Pérez, Leandro Leonel

Implementación de módulo para gestión de órdenes de despacho

2024

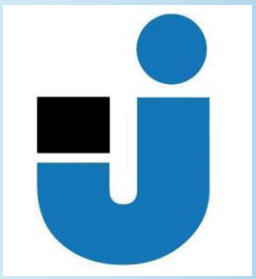
Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática



PRÁCTICA PROFESIONAL SUPERVISADA
Informe final

Implementación de módulo para gestión de órdenes de despacho

Leandro Leonel Pérez

Florencio Varela, diciembre 2024

ESTUDIANTE

Apellido y Nombres: Perez, Leandro Leonel
Correo electrónico: leanperez18@gmail.com

ORGANIZACIÓN DONDE SE REALIZA LA PRÁCTICA PROFESIONAL SUPERVISADA

Nombre o Razón social: Abstract Solutions SRL
Dirección: Av. Santa Fe 1378, Martínez, Buenos Aires
Teléfono: 1150245781
Sector: Desarrollo de Software

TUTOR ORGANIZACIONAL

Apellido y Nombres: Ing. Matías Luzuriaga
Correo electrónico: mluzuriaga@abstractsolutions.com.ar

DOCENTE SUPERVISOR

Apellido y Nombres: Dr. Ing. Martín Morales
Correo electrónico: martin.morales@unaj.edu.ar

DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS

Apellido y Nombres: Prof. Lavigna, Lía
Correo electrónico: llavigna@unaj.edu.ar

COORDINADOR DE LA CARRERA INGENIERIA EN INFORMÁTICA

Apellido y Nombres: Dr. Ing. Martín Morales
Correo electrónico: martin.morales@unaj.edu.ar

1. Resumen

En el presente trabajo se propone el desarrollo de un módulo backend para la gestión de órdenes de despacho para un MRP, diseñado como parte de un sistema web basado en microservicios. El proyecto fue realizado para la empresa Krah, dedicada a la fabricación de tuberías termoplásticas de gran diámetro para la conducción y almacenamiento de fluidos, con el objetivo de optimizar los procesos relacionados con la planificación y envío de productos. El módulo busca aportar mayor trazabilidad, precisión en las entregas y mejor coordinación operativa.

A lo largo del documento, se detalla el proceso del ciclo de desarrollo, desde la planificación, implementación y resultados obtenidos durante la Práctica Profesional Supervisada (PPS), se emplean tecnologías como Java, Spring Boot, Docker y Git. Además, se presentan diagramas UML, casos de uso y capturas de pantalla que ilustran las funcionalidades desarrolladas. Este proyecto pone en práctica metodologías ágiles como Scrum para la coordinación de tareas, garantizando una planificación integral con el equipo.

2. Abstract

This work proposes the development of a backend module for the management of dispatch orders for an MRP, designed as part of a web system based on microservices. The project was carried out for the company Krah, dedicated to the manufacture of large diameter thermoplastic pipes for the conduction and storage of fluids, with the aim of optimizing the processes related to the planning and shipment of products. The module seeks to provide greater traceability, delivery accuracy and better operational coordination.

Throughout the document, the development cycle process is detailed, from planning, implementation and results obtained during the Supervised Professional Practice (PPS), using technologies such as Java, Spring Boot, Docker and GitLab. In addition, UML diagrams, use cases and screenshots that illustrate the developed functionalities are presented. This project implements agile methodologies such as Scrum for the coordination of tasks, ensuring a comprehensive planning with the team.

3. Dedicatoria y Agradecimientos

Este trabajo representa no solo el cierre de una etapa académica muy importante, sino también el resultado del esfuerzo, la dedicación y el apoyo recibido a lo largo del proceso. Por ello, quiero dedicar este trabajo a todas las personas que hicieron posible que haya llegado a esta instancia.

En primer lugar, le quiero agradecer a mi familia y amigos que siempre estuvieron a mi lado, por el apoyo incondicional, compresión y respaldo. Especialmente a mi madre quien, con su ejemplo, motivación constante y palabras de aliento, me impulsó a superarme y esforzarme cada día más. Sin ella esto no habría sido posible.

A mis compañeros, profesores de la carrera y a la UNAJ, que no solo compartieron conmigo su conocimiento y experiencia, sino también su tiempo y dedicación, ayudándome a crecer profesional y personalmente.

También, quiero agradecer a Abstract Solutions por confiar en mí y brindarme la oportunidad de realizar mi Práctica Profesional Supervisada. Trabajar en un ambiente tan dinámico y profesional me permitió aplicar los conocimientos adquiridos durante la carrera y, a su vez, pude adquirir nuevas habilidades y experiencias valiosas para el futuro. Quiero destacar el papel del Ingeniero Matías Luzuriaga, mi tutor, quien me guió durante todo el proceso con paciencia y profesionalismo.

A todos los mencionados, expreso mi más sincero agradecimiento por ser parte de este camino.

4. Índice

1. Resumen
2. Abstract
3. Dedicatoria y Agradecimientos
4. Índice
5. Índice de figuras
6. Introducción
7. Objetivos
8. Tareas a ejecutar
9. Cronograma
10. Desarrollo
    - 10.1. Descripción del MRP
    - 10.2. Módulo gestión de órdenes de despacho
    - 10.3. Tecnologías
        - 10.3.1. Java
        - 10.3.2. Spring Boot
        - 10.3.3. Microservicios
        - 10.3.4. Docker
            - 10.3.4.1. Plataforma Docker
            - 10.3.4.2. ¿Para qué utilizar Docker?
            - 10.3.4.3. Arquitectura Docker
            - 10.3.4.4. Imágenes
            - 10.3.4.5. Contenedores
        - 10.3.5. MinIO
        - 10.3.6. Portainer
        - 10.3.6.1 Arquitectura Portainer
        - 10.3.7. Git
        - 10.3.8. Maven
        - 10.3.9. PostgreSQL
        - 10.3.10. Jenkins
        - 10.4.11. GitLab
    - 10.4. Definición Requerimientos
    - 10.5. Desarrollo API
        - 10.5.1. Metodologías ágiles

10.5.2. Scrum
10.5.3. Flujo de trabajo
10.5.4. Diagrama de componentes del Sistema
10.5.5. Diagrama UML
10.5.6. Casos de uso
10.6. Resultados
10.6.1. Pantalla principal
10.6.2. Módulo Transportista
10.6.2.1. Alta Transportista
10.6.2.2. Visualización lista transportista
10.6.2.3. Edición Transportista
10.6.2.4. Eliminar Transportista
10.6.3. Módulo Despacho
10.6.3.1. Alta Órdenes de Despacho
10.6.3.2. Filtros y calendario
10.6.3.3. Actualizar orden de despacho
10.6.3.4. Descarga orden de carga
10.7. Documentación
10.8. Diagrama de Gantt
11. Conclusiones
12. Reflexión sobre la Práctica Profesional Supervisada como espacio de formación
13. Bibliografía

5. Índice de figuras

Figura 1: Modelo Arquitectura Monolítica vs Microservicios
Figura 2: Arquitectura de Docker
Figura 3: Arquitectura de Portainer
Figura 4: Flujo de Git
Figura 5: CI/CD Pipeline
Figura 6: Ciclo del Sprint
Figura 7: Flujo de trabajo con Git
Figura 8: Diagrama de componentes del sistema
Figura 9: Diagrama UML Módulo Órdenes de Despacho
Figura 10: Casos de uso Transportistas

Figura 11: Casos de uso Visualización Órdenes
Figura 12: Pantalla principal MRP
Figura 13: Pantalla módulo Transportista
Figura 14: Formulario Alta Transportista
Figura 15: Lista paginada de Transportistas
Figura 16: Formulario edición Transportista
Figura 17: Transportista editado
Figura 18: Confirmación eliminación Transportista
Figura 19: Pantalla módulo Órdenes de Despacho
Figura 20: Vista calendarizada órdenes de despacho
Figura 21: Selección Tipo de Tuberías y cantidad
Figura 22: Lista de SKU
Figura 23: Tipo de Tubería cargada
Figura 24: Modal selección de cliente y PEC
Figura 25: Modal datos de envío
Figura 26: Confirmación creación de orden
Figura 27: Lista órdenes
Figura 28: Órdenes con distintos estados
Figura 29: Filtro de búsqueda
Figura 30: Filtro de estados
Figura 31: Filtro de fechas
Figura 32: Vista calendarizada
Figura 33: Lista de órdenes por día
Figura 34: Detalle orden de despacho
Figura 35: Edición orden de despacho
Figura 36: Confirmación edición orden
Figura 37: Orden actualizada
Figura 38: Estado orden actualizada
Figura 39: Acción descarga de orden
Figura 40: Orden de carga
Figura 41: Endpoints órdenes de despacho
Figura 42: Endpoints transportistas
Figura 43: Endpoint órdenes
Figura 44: Endpoint creación orden
Figura 45: Diagrama de Gantt

6. Introducción

Krah es una empresa dedicada a la fabricación de tuberías termoplásticas de gran diámetro para la conducción de y almacenamiento de fluidos. La compañía busca optimizar sus procesos para satisfacer las demandas del mercado, mejorar la eficiencia operativa y asegurar la calidad en sus entregas.

En este contexto, un sistema de planificación de requerimientos de materiales (MRP) es crucial para optimizar la producción y la gestión de inventarios en la fabricación de tubos. Este sistema permite planificar la producción de manera eficiente, asegurando que los materiales y recursos estén disponibles en el momento adecuado, minimizando costos y tiempos de espera. En la fabricación de tubos, el MRP se enfoca en la programación de la producción, la gestión de inventarios y la coordinación de compras, considerando factores como la demanda del mercado y los plazos de entrega.

Con el MRP establecido, se requiere el desarrollo de un módulo para gestionar las órdenes de despacho. Este módulo permitirá:

- Registro de órdenes: facilitar la entrada de nuevas órdenes de despacho, vinculados directamente con las órdenes de producción del MRP.
- Seguimiento de envíos: proporcionar visibilidad en tiempo real del estado de cada despacho, desde la preparación hasta la entrega.
- Gestión de inventarios: integrar con el MRP para asegurar que los productos despachados correspondan con los niveles de inventario actualizados.
- Reportes y visibilidad: crear informes que permitan analizar la eficiencia de los despachos.

Para la Práctica Profesional Supervisada (PPS) se propone el desarrollo e implementación de una API backend para incorporar un nuevo componente de software dentro de un sistema web, basado en una arquitectura de microservicios.
Se emplearán Java 17 y Spring Boot como framework para llevar adelante el trabajo. Además, se utilizarán herramientas como Maven y GIT para la gestión del proyecto y el control de versiones.

La API desarrollada será parte de un conjunto de microservicios que se comunicarán entre sí para gestionar diversas funcionalidades de la plataforma. El enfoque principal será implementar los servicios REST, la lógica de negocio, el modelado de datos y facilitar la integración con otros servicios, todo ello optimizando el rendimiento del sistema.

7. Objetivos

El objetivo general es desarrollar un nuevo módulo backend que permita gestionar las órdenes de despacho, con el propósito de facilitar el registro de órdenes, seguimiento de envíos, gestión de inventarios y generación de reportes.

A continuación, se listan los objetivos específicos que se pretenden alcanzar con el nuevo módulo:

1. Relevar, detallar y documentar las reglas de negocio necesarias para el posterior desarrollo de la API.
2. Diseñar, documentar con UML y presentar una propuesta de la solución a implementar.
3. Participar en la definición de una estrategia de implementación y buenas prácticas.
4. Definir y documentar la interfaz de la API para luego ser consumida por el frontend.
5. Definir y exponer el trabajo necesario para asegurar la calidad de la implementación.
6. Presentar el funcionamiento y desarrollar un manual de programador.

8. Tareas a ejecutar

A continuación, se detallan las tareas identificadas a ejecutar para poder cumplir con los objetivos específicos:

1. Análisis de requerimientos: reunión con el equipo, revisión de especificaciones, y análisis del sistema.
2. Diseño de la arquitectura del módulo: estructura interna, base de datos y planificación de APIs.
3. Desarrollo del módulo backend: implementación de servicios REST, integración con base de datos.
4. Integración con otros microservicios y pruebas de comunicación.
5. Pruebas unitarias e integración: validación de la funcionalidad interna del módulo.
6. Pruebas de rendimiento: optimización del rendimiento del módulo.
7. Documentación técnica y ajustes finales: redacción de la documentación y revisión del código.
8. Revisión final y entrega del módulo: últimas pruebas y despliegue en el entorno de producción.

9. Cronograma

Tarea/Semana	1	2	3	4	5	6	7	8	9	10	11	12
1	x											
2		x	x									
3				x	x	x						
4							x	x				
5									x			
6										x		
7											x	
8												x

10. Desarrollo

10.1. Descripción del MRP

El sistema de planificación de requerimiento de materiales o Material Requirements Planning (MRP) posee un circuito de fabricación que consta de 5 procesos claves:

- Conformado: proceso por el cual se conforma una tubería. Se utiliza un mandril (molde) y mediante un proceso de extrusión de plástico se fabrica el tubo.
- Enfriamiento: el conformado del tubo se realiza a altas temperaturas ya que se debe derretir el plástico para poder extruirlo en la conformadora. Es por esto, por lo que existe el proceso de enfriamiento en el cual se deja de enfriar a la tubería hasta la temperatura óptima de torneado y desmandrilado.
- Torneado: en este proceso se utiliza un torno para limar las puntas de las tuberías una vez llegado a la temperatura óptima.

- Desmandrilado: en esta etapa, se separa el tubo de su mandril (molde) y se realizan mediciones de calidad.
- Terminación: en esta fase, se le hacen modificaciones de estética y terminación para dejarlos listos para entregar.

Un operario se encarga de ejecutar las distintas fases del circuito de producción, iniciándolas progresivamente. Durante este proceso, tiene la capacidad de contabilizar el tiempo, registrar fallas y reiniciar cualquier fase si es necesario. Esto nos permite realizar un seguimiento detallado de todo el proceso productivo.

Además, el sistema cuenta con diferentes módulos, tales como:

- Programación y Producción: este módulo se utiliza para la planificación de la producción y el seguimiento de este.
- SCRAP (Desperdicio): este módulo se utiliza para gestionar todo el SCRAP que va generando el proceso productivo.
- Tuberías: en este módulo, se dan de alta los diferentes tipos de tuberías. Se llena un formulario con especificaciones técnicas que tiene cada tubería para darlo de alta en el sistema.
- Normas: se dan de alta las normas de fabricación bajo las cuales se puede fabricar una tubería.
- Cliente: se dan de alta clientes con datos básicos para la operación.
- Perfiles: Se genera un ABM (Alta, Baja, Modificación) de perfiles de tuberías. Se genera un formulario para dar de alta un perfil específico y luego asignarlo a una tubería a la hora de crearla.
- PEC: un Pedido de Ejecución de Contrato (PEC) es un pedido de fabricación de tuberías, se definen fechas de entrega, se le asigna un cliente y necesita de aprobaciones de otras áreas para validar el pedido.

10.2. Módulo gestión de órdenes de despacho

El nuevo módulo por implementar está enfocado en la gestión de órdenes de despacho. Este módulo permitirá realizar las siguientes acciones principales:

1. Alta de Transportistas: los usuarios podrán dar de alta a transportistas para asociarlos a las órdenes de despacho.

2. Visualización de Órdenes de Despacho: se habilitará una interfaz donde los usuarios podrán consultar las órdenes de despacho existentes, visualizando su estado actual en tiempo real.

3. Alta de Órdenes de Despacho: los usuarios podrán generar nuevas órdenes de despacho, lo que implica:

    a. Seleccionar un cliente y el PEC correspondiente.
    b. Asociar los tubos hechos que serán despachados.
    c. Completar la información de envío necesaria, como dirección y detalles del transportista.

4. Asignación de Estado de la Orden: dependiendo de la cantidad de datos completados al momento de la creación, la orden de despacho se generará con un estado específico, que reflejará si está pendiente de completar información o lista para ser procesada.

5. Actualización y Despacho de la Orden: una vez creadas, las órdenes podrán ser actualizadas con los datos faltantes. Cuando toda la información esté completa, la orden podrá ser despachada, cambiando su estado a "despachada".

Este módulo garantizará un mayor control y trazabilidad en el proceso de despacho, desde la creación hasta la finalización del envío.

10.3. Tecnologías

Para realizar la PPS se hará uso de varias tecnologías, entre las cuales se encuentran:

10.3.1. Java

Java es un lenguaje de programación orientado a objetos ampliamente utilizado en el mercado laboral. Es seguro, confiable, multiplataforma y de rápida programación, se puede utilizar para aplicaciones web, móviles y desarrollo de software empresarial.

Creado hace más de 20 años, hoy en día sigue siendo uno de los lenguajes más requeridos y populares. Además, posee ventajas como las siguientes:

- Lenguaje de alto nivel: esto significa que el lenguaje está orientado a que sea más parecido al lenguaje humano, permitiéndonos facilidad a la hora de escribir, leer y mantener el código.
- Estabilidad: el lenguaje continúa lanzando nuevas versiones mejoradas cada cierto periodo de tiempo.
- Portable: la portabilidad de plataformas permite ejecutar el código en cualquier plataforma que soporte JVM (Java Virtual Machine).
- Orientado a objetos: este paradigma permite estructurar y organizar el código a partir de objetos que representa entidades u conceptos del mundo real.
- Seguridad: java proporciona seguridad como parte integral de su diseño.
- Robusto: el compilador de Java es capaz de detectar errores en el código, además permite el manejo de excepciones (errores en tiempo de ejecución).

10.3.2. Spring Boot

Spring Boot es una herramienta que facilita la creación de aplicaciones web basadas en Java, proporciona una configuración predeterminada y automatizada. Su objetivo es reducir la complejidad del desarrollo al quitar ciertos aspectos comunes del desarrollo en las aplicaciones, tales como, la conexión de la base de datos, la seguridad, el servidor y otros componentes.

Entre sus características más destacadas se encuentran:

- Arquitectura modular: Gracias a la arquitectura modular de Spring, Spring Boot facilita la integración con otros módulos.
- Contenedor Embebido: Uno de los motivos por lo cual Spring Boot acelera el desarrollo es gracias a que contiene un servidor embebido (Tomcat), por lo cual, no es necesario desplegar la aplicación en un servidor externo, acelerando el ciclo de desarrollo.

- Configuración Automática: Spring Boot tiene la capacidad de detectar qué dependencias se incluyen en el proyecto y configurar los componentes necesarios para su correcto funcionamiento.
- Desarrollo rápido: la resolución de configuraciones complejas por parte de Spring Boot, hace posible que los desarrolladores se enfoquen en la lógica de negocio y no en configuraciones complejas.

Spring Boot es una extensión o módulo de Spring Framework, este último es un ecosistema de aplicaciones basadas en Java, entre sus proyectos más populares se encuentran Spring Data, Spring Security, Spring Cloud, entre otros.

Una de las características más importantes de Spring Framework es la inserción de dependencias (DI) que se usa para lograr la inversión de control (IoC). Este principio de diseño delega en el framework el control de la creación de los objetos y dependencias en lugar del propio desarrollador. Un ejemplo de lo expresado sería una analogía simple como la de un restaurante tradicional (sin IoC), el cliente tiene que cocinar su propia comida (control directo). En un restaurante con IoC, el cliente simplemente pide lo que quiere y el chef (el framework) se encarga de preparar la comida y entregarla al cliente.

10.3.3. Microservicios

Los microservicios son un enfoque arquitectónico compuesto por pequeños servicios independientes que se comunican entre sí mediante APIS.

Las ventajas de este modelo es la facilidad de escalar los servicios, la velocidad de desarrollos, la reutilización y la libertad en la elección de tecnologías.

Los microservicios surgieron como reemplazo a las arquitecturas monolíticas, este tipo de arquitectura se basa en un servicio grande que se ejecuta constantemente en un servidor físico o virtualizado. El hardware se vuelve un factor fundamental, ya que la disponibilidad y escalamiento de la aplicación depende enteramente de este.

Los microservicios, por el contrario, pueden operar varias instancias en un solo servidor o en varios servidores, a medida que los recursos escalan de forma dinámica para admitir las demandas de la carga de trabajo. Los microservicios individuales suelen estar en contenedores para mejorar la portabilidad y la escalabilidad.

