Logotipo

El contenido generado por IA puede ser incorrecto.

**PROCONNECT**

**Integrantes**

Javier Esteban Cruz Caucaly

Luna Alejandra Bautista Peña

Tania Nicol Reyes Ospina

Andrés Camilo Mejía Guerrero

Brayan Duwan Naranjo Gualdron

Kevin Santiago Cachaya Collazos

Jose Isais Barrios Machado

Gerson Santiago Saenz Gutiérrez

**Profesor**

Linda Caicedo Francoise

Universidad Libre

Ingeniería de Sistemas

**Tabla de Contenidos**

[1 Introducción 3](#_Toc207331432)

[1.1 Justificación 4](#_Toc207331433)

[2 Objetivos. 4](#_Toc207331434)

[2.1 Objetivo General 4](#_Toc207331435)

[2.2 Objetivo General 4](#_Toc207331436)

[3 Conceptos Básicos 4](#_Toc207331437)

[3.1 Estructura 4](#_Toc207331438)

[4 Patrones de Diseño 6](#_Toc207331439)

[4.1 Factory Method 6](#_Toc207331440)

[4.2 Builder 8](#_Toc207331441)

[4.3 Strategy 9](#_Toc207331442)

[4.4 Observer Pattern 10](#_Toc207331443)

[5 Microservicios 12](#_Toc207331444)

[5.1 Api Gateway. 12](#_Toc207331445)

[5.2 Login 13](#_Toc207331446)

[5.3 Professionals-Service 14](#_Toc207331447)

[5.4 Booking-Service 15](#_Toc207331448)

[5.5 Webapp Frontend 18](#_Toc207331449)

[5.6 PostgreSQL Database 20](#_Toc207331450)

[6 Contenedores 20](#_Toc207331451)

[7 Docker 22](#_Toc207331452)

[8 Kubernetes (K8s) 23](#_Toc207331453)

[9 Recomendaciones y Mejoras 25](#_Toc207331454)

[10 Conclusiones 25](#_Toc207331455)

[11 Referencias 26](#_Toc207331456)

**Tabla de** **Figura**

[Figura 1 Organización 5](#_Toc207329787)

[Figura 2 Factory Method 7](#_Toc207329788)

[Figura 3 Builder 8](#_Toc207329789)

[Figura 4 Strategy 9](#_Toc207329790)

[Figura 5 Observer 10](#_Toc207329791)

[Figura 6 Api Gateway 12](#_Toc207329792)

[Figura 7 Login 13](#_Toc207329793)

[Figura 8 Profesional 14](#_Toc207329794)

[Figura 9 Book 16](#_Toc207329795)

[Figura 10 Service 17](#_Toc207329796)

[Figura 11 Web 18](#_Toc207329797)

[Figura 12 Base de Datos 19](#_Toc207329798)

[Figura 13 Contenedores 20](#_Toc207329799)

[Figura 14 Docker 22](#_Toc207329800)

[Figura 15 Kubernets 23](#_Toc207329801)

**Tabla de Tablas**

[Tabla 1 3](#_Toc207329954)

# Introducción

**ProConnect** nace como una solución pensada para acercar a las personas con los profesionales que necesitan, de una forma práctica y confiable. La plataforma permite reservar citas y gestionar servicios fácilmente, apoyándose en una arquitectura moderna con **microservicios**, **Docker** y **Kubernetes**. Además, incorpora patrones de diseño y una organización clara de la información para que la experiencia del usuario sea simple, intuitiva y eficiente

## Justificación

Nace como una idea de una solución digital diseñada para conectar a los usuarios con profesionales en los distintos sectores, que permiten facilitar el acceso a estos servicios, ya sean para reservar, solicitar, o simplemente ser incluido y prestar los servicios de manera ágil, sencilla y confiable.

# Objetivos.

## Objetivo General

Desarrollar e implementar una plataforma Web que facilite la conexión entre profesionales y los usuarios que requieran un servicio.

## Objetivo General

Desarrollar e implementar una plataforma Web que facilite la conexión entre profesionales y los usuarios que requieran un servicio.

