

Universidad  
Tecnológica de  
Tecámac



# MEDICONNECT

## NEODIGITAL SOLUTIONS

REALIZADO POR:

ARTEAGA GAYOSSO ANGEL JESUS

LOPEZ NAVARIJO LAURA MICHELLE

PUEBLITA BAUTISTA PATRICK GUILLERMO

SANCHEZ TERAN JONATHAN DANIEL

10IDS1

Asesor::

DR. Téllez Barrientos Omar

# ÍNDICE

1.- Introducción al proyecto integrador .....	4
2.- Introducción al DevOps .....	5
3.- Etapa de planeación (DevOps) .....	6
3.1 Objetivo General.....	6
3.2 Comunicación Y Planeación Del Proyecto.....	8
3.2.1 Herramientas de comunicación .....	8
3.2.2 Planeación del Proyecto.....	10
3.3 Estructura de Roles .....	12
3.4 Recursos Técnicos .....	14
3.5 Análisis de requisitos .....	17
3.5.1 Requisitos funcionales .....	17
3.5.2 Requisitos no funcionales .....	18
3.5.3 Requisitos de infraestructura.....	19
3.6 Desarrollo del cronograma.....	21
3.7 Diseño de arquitectura.....	23
3.8 Gestión de riesgos.....	24
4. Etapa De Desarrollo Y Codificacion .....	28

4.1 Configuración del entorno de desarrollo .....	29
4.2 Modelo Relacional .....	31
4.3 Flujo de trabajo .....	33
4.4 Integración Continua (CI) .....	36
5. DESPLIEGUE CONTINUO (CD) .....	38
5.1.1 Automatización de Pipeline CI/CD .....	39
5.1.2 Rollback Automático .....	40
5.1.3 Revisión y actualización periódica del Pipeline .....	41
6.PRUEBAS .....	42
6.1.1 Pruebas Unitarias .....	43
6.1.2 Pruebas de integración .....	44
6.1.3 Pruebas funcionales .....	45
6.1.4 Pruebas de Interfaz de Usuario (UI) .....	46
7. OPERACIÓN .....	48
7.1 Monitoreo de la aplicación .....	49
7.1.2 Backup y Recuperación ante desastres .....	50
7.1.3 Optimización Continua del rendimiento .....	54
Conclusión .....	55

# **1.- INTRODUCCIÓN AL PROYECTO INTEGRADOR**

El proyecto consiste en el desarrollo e implementación de una aplicación web para gestionar las citas de un consultorio médico, que ofrecerá funcionalidades robustas y eficientes para la administración de los procesos relacionados con la atención médica. El sistema de citas en línea e inventario de medicamentos está diseñado como una solución integral. Este sistema permitirá registrar y administrar usuarios, incluyendo pacientes, doctores y personal administrativo, asegurando un control adecuado de los roles y funciones dentro de la plataforma. Los doctores tendrán la capacidad de gestionar citas y, durante las consultas, recetar medicamentos directamente a través del sistema. Cada receta generada será registrada de manera automática y vinculada al perfil del paciente, lo que permitirá que este consulte sus prescripciones en cualquier momento desde su cuenta. Esta funcionalidad garantizará que los pacientes tengan acceso rápido y seguro a sus recetas, facilitando el seguimiento de sus tratamientos. Además, la aplicación web incluirá un módulo de inventario de medicamentos que se actualizará en tiempo real. Cada vez que un medicamento sea recetado, su cantidad en el inventario se reducirá automáticamente, asegurando un control preciso de los insumos disponibles y minimizando el riesgo de errores o desabastecimientos. De esta manera, el sistema optimizará la experiencia tanto de los pacientes como de los doctores y fortalecerá la gestión interna de los recursos médicos, convirtiéndose en una herramienta esencial para la administración eficiente de clínicas o centros de salud.

## **2.- INTRODUCCIÓN AL DEVOPS**

DevOps es una metodología que combina desarrollo y operaciones para optimizar procesos, mejorando la colaboración y acelerar la entrega de software de alta calidad. DevOps no solo es una estrategia tecnológica, sino un cambio cultural que fomenta la integración entre los equipos. La automatización, la monitorización continua y la retroalimentación rápida son algunos de sus pilares fundamentales, los cuales considero esenciales para garantizar un ciclo de desarrollo ágil y eficiente, así como interiorizar la importancia de la comunicación abierta y los objetivos compartidos.

La importancia de DevOps radica en su capacidad para transformar la manera en que se desarrollan y entregan soluciones tecnológicas. DevOps es fundamental para garantizar que las empresas puedan mantenerse competitivas. Además, fomenta la automatización de procesos repetitivos y la integración continua, lo que reduce los errores humanos, acelera los tiempos de entrega y mejora la calidad del producto final. DevOps no solo es una solución técnica, sino un enfoque estratégico que impulsa la innovación, optimiza recursos y crea un entorno donde los equipos trabajan de manera más efectiva para alcanzar el objetivo del proyecto.

En conclusión, DevOps representa un cambio transformador en la forma en que las organizaciones desarrollan y entregan software. Al fomentar la colaboración, la automatización y la mejora continua, DevOps no solo impulsa la innovación, sino que también permite a las empresas responder rápidamente a un entorno tecnológico en constante evolución, asegurando su relevancia y competitividad en el mercado.

### **3.- ETAPA DE PLANEACIÓN (DevOps)**

#### **3.1 Objetivo General**

Desarrollar e implementar una aplicación web para la gestión de citas en un consultorio médico, que incluya un sistema de administración de usuarios, manejo de recetas médicas y un módulo de inventario de medicamentos en tiempo real, con el fin de optimizar los procesos internos, mejorar la experiencia de los pacientes y garantizar un control eficiente de los recursos médicos.

#### **Objetivos Específicos**

- Diseñar un sistema de registro y administración de usuarios que permita gestionar perfiles de pacientes, doctores y personal administrativo con roles y permisos diferenciados.
- Implementar un módulo de gestión de citas en línea que facilite la programación, modificación y cancelación de consultas médicas de manera ágil y accesible para los pacientes.
- Crear una funcionalidad que permita a los doctores generar recetas médicas durante las consultas, vinculándolas automáticamente al perfil del paciente para su consulta posterior.
- Desarrollar un sistema de inventario de medicamentos que registre y actualice en tiempo real las existencias, reduciendo automáticamente las cantidades al momento de recetar.
- Garantizar la seguridad de la información mediante la implementación de mecanismos de autenticación, autorización y encriptación de datos sensibles.

- Proporcionar una interfaz amigable y accesible, compatible con diferentes dispositivos, para mejorar la experiencia del usuario.
- Realizar pruebas de funcionalidad y rendimiento para asegurar la calidad del sistema antes de su implementación final.

## **3.2 Comunicación Y Planeación Del Proyecto**

### **3.2.1 Herramientas de comunicación**

Microsoft teams:

Microsoft Teams como herramienta principal de comunicación para el equipo de desarrollo y DevOps, facilitando la colaboración, seguimiento de tareas y toma de decisiones en el proyecto. A continuación, se presentan pruebas de reuniones para el desarrollo del proyecto:

#### **Canales de Comunicación**

Se utilizarán los siguientes canales en Microsoft Teams para organizar la información y mantener una comunicación eficiente. El canal General se utilizará para las comunicaciones globales del proyecto, anuncios importantes y documentación. Desarrollo estará destinado a la discusión sobre código, revisión de PRs y resolución de dudas técnicas. DevOps servirá para la coordinación de despliegues, monitoreo y resolución de problemas de infraestructura. Incidentes será el espacio para reportar y documentar la solución de problemas críticos en producción. Finalmente, Reuniones se usará para la organización de reuniones y la gestión de agendas.

#### **WhatsApp**

WhatsApp será un canal de comunicación complementario para agilizar la coordinación y resolver dudas urgentes. Se utilizará un grupo de equipo para mensajes rápidos, consultas inmediatas y notificaciones urgentes, mientras que los mensajes privados solo se emplearán para resolver dudas específicas sin



interrumpir la comunicación general. En caso de una falla crítica fuera del horario laboral, se recurrirá a WhatsApp para alertar al equipo responsable y escalar el incidente de manera inmediata.

## **Reglas y Buenas Prácticas**

Las decisiones técnicas deben registrarse en el canal correspondiente. Los reportes de incidentes deben documentarse en el canal de Incidentes con detalles del problema y la solución. Las reuniones deben programarse con anticipación y las actas deben compartirse en el canal de Reuniones. Se debe evitar el uso de mensajes privados para temas de interés general, priorizando siempre los canales abiertos. Además, se deben considerar integraciones con otras herramientas para mejorar la eficiencia y la comunicación del equipo. Para mejorar la productividad, se integrará Microsoft Teams con las siguientes herramientas:

- GitHub → Notificaciones de commits y PRs.
- Trello → Seguimiento de tareas y sprints.

Como evidencia del uso de estas herramientas, se han llevado a cabo conferencias sobre temas relevantes para el desarrollo del proyecto a través de Microsoft Teams y se han enviado mensajes a través de WhatsApp para la coordinación del equipo. A continuación, se presentan registros y capturas que demuestran la implementación efectiva de estos medios de comunicación.

