FACULTAD DE INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

Informe N°2 Arquitectura de Sistemas: Propuesta de Arquitectura Futura

Sección: 1

Profesor: Juan Gana

24 de junio de 2025

Integrantes:

Guillermo González

Tomás Poblete

Diego Gómez

Manuel Vergara

Juan de Dios Godoy

1. Evaluación Inicial (PoC)

Como la situación actual se trata de un sistema de registro manual de paquetes, las pruebas de estrés que se pueden realizar tienen que ver con la capacidad del conserje para hacer los registros, y cómo esta se ve afectada frente a varias situaciones críticas. Si se quisiera evaluar el desempeño del conserje se podría analizar el tiempo requerido para registrar los pedidos, así como medir el porcentaje de errores de registro y de entrega de estos. A continuación se presenta una tabla con situaciones críticas que estresan al sistema actual (manual) y cómo influyen en este:

Prueba de estrés	Descripción	Resultados observados
Llegada masiva de paquetes.	Simular ingreso de 30 paquetes en 1 hora. (Ej: en horario de almuerzo).	Tiempo de registro y errores.
Interrupciones.	Registrar paquetes mientras se atienden a residentes o se realizan otras tareas de gestión.	Retrasos y omisiones.
Pérdida del registro manual.	Simular daño o pérdida del cuaderno.	Tiempo de recuperación.
Consulta simultánea de residentes.	Poner a varios residentes retirando paquetes al mismo tiempo.	Tiempo de respuesta y errores de entrega.
Cambio de turno.	El personal nuevo debe continuar con el registro previo.	Facilidad de uso y continuidad en el proceso.

Tabla 1: Pruebas de estrés

Esta simulación deja en evidencia que el sistema actual funciona con lo justo y depende completamente de la memoria, el criterio y la disponibilidad del conserje. Cuando el ritmo de trabajo sube, como ocurre en edificios con muchas entregas diarias, el modelo manual no da abasto y empiezan a aparecer errores, olvidos y retrasos.

Esto confirma que es necesario avanzar hacia un sistema digital que se haga cargo de los puntos críticos del proceso, automatice lo repetitivo, entregue trazabilidad y facilite el trabajo tanto para el conserje como para los residentes. A partir de estas observaciones, se diseñó la propuesta de arquitectura futura que se presenta a continuación.

2. Propuesta de arquitectura To-Be

En esta sección se presenta la arquitectura futura del sistema, conocida como TO-BE. A partir del levantamiento realizado en la entrega anterior, donde se identificaron brechas importantes del sistema manual actual, se propone una nueva estructura que busca dar respuesta a las necesidades reales de los usuarios y mejorar los atributos de calidad priorizados, como la trazabilidad, la seguridad, la usabilidad y la disponibilidad.

Esta propuesta se compone de tres partes fundamentales: una arquitectura de procesos que digitaliza el flujo de trabajo actual, una arquitectura de aplicaciones y datos que define cómo se estructura la lógica interna del sistema, y una arquitectura de infraestructura que permite entender en qué entorno operará la solución, incluyendo elementos como la nube, dispositivos de los usuarios y servicios externos. Cada una de estas capas se detalla a continuación con su respectivo diagrama y explicación.

2.1 Arquitectura de Procesos

La arquitectura de procesos representa cómo funcionará el sistema una vez digitalizado. Se describe el flujo principal de trabajo que va desde la recepción del paquete hasta su retiro y eventual gestión de reclamos. Este proceso fue diseñado pensando en facilitar el trabajo del conserje, mejorar la experiencia del residente y garantizar mayor control y trazabilidad desde la consejería, que cumple el rol operativo de administrador del sistema.

