Proyecto Final de Carrera

Tema

"Diseño e implementación de una Plataforma de aplicaciones de productividad avanzada para PyMES"

Alumnos

Dalmasso, Fidel José

Hillar, Matías Agustín

Toniolo, Mateo Justo

Año

2024

Director

Dr. Silvio Gonnet

Carrera

Ingeniería en Sistemas de Información

Lugar

Universidad Tecnológica Nacional

Facultad Regional Santa Fe

Índice

1. Introducción

1.1. Contexto y motivación

En el Instituto de Desarrollo y Diseño INGAR - CONICET (Santa Fe), se identificó un problema recurrente relacionado con la finalización de proyectos técnicos avanzados destinados a la optimización de la producción. A pesar de contar con contribuciones de alta calidad técnica, muchos de estos proyectos no lograron avanzar más allá de la fase de prototipado. Este estancamiento se debió en gran parte a la falta de una plataforma informática adecuada que facilitara la gestión y ejecución eficiente de programas de optimización de la producción.

La necesidad de una plataforma de software accesible y eficiente surge de la dificultad que enfrentan las pequeñas y medianas empresas (PyMEs) para adoptar tecnologías avanzadas debido a limitaciones económicas y falta de personal calificado. Estas barreras impiden que las PyMEs mejoren su productividad con herramientas de optimización de producción que podrían ser altamente beneficiosas.

1.2. Proyecto Unidad Ejecutora PUE-INGAR

Para superar los obstáculos mencionados anteriormente, surge el "Proyecto Unidad Ejecutora (PUE-INGAR)", con el Dr. Aldo Vecchietti como responsable, y el Ing. Federico Hernández como líder de desarrollo. Este proyecto abarca los años 2016-2023 y tiene como objetivo principal el desarrollo de la "Plataforma de Unidad Ejecutora", o simplemente Plataforma PUE, un producto de software diseñado para ser fácil de implementar y escalar, permitiendo a las PyMEs integrar modelos de optimización sin necesidad de conocimientos técnicos profundos ni recursos significativos. La motivación principal es ofrecer una herramienta robusta y confiable que permita a estas empresas adoptar tecnologías avanzadas de manera práctica y efectiva.

El proyecto busca transformar la dinámica tecnológica en la región al proporcionar una solución que no solo mejore la productividad de las empresas locales, sino que también establezca un estándar tecnológico valioso tanto para el sector productivo como para la comunidad científica y empresarial.

El Proyecto Unidad Ejecutora PUE-INGAR consistió en tres etapas generales:

- **Relevamiento de Casos de Estudio (2016-2017)**
  Se analizaron las diferentes líneas de investigación y herramientas desarrolladas en el Instituto de Desarrollo y Diseño (INGAR), con el objetivo de identificar las potenciales herramientas informáticas a incluir en la plataforma, así como seleccionar los primeros casos de estudio que servirán para validar su funcionamiento.

- **Especificación de Requerimientos (2021)**
  Se definieron los requisitos funcionales y no funcionales de la plataforma de software basándose en los resultados del relevamiento de casos de estudio.

- **Diseño y Desarrollo de la plataforma (2022-2023)**
  Se diseñaron e implementaron los distintos módulos de la plataforma, tomando como base los requisitos especificados.

En el marco de la tercera etapa del proyecto, los autores de este trabajo son recomendados por el Dr. Silvio Gonnet, actual director del presente proyecto, para incorporarse al equipo de desarrollo de la Plataforma PUE. Tras un proceso de selección, los autores Dalmasso, Hillar y Toniolo se suman al equipo de desarrollo de software del INGAR - CONICET para participar en tareas relacionadas con el diseño y desarrollo de la plataforma PUE.

1.3. Objetivos de este proyecto

Durante el período de mayo a octubre de 2023, los desarrolladores de software del INGAR - CONICET participaron en el desarrollo de la Plataforma PUE, dirigida a pequeñas y medianas empresas.

1.3.1. Objetivos Generales

Este proyecto tiene como objetivo desarrollar e implementar una plataforma de software que permita a las pequeñas y medianas empresas (PyMEs) acceder a herramientas informáticas avanzadas, con el fin de mejorar la gestión productiva, especialmente en lo que respecta a la planificación avanzada. El propósito es facilitar la adopción e integración de sistemas productivos avanzados, optimizando la gestión sin requerir personal altamente especializado ni grandes recursos.

La plataforma está diseñada para cubrir las necesidades tanto de PyMEs como de usuarios expertos. Para las primeras, ofrecerá una solución sencilla y eficiente que incremente la productividad mediante tecnologías accesibles y escalables. Para los usuarios más avanzados, la plataforma incluirá funciones clave, como autenticación, gestión de seguridad, persistencia de datos, y acceso a interfaces de programación (API) y herramientas especializadas que simplifiquen el desarrollo de aplicaciones específicas.

Dicha plataforma de software consistirá de un producto de software basado en un esquema de microservicios, donde cada uno de estos servicios o aplicaciones contendrá las funcionalidades pertinentes para la resolución de un problema de optimización matemática complejo, y el análisis eventual de los resultados obtenidos. La plataforma incluirá también un servicio centralizador, destinado a la gestión de usuarios y demás aplicaciones específicas.

1.3.2. Objetivos Específicos

Se espera que la plataforma satisfaga los siguientes requisitos de funcionamiento:

- Administración de usuarios finales de la plataforma de software
- Administración de aplicaciones incluidas en la plataforma de software
- Lectura de datos desde fuentes internas y externas (planillas de cálculo, valores separados por coma, etc.)
- Ejecución de problemas matemáticos de optimización complejos

- Lectura y presentación de resultados
- Comparación de escenarios de un problema

1.4. Aportes Directos

La realización de este proyecto tiene distintos aportes según dos perfiles clave:

- **Miembros del sistema científico**
  Facilita la creación de aplicaciones de planificación avanzada basadas en modelos matemáticos para optimización de la producción.

- **Usuarios finales**
  Ofrece acceso a aplicaciones de alto valor agregado y sofisticación a un costo accesible, apoyando la adopción tecnológica y la integración de conocimiento experto.

La adopción de este software puede mejorar significativamente el proceso de toma de decisiones en las empresas, permitiendo la implementación de modelos matemáticos avanzados de manera amigable para el usuario inexperto. Este proyecto tiene relevancia tanto dentro del instituto como en el ámbito empresarial de la región, con el potencial de establecer un estándar tecnológico que contribuya a la mejora del rendimiento de las empresas locales y a la economía regional en general.

1.5. Estructura del Documento

El presente documento tiene como objetivos:

- Describir el trabajo realizado en el marco del Proyecto Unidad Ejecutora PUE-INGAR.
- Evidenciar cómo se ejecutaron las tareas y actividades pertinentes al desarrollo de un software innovador y tecnológicamente avanzado.
- Demostrar la utilidad potencial del software en el medio productivo, destacando su aporte tanto a nivel interno del instituto como en la esfera empresarial de la región.

Este documento se organiza en varios capítulos que cubren aspectos clave del proyecto. Inicia con una introducción que describe el contexto, los objetivos generales y específicos, así como los aportes del trabajo. Luego, en el Capítulo 2 se aborda la gestión del proyecto, explicando los requerimientos, etapas de desarrollo, metodología aplicada y la planificación de actividades y riesgos. A continuación, en el Capítulo 3, se presentan las tecnologías seleccionadas, justificando su elección tanto al inicio como durante el proceso de desarrollo. Posteriormente, en el Capítulo 4, se describe la arquitectura de la plataforma, incluyendo las vistas de despliegue, módulos y componentes. En el Capítulo 5 se detalla el diseño de la base de datos, explicando su estructura en MongoDB. En el Capítulo 6 de implementación se explica cómo se desarrollaron los requerimientos, las pruebas realizadas, los aspectos de seguridad, y el flujo de ejecución de las aplicaciones. Además, se dedica un capítulo a las aplicaciones desarrolladas, describiendo cada una en términos de marco teórico, entradas y salidas. Finalmente, el documento concluye con un análisis de los resultados obtenidos, reflexionando sobre el cumplimiento de los objetivos y las conclusiones del proyecto. Los últimos apartados incluyen las referencias bibliográficas y anexos que proporcionan información adicional sobre las reuniones y el diseño de la base de datos.

2. Gestión de proyecto

2.1. Requerimientos definidos

En la siguiente sección se explicitan los requerimientos inicialmente definidos por el equipo del PUE-INGAR. En la Tabla 1 se incluye un resumen de tales requerimientos.

id	Requerimiento definido
RQ1	Administración de usuarios y autenticación
RQ2	Administración de aplicaciones
RQ3	Lectura de datos de entrada
RQ4	Ejecución de una aplicación
RQ5	Visualización de resultados
RQ6	Gestión de escenarios

Tabla 1. *Requerimientos definidos.*

2.1.1. RQ1. Administración de usuarios y autenticación

Se debe poder acceder a la plataforma a través de usuarios con distintos perfiles y privilegios. Se prevén dos perfiles de usuario, a saber administrador general y usuario final. Se debe proveer una interfaz de gestión de usuarios, para su definición, autenticación y acceso la plataforma. Un usuario administrador general puede dar de alta nuevos usuarios, de ambos perfiles. Un usuario final podrá acceder solo a las aplicaciones permitidas a él por un administrador general. Luego del inicio de sesión, se debe visualizar un panel con las aplicaciones disponibles al usuario.

2.1.2. RQ2. Administración de aplicaciones

Se debe poder gestionar diferentes aplicaciones de productividad avanzada. Para ello se debe tener acceso a la creación de una nueva aplicación y modificación de las existentes, indicando:

- Nombre
- Dirección de acceso
- Descripción de uso
- Dirección de servicio web donde se aloja API
- Roles de usuario requeridos para su uso

Las aplicaciones podrán habilitarse y deshabilitarse desde el panel de inicio. Las acciones de habilitación e inhabilitación, así como el alta y baja de aplicaciones, podrán ser efectuadas solamente por un administrador general.

2.1.3. RQ3. Lectura de datos de entrada

Se debe contar con una interfaz de lectura de datos para la ejecución de modelos de planificación avanzada, por medio de archivos de tipo *Comma Separated Values* (CSV) o planillas de cálculo de tipo Excel. La secuencia de lectura deberá consistir en:

1. Seleccionar el tipo de archivo a ingresar.
2. Indicar ruta del archivo de datos.
3. Presentar datos en forma de tabla.

Para una planilla de cálculo, cada hoja del archivo representará una tabla tomando como nombre del parámetro de entrada el título de la hoja correspondiente.

Para un archivo CSV, se deberá ingresar un archivo por cada parámetro de entrada, siendo este equivalente a una hoja de un archivo de tipo Excel.

La estructura de datos propia del modelo estará dada por el o los archivos de entrada. Es decir, no se establecerá en la definición del mismo a nivel plataforma. Se precisa que el servicio deberá contemplar el manejo de errores en caso de se ingrese información que no coincida con los requerimientos de la aplicación.

Se debe poder modificar el valor de los datos ingresados dinámicamente a nivel aplicación, es decir, una vez procesados por el servicio y antes de la ejecución.

Se debe poder, una vez ejecutado un modelo, almacenar los datos de entrada y salida en una base de datos no relacional. Al momento de persistir esta información se debe indicar:

1. Identificador único del caso.
2. Fecha y hora de ejecución y almacenado.

2.1.4. RQ4. Ejecución de una aplicación

Se debe poder ejecutar un programa de planificación avanzada para cada aplicación en particular, a través del llamado a una interfaz de programación de aplicaciones (API). El usuario administrador general deberá indicar la dirección web de dicho servicio al configurar una aplicación.

La plataforma debe poder convertir los datos de entrada de una instancia de ejecución a un formato que se ajuste al estándar definido para la transferencia de datos con el API en cuestión, a la vez que debe poder convertir los datos provenientes de un API a un formato amigable con el usuario final.

El servicio web que reciba los datos de ejecución debe realizar las validaciones pertinentes, con el objetivo de contar con la información necesaria para la ejecución correcta del programa. Deberá retornar a la plataforma mensajes de error en caso de no haber sido provisto con la información correspondiente.

El servicio deberá funcionar de forma que el usuario provea a una aplicación específica datos de entrada, esta realice el procesamiento necesario y retorne la salida correspondiente. Donde los datos ingresados respeten el formato especificado en Lectura de Datos.

De esta forma, cada aplicación puede estar desarrollada en diferentes lenguajes como *GAMS*, Python o Java, o llamar la ejecución de un *solver* como *cplex*, *GLPK* o *Gurobi*, sin afectar la independencia de la plataforma con las aplicaciones.

El usuario deberá poder visualizar el estado de ejecución del modelo a medida que este se procesa. Cada aplicación deberá capturar los *logs* del programa en ejecución y mostrarlos por pantalla y eventualmente guardarlos en una base de datos.

2.1.5. RQ5. Visualización de resultados

La plataforma deberá proveer una interfaz de presentación de los resultados de la ejecución de un modelo. Para ello debe realizar el proceso inverso a la entrada de datos, es decir, transformar los datos provenientes de la API correspondiente a un formato reproducible por la plataforma.

Se deberá dividir la salida en secciones pertinentes a la naturaleza del problema (resultados, detalles, etc.), y permitir, en cada una de estas, filtrar y ordenar datos por campo.

Se deberá también mostrar una salida en forma de gráfica que se ajuste al modelo específico (diagramas de Gantt, gráficos de torta, lineales o de barra).

2.1.6. RQ6. Gestión de escenarios

La plataforma deberá proveer una interfaz de visualización y comparación de escenarios, donde se debe mostrar los resultados de la ejecución de una instancia del modelo.

Para la comparación, se deberá presentar una pantalla dividida en dos secciones, cada una con un buscador de escenarios por palabra clave. Una vez cargados dos escenarios, se deberá poder visualizar ambos en simultáneo.

2.2. Recursos humanos

El equipo de Diseño y Desarrollo de la plataforma estuvo compuesto de las siguientes personas

- Responsable de proyecto: Dr. Aldo Rodomiro Vechietti
- Líder de desarrollo y Desarrollador Senior: Ing. Federico Hernández
- Desarrollador Senior de soporte: Ing. Federico Mione
- Desarrolladores Junior:
    - Fidel José Dalmasso
    - Mateo Justo Toniolo
    - Matías Agustín Hillar

2.3. Etapas del proyecto

El proceso de desarrollo consistió en tres etapas bien definidas que se detallan a continuación.

- **Diseño e implementación de la arquitectura**
  Durante esta etapa se desarrolló la arquitectura que brinda soporte a todas las funcionalidades y servicios necesarios para cumplir con los requisitos establecidos en el apartado "2.1. Requerimientos Definidos".

- **Prototipado de aplicación de prueba**
  Durante esta etapa se coordinó el desarrollo en conjunto de una única aplicación a modo de prueba. Si bien se trató de un prototipo sin aplicación real, se planteó como objetivo alcanzar un producto funcional que cumpla con requisitos similares al resto de las aplicaciones a desarrollar.

- **Desarrollo de aplicaciones de productividad avanzada**
  Una vez alcanzado el prototipo básico con todas las funcionalidades requeridas, se replicó el producto obtenido para el desarrollo de tres aplicaciones independientes. Cada aplicación estuvo a cargo de un miembro del equipo de desarrolladores.

2.4. Metodología Utilizada

2.4.1. Prototipado evolutivo

El prototipado evolutivo es una metodología ágil que se centra en construir prototipos iniciales y mejorarlos gradualmente en iteraciones sucesivas. En lugar de intentar diseñar y desarrollar todo el sistema de una vez, se crea un prototipo funcional con las características más básicas y luego se mejora en versiones posteriores.

Esta metodología es especialmente conveniente en proyectos de software donde se estima que los requerimientos cambiarán conforme el desarrollo avanza, pero se conoce un conjunto básico de funcionalidades requeridas. Este aspecto resultó fundamental a la hora de decidir la utilización del modelo de prototipado evolutivo como guía del desarrollo de este proyecto: la existencia de dominios específicos diferentes y la necesidad de incorporar distintos productos de software de terceros para la resolución de modelos matemáticos complejos significó la obtención de un conjunto inicial de requerimientos funcionales poco definidos y/o inexistentes en ciertos casos. Asimismo, para poder abordar la diversidad de dominios se adoptó una arquitectura basada en microservicios, tal como se indica en el capítulo “*4. Arquitectura de la Plataforma*”. Este aspecto fue tenido en cuenta al momento de seleccionar esta metodología de desarrollo.

El modelo de prototipado evolutivo se trata de un proceso iterativo, en el cual la implementación de cada prototipo supone la ejecución de ciertas tareas inherentes a cada iteración (ver Figura 1) (Pressman & Maxim, 2014).

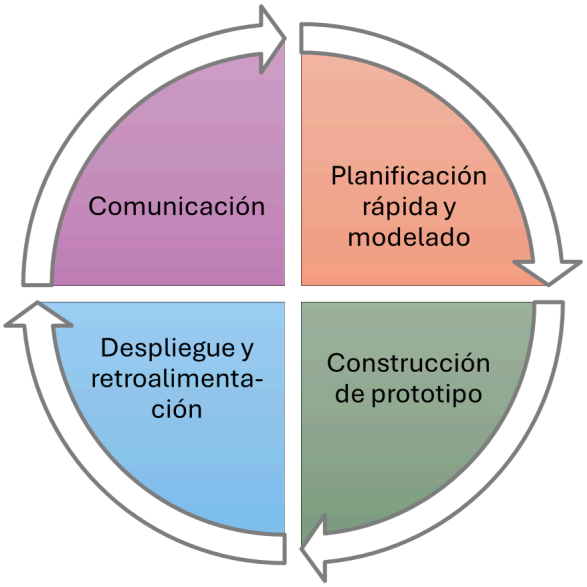


Figura 1. *Ciclo iterativo de prototipado evolutivo.*

Donde cada iteración consiste de las siguientes actividades:

1. **Comunicación**
   Consta de reuniones con stakeholders y usuarios finales con el fin de establecer objetivos principales y definir requerimientos conocidos. Se deben especificar también aspectos a definir posteriormente.

2. **Planificación rápida y modelado**
   Consta de planear una iteración completa del proceso de prototipado y realizar un diseño preliminar de los componentes a desarrollar. Se busca un enfoque en la representación de aspectos de software que serán visibles a los usuarios finales (como interfaces de usuario y diseño de salidas).

3. **Construcción de prototipo**
   La salida de la etapa anterior deriva en la construcción propiamente dicha del prototipo (desarrollo).

4. **Despliegue y retroalimentación**
   El prototipo es presentado a usuarios y stakeholders para su evaluación. La salida de esta fase se utiliza para el refinamiento de los requerimientos, alimentando la etapa de comunicación de una nueva iteración.

La aplicación de la metodología de prototipado evolutivo brinda al desarrollo las siguientes ventajas:

- **Retroalimentación temprana**
  Al construir prototipos iniciales rápidamente, se puede obtener comentarios valiosos de los usuarios o clientes antes de invertir tiempo y recursos significativos en el desarrollo completo de cada microservicio.

- **Iteraciones constantes**
  El proceso de prototipado evolutivo fomenta el desarrollo iterativo, lo que permite la mejora continua del producto de desarrollo de software, a lo largo del tiempo.

- **Flexibilidad y adaptabilidad**
  Con cada iteración, existe la oportunidad de adaptar y mejorar cada microservicio en función de los comentarios y las necesidades cambiantes del usuario.

- **Reducción de riesgos**
  Al abordar el desarrollo en pequeñas iteraciones, se reducen los riesgos de errores costosos que podrían ocurrir si se desarrollara toda la plataforma de una vez.

- **Entrega incremental**
  Permite crear versiones funcionales parciales a lo largo del tiempo, lo que permite a los usuarios comenzar a utilizar y beneficiarse de la aplicación en forma temprana en el proceso de desarrollo.

- **Escalabilidad**
  Permite optimizar y ajustar cada microservicio a medida que la demanda de la plataforma aumenta.

En resumen, el enfoque de prototipado evolutivo en el desarrollo de una aplicación basada en microservicios puede aumentar las posibilidades de éxito del proyecto, alentando una entrega más rápida, una mayor adaptabilidad a las necesidades del usuario y una reducción de riesgos durante el proceso de desarrollo.

2.4.2. Metodología adaptada al proyecto

Este proyecto se enfocó en el modelo de desarrollo de Prototipado Evolutivo (Pressman & Maxim, 2014). Según las distintas etapas del proyecto, la metodología fue variando ligeramente de la siguiente manera:

- **Etapa de "Diseño e implementación de la arquitectura a utilizar"**
  La actividad de comunicación para esta etapa se empleó para introducir a los desarrolladores el *stack* de tecnologías a utilizar para el desarrollo de la plataforma, así también como definir los objetivos del primer prototipo funcional: la aplicación *App0-Admin*.

Luego, en la planificación se diseñaron a nivel general los componentes necesarios para la implementación de la arquitectura de la plataforma, definiendo los microservicios y las interfaces de comunicación entre ellos. Además, se estableció la distribución de tareas y responsabilidades entre los miembros del equipo de desarrollo.

Durante la construcción del prototipo, el líder de desarrollo se encargó de implementar la arquitectura básica del sistema (contenedores, base de datos, servidor web) así como los servicios *back-end* e interfaces *front-end* de la aplicación de administración *App0-Admin*. Por su parte, los desarrolladores junior, junto con el desarrollador senior de soporte, resolvieron distintas tareas de *refactoring*, *troubleshooting* y documentación de la aplicación *App0*, con el objetivo de interiorizarse en el código existente y familiarizarse con la estructura del proyecto.

Las reuniones semanales consistieron en definir y asignar las tareas necesarias, discutir los avances individuales y re*solver* problemas de implementación en forma colaborativa.

En esta etapa, la retroalimentación se centró en la revisión de los avances y la validación de los componentes desarrollados, con el objetivo de garantizar la coherencia y la calidad del código.

El resultado de esta etapa fue una arquitectura de microservicios funcional, con un prototipo de aplicación de administración (*App0-Admin*) que serviría como base para la gestión de las aplicaciones de productividad avanzada.

- **Etapa de "Prototipo de la aplicación de prueba"**

  La comunicación en esta etapa se centró en la definición de los requerimientos de la aplicación de prueba, así como en la identificación de los distintos desafíos y limitaciones a resolver. Entre ellos se encontraban la integración del solver de optimización y la implementación de la interfaz de usuario.

  La planificación rápida y modelado se enfocó en la definición de los componentes de la aplicación de prueba, incluyendo la estructura de la base de datos, los servicios back-end y las interfaces front-end. Se establecieron los objetivos y las funcionalidades clave de la aplicación, así como los criterios de aceptación para la validación de los resultados.

  La construcción del prototipo consistió en la implementación de los distintos componentes de la aplicación de prueba, incluyendo su comunicación con el solver de optimización, la lectura y escritura de datos en la base de datos y la presentación tanto de entradas como de resultados en la interfaz de usuario.

  El resultado de esta etapa fue una aplicación de prueba llamada "*App1*" (detallada en la sección "8.1.1. App 1: Optimización en Problemas de Corte"). Dicha aplicación se refinó en sucesivas iteraciones, con el objetivo de alcanzar un prototipo estable y funcional que cumpliera con los requerimientos establecidos.

- **Etapa "Desarrollo de aplicaciones de productividad avanzada"**

  Una vez finalizado el desarrollo de las *App0* y *App1*, cada desarrollador fue asignado a la creación de una aplicación individual (*App2, App3, App4*), basándose en el prototipo "*App1*". Los entregables de esta fase incluyeron tanto el código desarrollado como la documentación

asociada. Las reuniones semanales evolucionaron para incluir presentaciones de avances y resolución de dudas entre los programadores, asegurando la coherencia y alineación con los objetivos del proyecto.

Para esta etapa, se estableció un esquema de comunicación con los usuarios expertos que incluía tres reuniones clave: una al inicio del desarrollo, otra aproximadamente a la mitad, y la última al final del proyecto. Estas reuniones fueron cruciales para validar los avances, ajustar los requerimientos y recibir retroalimentación esencial. Detalles de las reuniones semanales pueden observarse en el Anexo I: Registros de Meetings.

Como resultado final de las tres etapas mencionadas, se obtuvo una plataforma de aplicaciones de productividad avanzada que cumple con los requerimientos funcionales y no funcionales establecidos en la sección "Requerimientos definidos" definida en el Capítulo 2, e integra cinco aplicaciones individuales: *App0-Admin, App1, App2, App3 y App4*.

2.5. Actividades por categoría y cronograma

Las siguientes actividades se llevaron a cabo en forma transversal a las tres etapas anteriormente definidas. Su presentación no se debe a un orden cronológico, sino más bien, a una agrupación según las capas: *back-end* (ver Tabla 2), base de datos (ver Tabla 3) y *front-end* (ver Tabla 4).

2.5.1. Actividades back-end de la plataforma de aplicaciones de productividad avanzada (BE)

ID	Actividad
BE1	*Evaluar herramientas y tecnologías a utilizar.*
	En la etapa inicial del desarrollo de *back-end*, se evaluó las herramientas y tecnologías disponibles. Esta evaluación permitió seleccionar las mejores opciones para garantizar la eficiencia, la escalabilidad y la seguridad de la plataforma, asegurando así una base sólida para el desarrollo.
BE2	*Diseñar concepto básico de la arquitectura del proyecto.*
	La arquitectura del proyecto establece la estructura fundamental del software, definiendo cómo los componentes principales interactúan entre sí. En esta etapa, se determinaron los módulos clave, los patrones de diseño y los principios de escalabilidad para garantizar una base sólida en el desarrollo de la plataforma.
BE3	*Desarrollar un módulo de software que pueda ser desplegado en un microservicio y ser utilizado como API siguiendo los lineamientos de Open API.*
	Se diseñaron los módulos de software para su implementación como microservicios, lo que permitió su despliegue independiente y sus usos como API. Siguiendo los lineamientos de *Open API*, estos módulos garantizan una interfaz bien definida y documentada, facilitando la integración y la comunicación con otros componentes de la plataforma de manera eficiente y estandarizada.
BE4	*Desarrollar un módulo de gestión de autenticación y autorización.*

	Este componente asegura la identificación segura de usuarios, la gestión de permisos y el acceso controlado a recursos dentro de la plataforma. Proporciona una capa de seguridad esencial, garantizando que los usuarios tengan acceso solo a las funcionalidades y datos para los que están autorizados, lo que contribuye a la protección de la integridad y la confidencialidad de la información.
BE5	*Programar un módulo que permita la integración de código Python con el programa de modelado matemático (GAMS y similares).*
	Se desarrolló un módulo que facilita la integración de código *Python* con programas de modelado matemático como GAMS. Este módulo actúa como un puente, permitiendo la comunicación eficiente entre *Python* y el software de modelado. Posibilitó la transferencia de datos y resultados entre ambas plataformas, mejorando la interoperabilidad y la automatización del proceso de modelado y análisis matemático.
BE6	*Generar contenedores de Docker.*
	Se crearon contenedores que proporcionaron un entorno encapsulado y portátil para el despliegue de las aplicaciones, lo que facilitó el despliegue y la administración de servicios de manera eficiente. *Docker* permitió empaquetar todas las dependencias y configuraciones necesarias en contenedores, garantizando la consistencia y la escalabilidad en distintos entornos de implementación.
BE7	*Desarrollar un módulo que permita la gestión de tareas complejas en un flujo de trabajo organizado con prioridades.*
	Se desarrolló un módulo que permite la gestión eficaz de tareas complejas en un flujo de trabajo organizado con prioridades. Este componente, integrado con *GitLab* y su plataforma de *'issues'*, facilitó la asignación de tareas, la definición de niveles de prioridad y el seguimiento del progreso de las actividades. Esto optimizó la planificación y ejecución eficiente de tareas dentro de la plataforma, garantizando que las actividades más críticas sean abordadas en primer lugar y permitiendo una colaboración efectiva a través de la plataforma *GitLab*.
BE8	*Documentar los módulos y diagramas generados (clases, casos de uso y actividad).*
	Se desarrolló un documento para facilitar la comprensión, el mantenimiento y la colaboración entre los miembros del equipo de desarrollo, asegurando la claridad y la transparencia en todo el proyecto.

Tabla 2. *Lista de actividades realizadas vinculadas a back-end.*

2.5.2 Administración de bases de datos de la plataforma de aplicaciones de productividad avanzada (BD)

ID	Actividad
BD1	*Realizar el diseño conceptual y lógico de la base de datos NoSQL (MongoDB).*
	Se definió la estructura de la base de datos en términos de las entidades, relaciones y atributos que la componen (diseño conceptual), así como la organización de los datos dentro de la base de datos, incluyendo índices y esquemas de colecciones (diseño lógico).
BD2	*Desarrollar un módulo de gestión de la información de las bases de datos.*
	Se creó un conjunto de funcionalidades para la administración eficiente de los datos almacenados en las bases de datos. Esto incluye operaciones como la creación, modificación y eliminación de registros.
BD3	*Desarrollar un módulo de gestión de las estructuras de las bases de datos.*
	Se creó un conjunto de herramientas y funcionalidades que permitan administrar eficientemente la arquitectura y la configuración de las bases de datos. Esto incluye operaciones como la creación, modificación y eliminación de tablas, índices, vistas y otras estructuras de la base de datos.
BD4	*Diseminar datos en las bases de datos.*
	Se realizó la distribución de información dentro de las bases de datos.
BD5	*Verificar la integración con otros componentes de la plataforma.*
	Se verificó que la base de datos pueda comunicarse de manera efectiva y sin problemas con otros módulos, servicios o aplicaciones que formen parte de la misma plataforma. Esto incluye la verificación de la compatibilidad de interfaces de programación de aplicaciones (API), la correcta transmisión de datos entre la base de datos y otros componentes, así como la validación de la interoperabilidad de funciones y la consistencia de los datos compartidos
BD6	*Documentar las estructuras de las bases de datos NoSQL.*
	Se definió documentación sobre las estructuras de las bases de datos que facilitó la comprensión, el mantenimiento y la colaboración entre el equipo de desarrollo, asegurando la claridad y la transparencia en todo el proyecto.
BD7	*Documentar los módulos generados.*
	Se definió documentación de los módulos generados que facilitó la comprensión, el mantenimiento y la colaboración entre el equipo de desarrollo, asegurando la claridad y la transparencia en todo el proyecto.

Tabla 3. *Lista de actividades realizadas vinculadas a Base de Datos.*

2.5.3. Front-end de la plataforma de aplicaciones de productividad avanzada (FE)

ID	Actividad
FE1	*Diseñar y desarrollar interfaces de usuario avanzadas que involucran un gran volumen de información y gráficas complejas para ser desplegados en microservicios.*
	Se diseñaron componentes de interfaz de usuario (UI) intuitivos y responsivos, se implementaron técnicas de visualización de datos eficientes y escalables, y se integraron con los microservicios que proporcionan la información necesaria para la interfaz.
FE2	*Integrar los módulos de la interfaz de usuario con otros microservicios para obtener información a través de API.*
	Se realizó la configuración de solicitudes *HTTP* o llamadas a API específicas para recuperar datos relevantes y enviar o recibir información entre la interfaz de usuario y los microservicios. El objetivo principal fue permitir que la interfaz de usuario acceda a la funcionalidad y los datos proporcionados por los microservicios de manera eficiente y segura.
FE3	*Desarrollar un componente que permita la interacción por medio de web sockets para obtener el estado de ejecución de una tarea.*
	Se facilitó la interacción entre componentes de front-end y back-end mediante el protocolo de *WebSockets*, lo que permite a los usuarios obtener en tiempo real el estado de ejecución de un modelo.
FE4	*Desarrollar un módulo para el manejo de credenciales de autenticación.*
	Se proporcionaron funcionalidades para almacenar, cifrar, recuperar y verificar credenciales, asegurando así la confidencialidad y la integridad de la información sensible. Además, se implementaron prácticas de seguridad recomendadas para proteger las credenciales contra posibles amenazas, garantizando la autenticación segura de los usuarios en el sistema.
FE5	*Documentar los módulos y mockups generados.*
	Se realizaron documentos que facilitan la comprensión, el mantenimiento y la colaboración entre el equipo de desarrollo, asegurando la claridad y la transparencia en todo el proyecto.

Tabla 4. *Lista de actividades realizadas vinculadas a front-end.*

2.6. Cronograma y estimación de esfuerzo

La planificación inicial del proyecto se encuentra detallada en el cronograma de la Figura 2.

Figura 2. *Cronograma de actividades.*

En el documento de plan de proyecto inicial, se presenta el cálculo completo de estimación usando el método de Puntos de Caso de Uso. En este, se menciona que el esfuerzo estimado para el desarrollo completo del proyecto es de, aproximadamente, 2400 horas-hombre (TEF, *Total Estimated Effort*).

A su vez, teniendo en cuenta tanto la planificación del proyecto (cronograma), como la cantidad y correspondiente asignación de los recursos humanos; se obtiene un total de 2340 horas-hombre disponibles. Este resultado se obtiene de:

$$(6hsJR * 3desarrolladores + 8hsSR * 1desarrollador) * 5das/semana * 18semanas = 2340horas$$

2.7. Riesgos

Como se plantea en el plan de proyecto, a continuación se muestra la identificación de riesgos previa al comienzo de este desarrollo, junto con el plan de contingencia propuesto y detalles sobre su implementación.

2.7.1. Identificación de Riesgos

Las categorías que se utilizaron en esta sección se observan en las Tablas 5, 6 y 7.

Categoría	Impacto
1	Despreciable
2	Marginal
3	Crítico
4	Catastrófico

Tabla 5. *Tipos de impacto para un riesgo.*

Categoría	Probabilidad
1	Ínfima
2	Baja
3	Media
4	Alta

Tabla 6. *Probabilidad de activación de un riesgo.*

Categoría	Tipo de riesgos
1	Análisis
2	Planificación
3	RRHH
4	Recursos
5	Producto
6	Calidad
7	Herramientas
8	Estimación

Tabla 7. *Tipos de riesgos.*

A continuación en la Tabla 8 se detallan los riesgos identificados en el presente proyecto, indicando el nivel de exposición al riesgo ($ER = P * I$).

ID	Riesgo	Tipo	Probabilidad	Impacto	ER
R1	Falta de requerimientos	Análisis	2	3	6
R2	Falla en el relevamiento de información	Análisis	3	3	9
R3	Imposibilidad de cumplir con la cantidad de horas de trabajo semanales de cada integrante del equipo	RRHH	3	2	6
R4	Un integrante del equipo abandona el proyecto	RRHH	1	4	4
R5	Un integrante se ausenta del proyecto temporalmente	RRHH	3	2	6
R6	Cantidad de integrantes insuficientes para desarrollar el proyecto	RRHH	2	3	6
R7	Desconocimiento del uso de las herramientas y tecnologías necesarias	Herramientas	3	2	6
R8	Problemas con la tecnología utilizada, como una falta de compatibilidad con otras aplicaciones o problemas con la escalabilidad.	Herramientas	3	3	9
R9	No entregar a tiempo el producto	Planificación	2	2	4
R10	Cambios en el diseño del producto	Producto	2	3	6
R11	Cambios en los requerimientos	Análisis	1	4	4
R12	Incumplimiento con los estándares de	Calidad	3	3	9

	calidad requeridos debido a problemas como un diseño defectuoso o una falta de pruebas adecuadas				
R13	El proyecto podría verse afectado por problemas con la adquisición de licencias, hardware o software necesario para el proyecto.	Recursos	4	2	8
R14	Demoras para el análisis de Aplicaciones de plataforma	Análisis / Planificación	1	2	2
R15	No disponibilidad de los usuarios expertos de las aplicaciones de optimización a programar	RRHH	3	2	6
R16	Incapacidad para cumplir con los plazos establecidos	Planificación	2	3	6
R17	Cambios en el equipo de proyecto durante la ejecución	RRHH	1	2	2
R18	Desviación significativa de ejecución del proyecto en la estimación original	Estimación	2	3	6

Tabla 8. *Lista de actividades realizadas vinculadas a Front-end.*

2.7.2. Evaluación de riesgos y plan de contingencia

La Tabla 9 presenta los riesgos mencionados anteriormente ordenados de mayor a menor según su ER. Además, se agregan las medidas preventivas y las medidas correctivas que pertenecen al plan de contingencia.

