TRABAJO DE FIN DE GRADO

DESARROLLO DE APLICACIONES WEB

CARLOS ARANA JIMÉNEZ

IES FRANCISCO AYALA

El presente Trabajo de Fin de Grado consiste en el desarrollo de una aplicación web orientada a la gestión integral de una barbería. La plataforma permite a los usuarios registrarse, reservar citas y realizar pagos anticipados de manera sencilla e intuitiva. Además, ofrece funcionalidades específicas para los diferentes roles dentro del sistema: los administradores pueden gestionar usuarios, empleados, servicios y consultar listas de citas tanto diarias como mensuales; los empleados, por su parte, tienen acceso a sus respectivas agendas para organizar su jornada laboral de forma eficiente.

La aplicación ha sido desarrollada utilizando tecnologías modernas tanto en el frontend como en el backend. Para la interfaz de usuario se ha empleado Vue.js, lo que permite una experiencia dinámica y reactiva, mientras que el servidor ha sido construido con Node.js, garantizando un rendimiento robusto y escalable. La persistencia de datos se gestiona mediante una base de datos PostgreSQL, que asegura integridad y eficiencia en el manejo de la información.

Este trabajo abarca todo el ciclo de vida del desarrollo de software, desde la recogida de requisitos y diseño inicial hasta la implementación, pruebas y despliegue final, siguiendo buenas prácticas de desarrollo web. El objetivo principal ha sido crear una herramienta útil y fácil de usar tanto para los clientes como para el personal de la barbería, mejorando la organización y optimizando el tiempo de gestión.

This Final Degree Project focuses on the development of a web application aimed at the comprehensive management of a barbershop. The platform allows users to register, book appointments, and make advance payments in a simple and intuitive way. It also offers specific functionalities for different roles within the system: administrators can manage users, employees, and services, as well as view daily and monthly appointment lists; employees, in turn, have access to their own schedules to efficiently organize their workday.

The application has been developed using modern technologies on both the frontend and backend. The user interface was built with Vue.js, providing a dynamic and responsive experience, while the server was developed with Node.js, ensuring robust and scalable performance. Data persistence is handled through a PostgreSQL database, which guarantees integrity and efficiency in information management.

This project covers the entire software development lifecycle, from requirements gathering and initial design to implementation, testing, and final deployment, following best practices in web development. The main goal has been to create a useful and user-friendly tool for both clients and barbershop staff, improving organization and optimizing time management.

https://github.com/carlos-aj/TFG

rasoio.es

1. Justificación

1.1. Características generales

La aplicación consiste en una plataforma web orientada a la gestión y reserva de servicios en una barbería. Desde la página principal, sin necesidad de iniciar sesión, los usuarios pueden consultar información general sobre la barbería, su historia, el equipo de trabajo y acceder fácilmente al registro o inicio de sesión.

El sistema está estructurado en tres tipos de roles: usuario, empleado y administrador.

Una vez registrado, el usuario puede reservar una cita seleccionando la fecha, el servicio y el barbero deseado. Además, la plataforma permite incluir a un invitado mediante un sistema de reservas dobles, siempre que haya disponibilidad de citas contiguas. El usuario puede elegir entre pagar en el local o realizar un prepago online. Al confirmar una reserva, el sistema envía automáticamente un correo electrónico con todos los detalles de la cita.

Los empleados disponen de un panel donde pueden consultar el listado diario de citas, con información detallada sobre los servicios y los clientes. Cada cita puede marcarse como realizada, y en caso de inasistencia, el cliente puede ser sancionado. El sistema de penalizaciones funciona del siguiente modo:

- Primera falta: advertencia enviada por correo electrónico.
- Segunda falta: recargo económico en la próxima cita.
- Tercera falta: bloqueo temporal de la cuenta del usuario.

Las sanciones caducan automáticamente si transcurren tres meses sin nuevas inasistencias. En el caso de haber realizado prepago, el importe no se reembolsa si el cliente no asiste. Además, los empleados tienen acceso a un calendario donde pueden consultar de forma visual todas las citas programadas por día.

El administrador tiene acceso completo al sistema: puede gestionar usuarios, empleados y servicios, además de visualizar todas las citas diarias de cada trabajador y consultar el calendario global de la barbería.

La plataforma también cuenta con una galería donde se muestran imágenes de los diferentes servicios ofrecidos, con el objetivo de inspirar a los clientes y mostrar el trabajo realizado en el establecimiento.

1.2. Restricciones generales

Actualmente, la aplicación presenta algunas limitaciones funcionales que podrían considerarse en futuras versiones. No se ha implementado un sistema de recuperación de

contraseñas, por lo que, en caso de olvido, el usuario no puede restablecer su acceso de forma autónoma.

Tampoco se contempla la posibilidad de que empleados o administradores creen o gestionen citas en nombre de los usuarios, lo que limita la intervención del personal en situaciones excepcionales, como la atención telefónica o la gestión manual de reservas.

Además, el sistema no cuenta con funcionalidades avanzadas como la integración con calendarios externos, la personalización de horarios por empleado o la gestión automática de días festivos, aspectos que podrían mejorar la flexibilidad y escalabilidad del servicio.

1.3. Aspectos a cubrir

El proyecto cubre las funcionalidades principales necesarias para la gestión de una barbería, como el registro y autenticación de usuarios, la reserva de citas con posibilidad de prepago, la gestión de empleados y servicios, y la diferenciación de roles entre usuarios, empleados y administradores. También se incluye un sistema de penalizaciones por inasistencia, el envío de correos de notificación y una galería de servicios.

Por otro lado, se han dejado fuera ciertas funcionalidades que podrían desarrollarse en futuras versiones, como la recuperación de contraseñas, la creación de citas por parte del personal, la integración con calendarios externos, y un sistema de valoraciones. Estas limitaciones se deben principalmente a razones de alcance y tiempo.

1.4. Estudio de las prestaciones de la herramienta

La aplicación desarrollada se sitúa dentro del ámbito de plataformas web para la gestión y reserva de servicios en barberías y salones de belleza. Frente a otras soluciones comerciales y de código abierto disponibles en el mercado, esta herramienta destaca por su adaptación específica a las necesidades de un negocio local de barbería, con una estructura clara y roles diferenciados para usuarios, empleados y administradores.

A diferencia de plataformas genéricas que ofrecen funcionalidades amplias pero poco personalizadas, esta aplicación incorpora un sistema exclusivo de reservas con opción de invitado, un mecanismo de penalización por inasistencias adaptado a la operativa del negocio y una gestión simplificada que permite un control eficaz tanto para el personal como para los administradores.