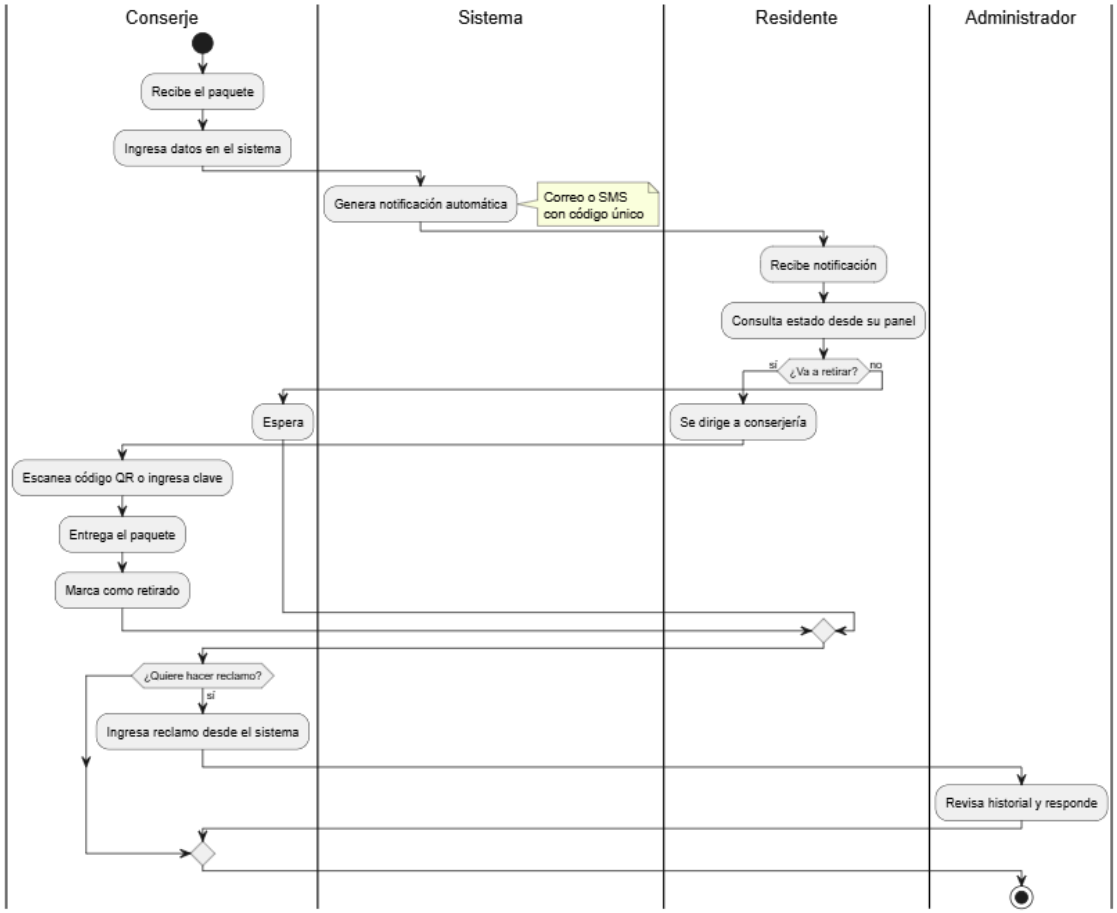

Figura 1: Diagrama Arquitectura de Procesos

El flujo comienza cuando el conserje recibe un paquete. En vez de anotarlo en papel, ahora lo ingresa al sistema, indicando el número de departamento y una pequeña descripción. El sistema genera automáticamente una notificación para el residente, que llega a través de un bot de WhatsApp e incluye un código único para el retiro.

El residente, al recibir la notificación, tiene la opción de revisar su panel personal para confirmar que el paquete está disponible. Cuando decide retirarlo, se presenta en conserjería, donde el conserje valida el código QR o la clave que recibió. Si todo está correcto, se entrega el paquete y se marca como retirado en el sistema. En caso de algún problema, el residente puede ingresar un reclamo, que será gestionado por el administrador del edificio.

Este nuevo proceso busca minimizar errores, reducir la carga manual del conserje y entregar mayor claridad y seguridad tanto para el residente como para la administración.

2.2 Arquitectura de Aplicaciones y Datos

Esta arquitectura permite visualizar cómo se estructuran internamente las aplicaciones del sistema y cómo interactúan con la base de datos. A diferencia del proceso manual, esta propuesta contempla un sistema donde tanto residentes como conserjes acceden a interfaces diferenciadas, cada una adaptada a sus funciones. La lógica de negocio se gestiona desde un backend centralizado, el cual está dividido en módulos según las funcionalidades del sistema.

