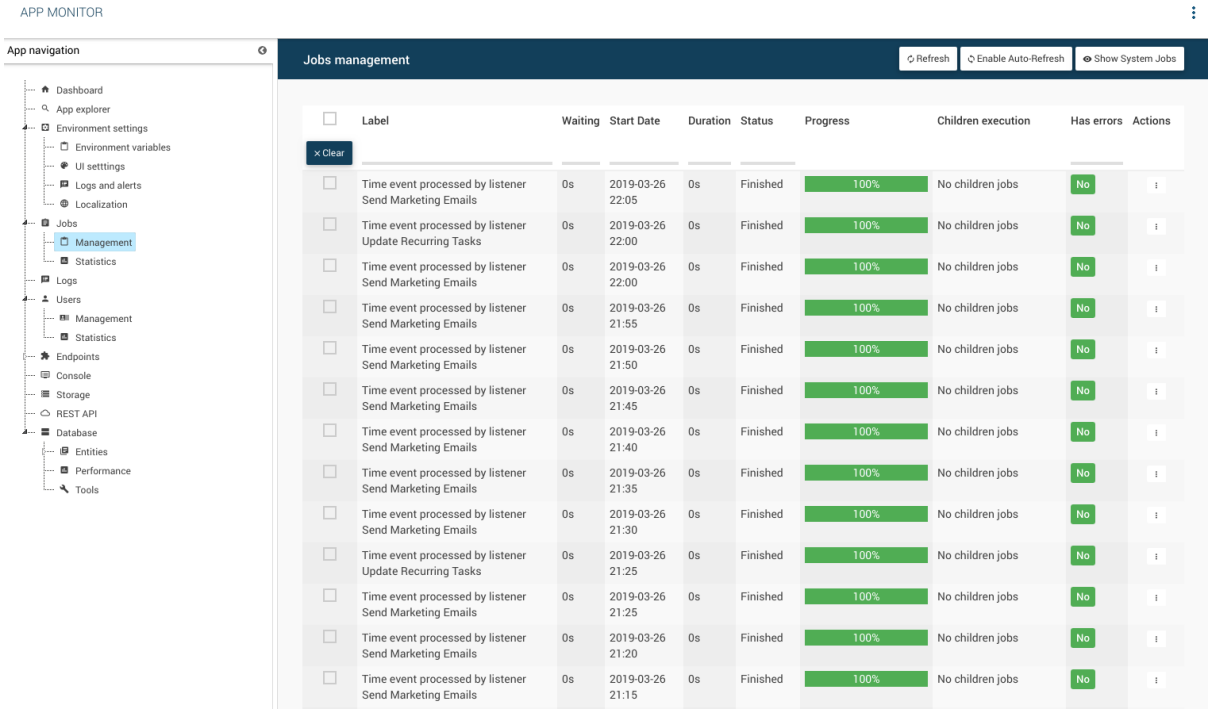


Figura 97. Ver estado de aplicaciones en ejecución.

1.5.2.2. Modelo Lógico

A continuación, se muestra el diagrama lógico del sistema:

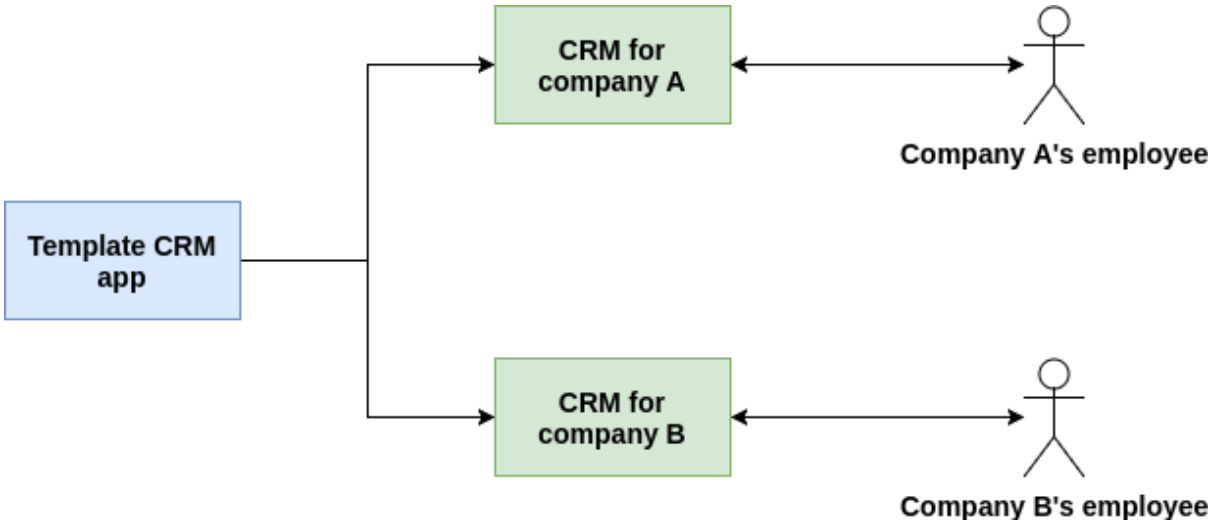


Figura 98. Modelo lógico del sistema1.

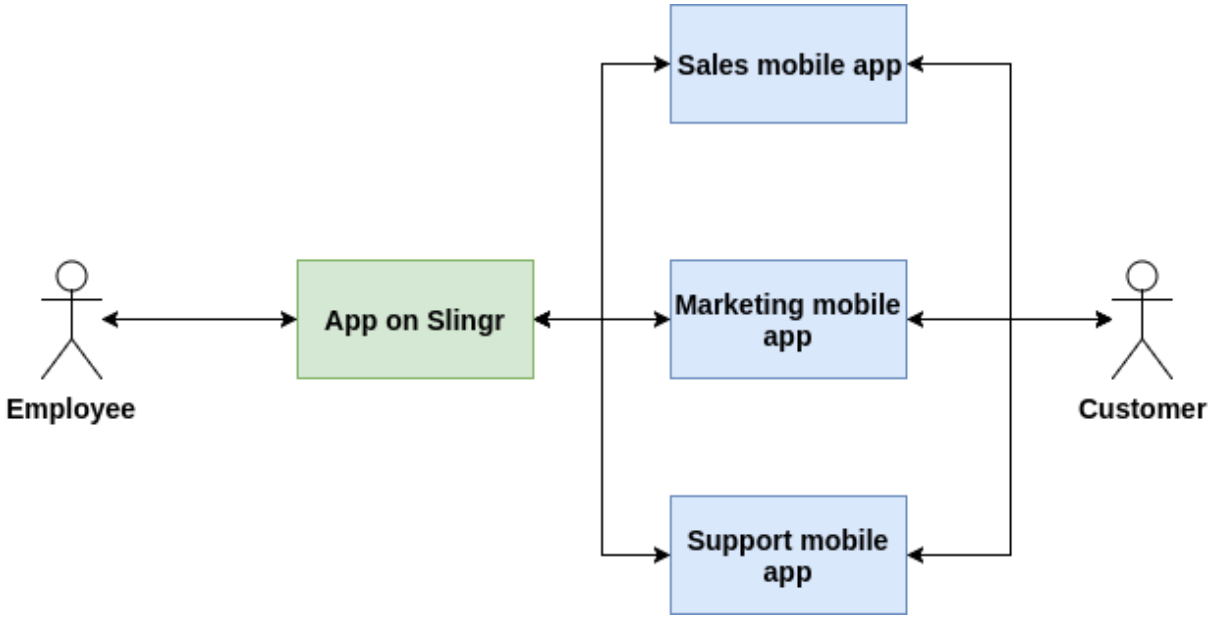


Figura 99. Modelo lógico del sistema2.

1.5.2.3. Problemas y necesidades

FUNCIONALIDAD	PROBLEMA
Crear la aplicación	Solo se puede hostear en servicios de Google no permite crear AWS.
Crear entidad de tareas	Poca visibilidad de todas las alternativas posibles.

Crea una vista básica	Demora significativa en el renderizado de la UI.
Prueba de aplicación	Es obligatorio ejecutar la acción push, podría ser renderizado dinámico.
Agregar acciones y definir el flujo de trabajo	Difícil integración con sistemas externos.
Agregar vista de flujo de trabajo	No aplican. No hay problemas.
Permisos	No se pueden establecer grupos de permisos
Integraciones	Las integraciones son extremadamente complejas
Supervise su aplicación	Los filtros de los logs son básicos, podría integrarse con kibana o algún motor junto con elastic searc

Tabla 10. Problemas y necesidades detectados SLINGR.

Generales:

- Documentación escasa y poco clara.

2. Cuadro comparativo de Sistemas

	JHipster	Altova UModel	Cuba	Slingr	OpenXava
Web	SI	NO	NO	SI	NO
Multilenguaje	SI	NO	NO	NO	NO
Devuelve código (flexible)	SI	NO	SI	NO	SI
Interfaz gráfica	NO	SI	SI	SI	SI
Base de datos	SI	SI	SI	SI	SI
Open Source	SI	NO	NO	NO	NO

Tabla 11. Cuadro comparativo de sistemas.

Objetivos y alcances preliminares del nuevo Sistema

1. Objetivos

El objetivo del proyecto es implementar un sistema generador del código de una aplicación web con arquitectura moderna, mediante un previo diseño de un diagrama relacional dirigido por un metamodelo que se ajusta a las restricciones del paradigma de la programación orientada a objetos.

Además, se pensó el sistema para que en un futuro se le brinde la posibilidad al usuario de desplegar su código en alguna plataforma Cloud de tal manera que quede 100% funcional.
También se tuvo en cuenta que en el futuro se puede dar la posibilidad de generar un diagrama desde uno ya existente realizado en generadores de modelos de otros softwares.

2. Alcance

En base a los datos recopilados del relevamiento, y a las posibilidades del proyecto, se decidió

que la aplicación cuente con los siguientes módulos que se encargan de dar soporte y tratar los requisitos específicos de cada funcionalidad prevista anteriormente:

2.1. Módulos

2.1.1. Módulo de Roles, Usuarios y Login

- Login mediante Google.
- Login mediante Github.
- Usuario estándar el cual va a tener permiso para la creación de un único diagrama y funciones limitadas para la creación de las UI.
- Usuario premium el cual va a tener permiso para la creación ilimitada de diagramas, lenguaje y features para las UI.
- Gestión de tipos de usuario.
- ABM Permisos.
- ABM Roles.
- Restablecer contraseña.

2.1.2. Módulo de Gestión de Diagramas

- ABM Diagrama.
- ABM Permisos de Diagrama.
- ABM Clase.
- ABM Interface.
- ABM Atributo.
- ABM Operación.
- ABM Relación.
- ABM Herencia.
- ABM Implementación.
- ABM Vista-UI: a definir.
- ABM Tipos de datos: a definir.

Módulo de proyectos que permita relacionar a la funcionalidad incluida en el módulo Gestión de diagramas.

2.1.3. Módulo generador código Api en C#

- Generar la arquitectura base del proyecto.
- Generar la configuración para la conexión a la base de datos seleccionada y la cadena de conexión brindada.
- Generar las distintas capas de servicio, repositorio, controladores.
- Generar las secciones de los modelos.
- Generar el DockerFile.
- Generar por cada clase del diagrama un archivo .cs.
- Generar por cada clase del diagrama un controlador con las acciones CRUD habilitadas.
- Generar por cada clase del diagrama un servicio con las acciones CRUD habilitadas.

- Generar por cada clase del diagrama una cada de repositorio con las acciones CRUD habilitadas y según la base de datos seleccionada.
- Generar por cada atributo de la clase un atributo en el archivo .cs.
- Generar por cada operación de la clase una operación en el archivo .cs.
- Generar por cada relación en la clase una relación en el archivo .cs y configurar para Entity Framework.

2.1.4. Módulo generador código Api en Java 8

- Generar la arquitectura base del proyecto.
- Generar la configuración para la conexión a la base de datos seleccionada y la cadena de conexión brindada.
- Generar las distintas capas de servicio, repositorio, controladores.
- Generar las secciones de los modelos.
- Generar el DockerFile.
- Generar por cada clase del diagrama un archivo .java.
- Generar por cada atributo de la clase un atributo en el archivo .java.
- Generar por cada operación de la clase una operación en el archivo .java.
- Generar por cada relación en la clase una relación en el archivo.java y configurar para Hibernate.

2.1.5. Módulo generador código Frontend Angular

- Generar la arquitectura básica del proyecto de angular.
- Generar los repositorios para la conexión con el Webapi.
- Generar por cada clase del diagrama un archivo .js.
- Generar por cada atributo de la clase un atributo en el archivo .js.
- Generar un componente para las vistas según vistas seleccionadas para cada entidad (Edit con acciones CRUD, listados con filtros y la visibilidad de cada atributo).

2.1.6. Módulo generador código Frontend React

- Generar la arquitectura básica del proyecto de React.
- Generar los repositorios para la conexión con el Webapi.
- Generar por cada clase del diagrama un archivo .js.
- Generar por cada atributo de la clase un atributo en el archivo .js.
- Generar un componente para las vistas según vistas seleccionadas para cada entidad (Edit con acciones CRUD, listados con filtros y la visibilidad de cada atributo).

2.1.7. Módulo exportador del código a un repositorio

- Establecer conexión con el repositorio.
- ABM Commit interno.
- ABM Push en el repositorio remoto.

2.1.8. Módulo de Reportes

- Cantidad de proyectos.
- Reportes de colaboración de usuarios en proyecto.
- Lenguajes más utilizados (para backend y frontend).

DISEÑO

Objetivo y alcances definidos del nuevo sistema

Según lo relevado exponemos lo siguientes objetivos definitivos del nuevo sistema:

- Implementar y desarrollar un sistema completo y consistente generador del código de una aplicación web utilizando una arquitectura moderna.
- El sistema debe ser desarrollado basado en un diseño previo de un diagrama relacional
- La base del diagrama relacional debe estar dirigido por metamodelos (ajustados a las restricciones del paradigma de POO).
- En próximas iteraciones se puede llegar a que el sistema brinde la posibilidad al usuario de desplegar su código en plataformas Cloud de tal manera que sea 100% funcional.
- También se prevé la posibilidad de generar un diagrama desde uno ya existente realizado en generadores de modelos de otros software.

1. Alcances y Límites del nuevo sistema

Buscar modelos ya existentes
Se brindará la opción de buscar diagramas filtrando por nombre o descripción de los diagramas públicos o con modo de solo lectura ya existentes en el sistema.

Documentación del dominio de un sistema
Mediante el diseño de un diagrama de dominio con las entidades, propiedades y sus relaciones quedará como documentación del sistema.

Facilitar las buenas prácticas y convenciones
La creación del diagrama solo permitirá buenas prácticas como el nombre de las variables, modificadores de accesos correctos y la correcta creación de las relaciones entre las entidades.

Abstraer a los analistas de sistemas del desarrollo del código
Cuando un analista diagrama a través de nuestra aplicación tiene la ventaja de poder obtener el código de un frontend y un backend.

Agilizar el desarrollo de una API
Una vez diagramado el dominio necesario, se podrá obtener el código de la API correspondiente con todas las operaciones CRUD (ABM) configuradas. Además, el código generado brindará un Swagger ya configurado que servirá como documentación de la API.

Agilizar el desarrollo de un Micro Frontend
Una vez diagramado el dominio necesario, se puede obtener el código de un Micro Frontend listo para usar. El código devuelto será en un lenguaje moderno seleccionado, con buenas prácticas y convenciones.

En base a los datos recopilados del relevamiento, y a las posibilidades del proyecto, se ha decidido en forma definitiva que la aplicación cuente con los siguientes módulos que se encargan de dar soporte y tratar los requisitos específicos de cada funcionalidad prevista anteriormente:

1.1. Módulo de Roles, Usuarios y Login

- Login mediante Github.
- Usuario estándar el cual tiene permiso para la creación de un único diagrama y funciones limitadas para la creación de las UI.
- Usuario premium el cual tiene permiso para la creación ilimitada de diagramas, lenguaje y features para las UI.
- Gestión de tipos de usuario.
- ABM Permisos.
- ABM Roles.
- Backup.
- Restore.

1.2. Módulo de Gestión de diagramas

- ABM Diagrama.
- ABM Permisos Diagrama.
- ABM Clase.
- ABM Atributo.
- ABM Relación.
- ABM Tipos de datos.

1.3. Módulo generador código Api en Java 8

- Generar la arquitectura base del proyecto.
- Generar la configuración para la conexión a la base de datos seleccionada y la cadena de conexión brindada.
- Generar las distintas capas de servicio, repositorio, controladores.
- Generar las secciones de los modelos.
- Generar el DockerFile.
- Generar por cada clase del diagrama un archivo .java.
- Generar por cada atributo de la clase un atributo en el archivo .java.
- Generar por cada operación de la clase una operación en el archivo .java.
- Generar por cada relación en la clase una relación en el archivo.java y configurar para Hibernate.
- Brinda un Swagger que sirve como documentación.

1.4. Módulo generador código Frontend React

- Generar la arquitectura básica del proyecto de React.
- Generar los repositorios para la conexión con el Webapi.

- Generar por cada clase del diagrama un archivo .js.
- Generar por cada atributo de la clase un atributo en el archivo .js.
- Generar un componente para las vistas según vistas seleccionadas para cada entidad (Edit con acciones CRUD, listados con filtros y la visibilidad de cada atributo).

1.5. Módulo exportador del código a un repositorio

- Establecer conexión con el repositorio.
- ABM Commit interno.
- ABM Push en el repositorio remoto.

1.6. Módulo de Reportes

- Cantidad de proyectos.
- Reportes de colaboración de usuarios en proyecto.
- Lenguajes más utilizados (para backend y frontend).

Salidas del Sistema

Describen las salidas que produce el sistema detallando por cada una de las funcionalidades, las salidas que producen.

Nombre	Crear Usuario mediante GitHub
Descripción	Solicita los datos del usuario username y password
Datos	• Username or email: String • Password: String • Mensaje de error: String
Caso de uso	Registrar en el sistema mediante GitHub

Tabla 12. Salida. Crear Usuario mediante GitHub.

Nombre	Loguear Usuario mediante GitHub
Descripción	Solicita los datos del usuario username y password
Datos	• Username or email: String • Password: String • Mensaje de error: String
Caso de uso	Ingresar al sistema mediante GitHub

Tabla 13. Salida. Loguear Usuario mediante GitHub.

Nombre	Consultar diagramas

Descripción	Se solicita la lista con los diagramas generados
Datos	• List<Diagramas> con ○ Nombre: string ○ BlackEndLenguaje: BlackEndLenguaje ○ FrontEndTool:FrontEndTool ○ DataBase: DataBase ○ DiagramaRol: string
Caso de uso	Ver listado de diagramas propios

Tabla 14. Salida. Consultar diagramas.

Nombre	Buscar diagramas públicos
Descripción	Se solicita una lista de diagramas según el nombre filtrado y las tecnologías escogidas
Datos	• List<Diagramas> con ○ Nombre: string ○ LenguajelBackEnd: LenguajelBackEnd ○ LenguajeFrontEnd:LenguajeFrontEnd ○ BaseDeDatos: BaseDeDatos
Caso de uso	Buscar diagramas públicos

Tabla 15. Salida. Buscar diagramas públicos.

Nombre	ABM Diagrama
Descripción	El usuario podrá crear, editar y eliminar un diagrama
Datos	• Nombre: string • ModificadorAccesoDiagrama: ModificadorAccesoDiagrama • FechaCreacion: Date • UltimaModificacion: Date • Clase: List<Clase> • TipoDato: List<dataType> • BlackEndLenguaje: blackEndLenguaje • FrontEndTool: frontEndTool • BaseDatos: DataBase • CadenaConexion: string
Caso de uso	ABM Diagrama

Tabla 16. Salida. ABM Diagrama.

Nombre	ABM Clase
Descripción	El usuario podrá crear, editar y eliminar una clase
Datos	• IdTipo: IdTipo • ModificadorAccesoClase:ModificadorAccesoClase • Nombre: string • Atributos: List<Atributo>

	• XCoord: decimal • YCoord: decimal • Ancho: decimal • esAbstracta: bool • esStatica: bool
Caso de uso	ABM Clase

Tabla 17. Salida. ABM Clase.

Nombre	ABM Atributo
Descripción	El usuario podrá crear, editar y eliminar un atributo de una clase
Datos	• Nombre: string • ModificadorAcceso: ModificadorAcceso • TipoAtributo: TipoAtributo • EsVisible: bool • PermiteNulos: bool • ValorPorDefecto: object • RegexExpresionPattern: string • EsRequerido: bool
Caso de uso	ABM Atributo

Tabla 18. Salida. ABM Atributo.

Nombre	ABM Relación
Descripción	El usuario podrá crear, editar y eliminar las relaciones de una clase
Datos	• ClaseOrigen: Clase • ClaseDestino: Clase • TipoRelacion: TipoRelacion • XCoordInicial: decimal • YCoordInicial: decimal • XCoordFinal: decimal • YCoordFinal: decimal
Caso de uso	ABM Relaciones

Tabla 19. Salida. ABM Relación.

Nombre	Generar código frontend
Descripción	El usuario generará el código frontend
Datos	• Componentes: React.Component • EstilosComponente: React.CSSProperites • DockerFile: DockerFile • ScriptImportadorDependencias: batch • ScriptIniciadorDeProyecto: batch

Caso de uso	Subir código frontend a GitHub, Descargar código frontend

Tabla 20. Salida. Generar código frontend.

Nombre	Generar código backend
Descripción	El usuario generará el código backend
Datos	• ClasesDominio: file • ArchivosCapaRepositorio: file • ArchivosCapaServicio: file • ArchivosCapaControlador: file • ConfiguracionEntityFramework: file
Caso de uso	Subir código backend a GitHub, Descargar código backend.

Tabla 21. Salida. Generar código backend.

Modelo funcional

1. Lista de Historias de Usuario

ID	HU001	
Título	Registro en el sistema mediante GitHub.	
Descripción	COMO	Usuario.
	QUIERO	Registrarme al sistema mediante mi cuenta de GitHub.
	PARA	Poder hacer uso de las funcionalidades del sistema.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione registrar mediante github.
	• Quiero
	  ○ Que se abra la ventana de github donde puedo ingresar mis credenciales.
	• **Cuando**
	  ○ El registro sea exitoso.
	• Quiero
	  ○ Que me dirija a la lista de diagramas.	

Tabla 22. Historia de usuario HU001.

ID	HU002	
Título	Ingreso al sistema con credenciales de GitHub.	
Descripción	COMO	Usuario.
	QUIERO	Ingresar al sistema utilizando mis credenciales de GitHub.
	PARA	Poder hacer uso de las funcionalidades del sistema.
Criterios de aceptación	• **Cuando** ○ Seleccione login mediante github.	

	• Quiero
	  ○ Que se abra la ventana de github donde puedo ingresar mis credenciales.
	
	• **Cuando**
	  ○ El login sea exitoso.
	• Quiero
	  ○ Que me dirija a la lista de mis diagramas.

Tabla 23. Historia de usuario HU002.

ID	**HU003**	
Título	Consultar mis diagramas.	
Descripción	**COMO**	Modelador.
	QUIERO	Consultar mis diagramas.
	PARA	Continuar desarrollando un proyecto existente.

Criterios de aceptación	• **Cuando**
	  ○ Ingreso a la sección de mis diagramas.
	• Quiero
	  ○ Ver una lista con mis diagramas que contenga.
	  ▪ El nombre de diagrama.
	  ▪ La tecnología frontend, backend y la base de datos usada.
	  ▪ La cantidad de clases que tiene mi diagrama.
	  ○ Un botón para cada diagrama para ingresar a la edición del diagrama.
	  ○ Un botón para crear un nuevo diagrama.	

Tabla 24. Historia de usuario HU003.

ID	**HU004**	
Título	Búsqueda de diagramas públicos disponibles.	
Descripción	**COMO**	Modelador.
	QUIERO	Buscar diagramas públicos disponibles.
	PARA	Reutilizar alguna solución ya existente.

Criterios de aceptación	• **Cuando**
	  ○ Ingrese a la sección de buscar diagramas.
	• Quiero
	  ○ Ver un listado de sugeridos entre los más populares y un cuadro de búsqueda.
	
	• **Cuando**
	  ○ Escriba en el cuadro de búsqueda.
	• Quiero
	  ○ Que vayan apareciendo los diagramas públicos que contengan en su título lo que escribí en el cuadro de búsqueda.	

Tabla 25. Historia de usuario HU004.

ID	HU005	
Título	Creación de diagrama.	
Descripción	**COMO**	Modelador.
	QUIERO	Crear un nuevo diagrama.
	PARA	Poder modelar el dominio de mi problema.

Criterios de aceptación	• **Cuando**
	  ○ Se cree un nuevo diagrama.
	• Quiero
	  ○ Que me dirija a la edición de dicho diagrama.
	
	• **Cuando**
	  ○ Cargue los datos del diagrama con un lenguaje de frontend vacío.
	• Quiero
	  ○ Que se muestre un mensaje el mensaje de error pertinente.
	
	• **Cuando**
	  ○ Cargue los datos del diagrama con un lenguaje de backend vacío.
	• Quiero
	  ○ Que se muestre un mensaje el mensaje de error pertinente.
	
	• **Cuando**
	  ○ Cargue los datos del diagrama con una opción de base de datos vacía o una cadena de conexión vacía.
	• Quiero
	  ○ Que se muestre un mensaje el mensaje de error pertinente.
	
	• **Cuando**
	  ○ Cargue los datos del diagrama con un nombre de diagrama vacío.
	• Quiero
	  ○ Que se muestre un error de “Nombre diagrama requerido”.	

Tabla 26. Historia de usuario HU005.

ID	HU006	
Título	Agregar clases a un diagrama.	
Descripción	**COMO**	Modelador.
	QUIERO	Poder agregar clases a mí diagrama.
	PARA	Poder representar mis entidades de dominio.

Criterios de aceptación	• **Cuando**
	  ○ Este en la vista de edición de un diagrama.
	• Quiero
	  ○ Ver un botón para crear una nueva clase.	

	• **Cuando**
	  ○ Seleccione el botón de crear clase.
	• Quiero
	  ○ Que se cree en el diagrama la nueva clase con sus campos vacíos.
	  ○ Que se abra un panel con las opciones de una nueva clase donde se vean los campos para editar: nombre, x, y, modificador de acceso, y el tipo de datos de id que va a tener la clase.
	
	• **Cuando**
	  ○ Vaya actualizando los campos en el panel de la creación.
	• Quiero
	  ○ Que se vaya actualizando la clase en el diagrama.
	
	• **Cuando**
	  ○ Cree mi clase sin un nombre válido (vacío o nombre de clase ya existente).
	• Quiero
	  ○ Que el recuadro de la clase en el diagrama se marque en rojo, indicando que hay un error.

Tabla 27. Historia de usuario HU006.

ID	**HU007**	
Título	Agregar atributos a una clase.	
Descripción	**COMO**	Modelador.
	QUIERO	Poder agregar atributos a mis clases.
	PARA	Poder representar las propiedades de las entidades.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione el botón de “Agregar atributo” en una clase.
	• Quiero
	  ○ Que se abra un panel con las opciones de un nuevo atributo donde se vean los campos para editar: nombre, clase.
	
	• **Cuando**
	  ○ Vaya actualizando los campos en el panel de la creación.
	• Quiero
	  ○ Que se vaya actualizando el atributo de la clase del diagrama.
	
	• **Cuando**
	  ○ Cree mi atributo sin un nombre válido (vacío).
	• Quiero
	  ○ Que el fondo del atributo en la clase del diagrama se establezca en rojo, indicando que hay un error.	

Tabla 28. Historia de usuario HU007.

ID	**HU008**

Título	Agregar relaciones a una clase.	
Descripción	**COMO**	Modelador.
	QUIERO	Poder agregar relaciones entre mis entidades.
	PARA	Poder representar las relaciones de mis objetos de dominio.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione el botón de “Agregar Relación” en una clase.
	• Quiero
	  ○ Que se abra un panel con las opciones de una nueva relación donde se vean los campos para editar: tipo de relación, clase fuente, clase destino, nombre, multiplicidad.
	• **Cuando**
	  ○ Vaya actualizando los campos en el panel de la creación.
	• Quiero
	  ○ Que se vaya actualizando la relación de la clase del diagrama mediante una flecha que representa la relación entre clases.
	• **Cuando**
	  ○ Cree mi relación inválida (nombre relación vacío).
	• Quiero
	  ○ Que la flecha que representa la relación en el diagrama se establezca en rojo, indicando que hay un error.	

Tabla 29. Historia de usuario HU008.

ID	**HU009**	
Título	Generar código frontend.	
Descripción	**COMO**	Modelador.
	QUIERO	Poder generar el código frontend.
	PARA	Poder ejecutar mi aplicación frontend o editarla.

Criterios de aceptación	• **Cuando**
	  ○ Ingrese a mi diagrama.
	• Quiero
	  ○ Ver un botón para exportar el código.
	• **Cuando**
	  ○ Seleccione el botón de generar código.
	• Quiero
	  ○ Ver la opción de generar mi código frontend.
	• **Cuando**
	  ○ Genere mi código frontend.
	• Quiero
	  ○ Ver la opción de descargarlo o exportarlo a mi repositorio.	

Tabla 30. Historia de usuario HU009.

ID	HU010	
Título	Generar código backend.	
Descripción	**COMO**	Modelador.
	QUIERO	Poder generar el código backend.
	PARA	Poder ejecutar mi aplicación backend o editarla.

Criterios de aceptación	• **Cuando**
	  ○ Ingrese a mi diagrama.
	• Quiero
	  ○ Ver un botón para exportar el código.
	
	• **Cuando**
	  ○ Seleccione el botón de generar código.
	• Quiero
	  ○ Ver la opción de generar mi código backend.
	
	• **Cuando**
	  ○ Genere mi código backend.
	• Quiero
	  ○ Ver la opción de descargarlo o exportarlo a mi repositorio.	

Tabla 31. Historia de usuario HU010.

ID	HU011	
Título	Consulta en el sistema los reportes sobre la aplicación**.**	
Descripción	**COMO**	Administrador.
	QUIERO	Consultar en el sistema los reportes por cantidad, colaboración, lenguaje y estadística.
	PARA	Poder hacer uso de las funcionalidades del sistema.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione el menú reporte.
	• Quiero
	  ○ Que se despliegue el menú de reportes donde puedo seleccionar tipo de Reportes.
	
	• **Cuando**
	  ○ Seleccione el tipo de reporte en el menú.
	• Quiero
	  ○ Que me muestre el reporte seleccionado.	

Tabla 32. Historia de usuario HU011.

ID	HU012	
Título	Consultar el informe de cantidad de proyectos.	
Descripción	**COMO**	Administrador.
	QUIERO	Poder consultar el informe de diagramas creados en un rango de fechas.
	PARA	Ver las estadísticas de cuantos informes se generaron en ese rango de fechas.
Criterios de aceptación	● **Cuando** ○ Seleccione el reporte de proyectos generados. ● Quiero ○ Poder seleccionar un rango de fechas en las quiero ver el informe. ● **Cuando** ○ Seleccione en el botón cargar ● Quiero ○ Ver un gráfico que indique la evolución de los proyectos generados mes a mes del rango de fechas seleccionados y ver el número total de proyectos creados.	

Tabla 33. Historia de usuario HU012.

ID	HU013	
Título	Consultar el informe de participación de usuario-diagrama.	
Descripción	**COMO**	Administrador.
	QUIERO	Poder consultar un informe que indique la participación de un usuario en un diagrama.
	PARA	Conocer en qué diagramas ha colaborado un usuario o saber quiénes han colaborado en un diagrama.
Criterios de aceptación	● **Cuando** ○ Seleccione el reporte de usuario-proyecto. ● Quiero ○ Poder seleccionar un proyecto o un usuario. ● **Cuando** ○ Seleccione en el botón cargar ● Quiero ○ Ver las estadísticas que indiquen en qué diagrama ha aportado un usuario o ver que usuario han aportado a un diagrama y cuantas veces.	

Tabla 34. Historia de usuario HU013.

ID	HU014	
Título	Consultar el informe de participación de usuario-diagrama.	
Descripción	**COMO**	Administrador.
	QUIERO	Poder consultar un informe que indique la participación de un usuario en un diagrama.
	PARA	Conocer en qué diagramas ha colaborado un usuario o saber quiénes han colaborado en un diagrama.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione el reporte de usuario-proyecto.
	• Quiero
	  ○ Poder seleccionar un proyecto o un usuario.
	
	• **Cuando**
	  ○ Seleccione en el botón cargar
	• Quiero
	  ○ Ver las estadísticas que indiquen en qué diagrama ha aportado un usuario o ver que usuario han aportado a un diagrama y cuantas veces.	

Tabla 35. Historia de usuario HU014.

ID	HU015	
Título	Consultar el informe de participación de los lenguajes más utilizados.	
Descripción	**COMO**	Administrador.
	QUIERO	Poder consultar el informe de los lenguajes más utilizados.
	PARA	Conocer el ranking de cada lenguaje separado por backend y frontend.

Criterios de aceptación	• **Cuando**
	  ○ Seleccione el reporte de ranking de proyectos.
	• Quiero
	  ○ Ver el ranking en modo de gráfico de los lenguajes más utilizados, tanto en frontend como en backend, de manera ordenada.	

Tabla 36. Historia de usuario HU015.

2. Modelo de casos de uso

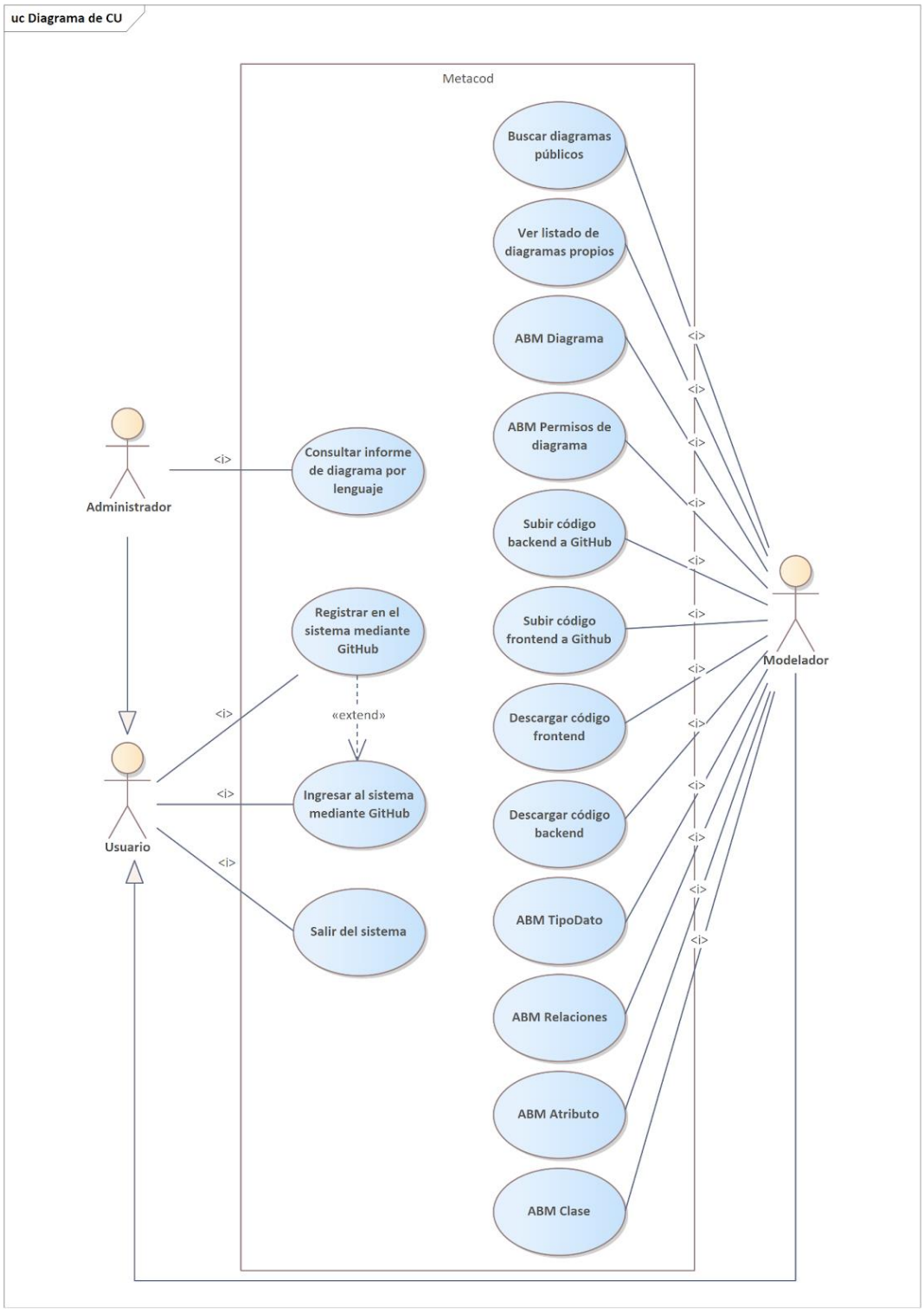


Figura 100. Modelo de Casos de Uso de todo el sistema METACOD.

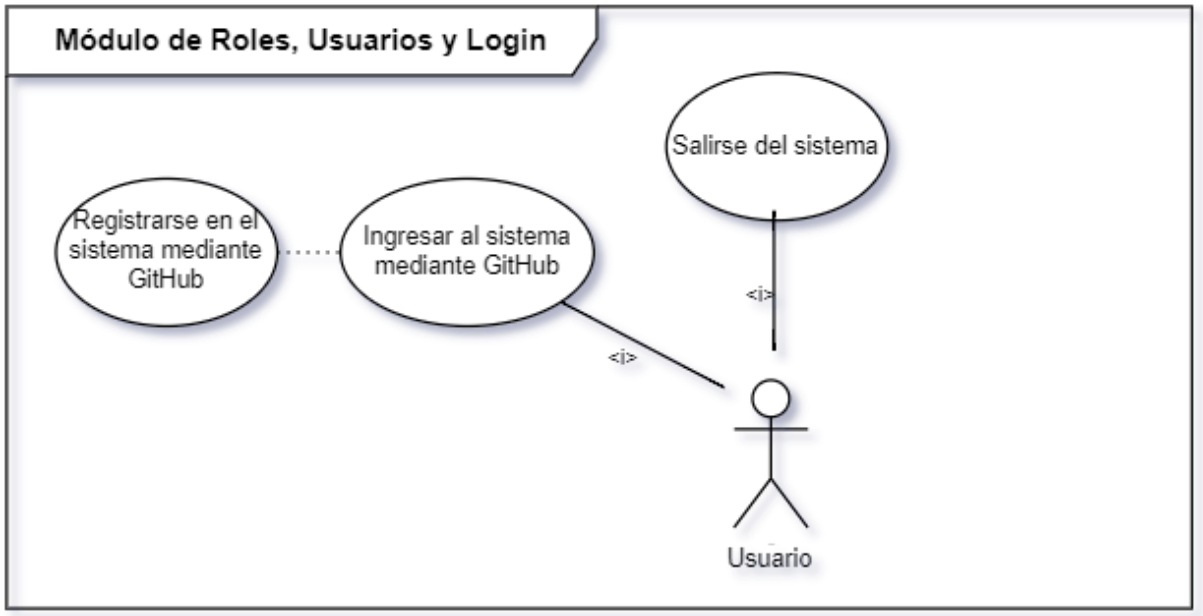


Figura 101. *Casos de uso del Módulo de Roles, Usuarios y Login.*

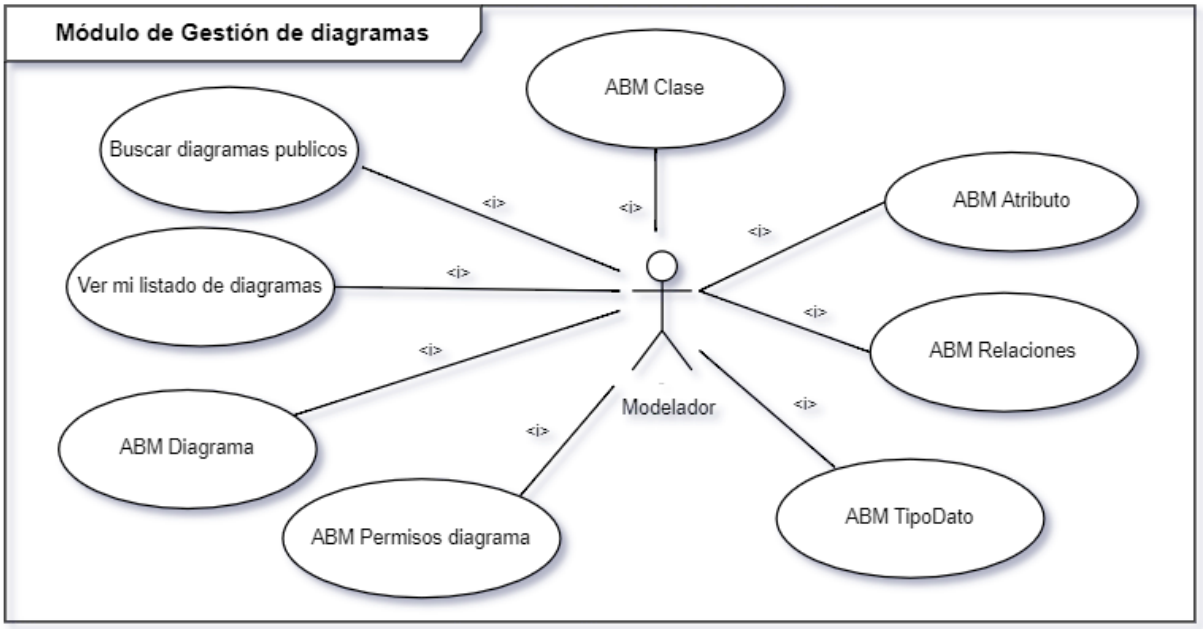


Figura 102. *Casos de uso del Módulo de Gestión de Diagramas.*

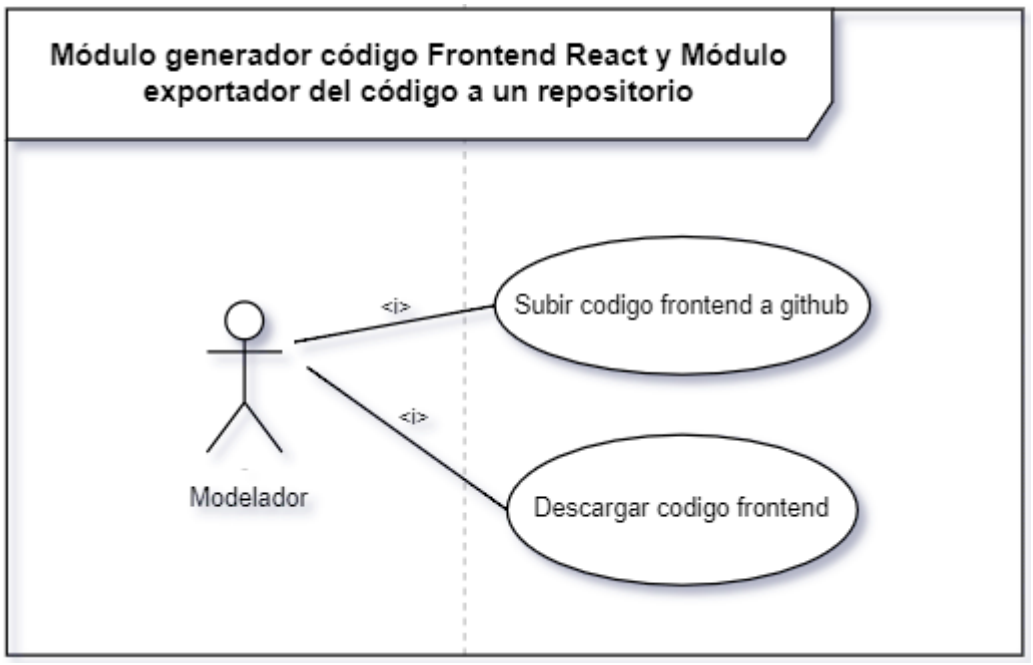


Figura 103. Casos de uso del Módulo generador código Frontend React y Módulo exportador del código a un repositorio.

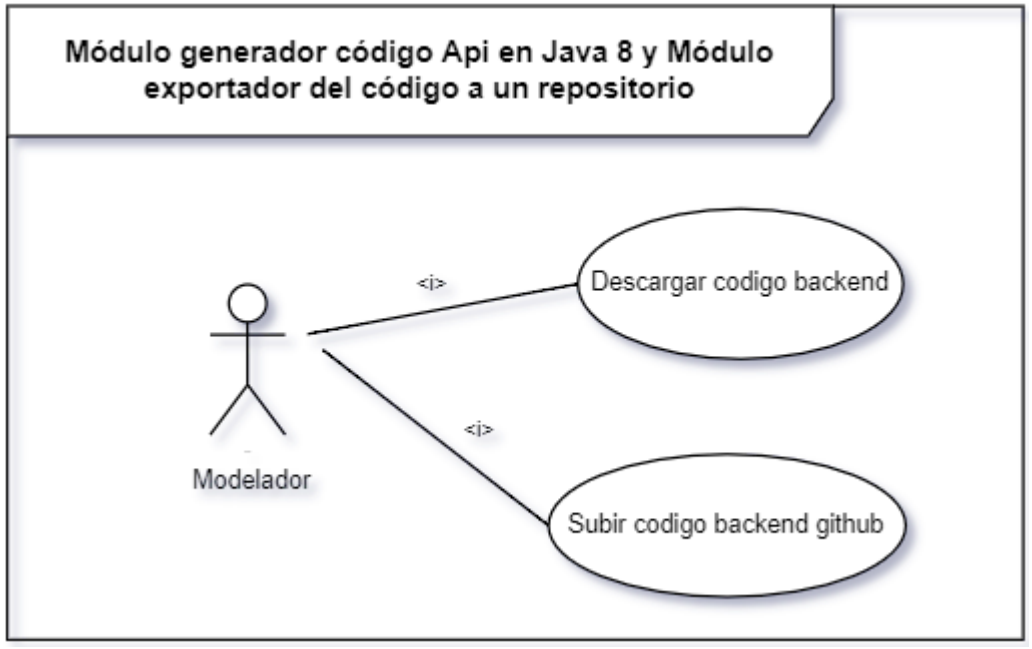


Figura 104. Casos de uso del Módulo generador código Api en Java 8 y Módulo exportador del código a un repositorio.

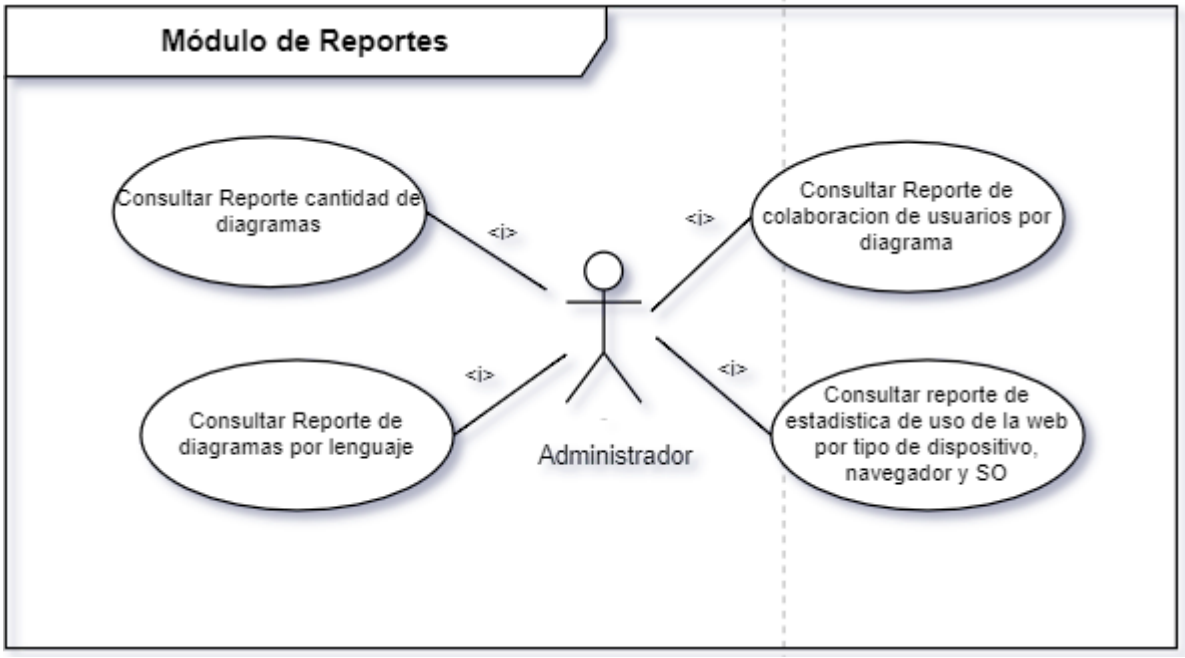


Figura 105. Casos de uso del Módulo de Reportes.

2.1. Cabeceras de Casos de Uso

1	Nombre del caso de uso	Registrar en el sistema mediante GitHub
	Actor	Usuario.
	Breve descripción	El actor selecciona la opción de ingresar y es redireccionado a una UI proporcionada por GitHub donde una vez ingresado y validados los datos de su cuenta se le asigna un usuario en el sistema.
	Prioridad	
	Parámetros de entrada	nombreUsuario, contraseñaUsuario.
	Estado inicial	
	Estado final	Usuario • con nombreUsuario definido. • con emailUsuario definido. • con rolUsuarioSistema definido con valor 'Modelador'.

Tabla 37. Cabecera CU 1.

2	Nombre del caso de uso	Ingresar al sistema mediante GitHub
	Actor	Usuario.

	Breve descripción	El actor ingresa al sistema con los datos de su cuenta de GitHub.
	Prioridad	
	Parámetros de entrada	nombreUsuario, contraseñaUsuario.
	Estado inicial	Usuario • con nombreUsuario existente. • con emailUsuario existente.
	Estado final	Acceso al Sistema (Metacod).

Tabla 38. Cabecera CU 2.

3	Nombre del caso de uso	Salir del sistema
	Actor	Usuario.
	Breve descripción	El usuario se desconecta del sistema.
	Prioridad	
	Parámetros de entrada	
	Estado inicial	Usuario logueado o conectado al sistema.
	Estado final	Usuario desconectado del sistema.

Tabla 39. Cabecera CU 3.

4	Nombre del caso de uso	Buscar diagramas públicos
	Actor	Modelador.
	Breve descripción	Se seleccionan todos los diagramas con modificador de acceso público dentro del sistema.
	Prioridad	
	Parámetros de entrada	
	Estado inicial	List<Diagrama>: • con modificadorDeAccesoDiagrama igual a público.
	Estado final	

Tabla 40. Cabecera CU 4.

5	Nombre del caso de uso	Ver listado de diagramas propios
	Actor	Usuario.
	Breve descripción	El usuario selecciona ver los diagramas que tiene en su propiedad.

	Prioridad	C.
	Parámetros de entrada	*idUsuario.*
	Estado inicial	Usuario existente: • con idUsuario igual a *idUsuario.* • con List<Diagrama> definido.
	Estado final	

Tabla 41. Cabecera CU 5.

6	**Nombre del caso de uso**	**ABM Diagrama**
	Actor	Usuario.
	Breve descripción	Permite crear, modificar o dar de baja un Diagrama.
	Prioridad	A.
	Parámetros de entrada	**Alta** • nombreDiagrama. • modificadorAccesoDiagrama. • lenguajeBackEnd. • lenguajeFrontEnd. • BaseDeDatos. • CadenaConexion. **Modificación** • nombreDiagrama. • modificadorAccesoDiagrama. • ultimaModificacion. • BaseDeDatos. • CadenaConexion. **Baja** • IdDiagrama.
	Estado inicial	**Alta** • Instancia de Usuario existente. **Baja** • Instancia de diagrama existente. **Modificación**

		• Instancia de diagrama existente.
	Estado final	**Alta** Instancia de diagrama creada con: • nombreDiagrama **definido.** • modificadorAccesoDiagrama **definido.** • fechaCreacion **igual a fechaActual.** • ultimaModificacion **igual a fechaActual.** • clase **vacío.** • tiposDatos **vacío.** • lenguajeBackEnd **definido.** • lenguajeFrontEnd **definido.** • BaseDeDatos **definido.** • CadenaConexion **definido.** **Modificación** Instancia de diagrama modificada con: • nombreDiagrama **modificado.** • modificadorAccesoDiagrama **modificado.** • ultimaModificacion **igual a fechaActual.** • BaseDeDatos **modificado.** • CadenaConexion **modificado.** **Baja** Instancia de diagrama eliminada. Instancias de clases asociadas eliminadas. Instancias de atributos asociados eliminados. Instancias de relaciones asociadas eliminadas.

Tabla 42. Cabecera CU 6.

7	Nombre del caso de uso	ABM Permisos diagrama
	Actor	Usuario.
	Breve descripción	Permite crear, modificar o dar de baja un Permisos de Diagrama.
	Prioridad	
	Parámetros de entrada	Usuario, Diagrama, Rol.
	Estado inicial	**Alta** • Instancia de PermisosDiagrama inexistente. **Modificación** • Instancia de PermisosDiagrama existente. **Baja**

		• Instancia de PermisosDiagrama existente.
	Estado final	**Alta** • Instancia de PermisosDiagrama creada. **Modificación** • Instancia de PermisosDiagrama modificada. **Baja** • Instancia de PermisosDiagrama eliminada.

Tabla 43. Cabecera CU 7.

8	**Nombre del caso de uso**	**ABM Clase**
	Actor	Usuario.
	Breve descripción	Permite al actor crear, modificar o dar de baja una Clase.
	Prioridad	A.
	Parámetros de entrada	**Alta** • idDiagrama. **Modificación** • idTipo. • modificadorAccesoClase. • nombreClase. • coordenadaX. • coordenadaY. • Ancho. • esAbstracta. • esEstatica. **Baja** • IdClase.
	Estado inicial	**Alta** Instancia de diagrama existente. **Modificación** Instancia de clase existente. **Baja** Instancia de clase existente.
	Estado final	**Alta** Instancia de clase creada con: • idTipo **definido con valor int.** • modificadorAccesoClase **definido con valor 'public'.** • nombreClase **sin definir.** • coordenadaX **definido con valor 0.0.** • coordenadaY **definido con valor 0.0.**

		• ancho **definido con valor 180.0.** • esAbstracta **definido con valor false.** • esEstatica **definido con valor false.** **Modificación** Instancia de clase modificada con: • idTipo **modificado.** • modificadorAccesoClase **modificado.** • nombreClase **modificado.** • coordenadaX **modificado.** • coordenadaY **modificado.** • ancho **modificado.** • esAbstracta **modificado.** • esEstatica **modificado.** **Baja** Instancia de clase eliminada. Instancias de atributos asociados eliminados. Instancias de relaciones asociadas eliminadas.

