ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA DE GESTIÓN DE CITAS MÉDICAS PARA CONSULTORIO ODONTOARIAS. DESARROLLO DE UN BACKEND

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

JHON HENRY TORRES CARRERA

DIRECTOR: IVONNE FERNANDA MALDONADO SOLIZ

DMQ, agosto 2023

CERTIFICACIONES

Yo, Jhon Henry Torres Carrera declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JHON HENRY TORRES CARRERA

jhon.torres01@epn.edu.ec

jhon-3316@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Jhon Henry Torres Carrera, bajo mi supervisión.

IVONNE FERNANDA MALDONADO SOLIZ

DIRECTOR

ivonne.maldonadof@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JHON HENRY TORRES CARRERA

DEDICATORIA

Es un honor para mí presentar este Trabajo de Integración Curricular, un proyecto que refleja tiempo y esfuerzo, colaboración con compañeras y la pasión por el aprendizaje.

Dedico este trabajo a todas esas personas que forman parte importante en mi vida, quiero recordarles que valoro su tiempo y apoyo. Una mención especial a mi madre, a todas mis hermanas y a mi novia, quienes se dieron el tiempo para darme su aliento e impulso ante el proceso del proyecto.

Jhon Torres

AGRADECIMIENTO

Quiero expresar un agradecimiento a mis familiares, pareja, amigos y profesores que se hicieron presentes durante mi evolución al cursar la carrera.

Un agradecimiento especial a mi madre y Marcelo Ramírez que siempre han sido una guía y me han enriquecido de valores y consejos, personas incondicionales al apoyarme en cualquier meta de mi vida. A todas mis hermanas quienes de ellas nunca falta el interés sobre las evoluciones importantes en mi vida incluyendo a mis primos cercanos. A mi novia que tuvo el tiempo para darme su aliento y ánimos durante la evolución del proyecto. A mis compañeras que tuvieron la capacidad de colaboración a lo largo del proceso.

Agradecimientos a los profesores, Ing. Ivonne Maldonado, Ing. Byron Loarte e Ing. Juan P. Zaldumbide, quienes han tenido la dedicación y la guía oportuna para mi crecimiento académico, de igual manera por la ayuda brindada durante el proceso del Trabajo de Integración Curricular.

Soy muy agradecido de formar parte de la institución Escuela Politécnica Nacional, y de contar con personas increíbles en mi vida.

Con gratitud y orgullo.

Jhon Torres

ÍNDICE DE CONTENIDO

RESUMEN

Para un negocio, en estos tiempos, es esencial contar con presencia tecnológica en el mercado digital, más aún hablando de un sistema que permita automatizar procesos en el negocio.

OdontoArias es un consultorio que ha venido realizando gestión de agendamiento de citas y de historias clínicas de manera manual, por lo que contar con un sistema le permite un mejor enfoque, un plus frente al mercado a su alrededor y sobre todo una mejor atención de sus pacientes. En general, la presencia en el mundo digital le ayuda impulsando el negocio y el crecimiento potencial, además permite que el paciente tenga mayor facilidad y este más involucrado en el proceso de agendamiento de citas y de historias clínicas.

El desarrollo de un *backend* dedicado responde a la necesidad de automatizar y optimizar procesos específicos. Al contar con *endpoints* concretos, se puede gestionar de manera más eficiente y efectiva citas médicas e historias clínicas. Permitiendo que el personal del consultorio se concentre más en brindar una atención de calidad a los pacientes que en el proceso en si de gestión, lo que se traduce en una mayor satisfacción de los pacientes y una reputación positiva para el negocio.

El presente desarrollo del componente *backend*, cuenta con *endpoints* para la automatización del proceso de agendamiento de citas y de historias clínicas en el consultorio OdontoArias. La construcción del *backend* ha seguido la guía de la metodología *Scrum*, permitiendo el cumplimiento de los objetivos en los tiempos establecidos, adaptándose rápidamente a los cambios, además de asegurar la calidad a lo largo del desarrollo.

Este documento se encuentra dividido en 5 secciones: la sección 1 corresponde a la descripción del componente (objetivos y alcance) y el marco teórico, la sección 2 presenta la metodología (roles y artefactos), herramientas y patrón arquitectónico utilizado en el desarrollo del *backend*, la sección 3 muestra los resultados por cada sp*rint* y las secciones 4 y 5 listan una serie de conclusiones y recomendación encontradas durante el desarrollo de este trabajo de titulación.