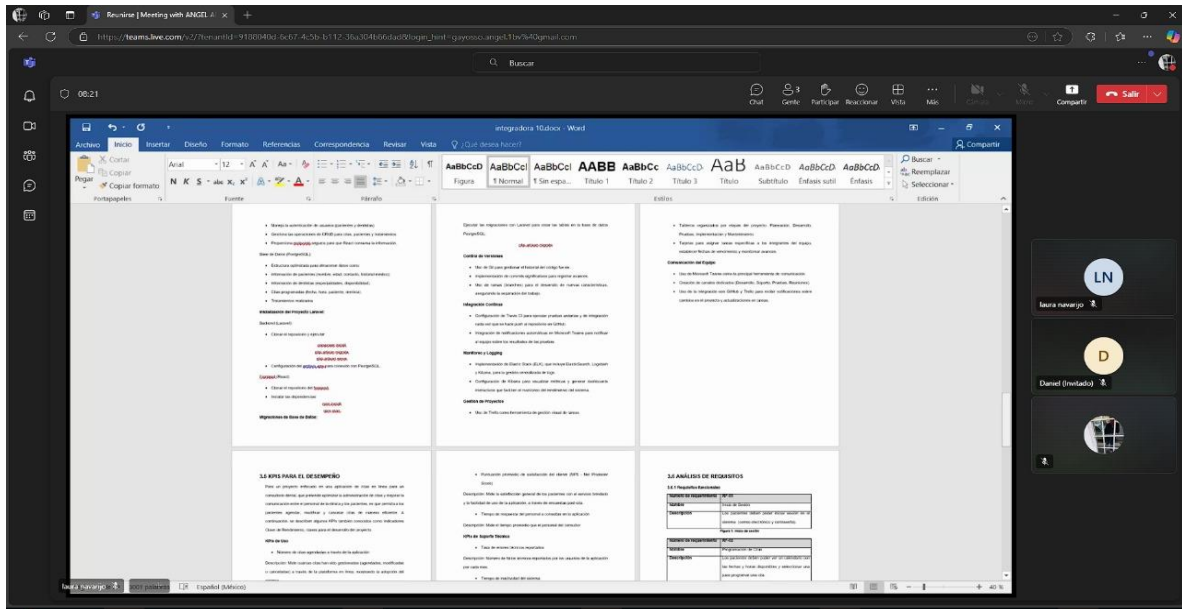


Figura 1. Devops

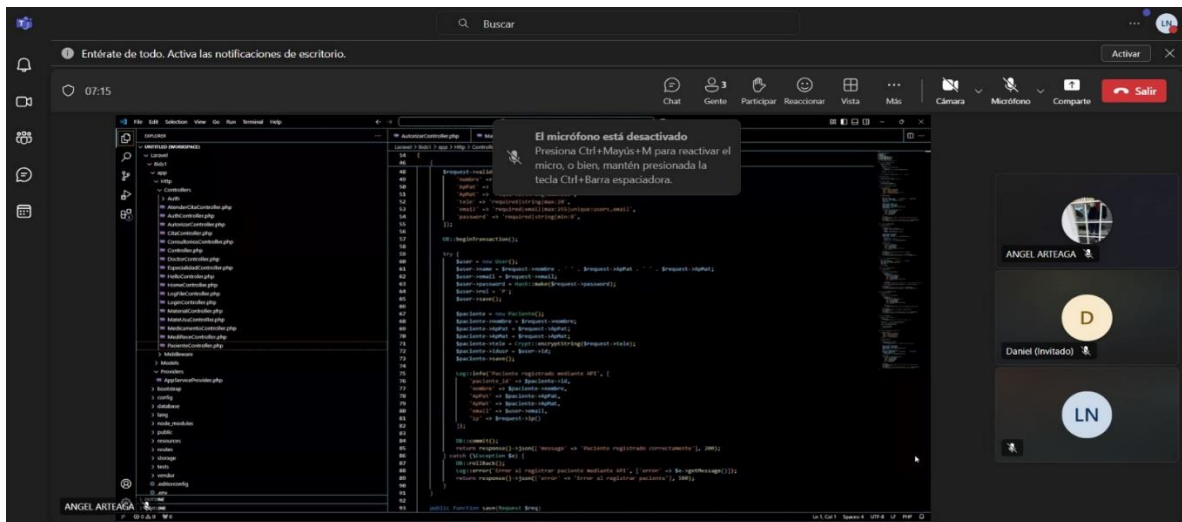


Figura 2. Codificación del proyecto

### 3.2.2 Planeación del Proyecto

Para la gestión del desarrollo del proyecto, se implementará la metodología Scrumban, una combinación de Scrum y Kanban, con el objetivo de optimizar la planificación y el flujo de trabajo en un periodo de dos meses.

## **Estructura y Organización**

El equipo de desarrollo organizará el trabajo en sprints de 1 a 2 semanas, estableciendo objetivos específicos y entregables parciales de la aplicación web. Cada sprint comenzará con una reunión de planificación y concluirá con una revisión de avances. Para mejorar el seguimiento de tareas y la comunicación, se utilizará un tablero Kanban digital en herramientas como Trello, el cual contará con las siguientes columnas: Backlog, donde se listarán tareas planificadas pero aún no priorizadas; Por hacer (To Do), con actividades seleccionadas para el sprint actual; En proceso (In Progress), para las tareas en desarrollo; En revisión (Review), donde estarán las tareas completadas pendientes de validación; y Completado (Done), donde se ubicarán las tareas finalizadas y aprobadas.

### 3.3 Estructura de Roles



**Ángel Jesús Arteaga Gayosso**

- **Líder de Proyecto**
- Coordinar y supervisar el desarrollo del proyecto.
- Facilitar la comunicación entre los integrantes.
- Gestionar el backlog y priorizar tareas.
- Asegurar que se sigan las metodologías ágiles establecidas.



**Patrick Guillermo Pueblita Bautista**

- **Desarrollador Backend**
- Diseñar y desarrollar la lógica de negocio de la aplicación.
- Implementar la API y gestionar la base de datos.
- Asegurar la seguridad y rendimiento del sistema.
- Realizar pruebas unitarias y de integración.



**Laura Michelle López Navarajo**

- **Desarrollador Frontend**
- Diseñar e implementar la interfaz de usuario.
- Garantizar una experiencia de usuario óptima y accesible.
- Integrar el frontend con el backend.
- Optimizar la aplicación para diferentes dispositivos y navegadores.



**Jonathan Daniel Sánchez Terán**

- **Tester de Software**

- Diseñar y ejecutar casos de prueba para verificar el funcionamiento de la aplicación.
- Identificar, documentar y reportar errores o fallos en el sistema.
- Realizar pruebas manuales y automatizadas en frontend y backend.

## **3.4 Recursos Técnicos**

### **Tecnologías Utilizadas**

Las tecnologías utilizadas en el proyecto de consultorio médico incluyen React para el frontend, proporcionando una interfaz de usuario interactiva y dinámica para pacientes y dentistas, con componentes reutilizables y diseño responsivo. En el backend, se emplea Laravel (PHP) para gestionar la lógica de negocio y generar una API RESTful que facilite la comunicación con el frontend. La base de datos utilizada es PostgreSQL, encargada de almacenar información estructurada de pacientes, dentistas y citas. Para el control de versiones, se centraliza el repositorio del código fuente en GitHub. En cuanto a la integración continua, se utiliza Travis CI para la ejecución de pruebas automatizadas y la verificación de la calidad del código en cada push. Docker se emplea para la gestión de entornos de desarrollo y producción uniformes. Para el monitoreo y logging, se implementa Elastic Stack (ELK) con Kibana, permitiendo la visualización de métricas y logs del sistema. La comunicación entre los miembros del equipo se realiza a través de Microsoft Teams, asegurando una colaboración efectiva. Finalmente, la gestión del proyecto se lleva a cabo con Trello, permitiendo la planificación de tareas, la organización de sprints y el seguimiento del avance del proyecto.

### **Arquitectura del Sistema**

La arquitectura del sistema se compone de tres elementos principales. Frontend (React): una aplicación web donde los pacientes pueden registrarse, iniciar sesión, programar citas dentales, ver su historial de tratamientos y recibir notificaciones, mientras que los dentistas pueden gestionar citas, acceder al historial de cada

paciente y actualizar tratamientos. Backend (Laravel): una API RESTful encargada de manejar la autenticación de usuarios (pacientes y dentistas), gestionar operaciones CRUD para citas, pacientes y tratamientos, y proporcionar endpoints seguros para la comunicación con React. Base de Datos (PostgreSQL): diseñada para almacenar información optimizada, incluyendo datos de pacientes como nombre, edad, contacto e historial médico, información de dentistas con especialidades y disponibilidad, citas programadas con fecha, hora, paciente y dentista, y tratamientos realizados. Para la inicialización del proyecto Laravel, se debe clonar el repositorio, ejecutar `composer install`, `php artisan migrate`, `php artisan serve`, y configurar el archivo `.env` para la conexión con PostgreSQL. En el frontend (React), se debe clonar el repositorio, instalar las dependencias con `npm install` y ejecutar `npm start`. La migración de base de datos se realiza con `php artisan migrate` para la creación de las tablas en PostgreSQL. En cuanto al control de versiones, se utiliza Git para gestionar el historial del código fuente, implementando commits significativos y ramas (branches) para el desarrollo de nuevas características, asegurando la separación del trabajo. Para la integración continua, Travis CI ejecutará pruebas unitarias e integración en cada push al repositorio en GitHub, con notificaciones automáticas en Microsoft Teams para informar al equipo sobre los resultados. El monitoreo y logging se gestionará mediante Elastic Stack (ELK), compuesto por Elasticsearch, Logstash y Kibana, lo que permitirá centralizar logs, visualizar métricas y generar dashboards interactivos para el monitoreo del rendimiento del sistema. La gestión de proyectos se llevará a cabo con Trello, organizando tableros por etapas como Planeación, Desarrollo, Pruebas, Implementación y Mantenimiento, con tarjetas asignadas a los integrantes del

equipo para establecer fechas de vencimiento y dar seguimiento a los avances. Finalmente, la comunicación del equipo se realizará mediante Microsoft Teams, con canales dedicados a Desarrollo, Soporte, Pruebas y Reuniones, integrando GitHub y Trello para recibir notificaciones sobre cambios en el proyecto y actualizaciones en las tareas.

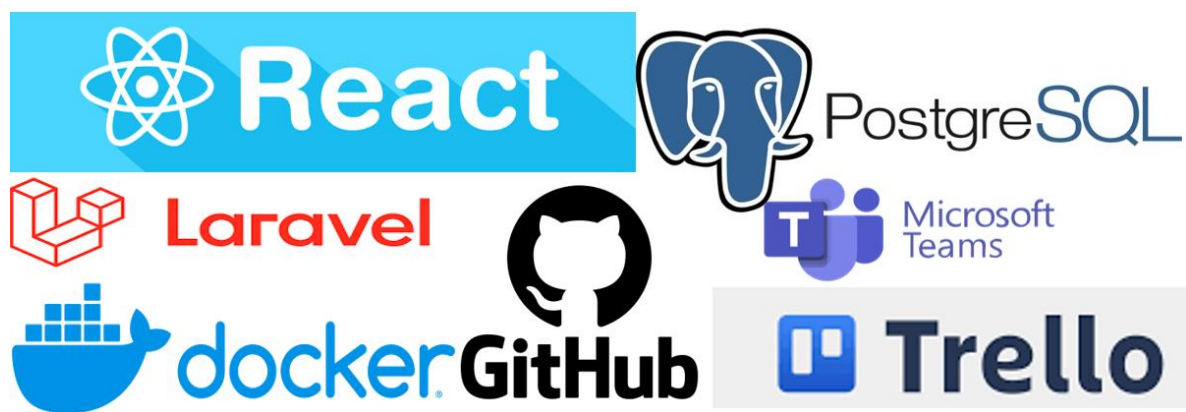


Figura 3. Herramientas



## 3.5 Análisis de requisitos

### 3.5.1 Requisitos funcionales

<b>Número de requerimiento</b>	<b>RF-01</b>
<b>Nombre</b>	Inicio de Sesión
<b>Descripción</b>	Los pacientes deben poder iniciar sesión en el sistema (correo electrónico y contraseña).

**Figura 4. Inicio de sesión**

<b>Número de requerimiento</b>	<b>RF-02</b>
<b>Nombre</b>	Programación de Citas
<b>Descripción</b>	Los pacientes deben poder ver un calendario con las fechas y horas disponibles y seleccionar una para programar una cita.

**Figura 5. Programación de citas**

<b>Número de requerimiento</b>	<b>RF-03</b>
<b>Nombre</b>	Cancelación de Citas
<b>Descripción</b>	Los pacientes deben poder cancelar una cita programada a través de su cuenta.