Tabla 44. Cabecera CU 8.

9	Nombre del caso de uso	ABM Atributo
	Actor	Usuario.
	Breve descripción	Permite el alta, modificación o baja de un atributo.
	Prioridad	A.
	Parámetros de entrada	**Alta** • claseId. **Modificación** • nombreAtributo. • modificadorAccesoAtributo. • tipo. • esVisibleEnUi. • admiteValorNulo. • valorPorDefecto. • patronExpresionRegex. • esRequerido. **Baja** • atributoId.
	Estado inicial	**Alta**

		Instancia de clase creada. **Modificación** Instancia de atributo creada. **Baja** Instancia de atributo creada.
	Estado final	**Alta** Instancia de atributo creada con • nombreAtributo **definido sin valor.** • modificadorAccesoAtributo **definido con valor 'public'.** • tipo **definido con valor 'string'.** • esVisibleEnUi **definido con valor true.** • admiteValorNulo **definido con valor false.** • valorPorDefecto **definido sin valor.** • patronExpresionRegex **definido sin valor.** • esRequerido **definido con valor true.** **Modificación** Instancia de atributo modificada con • nombreAtributo **modificado.** • modificadorAccesoAtributo **modificado.** • tipo **modificado.** • esVisibleEnUi **modificado.** • admiteValorNulo **modificado.** • valorPorDefecto **modificado.** • patronExpresionRegex **modificado.** • esRequerido **modificado.** **Baja** Instancia de atributo eliminada.

Tabla 45. Cabecera CU 9.

10	Nombre del caso de uso	ABM Relaciones
	Actor	Usuario.
	Breve descripción	Permite dar de alta, modificar o dar de baja relaciones.
	Prioridad	A.
	Parámetros de entrada	**Alta** • claseOrigen. **Modificación** • multiplicidadFuente. • claseDestino. • multiplicidadDestino. • tipoRelacion.

		• coordenadaXInicial. • coordenadaYInicial. • coordenadaXFinal. • coordenadaYFinal. **Baja** • relacionId.
	Estado inicial	**Alta** Instancia de clase creada. **Modificación** Instancia de relacion creada. **Baja** Instancia de relacion creada.
	Estado final	**Alta** Instancia de clase creada con: • multiplicidadFuente **con valor definito en 1.** • claseDestino **sin valor definido.** • multiplicidadDestino **con valor definito en 1.** • tipoRelacion **con valor definito en 'asociacion'.** • coordenadaXInicial **sin valor definido.** • coordenadaYInicial **sin valor definido.** • coordenadaXFinal **sin valor definido.** • coordenadaYFinal **sin valor definido.** **Modificación** Instancia de clase modificada con: • multiplicidadFuente **modificado.** • claseDestino **modificado.** • multiplicidadDestino **modificado.** • tipoRelacion **modificado.** • coordenadaXInicial **modificado.** • coordenadaYInicial **modificado.** • coordenadaXFinal **modificado.** • coordenadaYFinal **modificado.** **Baja** Instancia de clase eliminada.

Tabla 46. Cabecera CU 10.

11	Nombre del caso de uso	ABM TipoDato
	Actor	Usuario.

	Breve descripción	Permite dar de alta, modificar o dar de baja un tipo de dato.
	Prioridad	B.
	Parámetros de entrada	**Alta** • nombreTipoDeDato. • diagramaId. **Modificación** • nombreTipoDeDato. • datos. **Baja** • tipoDatoId.
	Estado inicial	**Alta** Instancia de diagrama creada. **Modificación** Instancia de tipoDeDato creada. **Baja** Instancia de tipoDeDato creada.
	Estado final	**Alta** Instancia de tipoDeDato creada con: • nombreTipoDeDato **definido.** • datos **sin definir.** **Modificación** Instancia de tipoDeDato creada con: • nombreTipoDeDato **modificado.** • datos **modificado.** **Baja** Instancia de tipoDeDato eliminada.

Tabla 47. Cabecera CU 11.

12	Nombre del caso de uso	Descargar código backend
	Actor	Usuario.
	Breve descripción	Permite descargar al usuario el codigo del backend en formato .zip/.rar.
	Prioridad	A.
	Parámetros de entrada	• diagramaId.
	Estado inicial	Instancia de diagrama con: • nombreDiagrama **definido.** • modificadorAccesoDiagrama **definido.** • lenguajeBackEnd **definido.** • lenguajeFrontEnd **definido.** • baseDeDatos **definido.** • cadenaConexion **definido.** • **clases definidas con:**

		○ idTipo **definido .** ○ modificadorAccesoClase **definido .** ○ nombreClase **definido.** ○ coordenadaX **definido.** ○ coordenadaY. ○ ancho **definido.** ○ esAbstracta **definido.** ○ esEstatica **definido.** ○ **atributos definidos con:** ▪ nombreAtributo **definido.** ▪ modificadorAccesoAtributo **definido.** ▪ tipo **definido.** ▪ esVisibleEnUi **definido.** ▪ admiteValorNulo **definido.** ▪ valorPorDefecto **definido.** ▪ patronExpresionRegex **definido.** ▪ esRequerido **definido.** ○ **relaciones definidas con:** ▪ claseOrigen **definido.** ▪ multiplicidadFuente **definido.** ▪ claseDestino **definido.** ▪ multiplicidadDestino **definido.** ▪ tipoRelacion **definido.** ▪ coordenadaXInicial **definido.** ▪ coordenadaYInicial **definido.** ▪ coordenadaXFinal **definido.** ▪ coordenadaYFinal **definido.**
	Estado final	

Tabla 48. Cabecera CU 12.

13	**Nombre del caso de uso**	**Descargar código frontend**
	Actor	Usuario.
	Breve descripción	Permite descargar al usuario el código del frontend en formato .zip/.rar.
	Prioridad	A.
	Parámetros de entrada	• diagramaId.
	Estado inicial	Instancia de diagrama con: • nombreDiagrama **definido.** • modificadorAccesoDiagrama **definido.** • lenguajeBackEnd **definido.** • lenguajeFrontEnd **definido.** • baseDeDatos **definido.**

		• cadenaConexion **definido.**
		• **clases definidas con**
		  ○ idTipo **definido.**
		  ○ modificadorAccesoClase **definido.**
		  ○ nombreClase **definido.**
		  ○ coordenadaX **definido.**
		  ○ coordenadaY **definido**.
		  ○ ancho **definido.**
		  ○ esAbstracta **definido.**
		  ○ esEstatica **definido.**
		  ○ **atributos definidos con:**
		  ▪ nombreAtributo **definido.**
		  ▪ modificadorAccesoAtributo **definido.**
		  ▪ tipo **definido.**
		  ▪ esVisibleEnUi **definido.**
		  ▪ admiteValorNulo **definido.**
		  ▪ valorPorDefecto **definido.**
		  ▪ patronExpresionRegex **definido.**
		  ▪ esRequerido **definido.**
		  ○ **relaciones definidas con**:
		  ▪ claseOrigen **definido.**
		  ▪ multiplicidadFuente **definido.**
		  ▪ claseDestino **definido.**
		  ▪ multiplicidadDestino **definido.**
		  ▪ tipoRelacion **definido.**
		  ▪ coordenadaXInicial **definido.**
		  ▪ coordenadaYInicial **definido.**
		  ▪ coordenadaXFinal **definido.**
		  ▪ coordenadaYFinal **definido.**

	Estado final	

Tabla 49. Cabecera CU 13.

14	Nombre del caso de uso	Subir código frontend a GitHub
	Actor	Usuario.
	Breve descripción	Permite subir al usuario el código del frontend a su GitHub en un repositorio con el nombre del proyecto.
	Prioridad	A.
	Parámetros de entrada	• diagramaId.
	Estado inicial	Instancia de diagrama con: • nombreDiagrama **definido.** • modificadorAccesoDiagrama **definido.** • lenguajeBackEnd **definido.**

		• lenguajeFrontEnd **definido.** • baseDeDatos **definido.** • cadenaConexion **definido.** • **clases definidas con:** ○ idTipo **definido.** ○ modificadorAccesoClase **definido.** ○ nombreClase **definido.** ○ coordenadaX **definido.** ○ coordenadaY. ○ ancho **definido.** ○ esAbstracta **definido.** ○ esEstatica **definido.** ○ **atributos definidos con:** ▪ nombreAtributo **definido.** ▪ modificadorAccesoAtributo **definido.** ▪ tipo **definido.** ▪ esVisibleEnUi **definido.** ▪ admiteValorNulo **definido.** ▪ valorPorDefecto **definido.** ▪ patronExpresionRegex **definido.** ▪ esRequerido **definido.** ○ **relaciones definidas con:** ▪ claseOrigen **definido.** ▪ multiplicidadFuente **definido.** ▪ claseDestino **definido.** ▪ multiplicidadDestino **definido.** ▪ tipoRelacion **definido.** ▪ coordenadaXInicial **definido.** ▪ coordenadaYInicial **definido.** ▪ coordenadaXFinal **definido.** ▪ coordenadaYFinal **definido.**
	Estado final	

Tabla 50. Cabecera CU 14.

15	Nombre del caso de uso	Subir código backend a GitHub
	Actor	Usuario.
	Breve descripción	Permite subir al usuario el código del backend a su GitHub en un repositorio con el nombre del proyecto.
	Prioridad	A.
	Parámetros de entrada	• diagramaId.
	Estado inicial	Instancia de diagrama con: • nombreDiagrama **definido.** • modificadorAccesoDiagrama **definido.**

		• lenguajeBackEnd **definido.** • lenguajeFrontEnd **definido.** • baseDeDatos **definido.** • cadenaConexion **definido.** • **clases definidas con:** o idTipo **definido.** o modificadorAccesoClase **definido.** o nombreClase **definido.** o coordenadaX **definido.** o coordenadaY **definido.** o ancho **definido.** o esAbstracta **definido.** o esEstatica **definido.** o **atributos definidos con:** ▪ nombreAtributo **definido.** ▪ modificadorAccesoAtributo **definido.** ▪ tipo **definido.** ▪ esVisibleEnUi **definido.** ▪ admiteValorNulo **definido.** ▪ valorPorDefecto **definido.** ▪ patronExpresionRegex **definido.** ▪ esRequerido **definido.** o **relaciones definidas con:** ▪ claseOrigen **definido.** ▪ multiplicidadFuente **definido.** ▪ claseDestino **definido.** ▪ multiplicidadDestino **definido.** ▪ tipoRelacion **definido.** ▪ coordenadaXInicial **definido.** ▪ coordenadaYInicial **definido.** ▪ coordenadaXFinal **definido.** ▪ coordenadaYFinal **definido.**
	Estado final	

Tabla 51. Cabecera CU 15.

Pantallas

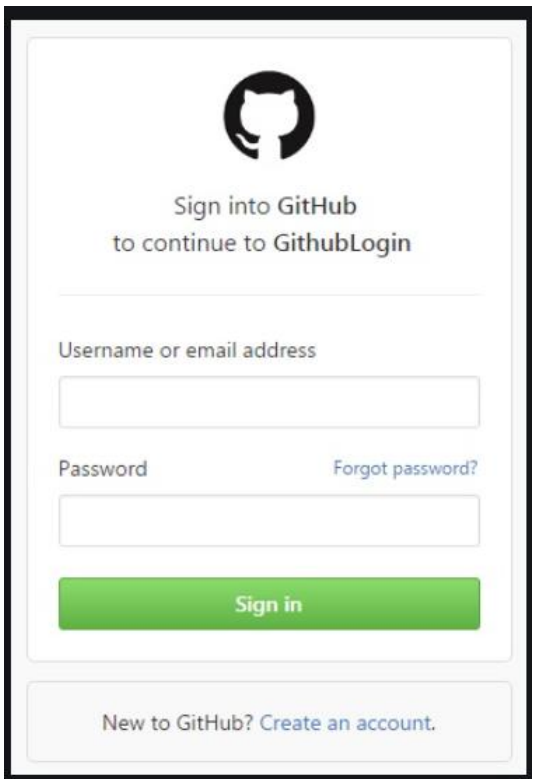


Figura 106. Login/Registrar GitHub.

1. Diagramas

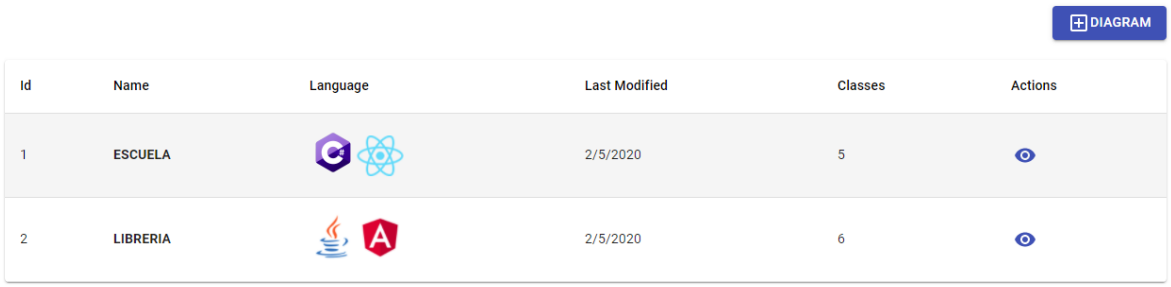


Figura 107. Listado de diagramas propios.

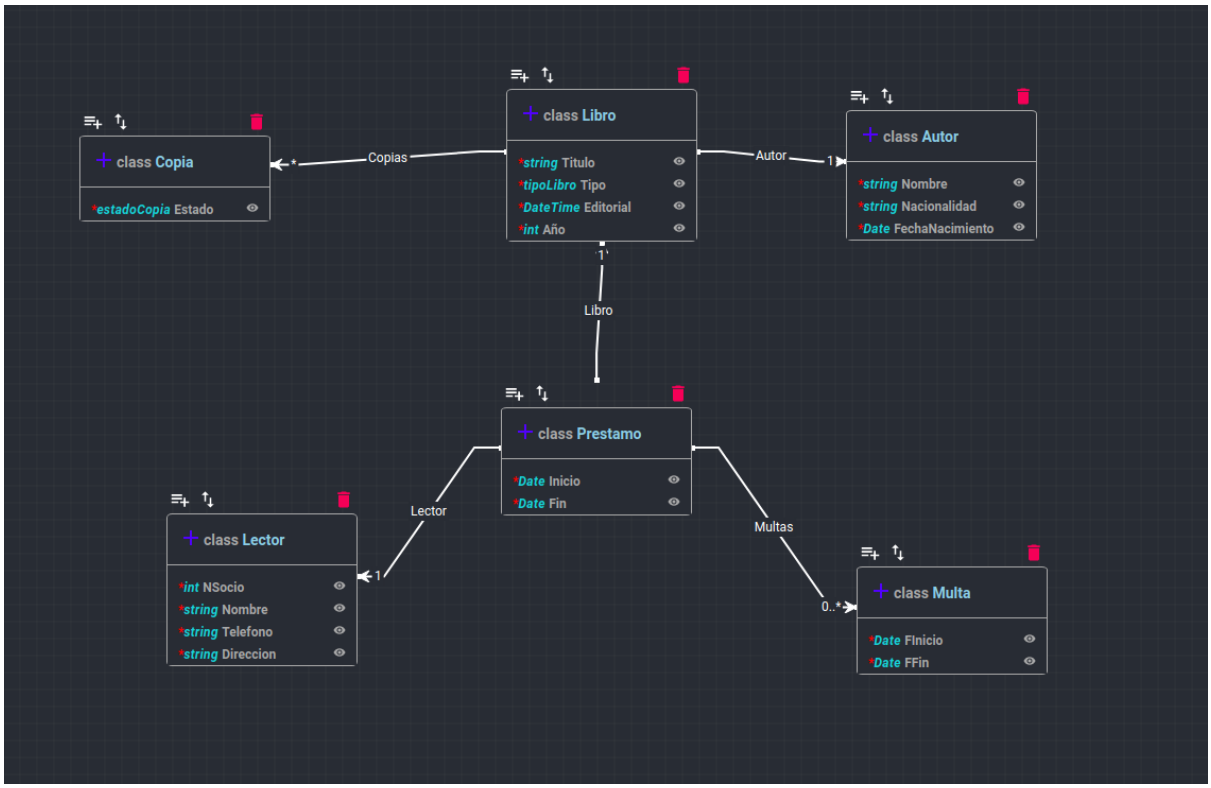


Figura 108. *Diagrama de clases completo, disponible para manipular.*

2. Clases

Figura 109. *Botón agregar clase a un diagrama.*

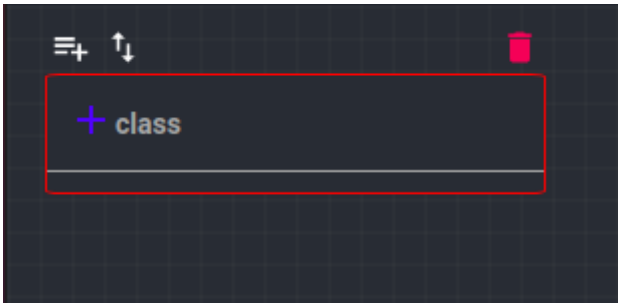


Figura 110. *Clase recién creada (bordes en rojo indicando que es inválida al no tener nombre).*

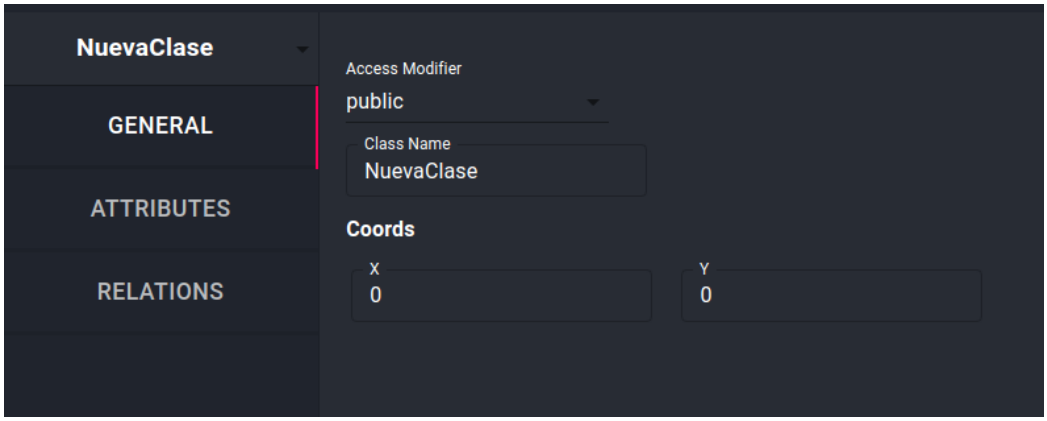


Figura 111. *Editar configuraciones básicas de una clase.*

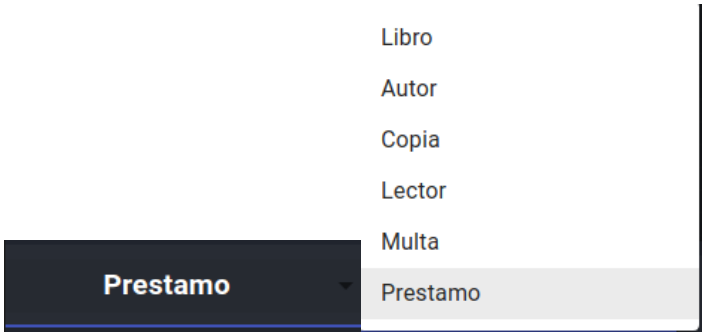


Figura 112. *Listado de clases de un diagrama para seleccionar.*

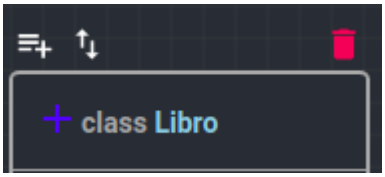


Figura 113. *Opciones de una clase (Agregar atributo, agregar relación o eliminar clase).*

3. Atributos

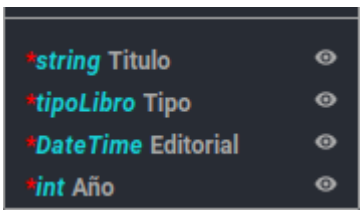


Figura 114. *Atributos de una clase de distintos tipos de datos, todos requeridos y visibles en la UI.*

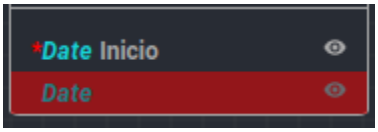


Figura 115. Atributo invalido en el diagrama (en rojo) por no poseer nombre.

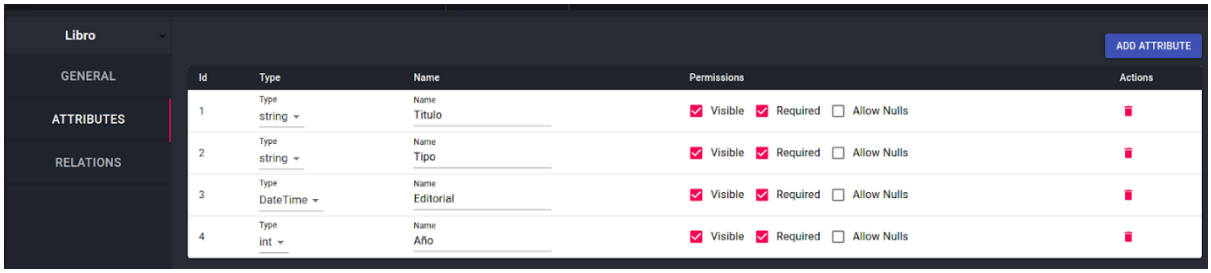


Figura 116. Listado de atributos de una clase para editar.

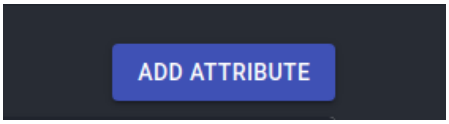


Figura 117. Botón para crear un nuevo atributo en una clase.

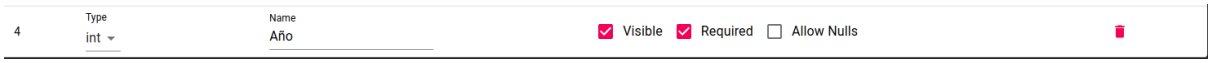


Figura 118. Edición de un atributo de una clase.

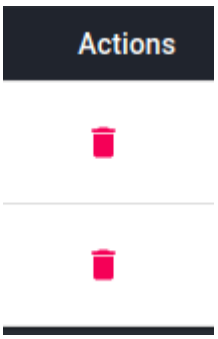


Figura 119. Botón para eliminar un atributo de una clase.

4. Relaciones

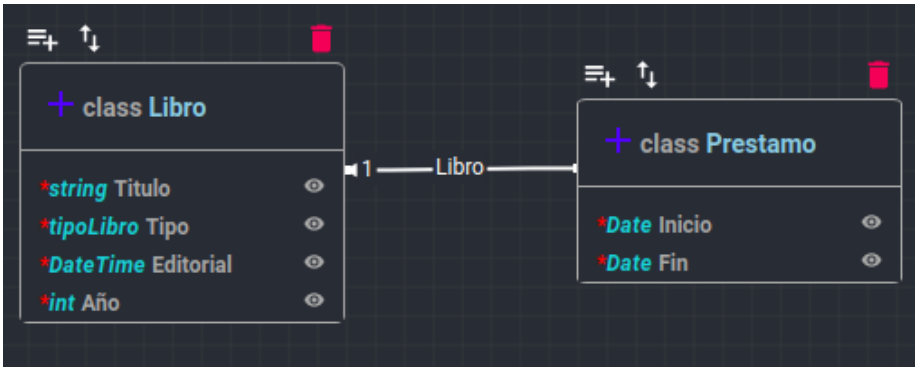


Figura 120. Relación dibujada en diagrama de préstamo a libro de tipo OneToOne.

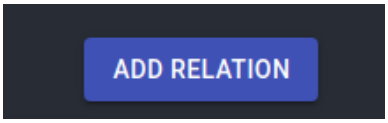


Figura 121. Botón para agregar una nueva relación a una clase.

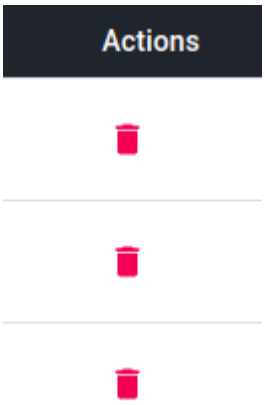


Figura 122. Botón para eliminar una relación de una clase.

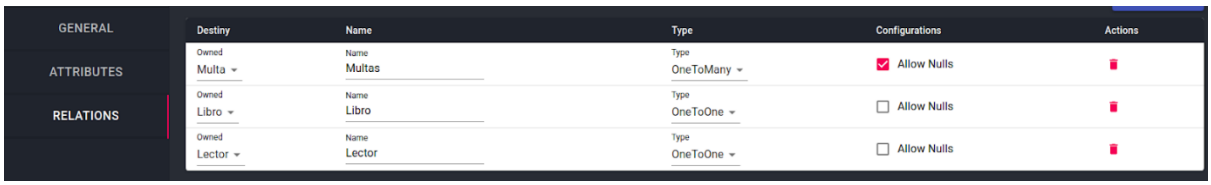


Figura 123. Listado de relaciones listo para manipular.

5. Reportes

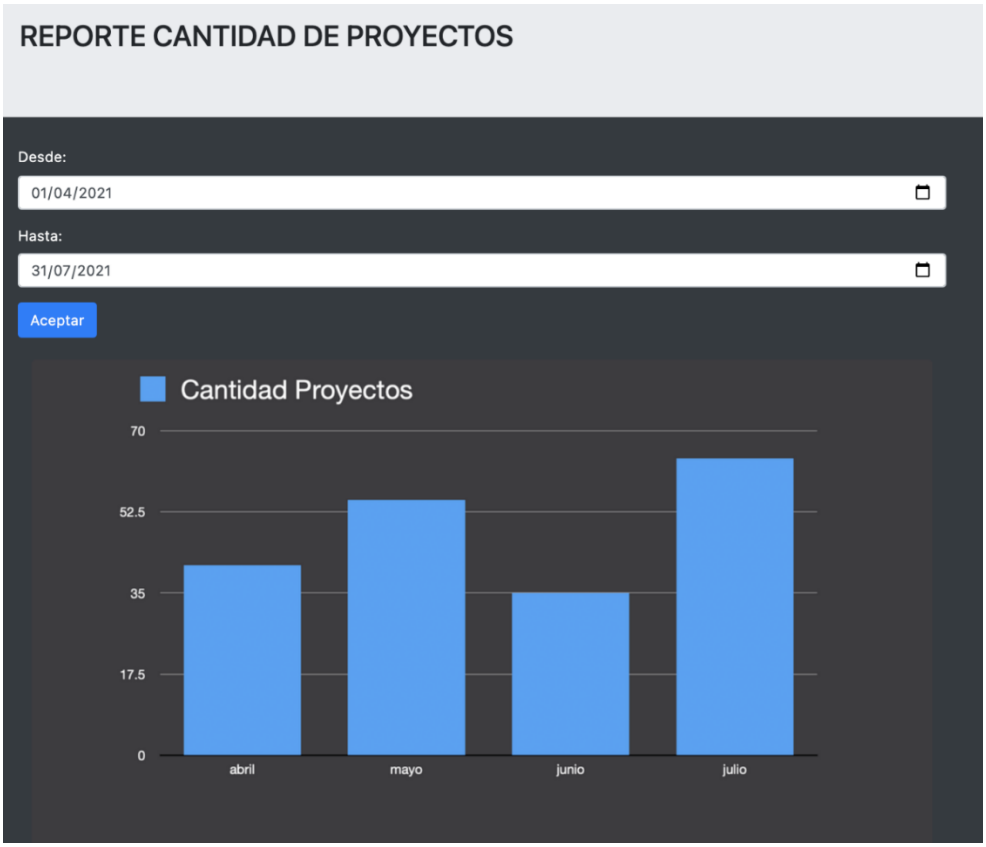


Figura 124. Reporte "Cantidad de proyectos".

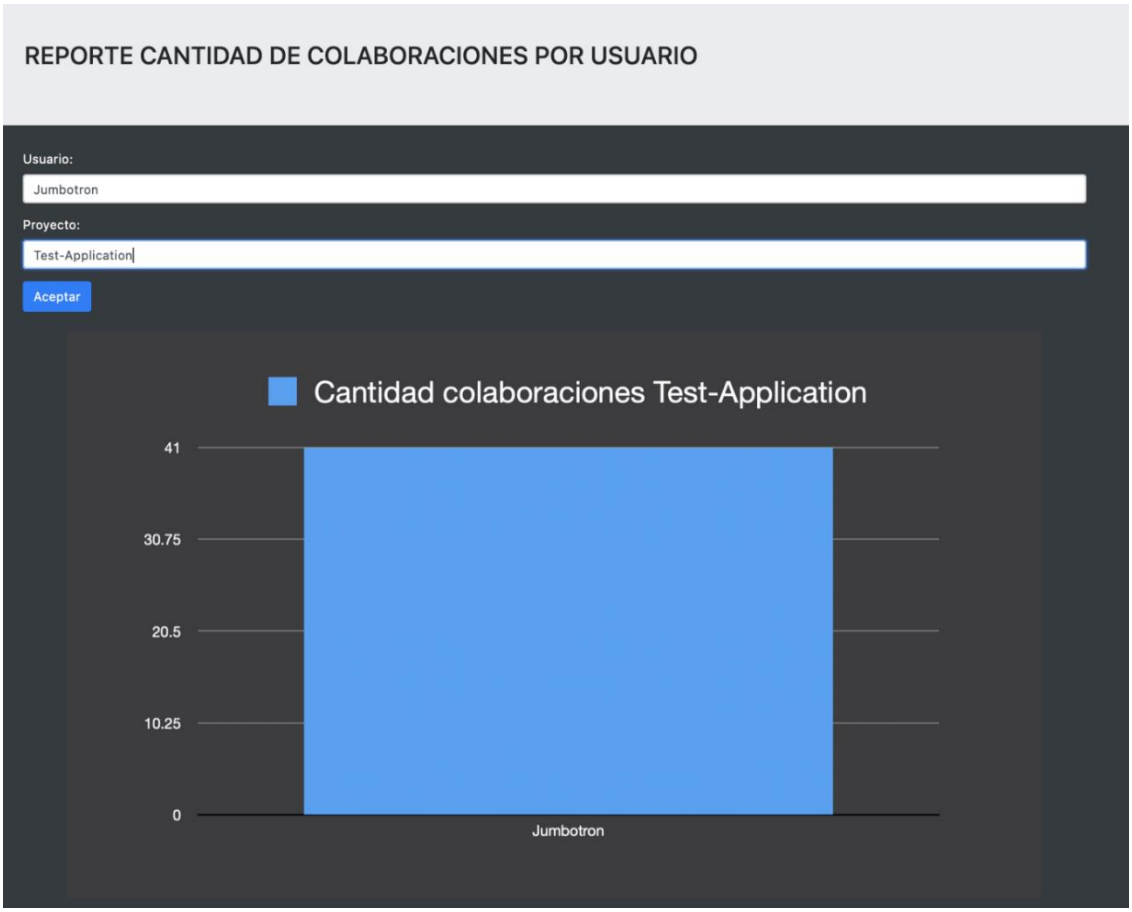


Figura 125. Reporte "Cantidad de colaboraciones por usuario".

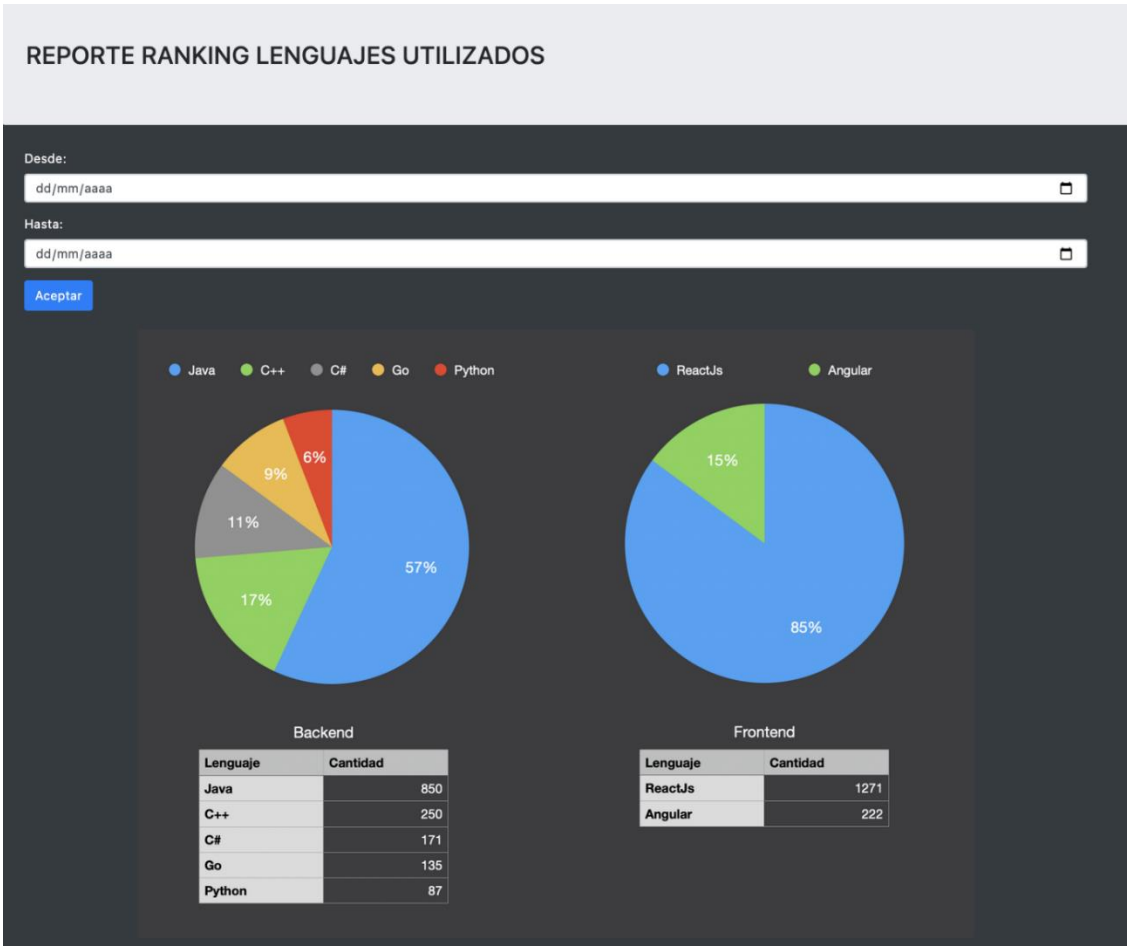


Figura 126. Reporte "Ranking Lenguajes Utilizados".

Modelo de datos

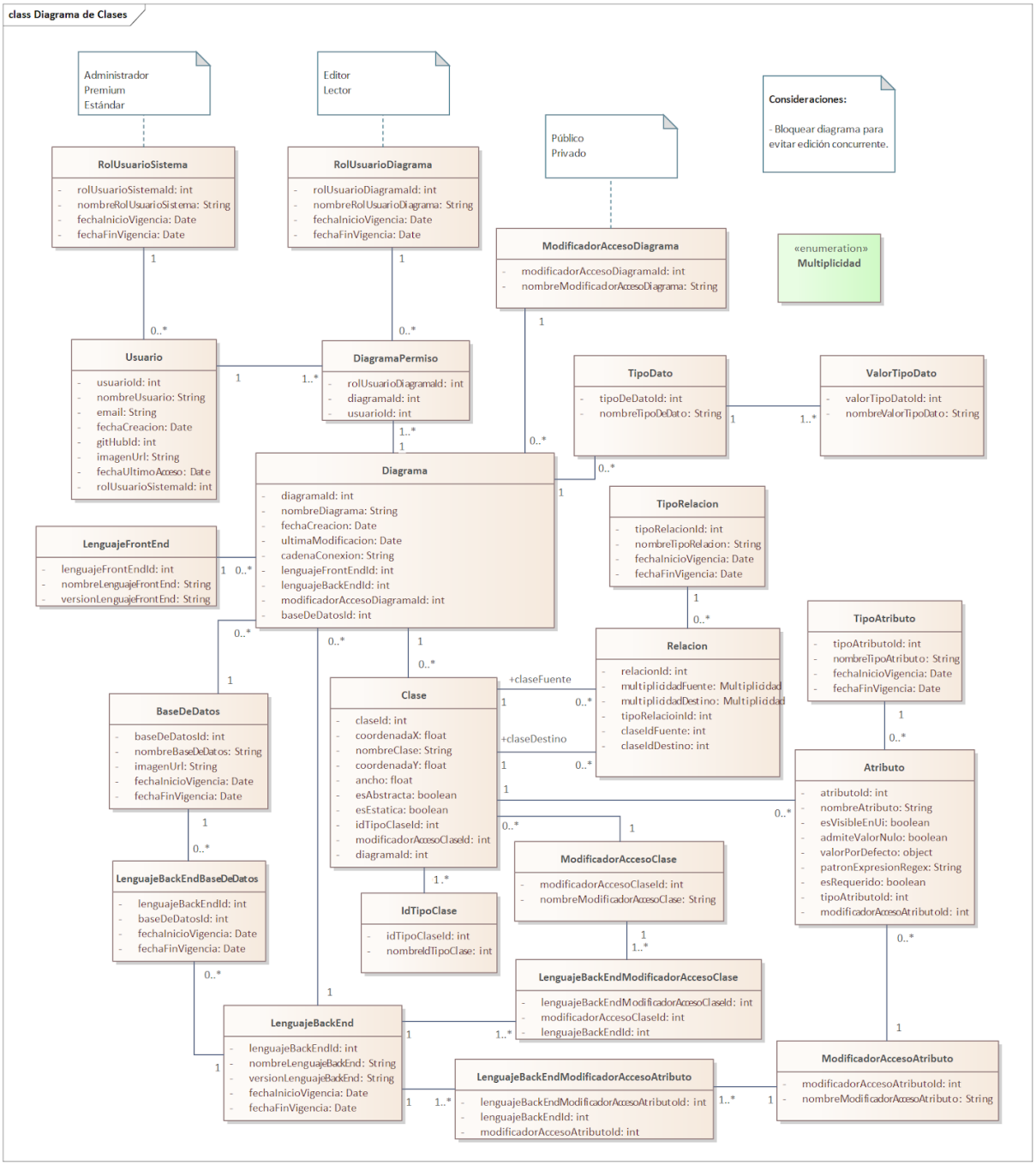


Figura 127. "Diagrama de clases del sistema".

Usuario:

- **usuarioId**
- **gitHubId**
- nombreUsuario: string
- email: string
- urlImg: string

- fechaCreacion: Date

Diagrama

- **diagramId: guid**
- nombreDiagrama: string
- modificadorAccesoDiagrama: ModificadorAccesoDiagrama (string enum)
- fechaCreacion: Date
- ultimaModificacion: Date
- clase: List<Clase>
- tiposDatos: List<TipoDato>
- lenguajeBackEnd: LenguajeBackEnd
- lenguajeFrontEnd: LenguajeFrontEnd
- BaseDeDatos: BaseDeDatos
- CadenaConexion: String

LenguajeBackEnd

- **lenguajeBackEndId: int**
- nombreLenguajeBackEnd: String

BaseDeDatos

- **baseDeDatosId**
- nombreBaseDeDatos: string
- imagenUrl: string
- List<LenguajeBackEnd>

DiagramaPermiso

- diagrama: Diagrama
- usuario: User
- rolUsuario: UserRol

Clase

- **claseId: int**
- idTipo: IdTipo
- modificadorAccesoClase:ModificadorAccesoClase
- nombreClase: string
- atributos: List<Atributo>
- coordenadaX: float
- coordenadaY: float
- ancho: float
- esAbstracta: boolean
- esEstatica: boolean

IdTipoClase

- **IdTipoClaseId: int**
- **IdTipoClasNombre: string**

Atributo

- **atributoId**
- nombreAtributo: string
- modificadorAccesoAtributo: ModificadorAccesoAtributo
- tipo: TipoAtributo
- esVisibleEnUi: boolean
- admiteValorNulo: boolean
- valorPorDefecto: object
- patronExpresionRegex: string
- esRequerido: boolean

Relación

- **relacionId**
- claseFuente: Clase
- multiplicidadFuente: Multiplicidad
- claseDestino: Clase
- multiplicidadDestino: Multiplicidad
- tipoRelacion: TipoRelacion
- coordenadaXInicial: float
- coordenadaYInicial: float
- coordenadaXFinal: float
- coordenadaYFinal: float

TipoDeDato:

- **tipoDeDatoId**
- nombreTipoDeDato: String
- datos: List<string>

DESARROLLO E IMPLEMENTACIÓN

Programación y documentación

El sistema fue desarrollado en dos repositorios de código distintos, donde uno corresponde a la aplicación backend y otro al frontend. El backend de la aplicación fue desarrollado en el lenguaje Java a través del framework para desarrollo web Spring boot, mientras que el frontend fue desarrollado en el lenguaje JavaScript a través de la librería React. El versionado de código se manejó a través de Git, usando como servidor el servicio en la nube GitHub y el estándar GitFlow para el manejo de ramas.

Otras tecnologías utilizadas:

- HTML 5
- CSS 3
- Javascript
- Typescript
- Material UI
- Java
- Spring boot
- Spring data
- Spring security
- Hibernate
- Swagger
- Microsoft SQL Server
- Docker
- Git

Endpoints Generados:

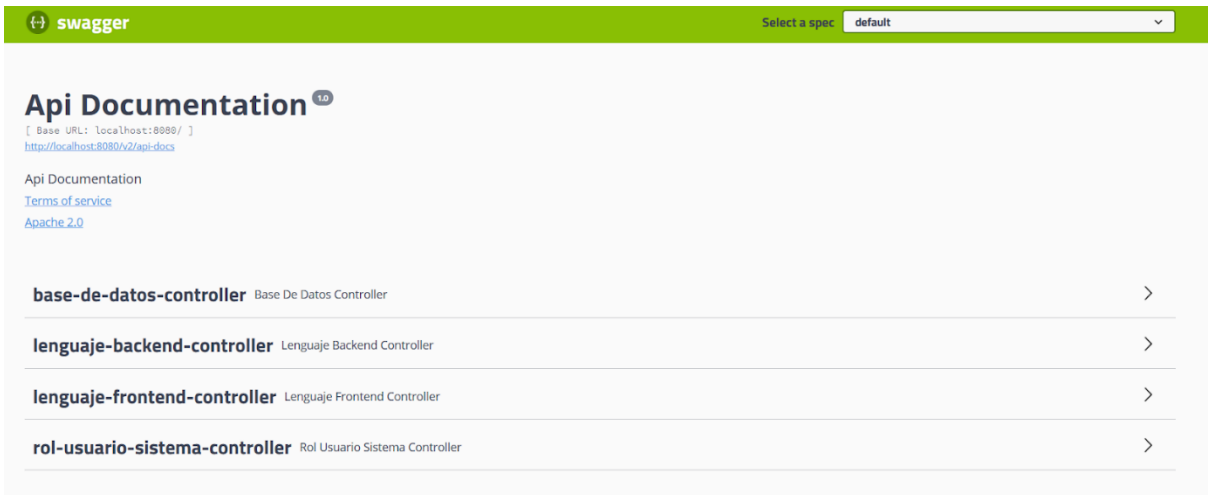


Figura 128. Interface de Swagger.

Figura 129. Endpoints Generados del sistema.

6. Documentación de Caso de Uso: ABM Diagrama

Nombre del CU	ABM Diagrama
Módulo seleccionado	Módulo de Gestión de diagramas.
Descripción breve del CU	El usuario podrá crear, editar y eliminar un diagrama.
Actor	Modelador.
Prioridad	A.
Programador a cargo	• Leandro Cruz. • Santiago Perez. • Agustín de Sautu Riestra.
Fecha inicio programación	1/08/2020.
Fecha finalización programación	10/09/2020.
Base de datos	SQL Server.
Clases involucradas	• Diagrama. • Clase. • LenguajeBackend. • LenguajeFrontend. • DiagramaPermiso. • Atributo. • Relacion. • Enum.
Historias de usuarios relacionadas	• HU005: Creación de diagrama. • HU006: Agregar clases a un diagrama. • HU007: Agregar atributos a una clase. • HU008: Agregar relaciones a una clase.
Casos de prueba relacionados	
Versionado	1.

Tabla 52. ABM Diagrama.

Código frontend:

```
export default function CreateDiagram() {
 const [diagram, setDiagram] = useState<postDiagram>({
   baseDeDatosId: 0,
   cadenaConexion: "",
   descripcion: "",
   lenguajeBackEndId: 0,
   lenguajeFrontEndId: 0,
   modificadorAccesoDiagramaId: 0,
```

```
  nombre: "",
});

const [lenguajeBackend, setLenguajesBackend] = useState<
  getBackendLenguajes[]
>([]);
const [lenguajeFrontend, setLenguajeFrontend] = useState<
  getFrontendLenguajes[]
>([]);

const [accessModifiers, setAccessModifiers] = useState<
  getAccessModifiersResponse[]
>([]);
const [databases, setDatabases] = useState<getDatabasesForBackendLenguaje>();
const history = useHistory();

useEffect(() => {
  const getInitialData = async () => {
    const backendLenguajes = await diagramServices.getBackendLenguajes();
    const frontendLenguajes = await diagramServices.getFrontendLenguajes();
    const accessModifiers = await diagramServices.getAccessModifiers();
    setLenguajesBackend(backendLenguajes);
    setLenguajeFrontend(frontendLenguajes);
    setAccessModifiers(accessModifiers);

    setDiagram((prevState) => ({
      ...prevState,
      lenguajeFrontEndId: frontendLenguajes[0].lenguajeFrontEndId,
      lenguajeBackEndId: backendLenguajes[0].lenguajeBackEndId,
      modificadorAccesoDiagramaId:
        accessModifiers[0].modificadorAccesoDiagramaId,
    }));
    getBatabasesForBackendLenguajes(backendLenguajes[0].lenguajeBackEndId);
  };
  getInitialData();
}, []);

const getBatabasesForBackendLenguajes = async (
  backendLenguajesId: number
) => {
  const databases = await diagramServices.getDatabases(backendLenguajesId);
  setDatabases(databases);
};

const changeBackendLenguaje = (e: React.ChangeEvent<{ value: unknown }>) => {
  const backendId = e.target.value as number;
  setDiagram((prevState) => ({ ...prevState, lenguajeBackEndId: backendId }));
  getBatabasesForBackendLenguajes(backendId);
};
const changeFrontendLenguaje = (e: React.ChangeEvent<{ value: unknown }>) => {
  const frotendId = e.target.value as number;
  setDiagram((prevState) => ({
    ...prevState,
    lenguajeFrontEndId: frotendId,
  }));
};
const changeName = (e: React.ChangeEvent<HTMLInputElement>) => {
  var value = firstLetterUpperCase(e.target.value);
  setDiagram((prevState) => ({ ...prevState, nombre: value }));
};
const changeConexionString = (e: React.ChangeEvent<HTMLInputElement>) => {
  var value = e.target.value;
  setDiagram((prevState) => ({ ...prevState, cadenaConexion: value }));
```

```
};

const changeDescription = (e: React.ChangeEvent<HTMLInputElement>) => {
  var value = e.target.value;
  setDiagram((prevState) => ({ ...prevState, descripcion: value }));
};
const changeAccessModifier = (accessModifierId: number) => {
  setDiagram((prevState) => ({
    ...prevState,
    modificadorAccesoDiagramaId: accessModifierId,
  }));
};

const changeDataBase = (
  e: React.ChangeEvent<HTMLInputElement>,
  value: string
) => {
  e.persist();
  const databaseId = e.target.value as unknown as number;
  setDiagram((prevState) => ({ ...prevState, baseDeDatosId: databaseId }));
};

const createDiagram = async () => {
  const response = await diagramServices.postDiagram(diagram);
  if (response.ok) {
    history.push(`/diagrams`);
  } else {
    alert("ocurrió un error");
  }
};
return (
  <>
    <Navbar></Navbar>
    <Container>
      <h1>New Diagram</h1>
      <Grid container>
        <h4>General Settings</h4>
        <Grid container item>
          <ButtonGroup aria-label="outlined primary button group">
            {accessModifiers.map((accessModifier) => (
              <Button
                key={accessModifier.modificadorAccesoDiagramaId}
                onClick={() =>
                  changeAccessModifier(
                    accessModifier.modificadorAccesoDiagramaId
                  )
                }
                color="primary"
                variant={
                  diagram.modificadorAccesoDiagramaId ===
                  accessModifier.modificadorAccesoDiagramaId
                    ? "contained"
                    : "outlined"
                }
              >
                {accessModifier.nombremodificadorAccesoDiagrama}
              </Button>
            ))}
          </ButtonGroup>
        </Grid>
        <Grid container item style={{ marginTop: "10px" }}>
          <TextField
            size="small"
            id="name"
```

```
        type="text"
        variant="outlined"
        label="Diagram name"
        fullWidth
        InputLabelProps={{
          shrink: true,
        }}
        onChange={changeName}
        value={diagram.nombre}
      />
    </Grid>
    <Grid container item style={{ marginTop: "10px" }}>
      <TextField
        size="small"
        id="description"
        type="text"
        variant="outlined"
        label="Description"
        multiline
        fullWidth
        minRows={3}
        InputLabelProps={{
          shrink: true,
        }}
        onChange={changeDescription}
        value={diagram.descripcion}
      />
    </Grid>
  </Grid>
  <Divider />

  <Grid container>
    <FormControl size="small">
      <h4>Backend Lenguaje</h4>

      <Select
        labelId="backendlengauje"
        label="Backend Lenguaje"
        id="demo-simple-select"
        value={diagram?.lenguajeBackEndId}
        onChange={changeBackendLenguaje}
      >
        {lenguajeBackend.map((lenguaje) => (
          <MenuItem
            key={lenguaje.lenguajeBackEndId}
            value={lenguaje.lenguajeBackEndId}
          >
            <Grid container justifyContent="center" alignItems="center">
              <img
                src={lenguaje.urlImagenLenguajeBackend}
                style={{
                  maxHeight: "50px",
                  borderRadius: "50%",
                }}
              ></img>
              <strong>{lenguaje.nombreLenguajeBackEnd}</strong>
            </Grid>
          </MenuItem>
        ))}
      </Select>
    </FormControl>
  </Grid>

  {diagram.lenguajeBackEndId !== 0 ? (
```

```
    <Grid container>
      <h4>Database</h4>
      <Grid
        container
        justifyContent="flex-start"
        alignItems="center"
        direction="row"
      >
        <FormControl component="fieldset">
          <RadioGroup
            row
            aria-label="databases"
            name="databases"
            value={diagram.baseDeDatosId}
            onChange={changeDataBase}
          >
            {databases?.baseDeDatos.map((x) => (
              <FormControlLabel
                value={x.baseDeDatosId}
                key={x.baseDeDatosId}
                control={<Radio color="primary" />}
                label={
                  <div style={{ display: "grid" }}>
                    <img
                      src={x.imagenUrl}
                      style={{
                        maxHeight: "30px",
                      }}
                    ></img>
                    <strong>{x.nombreBaseDeDatos}</strong>
                  </div>
                }
                labelPlacement="end"
              />
            ))}
          </RadioGroup>
        </FormControl>
      </Grid>
      <Grid container>
        <TextField
          style={{ marginTop: "10px" }}
          size="small"
          id="attr-name"
          type="text"
          variant="outlined"
          label="Conexion string"
          fullWidth
          InputLabelProps={{
            shrink: true,
          }}
          onChange={changeConexionString}
          value={diagram.cadenaConexion}
        />
      </Grid>
    </Grid>
  ) : null}
  <Grid container>
    <Grid item>
      <h4>Front Lenguaje</h4>
    </Grid>
    <Grid container>
      <FormControl size="small">
        <Select
          labelId="demo-simple-select-label"
          label="Front Lenguaje"
```

```
            id="demo-simple-select"
            value={diagram.lenguajeFrontEndId}
            onChange={changeFrontendLenguaje}
          >
            {lenguajeFrontend.map((lenguaje) => (
              <MenuItem
                key={lenguaje.lenguajeFrontEndId}
                value={lenguaje.lenguajeFrontEndId}
              >
                <Grid container justifyContent="center" alignItems="center">
                  <img
                    src={lenguaje.urlImagenLenguajeFrontend}
                    style={{
                      maxHeight: "30px",
                      borderRadius: "50%",
                    }}
                  ></img>
                  <strong>{lenguaje.nombreLenguajeFrontEnd}</strong>
                </Grid>
              </MenuItem>
            ))}
          </Select>
        </FormControl>
      </Grid>
    </Grid>
    <Grid container>
      <Button
        onClick={async () => {
          await createDiagram();
        }}
        fullWidth
        style={{ marginTop: "10px" }}
        color="primary"
        variant="contained"
      >
        CREATE
      </Button>
    </Grid>
  </Container>
 </>
);
}
```

Código backend:

Diagram.java

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "diagrama")
public class Diagrama {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long diagramaId;
  private String nombre;
  private String descripcion;
  private LocalDate fechaInicioVigencia;
  private LocalDate fechaFinVigencia;
  private LocalDate fechaCreacion;
  private LocalDate ultimaModificacion;
  private String cadenaConexion;
```

```
  @ManyToOne()
  @JoinColumn(name = "fk_lenguaje_frontend")
  private LenguajeFrontEnd lenguajeFrontEnd;

  @ManyToOne()
  @JoinColumn(name = "fk_lenguaje_backend")
  private LenguajeBackEnd lenguajeBackEnd;

  @ManyToOne()
  @JoinColumn(name = "fk_modificador_acceso_diagrama")
  private ModificadorAccesoDiagrama modificadorAccesoDiagrama;

  @ManyToOne()
  @JoinColumn(name = "fk_base_de_datos")
  private BaseDeDatos baseDeDatos;

  @OneToMany(cascade = ALL)
  @JoinColumn(name = "fk_diagrama")
  private List<DiagramaPermiso> diagramaPermisos = new ArrayList<>();

  @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
  @JoinColumn(name = "fk_diagrama")
  private List<Enum> enums = new ArrayList<>();

  @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
  @JoinColumn(name = "fk_diagrama")
  private List<Clase> clases = new ArrayList<>();
}
```

DiagramaController.java