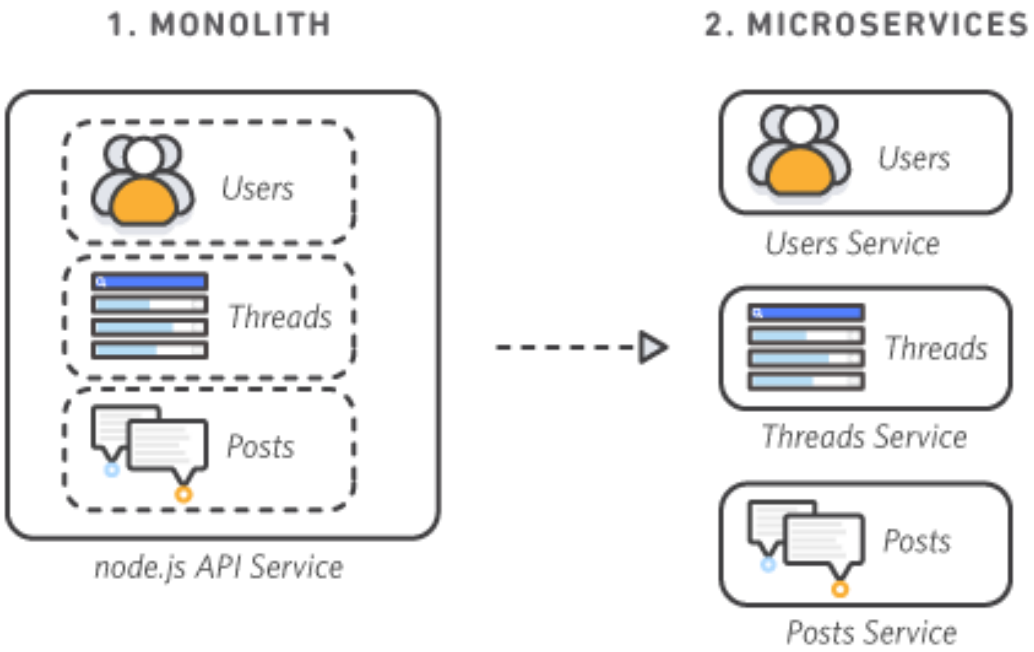


Figura 1: Modelo Arquitectura Monolítica vs Microservicios

Fuente: Recuperado de https://aws.amazon.com/es/microservices/

Como beneficios se pueden nombrar:

- **Escalado sencillo**: los microservicios permiten que cada servicio se escale de forma independiente para satisfacer la demanda de la característica de la aplicación que respalda. Esto permite a los equipos adecuarse a las necesidades de la infraestructura, medir con precisión el costo de una característica y mantener la disponibilidad si un servicio experimenta un aumento en la demanda.
- **Implementación sencilla**: los microservicios permiten la integración y la entrega continua, lo que facilita probar nuevas ideas y revertirlas si algo no funciona. El bajo costo de los errores permite experimentar, facilita la actualización del código y acelera el tiempo de comercialización de las nuevas características.
- **Libertad tecnológica**: al ser independientes, cada servicio puede ser desarrollado por un stack tecnológico diferente, dependiendo del problema y las soluciones que pueden brindar las diferentes tecnologías.
- **Código reutilizable** dividir el software en pequeños módulos definidos permite a los equipos reutilizar funciones para distintos propósitos. Un servicio creado para

una tarea puede servir como base para otra, lo que facilita que una aplicación crezca sin tener que escribir todo el código desde cero.

- **Resistencia**: en un sistema de microservicios, un error en uno de los servicios no afecta al sistema entero, las aplicaciones pueden manejar el degradado de funcionalidad, pero no bloquea todo el sistema.

10.3.4. Docker

Docker es una plataforma abierta que permite el desarrollo, distribución y ejecución de aplicaciones. Facilita la separación entre infraestructura y aplicación, acelerando la entrega de software.

Con Docker, se puede gestionar la infraestructura de una manera similar a la que manejan las aplicaciones, lo que reduce el tiempo entre el desarrollo y despliegue a producción.

10.3.4.1. Plataforma Docker

Docker permite empaquetar y ejecutar aplicaciones en entornos aislados llamados contenedores. Estos contenedores tienen la característica de ser ligeros, contienen lo necesario para que la aplicación se pueda ejecutar y pueden compartirse de forma que se garantiza su consistencia en diferentes entornos. Además, otorga la posibilidad de gestionar el ciclo de vida de los contenedores, desde el desarrollo hasta producción, en sus diferentes entornos, sean locales, proveedores de procesamiento en la nube o híbridos.

10.3.4.2. ¿Para qué utilizar Docker?

Para una entrega rápida y consistente de aplicaciones, facilita la creación de entornos estandarizados. También, es ideal para entornos de integración y entrega continua.

10.3.4.3. Arquitectura Docker

Docker utiliza una arquitectura cliente-servidor. El cliente Docker se comunica con el Docker Deamon, quien se encarga de realizar el trabajo pesado de construir, ejecutar y distribuir los contenedores. El cliente y el deamon pueden trabajan en el mismo sistema o el cliente se puede comunicar con un deamon de manera remota.

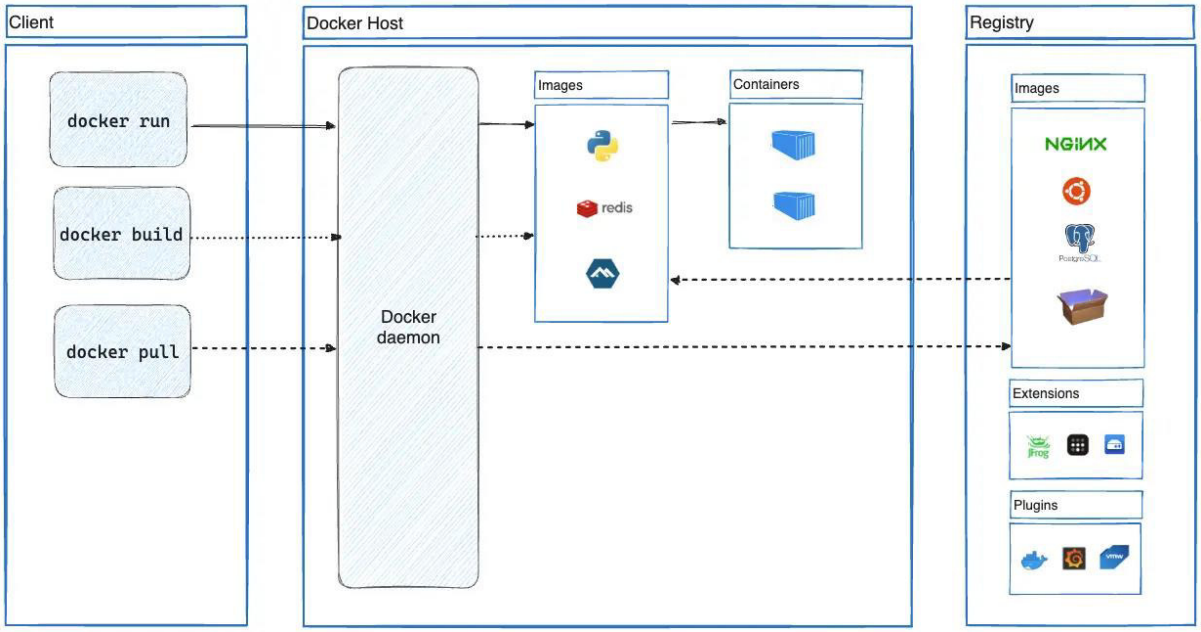


Figura 2: Arquitectura de Docker

Fuente: Recuperado de https://docs.docker.com/get-started/docker-overview/

10.3.4.4. Imágenes

Una imagen de Docker es un archivo ejecutable e independiente que se utiliza para crear un contenedor de Docker. Dentro de esta imagen se encuentran las bibliotecas, dependencias y archivos necesarios para que el software se ejecute, incluyendo el código de la aplicación.

Características de una imagen:

- Se utiliza como plantilla para crear contenedores.
- Define la configuración de un contenedor.
- Es un archivo de solo lectura.

10.3.4.5. Contenedores

Un contenedor de Docker es una unidad estandarizada de software que incluye todo lo necesario para ejecutar una aplicación, como el código fuente, bibliotecas, configuración, herramientas del sistema.

Las ventajas de utilizar contenedores son los siguientes:

- Portabilidad: los contenedores pueden ejecutarse en cualquier máquina que tenga Docker instalado.
- Isolación: cada contenedor se ejecuta en un espacio aislado, evitando conflictos entre dependencias y permite flexibilidad de desarrollo.
- Eficiencia: los contenedores son livianos a comparación de los hipervisor que requieren las máquinas virtuales tradicionales.
- Escalabilidad: se pueden ejecutar múltiples contenedores en la misma máquina.

10.3.5. MinIO

MinIO es una plataforma de almacenamiento en la nube compatible con Amazon S3. Esta plataforma de alta performance puede almacenar objetos como fotos, videos, imágenes de contenedores, archivos con formato pdf, csv, etc. y está pensado para que se pueda desplegar tanto en nubes públicas o privadas, entornos orquestados e infraestructura Edge.

Los requisitos previos para utilizar MinIO en un despliegue contenerizado son los que se detallan a continuación:

- Tener Podman o Docker instalado.
- Acceso de lectura, escritura y eliminación a la carpeta o unidad de almacenamiento para el volumen persistente.

10.3.6. Portainer

Portainer es una plataforma de ligera de prestaciones de servicios para aplicaciones contenerizadas, oculta la complejidad de manejar contenedores mediante una interfaz de usuario (UI). Esta UI permite la orquestación de los diferentes recursos (imágenes, contenedores, volúmenes, redes y más).

10.3.6.1. Arquitectura Portainer

Para entender cómo funciona la arquitectura, se necesitan conocer 2 conceptos: Portainer Server y Portainer Agent.

Ambos son contenedores ligeros que existen en la infraestructura de contenedores de la aplicación. Cada Portainer Agent debe desplegarse en cada nodo del cluster (conjunto de nodos) y reportarse ante el Portainer Server.

Una sola instancia del Portainer Server puede aceptar conexiones de múltiples Agent, permitiendo un manejo centralizado de clusters desde una UI.

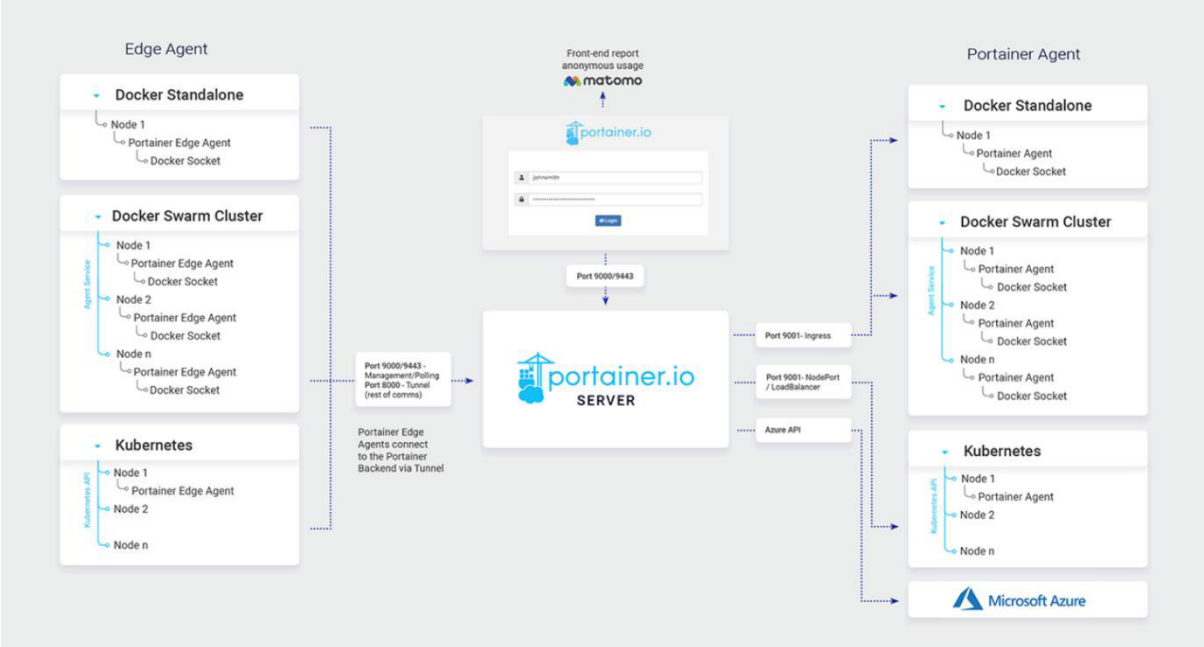


Figura 3: Arquitectura de Portainer

Fuente: Recuperado de https://docs.portainer.io/start/architecture

10.3.7. Git

Git es un software de control de versiones distribuido. Es sumamente conocido por su eficiencia, flexibilidad y capacidad de gestionar proyectos de cualquier tamaño.

Git permite llevar un registro de los cambios de código del proyecto, facilitando la gestión de diferentes versiones del proyecto, en caso de ser necesario se puede volver a una versión anterior. También, favorece el trabajo colaborativo entre desarrolladores, distintos colaboradores pueden trabajar en el mismo proyecto sin estar sobrescribiendo el trabajo del otro.

Git maneja tres áreas principales:

- Directorio de trabajo (working directory): es donde se trabaja y modifican los archivos.
- Área de preparación (staging area): es un área temporal donde se almacenan los archivos antes de ser confirmados.
- Repositorio local: es la base de datos donde Git guarda de forma permanente las confirmaciones (commits). Cada vez que se hace un commit, los archivos confirmados pasan al repositorio local.

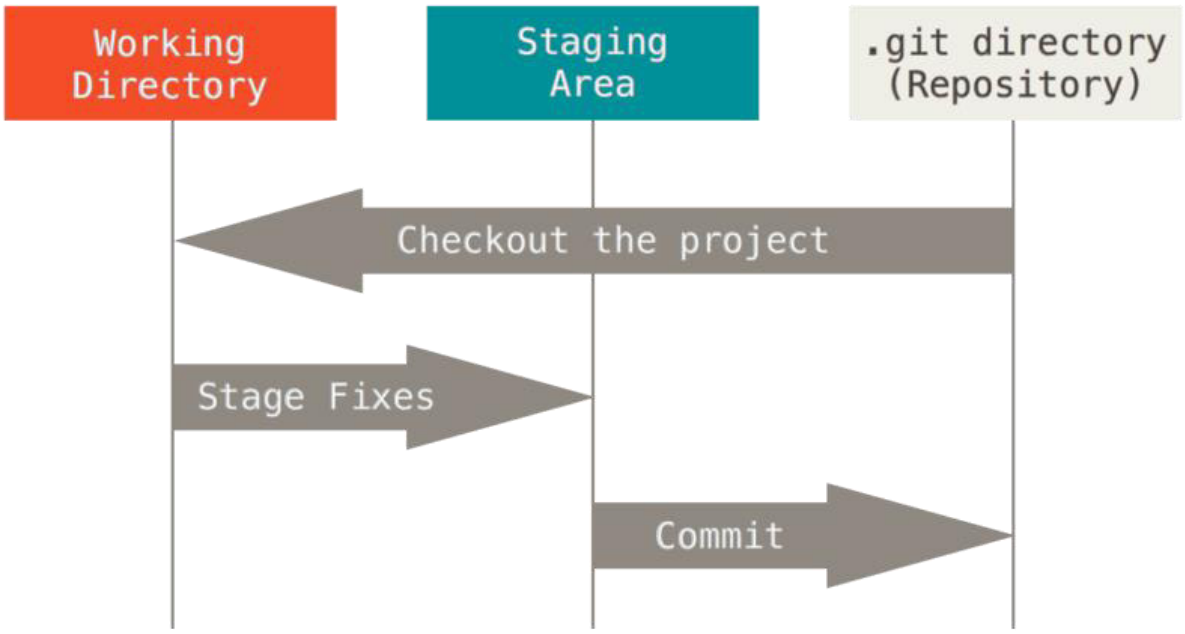


Figura 4: Flujo de Git

Fuente: Recuperado de https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git

10.3.8. Maven

Maven es una herramienta de gestión y comprensión de proyectos desarrollada por Apache. Es muy utilizada en el ecosistema de Java y agrega una gran cantidad de utilidades y beneficios a un proyecto. A continuación, se describen algunas de las principales utilidades y características que Maven aporta a un proyecto:

1. **Gestión de Dependencias**: Maven permite declarar y gestionar las dependencias del proyecto de manera sencilla en el archivo `pom.xml`. Esto facilita la inclusión de bibliotecas externas sin necesidad de descargarlas y configurarlas manualmente.

2. **Construcción del Proyecto**: Maven automatiza el proceso de compilación, ejecución de pruebas y empaquetado del proyecto en artefactos como JARs o WARs. Esto asegura que los proyectos se construyan de manera consistente y reproducible.

3. **Estructura Estándar del Proyecto**: Maven promueve una estructura estándar para proyectos Java, lo que facilita la organización y el mantenimiento del código. La estructura típica de un proyecto Maven incluye directorios como ‘src/main/java’, ‘src/test/java’, ‘src/main/resources’, entre otros.

4. **Gestión del Ciclo de Vida**: Maven define un ciclo de vida de construcción, que incluye fases como ‘validate’, ‘compile’, ‘test’, ‘package’, ‘verify’, ‘install’, y ‘deploy’. Esto permite controlar el proceso de construcción y agregar tareas personalizadas en puntos específicos del ciclo de vida.

5. **Plugins**: Maven tiene un sistema extensible de plugins, que permite agregar funcionalidades adicionales al proceso de construcción. Existen plugins para tareas comunes como la generación de documentación, la creación de informes de cobertura de pruebas, la ejecución de servidores de aplicaciones, entre otros.

6**. Integración con Sistemas de CI/CD**: Maven se integra fácilmente con sistemas de integración continua (CI) y entrega continua (CD) como Jenkins, Travis CI, GitLab CI, y otros.

7. **Dependencias Transitivas**: Maven resuelve automáticamente las dependencias transitivas, es decir, si el proyecto depende de una biblioteca que a su vez depende de otras bibliotecas, Maven se encargará de resolver y descargar todas esas dependencias.

8. **Repositorios de Artefactos**: Maven utiliza repositorios locales y remotos para almacenar y compartir artefactos. El repositorio central de Maven (Maven Central) contiene una gran cantidad de bibliotecas de uso común, que pueden ser incluidas fácilmente en el proyecto.

9. **Facilidad de Configuración**: el archivo `pom.xml` centraliza la configuración del proyecto, lo que facilita la gestión y el mantenimiento de las dependencias, plugins, y otras configuraciones del proyecto.

10.3.9. PostgreSQL

PostgresSQL es un sistema de gestión de base de datos relacionales de objetos (ORDBMS, por sus siglas en ingles), esto quiere decir, que combina el modelo de bases de datos relacionados y el modelo orientado a objetos. Como una base de datos tradicional que utiliza tablas y relaciones para persistir y recuperar datos, también admite la definición de objetos y clases, herencia, polimorfismo y encapsulamiento, típico de los modelos de bases de datos orientados a objetos.

Una de las características que ofrece es el cumplimiento de las propiedades ACID (Atomicity, Consistency, Isolation, Durability) para las transacciones, lo que garantiza la integridad y fiabilidad de los datos.

Las transacciones habitualmente se componen de múltiples declaraciones, la atomicidad (atomicity) garantiza que cada transacción será tratada como una sola unidad, en donde si una transacción falla, todas fallarán, de misma forma si los resultados son exitosos. La consistencia (consistency) asegura que los datos pasan de un estado válido a otro, los datos deben ser validos según las reglas de la base de datos. Para los casos que las transacciones se ejecuten de manera concurrente, el aislamiento (isolation) asegura que las transacciones serán manejadas como si se estuvieran ejecutando de manera secuencial. La durabilidad (durability) garantiza que los datos permanecerán persistidos para siempre, hasta en caso de fallas en el sistema.

Ventajas:

- Es gratuito y de código abierto, lo que reduce los costos y permite la modificación y personalización del código, según las necesidades específicas de cada proyecto.
- Es escalable y seguro, lo que lo hace adecuado para proyectos grandes y complejos.
- Tiene una comunidad activa y una documentación extensa, lo que facilita su implementación y mantenimiento.

10.3.10. Jenkins

Jenkins es un software de automatización de tareas de código libre escrito en Java, se utiliza para implementar flujos de integración continua/ entrega continua (CI/CD), a este flujo se le llama pipeline.