PALABRAS CLAVE: *Backend*, Consultorio odontológico, *endpoints*, *Scrum*.

ABSTRACT

For a business in these times, having a technological presence in the digital market is essential, especially when talking about a system that enables process automation within the business.

OdontoArias is a dental office that has been managing appointment scheduling and medical records manually. Having a system in place allows them a better focus, a competitive edge in the market, and above all, improved patient care. In general, having a digital presence helps drive the business and its potential growth. It also enables patients to have greater ease and involvement in the appointment scheduling and medical records process.

The development of a dedicated backend responds to the need for automating and optimizing specific processes. By having specific endpoints, medical appointments and medical records can be managed more efficiently and effectively. This allows the dental office staff to concentrate more on providing quality patient care rather than the management process itself, resulting in higher patient satisfaction and a positive reputation for the business.

The current development of the backend component includes endpoints for automating the appointment scheduling and medical records process at OdontoArias dental office. The construction of the backend has followed the Scrum methodology guide, enabling the achievement of objectives within established timelines, quick adaptation to changes, and ensuring quality throughout the development.

This document is divided into 5 sections: Section 1 corresponds to the component description (objectives and scope) and the theoretical framework. Section 2 presents the methodology (roles and artifacts), tools, and architectural pattern used in backend development. Section 3 showcases the results for each sprint, and Sections 4 and 5 provide a series of conclusions and recommendations found during the development of this thesis work.

KEYWORDS: *Backend,* Dental clinic, *endpoints,* Scrum.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Actualmente el consultorio odontológico OdontoArias enfrenta diversos desafíos en la gestión de sus servicios y atención a los pacientes, esto debido a que las citas médicas, agendamiento de citas y disponibilidad (horario) de odontólogos se maneja con registros manuales, es decir esta información únicamente se encuentra en hojas de cálculo (Excel) y en ocasiones en hojas de papel físicas, ocasionando problemas de organización, resultado en largos tiempos de espera para los pacientes, un aprovechamiento subóptimo de los recursos disponibles en el consultorio y hasta en la pérdida de información.

El solucionar el problema mencionado y automatizar los procesos del consultorio permite una mejor eficiencia para evitar generar malestar e inconformidad en el personal del consultorio y en pacientes; obteniendo resultados positivos al no afectar la imagen y reputación del centro odontológico en el ámbito de gestión de procesos y atención eficiente, fomentando una mayor productividad y satisfacción al tener acceso rápido y preciso a la información relevante.

La solución propuesta contempla el diseño e implementación de API's seguras y eficientes que permitan la comunicación entre los componentes *backend*, *frontend* y aplicación móvil. Añadiendo al sistema algoritmos de asignación de citas que optimicen la disponibilidad y el tiempo de los odontólogos, sin perder el enfoque de los criterios o requerimientos establecidos por el consultorio. Por ello el objetivo del presente proyecto de integración curricular es desarrollar el *backend* para la gestión de citas odontológicas en el consultorio OdontoArias. Dicho desarrollo cuenta con la implementación de una gestión de seguridad para el manejo de los datos en todos los procesos requeridos para el buen funcionamiento del *backend*. El consumo de los servicios del *backend* mantiene los datos seguros y confiables, datos a los que se puede acceder únicamente por los usuarios autorizados esto mediante la asignación de roles con la finalidad de protegerlos en la recopilación, almacenamiento y uso dentro del componente *backend* que se presentan por parte del *frontend* al usuario.

En resumen, el *backend* para la gestión de citas odontológicas en el consultorio OdontoArias permite la planificación y asignación eficaz de citas médicas, respaldado en el uso de herramientas tecnológicas, ayudando al consultorio a mejorar la gestión de estos procesos, reduciendo los tiempos de espera y optimizando el aprovechamiento de los recursos disponibles; manejando y manteniendo la lógica del negocio, resultando en una mayor satisfacción de los pacientes, una mayor productividad del personal y un consultorio más eficiente.

1.1 Objetivo general

Desarrollar el *backend* para la gestión de citas odontológicas en el consultorio OdontoArias.

