PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SERVICIO DE BOOKING PARA SALON DE UÑAS

Proyecto de titulación previo a la obtención del título de: Tecnología Superior En Desarrollo De Software

Autor: Isabela Mei Pesantez Choi

Tutor: Mgtr. Carlos Miguel Cárdenas Riofrio

Quito, Ecuador

2026

Tabla de contenidos

Lista de tablas

Lista de figuras

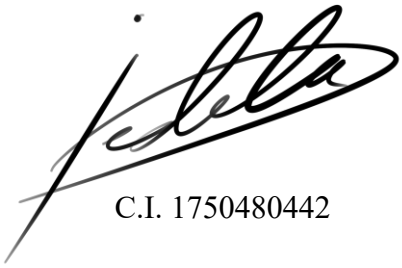
DECLARACIÓN y AUTORIZACIÓN

Yo, **ISABELA MEI PESANTEZ CHOI** con C.I. 1750480442 autor(a) del trabajo de titulación intitulado: **"SERVICIO DE BOOKING PARA SALON DE UÑAS"**, previa a la obtención del título de **Tecnología Superior En Desarrollo De Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 5 de febrero del 2026



C.I. 1750480442

Introducción

El avance de las tecnologías de la información ha transformado de manera significativa la forma en que los negocios gestionan sus procesos internos y se relacionan con sus clientes. En este contexto, los sistemas de información se han consolidado como herramientas fundamentales para optimizar recursos, mejorar la eficiencia operativa y elevar la calidad del servicio ofrecido. Entre estas soluciones, los servicios de reserva en línea han adquirido una amplia adopción en distintos sectores, especialmente en aquellos orientados a la atención directa al cliente.

En el sector de la belleza y el cuidado personal, particularmente en los salones de uñas, la gestión de citas se realiza con frecuencia de manera manual. Esta práctica genera diversos inconvenientes, tales como la duplicación de reservas, desorganización de horarios, pérdida de información y dificultades en el control de los servicios prestados. Estas problemáticas impactan de forma negativa tanto en la experiencia del cliente como en la productividad y organización del negocio.

El presente proyecto de titulación tiene como objetivo el desarrollo de un servicio de booking para un salón de uñas, orientado a automatizar el proceso de reservas y optimizar la administración de citas y servicios. La solución propuesta permite a los administradores gestionar la información de manera centralizada, segura y eficiente, mientras que los clientes disponen de una plataforma intuitiva que facilita la reserva de servicios de forma rápida y accesible. El salón al que le proveeré el servicio actualmente no maneja sus servicios de booking con ninguna tecnología, la idea del proyecto es optimizar el manejo de su local y facilitar el trabajo de las recepcionistas, evitando confusiones, y brindando una mejor experiencia para el cliente.

Para el desarrollo del sistema se aplicaron los conocimientos adquiridos durante la formación académica, abarcando las etapas de análisis de requerimientos, diseño, construcción de la aplicación y ejecución de pruebas. La solución tecnológica se basa en una arquitectura moderna, integrando herramientas y tecnologías actuales que garantizan su funcionalidad, escalabilidad y seguridad.

Este documento describe de manera detallada el proceso de desarrollo del sistema que se ha creado, organizado en capítulos que abordan el levantamiento de requisitos y el diseño de la solución, su implementación y las pruebas realizadas, finalizando con los resultados obtenidos, conclusiones y recomendaciones para futuros trabajos.

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema

1.1 Introducción al capítulo

En este capítulo se expone el proceso de identificación, análisis y definición de los requisitos del sistema desarrollado para el servicio de reservas de un salón de uñas, así como las decisiones tomadas en el diseño general de la solución tecnológica. El levantamiento de requisitos permitió comprender las necesidades reales del negocio y de los usuarios finales, sirviendo como base para la estructuración del sistema.

Asimismo, se describe el enfoque de diseño aplicado, considerando aspectos funcionales, técnicos y de usabilidad, con el objetivo de garantizar un sistema eficiente, seguro y alineado con los procesos operativos del salón de uñas.

1.2 Descripción del problema

En el sector de la belleza y el cuidado personal, muchos salones de uñas continúan gestionando sus citas de manera manual, utilizando medios informales como llamadas telefónicas, mensajes por aplicaciones de mensajería o redes sociales. Este tipo de gestión presenta múltiples inconvenientes, entre los que destacan la duplicación de reservas, errores en la asignación de horarios, falta de control sobre la disponibilidad y pérdida de información relevante de los clientes, y sobre todo queda la puerta abierta al error humano, provocando confusión en citas, desorden y esperas largas con clientes, y problemas de comunicación.

Estas deficiencias afectan negativamente tanto a los administradores del negocio como a los clientes, generando desorganización interna, disminución de la productividad y una experiencia de usuario poco satisfactoria. Frente a esta problemática, se evidencia la necesidad de implementar un sistema digital que permita automatizar y centralizar el proceso de reservas, mejorando la eficiencia operativa y la calidad del servicio ofrecido.