2. Tecnología empleada

Frontend:

Vue.js 3: Framework JavaScript para interfaces de usuario reactivas
Vite: Herramienta de compilación rápida para desarrollo web
Vue Router: Sistema de navegación para aplicaciones de una sola página

Vuetify: Biblioteca de componentes Material Design para Vue
GSAP: Plataforma para animaciones web avanzadas
Stripe.js: Integración de pagos en el frontend
Vue Google Charts: Visualización de datos y gráficos

Backend:

Node.js: Entorno de ejecución JavaScript del lado del servidor
TypeScript: Superconjunto tipado de JavaScript
Express: Framework web minimalista para Node.js
PostgreSQL: Sistema de base de datos relacional
Knex.js: Constructor de consultas SQL para Node.js
Objection.js: ORM basado en Knex
JWT: Autenticación basada en tokens
Bcrypt: Cifrado de contraseñas
Multer: Manejo de subida de archivos
Nodemailer: Envío de correos electrónicos
Stripe API: Procesamiento de pagos

Infraestructura:

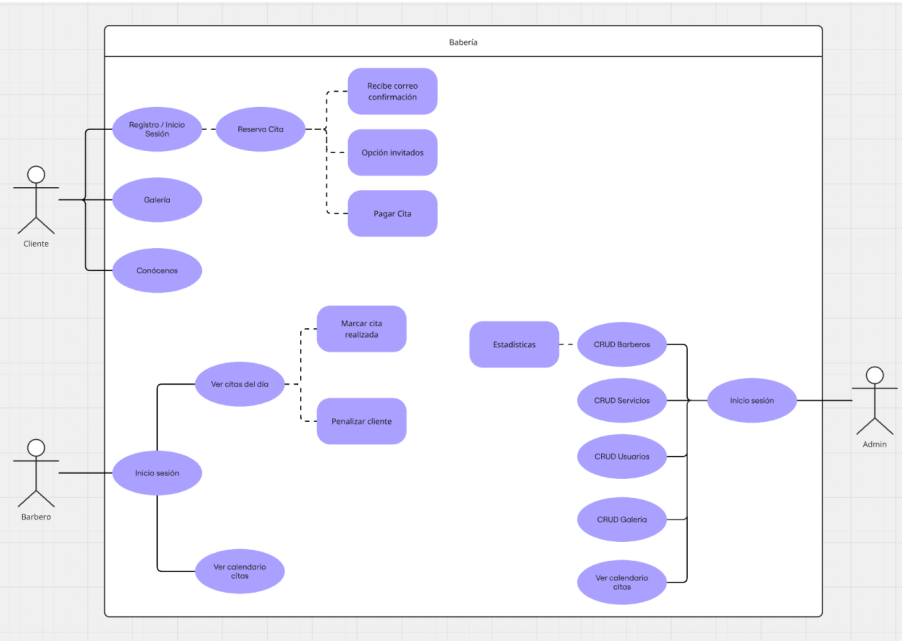
Docker: Contenedorización de aplicaciones (desarrollo)
Vercel: Plataforma de despliegue para frontend
Render: Plataforma de despliegue para backend
Railway: Plataforma de hosting para base de datos PostgreSQL

3. Requerimientos de hardware y software

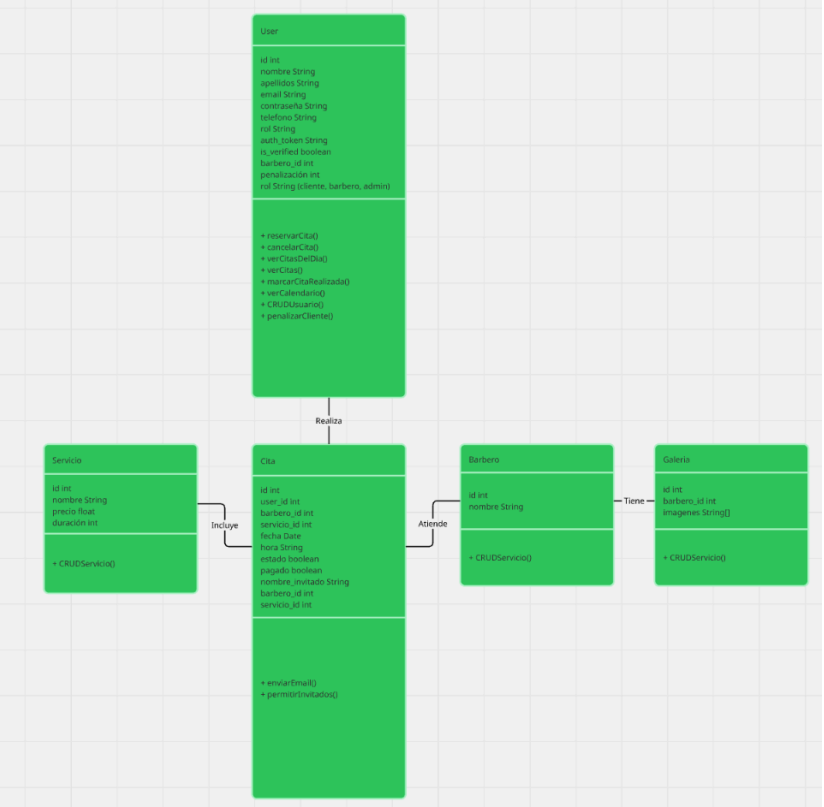
Los requerimientos de software se dividen según el tipo de ejecución, ya sea local o en producción.
Para la ejecución en producción, únicamente es necesario contar con una conexión a internet estable y un navegador web actualizado.
En cambio, para la ejecución en modo local, se requiere tener instalado Docker, así como instalar las dependencias mediante npm. Finalmente, para poner en marcha la aplicación en local, es necesario iniciar los contenedores utilizando docker-compose.

4. Análisis y diseño

4.1. Diagrama de casos de uso



4.2. Diagrama de clases



4.3. Descripción de base de datos

La base de datos tendrá la siguiente estructura:

Usuarios:

Estos son las personas que usarán la aplicación web y podrán tener roles como clientes, empleados o administradores. De ellos se almacena un identificador único llamado id, el nombre y apellidos, el correo electrónico que es único y se utiliza para la autenticación, la contraseña cifrada para garantizar un acceso seguro, el número de contacto telefónico, un contador de penalizaciones por faltas de asistencia, el tipo de usuario que puede ser 'user', 'empleado' o 'admin', un token de autenticación para las sesiones, un indicador que refleja si la cuenta está verificada, así como las fechas de creación y última modificación.

Barberos:

Son los profesionales que ofrecen los servicios en la barbería. De cada barbero se registra un identificador único (id), su nombre, la especialidad o técnica en la que destaca, junto con las fechas de registro y última actualización.

Servicios:

Estos corresponden a los servicios que se ofrecen en la barbería. Cada servicio tiene un identificador único, un nombre, el precio o coste asociado, la duración que representa el tiempo necesario para realizar el servicio expresado en horas, además de las fechas de creación y última modificación.

Citas:

Representan las reservas de servicios que realizan los usuarios. Cada cita incluye un identificador único, el cliente que realiza la reserva (user_id), el profesional asignado (barbero_id), el servicio a realizar (servicio_id), la fecha y hora programada, el estado de la cita que puede ser realizada o pendiente, un indicador que muestra si el servicio ha sido pagado, y opcionalmente el nombre de una persona invitada junto con el servicio, barbero y hora asignados para ese invitado. También se almacenan las fechas de creación y última modificación.