**Figura 6. Cancelación de citas**

<b>Número de requerimiento</b>	<b>RF-05</b>
<b>Nombre</b>	Modificación de Citas
<b>Descripción</b>	Los pacientes deben poder modificar los detalles de una cita programada a través de su cuenta.

**Figura 7. Modificación de citas**

<b>Número de requerimiento</b>	<b>RF-06</b>
<b>Nombre</b>	Registro de pacientes
<b>Descripción</b>	Los pacientes deben poder registrarse en el sistema.

**Figura 8. Registro de pacientes**

<b>Número de requerimiento</b>	<b>RF-06</b>
<b>Nombre</b>	Eliminar citas
<b>Descripción</b>	Los pacientes pueden eliminar citas previamente programadas.

**Figura 9. Eliminar citas**

### 3.5.2 Requisitos no funcionales

<b>Número de requerimiento:</b>	<b>RNF-01</b>
<b>Nombre</b>	Interfaz del sistema
<b>Descripción</b>	La interfaz de usuario debe ser intuitiva y fácil de usar, así como ser accesible desde dispositivos móviles y de escritorio.

**Figura 10. Interfaz del sistema**

<b>Número de requerimiento</b>	<b>RNF-02</b>
<b>Nombre</b>	Disponibilidad
<b>Descripción</b>	El sistema deber estar disponible en todo momento.

**Figura 11. Disponibilidad**

<b>Número de requerimiento</b>	<b>RNF-03</b>
<b>Nombre</b>	Colores del sistema
<b>Descripción</b>	El sistema debe contar con los colores deseados por la clínica.

**Figura 12. Colores del sistema**

<b>Número de requerimiento</b>	<b>RNF-04</b>
<b>Nombre</b>	Compatibilidad
<b>Descripción</b>	El sistema debe ser compatible con los principales navegadores web (Chrome y Microsoft Edge) y funcionar correctamente en diferentes sistemas operativos (Windows y Android).

**Figura 13. Compatibilidad**

### 3.5.3 Requisitos de infraestructura

<b>Número de requerimiento</b>	<b>RI-01</b>
<b>Nombre:</b>	Hardware
<b>Descripción:</b>	Se requiere un procesador Intel Core I7, RAM 16 GB, Windows 11 Pro.

**Figura 14. Hardware a utilizar.**

<b>Número de requerimiento</b>	<b>RI-03</b>
<b>Nombre:</b>	Copias de seguridad

<b>Descripción:</b>	Implementar un sistema de copias de seguridad automáticas para la base de datos y archivos críticos, con frecuencia diaria.
---------------------	---

**Figura 15. Herramientas de seguridad.**

<b>Número de requerimiento</b>	<b>RI-04</b>
<b>Nombre:</b>	Escalabilidad
<b>Descripción:</b>	La infraestructura debe permitir escalabilidad horizontal mediante el uso de Docker para gestionar contenedores.

**Figura 16. Escalabilidad.**

<b>Número de requerimiento</b>	<b>RI-05</b>
<b>Nombre:</b>	Aplicaciones
<b>Descripción:</b>	El sistema debe utilizar Laravel, React GitHub, Travis CI, Docker, Microsoft Teams y Trello para asegurar un desarrollo ágil y eficiente.

**Figura 17. Aplicaciones**

### 3.6 Desarrollo del cronograma

Este cronograma proporciona una visión clara de las actividades y su secuencia, con la finalidad de que cada fase del proyecto se complete de manera organizada y eficiente.

[illegible]



### **3.7 Diseño de arquitectura**

La arquitectura del sistema sigue el patrón Modelo-Vista-Controlador (MVC) para garantizar escalabilidad, flexibilidad y robustez en la gestión de citas médicas. Los componentes principales son los siguientes: Front-End (Vista): desarrollado con React, proporcionando una interfaz de usuario moderna, dinámica y adaptable a múltiples dispositivos. Se comunica con el backend a través de APIs RESTful y gestiona el estado global con Context API o Redux para una mejor experiencia de usuario. Back-End (Controlador y Modelo): basado en Laravel, encargado de desarrollar la API RESTful, gestionar la lógica del negocio y manejar la autenticación mediante JWT (JSON Web Tokens) para proteger las sesiones de los usuarios. Incluye endpoints para la gestión de citas, pacientes y médicos, además de integraciones con servicios de mensajería como correo electrónico, WhatsApp o SMS para recordatorios. Base de Datos (Modelo): utiliza PostgreSQL como sistema relacional para almacenar y gestionar datos estructurados de pacientes, médicos, citas y expedientes médicos. Se optimizan consultas con índices y se aplican estrategias de backup automático para garantizar la disponibilidad y recuperación de datos en caso de fallos.

### 3.8 Gestión de riesgos

A continuación, se presenta cada una de las tablas que detalla algunos de los riesgos más comunes en la administración de proyectos, personal, tecnológicos y de entrega, junto con su impacto, probabilidad, prioridad, consecuencias y clasificación de riesgo.

Riesgos del Personal								
ID	Riesgo	Impacto (1-3)	Probabilidad (1-3)	Prioridad (1-3)	Consecuencias	Tipo de riesgo		
						Alto	Medio	Bajo
1 AVES	Enfermedad	2	2	2	No poder terminar avances a tiempo			
2 IACP	Abandono del proyecto	3	2	3	Interrupción del flujo de trabajo y falta de equipo			
3 CMWO	Resistencia al cambio	2	3	2	Dificultad para implementar nuevas metodologías o tecnologías.			

Figura 19. Riesgos Personales



Riesgos Tecnológicos								
ID	Riesgo	Impacto (1-3)	Probabilidad (1-3)	Prioridad (1-3)	Consecuencias	Tipo de riesgo		
						Alto	Medio	Bajo
1 AVES	Problemas de compatibilidad con versiones antiguas	2	2	2	Necesidad de rediseño, aumento de costos, retrasos en el proyecto			
2 IACP	Dependencia de una tecnología específica	3	2	3	Dificultad para encontrar expertos, aumento de costos, posibles retrasos			
3 CMW O	Fallo del servidor	3	2	3	Interrupción del servidor, pérdida de datos, impacto en la continuidad del sistema.			

Figura 20. Riesgos Tecnológicos

Riesgos de Entrega								
ID	Riesgo	Impacto (1-3)	Probabilidad (1-3)	Prioridad (1-3)	Consecuencias	Tipo de riesgo		
						Alto	Medio	Bajo
1 AVES	Cambio de requerimientos	2	2	3	Revisión de alcance, retrasos en la entrega			
2 IACP	Retrasos en la planificación	3	2	3	Aumento en los costos, pérdida de confianza del cliente			
3 CMW O	Dependencia de terceros	2	2	3	Retrasos debido a problemas externos, incumplimiento de plazos			

Figura 21. Riesgos de Entrega

Riesgos de Administración del Proyecto								
ID	Riesgo	Impacto (1-3)	Probabilidad (1-3)	Prioridad (1-3)	Consecuencias	Tipo de riesgo		
						Alto	Medio	Bajo
1 AVES	Falta de líder efectivo	3	2	3	Desorganización, falta de dirección clara, conflictos internos			
2 IACP	Comunicación deficiente	3	2	3	Malentendidos, entregables incorrectos, retrasos en las decisiones			
3 CMW O	Riesgo de presupuesto	3	3	3	Sobrecostos, necesidad de financiamiento adicional			

Figura 22. Riesgos de Administración del Proyecto

## **4. ETAPA DE DESARROLLO Y CODIFICACION**

La etapa de desarrollo y codificación es fundamental en el ciclo de vida del proyecto de gestión de citas e inventario para el consultorio médico. Para garantizar un flujo de trabajo eficiente y colaborativo, se adopta la metodología DevOps, integrando prácticas de desarrollo ágil y automatización de procesos. Durante esta fase, el equipo de desarrollo implementa un sistema en línea que incluye dos aplicaciones móviles: una para los pacientes, que les permite programar, modificar y cancelar citas de manera intuitiva, y otra para los doctores, donde pueden gestionar sus consultas y prescribir medicamentos de manera digital. Además, se desarrolla un sistema de administración dirigido al personal del consultorio, que permite gestionar usuarios, citas, recetas médicas y el inventario de medicamentos en tiempo real. Desde un enfoque de consultoría tecnológica, se diseñan e implementan soluciones personalizadas para optimizar la operatividad del consultorio. Esto incluye la configuración de entornos de desarrollo y producción, la aplicación de prácticas de codificación segura y la implementación de procesos de integración y entrega continua (CI/CD) para garantizar la calidad del código y facilitar la colaboración entre desarrolladores. Este enfoque integrado no solo optimiza el proceso de desarrollo, sino que también permite una rápida adaptación a cambios y una respuesta eficiente ante posibles incidentes, asegurando una implementación exitosa del sistema de gestión de citas e inventario en el consultorio médico.

## **4.1 Configuración del entorno de desarrollo**

Para el desarrollo del sistema de gestión de citas e inventario del consultorio médico, se utilizará Laravel como framework base, garantizando una estructura robusta, segura y escalable. La arquitectura del sistema estará diseñada para optimizar la operatividad del consultorio mediante herramientas que faciliten el desarrollo, la integración y el despliegue eficiente del software. Desde un enfoque de consultoría tecnológica, se priorizará la selección de herramientas y tecnologías que mejor se adapten a las necesidades del consultorio médico, considerando factores como la seguridad de los datos, la accesibilidad de la información y la eficiencia en la gestión de los recursos. Para ello, se implementará un entorno de desarrollo que incluya las siguientes herramientas y configuraciones clave: El sistema está diseñado bajo la arquitectura Modelo-Vista-Controlador (MVC) para garantizar una mejor organización del código y facilitar futuras modificaciones. El backend, desarrollado en Laravel, cuenta con controladores bien estructurados que permiten una gestión eficiente de la lógica del negocio. Para el almacenamiento de datos, se utiliza PostgreSQL, seleccionado por su fiabilidad, alto rendimiento y capacidad para manejar grandes volúmenes de información médica con integridad y seguridad. El frontend, desarrollado en React, proporciona una interfaz de usuario dinámica e intuitiva, permitiendo que pacientes, doctores y personal administrativo interactúen con el sistema de manera fluida. La seguridad y autenticación se gestionan mediante OAuth 2.0 y JWT, garantizando el control de accesos, la protección de datos sensibles y el cumplimiento de normativas de seguridad en la gestión de información médica. En cuanto a la infraestructura y virtualización, se

emplea Docker para la contenedorización del sistema, asegurando entornos homogéneos en cada fase del desarrollo y facilitando la implementación en distintos servidores. Para la integración continua y el despliegue, se utilizan GitHub Actions y Laravel Forge, permitiendo la automatización de pruebas, integración y despliegue, reduciendo errores y optimizando los tiempos de entrega. Finalmente, el monitoreo y mantenimiento del sistema se gestiona con herramientas como Prometheus y Grafana, que permiten supervisar el rendimiento del sistema en tiempo real, asegurando su estabilidad y disponibilidad en todo momento. La correcta configuración del entorno de desarrollo garantizará que el sistema no solo cumpla con los requisitos funcionales del consultorio médico, sino que también se mantenga escalable, seguro y fácil de mantener, permitiendo futuras mejoras y adaptaciones según las necesidades del consultorio.