1.3 Objetivo del sistema

1.3.1 Objetivo general

Desarrollar un sistema de booking para un salón de uñas que permita automatizar la gestión de citas y servicios para las empleadas del local, mejorando la organización administrativa y la experiencia del cliente.

1.3.2 Objetivos específicos

- Facilitar la reserva de citas a través de una plataforma digital.
- Centralizar la información de servicios, clientes y horarios.
- Reducir errores en la asignación de citas.
- Mejorar la atención al cliente mediante un sistema organizado e intuitivo.
- Optimizar los procesos administrativos del salón.
- Permitir a los clientes visualizar la disponibilidad de citas.

1.4 Alcance del sistema

El sistema permitirá a los administradores gestionar servicios y citas, así como controlar los horarios disponibles. Los clientes podrán visualizar el calendario de disponibilidad que es el corazón del proyecto, se puede visualizar cuáles son las empleadas que están disponibles, que servicios ofrecen y qué días trabajan, sin revelar información sensible como números telefónicos o correos electrónicos. Se permitirá también que las recepcionistas actuales no pierdan su trabajo, simplemente que, dada a la gran afluencia de clientes en el salón, su trabajo sea más simple y un proceso más mecánico que no deje espacio a errores humanos que causen confusión. No se contempla en esta versión la gestión de pagos en línea ni la integración con sistemas contables externos.

1.5 Levantamiento de requisitos

El levantamiento de requisitos se realizó considerando el funcionamiento habitual de un salón de uñas y las necesidades de los usuarios involucrados que en este caso serían las dueñas y recepcionistas del negocio. A partir de un análisis y reuniones con la dueña principal del negocio, se definieron los requisitos funcionales y no funcionales del sistema.

1.5.1 Requisitos funcionales

Tabla 1 – Requisitos funcionales

Código	Descripción
RF01	El sistema debe permitir el inicio de sesión de usuarios administradores mediante credenciales seguras.
RF02	El sistema debe permitir la recuperación de contraseña a través de correo electrónico.
RF03	El sistema debe mostrar el listado de servicios disponibles, incluyendo nombre, descripción, precio y duración.
RF04	El sistema debe permitir al administrador registrar nuevos servicios.
RF05	El sistema debe permitir al administrador modificar la información de los servicios existentes.
RF06	El sistema debe permitir al administrador eliminar servicios.
RF07	El sistema debe permitir a los clientes reservar citas seleccionando servicio, fecha y hora.
RF08	El sistema debe mostrar un calendario con las citas registradas.
RF09	El sistema debe permitir la modificación o cancelación de citas existentes.

Código	Descripción
RF10	El sistema debe almacenar la información de las citas de forma centralizada y segura.

1.5.2 Requisitos no funcionales

Tabla 2 – Requisitos no funcionales

Código	Descripción
RNF01	El sistema debe contar con una interfaz intuitiva y fácil de usar.
RNF02	El sistema debe garantizar tiempos de respuesta adecuados ante las solicitudes del usuario.
RNF03	El sistema debe proteger la información almacenada mediante mecanismos de seguridad.
RNF04	El sistema debe ser accesible desde dispositivos de escritorio y móviles.
RNF05	El sistema debe ser escalable para permitir futuras mejoras o ampliaciones.
RNF06	El sistema debe garantizar la disponibilidad del servicio durante el horario de atención del salón.

1.6 Identificación de actores del sistema

Durante el análisis del sistema se identificaron los siguientes actores principales:

- **Administrador:** encargado de gestionar los servicios, citas, horarios de las empleadas y acceso al sistema.
- **Cliente:** usuario que consulta los servicios disponibles y disponibilidad de las citas.

- **Staff:** registro y control total sobre las citas, recibe llamadas y usa el Sistema para generar citas de forma eficaz.
- **Sistema:** plataforma tecnológica que procesa, almacena y administra la información.

1.7 Diseño del sistema

Con base en los requisitos funcionales y no funcionales definidos, se diseñó un sistema bajo una arquitectura cliente-servidor, orientada a la separación de responsabilidades y a la escalabilidad de la solución. Esta arquitectura permite desacoplar la capa de presentación de la lógica de negocio, facilitando el mantenimiento, la evolución del sistema y una mejor experiencia para el usuario final.

El frontend es responsable de la interacción directa con los usuarios, proporcionando interfaces intuitivas para el acceso al sistema, la visualización de los servicios disponibles y la gestión de citas. Esta capa se enfoca en la usabilidad y accesibilidad, garantizando una navegación clara y eficiente tanto para clientes como para administradores.

El backend se encarga de la gestión de la lógica del negocio, el procesamiento de las solicitudes, el almacenamiento de la información y la aplicación de mecanismos de seguridad. A través de esta capa se controlan los procesos relacionados con la autenticación de usuarios, la administración de servicios y la gestión de reservas, asegurando la integridad y confidencialidad de los datos

Asimismo, se definieron los principales flujos del sistema, tales como el proceso de autenticación, la administración de servicios y el flujo de reservas, los cuales sirvieron como base para la fase de construcción e implementación de la solución tecnológica.