```
1   @RestController
2   @RequestMapping("api/v1/diagramas")
3   @AllArgsConstructor
4   public class DiagramaController {
5
6       protected IDiagramaService iDiagramaService;
7
8       @PostMapping("/")
9       public ResponseEntity<DiagramaResponseDTO> create(@Valid @RequestBody DiagramaRequestDTO diagramaRequestDTO)
10      throws LenguajeFrontendDontFoundException, BaseDeDatosDontFoundException,
11              LenguajeBackendDontFoundException, ModificadorAccesoDiagramaDontFoundException
12      {
13          try {
14              return new ResponseEntity<>(iDiagramaService.create(diagramaRequestDTO), HttpStatus.CREATED);
15          }
16          catch (Exception e){
17              throw e;
18          }
19      }
20  }
21
```

DiagramService.java

```java
@Service
@AllArgsConstructor
public class DiagramaService implements IDiagramaService{

    protected DiagramaRepository diagramaRepository;
    protected LenguajeFrontendRepository lenguajeFrontendRepository;
    protected LenguajeBackendRepository lenguajeBackendRepository;
    protected BaseDeDatosRepository baseDeDatosRepository;
    protected ModoficadorAccesoDiagramaRepository modificadorAccesoDiagramaRepository;

    @Override
    public DiagramaResponseDTO create(DiagramaRequestDTO diagramaRequestDTO) throws
    BaseDeDatosDontFoundException, LenguajeFrontendDontFoundException,
    LenguajeBackendDontFoundException, ModificadorAccesoDiagramaDontFoundException
    {
        DiagramaResponseDTO diagramaResponseDTO;
        Diagrama diagrama;

        diagrama = MapperDiagrama.toDiagram(diagramaRequestDTO);
        diagrama.setFechaCreacion(LocalDate.now());

        Optional<BaseDeDatos> baseDeDatos = baseDeDatosRepository
        .findById(diagramaRequestDTO.getBaseDeDatosId());

        Optional<LenguajeFrontEnd> lenguajeFrontEnd = lenguajeFrontendRepository
        .findById(diagramaRequestDTO.getLenguajeFrontEndId());

        Optional<LenguajeBackEnd> lenguajeBackend = lenguajeBackendRepository
        .findById(diagramaRequestDTO.getLenguajeBackEndId());

        Optional<ModificadorAccesoDiagrama> modificadorAccesoDiagrama = modificadorAccesoDiagramaRepository
        .findById(diagramaRequestDTO.getModificadorAccesoDiagramaId());

        if(lenguajeFrontEnd.isEmpty()){
            throw new LenguajeFrontendDontFoundException(diagramaRequestDTO.getLenguajeFrontEndId());
        }
        if(lenguajeBackend.isEmpty()){
            throw new LenguajeBackendDontFoundException(diagramaRequestDTO.getLenguajeBackEndId());
        }

        if(modificadorAccesoDiagrama.isEmpty()){
            throw new ModificadorAccesoDiagramaDontFoundException(diagramaRequestDTO.getModificadorAccesoDiagramaId());
        }
        if(baseDeDatos.isEmpty()){
            throw new BaseDeDatosDontFoundException(diagramaRequestDTO.getBaseDeDatosId());
        }
        diagrama.setBaseDeDatos(baseDeDatos.get());
        diagrama.setLenguajeFrontEnd(lenguajeFrontEnd.get());
        diagrama.setLenguajeBackEnd(lenguajeBackend.get());
        diagrama.setModificadorAccesoDiagrama(modificadorAccesoDiagrama.get());

        diagramaRepository.save(diagrama);

        diagramaResponseDTO = MapperDiagrama.toDiagramaDTO(diagrama);