ID	Riesgo	ER	Medidas preventivas	Medidas correctivas
R2	Falla en el relevamiento de información	9	Realizar entrevistas y reuniones con los usuarios y los responsables del proyecto para obtener información detallada sobre los requisitos del software.	Redefinir los requisitos del software y volver a planificar el proyecto si los requisitos originales no pueden ser cumplidos.
R8	Problemas con la tecnología utilizada, como una falta de compatibilidad con otras aplicaciones o problemas con la escalabilidad.	9	Realizar pruebas de compatibilidad con otras aplicaciones y sistemas que serán integrados con el software. Evaluar la escalabilidad de la tecnología y asegurarse de que pueda manejar los requerimientos de volumen y tráfico que se espera para el software.	Evaluar la posibilidad de actualizar o cambiar la tecnología utilizada si se detecta que es la causa de los problemas.
R12	Incumplimiento con	9	Desarrollar un plan de pruebas	Identificar el problema

	los estándares de calidad requeridos debido a problemas como un diseño defectuoso o una falta de pruebas adecuadas		detallado que cubra todos los aspectos del software y que sea ejecutado por un equipo dedicado de prueba. Establecer un proceso de control de calidad que cubra todas las etapas del ciclo de vida del software, desde la planificación hasta la implementación.	específico y buscar soluciones que permitan corregir el diseño defectuoso o las pruebas insuficientes. Volver a planificar el proyecto y rediseñar el software si es necesario para cumplir con los estándares de calidad requeridos.
R13	El proyecto podría verse afectado por problemas con la adquisición de licencias, hardware o software necesario para el proyecto.	8	Identificar las licencias, hardware y software necesarios para el proyecto y realizar un análisis de costo-beneficio para determinar la viabilidad del proyecto. Asegurarse de que se cuenta con el presupuesto necesario para la adquisición de las licencias, hardware y software.	Evaluar la posibilidad de reemplazar el hardware o software adquirido si no cumple con los requerimientos del proyecto.
R1	Falta de requerimientos	6	Establecer un proceso formal para recopilar, documentar y validar los requerimientos del software. Utilizar herramientas de gestión de proyectos para documentar y realizar un seguimiento de los requerimientos del software.	Identificar los requerimientos faltantes y discutirlos con los interesados en el proyecto. Evaluar el impacto en el proyecto y realizar los ajustes necesarios en los plazos y recursos para adaptarse a los nuevos requerimientos.
R3	Imposibilidad de cumplir con la cantidad de horas de trabajo semanales de cada integrante del equipo	6	Identificar los recursos del equipo y sus disponibilidades para trabajar en el proyecto, asegurándose de que se tenga en cuenta la carga de trabajo de cada uno.	Redistribuir las tareas y responsabilidades del equipo para hacer frente a los problemas de carga de trabajo. Evaluar la posibilidad de ampliar el equipo o de contar con recursos adicionales para ayudar en el proyecto.
R5	Un integrante se ausenta del proyecto temporalmente	6	Realizar una distribución equilibrada de tareas y responsabilidades entre los miembros del equipo, para evitar la dependencia excesiva de un miembro del equipo en particular.	Identificar las tareas específicas que están afectando el progreso del proyecto debido a la ausencia de un miembro del equipo y asignarlas a otros miembros del equipo o buscar recursos adicionales que puedan ayudar a completar el trabajo a tiempo.
R6	Cantidad de integrantes	6	Establecer un plan de contingencia para cubrir la falta	Reasignar tareas y responsabilidades entre los

	insuficientes para desarrollar el proyecto		de personal, que incluya la identificación de posibles sustitutos o recursos adicionales que puedan ayudar en el proyecto.	miembros del equipo para hacer frente a la falta de personal. Evaluar la necesidad de buscar recursos adicionales que puedan ayudar a completar el trabajo a tiempo, como la externalización de tareas o la contratación de personal adicional.
R7	Desconocimiento del uso de las herramientas y tecnologías necesarias	6	Proporcionar acceso a documentación técnica y tutoriales para las herramientas y tecnologías necesarias.	Identificar las áreas específicas en las que los miembros del equipo tienen dificultades y proporcionar capacitación adicional.
R10	Cambios en el diseño del producto	6	Asegurarse de que los stakeholders relevantes participen en la especificación de requerimientos y en la validación del diseño para minimizar la probabilidad de cambios en el futuro	Establecer un proceso formal para implementar y validar los cambios en el diseño del producto.
R15	No disponibilidad de los usuarios expertos de las aplicaciones de optimización a programar	6	Proporcionar una capacitación básica sobre el software de optimización a los miembros del equipo que no son usuarios expertos, para que puedan apoyar en caso de que los usuarios expertos no estén disponibles temporalmente.	Trabajar con los usuarios expertos para establecer plazos y entregas adicionales, si es posible, para compensar por la pérdida de tiempo debido a su no disponibilidad.
R16	Incapacidad para cumplir con los plazos establecidos	6	Establecer plazos realistas y basados en una evaluación detallada de los requerimientos y el alcance del proyecto.	Identificar la causa raíz del retraso y desarrollar un plan de acción para corregir el problema.
R18	Desviación significativa de ejecución del proyecto en la estimación original	6	Realizar una evaluación detallada de los requerimientos y el alcance del proyecto para poder hacer una estimación realista del tiempo y los recursos necesarios.	Revisar y actualizar el plan de proyecto y los plazos para reflejar la nueva estimación y redefinir las tareas y responsabilidades.

Tabla 9. *Lista de actividades realizadas vinculadas a Front-end.*

2.7.3. Implementación de plan de contingencia de riesgos

El plan de contingencia de riesgos fue implementado con el objetivo de mitigar los posibles obstáculos que podrían surgir durante el desarrollo del proyecto. Este plan contemplaba diversas acciones preventivas y correctivas para abordar los riesgos identificados en la etapa de planificación, garantizando que el equipo de desarrollo pudiera reaccionar de manera eficaz ante cualquier eventualidad. A continuación, se detalla cómo se gestionaron y resolvieron algunos de los principales riesgos que se presentaron a lo largo del proyecto.

- **Riesgo R7: Desconocimiento del uso de las herramientas y tecnologías necesarias**
  Al inicio del proyecto, se identificó que los desarrolladores no estaban completamente familiarizados con todas las tecnologías necesarias para comenzar a trabajar, lo que motivó la implementación de un curso de capacitación centrado en herramientas clave, como Vue.js, antes de comenzar con el desarrollo. A lo largo del proyecto, surgieron nuevas tecnologías a medida que el desarrollo avanzaba, lo que requirió acciones adicionales. En particular, se llevaron a cabo reuniones de capacitación lideradas por el Jefe de Proyecto para abordar el uso de herramientas más complejas, como *HopeIT Engine*. Durante el desarrollo de cada aplicación individual, los desarrolladores también enfrentaron la necesidad de adaptarse a nuevas herramientas, como *GAMS* y *Gurobi*. En esta etapa, la resolución del riesgo se enfocó en la autoformación a través del análisis de documentación técnica, la búsqueda de cursos en línea y la consulta de ejemplos de código en repositorios de internet. Este riesgo se gestionó exitosamente, lo que permitió a los desarrolladores continuar con el proyecto sin retrasos significativos, aunque sí requirió una inversión de tiempo adicional para la capacitación continua, tanto individual como grupal.

- **Riesgo R8: Problemas con la tecnología utilizada, como una falta de compatibilidad con otras aplicaciones o problemas con la escalabilidad**
  Este riesgo surgió en las etapas tempranas del desarrollo, específicamente durante el trabajo en la aplicación prototipo (*App1*), que utilizaba *GAMS* como herramienta clave. La versión de *GAMS* empleada solo funcionaba en sistemas operativos *Windows*, lo que presentó un desafío de compatibilidad con el entorno de desarrollo. Aunque el plan de contingencia sugería la posibilidad de actualizar o cambiar la tecnología, se decidió implementar una máquina virtual con *Windows* en los servidores del instituto INGAR para permitir que *GAMS* operara dentro de ese entorno.

  Esta solución exigió un análisis de alternativas y tiempo adicional de trabajo. Los desarrolladores estuvieron involucrados en la evaluación y la toma de decisiones, mientras que la implementación quedó a cargo del jefe de proyecto. La gestión de este riesgo afectó la planificación del proyecto, ya que la decisión sobre cómo resolverlo tomó aproximadamente dos semanas, a lo que se sumó el tiempo necesario para configurar la máquina virtual y garantizar su correcto funcionamiento (solo afectó a las tareas del jefe de proyecto).

- **Riesgo R10: Cambios en el diseño del producto**
  En etapas avanzadas del desarrollo, surgió un riesgo relacionado con cambios en el diseño del producto. Aunque se contaba con un acceso inicial a los requerimientos y al dominio de cada

aplicación, durante el desarrollo, los usuarios de dominio comenzaron a tomar decisiones sobre cómo presentar la interfaz. Esto incluyó elecciones sobre tipos de gráficos, grillas y tablas, que variaban entre las aplicaciones debido a las diferencias en la información con la que se trabajaba.

Este riesgo, asociado principalmente al desarrollo de la interfaz gráfica (*front-end*), se materializó en cambios continuos en el diseño, lo que afectó las últimas iteraciones del proyecto y provocó un alargamiento en el cronograma de desarrollo de varias semanas.

Para abordar este riesgo, se decidió agilizar la comunicación entre cada desarrollador y su respectivo usuario, eliminando formalismos y adoptando un enfoque más informal. Esto permitió una retroalimentación más rápida y eficiente, facilitando la adaptación de los diseños a las nuevas solicitudes de los usuarios y minimizando el impacto en el calendario del proyecto. Esta medida ayudó a gestionar de manera más efectiva los cambios de diseño y a mantener el avance del desarrollo dentro de plazos razonables.

Aunque la prevención fue efectiva en parte, la necesidad de ajustar el diseño a las nuevas decisiones de los usuarios introdujo retrasos en la fase final del proyecto.

2.8. Esfuerzo total real y cumplimiento de plazos

A continuación, se presenta un análisis de cada etapa del proyecto. En la Figura 3 se ilustra el desvío de tiempo estimado frente al tiempo real empleado en cada una de estas etapas.

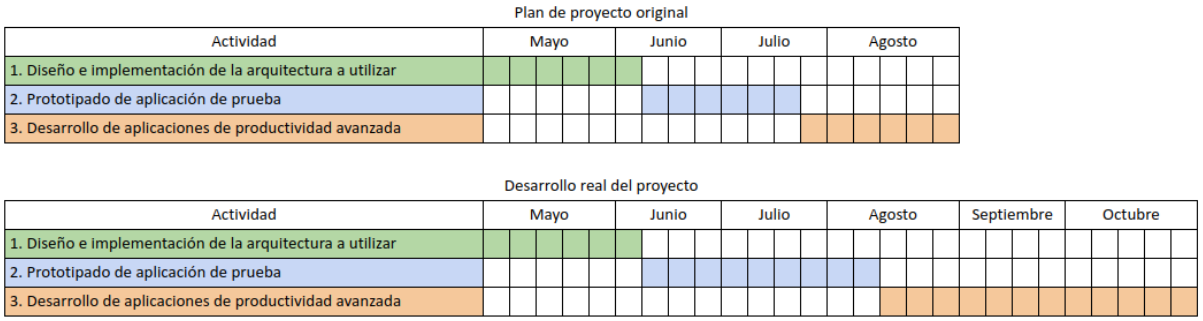


Figura 3. *Tiempo estimado vs. Tiempo real.*

En resumen, la estimación total de esfuerzo quedó por debajo de la situación real. En total, el proyecto se desarrolló en un plazo de 27 semanas (en lugar de 18), que equivalen, aproximadamente, 3510 horas-hombre. Esto representa un desvío del 50% por sobre la estimación inicial.

A continuación, se presenta un análisis retrospectivo de cada etapa del proyecto:

- **Diseño e implementación de la arquitectura a utilizar**
  Esta etapa pudo completarse en tiempo y forma, sin mayores desvíos.

- **Prototipado de aplicación de prueba**
  Esta etapa tomó más tiempo del esperado dado a diversas problemáticas que agregaron fricción al desarrollo. Entre ellas se encuentran: baja familiaridad con las tecnologías a utilizar, dificultades con la integración de los solvers a la plataforma y problemas técnicos de despliegue de servidores con sistema operativo *Windows*.

- **Desarrollo de aplicaciones de productividad avanzada**
  El desarrollo de las aplicaciones individuales también presentó desvíos en cuanto a la estimación de tiempo. En general, se debió a la complejidad inherente de las aplicaciones, donde cada una presentaba una serie de conjuntos, parámetros y variables específicos a un dominio particular. Esto requirió un mayor esfuerzo en la adaptación de los datos de entrada y salida en componentes visualmente amigables al usuario final.

3. Tecnologías utilizadas

En el desarrollo de este proyecto, se emplearon diversas tecnologías aquí clasificadas según su función específica en las categorías infraestructura, *back-end*, *front-end*, *solver* y gestión de proyecto.

En lo que respecta a las cuestiones de infraestructura, *back-end* y lenguajes de programación, las tecnologías a utilizar fueron dadas por el equipo de desarrollo de INGAR existente al momento de inicio de este proyecto. Esto quiere decir que la decisión de optar por las mismas no tuvo dependencia de los integrantes del presente trabajo.

3.1. Lista de tecnologías y herramientas utilizadas

La Tabla 10 detalla las tecnologías utilizadas, proporcionando información sobre su clasificación, nombre, la URL de su página oficial y la versión empleada.

Clasificación por Funciones		Tecnología	Link	Versión
Infraestructura	**Virtualización**	*Docker*	https://www.docker.com/	20.10.14
		Hyper-V	-	2023
	Base de Datos	*MinioBucket*	https://github.com/minio/minio/releases/tag/RELEASE.2022-01-25T19-56-04Z	RELEASE.2022-01-25T19-56-04Z
		MongoDB	https://www.mongodb.com/es	5.0.8-focal
		Redis	https://redis.io/	6.4.2
	Servidor Web	*Nginx*	https://www.nginx.com/	1.19
back-end		*Python*	https://www.python.org/	3.9.0
		Hopeit Engine	https://github.com/hopeit-git/hopeit.engine	0.16.5
		Open API	https://www.openapis.org/	3.0.3
		AIOHttp	https://docs.aiohttp.org/	3.8.6
front-end		*Node.js*	https://nodejs.org/es	14.17.0
		NPM	https://www.npmjs.com/	7
		Javascript	https://www.javascript.com/	v2020
		Typescript	https://www.typescriptlang.org/	4.6.3

	Vue.js	https://vuejs.org/	3.2.33
	Axios	https://axios-http.com/	0.26.1
***solver* / Optimizador**	*GAMS*	https://www.*GAMS*.com/	24.8.5
	Gurobi	https://www.*Gurobi*.com/	10.0.1
Gestión de Proyecto	*Google Meet*	-	-
	Miro Boards	https://miro.com/es/	-
	Gitlab Issues	-	-
Desarrollo	*Visual Studio Code*	https://code.visualstudio.com/	-
	Gitlab	https://about.gitlab.com/	-
	Git	https://git-scm.com/	-

Tabla 10. *Tecnologías utilizadas en el desarrollo del proyecto.*

3.2. Justificación de elección de tecnologías y herramientas

El proceso de selección de herramientas informáticas y tecnologías utilizadas en la implementación de la plataforma PUE se realizó parcialmente en etapas anteriores a la incorporación de los miembros de este trabajo al equipo de desarrollo.

3.2.1. Elección de tecnologías y herramientas definidas inicialmente

Siguiendo los lineamientos establecidos en el documento *Pliego - Proyecto Unidad Ejecutora,* se define explícitamente la decisión de utilizar las siguientes tecnologías:

- **Infraestructura y bases de datos:**
  MongoDB como base de datos no relacional centralizada, operativa para todas las aplicaciones conformantes de la plataforma.

  NGINX desplegado en contenedores *Docker* como servidor web para el acceso a aplicaciones a través de navegadores de internet.

  Hyper-V como software de virtualización para administración de sistema Windows necesario para ejecución de API *REST*.

- **Back-end:**
  OpenAPI como estándar de implementación de interfaces de aplicaciones de programación de tipo *REST*.

  HopeIt Engine: como librería de desarrollo e implementación de microservicios basados en flujo de datos para *Python*.

- **Lenguajes de programación y frameworks:**
  Python 3.9 como lenguaje de programación a utilizarse en extremo *backend* de aplicaciones de la plataforma.

  Vue.js o *React* como *framework* de desarrollo web. En este caso particular, la especificación técnica existente para el Proyecto Unidad Ejecutora PUE-INGAR dejó lugar a la elección entre las dos tecnologías propuestas. El equipo de desarrollo (con los integrantes de este trabajo ya incorporados) decidió la utilización del *framework* de desarrollo *Vue.js*, siguiendo las siguientes ventajas: facilidad de uso, mayor velocidad y performance, y disponibilidad de recursos de capacitación en la herramienta brindados por el instituto INGAR.

3.2.2. Elección de tecnologías y herramientas con participación del equipo de desarrollo completo

A partir de la incorporación de los participantes del presente al equipo de desarrollo de PUE- INGAR, las siguientes tecnologías y herramientas fueron seleccionadas de forma conjunta por todos los integrantes del equipo de desarrollo, siguiendo los siguientes criterios:

- **Infraestructura y bases de datos**
  Minio Bucket: debido a la necesidad eventual de almacenar objetos tipo *BLOB* (binary large objects) correspondientes al funcionamiento específico de las aplicaciones de la plataforma (como imágenes, gráficos, documentos de planillas de cálculo, etc.), se decidió la implementación de al menos una instancia de almacenamiento de tipo *bucket*.

- **Solver / optimizador**
  Distintos modelos matemáticos de optimización fueron provistos a los desarrolladores para la implementación de los mismos en las diferentes aplicaciones. Tanto *GAMS* como *Gurobi* son herramientas de resolución de modelos matemáticos, utilizadas originalmente por los usuarios expertos, anteriormente a la implementación de la plataforma. Se decidió mantener el uso de estas tecnologías, haciendo transparente al usuario la utilización de las mismas en la resolución de modelos matemáticos a través de las distintas aplicaciones, implementando mecanismos de acceso por parte de las aplicaciones de la plataforma a los mencionados solvers. Detalles sobre la implementación de *GAMS* y *Gurobi* pueden apreciarse en el apartado “Arquitectura de la plataforma”.

4. Arquitectura de la plataforma

Cuando se habla de la arquitectura de un sistema de software, se hace referencia al conjunto de estructuras que indican cuáles son los elementos que componen al producto, así como sus propiedades y la forma en que estos interactúan.

Cada una de estas estructuras corresponde a un enfoque arquitectónico específico, haciendo posible que su análisis de forma estructural logre dar idea del funcionamiento del sistema a distintos niveles de abstracción (tales como infraestructura de hardware, componentes de software, procesos computacionales)(*Clements y otros, 2003*).

En este trabajo se pueden destacar dos características principales que definen la arquitectura del producto a desarrollar: esquema de microservicios y despliegue local (*on premise*).

El hecho de tratarse de un despliegue local implica el alojamiento de todos los componentes del producto (desde datos hasta computadoras y dispositivos de red) en el ámbito físico de la organización desarrolladora. La elección de tal esquema se adecua a la naturaleza del producto; una plataforma de software sin requerimientos específicos de disponibilidad y seguridad y sin perspectivas de alta escala.

La opción por un esquema de microservicios se corresponde con la diversidad de dominios existentes en el ámbito. El deseo de plantear una solución de software a través de la cual se pueda hacer uso de modelos matemáticos disímiles e independientes, que utilizan herramientas tecnológicas variadas, y que permita la fácil adición de nuevos aplicativos específicos, hace natural la opción por un esquema de microservicios.

La arquitectura seleccionada facilitó que cada miembro del equipo de trabajo se haya desempeñado en el desarrollo de un microservicio diferente, con autonomía en cuanto a implementación de tecnologías, de funcionalidades y decisiones de código e infraestructura.

4.1. Definición de arquitectura de la plataforma PUE

Como parte de la etapa "Diseño e implementación de la arquitectura a utilizar", se realizó una formulación del diseño de arquitectura de la plataforma en este nivel. Las actividades que dieron resultado a este diseño van desde la BE1 a BD6 de dicha etapa, presente en el cronograma detallado en el Capítulo 2. Los integrantes de este trabajo no tuvieron participación en estas actividades.

La decisión de este esquema de despliegue está relacionada con el proceso de elección de tecnologías detallado anteriormente. Es decir, que la decisión de optar por ciertas herramientas influyó en el diseño de la arquitectura, a nivel de despliegue.

El diseño de la arquitectura de la plataforma en los niveles correspondientes a las vistas "de Módulo" y "de Componentes y Conectores" fue realizado de forma conjunta por el equipo de desarrollo completo en las actividades BE8 y BD7 del cronograma.

El diseño arquitectónico fue realizado con el propósito de regir la estructuración del diseño e implementación de las aplicaciones a desarrollar en las etapas siguientes. Es decir, que este pretendió

definir con el mayor grado de generalidad posible el funcionamiento de todas las aplicaciones de la plataforma y la interacción de sus componentes de software. Sobre la base de diseño que la documentación de la arquitectura supone, cada desarrollador implementó los aspectos específicos y funcionalidades propias de cada aplicación que están fuera del alcance de este, según su criterio.

4.2. Documento de arquitectura

La plataforma PUE se compone de un conjunto de microservicios independientes, cada uno de ellos destinado a la solución de un problema de optimización específico, sumado a un microservicio centralizador, encargado de la lógica de sesiones, usuarios y autenticación.

Cada aplicativo es desarrollado siguiendo un modelo de capas (presentación, negocio y persistencia de datos), siendo su implementación en cada uno de ellos independiente y autónoma.

A continuación se describe la arquitectura de la plataforma a través de las vistas de despliegue, módulos, y de componentes y conectores, siguiendo el modelo de documentación de arquitecturas Views and Beyond (*Clements y otros, 2003*).

4.2.1. Vista de despliegue

Esta vista describe la interacción entre componentes de hardware, como también con el usuario y sistemas externos.

En la Figura 4 se observa la segmentación del despliegue de la plataforma en capas de Cliente (*Client*), Microservicios (incluidos en *Application*) y Almacenamiento (*Storage*). Un cliente accede a la plataforma a través del servicio App0 y será redireccionado a los restantes según lo desee.

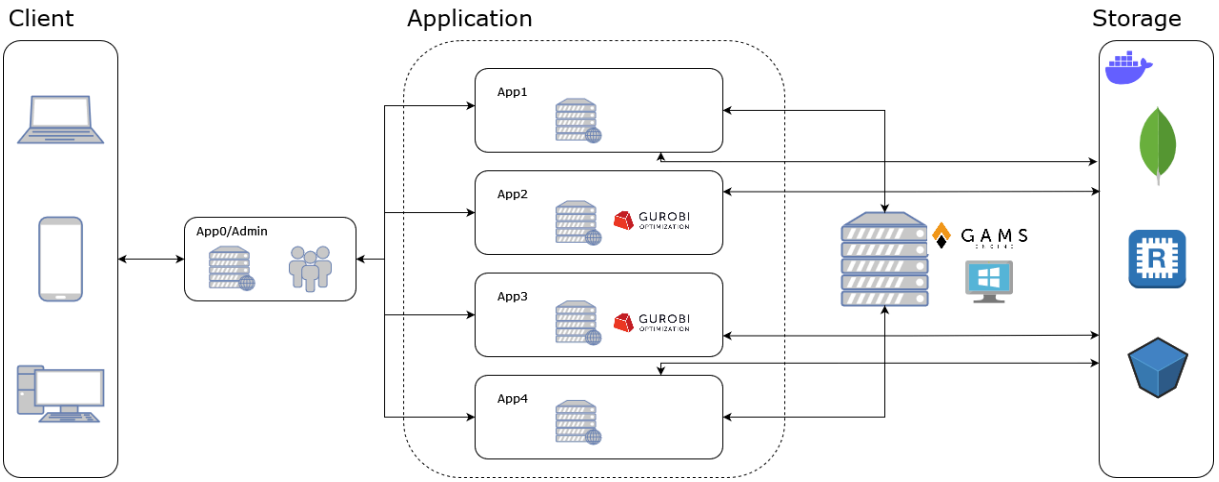


Figura 4. *Vista de despliegue de componentes.*

Cada aplicativo se comunica con un software de resolución de problemas de optimización (*solvers*) según la naturaleza del problema y accede a la capa de almacenamiento para persistir datos pertinentes a resoluciones, escenarios, y ejecuciones de tareas.

A continuación se describen las tecnologías informáticas y herramientas utilizadas en el desarrollo de la plataforma. El conjunto de ellas se presenta acorde a la siguiente categorización:

- **Front-end**
  Tecnologías utilizadas en el desarrollo de la capa de presentación de la plataforma (*framework* de *JavaScript*, lenguajes utilizados, clientes web, etc.).

- **Back-end**
  Tecnologías utilizadas para el desarrollo de la capa no visible al usuario o de lógica de negocio (lenguajes utilizados, bibliotecas de desarrollo de interfaces de aplicación, etc.).

- **Infraestructura**
  Herramientas y tecnologías utilizadas para el despliegue del producto de software. La utilización de las mismas no es inherente a los requerimientos de software definidos para la plataforma, sino que hacen a la definición de la infraestructura sobre la cual se desarrolló esta (servidor web, software de virtualización, motores de bases de datos, etc.).

- **Solver**
  Productos de software de resolución de modelos matemáticos de optimización utilizados para la resolución de problemas específicos.

En la Figura 5 se detallan las tecnologías específicas de cada categoría.

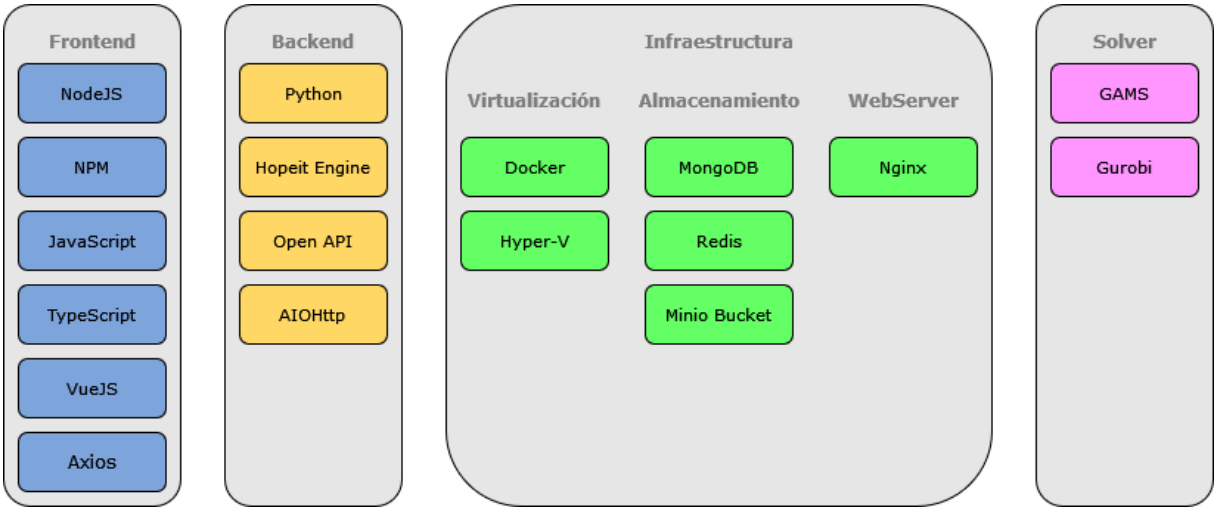


Figura 5. *Stack de tecnologías utilizadas en proyecto.*

Donde las aplicaciones se despliegan localmente en el entorno del equipo anfitrión, mientras que las instancias de almacenamiento (*MongoDB*, *Minio* y *Redis*) funcionan en contenedores *Docker*.

El esquema de despliegue del extremo web de la plataforma accesible por el usuario se define en la Figura 6.

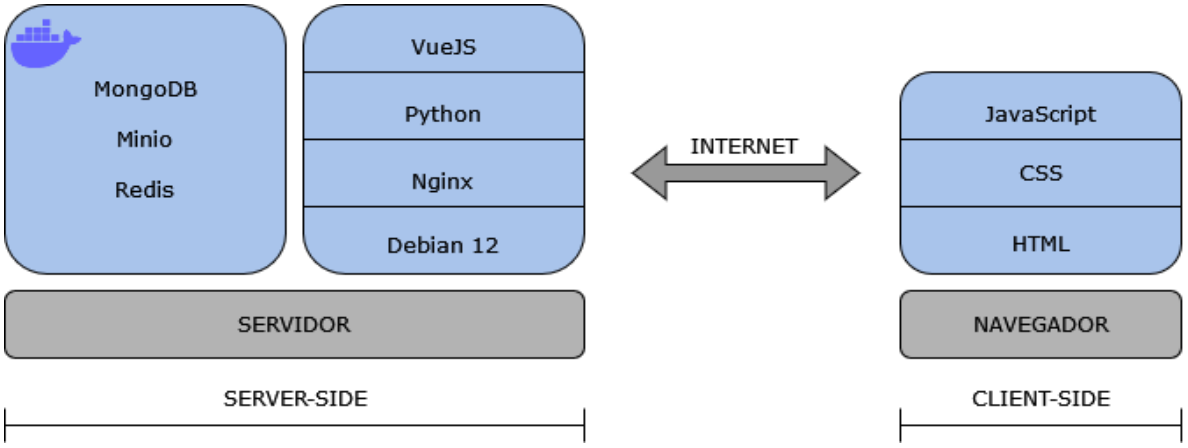


Figura 6. *Esquema de comunicación web de la plataforma.*

Se cuenta también con una aplicación de tipo *API REST* independiente, encargada de recibir peticiones para la resolución de modelos *GAMS*. Este escenario se presenta en las *App1* y *App4* (ver Figura 4).

4.1.2. Vista de módulos

Pretende ilustrar la composición del sistema a nivel módulos de código, detallando qué funcionalidades deben localizarse en qué lugar.

En la Figura 7 se presenta la vista de módulos de software para el servicio App0/Admin. El servicio App0/Admin es, como se menciona anteriormente, el encargado de procesar operaciones de usuario (autenticación, alta, baja y modificación). Un usuario accede a estas funcionalidades a través de la interfaz web provista, desarrollada en el *framework* de desarrollo web *VueJS*. La herramienta *AXIOS* permite transformar las peticiones hechas por un usuario en *requests HTTP* a ser enviadas a *endpoints* de una interfaz de aplicación tipo *API REST*.

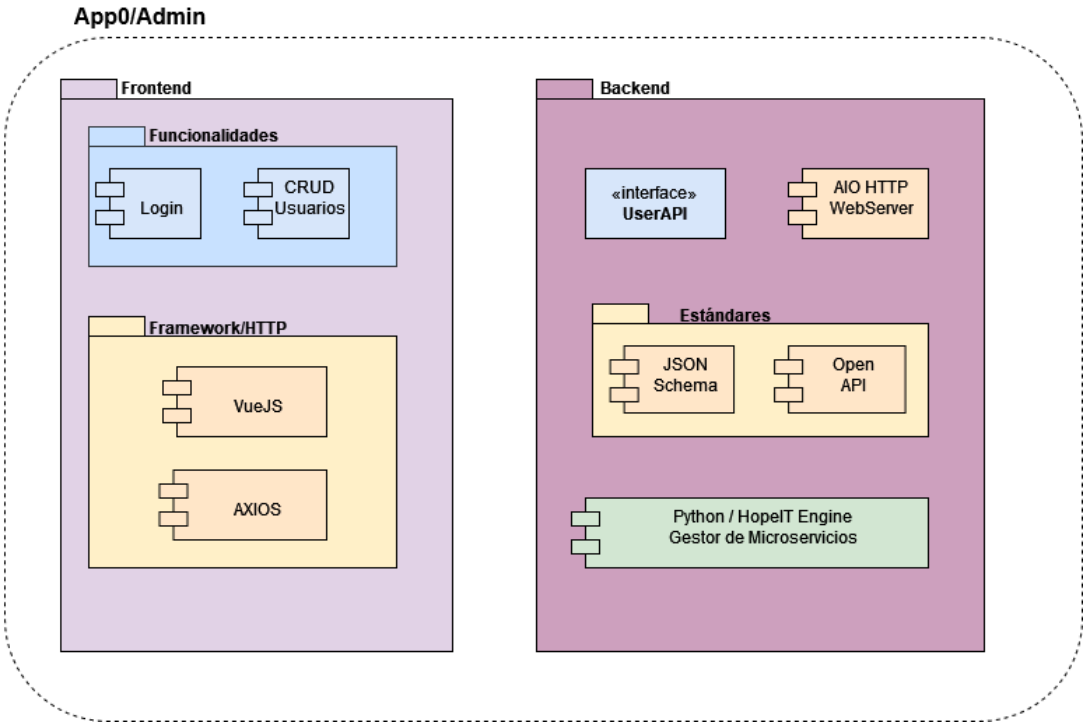


Figura 7. *Vista de módulos de software para App0/Admin.*

OpenAPI y *JSON Schema* proveen los estándares que permiten definir tanto el comportamiento del *API REST* de Usuarios como el flujo de información en formato *JSON*, respectivamente.

La herramienta *AIOHTTP Server* provee un servidor web óptimo para manejar operaciones de entrada y salida asíncronas (o concurrentes) en un entorno de aplicaciones *Python*.

La lógica de operaciones en *back-end* está escrita en lenguaje *Python*, haciendo uso del *framework HopeIT Engine*, tratándose este último de una biblioteca específica para el desarrollo de aplicaciones de microservicios basadas en flujos de datos concurrentes (Canto, 2024).

Las aplicaciones específicas tienen un comportamiento similar al de *App0* descrito anteriormente. En la Figura 8 se presenta la vista de módulos para *App1/App2/App3/App4*.