1.8 Conclusión del capítulo

En este capítulo se analizó la problemática asociada a la gestión manual de citas en los salones de uñas y se realizó el levantamiento de los requisitos necesarios para el

desarrollo del sistema de booking propuesto. De igual manera, se establecieron los lineamientos del diseño del sistema, definiendo una arquitectura clara y estructurada que servirá como base para la implementación de la solución tecnológica, la cual se desarrolla y detalla en el siguiente capítulo.

Capítulo II

Construcción del Sistema

En este capítulo se describe de manera detallada el proceso de construcción e implementación del sistema de booking desarrollado para un salón de uñas. Se explican las decisiones técnicas adoptadas, la estructura general del sistema, las tecnologías utilizadas y la forma en que se integraron los distintos componentes para conformar una solución funcional, segura y principalmente, escalable.

La construcción del sistema se realizó a partir de los requisitos funcionales y no funcionales definidos en el capítulo anterior, siguiendo una arquitectura moderna orientada a la separación de responsabilidades entre las capas de presentación, lógica de negocio y persistencia de datos. Este enfoque permitió desarrollar una plataforma robusta que optimiza la gestión de citas y reduce significativamente los errores asociados a procesos manuales.

2.1 Estructura general del desarrollo

El sistema fue desarrollado bajo el modelo de arquitectura cliente-servidor, en el cual las responsabilidades se distribuyen entre el cliente, encargado de la interacción con el usuario, y el servidor, responsable del procesamiento de datos, la lógica de negocio y el almacenamiento de la información.

En este enfoque, el cliente corresponde a una aplicación web desarrollada con React, que permite a los usuarios interactuar con el sistema mediante una interfaz dinámica e intuitiva. El servidor corresponde a un backend desarrollado con Spring Boot, el cual expone servicios REST que procesan las solicitudes provenientes del frontend y gestionan la información de forma centralizada. Adicionalmente, el sistema integra servicios externos para la autenticación de usuarios mediante Firebase Authentication, lo que permite un manejo seguro de credenciales y la recuperación de contraseñas sin necesidad de implementar mecanismos propios de gestión de usuarios en el backend.

Esta arquitectura favorece la escalabilidad del sistema, facilita su mantenimiento y permite la incorporación de nuevas funcionalidades en futuras versiones.

2.2 Organización del proyecto

El sistema se organizó en tres capas principales, además de unos servicios complementarios, cada una con responsabilidades claramente definidas:

Frontend web (React)

- Interfaz gráfica para la visualización de servicios y disponibilidad de citas.
- Pantalla de inicio de sesión para administradores y personal autorizado.
- Visualización del calendario de reservas, elemento central del sistema.
- Formularios para la creación, edición y eliminación de servicios.
- Gestión de roles de usuario (administrador, staff y cliente).
- Validación de datos antes del envío al servidor.

Backend (Spring Boot)

- Exposición de una API REST para la gestión de servicios y citas.
- Implementación de la lógica de negocio del sistema.
- Control de acceso a las funcionalidades según el rol del usuario.
- Persistencia de la información en la base de datos.
- Integración con Firebase para la autenticación de usuarios.

Base de datos (PostgreSQL y pgAdmin)

- Almacenamiento estructurado de la información del sistema.
- Gestión de tablas para servicios, técnicos, citas y usuarios.
- Mantenimiento de la integridad y relaciones entre entidades.
- Uso de pgAdmin como herramienta de administración y visualización de la base de datos.
- Ejecución de consultas y verificación de datos durante el desarrollo.

Servicios externos

- Firebase Authentication para la gestión de inicio de sesión y recuperación de contraseñas.
- Base de datos relacional para el almacenamiento de servicios y citas.

- Infraestructura de despliegue para garantizar la disponibilidad del sistema.

Esta organización permitió mantener un código modular, legible y fácil de mantener, alineado con buenas prácticas de ingeniería de software.

2.3 Desarrollo del frontend

El frontend del sistema fue cimentado utilizando la biblioteca React, la cual permite construir interfaces de usuario basadas en componentes reutilizables. Esta tecnología facilita la creación de aplicaciones web dinámicas, en las que los cambios de estado se reflejan automáticamente en la interfaz sin necesidad de recargar la página.

Durante el desarrollo se priorizó la experiencia de usuario, diseñando pantallas claras y funcionales que permiten una navegación fluida entre las distintas secciones del sistema. Entre las principales funcionalidades implementadas en el frontend se encuentran:

- Inicio de sesión para usuarios administradores y staff mediante correo electrónico y contraseña.
- Visualización del listado de servicios disponibles, organizados por categorías.
- Interfaz gráfica del calendario de disponibilidad de citas.
- Formularios para la gestión de servicios y citas.
- Interfaz diferenciada según el rol del usuario, restringiendo el acceso a funcionalidades administrativas.

Se aplicaron validaciones en el cliente para asegurar que los datos ingresados por el usuario cumplan con los formatos requeridos antes de ser enviados al backend, reduciendo errores y mejorando la confiabilidad del sistema.