## 4.2 Modelo Relacional

El modelo relacional del sistema está diseñado para gestionar de manera eficiente la información del consultorio médico, asegurando la integridad de los datos y optimizando el acceso a la información. Las entidades principales que componen este modelo incluyen Pacientes, que almacena los datos personales de los pacientes como nombre, apellidos, número de teléfono, dirección, entre otros, permitiendo realizar un seguimiento de los pacientes que asisten al consultorio; Doctores, en la que se guardan los datos relacionados con el personal médico, tales como nombre, especialidad, número de contacto y horario de atención, asociando a cada doctor una especialidad específica que se puede vincular con las citas programadas; Citas, que contiene los registros de las consultas médicas programadas, vinculando a los pacientes con los doctores y las especialidades correspondientes, detallando la fecha, hora y otros aspectos de cada cita; Medicamentos y Materiales, que gestiona los insumos y recursos utilizados en las consultas médicas, permitiendo mantener un control de inventario de los medicamentos y materiales disponibles en el consultorio; y Consultorios y Especialidades, que organiza la distribución de los espacios médicos (consultorios) y las áreas de atención disponibles, donde cada consultorio puede estar asociado a una o varias especialidades médicas. Este modelo relacional establece las relaciones entre las entidades a través de claves primarias (PK) y claves foráneas (FK). La clave primaria identifica de manera única a cada registro dentro de una tabla, mientras que las claves foráneas sirven para establecer vínculos entre las diferentes tablas del sistema, por ejemplo, las Citas tienen claves foráneas que se

refieren a los Pacientes y los Doctores, mientras que los Doctores están asociados con una Especialidad, garantizando así la integridad de los datos y permitiendo acceder de manera eficiente a la información relacionada.

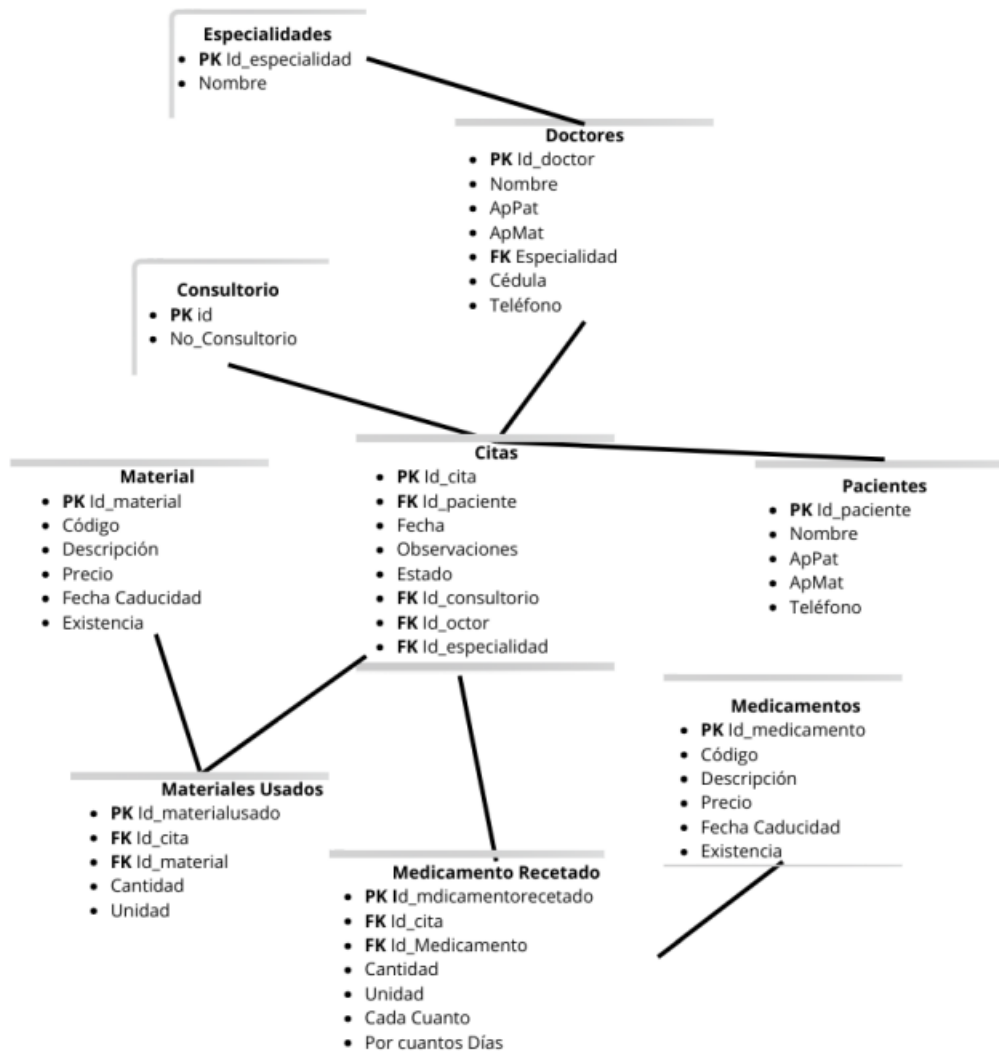


Figura 23. Modelo relacional



## 4.3 Flujo de trabajo

A continuación, se muestra el flujo de trabajo de Laravel, el framework utilizado para el desarrollo del sistema de gestión del consultorio médico. Laravel sigue la arquitectura MVC (Modelo-Vista-Controlador), lo que permite una separación clara de la lógica de negocio, la presentación y el acceso a datos. A continuación, se muestra el flujo de trabajo del proyecto bien definida ha sido fundamentada en GitHub para gestionar eficazmente el desarrollo de software:

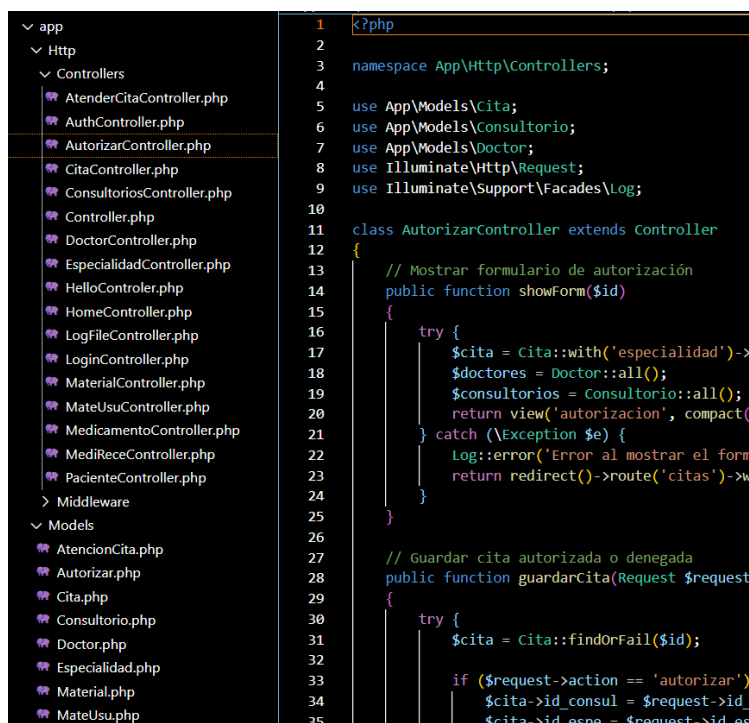
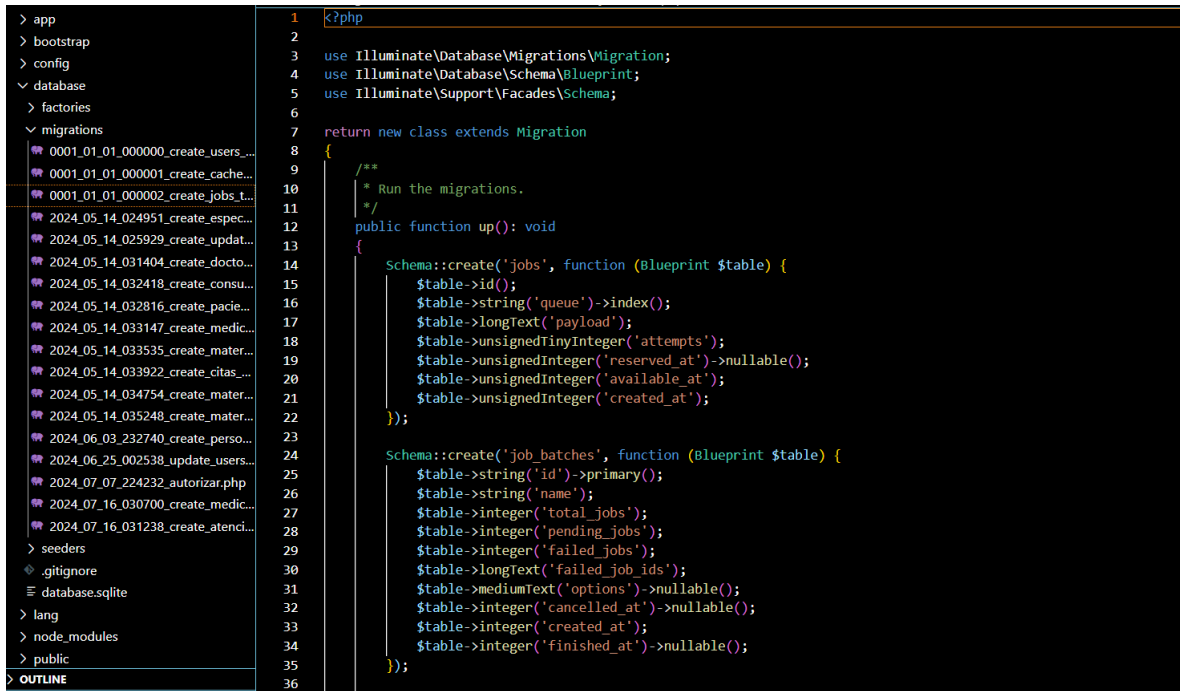


Figura 24. Flujo de trabajo

## Migraciones

Las migraciones en Laravel permiten controlar y versionar los cambios en la estructura de la base de datos del proyecto. Actúan como un controlador de

versiones para la base de datos, facilitando la definición, modificación y compartición de la estructura de manera programática.