1.2 Objetivos específicos

1. Identificar los requerimientos para el *backend*.
2. Diseñar la base de datos relacional para el componente *backend*.
3. Codificar los diferentes *endpoints* para el componente *backend*.
4. Validar la funcionalidad de los *endpoints*.
5. Desplegar los *endpoints.*

1.3 Alcance

El alcance del proyecto se centra en el desarrollo del *backend* para el sistema de gestión del consultorio OdontoArias. El objetivo principal es mejorar la eficiencia y la calidad de los servicios a través de la automatización y optimización de los procesos relacionados con la gestión de citas médicas e historias clínicas; en otras palabras, proporcionar los servicios que se requiera para lograr un fácil agendamiento de citas y gestionar las historias clínicas de pacientes de manera eficiente, por medio de la gestión y autenticación de usuarios con sus distintos roles.

La interacción está basada en peticiones API REST, contando con diferentes CRUD (Crear, Leer, Actualizar y Borrar) dependiendo de las restricciones del rol autenticado.

Roles del componente *backend*:

- Paciente.
- Odontólogo.
- Administrador.

Roles de la aplicación móvil:

- Paciente.
- Odontólogo.

El rol Paciente puede:

- Registrarse.
- Autenticarse.
- Reestablecer contraseña.
- Agendar horarios de cita.
- Visualizar la información del odontólogo (quien le atendará).
- Visualizar su historia clínica propia.
- Cancelar cita médica.
- Recibir recordatorios, a su dispositivo móvil, de las citas que agendo.

El rol Odontólogo puede:

- Autenticarse.
- Reestablecer contraseña.
- Registrar horario de disponibilidad de atención de citas.
- Actualizar fecha y hora de la cita odontológica.
- Crear, leer y actualizar historias clínicas de los pacientes.

El rol Administrador puede:

- Autenticarse.
- Reestablecer contraseña.
- Crear, leer, actualizar, deshabilitar odontólogos.
- Crear, leer, actualizar, deshabilitar pacientes.
- Visualizar la información de las citas creadas.
- Agendar horarios de cita (para un paciente que visite el consultorio).
- Visualizar las historias clínicas de todos los pacientes del consultorio.
- Creas, leer, actualizar y eliminar servicios.

1.4 Marco Teórico

Sistema Web

Un sistema web es una aplicación o plataforma que se ejecuta en un navegador web y permite a los usuarios interactuar y acceder a servicios, información y funcionalidades a través de Internet. Está compuesto por una combinación de tecnologías, que permiten la creación de interfaces de usuario interactivas y la comunicación con servidores para procesar y almacenar datos [1].

Backend

El *backend*, también conocido como el lado del servidor, es la parte de un sistema informático que se encarga de procesar y gestionar la lógica de negocio, la persistencia de datos y la comunicación con otros sistemas o componentes. En el contexto de una aplicación, el *backend* se encarga de recibir las solicitudes del cliente, procesarlas, interactuar con la base de datos y devolver la respuesta adecuada al cliente.

El *backend* se compone de diferentes elementos, como servidores, *frameworks*, bibliotecas y bases de datos, que trabajan juntos para proporcionar funcionalidades, seguridad y rendimiento al sistema. En general, el *backend* es responsable de manejar la lógica del sistema, la autenticación de usuarios, la gestión de datos, la ejecución de algoritmos y cualquier otro procesamiento necesario para que la aplicación funcione correctamente [2].

Arquitectura MVC

Es un patrón de diseño de software que divide una aplicación en tres componentes principales: el modelo, la vista y el controlador. El modelo se encarga de la lógica y el manejo de los datos, la vista se encarga de la presentación visual y el controlador actúa como intermediario que gestiona la interacción entre el modelo y la vista [3].

Base de datos relacional

Es un tipo de base de datos que organiza los datos en tablas relacionadas entre sí mediante claves primarias y claves externas. Utiliza un lenguaje de consulta estructurado (SQL) para realizar consultas y manipulaciones de datos. Las bases de datos relacionales son conocidas por su estructura predefinida y la capacidad de establecer relaciones entre tablas [4].

Alwaysdata

Es un proveedor de alojamiento *web* y servicios en la nube que ofrece opciones para alojar aplicaciones y sitios *web*. Proporciona servidores compartidos y dedicados, bases de datos, almacenamiento y otros recursos para el alojamiento de aplicaciones y sitios *web* [5].

API RESTful

Es un estilo de arquitectura de servicios *web* que sigue los principios y convenciones del protocolo HTTP. Una API RESTful expone recursos a través de URLs (*Uniform Resource Locators*) y utiliza los métodos HTTP (*GET, POST, PUT, DELETE*, etc.) para acceder y manipular esos recursos. Permite una comunicación eficiente y estructurada entre aplicaciones o servicios [6].