Figura 1

Idea inicial de interfaz de login

Figura 2

Idea inicial de creación de cita

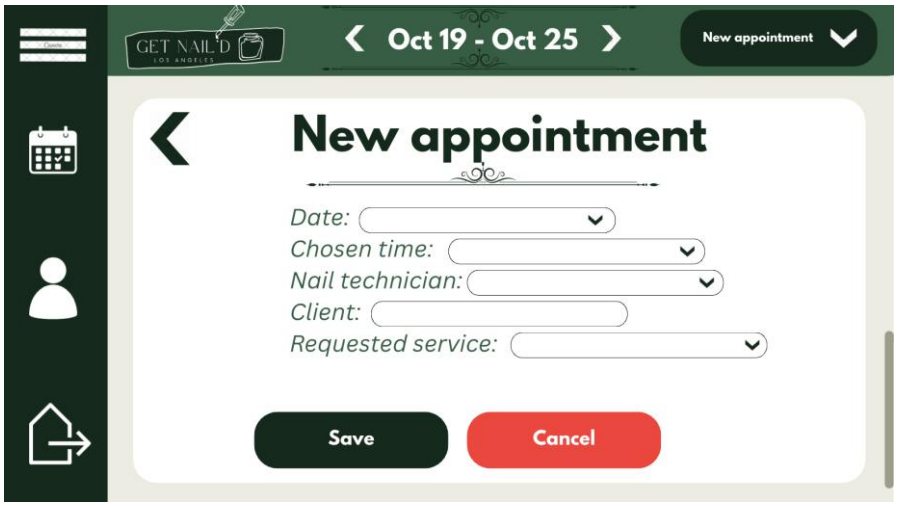
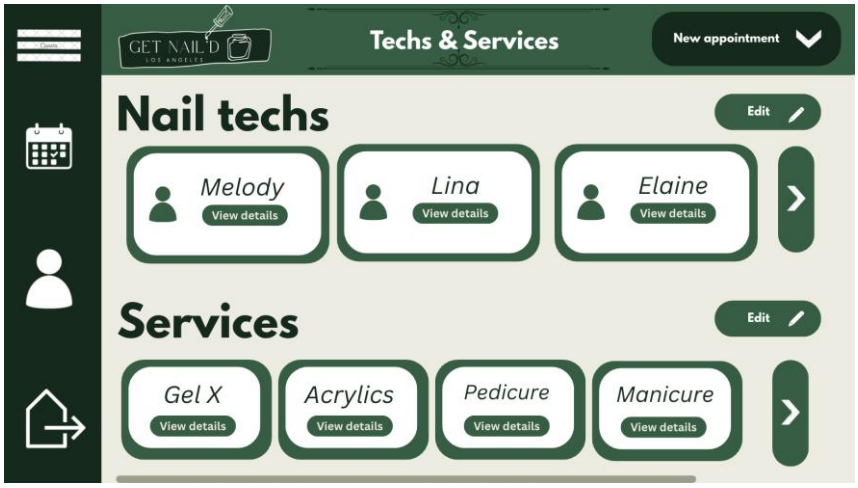


Figura 3

Idea inicial de muestra de empleadas y servicios



2.4 Desarrollo del backend

El backend fue desarrollado utilizando Spring Boot, un framework que facilita la creación de aplicaciones empresariales basadas en Java. Este framework permite estructurar el proyecto en capas, separando controladores, servicios y repositorios, lo que contribuye a un código más limpio y organizado.

El backend se encarga de procesar las solicitudes enviadas desde el frontend, aplicar las reglas de negocio y gestionar la comunicación con la base de datos. Entre las principales responsabilidades del backend se incluyen:

- Gestión de servicios ofrecidos por el salón (crear, modificar y eliminar).
- Gestión de citas y control de disponibilidad de horarios.
- Validación de datos recibidos desde el frontend.
- Persistencia de la información en la base de datos.
- Control de acceso a las funcionalidades según el rol del usuario autenticado.

Para la autenticación, el backend se integra con Firebase Authentication, este se usa únicamente para el servicio de login de los administradores, cuentas que también están integradas en el código del frontend para que solo ciertos correos puedan tener acceso a la creación de cita y edición de servicios y empleadas, no se ofrece un servicio de registro de cuenta, pero sí de recuperación de contraseña.

La comunicación entre el frontend y el backend se realiza mediante peticiones HTTP enviadas desde la aplicación React hacia los endpoints de la API REST desarrollada en Spring Boot. Estas solicitudes se implementan utilizando la librería **Axios**, específicamente con métodos como *axios.get, axios.post, axios.put* y *axios.delete.* Por ejemplo, cuando el sistema necesita obtener la lista de servicios o las citas registradas, el frontend realiza peticiones como GET */api/services* o GET */api/appointments*. De igual manera, cuando se crea o modifica una cita, se envían solicitudes POST o PUT con la información correspondiente en formato JSON. Este mecanismo permite que el frontend y el backend trabajen de forma desacoplada, comunicándose a través de la API de manera clara y estructurada, lo que facilita futuras mejoras, mantenimiento y escalabilidad del sistema.