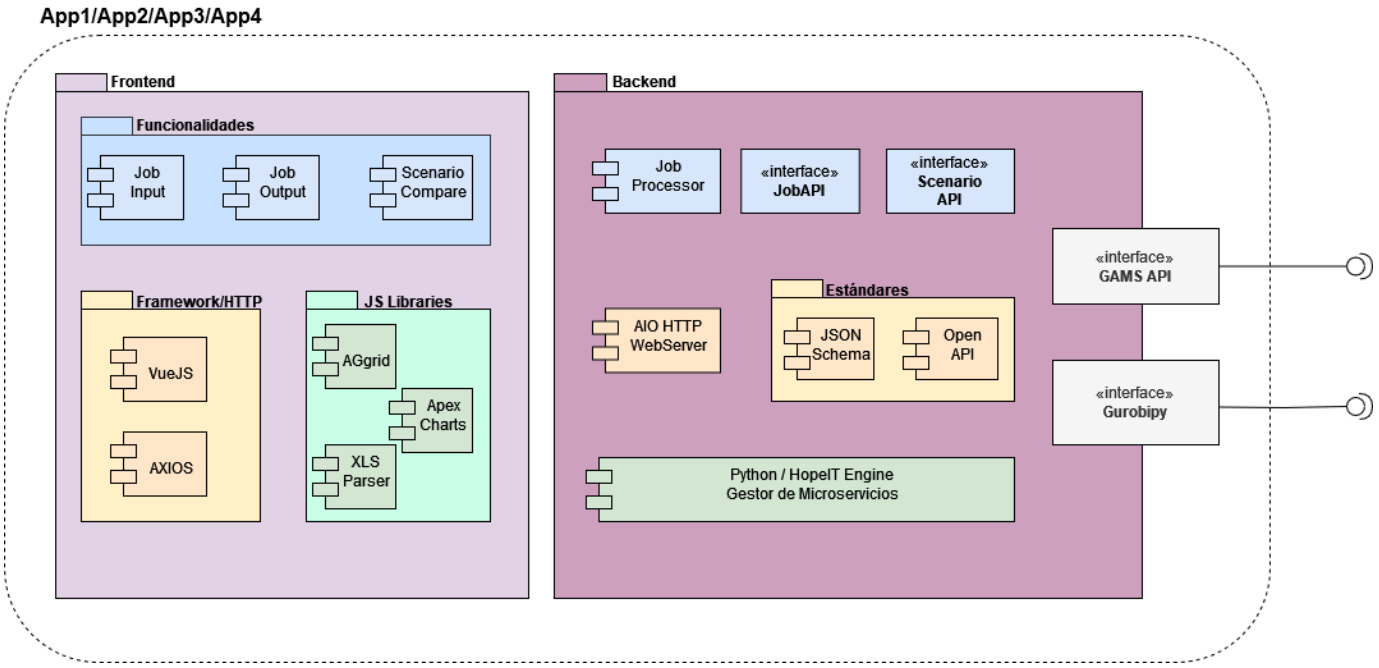


Figura 8. *Vista de módulos de software para App1/App2/App3/App4.*

En lo que respecta al extremo *front-end* de las mismas, se aprecia la utilización de las siguientes bibliotecas de *JavaScript*:

- *AGgrid*: herramienta de gestión de tablas en interfaces de usuario.
- *ApexCharts*: herramienta de generación de gráficos.
- *XLS Parser*: herramienta de conversión entre archivos de formato *xls* y *csv* a JSON, y viceversa.

A nivel *back-end*, el comportamiento es análogo a aquel en App0, estando presente en este caso las API correspondientes al manejo de los elementos *Job* y *Scenario* (más adelante se introducen estos conceptos).

Existen para *App1, App2, App3* y *App4* módulos destinados a la comunicación con el software de resolución de modelos matemáticos que corresponde en cada una de ellas. En los casos de las aplicaciones 1 y 4, el intercambio se realiza a través de una *API REST* ejecutándose en un equipo externo. En las aplicaciones 2 y 3, se accede al solver *Gurobi*, ejecutándose nativamente en el mismo entorno local, a través del módulo *Gurobipy*.

La capa de almacenamiento (Figura 9) consta de las instancias de bases de datos *MongoDB*, de almacenamiento tipo *bucket Minio* y de cacheo de bases de datos en memoria *Redis*. Cada una de ellas se ejecuta aisladas en contenedores *Docker*. Las aplicaciones acceden a las funcionalidades de estas instancias a través de *endpoints* expuestos por las mismas.

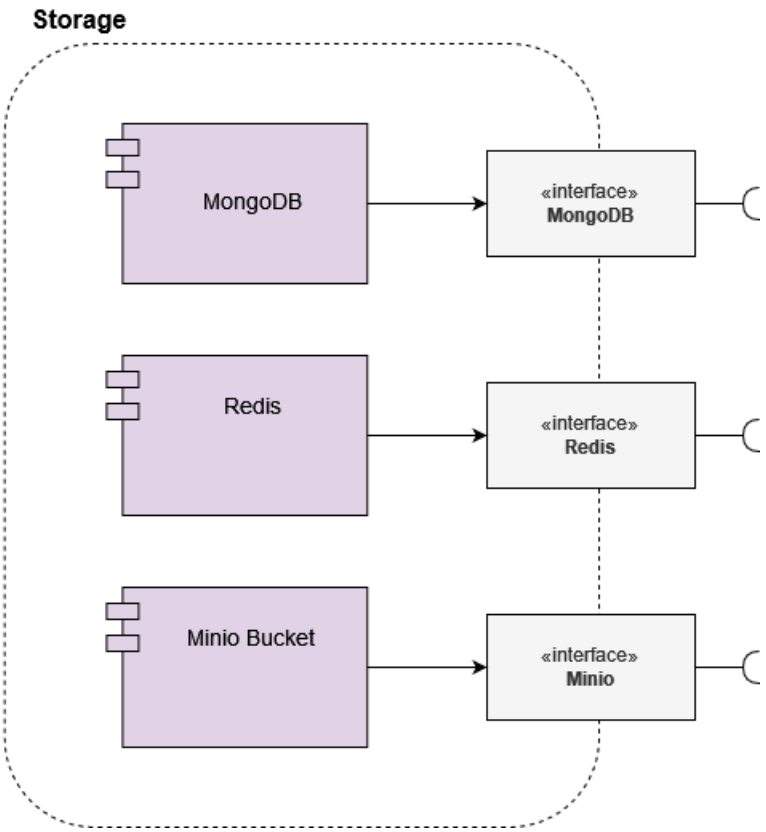


Figura 9. *Vista de módulos de software para la capa de almacenamiento.*

4.1.3. Vista de componentes y conectores

Esta vista describe la interacción específica entre módulos de software dentro de la plataforma.

En la Figura 10 se presenta la vista de componentes y conectores de *App0/Admin*. Según la vista, el usuario solo accede a las funciones de gestión de usuario, autenticación, y a la redirección al resto de las aplicaciones.

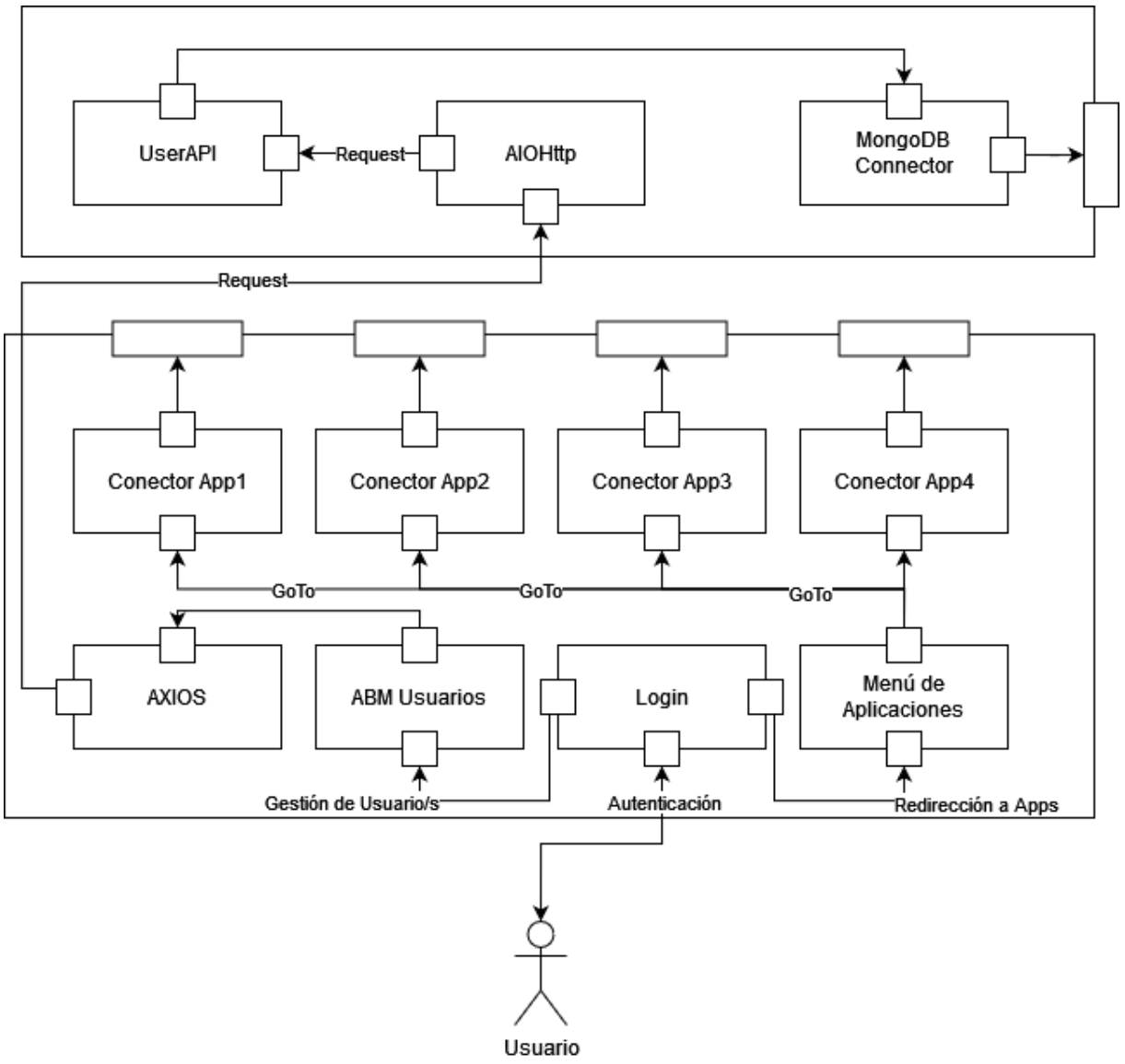


Figura 10. *Vista de componentes y conectores para App0/Admin.*

En el esquema de las aplicaciones restantes (Figura 11), se observa que el usuario puede realizar ejecuciones de un trabajo (*jobInput*), visualizar su resultado (*jobOutput*), crear y visualizar escenarios (*ScenarioCompare*). Cada solicitud realizada por el usuario se traduce en llamadas a funciones y operaciones de datos en la capa *back-end*, via solicitudes (*Request*) gestionados por *AXIOS* (del lado del usuario) y *AIOHttp* (del lado del servidor).

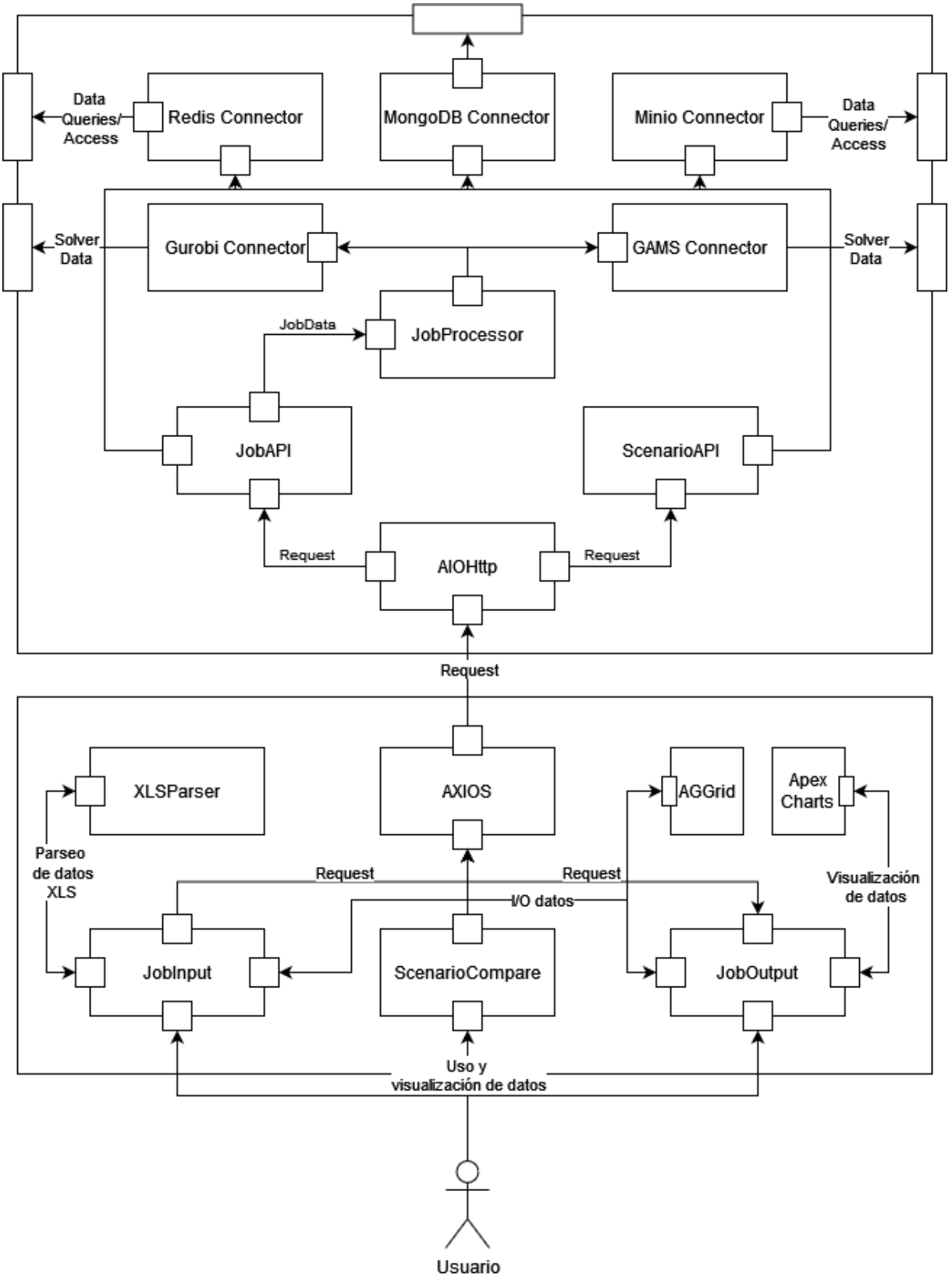


Figura 11. *Vista de componentes y conectores para App1/App2/App3/App4.*

5. Diseño de la base de datos

El diseño del modelo de base de datos a implementar en la plataforma se desarrolló en las actividades BD1 y BD6 correspondientes a la etapa “Diseño e implementación de la arquitectura a utilizar", presente en el cronograma detallado en el Capítulo 2. Este diseño fue realizado con el objetivo de ser universal para todas las aplicaciones de la plataforma. Es decir que la documentación que atiende al diseño de base de datos es una entrada al proceso de diseño e implementación de cada aplicación particular. Los desarrolladores de estas últimas debieron aplicar el modelo aquí detallado a la hora de definir estructuras de datos y funcionalidades específicas de cada una de las aplicaciones.

5.1. Modelo conceptual - Diagrama ERD

Se realizó un modelado conceptual del diseño estructural de la base de datos de la plataforma a través de un diagrama entidad-relación o ERD, por sus siglas en inglés *Entity Relationship Diagram*. En la Figura 12 se presentan las entidades fundamentales del sistema desarrollado, junto con sus atributos y la forma en que se relacionan con las demás entidades.

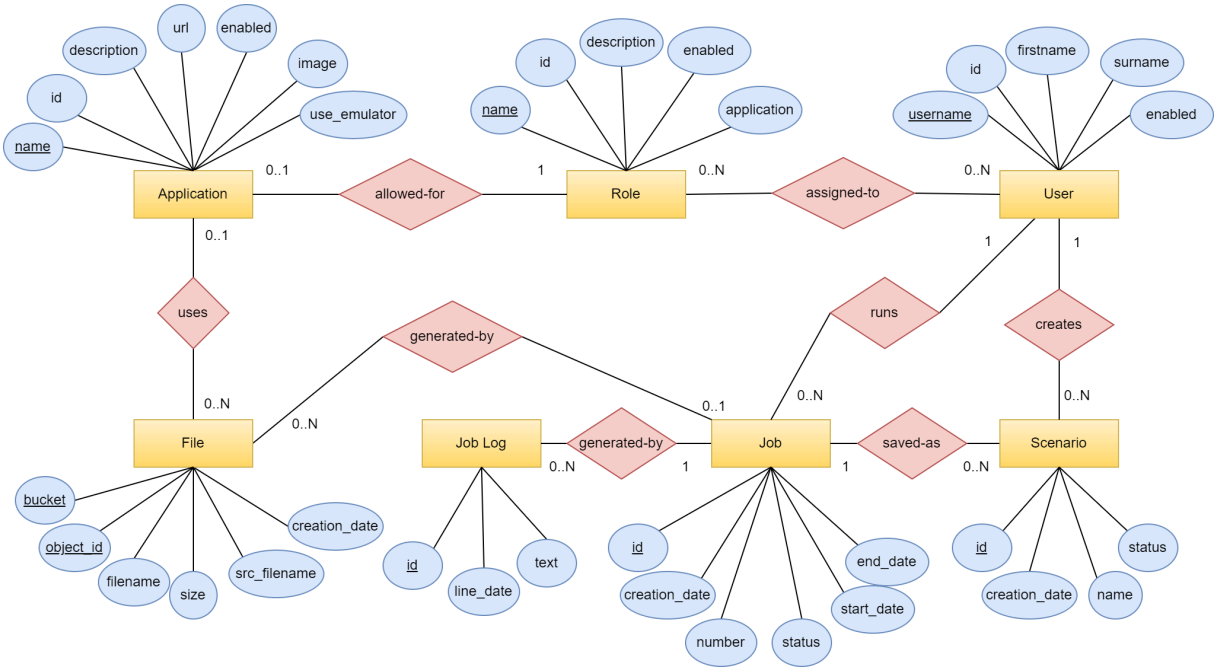


Figura 12. *Diagrama Entidad-Relación.*

Cada entidad es modelada en el diagrama como un rectángulo, del cual se desprende una serie de atributos específicos de la misma. Una relación entre dos entidades se modela con un rombo, indicando verbalmente la forma en que estas interactúan. Cada relación posee un par de cardinalidades, significando estas el posible número de ocurrencias de una entidad respecto de la otra. Por ejemplo, si se toman las entidades *Role* y *User*, la relación entre ambas entidades, descrita como *Assigned-to* (asignado a) posee cardinalidades 0..N en ambos extremos. Esto quiere decir que

un rol puede ser asignado a cero o más usuarios, a la vez que un usuario puede tener asignado cero o más roles.

Este modelo tiene como objetivo definir el esquema de funcionamiento de la capa de almacenamiento de datos en la plataforma, en su forma más abstracta, siendo esta una representación conceptual, libre de detalles de tecnología e implementación. Este modelo funcionó como entrada del diseño lógico e implementación de la capa de datos del producto desarrollado.

5.2. Modelo lógico - Diagrama IE

Un modelo lógico supone el nivel de concreción siguiente a un modelo conceptual de bases de datos. La representación del mismo se realizó a través de un diagrama de *Information Engineering* o diagrama IE (ver Figura 13). En este se observa el instanciamiento de las entidades existentes en la plataforma, junto con sus atributos y relaciones, dentro de un módulo de software.

Tomando como ejemplo las entidades *Rol* y *User* descritas en la sección anterior, observamos:

- *app0_user_role*: es una instancia de la entidad Rol existente en el contexto de la aplicación *Administration*. El atributo ID (identificador) supone la clave primaria de la instancia específica de Rol, mientras que *username* y *role* son claves foráneas de esta.
- *app0_user*: es una instancia de la entidad Usuario, existente también el contexto de la aplicación *Administration*. Su atributo *username* supone la clave privada.

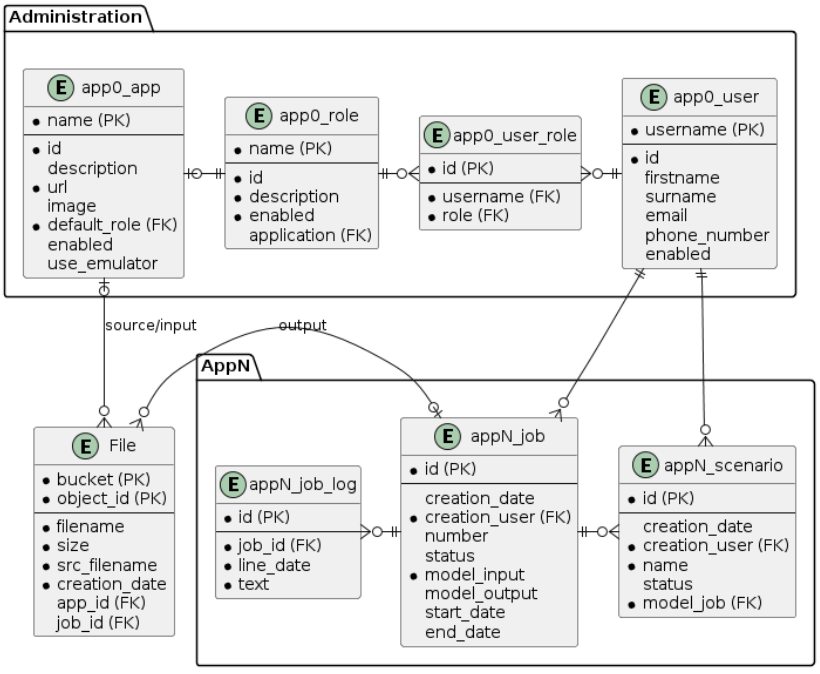


Figura 13. *Diagrama Information Engineering.*

5.3. Modelo físico - Implementación en MongoDB

La forma en que se implementa una base de datos se especifica a través de un modelo físico. En este se especifican cómo se estructuran las entidades y las relaciones entre ellas. Se definen todos los campos específicos de una entidad, junto con las relaciones entre ellas y cardinalidades correspondientes. Se trata de la representación más concreta de un modelo de bases de datos, a partir de la cual se debe implementar esta efectivamente.

En los diagramas de este apartado, las entidades están modeladas como colecciones, atendiendo al esquema de representación de datos de *MongoDB*.

La base de datos instanciada en MongoDB se denomina *puedb* e incluye tanto colecciones generales de administración como colecciones específicas a cada una de las aplicaciones. En la Tabla 11 se detallan las colecciones incluidas en *puedb*.

Colecciones de administración	Colecciones específicas a las aplicaciones			
app0.app *app0.role* *app0.user* *app0.user_role*	*app1.job* *app1.job_log* *app1.scenario*	*app2.job* *app2.job_log* *app2.scenario*	*app3.job* *app3.job_log* *app3.scenario*	*app4.job* *app4.job_log* *app4.scenario*

Tabla 11. *Colecciones incluidas en puedb.*

Con respecto al diseño físico en particular a este proyecto, cabe destacar dos aspectos importantes. Por un lado, todas las aplicaciones de productividad avanzada (*App1, App2, App3, App4*) comparten el mismo diseño físico de base de datos, a excepción de la colección *app{N}.job*, que varía según la aplicación. Por otro lado, el diseño físico de la colección *app{N}.job* se llevó a cabo durante la implementación de las aplicaciones de productividad avanzada, puesto que se necesitaron conocer los requerimientos específicos de cada una de ellas.

A continuación se hace una descripción de cada una de las colecciones y su participación en la plataforma PUE, tomando como ejemplo la *App0*.

5.3.1. Colección app0.app

En esta colección se almacenan los datos de las aplicaciones de productividad avanzada, siguiendo la estructura indicada en la Figura 14.

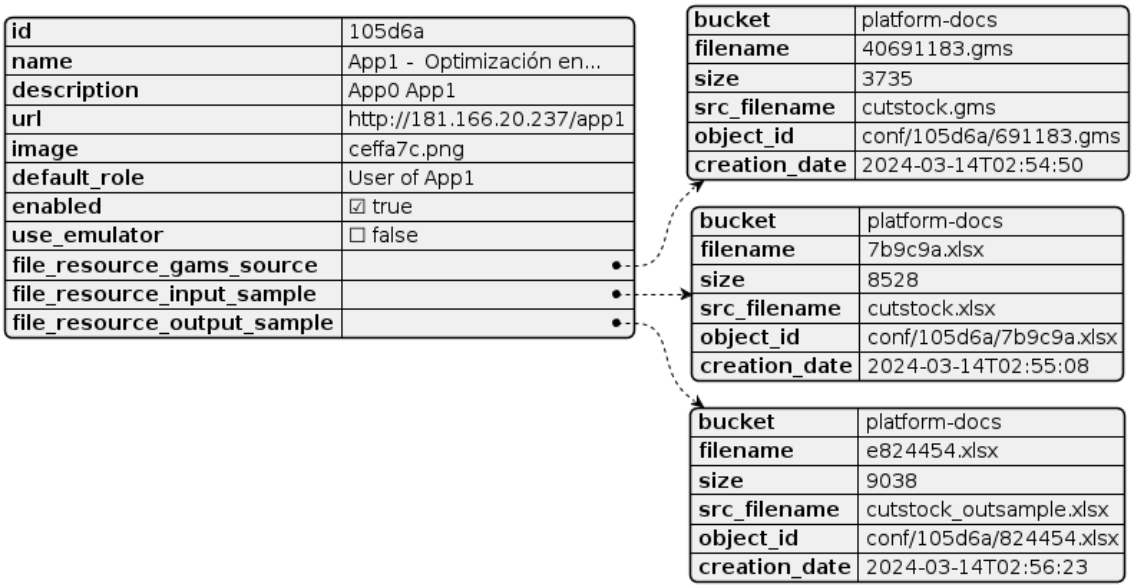


Figura 14. *Ejemplo de documento en puedb.app0.app.*

Por cada aplicación se almacenan los siguientes campos obligatorios: identificador (*id*), nombre (*name*), descripción (*description*), dirección (*url*), logo (*image*), rol de acceso (*default_role*), flag de habilitación (*enabled*) y flag de testing (*use_emulator*).

Además, sólo si la aplicación utiliza el *solver GAMS*, se embeben datos sobre el archivo fuente (*file_resource_GAMS_source*), archivo Excel de entrada por defecto (file_resource_input_sample) y archivo Excel de salida de ejemplo (file_resource_output_sample). Por cada uno de los archivos, se requiere conocer nombre del *bucket* donde se encuentran, el nombre de archivo (*filename*), el tamaño en bytes (*size*), el nombre original del archivo (*src_filename*), la ruta de acceso absoluto (*object_id*) y la fecha de creación (*creation_date*).

5.3.2. Colección app0.role

En los documentos de la colección *role* se tiene la estructura ilustrada en la Figura 15. Los campos son: identificador (id), nombre (*name*), breve descripción (*description*), así también como una bandera de habilitación (*enabled*) y una copia del nombre de la aplicación (*application*). El campo *application* es utilizado únicamente para asignarle una "etiqueta" al rol, sin embargo no tiene efecto en el sistema de roles-permisos.

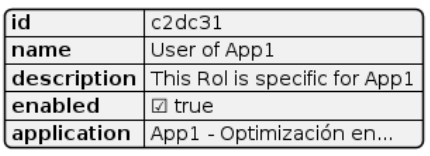


Figura 15. *Ejemplo de documento en puedb.app0.role.*

5.3.3. Colección app0.user

En la Figura 16 se muestran los campos almacenados para los usuarios creados por los administradores desde la interfaz gráfica. Estos incluyen identificador (id), primer nombre (*firstname*), apellido (*surname*), nombre de usuario (*username*), email, número de teléfono (*phone_number*) y flag de habilitación (*enabled*). Todos los atributos son opcionales excepto por *id* y *username*.

id	b88aa0
firstname	Mateo
surname	Toniolo
username	mtoniolo
email	mateojustotoniolo@gmail.com
phone_number	3427594619
enabled	☑ true

Figura 16. *Ejemplo de documento en puedb.app0.user.*

5.3.4. Colección app0.user_role

En esta colección se almacenan las relaciones 1-N de rol-usuario, indicando identificador (*id*), nombre de usuario (*username*) y rol (*role*). Todos los atributos son obligatorios (Figura 17).

id	9b88b1f
username	mtoniolo
role	User of App2

Figura 17. *Ejemplo de documento en puedb.app0.user_role.*

5.3.5. Colección app0.job

Los documentos de esta colección almacenan datos sobre cada una de las ejecuciones de las aplicaciones, tanto las exitosas como las fallidas.

Por cada ejecución exitosa se almacenan, atributos básicos como identificador (id), fecha de creación, de inicio y de finalización, código de estado (status), entre otros. Además, se embeben las entradas ingresadas por el usuario (*model_input*) y las salidas generadas por el *solver* (*model_output*) en formato JSON; así como información sobre archivos relacionados a *job* en particular (*result_files*) como archivos de *log* generados por el API, archivos de solución crudos con extensión *.lst* para *GAMS* o *.sol* para *Gurobi*, entre otros.

En la Figura 18 se presenta un ejemplo de la representación física de la entidad *job* para la App1. Nótese que la estructura de los campos *model_input*, *model_output* y *result_files* son específicos a cada aplicación. Por esta razón, en el Anexo II se incluyen el resto de los diagramas de diseño físico para cada una de las aplicaciones faltantes en las figuras 65, 66 y 67.

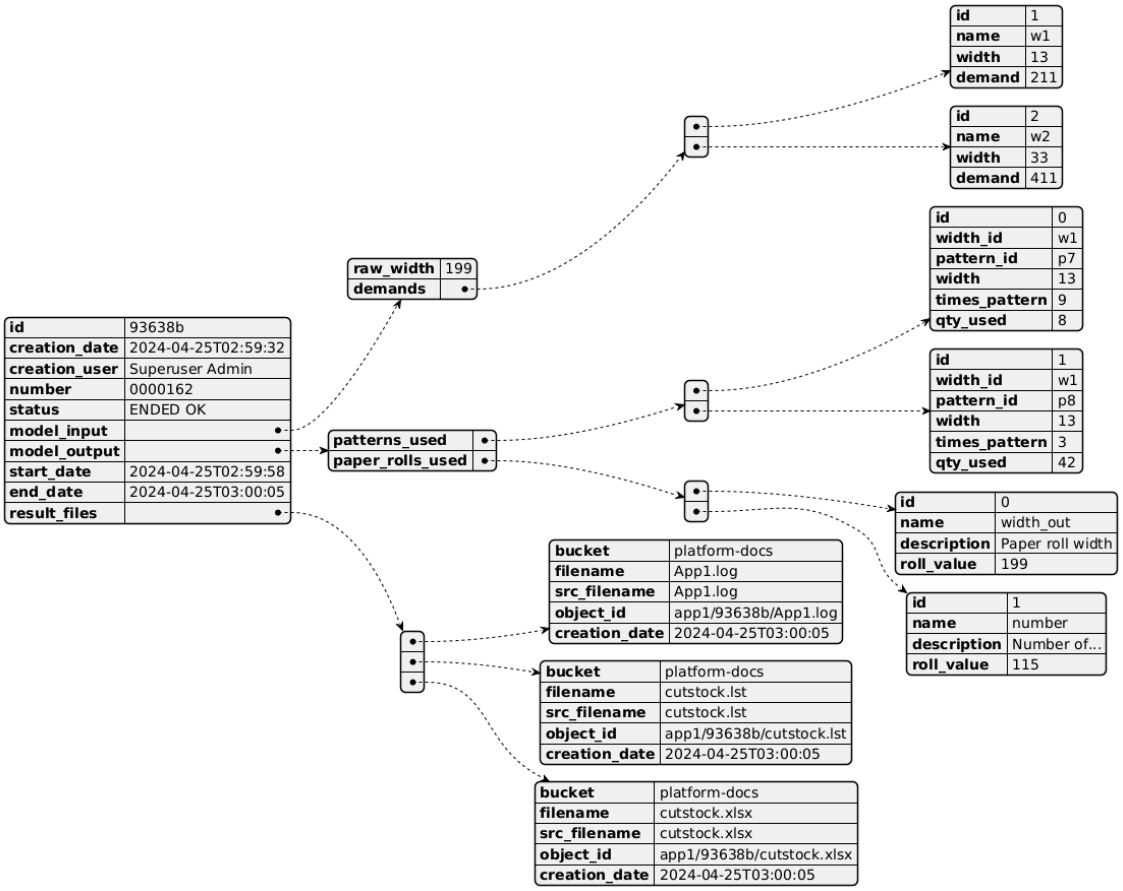


Figura 18. *Ejemplo de documento en puedb.app1.job.*

5.3.5. Colección app0.job_log

En esta colección se almacena cada *log* impreso por el API durante la ejecución del *job* N, incluyendo documentos para mensajes de error, mensajes de éxito y mensajes de carga.

En la Figura 19 puede verse que por cada documento se indica: identificador (*id*), identificador de ejecución (*job*_id), fecha (*line_date*) y contenido del *log* (*text*).

id	9362cf
job_id	9362b8
line_date	2024-04-25T02:29:52
text	cutstock.lst uploaded to s3 repo

Figura 19. *Ejemplo de documento en puedb.app1.job_log.*

5.3.6. Colección app0.scenario

La estructuras de estos documentos se puede observar en la Figura 20. Conceptualmente, un escenario no es más que un *model_job* al que se le asigna un nombre (*name*). Por ello, los documentos de la colección *app{N}.scenario* tienen embebida una copia exacta del *model_job* al que hacen referencia. A fines prácticos, en la Figura 20 se sintetiza la estructura completa de la ejecución en *model_job_copy*.

Otros campos presentes en los documentos son: el identificador (*id*), la fecha de creación (*creation_date*), el usuario creador (*creation_user*) y el código de estado (*status*).

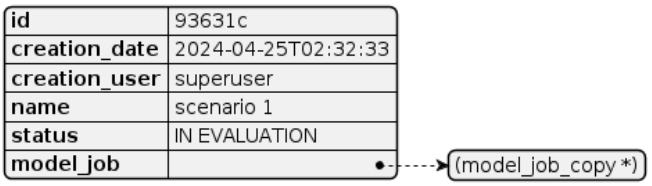


Figura 20. *Ejemplo de documento en puedb.app1.scenario.*

6. Implementación

Este capítulo detalla los aspectos involucrados en la etapa de implementación de las aplicaciones que conforman la plataforma UE-INGAR. Se define cómo se abordaron cuestiones funcionales (requerimientos funcionales definidos) y no funcionales (seguridad y flujo de ejecución de aplicaciones).

6.1. Desarrollo según requerimientos

Esta sección pretende ilustrar la forma en que las funcionalidades desarrolladas como parte de la plataforma satisfacen los requerimientos definidos en la sección “Gestión de Proyecto”. Las imágenes presentadas a continuación muestran cómo el usuario final navega dentro del sitio de la plataforma, realizando acciones que lo llevan a ejecutar de forma exitosa las funciones que se espera el producto de *software* ejecute.

6.1.1. R1. Administración de usuarios y autenticación

En el siguiente conjunto de imágenes se muestra la interfaz de usuario correspondiente a la administración de usuarios y autenticación dentro de la plataforma.

Estas capturas de pantalla ilustran la variedad de funcionalidades relacionadas con la autenticación (Figura 21 y Figura 22) y gestión de usuarios (Figura 23), incluyendo el proceso de inicio de sesión (Figura 22), la asignación de aplicaciones (Figura 22 y Figura 24), la administración de perfiles de usuario (Figura 24) y la creación (Fig.25), modificación (Fig.24) y visualización (Fig.26) de usuarios por parte de los administradores generales.