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('jobs', function (Blueprint $table) {
15             $table->id();
16             $table->string('queue')->index();
17             $table->longText('payload');
18             $table->unsignedTinyInteger('attempts');
19             $table->unsignedInteger('reserved_at')->nullable();
20             $table->unsignedInteger('available_at');
21             $table->unsignedInteger('created_at');
22         });
23
24         Schema::create('job_batches', function (Blueprint $table) {
25             $table->string('id')->primary();
26             $table->string('name');
27             $table->integer('total_jobs');
28             $table->integer('pending_jobs');
29             $table->integer('failed_jobs');
30             $table->longText('failed_job_ids');
31             $table->mediumText('options')->nullable();
32             $table->integer('cancelled_at')->nullable();
33             $table->integer('created_at');
34             $table->integer('finished_at')->nullable();
35         });
36     }
37 }
```

Figura 25. Migraciones

## Procesamiento de controladores

Una vez que el usuario realiza una solicitud dentro del sistema (por ejemplo, agendar una cita, actualizar su información o consultar el inventario de medicamentos), Laravel dirige esta solicitud a un **Controlador**, que es el encargado de manejar la lógica del sistema y determinar qué acción se debe realizar.

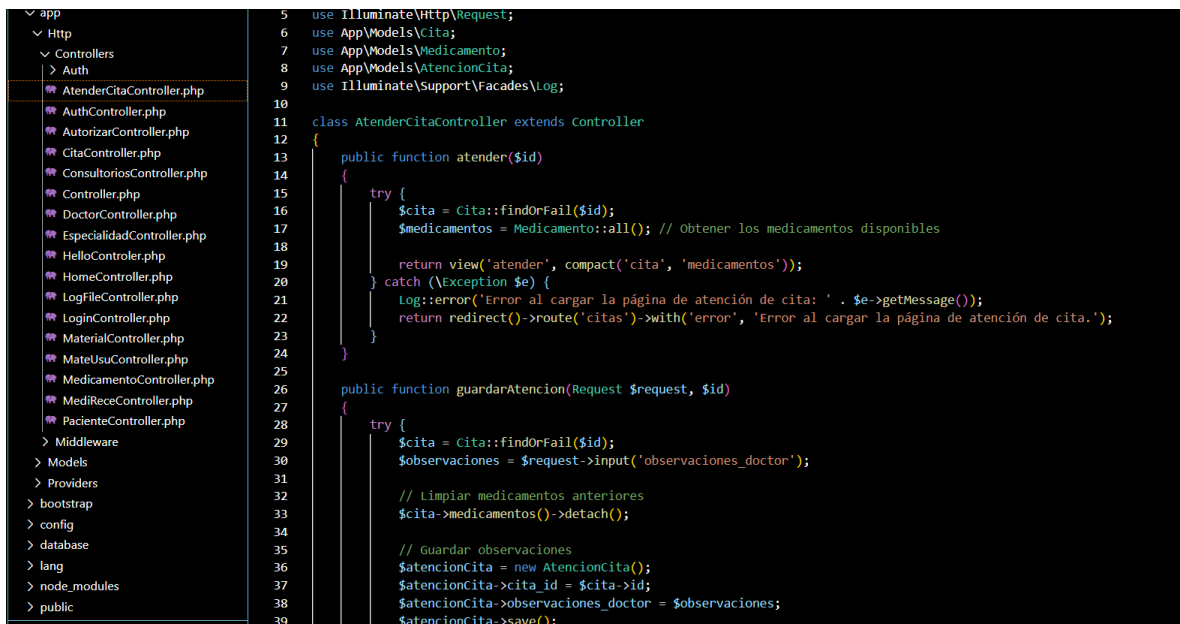


Figura 26. Controller

## PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales y objeto-relacional de código abierto que nos permitirá gestionar de manera eficiente la base de datos del proyecto. En el archivo “.env” del proyecto, la configuración de la base de datos sería:

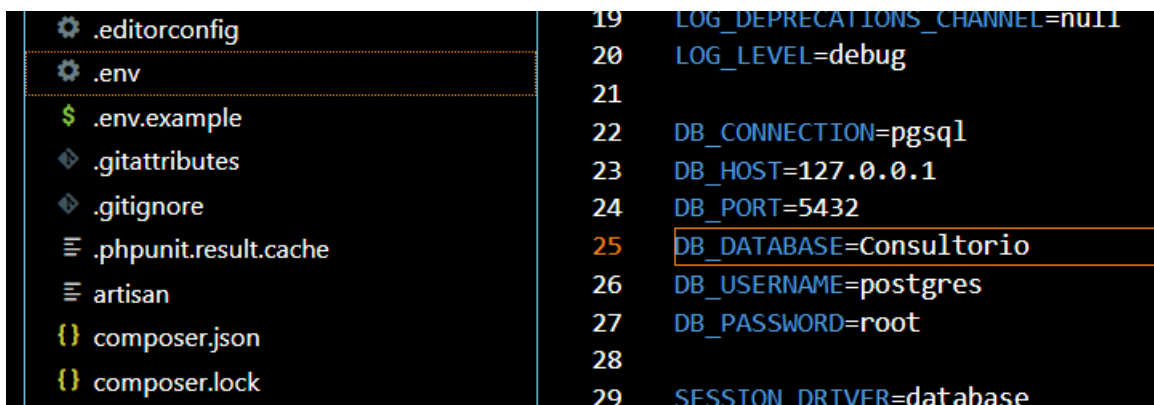


Figura 27. PostgreSQL

## **4.4 Integración Continua (CI)**

Es una práctica clave en el desarrollo moderno de software que permite a los equipos integrar su trabajo de forma frecuente y automatizada. Proporciona una plataforma integral para la gestión del ciclo de vida del desarrollo de software, desde el control de versiones hasta la integración y entrega continua (CI/CD), con una experiencia de usuario colaborativa y centralizada.

Se utiliza GitHub como repositorio centralizado para gestionar el código fuente del sistema de gestión de citas e inventario del consultorio médico, lo que permite a los equipos colaborar en tiempo real, gestionar versiones y realizar revisiones de código para asegurar la calidad del desarrollo. Se implementan GitHub Actions y Laravel Forge para automatizar las tareas de construcción, pruebas y despliegue, garantizando que cada cambio en el código sea validado automáticamente antes de su implementación, reduciendo errores y optimizando los tiempos de entrega. Para la planificación y organización del desarrollo, se emplea Trello, utilizando tableros Kanban para gestionar tareas, asignación de responsables y visualizar el flujo de trabajo, lo que optimiza la productividad del equipo. Las revisiones de código se realizan a través de pull requests en GitHub, asegurando la calidad antes de fusionar nuevas funcionalidades en la rama principal y mejorando la eficiencia del desarrollo. Prometheus y Grafana se implementan para monitorear el rendimiento del sistema en tiempo real, supervisando la estabilidad y disponibilidad del sistema en entornos de producción. La funcionalidad de Wiki en GitHub se utiliza para documentar procesos, guías de desarrollo y pautas del proyecto, facilitando la transferencia de conocimiento y la colaboración entre los miembros del equipo. Para

garantizar un control de accesos seguro y proteger los datos sensibles, se integran OAuth 2.0 y JWT para la autenticación de usuarios, además de implementar Elastic Stack (ELK) con Kibana para la visualización de logs y auditoría de eventos, permitiendo la detección temprana de posibles vulnerabilidades. La implementación de estas herramientas y prácticas DevOps asegura que el sistema de gestión de citas e inventario del consultorio médico sea escalable, seguro y fácil de mantener, permitiendo futuras mejoras y adaptaciones según las necesidades del consultorio.

## 5. DESPLIEGUE CONTINUO (CD)

El despliegue continuo es una práctica clave en el desarrollo moderno de software, permitiendo entregas frecuentes y organizadas con un alto nivel de automatización. Para el desarrollo del sistema de gestión de citas e inventario del consultorio médico, la implementación de esta metodología garantizará una operatividad eficiente y segura. Automatizar el despliegue del sistema de gestión de citas e inventario permitirá implementar nuevas funciones y corregir errores sin intervención manual, asegurando una experiencia fluida tanto para pacientes como para el personal del consultorio. Herramientas como GitHub Actions y Laravel Forge serán clave en la optimización del flujo de integración y entrega continua (CI/CD), permitiendo que cada cambio en el código se valide y despliegue de manera eficiente. Dado que el sistema maneja información médica sensible, es fundamental garantizar su correcto funcionamiento mediante pruebas automatizadas. Se implementarán pruebas de unidad, de integración y funcionales para validar la autenticación con OAuth 2.0 y JWT, la gestión de citas y la administración del inventario. Travis CI se utilizará para ejecutar estas pruebas con cada cambio en el código, asegurando la estabilidad y seguridad del sistema. La adopción del despliegue continuo en este proyecto permitirá iteraciones rápidas, mejorando la calidad del software y reduciendo el tiempo de entrega de nuevas funcionalidades. Además, facilitará la detección temprana de errores, optimizando la experiencia del usuario y garantizando la disponibilidad del sistema en todo momento. El monitoreo y mantenimiento se realizarán con Prometheus y Grafana, permitiendo supervisar métricas de rendimiento en tiempo real y detectar posibles fallos antes de que

afecten a los usuarios. Con este enfoque, el sistema de gestión de citas e inventario del consultorio médico se mantendrá escalable, seguro y adaptable a futuras necesidades del consultorio.

### **5.1.1 Automatización de Pipeline CI/CD**

El pipeline de CI/CD es una herramienta fundamental que automatiza el proceso de construcción, prueba y despliegue del sistema. Gracias a esta configuración, cada vez que se realice un cambio en el código, se verificará y se implementará automáticamente, reduciendo la necesidad de intervención manual y optimizando el flujo de trabajo.

#### **Configuración en GitHub Actions**

Se establecerá un pipeline en GitHub Actions que se activará cada vez que un desarrollador haga un commit en el repositorio, asegurando que todos los cambios se procesen sin demora y cumplan con los estándares de calidad del proyecto.