2.5 Integración y comunicación entre capas

La comunicación entre el frontend y el backend se realiza mediante solicitudes HTTP utilizando el protocolo REST. El frontend envía peticiones al backend para consultar información, registrar servicios o gestionar citas, y el backend responde con los datos solicitados o con mensajes de confirmación según la operación realizada.

Esta comunicación se implementó utilizando librerías de consumo de APIs en el frontend, encapsulando así las solicitudes en funciones reutilizables. Gracias a este enfoque, se mantiene un código ordenado que facilita la modificación de endpoints en caso de cambios futuros.

El backend, por su parte, se encarga de procesar las solicitudes, interactuando con la base de datos y devolviendo respuestas estructuradas en formato JSON, asegurando la interoperabilidad entre las capas del sistema.

2.6 Control de versiones y pruebas iniciales

Durante el desarrollo del sistema se utilizó Git como herramienta de control de versiones. Esta herramienta permitió registrar los cambios realizados en el código,

facilitar la recuperación de versiones anteriores y mantener un historial claro del proceso de desarrollo.

Las pruebas iniciales se realizaron de manera progresiva durante la implementación de cada funcionalidad. Se verificó el correcto funcionamiento de:

- Inicio de sesión y recuperación de contraseñas.
- Gestión de servicios por parte del administrador.
- Visualización y gestión del calendario de citas.
- Control de roles y restricciones de acceso.

Estas pruebas permitieron identificar errores tempranamente y realizar ajustes antes de consolidar una versión estable del sistema, contribuyendo a una mayor calidad del producto final.

2.7 Conclusión del capítulo

En este capítulo se detalló el proceso de construcción del sistema de booking para un salón de uñas, describiendo la arquitectura adoptada, las tecnologías empleadas y la implementación de las principales funcionalidades. La correcta integración entre el frontend y el backend permitió desarrollar una solución robusta, segura y alineada con los requisitos establecidos.

El sistema construido sienta las bases para la fase de pruebas y estabilización, la cual se aborda en el siguiente capítulo, donde se evaluará el desempeño del sistema y su correcto funcionamiento en un entorno real simulando cargas reales.

Capítulo III

Pruebas y Estabilización

3.1 Enfoque general de pruebas

Con el objetivo de validar la estabilidad y el rendimiento del backend del sistema de gestión del salón de uñas, se realizaron pruebas de carga utilizando Apache JMeter como herramienta principal. Estas pruebas se centraron en simular múltiples usuarios accediendo de forma simultánea a los servicios más importantes de la API, con el fin de observar el comportamiento del servidor bajo condiciones de uso repetitivo y concurrente. Aunque se consideraron otros tipos de pruebas funcionales y manuales durante el desarrollo, el enfoque principal de esta etapa estuvo en las pruebas de rendimiento automatizadas con JMeter, ya que permitieron medir tiempos de respuesta, detectar posibles fallos y comprobar la capacidad de la aplicación para mantenerse estable bajo carga.

3.2 Configuración del escenario de carga

Para ejecutar las pruebas, se configuró un *Thread Group* en JMeter, que representa a los usuarios virtuales que interactúan con el sistema. En este caso se definieron 10 hilos (usuarios simulados), un *ramp-up* **de** 5 segundos y un *Loop Count* **de 10 iteraciones** por cada hilo. Significa que los 10 usuarios no comenzaron a enviar solicitudes al mismo tiempo, sino que se distribuyeron a lo largo de 5 segundos, simulando un crecimiento gradual del tráfico. Cada usuario realizó 10 peticiones consecutivas, lo que permitió evaluar no solo la respuesta inicial del sistema, sino también su comportamiento cuando las solicitudes se repiten en un corto período de tiempo.

Además, se activó la opción **"***Same user on each iteration"*, lo cual hace que cada hilo se comporte como un mismo usuario a lo largo de todas sus repeticiones. Esta configuración se eligió para representar más realista el uso del sistema, donde una misma persona puede consultar varias veces la información en vez de generar un usuario nuevo en cada solicitud. No se definieron tiempos máximos de duración ni retrasos adicionales, ya que el objetivo fue observar la respuesta directa del sistema ante una carga constante y controlada.

3.3 Pruebas del endpoint de servicios

La primera serie de pruebas se realizó sobre el endpoint **GET /api/services**, encargado de devolver la lista de servicios ofrecidos por el salón. Este endpoint es fundamental, ya que la información de los servicios se utiliza posteriormente en la creación de citas y en la visualización de opciones para los usuarios. Durante la ejecución de la prueba, los hilos de JMeter enviaron múltiples solicitudes simultáneas para consultar estos datos. Los resultados mostraron tiempos de respuesta bajos y consistentes, sin presencia de errores. Esto indica que las consultas relacionadas con los servicios están bien optimizadas y que el sistema puede soportar accesos concurrentes a esta información sin afectar su estabilidad.