Galería:

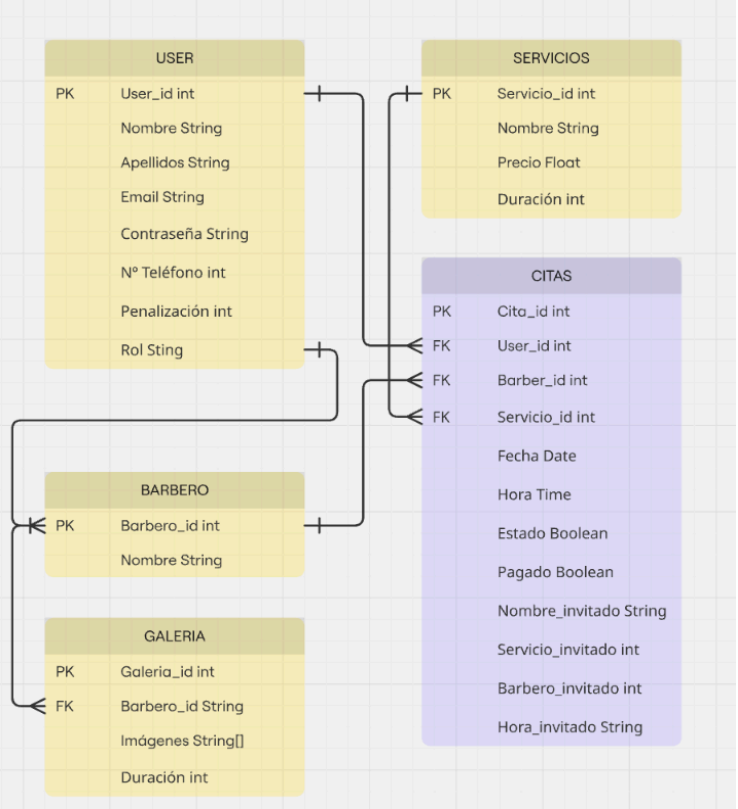
Almacena imágenes de los trabajos realizados por los barberos. Cada registro contiene un identificador único, la referencia al barbero asociado, una colección de rutas hacia las imágenes, y las fechas de subida y última actualización.

Relaciones:

Un usuario puede realizar múltiples citas, un barbero puede atender varias citas, un servicio puede estar incluido en muchas citas, y un barbero puede tener múltiples entradas en la galería. Además, un usuario con rol de empleado puede estar asociado a un registro de barbero.

La base de datos está diseñada para soportar todas las funcionalidades del sistema de gestión de barbería, abarcando desde la reserva de citas hasta la administración de usuarios, servicios y barberos.

4.4. Diagrama de base de datos



5. Implementación

A continuación se detalla y explica la estructura de archivos y directorios del proyecto de barbería, describiendo la función de cada componente y cómo se relacionan entre sí.

5.1. Archivos Raíz

docker-compose.yml: Archivo de configuración para Docker Compose que define y orquesta los tres servicios principales: backend (API REST en Node.js/Express), frontend (aplicación Vue.js) y base de datos PostgreSQL. Establece los puertos, volúmenes y variables de entorno necesarios para cada servicio, permitiendo un despliegue consistente en cualquier entorno.

5.2. Backend

El backend está implementado como una API RESTful usando Node.js, Express, TypeScript y PostgreSQL como base de datos, con Knex.js como constructor de consultas y Objection.js como ORM.

5.2.1. Archivos Raíz del Backend

knexfile.js: Archivo de configuración de Knex.js que define las conexiones a la base de datos para diferentes entornos (desarrollo, staging y producción). Configura los parámetros de conexión como host, puerto, usuario y contraseña, así como las rutas para las migraciones y semillas.
package.json: Define las dependencias del backend, incluyendo Express, TypeScript, Knex, Objection.js, bcrypt para cifrado de contraseñas, JWT para autenticación, Nodemailer para envío de correos, Multer para manejo de archivos, Stripe para pagos, entre otros. También define los scripts para iniciar, desarrollar y construir la aplicación.
tsconfig.json: Configuración del compilador TypeScript, especificando opciones como el target de compilación (ES2020), el módulo de salida (CommonJS), directorios de entrada y salida, y otras opciones de compilación.
dockerfile: Instrucciones para construir la imagen Docker del backend, incluyendo la instalación de dependencias, compilación del código TypeScript y configuración del entorno de ejecución.

5.2.2. Directorios del Backend

src/: Contiene todo el código fuente del backend organizado en una estructura modular.
index.ts: Punto de entrada de la aplicación que configura el servidor Express, establece middleware como CORS y cookie-parser, define las rutas API y configura el manejo de errores. También inicia la sincronización de secuencias de la base de datos.
controllers/: Implementan la lógica de negocio y manejan las solicitudes HTTP.
user.controller.ts: Gestiona operaciones relacionadas con usuarios como registro, inicio de sesión, actualización de perfil, y administración de usuarios. Implementa la autenticación con JWT y el cifrado de contraseñas con bcrypt.
cita.controller.ts: Maneja la creación, modificación, cancelación y consulta de citas. Implementa lógica para verificar disponibilidad, gestionar invitados y procesar pagos.
barbero.controller.ts: Gestiona operaciones relacionadas con barberos, como creación, actualización, eliminación y consulta de información de los profesionales.
servicio.controller.ts: Maneja los servicios ofrecidos por la barbería, incluyendo la creación, actualización, eliminación y consulta de servicios con sus precios y duraciones.
galeria.controller.ts: Gestiona la galería de imágenes, permitiendo subir, eliminar y consultar imágenes asociadas a barberos.
payment.controller.ts: Maneja los pagos con Stripe, procesando las transacciones y actualizando el estado de las citas.
upload.controller.ts: Gestiona la subida de archivos como imágenes de perfil y fotos para la galería.
routes/: Define las rutas de la API y asocia cada endpoint con su controlador correspondiente.
user.routes.ts: Rutas para gestión de usuarios (/api/user), incluyendo registro, inicio de sesión, cierre de sesión y administración de perfiles.
cita.routes.ts: Rutas para gestión de citas (/api/cita), permitiendo crear, consultar, actualizar y cancelar reservas.

barbero.routes.ts: Rutas para gestión de barberos (/api/barbero), para administrar los profesionales.
servicio.routes.ts: Rutas para gestión de servicios (/api/servicio), para administrar el catálogo de servicios.
galeria.routes.ts: Rutas para gestión de la galería (/api/galeria), incluyendo subida y eliminación de imágenes.
public.routes.ts: Rutas públicas, incluyendo webhooks de Stripe para procesar eventos de pago.
models/: Define los modelos de datos usando Objection.js, estableciendo la estructura de las tablas y las relaciones entre ellas.
User.ts: Modelo para usuarios con campos como nombre, apellidos, email, contraseña, rol, etc.
Barbero.ts: Modelo para barberos con campos como nombre y especialidad.
Servicio.ts: Modelo para servicios con campos como nombre, precio y duración.
Cita.ts: Modelo para citas con campos como usuario, barbero, servicio, fecha, hora, estado, etc.
Galeria.ts: Modelo para la galería con campos como barbero e imágenes.
middlewares/: Middleware para autenticación, autorización y seguridad.
auth.middleware.ts: Verifica tokens JWT y gestiona roles de usuario.
security.middleware.ts: Implementa protecciones de seguridad como rate limiting y validación de datos.
db/: Configuración de la conexión a la base de datos PostgreSQL usando Knex.js.
knex.ts: Configura la conexión a la base de datos y la integra con Objection.js.
utils/: Funciones de utilidad reutilizables.
emailSender.ts: Configura Nodemailer para enviar correos de confirmación y notificaciones.
tokenManager.ts: Gestiona la creación y verificación de tokens JWT.
types/: Definiciones de tipos TypeScript para mejorar el desarrollo y la seguridad del código.
express/: Extensiones de tipos para Express, incluyendo interfaces personalizadas para Request y Response.
ApiGaleria/: Directorio donde se almacenan físicamente las imágenes subidas para la galería.
migrations/: Archivos de migración de Knex que definen la estructura de la base de datos.
create_users_table.js: Crea la tabla de usuarios con campos como nombre, email, contraseña, etc.
create_barbero_table.js: Crea la tabla de barberos con campos como nombre y especialidad.
create_servicios_table.js: Crea la tabla de servicios con campos como nombre, precio y duración.
create_cita_table.js: Crea la tabla de citas con campos como usuario, barbero, servicio, fecha, etc.
create_galeria_table.js: Crea la tabla de galería para almacenar imágenes asociadas a barberos.
add_nombre_invitado.js: Añade el campo nombre_invitado a la tabla de citas para soportar la funcionalidad de invitados.
seeds/: Datos iniciales para poblar la base de datos con información de prueba.