        return diagramaResponseDTO;
    }
}
```

Planificación de Capacitación

A continuación, se describe la estructura, personalizada especialmente para el sistema, del plan de capacitación para el uso del sistema METACOD.

1. Objetivo

1.1. Objetivos Generales

- Adquirir las herramientas básicas y necesarias para el uso del sistema y de sus funciones principales.
- Optimizar el uso apropiado y reducir dudas o dificultades en el uso del sistema.
- Instruir a usuarios en la práctica y uso del sistema y documentación para obtener conocimientos para capacitar a otros usuarios.

1.2. Objetivos Específicos

- Generar conocimiento y técnicas para la implementación del sistema en diversos entornos similares.
- Presentar el Marco legal, referido a la creación y publicación de Planes de Estudios y de contenidos digitales.
- Adquirir conocimientos para el mantenimiento autónomo de los sistemas de datos y archivos.

2. Alcance

Los Usuarios a quién está dirigido este Plan de Capacitación son aquellos que tendrán los roles principales de “Administrador” y de “Modelador”.

3. Contenidos de la Capacitación

3.1. Prerrequisitos

Los usuarios que tomen la capacitación deben estar familiarizados con herramientas de modelado básicas.

En el caso del administrador del sistema debe tener conocimientos específicos del entorno en donde se ejecuta actualmente el sistema y la infraestructura en donde se implementa.

3.2. Desarrollo

Para lograr los objetivos mínimos de nuestro plan de capacitación debemos desarrollar los siguientes temas:

- Realizar un reconocimiento de todas las funciones del sistema a nivel general, para todos los usuarios participantes.
- Exponer el Marco legal sobre la propiedad intelectual de los sistemas a modelar, describiendo los artefactos y limitando el modelado especificado por el sistema.
- Explicar y guiar a los usuarios que toman el rol de “Administrador” en la interpretación de la arquitectura general del sistema descrita en la documentación.

3.3. Duración de la Capacitación

La carga horaria será proporcional a la cantidad de usuarios que se capaciten y sus Roles.

Para los usuarios “Administrador”: se realizará una capacitación de 3 horas por 30 personas con el siguiente temario:

- Introducción al sistema.
- Propósito del sistema.
- Alcance del sistema.
- Módulos del sistema.
    - Módulo de Roles, Usuarios y Login.
    - Módulo generador código Api en Java 8.
    - Módulo generador código Frontend React.
    - Módulo de Reportes.
- Seguridad.
- Informes.

Para los usuarios “Modelador”: se realizará una capacitación de 4 horas por 20 personas con el siguiente temario:

- Introducción al sistema.
- Propósito del sistema.
- Alcance del sistema.
- Módulos del sistema.
    - Módulo de Roles, Usuarios y Login.
    - Módulo de Gestión de diagramas.
    - Módulo exportador del código a un repositorio.
- Github.
- Orientación a objetos.
- Diagrama de objetos.

3.4. Modalidad

Para la capacitación de usuarios con rol “Administrador” se deberá utilizar el Manual del sistema o una simplificación de este con todas las funciones. El capacitador realizará una exposición sobre una presentación o realizar videos tutoriales con los procedimientos de las funciones. Los participantes podrán ver un procedimiento de las funciones críticas para su rol, y generar consultas que se resolverán en el momento o de manera virtual con un soporte del capacitador. Se puede realizar tanto de manera presencial como virtual.

Para la capacitación de usuarios que toman roles específicos de “Modelador”, se deberá utilizar el Manual del sistema o una simplificación de este con todas las funciones. El capacitador realizará una exposición, se apoyará sobre una presentación y presentará un entorno para realizar una práctica de las funciones críticas. El capacitador puede resolver problemas en el momento. Se puede realizar tanto de manera presencial como virtual.

3.5. Práctica y Evaluación

Las prácticas para el/los “Administrador” para la modalidad presencial, la evaluación será por medio de observación de parte del capacitador con el objetivo de lograr realizar todas las operaciones básicas y un reconocimiento de las funciones específicas.

Las prácticas de la capacitación de “Modelador” se evaluará por medio de un test de prueba práctica con objetivo de lograr todo un proceso funcional completo.

3.6. Actividades de la Capacitación

Actividad	Duración (Días)	Recurso
Realizar Cronograma de Capacitación	5	Líder de Proyecto / Analista
Realizar de simplificación de Manual de Usuario	2	Analista / Desarrollador Frontend / Líder de Proyecto
Elaboración de Temario de Capacitación	5	Líder de Proyecto / Analista
Instalación de software para la grabación y edición de videos para usuario Administrador	0,25	Líder de Proyecto
Diseño de videos tutoriales para usuario Administrador	0,75	Desarrollador Frontend / Desarrollador Backend / Líder de Proyecto
Grabación de videos tutoriales para usuario Administrador	5	Tester / Desarrollador Frontend
Diseño de documento de "Preguntas Frecuentes"	3	Analista
Elaboración de Temario de prueba de funciones básicas de usuario Administrador	3	Líder de Proyecto
Elaboración de Temario de prueba de proceso funcional de usuario Modelador	3	Desarrollador Frontend / Desarrollador Backend

Tabla 53.. Actividades de la Capacitación

(Anexo 2: Diagrama de Tiempos)

Planificación, Ejecución y Documentación de pruebas

A continuación, se presentarán las pruebas que se llevarán a cabo para validar el funcionamiento del sistema.

4. Objetivos

El objetivo que se pretende alcanzar con el siguiente plan de pruebas es detectar fallas en los módulos del sistema, verificando que cada uno de ellos efectúe las funciones correspondientes y, a su vez, que imposibilite llevar a cabo las tareas en caso de utilizar datos erróneos.

Además, resulta de suma importancia validar el rendimiento de las funciones del sistema, en relación a los accesos a la base de datos y la complejidad de procesamiento. Para tal fin, fue utilizada la herramienta JMeter, mediante la cual es posible realizar pruebas de rendimiento y de carga.

5. Alcances

Con respecto a los alcances del plan de prueba, se efectuarán distintos tipos de pruebas sobre diferentes módulos del sistema, los cuales se detallan a continuación.

5.1. Tipos de prueba

- 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 usuario.

5.2. Módulos a probar

- Gestión de Diagrama.
- Gestión de Roles y Usuarios.
- Generación de código Backend en Java.
- Generación de código Frontend en React.
- Gestión de exportación a repositorio GitHub.
- Reportes.

6. Ejecución

Nº de Caso de Prueba	Nombre	Resultado	Tipo de Prueba
CP001	Ingreso de nombre de clase vacío	✗	Prueba de validación
CP002	Ingreso de nombre de atributo de clase vacío	✗	Prueba de validación
CP003	Ingreso de dos atributos de clase con el mismo nombre	✗	Prueba de validación
CP004	Alta de Lenguaje Backend ya existente	✓	Prueba de lógica
CP005	Login de usuario mediante GitHub	✗	Prueba de lógica

CP005	Login de usuario mediante GitHub (Con medidas Correctivas)	✓	Prueba de lógica
CP006	Relación de composición con la misma clase tanto en origen como destino	✗	Prueba de lógica
CP007	Crear un nuevo diagrama	✓	Prueba de integración
CP008	Exportar código generado a partir de un diagrama vacío a GitHub	✗	Prueba de integración
CP008	Exportar código generado a partir de un diagrama vacío a GitHub	✓	Prueba de integración
CP009	Generar reporte de lenguajes backend más utilizados	✓	Prueba de integración
CP010	Consultas concurrentes de diagramas públicos	✗	Prueba de carga
CP011	Creación concurrente de clases en diagramas	✓	Prueba de carga
CP012	Solicitudes de logueo concurrentes	✓	Prueba de carga
CP013	Cerrar sesión y volver a la página anterior	✗	Prueba de seguridad
CP014	Intentar visualizar un diagrama ajeno privado	✗	Prueba de seguridad
CP015	Intentar generar un reporte sin tener rol de administrador	✓	Prueba de seguridad

Tabla 54. Casos de prueba a realizar.

7. Pruebas de validación de ingreso de datos

7.1. Objetivo

Validar el formato de los datos ingresados en los módulos principales del sistema y asegurar que no sea permitido completar una acción utilizando datos de entrada erróneos.

7.2. Alcance

Las pruebas de validación de datos buscarán comprobar que el tipo de datos ingresado corresponde al campo en el que se esté introduciendo, que los datos respeten el formato esperado para su adecuado procesamiento, que la longitud de los mismos sean la correcta, que no se admita duplicación de datos para evitar inconsistencias y problemas en la gestión de la base de datos, entre otros.

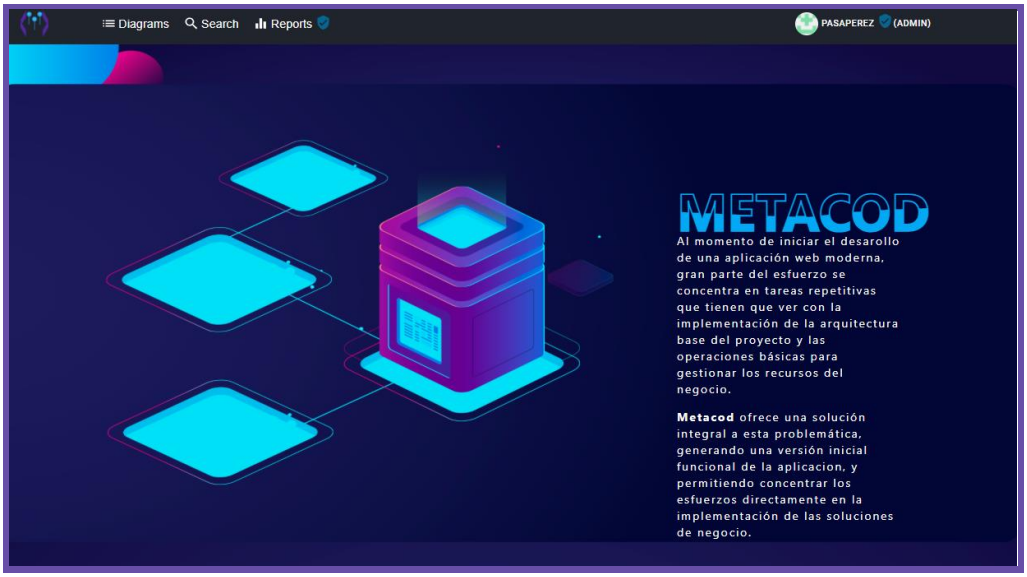
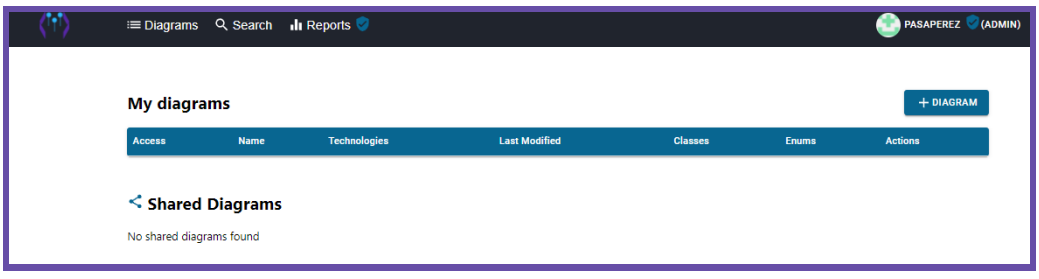
7.3. Realización

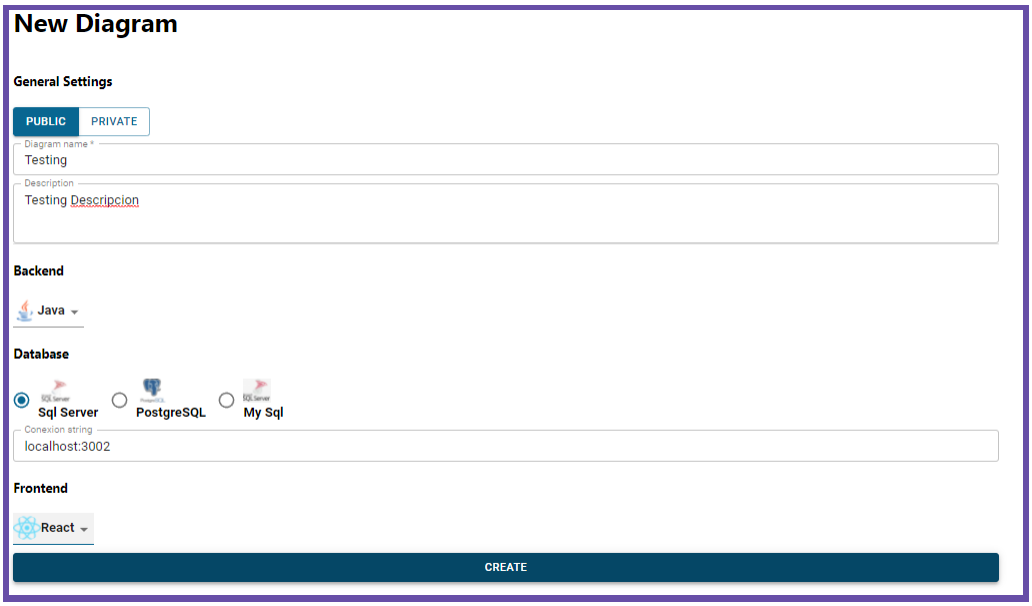
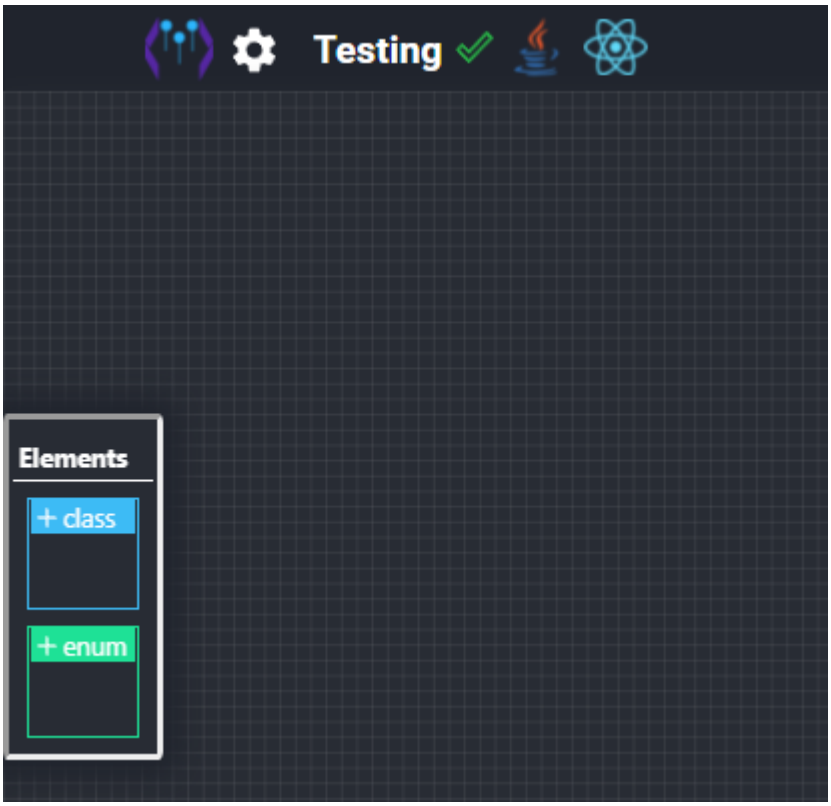
Estas pruebas se irán ejecutando de forma concurrente a la programación de las funcionalidades que requieren el ingreso de los datos validados para conducir una acción a partir de estos.

CP001 - Nombre de Clase vacío	
User story	HU003, HU005, HU006.
Fecha de ejecución	16/10/2021.
Actor	Modelador.
Objetivo	Verificar que el sistema no admita que se agregue a un diagrama una clase con nombre vacío.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”.
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	Validación de datos.
Medidas correctivas	Validar que el nombre de clase ingresado no pueda quedar vacío.

Nº de Paso	**Acción a realizar**	**Resultado Esperado**	**Resultado Obtenido**
01	Hacer click en el botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en el botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama.		

	(Captura de pantalla 03)		
04	Clickear el botón “Create”. *(Captura de pantalla 03)*		
05	Arrastrar elemento “Class” al canvas. *(Captura de pantalla 04)*	El elemento gráfico que representa a la clase aparece delineado en rojo y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 05)*	El elemento gráfico que representa a la clase aparece delineado en verde y a la derecha del nombre del diagrama se muestra un ícono de tick verde, indicando validación en la integridad del diagrama. *(Captura de pantalla 06)*

Nº de Captura	Captura de Pantalla
01	
02	

03	 New Diagram General Settings PUBLIC PRIVATE Diagram name * Testing Description Testing Descripcion Backend Java Database Sql Server PostgreSQL My Sql Conexion string localhost:3002 Frontend React CREATE
04	 Testing Elements + class + enum

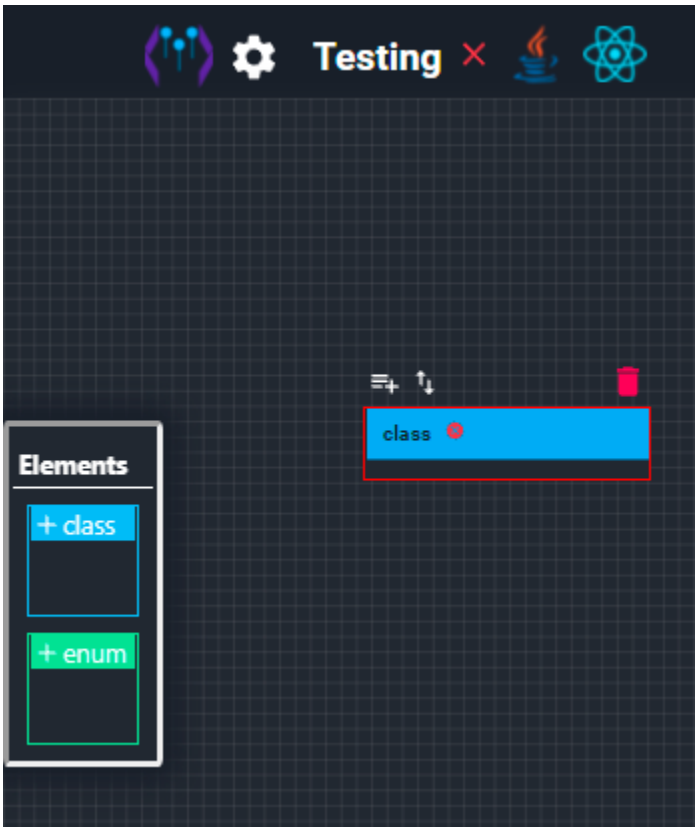
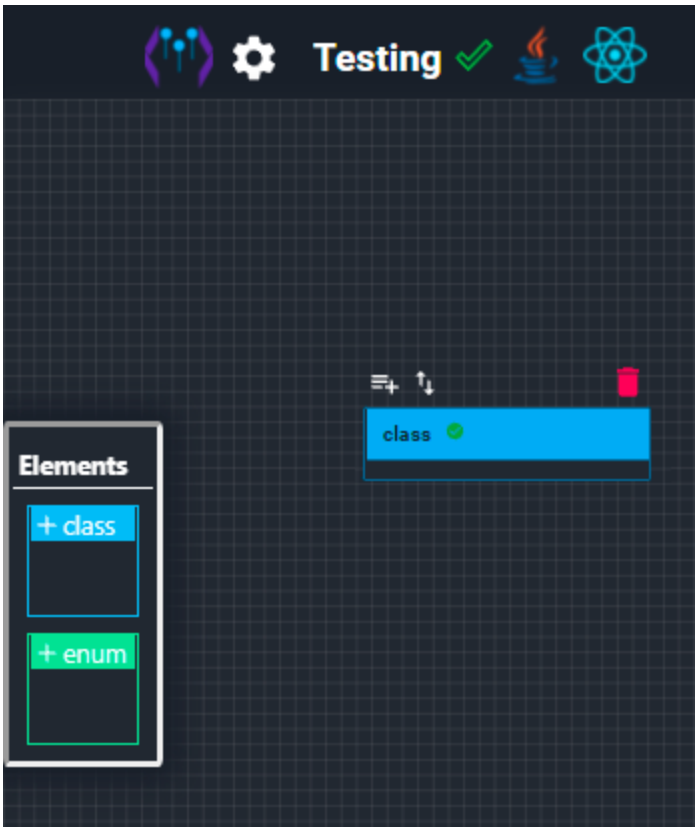
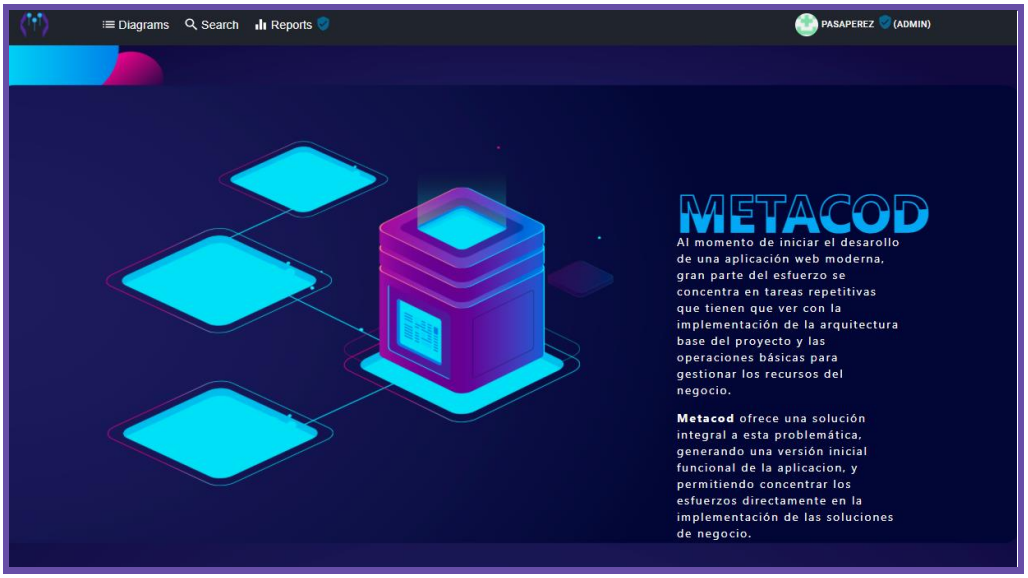
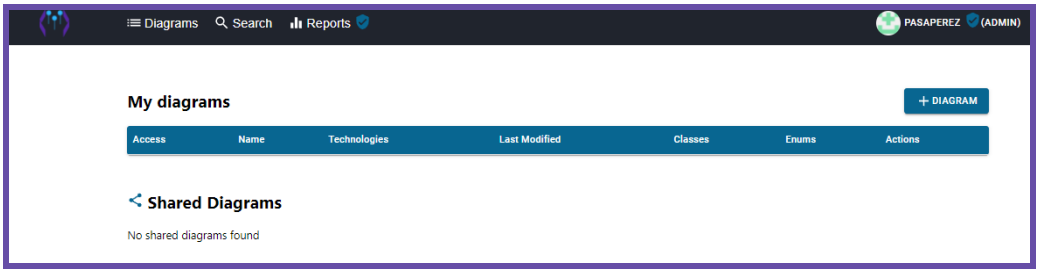
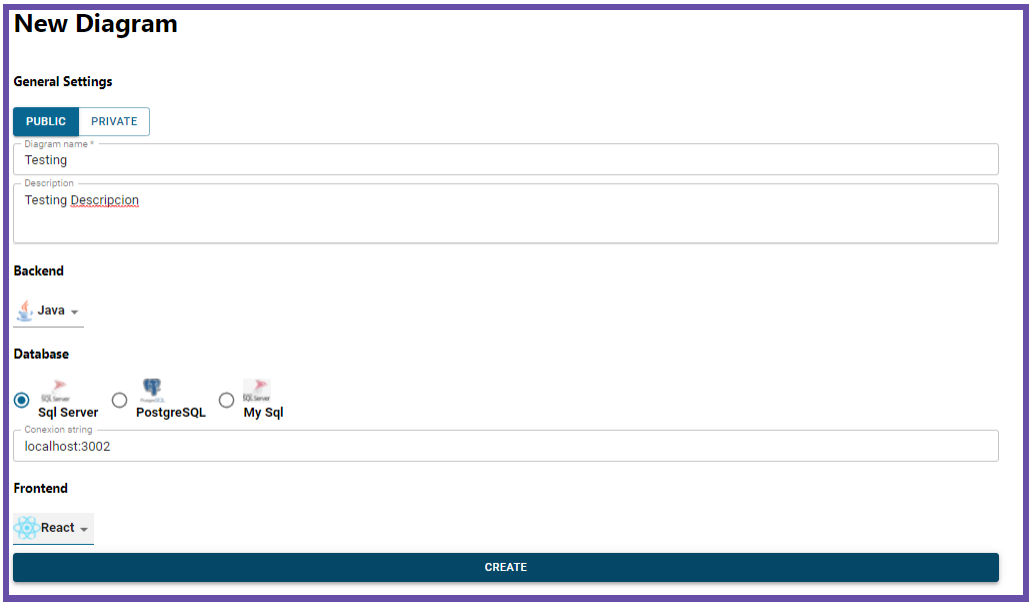
05	
06	

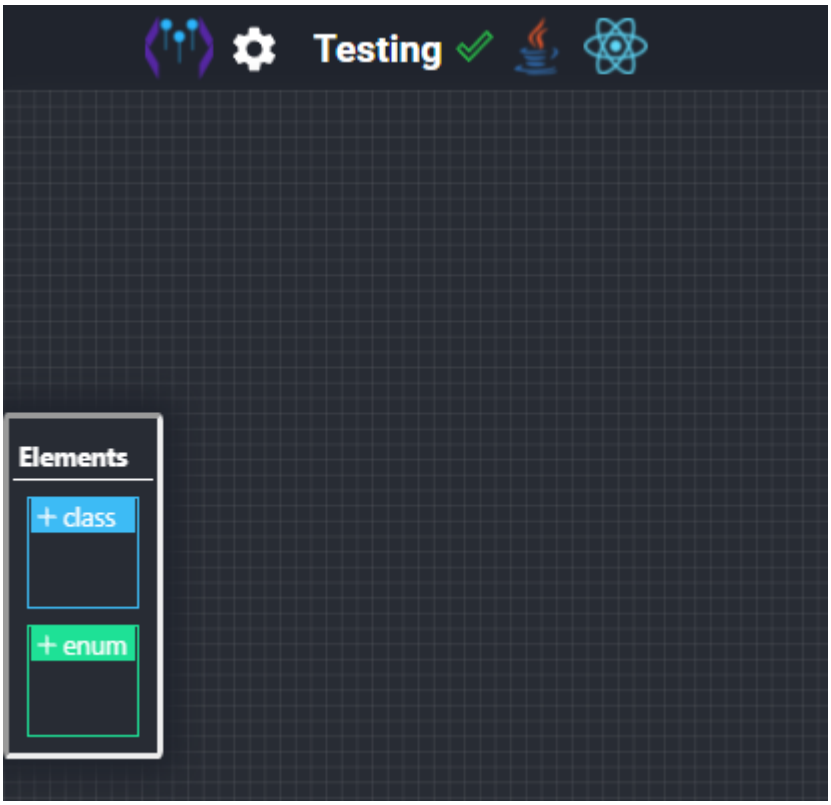
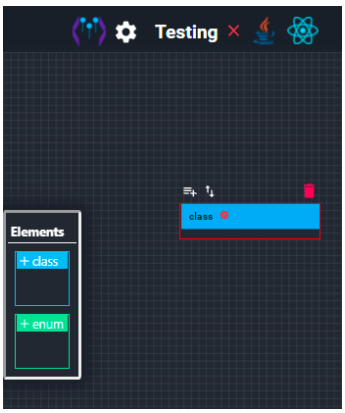
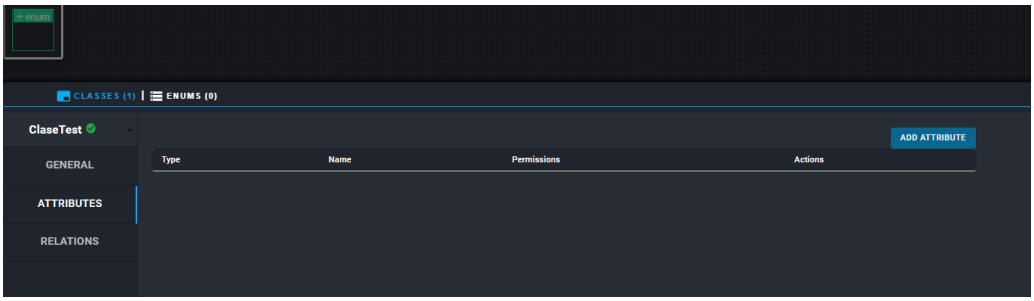
Tabla 55. CP003 Nombre de clase vacío.

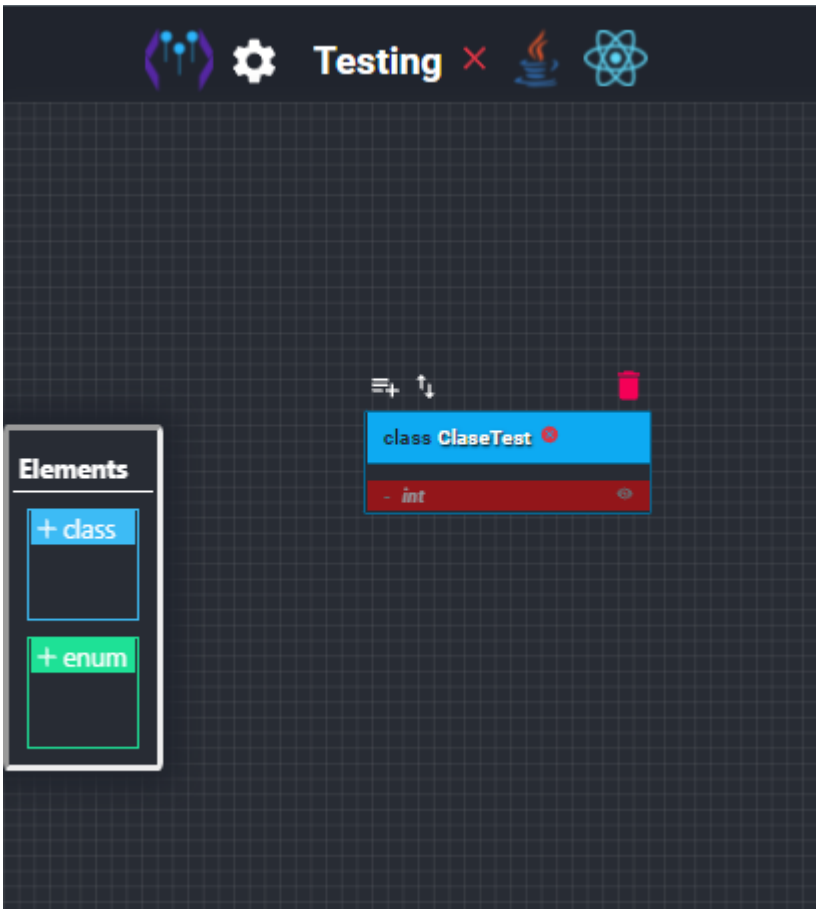
CP002 - Ingreso de nombre de atributo de clase vacío	
User story	HU003, HU005, HU006.
Fecha de ejecución	16/10/2021.
Actor	Modelador.
Objetivo	Verificar que el sistema no admita que se agregue a una clase un atributo cuyo nombre esté vacío.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”, nombreClase: “ClaseTest”.
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	Validación de datos.
Medidas correctivas	Validar que el nombre de atributo de clase ingresado no pueda quedar vacío.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en el botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en el botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear el botón “Create”. *(Captura de pantalla 03)*		
05	Arrastrar elemento “Class” al canvas.		

	(Captura de pantalla 04)		
06	Hacer doble click en la clase agregada al canvas.		
07	Ingresar “ClaseTest” como nombre de la clase.		
08	Hacer click en la pestaña “Attributes” del panel inferior.		
09	Hacer click en el botón “Add attribute”. *Captura de pantalla 06)*	El elemento gráfico que representa al atributo aparece remarcado en rojo y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 07)*	El elemento gráfico que representa al atributo no aparece remarcado y a la derecha del nombre del diagrama se muestra un ícono de tick verde, indicando validación en la integridad del diagrama. *(Captura de pantalla 08)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

04	 Testing Elements + class + enum
05	 Testing Elements + class + enum class
06	 CLASSES (1) ENUMS (0) ClaseTest GENERAL ATTRIBUTES RELATIONS ADD ATTRIBUTE Type Name Permissions Actions

07	 Testing Elements + class + enum class ClaseTest - int

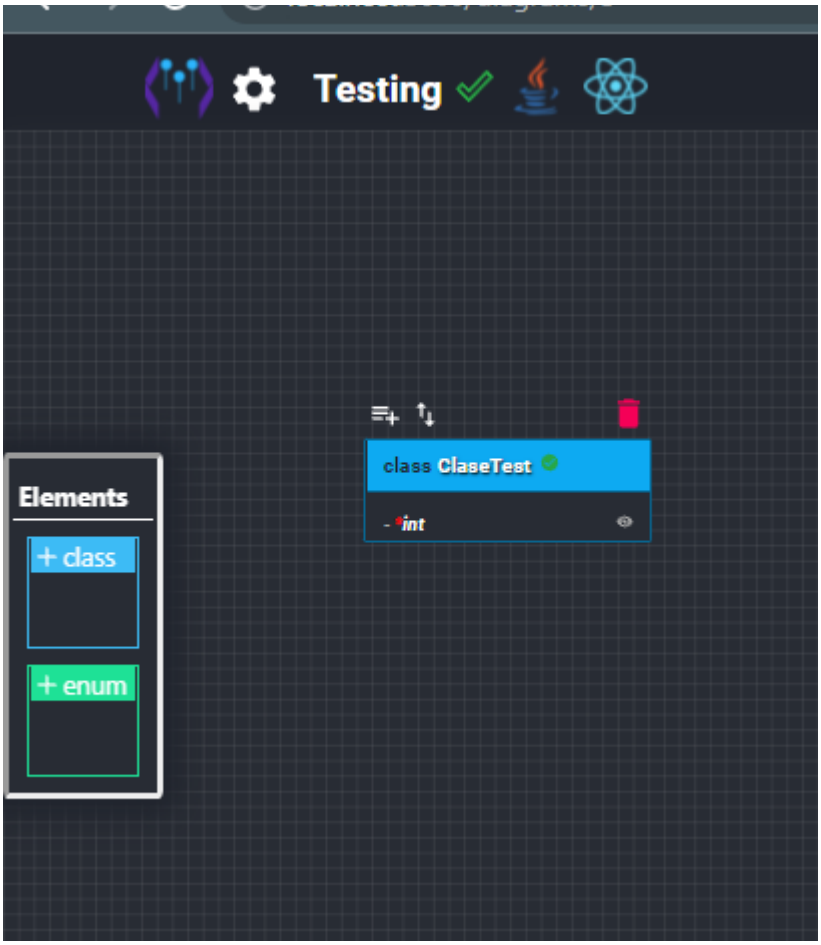
08	

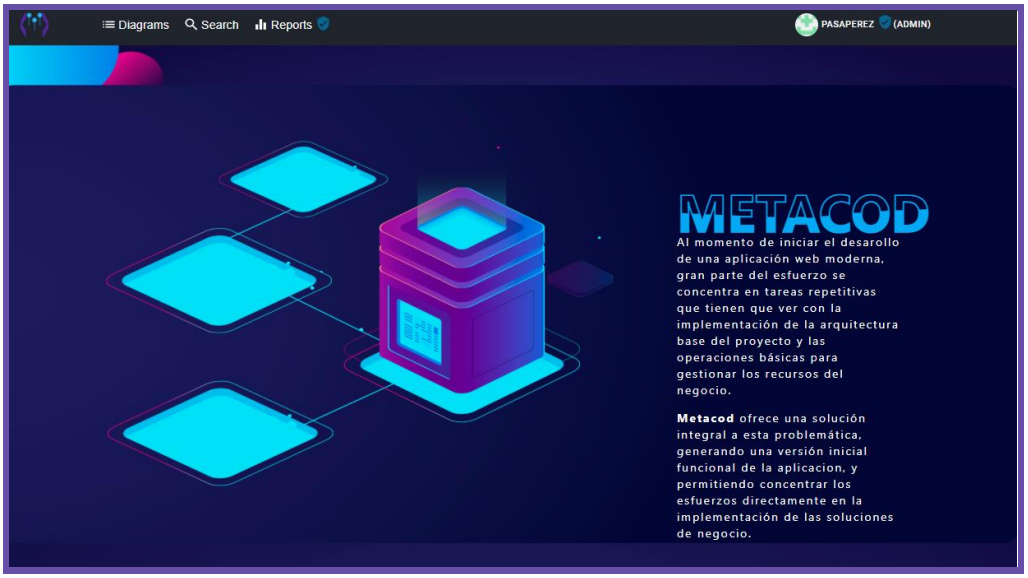
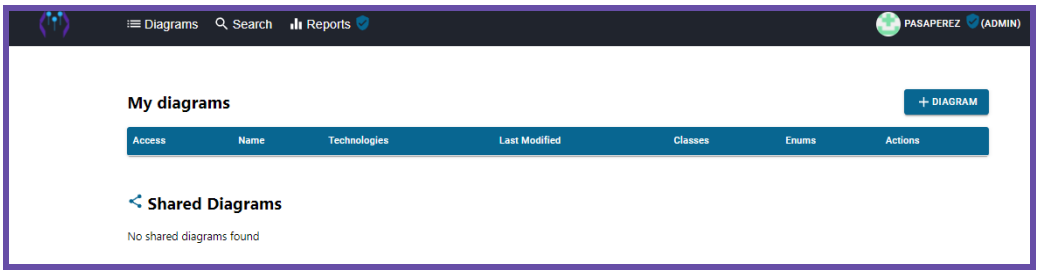
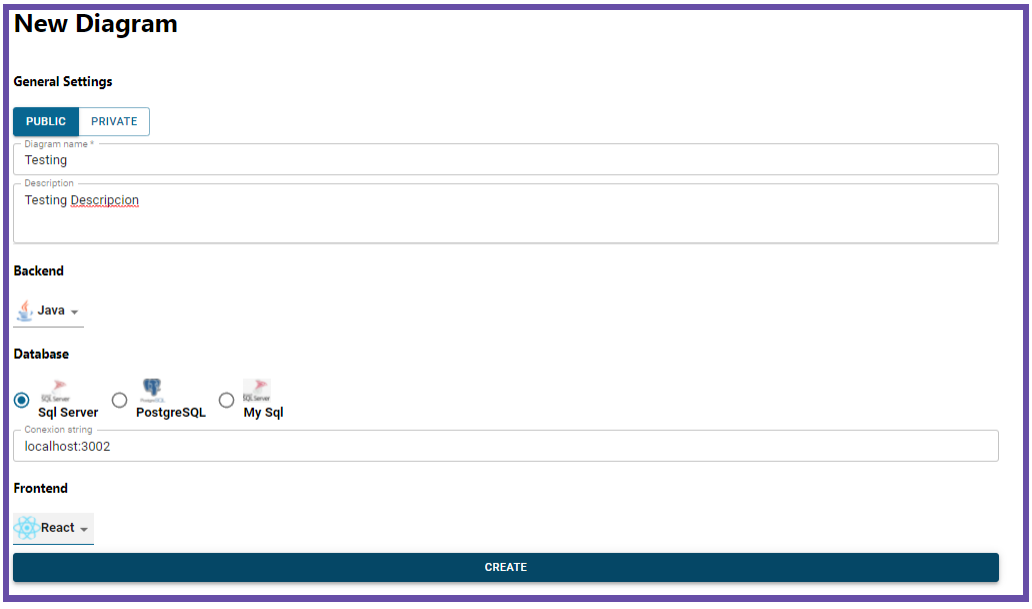
Tabla 56. CP002 Ingreso de nombre de atributo de clase vacío.

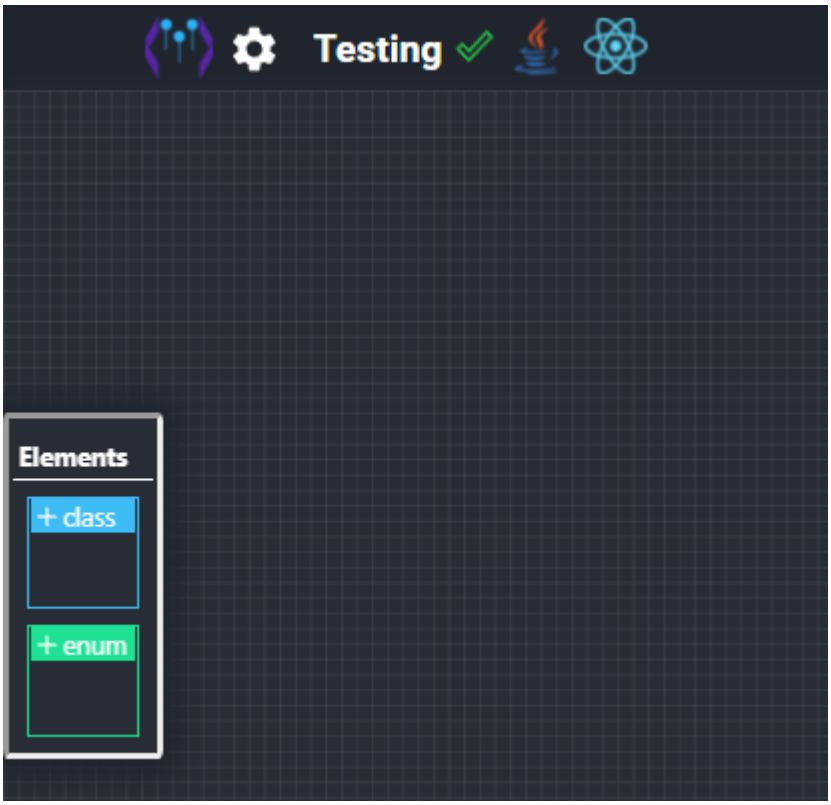
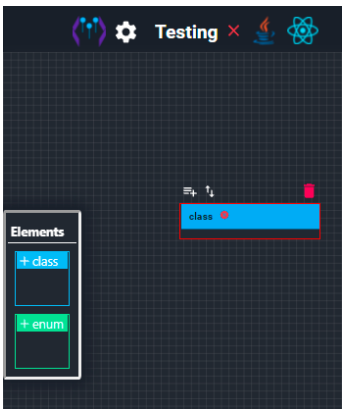
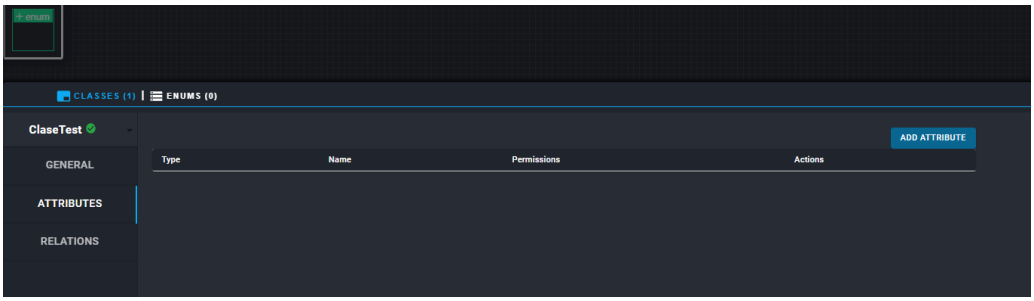
CP003 - Ingreso de dos atributos de clase con el mismo nombre	
User story	HU003, HU005, HU006, HU007.
Fecha de ejecución	18/10/2021.
Actor	Modelador.
Objetivo	Verificar que el sistema no admita que se agreguen a una clase dos atributos con el mismo nombre.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”, nombreClase: “ClaseTest”, nombreAtributo1= “Atributo1”, nombreAtributo2= “Atributo1”.

Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	Validación de datos.
Medidas correctivas	N/A.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en el botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en el botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear el botón “Create”. *(Captura de pantalla 03)*		
05	Arrastrar elemento “Class” al canvas. *(Captura de pantalla 04)*		
06	Hacer doble click en la clase agregada al canvas.		
07	Ingresar “ClaseTest” como nombre de la clase.		
08	Hacer click en la pestaña “Attributes” del panel inferior.		
09	Hacer click en el botón “Add attribute”. *Captura de pantalla 06)*		
10	Clickear sobre el campo “Name		

	Attribute” e introducir el nombre.		
11	Hacer click en el botón “Add attribute”. *Captura de pantalla)*		
12	Clickear sobre el campo “Name Attribute” e introducir el mismo nombre que en el paso 10.	En el elemento gráfico que representa a la clase, aparece un ícono rojo con una cruz del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 07)*	En el elemento gráfico que representa a la clase, aparece un ícono verde con un tick del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de tick verde, indicando que hay integridad en el diagrama. *(Captura de pantalla 08)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

04	 Testing Elements + class + enum
05	 Testing Elements + class + enum class
06	 CLASSES (1) ENUMS (0) ClaseTest GENERAL ATTRIBUTES RELATIONS ADD ATTRIBUTE Type Name Permissions Actions

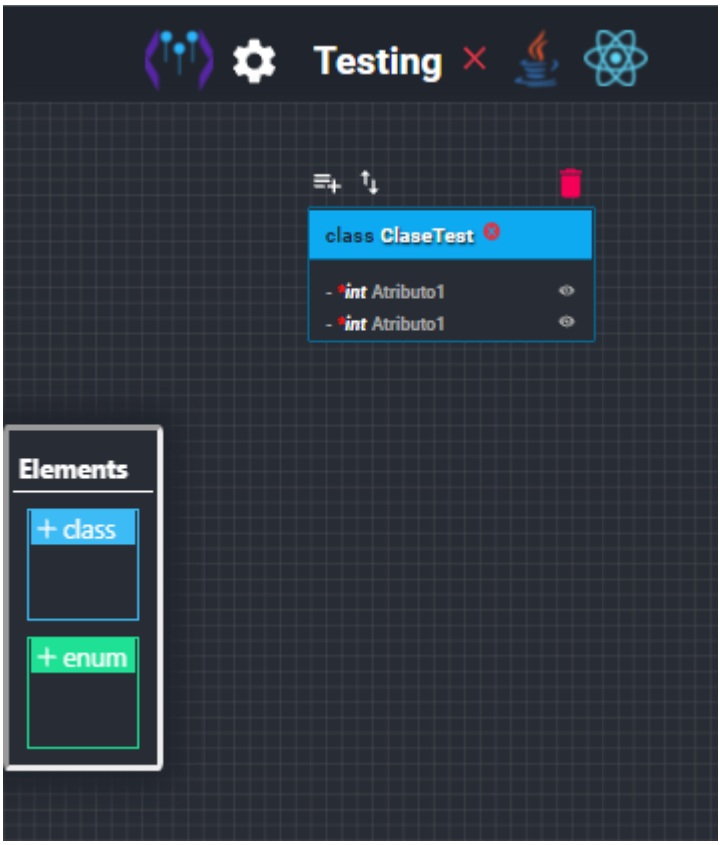
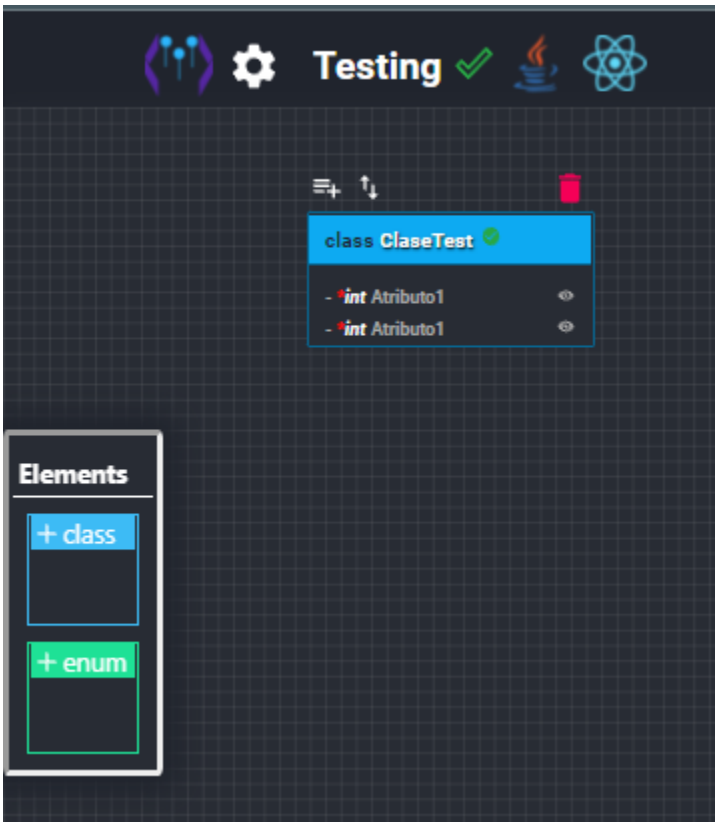
07	
08	

Tabla 57. CP003 Ingreso de dos atributos de clase con el mismo nombre.

8. Pruebas de lógica de los módulos principales

8.1. Objetivo

Detectar fallas en el funcionamiento de cada uno de los componentes unitarios del sistema y verificar que cada módulo realiza las funciones que corresponden de forma correcta.

8.2. Alcance

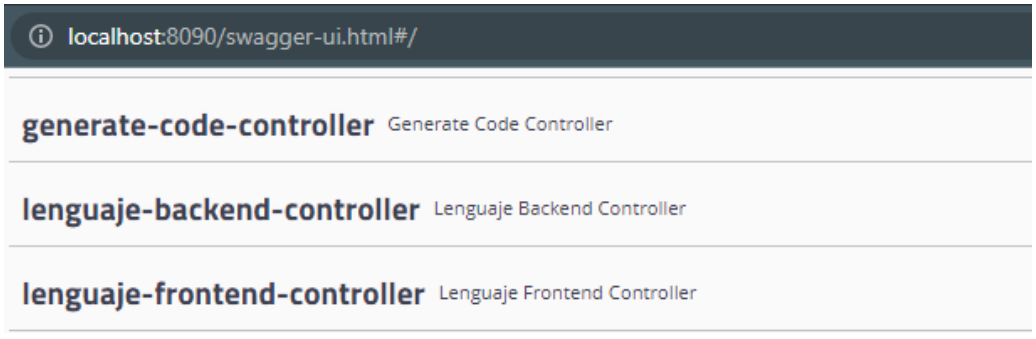
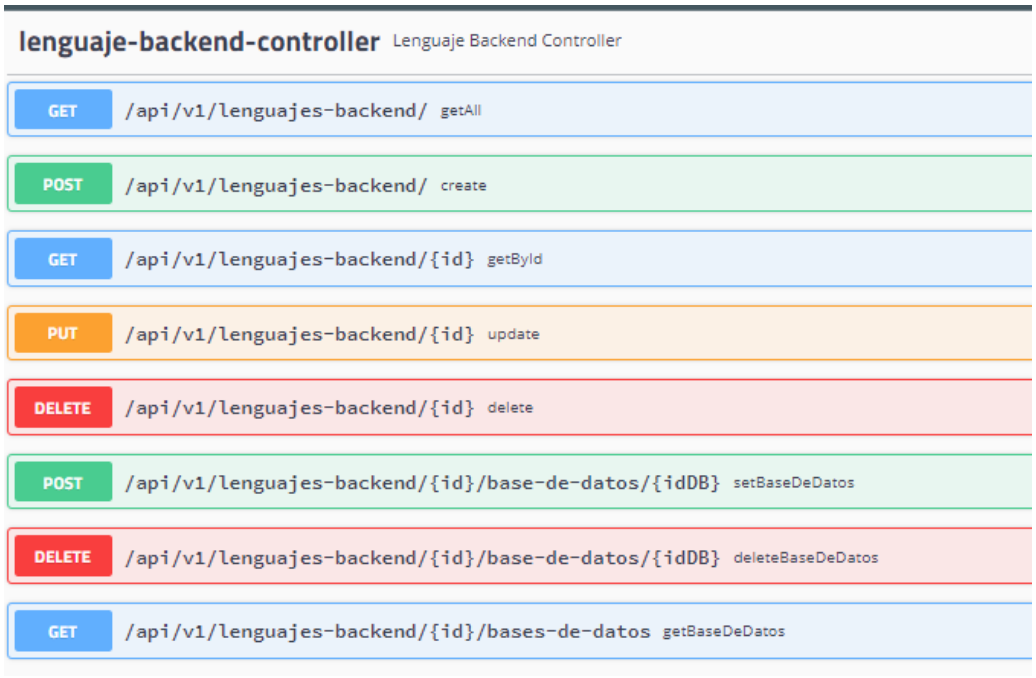
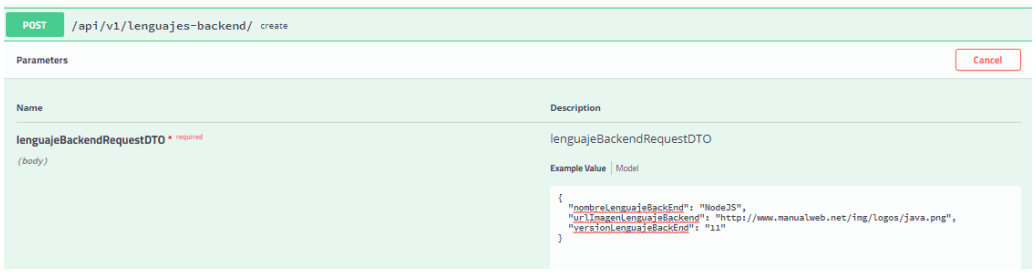
Las pruebas de lógica de los módulos principales se realizarán sobre los módulos de Gestión de Diagramas y Gestión de Roles y Usuarios, e incluirán validaciones de características particulares de cada módulo.

8.3. Realización

Estas pruebas serán ejecutadas al momento de finalizar la programación de cada módulo.

CP004 - Alta de Lenguaje Backend ya existente	
User story	HU010.
Fecha de ejecución	18/10/21.
Actor	Administrador.
Objetivo	Validar que el sistema no admite dos lenguajes backend con el mismo nombre y la misma versión.
Datos de prueba	nombreLenguajeBackEnd1: "NodeJS", urlImagenLenguajeBackend1: "http://www.manualweb.net/img/logos/java.png", versionLenguajeBackEnd1: "11". nombreLenguajeBackEnd2: "NodeJS", urlImagenLenguajeBackend2: "http://www.manualweb.net/img/logos/java.png", versionLenguajeBackEnd: "11".
Precondiciones	Usuario logueado con rol “Administrador”.
Tipo de ejecución	Manual.
Tipo de prueba	Lógica.
Medidas correctivas	N/A.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Ingresar en el navegador la URL "http://localhost:8090/swagger-ui.html#/". *(Captura de pantalla 01)*		
02	Clickear el botón "POST" de la fila con URL "/api/v1/lenguajes-backend/". *(Captura de pantalla 02)*.		
03	Introducir, en el campo "Example Value", datos del lenguaje backend a registrar en el sistema. *(Captura de pantalla 03)*		
04	Clickear el botón "Execute". *(Captura de pantalla 03)*		
05	El sistema muestra detalles de creación de lenguaje backend exitosa. *(Captura de pantalla 04)*		