# 

# Conceptos Básicos

## Estructura

Mayor mente compuesta por la segmentación de espacios que incluye algunas partes como:

1. **Encabezado fijo (topbar):** marca ProConnect, enlace a login/salir.
2. **Breadcrumbs (migas):** franja superior bajo el header
3. **Contenedor principal:** zona de vistas.
4. **Vistas/Secciones:**
5. **Home**
6. **Login / Registro: con** dos flujos profesional y contratador
7. **Servicios**: tarjetas por tipo de servicio
8. **Profesionales**: tarjetas (nombre, bio, “Ver detalle”)
9. **Detalle del profesional**: información + botón “Solicitar cita”
10. **Solicitar cita**; formulario de reserva
11. **Dashboards** por rol: acciones rápidas (ver citas, gestionar perfil)

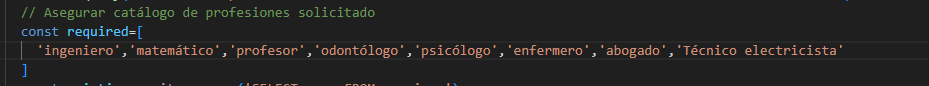
Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Figura Organización

En esta imagen se da a entender de como se muestra la organización general del aplicativo

Texto

El contenido generado por IA puede ser incorrecto.

# Patrones de Diseño

Los patrones de diseño son soluciones reutilizables y probadas para problemas comunes en el desarrollo de software. No son código específico, sino plantillas o guías que ayudan a estructurar el código de manera eficiente, mejorando su mantenibilidad, escalabilidad y reutilización

## Factory Method

Se usa para los datos de profesionales y servicios se cargan consistentemente dentro de la Interfaz para diferentes tipos de datos que se van guardando (usuarios, profesionales, reservas).

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

Figura Factory Method

En esta parte de la imagen se logra visualizaras diferentes servicios usan el mismo patrón para conectarse a las diferentes bases de datos

## Builder

El patrón Builder nos permite construir las tarjetas de profesionales mediante una secuencia clara de pasos para así facilitar el reutilizar el código después.

Texto

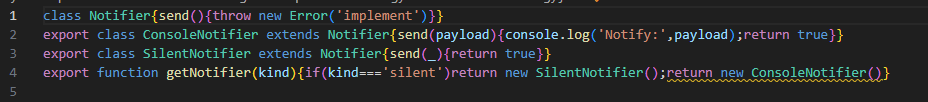
El contenido generado por IA puede ser incorrecto.



Figura Builder

En esta parte se evidencia como las tarjetas siguien la misma estructura para garantizar coherencia en la interfaz

## Strategy

Al crear una reserva, se ejecutan diferentes estrategias de notificación que permiten al usuario ver una confirmación de su reserva gracias al patrón.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Figura Strategy

La Imagen hace referencia al como al hacer una solicitud se ve el cómo llega una notificación de que su reserva la recibió el profesional

## Observer Pattern

Cuando un usuario crea una reserva, múltiples componentes reaccionan y al concluir se puede ver cómo se recibe una confirmación visual de la acción que realizo.

Captura de pantalla con letras blancas

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Figura Observer

La siguiente imagen muestra como se espera que el usuario vea una notificación de como la reserva ha sido exitosa

# Microservicios

Es un **estilo arquitectónico** donde una aplicación se divide en servicios pequeños, independientes y altamente cohesionados, donde cada microservicio se enfoca en una funcionalidad especifica ya sea para autentificación como un Login o un sistema de pagos

## Api Gateway.

Este funciona a través de las peticiones que se van realizando a través de los microservicios adecuados por lo que se van navegando.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura Api Gateway

En esta zona se visualiza como las peticiones se van a redigir a los microservicios adecuados que sirven para gestionar la reserva.