Figura 4

HTTP request de servicios

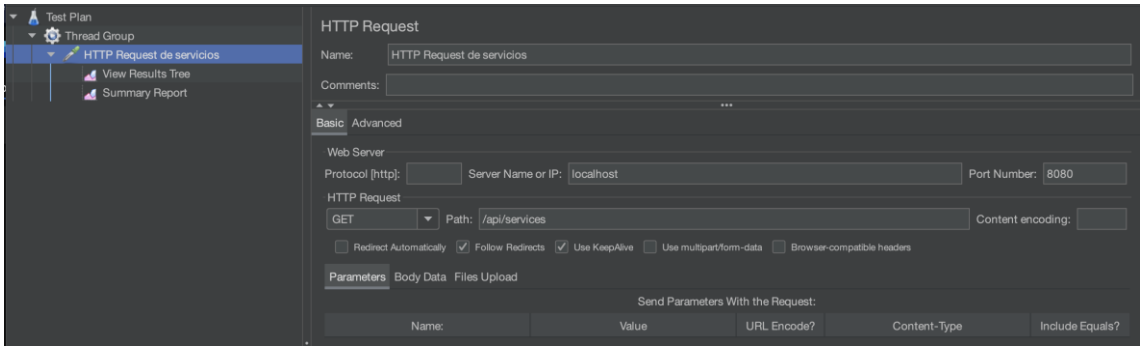
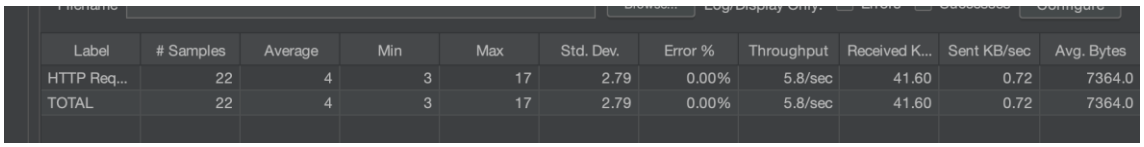


Figura 5

Resultados de HTTP request de citas



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
HTTP Req...	22	4	3	17	2.79	0.00%	5.8/sec	41.60	0.72	7364.0
TOTAL	22	4	3	17	2.79	0.00%	5.8/sec	41.60	0.72	7364.0

3.5 Pruebas del endpoint de citas

Finalmente, cuando se evaluó el endpoint **GET /api/appointments**, encargado de recuperar la información de las citas registradas. Este es uno de los componentes más críticos del sistema, ya que las citas representan datos dinámicos que se actualizan constantemente. Durante la prueba, JMeter simuló múltiples usuarios consultando la lista de citas al mismo tiempo, similar a lo que ocurriría cuando varios empleados o administradores revisan el calendario del salón. Los resultados mostraron un comportamiento estable del servidor, con tiempos de respuesta consistentes y sin errores de conexión o fallos en las consultas. Esto confirma que el sistema puede soportar la carga de consultas de citas sin comprometer la disponibilidad del servicio.

Figura 6

HTTP request de citas

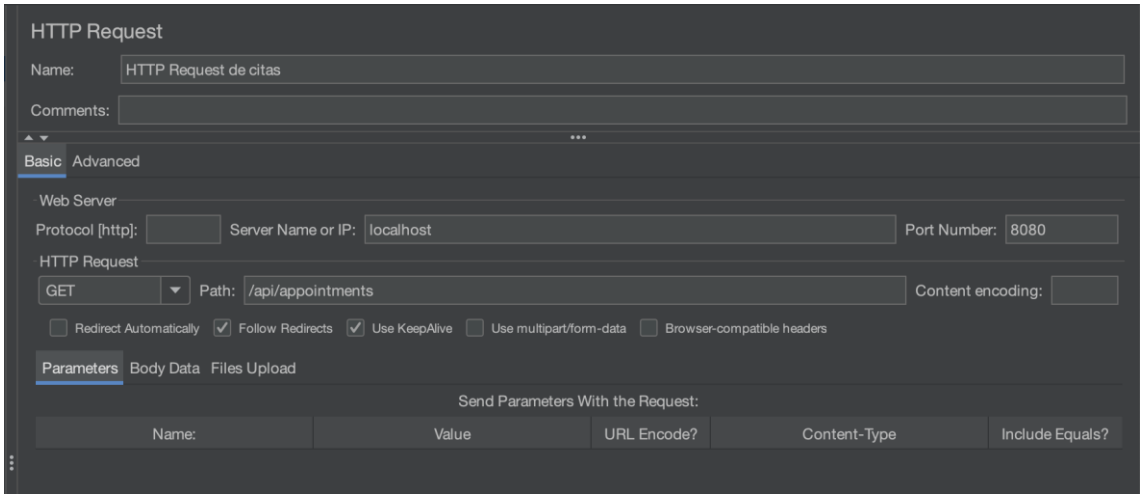
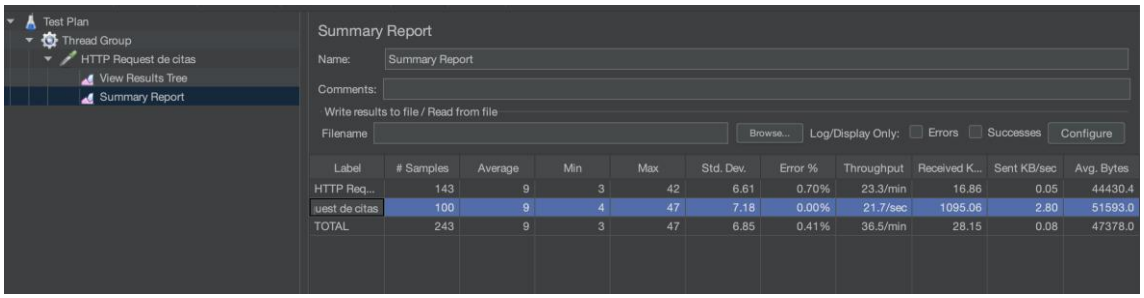