#### **Pasos del Pipeline**

En la fase de construcción, se prepara la aplicación para su lanzamiento, lo que implica compilar el código, instalar dependencias y configurar el entorno necesario para su ejecución. En la fase de pruebas, se ejecutarán pruebas automatizadas de unidad, integración y funcionales para garantizar que cada componente del sistema funcione correctamente. Se validará la autenticación con OAuth 2.0 y JWT, la gestión de citas y la administración del inventario. Finalmente, en la fase de despliegue, si todas las pruebas son exitosas, el pipeline lanzará automáticamente

la nueva versión del sistema en el entorno de prueba o producción, asegurando que los usuarios accedan a las mejoras sin interrupciones.

### **Beneficios del Despliegue Continuo**

La adopción del pipeline de CI/CD en este proyecto permitirá iteraciones rápidas, mejorando la calidad del software y reduciendo el tiempo de entrega de nuevas funcionalidades. Además, facilitará la detección temprana de errores, optimizando la experiencia del usuario y garantizando la disponibilidad del sistema en todo momento. El monitoreo y mantenimiento se realizarán con Prometheus y Grafana, permitiendo supervisar métricas de rendimiento en tiempo real y detectar posibles fallos antes de que afecten a los usuarios. Con este enfoque, el sistema de gestión de citas e inventario del consultorio médico se mantendrá escalable, seguro y adaptable a futuras necesidades del consultorio.

#### **5.1.2 Rollback Automático**

Se configurará un sistema dentro del proceso de despliegue que se activará de inmediato si algo falla durante la implementación o si surgen problemas importantes después de que la nueva versión esté en funcionamiento.

Usaremos GitLab CI/CD para el control de versiones, lo que nos permitirá guardar y etiquetar las diferentes versiones del código y los contenedores Docker. Así, si hay un error, el sistema podrá regresar sin problemas a la versión anterior que estaba funcionando bien. Además, es fundamental que todo el proceso de regreso a la versión anterior esté bien documentado, lo que garantizará que todos sepan



cómo actuar rápidamente si algo sale mal y que las configuraciones estén en línea con las políticas de recuperación del sistema.

### **5.1.3 Revisión y actualización periódica del Pipeline**

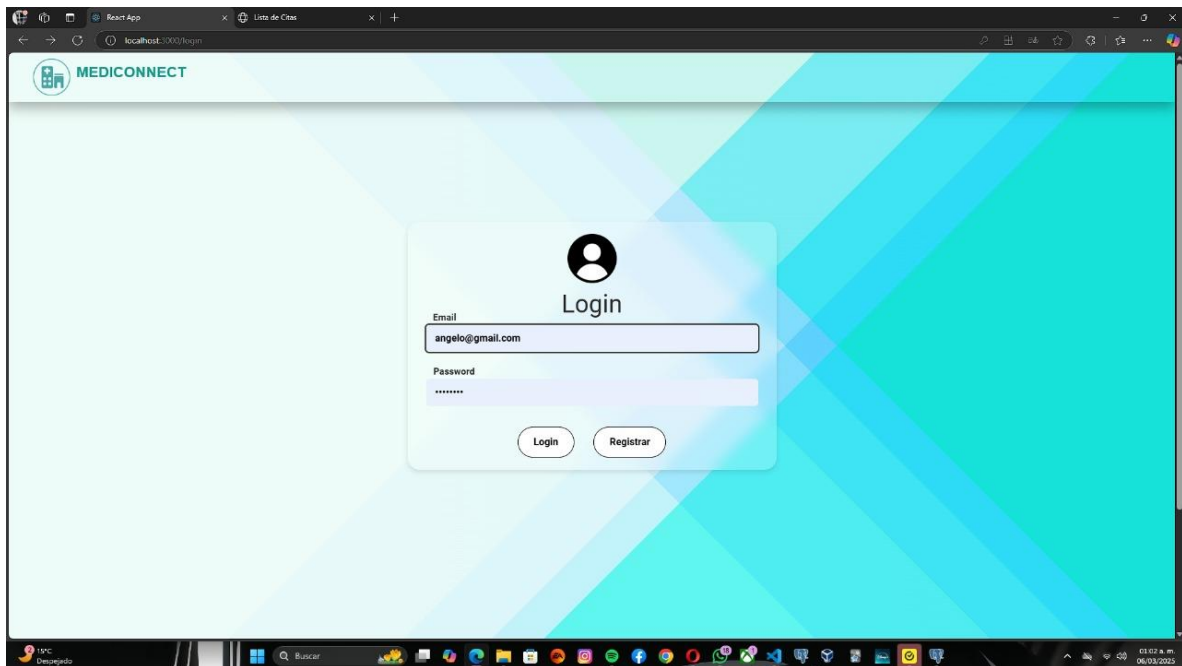
Proceso clave en la gestión de proyectos de software que se centra en evaluar y mejorar continuamente el flujo de trabajo de CI/CD (Integración Continua y Despliegue Continuo). La evaluación regular consiste en realizar revisiones periódicas del pipeline para analizar su desempeño y eficacia la recopilación de retroalimentación permite obtener comentarios de los miembros del equipo sobre su experiencia con el pipeline y las áreas que consideran que necesitan mejoras la implementación de cambios se basa en las evaluaciones y la retroalimentación realizando ajustes o actualizaciones en el pipeline como añadir nuevas etapas modificar las existentes o integrar nuevas herramientas la documentación se actualiza para reflejar cualquier cambio realizado asegurando que todos los miembros del equipo estén al tanto de los nuevos procesos la capacitación proporciona formación adicional al equipo si se introducen nuevas herramientas o procesos para asegurar que todos estén cómodos con las actualizaciones La Revisión y Actualización Periódica del Pipeline es esencial para mantener un proceso de desarrollo ágil y eficiente, asegurando que el equipo pueda adaptarse a nuevos desafíos y seguir entregando un software de alta calidad.

## 6.PRUEBAS

Las pruebas en un sistema son un proceso esencial que busca asegurar la calidad, funcionalidad y seguridad de una aplicación antes de su lanzamiento. Su propósito principal es detectar y corregir errores o inconsistencias en el software, garantizando que el producto final cumpla con las especificaciones requeridas y ofrezca una experiencia confiable a los usuarios. Las pruebas permiten validar la funcionalidad comprobar que todas las características y flujos de la aplicación como la creación y envío de encuestas operen de manera adecuada detectar y corregir errores tempranamente identificar problemas en las fases iniciales del desarrollo lo que reduce costos y tiempos de corrección fortalecer la seguridad evaluar y remediar posibles vulnerabilidades para proteger los datos de las encuestas y la información personal de los usuarios optimizar el rendimiento medir y ajustar el rendimiento de la aplicación asegurando tiempos de respuesta eficientes y una gestión adecuada de la carga garantizar la usabilidad verificar que la interfaz de usuario sea intuitiva y accesible en distintos dispositivos y navegadores mejorando así la experiencia del usuario Implementar estas pruebas siguiendo las mejores prácticas de desarrollo ágil y DevOps no solo asegura que el proyecto alcance sus objetivos, sino que también promueve un proceso de desarrollo continuo y automatizado. Esto permite realizar mejoras y actualizaciones en la aplicación de manera ágil y segura, brindando un producto de alta calidad a los usuarios.

### **6.1.1 Pruebas Unitarias**

Las pruebas unitarias son una técnica de verificación que se aplica a los componentes individuales del sistema para asegurarse de que cada módulo funcione correctamente de manera aislada. Estas pruebas se realizan a nivel de código y permiten detectar errores en las funciones o métodos antes de integrarlos con otros componentes. En el sistema, las pruebas unitarias se aplicaron principalmente al módulo de inicio de sesión para garantizar que los datos ingresados por el usuario sean procesados correctamente y se cumplan las reglas de autenticación. Las pruebas unitarias se realizaron para verificar el correcto funcionamiento del inicio de sesión, evaluando que el sistema permita el acceso únicamente a usuarios con credenciales válidas. Se comprobó que los campos de correo electrónico y contraseña acepten datos correctos y que se generen los mensajes de error adecuados en caso de ingresar información incorrecta. Además, se validó la redirección automática a la página principal al iniciar sesión exitosamente. Estas pruebas unitarias garantizan la calidad y fiabilidad del sistema antes de su integración con otros módulos, lo que ayuda a prevenir errores durante la ejecución. A continuación, se presenta la figura donde se observa la interfaz de inicio de sesión del sistema.

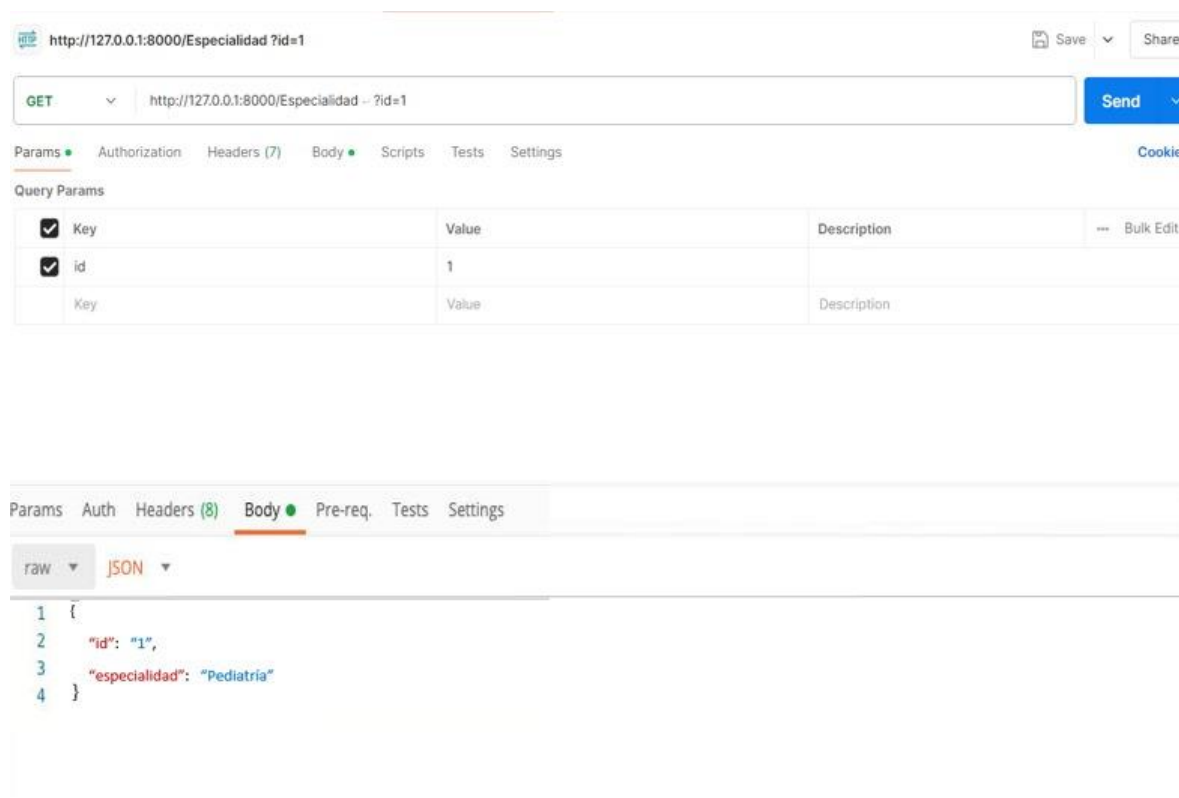