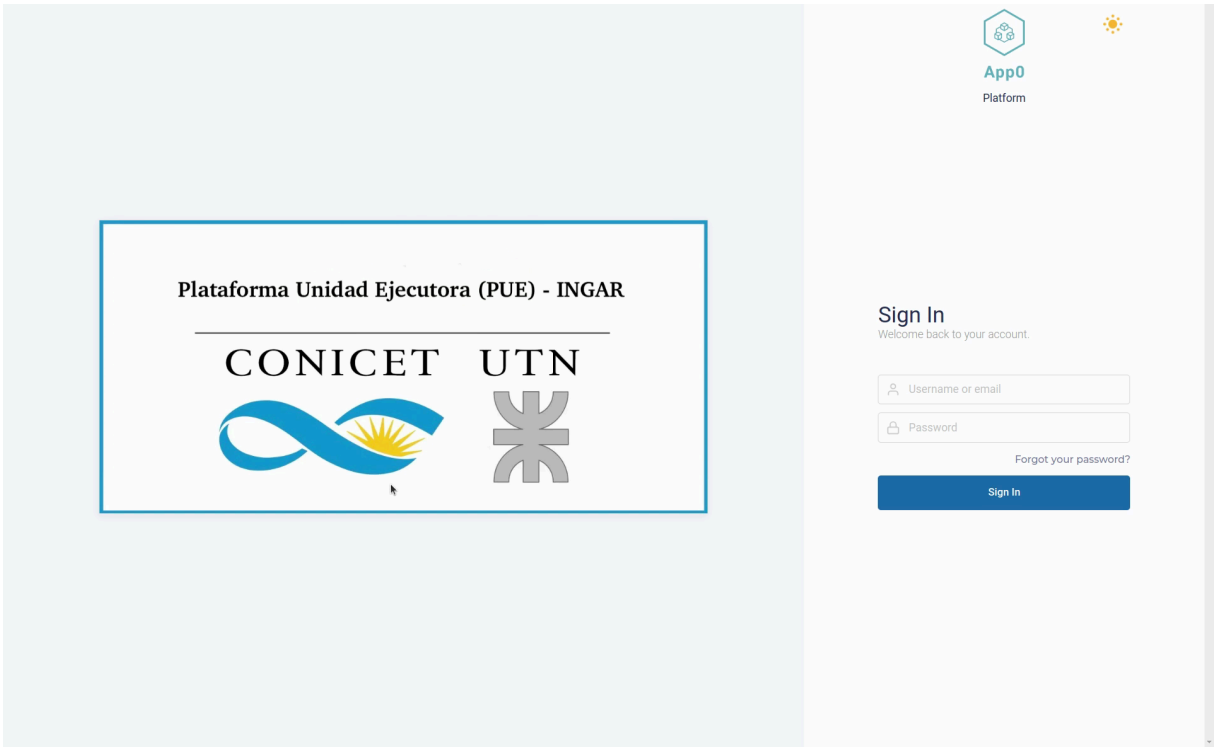


Figura 21. *Pantalla de Login.*

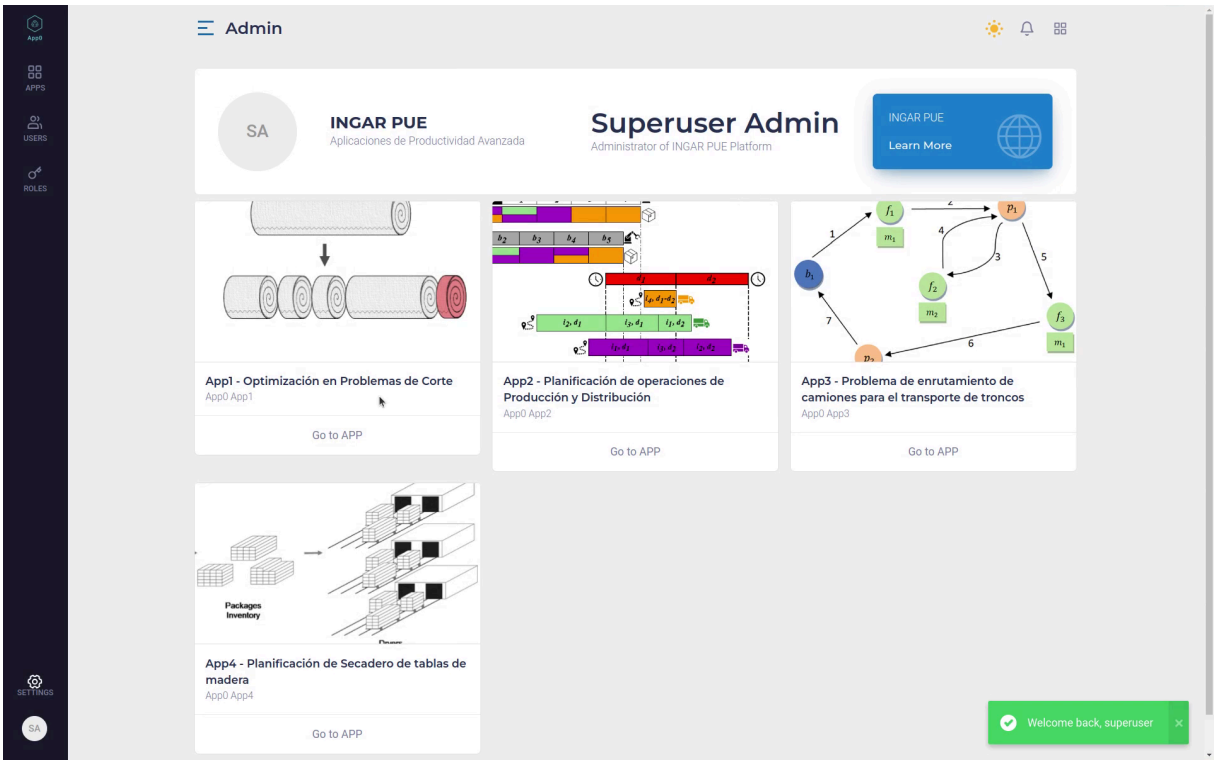


Figura 22. *Pantalla de inicio para el usuario Superuser.*

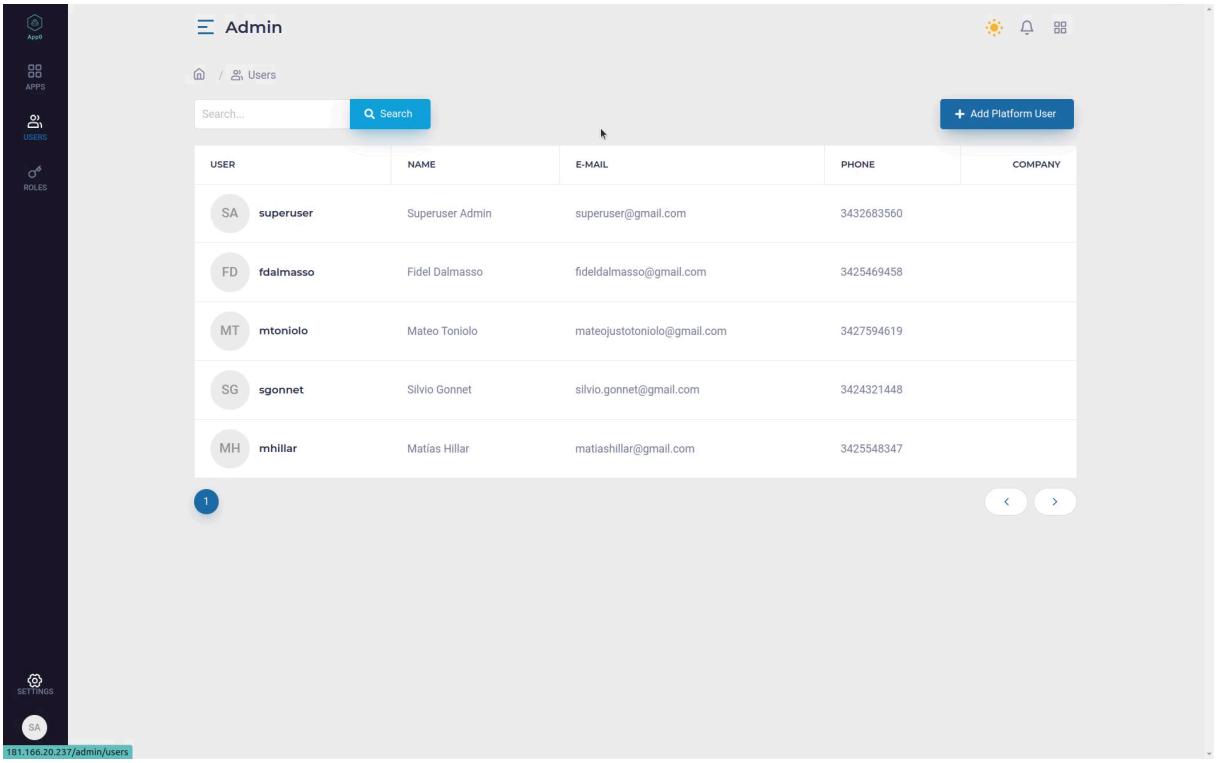


Figura 23. *Pantalla de menú de usuarios.*

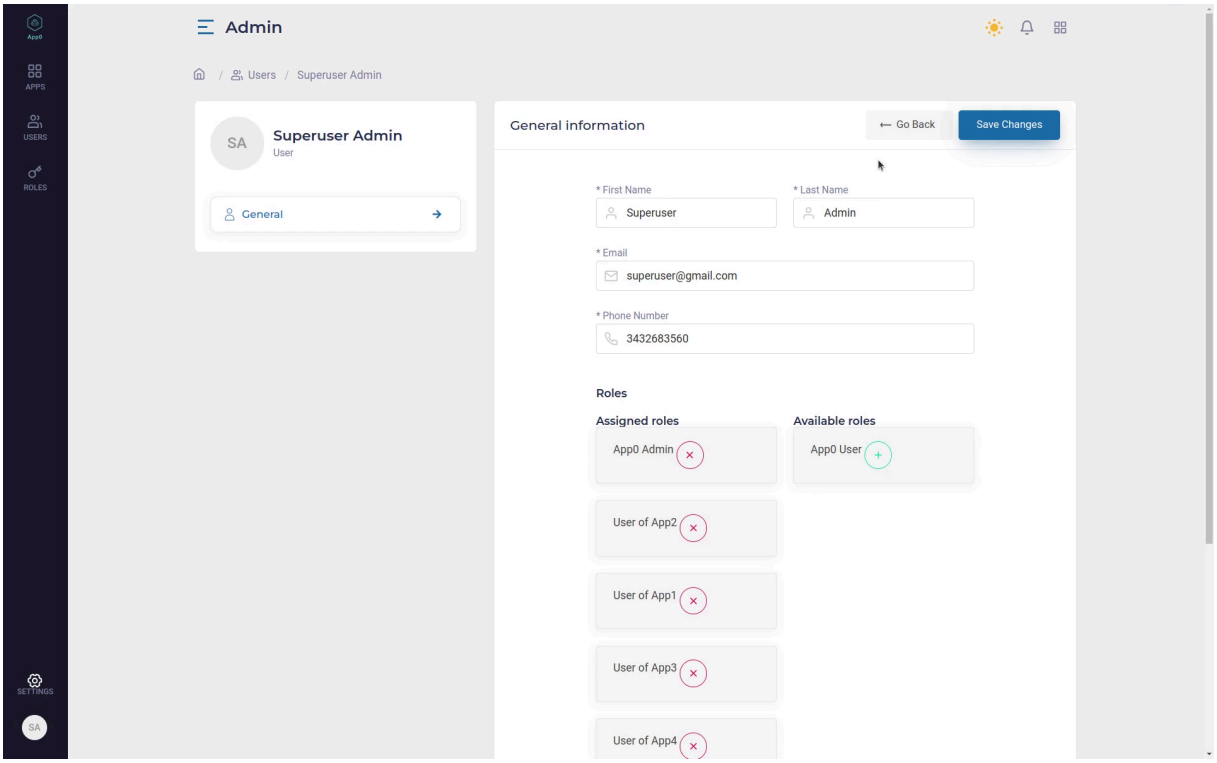


Figura 24. *Pantalla de configuración de perfil de usuario específico.*

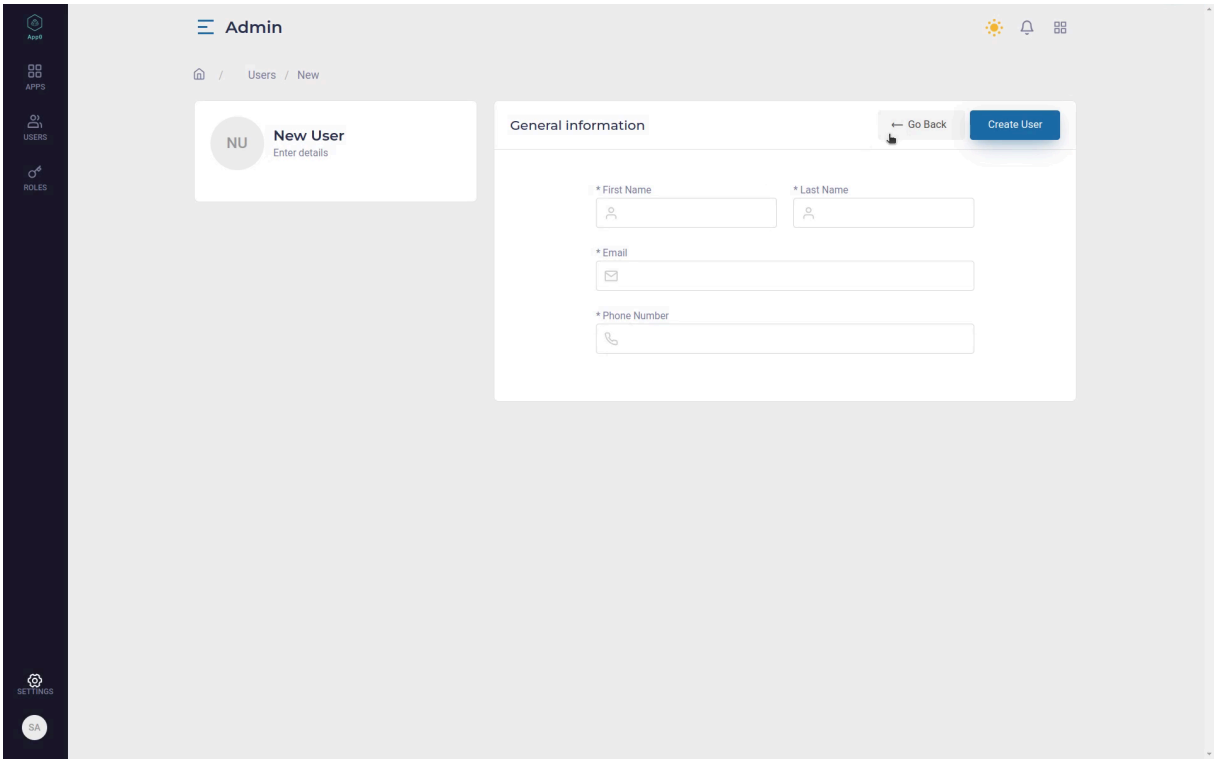


Figura 25. *Pantalla de creación de un nuevo usuario.*

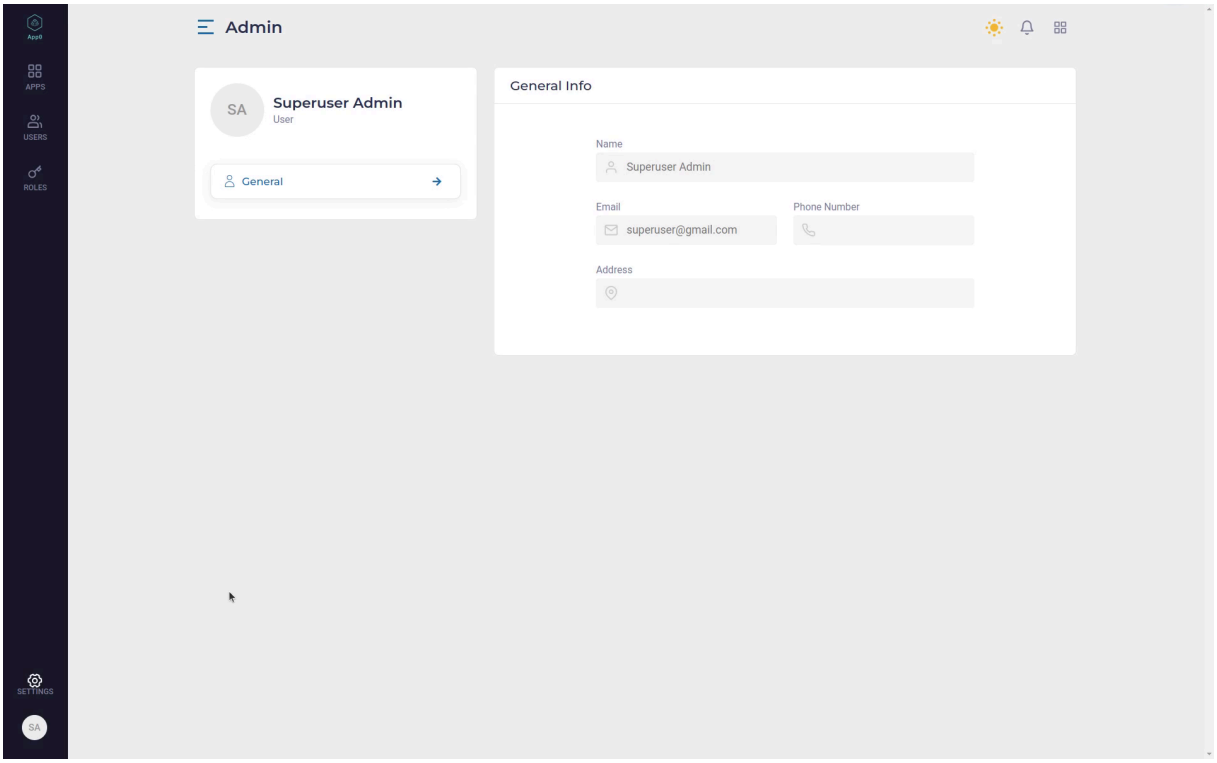


Figura 26. *Pantalla de visualización de perfil de usuario logueado.*

6.1.2. R2. Administración de aplicaciones

Estas capturas de pantalla ilustran la variedad de funcionalidades relacionadas con el listado de apps creadas (Figura 27), la creación de una nueva app (Figura 28) y la modificación de una app existente (Figura 29).

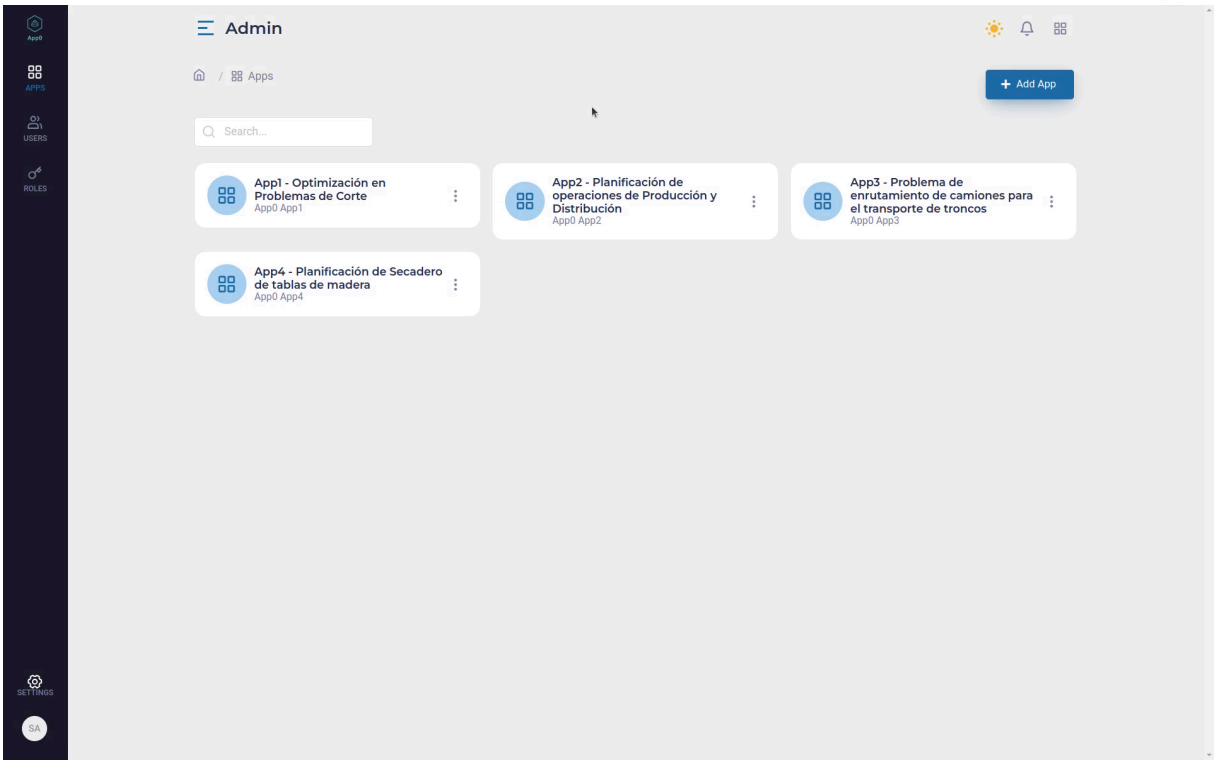


Figura 27. *Pantalla de listado de aplicaciones.*

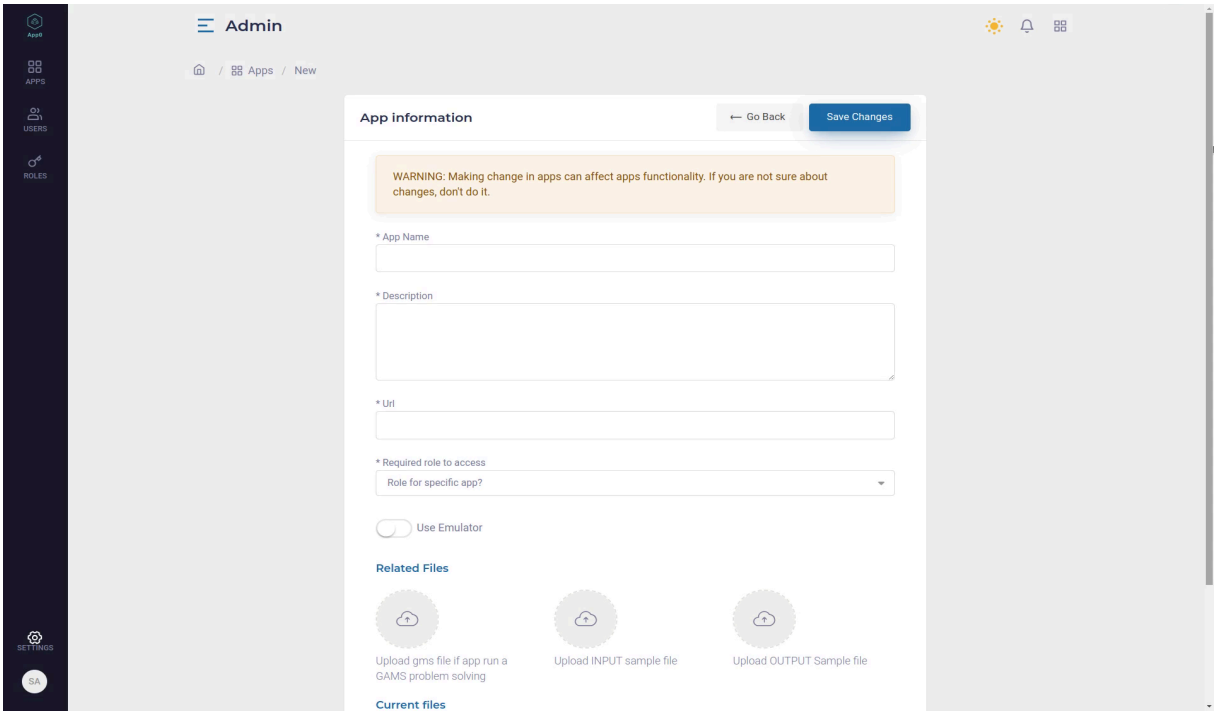


Figura 28. *Pantalla de configuración de una nueva aplicación.*

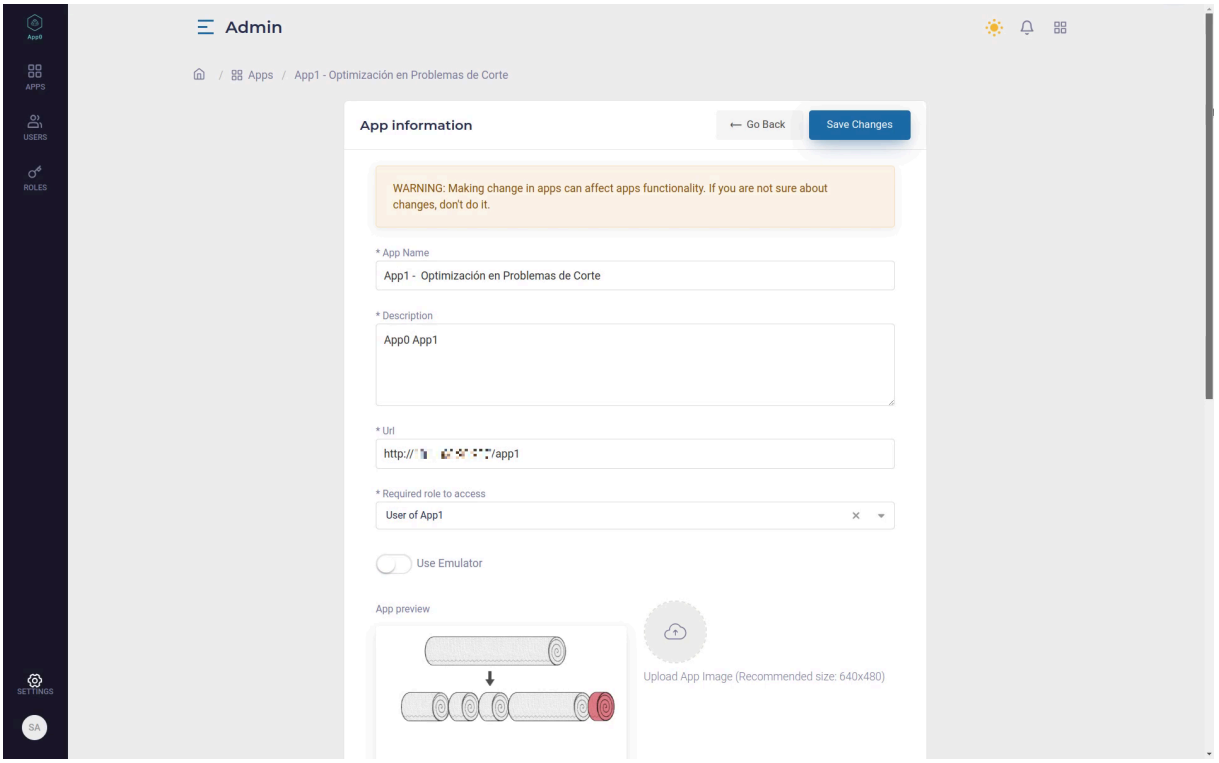


Figura 29. *Pantalla de configuración de una aplicación.*

6.1.3. R3. Lectura de datos de entrada

El conjunto de esta información conformará un escenario de ejecución, disponible para ser utilizado posteriormente para su visualización y comparación con otros escenarios.

6.1.4. R4. Ejecución de una aplicación

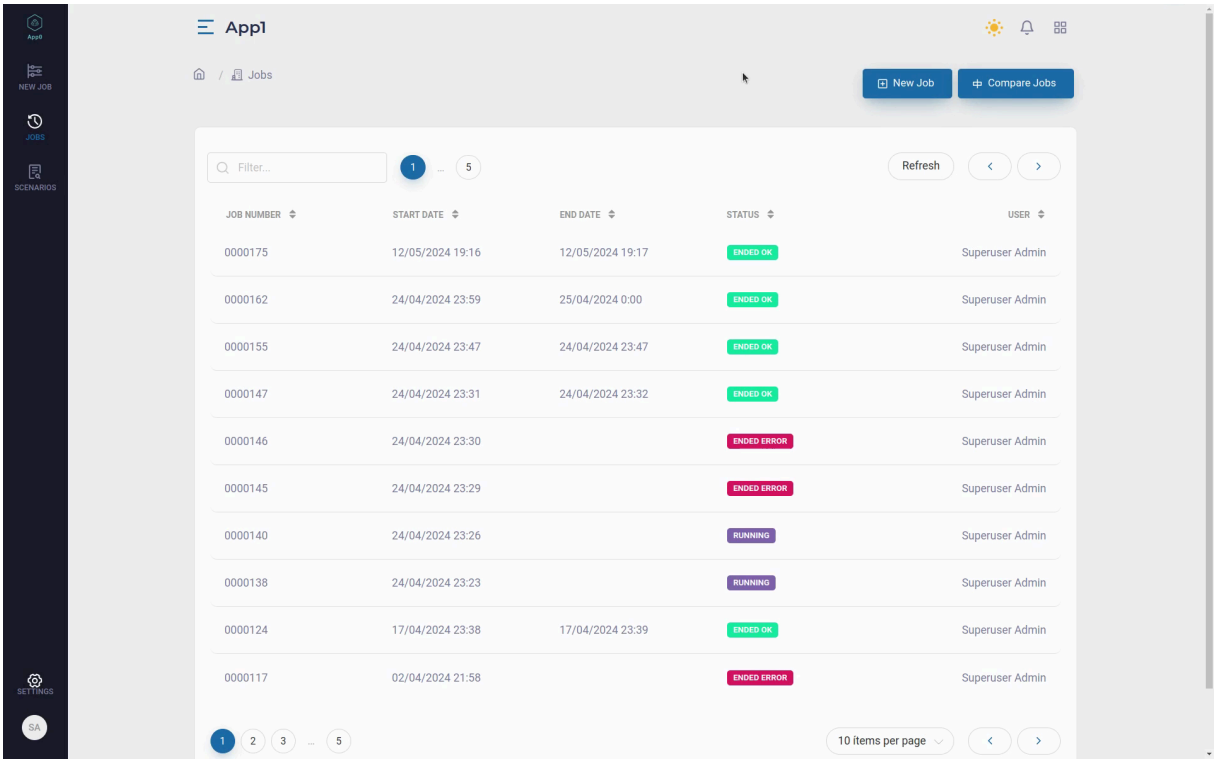


Figura 30. *Pantalla de listado de jobs.*

Llamamos *job* a la ejecución de un modelo, en el listado que se muestra en la Figura 30 cada entrada de la lista es un *job*, esto permite que se pueda acceder a cada uno, es decir, visualizar las entradas junto con los resultados en caso de éxito o el error en caso de haber fallado. Para evitar imágenes duplicadas al final de este trabajo se explican las apps de manera detallada, allí se puede observar como es la salida de cada ejecución (Figura 43).

6.1.5. R5. Visualización de resultados

Como se explica en la sección anterior, las imágenes de las salidas de ejecución se pueden encontrar en los apartados de cada App al final de este informe.

6.1.6. R6. Gestión de escenarios

A continuación se usan dos escenarios correspondientes a la App4 como ejemplos. En la Figura 31 se muestra una lista de escenarios guardados y luego los resultados de dos escenarios seleccionados se comparan en la Figura 32.a y 32.b.

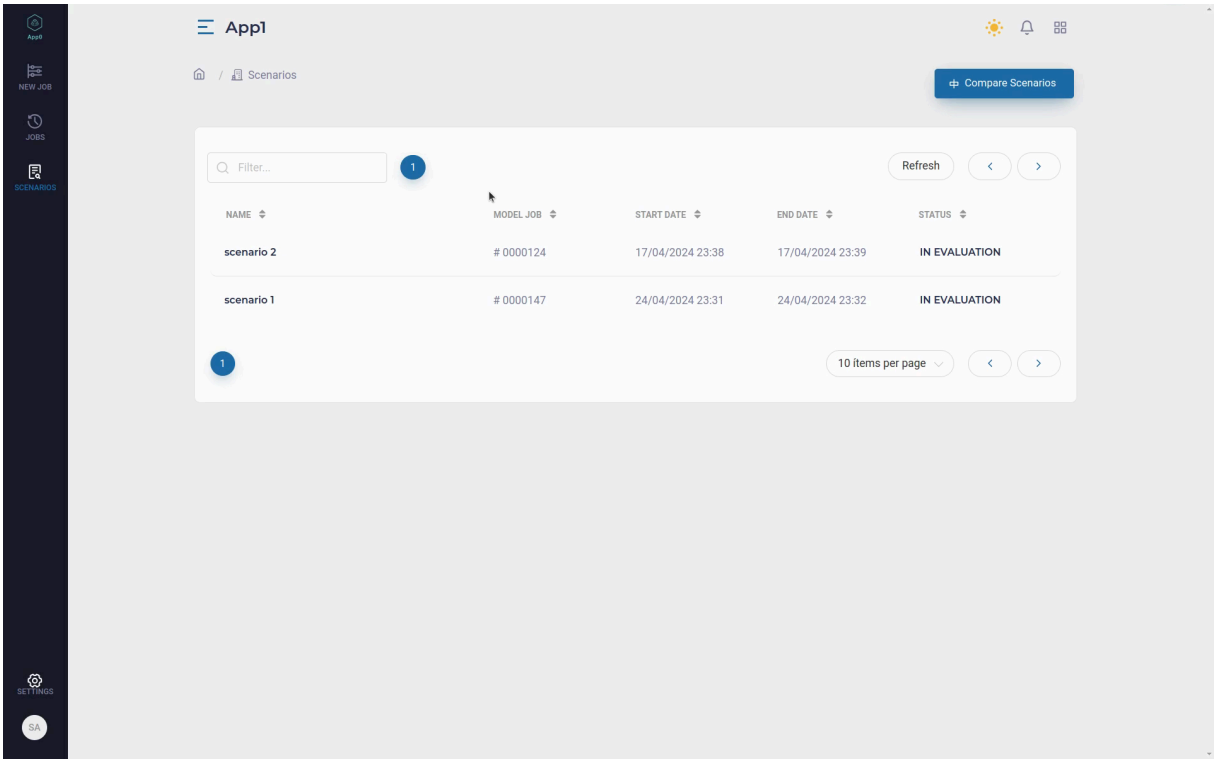


Figura 31. *Pantalla de listado de Escenarios.*

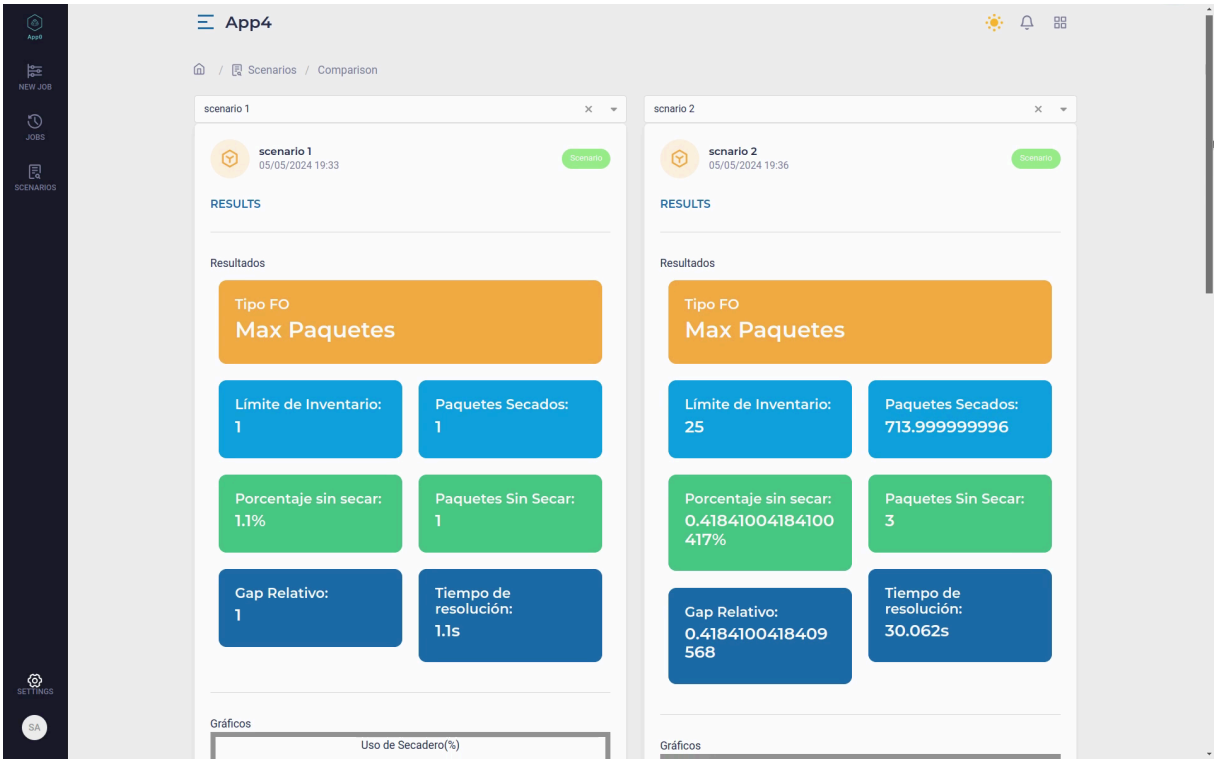


Figura 32.a. *Pantalla de Comparación de Escenarios.*

Figura 32.b. *Pantalla de Comparación de Escenarios.*

6.2. Ejecución de pruebas y testing

En esta sección, se detalla el proceso de testing implementado para evaluar el desarrollo de la interfaz web. Se describen las metodologías, herramientas y técnicas utilizadas en las pruebas de unidad, integración y aceptación, destacando cómo cada fase del *testing* contribuyó a la identificación y corrección de errores, así como a la validación de las funcionalidades implementadas y definidas en los casos de uso.