Los pipelines se encargan de automatizar las pruebas y reportar cambios en tiempo real sobre el código fuente.

Jenkins provee una plataforma flexible y extensa para automatizar tareas a través de su arquitectura de plugin. Se logra integrar con varias tecnologías como:

- Versiones de control (Git).
- Herramientas de construcción de proyectos (Maven, Gradle).
- Testing (Junit).
- Herramientas de despliegue (Docker, Kubernetes).

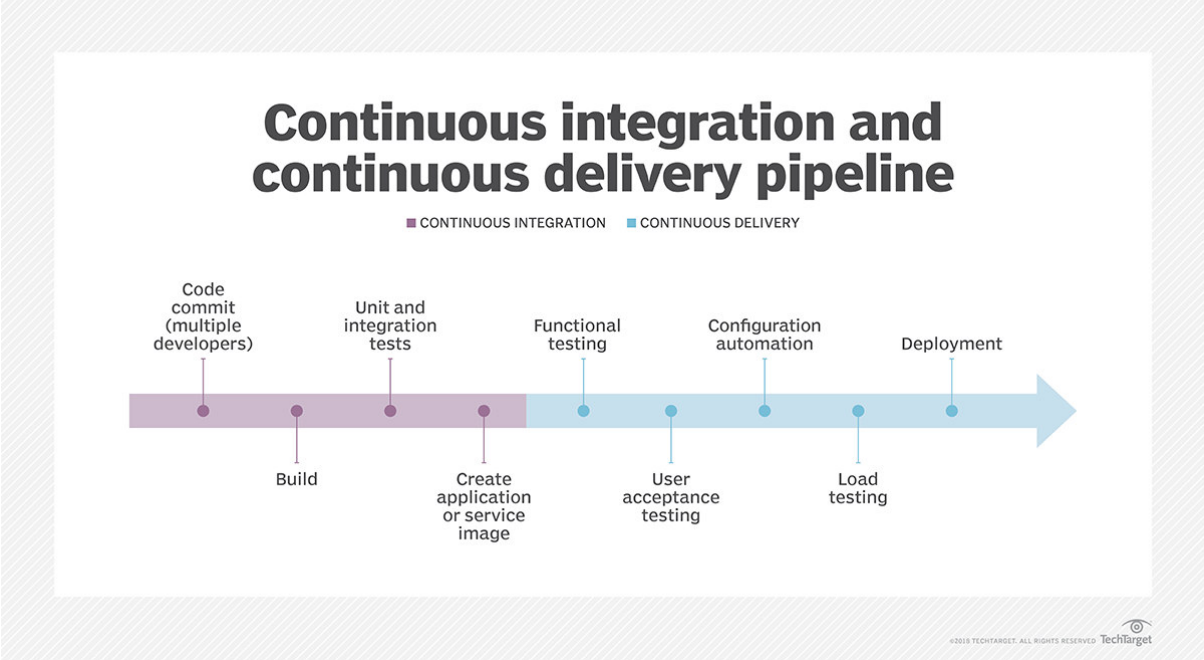


Figura 5: CI/CD Pipeline

Fuente: Recuperado de https://www.techtarget.com/searchsoftwarequality/definition/Jenkins

Los pipelines se pueden dividir en dos partes, integración continua (CI) y entrega continua (CD):

- CI: En esta etapa, los desarrolladores añaden nuevo código al proyecto. La integración continua detecta errores y defectos antes de integrar el código nuevo.
- CD: En esta etapa posterior a la CI, se automatiza la creación y empaquetado del proyecto, para su despliegue en un entorno de prueba o producción.

10.3.11. GitLab

GitLab es una plataforma de gestión de proyectos y control de versiones que se utiliza para el desarrollo colaborativo de software. Ofrece una amplia gama de características y herramientas para gestionar el ciclo de vida del desarrollo, desde la creación de proyectos hasta la entrega final, y es compatible con Git. Es una de las opciones más populares y útiles en la industria del software para equipos de desarrollo y programadores.

10.4. Definición Requerimientos

Como primera tarea, se realizó una serie de reuniones con el equipo para entender la problemática y definir las tareas necesarias a llevar a cabo.

A continuación, se listan las tareas identificadas:

KRH1 – Alta transportista.

Descripción: Como usuario debo ser capaz de registrar un nuevo Transportista. Para dar de alta un transportista, se debe completar un formulario que incluya los siguientes datos obligatorios:

- Nombre
- CUIT
- Razón Social

Prioridad: Media

Criterios de Aceptación:

- El formulario de registro de transportistas debe ser accesible solo para usuarios autorizados.
- Al guardar los datos, el sistema debe verificar la unicidad del número de CUIT y mostrar un mensaje de error en caso de duplicidad.
- El transportista debe quedar registrado en la base de datos y estar disponible para su selección en la creación de órdenes de despacho.

KRH2 – Edición Transportista.

Descripción: Como usuario debo ser capaz de editar los datos de un Transportista previamente registrado.

Prioridad: Media

Criterios de Aceptación:

- Los datos modificados deben verse reflejados en la base de datos.

KRH3 – Eliminación Transportista.

Descripción: Como usuario debo ser capaz de eliminar a un Transportista previamente registrado.
Prioridad: Media
Criterios de aceptación:

- La eliminación debe verse reflejado en la base de datos.

KRH4 – Listado de Transportistas.

Descripción: Como usuario debo ser capaz de visualizar una lista de las Transportistas que se encuentran en el sistema.
Prioridad: Media
Criterio de aceptación:

- Se debe observar la información correspondiente a los Transportistas.

KRH5 – Obtener Transportista.

Descripción: Como usuario debo ser capaz de obtener la información de un Transportista.
Prioridad: Media
Criterio de aceptación:

- Se valida que el usuario solicitado existe previamente.
- Se debe observar el nombre, CUIT y razón social.

KRH6 – Alta Órdenes de Despacho.

Descripción: Como usuario debo ser capaz de dar de alta una nueva Orden de Despacho. El usuario debe completar qué Tipo de Tubería y cantidad a despachar, cargar el cliente y su PEC y, por último, cargar los datos de envío. Una vez completado estos pasos, se crea una Orden de Despacho con un estado. Los estados posibles son:

- PLANIFICACIÓN
- PROGRAMADOS
- CONFIRMADOS
- DESPACHADO

Prioridad: Alta

Criterio de aceptación:

- Si solo se completa la fecha de entrega en los datos de envío, se crea la orden en estado PLANIFICACIÓN.
- Si se completa el cliente/PEC y se le carga Transportista, se crea la orden en estado PROGRAMADOS.
- Si se completa el cliente/PEC, Transportista y se le cargan los tubos, se crea la orden en estado CONFIRMADOS.
- Si se completa el cliente/PEC, Transportista, Tubos y se carga la hora de carga y descarga en la sección de datos de envío, se crea una orden con estado DESPACHADO.
- La Orden de Despacho debe verse reflejada en la base de datos.

KRH7 – Listado de órdenes de despacho.

Descripción: Como usuario debo ser capaz de visualizar una lista de las Órdenes de Despacho que se encuentran en el sistema. Además, debo tener 3 filtros de búsqueda para filtrar el listado. La información que se debe observar de cada orden es la siguiente:

- OOD (Numero de Orden de Despacho)
- Cliente
- Entrega
- Provincia
- Localidad
- Estado

Los filtros de búsqueda serán los siguientes:

- Búsqueda por Cliente
- Búsqueda por Estado
- Búsqueda por Fecha (mes y año)

Prioridad: Media

Criterio de aceptación:

- Se visualizan todas las órdenes de manera paginada.
- Si se filtra por Cliente, se deben observar todas las órdenes pertenecientes al cliente.

- Si se filtra por Estado, se deben observar todas las órdenes que tengan el estado seleccionado.
- Si se filtra por Fecha, se deben observar todas las órdenes de la fecha seleccionada.

KRH8 – Edición Órdenes de Despacho.

Descripción: Como usuario debo ser capaz de editar los datos de envío de una Orden de Despacho y completar los datos faltantes. Los datos de envío son los siguientes:

- Fecha de Entrega
- Provincia
- Localidad
- Transportista
- Tipo de Contratación
- Comentarios
- Horario Inicio Carga
- Horario Fin Carga

Prioridad: Media

Criterio de aceptación:

- No debe permitir dejar campos sin completar.
- Se debe actualizar el estado de la orden.
- Los cambios deben verse reflejado en la base de datos.

KRH9 – Obtener Orden de Despacho.

Descripción: Como usuario debo ser capaz de ver la información de una Orden de Despacho.

Prioridad: Media

Criterio de aceptación:

- Se debe observar los datos de envío.
- Se debe observar la lista de tipo de tuberías.
- Se debe observar el PEC.

KRH10 – Obtener Órdenes de Despacho por día.

Descripción: Como usuario debo ser capaz de ver una lista de las Órdenes de Despacho que hay en un día específico.
Prioridad: Media
Criterios de aceptación:

- Se debe observar el identificador (id) de la Orden de Despacho.
- Se debe observar el PEC en caso de que lo tenga.
- Se debe observar el Transporte en caso de que lo tenga.
- Se debe observar la fecha de entrega en caso de que lo tenga.

KRH11 – Crear Orden de Carga en formato PDF.

Descripción: Como usuario debo ser capaz de descargar una Orden de Carga en formato PDF.
Prioridad: Media
Criterio de aceptación:

- El pdf es persistido en MinIO.
- Se retorna un link al frontend que te redirecciona al recurso.
- Se le asigna un identificador único al archivo.

KRH12 – Asociación Tuberías y Tipo Tuberías.

Descripción: Como usuario debo ser capaz de asociar un Tipo de Tubería con una Tubería real ya producida. A su vez, se incorporan 2 checkbox (Etiqueta y Cuño), los cuales los usuarios tendrán la posibilidad de marcarlos o no.
Prioridad: Media
Criterio de aceptación:

- Se debe desplegar una lista de Tuberías.
- Se le debe asignar a un Tipo de Tubería una Tubería.
- Se debe poder hacer un check al campo Etiqueta y Cuño.
- Se debe persistir en la base de datos

KRH13 – Obtener Lista de SKU de los Tipo de Tuberías.

Descripción: Como usuario debo ser capaz de ver una lista de los SKU de los Tipos de Tuberías.
Prioridad: Media
Criterio de aceptación:

- Se retorna de la base de datos una lista solo de SKU de Tipo de Tuberías.

KRH14 – Obtener Detalle de los Tipo de Tuberías.

Descripción: Como usuario debo ser capaz de ver en detalle la información de un Tipo de Tubería seleccionado.
Prioridad: Media
Criterio de aceptación:

- Se debe observar el SKU.
- Se debe observar el Perfil
- Se debe observar la Coextrusión.
- Se debe observar las Normas

KRH15 – Obtener Tuberías.

Descripción: Como usuario debo ser capaz de obtener una lista de Tuberías.
Prioridad: Media
Criterio de aceptación:

- Se debe observar el Id.
- Se debe observar el número de serie.

KRH16 – Obtener Lista de Clientes.

Descripción: Como usuario debo ser capaz de obtener una lista de clientes regulares.
Prioridad: Media
Criterio de aceptación:

- Se recuperan los clientes con estado REGULAR.

KRH17 – Obtener PEC asociado al Cliente.

Descripción: Como usuario debo ser capaz de recuperar el PEC asociado a un cliente.
Prioridad: Media
Criterio de aceptación:

- Se observa el Id.
- Se observa el tipo de cliente.

10.5. Desarrollo API

En esta sección se describe el proceso de desarrollo de API implementada durante las prácticas. Para organizar el trabajo se adoptó Scrum como marco ágil de desarrollo, facilitando la gestión iterativa de tareas. A lo largo de este proceso, se llevaron a cabo diferentes actividades claves, como la definición de casos de uso, el diseño de un diagrama de clases UML y la elaboración de un diagrama de componentes, que ilustra la estructura general y las ilustraciones de los módulos del sistema.

10.5.1. Metodologías ágiles

Las metodologías ágiles son marcos de trabajo que tienen el objetivo de adaptar la forma de trabajo según las condiciones del proyecto, se busca conseguir una rápida respuesta a las diferentes circunstancias por las que puede pasar un proyecto, teniendo flexibilidad para amoldarse durante su implementación.

Las metodologías ágiles tienen varias ventajas sobre las tradicionales, entre ellas se pueden nombrar:

- **Mejora la calidad del producto**: se fomenta el enfoque proactivo de los miembros del equipo
- **Mayor satisfacción del cliente**: se involucra al cliente en el proceso, mediante presentaciones y entregas puede ver las mejoras en tiempo real.
- **Mayor motivación de los trabajadores**: los equipos de trabajo autónomos fomentan la creatividad y promueven la innovación entre sus integrantes.

- **Trabajo colaborativo**: se permite una mejor organización de trabajo, con diferentes roles, equipos y reuniones periódicas.
- **Uso de métricas más relevantes**: se utilizan métricas para estimar costes, rendimiento, tiempo, etc.
- **Mayor control y capacidad de predicción**: cada integrante es capaz de revisar y adaptar su trabajo a lo largo del proceso productivo.
- **Reducción de costes**: los errores se van viendo a lo largo del desarrollo, permitiendo identificar los puntos de mejora en el mismo proceso, evitando mostrar un producto final que no satisfaga al cliente y el proyecto fracase.

10.5.2. Scrum

Scrum es un marco de trabajo ágil para la gestión de proyectos, principalmente es usado en equipos de desarrollo de software, pero puede ser utilizado por cualquier equipo de trabajo. Como marco de trabajo define roles y ceremonias que el equipo debe llevar a cabo para garantizar la mejora continua del producto o servicio brindado. El trabajo se divide en periodos de tiempo llamados sprint, pueden ser de 15 o 30 días. Se define una serie de tareas que el equipo se compromete a terminar en este periodo de tiempo.

Un elemento fundamental de Scrum es el concepto de backlog, se trata de una lista de tareas a cargo del producto owner, las tareas varían desde nuevas funcionalidades, correcciones o mejoras.

En cuanto a los roles, se tienen tres roles bien definidos: product owner, scrum master y el equipo de desarrollo.

El product owner es quien tiene la visión del producto, está centrado en entender los requisitos del producto y gestionar el trabajo que será realizado por el equipo. Por su lado, el scrum master se encarga de la gestión del equipo, lidera las ceremonias de la metodología y se encarga de gestionar los recursos para que el equipo de desarrollo pueda realizar su trabajo de la manera más eficiente posible. Por último, el equipo de desarrollo se encarga de trabajar en los requisitos definidos por el producto owner.

Existen una serie de ceremonias que facilita la implementación de scrum, a continuación, se detallan:

- Sprint Planning: esta reunión se utiliza para planificar el alcance del sprint, cada miembro del equipo debe poder identificar qué tareas podrá cumplir durante el sprint.

- Daily Scrum: es una reunión diaria (usualmente a la mañana) de corta duración, cada miembro del equipo deberá contestar tres preguntas: ¿Qué hice ayer, ¿Qué tengo pensado hacer hoy? y ¿Tengo algún bloqueante?. Cada miembro debe ser muy conciso ya que la reunión idealmente debe durar 15 minutos. El objetivo es que cada miembro esté al tanto del trabajo del resto del equipo.
- Sprint Review: al final del sprint, el equipo se reúne y tiene una demo para revisar los avances obtenido. Las partes interesadas participan y dan un feedback.
- Sprint Retro: durante esta ceremonia el equipo se reúne a analizar qué se hizo bien y qué se hizo mal, qué cosas se pueden mejorar para el siguiente sprint.

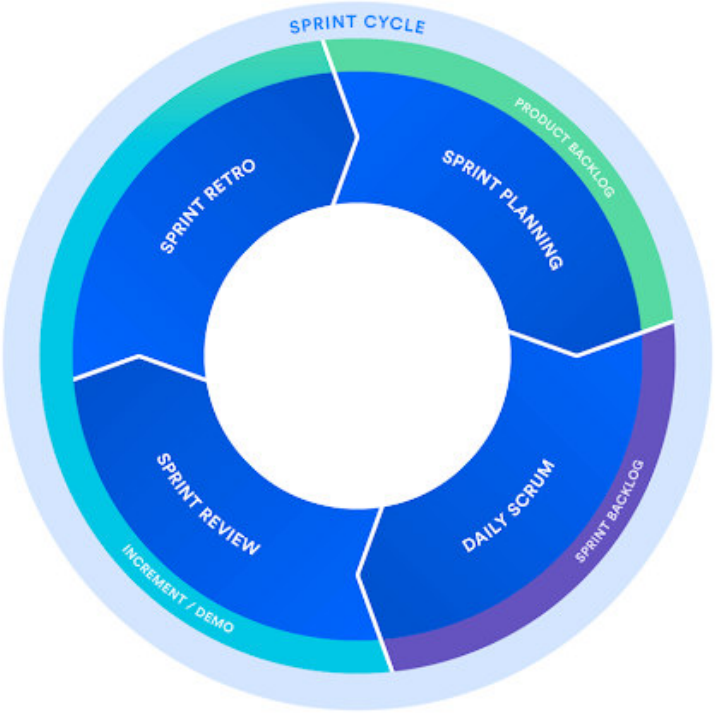


Figura 6: Ciclo del Sprint

Fuente: Recuperado de https://www.atlassian.com/es/agile/scrum

10.5.3. Flujo de trabajo

Una vez iniciado el desarrollo de la API, siguiendo los requerimientos previamente descritos, se adoptó un flujo de trabajo basado en Git. Para cada tarea asignada, se creó una nueva rama (branch) a partir de la rama principal (main branch), la cual contiene la versión más reciente y estable del proyecto. Este enfoque permite que cada miembro del equipo pueda trabajar de manera independiente, sin interferir en las actividades de los demás. Al finalizar el desarrollo en cada rama, el código se integra en la main branch mediante un proceso de fusión de código llamado merge, gestionando y resolviendo conflictos en caso de ser necesario. Este flujo garantiza un control más eficiente de versiones y facilita el trabajo en grupo.

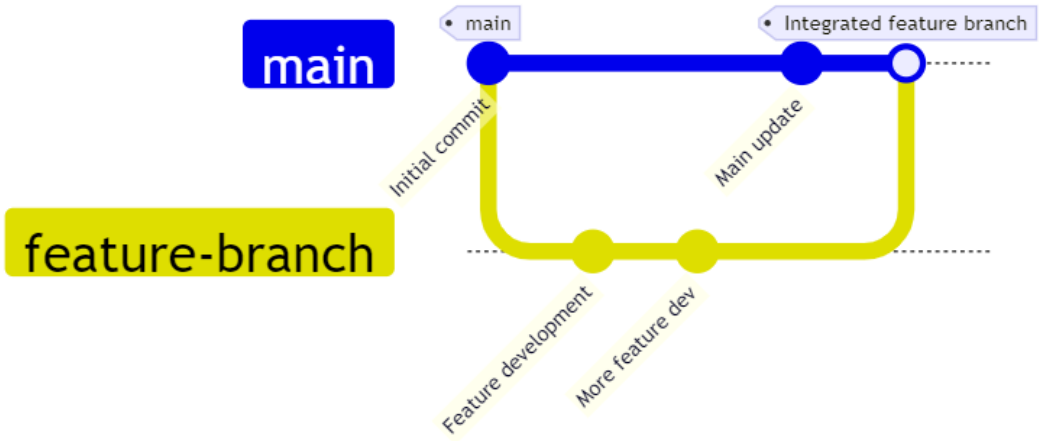


Figura 7: Flujo de trabajo con Git

Fuente: Elaboración propia, basada en la práctica.