06	Introducir, en el campo "Example Value", datos del lenguaje backend a registrar en el sistema, ingresando los mismos que los del paso 03. *(Captura de pantalla 03)*		
07	Clickear el botón "Execute". *(Captura de pantalla 03)*	El sistema muestra detalles de error en la creación de lenguaje backend. *(Captura de pantalla 05)*	El sistema muestra detalles de error en la creación de lenguaje backend. *(Captura de pantalla 05)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

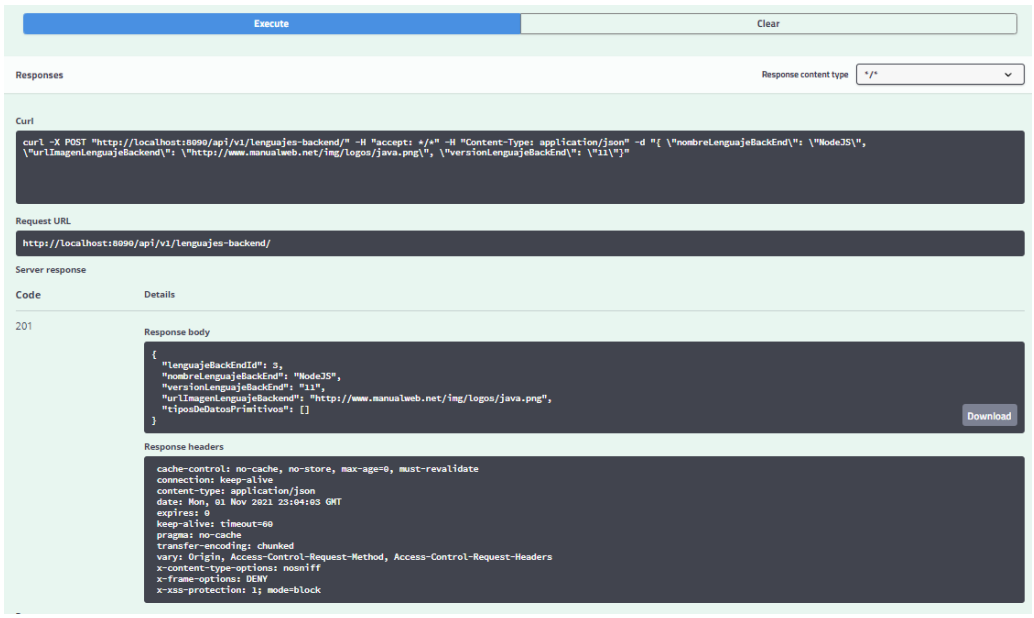
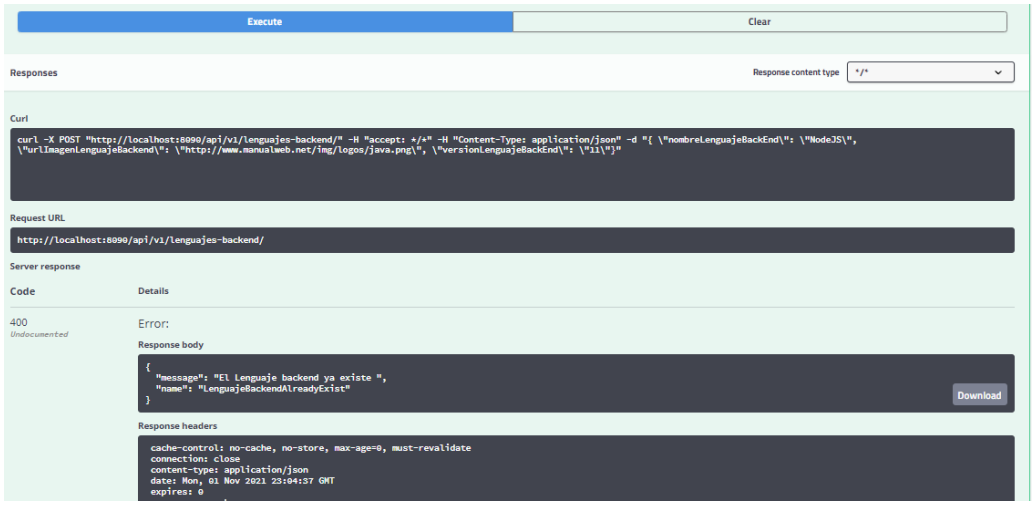
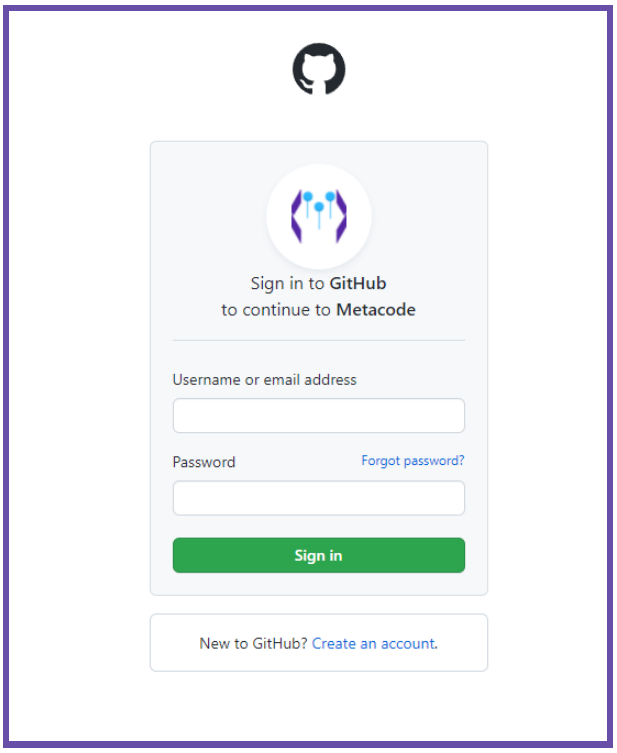
04	
05	

Tabla 58. CP004 Alta de Lenguaje Backend ya existente.

CP005 - Login de usuario mediante GitHub	
User story	HU001, HU002.
Fecha de ejecución	18/10/2021.
Actor	Modelador.
Objetivo	Verificar que el usuario esté autenticado y autorizado para utilizar el sistema.

Datos de prueba	nombreUsuario: PASAPEREZ, contraseña: pass1234.
Precondiciones	N/A.
Tipo de ejecución	Manual.
Tipo de prueba	Prueba de lógica.
Medidas correctivas	N/A.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Clickear botón de “Login with GitHub” *(Captura de pantalla 01)*		
02	Ingresar credenciales de login de GitHub *(Captura de pantalla 02)*		
03	Clickear botón “Sign in” para confirmar inicio de sesión	Ver pantalla de listado de diagramas personales del usuario.	Ver pantalla de acceso no permitido. *(Captura de pantalla 03)*

Nº de Captura	Captura de Pantalla
01	
02	

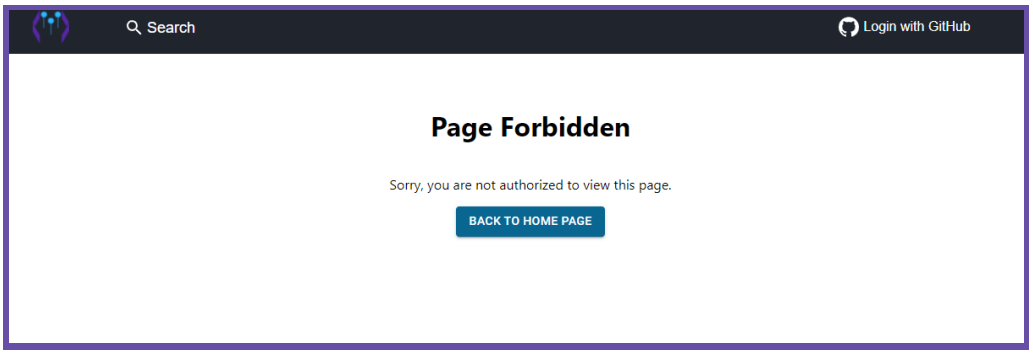
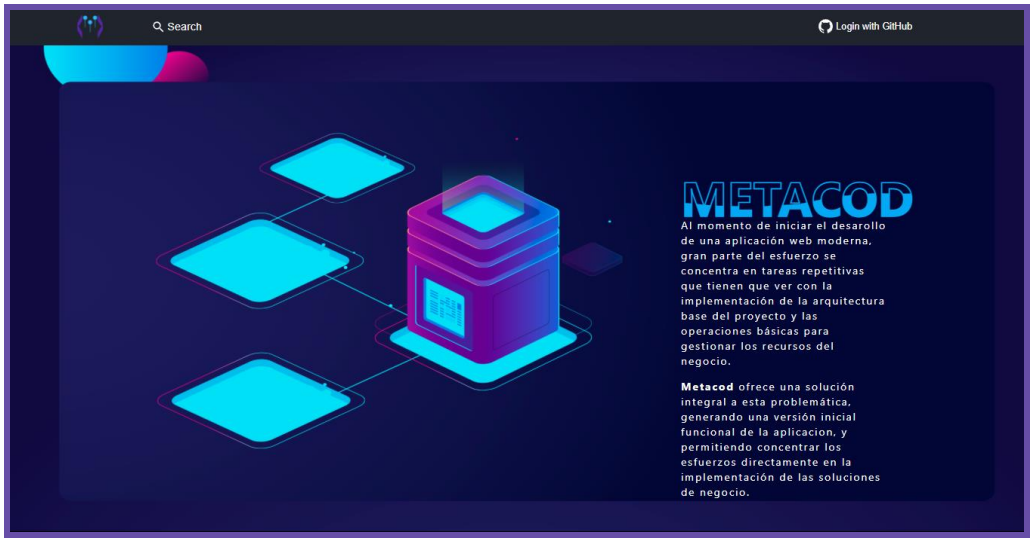
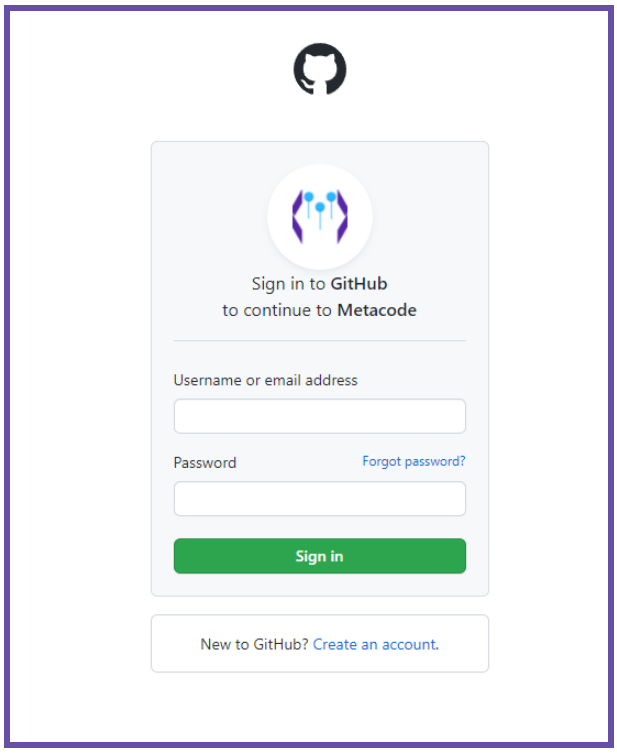
03	

Tabla 59. CP005 Login de usuario mediante GitHub.

CP005 - Login de usuario mediante GitHub	
User story	HU001, HU002.
Fecha de ejecución	18/10/2021.
Actor	Modelador.
Objetivo	Verificar que el usuario esté autenticado y autorizado para utilizar el sistema.
Datos de prueba	nombreUsuario: PASAPEREZ, contraseña: pass1234.
Precondiciones	N/A.
Tipo de ejecución	Manual.
Tipo de prueba	Lógica.
Medidas correctivas	Asignar a los roles de usuario los permisos de acceso a las funcionalidades del sistema correctamente.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Clickear botón de “Login with GitHub” *(Captura de pantalla 01)*		
02	Ingresar credenciales de login de GitHub *(Captura de pantalla 02)*		

03	Clickear botón “Sign in” para confirmar inicio de sesión	Ver pantalla de listado de diagramas personales del usuario. *(Captura de pantalla 03)*	Ver pantalla de listado de diagramas personales del usuario. *(Captura de pantalla 03)*

Nº de Captura	Captura de Pantalla
01	
02	

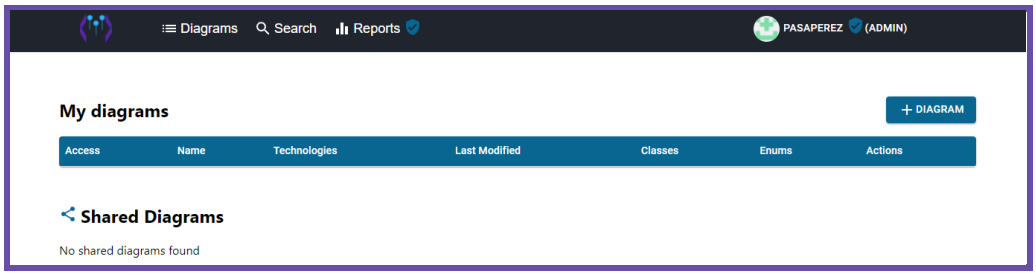
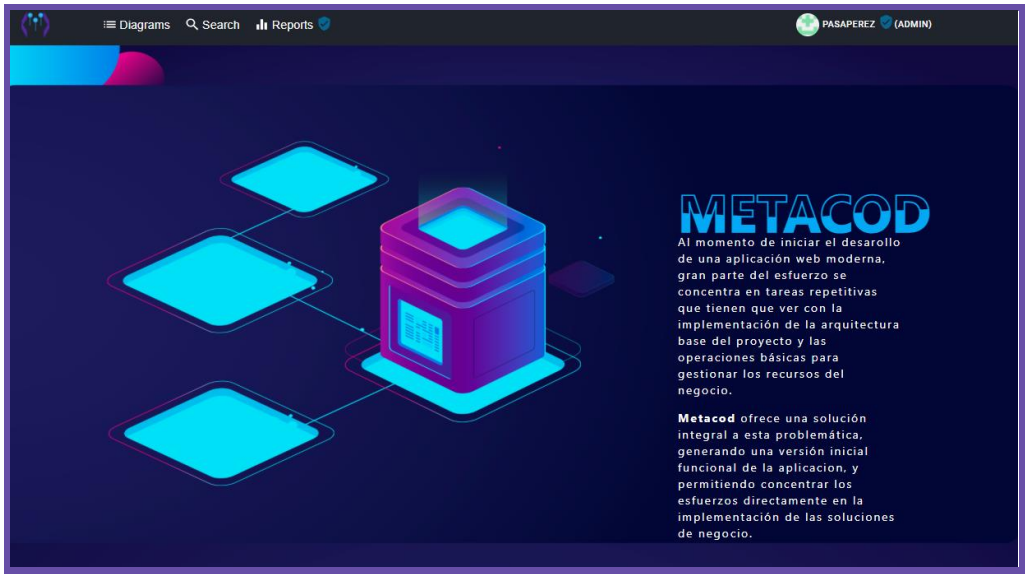
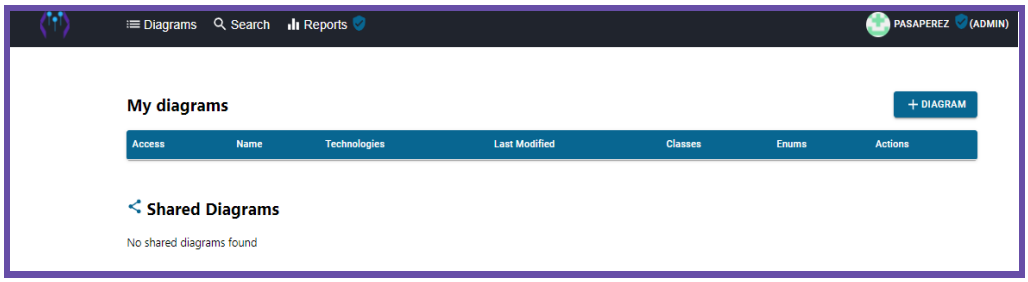
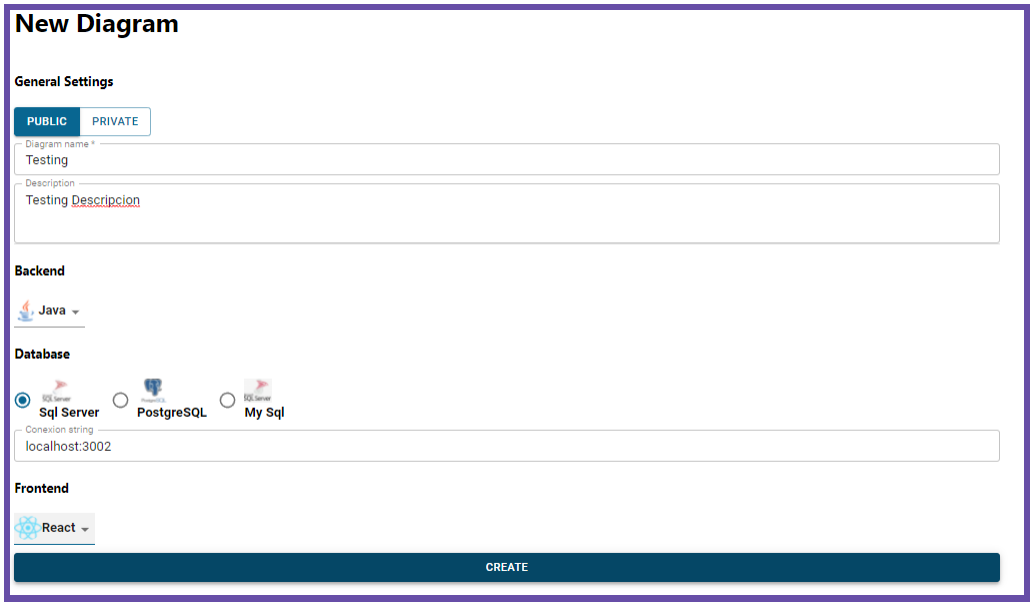
	

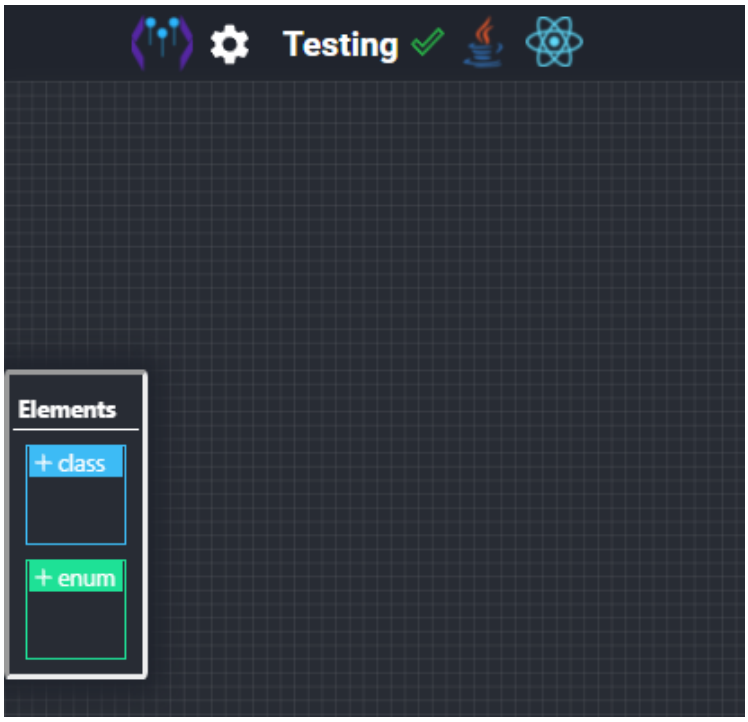
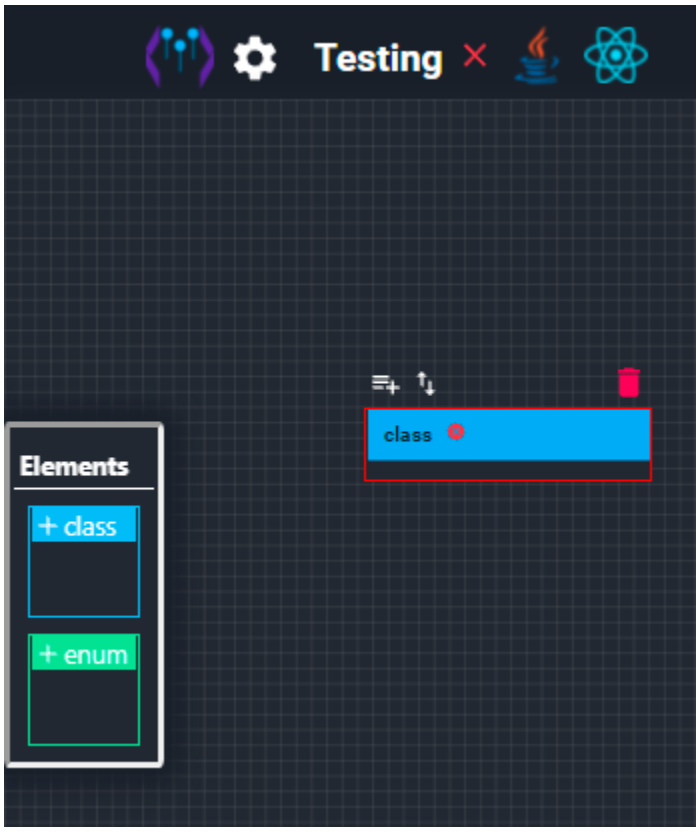
Tabla 60. CP005 Login de usuario mediante GitHub (Corregido).

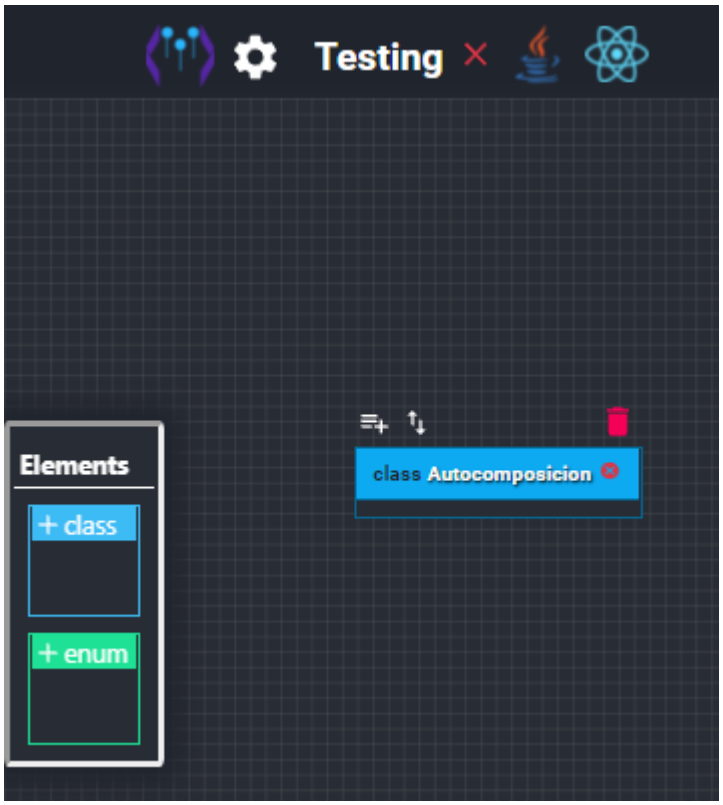
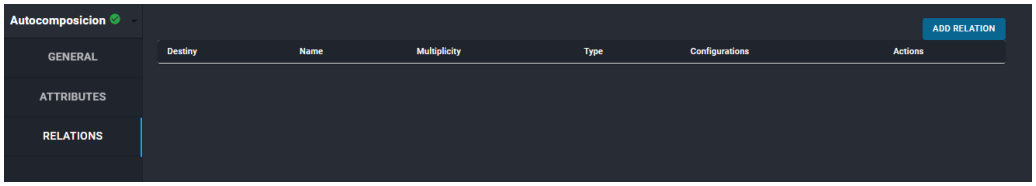
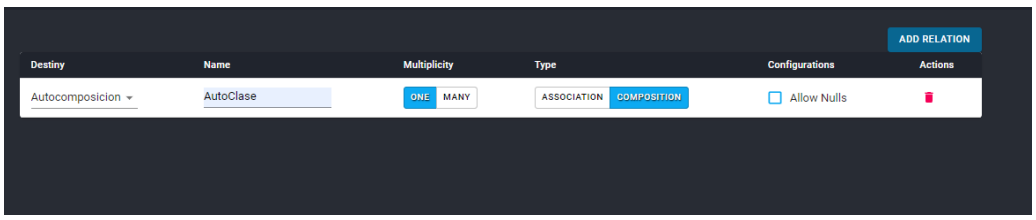
CP006 - Relación de composición con la misma clase tanto en origen como destino	
User story	HU003, HU005, HU006, HU007.
Fecha de ejecución	19/10/21.
Actor	Modelador.
Objetivo	Validar que el sistema no permita agregar una relación cuya clase origen y destino sea la misma.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”, nombreClase: “Autocomposicion”, nombreRelacion: “Autoclase”.
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	Lógica.
Medidas correctivas	N/A.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en el botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en el botón “+ Diagram”.		

	(Captura de pantalla 02)		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear el botón “Create”. *(Captura de pantalla 03)*		
05	Arrastrar elemento “Class” al canvas. *(Captura de pantalla 04)*		
06	Hacer doble click en la clase agregada al canvas. *(Captura de pantalla 05)*		
07	Ingresar nombre de la clase. *(Captura de pantalla 06)*		
08	Click en la pestaña “Relations”. *(Captura de pantalla 07)*		
09	Click en botón “Add Relation”. *(Captura de pantalla 08)*		
10	En el campo “Destination”, seleccionar la misma clase creada. *(Captura de pantalla 09)*	En el elemento gráfico que representa a la clase, aparece un ícono rojo con una cruz del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 09)*	En el elemento gráfico que representa a la clase, aparece un ícono verde con un tick del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de tick verde, indicando que hay integridad en el diagrama. *(Captura de pantalla 10)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

04	 Testing Elements + class + enum
05	 Testing class Elements + class + enum

06	 Testing Elements + class + enum class Autocomposicion
07	 Autocomposicion GENERAL ATTRIBUTES RELATIONS ADD RELATION Destiny Name Multiplicity Type Configurations Actions
08	 ADD RELATION Destiny Name Multiplicity Type Configurations Actions Autocomposicion AutoClase ONE MANY ASSOCIATION COMPOSITION Allow Nulls

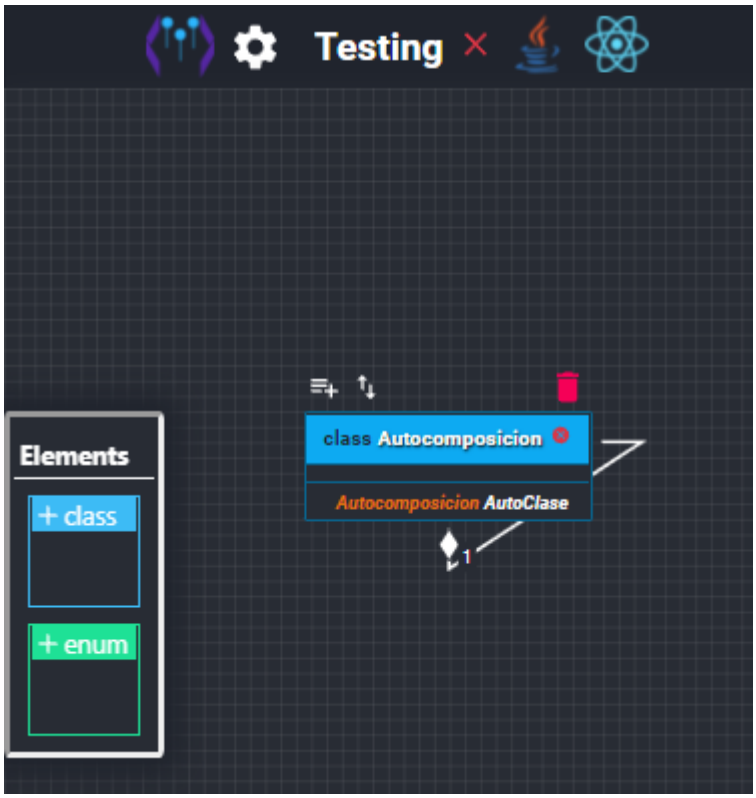
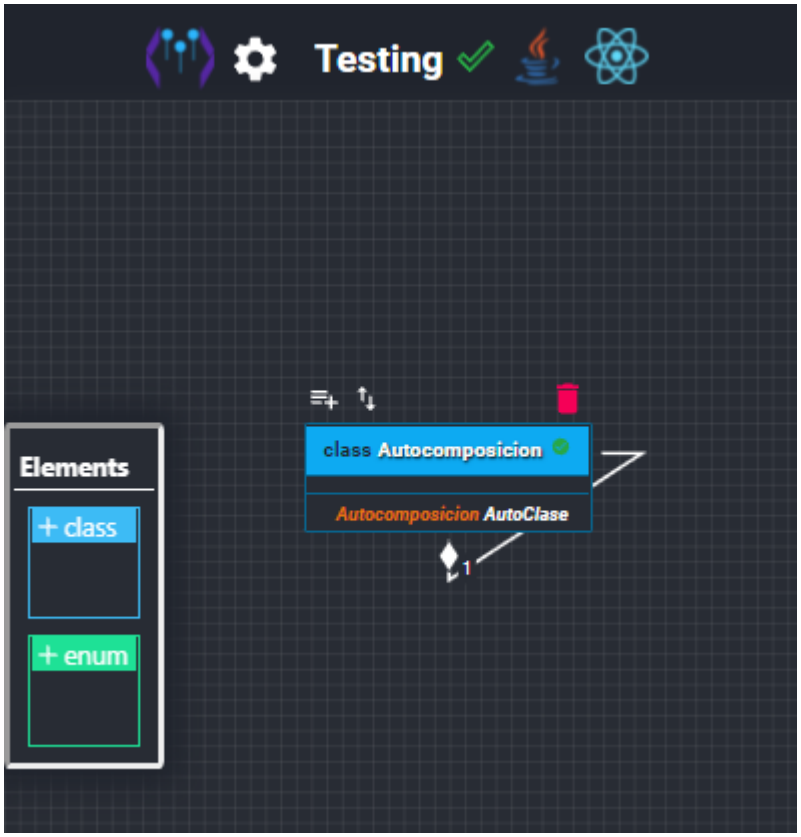
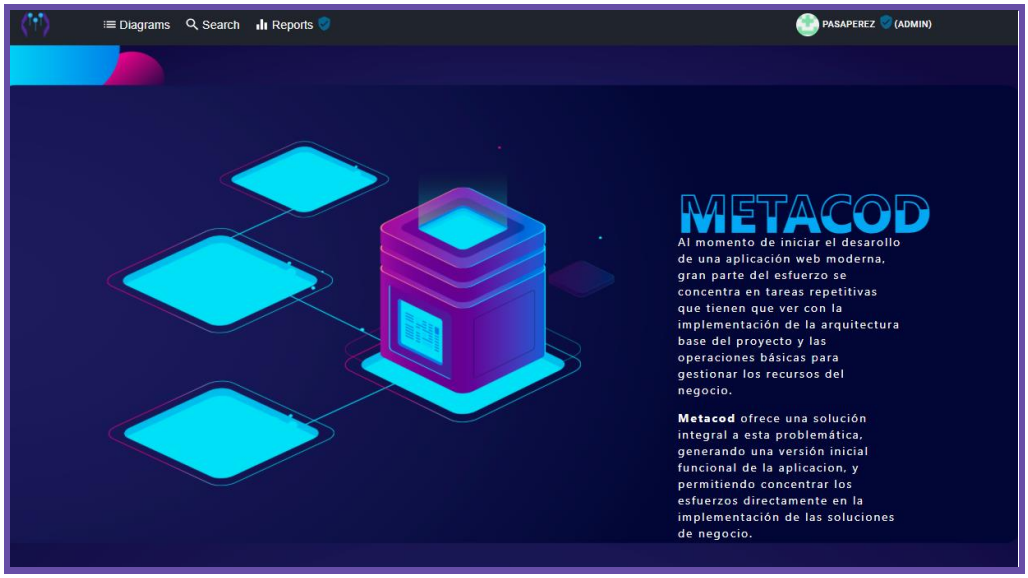
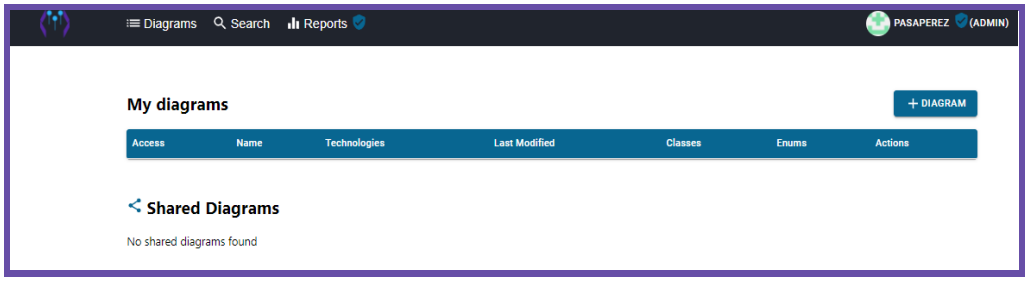
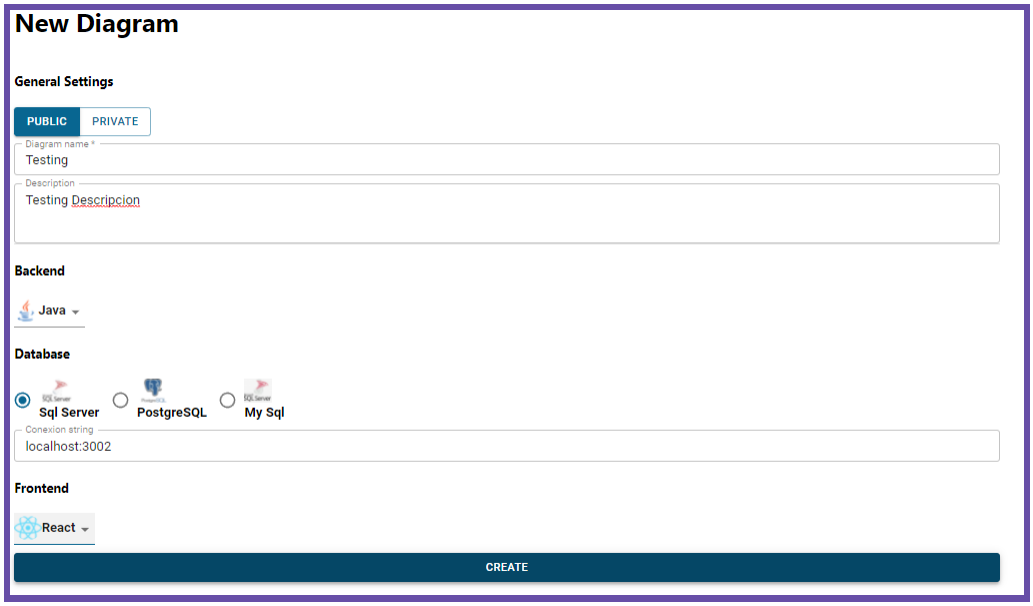
09	
10	

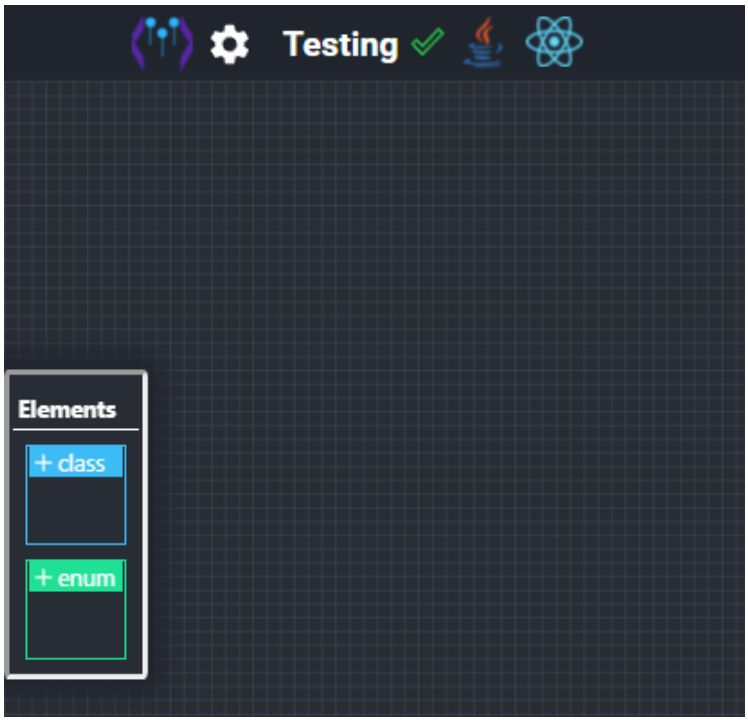
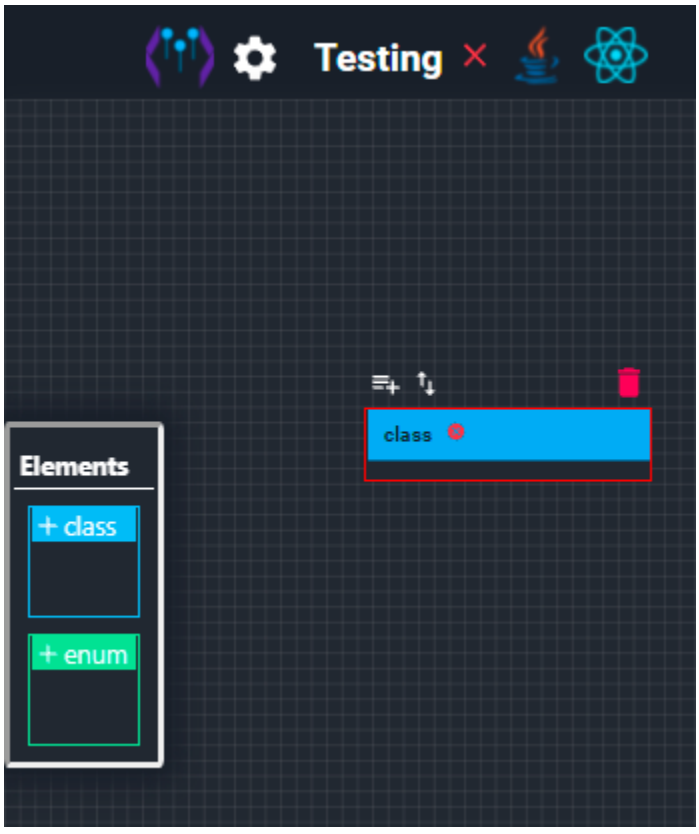
Tabla 61. CP006 Relación de composición con la misma clase tanto en origen como destino.

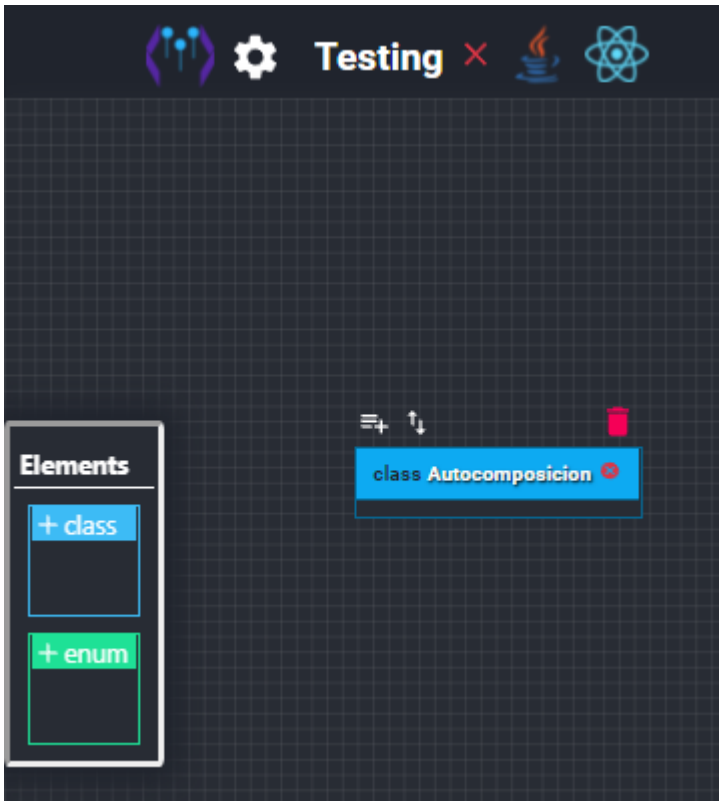
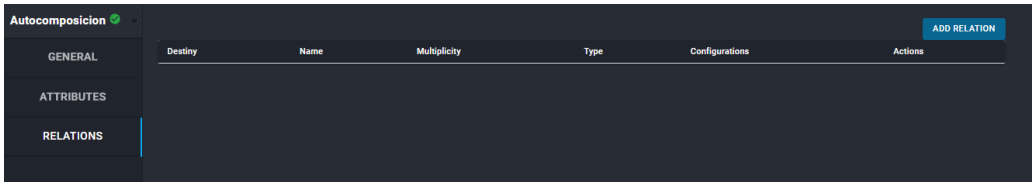
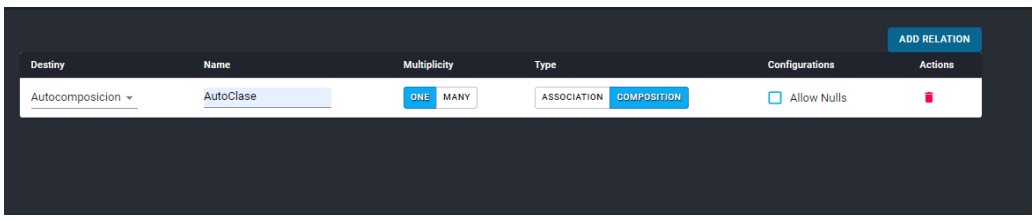
CP006 - Relación de composición con la misma clase tanto en origen como destino	
User story	HU003, HU005, HU006, HU007.
Fecha de ejecución	19/10/21.
Actor	Modelador.
Objetivo	Validar que el sistema no permita agregar una relación cuya clase origen y destino sea la misma.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”, nombreClase: “Autocomposicion”, nombreRelacion: “Autoclase”.
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	Lógica.
Medidas correctivas	Validar, al momento de seleccionar el destino de la relación que se está creando en la clase, que la misma no pasa el control de integridad.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear botón “Create”. *(Captura de pantalla 03)*		

05	Arrastrar elemento “Class” al canvas. *(Captura de pantalla 04)*		
06	Hacer doble click en la clase agregada al canvas. *(Captura de pantalla 05)*		
07	Ingresar nombre de la clase. *(Captura de pantalla 06)*		
08	Click en la pestaña “Relations”. *(Captura de pantalla 07)*		
09	Click en botón “Add Relation”. *(Captura de pantalla 08)*		
10	En el campo “Destination”, seleccionar la misma clase creada. *(Captura de pantalla 09)*	En el elemento gráfico que representa a la clase, aparece un ícono rojo con una cruz del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 09)*	En el elemento gráfico que representa a la clase, aparece un ícono rojo con una cruz del lado derecho del nombre de la clase, y a la derecha del nombre del diagrama se muestra un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 09)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

04	 Testing Elements + class + enum
05	 Testing class Elements + class + enum

06	 Testing Elements + class + enum class Autocomposicion
07	 Autocomposicion GENERAL ATTRIBUTES RELATIONS ADD RELATION Destiny Name Multiplicity Type Configurations Actions
08	 ADD RELATION Destiny Name Multiplicity Type Configurations Actions Autocomposicion AutoClase ONE MANY ASSOCIATION COMPOSITION Allow Nulls

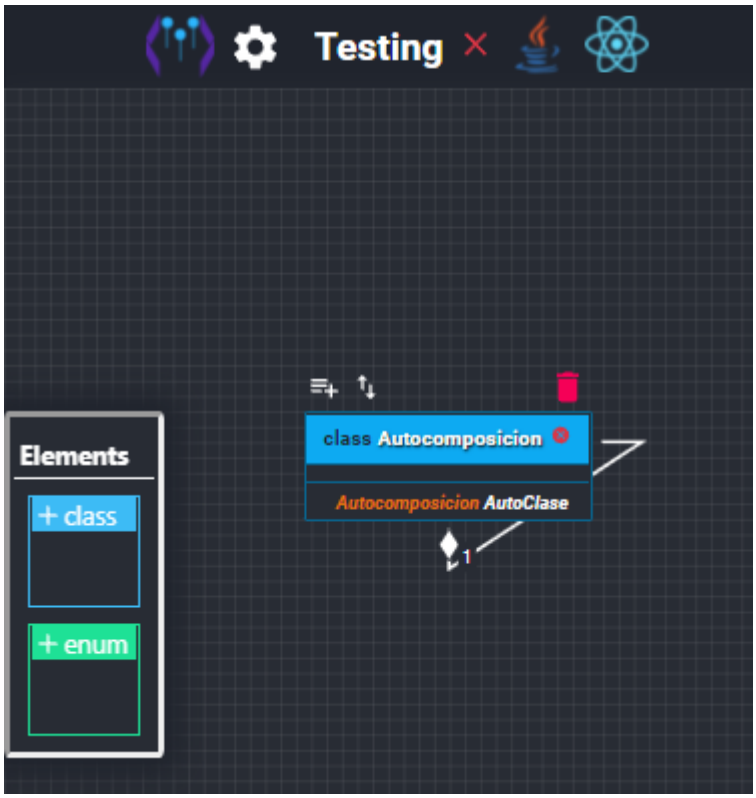
09	

Tabla 62. CP006 Relación de composición con la misma clase tanto en origen como destino (Corregido).

9. Pruebas de integración entre módulos

9.1. Objetivo

Verificar que los diferentes módulos desarrollados se comunican e interactúan entre sí de forma apropiada, y que, una vez integrados los mismos, el sistema responde a las funcionalidades especificadas correctamente.

9.2. Alcance

Las pruebas de integración entre módulos contemplarán funcionalidades que requieran para su realización una ejecución transversal entre módulos, es decir, donde los módulos incluidos en la prueba interactúen entre sí para la obtención de un resultado.

9.3. Realización

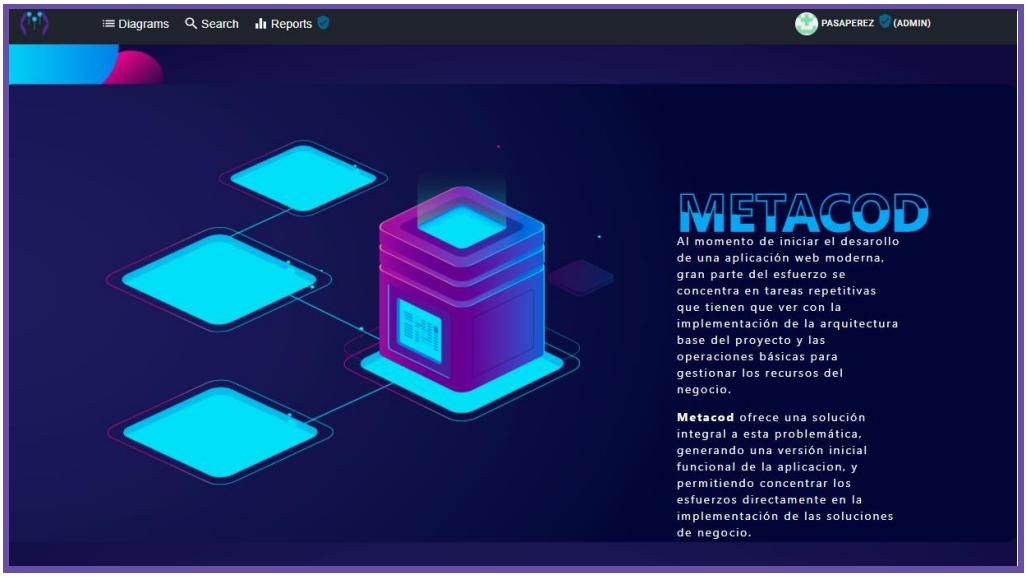
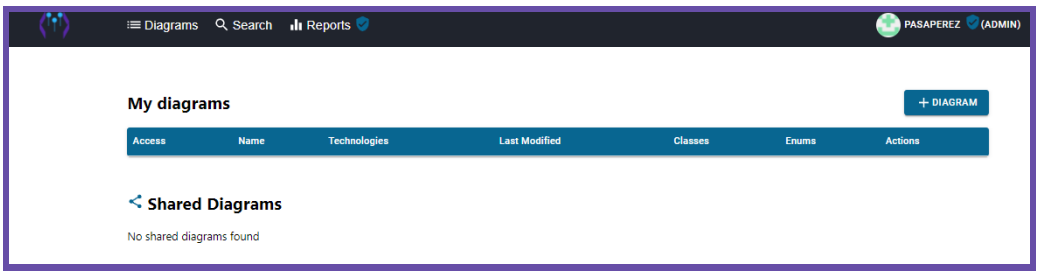
Estas pruebas se ejecutarán progresivamente, a medida que vaya concluyendo el desarrollo de los módulos incluidos en la prueba o las principales funcionalidades de los mismos que serán probadas.

CP007 - Crear un nuevo diagrama

User story	HU003, HU005.
Fecha de ejecución	20/10/2021.
Actor	Modelador.
Objetivo	Verificar que el sistema cree correctamente un diagrama asociado a un lenguaje backend, un lenguaje frontend, una base de datos, al rol “Propietario” del usuario que lo crea.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	De Integración.
Medidas correctivas	N/A.
Módulos involucrados	Módulo de Roles, Usuarios y Login. Módulo de Gestión de diagramas. Módulo de Reportes.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear botón “Create”.		

	(Captura de pantalla 03)		
05	Clickear botón de inicio (ícono de Metacod). *(Captura de pantalla 04)*		
06	Clickear botón “Diagrams”. *(Captura de pantalla 01)*	Se muestra la lista de los diagramas pertenecientes al usuario, la cual incluye el diagrama recientemente creado, en donde se detallan las configuraciones ingresadas en el paso 03. *(Captura de pantalla 05)*	Se muestra la lista de los diagramas pertenecientes al usuario, la cual incluye el diagrama recientemente creado, en donde se detallan las configuraciones ingresadas en el paso 03. *(Captura de pantalla 05)*

Nº de Captura	Captura de Pantalla
01	
02	

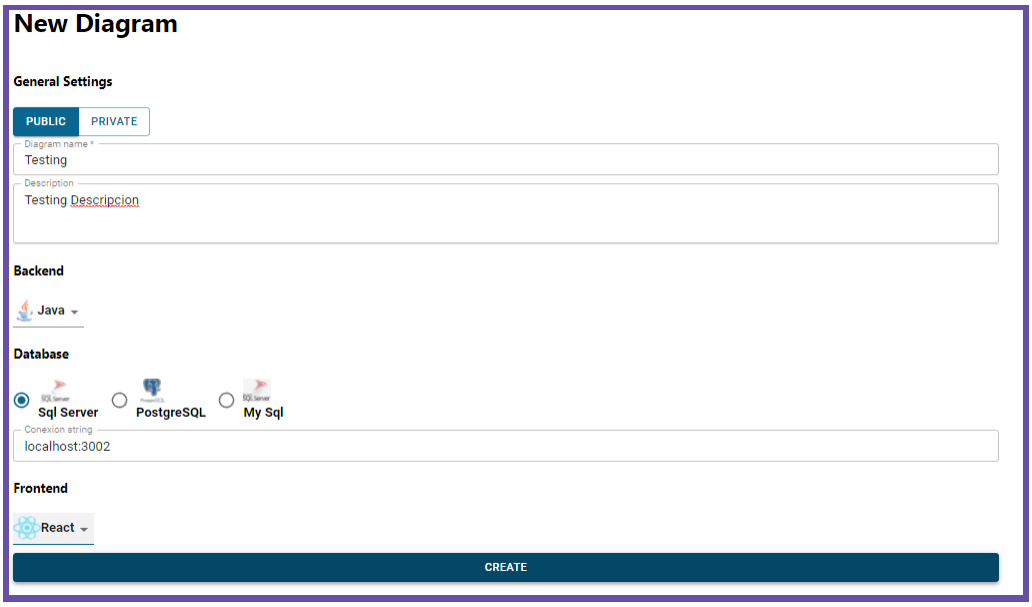
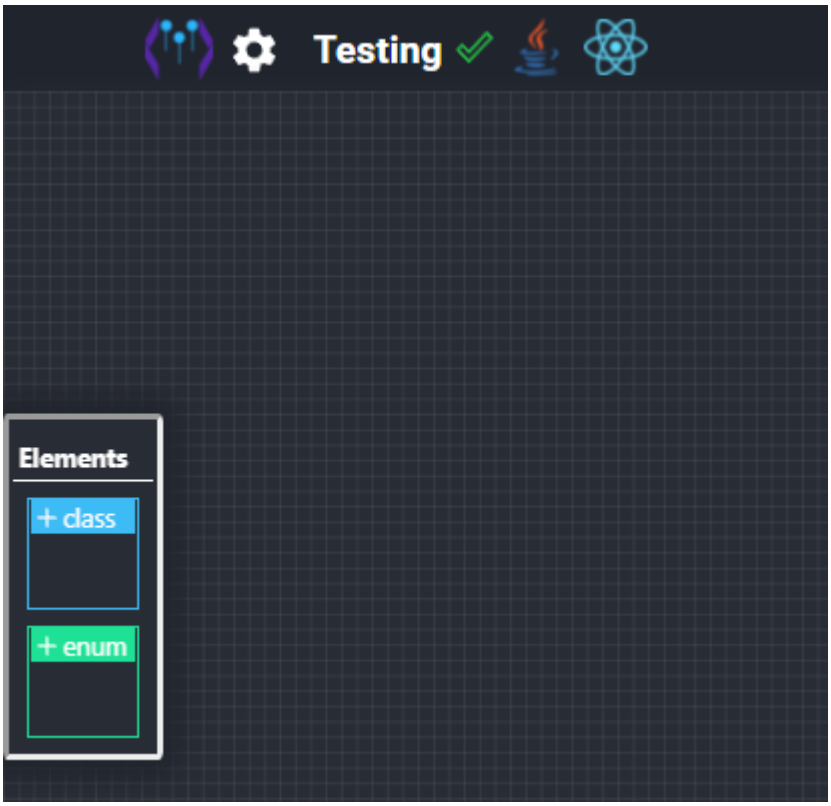
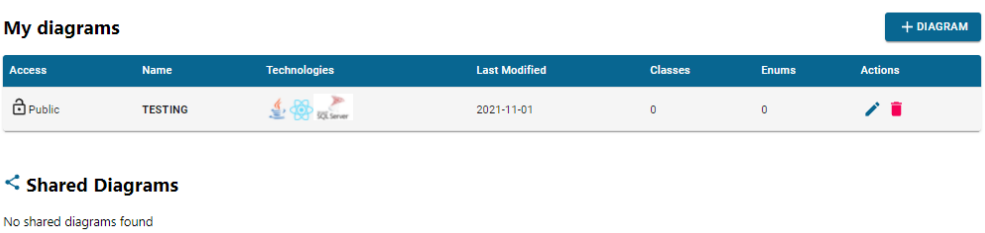
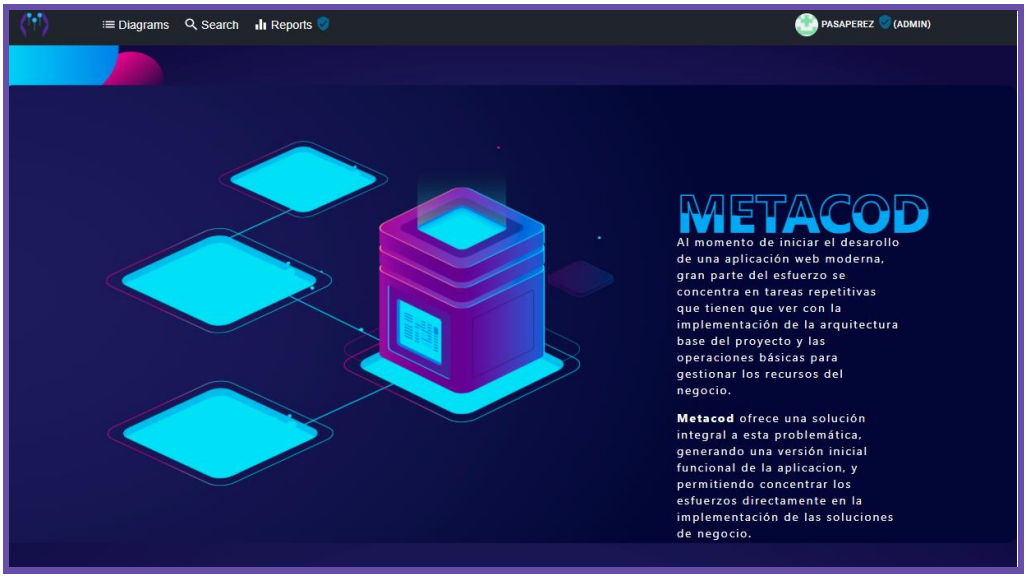
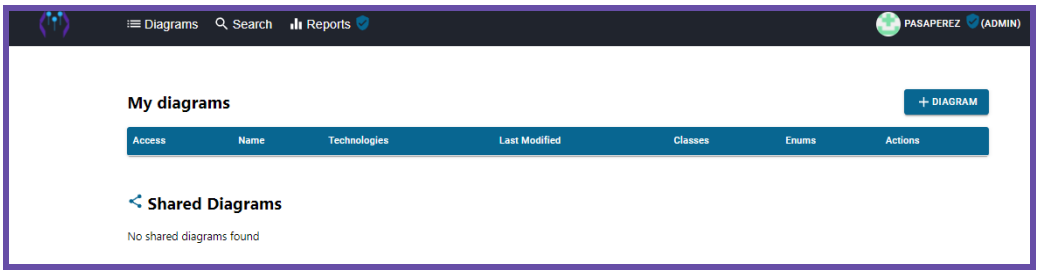
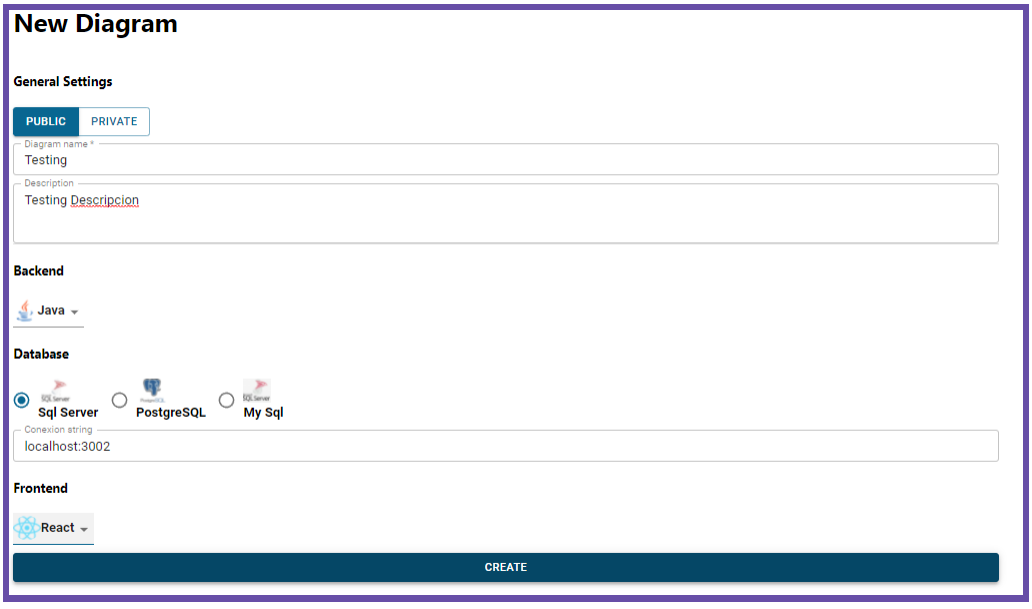
03	 New Diagram General Settings PUBLIC PRIVATE Diagram name * Testing Description Testing Descripcion Backend Java Database Sql Server PostgreSQL My Sql Conexion string localhost:3002 Frontend React CREATE
04	 Testing Elements + class + enum
05	 My diagrams + DIAGRAM Access Name Technologies Last Modified Classes Enums Actions Public TESTING 2021-11-01 0 0 Shared Diagrams No shared diagrams found

Tabla 63. CP007 Crear un nuevo diagrama.

CP008 - Exportar código generado a partir de un diagrama vacío a GitHub	
User story	HU003, HU005, HU009, HU010.
Fecha de ejecución	20/10/21.
Actor	Modelador.
Objetivo	Crear un diagrama vacío y verificar que la generación del código y posterior exportación a partir del mismo no es permitida por el sistema.
Datos de prueba	nombreDiagrama: “Testing”, descripciónDiagrama: “Testing descripción”, lenguajeBackend: “Java”, lenguajeFrontend: “React”, cadenaDeConexión: “http://localhost:3002;database=db_testing”.
Precondiciones	Usuario logueado con rol “Modelador”.
Tipo de ejecución	Manual.
Tipo de prueba	De integración.
Medidas correctivas	N/A.
Módulos involucrados	Módulo de Gestión de diagramas. Módulo generador código Api en Java 8. Módulo generador código Frontend React. Módulo exportador del código a un repositorio. Módulo de Reportes.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Hacer click en botón “Diagrams”. *(Captura de pantalla 01)*		
02	Hacer click en botón “+ Diagram”. *(Captura de pantalla 02)*		
03	Ingresar datos de diagrama. *(Captura de pantalla 03)*		
04	Clickear botón “Create”. *(Captura de pantalla 03)*	Se muestra, a la derecha del nombre del diagrama, un ícono de cruz roja, indicando un fallo en la integridad del diagrama. *(Captura de pantalla 07)*	Se muestra, a la derecha del nombre del diagrama, un ícono de tick verde, indicando que hay integridad en el diagrama. *(Captura de pantalla 04)*
05	Clickear botón “Generate Code”. *(Captura de pantalla 04)*	El botón se encuentra deshabilitado, y no ejecuta la funcionalidad de generar y exportar el código a GitHub. *(Captura de pantalla 07)*	El botón se encuentra habilitado, y se ejecuta la funcionalidad de generar y exportar el código a GitHub. *(Captura de pantalla 04)*
06	Clickear botón “Generate”. *(Captura de pantalla 05)*	La ventana de confirmación de generación de código y exportación no se muestra. *(Captura de pantalla 07)*	El código es generado a partir del diagrama vacío y exportado a GitHub. *(Captura de pantalla 06)*

Nº de Captura	Captura de Pantalla
01	
02	
03	

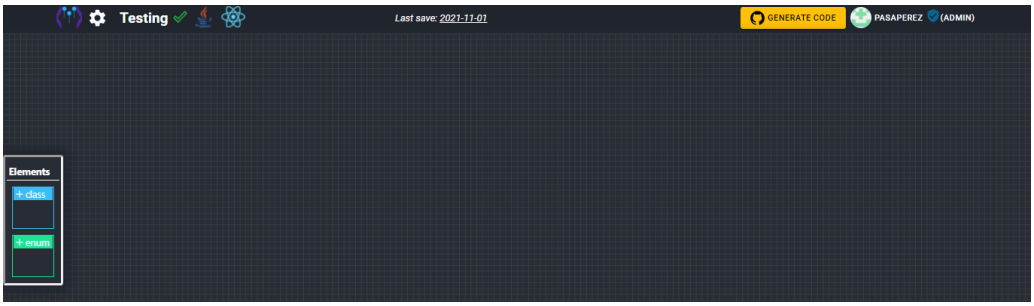
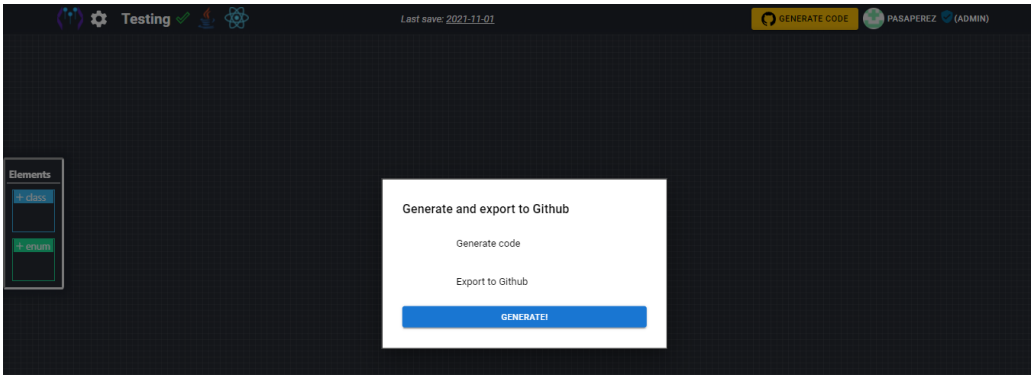
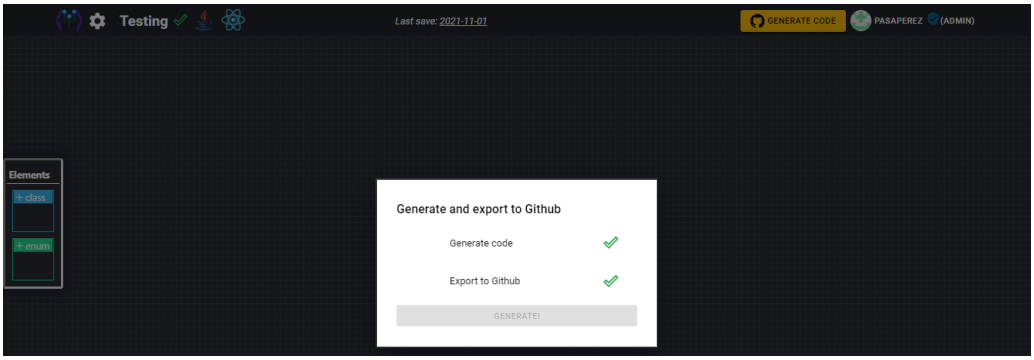
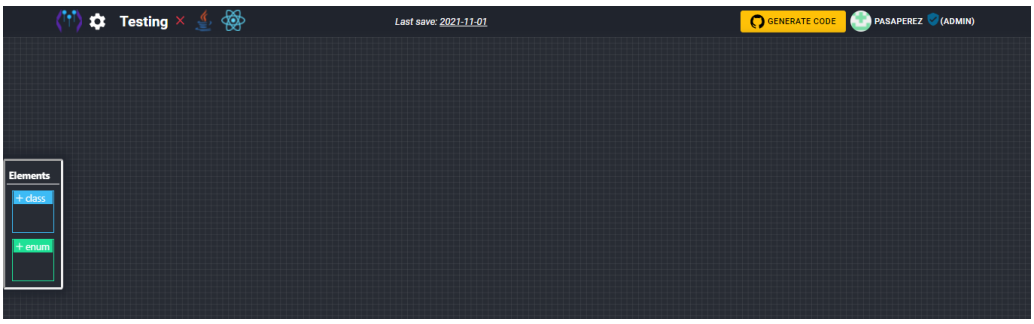
04	
05	
06	
07	

Tabla 64. CP008 Exportar código generado a partir de un diagrama vacío a GitHub.

CP009 - Generar reporte de lenguajes backend más utilizados	
User story	HU003, HU005, HU010, HU015.
Fecha de ejecución	21/10/21.
Actor	Administrador.
Objetivo	Verificar que el módulo de la generación de reportes puede acceder y presentar la información del sistema correctamente.
Datos de prueba	tipoDeReporte: “tecnologíasMasUsadas”.
Precondiciones	Usuario logueado con rol “Administrador”.
Tipo de ejecución	Manual.
Tipo de prueba	De integración.
Medidas correctivas	N/A.
Módulos involucrados	Módulo de Reportes. Módulo de Gestión de diagramas. Módulo generador código Api en Java 8.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Clickear el botón “Reports”. *(Captura de pantalla 01)*		
02	En el menú desplegable, clickear “Technology Statistics”. *(Captura de pantalla 02)*	Se muestra una pantalla que sintetiza la información de los lenguajes backend más utilizados en un gráfico de dona. *(Captura de pantalla 03)*	Se muestra una pantalla que sintetiza la información de los lenguajes backend más utilizados en un gráfico de dona. *(Captura de pantalla 03)*

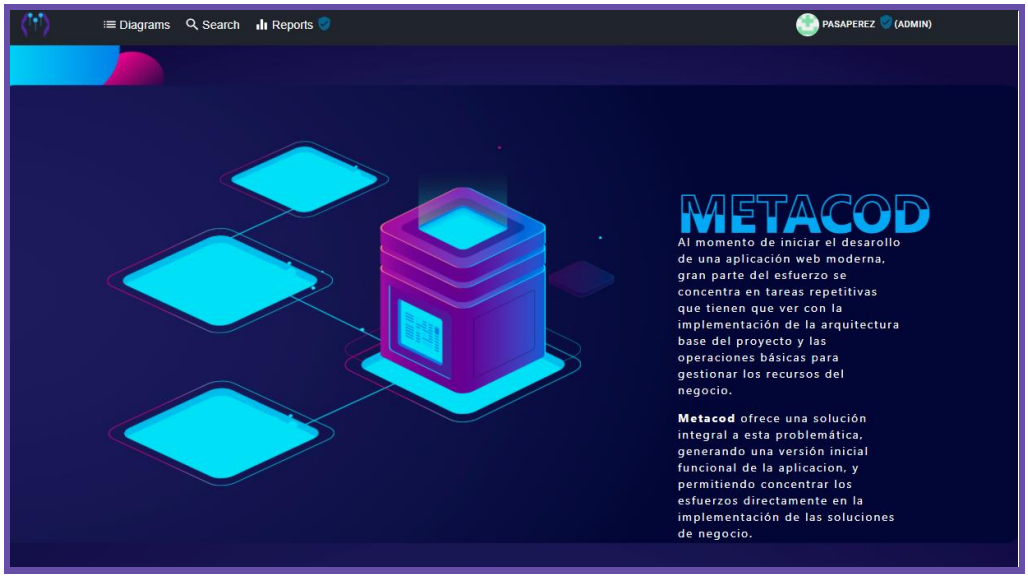
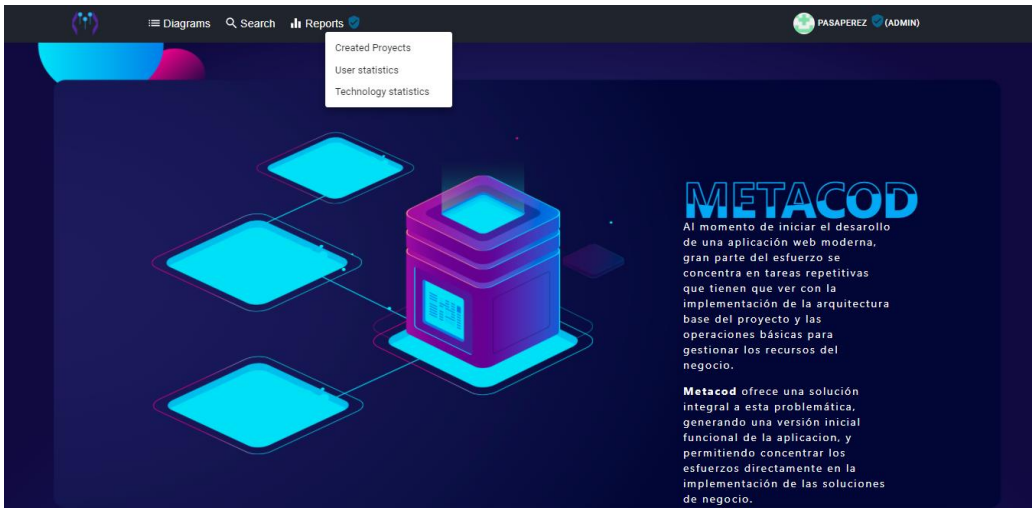
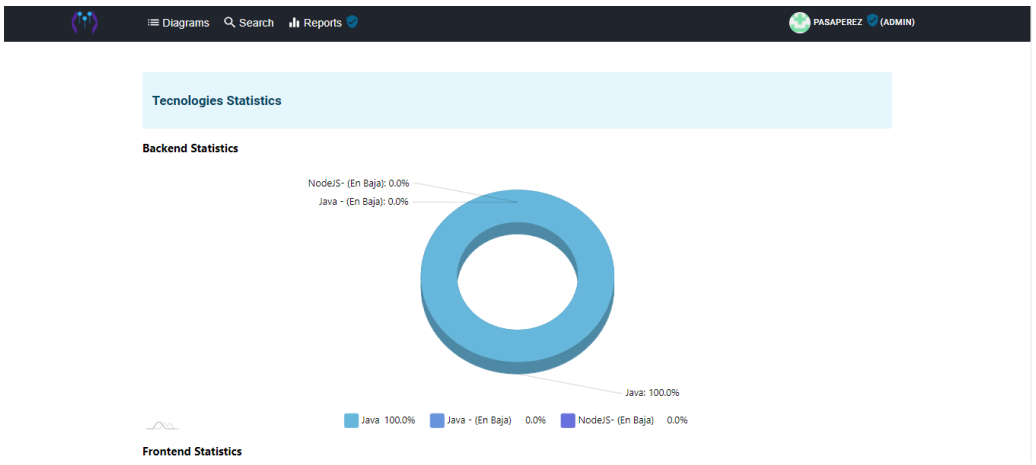
Nº de Captura	Captura de Pantalla
01	
02	
03	

Tabla 65. CP009 Generar reporte de lenguajes backend más utilizados.

10. Pruebas de carga

10.1. Objetivo

Validar el rendimiento de las funcionalidades del sistema, en lo que respecta a complejidad de procesamiento y consultas HTTP a la API del Sistema.

10.2. Alcance

Las pruebas de carga consistirán en la creación de muchas instancias de entidades del sistema de manera simultánea.

10.3. Realización

Estas pruebas requieren el apropiado funcionamiento e interacción entre distintos módulos, razón por la cual se llevarán a cabo luego de concluir con las pruebas de integración.

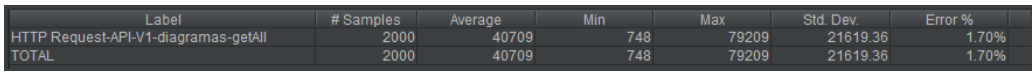
Se utilizará el software Apache JMeter para la realización de las pruebas de carga.

CP010 - Consultas concurrentes de diagramas públicos	
User story	HU003, HU004.
Fecha de ejecución	16/11/21.
Actor	Administrador.
Objetivo	Verificar que el sistema puede consultar todos los diagramas concurrentemente con 2000 usuarios (o consultas), en un periodo de 10 segundos, sin superar los 60 segundos de espera, y con una tasa de éxito del 99%.
Datos de prueba	Token JWT de seguridad de Actor logueado con permisos: “eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJwYXNhcGVyZXoiLCJQZX JtaXNvcyI6W3siYXV0aG9yaXR5IjoiUk9MRV9BRE1JTklTVFJBR E9SIn1dLCJGZWNoYV9FeHBpcmFjaW9uX1Rva2VuIjoiMDYtMT ItMjAyMSAwMTo1NjowOSIsIlVzdWFyaW9fTWV0YWNvZCI6eyJu b21icmVVc3VhcmlvIjoicGFzYXBlcmV6IiwidXN1YXJpb0lkIjoiMyIs ImVtYWlsIjoicGFzYXBlcmV6QHlhaG9vLmNvbS5hciIsImltYWdlbl VybCI6Imh0dHBzOi8vYXZhdGFycy5naXRodWJ1c2VyY29udGVu dC5jb20vdS8yMDU0NDY5P3Y9NCJ9LCJleHAiOjE2Mzg3NjY1Nj l9.5a33xi7HhFUhrp-set-awRAFseQwcT0x8GAou4aaZhqsmn7N SmJBZ-UEbCPxR-mLHOSnGwpbroItmpB2JVvU8Q”.
Precondiciones	Al menos un diagrama existente en Base de Datos.
Tipo de ejecución	Automática.
Tipo de prueba	Carga.

Como resultado esperado se tiene que con 2000 usuarios (o consultas), en un periodo de 10 segundos la API debe responder sin superar los 60 segundos de espera, y tener una tasa de éxito del 99%.

Como resultado obtenido *(Captura de pantalla 01)* algunas consultas superan hasta los 79 segundos y la tasa de éxito es inferior a 99%.

El caso de prueba falló en cumplir con las expectativas de rendimiento esperadas.

Nº de Captura	Captura de Pantalla
01	

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request-API-V1-diagramas-getAll	2000	40709	748	79209	21619.36	1.70%
TOTAL	2000	40709	748	79209	21619.36	1.70%

Tabla 66. CP010 Consultas concurrentes de diagramas públicos.

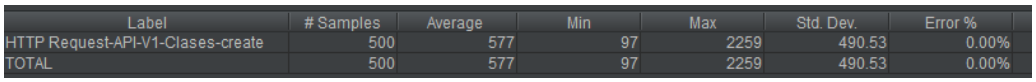
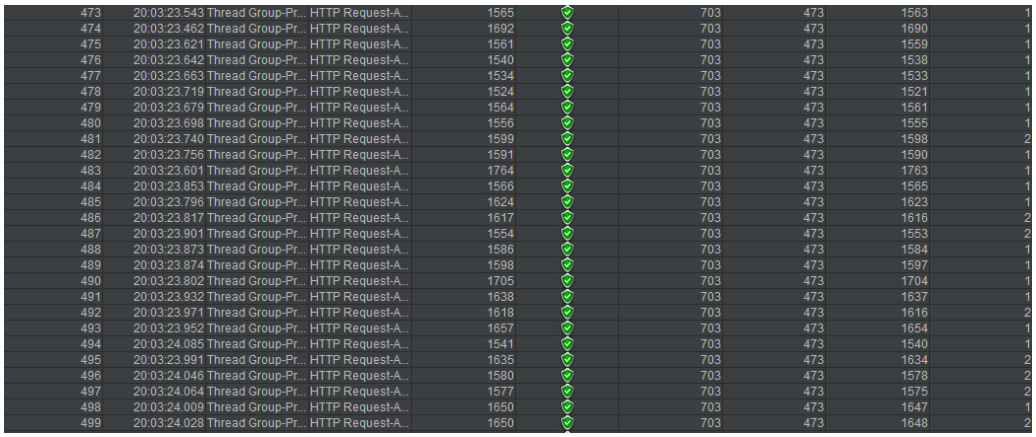
CP011 - Creación concurrente de clases en diagramas	
User story	HU006.
Fecha de ejecución	17/11/21.
Actor	Administrador.
Objetivo	Verificar que el sistema puede soportar la creación concurrente de 500 usuarios (o consultas), en un periodo de 10 segundos, sin superar los 30 segundos de espera, y con una tasa de éxito del 99%.