Figura 7

Resutados de HTTP request de citas



3.4 Pruebas del endpoint de técnicos

Posteriormente se probó el endpoint **GET /api/nailtechs**, que devuelve la información de los técnicos de uñas registrados en el sistema. Este endpoint implica una mayor complejidad, ya que cada técnico puede estar relacionado con varios servicios, lo que genera consultas más elaboradas en la base de datos. Pese a esto, el servidor respondió establemente durante todas las iteraciones de los usuarios simulados. Los tiempos de respuesta se mantuvieron dentro de rangos aceptables y no se registraron errores en las solicitudes. Estos resultados demuestran que la estructura de relaciones entre técnicos y servicios está correctamente gestionada y que el sistema puede manejar múltiples consultas simultáneas a este módulo sin degradar su rendimiento.

Figura 8

HTTP request de empleadas

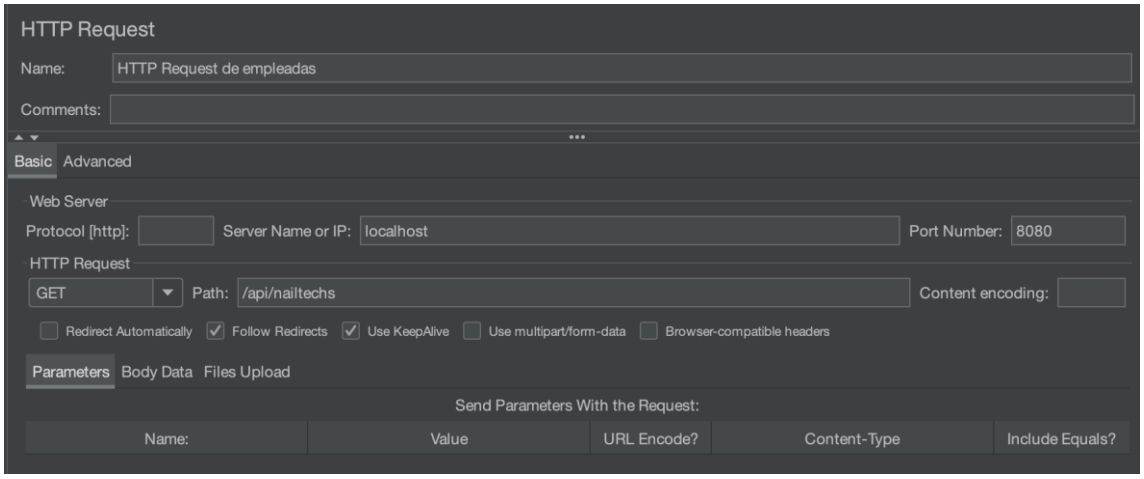
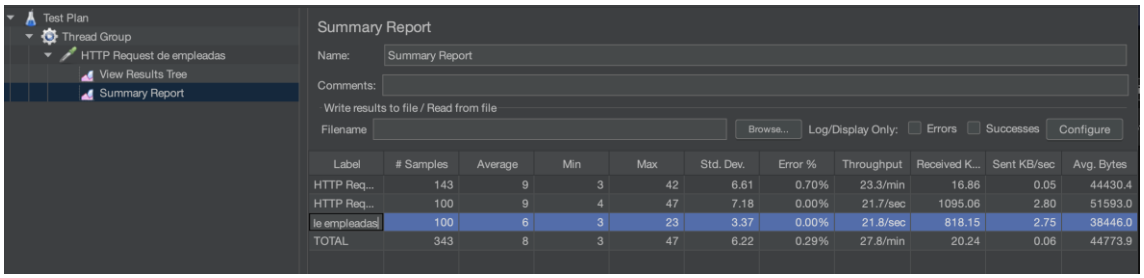


Figura 9

Resultados de HTTP request de empleadas



3.6 Observaciones generales de rendimiento

En conjunto, las pruebas realizadas con Apache JMeter evidencian que el backend del sistema presenta un comportamiento estable frente a una carga moderada de usuarios concurrentes. Durante las ejecuciones se simularon múltiples peticiones simultáneas hacia los endpoints principales de la API, específicamente aquellos encargados de gestionar servicios, técnicos y citas, que representan las operaciones más frecuentes dentro del flujo normal del sistema.