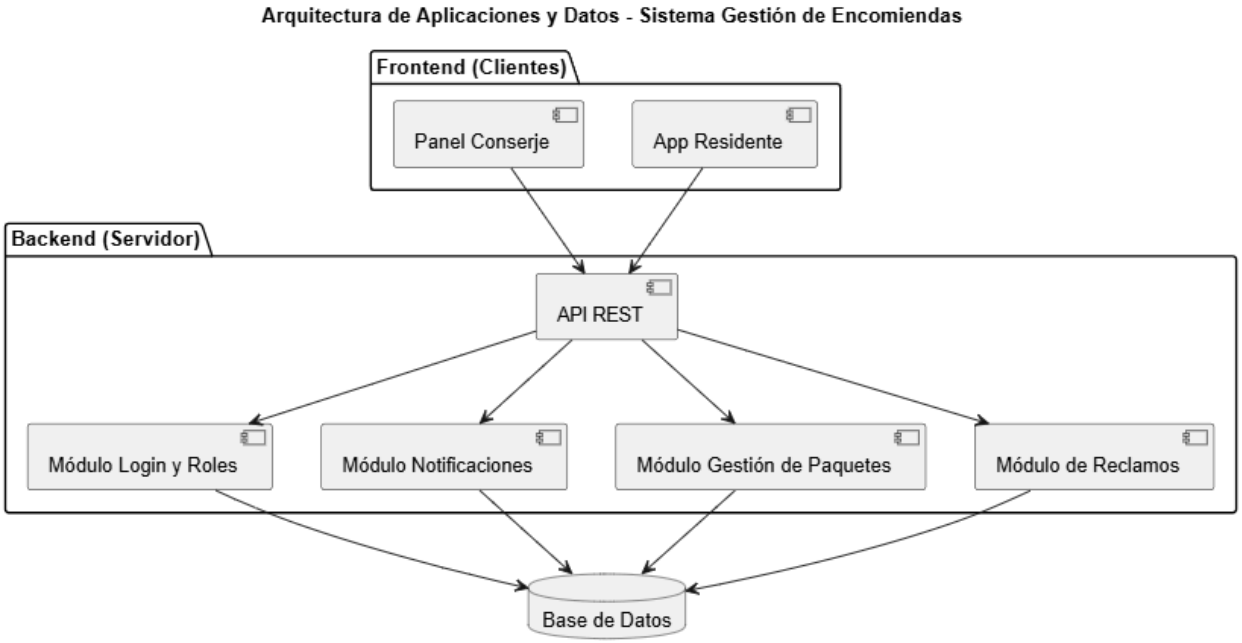

Figura 2: Diagrama Arquitectura de Aplicaciones y Datos

El sistema se divide en tres capas principales: interfaz de usuario (frontend), lógica del negocio (backend) y almacenamiento (base de datos).

Por un lado, los residentes acceden a una aplicación móvil o web donde pueden ver el estado de sus encomiendas, recibir notificaciones, y en caso necesario, generar reclamos.

Por otro lado, los conserjes acceden a un panel administrativo, desde donde pueden registrar paquetes, validarlos al momento del retiro, y gestionar reclamos ingresados por los usuarios.

Ambos actores interactúan con una API REST central, que distribuye las solicitudes hacia los distintos módulos del sistema: uno para el inicio de sesión y gestión de roles, otro para la administración de los paquetes, uno encargado de enviar notificaciones automáticas, y finalmente un módulo para el manejo de reclamos.

Toda la información es almacenada en una base de datos única, que contiene usuarios, historial de entregas, claves de validación, notificaciones enviadas y reclamos abiertos o resueltos.

2.3 Arquitectura de Infraestructura

Esta capa busca mostrar de forma clara el entorno técnico donde se desplegará el sistema. A diferencia del proceso manual, que depende completamente de medios físicos como cuadernos y llamadas, esta propuesta considera una infraestructura digital basada en tecnología accesible, segura y simple de mantener.

El sistema se apoyará en una arquitectura web, donde el frontend será accesible desde navegadores o dispositivos móviles. El backend estará alojado en la nube, permitiendo alta disponibilidad, escalabilidad y facilidad para mantener los servicios activos. La base de datos también residirá en servidores cloud, asegurando respaldo, integridad y trazabilidad de la información. Además, se incorporan servicios externos para el envío de notificaciones automatizadas mediante WhatsApp, gestionadas por un bot conectado al sistema, y lectores QR o códigos únicos para validar la entrega de paquetes en terreno.