6.2.1. Prueba de unidad

Se decidió implementar un plan de testeo unitario básico que cada desarrollador replicó en su aplicación. Esta decisión se tomó para ahorrar tiempo y recursos, ya que esto permite mantener una estructura consistente en las pruebas a pesar de tener múltiples apps. Cada desarrollador adaptó el plan según las necesidades específicas de su módulo, asegurando así una evaluación completa de toda la plataforma desarrollada.

El plan de prueba unitario se basa en el flujo de acciones definido en la Figura 33.

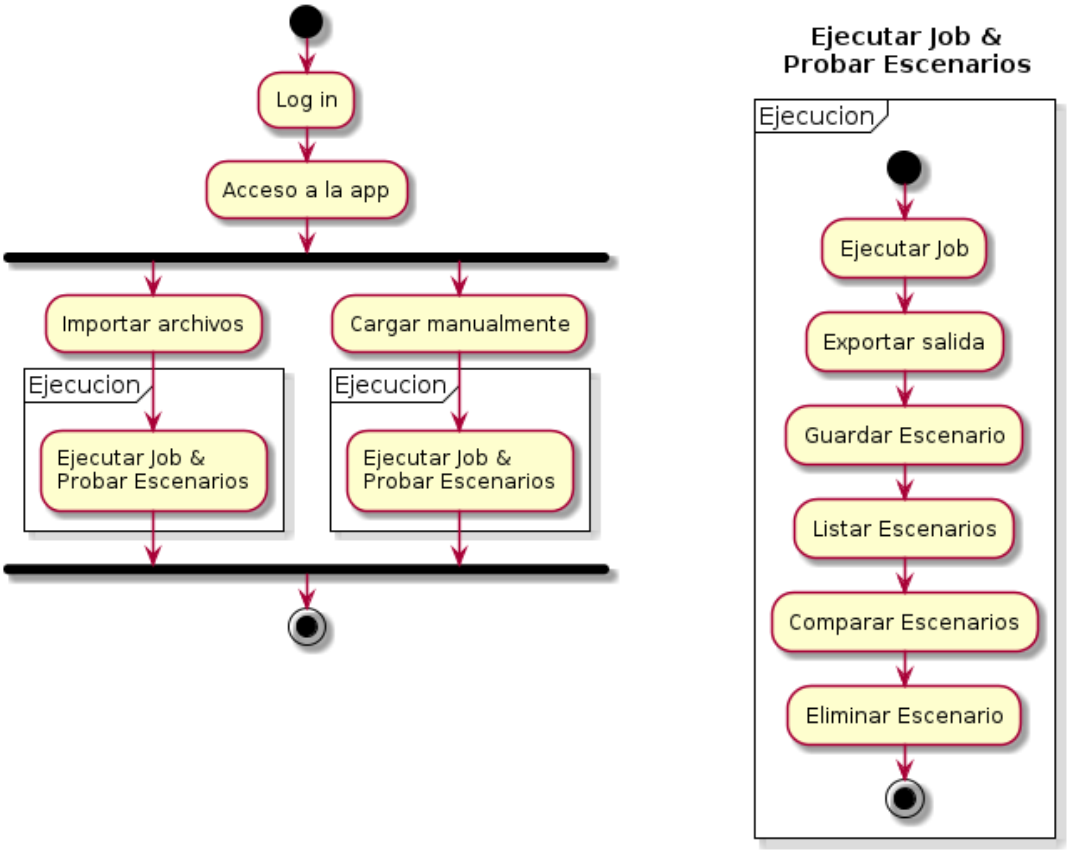


Figura 33. *Flujo de acciones de plan de prueba unitario.*

En la Figura 33 se pueden observar todas las funciones a probar por cada uno de los desarrolladores en sus respectivas apps. Cabe destacar que este tipo de pruebas se realizan contemplando el rol del usuario final, ya que son ellos quienes más contacto tendrán con la ejecución de aplicaciones, más adelante se explica cómo se prueban las funciones de usuario administrador.

6.2.2. Prueba de Usuario en Vivo

Una vez finalizadas las pruebas unitarias se decidió realizar un encuentro con los usuarios finales (investigadores del INGAR) para realizar al mismo tiempo pruebas de usuario en vivo y validar los modelos de las apps. Este enfoque permite una retroalimentación inmediata del usuario sobre el software mientras el desarrollador está presente para observar y tomar nota de cualquier problema o sugerencia que surja durante la sesión de prueba. Esta interacción directa entre el usuario y el desarrollador puede ser muy útil para identificar rápidamente áreas de mejora y realizar ajustes en el software en tiempo real.

6.2.3. Prueba de Integración

El objetivo principal de las pruebas de integración es detectar errores en la interacción entre los componentes del sistema cuando se combinan. Esto es especialmente importante porque los componentes individuales pueden funcionar correctamente por separado, pero pueden surgir

problemas cuando se integran. En este caso, el objetivo es probar la integración de las apps desarrolladas en el núcleo de la aplicación web, también llamado app0.

Previo a las pruebas de integración es necesario probar ciertas funcionalidades que dependen de la existencia de las apps, en la Figura 34 se muestra el flujo de las mismas (cabe destacar que todas deben hacerse desde el rol de Usuario Administrador).

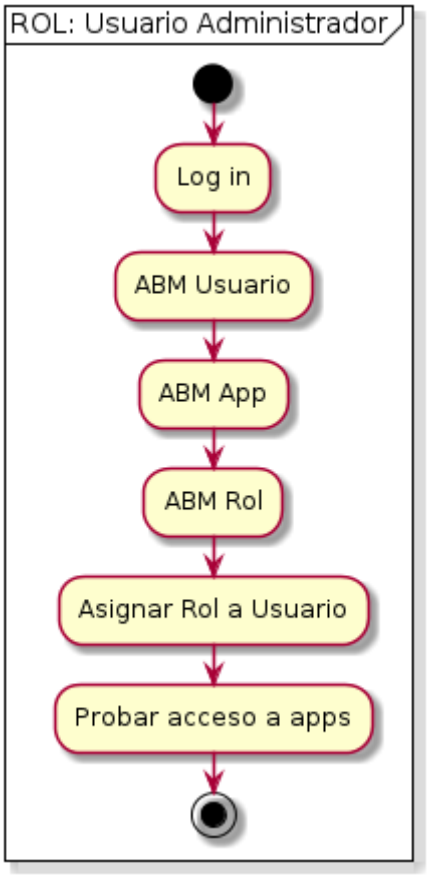


Figura 34. *Pruebas unitarias previas a las pruebas de integración.*

El Alta, Baja y Modificación de Usuarios, Apps y Roles, así como también la asignación de roles a usuarios, se realizaron con un Usuario con permisos de administrador. Para corroborar la integración exitosa de las apps se utilizaron usuarios de prueba que simulan los usuarios finales de las mismas.

6.3. Seguridad

Como es mencionado anteriormente, no existió un proceso de elicitación de requerimientos no funcionales en la etapa de análisis del proyecto. Considerando a la seguridad como un atributo de calidad que se corresponde con aspectos no funcionales de un sistema, no se proveyeron entonces requerimientos específicos de seguridad. De esta forma, los aspectos relacionados a la seguridad de la plataforma fueron abordados según el criterio del equipo de desarrollo.

Cuando se habla de seguridad informática, es importante establecer cuál es la superficie de ataque del producto que se desarrolla. En el caso presente, la plataforma PUE fue pensada para desplegarse en el ámbito interno de la institución cliente, por lo que la plataforma es eventualmente susceptible solamente a ataques internos. Luego al no requerirse la exposición de los puntos de acceso de la plataforma a la Internet, la necesidad de implementar tácticas de seguridad se reduce considerablemente, en comparación a un producto de software presentado al acceso público externo.

Se determinó entonces, la implementación de un esquema de control de acceso basado en roles (*RBAC*), con el objetivo de evitar el escalamiento vertical de privilegios.

En este esquema, se definen roles en función de los tipos de usuarios que tendrán acceso a las funcionalidades de la plataforma de acuerdo con las políticas definidas por el cliente.

El esquema de *RBAC* es particularmente útil para arquitecturas como la que se presenta en este trabajo, donde varios tipos de usuarios comparten acceso a un mismo sistema. Los permisos se gestionan a nivel de rol en lugar de a nivel individual, lo que facilita la auditoría y la modificación de reglas.

La estrategia planteada consiste en crear un rol por cada aplicación existente. Además, por defecto, cada usuario administrador recibe el rol de "Usuario Administrador", mientras que cada usuario final recibe el rol de "Acceso Inicial a la Plataforma" (rol base que le permite acceder a la plataforma web).

Desde ahí, cada usuario creado tendrá el rol base llamado "app0" por defecto, y se le debe asignar un rol adicional por cada aplicación a la que se le permita acceder. Esta estructura garantiza un acceso controlado y seguro a las distintas funcionalidades de la plataforma, adaptándose a las necesidades específicas de cada usuario dentro de la organización.

En la Figura 35 se presenta un ejemplo mostrando cómo funciona el esquema *RBAC* en la plataforma PUE, para el mismo se crean cuatro usuarios (Federico, Mateo, Matías y Fidel). En este ejemplo, Federico tiene asignado el rol de administrador y acceso a todas las apps creadas (App1, App2 y App3), mientras que el resto de usuarios solo pueden acceder al conjunto de aplicaciones asignadas.

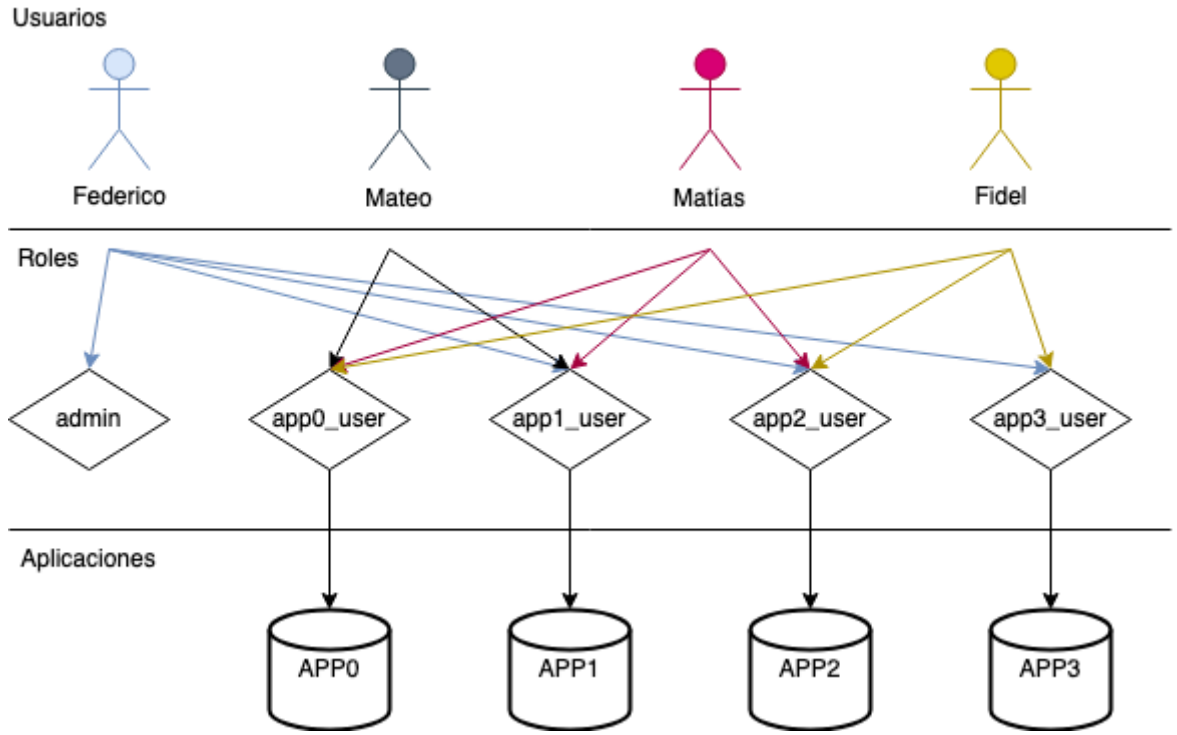


Figura 35. *Ejemplo de esquema RBAC.*

A continuación, se presentan las capturas de pantalla de la interfaz de usuario que ilustran los casos de uso relacionados con la gestión de roles dentro del esquema de control de acceso basado en roles (*RBAC*). En la Figura 36 se presenta el listado de roles creados, en la Figura 37 se muestra la pantalla de creación de un nuevo rol y en las Figuras 38 y 39 se observa la configuración de "*rol administrador*" y "*rol de acceso app1*" respectivamente.

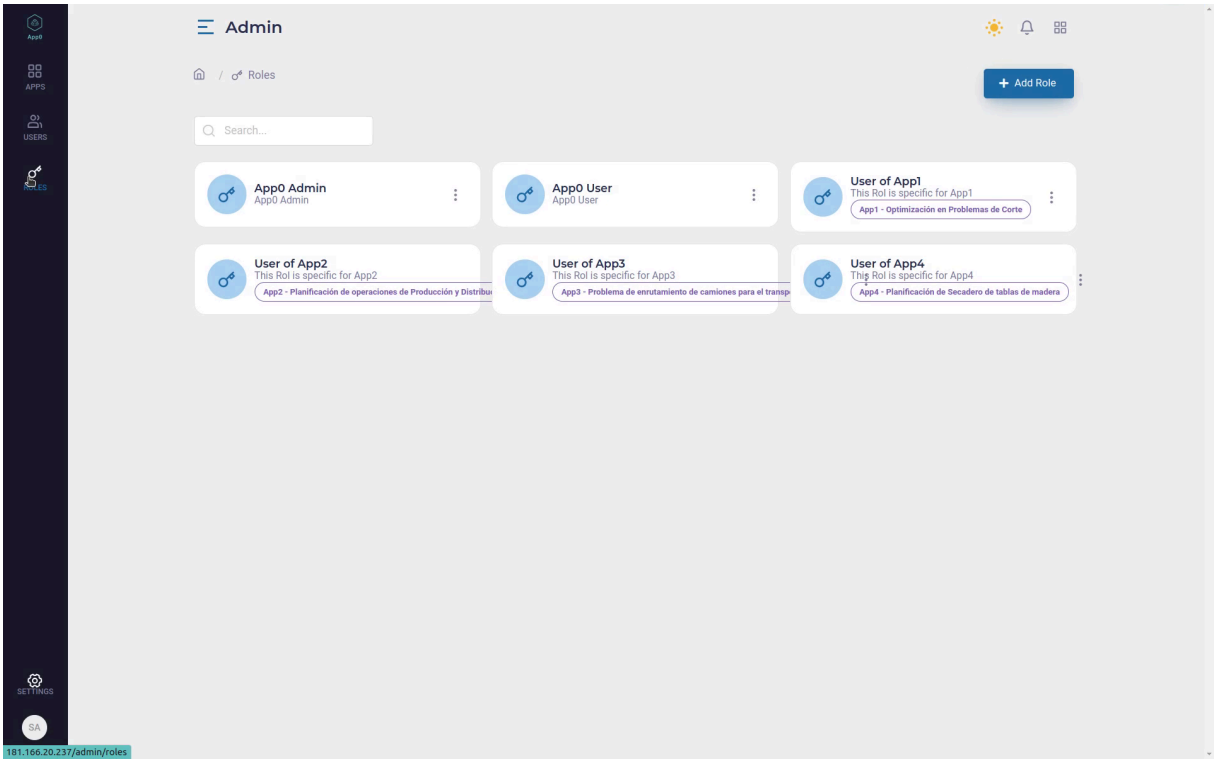


Figura 36. *Pantalla de gestión de roles.*

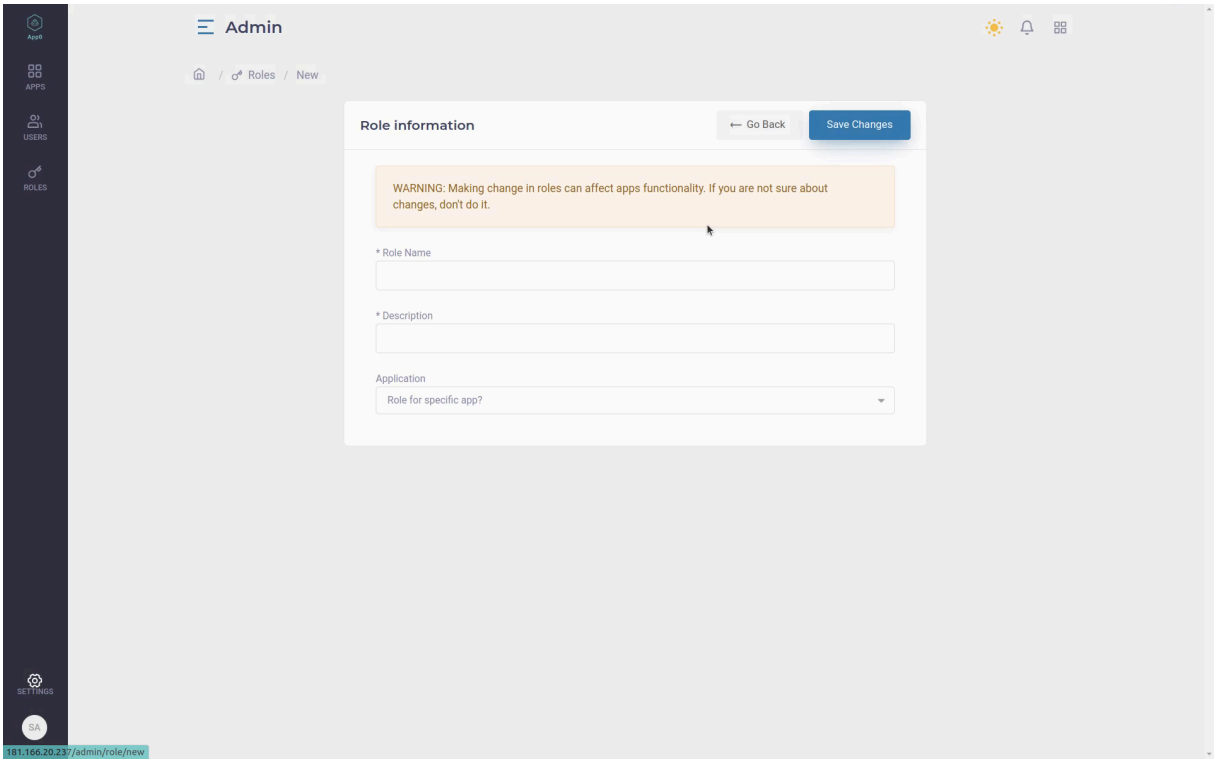


Figura 37. *Pantalla de creación de un rol nuevo.*

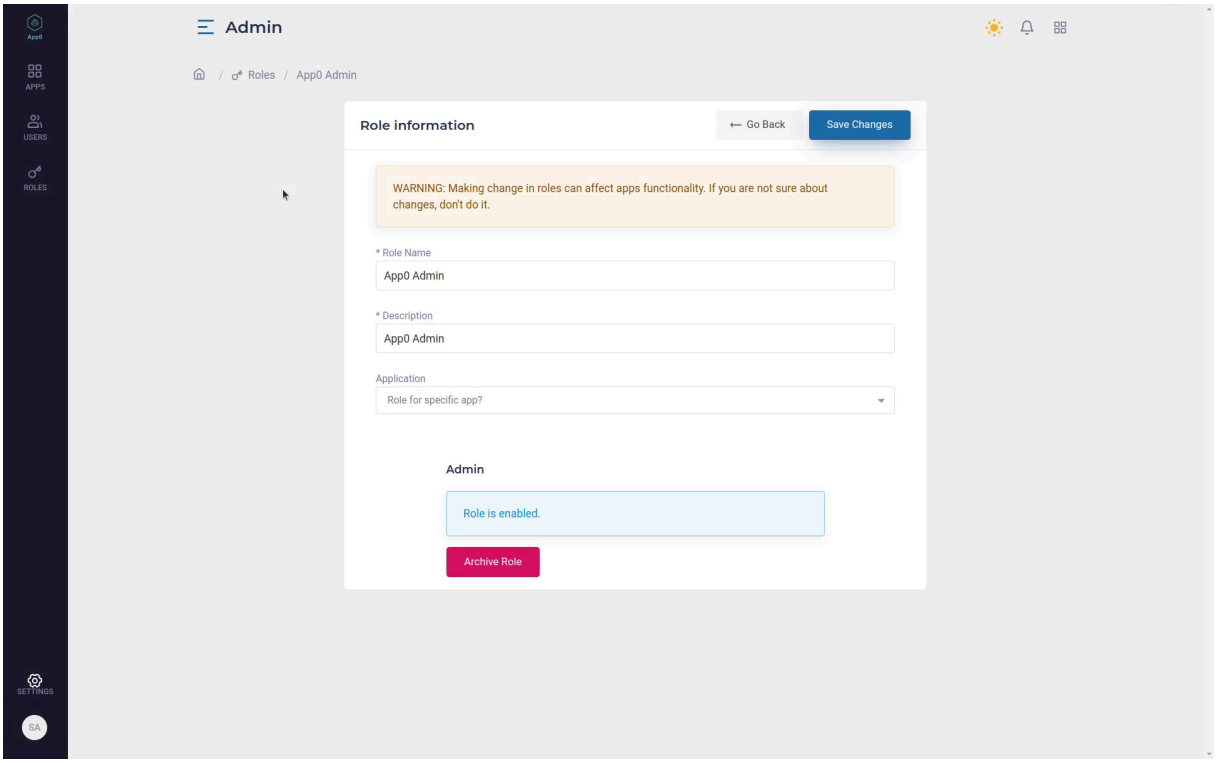


Figura 38. *Pantalla de configuración de rol para App0/Admin.*

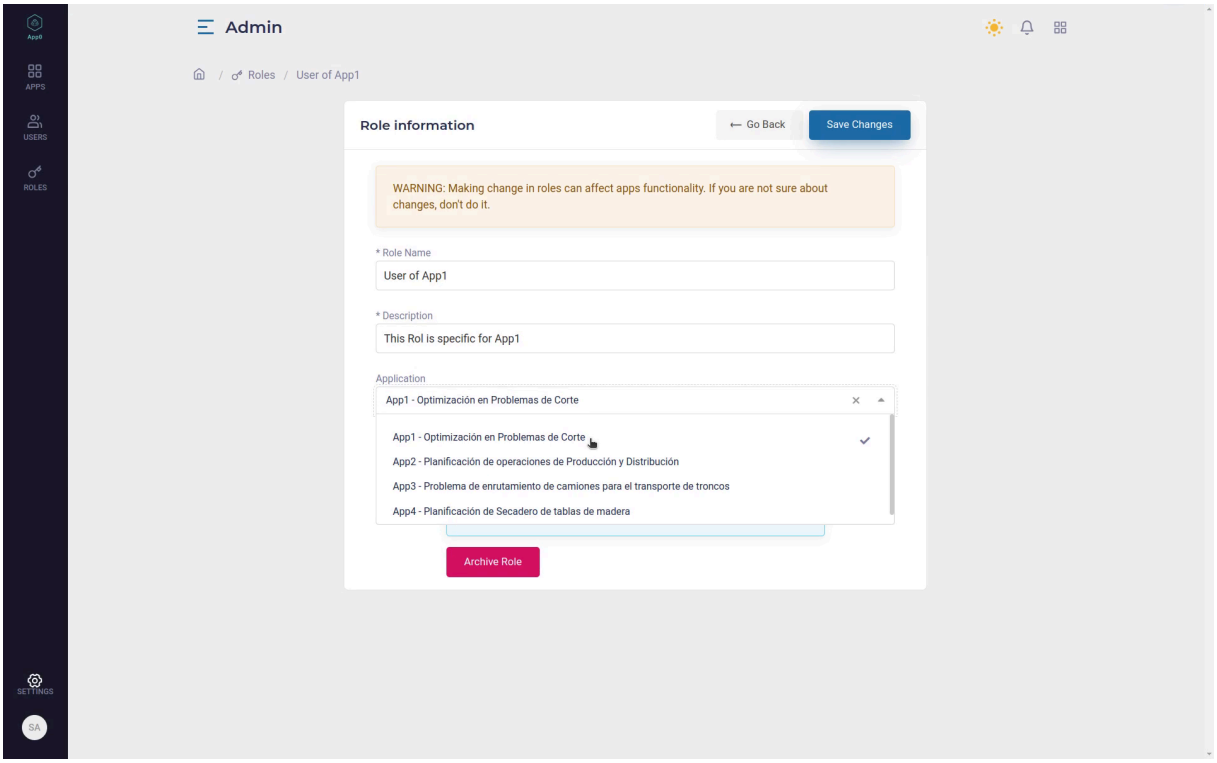


Figura 39. *Pantalla de configuración de rol para App1.*

Estas capturas de pantalla muestran las interfaces de usuario diseñadas para facilitar la administración de roles dentro del sistema, permitiendo una gestión eficiente y segura de los permisos de acceso.

6.4. Flujo de ejecución de aplicación según *solver*

La plataforma PUE fue diseñado originalmente con la intención de ser solver agnóstico, permitiendo la implementación de aplicaciones que utilicen diversas herramientas de software de resolución matemática y optimización de modelos, de forma transparente.

Por el momento, la plataforma soporta la implementación de aplicaciones que utilicen una de las siguientes herramientas de modelado matemático:

- *Gurobi*
- *GAMS* (*General algebraic modeling system*)

A continuación se detallan las diferencias en el proceso de ejecución para cada tipo de aplicación.

6.4.1. Aplicación con solver Gurobi

En la Figura 40 se presenta un diagrama de secuencia que describe el funcionamiento de la plataforma al ejecutar una aplicación que emplea el *solver Gurobi* (*Gurobi Optimization*, 2020), incluyendo la interacción del usuario con la interfaz, el procesamiento interno, la resolución del *solver Gurobi* y el guardado y visualización de los resultados.

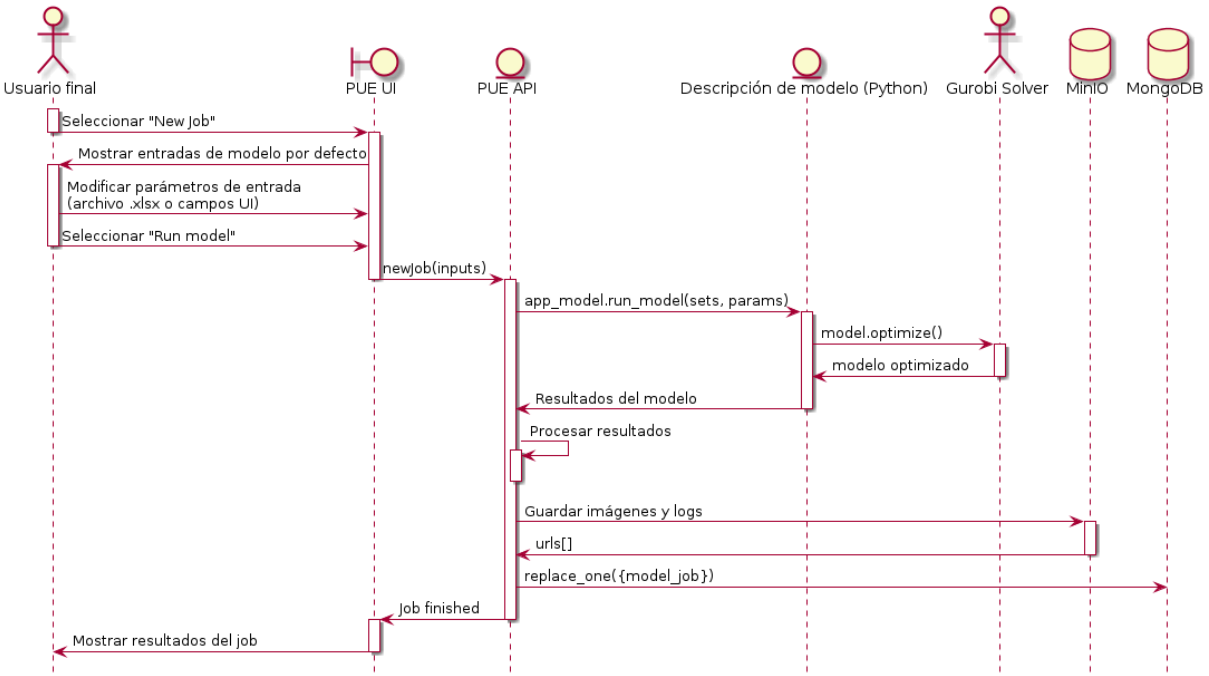


Figura 40. *Flujo ejecución de aplicación con solver Gurobi.*

6.4.2. Aplicación con solver GAMS

La Figura 41 describe la secuencia de interacciones para el caso de una aplicación que utiliza *solver GAMS*. A diferencia de la resolución por *solver Gurobi*, en este escenario se agrega *GAMS* API como un nuevo actor fundamental, intermediario entre PUE API y el *solver GAMS*. Dicho actor surge como respuesta a la restricción del *solver* para correr únicamente en dispositivos con sistema operativo Windows (*GAMS* Development Corporation, 2017).

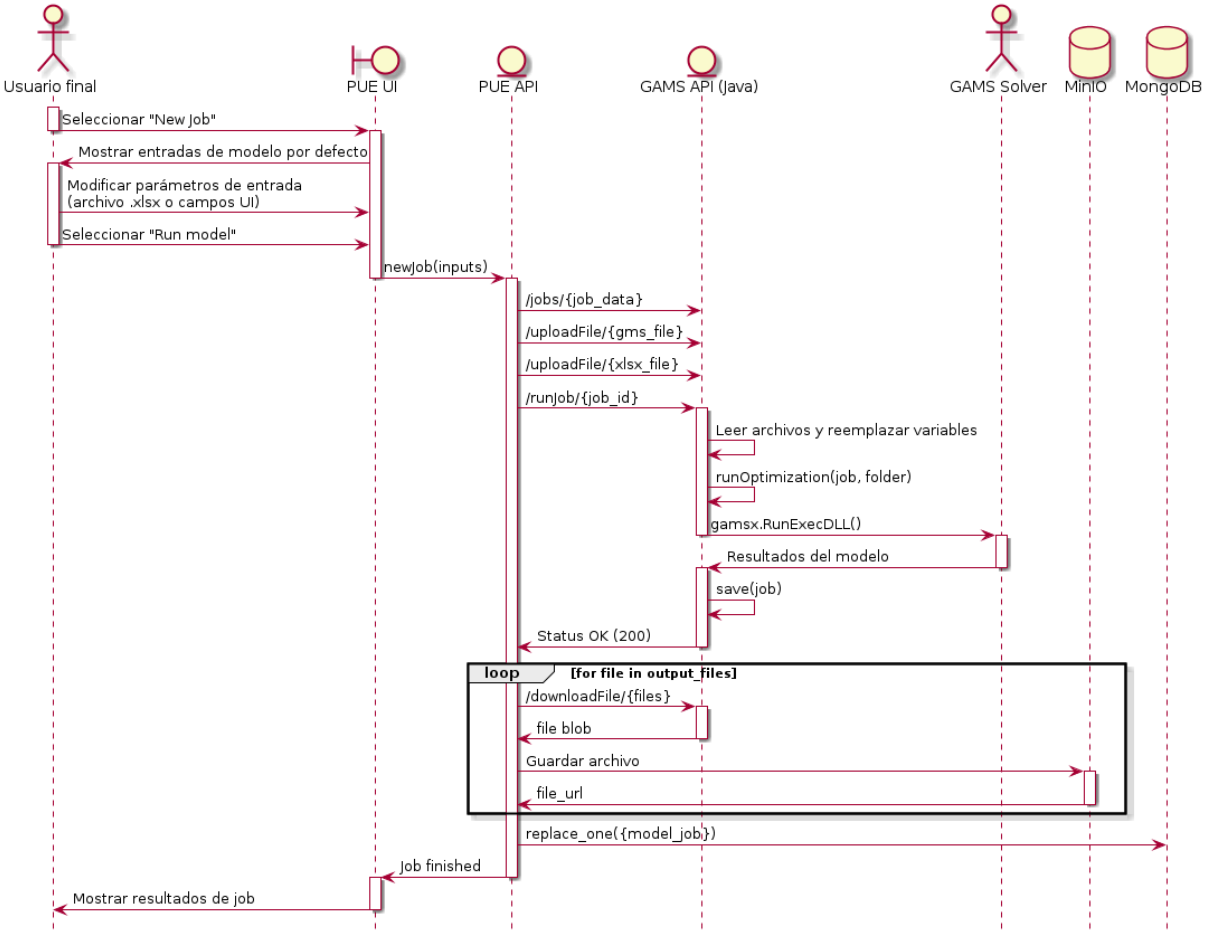


Figura 41. *Flujo ejecución de aplicación con solver GAMS.*

7. Aplicaciones Desarrolladas

7.1. App 1: "Optimización en Problemas de Corte"

7.1.1. Introducción

Esta aplicación se basa en un caso de ejemplo extraído de la galería de modelos de ejemplo de la aplicación *GAMS* Miro (https://miro.*GAMS*.com/). Se ha utilizado como prototipo inicial para el desarrollo y la integración de la primera aplicación en el conjunto de microservicios.

El propósito de esta app es proporcionar a los desarrolladores una oportunidad para experimentar con el proceso de desarrollo e integración de una aplicación web de microservicios. Acorde a la metodología propuesta, este sería el prototipo inicial.

7.1.2. Marco Teórico

El problema del material de corte tiene su origen en la industria papelera. Un fabricante tiene que satisfacer la demanda de rollos de papel de diferentes anchos. Para ello, se cortan rollos más pequeños a partir de rollos de papel anchos de tamaño estandarizado. Dependiendo del patrón, no siempre es posible utilizar todo el rollo de papel, por lo que habrá desperdicio. El objetivo es satisfacer la demanda minimizando el número de rollos de papel necesarios.