## Login

Este encargado de la verificación de los usuarios por el tipo de rol que cumplen dentro de la página y así les permite redirigirse hacia la interfaz a donde va

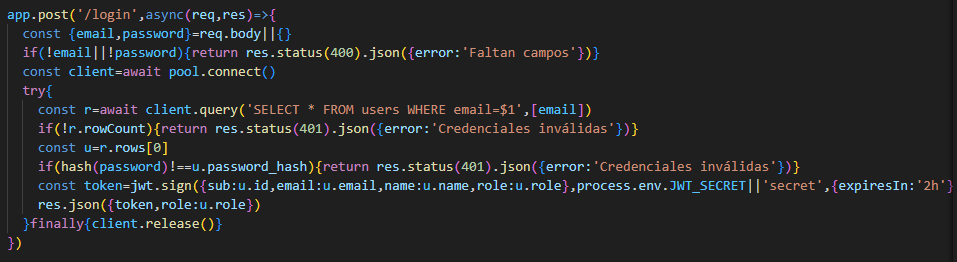




Figura Login

En esta parte de la pagina podremos darnos cuenta de cómo identificamos el microservicio que se encarga de validar si un usuario existe dentro de la base de datos.

## Professionals-Service

Esta tiene como principal fin administrar el catalogó de los profesionales que están disponibles para ofrecer sus servicios

Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Figura Profesional

## Booking-Service

En esta parte del servicio funciona ya sea para gestionar una reserva

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Figura Book



Figura Service

Tanto para la imagen 8 como 9 se puede distinguir como se esta comportando el servicio al mencionar las citas pendientes que tiene tanto el profesional como el usuario que espera que le acepten la petición.

## Webapp Frontend

Contiende todo lo que es la intefaz del usuario como visualiza toda la parte porque la que está navegando

Texto

El contenido generado por IA puede ser incorrecto.

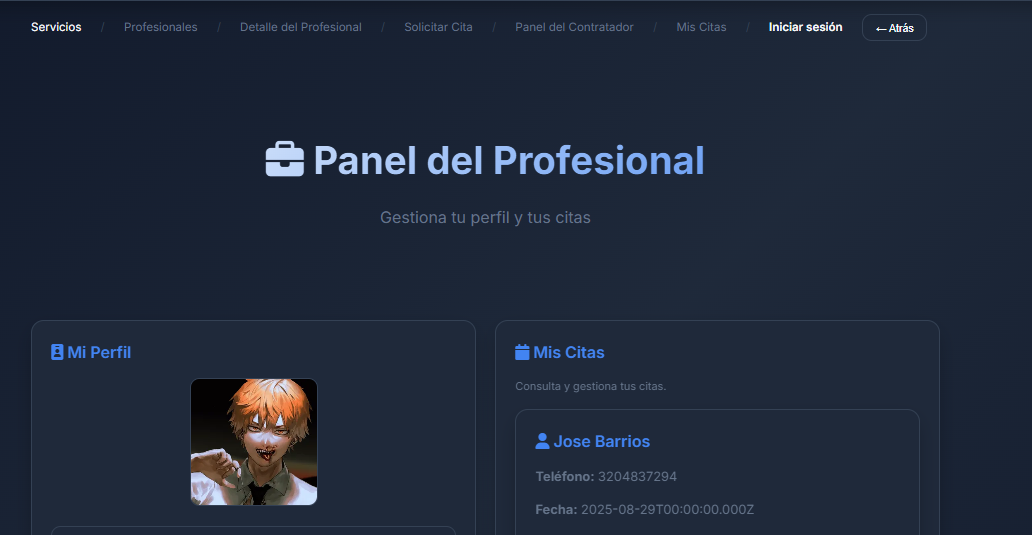


Figura Web

## PostgreSQL Database

En esta parte observamos como utilizamos una base de datos para guardar los datos de los usuarios que van ingresando en el aplicativo

Texto

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Figura Base de Datos

# Contenedores

Son **entornos aislados** que empaquetan una aplicación junto con sus dependencias (librerías, configuraciones, etc.). Permiten que el software se ejecute de manera consistente en cualquier entorno (desarrollo, producción).