10.5.4. Diagrama de componentes del sistema

El siguiente diagrama ilustra la arquitectura del sistema basada en microservicios, mostrando cómo interactúan los componentes:

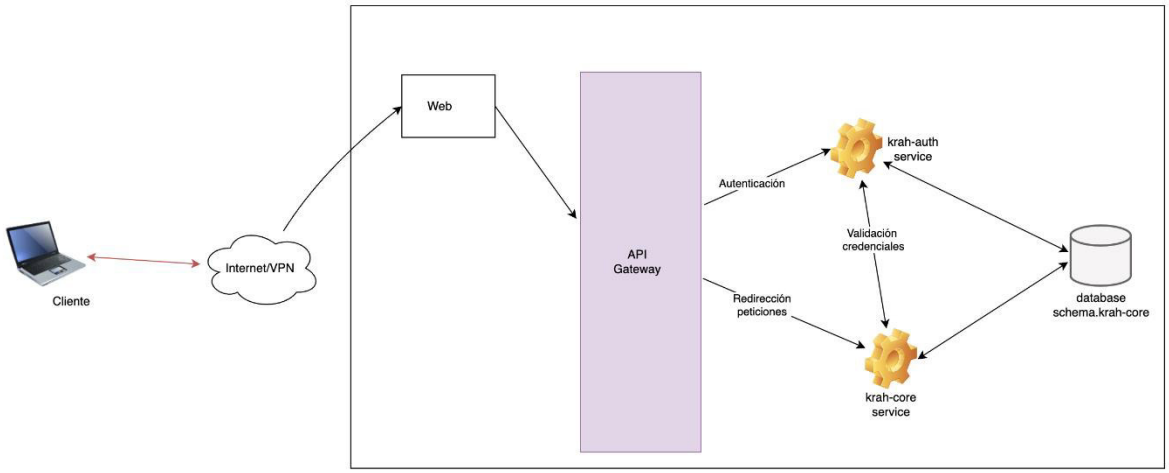


Figura 8: Diagrama de componentes del sistema

Fuente: Elaboración propia, basada en la práctica.

- Cliente: representa cualquier usuario que realiza peticiones al sistema, como solicitudes de datos o envío de datos. Las peticiones son recibidas por el API Gateway.
- API Gateway: actúa como punto central de entrada para todas las solicitudes externas. Su función principal es recibir las peticiones y redireccionarlas a su microservicio correspondiente. Además, garantiza que las peticiones estén previamente autenticadas.
- Krah-auth service: es responsable de autenticar a los usuarios, es decir, verifica las credenciales y, si son correctas, valida el acceso de los usuarios para las operaciones solicitadas.
- Krah-core service: es el servicio principal que gestiona la lógica de negocio. Procesa las solicitudes que hayan sido autenticadas y validadas. Se comunica con la DDBB para obtener la información requerida o persistir datos.
- Database: es responsable de persistir la información requerida por parte del negocio.

10.5.5. Diagrama UML

El siguiente diagrama UML refleja las entidades principales que intervienen en la gestión de órdenes de despacho, junto con sus atributos y relaciones entre las mismas. Cada clase representan un aspecto relevante en el proceso, como cliente, transportista, pedidos de ejecución de contrato y tubos, lo que permite gestionar de manera más estructurada las órdenes de despacho.

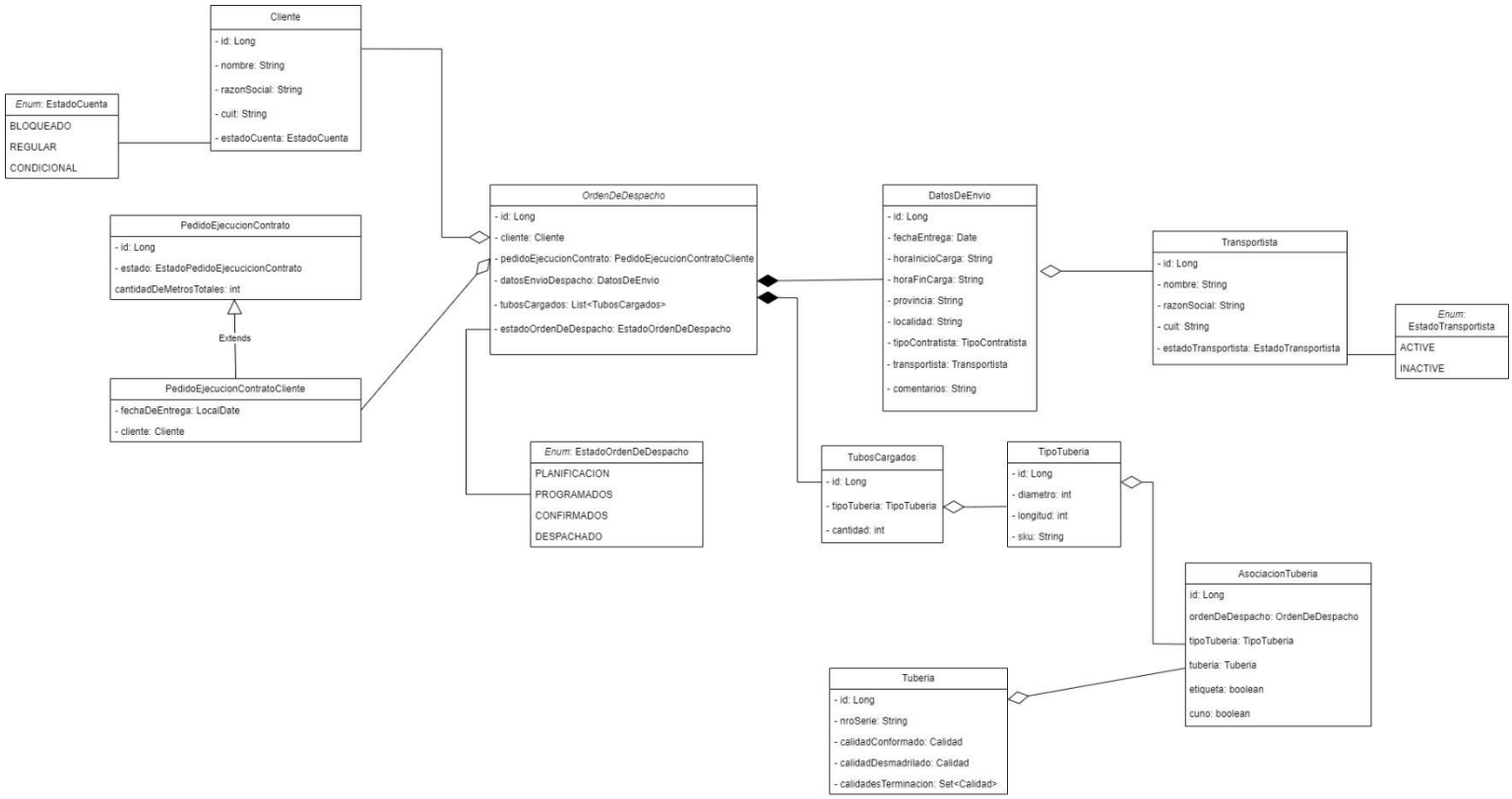


Figura 9: Diagrama UML Módulo Órdenes de Despacho

Fuente: Elaboración propia, basada en la práctica.

Cliente

- Atributos: id, nombre, razonSocial, cuit, estadoCuenta.
- Relación: Un cliente puede tener **múltiples pedidos** de ejecución de contrato (relación *1 a n*).
- El **EstadoCuenta** es un enum que puede tomar los valores: **BLOQUEADO, REGULAR, CONDICIONAL**.

PedidoEjecucionContrato

- Clase base con atributos como id, estado y cantidadDeMetrosTotales.

- Relación: esta clase se **especializa** en PedidoEjecucionContratoCliente, que incluye la **fecha de entrega** y una relación directa con el **cliente**.

OrdenDeDespacho

- Atributos: id, cliente, pedidoEjecucionContrato, datosEnvioDespacho, tubosCargados, estadoOrdenDeDespacho.
- Relación: cada orden de despacho está vinculada a **un cliente** y a un **pedido de ejecución de contrato**, y tiene un estado definido en el enum **EstadoOrdenDeDespacho**: **PLANIFICACION, PROGRAMADOS, CONFIRMADOS, DESPACHADO**.
- También contiene una lista de **tubos cargados**.

DatosDeEnvio

- Contiene información relevante para el despacho, como **fecha de entrega, localidad, transportista** asignado, y horarios de carga.
- Cada orden de despacho tiene un conjunto de datos de envío asociado.

Transportista

- Atributos: id, nombre, razonSocial, cuit, estadoTransportista.
- Relación: un transportista puede estar asignado a **varios envíos**.
- El estado del transportista es un enum con valores **ACTIVE** e **INACTIVE**.

TubosCargados y Tubería

- Cada tubo cargado se vincula con un **tipo de tubería** específico.
- Atributos de Tubería: id, nroSerie, calidadConformado y **calidades de terminación**.

TipoTubería

- Atributos como diámetro, longitud y sku.
- Relación: cada tipo de tubería puede ser utilizado en **varios tubos cargados** y puede aparecer en diferentes órdenes de despacho.

AsociacionTuberia

- Esta clase modela la **asociación entre una orden de despacho y los tubos cargados**, con atributos adicionales como etiqueta y cuno.

10.5.6. Casos de uso

Los casos de uso son una herramienta que nos permite describir cómo los distintos actores interactúan con el sistema para alcanzar un objetivo. En el módulo de gestión de órdenes de despacho, los casos de uso nos permiten detallar los procesos desde el punto de vista funcional, asegurando que se cubran las posibles interacciones que tendrán los usuarios con el sistema.

En este apartado se presentarán los casos de uso principales, tales como:

- Alta y gestión de Transportista.
- Visualización de Órdenes de Despacho
- Registro y actualización de órdenes de despacho.

El siguiente caso de uso hace foco en la gestión de Transportistas dentro del sistema. En este, un usuario con rol de operario tiene la capacidad de realizar varias operaciones fundamentales:

- Alta Transportista: permite registrar nuevos Transportistas en el sistema.
- Visualizar Lista Transportistas: permite visualizar una lista paginada de todos los transportistas del sistema.
- Editar Transportista: permite modificar los datos de un Transportista. Esta acción incluye la verificación de la existencia de un Transportista.
- Eliminar Transportista: permite eliminar a un Transportista del sistema. Esta acción incluye la verificación de la existencia de un Transportista.

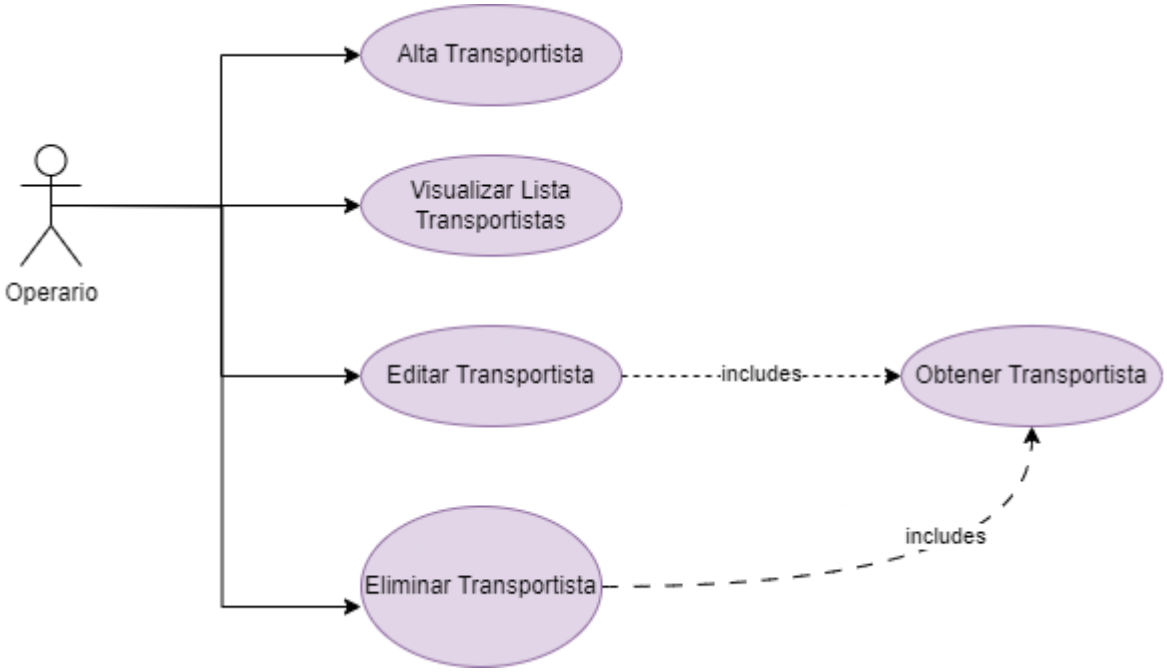


Figura 10: Casos de uso Transportistas

Fuente: Elaboración propia, basada en la práctica.

Este caso de uso aborda cómo un operario interactúa con diferentes funcionalidades relacionadas a la gestión de órdenes de despacho. Las principales acciones incluyen:

- Visualizar orden de despacho: permite visualizar al detalle una orden de despacho. Esta funcionalidad incluye la posibilidad de editar datos existentes, así como agregar los faltantes en caso de ser necesario.
- Visualizar lista órdenes de despacho: permite acceder a las órdenes de despacho creadas, facilitando la visualización de datos importantes como el estado actual y cliente.
- Descarga orden de carga: permite descargar un archivo pdf con toda la información de las tuberías a despachar, así como los datos de envío.
- Alta orden de despacho: permite registrar una nueva orden de despacho en el sistema.

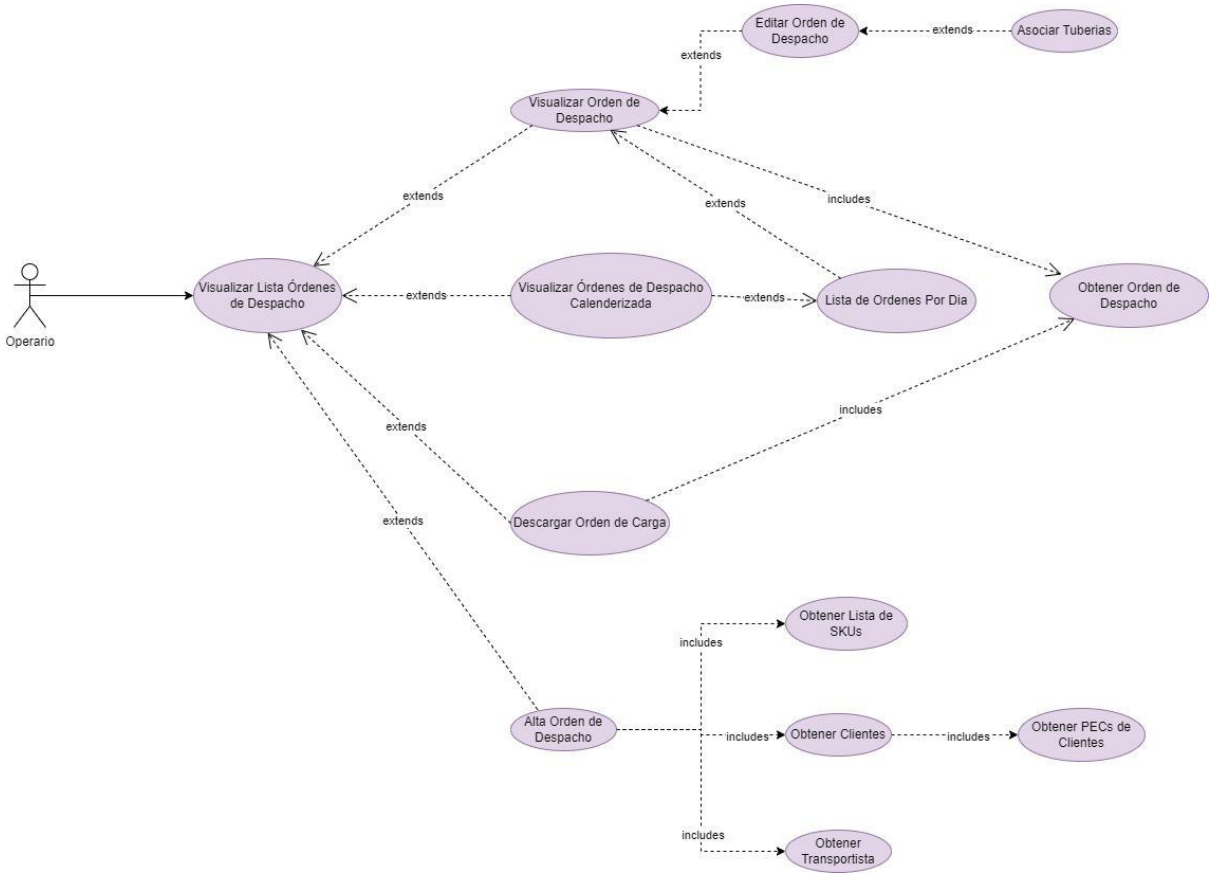


Figura 11: Casos de uso Visualización Órdenes

Fuente: Elaboración propia, basada en la práctica.

10.6. Resultados

En esta sección, se presentarán las pantallas de los módulos desarrollados durante el proyecto, con el objetivo de ilustrar las funcionalidades implementadas en cada módulo. Es importante destacar que el desarrollo del sistema se dividió en dos áreas: el frontend, este fue diseñado y construido por un especialista en interfaces de usuarios y, el backend, del cual fui responsable.

A través de estas imágenes, se podrá observar las principales características de la API y los procesos de backend implementados e integrados con la interfaz de usuario para proporcionar un flujo de trabajo funcional y eficiente. Esto permitirá una visualización clara de los resultados obtenidos, evidenciando el correcto funcionamiento y la colaboración entre ambas capas de la aplicación.

10.6.1. Pantalla principal

La pantalla principal del sistema cuenta con un menú lateral donde se encuentran todos los módulos de la aplicación, dentro de este se incorporan dos nuevos módulos: **Transportistas** y **Despacho**, los cuales se encuentran resaltados en el cuadro rojo.

Figura 12: Pantalla principal MRP

Fuente: Elaboración propia, basada en la práctica.

10.6.2. Módulo Transportista

Dentro del módulo Transportista, se podrá encontrar una lista paginada de los transportistas dados de alta en el sistema, por el momento, no existe ninguno, por lo cual, aparece el mensaje “No hay información”. También, se encuentran un filtro de búsqueda que permitirá filtrar transportistas por nombre y un botón Alta Transportista que redirigirá hacia un formulario para dar de alta un nuevo transportista en el sistema.

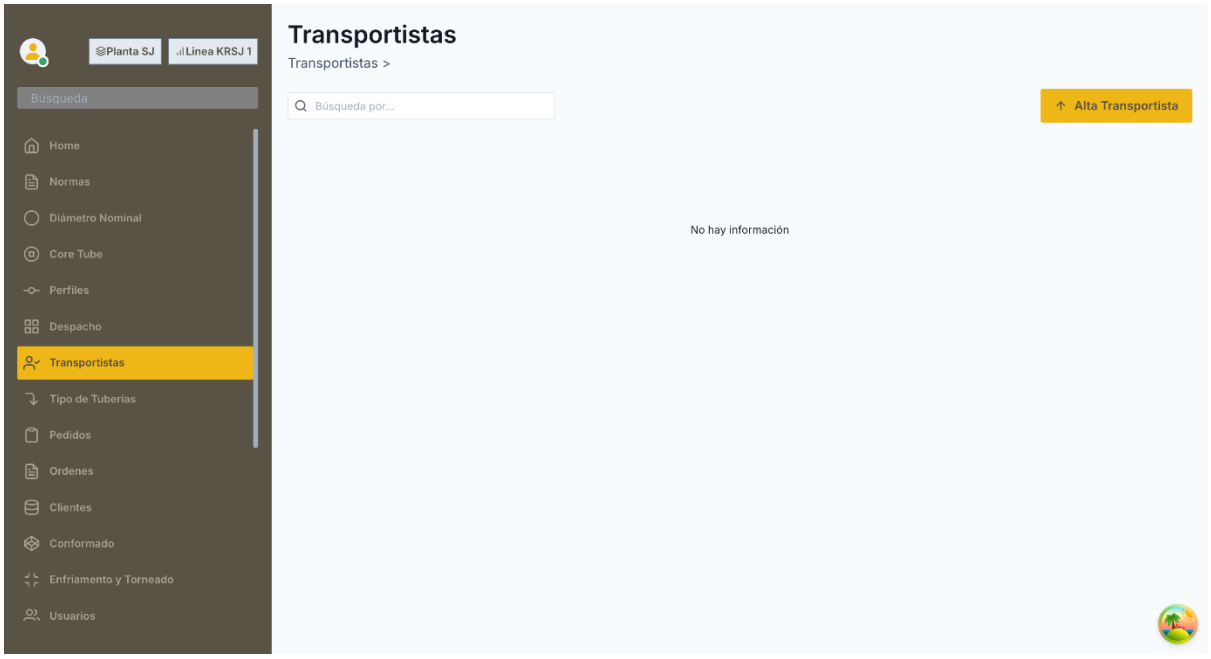


Figura 13: Pantalla módulo Transportista

Fuente: Elaboración propia, basada en la práctica.