01_barberos_seed.js: Crea barberos iniciales con nombres y especialidades.
01_servicios_seed.js: Crea servicios iniciales con nombres, precios y duraciones.
01_usuarios_seed.js: Crea usuarios administradores y empleados iniciales.
01_galeria_seed.js: Crea entradas iniciales en la galería con imágenes de ejemplo.
02_usuarios_seed.js: Crea usuarios clientes de prueba.
02_citas_seed.js: Crea citas de ejemplo para demostrar la funcionalidad.
dist/: Código compilado de TypeScript a JavaScript, listo para producción. Contiene la misma estructura de directorios que src/ pero con archivos .js en lugar de .ts.

5.3. Frontend

El frontend está implementado como una aplicación de página única (SPA) usando Vue.js 3, Vite, Vuetify para la interfaz de usuario y Vue Router para la navegación.

5.3.1. Archivos Raíz del Frontend

index.html: Punto de entrada HTML de la aplicación que carga el script principal y define el contenedor raíz donde se montará la aplicación Vue.
vite.config.js: Configuración de Vite para el desarrollo y construcción de la aplicación. Define plugins como Vue y Vuetify, establece alias para importaciones y configura el servidor de desarrollo.
package.json: Define las dependencias del frontend como Vue, Vuetify, Vue Router, GSAP para animaciones, Stripe.js para pagos, Vue Google Charts para visualizaciones, entre otras. También define scripts para desarrollo, construcción y vista previa.
dockerfile: Instrucciones para construir la imagen Docker del frontend, incluyendo la instalación de dependencias y la configuración del servidor de desarrollo Vite.
vercel.json: Configuración para el despliegue en Vercel, especificando rutas y comportamiento del servidor.

5.3.2. Directorios del Frontend

src/: Código fuente del frontend organizado en una estructura modular.
main.js: Punto de entrada de la aplicación Vue que crea la instancia de Vue, registra plugins como Vue Router y Vuetify, y monta la aplicación en el elemento raíz.
App.vue: Componente raíz de la aplicación que define la estructura básica con el encabezado, el contenido principal y el pie de página.
config.ts: Configuración global, como la URL de la API backend para las solicitudes HTTP.
style.css: Estilos globales de la aplicación, definiendo variables CSS para colores, fuentes y otros estilos compartidos.
components/: Componentes Vue reutilizables que forman la interfaz de usuario.
Header.vue: Barra de navegación superior que muestra el logo, menú de navegación y opciones de usuario según su rol.
Footer.vue: Pie de página con información de contacto y enlaces útiles.

Citas.vue: Componente complejo para gestionar citas, permitiendo seleccionar barbero, servicio, fecha y hora, y opcionalmente traer un invitado o pagar online.
CitasEmpleadosAdmin.vue: Vista especializada de citas para empleados y administradores, mostrando todas las citas asignadas y permitiendo marcarlas como realizadas.
MesCompleto.vue: Vista de calendario mensual que muestra todas las citas programadas, útil para administradores y empleados.
Servicios.vue: Interfaz para administrar los servicios ofrecidos, permitiendo crear, editar y eliminar servicios.
Empleados.vue: Gestión de empleados/barberos, con funcionalidades para añadir, editar y eliminar profesionales, y ver estadísticas de rendimiento.
Usuarios.vue: Administración de usuarios, permitiendo ver, crear, editar y eliminar cuentas.
Galeria.vue: Gestión de la galería de imágenes, permitiendo subir y organizar fotos de trabajos realizados.
Login.vue: Formulario de inicio de sesión con validación y opción de recordar usuario.
Register.vue: Formulario de registro para nuevos usuarios con validación de campos.
Landing.vue: Página principal atractiva que presenta la barbería y sus servicios.
Conocenos.vue: Página informativa sobre la historia y valores de la barbería.
CitaExito.vue: Página de confirmación tras reservar una cita exitosamente.
Confirm.vue: Componente para confirmar acciones importantes como eliminaciones.
404.vue: Página de error para rutas no encontradas.
plugins/: Configuración de plugins utilizados en la aplicación.
router.js: Configuración de Vue Router, definiendo las rutas de la aplicación y la protección de rutas según el rol del usuario.
vuetify.js: Configuración de Vuetify, definiendo el tema personalizado para la barbería con colores, fuentes y componentes.
assets/: Recursos estáticos como imágenes, iconos y estilos adicionales.
Logo.png: Logo de la barbería.
animations.css: Estilos para animaciones utilizadas en toda la aplicación.
utils/: Funciones de utilidad reutilizables en el frontend.
auth.js: Funciones para gestionar la autenticación, como verificar tokens, obtener roles de usuario y almacenar datos de sesión.
animations.js: Funciones para crear animaciones con GSAP.
formatters.js: Funciones para formatear fechas, horas, precios, etc.
public/: Archivos estáticos que se sirven directamente sin procesamiento, como favicon e imágenes de fondo.
dist/: Código compilado y optimizado listo para producción, generado por Vite al construir la aplicación.

5.4. Docker

docker/: Contiene archivos de configuración y variables de entorno para Docker.
.env.backend: Variables de entorno para el servicio de backend, como credenciales de base de datos, claves API, etc.

.env.frontend: Variables de entorno para el servicio de frontend, como URL de la API backend.
docker-compose.yml: Define los tres servicios principales de la aplicación:
backend: Servicio Node.js/Express que ejecuta la API REST.
frontend: Servicio Vite/Vue que sirve la interfaz de usuario.
db: Servicio PostgreSQL que almacena los datos de la aplicación.

6. Evaluación y prueba

Durante el desarrollo del proyecto, se han incorporado validaciones exhaustivas en todas las funcionalidades con el objetivo de garantizar tanto la integridad de los datos como la seguridad general de la aplicación.