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

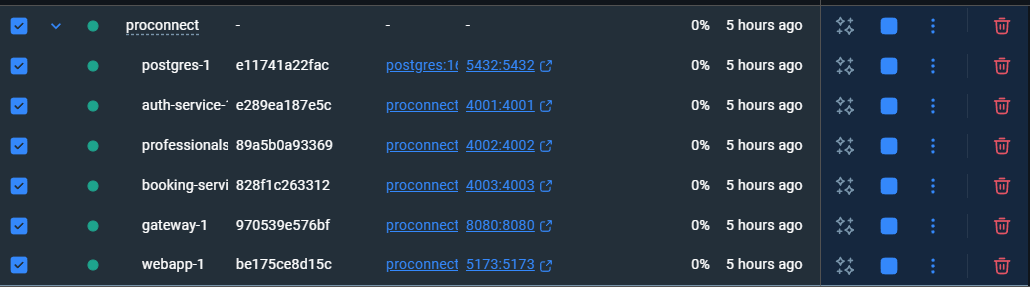


Figura Contenedores

En esta parte observamos como están creados los contenedores de los cuales disponemos para que el aplicativo funcione.

# Docker

Es la **plataforma más popular para crear y gestionar contenedores**. Automatiza el despliegue de aplicaciones empaquetándolas en contenedores estandarizados.



Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Figura Docker

Observamos como están construidos y almacenados para microservicio los dockers

# Kubernetes (K8s)

Es un **orquestador de contenedores** open-source que automatiza el despliegue, escalado y gestión de aplicaciones empaquetadas en contenedores, que tiene relación con Docker.

Kubernetes no depende de Docker, pero lo usa comúnmente para crear contenedores.

Se configura mediante archivos **YAML** llamados **manifiestos**, que definen **Deployments**, **Services**, **Ingress** y otros recursos.

Texto

El contenido generado por IA puede ser incorrecto.

Figura Kubernets

En esta parte se oberserva como esta siendo gestionado los kubernets a través de un cluster

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

# Recomendaciones y Mejoras

Seguridad

* Usar bcrypt o argon2 para almacenar contraseñas de forma segura.
* Mover secretos y llaves a *Kubernetes Secrets* o gestores externos como Vault.
* Implementar rate limiting en endpoints sensibles.

Infraestructura y Contenedores

* Endurecer Dockerfiles con builds multistage, uso de USER no root y .dockerignore.
* Evitar imágenes con :latest y usar versiones fijas.
* Definir resources.requests y resources.limits en cada deployment.

Escalabilidad y Resiliencia

* Implementar colas asíncronas para notificaciones y procesos tardados.
* Añadir retry y circuit breakers en llamadas entre servicios.

Calidad y Mantenimiento

* Incorporar validación de datos con librerías como Joi o Zod.
* Documentar APIs con OpenAPI/Swagger.

Observabilidad

* Configurar logging estructurado con correlation IDs.
* Agregar métricas (Prometheus) y trazas distribuidas (OpenTelemetry).

# Conclusiones

El desarrollo de **ProConnect** evidencia cómo la combinación de **microservicios**, **contenedores** y **patrones de diseño** puede dar origen a una plataforma robusta, modular y altamente adaptable. Gracias a su arquitectura distribuida, la aplicación no solo simplifica la interacción entre usuarios y profesionales, sino que también ofrece una base sólida para futuras extensiones y mejoras tecnológicas. La integración de principios de **arquitectura de la información** asegura una experiencia de usuario clara y coherente, mientras que la aplicación de patrones de diseño fomenta la reutilización de código y la reducción del acoplamiento. Asimismo, el uso de **Docker** y **Kubernetes** proporciona una solución escalable y portable que responde a los desafíos actuales de despliegue en entornos de producción.

# Referencias

Susnjara, S., & Smalley, I. (2025, 22 julio). Docker. *Docker*. https://www.ibm.com/think/topics/docker

*Kubernetes documentation*. (s. f.). Kubernetes. https://kubernetes.io/docs/home/

*Patrones de diseño / Design patterns*. (s. f.). https://refactoring.guru/es/design-patterns

Del Prado, J. A. (2020, 29 septiembre). *La Arquitectura de Información Web y sus tipos*. UXABLES | Blog. https://www.uxables.com/diseno-ux-ui/la-arquitectura-de-informacion-web-y-sus-tipos/

Braeuner, M. (2022, 18 julio). *Guía de Docker para principiantes: cómo crear tu primera aplicación Docker*. freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/guia-de-docker-para-principiantes-como-crear-tu-primera-aplicacion-docker/>

*«Deploy on Kubernetes»*. (2025, 6 agosto). Docker Documentation. https://docs.docker.com/desktop/features/kubernetes/