Datos de prueba	Content-Type= application/json, accept = */*, { "ancho": 1, "atributoDefaultId": 1, "clasePadreId": 0, "coordenadaX": 100, "coordenadaY": 100, "esAbstracta": true, "nombre": "Clase1", "sePuedeCrear": true, "sePuedeEditar": true, "sePuedeEliminar": true, "sePuedeLeer": true }.
Precondiciones	Al menos un diagrama existente en Base de Datos.

Tipo de ejecución	Automática.
Tipo de prueba	De Carga.

Como resultado esperado se tiene que con 500 usuarios (o consultas), en un periodo de 10 segundos la API debe responder sin superar los 30 segundos de espera, y tener una tasa de éxito del 99%.

Como resultado obtenido *(Captura de pantalla 01)* la tasa de éxito fue del 100%, y el tiempo de espera fue de poco más de 2 segundos.

El caso de prueba paso con éxito y cumplió el objetivo.

Nº de Captura	Captura de Pantalla
01	
02	

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request-API-V1-Clases-create	500	577	97	2259	490.53	0.00%
TOTAL	500	577	97	2259	490.53	0.00%

473	20:03:23.543	Thread Group-Pr...	HTTP Request-A...	1565		703	473	1563	1
474	20:03:23.462	Thread Group-Pr...	HTTP Request-A...	1692		703	473	1690	1
475	20:03:23.621	Thread Group-Pr...	HTTP Request-A...	1561		703	473	1559	1
476	20:03:23.642	Thread Group-Pr...	HTTP Request-A...	1540		703	473	1538	1
477	20:03:23.663	Thread Group-Pr...	HTTP Request-A...	1534		703	473	1533	1
478	20:03:23.719	Thread Group-Pr...	HTTP Request-A...	1524		703	473	1521	1
479	20:03:23.679	Thread Group-Pr...	HTTP Request-A...	1564		703	473	1561	1
480	20:03:23.698	Thread Group-Pr...	HTTP Request-A...	1556		703	473	1555	1
481	20:03:23.740	Thread Group-Pr...	HTTP Request-A...	1599		703	473	1598	2
482	20:03:23.756	Thread Group-Pr...	HTTP Request-A...	1591		703	473	1590	1
483	20:03:23.601	Thread Group-Pr...	HTTP Request-A...	1764		703	473	1763	1
484	20:03:23.853	Thread Group-Pr...	HTTP Request-A...	1566		703	473	1565	1
485	20:03:23.796	Thread Group-Pr...	HTTP Request-A...	1624		703	473	1623	1
486	20:03:23.817	Thread Group-Pr...	HTTP Request-A...	1617		703	473	1616	2
487	20:03:23.901	Thread Group-Pr...	HTTP Request-A...	1554		703	473	1553	2
488	20:03:23.873	Thread Group-Pr...	HTTP Request-A...	1586		703	473	1584	1
489	20:03:23.874	Thread Group-Pr...	HTTP Request-A...	1598		703	473	1597	1
490	20:03:23.802	Thread Group-Pr...	HTTP Request-A...	1705		703	473	1704	1
491	20:03:23.932	Thread Group-Pr...	HTTP Request-A...	1638		703	473	1637	1
492	20:03:23.971	Thread Group-Pr...	HTTP Request-A...	1618		703	473	1616	2
493	20:03:23.952	Thread Group-Pr...	HTTP Request-A...	1657		703	473	1654	1
494	20:03:24.085	Thread Group-Pr...	HTTP Request-A...	1541		703	473	1540	1
495	20:03:23.991	Thread Group-Pr...	HTTP Request-A...	1635		703	473	1634	2
496	20:03:24.046	Thread Group-Pr...	HTTP Request-A...	1580		703	473	1578	2
497	20:03:24.064	Thread Group-Pr...	HTTP Request-A...	1577		703	473	1575	2
498	20:03:24.009	Thread Group-Pr...	HTTP Request-A...	1650		703	473	1647	1
499	20:03:24.028	Thread Group-Pr...	HTTP Request-A...	1650		703	473	1648	2

Tabla 67. CP011 Creación concurrente de clases en diagramas.

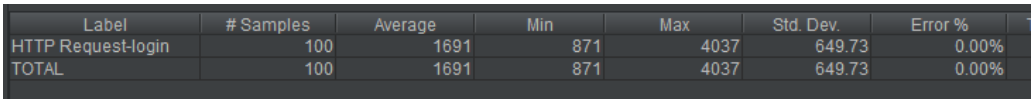
CP012 - Solicitudes de logueo concurrentes	
User story	HU002.
Fecha de ejecución	18/11/21.
Actor	Usuario Genérico.

Objetivo	Verificar que el sistema puede soportar la solicitud de logueo de 100 usuarios (o consultas), en un periodo de 3 segundos, sin superar los 30 segundos de espera, y con una tasa de éxito del 99%.
Datos de prueba	code: f3d1c5620243de57a797.
Precondiciones	-
Tipo de ejecución	Automática.
Tipo de prueba	De Carga.

Como resultado esperado se tiene que con 100 usuarios (o consultas), en un periodo de 3 segundos la API debe responder sin superar los 30 segundos de espera, y tener una tasa de éxito del 99%.

Como resultado obtenido *(Captura de pantalla 01)* la tasa de éxito fue del 100%, y el tiempo de espera fue de poco más de 4 segundos.

El caso de prueba paso con éxito y cumplió el objetivo.

Nº de Captura	Captura de Pantalla
01	

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
HTTP Request-login	100	1691	871	4037	649.73	0.00%
TOTAL	100	1691	871	4037	649.73	0.00%

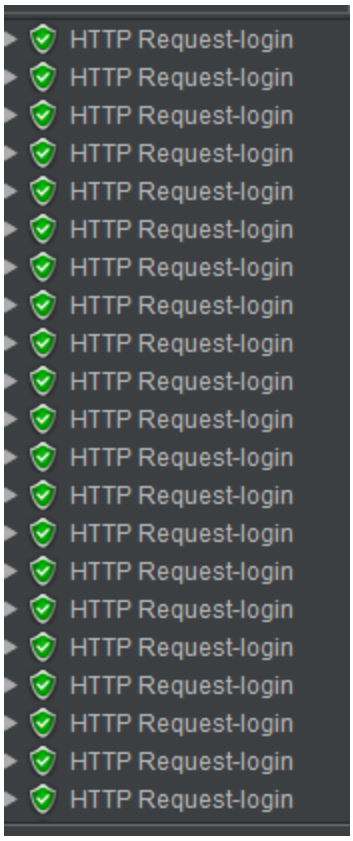
02	

Tabla 68. CP012 Solicitudes de logueo concurrentes.

11. Pruebas de seguridad por niveles de usuario

11.1. Objetivo

Verificar que cada usuario pueda hacer uso del sistema sólo si se ha autenticado correctamente, y que únicamente tenga permitido acceder a las funcionalidades que le corresponden según su rol.

11.2. Alcance

Las pruebas de seguridad por niveles de usuario buscarán comprobar que, para cada rol de usuario, la sesión abierta en el sistema les permite hacer uso de las funcionalidades definidas y limitadas para el rol correspondiente, pero les impide acceder a cualquier otra funcionalidad para cuyo uso el rol no tenga autorización.

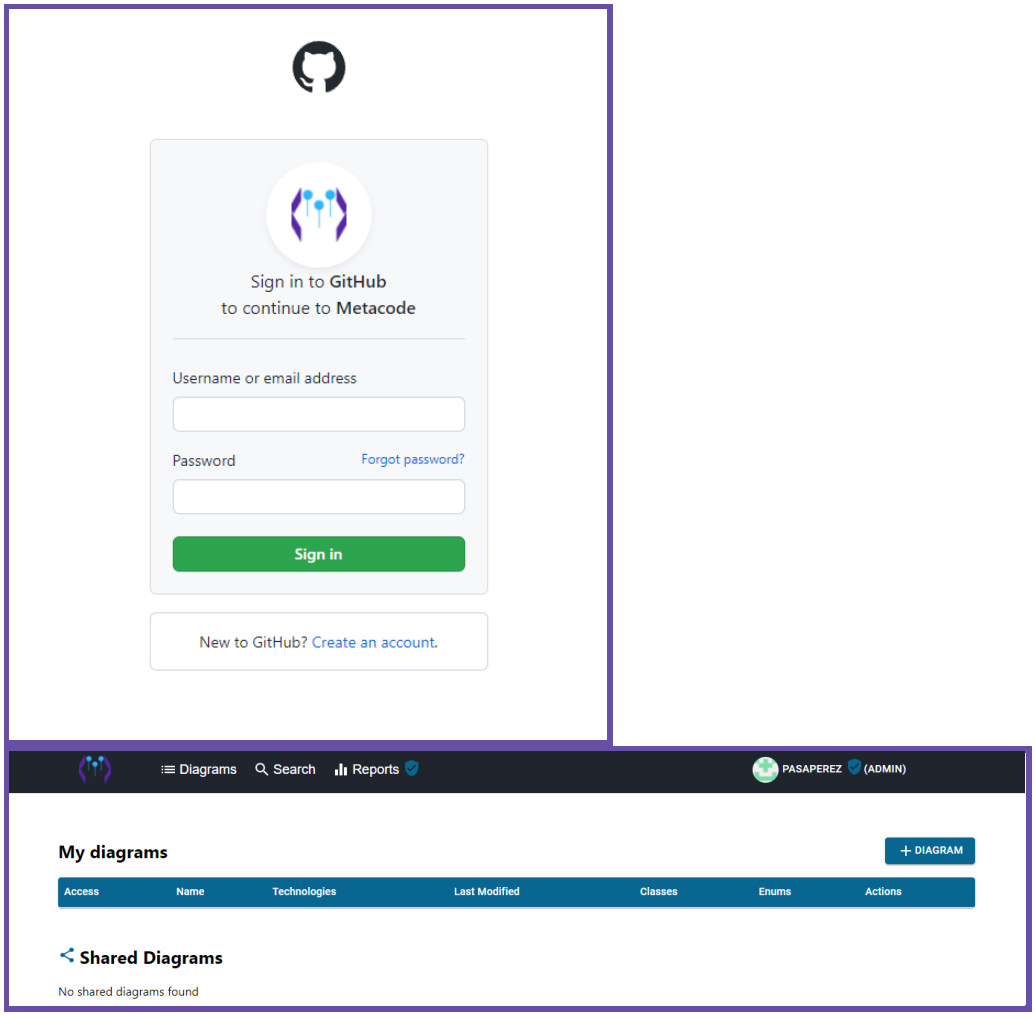
11.3. Realización

Estas pruebas se llevarán a cabo antes de las pruebas de integración.

CP013 - Cerrar sesión y volver a la página anterior

User story	HU002.
Fecha de ejecución	21/10/21.
Actor	Modelador.
Objetivo	Verificar que, al cerrar sesión y navegar luego a la página anterior, el sistema no permite visualizar información.
Datos de prueba	nombreUsuario: PASAPEREZ, contraseña: pass1234.
Precondiciones	Usuario registrado con rol básico de Editor.
Tipo de ejecución	Manual.
Tipo de prueba	De seguridad por niveles de usuario.
Medidas correctivas	N/A.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Clickear botón de “Login with GitHub” *(Captura de pantalla 01)*		
02	Ingresar credenciales de login de GitHub *(Captura de pantalla 02)*		
03	Clickear botón “Sign in” para confirmar inicio de sesión		
04	Clickear encima del nombre de Usuario.		
05	Hacer click en el botón “Logout” *(Captura de pantalla 03)*		
06	Retroceder con el navegador a una página de atrás.	El sistema muestra un mensaje de error que el usuario no está autorizado para ver el contenido. *(Captura de pantalla 04)*	El sistema muestra un mensaje de error que el usuario no está autorizado para ver el contenido. *(Captura de pantalla 04)*

Nº de Captura	Captura de Pantalla
01	
02	

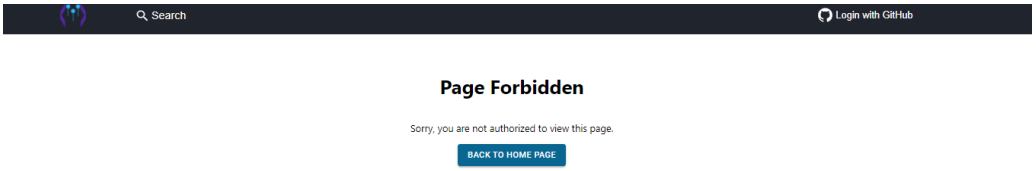
03	
04	

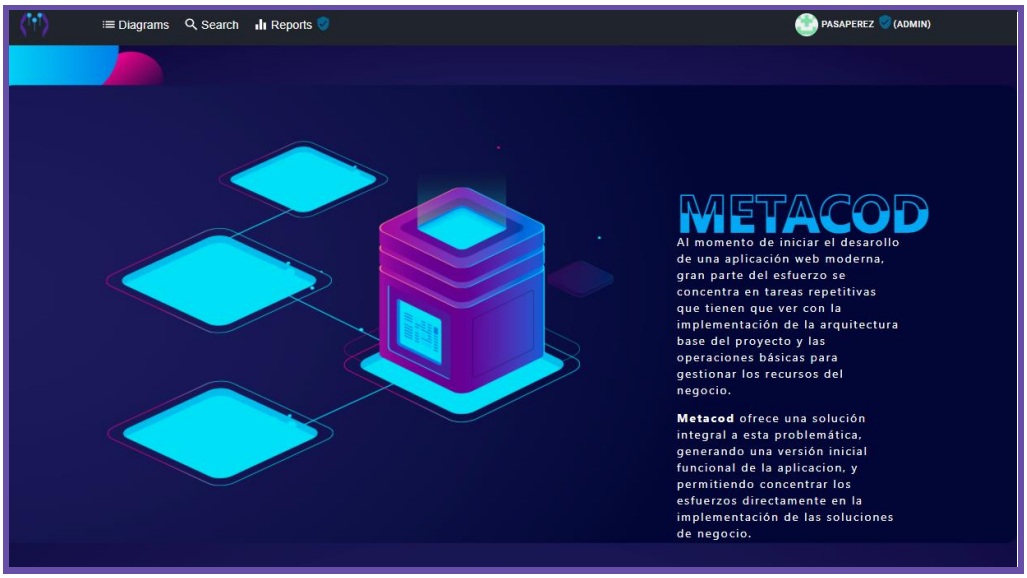
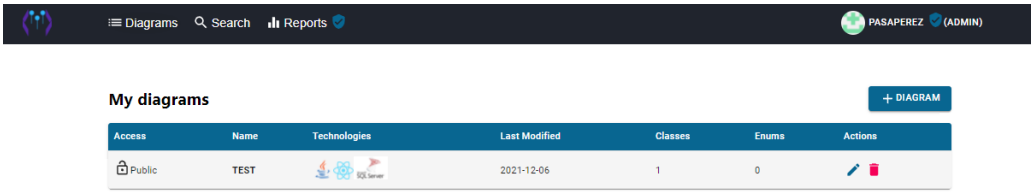
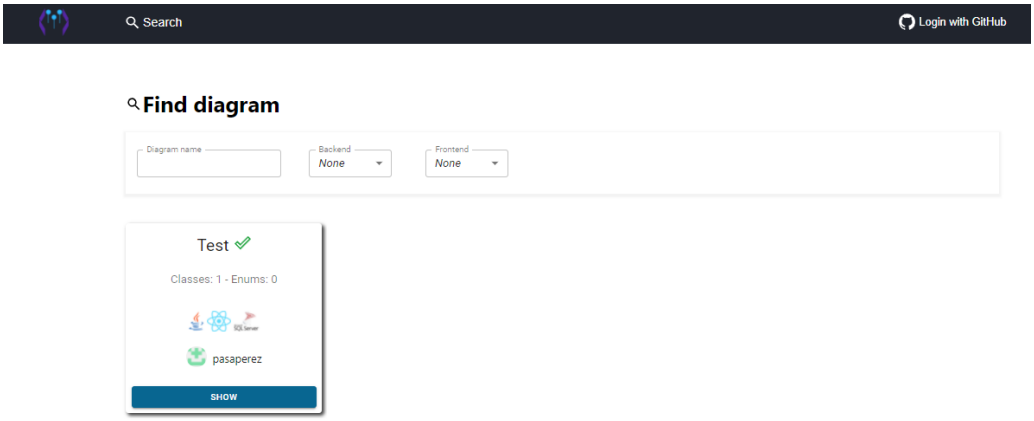
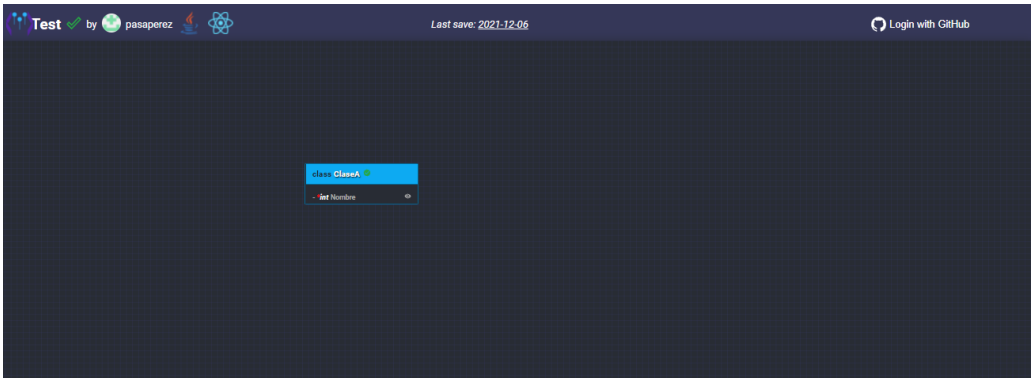
Tabla 69. CP013 Cerrar sesión y volver a la página anterior.

CP014 - Intentar visualizar un diagrama ajeno privado	
User story	HU003, HU004.
Fecha de ejecución	22/10/2021.
Actor	Modelador.
Objetivo	Verificar que, al cambiar la visibilidad de un diagrama de “Público” a “Privado” por el propietario del mismo, un usuario que se encuentra actualmente visualizando deja de tener acceso al mismo al refrescar la página del navegador.
Datos de prueba	idDiagrama: 2, visibilidadDiagrama: “Privado”.
Precondiciones	Diagrama existente con visibilidad “Público”, Usuario propietario del diagrama logueado en el sistema.
Tipo de ejecución	Manual.
Tipo de prueba	De seguridad por niveles de usuario.
Medidas correctivas	Verificar la visibilidad configurada al momento de abrir un diagrama.

N° de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Desde la interfaz correspondiente al usuario que accede al diagrama público, clickear botón “Search” *(Captura de pantalla 04)*		

02	Desde la interfaz correspondiente al usuario que accede al diagrama público, clickear botón “Show” del diagrama cuyo id es “2” *(Captura de pantalla 03)*		
03	Desde la interfaz correspondiente al usuario propietario del diagrama, clickear botón “Diagrams” *(Captura de pantalla 01)*		
04	Desde la interfaz correspondiente al usuario propietario del diagrama, clickear ícono “Editar” del diagrama con visibilidad “Público” existente *(Captura de pantalla 02)*		
05	Desde la interfaz correspondiente al usuario propietario del diagrama, clickear ícono “Configuración” *(Captura de pantalla 03)*		
06	Desde la interfaz correspondiente al usuario propietario del diagrama, dentro del diagrama, el usuario debe clickear en la opción “Privado” *(Captura de pantalla 05)*		
07	Desde la interfaz correspondiente al usuario propietario del diagrama, dentro del diagrama, el usuario debe clickear en el botón de “Save” *(Captura de pantalla 05)*	El usuario que accede al diagrama público le debe salir un mensaje de “Este diagrama es privado” y se le debe redireccionar a la página de sus diagramas	El usuario puede seguir viendo el diagrama, y si desea puede clonar este diagrama.

Nº de Captura	**Captura de Pantalla**

01	 Diagrams Search Reports PASAPEREZ (ADMIN) METACOD Al momento de iniciar el desarollo de una aplicación web moderna, gran parte del esfuerzo se concentra en tareas repetitivas que tienen que ver con la implementación de la arquitectura base del proyecto y las operaciones básicas para gestionar los recursos del negocio. Metacod ofrece una solución integral a esta problemática, generando una versión inicial funcional de la aplicacion, y permitiendo concentrar los esfuerzos directamente en la implementación de las soluciones de negocio.
02	 Diagrams Search Reports PASAPEREZ (ADMIN) My diagrams + DIAGRAM Access Name Technologies Last Modified Classes Enums Actions Public TEST 2021-12-06 1 0
03	 Search Login with GitHub Find diagram Diagram name Backend None Frontend None Test Classes: 1 - Enums: 0 pasaperez SHOW
04	 Test by pasaperez Last save: 2021-12-06 Login with GitHub class ClaseA - int Nombre

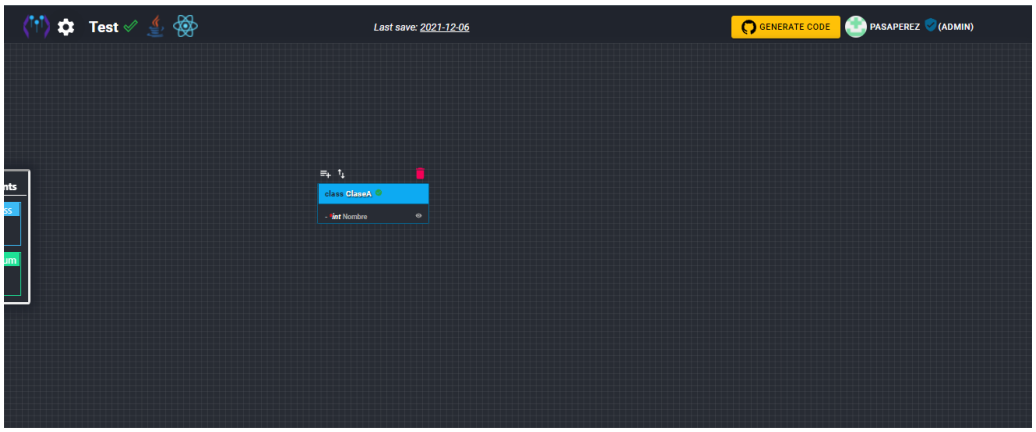
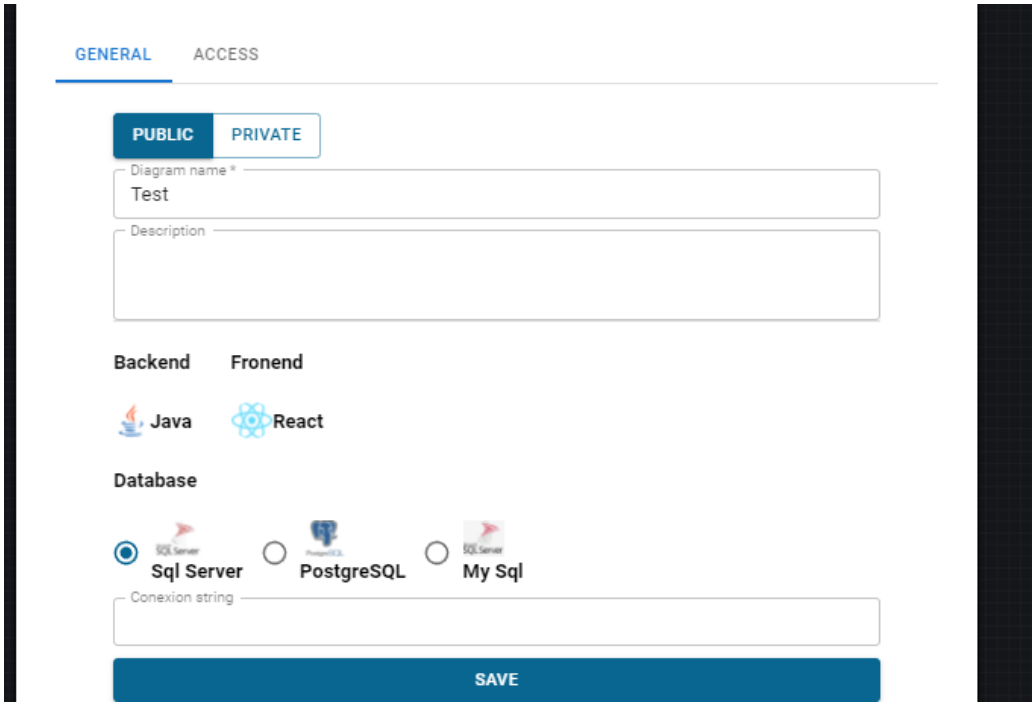
05	
05	

Tabla 70. CP014 Intentar visualizar un diagrama ajeno privado.

CP015 - Intentar generar un reporte sin tener rol de administrador	
User story	HU011, HU012, HU013, HU014, HU015.
Fecha de ejecución	21/11/21.
Actor	Modelador.
Objetivo	Verificar que un usuario que se encuentra actualmente logueado y que no tiene permisos de administrador, puede ver y acceder a los menús de generación de reportes.

Datos de prueba	-
Precondiciones	Usuario logueado en el sistema con rol de Editor y sin rol de Administrador.
Tipo de ejecución	Manual.
Tipo de prueba	De seguridad por niveles de usuario.
Medidas correctivas	Verificar que el menú de generación de reportes solicite una confirmación de credenciales al ingresar.

Nº de Paso	Acción a realizar	Resultado Esperado	Resultado Obtenido
01	Desde la interfaz correspondiente al usuario que accede al diagrama público, clickear el botón “Diagrams” *(Captura de pantalla 01)*		
02	Desde la interfaz correspondiente al usuario que muestra la lista de sus diagramas *(Captura de pantalla 02)*	El sistema muestra una opción de los diagramas “Diagrams” una opción de diagramas públicos “Search” y el nombre de usuario que permite al usuario una única acción de desloguear del sistema y no muestra la opción de ingresar a reportes “Reports” *(Captura de pantalla 03)*	El sistema muestra una opción de los diagramas “Diagrams” una opción de diagramas públicos “Search” y el nombre de usuario que permite al usuario una única acción de desloguear del sistema y no muestra la opción de ingresar a reportes “Reports” *(Captura de pantalla 03)*

Nº de Captura	Captura de Pantalla

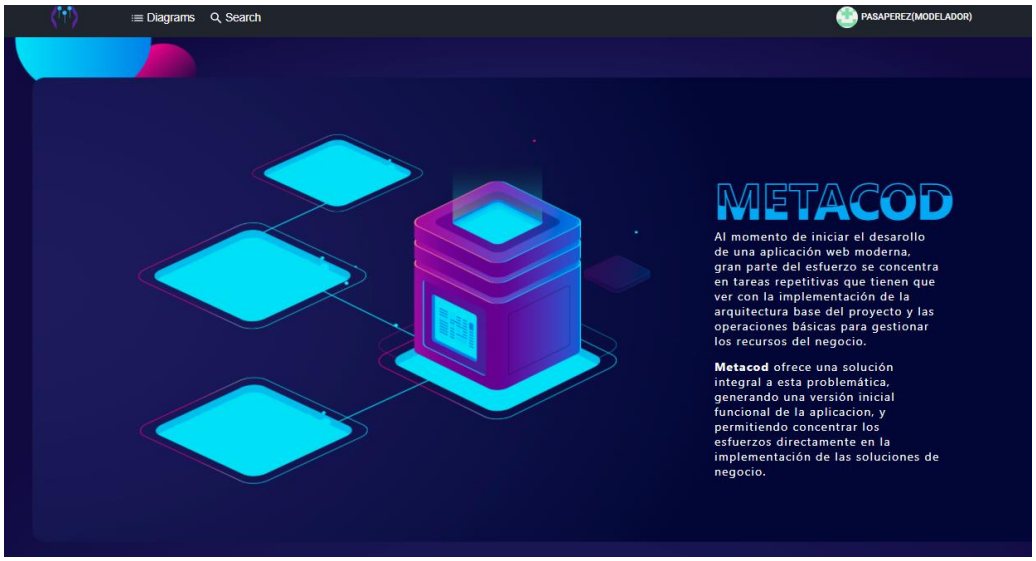
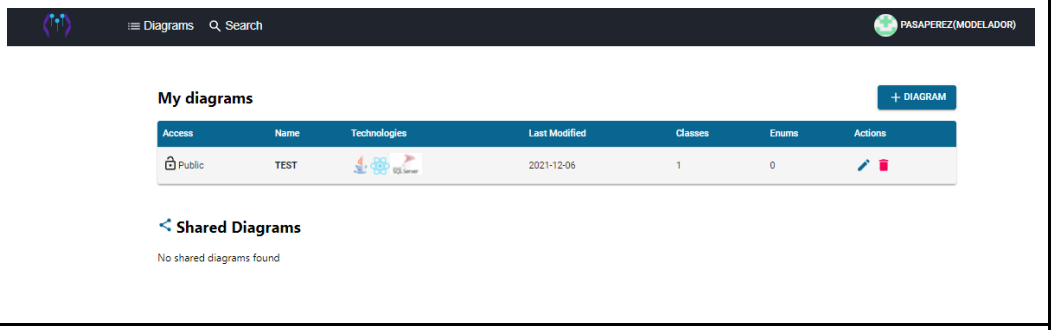
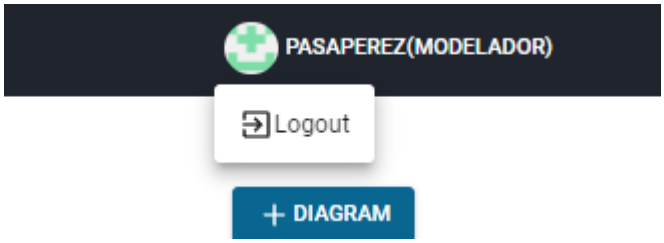
01	
02	
03	

Tabla 71. CP015 Intentar generar un reporte sin tener rol de administrador.

Manual de usuario del Sistema completo

(Anexo 1: Manual de usuario del Sistema completo)

Planificación de Implementación del Sistema

12. Plan de implementación

El sistema está pensado para funcionar en la web y para ser utilizado por múltiples usuarios como repositorio de sus diagramas por ende debe estar montado en un servidor.

12.1. Estrategias del plan de implementación / Método de Conversión

Se debe tener en cuenta, que el sistema provee un servicio nuevo, por lo que no deberá reemplazar a un sistema ya existente, por lo que el método de conversión elegido para su implementación es directo.

Estrategias de implementación: **Directa**

Para ello se harán los siguientes pasos:

12.2. Determinación del equipo de implementación

El equipo de trabajo consta de un implementador encargado de realizar una revisión sobre el entorno de producción y de la instalación propiamente dicha, y un analista que tiene la tarea de dar soporte al cliente.

12.3. Instalación y configuración del servidor web

Como servidor web hemos elegido un servidor web en Amazon (Nginx) para realizar la instalación se harán los siguientes pasos:

- Login en AWS.
- Elegir servicio.
- Ubicación del servicio.
- Elegir Sistema Operativo.
- Elegir tipo de instancia.
- Configuración de la instancia.
- Configurar almacenamiento.
- Añadir Tag.
- Configurar grupo de seguridad.
- Previsualización y lanzamiento.
- Llaves SSH.
- Instancia instalada.
- Conectarse a la instancia de Linux.

12.4. Instalación y configuración de la base de datos

La base de datos será SQL Server y se montará en AWS. para realizar la instalación se harán los siguientes pasos:

- Acceder a la consola de RDS

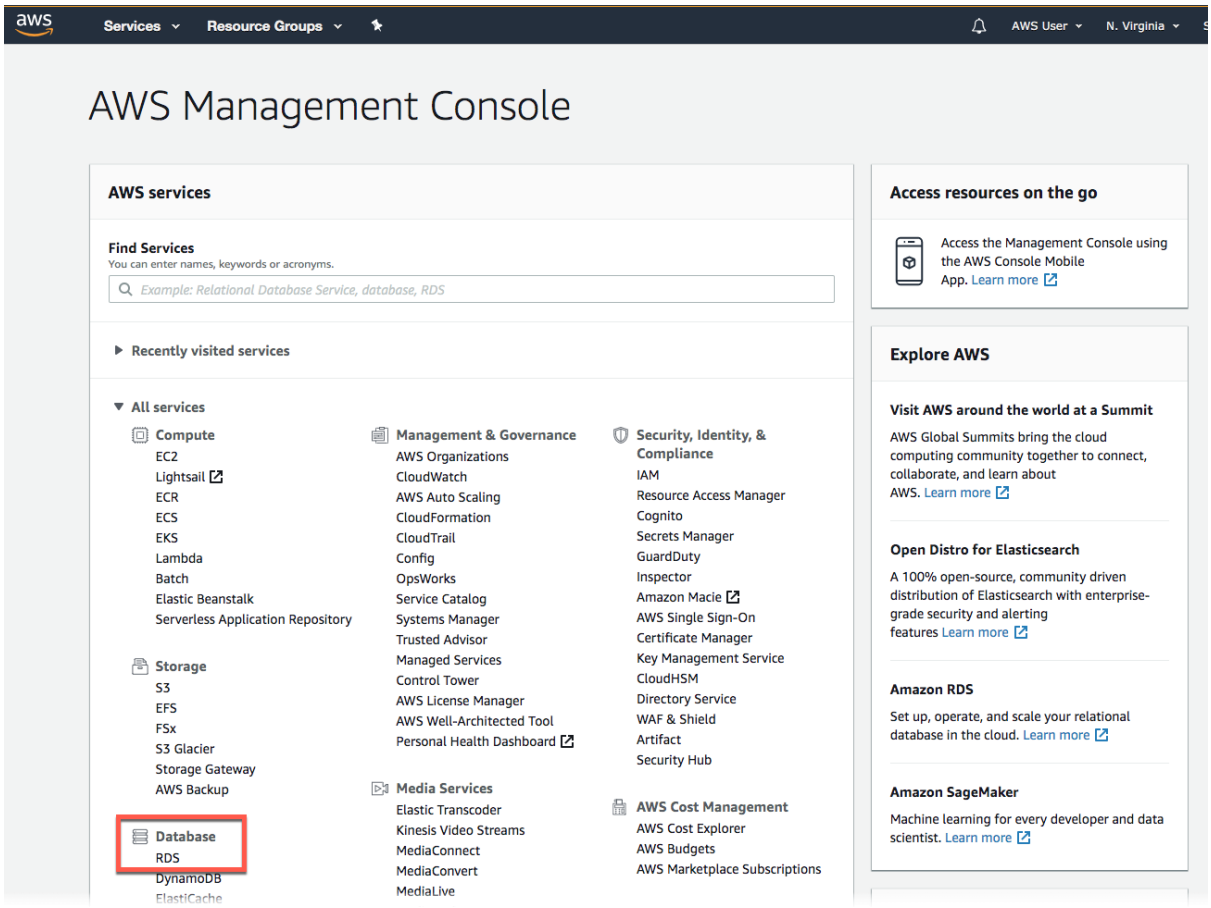


Figura 130. Acceder a consola RDS.

- Crear una instancia de base de datos Microsoft SQL Server:
    - En este paso, utilizaremos Amazon RDS para crear una instancia de base de datos Microsoft SQL Server con una instancia de clase db.t2.micro, 20 GB de almacenamiento y copias de seguridad automatizados activados con un periodo de retención de un día.
- Seleccionamos la *región* en la que deseamos crear la instancia de base de datos.

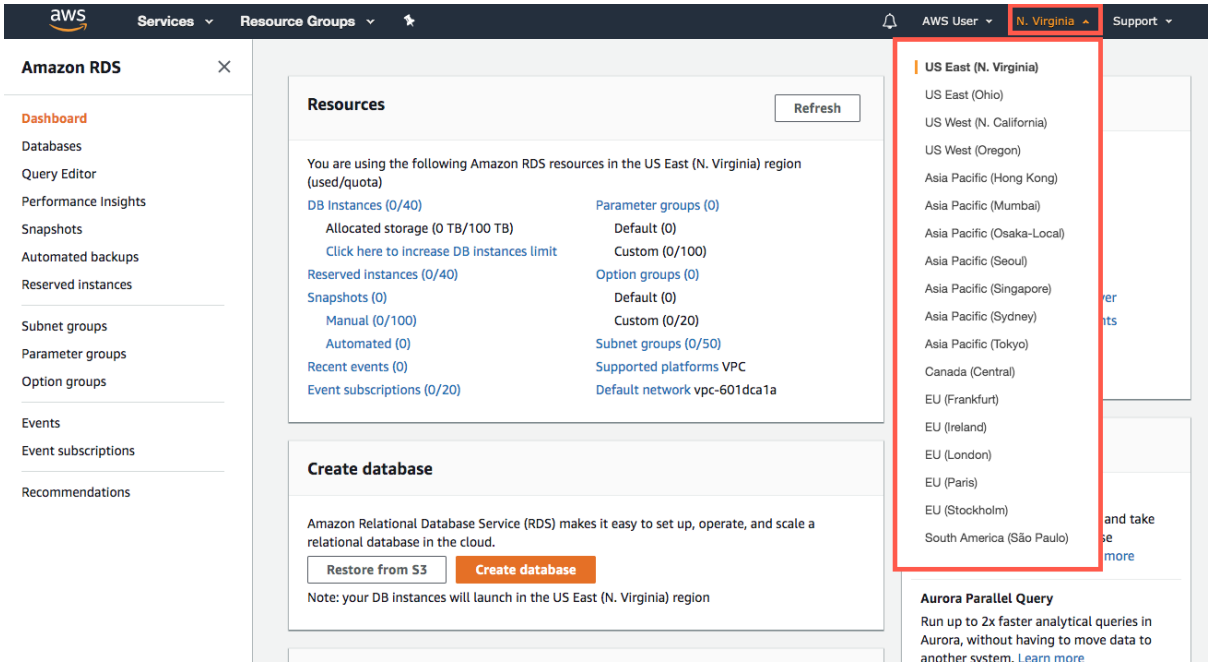


Figura 131. Selección de región de BD.

- Damos click en “create database”.

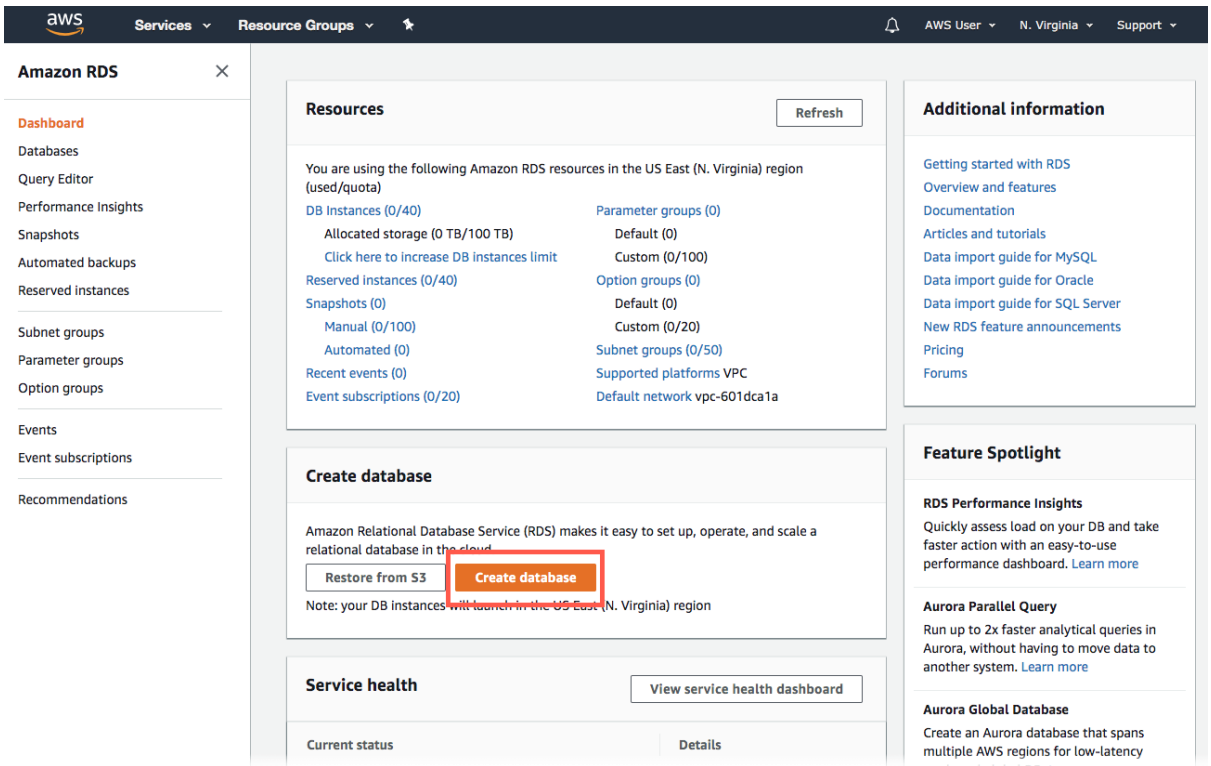


Figura 132. Create database.

- Seleccionamos “SQL Server Express Edition” y “Only enable options eligible for RDS Free Usage Tier” (Permitir solo opciones elegibles para la capa de uso gratuita de RDS) y luego hacemos clic en “Next”.

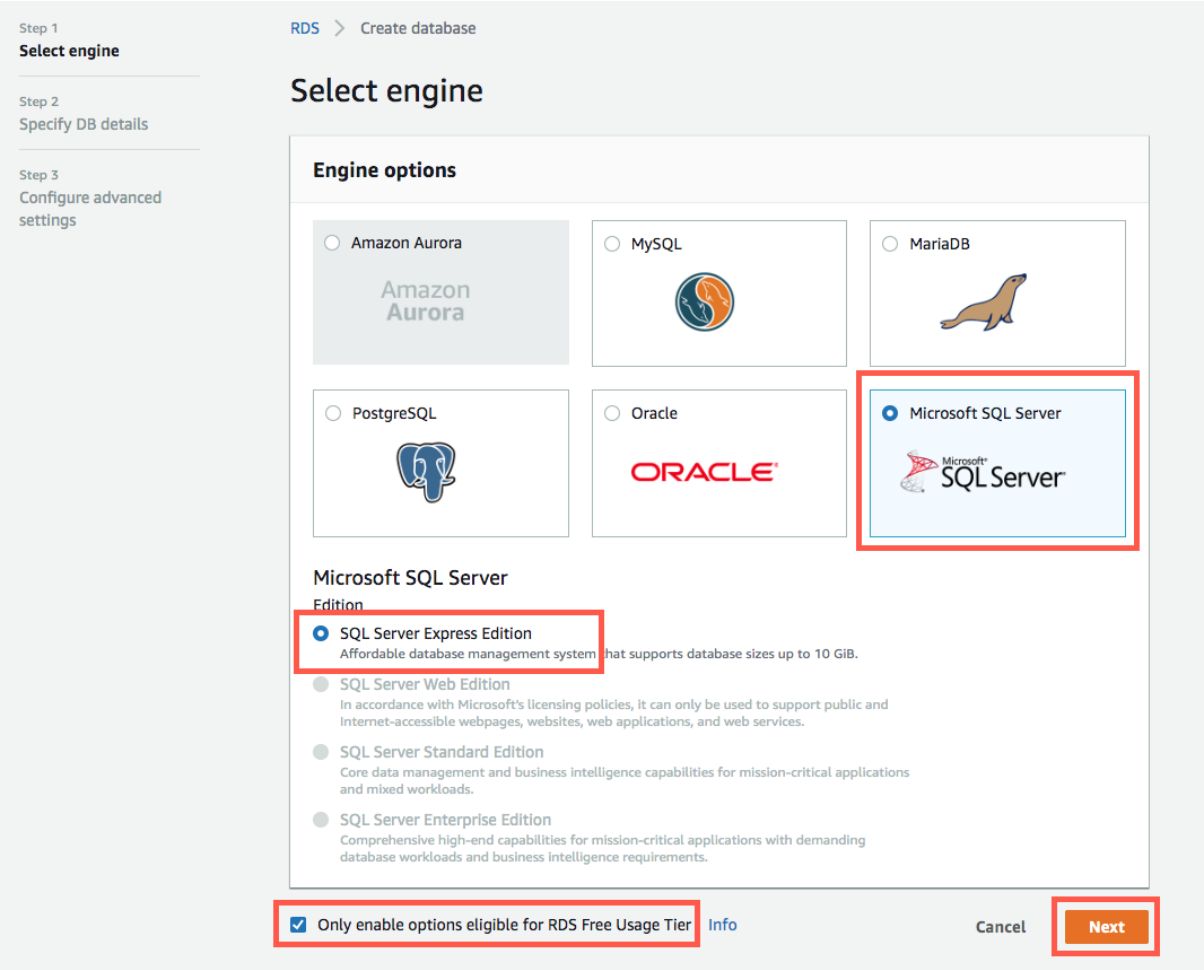


Figura 133. Selección SQLServer.

- Configuramos y creamos la instancia de base de datos.

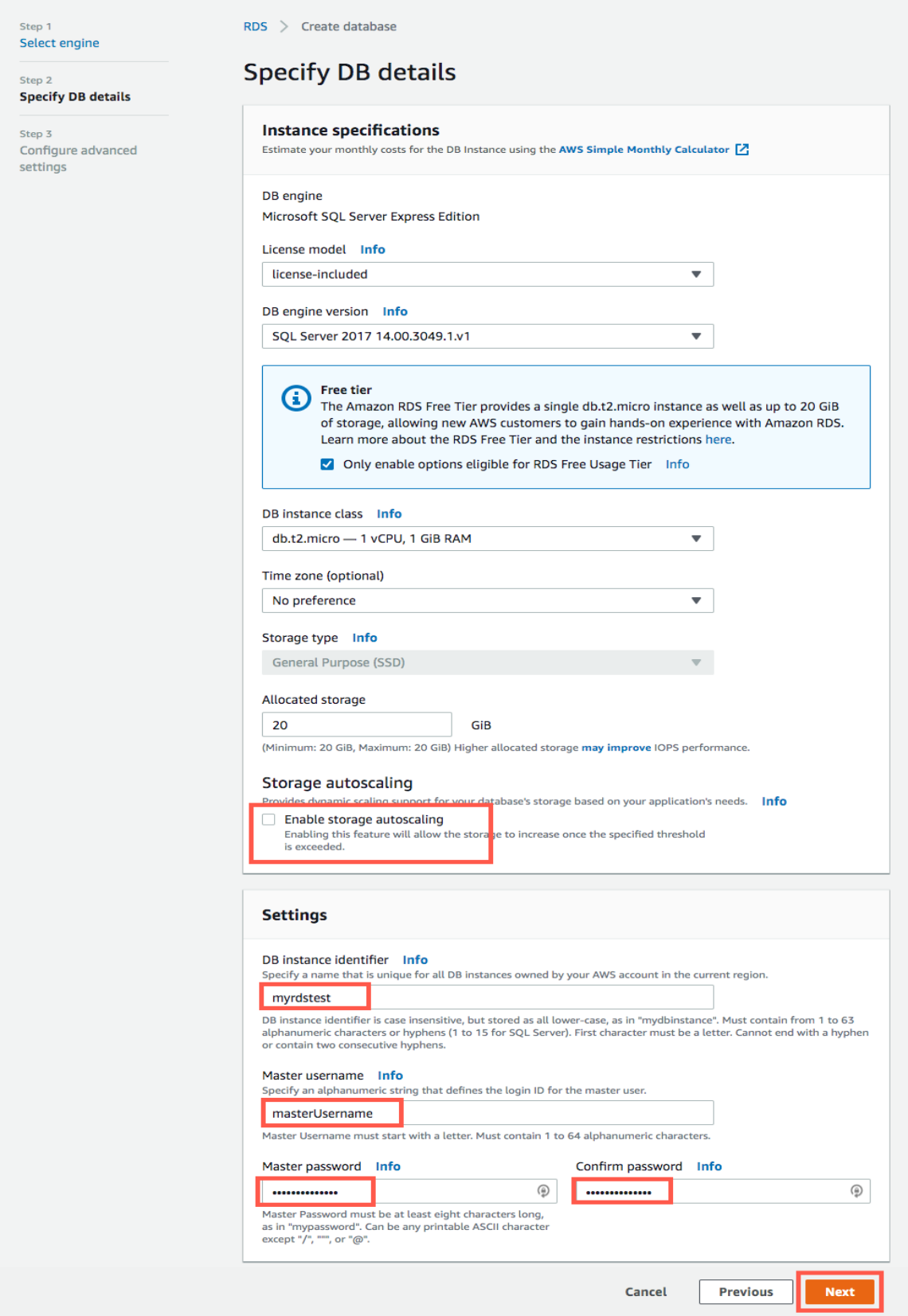


Figura 134. Crear instancia de BD.

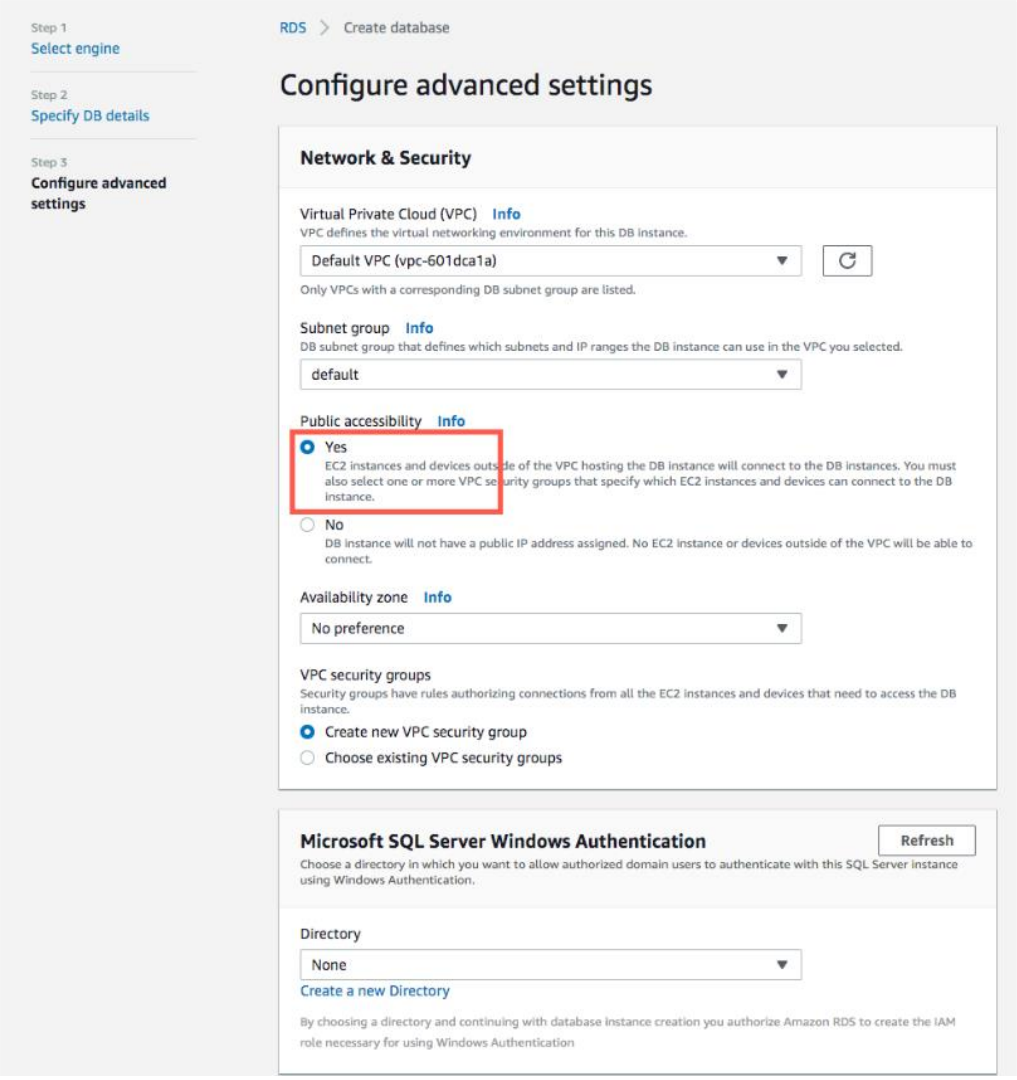


Figura 135. Configurar instancia de BD.

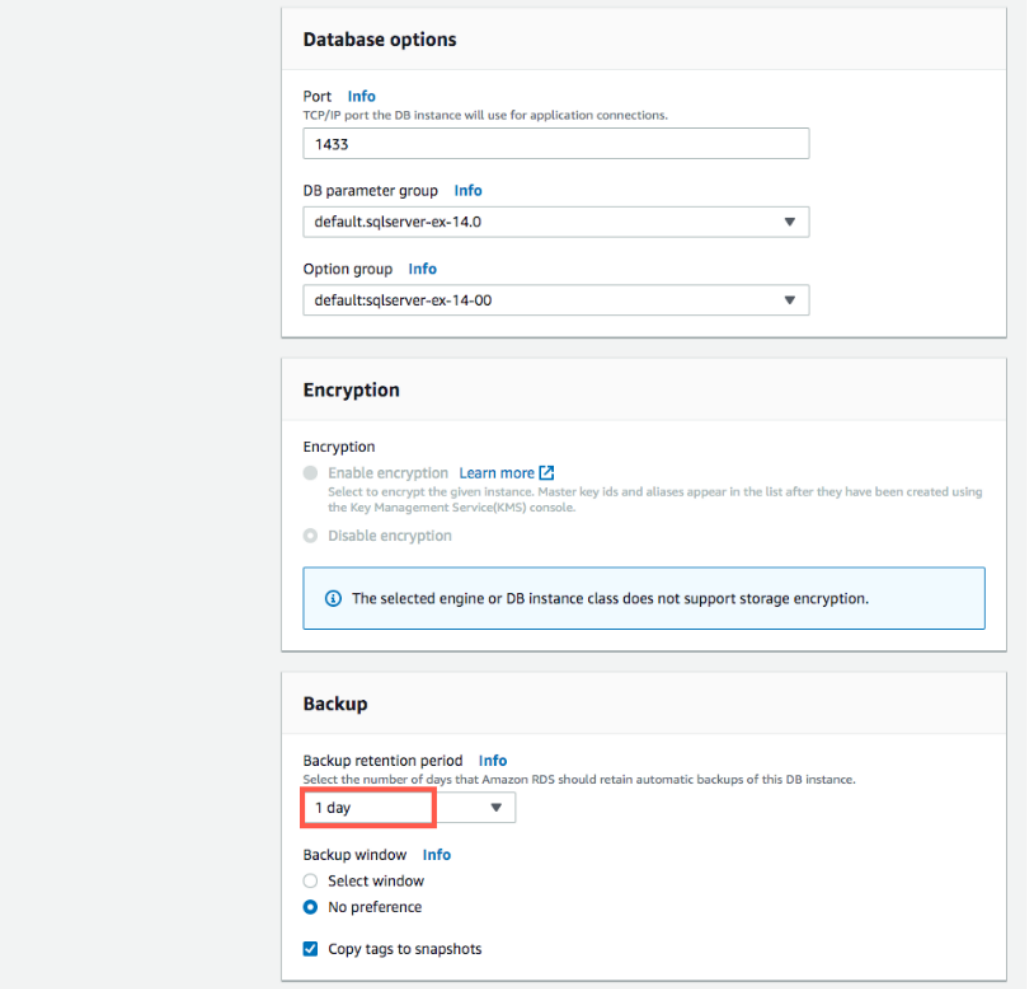


Figura 136. Configurar instancia de BD

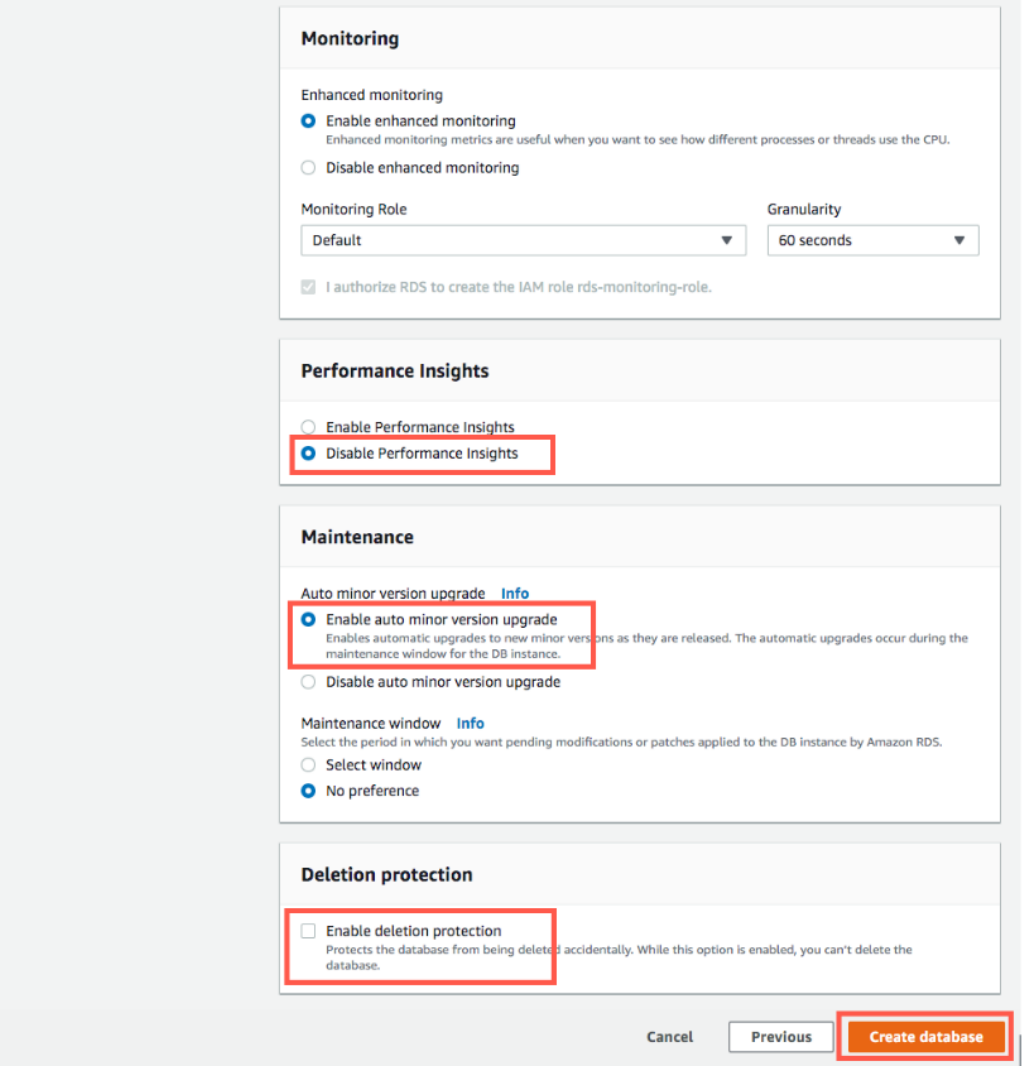


Figura 137. Configurar instancia de BD.

12.5. Inicio del sistema y carga inicial de datos

Una vez que se haya creado la instancia de base de datos y se encuentre en estado "available" (disponible), puede conectarse a una base de datos de la instancia de base de datos con cualquier cliente SQL. En este paso, descargamos Microsoft SQL Server Management Studio Express, un cliente popular para SQL Server.
Nos conectaremos a la base de datos Microsoft SQL Server y se despliegan los scripts iniciales en la instancia de la base de datos creada para configurar los datos iniciales de la aplicación.

El script de inicialización contendrá la siguiente información:

- Roles del usuario en el sistema.
- Roles del usuario en un diagrama.
- Usuario administrador.
- Tipos de relaciones.

- Multiplicidades.
- Lenguajes Backend.
- Lenguajes Frontend.
- Tipos de datos por lenguaje backend.
- Modificadores de acceso de diagrama.

12.6. Implementación de políticas de backup

Para obtener una capacidad adicional de recuperación ante desastres, configuraremos una instancia de base de datos de Amazon RDS para replicar las instantáneas y los registros de transacciones a una región AWS de destino de su elección. Cuando se configura la replicación de la copia de seguridad para una instancia de base de datos, RDS inicia una copia entre regiones de todas las instantáneas y los registros de transacciones tan pronto como estén listos en la instancia de base de datos.

El almacenamiento de copia de seguridad de Amazon RDS de cada región de AWS se compone de las copias de seguridad automatizadas y las instantáneas de base de datos manuales de esa región. El total del espacio de almacenamiento de copias de seguridad equivale a la suma del almacenamiento de todas las copias de seguridad de una región. Mover una instantánea de base de datos a otra región incrementa el almacenamiento de backup en la región de destino. Las copias de seguridad se almacenan en Amazon S3.

Para habilitar las copias de seguridad automatizadas, usaremos el comando de la AWS CLI `modify-db-instance`.
Incluiremos los siguientes parámetros:

- `--db-instance-identifier`
- `--backup-retention-period`
- `--apply-immediately` o bien `--no-apply-immediately`

Habilitaremos las copias de seguridad automatizadas estableciendo el periodo de retención de copia de seguridad en 3 días. Los cambios se aplican inmediatamente.