Los resultados obtenidos muestran que los tiempos de respuesta promedio se mantuvieron dentro de rangos adecuados para una aplicación web de este tipo, sin presentar incrementos abruptos ni degradación progresiva del rendimiento a medida que avanzaban las iteraciones. Asimismo, el porcentaje de errores registrado fue nulo o extremadamente bajo, lo cual indica que el servidor logró procesar las solicitudes correctamente incluso bajo condiciones de concurrencia. Esto demuestra que tanto la configuración del servidor como la estructura de la base de datos y las consultas implementadas soportan correctamente múltiples accesos simultáneos.

Otro aspecto relevante observado durante las pruebas fue la consistencia en el rendimiento. Las variaciones entre tiempos mínimos y máximos de respuesta se mantuvieron controladas, lo que sugiere que no existen cuellos de botella críticos en las operaciones evaluadas. Además, la tasa de transferencia de datos y el throughput obtenido reflejan que el sistema es capaz de atender un volumen constante de solicitudes por segundo sin interrupciones.

Es importante señalar que estas pruebas no buscan determinar el límite máximo de carga que el sistema puede soportar, sino validar su funcionamiento en un escenario representativo de uso real. Bajo esta perspectiva, los resultados permiten concluir que la aplicación está preparada para operar en un entorno con múltiples usuarios activos al mismo tiempo, manteniendo un servicio confiable, consistente y con tiempos de respuesta adecuados. Esto refuerza la idea de que la arquitectura implementada es sólida y constituye una base adecuada para futuras ampliaciones o incrementos en la demanda.

Conclusiones

El desarrollo del sistema de gestión para el salón de uñas permitió integrar de manera exitosa conocimientos de arquitectura de software, desarrollo backend, bases de datos, desarrollo frontend y pruebas de rendimiento. A lo largo del proyecto se construyó una aplicación completa que permite administrar servicios, técnicos y citas, cubriendo así los procesos fundamentales de operación de un salón de belleza moderno.

Las pruebas realizadas, especialmente las de carga con Apache JMeter, demostraron que el sistema mantiene tiempos de respuesta estables y un comportamiento confiable ante múltiples solicitudes simultáneas. Esto indica que la estructura de la base de datos, las relaciones entre entidades y la implementación de la API REST fueron diseñadas de forma adecuada, permitiendo que la aplicación soporte un uso concurrente sin degradar significativamente su rendimiento.

Un aspecto importante a destacar es que el sistema fue desarrollado con una arquitectura que favorece la escalabilidad. El backend está desacoplado del frontend, lo que facilita futuras mejoras, ampliaciones de funcionalidad o incluso la integración con aplicaciones móviles u otros servicios externos. Además, el uso de tecnologías estándar

como Spring Boot, React y PostgreSQL permite que el proyecto pueda crecer sin necesidad de rehacer su base tecnológica.

Como parte del trabajo futuro, se contempla la adquisición de un dominio propio y la publicación oficial del sistema en la nube mediante la plataforma Render, lo que permitirá que el sistema esté disponible en línea de forma permanente. Este paso consolidará el proyecto no solo como un ejercicio académico, sino como una solución real y funcional que podría ser utilizada en un entorno comercial. Asimismo, se podrían añadir nuevas funcionalidades como recordatorios automáticos por correo, reportes avanzados o un panel para clientes.

Recomendaciones

- Se recomienda desplegar el sistema en un entorno de producción (por ejemplo, mediante servicios en la nube) para permitir su acceso remoto y facilitar su uso en un entorno real. Esto permitiría evaluar el desempeño del sistema fuera del entorno local y garantizar una mayor disponibilidad.
- Aunque el sistema funciona correctamente en condiciones normales, es recomendable realizar pruebas de carga y estrés de manera periódica para evaluar su comportamiento ante múltiples usuarios concurrentes. Esto ayudará a identificar posibles cuellos de botella y mejorar la escalabilidad del sistema.
- Se sugiere implementar medidas adicionales de seguridad, como la protección de endpoints sensibles, validaciones más estrictas de entrada de datos y el manejo seguro de credenciales, con el objetivo de reducir vulnerabilidades y proteger la información del negocio y de los usuarios.
- Aunque la interfaz es funcional y clara, se recomienda realizar pruebas de usabilidad con usuarios reales para identificar oportunidades de mejora en la navegación, el diseño visual y la accesibilidad, garantizando una experiencia más intuitiva y agradable.
- Finalmente, se recomienda diseñar futuras ampliaciones del sistema considerando la posibilidad de agregar nuevas funcionalidades, como gestión de clientes, historial de servicios o integración con sistemas de pago, manteniendo una arquitectura modular y escalable.

Referencias bibliográficas

- Apache Software Foundation. (2024). *Apache JMeter user manual*. https://jmeter.apache.org
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California, Irvine).
- Oracle. (2024). *Java Platform, Standard Edition documentation*. https://docs.oracle.com/javase/
- PostgreSQL Global Development Group. (2024). *PostgreSQL documentation*. https://www.postgresql.org/docs/
- Render. (2024). *Render documentation*. https://render.com/docs
- React. (2024). *React documentation*. https://react.dev
- Spring. (2024). *Spring Boot reference documentation*. https://docs.spring.io/spring-boot/docs/current/reference/html/

Anexos

Manual de usuario en inglés y español

Repositorio para el frontend

Repositorio para el backend