7.1.3. Uso de GAMS en el Proyecto

El uso de GAMS como solver en el desarrollo de aplicaciones requiere de tres tipos de archivos, como se observa en la Figura 28: el archivo GAMS (que contiene el código de resolución), un archivo de entrada y un archivo de salida. El solver GAMS lee las entradas desde un archivo Excel y escribe las salidas en otro archivo Excel. Esto significa que el solver GAMS recibe un archivo Excel con datos de entrada, los procesa según el modelo matemático definido y sobrescribe los resultados en un archivo de salida en formato Excel.

Cuando se crea una aplicación con solver GAMS en nuestra plataforma, se requiere que el usuario cargue tres archivos: el archivo de entrada, el archivo de salida y el archivo con la solución GAMS. Es importante destacar que el tipo de archivos utilizados como entrada y salida (en este caso Excel) es una decisión de los diseñadores del modelo matemático, quienes optaron por este formato por su compatibilidad y conveniencia operativa. Para este proyecto, tanto la App1 como la App4 (que utilizan GAMS) manejan archivos en formato Excel para sus operaciones.

Específicamente, el proceso se desarrolla de la siguiente manera:

- **Carga Inicial del Archivo de Entrada**
  Al crear la aplicación, el usuario carga un archivo Excel donde se definen los parámetros de entrada y sus valores. Esta acción se realiza una sola vez durante el ciclo de desarrollo de la aplicación.

- **Interfaz de Entrada**
  La interfaz gráfica permite a los usuarios ingresar datos, los cuales corresponden a las entradas del archivo Excel de entrada.

- **Sobrescritura del Archivo Excel de Entrada**
  Cada vez que el usuario ingresa los valores de entrada y ejecuta la aplicación, el primer paso es sobrescribir esos valores en el archivo Excel de entrada con la información proporcionada.

- **Envío de Archivos al Servidor GAMS**
  Una vez sobrescritos los datos de entrada, siempre se envían al servidor *GAMS* los tres archivos: el archivo de entrada actualizado, el archivo de salida y el archivo de solución *GAMS* para ser procesados.

- **Sobrescritura del Archivo Excel de Salida**
  Una vez finalizada la ejecución, los datos de salida se sobrescriben en el archivo de salida enviado y luego se devuelven para que la interfaz de salida tome los datos de allí y los muestre al usuario.

Esta estructura asegura que por cada entrada en el archivo Excel, exista una entrada asociada en la interfaz gráfica, lo que facilita el proceso de ingreso de datos y su posterior procesamiento por el solver GAMS. Este enfoque permite una integración efectiva entre la interfaz de usuario y el sistema de resolución de problemas de GAMS, garantizando que los datos ingresados se procesen correctamente y se obtengan los resultados esperados.

7.1.4. Entradas de la aplicación

Según lo mencionado en el apartado “7.1.3. Uso de *GAMS* en el proyecto”, a continuación se detallan los parámetros de entrada definidos en formato libro de Excel (*.xlsx*). Todos estos campos pueden verse reflejados en la interfaz desarrollada (Figura 42).

- *Raw width*: El ancho estandarizado del rollo de papel del que se cortan los patrones.
- *Demand*: El número de cada ancho que se demanda.

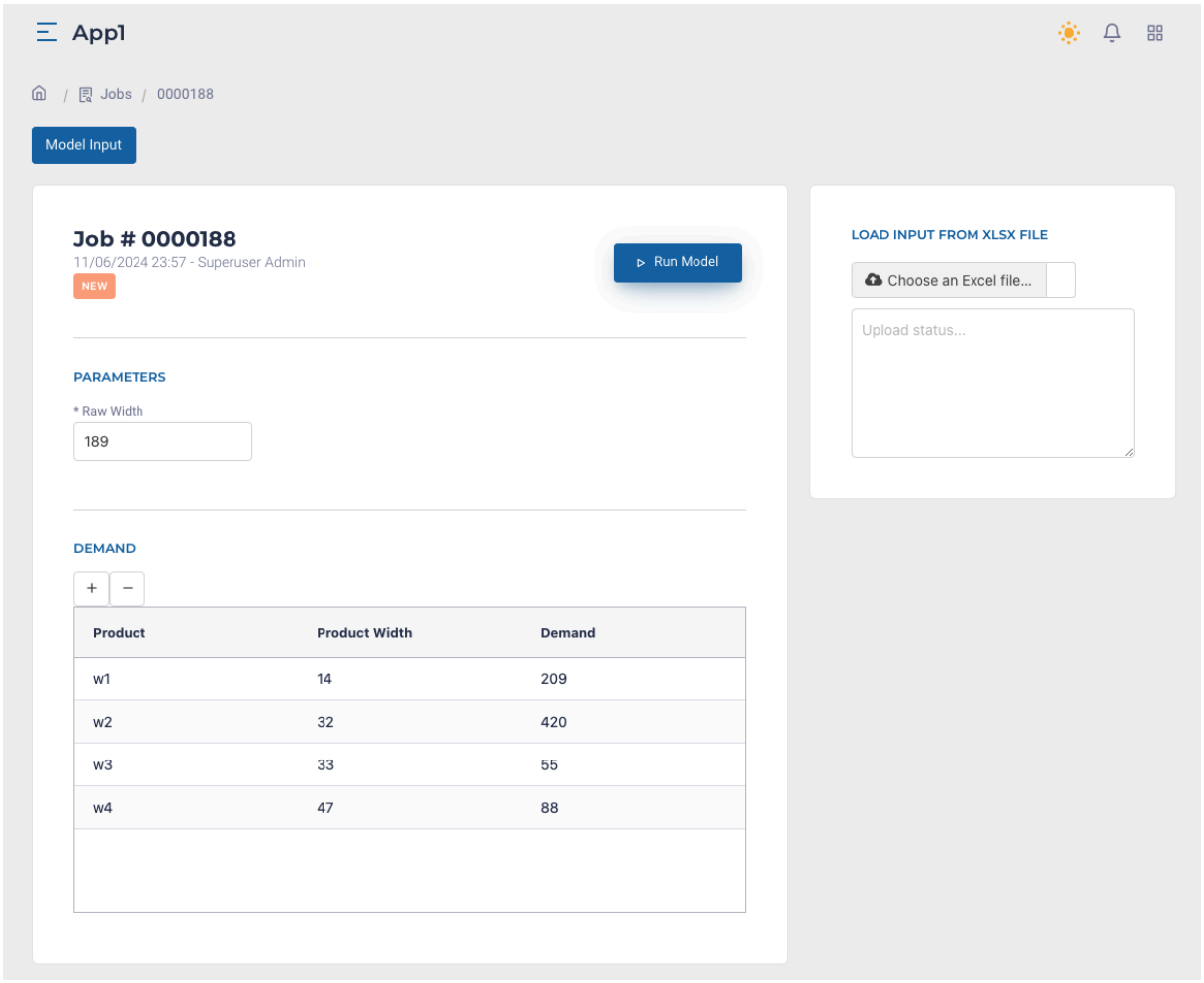


Figura 42. *Pantalla de entradas de App1.*

7.1.5. Salidas (Gráficas y analíticas)

Al igual que en los parámetros de entrada, la salida de cada ejecución se escribe en un archivo Excel, todos los campos se ven reflejados en la interfaz presentada en la Figura 43.

- *Patterns Used:* La hoja de salida 'Patrones utilizados' muestra los diferentes patrones y la cantidad utilizada.
- *Paper rolls Used:* La hoja de salida 'Rollos de papel utilizados' muestra la cantidad de rollos de papel utilizados por medio de la producción y demanda de cada familia de rollos.

A continuación, en la Figura 43, se muestra la interfaz web que representa la salida de una ejecución de la *App1*. Se diseñó un diagrama de barras para conocer la cantidad y tipo de rollos utilizados en cada patrón, donde los patrones se ven en el eje vertical y cada color representa un ancho de rollo.

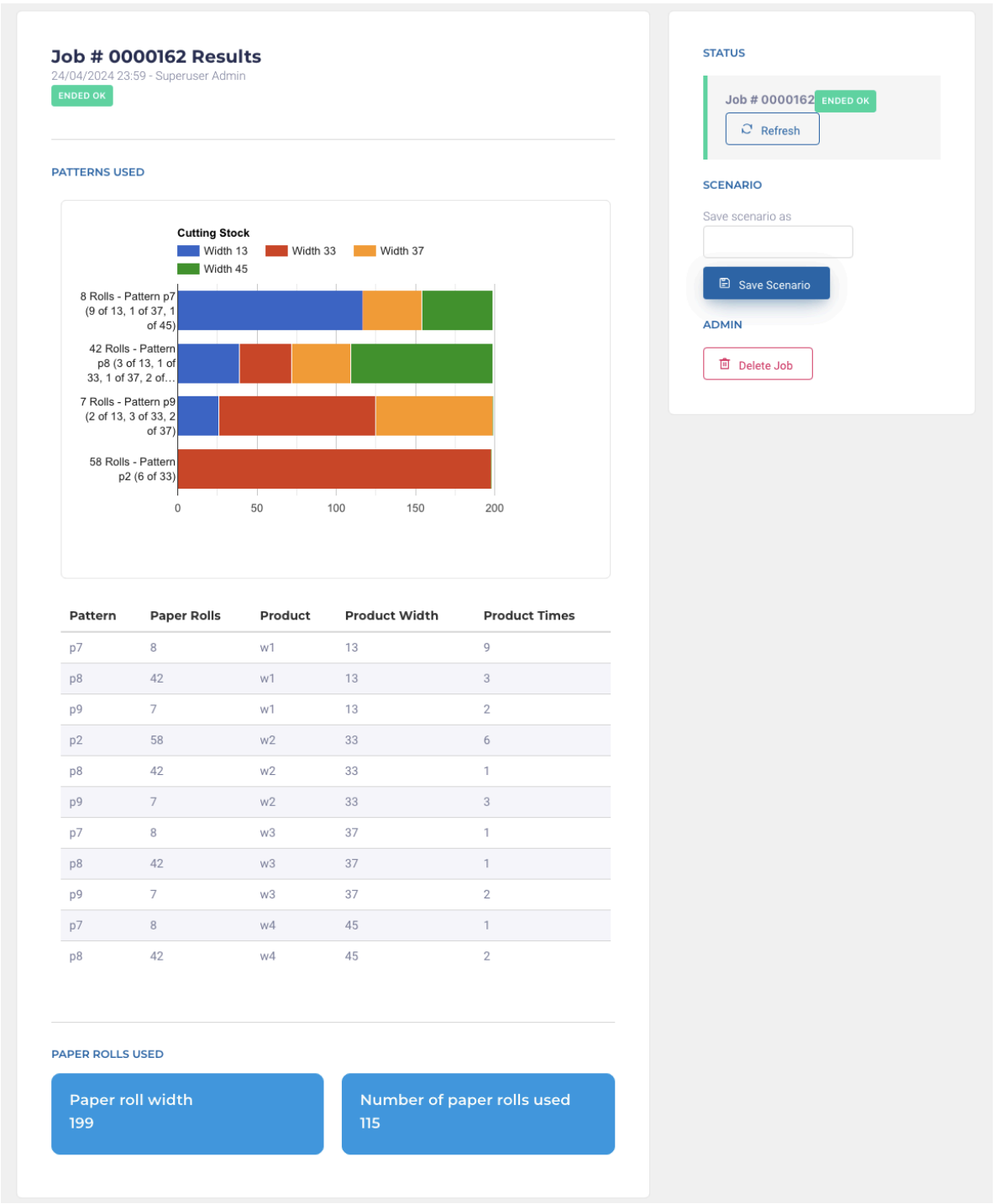


Figura 43. *Visualización de resultados de App1.*

7.2. App 2: *"Modelo MILP para el tratamiento simultáneo de las operaciones de producción y distribución en plantas batch multiproducto"*

7.2.1. Introducción

Esta aplicación se hizo en respuesta al trabajo realizado por los autores Aldana Tibaldo, Jorge Marcelo Montagna y Yanina Fumero titulado *"Modelo MILP para el tratamiento simultáneo de las operaciones de producción y distribución en plantas batch multiproducto"* (Tibaldo y otros, 2021). En este trabajo se presenta un modelo que permite determinar la programación tanto de la producción de lotes como de la distribución de los productos finales.

7.2.2. Marco Teórico

Para las empresas manufactureras, la programación de la producción y la distribución son dos actividades íntimamente ligadas, ya que la segunda solo puede iniciarse una vez que se han completado determinadas tareas del proceso productivo. Esta integración es crucial ante la alta variabilidad de los mercados y la necesidad de fabricar productos personalizados, o en industrias con productos de vida útil muy corta, que exigen una creciente flexibilidad y sincronización entre producción y distribución. En estos contextos, los productos finales deben ser entregados a los clientes poco después de su fabricación.

Dada una planta de producción con una instalación batch de unidades de procesamiento de capacidad variable para la producción de distintos productos, una serie de vehículos para transportar los productos y una serie de pedidos de clientes; el problema que busca resolver el modelo planteado en (Tibaldo y otros, 2021) consiste en determinar: el programa de producción óptimo que contemple el número, tamaño, asignación, secuenciación y temporización detallada de los lotes a procesar en cada unidad de la planta, y el plan de distribución que indique los vehículos a utilizar, la asignación de pedidos a cada unidad de transporte, la ruta y la temporización precisa de los clientes visitados por cada vehículo, a fin de minimizar el costo total de las operaciones de producción y distribución (Tibaldo y otros, 2021).

Para lograr esto, en (Tibaldo y otros, 2021) se propone un modelo de programación lineal mixta-entera (MILP) que resuelve simultáneamente la programación de la producción y distribución para el caso de una instalación batch multiproducto monoetapa con múltiples unidades no idénticas operando en paralelo. Para las operaciones de transporte el modelo considera una flota heterogénea de vehículos de diferentes capacidades y costos asociados. El modelo posee como objetivo satisfacer el total de órdenes que emiten los clientes para cada intervalo de caducidad dado, minimizando los costos totales (Tibaldo y otros, 2021).

7.2.3. Conjuntos (sets) del modelo

Tibaldo y otros (2021) proponen una serie de conjuntos para modelar el problema planteado. Entre ellos se encuentran los Clientes, los Productos y los Lotes. La Tabla 12 detalla el listado completo de conjuntos involucrados en el modelo de scheduling de producción, asignación y distribución.

Conjunto	Descripción	Ejemplo
I	Locaciones fijas (o Nodos)	$i0, i1, i2, ...$
$IP(I)$	Plantas manufactureras	$i0$
$IC(I)$	Clientes	$i1, i2, i3, ...$
M	Unidades de procesamiento (o Máquinas)	$m1, m2, m3, ...$
P	Productos	$p1, p2, p3, ...$
$L(M)$	Ranuras de tiempo (o Time-slots)	$l1, l2, l3, ...$
$B(M)$	Lotes (o Batches)	$b1, b2, b3, ...$
VT	Tipos de vehículos	$vt1, vt2, vt3, ...$
V	Vehículos	$v1, v2, v3, ...$
D	Ventanas de tiempo	$d1, d2, d3, ...$

Tabla 12. *Conjuntos involucrados en App2.*

En el modelo, las plantas manufactureras y los clientes se representan como nodos en el espacio, sobre los cuales se define la distancia entre ellos. Cabe aclarar, que si bien "Plantas manufactureras" es un conjunto, el modelo admite la definición de una única planta $i0$. El resto de los nodos i_n subsiguientes constituyen los clientes del problema.

Las ranuras de tiempo (o *time-slots*) son subdivisiones del tiempo total de operación de una máquina, y se utilizan para identificar con precisión el procesamiento de un lote de producto en un tiempo definido. Por cada *time-slot*, el tiempo de inicio y fin constituyen incógnitas del problema.

Los lotes (o *batches*) son conjuntos de productos que se procesan en una máquina particular. Un lote puede contener uno o más productos, y procesarse a lo largo de una o más ranuras de tiempo. Además, cada lote puede ser asignado a uno o más vehículos para su distribución.

Los tipos de vehículos se utilizan para agrupar los vehículos según sus características, como capacidad de carga y costo de distribución.

Las ventanas de tiempo definen intervalos de tiempo en los cuales los clientes pueden solicitar la entrega de uno o más productos. El modelo debe garantizar, en lo posible, que los productos sean entregados dentro de la ventana de tiempo correspondiente, minimizando los costos asociados a la tempranez o tardanza en la entrega.

7.2.4. Entradas de la aplicación

Por medio de las entradas de la aplicación se definen, no solo los parámetros configurables del modelo como la velocidad promedio de los vehículos vel, sino también, valores de tipo "cantidad" para definir el tamaño de determinados conjuntos, como por ejemplo, la cantidad de clientes CC.

Para facilitar la comprensión del modelo, las entradas fueron agrupadas en 6 categorías: Productos, Máquinas, Tipos de vehículos, Vehículos, Clientes y Demanda. La Tabla 13 presenta en forma completa todas las entradas configurables del modelo.

Categoría	Símbolo	Descripción	Unidad
Productos	PC	Cantidad de tipos de productos {p1, p2, p3, ...}	$\mathbb{N}$
	α_p	Factor de conversión de unidades de producto p a kg [KG:U]	kg/u
Máquinas (o Unidades de procesamiento)	MC	Cantidad de máquinas {m1, m2, m3, ...}	$\mathbb{N}$
	$capmax_{m,p}$	Capacidad máxima de la unidad m para el procesamiento del producto p	u
	$capmin_{m,p}$	Capacidad mínima de la unidad m para el procesamiento del producto p	u
	$ftp_{m,p}$	Tiempo de procesamiento fijo de cada lote de producto p en la unidad m	h/batch
	$fpc_{m,c}$	Costo de procesamiento fijo de producto p en la unidad m	$
Tipos de vehículos	VTC	Cantidad de tipos de vehículos {vt1, vt2, vt3, ...}	$\mathbb{N}$
	cap_{vt}	Capacidad máxima admisible del vehículo vt	kg
	prc_{vt}	Porcentaje mínimo de ocupación del vehículo vt	%
	cf_{vt}	Costo de distribución fijo del vehículo vt	$
	cv_{vt}	Costo de distribución variable del vehículo vt	$/km
Vehículos	VC	Cantidad de vehículos {v1, v2, v3, ...}	$\mathbb{N}$
	vel	Velocidad promedio	km/h
	$VVT(VT)$	Distribución de vehículos según su tipo vt	-
Clientes	CC	Cantidad de clientes {i1 ,i2, i3, ...}	$\mathbb{N}$
	$dist_{i,j}$	Distancia entre nodo i y nodo j	km
Demanda	DC	Cantidad de ventanas de tiempo	$\mathbb{N}$
	$limmin_d$	Límite inferior de la ventana de tiempo d	h
	$limmax_d$	Límite superior de la ventana de tiempo d	h
	ctt	Costo tardanza / tempranez	$/h

	$dem_{i,p,d}$	Cantidad de producto p demandada por el cliente i en la ventana de tiempo d	u

Tabla 13. *Entradas definidas en App2.*

A continuación se expone la representación por medio de la interfaz gráfica de las entradas definidas anteriormente.

La Figura 44 muestra las entradas relacionadas a los Productos.

Figura 44. Definición de PC y α_p.

La Figura 45 presenta las entradas relacionadas a las Máquinas (o Unidades de procesamiento).

Figura 45. *Definición de* MC, $capmax_{m,p}$, $capmin_{m,p}$, $ftp_{m,p}$ y $fpc_{m,c}$.

La Figura 46 muestra las entradas relacionadas con los Tipos de Vehículos.

Figura 46. *Definición de* VTC, cap_{vt}, prc_{vt}, cf_{vt} *y* cv_{vt}.

La Figura 47 presenta las entradas relacionadas a los Vehículos.

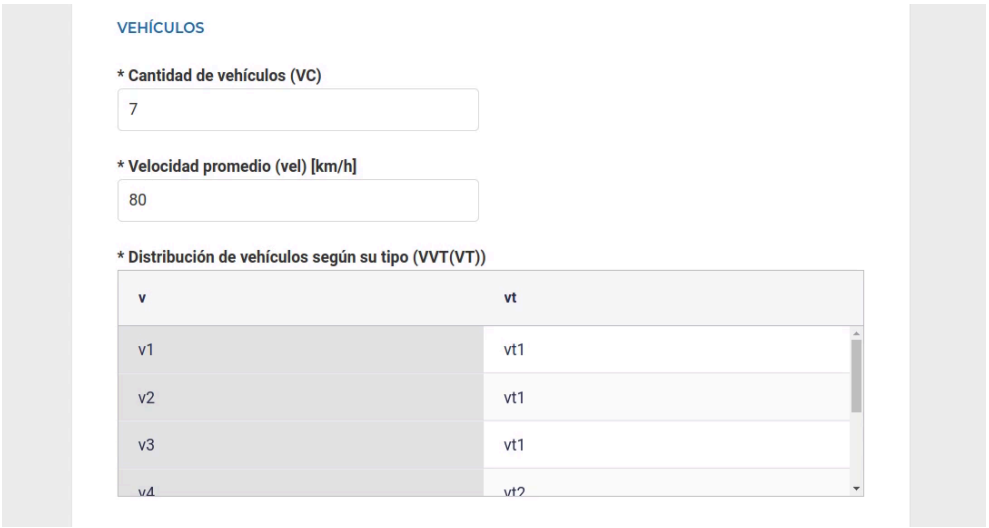


Figura 47. *Definición de* VC, vel *y* $VVT(VT)$.

Las entradas relacionadas a los Clientes se ven en la Figura 48.

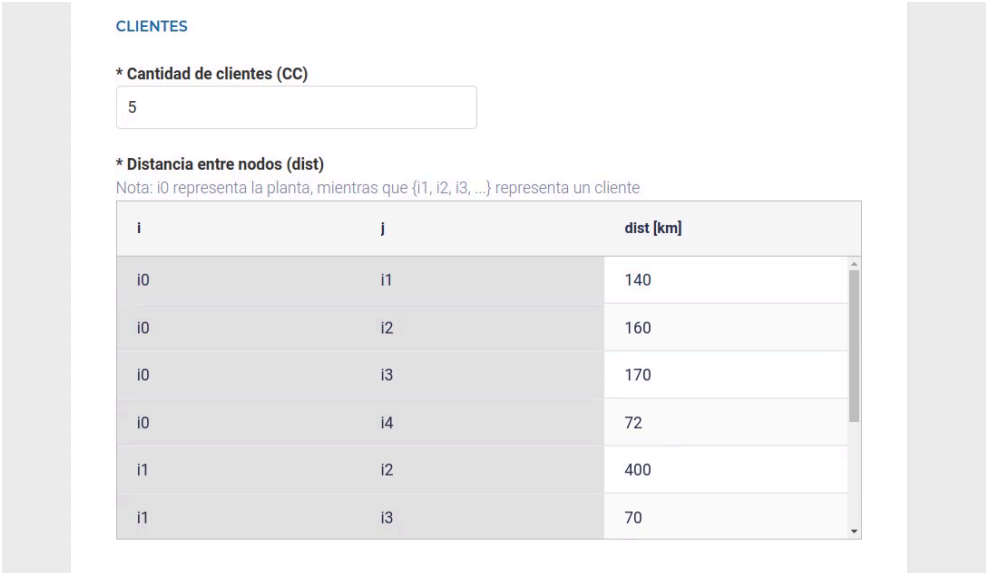


Figura 48. *Definición de* CC *y* $dist_{i,j}$.

Las entradas relacionadas a la Demanda se muestran en la Figura 49.

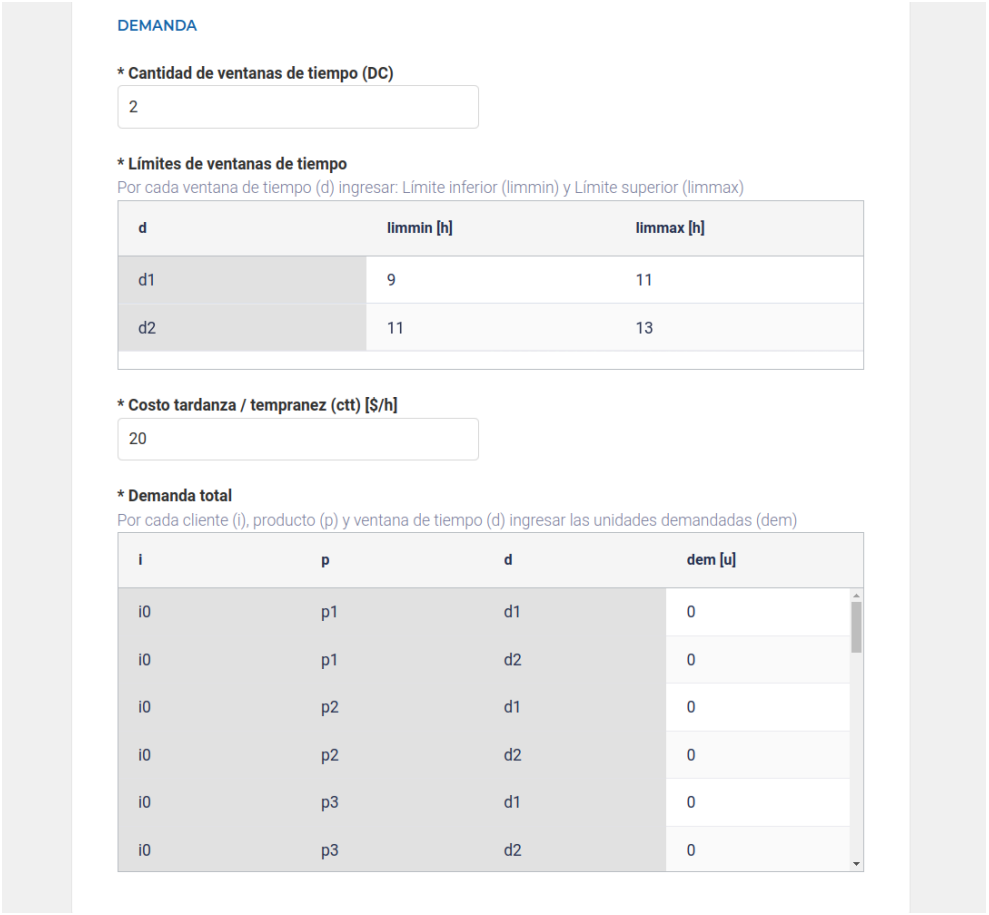


Figura 49. *Definición de* DC, $limmin_d$, $limmax_d$ *y* ctt *y* $dem_{i,p,d}$.

7.2.5. Variables de decisión

Las variables de decisión representan las incógnitas del problema, y por ende, constituyen las salidas del modelo. A continuación, se presenta en la Tabla 14 el conjunto completo de variables de decisión del problema.

Variable de decisión	Descripción	Unidad
$X_{b,p,m,l}$	Si el lote b de producto p es procesado en el *slot* l de la unidad m	{0,1}
$R_{b,p,v}$	Si el lote b del producto p es asignado al vehículo v	{0,1}
W_v	Si el vehículo v es o no utilizado	{0,1}
$Z_{i,d,v}$	Si el pedido del cliente i para la ventana de tiempo d es asignado al vehículo v	{0,1}
$ZP_{i,v}$	Si el cliente i es el primero en ser visitado en el recorrido del vehículo v	{0,1}
$Y_{i,j,v}$	Si el cliente i se le entrega inmediatamente antes del cliente j con el camión v	{0,1}
$ZU_{i,v}$	Si el cliente i es el último en la ruta del vehículo v	{0,1}
$BS_{b,p}$	Tamaño del lote b de producto p	u
$ST_{m,l}$	Tiempo inicial de procesamiento del *slot* l en la unidad m	h
$FT_{m,l}$	Tiempo final de procesamiento del *slot* l en la unidad m	h
DT_v	Tiempo de partida del vehículo v	h
$DET_{i,v}$	Tiempo de arribo al cliente i en el vehículo v	h
$T_{i,d}$	Tempranez de entrega al cliente i en la ventana d	h
$Ta_{i,d}$	Tardanza de entrega al cliente i en la ventana d	h
$QT_{b,p,v}$	Cantidad de unidades de producto p del lote b que son cargadas en el vehículo v	u

Tabla 14. *Variables de decisión del problema.*

7.2.6. Función objetivo

La función objetivo es la minimización del costo operativo total dado por los costos de producción y los costos de distribución. Dicha función objetivo se expresa como:

$$Minimizar(CPROD + CTRANS)$$

Donde

$$CPROD = \sum_{b \in B_p} \sum_{p \in P} \sum_{m \in M} fpc_{m,p} X_{b,p,m,l}$$

$$CTRANS = \sum_{j \in IC} \sum_{vt \in VT} \sum_{v \in VVT_{vt}} (tc_{i1,j,vt} ZP_{j,v} Q_{j,v}) + \sum_{i \in IC} \sum_{j \in IC} \sum_{vt \in VT} \sum_{v \in VVT_{vt}} (tc_{i,j,vt} Y_{i,j,v} Q_{j,v}) + \sum_{i \in IC} \sum_{vt \in VT} \sum_{v \in VVT_{vt}} (tc_{i,i1,j,vt} ZU_{i,v})$$

$$Q_{i,v} = \sum_{p \in P} \sum_{d \in D} dem_{i,p,d} \alpha_p Z_{i,d,v} \forall i \in IC, \forall v \in V$$

7.2.7. Salidas (Gráficas y analíticas)

Las salidas de App 2 son cinco: Resumen de ejecución, Tabla de Producción, Tabla de Asignación de lotes a vehículos, Tabla de Distribución, y gráfico *Gantt* de Producción, Asignación y Distribución. Con dichas salidas se pretende presentar al usuario final una interpretación amigable de las variables de decisión resultantes tras ejecutar el modelo.

Con el objetivo de reducir el ruido en la información presentada al usuario, las variables de decisión tipo booleanas ($X_{b,p,m,l}$, $R_{b,p,v}$, W_v, $Z_{i,d,v}$, $ZP_{i,v}$, $Y_{i,j,v}$ y $ZU_{i,v}$) no figuran en forma explícita en las salidas presentadas. Más bien, se utilizan para determinar cuáles de todas las posibles combinaciones entre las variables m, l, b, p, v, i y d deben mostrarse (o no) al usuario.

En la Figura 50 se presenta el Resumen de ejecución, donde se indica al usuario el Estado del modelo, el Valor objetivo y el Tiempo de ejecución.

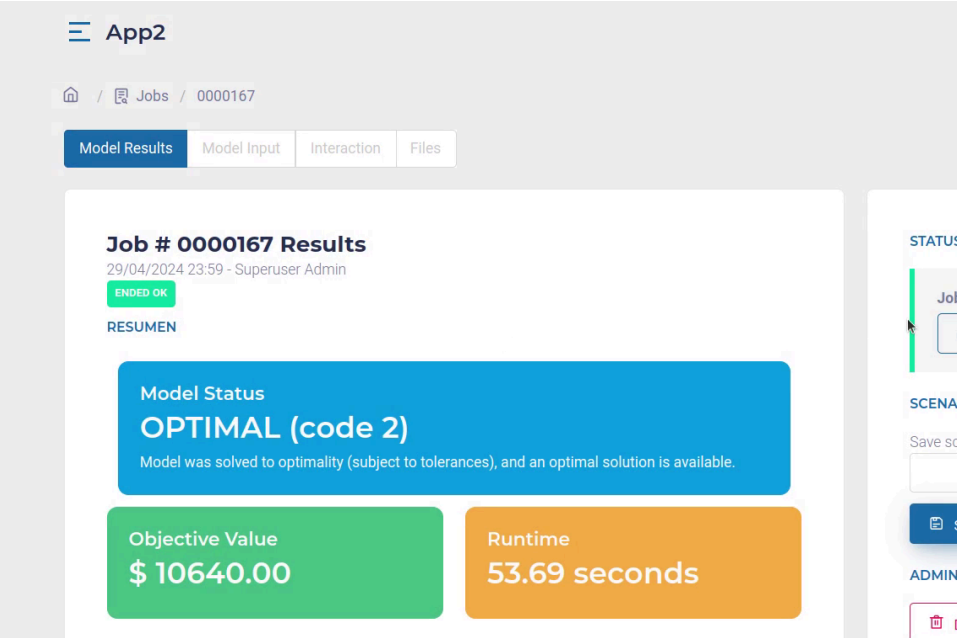


Figura 50. *Resumen de salida de ejecución: Model Status, Objective Value, Runtime.*

La Tabla de Producción (Figura 51) muestra cómo se planifica la producción de los productos en las distintas máquinas de la planta. Para ello, indica el lote (y su tamaño) y la ranura de tiempo en el que se procesa el producto, junto con tiempo de inicio y tiempo de finalización. Además, se incluye el porcentaje de utilización de la máquina durante la producción.

Para el ejemplo señalado en la Figura 51, la máquina $m1$, en su *time-slot* $l3$ produce producto $p2$ para el lote $b2$. El intervalo de procesamiento tiene inicio en $ST = 2.5$ y fin en $FT = 4$.

TABLA 1: PRODUCCIÓN

Por cada combinación de máquina (m), slot (l), lote (b) y producto (p):
$BS_{b,p}$ es el tamaño del lote en unidades.
$ST_{m,l}$ es el tiempo de inicio de producción.
$FT_{m,l}$ es el tiempo de finalización de producción.

m	l	b	p	$BS_{b,p}$ [u]	$ST_{m,l}$ [h]	$FT_{m,l}$ [h]	% utilización
m1	l1	b1	p3	150	0	1	100
m1	l2	b1	p2	200	1	2.5	100
m1	l3	b2	p2	200	2.5	4	100
m1	l4	b3	p2	200	4	5.5	100
m1	l5	b2	p3	150	5.5	6.5	100
m1	l6	b3	p3	150	6.5	7.5	100
m1	l7	b4	p3	150	7.5	8.5	100
m1	l8	b4	p2	200	8.5	10	100
m1	l9	b5	p3	150	10	11	100
m2	l1	b1	p1	150	0	1	100
m2	l2	b2	p1	150	1	2	100
m2	l3	b3	p1	150	2	3	100
m2	l4	b4	p1	150	3	4	100
m2	l5	b5	p1	150	4	5	100
m2	l6	b6	p3	100	5	6.5	100

Figura 51. *Tabla de producción. Incluye $BS_{b,p}$, $ST_{m,l}$, $FT_{m,l}$ y % de utilización.*

La Tabla de Asignación de lotes a vehículos (Figura 52) indica, dado un determinado lote b de tamaño $BS_{b,p}$, qué cantidad de producto $QT_{b,p,v}$ es asignado al vehículo v.

Siguiendo el ejemplo presentado anteriormente, el lote $b2$ de producto $p2$ tiene un tamaño $BS = 200$ unidades y es asignado a los vehículos $v6$ y $v7$ para su distribución, donde cada uno transporta 100 unidades por separado.

TABLA 2: ASIGNACIÓN DE LOTES A VEHÍCULOS

Por cada combinación de lote (b), producto (p) y vehículo (v):
$BS_{b,p}$ es el tamaño del lote en unidades.
$QT_{b,p,v}$ es la cantidad de producto asignada al vehículo v.

b	p	v	$BS_{b,p}$ [u]	$QT_{b,p,v}$ [u]
b1	p1	v6	150	150
b1	p2	v6	200	200
b1	p3	v6	150	150
b2	p1	v6	150	60
b2	p1	v7	150	90
b2	p2	v6	200	100
b2	p2	v7	200	100
b2	p3	v6	150	150
b3	p1	v7	150	110
b3	p2	v4	200	50
b3	p2	v7	200	150
b3	p3	v7	150	150
b4	p1	v4	150	50
b4	p1	v7	150	100
b4	p2	v4	200	200
b4	p3	v7	150	150
b5	p1	v4	150	150
b5	p3	v4	150	150
b6	p3	v4	100	60

Figura 52. *Tabla de Asignación de lotes a vehículos. Incluye* $BS_{b,p}$ *y* $QT_{b,p,v}$.