10.6.2.1. Alta Transportista

Al presionar el botón Alta Transportista se puede observar el formulario que permite cargar los datos del transportista: nombre, cuit y razón social. Una vez completados los datos, se presiona el botón Confirmar Transportista y el sistema se encarga de persistir el nuevo transportista. Al momento de la creación, el nuevo transportista tiene por default el estado **Activo**.

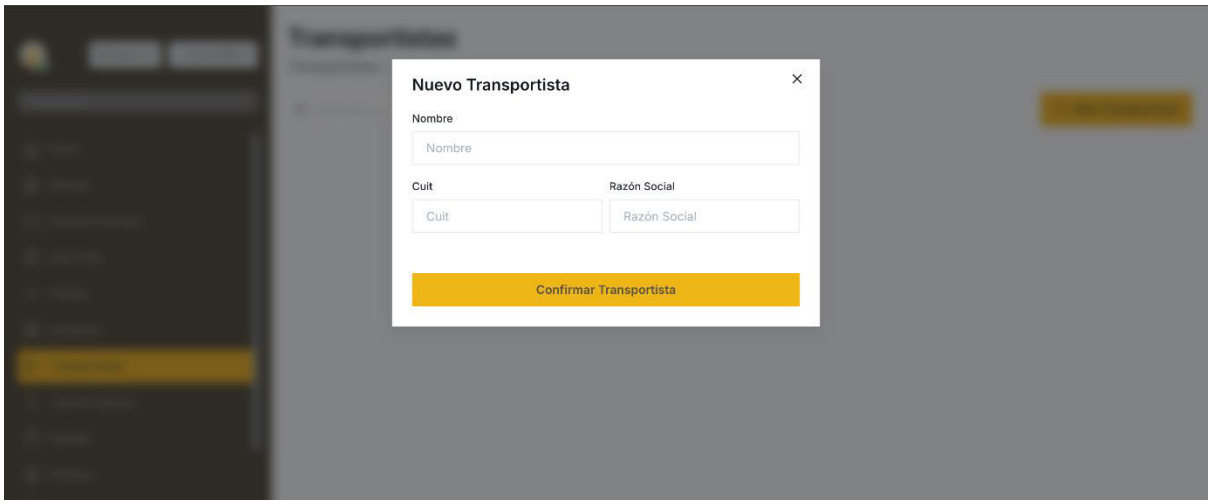


Figura 14: Formulario Alta Transportista

Fuente: Elaboración propia, basada en la práctica.

10.6.2.2. Visualización lista transportistas

Al crear un nuevo transportista, ahora se podrá observar una lista paginada de estos. Junto con esta lista, se realizarán dos acciones diferentes, editar los datos del transportista o eliminarlo del sistema. Estas acciones se las podrá observar dentro de la columna *Acciones* en la tabla.

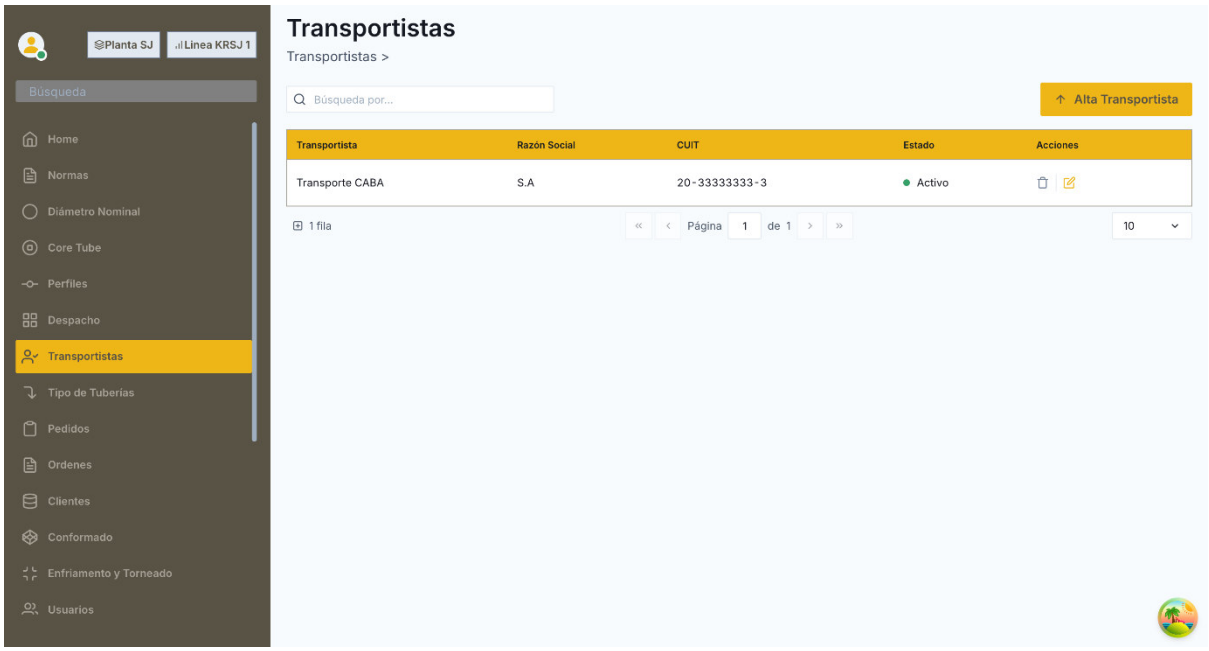


Figura 15: Lista paginada de Transportistas

Fuente: Elaboración propia, basada en la práctica.

10.6.2.3. Edición Transportista

Al presionar el icono de edición, se obtendrá un nuevo formulario con los datos del transportista y se podrá modificar cualquiera de ellos. En este ejemplo, se cambiará el estado a **Inactivo**.

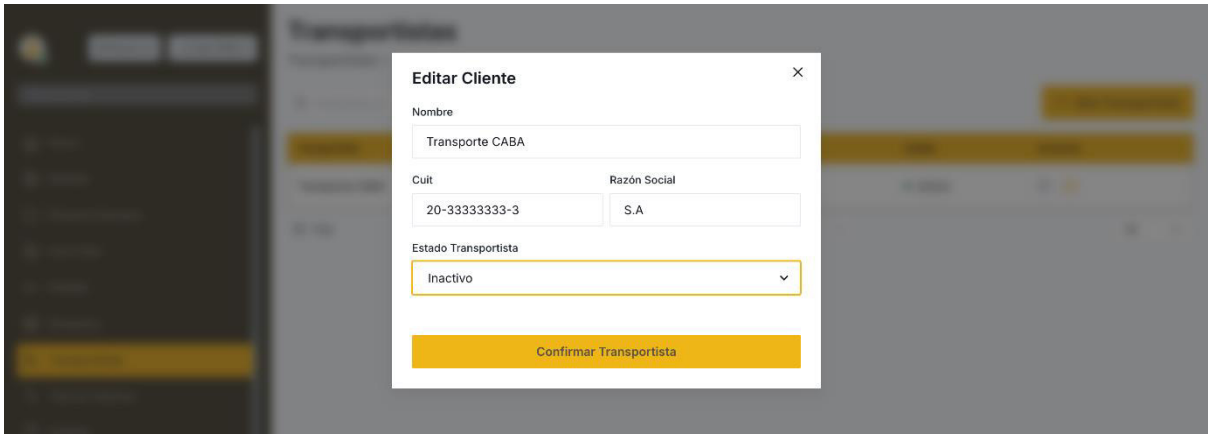


Figura 16: Formulario edición Transportista

Fuente: Elaboración propia, basada en la práctica.

Luego, se presionará el botón *Confirmar Transportista*, el cual devolverá a la lista paginada y en el mismo, se podrá observar que el estado efectivamente ha cambiado a **Inactivo**.

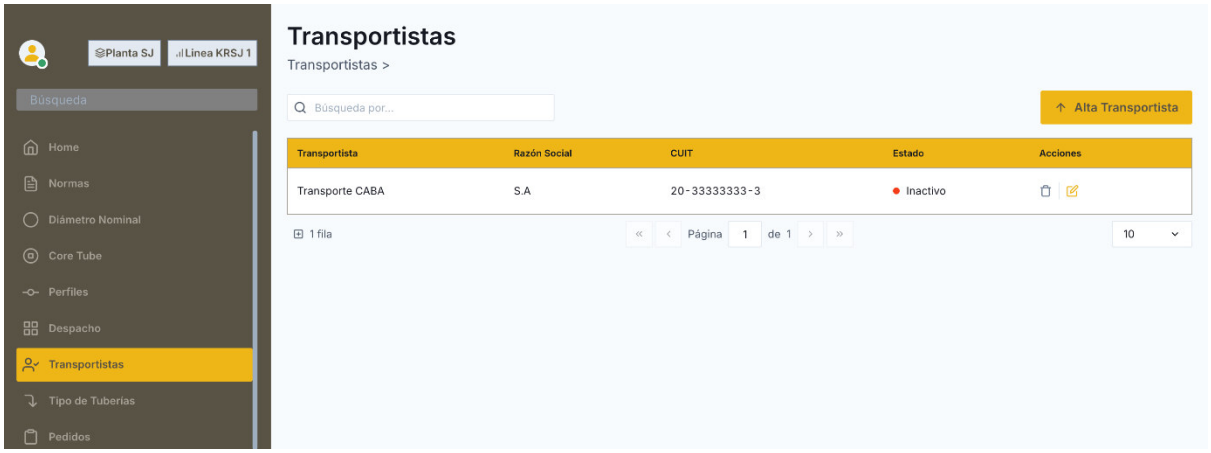


Figura 17: Transportista editado

Fuente: Elaboración propia, basada en la práctica.

10.6.2.4. Eliminar transportista

Para el caso de que el usuario quisiera eliminar a un transportista, debe presionar el ícono de eliminación. Una vez presionado, se obtendrá un mensaje de confirmación preliminar, en caso de aceptar, el transportista se elimina del sistema.

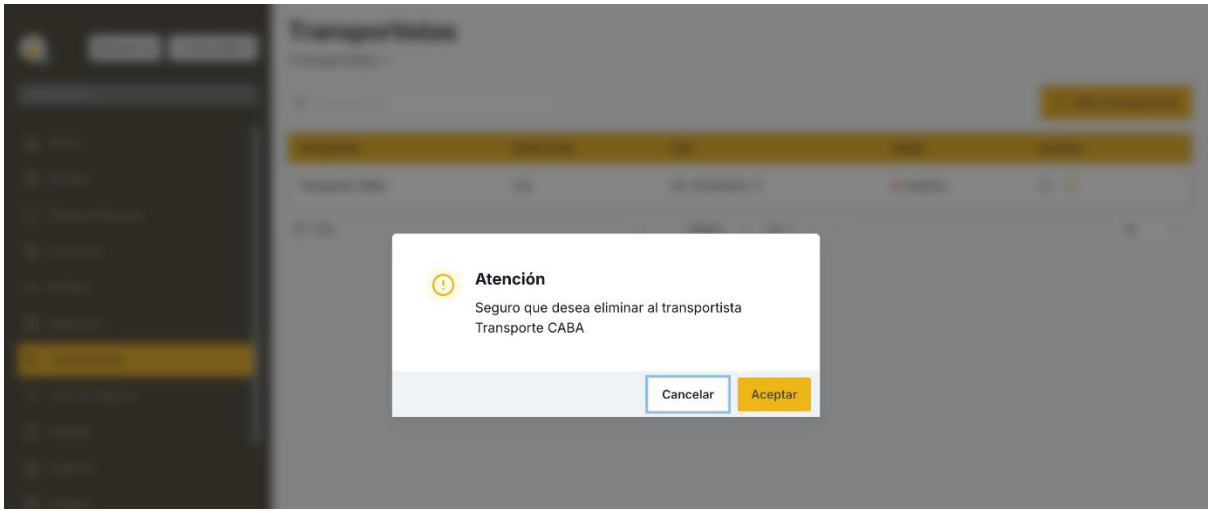


Figura 18: Confirmación eliminación Transportista

Fuente: Elaboración propia, basada en la práctica.

10.6.3. Módulo Despacho

En este módulo, se contará con dos vistas, una lista paginada de las órdenes de despacho y otra vista calendarizada donde se podrán ver las órdenes programadas para los días. Además, se cuenta con tres filtros: filtro por cliente, estado y mes y año.

En esta pantalla, se tendrá un botón *Alta ODD*, el cual permitirá comenzar el proceso para crear una nueva orden de despacho.

En la figura 20, se puede observar la vista Lista, en donde se mostrarán todas las órdenes creadas, como todavía no existe ninguna aparece el mensaje “No hay información”.

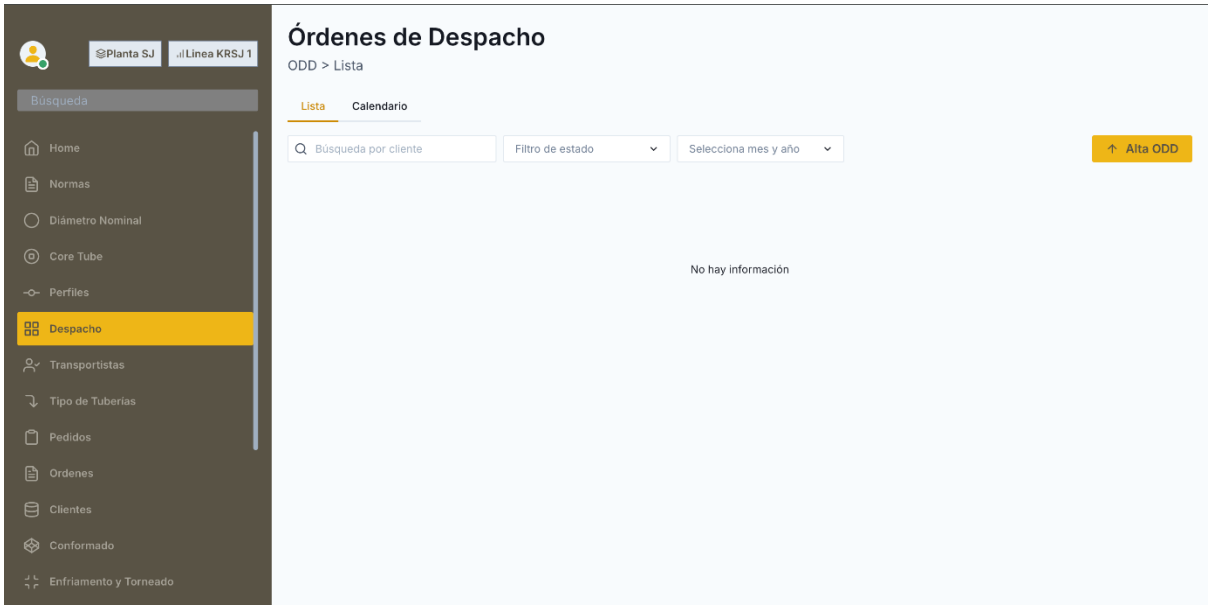


Figura 19: Pantalla módulo Órdenes de Despacho

Fuente: Elaboración propia, basada en la práctica.

Luego, en la vista *Calendario*, se podrá observar un calendario en el cual aparecerán las órdenes que tienen como fecha de entrega ese día.

Figura 20: Vista calendarizada órdenes de despacho

Fuente: Elaboración propia, basada en la práctica.

10.6.3.1. Alta Órdenes de Despacho

En esta subsección, se presentan las pantallas para el registro de nuevas órdenes de despacho, destacando el flujo de trabajo que permite al usuario gestionar órdenes de manera eficiente.

Como primer paso para la creación, el usuario deberá seleccionar de manera obligatoria qué tipos de tuberías se despacharán junto con la cantidad correspondiente.

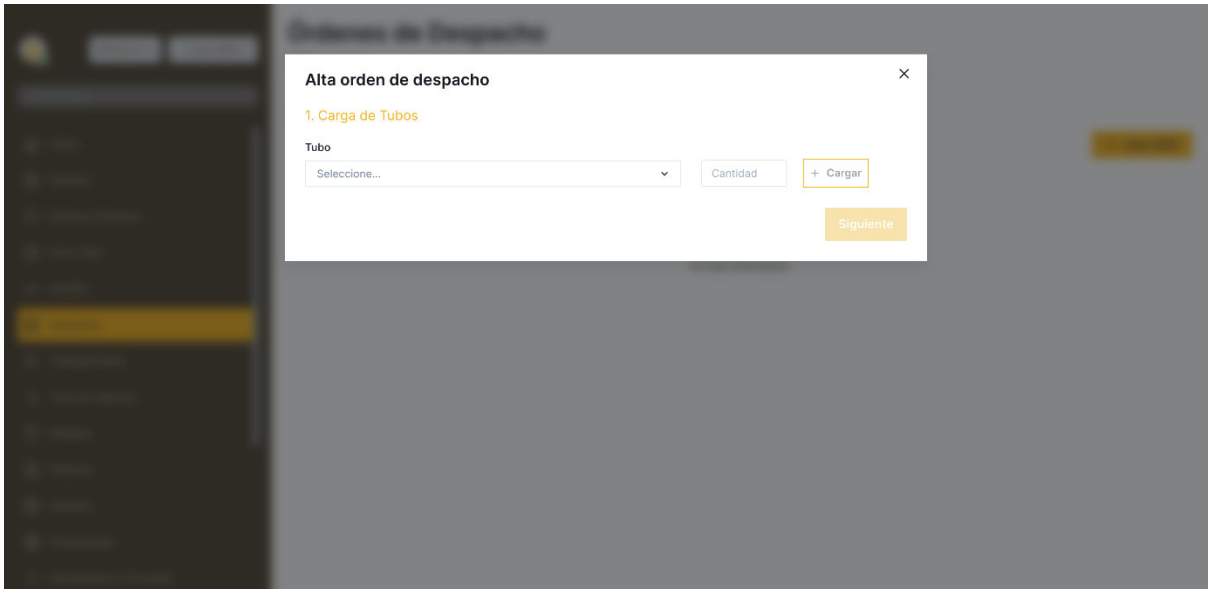


Figura 21: Selección Tipo de Tuberías y cantidad

Fuente: Elaboración propia, basada en la práctica.

Al presionar sobre la lista desplegable Tubo, se observará una lista de los SKU (identificador único de los tipos de tubería). Al seleccionar uno y escribir la cantidad correspondiente se habilitará el botón *Cargar.*

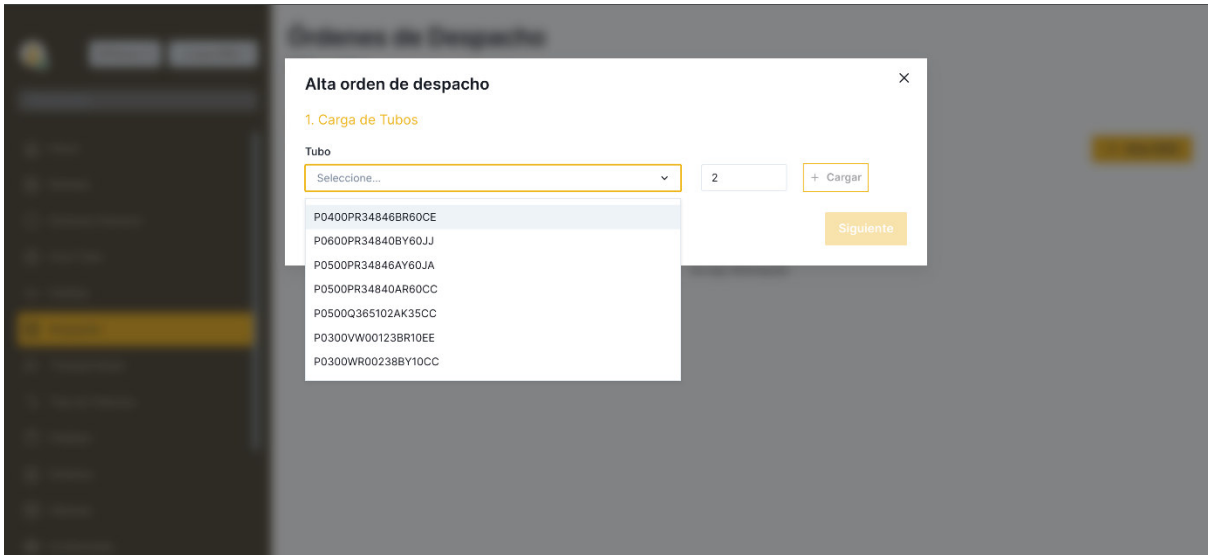


Figura 22: Lista de SKU

Fuente: Elaboración propia, basada en la práctica.