```
aws rds modify-db-instance \ --db-instance-identifier mydbinstance \ --backup-retention-period 3 \ --apply-immediately
```

13. Duración de la implementación

La duración de la implementación está estrictamente ligada a los resultados de la implementación de los servicios en AWS.

Dado que es un proveedor confiable y no se esperan errores de funcionamiento salvo algún problema de configuración, se estima que el proceso no durará más de 2 semanas.

Trabajo Práctico Anual Nº2

CAPITULO 1

Actividades

En este capítulo se realiza un análisis de todas las fases, actividades y tareas que son propias de un proyecto estandarizado y a la vez se aplica una evaluación para elegir las tareas necesarias a aplicar en el proyecto propio de Metacod. Además, se realiza una evaluación subjetiva de los integrantes para poder realizar una estimación de la duración de cada tarea a aplicar.

1. Definición y descripción de actividades

Se define cada actividad a realizar y se escribe una breve descripción de la misma. Entre las actividades que realizamos para la realización de este proyecto tenemos:

1.1. Planificación

1.1.1. Formación del Equipo

Se conforma un grupo de 5 participantes para realizar todo el proyecto en el plazo de un ciclo educativo de un año (9 meses calendario), todos los integrantes pueden cursar la catedra Proyecto Final.

1.1.2. Presentación de distintas propuestas de sistemas a realizar

Reunión al inicio del cursado para que cada miembro del equipo presente ideas para realizar en el proyecto. Y la toma de decisión de la idea aceptada por todos los integrantes del equipo.

1.1.3. Presentación de la idea a los docentes sobre el proyecto

Presentación de la idea elegida en las consultas de Proyecto Final a los profesores de la catedra.

1.1.4. Definición de Herramientas internas de comunicación

Se seleccionan las herramientas que serán utilizadas para la comunicación entre los miembros del equipo, ya sea síncrona o asíncronamente y todas sus alternativas por si estas fallan.

1.1.5. Definición de Herramientas de Gestión de la configuración

Se evalúan las posibilidades y se definen las herramientas a aplicar en cuanto a cómo se lleva adelante un control sobre los artefactos de software generados durante la ejecución del proyecto.

1.1.6. Definición de metodología a utilizar

Se evalúan las metodologías conocidas por cada integrante y se decide cuál de estas se utilizará en la codificación, diseño e incluso todo el desarrollo del proyecto.

1.1.7. Definir todas las actividades para el desarrollo del proyecto

Se elabora una lista detallada de todas las etapas y actividades que están involucradas durante todo el proceso del proyecto, las cuales siguen la metodología sugerida por la catedra (Planificación Investigación y Relevamiento, Análisis, Diseño, Desarrollo, Prueba o Testing, Implementación y Realización de Documentación), cada Etapa está compuesta por diferentes actividades y tareas.

1.1.8. Estimación temporal de todas las actividades para el desarrollo del proyecto

Se determina la duración aproximada de cada una de las actividades definidas en base a diferentes criterios como: experiencia previa en planificación, consultas con integrantes de años previos, fechas de planificación de la catedra, e hitos u entregables solicitados. Se establece a su vez un orden en el cual deben ser realizadas.

1.1.9. Realizar diagrama de tiempos

Para una organización sencilla de tanta información se la presenta a través de un diagrama de tiempos o Gantt.

1.1.10. Diseñar los perfiles para cada puesto y Asignación entre los miembros del equipo

De acuerdo con las características del Proyecto, se definen los puestos de trabajo que serán requeridos para su ejecución, a su vez con las estimaciones se determina cómo está compuesto el equipo, en cuanto a la cantidad de Puestos por perfil, y se asignan las responsabilidades a los miembros con los que cuenta el equipo.

1.1.11. Realizar estudios de factibilidad

Realizar los estudios de:

- Factibilidad técnica
- Factibilidad operativa
- Factibilidad ambiental
- Factibilidad legal
- Factibilidad económica y financiera

1.1.12. Realizar Análisis de Riesgos e impacto ambiental

Con la finalidad de orientar la toma de decisiones en la evaluación del proyecto se hace este análisis que se formula con base en información que tiene la menor incertidumbre posible para medir las posibilidades de éxito o fracaso de una inversión y el impacto que tendrá el proyecto en el contexto que se implementará.

1.2. Relevamiento y Análisis de Sistemas

1.2.1. Definición de Objetivos y Descripción general del proyecto

Se definen los objetivos y subobjetivos, a nivel general, necesarios para llevar a cabo el proyecto desde cero según las necesidades que nos brinda la catedra.

1.2.2. Definición de Marco Teórico o Investigación Preliminar

Se realiza un análisis con base en la propuesta presentada y se investiga que áreas y actividades se pueden desarrollar sobre esta base. De esta manera se comprende mejor el tipo de sistema que se quiere desarrollar, la oferta del mercado, y se puede detectar potenciales funcionalidades a utilizar.

1.2.3. Relevamiento General de 5 sistemas similares

Se buscan sistemas similares ya desarrollados para poder comparar con las potenciales funcionalidades que se necesitan. Se analiza brevemente la documentación de estos sistemas o aplicaciones.

1.2.4. Relevamiento Detallado de 5 sistemas similares

- Análisis y relevamiento de funciones propias de cada sistema
  Teniendo como base los sistemas o aplicaciones que ya se estudiaron se analiza detalladamente cada funcionalidad, y se observa mediante su documentación como realiza sus tareas.
- Desarrollo de modelo lógico de cada sistema
  Se realiza un modelo de todas las partes involucradas del sistema que permiten llevar a cabo su tarea principal. Este modelo nos permite comprender el sistema en forma global.
- Evaluación de ventajas y desventajas de cada sistema
  Se evalúan las ventajas de estos sistemas para usar como referencia a aplicar en nuestro desarrollo, y se evalúan las desventajas, para evitarlas o para resolver necesidades que presentan.

1.2.5. Análisis y evaluación de problemáticas y necesidades encontradas del contexto de los sistemas

Una vez estudiados los diferentes sistemas competentes, se detectan funcionalidades no desarrolladas y/o necesidades no cubiertas en donde se plantean soluciones posibles para poder satisfacer estas necesidades. Esto se realiza a nivel general con respecto a los 5 sistemas similares relevados.

1.2.6. Redacción de los objetivos y alcances del proyecto aplicado a nuestro sistema

Se analizan las funcionalidades principales del proyecto, basándose en los objetivos principales del proyecto, para poder delimitar su alcance y enfocarse en aquellas necesidades que no están siendo solucionadas y cómo se pretende dar solución a ellas.

1.2.7. Definición y división preliminar de módulos de nuestro sistema

Se hace una primera división del sistema de forma modular para poder simplificar el desarrollo de este. Se enumeran todos sus módulos y se definen sus primeras tareas principales para poder dar soporte a los objetivos y necesidades, llegando a los alcances previstos, pero no superando los límites del sistema.

1.3. Investigación y capacitación

1.3.1. Investigación general de tecnologías

Se investiga a nivel general las tecnologías conocidas por los integrantes del equipo, y se investiga en tecnologías que complementen o sean necesarias para cubrir la base arquitectónica que requiere el sistema según sus objetivos y alcances.

1.3.2. Investigación y capacitación sobre herramientas de desarrollo web

Los integrantes se capacitan en tecnologías de desarrollo web y los integrantes que se les asignara el rol de desarrollador en esta área se deben capacitar con mayor detalle.

1.3.3. Profundización de conocimientos de Bases de Datos

Se hace un repaso general y detallado de las tecnologías y procedimientos de bases de datos, con respecto a cátedras ya cursadas y enfocándose en las necesidades tanto desde el Frontend como de Backend, ya que ambas interactuaran con estos motores al menos de forma indirecta.

1.3.4. Investigación de DDD

Se hace una investigación y capacitación de parte de todos los integrantes del enfoque de “Diseño Guiado Por El Dominio” ya que este enfoque guía a todo el proyecto.

1.3.5. Profundizar conocimiento en Virtualizadores

Todos los integrantes hacen un repaso general de las tecnologías de virtualización y de sus alternativas para comprender las posibles opciones a utilizar en la implementación del sistema final.

1.3.6. Recordar y profundizar sobre Versionadores

Los integrantes desarrolladores hacen un repaso y una capacitación practica sobre versionadores de código para aprender buenas prácticas y poder aplicarlas.

1.4. Diseño

1.4.1. Definición formal de objetivos, alcances del sistema y límites del sistema

Se realiza una definición más exhaustiva, formal y orientada al diseño de los objetivos, límites y alcances definidos anteriormente, para brindar un enfoque más detallado de los mismos. Se refinan las funcionalidades pensadas para el sistema y las funcionalidades que deben ser implementadas para satisfacer las necesidades detectadas. Esta etapa puede ser realizada múltiples veces a lo largo del diseño para refinar detalles.

1.4.2. Definición de Salidas del Sistema

Se definen las salidas de información que va a mostrar nuestro sistema por cada funcionalidad, enfocándonos en el tipo y como será mostrada en las posibles implementaciones.

1.4.3. Definir los roles y las funcionalidades permitidos para cada rol

Análisis y selección de los roles para los usuarios del sistema (nombre, descripción y tipo) y redacción de las funcionalidades y permisos que tendrá cada uno al usar el sistema. Diagrama de Actores.

1.4.4. Diagramación de Casos de Uso

A partir de las funciones y de los roles analizados, se realiza el diagrama de casos de usos del sistema con todos los actores y los casos de uso que se tiene cada módulo para poder dar soporte a todas las salidas que se han definido.
Lista de Casos de Uso:

- ABM Usuario.
- ABM Rol de Usuario.
- ABM Diagrama.
- ABM Permisos de Diagrama.
- ABM Clase.
- ABM Interface.
- ABM Atributo.
- ABM Operación.

- ABM Relación.
- ABM Herencia.
- ABM Implementación.
- ABM Tipo de datos.
- Consultar Diagrama.
- Generar Reporte/Estadística.
- Iniciar Sesión.
- Recupero de contraseña.
- Buscar Diagrama.
- Exportar Diagrama.
- Compartir Diagrama.

1.4.4.1. Descripción y flujos de sucesos de los casos de uso

Para mayor comprensión de cada uno de estos Casos de Uso, vamos a realizar la descripción de estos teniendo en cuenta el siguiente formato:

- Nombre del caso de uso
- Número: según la numeración dada anteriormente.
- Prioridad (A/B/C): considerando A al de mayor prioridad y C al de menor.
- Camino Alterno: posibles derivaciones de funcionamiento que puede tener el caso de uso.
- Descripción: para dar una pequeña narración explicativa del funcionamiento del caso de uso.
- Actor: agente que se encarga de realizar o disparar la ejecución de la funcionalidad.
- Precondición: aspecto previo considerado como verdadero que no debe ser comprobado.
- Parámetros de entrada: datos externos ingresados por el actor para llevar a cabo la funcionalidad.
- Estado inicial: son las entidades, atributos y relaciones que deben ser verificadas en la realización del caso de uso. Nota: en el presente proyecto, el estado inicial de cada uno de los Casos de Uso se considera verificado, para simplificar la lectura de este.
- Estado final: son las entidades, atributos y relaciones que son el resultado de la ejecución del caso de uso.

1.4.5. Diagramación de Modelo de Clases

Se representan los objetos de nuestro sistema mediante clases que permitirá dar soporte a todos los casos de uso. Diagrama de clases.

1.4.6. Realización de descripción de Casos de uso

Se realiza la secuencia detallada de pasos que debe realizar el usuario para lograr ejecutar un caso de uso correctamente, incluyendo control de errores y salidas planeadas. Descripción de Flujo de Sucesos.

1.4.7. Diagramación de base de datos

Se define el modelo a implementar en las diversas bases de datos que permitirán persistir la información relevante de nuestro sistema. Modelo Entidad Relación.

1.4.8. Maquetación de Interfaces gráficas

Se diseñan las interfaces de usuario de cada caso de uso, para lograr un mejor entendimiento de su funcionamiento y ver cómo interactúa el usuario con el mismo. Prototipo de Interfaces Graficas UI.

1.5. Desarrollo

1.5.1. Configuración de Gestionador de Versionado

Se hace la configuración que pueda soportar todo el versionado y trabajo en equipo para el desarrollo de todo el sistema diseñado, respetando las buenas prácticas aprendidas. Se hace la configuración del repositorio remoto y la prueba del acceso de todos los integrantes, tanto desarrolladores como documentadores.

1.5.2. Configuración de entornos de desarrollo a utilizar

Se realiza una configuración uniforme del entorno de desarrollo, IDEs y de ejecución de pruebas, de cada uno de los integrantes para mantener la compatibilidad al desarrollar.

1.5.3. Llevar a cabo codificación

Se realiza la implementación del diseño. Esta actividad también puede ser modular, iterativa, incremental o de división arbitraria. Se realiza según la metodología elegida para el proyecto, sea desarrollo Tradicional, Ágil, híbrido o una alternativa.

- Módulo de Roles, Usuarios y Login.
- Módulo de Gestión de diagramas.
- Módulo generador código Api en C#.
- Módulo generador código Api en Java 8.
- Módulo generador código Frontend Angular.
- Módulo generador código Frontend React.
- Módulo exportador del código a un repositorio.
- Módulo de Reportes

1.6. Testing

1.6.1. Descripción de Test Unitarios, Test cases y Test de componentes

Se realiza la descripción de los test a utilizar para probar cada porción de codigo o componente a fin de evaluar que se siguen las líneas base para poder obtener las salidas.

1.6.2. Descripción de Test Modulares

Se describe el paso a paso o la manera o procedimiento de probar todos los componentes de un módulo al completo.

1.6.3. Descripción de Test de integración

Se realiza la descripción del procedimiento a realizar cuando se haga la ejecución de pruebas de distintos módulos finalizados para evaluar cómo funcionan sus interfaces.

1.6.4. Descripción de Test de aceptación

Por último, se especifica como se realizarán las pruebas a nivel global que nos mostraran las salidas que se han definido para todo el sistema.

1.7. Documentación

1.7.1. Desarrollar manual de usuario

Se hace la documentación de todo el sistema y de cada una de las funcionalidades que están disponibles al usuario, además de los procedimientos que debe realizar este para poder obtener las salidas necesarias.

1.7.2. Realizar plan de capacitación

Se define un plan para realizar una posible capacitación a distintos usuarios a donde estos necesiten una implementación del sistema, que, como y de qué manera se realizara la posible capacitación de nuestro sistema.

1.8. Implementación

1.8.1. Ejecución de Test Modulares

Se realiza la ejecución y documentación de los test. Se realiza especial cobertura en los módulos que se encargan de la seguridad y de la carga de datos clave para el sistema.

1.8.2. Ejecución de Test de Integración

Se realiza la ejecución planeada de los módulos y el funcionamiento de sus interfases y se documenta su integración global.

1.8.3. Ejecución de Test de aceptación

Se ejecutan las pruebas del sistema en su conjunto, evaluando cada una de las funcionalidades.

1.8.4. Instalación y configuración de Backend y Frontend

Se hace el paso de los componentes desarrollados del área de desarrollo al área de producción en la que se presentara el sistema.

1.8.5. Instalación y configuración de Base de Datos

Se hace el paso de las estructuras de datos desarrolladas del área de desarrollo al área de producción en la que se presentara el sistema. Además, se hace la carga de Datos de presentación.

1.9. Hitos

- Presentación TP1 anual
  - Entregable 1 TP1: 23/3/2021 (Formación de Grupo e Idea inicial)
  - Entregable 2 TP1: 4/5/2021 (Relevamiento y Análisis de Sistemas)
  - Entregable 3 TP1: 8/6/2021 (Diseño y Testing)
  - Entregable 4 TP1: 2/11/2021 (Investigación y Capacitación, Documentación e Implementación)
- Presentación TP2 anual
  - Entregable 1 TP2: 4/5/2021 (Planificación Inicial)
  - Entregable 2 TP2: 8/6/2021 (Planificación Análisis de Factibilidad)
- Presentación TP1 Integrador (20/4/2021-18/5/2021)
- Presentación TP2 Integrador (17/8/2021-24/8/2021)
  - Diseño inicial de Póster para congreso CONAIISI
  - Entregable 1 Poster: 28/9/2021 (1° Revisión)
  - Entregable 2 Poster: 12/10/2021 (2° Revisión)
  - Diseño final y exposición de Póster para congreso CONAIISI
  - 16/11/2021 (15° Exposición de Proyectos de Sistemas)
  - Presentación de DEMO 1
  - 14/9/2021 (Según el Diseño y lo alcanzado a Implementar)
  - Presentación de DEMO 2
  - 12/10/2021 (Implementación completa)
  - Presentación de Sistema DEMO 3
  - 9/11/2021 (Implementación completa con correcciones y necesidades agregadas)

2. Diagrama de tiempos

A continuación, se realiza el Diagrama de Tiempos en el que se representan todas las fases, actividades y tareas a aplicar al proyecto y se plasman en un Diagrama de Gantt.
(Anexo 2: Diagrama de Tiempos)

CAPITULO 2

Organización para la ejecución del proyecto

En este capítulo se desarrollan todos los perfiles y roles que deben tener los integrantes del equipo para poder llevar a cabo toda la ejecución de las tareas definidas del proyecto y llegar a lograr el objetivo de este. Además, se describen las herramientas que utilizaran los integrantes para poder llevar a cabo el proyecto.

1. Equipo de trabajo

Puesto	Líder de Proyecto
Superior	-
Subordinación	Desarrollador, Analista/Diseñador, DBA, Tester.
Formación Académica	Ingeniería en Sistemas de Información. Técnico superior en desarrollo de software.
Competencias	• Conocimiento en testing • Conocimiento en lenguaje orientada a objetos (Java) • Capacidad de planificación. • Debe ser capaz de trabajar bajo presión. • Habilidades de liderazgo y comunicación. • Ingles técnico medio.
Experiencia	3 años siendo líder en otros proyectos.
Cantidad de empleados requeridos	1.

Tabla 72. Puesto: Líder de Proyecto.

Puesto	Desarrollador
Superior	Líder de Proyecto.
Subordinación	-
Formación Académica	Ingeniería en Sistemas de Información. Técnico superior en desarrollo de software.
Competencias	• Conocimientos en diagramación lógica, algoritmos y estructura de datos. • Conocimiento sobre programación orientada a objetos (Java) y estructurada. • Uso de herramientas para el desarrollo de aplicación web. • Actitud crítica, de perfeccionamiento y actualización permanente. • Ingles técnico medio.
Experiencia	1 año de programación de sistemas informáticos.
Cantidad de empleados requeridos	5.

Tabla 73. Puesto: Desarrollador.

Puesto	Analista/Diseñador

Superior	Líder de Proyecto.
Subordinación	Desarrollador, Analista/Diseñador, DBA, Tester
Formación Académica	Ingeniería en Sistemas de Información. Técnico superior en desarrollo.
Competencias	• Conocimientos en Base de Datos. • Conocimiento de lenguajes de programación y patrones de diseño. • Uso de herramientas para diseño y análisis. • Conocimiento de técnicas de modelado. • Ingles técnico medio.
Experiencia	1 año analizando y diseñando aplicaciones y proyectos.
Cantidad de empleados requeridos	5.

Tabla 74. Puesto: Analista/Diseñador.

Puesto	DBA
Superior	Líder de Proyecto.
Subordinación	-
Formación Académica	Ingeniería en Sistemas de Información. Técnico superior en desarrollo.
Competencias	• Conocimiento de instalación y configuración. • Conocimiento de Tuning de parámetros. • Ajustes de consultas SQL. • Conocimientos en programación PL/SQL. • Nivel medio de inglés.
Experiencia	1 año diseñando.
Cantidad de empleados requeridos	1.

Tabla 75. Puesto: DBA.

Puesto	Tester
Superior	-
Subordinación	Desarrollador, Analista/Diseñador, DBA, Tester.
Formación Académica	Ingeniería en Sistemas de Información. Técnico superior en desarrollo.
Competencias	• Capacidad de abstracción y modelado. • Facilidad de comunicación oral y escrita. • Creatividad. • Aptitudes para trabajo en equipo. • Diseñar plan de testing. • Conocimientos de ingeniería de Software. • Conocimientos de metodologías y modelos de calidad para la industria del software. • Poder hacer pruebas unitarias, funcionales y de estrés de aplicaciones. • Conocimientos de lenguajes de consulta de Base de Datos.

	• Ingles técnico medio.
Experiencia	1 año de experiencia en el rubro
Cantidad de empleados requeridos	3

Tabla 76. Puesto: Tester.

2. Funciones principales de los miembros del equipo

2.1. Líder de proyecto

Coordinar y gestionar tareas relacionadas con el proceso de desarrollo de software, daily meetings, Sprints, puesta a punto y mantenimiento de software. Administrar el product backlog. Supervisar las tareas. Evaluar el avance del proyecto y asegurar que cuenten con los estándares estipulados por la cátedra, establecer políticas y metodologías de testing.

2.2. Desarrollador frontend

Codificar todos los componentes de interacción con el usuario, diseñar y maquetar la estructura de UI web. También será el responsable de la UX (experiencia de usuario) y del manejo de las mejores prácticas para consumir el backend.

Además, será encargado de codificar los componentes generados según el diagrama de dominio establecido.

2.3. Desarrollador backend

Codificar la API para el manejo del diagrama de dominio, documentar la misma y asegurarse de la escalabilidad en el tiempo.

También será el responsable de codificar la API generada en el lenguaje seleccionado.

2.4. Analista/Diseñador

Realizar el relevamiento y análisis de los requerimientos del sistema, detectando los problemas y las necesidades. Diseñar el sistema de acuerdo con ello y aprovechando al máximo las arquitecturas a utilizar. Decidir los patrones más adecuados para implementar

2.5. Tester

Diseñar el plan de testeo, casos de prueba en base a los requisitos. Realizar pruebas manuales automatizadas como test unitarios, pruebas de integración, de aceptación y de stress. Armado de ambientes de prueba y administrar los datos o lotes de prueba. Ejecutar los casos de prueba.

Realizar la documentación de las pruebas registrando incidentes en base a los defectos encontrados y su seguimiento correspondiente, reportar los resultados de las pruebas.

2.6. DBA

Crear y mantener la base de datos estable y en línea, realizar tareas relacionadas al Back up, recuperación y optimización de la base. Crear y administrar la estructura de la base de datos, la actividad de los datos y el sistema manejador de base de datos. Gestionar el creciente volumen de datos y diseñar los planes apropiados para administrarlos. Establecer el diccionario de datos, asegurar la confiabilidad de la base, gestionar la seguridad de esta.

Configurar la base de datos parar poder ser alojada en la nube y administrar según los recursos disponibles. Gestionar la seguridad y permisos de la misma a los distintos usuarios e IPS.

	Líder de Proyecto	Desarrollador Backend	Desarrollador Frontend	Tester	DBA	Analista/ Diseñador
Cruz, Leandro		X	X	X		X
de Sautu Riestra, Agustín	X	X	X			X
Manfredi, Gustavo		X	X	X		X
Perez, Angel Santiago		X		X	X	X
Vasquez, Alesis		X	X	X		X

Tabla 77. Asignación de funciones a desempeñar por los integrantes.

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

A lo largo de toda la duración del proyecto se utilizan diversas herramientas digitales, tanto para comunicación, como para control, planificación, y para desarrollo.

3.1. Método de Comunicación asíncrono

Para la comunicación permanente que se enfocan en la coordinación rápida de tareas, consultas cortas o de breve respuesta y de recordatorios de avisos urgentes se utilizan 2 grupos creados por los integrantes del equipo. Se utilizan 2 aplicaciones para tener una vía de comunicación auxiliar por si no funciona el servicio principal.

3.1.1. Grupo de WhatsApp (Aplicación de comunicación)

WhatsApp Messenger (o simplemente WhatsApp) es una aplicación de mensajería instantánea para teléfonos inteligentes, en la que se envían y reciben mensajes mediante Internet, así como imágenes, vídeos, audios, grabaciones de audio (notas de voz).

Figura 138. Logo oficial de WhatsApp.

3.1.2. Grupo de Telegram (Aplicación de comunicación)

Telegram es una plataforma de mensajería y VOIP, la aplicación está enfocada en la mensajería instantánea, el envío de varios archivos y la comunicación en masa.

Figura 139. Logo oficial de Telegram.

3.2. Método de Comunicación síncrona

Para la comunicación instantánea en la que previamente se ha planeado una reunión se utiliza:

3.2.1. Servidor privado de Discord

Discord es un servicio de mensajería instantánea freeware de chat de voz VoIP, video y chat por texto; este funciona a través de servidores; este está separado en canales ya sea de texto o de voz.

Figura 140. Logo oficial de Discord.

3.3. Control de avances del proyecto

3.3.1. Repositorio remoto privado de Enterprise Architect

La solución perfecta para todo proyecto, para visualizar, analizar, modelar, probar y mantener todos sus sistemas, software, procesos y arquitecturas. Enterprise Architect es la plataforma ideal para ayudar a mantener el control de su espacio de trabajo, apoyo colaborativo de equipo, y genera confianza en el equipo para los proyectos más complejos.

Figura 141. Logo oficial de Enterprise Architect.

3.4. Método de Documentación

El método que se utiliza para la documentación, y versionado, del presente informe de Tesis es:

3.4.1. Carpeta privada de OneDrive con Word

Para el control de Versiones de Documentación:
Microsoft OneDrive es un servicio de alojamiento de archivos. Es accesible por su página web desde ordenadores y dispone de aplicaciones para Sistemas Operativos Microsoft Windows. Y para la edición de la documentación:
Microsoft Word es un programa informático orientado al procesamiento de textos. Fue creado por la empresa Microsoft, y viene integrado de manera predeterminada en el paquete ofimático denominado Microsoft Office.

Figura 142. (Izquierda) Logo oficial de OneDrive. (Derecha) Logo oficial de Microsoft Word.

4. Gestión de Configuración del Software

Los métodos de gestión de versionado que se utiliza a lo largo de todo el desarrollo del proyecto son:

4.1. Repositorio privado de GitHub con software local de Git.

El Software Git es un software de control de versiones. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

GitHub es una plataforma de desarrollo colaborativa para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails.

Como estrategia de manejo de branches vamos a utilizar git-flow usando las ramas de función: Master, Develop, Feature y HotFix.

Figura 143. (Izquierda) Logo oficial de Git. (Derecha) Logo oficial de GitHub.

CAPITULO 3

Factibilidad

En este capítulo se desarrolla el estudio de factibilidad del proyecto, en el cual se hace un análisis de los riesgos en el desarrollo del sistema, los costos de los recursos para el desarrollo del mismo y finalmente el análisis del impacto ambiental que producirá el sistema.

1. Definición y descripción de recursos para cada una de las actividades

1.1. Recursos Humanos

- 1 líder de proyecto.
- Desarrollador Backend 1.
- Desarrollador Backend 2.
- Desarrollador Backend 3.
- Desarrollador Backend 4.
- Desarrollador Backend 5.
- Desarrollador Frontend 1.
- Desarrollador Frontend 2.
- Desarrollador Frontend 3.
- Desarrollador Frontend 4.
- Tester 1.
- Tester 2.
- Tester 3.
- Tester 4.
- DBA 1.
- Analista 1.
- Analista 2.
- Analista 3.
- Analista 4.

Nombre de Tarea	Nombre de los Recursos
PLANIFICACIÓN	
Formación del Equipo	Líder del Proyecto
Presentación de distintas propuestas de sistemas a realizar	Líder del Proyecto
Presentación de la idea a los docentes sobre el proyecto	Líder del Proyecto
Definición de Herramientas internas de comunicación.	Líder del Proyecto
Definición de herramientas de gestión de la configuración	Analista/ Diseñador
Definición de metodología a utilizar	Analista /Diseñador
Definir todas las actividades para el desarrollo del proyecto	Líder del Proyecto Analista / Diseñador
Estimación temporal de todas las actividades para el desarrollo del proyecto	Líder del Proyecto Analista / Diseñador
Realizar diagrama de tiempo	Analista / Diseñador
Diseñar los perfiles para cada puesto y Asignación entre los miembros del equipo	Líder del Proyecto
Realizar estudios de factibilidad	Analista / Diseñador
Realizar Análisis de riesgos e impacto ambiental	Analista / Diseñador
RELEVAMIENTO Y ANÁLISIS DE SISTEMAS	
Definición de Objetivos y Descripción general del proyecto	Líder de Proyecto
Definición de Marco Teórico o Investigación Preliminar	Analista / Diseñador
Relevamiento General de 5 sistemas similares	Analista / Diseñador
Relevamiento Detallado de 5 sistemas similares	Analista / Diseñador
Análisis y evaluación de problemáticas y necesidades encontradas del contexto de los sistemas	Analista / Diseñador
Redacción de los objetivos y alcances del proyecto aplicado a nuestro sistema	Analista / Diseñador
Definición y división preliminar de módulos de nuestro sistema	Analista / Diseñador
INVESTIGACIÓN Y CAPACITACIÓN	
Investigación general de tecnologías	Desarrollador Backend - Frontend
Investigación y capacitación sobre herramientas de desarrollo web	Desarrollador Backend - Frontend
Profundización de conocimientos de Bases de Datos	BDA
Investigación de DDD	Desarrollador Backend - Frontend
Profundizar conocimiento en Virtualizadores	Desarrollador Backend - Frontend

Recordar y profundizar sobre Versionadores	Desarrollador Backend - Frontend
DISEÑO DEL PROYECTO	
Definición formal de objetivos, alcances del sistema y límites del sistema	Analista / Diseñador
Definición de salidas del sistema	Analista / Diseñador
Definir los roles y las funcionalidades permitidos para cada rol	Analista / Diseñador
Diagramación de CU	Analista / Diseñador
Diagramación de Modelos de Clases	Analista / Diseñador
Realización de descripción de CU	Analista / Diseñador
Diagrama de Base de Datos	DBA
Maquetación de interfaces graficas	Analista / Diseñador
DESARROLLO	
Configuración de Gestionador de Versionado	Desarrollador Frontend – Backend
Configuración de entornos de desarrollo a utilizar	Desarrollador Frontend – Backend
Levar a cabo codificación	Desarrollador Frontend – Backend
TESTING	
Descripción de Test Unitarios o Test cases y Test de componentes	Tester
Descripción de Test Modulares	Tester
Descripción de Test de Integración	Tester
Descripción de Test de Aceptación	Tester
DOCUMENTACIÓN	
Desarrollar manual de usuario	Desarrollador Frontend - Backend
Realizar plan de capacitación	Líder de Proyecto
IMPLEMENTACIÓN	
Ejecución de Test Modulares	Tester
Ejecución de Test de Integración	Tester
Ejecución de Test de aceptación	Tester
Instalación y configuración de Backend y Frontend	Desarrollador Frontend – Backend
Instalación y configuración de Bases de Datos	DBA

Tabla 78. Asignación de tareas a recursos humanos disponibles.

2. Diagrama de recursos

A continuación, se indica los recursos y costos asociados:

Nombre del 0'Recurso	Capacidad Máxima/Persona	Tasa estándar Precio /Hora
Líder del Proyecto	45%	900 $/hora
Analista / Diseñador	20%	800 $/hora

Desarrollador Frontend	25%	800 $/hora
Desarrollador Backend	20%	850 $/hora
Tester	25%	700$/hora
DBA	45%	700$/hora

Tabla 79. Asignación de costos a recursos humanos.

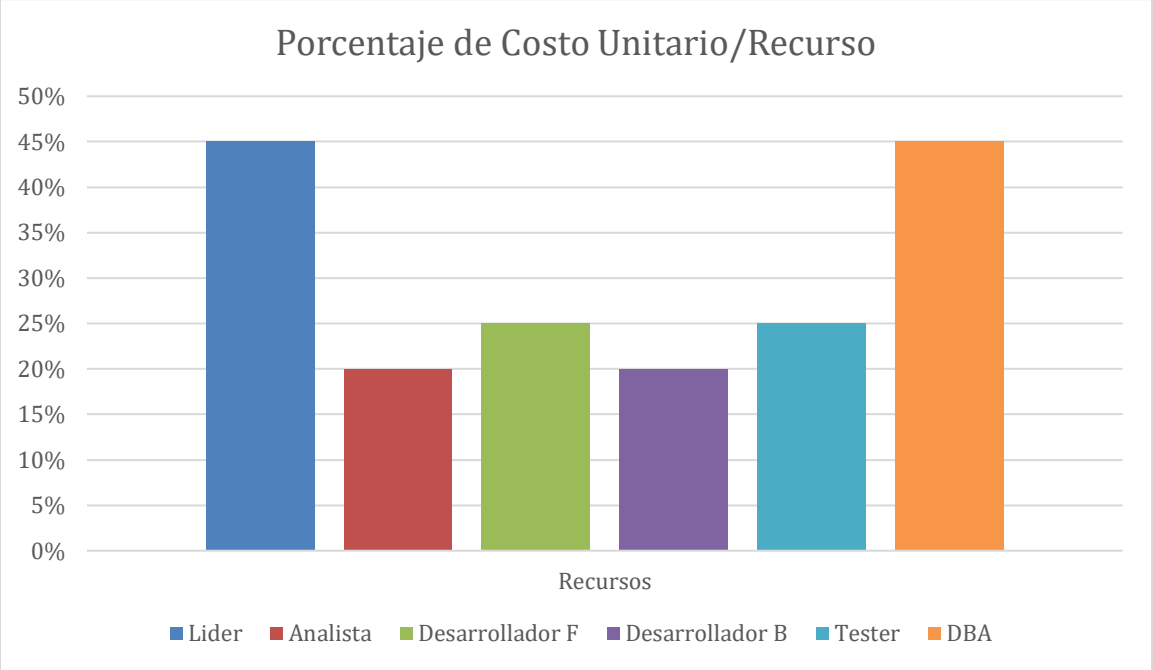


Figura 144. Porcentaje de Costo Unitario por Recurso.

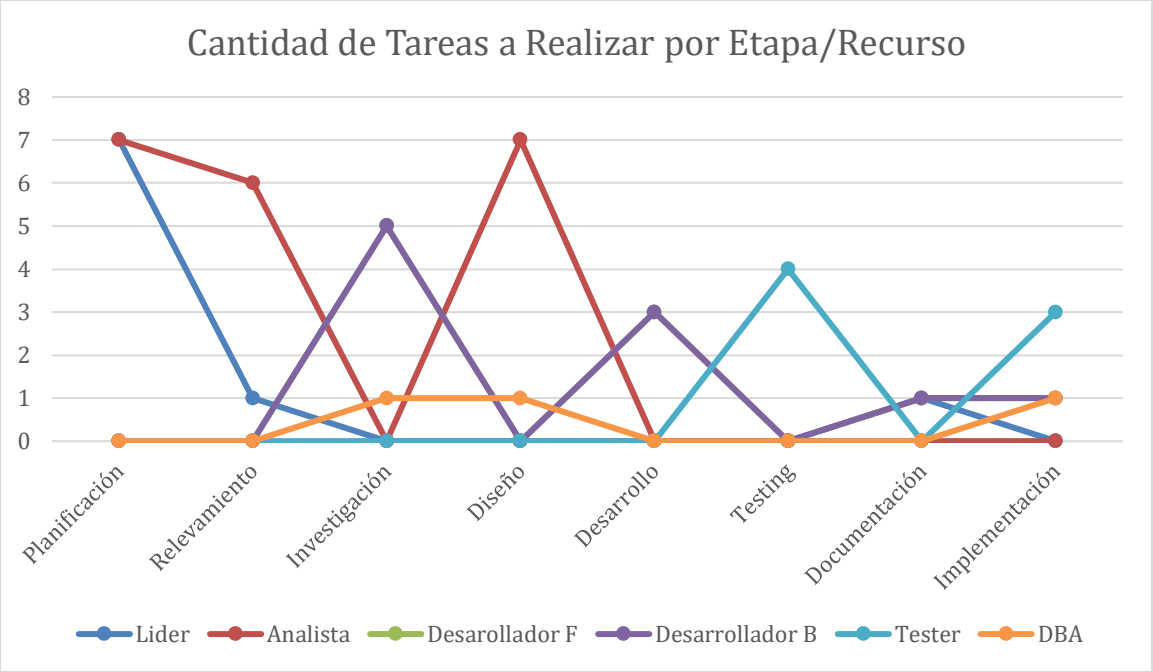


Figura 145. Cantidad de tareas a realizar por Etapa/Recurso.

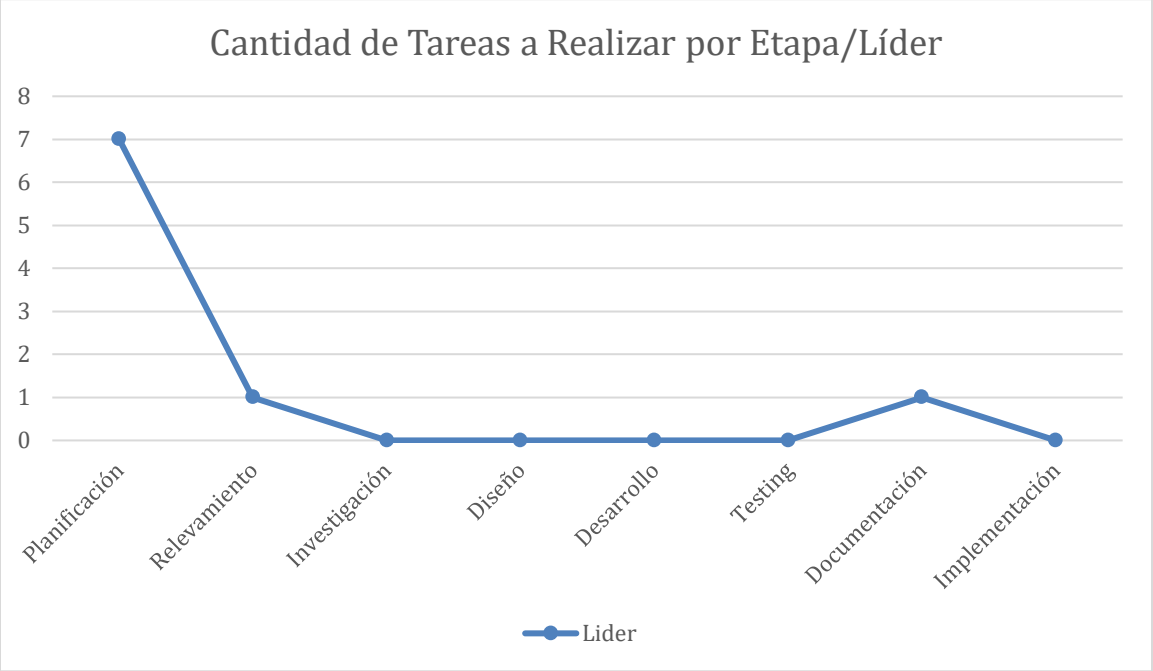


Figura 146. Cantidad de tareas a realizar por Etapa/Líder.

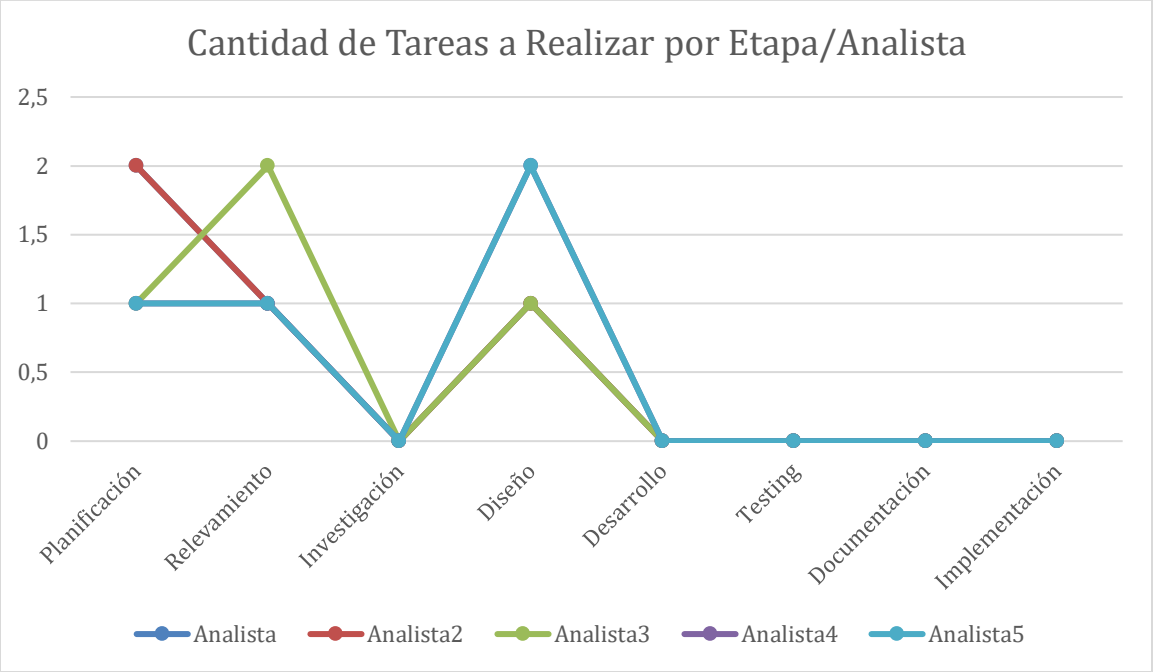


Figura 147. Cantidad de tareas a realizar por Etapa/Analista.

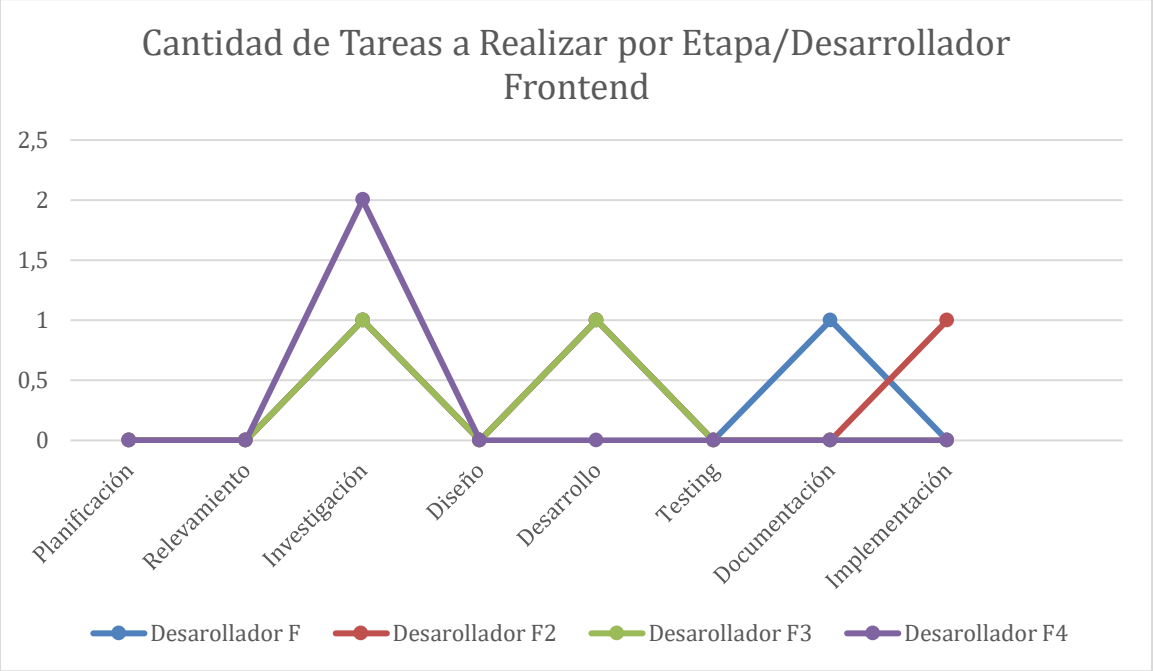


Figura 148. Cantidad de tareas a realizar por Etapa/Desarrollador Frontend.

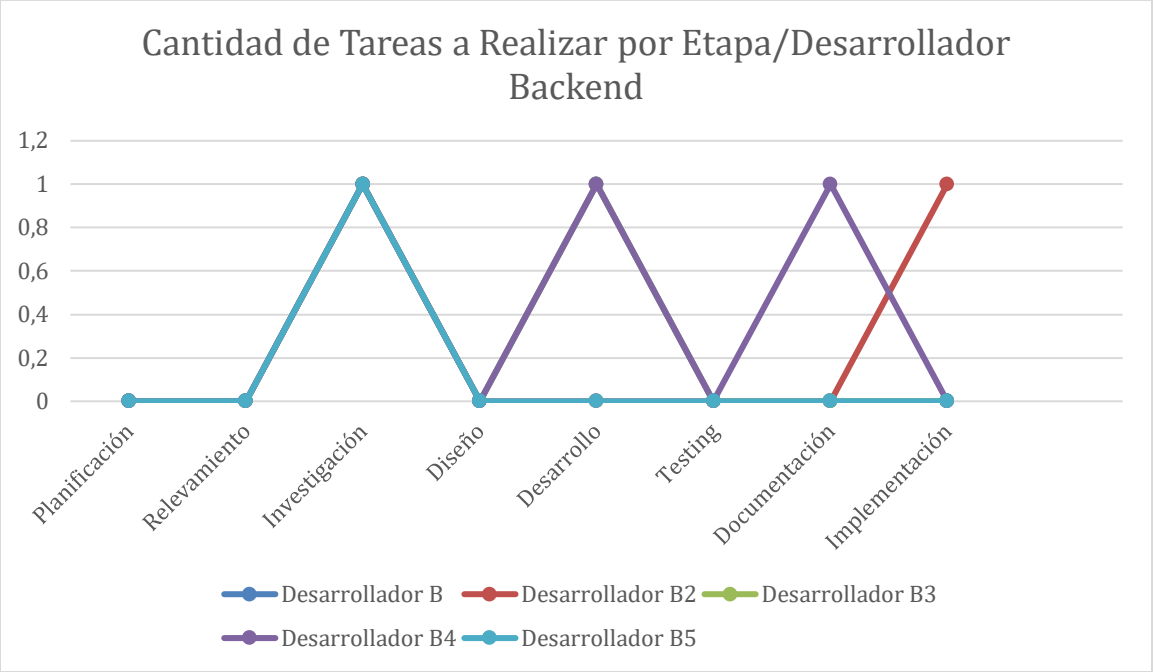


Figura 149. Cantidad de tareas a realizar por Etapa/Desarrollador Backend.

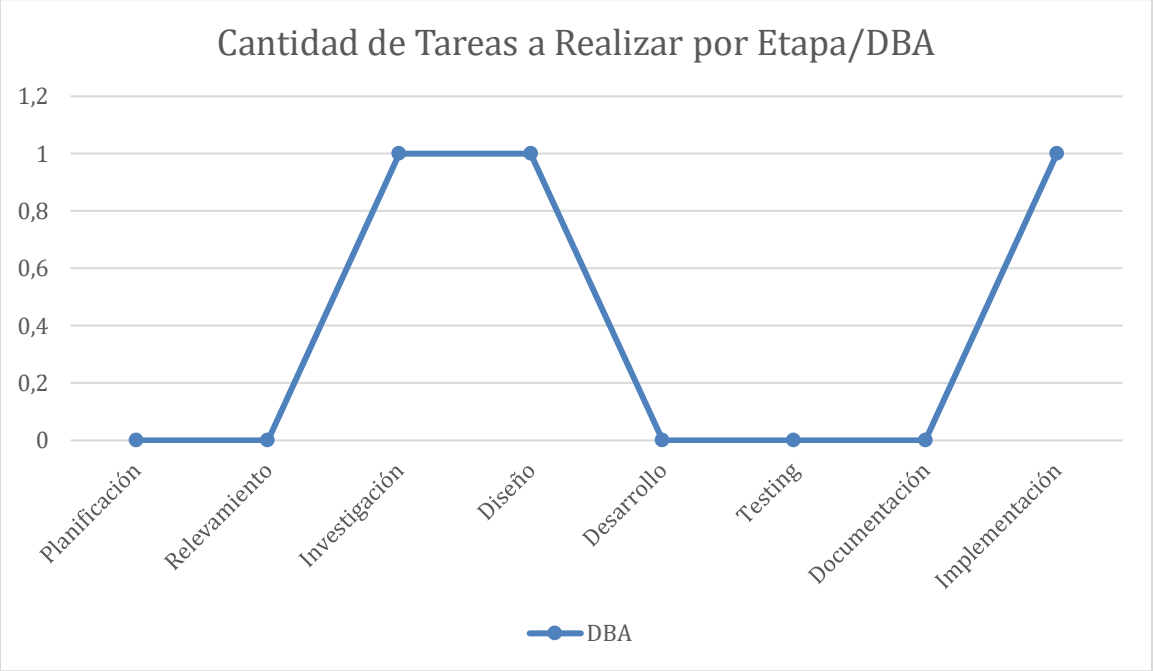


Figura 150. Cantidad de tareas a realizar por Etapa/DBA.

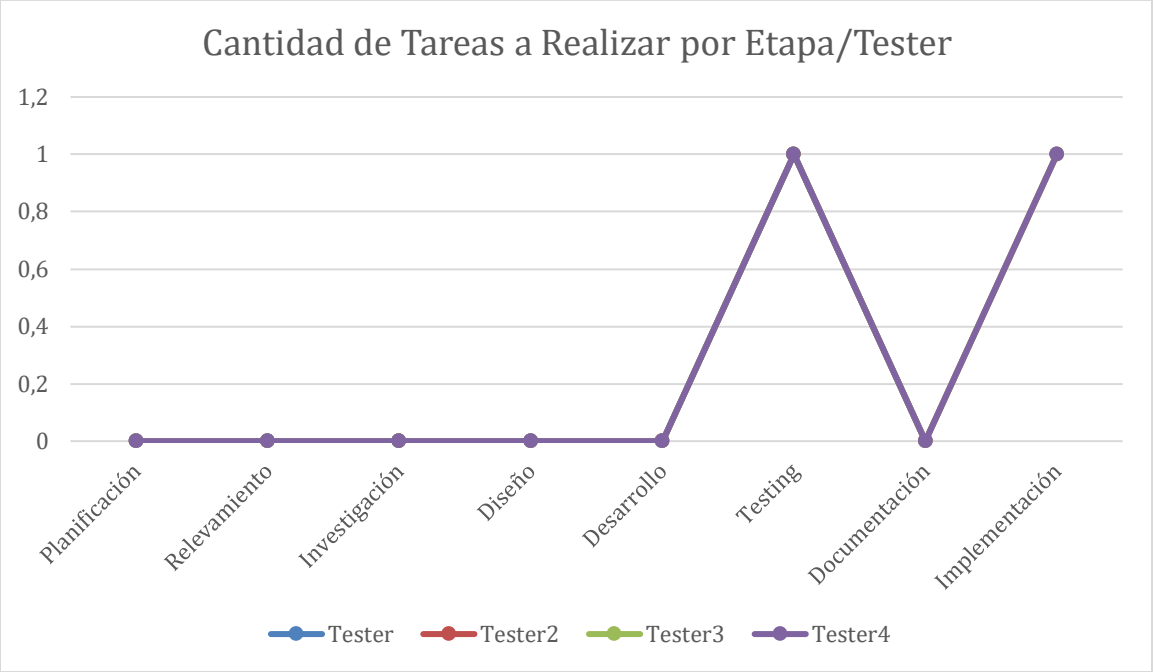


Figura 151. Cantidad de tareas a realizar por Etapa/Tester.

2.1. Conclusión

Luego del análisis de distribución de tiempos de cada uno de los Recursos, se puede observar que ningún rol de Recurso pasa las 2 tareas asignadas en cada etapa, a excepción del Líder del proyecto en la etapa de planificación, de la cual él es el principal responsable. Por lo que se puede concluir que, en general, no hay sobrecarga de tareas en los recursos, por cada etapa.

3. Análisis de factibilidad

Es un análisis que se hace para determinar si es viable la realización de una solución en un contexto determinado, se realiza teniendo en cuenta cuatro enfoques:

- Operativo
- Legal
- Técnica
- Económica

3.1. Factibilidad Operativa

La Factibilidad Operativa de sistemas, tiene como objetivo comprobar que el equipo será capaz de implementar el Sistema.
Cuenta con el personal capacitado para hacerlo o tiene los recursos humanos necesarios para mantener el Sistema.
Para esto, el sistema debe contemplar cinco puntos importantes al momento de desarrollarse.

1. Recursos de hardware

2. Recursos de Infraestructura
3. Recursos humanos
4. Conocimiento sobre las tecnologías y Capacitaciones
5. Mantenimiento del software
6. Mantenimiento del hardware

Aspecto	Requiere	Aspecto Requiere Análisis de factibilidad
Recursos de hardware	5 notebooks 1 servidor 5 celulares	Solo se necesita un único servidor, el elegido, para mantener toda la aplicación funcionando en forma óptima. Los celulares a usar en el momento de realizar las pruebas son los propios de los integrantes del equipo.
Recursos de infraestructura	Servicio de internet	La conexión a internet es uno de los puntos más importantes debido a que el equipo la necesita para realizar el versionado del código, implementación de pruebas y desarrollo de algunos módulos del sistema.
Recursos humanos	1 líder de proyecto 5 testers 5 analistas de sistemas 5 diseñadores 4 desarrolladores Frontend 5 desarrolladores Backend 1 DBA	Los roles descriptos están previstos para ser cumplidos por un total de cinco personas, los integrantes del equipo, que se encargan de cumplir cada rol acorde a la etapa del proyecto en la que se encuentren. En la etapa de definición de requerimientos emplean el rol de analistas de sistema descartando las otras. En la etapa de diseño se utiliza el rol de diseñador, testers y desarrolladores Front end Por último, en la etapa de desarrollo e implementación y pruebas se aplican los roles de desarrolladores tanto Frontend

		como Back-end y el DBA. Además, el rol de líder es transversal a todas las etapas.
Conocimiento sobre las tecnologías y capacitaciones	ECMA 6 React Css3 Html5 Sql Server Java Material-UI	El nivel de conocimiento de los miembros del equipo en general es intermedio. Varios de los integrantes tienen experiencia en alguna de las tecnologías nombradas con lo que pueden ayudar a sus pares. Las tecnologías en las que ninguno de los integrantes tiene experiencia se van a estudiar mediante un plan de capacitaciones previamente descripto.
Mantenimiento del software	Hardware	Desde el lado del Sistema, los equipos para el desarrollo tienen asociado un plan de mantenimiento anual realizado por los miembros del equipo
Mantenimiento del hardware	Software	El mantenimiento y actualización de servidores y dispositivos lo realizan los integrantes del equipo acorde a las necesidades de los mismos. En cambio, la aplicación se mantiene mediante un proceso de integración continua

Tabla 80. Detalle de aspectos operativos.

3.1.1. Conclusión

Se concluye, analizando lo anterior, que es factible realizar el sistema operativamente dado que no son necesarios equipos muy sofisticados o conocimientos muy avanzados, y todo lo que no se sepa se capacitará dentro del equipo de desarrollo.

3.2. Factibilidad Legal

El análisis de esta factibilidad nos permite corroborar si el proyecto propuesto cumple con el marco legal en el cual se encuentra inmerso. Para ello se debe realizar un análisis de los siguientes factores:

- Privacidad de la información
- Propiedad intelectual
- Propiedad de software
- Licenciamiento de software

- Y las Leyes que regulen a estos factores

3.2.1. Privacidad de la información

Un aspecto muy importante a tener en cuenta es sobre la información es que se encontrará guardada en las bases de datos del sistema.

Los datos personales de cada usuario son de carácter sensible, por lo que la difusión de estos no está permitida, esta expresado en nuestra Constitución Nacional en el art. 43 párrafo tercero "Protección de datos personales" y en la ley 25.326.

La información que se puede compartir es la correspondiente a la bitácora que el mismo desee y en la red social elegida.

3.2.2. Propiedad intelectual

La propiedad intelectual se refiere a un bien económico que incluye productos intangibles, al igual que productos físicos, reconocido en la mayor parte de legislaciones de los países y sujeto a explotación económica por parte de los poseedores legales de dicha propiedad.

Según esta definición es posible definir que todo medio de información, sean imágenes, iconos, videos, textos, etc.; deben ser propiedad del sistema o estar libres de derechos de autor para evitar infringir la Ley de Propiedad Intelectual. En la Argentina, hay tres maneras de registrar un software en Propiedad Intelectual:

- Obras inéditas: Aquellas obras que los autores o titulares solamente utilizan en forma personal o dentro de una empresa.

- Obras publicadas: Obras que se venden, regalan, donan, distribuyen gratuitamente, etc. Este trámite contempla la inscripción de obras de software puestas en conocimiento del público.

- Contratos de Software (licencias de uso, cesión de derechos y otros).

Ley 11.723 - Régimen legal de la propiedad intelectual

3.2.3. Propiedad del software

Al igual que con los medios de información, es necesario poseer los permisos necesarios para el uso del software a utilizar.

El sistema a desarrollar será propiedad de la organización, por lo que los derechos de propiedad y de distribución le pertenecen a la organización que ha solicitado y financia el proyecto. En el caso de que alguna persona realice modificaciones o distribuciones no autorizadas, recibirá una pena dictada por la ley.

3.2.4. Software Licenciado

Software Licenciado es un producto de software que es comercializado con ciertas condiciones o términos de uso, establecidas por el fabricante. Estas condiciones generalmente limitan el número de equipos en los que el producto puede ser instalado y son conocidas como “licencias”, juegan un papel variable sobre el precio final del producto, pues entre más equipos pueda instalar, más costoso resultará el producto.

Esto significa que mientras la organización mantenga la licencia, el software podrá ser utilizado libremente.

3.2.5. Conclusión

Tras analizar los puntos anteriormente detallados es posible concluir que el Sistema no viola ninguna disposición legal y por ende el análisis de la factibilidad legal es positivo.

3.3. Factibilidad Técnica

Con este análisis se pretende determinar si el equipo de trabajo cuenta con las herramientas y la tecnología necesaria para llevar a cabo el desarrollo del proyecto y su puesta en funcionamiento en tiempo y forma.
Al tratarse de una plataforma web, se requiere de un servicio de hosting y almacenamiento que brinda la capacidad de recibir grandes cantidades de peticiones, ya que son muchas las instituciones que se esperan usarán este sistema.

Actualmente, existen muchos servicios de alojamiento web dinámico. Amazon Web Services, por ejemplo, brinda soluciones de almacenamiento de base de datos, potencia de cómputos, entre otras, pensadas para adaptarse al cambio de la aplicación de manera escalable y flexible. Por lo que esta opción es óptima para desarrollo como para producción.
Para determinar si el proyecto es factible de realizar, desde el punto de vista técnico, se analizarán los siguientes aspectos:

1. Volumen de datos.
2. Cantidad de peticiones por día al sistema.
3. Tiempo de permanencia promedio.
4. Usuarios conectados en simultáneo.
5. Tipos de datos.
6. Herramienta de desarrollo.
7. Integración con sistemas externos.
8. Backup y recuperación.

Aspecto	Requiere	Aspecto Requiere Análisis de factibilidad Técnica
Volumen de datos	Base de datos SQL (hosteada en AWS) Repositorio de versionado de archivos.	Los volúmenes de datos que requiere nuestro sistema son los mínimos soportados por las bases de datos convencionales.

		Además, los datos generados por el sistema al exportar los modelos a código son todos exportados a archivos en un servicio repositorio de controlador de versiones.
Cantidad de peticiones por día al sistema	Se estiman cerca de 1000 a 5000 peticiones diarias por usuario.	Todas las peticiones que puede realizar un usuario al utilizar al sistema deben ser soportadas por el servidor donde este hosteado el sistema.
Tiempo de permanencia promedio	Se estima que los usuarios pueden permanecer conectados al sistema entre 30 minutos y 8 horas.	La permanencia en el sistema no es una limitación ya que toda la arquitectura está pensada para evolucionar con peticiones del usuario.
Usuarios conectados en simultaneo	Se estima que puede haber 10 usuarios conectados al mismo tiempo por equipo, por organización que requiera los servicios del sistema.	La cantidad de usuarios conectados realizando peticiones puede variar luego de implantado el sistema, pero los hosts en servicios de la nube se pueden balancear constantemente, escalando según las necesidades.
Tipos de datos	Los datos que utilizara el sistema son utilizados primeramente en texto plano, que serán asignados a archivos. Los cuales al ser salida irán destinados al repositorio externo.	No es necesario análisis para los tipos de datos.
Herramientas de desarrollo	Se debe contar con conocimiento de distintos lenguajes de programación y el uso de frameworks y herramientas de desarrollo web.	El equipo de desarrollo cuenta con los fundamentos necesarios para llevar a cabo el proyecto y capacitarse en las herramientas necesarias. Por lo que es factible
Integración con sistemas externos	Se requiere acceso al sistema de API de github para tener el acceso al sistema de usuarios de este servicio y para acceder a su sistema de repositorio y versionado.	La conexión a la interface de este sistema es publica, así como su documentación, asique es factible.
Backup y recuperación.	Se debe soportar el backup de datos diariamente y del sistema en sí.	Gracias a la plataforma que se utiliza en la nube se tiene un sistema de backup automatizado.

Tabla 81. Detalle de aspectos técnicos.

Herramientas	Ventajas	Desventajas
Base de datos SQL (hosteada en AWS)	• Flexibilidad permite seleccionar plataformas de desarrollo, Bases datos, etc. • Ofrece precios bajos por sus servicios. • Gran velocidad de acceso y agilidad en plataformas y bases de datos. • Estabilidad y elasticidad debido al respaldo de su infraestructura. • Seguridad para sus sistemas y bases de datos cuenta con certificaciones y acreditaciones.	• Algunos precios de los servicios pueden estar en moneda extranjera.
Framework React.	• Replicación de componentes • Variedad de componentes • Uso del Dom con actualizaciones rápidas. • Comunidad prospera • Componentes aislados lo que implica ahorro de tiempo.	• Alta velocidad de desarrollo lo que implica que los desarrolladores deben adaptarse a los nuevos cambios. • Documentación deficiente. • Muchos desarrolladore encuentran el uso de Js complejo
Framework Spring Boot.	• Código abierto • Funciones avanzadas al ser una extensión de Spring • Marco protegido, para la mayoría de ataques cibernéticos • Tiempo desarrollo corto	• Mayor tiempo de ejecución del código de software. • Problemas con la JVM
GitHub	• Servicio gratuito, aunque también tiene servicios de pago.	• No es absolutamente abierto. • Tiene limitaciones de espacio, ya que no

	• Búsqueda muy rápida en la estructura de los repositorios. • Amplia comunidad y fácil encontrar ayuda. • Ofrece prácticas herramientas de cooperación y buena integración con Git. • Fácil integrar con otros servicios de terceros. • Trabaja también con TFS, HG y SVN	puedes exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis.

Tabla 82. Detalle de herramientas.

3.3.1. Conclusión

Tras analizar los puntos anteriormente detallados, para el equipo de desarrollo (además de que se tiene acceso al Tier free de diferentes servicios de la nube) y teniendo los conocimientos necesarios, es posible concluir que el Sistema es factible Técnicamente.

3.4. Factibilidad Económica

En esta sección se explica en que consiste la factibilidad económica para el sistema que se va a desarrollar, METACOD.

Para que el proyecto resulte económicamente viable se busca justificar el costo de desarrollar el sistema estimando utilidades obtenidas luego de su puesta en producción.
Para poder conocer estos costos se realiza un análisis de los costos totales del proyecto:

Recurso	Costo
Recursos Humanos	$ 4.326.000
Puestos o equipamiento de Trabajo	$ 405.000
Servicios en la nube	$ 0
Capacitación	$ 7.602
Servicio de Internet	$ 90.000
Marketing y Publicidad	$ 2.007
Costo Total	$ 4.823.007

Tabla 83. Detalle de costos.

Aclaración, los costos de servicios en la nube, en el desarrollo, son de $ 0 ya que se utilizan los Tier free antes descriptos y se limita los costos a cambio de restricciones de rendimiento en el desarrollo.

3.4.1. Retorno de la Inversión

Retorno: Se calcula que con la puesta en producción del sistema se podrá realizar una venta a distintas organizaciones. Se calcula que se realizan ventas a 5 empresas (en los primeros 6 meses con crecimiento de 2 empresas/3 meses) a razón de $ 10.000/mes.

En base a el costo total del proyecto ($ $ 4.823.007) y los beneficios previstos se puede inferir que el retorno de la inversión se dará en 1.5 Años debido a que primero hay que lograr posicionar el sistema en el mercado, en base a ello comenzar a obtener un mayor ingreso por parte de publicidad y obtener un mínimo de 5 empresas (en los primeros 6 meses y 11 luego de los primeros 18).

3.4.2. Conclusión

Según lo analizado anteriormente se puede concluir que el proyecto es factible económicamente en el corto y mediano plazo, luego de estimar riesgos.

4. Costos desagregados por recursos con periodicidad mensual

Costo mensual	2021									Total
	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	
Total del mes	771223	366223	366223	366223	568223	568223	624223	624223	568223	4823007
Recursos Humanos	356000	356000	356000	356000	558000	558000	614000	614000	558000	4326000
Líder del Proyecto	36000	36000	36000	36000	36000	36000	36000	36000	36000	324000
Analista / Diseñador	64000	64000	64000	64000	16000	16000	16000	16000	16000	336000
Desarrollador Frontend	0	0	0	0	32000	32000	32000	32000	32000	160000
Desarrollador Backend	0	0	0	0	34000	34000	34000	34000	34000	170000
Tester	0	0	0	0	28000	28000	28000	28000	28000	140000
DBA	0	0	0	0	0	0	56000	56000	0	112000
Puestos o equipamiento de Trabajo	405000	0	0	0	0	0	0	0	0	405000
Notebook 1	71000	0	0	0	0	0	0	0	0	71000
Notebook 2	71000	0	0	0	0	0	0	0	0	71000
Notebook 3	71000	0	0	0	0	0	0	0	0	71000
Notebook 4	71000	0	0	0	0	0	0	0	0	71000
Notebook 5	71000	0	0	0	0	0	0	0	0	71000
Celular 1	10000	0	0	0	0	0	0	0	0	10000
Celular 2	10000	0	0	0	0	0	0	0	0	10000
Celular 3	10000	0	0	0	0	0	0	0	0	10000
Celular 4	10000	0	0	0	0	0	0	0	0	10000
Celular 5	10000	0	0	0	0	0	0	0	0	10000
Servicios en la nube	0	0	0	0	0	0	0	0	0	0
Servidor EC2	0	0	0	0	0	0	0	0	0	0
Base de datos	0	0	0	0	0	0	0	0	0	0
Servicios de Red	0	0	0	0	0	0	0	0	0	0
Capacitación	0	0	0	7602	0	0	0	0	0	7602
Curso 1	0	0	0	2534	0	0	0	0	0	2534
Curso 2	0	0	0	2534	0	0	0	0	0	2534
Curso 3	0	0	0	2534	0	0	0	0	0	2534
Servicio de Internet	10000	10000	10000	10000	10000	10000	10000	10000	10000	90000
Servicio 1	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 2	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 3	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 4	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 5	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Marketing y Publicidad	223	223	223	223	223	223	223	223	223	2007

Tabla 84. Costos desagregados.

(Anexo 3: Costos Desagregados)

5. Análisis de riesgos

En esta sección se analizarán los posibles riesgos a los que puede estar expuesto este proyecto, estos son:

- Fallas en la programación que puedan ocurrir en el software y herramientas de desarrollo.
- Incumplimiento en el rendimiento general de la aplicación en cuanto a tiempo de respuesta, usabilidad y funcionalidades.
- Ausencia de un registro apropiado a los eventos del sistema que permita la trazabilidad de las actividades que se desarrollaron.
- Falta de protocolo para el Establecimiento de acciones para los casos de excepción y su manejo adecuado.
- Los entregables no cumplen con lo solicitado y no son satisfactorios con las expectativas del cliente.
- El desarrollo de pruebas de aceptación para la recepción del no es el adecuado.
- Los canales de comunicación que se establecieron no cumplen con las necesidades del proyecto.
- Plazos de revisión y control del sistema son breves e irreales.

- Plazos establecidos para las pruebas del sistema son breves e irreales.
- Plazos definidos para el desarrollo del sistema son breves e irreales.
- Los patrocinadores del proyecto no tengan una clara dimensión de los alcances y recursos que requiere el proyecto.
- La infraestructura física, equipos, son insuficientes y/o inadecuadas para atender las necesidades.
- Fallas eléctricas que puedan ocurrir y daños en los equipos informáticos debido a los mismos
- Falta de personal con las capacidades necesarias.
- Planes de capacitación no satisfactorios para los usuarios del sistema.
- Falta de coordinación para producir copias de seguridad de los archivos existentes en forma periódica.
- Accesos indebidos de terceros en los perfiles de los derechos de acceso del usuario.
- Perdidas de información institucional, por intromisión de entes externos y/o inescrupulosos, los cuales pueden generar daño y perdida de información institucional.
- Dificultad para acceder a los servicios utilizados.
- No se cuenta con el personal adecuado para la administración de la base de datos.
- Ausencia de controles en los mecanismos para la administración de la información.

Para el análisis de riesgos también se tiene en cuenta el Impacto de la Amenaza y la probabilidad de ocurrencia.

Se utilizan 2 escalas del 1 al 3 para clasificar el nivel de estos criterios.

Probabilidad de ocurrencia	Impacto
3 Alta probabilidad	3 Mucho impacto
2 Probabilidad media	2 Impacto medio
1 Baja probabilidad	1 Bajo impacto

Tabla 85. Escala de clasificación de riesgos.

Posteriormente se analizaron los niveles de riesgo generando una escala, y qué significaría cada nivel. El riesgo se calcula como R=PO*I.

Nivel de Riesgo	Descripción
Alto 7-9	Impiden el correcto desarrollo o progreso del proyecto.
Medio 4-6	Deteriora o retrasa las actividades del desarrollo o progreso del Proyecto.
Bajo 1-3	Fallas menores o con muy baja repercusión en el desarrollo del proyecto, permiten continuar con el mismo.

Tabla 86. Escala de niveles de riesgo.

Riesgo	Probabilidad de ocurrencia	Impacto	Nivel de Riesgo
Fallas en la programación que puedan ocurrir en el software y herramientas de desarrollo.	3	3	9
Incumplimiento en el rendimiento general de la aplicación en cuanto a tiempo de respuesta, usabilidad y funcionalidades.	2	1	2

Ausencia de un registro apropiado a los eventos del sistema que permita la trazabilidad de las actividades que se desarrollaron.	2	1	2
Falta de protocolo para el Establecimiento de acciones para los casos de excepción y su manejo adecuado	1	1	1
Los entregables no cumplen con lo solicitado y no son satisfactorios con las expectativas del cliente	1	1	1
El desarrollo de pruebas de aceptación para la recepción del no es el adecuado.	1	2	2
Los canales de comunicación que se establecieron no cumplen con las necesidades del proyecto	1	1	1
Plazos de revisión y control del sistema son breves e irreales	3	3	9
Plazos establecidos para las pruebas del sistema son breves e irreales	3	1	3
Plazos definidos para el desarrollo del sistema son breves e irreales	3	3	9
Los patrocinadores del proyecto no tengan una clara dimensión de los alcances y recursos que requiere el proyecto	1	2	2
La infraestructura física, equipos, son insuficientes y/o inadecuadas para atender las necesidades	1	2	2
Fallas eléctricas que puedan ocurrir y daños en los equipos informáticos debido a los mismos	1	1	1
Falta de personal con las capacidades necesarias	3	3	9
Planes de capacitación no satisfactorios para los usuarios del sistema	2	1	2
Falta de coordinación para producir copias de seguridad de los archivos existentes en forma periódica	2	1	2
Accesos indebidos de terceros en los perfiles de los derechos de acceso del usuario	3	2	6
Perdidas de información institucional, por intromisión de entes externos y/o inescrupulosos, los cuales pueden generar daño y perdida de información institucional	3	2	6
Dificultad para acceder a los servicios utilizados	2	2	4
No se cuenta con el personal adecuado para la administración de la base de datos	2	2	4
Ausencia de controles en los mecanismos para la administración de la información	3	2	6

Tabla 87. Evaluación de riesgos.

Luego, una vez analizados todos los riesgos del proyecto y los niveles de impacto, se decide a que nivel prestar atención y analizar como mitigarlos, se presentan los riesgos más importantes para el mismo, ósea los de riesgo Alto. Los riesgos de nivel bajo o medio se deciden aceptar.

- Fallas en la programación que puedan ocurrir en el software y herramientas de desarrollo.

- Plazos de revisión y control del sistema son breves e irreales.
- Plazos definidos para el desarrollo del sistema son breves e irreales.
- Falta de personal con las capacidades necesarias.

Para estas fallas se proponen acciones preventivas para intentar evitar que sucedan y en caso que estas sucedan se proponen acciones correctivas.

Riesgo	Acción preventiva	Acción correctiva
Fallas en la programación que puedan ocurrir en el software y herramientas de desarrollo.	• Contar con al menos un miembro del equipo que tenga conocimientos acerca de testing. • Revisión constante del software. • Realizar planificaciones y ejecución de distintos tipos de pruebas. • Desarrollar el sistema de manera que sea de fácil mantenimiento.	• Establecer User Stories de deuda técnica para atacar los problemas de calidad. • Asignar los recursos más capacitados a la refactorización de los componentes/módulos más comprometidos en calidad. • Programación de a pares.
Plazos de revisión y control del sistema son breves e irreales.	• Realizar planificación más exhaustiva de las tareas. • Usar más recursos para la etapa de planificación y que no sea unipersonal. • Ajustar o reforzar el tiempo de los recursos en las tareas críticas.	
Plazos definidos para el desarrollo del sistema son breves e irreales.		
Falta de personal con las capacidades necesarias.	• Llevar a cabo una planificación para las capacitaciones en los lenguajes de programación a utilizar. • Realizar cursos de capacitación sobre las herramientas a utilizar.	• Aumentar la dedicación horaria a la capacitación. • Cursos internos en el equipo dictados por especialistas. • Contratación de recursos ya capacitados. • Programación de a pares.

Tabla 88. Mitigación de riesgos.

5.1. Conclusión

De acuerdo con el análisis de riesgo realizado, se han identificado las potenciales amenazas a la correcta ejecución del proyecto y el impacto que las mismas producirían en caso de materializarse, se establecieron medidas preventivas y correctivas para los casos más graves. De este análisis se desprende la necesidad de adicionar medidas preventivas para mitigar los riesgos de mayor impacto, los cuales son tenidos en cuenta en toda la ejecución del proyecto.

6. Análisis de impacto ambiental

Es la alteración que se produce en el ambiente cuando se lleva a cabo un proyecto o una actividad. La alteración no siempre es negativa. Puede ser favorable o desfavorable para el medio.
El impacto de los sistemas de información puede ser:

- Por el propio desarrollo, implementación y operación de los nuevos sistemas
- Por la actividad para la que se están desarrollando e implementando los nuevos sistemas