Scrum

Es un marco de trabajo ágil para la gestión y desarrollo de proyectos. Se basa en la colaboración, la adaptabilidad y la entrega incremental. *Scrum* divide el trabajo en iteraciones llamadas "*sprints*" y utiliza roles definidos (*Product Owner, Scrum Master* y *Development Team*) y artefactos (Historias de Usuario, *Product Backlog* y *Sprint Backlog*) para facilitar la comunicación y el seguimiento del progreso del proyecto [7].

Postman

Es una herramienta de colaboración y desarrollo de API. Permite a los desarrolladores diseñar, probar y documentar API’s de manera eficiente. Postman ofrece una interfaz gráfica fácil de usar para enviar solicitudes HTTP a diferentes puntos finales, ver las respuestas, crear y guardar colecciones de solicitudes, y automatizar pruebas y tareas relacionadas con API’s [8].

Laravel

Es un *framework* de desarrollo web basado en PHP. Proporciona una estructura y un conjunto de componentes predefinidos que facilitan el desarrollo rápido y eficiente de aplicaciones *web* [9]. Laravel incluye características como enrutamiento, manejo de bases de datos, plantillas de vistas, autenticación de usuarios y mucho más [10].

Eloquent (ORM)

Incluido en Laravel que simplifica la interacción con la base de datos. Proporciona modelos para representar las tablas y establecer relaciones entre ellos. Con una sintaxis orientada

a objetos, permite realizar consultas de manera concisa y expresiva. Además, las migraciones ayudan a mantener la estructura de la base de datos actualizada. En resumen, Eloquent es una herramienta poderosa que facilita el trabajo con la base de datos en Laravel, mejorando la legibilidad y la eficiencia del código [11].

Railway

Es una plataforma de alojamiento *web* que facilita la implementación y el despliegue de aplicaciones *web* y proyectos en línea [12].

Railway permite a los desarrolladores hospedar sus aplicaciones *web* de manera sencilla, ofreciendo una variedad de características para agilizar el proceso de implementación. Esto puede incluir integración con sistemas de control de versiones, *GitHub*.

Apache Bench

Es una herramienta de línea de comandos que viene con el servidor web Apache HTTP Server. Se emplea para medir y valorar el rendimiento de un servidor *web* o aplicación *web* al simular múltiples solicitudes concurrentes [13].

La herramienta realiza pruebas de carga al enviar solicitudes HTTP a un servidor y evaluar sus respuestas. Es útil para evaluar la capacidad de un servidor ante cierta carga de trabajo y detectar posibles problemas de rendimiento o cuellos de botella en una aplicación web.

2 METODOLOGÍA

El estudio de casos es una metodología que se aplica mayormente en fenómenos sociales y educativos. Se basa en el estudio particular y complejo de un caso singular para entender su actividad en circunstancias importantes [14], permitiendo un estudio más profundo de una situación en particular.

El sujeto de estudio puede ser un fenómeno, una persona, un evento o algo más concreto como es el caso del presente trabajo de titulación en el cual se realiza el análisis particular del consultorio odontológico OdontoArias.

2.1 Metodología de Desarrollo

Las metodologías ágiles son enfoques de desarrollo de *software* que se centran en la flexibilidad, la colaboración y la adaptabilidad en lugar de los procesos y la planificación rígida. Estas metodologías valoran la entrega temprana y continua de *software* funcional, así como la capacidad de responder rápidamente a los cambios y retroalimentación del cliente.

Scrum es un marco de trabajo ágil que se enfoca en la entrega iterativa e incremental de un producto, promoviendo la colaboración, la adaptabilidad y la respuesta rápida a los cambios. Lo que permite esta metodología es que todos los miembros implicados en el proyecto conozcan lo que va sucediendo con el proyecto, de manera global se conoce el avance. Permite la inspección del progreso para tomar decisiones para detectar posibles cambios y evitar problemas graves durante el desarrollo. Además, logra una mejor adaptación del equipo de trabajo que puede estar expuesto a los escenarios cambiantes en la tecnología [15].

Roles

En el marco de trabajo Scrum, se definen tres roles principales que desempeñan funciones clave en el proyecto.