La Tabla de Distribución (Figura 53) presenta cada uno de los tramos del recorrido de cada vehículo v abarcando Tiempo de Partida y Tiempo de Entrega, además de Tempranez y Tardanza en caso de que existiera.

En el ejemplo señalado en la Figura 53, el vehículo $v7$ comienza su recorrido en $t = 9.03$, entregando productos a los clientes $i4$, $i3$ e $i1$ en ese orden, sin presentar demora ni adelanto.

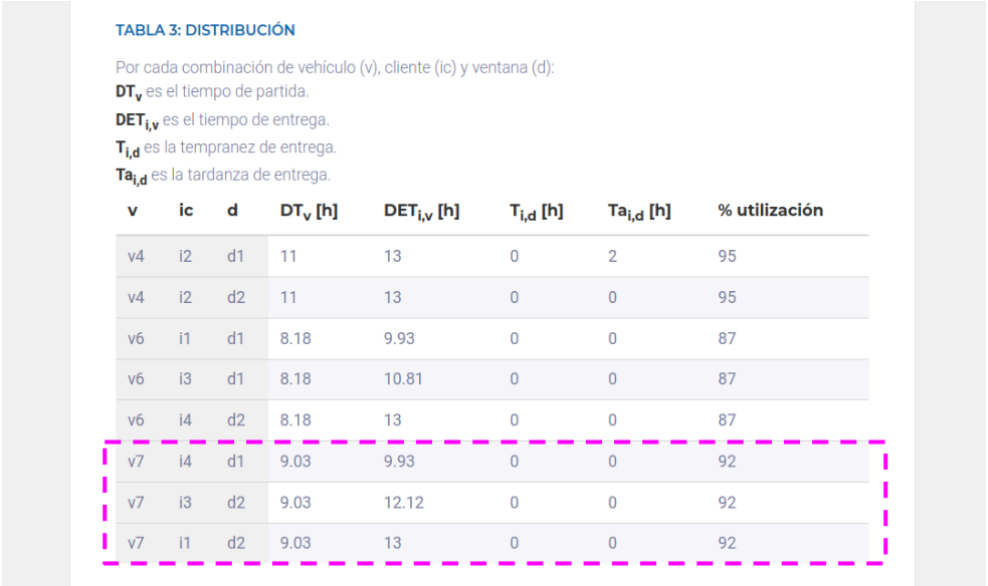


TABLA 3: DISTRIBUCIÓN

Por cada combinación de vehículo (v), cliente (ic) y ventana (d):
DT_v es el tiempo de partida.
$DET_{i,v}$ es el tiempo de entrega.
$T_{i,d}$ es la tempranez de entrega.
$Ta_{i,d}$ es la tardanza de entrega.

v	ic	d	DT_v [h]	$DET_{i,v}$ [h]	$T_{i,d}$ [h]	$Ta_{i,d}$ [h]	% utilización
v4	i2	d1	11	13	0	2	95
v4	i2	d2	11	13	0	0	95
v6	i1	d1	8.18	9.93	0	0	87
v6	i3	d1	8.18	10.81	0	0	87
v6	i4	d2	8.18	13	0	0	87
v7	i4	d1	9.03	9.93	0	0	92
v7	i3	d2	9.03	12.12	0	0	92
v7	i1	d2	9.03	13	0	0	92

Figura 53. *Tabla de Distribución. Incluye* DT_v, $DET_{i,v}$, $T_{i,d}$ *y* $Ta_{i,d}$.

Por último, el Gráfico *Gantt* de Producción, Asignación y Distribución (Figura 54) ofrece una visualización compacta de las salidas presentadas anteriormente.

Este gráfico incluye

- *Scheduling* de Producción (Figura 54.i).
- Ubicación temporal de las Ventanas de Tiempo (Figura 54.ii).
- Asignación de lotes de producto y Recorrido de los vehículos (Figura 54.iii).

Tómese como ejemplo el lote $b2$ de producto $p2$ (Figura 54.iv). Este lote tiene un tamaño $BS = 200$ unidades y se procesa en la máquina $m1$ en el time-slot $l3$ que comienza en $ST = 2.5$ horas y finaliza en $FT = 4$ horas. A partir de este lote, se asignan 100 unidades de producto $p2$ al vehículo $v6$ y 100 unidades $p2$ al vehículo $v7$.

Además, en Figura 54.v se observa cómo al vehículo $v7$ se le es asignado fracciones de los lotes $b2$, $b3$ y $b4$, alcanzando el 92% sobre su carga máxima. Dicho vehículo comienza su recorrido en $t = 9$ partiendo desde la planta $i0$, entrega primero parte de su carga al cliente $i4$ en $t = 10$, luego al cliente $i3$ en $t = 12.12$ y por último a $i4$ en $t = 13$.

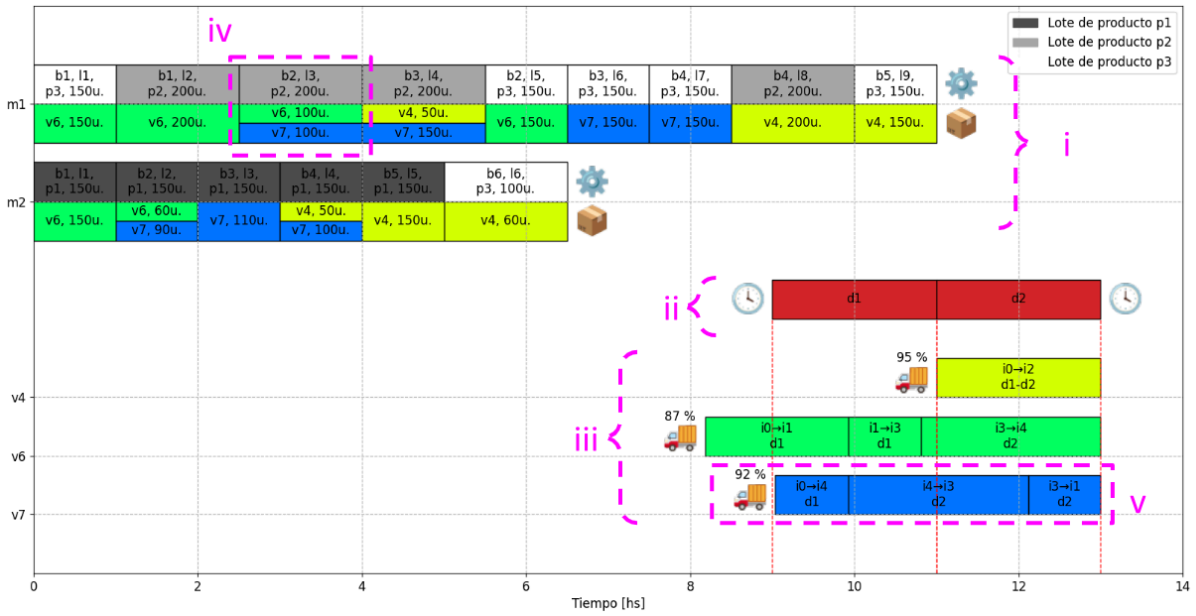


Figura 54. *Gráfico Gantt de Producción, Asignación y Distribución.*

7.3. *App 3: "Problema MILP para la optimización de rutas de camiones en la industria forestal"*

7.3.1. Introducción

Este programa aplica el modelo de optimización descrito en *Melchiori, L., Nasini, G., Montagna, J. M., & Corsano, G. . Problema de Recogida y Entrega en la Industria Forestal: nuevos modelos y estudio de su performance* (Melchiori y otros, 2020). El mismo consiste en una solución al problema de programación lineal entera correspondiente al ruteo óptimo de camiones transportadores de materia prima en una industria forestal.

7.3.2. Marco Teórico

En el contexto de una industria forestal, donde se debe transportar materia prima entre frentes de cosecha y plantas industriales, se debe planificar diariamente una serie de viajes, utilizando una flota de camiones distribuidos en diferentes bases regionales.

El transporte eficiente de troncos en una industria forestal es esencial para reducir costos. Esta aplicación presenta un modelo de Programación Lineal Entera (PLE) para resolver de manera exacta el Problema de Recogida y Entrega (PRE) específico del transporte de troncos de madera. En el mismo se deben planificar rutas para satisfacer la demanda de puntos de suministro, considerando diversas restricciones como limitaciones de tiempo y características de los vehículos. El problema implica tomar decisiones complejas debido a la gran cantidad de camiones, nodos de oferta y demanda, tipos de materia prima y aspectos legales.

La definición de restricciones y de función objetivo van en línea con el objetivo de mejorar la gestión logística y reducir costos en la industria forestal.

Las entidades principales del problema en cuestión son:

- Frente de cosecha: puntos de extracción de materia prima.
- Planta Industrial: punto de procesamiento de materia prima.
- Base Regional: punto de descanso de camiones.
- Ruta: conjunto de viajes de un camión entre frentes de cosecha, plantas industriales y bases regionales. Cada ruta puede componerse de cuatro tipos de viaje (ver Figura 55):
    a. Descargado desde una base regional a un frente de cosecha (1 en Figura 55).
    b. Cargado desde un frente de cosecha a una planta industrial (2, 4 y 6 en Figura 55).
    c. Descargado desde una planta industrial a un frente de cosecha (3 y 5 en Figura 55).
    d. Descargado desde una planta industrial a base regional (7 en Figura 55).

Se conoce inicialmente cuál es la demanda y la oferta diaria de cada frente de cosecha y planta industrial, respectivamente, en cantidad de troncos.

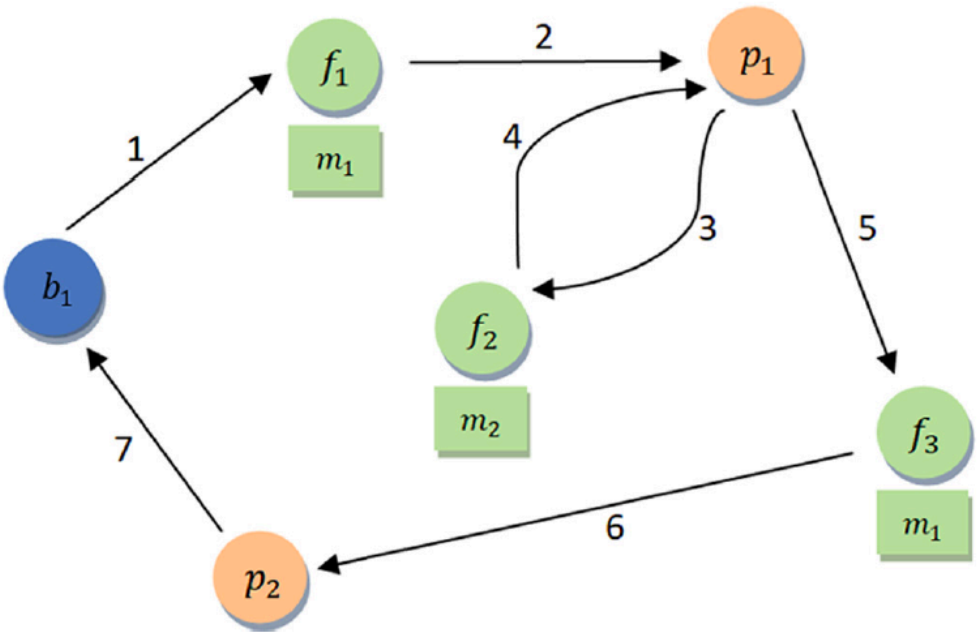


Figura 55. *Representación gráfica de una ruta correspondiente a un camión.*

El objetivo es generar un plan diario de ruteo de los camiones, minimizando el costo de los viajes que componen las rutas, a la vez que satisfaciendo las demandas de cada planta industrial y respetando las restricciones establecidas, teniendo presente las siguientes consideraciones:

- Cada ruta comienza y termina en la base regional correspondiente a cada camión.
- Un camión debe obligatoriamente dirigirse a una planta industrial luego de llenar su carga en un frente de cosecha (es decir, no puede visitar varios frentes consecutivamente).
- Las rutas deben componerse de a lo sumo siete viajes.
- Existe un valor máximo de tiempo que puede requerir una ruta.
- El costo a minimizar es el costo de la suma de los viajes de una ruta.
- El costo de una ruta se compone de un costo fijo por utilización de camión, y un costo variable en función de la distancia recorrida.

7.3.3. Entradas de la aplicación

En la Tabla 15 se presenta en forma completa todas las entradas configurables del modelo.

Nombre	Descripción	Unidad
N_{base}	Número de bases regionales	$\mathbb{N}$
N_{frente}	Número de frentes de cosecha	$\mathbb{N}$
N_{planta}	Número de plantas industriales	$\mathbb{N}$

$N_{matprima}$	Número de materias primas involucradas	$\mathbb{N}$
Cam_i	Cantidad de camiones por base i	$\mathbb{N}$
O_{fm}	Número de cargas completas de camiones ofertadas por cada frente f de cada materia prima m	$\mathbb{N}$
D_{pm}	Número de cargas completas de camiones demandadas por cada planta p de cada materia prima m	$\mathbb{N}$
$Dist_{bf}$	Distancia entre base b y frente de cosecha f	Km
$Dist_{pb}$	Distancia entre planta p y base b	Km
$Dist_{fp}$	Distancia entre frente f y planta p	Km
LRT	(*Limit route time*) Límite de tiempo de duración de una ruta	H
MTR	(*Maximum trips per route*) Cantidad máxima de viajes permitidos por ruta	$\mathbb{N}$
C_l	Costo por km recorrido con carga	$
C_u	Costo por km recorrido sin carga	$
V_l	Velocidad promedio de camión con carga	Km/h
V_u	Velocidad promedio de camión sin carga	Km/h

Tabla 15. *Variables de entrada App3.*

En una ventana de creación de un trabajo nuevo (New job) el usuario puede modificar el valor de las variables de entrada de la siguiente manera:

Para las variables N_{base}, N_{frente}, N_{planta}, $N_{matprima}$ ver Figura 56:

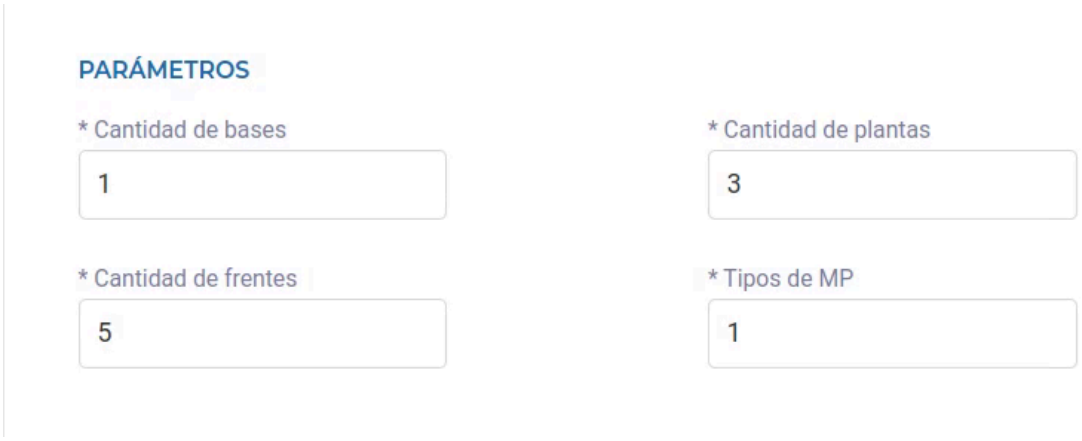


Figura 56. *Sección "Parámetros" en "New job".*

Para la variable Cam_i ver Figura 57:

Figura 57. *Sección "Bases, Plantas y Tipos de Materia Prima" en "New job".*

En las sucesivas tablas (*MP requerida por planta* y *MP ofertada por frente*) debe definirse qué materias primas son requeridas y ofertadas por cada planta industrial y frente de cosecha, respectivamente.

Para las variables O_{fm}, D_{pm} ver Figura 58:

OFERTAS Y DEMANDAS

Cantidad de MP ofertada por frente

Frente	Tipo MP	Cantidad
1	1	83
2	1	42
3	1	65
4	1	70
5	1	51

Cantidad de MP demandada por planta

Planta	Tipo MP	Cantidad
1	1	150
2	1	70
3	1	80

Figura 58. *Sección "Ofertas y Demandas" en "New job".*

Para las variables correspondientes a distancias $Dist_{bf}$, $Dist_{pb}$, $Dist_{fp}$ ver Figura 59:

DISTANCIAS

Distancias de base a frente (en kilómetros)

Base	Frente	Distancia
1	1	72
1	2	32
1	3	30
1	4	67
1	5	50

Distancias de frente a planta (en kilómetros)

Frente	Planta	Distancia
1	1	50
1	2	41
1	3	72
2	1	50
2	2	100
2	3	58

Distancias de planta a base (en kilómetros)

Planta	Base	Distancia
1	1	41
2	1	70
3	1	28

Figura 59. *Sección "Distancias" en "New job".*

Para las variables correspondientes a restricciones del modelo LRT, MTR, C_l, C_u, V_l, V_u ver Figura 60.

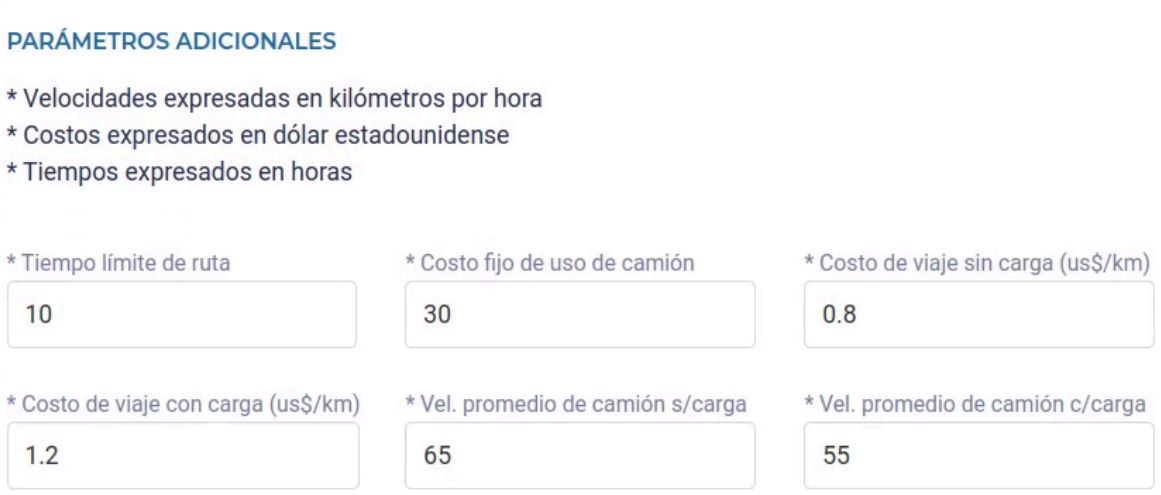


Figura 60. *Sección "Parámetros Adicionales" en "New job".*

7.3.4. Variables de decisión

Como se mencionó previamente, las variables de decisión representan las incógnitas del problema, y por ende, constituyen las salidas del modelo. A continuación, se presenta en la Tabla 16 el conjunto completo de variables de decisión del problema.

Nombre	Descripción	Unidad
$x_{c,f,m}^{V1}$	Comienzo de la ruta. 1 si el camión c se dirige al frente f a buscar materia prima m	{0,1}
$x_{c,f,p}^{Vi}$	Con i par. 1 si el camión c se dirige desde el frente f a la planta p	{0,1}
$x_{c,p,f,m}^{Vi}$	Con i impar. 1 si el camión c se dirige desde la planta p al frente f a buscar m	{0,1}
$x_{c,f,m}^{Vi}$	Fin de la ruta, con i impar. 1 si el camión c regresa a su base regional desde la planta p	{0,1}

Tabla 16. *Variables de decisión App3.*

7.3.5. Función objetivo

$$\begin{aligned}\text{TCOST} = &\sum_{b\in B}\sum_{c\in b}\sum_{m\in M}\sum_{f\in F_m} WUa_{b,f}\, x^{v_1}_{c,f,m} + \\ &\sum_{c\in C}\sum_{v\in V_{par}}\sum_{m\in M}\sum_{f\in F_m}\sum_{p\in P_m} WL_{f,p}\, x^{v}_{c,f,p} + \\ &\sum_{c\in C}\sum_{v\in V_{impar}}\sum_{p\in P}\sum_{m\in M}\sum_{f\in F_m} WUb_{p,f}\, x^{v}_{c,p,f,m} + \\ &\sum_{b\in B}\sum_{c\in b}\sum_{v\in V_{impar}}\sum_{p\in P} WUc_{p,b}\, x^{v}_{p,c}, \end{aligned} \tag{1}$$

donde V_{impar} y V_{par} denotan los subconjuntos de V dados por

$$V_{impar} = \{v_i \in V : i \text{ es impar }, i \geq 3\} \quad \text{y} \quad V_{par} = \{v_i \in V : i \text{ es par }\}.$$

7.3.6. Salidas (Gráficas y analíticas)

La salida del presente modelo puede distinguirse en dos partes; la primera correspondiente a las variables representativas de la respuesta al problema planteado, en términos analíticos y de performance (ver Figura 61), mientras que la parte ulterior muestra en detalle la solución a aplicar para alcanzar el valor óptimo de la función objetivo.

Figura 61. *Sección "Resultados Generales" en "job Results".*

La sección *Resultados Generales* indica las siguientes variables de salida:

- Valor de función objetivo: valor óptimo hallado para la función objetivo planteada, afectada por las restricciones definidas. La unidad en cuestión es pesos argentinos.
- Gap de optimalidad: distancia porcentual existente entre la mejor solución posible y la solución hallada. Se trata de una variable específica del solver Gurobi. Un gap de optimalidad igual a cero indica que se ha alcanzado la mejor solución posible.
- Camiones utilizados: número de camiones a conformar la flota para alcanzar la solución óptima.
- Tiempo de resolución: tiempo transcurrido para el hallazgo de la solución óptima.

La sección *Rutas de Camiones* (Figura 62) muestra detalladamente la solución específica al problema, indicando por cada entrada de la lista una ruta. Cada ruta contiene:

- Número de camión.
- Base regional de origen.
- Conjunto de viajes:
    - Número indicador de viaje.
    - Base regional origen del viaje.
    - Frente origen del viaje.
    - Tipo de materia prima.
    - Planta origen del viaje.

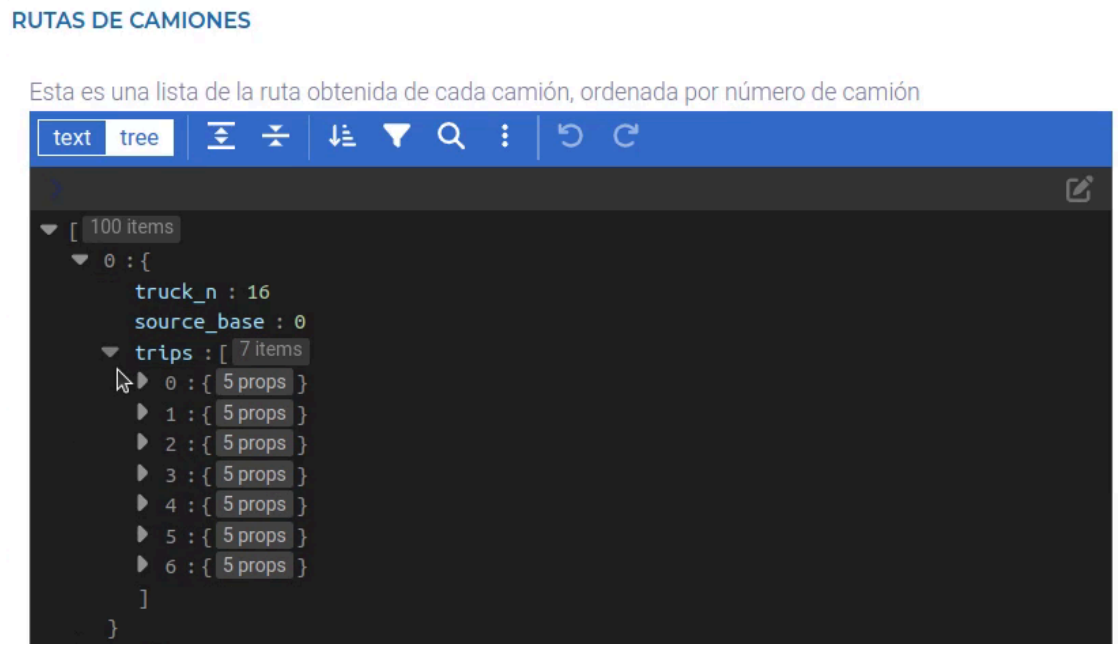


Figura 62. *Sección "Rutas de Camiones" en "job Results".*

7.4. App 4: "*Planificación de Secadero de tablas de madera*"

7.4.1. Introducción

Esta aplicación se fundamenta en el trabajo presentado en el artículo "Drying operation planning in a sawmill," publicado en 2020 en la revista Elsevier por los autores Nicolás Vanzetti, Gabriela Corsano y Jorge Montagna (Vanzetti y otros, 2020). En este artículo, se aborda la importancia crucial de la etapa de secado en la industria forestal, resaltando que el éxito en la fabricación de madera radica en la capacidad de mantener la calidad de la madera durante todo el proceso de producción. La etapa de secado en un aserradero es particularmente destacada, ya que determina tanto el valor como las características necesarias para satisfacer las demandas de los clientes. En esta investigación, se explora un enfoque basado en un modelo de Programación Disyuntiva Generalizada para abordar las decisiones clave relacionadas con la asignación de paquetes de tablas a hornos de secado y la programación de los ciclos de secado. Los resultados y capacidades de este enfoque se examinan a través de ejemplos prácticos.

7.4.2. Marco Teórico

El artículo "*Drying operation planning in a sawmill*" aborda la significativa función económica, ambiental y social de la industria forestal en el noreste de Argentina (Vanzetti y otros, 2020). El enfoque principal es en los aserraderos, que constituyen el 95% de las instalaciones de la industria. El trabajo, proveniente del artículo mencionado, busca re*solver* la carencia de estudios en la planificación de la etapa de secado en estos aserraderos. El artículo propone un enfoque innovador mediante la Programación Disyuntiva Generalizada (GDP) para optimizar tanto el llenado de los hornos con tablas como su programación. A través de ejemplos, se demuestra la efectividad de este enfoque en mejorar el rendimiento de una operación crítica en la industria forestal de la región.

En Vanzetti y otros (2020) se aborda la fase crítica de secado en la industria forestal, donde las tablas resultantes del proceso de aserrado deben alcanzar la humedad requerida para su comercialización. Estas tablas se agrupan en paquetes, que consisten en tablas de una familia y longitud específicas, basándose en la igualdad de grosor para definir la familia. Los paquetes se organizan en vagones que deben tener la misma longitud y contener tablas de la misma familia. Estos vagones cargados son introducidos en secadores a través de rieles y se acomodan en filas secuenciales, siendo el número de vagones por secador dependiente de la longitud de los vagones y el tamaño del secador.

El proceso de secado implica ciclos de secado con duraciones variables según el tamaño del secador y la familia de tablas. Esto garantiza un secado uniforme al procesar paquetes de la misma familia en cada ciclo. Para optimizar los recursos, se establece una carga mínima en cada secador. La planificación de la operación de secado se desarrolla en períodos de tiempo definidos, coincidiendo con la llegada de nuevos paquetes desde la etapa de aserrado. El objetivo principal del enfoque es optimizar la asignación de paquetes a hornos y programar los ciclos de secado dentro de este horizonte temporal, asumiendo recursos energéticos adecuados.

En resumen, la propuesta Vanzetti y otros (2020) se centra en la optimización de la planificación de secado en la industria forestal, donde la asignación eficiente de paquetes a hornos y la programación

estratégica de ciclos de secado son cruciales para garantizar un secado uniforme y eficaz de las tablas aserradas.

7.4.3. Entradas de la aplicación

Como se menciona en el desarrollo de App1, el modelo de planificación de secado también utiliza Excel como entrada de datos y un *solver GAMS*. En esta solución se secan paquetes de tablas, caracterizadas por el espesor de tabla (denominados “familia”) y el largo. Para secar hay un conjunto de secaderos. Cada secadero puede tener una o más filas donde ingresan “vagones” con una capacidad mínima y máxima de paquetes que pueden cargar. Cada vagón carga un tipo de familia y largo de paquetes. El tiempo de secado dependerá del secadero y de la familia de tablas que se quiera secar.

A continuación, en la Tabla 17 se detallan los parámetros de entrada utilizados en las hojas de cálculo, los cuales pueden verse reflejados en la interfaz desarrollada (ver Figuras 63.a y 63.b):

Nombre	Descripción	Unidad
Función Objetivo	Se puede elegir dos funciones objetivo (maximizar la cantidad de paquetes a secar y minimizar el espacio no utilizado de los secaderos). Además, el porcentaje de paquetes que pueden quedar en inventario al final del periodo de planificaciones (“Imin”), teniendo en cuenta el total de paquetes disponibles. Y a partir de qué momento van a estar disponibles los secaderos al comienzo del periodo (se consideran que algunos pueden estar siendo utilizados por la planificación anterior, denominado “tocu”).	{“Maximizar tablas secadas”, “Minimizar espacio secadero”}
Tiempo	El horizonte de planificación está dividido en varios periodos, siendo la unidad de tiempo el día. Especifica la hora de inicio y de fin de cada periodo.	ℕ
Largos	Se cargan los distintos largos de los paquetes de tablas	ℕ
PaqDisp	Se carga la cantidad de paquetes que ingresan para ser secados en cada periodo de tiempo.	ℕ

Tabla 17. *Parámetros de entrada de App4.*

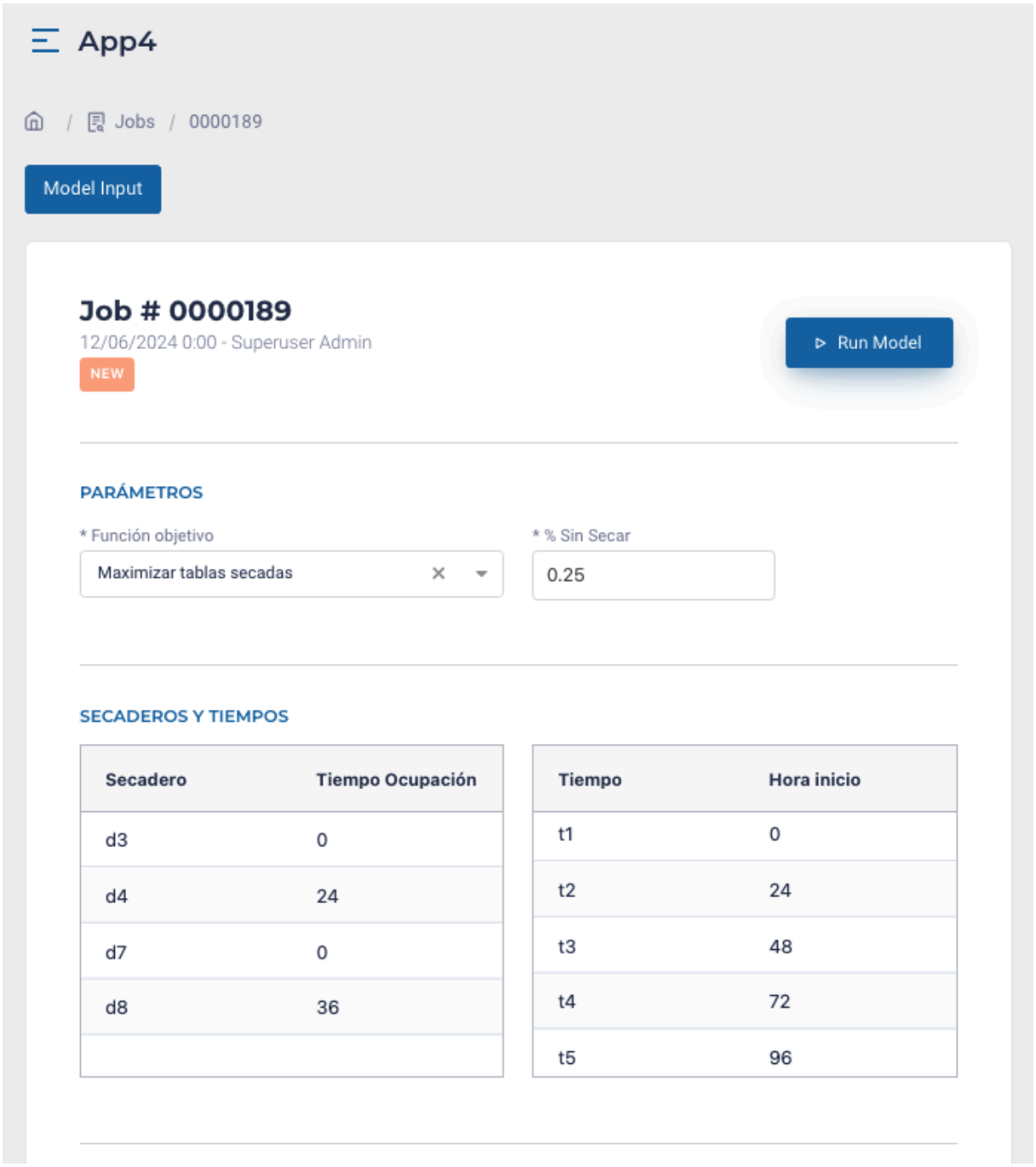
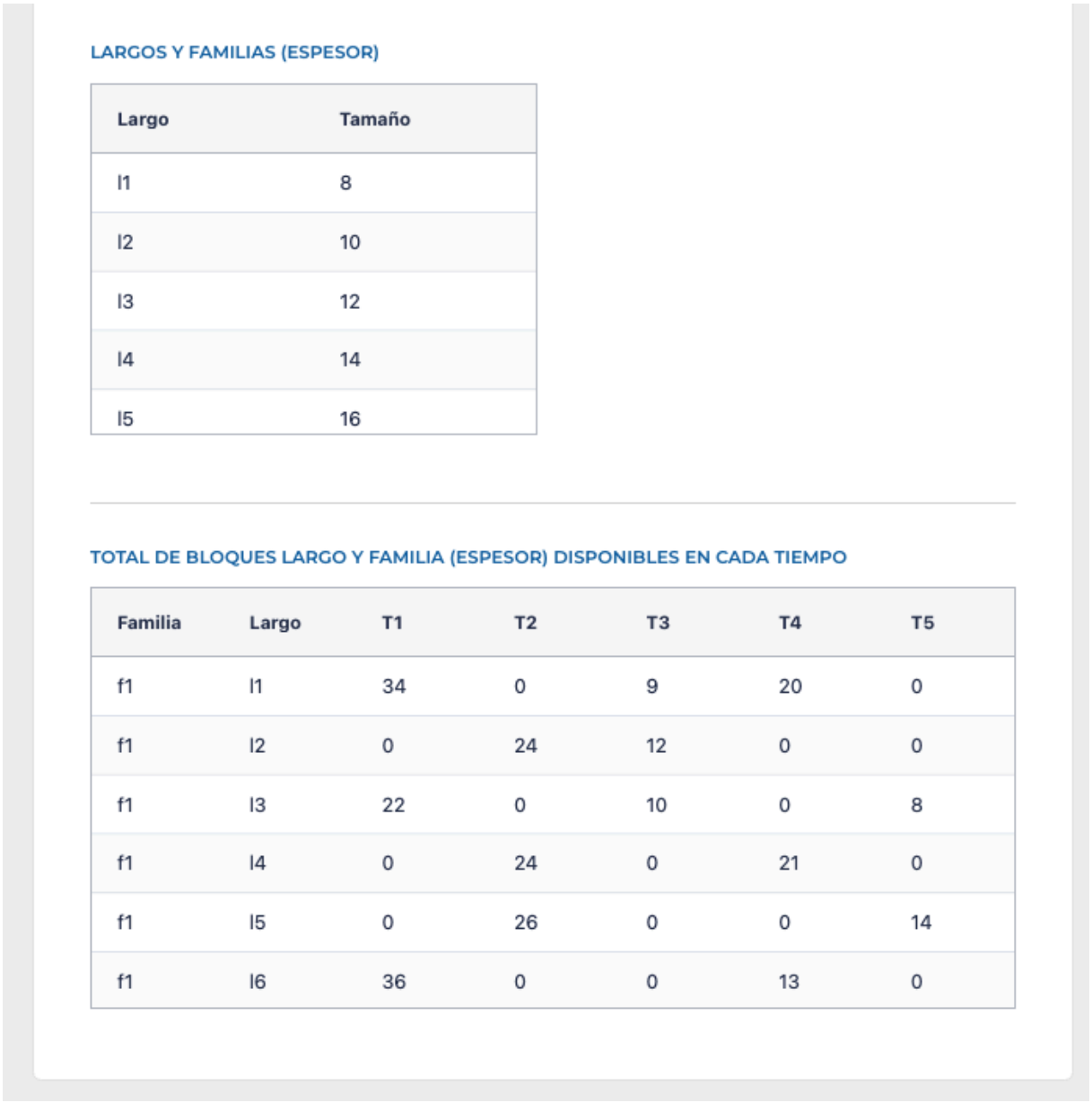


Figura 63.a. *Pantalla de entradas de App4.*



LARGOS Y FAMILIAS (ESPESOR)

Largo	Tamaño
l1	8
l2	10
l3	12
l4	14
l5	16

TOTAL DE BLOQUES LARGO Y FAMILIA (ESPESOR) DISPONIBLES EN CADA TIEMPO

Familia	Largo	T1	T2	T3	T4	T5
f1	l1	34	0	9	20	0
f1	l2	0	24	12	0	0
f1	l3	22	0	10	0	8
f1	l4	0	24	0	21	0
f1	l5	0	26	0	0	14
f1	l6	36	0	0	13	0

Figura 63.b. *Pantalla de entradas de App4.*

7.4.4. Salidas (Gráficas y analíticas)

Como resultado se muestra qué familia de tablas se seca en cada ciclo de secado, qué cantidad de paquetes entra en cada ciclo y en qué fila del secadero se debería poner.