6.1. Enfoque de Validación

La validación de datos se ha abordado desde tres niveles complementarios. En el frontend, se utiliza Vue.js para realizar validaciones en tiempo real que mejoran la experiencia del usuario y previenen errores comunes antes de que lleguen al servidor. En el backend, los controladores desarrollados con Node.js y Express llevan a cabo verificaciones adicionales para asegurar que los datos cumplan con las reglas de negocio establecidas. Finalmente, en la base de datos, se han definido restricciones directamente en PostgreSQL, reforzadas por validaciones adicionales mediante el uso del ORM Objection.js.

6.2. Medidas de Seguridad

Para proteger la aplicación ante posibles amenazas, se han adoptado diversas medidas de seguridad. Se previene la inyección SQL a través del uso de Knex.js y Objection.js, los cuales emplean consultas parametrizadas. La protección contra ataques XSS se garantiza escapando correctamente el contenido generado por los usuarios. La autenticación se implementa mediante tokens JWT y las contraseñas se cifran con bcrypt para asegurar su confidencialidad. El sistema de control de acceso se basa en roles, lo que permite restringir funcionalidades según el perfil del usuario. Además, se ha aplicado limitación de peticiones (rate limiting) para reducir el riesgo de ataques de fuerza bruta en los puntos de autenticación.

6.3. Validaciones Principales

En cuanto a la gestión de usuarios, durante el registro y edición se valida que el nombre y apellidos estén completos, que el correo electrónico tenga el formato correcto y sea único, que la contraseña cumpla con requisitos de complejidad y que el número de teléfono sea

válido. Para la autenticación, se verifica tanto la validez de las credenciales como los permisos asignados al usuario.

Respecto a los barberos, se validan campos como el nombre y la especialidad, además de comprobar que el usuario que realiza la operación tenga los permisos administrativos necesarios. Para los servicios ofrecidos, se comprueba que el nombre esté presente, que el precio sea un valor positivo y que la duración esté expresada como un número entero.

Las citas requieren una validación cuidadosa, asegurando que el servicio, barbero, fecha y hora seleccionados sean coherentes, que la fecha sea futura y que la hora esté disponible. En caso de incluir un invitado, también se validan sus datos correspondientes. La cancelación de citas implica una verificación adicional, ya sea para confirmar que el usuario es el propietario de la cita o que posee permisos administrativos.

En la galería, se controla que las imágenes cargadas tengan un formato permitido, no superen un tamaño máximo y no excedan un límite de cantidad definido. Finalmente, los pagos son validados mediante la pasarela Stripe, asegurando que los datos proporcionados sean válidos y que los montos coincidan con los servicios seleccionados.

Todas las validaciones están diseñadas para ofrecer retroalimentación clara y específica, indicando al usuario el problema exacto y cómo corregirlo para completar exitosamente la operación.

7. Manual de estilos

7.1. Sketches

Aunque no han sido los estilos finales, han sido una base firme para la realización de estos: https://github.com/carlos-aj/TFG/blob/main/mockupTFGDesktop.pdf

7.2. Criterios de accesibilidad

La aplicación ha sido diseñada considerando criterios de accesibilidad con el objetivo de garantizar que todas las personas, independientemente de sus capacidades, puedan interactuar con el sistema de forma efectiva y sin barreras.

A nivel estructural, se ha utilizado una base semántica sólida mediante el uso de etiquetas HTML5 como <header>, <nav>, <main>, <footer>, <section> y <article>. Esta estructura permite a los lectores de pantalla interpretar correctamente la disposición del contenido, mientras que la jerarquía de encabezados, desde <h1> hasta <h6>, organiza la información de manera lógica y facilita la navegación asistida.

En cuanto al contenido visual, todas las imágenes cuentan con textos alternativos que describen su contenido o función utilizando el atributo alt, lo que mejora la accesibilidad para usuarios que dependen de tecnologías lectoras. Se ha cuidado especialmente el contraste entre texto y fondo, manteniendo una proporción mínima de 4.5:1 para asegurar una adecuada legibilidad, incluso en usuarios con visión reducida. Los iconos están acompañados de texto explicativo para evitar depender exclusivamente de representaciones visuales, y el diseño es completamente adaptable a distintos tamaños de pantalla, permitiendo el uso de zoom sin pérdida de funcionalidad.

La interacción con la interfaz se ha optimizado para usuarios que navegan exclusivamente con el teclado. Se ha implementado una navegación completa mediante teclas, con un orden de tabulación coherente que respeta el flujo natural de la página, y con indicadores de foco claramente visibles. Asimismo, se han incluido atajos de teclado para acciones frecuentes, los cuales están documentados dentro de la interfaz. Todos los componentes interactivos basados en Vuetify cumplen con las pautas ARIA, lo que garantiza su accesibilidad técnica.

En los formularios, cada campo está correctamente etiquetado mediante el atributo for en las etiquetas <label>, lo que permite una clara asociación entre los elementos. Se ofrecen instrucciones explícitas sobre el formato esperado en los distintos campos, y los mensajes de error son descriptivos, facilitando al usuario la identificación y corrección de problemas. Además, los campos relacionados se agrupan lógicamente utilizando elementos como <fieldset> y <legend>, lo que mejora la comprensión general del formulario.

La compatibilidad con tecnologías asistivas se ha abordado mediante la incorporación de atributos ARIA como role, aria-label y aria-describedby, que optimizan la interpretación por parte de lectores de pantalla. Se han realizado pruebas específicas utilizando herramientas como NVDA y VoiceOver, verificando la experiencia de usuario en entornos reales. Asimismo, la interfaz es compatible con funciones de ampliación, modos de alto contraste y otras utilidades de accesibilidad integradas en sistemas operativos y navegadores.

Adicionalmente, se ha procurado utilizar un lenguaje claro y directo en todos los elementos de la interfaz, evitando tecnicismos innecesarios que puedan dificultar la comprensión. Los usuarios disponen del tiempo suficiente para leer, entender y completar las tareas sin presiones por limitaciones temporales estrictas. Las funcionalidades principales están disponibles a través de diferentes mecanismos de acceso, como menús, botones y enlaces, lo que otorga mayor flexibilidad en la navegación.

7.3. Criterios de usabilidad

La aplicación ha sido concebida con un fuerte enfoque en la usabilidad y en la experiencia de usuario. Desde su diseño inicial, se han aplicado principios de diseño centrado en el usuario para asegurar que tanto clientes como empleados y administradores puedan interactuar con el sistema de manera eficiente, intuitiva y agradable.

La estructura general de la interfaz presenta una jerarquía clara de contenidos, con una organización lógica que permite al usuario localizar la información que necesita de forma rápida. La navegación se mantiene consistente en todas las secciones de la aplicación, gracias a un menú principal que adapta sus opciones según el rol del usuario. Para facilitar la orientación dentro del sistema, se han incorporado migas de pan que indican la ubicación actual del usuario. Las funcionalidades relacionadas se agrupan de manera coherente, lo que contribuye a una navegación más intuitiva.

El diseño es completamente adaptable a distintos dispositivos y resoluciones de pantalla. Ya sea en móviles, tabletas u ordenadores, los elementos de la interfaz se reorganizan de forma fluida para aprovechar el espacio disponible sin comprometer la funcionalidad. Las imágenes y gráficos se escalan proporcionalmente, y en dispositivos móviles, el menú se transforma en un menú hamburguesa que optimiza el uso del espacio.