Product Owner

Es el representante del cliente o del área de negocio. Encargado de definir y priorizar los requisitos y funcionalidades del sistema en el *Product Backlog*; debe mantener una comunicación constante para comprender sus necesidades y expectativas. Toma decisiones sobre qué funcionalidades se deben desarrollar en cada *Sprint*, basándose en el valor y la importancia para el cliente, además establece criterios de aceptación claros para cada funcionalidad [15].

Scrum Master

Es un facilitador del proceso dentro de la metodología Scrum y tiene un papel clave en asegurar que se sigan los principios y prácticas de Scrum. Entre sus responsabilidades están: el garantizar que el equipo de desarrollo entienda y aplique correctamente los principios y prácticas de Scrum, eliminar obstáculos y problemas que puedan afectar el progreso del equipo, facilitar las reuniones de Scrum, promover un ambiente de trabajo colaborativo y de mejora continua, y ayudar al equipo a definir y alcanzar sus objetivos, fomentando la autonomía y la autogestión [15].

Development Team

El equipo de desarrollo está compuesto por profesionales multidisciplinarios encargados de implementar las funcionalidades del sistema. Trabajan de forma colaborativa para planificar y realizar el trabajo necesario para entregar las funcionalidades definidas en el *Sprint*. El *Development Team* es auto organizado y toma decisiones sobre cómo llevar a cabo las tareas asignadas, participar activamente en las reuniones diarias de Scrum, aclara los requisitos y valida las funcionalidades desarrolladas [15].

A continuación, en la **Tabla 2.1** se muestra la disposición del equipo dentro del desarrollo del sistema con la determinación de los roles para la aplicación de la metodología *Scrum*.

Tabla 2.1 Roles designados

ROLES	NOMBRES
Product Owner	Dra. Lisseth Sánchez
Scrum Master	Ing. Ivonne Maldonado
Development Team	Sr. Jhon Torres

Artefactos

Los artefactos son útiles ya que ayudan a mantener un seguimiento claro de los requisitos, prioridades y progreso del proyecto, facilitando la colaboración y la toma de decisiones informadas por parte del equipo y los interesados [16].

Recopilación de Requerimientos

La recopilación de requerimientos en Scrum se basa en la comunicación constante y la colaboración entre el *Product Owner* y el *Development Team*. Es un proceso iterativo y colaborativo que se enfoca en identificar las necesidades clave del cliente y transformarlas

en elementos del *Product Backlog*, que sirven como guía para el desarrollo de cada *Sprint* [17].

Es importante destacar que en Scrum se promueve un enfoque iterativo e incremental en la recopilación de requerimientos. Esto significa que los requerimientos pueden evolucionar a lo largo del proyecto a medida que se obtiene retroalimentación del cliente y se adquiere mayor comprensión del dominio. Se priorizan las funcionalidades más valiosas y se trabaja en ellas en cada iteración, lo que permite una entrega temprana y continua de valor al cliente.

En el **ANEXO II-Recopilación de requerimientos** se encuentra la lista de requerimientos recopilados.

Historias de Usuario

Este artefacto permite capturar los requisitos y necesidades del usuario, los cuales describen una funcionalidad específica del sistema.

La **Tabla 2.2** es un ejemplo de Historia de Usuario, misma que se utiliza para representar las funcionalidades requeridas del sistema. En el **ANEXO II-Historias de Usuario** se encuentran todas las Historias de Usuario.

Tabla 2.2 Formato de Historias de Usuario – HUB002

HISTORIA DE USUARIO	
Identificador (ID): HUB002	**Usuario:** Administrador, paciente y odontólogo
Nombre Historia: *Endpoint* de Autenticación, Cierre de sesión y Reestablecer contraseña.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Jhon Torres	
Descripción: El usuario proporciona las credenciales de correo junto a su contraseña para poder autenticarse en el sistema y cerrar sesión. El usuario que olvide sus credenciales puede cambiar su contraseña mediante un enlace que se le enviara al correo ingresado por interfaz de usuario.	
Observación: El usuario debe estar registrado para poder autenticarse. Para cambiar de contraseña el usuario con correo ingresado debe existir en la base de datos.	