Al cargar un nuevo tipo de tubería, aparecerá en la parte de abajo alguna información perteneciente al mismo. A su vez, tendrán la posibilidad de seguir seleccionando tipos de tuberías para despachar.

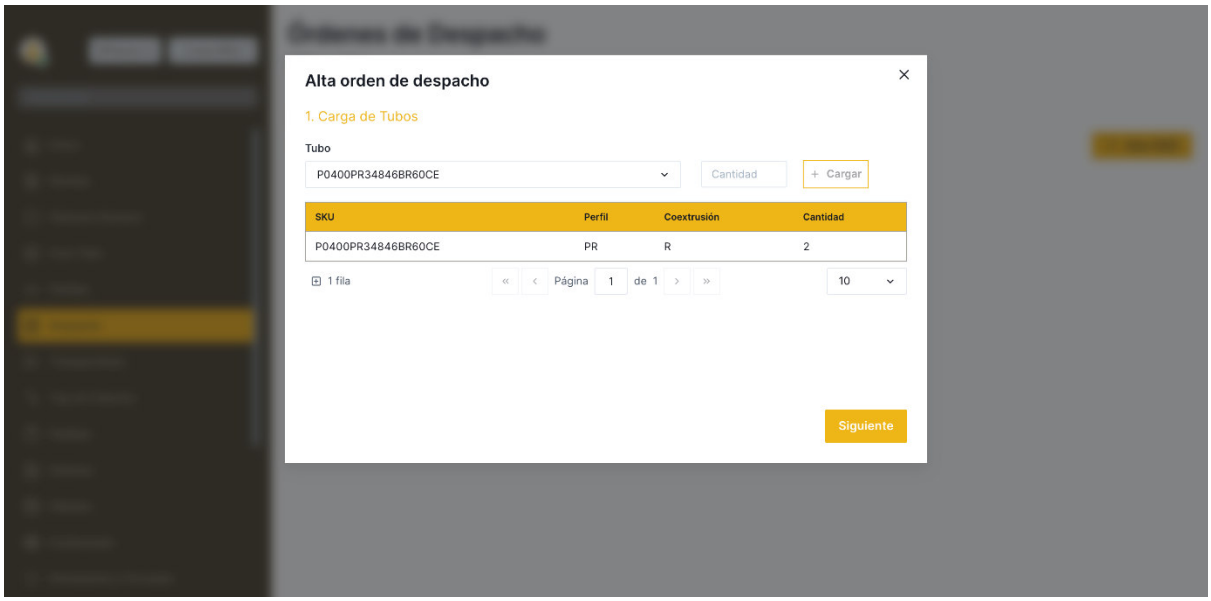


Figura 23: Tipo de Tubería cargada

Fuente: Elaboración propia, basada en la práctica.

Una vez cargada la cantidad de tuberías requerida se presiona el botón *Siguiente* y se pasa a la siguiente pantalla.

En esta segunda instancia existen dos campos opcionales: Cliente y PEC. En el campo cliente tendrá una lista desplegable con los clientes activos dentro del sistema. Al seleccionar un cliente, la lista desplegable de PEC se actualiza para mostrar únicamente los PEC asociados al cliente seleccionado.

Para el ejemplo actual, no se completarán estos campos y se presionará el botón *Siguiente* para continuar con el proceso de creación.

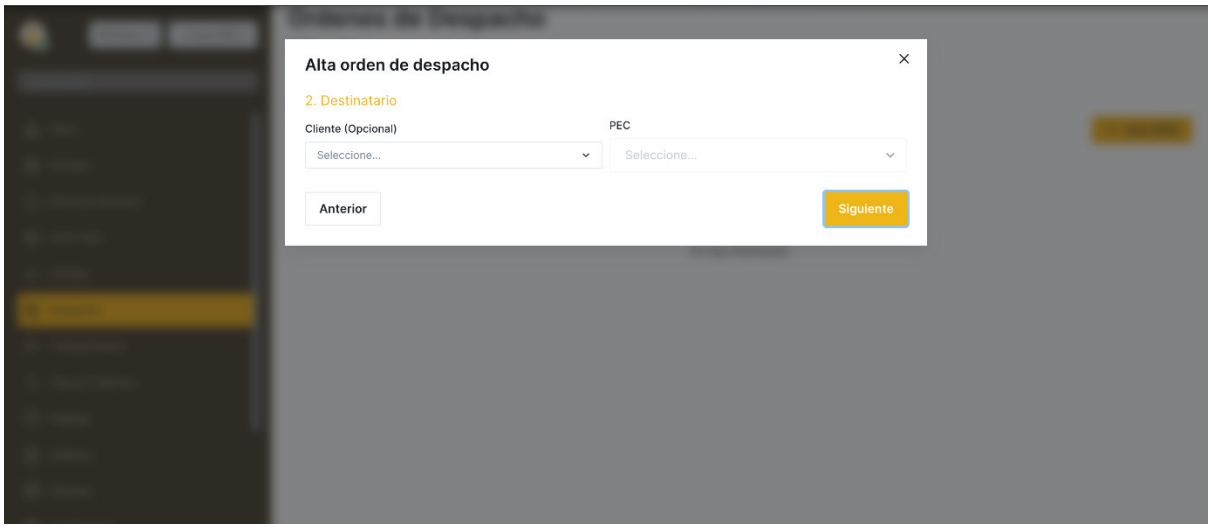


Figura 24: Modal selección de cliente y PEC

Fuente: Elaboración propia, basada en la práctica.

Como último paso, es necesario cargar los datos de envío. El único dato obligatorio es la Fecha de Entrega, al ingresar esta información se podrá confirmar la creación de la orden.

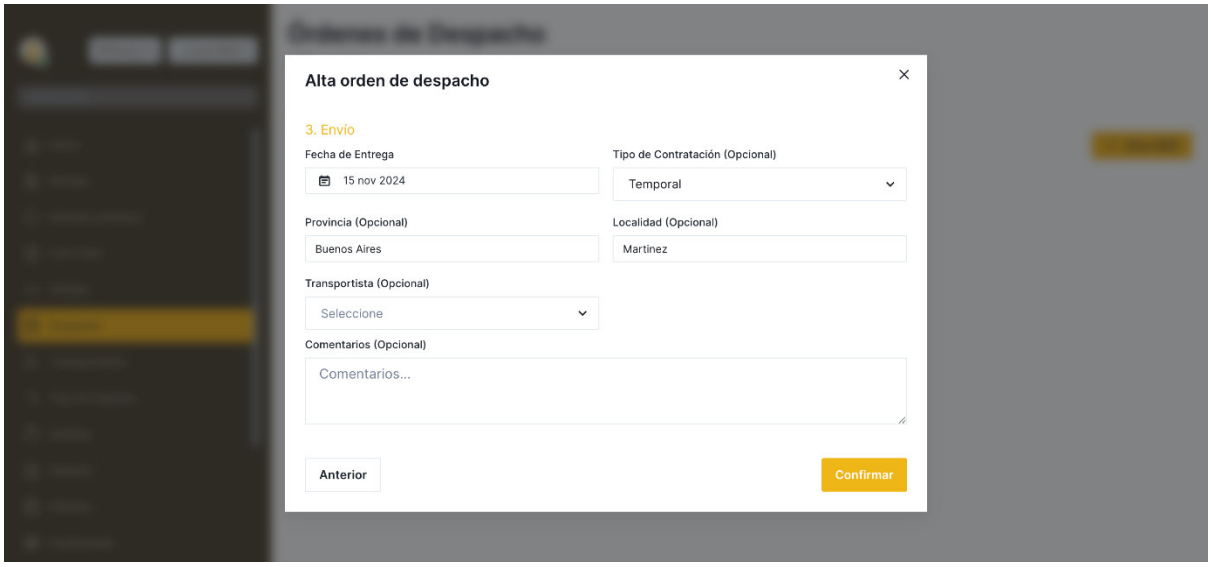


Figura 25: Modal datos de envío

Fuente: Elaboración propia, basada en la práctica.

Antes de la creación se tendrá un paso que es la confirmación definitiva, al presionar *Aceptar* la orden se creará y el sistema se redirigirá al módulo de despacho.

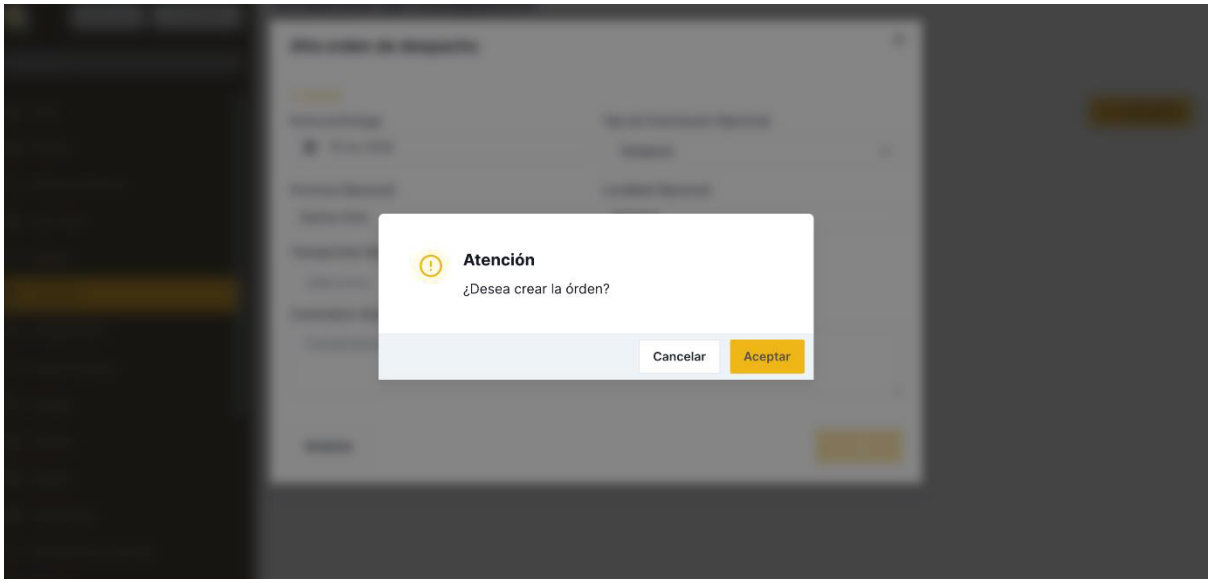


Figura 26: Confirmación creación de orden

Fuente: Elaboración propia, basada en la práctica.

Una vez completado todos los pasos y confirmado la orden se puede observar la orden creada en la lista.

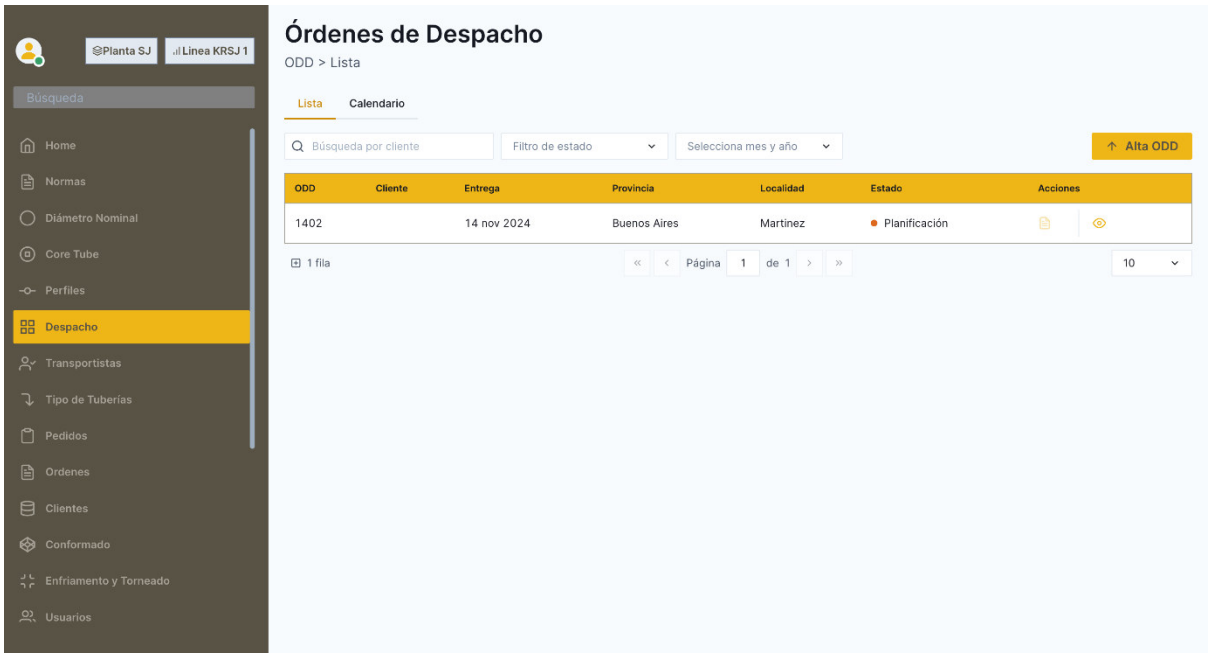


Figura 27: Lista órdenes

Fuente: Elaboración propia, basada en la práctica.

10.6.3.2. Filtros y calendario

El módulo de despacho cuenta con tres filtros que facilitan la búsqueda y organización de las órdenes. Para visualizar el funcionamiento de los filtros se crearon órdenes con distintos estados y fechas de entrega como se observa en la siguiente imagen:

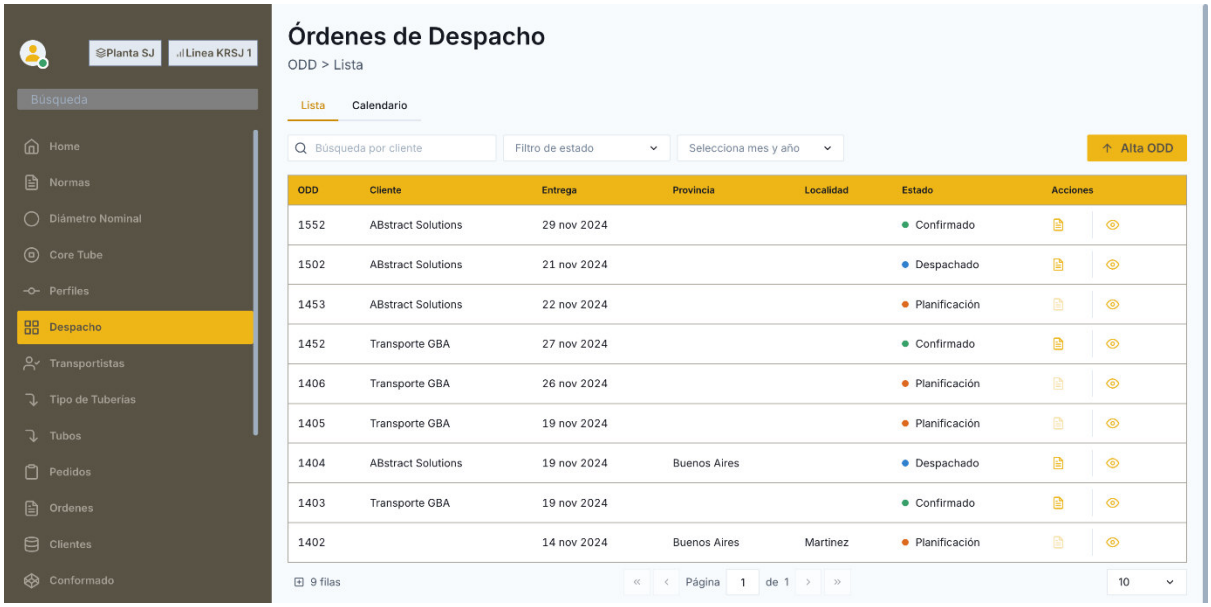


Figura 28: Órdenes con distintos estados

Fuente: Elaboración propia, basada en la práctica.

Para utilizar el filtro de búsqueda por cliente, simplemente se debe escribir el nombre del cliente de la orden y automáticamente la lista se comienza a actualizar con las coincidencias que vaya encontrando:

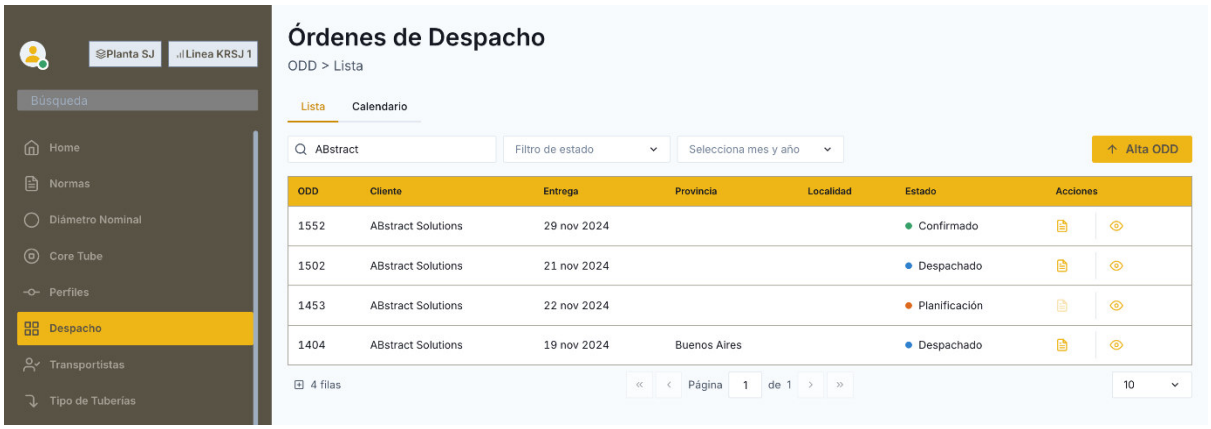


Figura 29: Filtro de búsqueda

Fuente: Elaboración propia, basada en la práctica.

Para el caso de los estados, se cuenta con una lista desplegable que muestra los estados posibles de las órdenes, al seleccionar uno se actualizan las órdenes con las que coincidan con el estado seleccionado:

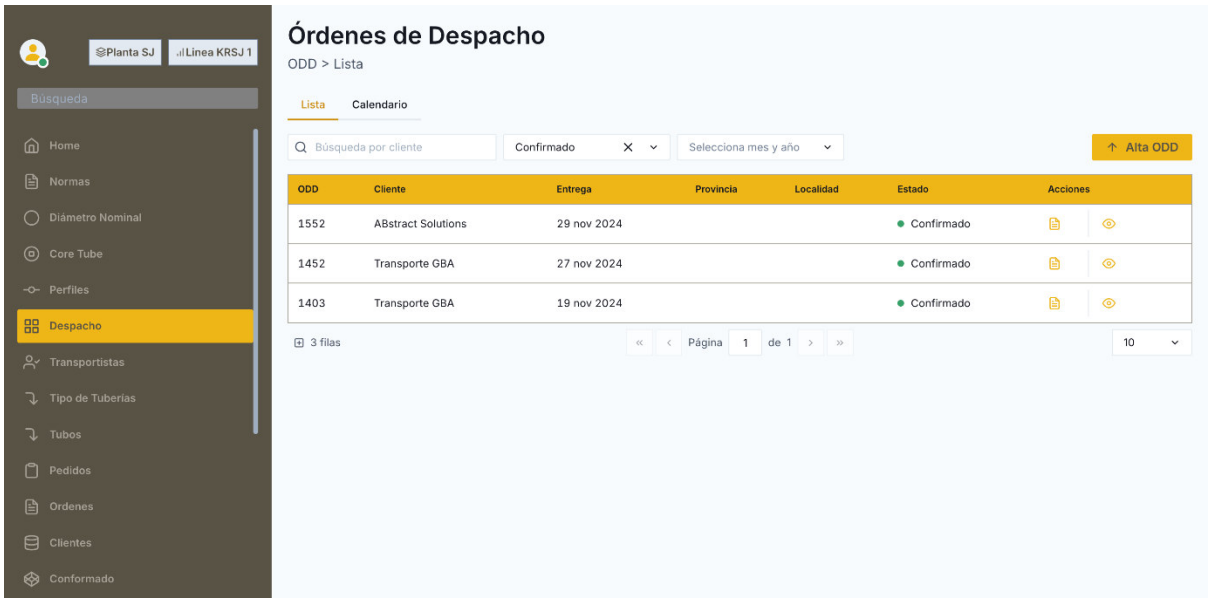


Figura 30: Filtro de estados

Fuente: Elaboración propia, basada en la práctica.