**Figura 28. Prueba unitaria**

### **6.1.2 Pruebas de integración**

Las pruebas de integración son un paso crítico en el proceso de desarrollo de software que se enfocan en verificar cómo interactúan diferentes componentes o módulos de una aplicación entre sí. El objetivo es asegurarse de que, al combinar distintos elementos del sistema, se mantenga la funcionalidad y se minimicen los errores. Este tipo de pruebas se realizó con la herramienta postman es especialmente importante para detectar problemas que no se pueden identificar en pruebas unitarias, donde cada componente se evalúa de manera aislada. Mejor calidad de software al garantizar que los componentes trabajen juntos sin problemas, las pruebas de integración contribuyen a la calidad general del software reducción de costos detectar y corregir problemas en fases tempranas del desarrollo

es generalmente menos costoso que solucionar fallos en producción facilitación del desarrollo ágil con pruebas de integración efectivas los equipos de desarrollo pueden realizar cambios en el código de manera más rápida y segura facilitando un enfoque ágil en el desarrollo de software.



**Figura 29. Pruebas de integración**

### 6.1.3 Pruebas funcionales

Las pruebas funcionales son un tipo de prueba de software que verifica que las funciones de una aplicación cumplan con los requisitos especificados. Estas pruebas se centran en la funcionalidad de la aplicación desde la perspectiva del usuario final. Aquí tienes una descripción detallada de cómo se realizan las pruebas funcionales, usando el ejemplo del sistema de gestión de citas de un consultorio

dental. Antes de comenzar con las pruebas, es crucial tener una comprensión clara de los requisitos funcionales de la aplicación.

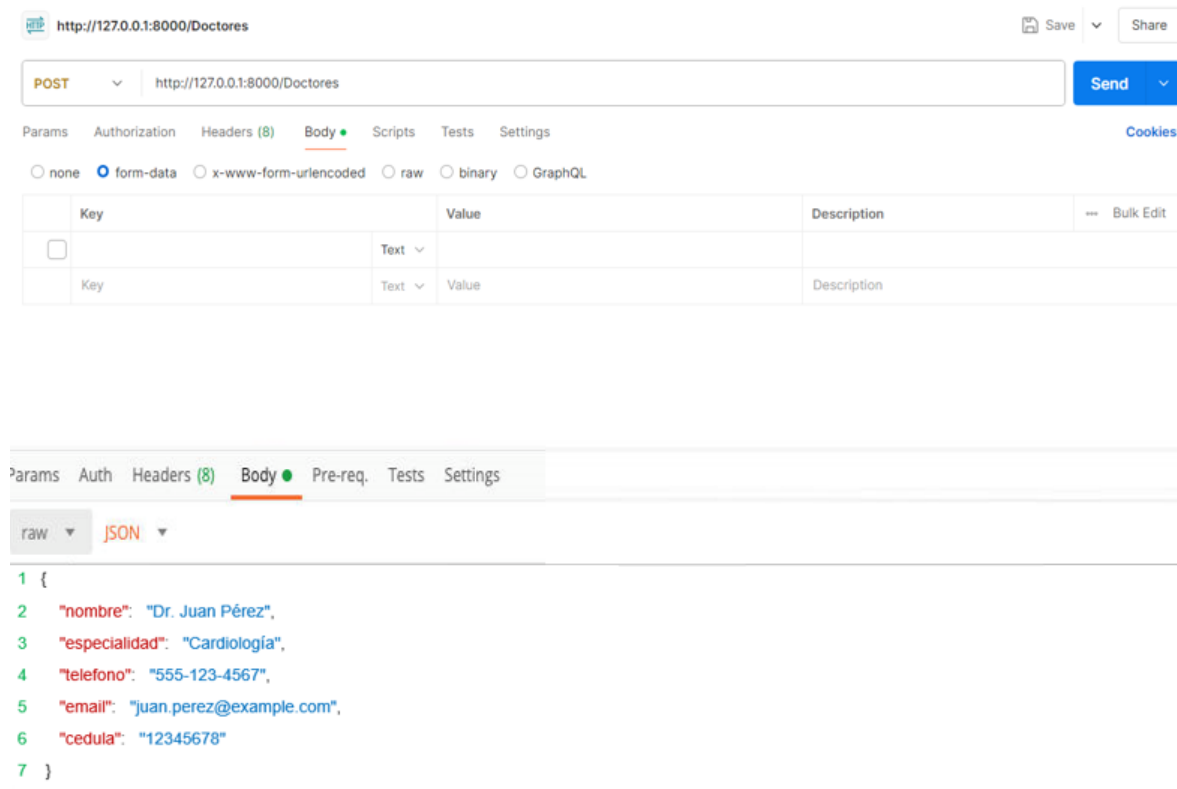
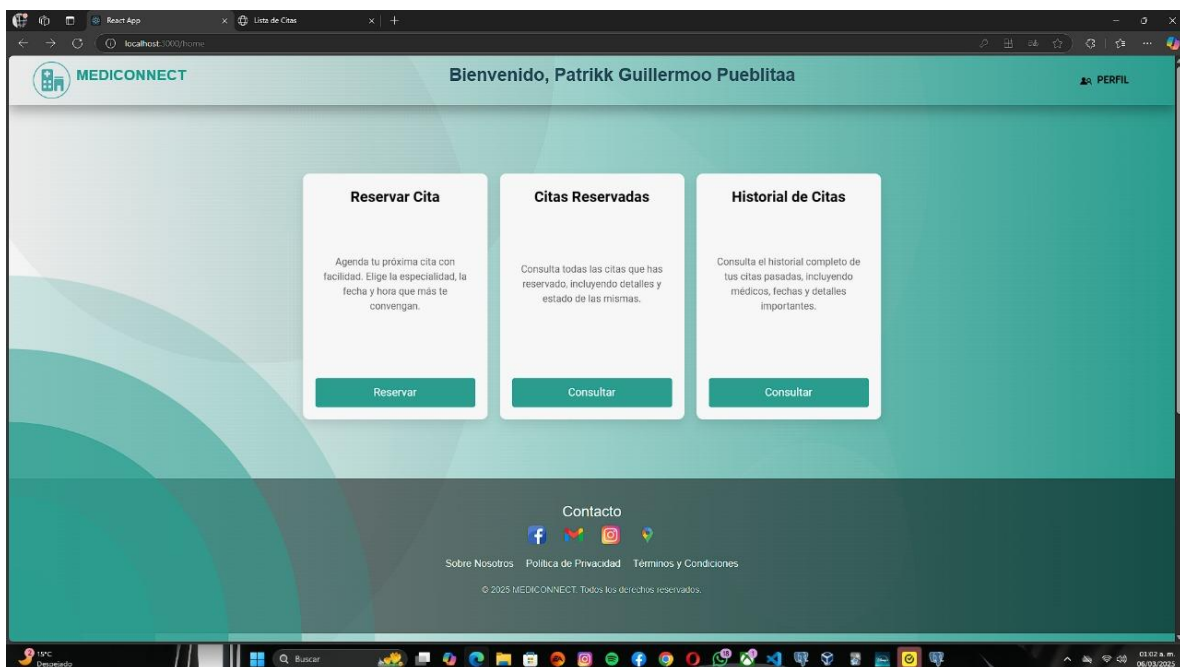


Figura 30. Pruebas funcionales

### 6.1.4 Pruebas de Interfaz de Usuario (UI)

El objetivo de esta prueba es validar que la interfaz gráfica del sistema sea intuitiva, fácil de usar y cumpla con los estándares de diseño establecidos. El procedimiento consiste en verificar que el diseño sea consistente en toda la página, comprobar que los botones, imágenes e íconos se visualicen correctamente, evaluar la disposición de los elementos como tarjetas, botones y encabezado, probar que los botones de navegación respondan al hacer clic y revisar que la página se adapte correctamente en diferentes tamaños de pantalla. El resultado esperado es que el diseño sea uniforme con la paleta de colores definida, los botones tengan etiquetas

claras y funcionales, los íconos e imágenes sean visibles sin distorsión y la navegación sea fluida entre las diferentes secciones. La evidencia se presenta con la captura de pantalla de la página principal con las tres opciones disponibles y capturas adicionales de la versión móvil si se realizó. El resultado obtenido es que la interfaz cumple con los estándares de diseño y es fácil de usar. A continuación, se le presenta la figura 28 donde se observa el diseño de la página principal, en la cual se muestran tres opciones disponibles para el usuario: Reservar Cita, Citas Reservadas e Historial de Citas. Cada tarjeta contiene una breve descripción de la funcionalidad que ofrece, acompañada de un botón que permite acceder a la sección correspondiente. Además, se aprecia la sección de contacto con enlaces a redes sociales y políticas de privacidad, garantizando una navegación intuitiva y agradable para el usuario.



**Figura 31. Pruebas de usuario**

## 7. OPERACIÓN

La fase de operación se centra en la gestión y supervisión de las aplicaciones y servicios en un entorno de producción, con el objetivo principal de asegurar que las aplicaciones funcionen de manera eficiente y confiable, brindando soporte continuo a los usuarios finales. En esta fase, se abordan varios aspectos clave para el buen funcionamiento del sistema y la satisfacción de los usuarios. El monitoreo y rendimiento es esencial, garantizando que las aplicaciones estén funcionando de manera óptima. Esto implica la supervisión de recursos, tiempos de respuesta y carga del sistema, asegurando que todo esté dentro de parámetros aceptables para un funcionamiento eficiente. Además, se lleva a cabo un mantenimiento proactivo para identificar y resolver problemas antes de que afecten a los usuarios. Esto incluye realizar actualizaciones periódicas, aplicar parches de seguridad y optimizar el rendimiento de las aplicaciones. La gestión de incidencias es otra área crucial, ya que se deben manejar y responder a cualquier incidente que pueda surgir, minimizando el tiempo de inactividad y asegurando que el servicio se mantenga disponible y funcional en todo momento. La automatización de procesos también juega un papel importante en esta fase, implementando herramientas y prácticas que automaticen tareas repetitivas, como despliegues y monitoreo, lo cual reduce los errores humanos y mejora la eficiencia. Finalmente, se asegura un feedback continuo, recopilando y analizando datos tanto de los usuarios como de los sistemas para mejorar constantemente el producto y la experiencia del usuario.



## 7.1 Monitoreo de la aplicación

El monitoreo de la aplicación es una parte esencial para garantizar un funcionamiento óptimo y eficiente, especialmente en un entorno crítico como un consultorio médico. Utilizar Kibana junto con Elasticsearch y Logstash (parte del stack ELK) permite visualizar y analizar datos en tiempo real, ayudando a identificar problemas y mejorar la experiencia del usuario. Para implementar el monitoreo de una aplicación en un consultorio médico, se debe configurar la aplicación para que genere logs de eventos importantes, como la creación, modificación y cancelación de citas médicas, el registro de pacientes y usuarios en el sistema, así como errores y excepciones en la aplicación, tales como problemas de conexión o fallos en el sistema. Es fundamental que los logs se generen en un formato que Logstash pueda interpretar fácilmente, como JSON o CSV, lo que facilitará su integración y análisis posterior, permitiendo un monitoreo eficiente y en tiempo real del sistema.

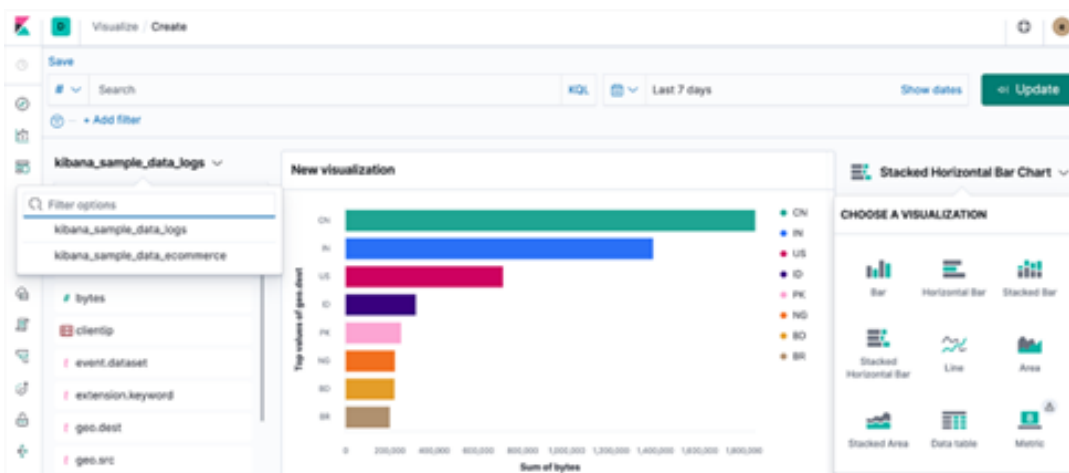


Figura 32. Monitoreo

### **7.1.2 Backup y Recuperación ante desastres**

Desplegar una aplicación web en AWS implica configurar una infraestructura que abarque varios componentes esenciales para asegurar su correcto funcionamiento y disponibilidad. Aquí se detallan los componentes necesarios:

#### **Servidor de aplicaciones**

Este es el componente que alberga el código de la aplicación web. En AWS, se utilizó EC2 (Elastic Compute Cloud) para crear y gestionar instancias de servidores virtuales. Las instancias EC2 permiten escalar el número de servidores según la demanda y configurar el entorno necesario para ejecutar la aplicación, como servidores web (Nginx, Apache) y entornos de ejecución específicos (Node.js, Java, etc.).