Al igual que los datos de entrada, la ejecución del modelo matemático devuelve los resultados en una planilla Excel, a continuación se detalla cada variable de salida.

- Resumen: Se muestran los resultados más relevantes como ‘Cantidad de paquetes secados’, ‘Cantidad de ciclos necesarios’, ‘Cantidad de paquetes sin secar’, ‘Porcentaje de paquetes sin secar’, ‘Porcentaje de uso de secadero’, ‘Tiempo de resolución’ y ‘GAP relativo’.

- Q: Muestra la cantidad de paquetes según familia y largo de tablas que se secan en cada ciclo de secado en cada secadero.
- Llenado: Muestra la cantidad de vagones que ingresan en cada fila del secadero, junto con tipo y tamaño de las tablas que transporta.
- inventarioFinal: Muestra el inventario de paquetes de tablas sin secar al final de cada periodo de tiempo.
- Volusociclo: Porcentaje de llenado que tiene cada secadero en cada ciclo de secado.
- TIS: Momento en el que inicia cada ciclo de secado en cada secadero.
- TFS: Momento en el que finaliza cada ciclo de secado en cada secadero.

Además, la interfaz web desarrollada para esta app muestra los resultados de manera gráfica para facilitar la comprensión de los mismos, a continuación se muestra como ejemplo la salida de una ejecución (ver Figura 64).

La pantalla de “Resultados” muestra:

- Tipo FO
- Límite de inventario
- Cantidad de Paquetes secados
- Porcentaje de paquetes sin secar
- Cantidad de paquetes secados
- Tiempos de resolución

En la pantalla de gráficos se generan dos tipos. El primero muestra un gráfico de torta para representar el porcentaje de uso del secadero, y el segundo genera un gráfico de *Gantt* para mostrar la duración de los ciclos de secado de cada secadero.

La integración de esta funcionalidad gráfica se realizó mediante la incorporación de la biblioteca *vue-google-charts* (Terekhov, 2022). Esta biblioteca importa un conjunto predefinido de gráficos de *Google* y simplifica su uso dentro del entorno de Vue.js. Con *vue-google-charts*, se pueden crear gráficos fácilmente en aplicaciones Vue.js utilizando la sintaxis familiar de *Vue*. Esta extensión maneja la configuración y la integración con la *API* de *Google Charts*, lo que hace que sea más sencillo para los desarrolladores implementar gráficos en sus aplicaciones Vue.js sin tener que preocuparse por la complejidad de la integración con *Google Charts*.

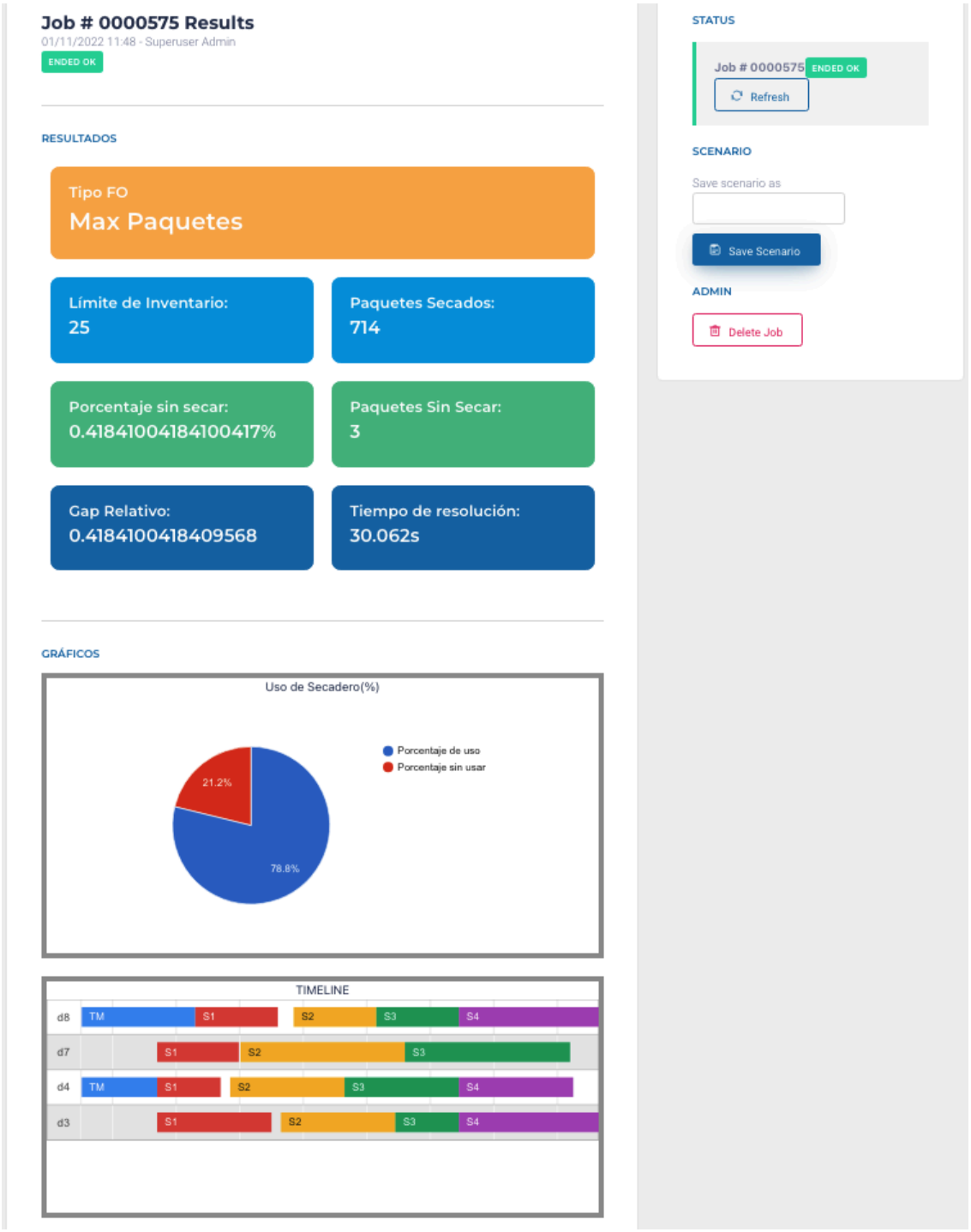


Figura 64. *Visualización de resultados de App4.*

8. Conclusiones

El proyecto desarrollado ha cumplido con éxito los objetivos planteados, proporcionando soluciones innovadoras basadas en tecnologías avanzadas de optimización matemática para pequeñas y medianas empresas. A continuación, se presentan las conclusiones obtenidas a partir de este trabajo.

8.1. Conclusiones sobre las Aplicaciones Desarrolladas

Las aplicaciones desarrolladas han demostrado ser efectivas en la implementación de modelos de optimización matemática en distintos dominios industriales. La arquitectura de la plataforma permitió adaptar cada aplicación a las necesidades específicas de optimización de producción, ofreciendo soluciones robustas y escalables.

8.1.1. App 1: "Optimización en Problemas de Corte"

Esta aplicación funcionó como un prototipo de validación, replicando de manera efectiva un ejemplo preexistente. Se logró establecer con éxito la conexión con el servidor *GAMS* y familiarizar a los desarrolladores con la tecnología a utilizar.

Siguiendo la metodología de trabajo de prototipado evolutivo, la App 1 funcionó como el prototipo inicial que estableció las bases para el desarrollo de las siguientes aplicaciones. Este enfoque permitió mejorar y adaptar progresivamente las funcionalidades y características de la plataforma, garantizando que cada nueva aplicación se construyera sobre una base sólida y probada.

8.1.2. App 2: "Modelo MILP para el tratamiento simultáneo de las operaciones de producción y distribución en plantas batch multiproducto"

La aplicación logró integrar modelos matemáticos avanzados para optimizar operaciones complejas de producción y distribución en plantas batch, permitiendo una planificación personalizada y flexible.

Además, la implementación permitió integrar el modelo analítico dentro de la plataforma unificadora, la cual permite visualizar tanto los datos de entrada como la salida del *solver* matemático *Gurobi* de una forma amigable para el usuario final.

8.1.3. App 3: "Problema MILP para la optimización de rutas de camiones en la industria forestal"

La aplicación fue exitosa en la obtención de resultados de fácil visualización al usuario, logrando adaptar un dominio complejo a los conocimientos del usuario final promedio.

Al mismo tiempo, la capacidad de la arquitectura para manejar grandes volúmenes de datos y realizar cálculos complejos permitió una implementación efectiva del modelo matemático, proporcionando resultados precisos y rápidos.

La implementación local del software de resolución de problemas de optimización logró la eficiente interacción entre este y la aplicación correspondiente, alcanzando alta velocidad de ejecución y escaso uso de recursos de hardware.

8.1.4. App 4: "Planificación de Secadero de tablas de madera"

- Esta aplicación facilita la planificación de operaciones en secaderos de tablas de madera, optimizando los tiempos de secado y maximizando la producción.
- Se logró conectar la aplicación a los servidores de *GAMS* para su ejecución, adaptando las entradas y salidas para facilitar el uso del programa por parte de los usuarios finales. La integración con la plataforma de aplicaciones de productividad avanzada permitió una visualización clara y detallada de los resultados, facilitando la toma de decisiones por parte de los gestores y operadores.

8.2. Conclusiones sobre la implementación de aplicaciones

Al finalizar el proyecto, se realizó una estimación de los recursos necesarios para desarrollar una nueva aplicación, basada en la experiencia obtenida durante el desarrollo de las cuatro aplicaciones anteriores. Se concluyó que el desarrollo de una nueva app llevaría entre 30 y 40 días, dependiendo de varios factores.

Uno de los principales determinantes es la familiaridad del desarrollador con la plataforma. Si es la primera vez que un programador trabaja con esta plataforma, el tiempo necesario será mayor debido al proceso de adaptación. En cambio, si el desarrollador ya ha participado en la creación de alguna de las aplicaciones previas, el tiempo de desarrollo se reducirá, ya que podrá aprovechar las aplicaciones anteriores como base o prototipo.

Otro factor que influye en el tiempo de desarrollo es la necesidad de que el programador se familiarice con el dominio específico de la nueva aplicación. Esto puede agregar variabilidad a los plazos, ya que algunos dominios pueden ser más complejos de entender que otros.

En resumen, estimamos que el tiempo necesario para desarrollar una nueva aplicación oscilaría entre 30 y 40 días, dependiendo de la experiencia previa del equipo y la complejidad del dominio de la aplicación.

8.3. Conclusiones sobre el Proyecto

El proyecto ha sido una oportunidad invaluable para aplicar y validar la metodología de desarrollo utilizada. El enfoque de prototipado evolutivo facilitó la iteración rápida y la adaptación a los requisitos de los usuarios finales. A través de este enfoque, se logró el alcance de los objetivos establecidos, proporcionando una plataforma de software que sea de fácil implementación, que soporte el escalamiento de negocio, y fundamentalmente que provea un soporte robusto y confiable a la toma de decisiones de negocio del eventual cliente.

- **Metodología Implementada**

  La metodología de prototipado evolutivo permitió una respuesta ágil a los cambios y necesidades del proyecto, asegurando la entrega iterativa de productos que cumplen con las expectativas tanto para usuarios avanzados como usuarios finales.

  Las reuniones periódicas y la comunicación constante con los stakeholders fueron fundamentales para alinear el desarrollo con los objetivos estratégicos del proyecto.

- **Alcance de los Objetivos**

  El proyecto logró desarrollar aplicaciones funcionales y validadas en entornos controlados, demostrando la viabilidad de la arquitectura y la eficacia de los modelos matemáticos implementados.

  Si bien las aplicaciones no fueron probadas en entornos de producción reales de PyMEs, se logró implementar soluciones que cumplen con los requisitos técnicos y operativos establecidos.

- **Experiencia Adquirida**

  El equipo adquirió una experiencia invaluable en el desarrollo de soluciones tecnológicas complejas, fortaleciendo sus habilidades en programación, gestión de proyectos y optimización de procesos industriales.

  Se obtuvo experiencia aplicando tecnologías estudiadas durante la carrera, como MongoDB y *Python*, así como en nuevas tecnologías que eran desconocidas para todos los miembros del equipo, como *Vue.js*.

- **Desvíos con respecto al cronograma**

  Si bien no se cumplieron los plazos proyectados inicialmente, las desviaciones no fueron significativas ni impactaron negativamente en la calidad o el alcance del proyecto. Las dificultades encontradas fueron abordadas de manera efectiva, asegurando la entrega de una solución funcional dentro de los plazos de contratación establecidos.

8.4. Alcance de los Objetivos

Desde el comienzo del proyecto, se establecieron tanto objetivos específicos como un objetivo general, los cuales guiaron el desarrollo de la plataforma de software. Para asegurar el cumplimiento del objetivo general, se puso especial atención en alcanzar cada uno de los objetivos específicos definidos, ya que estos son fundamentales para la funcionalidad global de la plataforma.

8.4.1. Objetivos Específicos:

- **Administración de usuarios y aplicaciones**
  Se implementaron con éxito funcionalidades de autenticación y gestión de usuarios, permitiendo un control efectivo de la administración de aplicaciones dentro de la plataforma. Esto asegura que tanto usuarios finales como expertos puedan acceder de manera segura a las herramientas de optimización.

- **Lectura de datos y ejecución de problemas**
  Las aplicaciones lograron integrar modelos matemáticos avanzados que permiten la lectura de datos y la ejecución de problemas de optimización en diferentes dominios industriales. De esta forma, se logró cumplir con los requisitos de funcionamiento establecidos al inicio del proyecto, proporcionando una solución adaptada a las necesidades de las PyMEs.

- **Lectura y presentación de resultados**
  La plataforma desarrollada permite una visualización clara y comprensible de los resultados generados por los solvers matemáticos, facilitando la interpretación por parte de los usuarios finales, con la expectativa de facilitar la toma de decisiones basadas en estos resultados.

- **Comparación de escenarios**
  Se implementó la capacidad de comparar diferentes escenarios dentro de la plataforma, permitiendo a los usuarios evaluar varias soluciones a problemas de optimización. Esta funcionalidad es clave para que las PyMEs puedan analizar diferentes enfoques y seleccionar la estrategia más adecuada para sus procesos productivos.

8.4.2. Objetivo General:

El objetivo general de desarrollar e implementar una plataforma que permitiera a las pequeñas y medianas empresas acceder a herramientas informáticas avanzadas y mejorar su gestión productiva fue alcanzado con éxito. La plataforma no solo ofreció soluciones escalables, sino que también proporcionó un entorno robusto y accesible para la optimización de procesos industriales, sin la necesidad de personal altamente especializado.

El cumplimiento exitoso de los objetivos específicos permitió asegurar que el objetivo general se alcanzara de manera efectiva. Cada uno de estos objetivos específicos contribuyó de manera integral al desarrollo de una plataforma funcional, escalable y eficiente para el beneficio de las PyMEs.

8.5. Reflexión Final

En resumen, el proyecto ha sido un éxito en términos de implementación tecnológica y logro de objetivos estratégicos. La plataforma desarrollada no solo ha demostrado ser efectiva en la optimización de procesos industriales, sino que también ha sentado las bases para futuras aplicaciones y mejoras. Si bien esta versión puede considerarse funcional, se recomienda continuar trabajando en futuras iteraciones para pulir funcionalidades adicionales y mejorar la experiencia del usuario. Estamos satisfechos por haber desarrollado una primera versión funcional dentro de los plazos de contratación, y esperamos que esta plataforma siga evolucionando para beneficiar a un número aún mayor de pequeñas y medianas empresas.

Bibliografía

Canto, P. (2024). *hopeit-git/hopeit.engine: Python engine that allows creation and operations of reactive microservices*.https://github.com/hopeit-git/hopeit.engine

Clements, P., Garlan, D., Little, R., Nord, R., & Stafford, J. (2003). Documenting software architectures: views and beyond. *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, 740-741. https://dl.acm.org/doi/10.5555/776816.776928

GAMS Development Corporation. (2017). *General Algebraic Modeling System (GAMS) Release 24.8.5*. https://www.*GAMS*.com/download/

Gurobi Optimization, LLC. (2020). *Gurobi Optimizer Reference Manual*. https://www.*Gurobi*.com

Melchiori, Luciana & Nasini, Graciela & Montagna, Jorge & Corsano, Gabriela. (2020). A mathematical modeling for simultaneous routing and scheduling of logging trucks in the forest supply chain. Forest Policy and Economics. 136. 102693. 10.1016/j.forpol.2022.102693.

Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

Terekhov, S. (2022). *vue-google-charts: Reactive Vue.js wrapper for Google Charts lib*. https://github.com/devstark-com/vue-google-charts

Tibaldo, A., Montagna, J. M., & Fumero, Y. (2021). Modelo MILP para el tratamiento simultáneo de las operaciones de producción y distribución en plantas batch multiproducto. *Sociedad Argentina de Informática e Investigación Operativa*, (IV Simposio Argentino de Informática Industrial e Investigación Operativa (SIIIO 2021) - JAIIO 50 (Modalidad virtual)), 105-118. https://sedici.unlp.edu.ar/handle/10915/141777

Vanzetti, N. A., Corsarno, G., & Montagna, J. M. (2020). Drying operation planning in a sawmill. *Computers and Chemical Engineering*. https://ri.conicet.gov.ar/handle/11336/144310

Anexo I: Registros de Meetings

Martes 2023/5/2
* Duración: 90 minutos.
* Participantes: Hernández, Mione, Hillar, Toniolo, Dalmasso
* Temas: idea de proyecto final de carrera, diseño del esqueleto de plataforma, presentación de prototipo de API de *back-end*.

Lunes 2023/5/8
* Duración: 60 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas: temas generales de python y *back-end*.

Martes 2023/5/16
* Duración: 60 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso, Mione
* Temas: muestra de estructura de api app01, distribución de primeras tareas para desarrollar la app1 (App inicial).

Miércoles 2023/5/24
* Duración: 60 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso, Aldo Vecchietti
* Temas: demostración de *GAMS* por parte de Vecchietti, definición de UI topics.

Lunes 2023/5/29
* Duración: 90 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso, Developer *back-end* Hopeit engine
* Temas: Capacitación y presentación de la herramienta Hope It Engine.

Viernes 2023/6/9
* Duración: 60 minutos.

* Participantes: Hernández, Mione
* Temas: Definición de estrategias, objetivos y modelos a implementar.

Jueves 2023/6/15
* Duración: 60 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso, Mione
* Temas:
* Presentación de App1
* Definición de tareas a realizar en App1
* Platform troubleshooting

Viernes 2023/6/23
* Duración: 50 minutos.
* Participantes: Toniolo, Dalmasso, Mione
* Temas:
* Pruebas y avances sobre AGgrid
* Prueba de [librería para conversión JSON-Excel y viceversa](https://github.com/mustafacagri/vue-js-excel).

Martes 2023/6/27
* Duración: 60 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso, Mione
* Temas:
* *GAMS* Engine setup - work in progress
* Developer roadmap
* xlsx to json tools
* Fidel work on python xlsx manipulation
* Blocking:
* Definir donde colocar AgGrid (by Toniolo)
* Login problems (by Dalmasso/Hillar)
* Next steps:
* Distribución de tareas definidas en gitlab issues

Miércoles 2023/7/5
* Duración: 180 minutos.
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Alternativas de arquitectura para soporte de aplicaciones *GAMS*

* Troubleshooting de:
* Contenedores Docker.
* Comunicación con API admin.
* Instalación y uso de AG Grid.

Viernes 2023/7/14
* Duración: 180 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Se propone la virtualización para correr *GAMS* en Windows
* Continúa el desarrollo de SaveScenario (by Hillar)
* Finaliza la tarea de leer un Excel y transformarlo a JSON (by Dalmasso)
* Finaliza AGgrid (by Toniolo)
* Se realiza un Merge por cada feature finalizado

Lunes 2023/7/17
* Duración: 180 minutos
* Participantes: Hernández, Vecchietti, Montagna, Corsano, Fumero, Melchiori, Tibaldo, Vanzetti, Ackermann
* Temas:
* Presentación y distribución de nuevos modelos a implementar

---

Miércoles 2023/7/19
* Duración: 180 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Troubleshooting

Jueves 2023/7/27
* Duración: 180 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Troubleshooting
* test, VM

Viernes 2023/8/25
* Duración: 60 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Revision final de app1
* Discusión sobre optimización de código y como mejorar la interfaz del usuario

---

Viernes 2023/9/15
* Duración: 45 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Muestra de avances en apps y estimación de fecha de presentación.

Martes 2023/10/10
* Duración: 45 minutos
* Participantes: Hernández, Hillar, Toniolo, Dalmasso
* Temas:
* Status APIs y UI
* Los APIs se encuentran en un 70% y las UI en 15%.
* Se organizaron reuniones con los dueños de los modelos.

Martes 2023/10/31
* Duración: 180 minutos
* Participantes: Hernández, Vecchietti, Montagna, Corsano, Fumero, Melchiori, Tibaldo, Vanzetti, Hillar, Toniolo, Dalmasso
* Evaluación final y presentación de los modelos implementados.
* Discusión de los resultados obtenidos y próximos pasos.

Anexo II: Diseño físico de colección *job* según aplicación

A continuación se presentan los diagramas de diseño físico de la colección *job* para las aplicaciones *App2, App3* y *App4*.

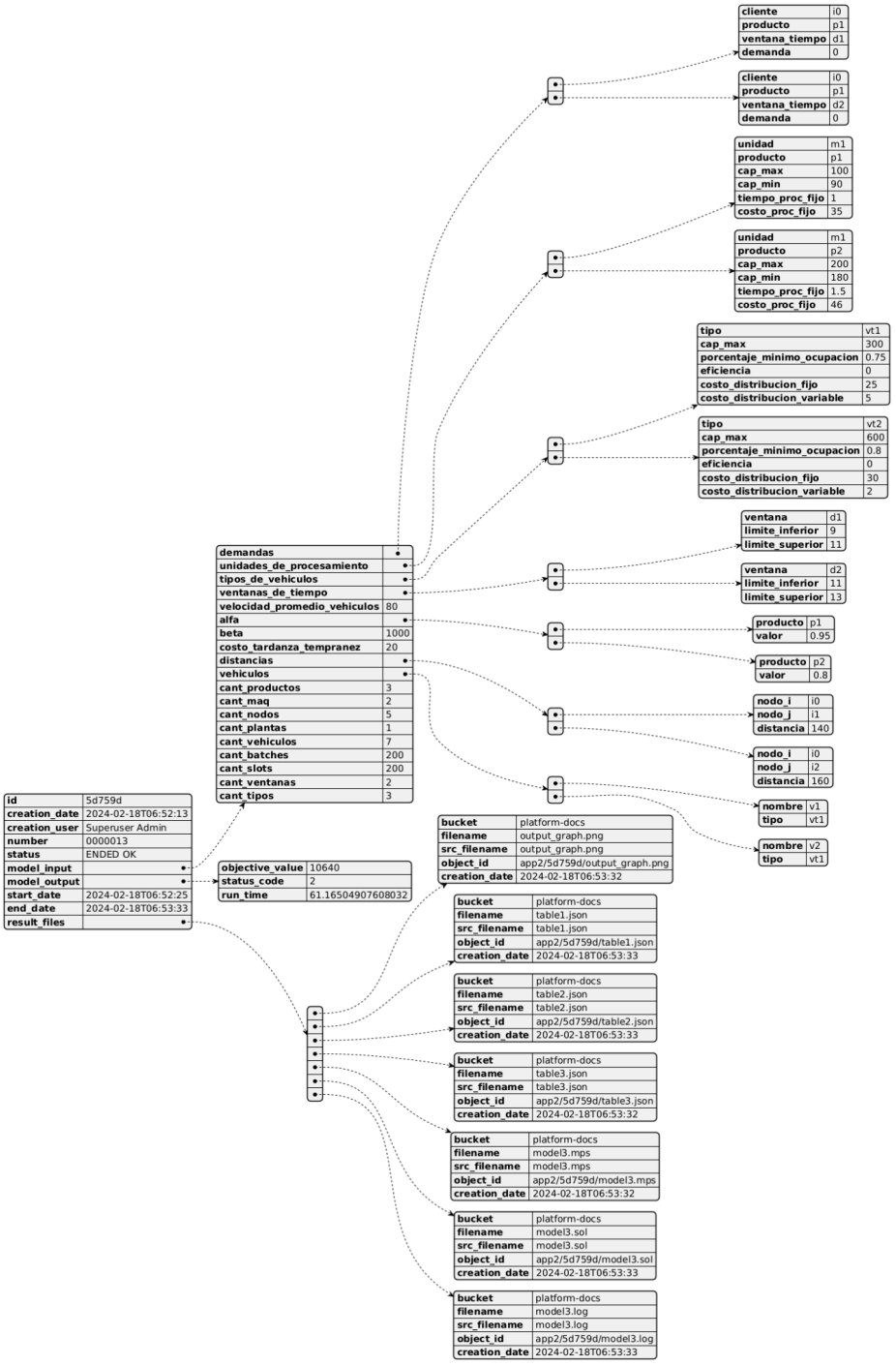


Figura 65. *Ejemplo de documento en puedb.app2.job.*

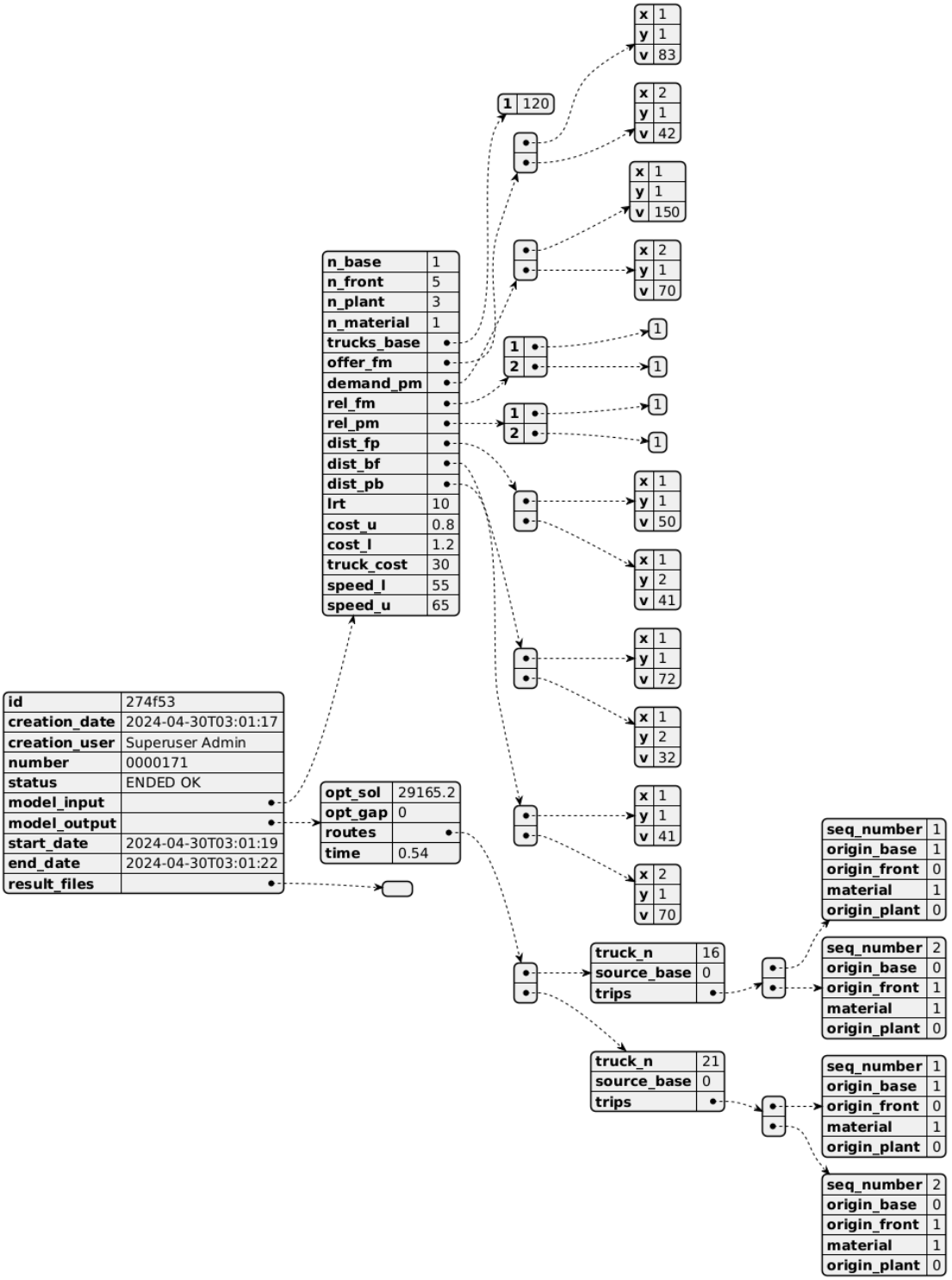


Figura 66. *Ejemplo de documento en puedb.app3.job.*

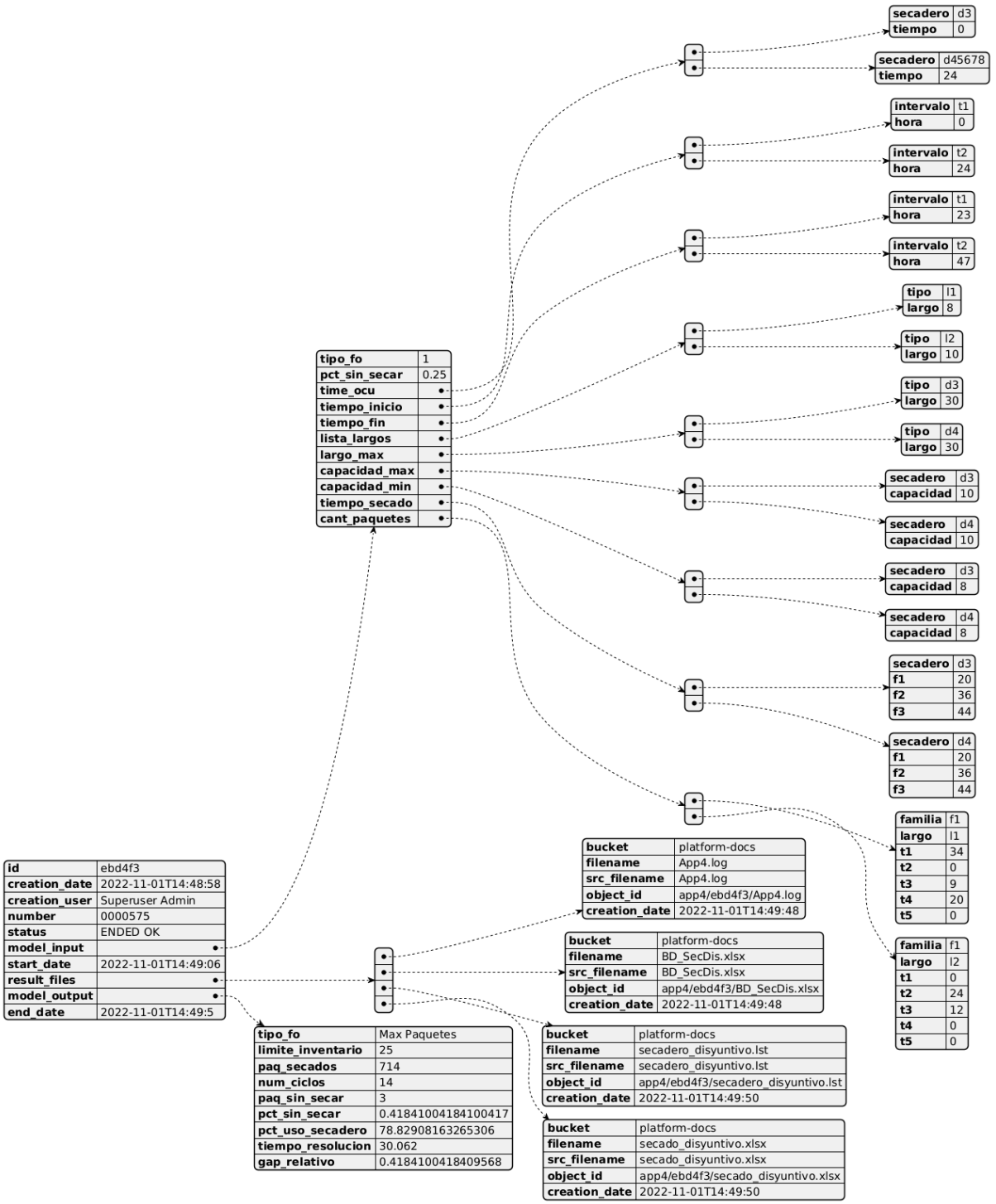


Figura 67. *Ejemplo de documento en puedb.app4.job.*