Por último, se tendrá otra lista desplegable en el cual se tiene la opción de elegir mes y año de las órdenes, por lo tanto, solo se observarán en la lista aquellas órdenes que coincidan con el mes y año seleccionado:

Figura 31: Filtro de fechas

Fuente: Elaboración propia, basada en la práctica.

En el caso de seleccionar la pestaña *Calendario*, se podrán observar las órdenes existentes para cada día del mes.

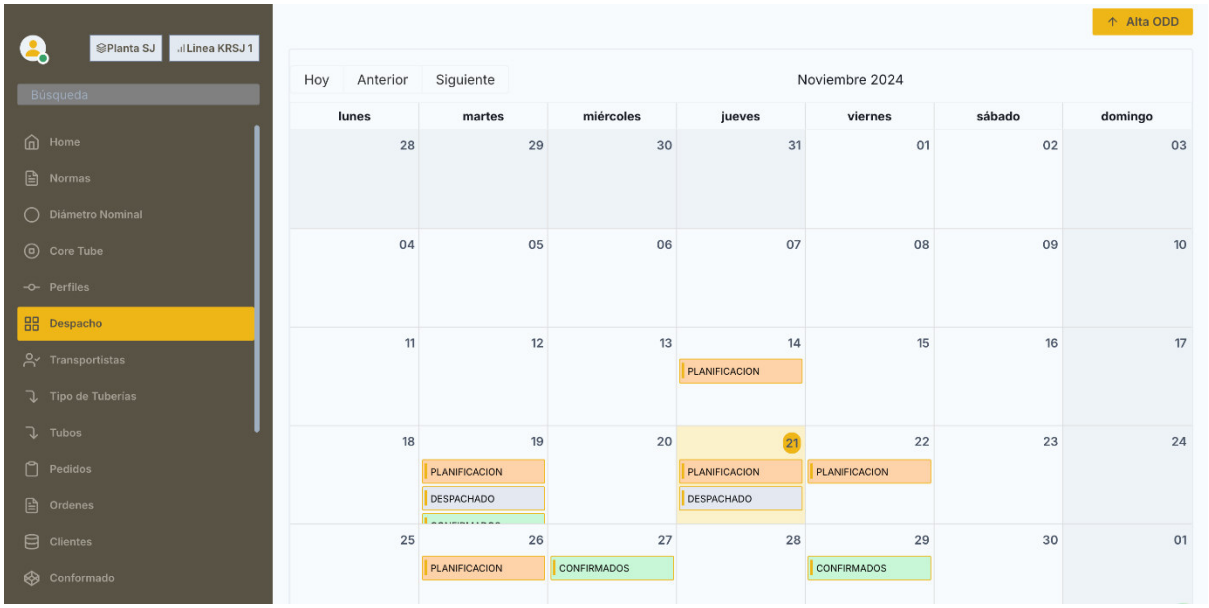


Figura 32: Vista calendarizada

Fuente: Elaboración propia, basada en la práctica.

Al presionar una fecha en el calendario, se abrirá una ventana con una lista de las órdenes correspondientes al día.

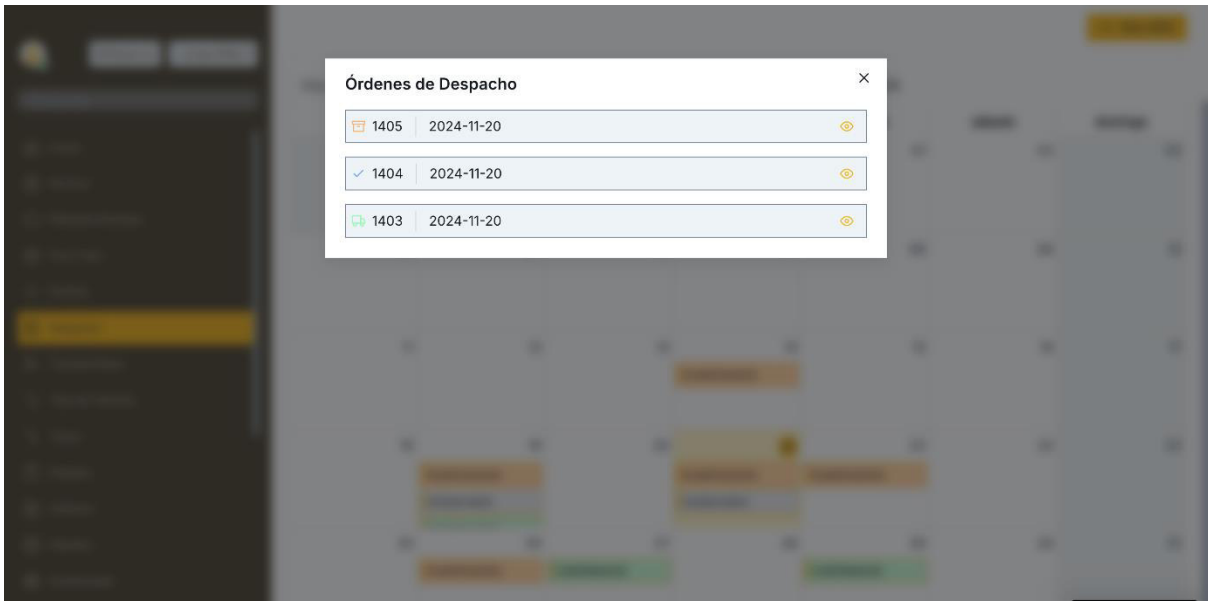


Figura 33: Lista de órdenes por día

Fuente: Elaboración propia, basada en la práctica.

10.6.3.3. Actualizar orden de despacho

Una vez que una orden es creada, ya sea con datos incompletos o completos, se tiene la posibilidad de editar los datos cargados.

Dependiendo de los datos cargados la orden tendrá un estado asignado:

- Si cuenta con tubos cargados y fecha de entrega se le asigna el estado PLANIFICACION.
- Si cuenta con lo nombrado en punto anterior y, además, se le carga un cliente/PEC y transportista, se le asigna el estado CONFIRMADO.
- Si se le cargan los datos Hora Inicio Carga y Hora Fin Carga, se le asigna el estado DESPACHADO.

Para actualizar una orden, se tomará como ejemplo la orden 1403, la cual tiene el estado CONFIRMADO.

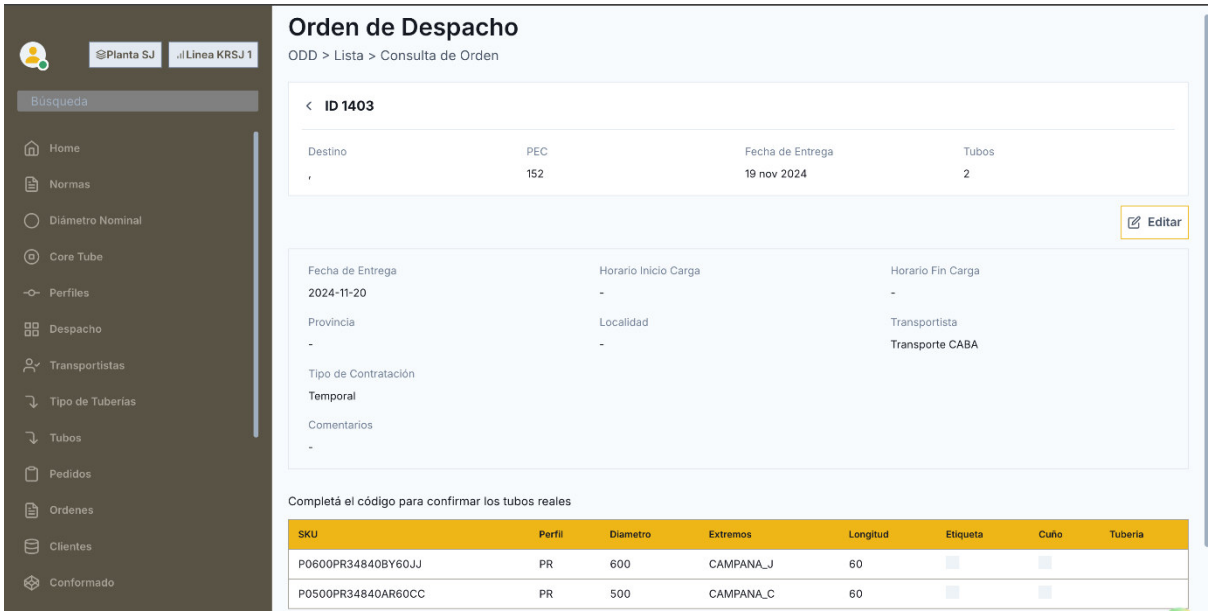


Figura 34: Detalle orden de despacho

Fuente: Elaboración propia, basada en la práctica.

Al presionar el botón *Editar*, se habilitarán los campos para su modificación. En la sección baja de la pantalla se tiene la posibilidad de asignarle un tubo a un tipo de tubería seleccionado en el primer paso de la creación de la orden.

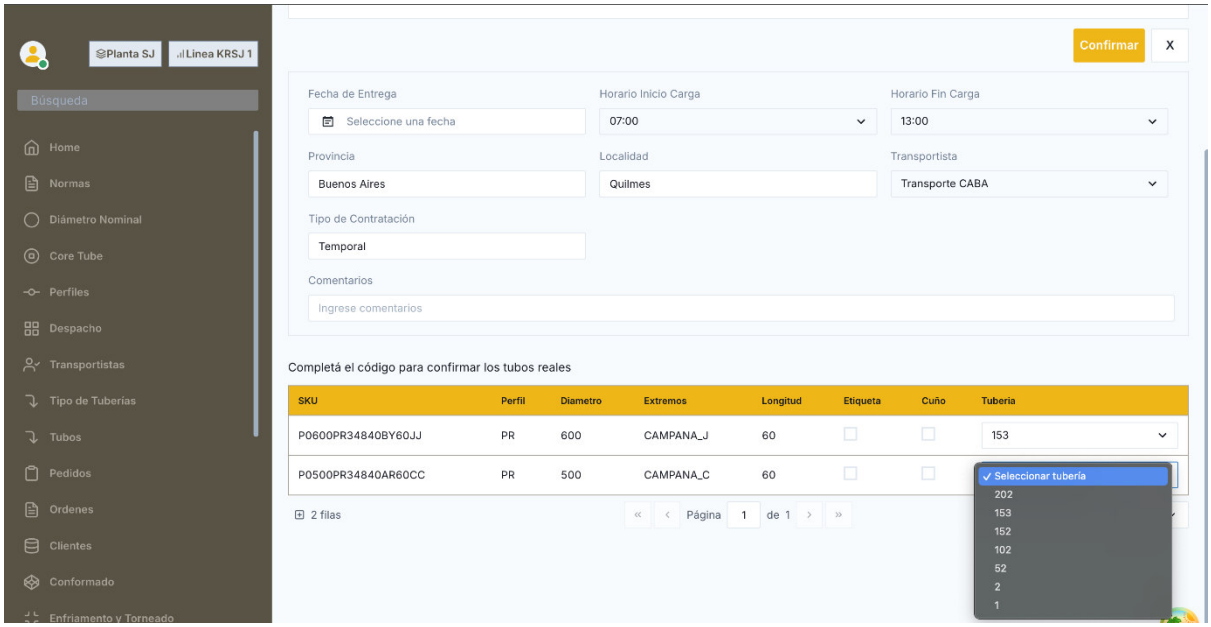


Figura 35: Edición orden de despacho

Fuente: Elaboración propia, basada en la práctica.

Una vez completado los campos, se presiona el botón *Confirmar* y se aparecerá un mensaje de confirmación, se aceptan los cambios y se persisten en la base de datos.

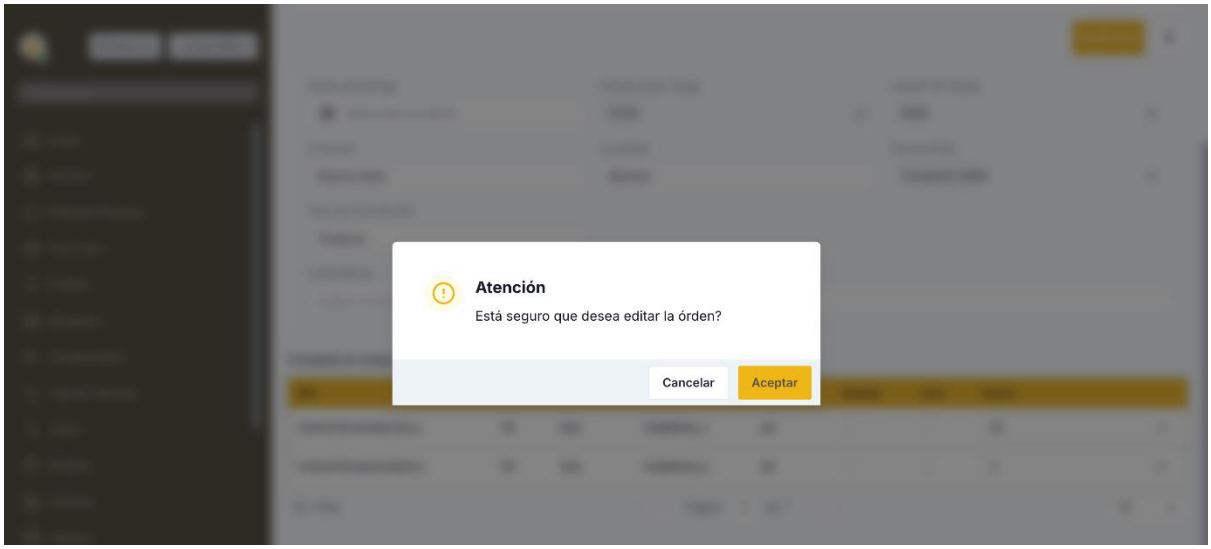


Figura 36: Confirmación edición orden

Fuente: Elaboración propia, basada en la práctica.

Luego de aceptar los cambios, se podrán observar los cambios:

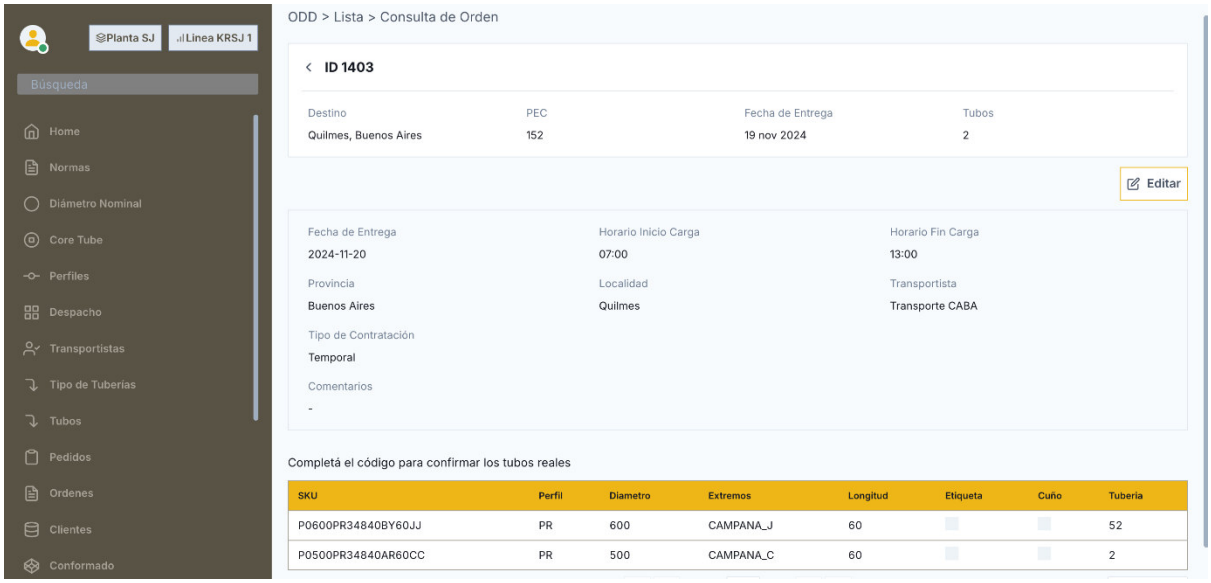


Figura 37: Orden actualizada

Fuente: Elaboración propia, basada en la práctica.

En la lista de órdenes se podrá apreciar el cambio de estado y los datos como provincia y localidad:

Figura 38: Estado orden actualizada

Fuente: Elaboración propia, basada en la práctica.

10.6.3.4. Descarga orden de carga

Cuando una orden se encuentra en estado CONFIRMADO o DESPACHADO en la lista de las órdenes, en la columna de *Acciones* se habilitará la opción de descargar la orden de carga, que consiste en un archivo PDF con la información de los tubos a despachar, así como datos de envío.

En la siguiente imagen se puede observar que las órdenes con los estados nombrados no se encuentran griseados:

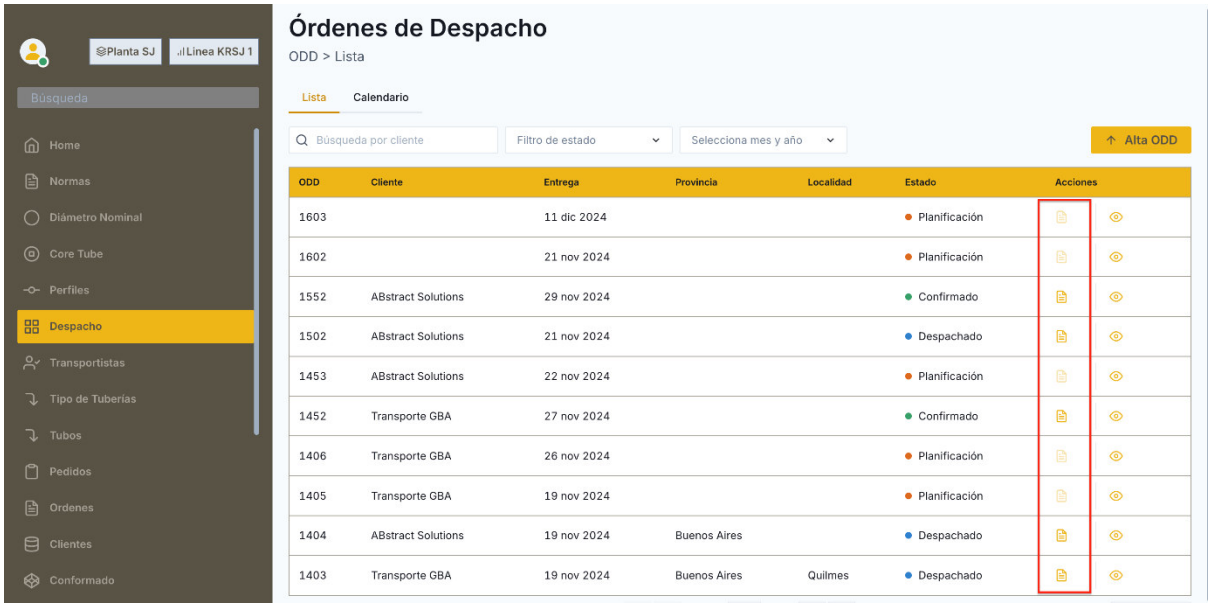


Figura 39: Acción descarga de orden

Fuente: Elaboración propia, basada en la práctica.