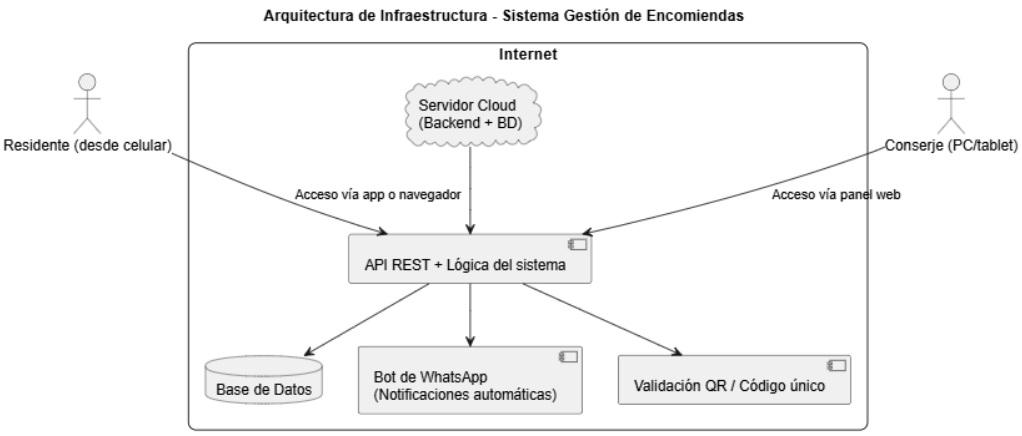

Figura 3: Diagrama Arquitectura de Infraestructura

El diagrama muestra cómo los dos actores principales (residente y conserje) interactúan con el sistema a través de Internet. Ambos se conectan con una API central alojada en la nube, la cual concentra la lógica del sistema. Esta API se comunica directamente con la base de datos y también con servicios externos, como el módulo de envío de notificaciones automatizadas por WhatsApp y el sistema de validación QR.

Esta infraestructura permite que el sistema sea accesible desde cualquier lugar con conexión, tenga buena disponibilidad y escalabilidad, y reduce la dependencia de equipos locales complejos o mantenimiento técnico especializado dentro del edificio. Además, el uso de un bot de WhatsApp para las notificaciones automatizadas mejora la experiencia del usuario, ya que es un canal directo, familiar y de alta efectividad para los residentes.

2.4 Cierre de la propuesta arquitectónica

Los tres niveles presentados en esta propuesta, que incluyen procesos, aplicaciones y datos, e infraestructura, forman una arquitectura clara, concreta y realista. El objetivo no es solo digitalizar lo que hoy se hace de forma manual, sino también mejorar la experiencia de quienes usan el sistema, haciéndolo más claro, más seguro y con menos margen de error. Cada decisión fue tomada en base a los problemas identificados en la entrega anterior, con la intención de construir una solución que pueda crecer con el tiempo y mantenerse sin mayores complicaciones.

3. Justificación de decisiones arquitectónicas

La arquitectura futura propuesta no surge de manera aislada, sino como una respuesta directa a las brechas identificadas en el análisis del sistema manual actual. A lo largo del informe anterior se evidenció que el modelo basado en registros en papel y verificación informal presenta limitaciones importantes, tanto en eficiencia como en seguridad. Por eso, cada una de las decisiones tomadas en el diseño del sistema digital busca abordar de forma concreta esas debilidades y mejorar los atributos de calidad que se definieron como prioritarios: trazabilidad, seguridad, usabilidad y disponibilidad.

Una de las brechas más importantes detectadas fue la necesidad de que el conserje conozca a los residentes para evitar entregar paquetes a personas equivocadas. Esta carga, que recae completamente sobre la memoria del conserje, se soluciona en el nuevo sistema mediante la validación con códigos únicos o QR, que permiten confirmar de forma segura la identidad de quien retira la encomienda.

También se observó que el registro manual, hecho con lápiz y papel, es propenso a errores de legibilidad y pérdida de información. En el diseño propuesto, estos registros pasan a un formato digital claro, con fuentes legibles, almacenamiento ordenado y acceso rápido. Esto no solo mejora la calidad de los datos, sino que permite consultarlos en cualquier momento desde una base de datos en la nube.

Otro punto crítico fue la forma en que los residentes se enteran de que llegó un paquete. Al depender del citófono o del recuerdo del conserje para avisar más tarde, existe un alto riesgo de que la notificación nunca llegue. La arquitectura futura incluye un bot de WhatsApp para el envío automático de alertas, asegurando que el residente reciba la información de forma inmediata y por un canal que ya usa en su día a día.

En conjunto, estas decisiones permiten reducir la carga sobre el conserje, mejorar la experiencia del residente y entregar una solución más organizada, con respaldo, control y escalabilidad. El paso a una arquitectura digital no busca simplemente modernizar el sistema, sino resolver problemas reales y repetitivos que afectan directamente la operación diaria en el edificio.