El objetivo consiste identificar, predecir y entender el impacto de una acción/componente sobre el medio ambiente. Y a partir de esa información determinar, en caso de ser necesario, las acciones preventivas o correctivas que se deban implementar.

Para la **evaluación** del **impacto ambiental** hay que tener en cuenta:

SIGNO: POSITIVO o NEGATIVO
Según si es positivo y sirve para mejorar el medio ambiente o si es negativo y degrada.

INTENSIDAD: ALTA (3), MEDIA (2), BAJA (1)
Según afecte la destrucción del ambiente sea total, alta, media o baja.

EXTENSION: TOTAL (3), PARCIAL (2), PUNTUAL (1)
Según afecte a un lugar muy concreto y se llama puntual, o a una zona algo mayor –parcial, o una gran parte del medio –impacto extremo- o a todo total.

PERSISTENCIA: PERTINAZ (3), TEMPORAL (2) o FUGAZ (1)
Según si es fugaz si dura 1 año o menos; temporal si dura entre 1 a 3 años y pertinaz si dura de 3 a 10 años y permanente si es para siempre.

RECUPERACION: IRRECUPERABLES (3), REVERSIBLE (2), MITIGABLE (2), RECUPERABLE (1)
Según sea más o menos fácil de reparar distinguimos irrecuperables, reversibles, mitigables, recuperables.

PERIODICIDAD: CONTINUO (2), DISCONTINUO (1)
Según distinguimos si el impacto es continuo como un Sistema que funciona las 24 horas los 365 días del año; o discontinuo.
Listamos las acciones/componentes que producen impactos ecológicos en el medio ambiente.

Uso de energía eléctrica
El sistema desarrollado será alojado en un servidor que estará disponible las 24 horas todos los días, esto conlleva un gasto de energía eléctrica.
También tendremos en cuentas de los distintos dispositivos a través de los cuales se hace uso del sistema.

Uso de internet
El uso de internet y forma parte de esta clasificación ya que genera contaminación y es que las tecnologías de la información son responsables de las emisiones de gases contaminantes que contribuyen al efecto invernadero. Varias empresas están incorporando el uso de energía renovable debido a la creciente contaminación.

Consumo Papel
La utilización del sistema permite reducir el consumo de papel para la generación de diagramas en el desarrollo y documentación, permitiendo una reducción tanto en el uso de papel que utiliza cada empresa que se subscribe, como de la tinta necesaria para esa gestión, ya que esta tiene una alta toxicidad y genera residuos. La fabricación de papel representa una enorme fuente de contaminación y tiene un gran impacto medioambiental, es uno de los mayores contaminantes del agua y del aire, y una de las que más gases efecto invernadero emite.

Componentes de Hardware
También tendremos en cuenta la gran cantidad componentes de hardware desechados que se realizan, en donde el uso irresponsable de estos desperdicios electrónicos tiene un gran impacto en el planeta y una gran contaminación.

Impacto Visual
El impacto visual de nuestra organización es casi nulo, ya que el mismo no debe ser realizado sin tener en cuenta su contexto considerando la gente que habita y las áreas circundantes.
Hemos considerado que nuestra organización pueda mezclarse en su entorno y apenas pueda ser visible ya que las actividades desarrolladas, tamaño de los proyectos y cantidad de personal para la realización del mismo no es necesario una infraestructura lo suficientemente grande para que desarrolle un impacto visual negativo, Además el mismo cuenta con superficies no refractantes que puedan llegar a disipar calor y causar malestar e incomodidad, a su entorno y a la gente.

Impacto Auditivo
El impacto auditivo debido a las actividades que desarrolladas y el tipo de organización es mínimo, la consideración es baja.

Impacto Social
Por último, consideramos el impacto social del desarrollo de nuestro proyecto, ya que
La implementación del mismo ayuda a la comunidad de desarrollador y analistas ya que el mismo tiene como objetivo el de abstraer a los analistas de sistemas del desarrollo de código y a los desarrolladores que quieren iniciar rápidamente un proyecto con buenas prácticas.
El desarrollo e implementación del proyecto beneficia en la generación de diagramas de dominio restringidos por metamodelos orientado a objetos y luego poder a partir de este obtener el código de una interfaz web construida con tecnologías modernas y API con conexión a una base de datos.

ACCION: Desarrollo / Implementación del Sistema						
	Signo	Intensidad	Extensión	Persistencia	Recuperación	Periodicidad
Energía Eléctrica	-	1	2	2	1	2
Uso de Internet	-	1	3	2	1	2
Consumo Papel	-	2	2	2	2	1
Desperdicios HW	-	3	2	3	2	1
Impacto Visual	-	1	3	3	3	3
Impacto Auditivo	-	1	3	1	1	2
Impacto Social	+	3	2	3	2	3

Tabla 89. Evaluación de impacto ambiental.

6.1. Conclusión

Los factores que hemos estudiado en esta sección afectan al ambiente tanto a nivel de Hardware, energético y social. Analizando la tabla podemos obtener la conclusión de que tanto el desarrollo e implementación (hosting) del sistema genera un impacto considerablemente viable teniendo en cuenta su contrapartida del impacto social positivo hacia la sociedad, además podemos observar que aumenta un poco más el impacto durante la etapa de Implementación por el uso de servidores y energía constantes que aun así termina siendo insignificante.

Trabajo Práctico Integrador Nº1

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.

1. **Estimar tiempo y recursos necesarios para el desarrollo de un proyecto**.
2. **Definir el presupuesto.**
3. **Liderar y gestionar los proyectos a su cargo**.
4. **Establecer un plan del proyecto**: controlando su progreso y efectuando el seguimiento de los desvíos.
5. **Administrar los grupos de trabajo a su cargo**: motivando a sus integrantes, cuidando su adecuada capacitación o coaching y resolviendo eventuales conflictos.
6. **Colaborar con el analista funcional**: en la interpretación y comprensión de las necesidades.
7. **Participar en el diseño:** sobre todo de soluciones asociadas a los requerimientos y colaborar en la definición de las arquitecturas.
8. **Hacer seguimiento de cada fase y cada hito del proyecto**: monitoreando tiempos, costes, calidad y riesgos.
9. **Asegurar la implementación de productos o servicios:** de acuerdo al alcance acordado para lograr la satisfacción del cliente.
10. **Involucrar a los usuarios o clientes cuando haga falta**: negociando compromisos con los mismos o con otros grupos o personas involucradas.
11. **Asegurar el cumplimiento de los procesos definidos por el área de calidad**.
12. **Cuidar los aspectos de un proyecto**: Como aquellos que pueden incidir en sus alcances, plazos y calidad, incluyendo riesgos y manejo de los cambios, realizando el análisis de impacto y negociando compromisos.
13. **Elaborar informes de avance del proyecto**: tomar decisiones correctivas o proponer soluciones alternativas a la gerencia.
14. **Asegurar que los proyectos sean cerrados apropiada y formalmente.**
15. **Colaborar con la unidad de negocios**: ayudando para establecer el precio de venta, si así se requiere.

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. Estimar tiempo y recursos necesarios para el desarrollo de un proyecto.
2. Definir el presupuesto y obtener el mejor rendimiento del mismo.
3. Establecer un plan del proyecto.
4. Colaborar con el analista funcional.
5. Participar en el diseño.

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).

- Administrar los grupos de trabajo a su cargo.
- Elaborar informes de avance del proyecto.
- Involucrar a los usuarios o clientes cuando haga falta.
- Asegurar el cumplimiento de los procesos definidos por el área de calidad.
- Hacer seguimiento de cada fase y cada hito del proyecto.

4. Cuáles son las 3 principales funciones que cumplirá durante la fase de post ejecución del proyecto, el "Jefe (o Director) de Proyecto" (pueden repetirse con las del punto 1).

- Asegurar la implementación de productos o servicios.
- Asegurar que los proyectos sean cerrados apropiada y formalmente.
- Colaborar con la unidad de negocios.

5. 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.

- **Riesgos de Alcance**: A lo largo del desarrollo de un proyecto, el alcance puede experimentar cambios y crecer en complejidad a medida que los clientes añaden nuevos requerimientos y esto puede ampliar el alcance o modificarlo en consecuencia esto puede provocar mayores costos económicos y atrasos en el calendario del proyecto.
- **Riesgos de la Planificación**: El proyecto no se desarrolle de la manera en la que se había planificado inicialmente no necesariamente tienen por qué ser errores propios de nuestro equipo, sino que pueden obedecer a causas externas.
- Retrasos en el suministro por parte de un proveedor externo, accidentes u otros imprevistos no controlados, pueden alterar la planificación inicial. Obviamente las consecuencias de este riesgo es un atraso en los entregables y calendarios del proyecto.
- **Riesgos de los recursos**: Los recursos de los que se dispone para realizar un proyecto también experimentan modificaciones durante el transcurso de este. Aunque inicialmente los recursos presupuestarios sean de una determinada cantidad, es posible que durante el desarrollo del proyecto se presenten cambios en la situación económica del sector al que se dedica la empresa o cambios macroeconómicos.
- Por otro lado, los recursos humanos también pueden experimentar modificaciones. El personal que comienza realizar un proyecto no necesariamente será el mismo que lo termine. Además, los nuevos miembros que se incorporen al equipo cuando el proyecto ya se encuentre comenzado, deberán emplear un tiempo en adaptarse. Esto puede llevar a consecuencias desde disminución de calidad a aumento de los costes del proyecto y productividad afectando el calendario.
- **Riesgos tecnológicos**: Utilizar software u otras utilidades informáticas inadecuadas supondrá una merma en tu productividad. Si existen problemas tecnológicos, esto retrasa o dificulta la entrega de tus proyectos. La consecuencia principal va a afectar la productividad y calidad de software.

Riesgo	Probabilidad de ocurrencia (baja / media / alta)	Impacto (baja / media / alta)	Requiere Tratamiento o es Aceptable
Malware	Alta	Alta	Requiere Tratamiento
Ransomware	Alta	Alta	Requiere Tratamiento

Acceso no autorizado o Hackeo	Media	Alto	Requiere Tratamiento
Fuga de información por Malas configuraciones de las redes internas y los accesos a los servidores	Baja	Alta	Requiere Tratamiento
Permisos mal asignados	Media	Media	Requiere Tratamiento

Tabla 90. Análisis de riesgos tecnológicos.

Riesgo	Tratamiento	Acciones
Malware	Reducido	• Instalar un buen antivirus y antimalware. • Bloquear páginas que son propensas a contenerlos.
Ransomware	Reducido	• Instalar antivirus y firewall. • Bloquear páginas que son propensas a contenerlos. • Actualizar Sistemas Operativos. • Realizar copias de seguridad en la nube.
Acceso no autorizado	Reducido	• Correcta configuración de usuarios con privilegios. • Control de acceso físico a sala de servidores.
Hackeo	Reducido	• Concientizar a los empleados sobre la importancia de cambiar las contraseñas cada 3 o 6 meses. • Crear una política que obligue a generar contraseñas robustas.
Fuga de información por Malas configuraciones de las redes internas y los accesos a los servidores	Reducido	• Realizar capacitaciones periódicas en seguridad informática, sobre tecnologías nuevas y más robustas. • Encargar la tarea de la seguridad a un experto con amplia experiencia y habilidades. • Realizar auditorías enfocadas en la seguridad para detectar y mejorar los sistemas lo antes posible.

Tabla 91. Acciones preventivas para riesgos tecnológicos.

6. Si los obligaran a incorporar al equipo del Proyecto a 2 personas, en qué momento los incorporaría, en cuál puesto y perfil y qué actividades les asignaría.

La incorporación al equipo de dos personas sería más beneficiosa hacerla durante las primeras etapas, ya que mientras más tiempo tenga de desarrollo el proyecto más tiempo llevará la adaptación al proyecto una vez comenzado, esto supondrá, al menos por un tiempo, una menor eficiencia y productividad.

Además de que a veces colocar más recursos a una tarea puede tener un efecto contrario en cuanto problemas de comunicaciones con el resto del equipo.

Generalmente y en nuestro caso lo más adecuado sería agregar dos Desarrolladores o Analistas ya que sus etapas son próximas al inicio del proyecto y las más demandantes.

7. 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.

Democrático: El mejor enfoque que podríamos elegir en cuanto a liderazgo sería el democrático, ya que el equipo a pesar de estar formado por estudiantes avanzados de la carrera se lograría una mejor solución mediante un ambiente participativo de todos los temas y situaciones logrando decisiones consensuadas.

8. 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.

- Falta de claridad en la estrategia, dirección, objetivos y liderazgo del proyecto.
  Enfoques agresivos:
  - Cuando es vital una acción rápida y decisiva.
  - Cuestiones importantes en las que es necesario llevar a cabo acciones impopulares.
  - En contra de persona que se aprovechan de un comportamiento no agresivo.

- Errores en el manejo de la comunicación y malentendidos.
  Enfoques de colaboración:
  - Para combinar intuiciones de personas con distintas perspectivas.
  - Para obtener compromisos incorporando las inquietudes en un consenso.

- Costos y presupuesto del proyecto.
  Enfoques de arreglo:
  - Para conseguir acuerdos temporales en cuestiones complejas.
  - Para llegar a soluciones rápidas cuando el tiempo apremia.

- Ego.
  Enfoques acomodativos:
  - Cuando la armonía y estabilidad son especialmente importantes.
  - Con el propósito de acumular crédito social para cuestiones posteriores.
  - Cuando uno se da cuenta que está equivocado; con el fin de estar en una mejor posición para ser escuchado.

- Conflictos personales y choques de personalidades.
  Enfoques acomodativos:
  - Cuando la armonía y estabilidad son especialmente importantes.
  - Con el propósito de acumular crédito social para cuestiones posteriores.

- Falta de confianza.
  Enfoques de colaboración:
  - Para abrir caminos entre sentimientos que han interferido en una relación.

9. 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 y explicar detalladamente.

1. **Clarificar los objetivos (Positiva):** Esto es una motivación que ayudará a los miembros del equipo que no vean un panorama claro. Si el integrante tiene claro su objetivo, pondrá todos sus esfuerzos por sacarlo adelante.
   Por ej: Si los objetivos varían de un día para otro y cambian la manera de afrontar su trabajo, las consecuencias serán altamente negativas. Consecuencia de ello, el integrante perderá la motivación en el proyecto en el que está inmerso.

2. **Libertad y responsabilidad (Positiva):** Se trata de que los integrantes del equipo tengan total libertad para definir su horario. Si cada persona se siente totalmente responsable de sus funciones y tiene conciencia de que sus compañeros dependen de ellas, lo tendrá en cuenta a la hora de escoger sus días de descanso.

3. **Buena comunicación (Positiva):** La comunicación y la escucha debe ser clara y asertiva entre los integrantes del equipo Prestar atención a la comunicación no verbal entre los integrantes y tratar de anticiparse a sus problemas y necesidades.

   Si se logra conectar emocionalmente, se reforzará la confianza mutua, su compromiso con el proyecto y su motivación para alcanzar los objetivos de la empresa.

   Por ejemplo, un integrante del equipo que no está participando activamente, quizás al hablar podemos entender si tiene algún problema y ajustar las tareas a su conveniencia.

4. **Reconocimiento (Negativa):** Todos necesitamos crédito de nuestros logros personales y que aprecien nuestras contribuciones. Reconocer un trabajo bien hecho es otra manera de dar las gracias.

   Los logros personales motivan, pero lo hacen mucho más si alguien se da cuenta y lo sabe valorar y reconocer. El reconocimiento es uno de los caminos para dar significado a la existencia de las personas.

Por ejemplo, un miembro del equipo que no está tan capacitado y no puede cumplir al mismo ritmo que los demás puede salir exhausto o estresado psicológicamente. El reconocer su esfuerzo por más mínimo que sea puede darle orgullo para continuar.

5. **Llamado de atención, corrección o reclamo (Negativo):** Si uno de los integrantes no está cumpliendo con las tareas que les corresponde, se le realizará un llamado de atención. Esto genera un cierto nivel de estrés o presión a los integrantes que permite obtener mejores resultados.

10. 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 que se utilizara para hacer la implementación del sistema nuestro con respecto a los sistemas que usan actualmente otro, será en forma de piloto primeramente como forma de prueba, luego se puede implementar en etapas combinado de forma paralela, o se puede implementar de forma directa. A continuación, se explica más detalladamente.

Al tener un sistema modularizado, y muy especializado nuestro sistema ofrece la posibilidad a las empresas de utilizar opciones de prueba, que permiten usar sus funciones limitadas, lo cual permite usar el sistema en una especia de forma piloto para probar sus funciones previamente a la conversión completa, y de esta manera evaluar el mejor método de implementación:

- **Etapas y paralelismo:** Nuevamente como nuestro sistema implementa grandes funcionalidades divididas estratégicamente por módulos, se puede realizar una conversión por etapas en la que cada etapa se define como el reemplazo íntegro de un módulo del sistema actual por la tecnología utilizada por la empresa, esta etapa puede entrar en funcionamiento reemplazando solo esa tecnología mientras el resto de la organización utiliza paralelamente el sistema o método que se utilizaba anteriormente. Esta conversión del sistema nuevo al actual puede ser incremental etapa a etapa o utilizar los módulos a conveniencia.

- **Directa:** También se puede hacer una conversión directa y reemplazando el sistema actual directamente por el nuevo, las únicas desventajas son que se perderán el avance de todos los proyectos que hayan empezado su desarrollo con el sistema antiguo o se deberá esperar a que todos los proyectos que están desarrollando se terminen para hacer una implementación pareja y completa a nivel empresa.

Trabajo Práctico Integrador Nº2

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 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.

Data Center: Unidad organizativa de Servicios que implica la combinación adecuada y eficiente de equipos, software, personas, instalaciones, sistemas, datos, documentación, tiempo y dinero.

Principales Recomendaciones

La empresa tendrá un adecuado funcionamiento en la distribución y características de los locales de trabajo, previendo condiciones de higiene y seguridad en sus construcciones e instalaciones, en las formas, en los lugares de trabajo y en el ingreso, tránsito y egreso del personal, tanto para los momentos de desarrollo normal de tareas como para las situaciones de emergencia. Con igual criterio deberán ser proyectadas las distribuciones, construcciones y montaje de los equipos industriales y las instalaciones de servicio. Los equipos, depósitos y procesos riesgosos deberán quedar aislados o adecuadamente protegidos tal y como lo indica el artículo 42 sobre Proyecto, Instalación, Ampliación, Acondicionamiento y Modificación de establecimientos de la Ley N° 19.587, aprobada por Decreto N° 351/79.

Extinción de incendios automático y manuales, sensores

Matafuegos Manuales*:* Siguiendo el Artículo 176 de la Ley La cantidad de matafuegos necesarios en los lugares de trabajo, se determinarán según las características y áreas de los mismos, importancia del riesgo, carga de fuego, clases de fuegos involucrados y distancia a recorrer para alcanzarlos., Las clases de fuegos se designarán con las letras A-B-C y D.

En todos los casos deberá instalarse como mínimo un matafuego cada 200 metros cuadrados de superficie a ser protegida. La máxima distancia a recorrer hasta el matafuego será de 20 metros para fuegos de clase A y 15 metros para fuegos de clase B.

Matafuegos Automáticos**:** Se deben cumplimentar las mismas restricciones que para los manuales.

Cerraduras eléctricas con sistema de control de acceso

Se debe acceder a ellos desde oficinas privadas, deben contar con cerraduras magnéticas prohibiendo el acceso de personal no autorizado.

Piso técnico con bloques reubicables, debajo cableado libre, sensores

- Los CPD deben tener buenos equipos de climatización, de no ser así los costos por problemas de una mala refrigeración serán mayores a los del equipo.
- Los cableados pueden ir con piso falso o canaletas para evitar interferencias.

Ups, Grupo Electrógeno con Monitoreo

- Los Sistemas de Alimentación Ininterrumpida permiten tener flujo de energía eléctrica mediante baterías, cuando el suministro eléctrico falla. De la misma manera, sirven para proteger los dispositivos que se encuentran conectados cuando hay una elevación o disminución de tensión, o sostener su funcionamiento cuando suceden pequeños cortes de energía. Y generalmente son de doble conversión (Es decir no cuentan con relé, lo que provocará fallos de microsegundos que puede llevar a pérdida del trabajo del día).

Ubicación Lógica y Ubicación Física

En cuanto a la ubicación debemos tener en cuenta de tener servicios esenciales cercanos:

- Servicios de emergencias de salud y policía.

- Agua, electricidad e internet.
- No debe dar directamente a la luz solar.
- No debe obstruir la vista en la zona de localización.
- Además, el data center debe estar ubicado de forma que no den a la calle principal donde se encuentran instaladas la sede.

Servicios de emergencias y salud centrales de detección y extinción de incendios.

- Deben estar en las cercanías en caso de siniestros y robos.

Entradas y Salidas de Emergencia

Capítulo 18. Art. 172. Los medios de escape destinados en las plantas de desarrollo cumplimentan lo siguiente:

- El trayecto a través de los mismos deberá realizarse por pasos comunes libres de obstrucciones y no estará entorpecido por locales o lugares de uso o destino diferenciado.
- Colocamos señales para indicar la salida en caso de fuerza mayor.
- Ninguna puerta, vestíbulo, corredor, pasaje, escalera u otro medio de escape, será obstruido o reducido en el ancho reglamentario.

Salas y cajas de Backup.

Deberá contar con un backup y una relación de todos los sistemas de información municipal, tanto fuentes como ejecutables y la base de datos, así mismo, software de fábrica de desarrollo, así como de gestión, lo cual permitirá la rápida identificación de la información, la relación debe contener:

- Nombre del sistema.
- Lenguaje o paquete con el que fue creado el sistema (programas que lo conforman tanto fuentes como ejecutables).
- Área que genera la información.
- Áreas que utilizan la información.
- Volumen de los archivos que trabaja el sistema.
- Volumen de transacciones que maneja el sistema, diarias, semanales, mensuales.
- Equipamiento necesario para el manejo óptimo del sistema.
- Fecha en que la información fue ingresada.
- Fecha de creación de los sistemas.
- Nivel de importancia estratégica que tiene la información del sistema para la institución.
- Actividades a realizar para restablecer el sistema de información.

Cableado eléctrico, puestas a tierra y tableros

- Cableado estructurado eléctrico y distribución: Conexiones de punto a punto, respetando todas las normas y estándares que exige la ley. La electricidad debe ir separada a los cables de red de nuestras instalaciones.
- Subestaciones eléctricas: Organización de la distribución eléctrica de manera eficiente y ordenada, permitiendo que se pueda administrar/gestionar a la manera deseada.
- Tableros eléctricos: Administración de circuitos y aumento del nivel de seguridad de la empresa de manera instantánea.
- Pozos a tierra: Aíslan la electricidad excedente hacia una zona de baja resistencia, incrementando así la seguridad de la empresa.

Cableado de datos y telefonía y centrales telefónicas

Para poder llevar a cabo una mejor comunicación entre los empleados de la empresa como así también con los clientes, y una mejor distribución de las llamadas con los profesionales adecuados a atender cada solicitud de servicio, tendremos telefonía fija.

Un plan de línea fija es útil para la empresa porque:

- Se trata de un negocio local con sedes y los empleados no viajan mucho.
- Sirve para proporcionar un buen servicio al cliente, los empleados necesitan transferir llamadas para contactar con la persona adecuada que pueda ayudar a un cliente.
- El negocio depende en gran medida de la comunicación telefónica, por lo que necesitamos una conexión clara que siempre esté disponible.
- El presupuesto permite pagar por los costos de instalaciones complejas y hardware específico.
- Los empleados pueden tener un equilibrio entre el trabajo y la vida privada y no se llevan el trabajo a casa.

Sistemas de refrigeración y calefacción

Debido a que las actividades a realizar entran en las categorías reposada y ligera de gasto energético, solamente serán necesarios aparatos de calefacción de calor radiante (quemadores de fuel-oil o radiadores eléctricos) o placas de contacto calientes, para temperaturas bajas y equipos de ventilación y aires acondicionados para temperaturas altas (1 aparato cada 50mt2). El apartado de estrés térmico no lo consideramos necesario para implementarlo debido a las mismas actividades a realizar.

Tipos de iluminación

Sistema de iluminación: Suministro de corriente eléctrica iluminando las habitaciones e incrementando así la seguridad de las instalaciones.

La iluminación en los lugares de trabajo de las diferentes cumplirá lo siguiente:

1. La composición espectral de la luz deberá ser adecuada a la tarea a realizar, de modo que permita observar o reproducir los colores en la medida que sea necesario.

2. El efecto estroboscópico, será evitado.

3. La iluminación será adecuada a la tarea a efectuar, teniendo en cuenta el mínimo tamaño a percibir, la reflexión de los elementos, el contraste y el movimiento.

4. Las fuentes de iluminación no deberán producir deslumbramientos, directo o reflejado, para lo que se distribuirán y orientarán convenientemente las luminarias y superficies reflectantes existentes en el local.

5. La uniformidad de la iluminación, así como las sombras y contrastes serán adecuados a la tarea que se realice.

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. Las estructuras organizacionales son los diferentes patrones de diseño para constituir una empresa, con el fin de cumplir las metas propuestas y lograr el objetivo deseado.

El tipo de estructura Organizativa utilizado en el Organigrama es: **Estructura Departamental.**

Departamentalización

La departamentalización, característica típica de las grandes empresas, se relaciona con el tamaño de la empresa y la naturaleza de las operaciones. Cuando la empresa crece, sus actividades no pueden ser supervisadas directamente por

el propietario o el director. Esta tarea de supervisión puede facilitarse asignando a diversos departamentos la responsabilidad de las diferentes fases o aspectos de esta actividad. El diseño departamental o departamentalización presenta una variedad de tipos.

Se pueden aplicar varios tipos de departamentalización son:

a. Funcional.

b. Por productos y servicios.

c. Por base territorial (geográfica).

d. Por cliente.

e. Por proceso.

f. Por proyecto.

g. Matricial.

a. Departamentalización Funcional

También denominada agrupación por función, departamentalización por funciones o incluso estructura funcional, es la organización basada en funciones que requieren actividades semejantes y que se agrupan e identifican de acuerdo con alguna clasificación funcional, como finanzas, recursos humanos, mercadeo, producción, etc. La agrupación por áreas funcionales, -conocimiento, habilidades, procesos de trabajo o función de trabajo- refleja el énfasis en las interdependencias de procesos y de escala o interdependencias sociales, en detrimento de las interdependencias del flujo de trabajo. Al departamentalizar con un criterio funcional la empresa estimula la especialización, bien sea estableciendo carreras para los especialistas dentro de su área de especialización, supervisándolos mediante personas de su propia especialidad o estimulando su interacción social,

b. Departamentalización por Productos o Servicios

La organización basada en los productos o servicios incluye la diferenciación y la agrupación de las actividades de acuerdo con las salidas o resultados (output) de la empresa. Los principales deberes y tareas relacionadas con un producto o servicio se agrupan y asignan a un departamento específico para coordinar las actividades requeridas en cada tipo de salida o resultado (output).

c. Departamentalización por Áreas Geográficas

La organización basada en la situación geográfica o territorial requiere diferenciación y agrupación de las actividades de acuerdo con la localización del sitio donde se desempeñará el trabajo, o el área de mercado servida por la empresa. La presuposición implícita en esta estructura es que, donde los mercados estén dispersos, la eficiencia se mejorará si todas las actividades relacionadas con la tarea se agrupan en un área geográfica específica. En consecuencia, las funciones y los productos o servicios, sean semejantes o no, deben agruparse sobre la base de intereses geográficos. La departamentalización por base territorial la utilizan las empresas que cubren grandes áreas geográficas y mercados extensos. Es especialmente atractiva para empresas de gran escala cuyas actividades están dispersas física o geográficamente.

d) Departamentalización por Clientes

La organización basada en la clientela (consumidores o usuarios o clientes) implica la diferenciación y agrupación de las actividades según el tipo de persona o agencia para la que se realiza el trabajo. Las características de los clientes (edad, nivel socioeconómico, hábitos de compras, etc.) constituyen la base de esta estrategia totalmente centrada en el cliente, que refleja el énfasis en el consumidor del producto o servicio ofrecido por la empresa y es recomendada cuando la empresa trata con diferentes tipos de clientes que presentan características y necesidades diferentes. La departamentalización por clientela divide las

unidades organizacionales de modo que puedan servir a un diferente tipo de cliente, cuando diferentes clientes requieren diferentes métodos y características de ventas, diferentes servicios adicionales, etc.

e) Departamentalización por Proceso

Denominada también agrupación por proceso o departamentalización por fases del proceso o por procesamiento, e incluso por el tipo de equipo. Está restringida prácticamente a aplicaciones del nivel operacional de las empresas industriales y de servicios, en especial en las áreas productivas o de operaciones. La diferenciación y agrupación se realizan mediante la secuencia del proceso productivo u operacional o mediante la distribución física y la disposición racional del equipo utilizado. La estrategia de agrupación y diferenciación está determinada por el proceso de producción de bienes y servicios.

f) Departamentalización por Proyectos

La organización basada en proyectos implica la diferenciación y agrupación de las actividades de acuerdo con las salidas y los resultados (outputs) relativos a uno o varios proyectos de la empresa. Esta estrategia se utiliza en empresas que elaboran productos que exigen gran concentración de recursos y tiempo prolongado para fabricarlos. Es el caso de los astilleros navales que producen navíos, las obras de construcción civil (edificios) o industriales (fábricas o hidroeléctricas) que exigen tecnología compleja, personal especializado y reunión de recursos diferentes durante la elaboración del producto. Como el producto es de gran tamaño y exige planeación individual y detallada y un extenso periodo para la ejecución, cada producto se trata como un proyecto especial por encargo. Esa estrategia de organización adapta la estructura de la empresa a los proyectos que se propone construir: se destacan y concentran unidades y grupos de empleados durante un largo periodo, en proyectos específicos, y se asignan los recursos necesarios para cada proyecto.

g) Estructura Matricial

Es una de las formas de organización recientemente desarrolladas, cuya utilización ha tenido éxito en situaciones en que la complejidad constituye el mayor desafío. También se denomina matriz u organización en grid. La esencia de la organización matricial es la combinación de formas de departamentalización funcional y de producto o de proyecto en la misma estructura organizacional. Es una estructura mixta, de doble entrada. La organización funcional es vertical y la organización por producto o proyecto es horizontal, y ambas se encuentran superpuestas. Ninguna de las formas de departamentalización empleadas individualmente puede contener todas las interdependencias necesarias: la organización funcional afecta el flujo de trabajo mientras la organización por proyecto impide los contactos entre los especialistas situados en los diversos departamentos. Una de las maneras de abarcar todas las interdependencias entre los departamentos es mantener las interdependencias en los niveles más altos de la jerarquía mediante la centralización exagerada, lo cual provoca innumerables problemas ya analizados. Otra manera consiste en establecer una estructura dual línea-staff. Los departamentos de línea, que tienen autoridad formal para decidir, contienen las interdependencias principales, mientras que los departamentos de staff, cuya autoridad sólo puede aconsejar, contienen las interdependencias residuales. Una tercera manera consiste en utilizar esquemas de ligación: la organización preserva la estructura tradicional de autoridad, pero superpone una sobrecarga de papeles de ligación, como comités, gerentes coordinadores, fuerzas de tarea, etc. (véase el cuadro para tratar con las interdependencias residuales, es decir, las interdependencias secundarias.

3. Detallar como mínimo seis servicios que brinde el área seleccionada (sea interna o externa a la empresa).

Servicios brindados por el área.

1. Atender los requerimientos de desarrollo de sistemas informáticos sobre la base de especificaciones funcionales elaboradas por las distintas áreas de la institución.
2. Realizar tareas de análisis, diseño, desarrollo, prueba-depuración y mantenimiento de los sistemas informáticos requeridos.
3. Documentar las actividades de diseño, desarrollo y mantenimiento de los sistemas informáticos.
4. Efectuar el control de calidad de los sistemas informáticos desarrollados o sujetos de mantenimiento.

5. Transferir los sistemas desarrollados o sujetos de mantenimiento al Departamento de Administración de Sistemas, para su implementación, soporte y capacitación.
6. Establecer normas, estándares y procedimientos para las tareas de desarrollo, control de calidad y mantenimiento de sistemas informáticos.
7. Mantener un repositorio y realizar el control de versiones del código fuente de los sistemas desarrollados.
8. Promover la adopción de nuevas herramientas, metodologías, estándares y tecnologías aplicables al ciclo de vida de los sistemas informáticos.
9. Brindar apoyo técnico a las distintas áreas de la institución, en proyectos orientados a la automatización de sus procesos.
10. Promover el uso de herramientas de Software libre y estándares abiertos.
11. Proponer normativa interna para el ejercicio de sus funciones en el área de su competencia.
12. Proporcionar información oportuna, veraz y precisa en el ámbito de sus competencias a requerimiento de su inmediato superior o autoridades superiores.

4. 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.

Jefe de desarrollo y mantenimiento.

Seleccionamos los participantes – Fuentes de información**.**

- Director de TI.
- Jefes del Mismo nivel (Producción, Administración, BD, etc.).
- Empleados de desarrollo, análisis, diseño y testing.
- Stakeholders (Clientes externos).

Acciones

Las acciones dispuestas, para la recopilación de información se realiza a través de la distribución de encuestas, cuestionarios y reuniones semanales para los distintos participantes de la retroalimentación 360.

Dichos cuestionarios y encuestas cuentan con preguntas dicotómicas, ordinales y abiertas, en el cual dependiendo del participante se recopila información sobre las competencias profesionales y personales. Por otro lado las reuniones se realizan semanalmente en cual el jefe realiza preguntas abiertas.

Entre las competencias de las cuales se recopila información se detalla:

- Liderazgo.
- Inteligencia Emocional.
- Valores Personales.
- Trabajo en Equipo.
- Comunicación.
- Creatividad.
- Organización.
- Valores Organizacionales.
- Conocimientos Técnicos.
- Dominio del Servicio.
- Servicios al Cliente.

5. Analizar la aplicación del "Coaching Eficaz" en el área seleccionada. O sea, de qué forma relevaría la situación del personal y cuáles acciones realizaría Ud. como Jefe del área seleccionada para poder aplicar correctamente el coaching.

Coaching eficaz:

- Escuchar (para detectar a tiempo problemas, inconvenientes, Ideas, sugerencias, necesidades de cada persona). Se coloca un buzón de problemas/Ideas que se revisa diariamente en cada uno de los sectores de análisis, diseño, desarrollo y testing.
  Esto dicho se complementa con reuniones de 10 minutos donde indican que problemas que tuvieron o ideas surgieron.

- Respaldar (a cada persona por sus esfuerzos, sus logros, sus problemas). También se premia con premios a los empleados del mes por participaciones e ideas y dificultades superadas.

- Ayudar (a cada persona a resolver sus inconvenientes, desarrollar su efectividad).
  En las reuniones diarias, se ayuda en conjunto a los problemas que aparecieron y se revisa al otro día si fue solucionado.

- Analizar y monitorear (los resultados, logros y tropiezos de cada persona).
  En la reunión diaria cuenta con una revisión, donde marcará en un tablero el progreso del problema/ idea.

- Proveer (posibilidades de crecimiento y capacitación para todas las personas).
  Si se repite con frecuencia cierto error, se dictan capacitaciones sobre el tema a través de videos tutoriales. También se contará con capacitaciones para nuevas tecnologías que surjan.

6. Con ejemplos del área seleccionada, explique las características de un equipo de trabajo efectivo y un equipo de trabajo equilibrado.

En el área de desarrollo y mantenimiento de SW, existen tipos de desarrollo y mantenimiento como lo son el tradicional basado en Proceso Unificado donde encontramos características de equipos equilibrados y Metodología Ágil con características de equipos efectivos con una breve comparación de las metodologías nos sirve para ejemplificar entre equipos de trabajo efectivo y equipos de trabajo equilibrado.

Equipos Equilbrados – Equipos Efectivos.

EFECTIVOS	EQUILIBRADOS
Libre expresión de todos los miembros.	Cantidad de integrantes, de acue con recomendaciones de alcance de control del líder.
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.	Disponibilidad de tiempo.
Todos están dispuestos a asumir riesgos, ya que hay una adecuada planificación y gestión de riesgos de parte del líder.	Necesidades personales y f propios.
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	Actitud (positiva, negativa, colaboración, egoísta, etc.).

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.	
Hay objetivos comunes y metas claras bien arraigados en todos los miembros.	Roles (orientado a la tarea, orienta a la relación, etc.).
Existe iniciativas, deseos y voluntad de participación, respeto por todos y siempre los miembros están dispuestos a colaborar.	Adaptabilidad al estrés.
Aceptación de decisiones por consenso general, aún cuando existan divergencias individuales.	Personalidad (introvertido, extrovertido, agresivo, sumiso, solitario, etc.).
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.	Ingenio, creatividad, generación de ideas, inquietudes, nuevos proyectos, etc.
Retroalimentación de todos los integrantes del equipo de trabajo a los efectos de pensar y poner en práctica permanente acciones de mejora continua.	Competencias técnicas y nivel de capacitación.

Tabla 92. Comparación equipos efectivos y equilibrados.

Ejemplos

- En un equipo ágil, los miembros se pueden expresar libremente cuando el Product Owner presenta los requerimientos al equipo (sprint Planning) y el equipo en conjunto con el cliente se expresa libremente fomentando iniciativas, deseos y voluntad de participación, respeto y siempre los miembros están dispuestos a colaborar y para finalizar se prioriza los requerimientos a ingresar en el próximo sprint logrando decisiones por consenso general aun así existiendo divergencias individuales.

  En cambio, en el proceso unificado los requerimientos son planificados desde el principio y no permiten la libre expresión y consensuada, lo cual hace que además el trabajo se orienta más a la tarea.

- En un equipo ágil se fomenta la delegación eficaz del líder Scrum Master o Autoorganizarse asignando a los miembros indicados para cada tarea, logrando que el trabajo en conjunto sea mejor que el trabajo individual es decir logrando un compromiso en conjunto.
  Por su parte el proceso unificado la asignación de tarea se hace al principio para los miembros especialistas que cuenten con las competencias técnicas y capacitación adecuada en cada tarea con recomendación del director del proyecto.

- En los equipos ágiles, como los miembros se pueden expresar libremente y autoorganizados, los riesgos se asumen en conjunto además de que hay una adecuada planificación y gestión de riesgos.
  En proceso unificado los riesgos son asumidos individualmente por el responsable especialista de la tarea.

- En los equipos ágiles, al ser equipos Autoorganizados y multitarea los miembros del equipo saben 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.
  En proceso unificado al estar cada miembro muy relacionado con su tarea anteponen las necesidades personales y fines propios.

- Los equipos ágiles los objetivos son planificados (Sprint Planning) a cortos plazos comunes y metas claras bien arraigados en todos los miembros
  En proceso unificado los objetivos son planificados al principio del proyecto, pero pueden no estar tan claros al principiopio del proyecto por parte del cliente.

- En los equipos ágiles, el último día del sprint se realiza la reunión de revisión de la iteración. Tiene dos partes:

  - Revisión (demostración).

- Retrospectiva: El equipo realiza retroalimentación y mecanismos de mejora continua es decir cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.
  En Proceso unificados no existe esta retroalimentación

- A diferencia de procesos ágiles que las iteraciones sprint (4 Semanas máximo) en proceso unificado hay disponibilidad de tiempo (8 a 12 meses).

- Por último, en proceso unificado no hay necesidad de perfiles, pueden tener distintos perfiles y actitud como Actitud (positiva, negativa, colaboración, egoísta, etc. y Personalidad (introvertido, extrovertido, agresivo, sumiso, solitario, etc.)

7. Detallar las funciones que podría tener un Tablero de Comandos del área seleccionada y el diseño de la pantalla principal del mismo.

- Tablero de comandos del jefe de desarrollo y mantenimiento.
- Tenemos gráficos de torta y barra para las siguientes áreas lo cual reflejan los siguientes conceptos:
- Indicadores.
- Objetivos.
- Tolerancias.
- Alarmas.
- Acciones.
- Compromisos.

Tenemos un gráfico por cada uno de los sectores del área para reflejar los conceptos mencionados:

- Análisis: Tenemos un gráfico de torta, que cuenta, con la cantidad de proyectos que estén en la etapa análisis iniciados, pendientes.

  Alarma: Cuando la cantidad de proyectos pendientes es igual o mayor a 40%.
  Objetivo: Se cumple cuando la cantidad de proyectos iniciados es igual a 80%.
  Unidad de tiempo: Se actualizan el tablero al cabo de dos meses.

- Diseño: Tenemos un gráfico de barra que muestra los proyectos en la etapa de diseño con su duración.

  Alarma: Cuando la duración de 4 o más proyectos en etapa de diseño es igual o mayor a 2 meses.
  Objetivo: Se cumple cuando ningún proyecto supera la duración de 1 mes.
  Unidad de tiempo: Se actualizan el tablero al cabo de dos meses.

- Desarrollo: Tenemos un gráfico de barra con la cantidad de funcionalidades implementadas por proyecto por mes.

  Alarma: Cuando la cantidad de funcionalidades es menor al 10%.
  Objetivo: Cuando la cantidad de funcionalidades es mayor al 40 %.
  Referencias: Cuenta con referencias con los lenguajes de programación utilizados.
  Unidad de tiempo: Se actualizan el tablero al cabo de dos meses.

- Testing: Tenemos gráfico de barras con la cantidad de bugs y errores encontrados por proyecto.

  Alarma: Cuando se encuentran más de 200 bugs y errores en un proyecto.
  Objetivo: Cuando se encuentran menos de 50 bugs y errores.
  Unidad de tiempo: Se actualizan el tablero al cabo de un mes.
  Referencias: Cuenta con referencias de los tipos de errores y bugs.

Cada una de las alarmas cuenta con simulaciones de acciones correctivas que están a disposición del jefe de área.

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.

Para la nueva estrategia, podemos hacer un cambio radical implementando otra metodología.

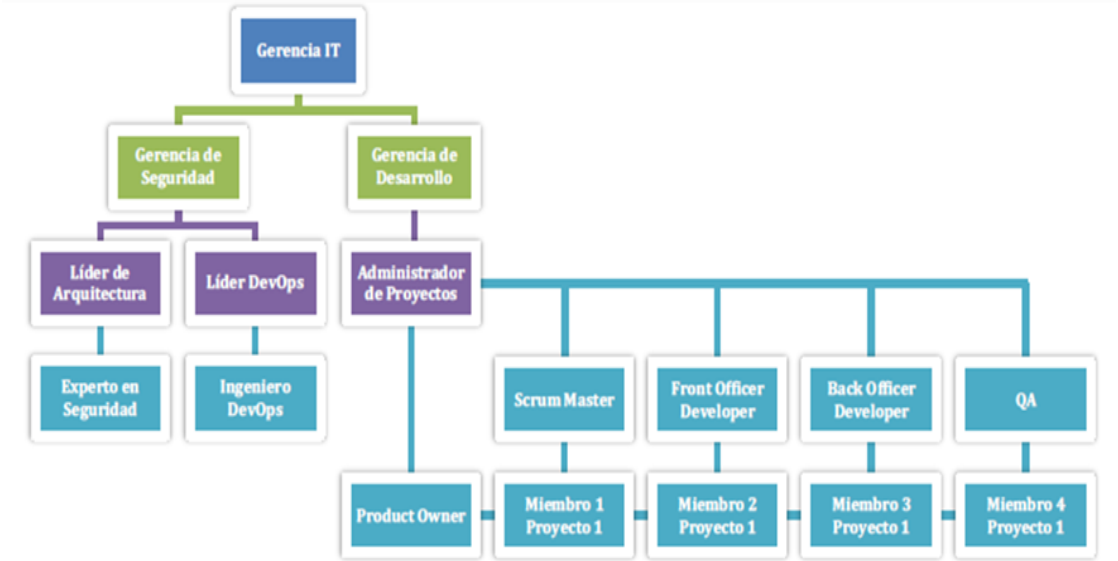


Figura 152. Nueva Estructura Organizacional

En la nueva estructura organizativa existirán tribus, una de ellas por proyecto a desarrollar, coordinadas por el Administrador de Proyectos. Cada tribu incluirá a varias cuadrillas.

Estas últimas estarán compuestas por un Product Owner, un Scrum Master, un Front Officer Developer, un Back Officer Developer y un QA.

La nueva estrategia se basa en un desarrollo iterativo e incremental utilizando metodologías ágiles, específicamente el framework scrum.

Las iteraciones se pueden entender como mini proyectos: en todas las iteraciones se repite un proceso de trabajo similar para proporcionar un resultado completo sobre producto final, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental.

Para ello, cada requisito se debe completar en una única iteración: el equipo debe realizar todas las tareas necesarias para completarlo (incluyendo pruebas y documentación) y que esté preparado para ser entregado al cliente con el mínimo esfuerzo necesario. De esta manera no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.
En cada iteración el equipo evoluciona el producto (hace una entrega incremental) a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados. Un aspecto fundamental para guiar el desarrollo iterativo e incremental es la priorización de los objetivos/requisitos en función del valor que aportan al cliente.

Cada uno de estos ciclos o sprint son de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas, límite máximo de feedback de producto real y reflexión). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

Planificación de la iteración
El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

• Selección de requisitos (2 horas). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que prevé que podrá completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

• Planificación de la iteración (2 horas). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos seleccionados. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se auto asignan las tareas, se autoorganizan para trabajar incluso en parejas (o grupos mayores) con el fin de compartir conocimiento (creando un equipo más resiliente) o para resolver juntos objetivos especialmente complejos.

Ejecución de la iteración
Con el fin de mejorar el día a día el equipo realiza una reunión de sincronización (15 minutos), normalmente delante de un tablero físico o pizarra (Scrum Taskboard). El equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con la previsión de objetivos a mostrar al final de la iteración. En la reunión cada miembro del equipo responde a tres preguntas:

• ¿Qué he hecho desde la última reunión de sincronización para ayudar al equipo a cumplir su objetivo?
• ¿Qué voy a hacer a partir de este momento para ayudar al equipo a cumplir su objetivo?
• ¿Qué impedimentos tengo o voy a tener que nos impiden conseguir nuestro objetivo?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda mantener el foco para cumplir con sus objetivos.
• Elimina los obstáculos que el equipo no puede resolver por sí mismo.
• Protege al equipo de interrupciones externas que puedan afectar el objetivo de la iteración o su productividad.

Durante la iteración, el cliente junto con el equipo refina la lista de requisitos (para prepararlos para las siguientes iteraciones) y, si es necesario, cambian o replanifican los objetivos del proyecto (10%-15% del tiempo de la iteración) con el objetivo de maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

Inspección y adaptación
El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

• Revisión (demostración) (1,5 horas). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, replanificando el proyecto.

• Retrospectiva (1,5 horas). Con el fin de mejorar la relación todo el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de eliminar o escalar los obstáculos identificados que estén más allá del ámbito de acción del equipo.

La nueva estrategia contiene más de 20 actividades, el cambio será gradual en 2 años, lo cual mejora el día a día de los integrantes del equipo y su calidad.
Mejora en la comunicación con las reuniones y planificaciones y retrospectivas, provoca mejoras en entregables más tempranamente y mejora los resultados tangibles al cliente ya que es parte del equipo, lo que provoca satisfacción continua de los clientes internos y externos (product owner) y las reuniones diarias ayuda a solucionar problemas más tempranamente.
Potencia fortalezas ya que los equipos son autoorganizados lo que reduce debilidades y amenazas.

Descripción	Inicio	Fin
Planificación.	01/01/2021	01/02/2021
Relevamiento del proceso.	01/02/2021	01/06/2021
Análisis del proceso.	01/06/2021	01/08/2021
Diseño de la solución.	01/08/2021	01/11/2021
Capacitación del personal.	01/11/2021	01/12/2021
Búsqueda de personal	01/12/2021	01/02/2022
Prueba piloto.	01/02/2022	01/04/2022
Evaluación de resultado y ajustes	01/04/2022	01/05/2022
Puesta en marcha.	01/05/2022	01/08/2022

Tabla 93. Tarea para puesta en marcha de Estrategias.

Cronograma Mensual

	Descripcion	Inicio	Fin	01/01/2021	01/02/2021	01/03/2021	01/04/2021	01/05/2021	01/06/2021	01/07/2021	01/08/2021	01/09/2021	01/10/2021	01/11/2021	01/12/2021	01/01/2022	01/02/2022	01/03/2022	01/04/2022	01/05/2022	01/06/2022	01/07/2022	01/08/2022	01/09/2022	01/10/2022	01/11/2022	01/12/2022
1																										
2	Planificación.	01/01/2021	01/02/2021																							
3	Relevamiento del proceso.	01/02/2021	01/06/2021																							
4	Análisis del proceso.	01/06/2021	01/08/2021																							
5	Diseño de la solución.	01/08/2021	01/11/2021																							
6	Capacitación del personal.	01/11/2021	01/12/2021																							
7	Búsqueda de personal	01/12/2021	01/02/2022																							
8	Prueba piloto.	01/02/2022	01/04/2022																							
9	Evaluación de resultado y ajuste	01/04/2022	01/05/2022																							
10	Puesta en marcha.	01/05/2022	01/08/2022																							

Tabla 94. Cronograma Mensual Para Puesta en Marcha de Estrategias.

ANEXOS

ANEXO 1: Manual de Usuario del Sistema Completo

Anexo 2: Diagrama de Tiempos

ANEXO 3: Costos Desagregados

Costo mensual	2021									Total
	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	
Total del mes	771223	366223	366223	366223	568223	568223	624223	624223	568223	4823007
Recursos Humanos	356000	356000	356000	356000	558000	558000	614000	614000	558000	4326000
Líder del Proyecto	36000	36000	36000	36000	36000	36000	36000	36000	36000	324000
Analista / Diseñador	64000	64000	64000	64000	16000	16000	16000	16000	16000	336000
Desarrollador Frontend	0	0	0	0	32000	32000	32000	32000	32000	160000
Desarrollador Backend	0	0	0	0	34000	34000	34000	34000	34000	170000
Tester	0	0	0	0	28000	28000	28000	28000	28000	140000
DBA	0	0	0	0	0	0	56000	56000	0	112000
Puestos o equipamiento de Trabajo	405000	0	0	0	0	0	0	0	0	405000
Notebook 1	71000	0	0	0	0	0	0	0	0	71000
Notebook 2	71000	0	0	0	0	0	0	0	0	71000
Notebook 3	71000	0	0	0	0	0	0	0	0	71000
Notebook 4	71000	0	0	0	0	0	0	0	0	71000
Notebook 5	71000	0	0	0	0	0	0	0	0	71000
Celular 1	10000	0	0	0	0	0	0	0	0	10000
Celular 2	10000	0	0	0	0	0	0	0	0	10000
Celular 3	10000	0	0	0	0	0	0	0	0	10000
Celular 4	10000	0	0	0	0	0	0	0	0	10000
Celular 5	10000	0	0	0	0	0	0	0	0	10000
Servicios en la nube	0	0	0	0	0	0	0	0	0	0
Servidor EC2	0	0	0	0	0	0	0	0	0	0
Base de datos	0	0	0	0	0	0	0	0	0	0
Servicios de Red	0	0	0	0	0	0	0	0	0	0
Capacitación	0	0	0	7602	0	0	0	0	0	7602
Curso 1	0	0	0	2534	0	0	0	0	0	2534
Curso 2	0	0	0	2534	0	0	0	0	0	2534
Curso 3	0	0	0	2534	0	0	0	0	0	2534
Servicio de Internet	10000	10000	10000	10000	10000	10000	10000	10000	10000	90000
Servicio 1	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 2	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 3	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 4	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Servicio 5	2000	2000	2000	2000	2000	2000	2000	2000	2000	18000
Marketing y Publicidad	223	223	223	223	223	223	223	223	223	2007

Tabla 95: Costos Desagregados.

CONCLUSIÓN

En el presente proyecto de tesis realizado para la carrera de Ingeniería en Sistemas de Información representa el trabajo de un año de dedicación y esfuerzo, en el que el grupo de trabajo ha afrontado todas las etapas esenciales, desde la investigación inicial hasta la implementación exitosa del Sistema Metacod. A lo largo de este proceso, se han demostrado los conocimientos adquiridos en la carrera relacionados a la gestion de proyectos, analisis de requerimientos, planificación, investigación, diseño, y se ha mantenido la aplicación de buenas prácticas para las distintas etapas trabajadas.

En la opinión de los miembros del equipo de trabajo este proyecto se destaca por su envergadura, e innovación con las soluciones propuestas al momento de realizarlo. Metacod se presenta como una solución prometedora que promete transformar la forma en que se abordan los proyectos de desarrollo de software. Permite a los usuarios concentrarse en la esencia del negocio y en la concepción de soluciones, mientras que delega la labor repetitiva de crear proyectos y códigos iniciales al sistema, de esta manera optimizando todo el proceso de desarrollo.

Además, destacamos que se ha logrado aprovechar los conocimientos adquiridos a lo largo de la carrera y nos hemos enfocado en el aprendizaje continuo junto a este grupo de cuatro personas para dominar las metodologias necesarias para el éxito de Metacod. Hemos llevado a la practica el analisis de factibilidades e impacto ambiental de un caso real.

En resumen, este proyecto de tesis es la pueba de las habilidades técnicas y capacidad para aplicar conocimientos en la resolución de problemas complejos. Más allá del trabajo requerido para la materia, Metacod tiene el potencial de servir a la industria del desarrollo de software al optimizar tiempos y costos.

BIBLIOGRAFÍA Y SITIOS WEB

- Altova, Altova UModel (2021). *Herramienta de modelado de software UML*: https://www.altova.com/es/umodel
- CUBA (2020). *Studio User Guide,* Haulmont: https://www.cuba-platform.com/documentation/
- Jacobson, I., Booch, G., Rumbaugh, J. (2008). *El Proceso Unificado de Desarrollo de Software*, Editoral Pearson Educación S.A. Madrid.
- JHipster, Dubois, J., Sasidharan, K., Pascal, G. (2021). *Release notes and Documentation*: https://www.jhipster.tech/releases/ - https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jhipster/
- Kelsey, H., Brendan, B., Beda, J. (2019) *Kubernetes Up and Running*, 2a edición, Editorial O'Reilly.
- OPENXAVA. (2021). *Documentation, Reference Guide*: https://www.openxava.org/doc/
- Palma, F. Ruiz, A. (2018) *Listas Descriptivas de Casos de Uso-Formato Recomendado*, Diseño de Sistemas FRM.
- Pressman, R. (2010) *Ingeniería de Software: Un enfoque práctico*, 7a edición, Editorial Mc Graw Hill.
- Ramakrishnan. (2006) *Sistema de Gestión de Base de Datos*, Editorial Mc Graw Hill.
- SLINGR. (2021) *Documentation, Where to start and What's SLINGR?*: https://slingr-stack.github.io/platform/