Al presionar el ícono, en una nueva pestaña del navegador aparecerá un PDF como el siguiente:

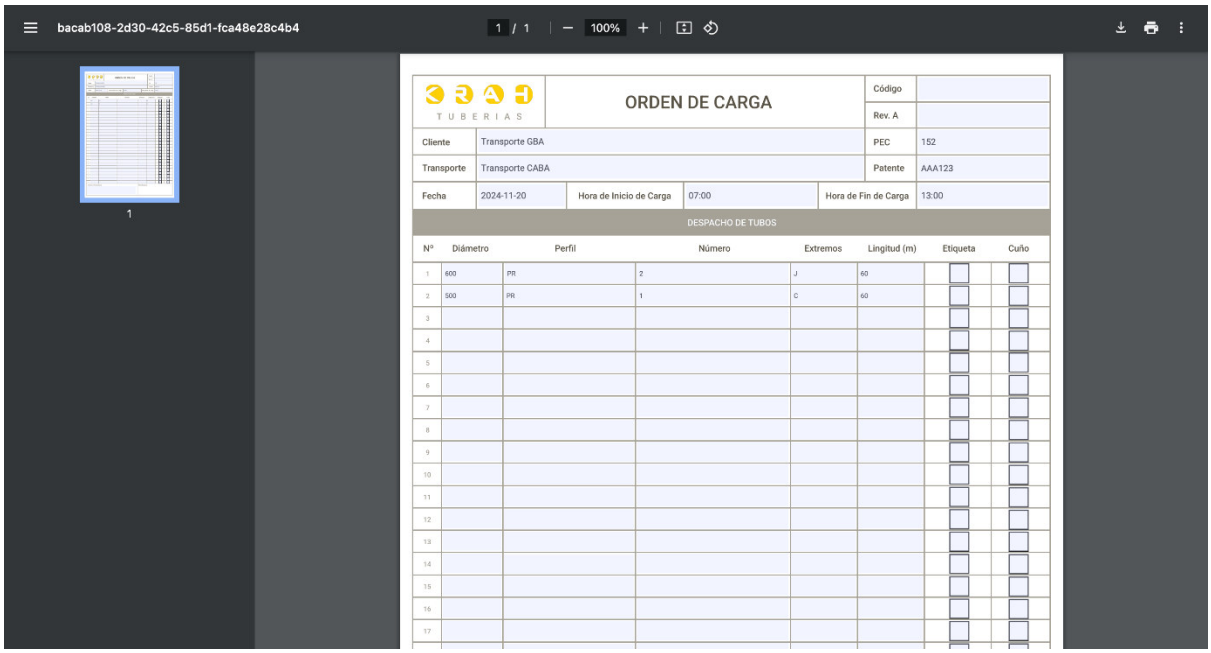


Figura 40: Orden de carga

Fuente: Elaboración propia, basada en la práctica.

10.7. Documentación

Para la documentación se utilizó Swagger, la cual se integró a través de la biblioteca **springdoc-openapi** para Spring Boot. Esta herramienta permite generar documentación interactiva para las APIs, la cual incluye:

- Descripción de los endpoints: métodos HTTP, rutas, parámetros y respuestas.
- Interfaz: permite probar los endpoints desde el navegador.
- Esquema OpenAPI: genera un archivo JSON o YAML, que describe la API de acuerdo con el estándar OpenAPI.

Con una configuración sencilla en el proyecto, se puede acceder a una interfaz que organiza los endpoints por controlador. A continuación, se presentan las listas generadas para los endpoints creados para el proyecto:

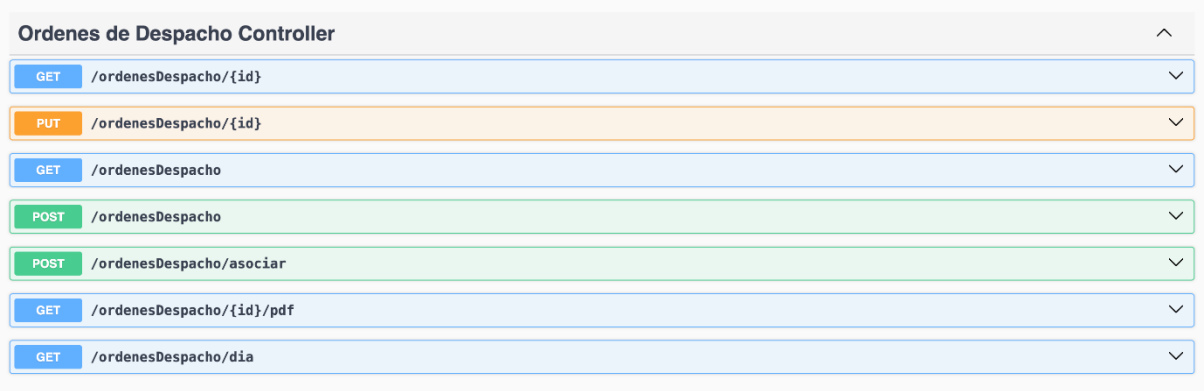


Figura 41: Endpoints órdenes de despacho

Fuente: Elaboración propia, basada en la práctica.

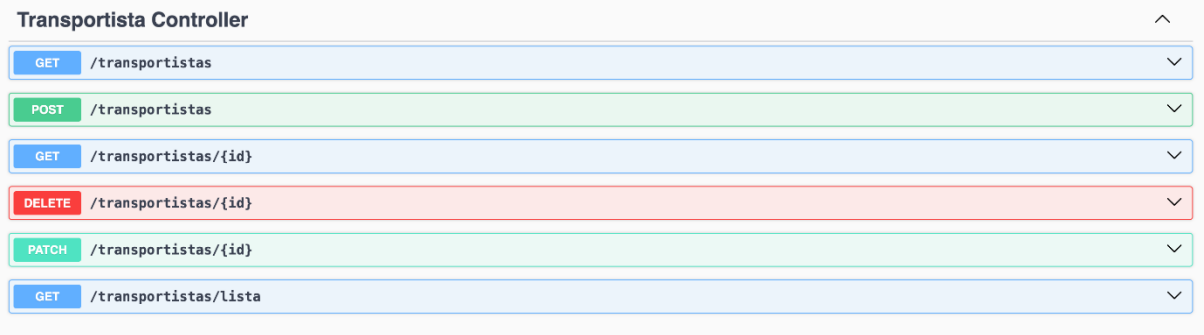


Figura 42: Endpoints transportistas

Fuente: Elaboración propia, basada en la práctica.

Cada uno de estos endpoints muestra una serie de detalles al desplegarlos. Para el caso del endpoint *GET /ordenesDespacho,* indica que tiene cinco parámetros opcionales, el código HTTP de respuesta y el cuerpo de la respuesta.

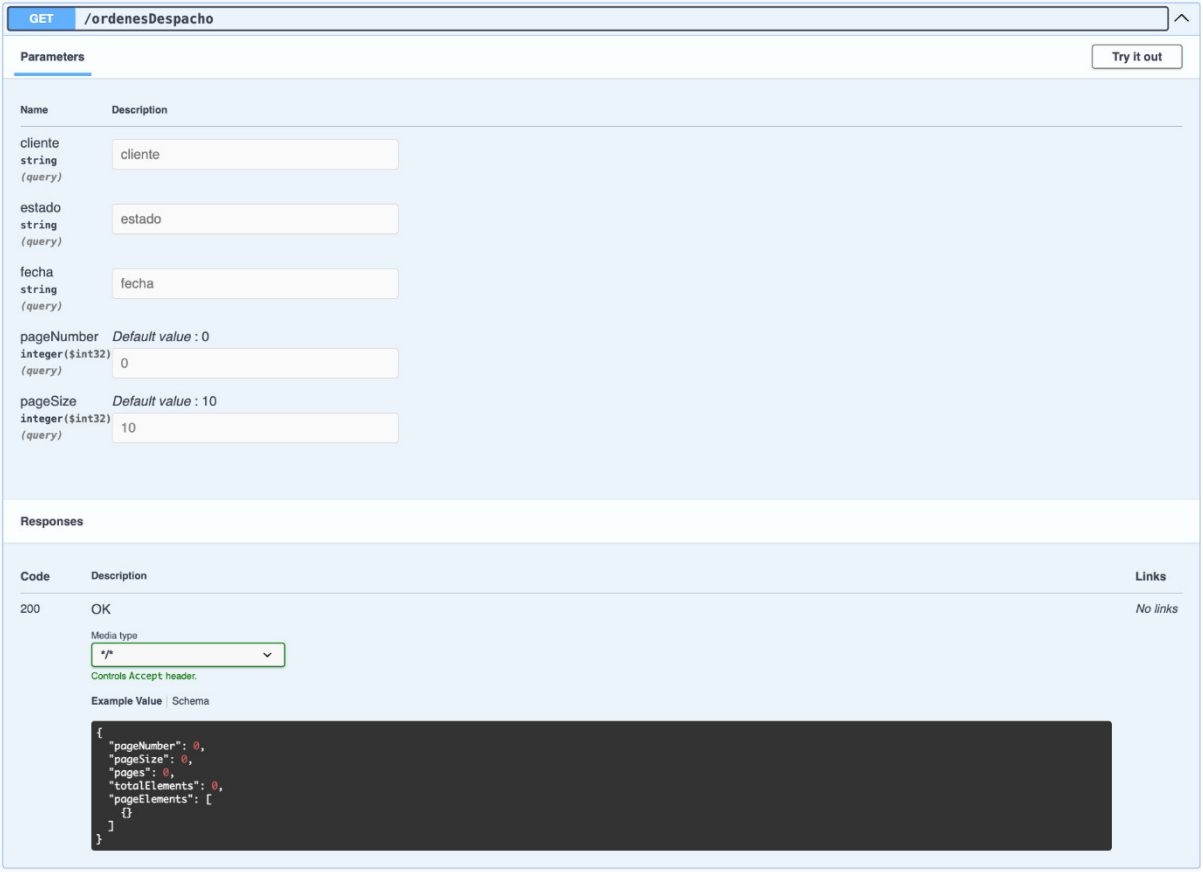


Figura 43: Endpoint órdenes

Fuente: Elaboración propia, basada en la práctica.

Otro ejemplo, es el *POST /ordenesDespacho* el cual no posee parámetros, pero indica el cuerpo de la petición.

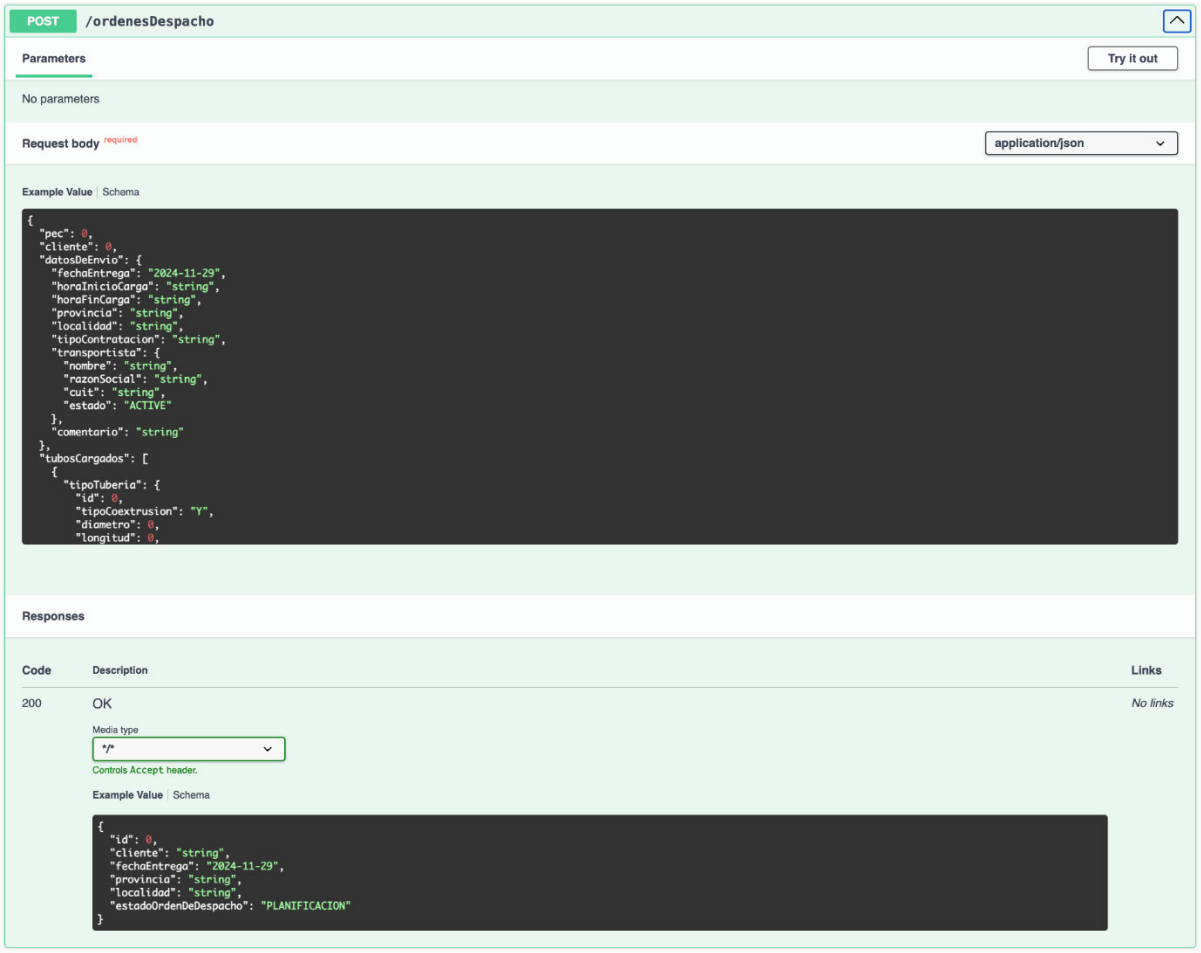


Figura 44: Endpoint creación orden

Fuente: Elaboración propia, basada en la práctica.

Contar con esta documentación es fundamental para los desarrolladores frontend, ya que les proporciona una guía clara de cómo interactuar con los servicios del backend.

10.8. Diagrama de Gantt

El siguiente diagrama de Gantt del proyecto muestra las diferentes tareas realizadas a lo largo del desarrollo del módulo de gestión de órdenes de despacho.

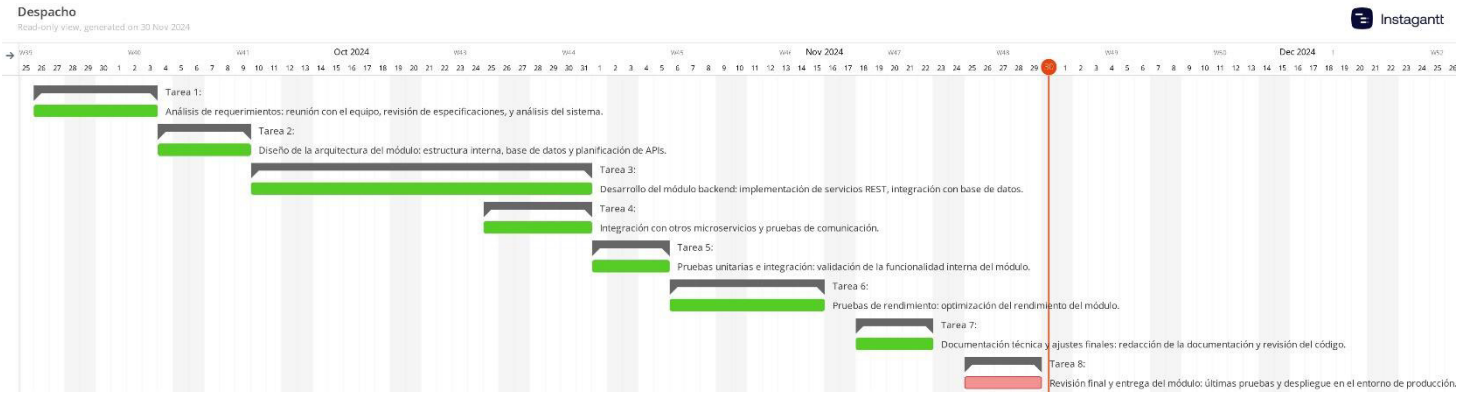


Figura 45: Diagrama de Gantt

Fuente: Elaboración propia, basada en la práctica.

11. Conclusiones

A través del desarrollo del módulo de gestión de órdenes de despacho para Krah, la Práctica Profesional Supervisada permitió abordar un proyecto integral desde el diseño, implementación y validación final. Este trabajo permitió cumplir con los objetivos propuestos, como la optimización de procesos logísticos, la trazabilidad de las órdenes y la integración con sistemas existentes en una arquitectura de microservicios.

El desarrollo del módulo de órdenes de despacho permitió lograr avances importantes como la formalización de las reglas de negocio, la creación de diagramas de arquitectura, diagramas de casos de usos y un diagrama de clases UML, que permiten tener una visión de los componentes que hacen parte del proyecto.

La experiencia adquirida fue positiva en muchos aspectos. En lo técnico, la implementación de herramientas y tecnologías actuales del mercado laboral, como Java, Spring Boot, Docker y Git, consolide habilidades prácticas esenciales para el desarrollo personal. Además, el uso de una metodología ágil como Scrum proporcionó una visión de la importancia de la organización, planificación y colaboración en un equipo de trabajo.

No obstante, el proyecto no estuvo exento de desafíos. Uno de los problemas recurrentes fue la demora en la respuesta del cliente para resolver las plantadas a lo largo del desarrollo. Para superar esta problemática, se realizaron una serie de reuniones donde se priorizó la resolución de preguntas críticas. Además, se solicitó documentación detallada sobre las reglas de negocio, lo que ayudó a reducir la dependencia de respuestas inmediatas y permitió mantener el ritmo de desarrollo.

En conclusión, este proyecto no solo cumplió con los objetivos planteados, sino que brindó una solución integral y sostenible que contribuyó directamente a la eficiencia y trazabilidad de los procesos logísticos de la empresa.

12. Reflexión sobre la Práctica Profesional Supervisada como espacio de formación

Durante la Práctica Profesional Supervisada, tuve la oportunidad de aplicar mis conocimientos de Ingeniería en Informática en un proyecto real para Krah. Desarrollé un módulo que optimiza la gestión de órdenes de despacho en la fabricación de tuberías usando tecnologías modernas y un enfoque ágil.

Fue una experiencia muy valiosa el poder vivir el ciclo de desarrollo completo, desde la planificación y el análisis de requerimientos, pasando el diseño, el desarrollo y las pruebas. Esta perspectiva integral me permitió comprender mejor cómo se conectan las diferentes etapas y la importancia desde cada una de ellas. Antes de esta experiencia, no había tenido la oportunidad de abordar un proyecto de manera tan estructurada y completa.

En conclusión, la Práctica Profesional Supervisada es una experiencia invaluable que nos permite experimentar lo que significa ser un profesional en Ingeniería en Informática. Nos prepara para lidiar con aspectos técnicos y humanos, y nos da la posibilidad de crecer tanto en conocimientos como en habilidades personales. Esta vivencia reafirma la importancia de combinar lo aprendido en el aula con la experiencia práctica, sentando las bases para una transición exitosa hacia el mundo laboral.

13. Bibliografía

Diagrama de clases. Teoria y ejemplos. (s.f.). DiagramasUML.com. Recuperado de https://diagramasuml.com/diagrama-de-clases/

Git - Fundamentos de Git. (s.f.). Recuperado de https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git

Hashemi-Pour, C., & Riglian, A. (2024, 27 de septiembre). What is Jenkins and How Does It Work? | Definition from TechTarget. Search Software Quality. Recuperado de https://www.techtarget.com/searchsoftwarequality/definition/Jenkins

IBM. (s.f.). ¿Qué es Java Spring Boot? | IBM - United States. Recuperado de https://www.ibm.com/mx-es/topics/java-spring-boot

Metodologías ágiles: ¿Qué son y cuáles son más utilizadas? (s.f.). Thinking for Innovation. Recuperado de https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

MinIO Object Storage for Container — MinIO Object Storage for Container. (s.f.). MinIO | S3 Compatible Storage for AI. Recupero de https://min.io/docs/minio/container/index.html

Portainer architecture | Portainer Documentation. (s.f.). Recuperado de https://docs.portainer.io/start/architecture

PostgreSQL 14.15 Documentation. (s.f.). PostgreSQL Documentation. Recuperado de https://www.postgresql.org/docs/14/index.html

¿Qué es Java? - Explicación del lenguaje de programación Java - AWS. (s.f.). Amazon Web Services, Inc. Recuperado de https://aws.amazon.com/es/what-is/java/

¿Qué es scrum y cómo empezar? (s.f.). Collaboration software for software, IT and business teams. Recuperado de https://www.atlassian.com/es/agile/scrum

¿Qué son los microservicios? | AWS. (s.f.). Amazon Web Services, Inc. Recuperado de https://aws.amazon.com/es/microservices/

Spring Boot. (s.f.). Spring Boot. Recuperado de https://spring.io/projects/spring-boot

Tutorial de diagramas de casos de uso (Guía con ejemplos). (s.f.). Creately Blog. Recuperado de https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/

Welcome | Portainer Documentation. (s.f.). Recuperado de https://docs.portainer.io/

What is Docker? (s.f.). Docker Documentation. Recuperado de https://docs.docker.com/get-started/docker-overview/