#### **Servidor de base de datos**

Para gestionar y almacenar los datos de la aplicación, AWS ofrece RDS (Relational Database Service), que permite configurar bases de datos relacionales como MySQL, PostgreSQL, o SQL Server. Al utilizar RDS, se obtiene una base de datos administrada que simplifica tareas como backups, replicación y actualizaciones.

#### **Balanceo de cargas**

El balanceo de cargas distribuye el tráfico entrante entre varias instancias de servidores para garantizar alta disponibilidad y fiabilidad. Elastic Load Balancing (ELB) en AWS ayuda a distribuir automáticamente el tráfico de aplicaciones entrantes a múltiples instancias EC2. Esto mejora la tolerancia a fallos y facilita la escalabilidad horizontal de la aplicación.

## **Configuración de dominio**

Para que en el sistema puedan acceder a través de un nombre de dominio, se utiliza Route 53, el servicio de DNS de AWS. Para gestionar la traducción de nombres de dominio a direcciones IP asociadas a recursos de AWS, como el ALB. Además, permite gestionar el enrutamiento y la disponibilidad de los servicios.

## **Certificados de seguridad**

Para asegurar la comunicación entre los usuarios y la aplicación, se implementan certificados SSL/TLS. AWS Certificate Manager (ACM) facilita la obtención, implementación y gestión de certificados SSL/TLS para aplicaciones ejecutadas en AWS. Los certificados se asocian con el ALB para habilitar conexiones seguras HTTPS, protegiendo los datos en tránsito.

Esta configuración asegura que la aplicación web sea escalable, segura y altamente disponible, utilizando los servicios gestionados que AWS ofrece. Para crear una instancia en AWS se puede realizar de esta manera:

1. Accede a la Consola de AWS
  - Abre tu navegador web y ve a [AWS Management Console] (<https://aws.amazon.com/console/>).
  - Inicia sesión con tu cuenta de AWS.
2. Navega a EC2
  - En la consola de AWS, busca y selecciona EC2.
3. Lanzar una nueva instancia
  - En el panel de EC2, haz clic en el botón Launch Instance.
4. Configurar la instancia

- Nombre y etiquetas de la instancia\*\*: Proporciona un nombre descriptivo para la instancia.

#### 5. Aplicaciones e imágenes (AMI)

- Selecciona una Amazon Machine Image (AMI). Por ejemplo, puedes elegir Amazon Linux 2 AMI o cualquier otra AMI que prefieras.

#### 6. Tipo de instancia

- Selecciona el tipo de instancia. Un tipo común para pruebas es `t2.micro`, que es elegible para el nivel gratuito.

#### 7. Configuración de la instancia

- Elige la opción predeterminada si no necesitas configuraciones específicas avanzadas.

#### 8. Opciones de compra (opcional)

- Configura las opciones de compra si necesitas instancias reservadas o spot.

#### 9. Configuración de red

- Selecciona la VPC y la subred donde deseas lanzar la instancia.
- Configura los grupos de seguridad para permitir tráfico específico. Por ejemplo, puedes agregar reglas para permitir SSH (puerto 22) y HTTP (puerto 80).

#### 10. Almacenamiento

- Configura el volumen de almacenamiento. Elige el tamaño y el tipo de almacenamiento, como EBS.

#### 11. Configuración de almacenamiento adicional

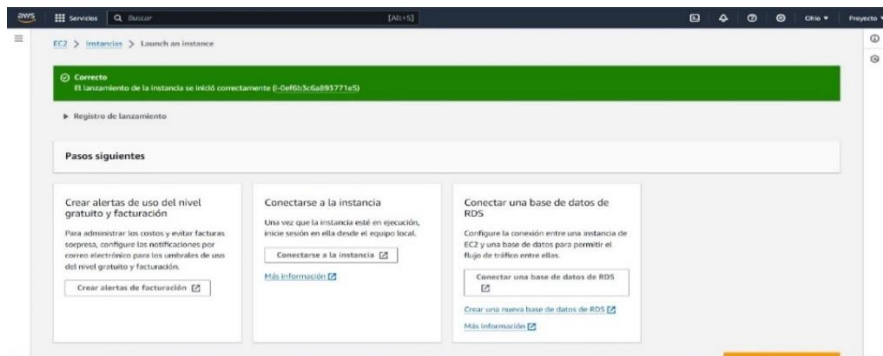
- Añade volúmenes adicionales si es necesario.

#### 12. Revisar y lanzar

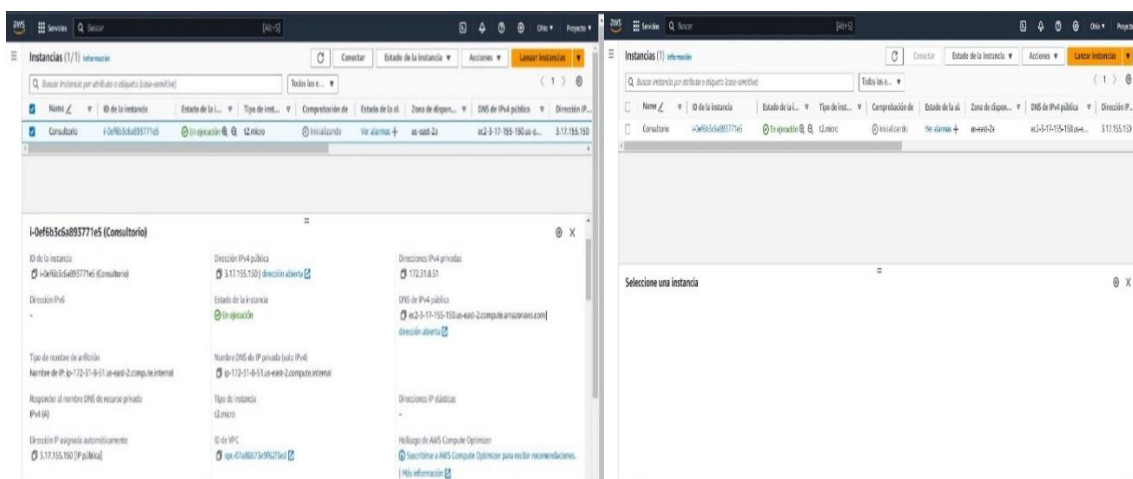
- Revisa todas las configuraciones para asegurarte de que sean correctas.
- Haz clic en Launch Instance

#### 13. Conectar a la instancia

- Una vez que la instancia esté en estado "running", selecciona la instancia en el panel de EC2.
- Haz clic en el botón y sigue las instrucciones para conectarte a la instancia mediante SSH.



A continuación, se anexa las evidencias, que detalla el flujo de trabajo para el despliegue de una instancia EC2 en AWS, diseñado para gestionar de manera eficiente la infraestructura en la nube. Esta configuración no solo facilita la administración del entorno de servidor, sino que también garantiza que la instancia esté configurada y operativa de forma coherente y sin interrupciones. Emplea prácticas como la selección adecuada de para el sistema operativo, el ajuste de tipos de instancia para satisfacer demandas específicas y la configuración de redes y almacenamiento para asegurar el rendimiento y la disponibilidad.



### **7.1.3 Optimización Continua del rendimiento**

Utilizando Kibana, el administrador puede visualizar el tiempo de espera promedio en tiempo real y descubrir que, durante ciertos días y horas, el tiempo de espera se incrementa considerablemente. Con esta información, pueden ajustar la programación de citas, agregar más personal durante los períodos más concurridos, o implementar un sistema de gestión de filas. Al implementar la optimización continua del rendimiento utilizando Kibana en un consultorio dental, se pueden mejorar tanto la experiencia del paciente como la eficiencia operativa. Este enfoque proactivo no solo asegura un servicio de calidad, sino que también permite identificar y abordar problemas antes de que afecten la satisfacción del paciente.

## CONCLUSIÓN

El desarrollo e implementación de MEDICONNECT para la gestión de citas médicas en un consultorio, basada en prácticas de DevOps, ha permitido la creación de un sistema eficiente, seguro y escalable. La adopción de DevOps en este proyecto ha sido clave para mejorar la calidad del software y acelerar la implementación de nuevas funcionalidades. A través de la integración y entrega continua (CI/CD), se logró automatizar los procesos de despliegue, reduciendo errores y asegurando que cada versión del software pasara por pruebas rigurosas antes de su lanzamiento en producción. Desde la perspectiva del usuario final, la implementación de esta aplicación ha generado múltiples beneficios para el consultorio médico. Los pacientes pueden programar citas de manera más rápida y sencilla mediante una interfaz intuitiva, lo que reduce los tiempos de espera y optimiza la gestión de la agenda médica. Gracias a la disponibilidad en la nube, el acceso a la información se mantiene seguro y eficiente desde cualquier dispositivo autorizado, lo que permite una operatividad continua y sin interrupciones. La combinación de un desarrollo web eficiente con metodologías DevOps ha resultado en una solución robusta, confiable y escalable para la gestión de citas médicas. Además de optimizar la administración de citas, la aplicación facilita la comunicación entre médicos y pacientes, mejora la toma de decisiones basada en datos y garantiza la continuidad operativa del consultorio a largo plazo con futuras actualizaciones para brindar un servicio más completo y eficiente.