4. Mejoras a nivel de código y patrones

Teniendo en cuenta que la arquitectura as-is era completamente manual, no existen líneas de código previas sobre las cuales realizar cambios. Sin embargo, en esta sección se describen los patrones de arquitectura y diseño definidos para el sistema to-be, que guiarán su desarrollo técnico.

Dado que el sistema opera en un entorno de baja demanda y sus principales funcionalidades dependen de operaciones sobre una base de datos, se optó por implementar el patrón arquitectónico cliente-servidor. En esta arquitectura, el cliente (frontend) se encarga de mostrar la información y enviar solicitudes, mientras que el servidor (backend) gestiona la lógica de negocio y el acceso a la base de datos. Este enfoque no solo reduce la complejidad y los costos de desarrollo en comparación con arquitecturas más escalables como microservicios, sino que también se ajusta mejor a la naturaleza interdependiente de las funcionalidades del sistema, como el registro y retiro de paquetes mediante códigos QR. Además del enfoque cliente-servidor, este sistema también responde a una arquitectura en capas, donde se distingue de forma clara la capa de presentación (frontend), la lógica de negocio (backend) y la capa de datos (base de datos). Esta separación favorece la mantenibilidad, la escalabilidad y la claridad del desarrollo.

A nivel de código, se introdujeron mejoras significativas como la digitalización del proceso de inscripción de paquetes, la automatización de notificaciones al destinatario y el registro sistemático de retiros. Estas mejoras permitieron reducir el tiempo promedio del proceso completo, que anteriormente podía variar entre cinco y quince minutos en formato manual, a un máximo de cinco minutos, aumentando así la eficiencia operativa y la trazabilidad del servicio.

En cuanto a tecnologías, este componente está desarrollado en TypeScript y usa React como biblioteca base, dentro del framework Next.js para el frontend. La interacción con el backend (servidor desarrollado en Deno) y la base de datos PostgreSQL permite estructurar una arquitectura cliente-servidor, donde este componente representa la capa de presentación del cliente. La UI también se apoya en la biblioteca de íconos Lucide-react y utiliza clases utilitarias de Tailwind CSS para estilización, lo que contribuye a una experiencia de usuario moderna y dinámica.

También utilizamos patrones de diseños para reutilizar componentes, como en el componente PackageDisplay, implementa el patrón de diseño Componente Presentacional (Presentational Component), característico del paradigma de diseño componente-contenedor, donde se encapsula la lógica de presentación y renderizado de un paquete sin acoplarse directamente a fuentes de datos ni lógica de negocio compleja. Este patrón favorece la reutilización, mantenibilidad y claridad del código.

Cabe destacar, que el repositorio y su respectiva imagen Docker, se encuentran en el **Anexo 6.2 Repositorio y despliegue técnico.**

5. Discusión y conclusiones

La transformación arquitectónica propuesta es una reconfiguración del modelo operativo original, donde se tomaron decisiones de diseño que reflejaran consideración de los usuarios y agentes relacionados.

Tras la evaluación inicial en el informe anterior, la dependencia del sistema era exclusiva del conserje, teniendo en cuenta factores como su disponibilidad, memoria, capacidad de organización, experiencia, etc. Esta dependencia es riesgosa, porque en caso de un momento de alta carga operativa (como un cyberweek) y otras situaciones inesperadas como pérdidas de registro o cambios de turnos, esa carga no se redistribuye, haciendo que el proceso sea más complicado.

Nuestra propuesta ofrece transformar el frustrante proceso manual a un formato digital, incorporando códigos QR, notificaciones automáticas, y un backend full stack integrado y centralizado en el registro de paquetes de usuarios, la autenticación con cookies, entre otras características. De esta forma, se distribuye la carga, automatizando las tareas repetitivas, garantizando la trazabilidad de las acciones. Estas decisiones arquitectónicas realizadas tienen como objetivo facilitar el trabajo de los conserjes de edificios, mejorar la experiencia de los residentes al tener una noción precisa del estado de sus paquetes y una continuidad operacional en caso de fallas del sistema.

A nivel de código se añadieron mejoras relevantes, en la sistematización del proceso de retiro, lo cual busca mejorar específicamente todos los tiempos operativos. La implementación de patrones como el Componente Presentacional en módulos clave del frontend han permitido una reutilización y claridad en el código.