Cada componente de la interfaz ha sido cuidadosamente diseñado para favorecer la claridad y la acción. Los botones están claramente destacados mediante colores contrastantes y etiquetas descriptivas que indican su función, como "Reservar Cita", "Confirmar" o "Cancelar". Los iconos, elegidos por su alto grado de reconocimiento, se presentan junto a textos explicativos para reforzar su significado. Los formularios han sido simplificados mediante la agrupación lógica de campos y la implementación de validación en tiempo real. Además, se proporciona retroalimentación visual inmediata tras cada acción del usuario, indicando el éxito, error o estado del proceso.

La consistencia visual de la aplicación se ha logrado mediante el uso de una paleta de colores coherente, donde el amarillo (#F5E009) actúa como color de acento sobre fondos predominantemente oscuros (#2B2B2B). La tipografía seleccionada, DM Serif, asegura una lectura cómoda en todos los tamaños de pantalla. La iconografía utilizada sigue los lineamientos de Material Design Icons, lo que refuerza la uniformidad visual. También se ha prestado especial atención al espaciado y la alineación, lo que contribuye a una interfaz limpia y bien estructurada.

Las interacciones del usuario han sido optimizadas con elementos como calendarios intuitivos para seleccionar fechas en la reserva de citas, autocompletado y sugerencias en campos de entrada para reducir errores, y confirmaciones previas en acciones que pueden tener consecuencias irreversibles, como la cancelación de citas o la eliminación de datos. Además, se han integrado transiciones y animaciones sutiles que mejoran la percepción de respuesta de la aplicación sin distraer al usuario.

La claridad en la presentación de la información también ha sido una prioridad. Se ofrecen mensajes de estado comprensibles tras cada acción, como la confirmación de una reserva o un error en el pago. En procesos más complejos, como la reserva de citas, se proporcionan instrucciones breves y precisas para guiar al usuario paso a paso. La disponibilidad de barberos y horarios se muestra de forma visualmente clara mediante códigos de colores, y la

información clave se presenta de manera ordenada en tablas y listas, facilitando así la comparación y selección de opciones.

7.4. Tipografía

La aplicación emplea un sistema tipográfico cuidadosamente seleccionado que busca mantener la coherencia visual, asegurar una excelente legibilidad y reforzar la identidad estética de la marca. La elección de la tipografía no responde únicamente a criterios funcionales, sino también al deseo de transmitir un estilo acorde con el carácter tradicional y refinado del negocio de la barbería.

Como fuente principal se ha optado por DM Serif, utilizada de manera exclusiva en toda la interfaz. Esta tipografía, de estilo serif, combina elegancia clásica con un diseño contemporáneo, evocando la tradición artesanal del oficio sin perder modernidad. Sus formas estilizadas permiten una lectura fluida tanto en pantallas grandes como en dispositivos móviles, y sus rasgos distintivos aportan una personalidad visual reconocible y con carácter. Gracias a su versatilidad, la fuente funciona con eficacia tanto en títulos como en cuerpos de texto, manteniendo armonía en todos los niveles jerárquicos de contenido.

La implementación de DM Serif se realiza mediante su carga directa desde Google Fonts, lo que garantiza compatibilidad y rendimiento en múltiples navegadores. En el sistema de estilos, la fuente se define como una variable CSS global, lo que permite aplicar de manera consistente la tipografía en todos los elementos textuales de la aplicación, asegurando así una experiencia visual unificada.

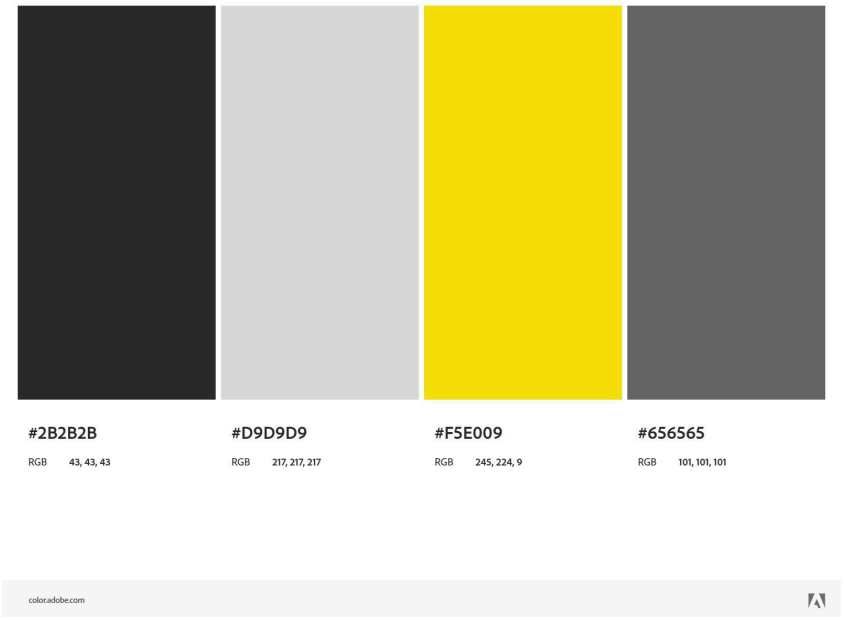
Para crear jerarquías visuales claras dentro de la interfaz, se han empleado distintas variantes de la fuente. La versión regular se utiliza en la mayoría del contenido, ofreciendo una apariencia equilibrada y profesional. La variante italic se reserva para títulos o elementos que requieren un énfasis especial, generando un contraste sutil y elegante. Por su parte, la versión display se aplica a títulos principales y componentes destacados, aprovechando al máximo la expresividad visual de la tipografía.

En cuanto a los tamaños y pesos tipográficos, se ha definido una escala coherente y adaptativa. Los títulos principales varían entre 24 y 32 píxeles en pantallas de escritorio, y entre 20 y 24 píxeles en dispositivos móviles. Los subtítulos oscilan entre 18 y 22 píxeles en escritorio, y de 16 a 18 píxeles en móvil. Para el texto de navegación, se utiliza un tamaño de 22 píxeles en escritorio y 18 en móvil. El cuerpo principal de los textos mantiene una medida estándar de 16 píxeles en escritorio y entre 14 y 16 píxeles en móvil, mientras que los textos secundarios se presentan con 14 píxeles en pantallas grandes y entre 12 y 14 píxeles en móviles.

En conjunto, este sistema tipográfico no solo facilita la lectura y la navegación, sino que además contribuye activamente a consolidar la identidad visual de la aplicación. A través de

su estética cuidada y su implementación precisa, la tipografía transmite profesionalidad, confianza y un carácter distintivo que refleja la esencia del negocio.

7.5. Mapa de colores



7.6. Dispositivos/vistas

La aplicación ha sido desarrollada bajo principios de diseño responsive, con el objetivo de adaptarse de manera fluida a una amplia variedad de dispositivos y tamaños de pantalla. Esto garantiza que todos los usuarios, independientemente del equipo que utilicen, puedan disfrutar de una experiencia óptima, tanto en términos de funcionalidad como de presentación visual.