La incorporación de patrones modernos de infraestructura prepara el sistema para escalar a contextos similares, como otros edificios o condominios, evitando tener que rediseñar la arquitectura. Además, la arquitectura en la nube y el uso de Deno en el backend también brinda escalabilidad, lo cual lo hace más sostenible a través del tiempo

La usabilidad también es un factor clave, ya que se buscaba que la interfaz fuese diseñada para usuarios con poca experiencia digital, y justamente la implementación de validaciones sencillas

como códigos QR, y una interfaz hecha con herramientas como TypeScript, React, Nextjs y Tailwind, responde a esa necesidad.

En conclusión, nuestra propuesta de diseño arquitectónico responde a los problemas detectados y se logra adaptar correctamente a condiciones de uso reales. Es una implementación que no es únicamente una solución tecnológica, sino que también es una herramienta escalable y funcional. Se optimiza el proceso de gestión de paquetes y se establece una mejora continua del sistema.

6. Anexo

6.1 Trabajo realizado por cada miembro

- Evaluación Inicial (PoC): Juan de Dios Godoy
- Propuesta de Arquitectura To-Be: Tomás Poblete
- Justificación de decisiones arquitectónicas: Diego Gómez
- Mejoras a nivel de código y patrones: Guillermo González
- Discusión y conclusiones: Manuel Vergara

6.2 Repositorio y despliegue técnico

La arquitectura de procesos representa cómo funcionará el sistema una vez digitalizado. Se describe el flujo principal de trabajo que va desde la recepción del paquete hasta su retiro y eventual gestión de reclamo.

- Repositorio Github (frontend, backend y automatizaciones): https://github.com/GGonzalezGG/PP05H-ges-paq-edificios
- Imagen Docker disponible en DockerHub (backend): https://hub.docker.com/repository/docker/ggonzalezgg/gestor-paquetes-edificios/general

6.3 Desarrollo del código

El código fuente de este proyecto fue desarrollado previamente por Guillermo González y Juan de Dios Godoy para el ramo de Programación Profesional, por lo que ya estaba implementado antes de comenzar este informe. Esto facilitó el trabajo técnico en esta entrega, aunque también significó que no todos los integrantes pudimos participar directamente en el repositorio, debido a restricciones de grupos en el curso original.

Aun así, trabajamos de forma colaborativa revisando el código, validando los patrones utilizados y asegurándose de que el sistema cumpliera con los objetivos definidos según nuestra propuesta arquitectónica. La calidad de este trabajo refleja nuestro esfuerzo en conjunto, más allá de quién escribió cada línea de código.

6.4 Resumen técnico del desarrollo

El sistema que usamos fue desarrollado con tecnologías modernas, pensando en que fuera fácil de mantener, que fuera escalable y con buena trazabilidad.

- Frontend:
  Está hecho en TypeScript, usando React con Next.js, y el diseño lo trabajamos con Tailwind CSS. Aplicamos patrones como "Presentational Components", que ayudan a que el código sea más claro y fácil de reutilizar.
- Backend:
  Lo desarrollamos en Deno, con una arquitectura en capas y una API REST. Incluye validación de paquetes con códigos QR y manejo de sesiones usando cookies seguras.
- Base de datos:
  Usamos PostgreSQL como base de datos relacional, para poder llevar un registro estructurado y trazable de usuarios, paquetes y acciones dentro del sistema.
- Notificaciones y automatización:
  Integramos un bot de WhatsApp que envía notificaciones automáticas cuando llega un paquete, lo que hace más rápida y efectiva la comunicación con los residentes.
- Contenerización y despliegue:
  El backend está empaquetado con Docker, lo que facilita su despliegue. La imagen está subida a DockerHub y lista para correr en un entorno en la nube.

6.5 Comentario final del equipo

Este proyecto fue un paso importante para dejar atrás el sistema manual y pasar a una solución digital que automatiza tareas repetitivas, mejora la trazabilidad y entrega una mejor experiencia para todos los usuarios. Aunque el código ya había sido desarrollado previamente para el ramo de Programación Profesional, lo adaptamos y lo reorganizamos siguiendo principios como la separación por capas, el uso de componentes reutilizables y una estructura clara entre frontend, backend y base de datos. La solución que armamos no está pensada solo para este caso puntual, sino que se puede aplicar fácilmente en otros edificios o contextos parecidos sin tener que hacer un nuevo código desde cero.