En cuanto a la compatibilidad, la aplicación mantiene un rendimiento estable y una apariencia coherente en dispositivos de escritorio y portátiles con resoluciones iguales o superiores a 1200 píxeles, así como en tablets y navegadores redimensionados entre 768 y 1199 píxeles. También ofrece soporte completo para smartphones y dispositivos móviles dentro del rango de 400 a 767 píxeles de ancho. Para preservar la experiencia de usuario, se han definido dos límites clave: un ancho mínimo estético de 400 píxeles, por debajo del cual la interfaz pierde cohesión visual, y un ancho mínimo de usabilidad de 360 píxeles, que representa el umbral funcional inferior para operar la aplicación sin comprometer su uso.

La implementación técnica del diseño adaptable se ha llevado a cabo principalmente mediante el uso de media queries en los archivos CSS, los cuales ajustan dinámicamente los estilos según el ancho disponible en pantalla. Asimismo, se ha aprovechado el sistema de rejilla que ofrece Vuetify, utilizando componentes como v-row y v-col para reorganizar

automáticamente los elementos según el espacio disponible, lo que permite mantener un diseño limpio y funcional sin importar el dispositivo.

Para optimizar aún más la experiencia, se han realizado adaptaciones específicas que incluyen un menú de navegación que varía según el contexto: mientras que en pantallas grandes se muestra en su versión completa, en dispositivos móviles se transforma en un menú hamburguesa con acceso lateral. Los tamaños de fuente, botones e imágenes se ajustan proporcionalmente en función del tamaño de pantalla, al igual que los márgenes y espacios, que han sido cuidadosamente calibrados para cada resolución. El contenido que en escritorio se organiza en múltiples columnas, se apila verticalmente en móviles para facilitar la lectura y la interacción. Las tablas, por su parte, se transforman en vistas de tarjetas en pantallas reducidas, conservando toda la información sin comprometer la legibilidad.

A nivel de comportamiento dinámico, también se han incorporado ajustes mediante JavaScript para adaptar la interfaz en tiempo real. Entre estos se encuentran la modificación automática del encabezado en función del tamaño de pantalla, así como la adaptación del comportamiento de scroll y las animaciones para garantizar una navegación fluida en todos los dispositivos.

La aplicación ha sido sometida a pruebas de compatibilidad en múltiples entornos, incluyendo los navegadores Chrome, Firefox, Safari y Edge, y en los sistemas operativos Windows, macOS, iOS y Android. Asimismo, se ha verificado su funcionamiento en una amplia gama de dispositivos, desde ordenadores de escritorio y portátiles hasta tablets y smartphones de distintos tamaños.

8. Software utilizado

El desarrollo de la aplicación ha implicado la utilización de diversas herramientas y plataformas que han contribuido a mantener un flujo de trabajo eficiente y a lograr un producto final robusto, escalable y de alta calidad.

Para el entorno de desarrollo se empleó Visual Studio Code, un editor de código fuente desarrollado por Microsoft que funcionó como la herramienta principal para escribir, depurar y mantener el código. Esta elección se justificó por su excelente integración con tecnologías como Vue.js, TypeScript y Node.js, así como por la disponibilidad de extensiones para ESLint y Prettier, que ayudaron a mantener la calidad del código. Además, su terminal integrada facilitó la ejecución de comandos npm y pnpm, y su soporte nativo para Git permitió un control de versiones fluido directamente desde el editor.

El uso de Docker y Docker Compose resultó clave para garantizar entornos de desarrollo consistentes y fácilmente replicables. Se contenedorizó cada servicio por separado (frontend, backend y base de datos), lo que permitió aislar los procesos y evitar conflictos entre

dependencias. La configuración se gestionó a través del archivo docker-compose.yml, que facilitó la orquestación de los servicios, y se utilizaron volúmenes para asegurar la persistencia de datos y permitir el desarrollo en tiempo real sin pérdida de información.

En cuanto a la base de datos, se utilizó PostgreSQL, un sistema de gestión relacional robusto y altamente compatible con aplicaciones modernas. La versión 15 fue desplegada en un contenedor Docker, lo que garantizó su integración con el resto del sistema. Se aprovecharon características avanzadas como los tipos de datos en arrays, útiles para funcionalidades como la galería de imágenes, y se implementaron claves foráneas para mantener la integridad referencial.

Para el backend se eligió Node.js, en su versión 16 o superior, como entorno de ejecución de JavaScript en el servidor. Esta elección permitió aprovechar las últimas características del estándar ECMAScript, así como una gestión eficaz de dependencias a través de npm. El framework Express y el ORM Objection.js se emplearon para facilitar la estructuración del código del servidor y su interacción con la base de datos.

En el desarrollo del frontend, se utilizó Vite como herramienta de construcción moderna. Esta tecnología ofreció un servidor de desarrollo con recarga en caliente (Hot Module Replacement), optimización automática para producción y compatibilidad con plugins esenciales como Vue y Vuetify, que fueron la base visual e interactiva de la interfaz.

Durante el desarrollo y las pruebas, se utilizó Google Chrome como navegador principal, complementado por pruebas de compatibilidad cruzada en Firefox, Safari y Microsoft Edge. Esto permitió verificar el correcto funcionamiento de la aplicación en los principales entornos del mercado.

Para el control de versiones se adoptó Git, que permitió gestionar de forma eficiente las ramas del proyecto, realizar seguimiento detallado de los cambios y facilitar la colaboración entre desarrolladores. El código fuente fue alojado en GitHub, donde además se gestionaron los problemas (issues), las solicitudes de cambios (pull requests), y la documentación técnica mediante archivos README y wiki internos.

La integración de pagos fue posible gracias a Stripe, una plataforma de procesamiento de pagos en línea. La aplicación se conectó a su API para manejar transacciones con tarjeta de forma segura. Durante el desarrollo, se utilizó el modo Sandbox para realizar pruebas sin afectar fondos reales, y se implementaron webhooks para gestionar los eventos de pago en tiempo real.

Para el despliegue, se utilizaron diferentes servicios según el componente. El frontend fue desplegado en Vercel, lo que facilitó la integración continua con GitHub, la generación automática de versiones de vista previa y la entrega rápida del contenido estático gracias a su red de distribución de contenido global (CDN). El backend fue alojado en Render, una plataforma que ofreció escalado automático y soporte nativo para aplicaciones Node.js.

Finalmente, la base de datos PostgreSQL fue alojada en Railway, una solución en la nube que proporcionó una gestión simplificada, copias de seguridad automáticas y herramientas de monitorización del rendimiento.

9. Mejoras posibles y aportaciones

9.1. Funcionalidad

A partir de las limitaciones actuales del sistema, se han identificado varias áreas clave en las que se pueden implementar mejoras funcionales para futuras versiones de la aplicación. Una de las primeras es la inclusión de un sistema completo de recuperación de contraseñas mediante correo electrónico. Esta funcionalidad permitiría generar tokens de un solo uso con validez temporal, habilitar una página segura para restablecer la contraseña, y enviar notificaciones automáticas de seguridad al realizar cambios en las credenciales de acceso.

Otro aspecto relevante es la posibilidad de habilitar una gestión de citas más avanzada por parte del personal. Esto incluiría una interfaz administrativa específica para que empleados y administradores puedan crear citas en nombre de los clientes, registrar qué empleado ha creado cada una para fines de auditoría, y añadir notas especiales como observaciones en reservas telefónicas. Además, se contempla permitir la creación de cuentas provisionales para clientes que aún no se hayan registrado.

En cuanto a la personalización del horario, se propone ofrecer mayor flexibilidad en la configuración, permitiendo definir horarios personalizados para cada barbero, registrar días libres y vacaciones individuales, e integrar un calendario de festivos actualizado automáticamente según la localidad. También se podría aplicar un sistema de bloqueo inteligente de franjas horarias, considerando la duración de los servicios.

La integración con sistemas externos es otra línea prioritaria de mejora. Se plantea permitir la sincronización bidireccional con Google Calendar, Apple Calendar y Outlook, además de ofrecer la exportación de citas en formatos estándar como iCal y CSV. La documentación de una API pública y la implementación de webhooks permitirían una conectividad más fluida con otros sistemas de gestión o herramientas externas.

En el ámbito de la búsqueda y la organización de la información, se proyecta un buscador unificado dentro del panel administrativo que facilite encontrar barberos, servicios, usuarios y citas. Esta funcionalidad se complementaría con filtros avanzados por fecha, estado, método de pago y profesional asignado, así como la posibilidad de ordenar los resultados por popularidad, valoraciones o disponibilidad. La incorporación de un sistema de etiquetas también permitiría categorizar y filtrar servicios con mayor precisión.

La gestión avanzada de disponibilidad incluye propuestas como la implementación de listas de espera para horarios saturados, notificaciones automáticas cuando se liberen franjas solicitadas y ajustes dinámicos del tiempo entre citas para permitir la preparación del espacio

o del barbero. También se estudia un sistema de sugerencia inteligente que proponga horarios alternativos cuando no haya disponibilidad inmediata.

9.2. Código

A nivel técnico, el código base puede beneficiarse de varias mejoras estructurales. Se contempla la refactorización del backend aplicando el patrón Repository, lo cual permitiría separar la lógica de acceso a datos de los controladores y mejorar la mantenibilidad. Asimismo, la implementación de pruebas unitarias y de integración incrementaría la cobertura y la fiabilidad del sistema.

En el frontend, se recomienda una migración progresiva hacia TypeScript para mejorar la detección temprana de errores y aprovechar un autocompletado más eficaz durante el desarrollo. También se sugiere la optimización de las consultas a la base de datos, mediante la revisión de índices y la eliminación de consultas tipo N+1, con el fin de mejorar el rendimiento general.

Otro avance técnico sería la implementación de un sistema de caché para reducir la carga del servidor y acelerar los tiempos de respuesta. Finalmente, la documentación de la API utilizando estándares como Swagger o OpenAPI facilitaría enormemente su adopción por terceros y mejoraría la interoperabilidad del sistema.

9.3. Información

Desde el punto de vista informativo, se plantea la creación de un sistema de estadísticas y analíticas avanzadas que proporcione a los administradores un panel de control con métricas relevantes sobre el rendimiento del negocio y las tendencias de uso. Además, se podrían generar informes personalizables por períodos, servicios o barberos, lo cual aportaría una visión más detallada de la actividad.

El sistema de notificaciones también podría ser ampliado para incluir más opciones de configuración y nuevos canales como SMS o notificaciones push. Paralelamente, se propone enriquecer la documentación de usuario mediante tutoriales en video y guías paso a paso que faciliten la adopción de la aplicación.

Con vistas a fortalecer la relación con los clientes, se podrían introducir un programa de fidelización basado en puntos, descuentos y promociones especiales, así como un sistema de encuestas de satisfacción que recoja automáticamente el feedback tras cada cita.

Estas propuestas de mejora abordan de manera directa las limitaciones actuales del sistema, ampliando sus funcionalidades y optimizando la experiencia de todos los usuarios involucrados. A medida que se integren, no solo aumentarán la flexibilidad y escalabilidad

del servicio, sino que también reforzarán el valor percibido tanto por clientes como por el equipo de trabajo.

10. Bibliografía y Referencias

Documentación Oficial de Tecnologías

Frontend:

- Vue.js. (2023). Vue.js - The Progressive JavaScript Framework. https://vuejs.org/guide/introduction.html
- Vite. (2023). Vite - Next Generation Frontend Tooling. https://vitejs.dev/guide/
- Vuetify. (2023). Vuetify - Material Design Framework. https://vuetifyjs.com/en/introduction/why-vuetify/
- Vue Router. (2023). Vue Router - The official router for Vue.js. https://router.vuejs.org/guide/
- GreenSock. (2023). GSAP (GreenSock Animation Platform). https://greensock.com/docs/
- Stripe. (2023). Stripe.js Reference. https://stripe.com/docs/js
- Vue Google Charts. (2023). Vue Google Charts Documentation. https://www.npmjs.com/package/vue-google-charts

Backend:

- Node.js. (2023). Node.js Documentation. https://nodejs.org/en/docs/
- TypeScript. (2023). TypeScript Documentation. https://www.typescriptlang.org/docs/
- Express. (2023). Express - Node.js web application framework. https://expressjs.com/en/guide/routing.html
- PostgreSQL. (2023). PostgreSQL Documentation. https://www.postgresql.org/docs/15/index.html
- Knex.js. (2023). Knex.js - A SQL Query Builder for JavaScript. https://knexjs.org/guide/
- Objection.js. (2023). Objection.js - SQL-friendly ORM for Node.js. https://vincit.github.io/objection.js/guide/
- JSON Web Tokens. (2023). JWT.io - JSON Web Tokens Introduction. https://jwt.io/introduction/
- Bcrypt. (2023). Bcrypt npm package. https://www.npmjs.com/package/bcrypt
- Multer. (2023). Multer - Node.js middleware for handling multipart/form-data. https://github.com/expressjs/multer
- Nodemailer. (2023). Nodemailer - Send emails from Node.js. https://nodemailer.com/about/
- Stripe API. (2023). Stripe API Reference. https://stripe.com/docs/api

Infraestructura:

- Docker. (2023). Docker Documentation. https://docs.docker.com/
- Docker Compose. (2023). Docker Compose Documentation. https://docs.docker.com/compose/
- PNPM. (2023). PNPM Documentation. https://pnpm.io/motivation
- NPM. (2023). npm Documentation. https://docs.npmjs.com/
- Vercel. (2023). Vercel Documentation. https://vercel.com/docs
- Render. (2023). Render Documentation. https://render.com/docs
- Railway. (2023). Railway Documentation. https://docs.railway.app/

Recursos adicionales:

Herramientas de Diseño y Planificación

- Miro. (2023). Miro - The Visual Collaboration Platform. https://miro.com/ Utilizado para la creación de diagramas de flujo, casos de uso y estructura de la base de datos.
- Adobe Color. (2023). Adobe Color - Color wheel, a color palette generator. https://color.adobe.com/ Empleado para la selección y armonización de la paleta de colores de la aplicación.
- ChatGPT. (2023). OpenAI ChatGPT. https://chat.openai.com/ Utilizado como asistente para la generación de algunas imágenes y contenido textual.

Recursos Visuales y Contenido

- Barbería local. (2025). https://www.instagram.com/olimpo_hairstyle/ Fuente de imágenes reales para la galería de la aplicación, utilizadas con permiso.