Product Backlog

Es un artefacto de Scrum que permite gestionar los requisitos y funcionalidades que se deben desarrollar en un proyecto. Prioriza el valor de los elementos en relación con las

necesidades del negocio [16]. En el **ANEXO II-*Product Backlog*** se encuentra la tabla completa del *Product Backlog.*

Sprint Backlog

Artefacto utilizado en Scrum para organizar y planificar el trabajo durante un *Sprint* determinado, es una lista de tareas seleccionadas del *Product Backlog* para ser abordadas durante un *Sprint* en específico [16] que ayuda a mantener el enfoque en la entrega de los elementos. En el **ANEXO II-*Sprint Backlog*** se encuentra la tabla completa del *Sprint Backlog.*

2.2 Diseño de la arquitectura

El diseño de la arquitectura de un software se refiere a la estructura y organización general del sistema, que incluye la distribución de componentes, la comunicación entre ellos y la forma en que se gestionan los datos. Un diseño de arquitectura bien pensado y robusto proporciona una serie de beneficios tales como, garantizar la escalabilidad del software, promover la calidad del sistema al permitir el impacto directo a favor de satisfacer las necesidades [18]. En general, el diseño de la arquitectura de un software bien elaborado es esencial para desarrollar sistemas confiables, flexibles y de alta calidad.

Arquitectura de Datos

La arquitectura de datos no es más que describir cómo serán los datos gestionados (recopilación, transformación, distribución y consumo) dentro de un sistema. Su principal valor radica en entender de manera clara el plan y la forma en que fluyen los datos a través del sistema de almacenamiento de datos (BDD). El buen diseño de la arquitectura de datos está basado en los requisitos del negocio, lo que ayuda a definir el modelo de datos y estructuras de datos [19].

La **Figura 2.1** presenta el Modelado de la Base de Datos del componente *backend*, es decir se puede observar el modelado de las entidades y relaciones implementados para la gestión de datos que almacene y procese el sistema.

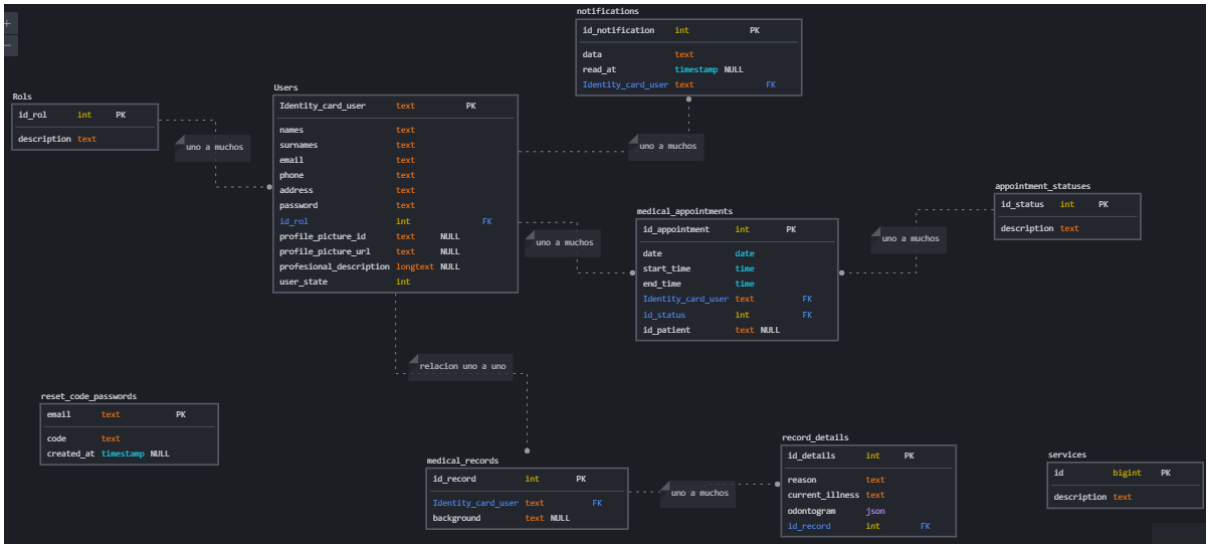

Figura 2.1 Modelado de BDD

Patrón arquitectónico

El patrón arquitectónico, es un patrón de diseño de software que divide una aplicación en tres componentes principales: modelo, la vista y el controlador [3]:

- El modelo se encarga del manejo y actualización de los datos,
- La vista se encarga de la presentación visual y,
- El controlador actúa como intermediario que gestiona la interacción entre el modelo y la vista, componente que contiene la lógica.

En la **Figura 2.2** se muestra el patrón arquitectónico implementado en el desarrollo *backend*, además las herramientas aplicadas para el funcionamiento del componente.

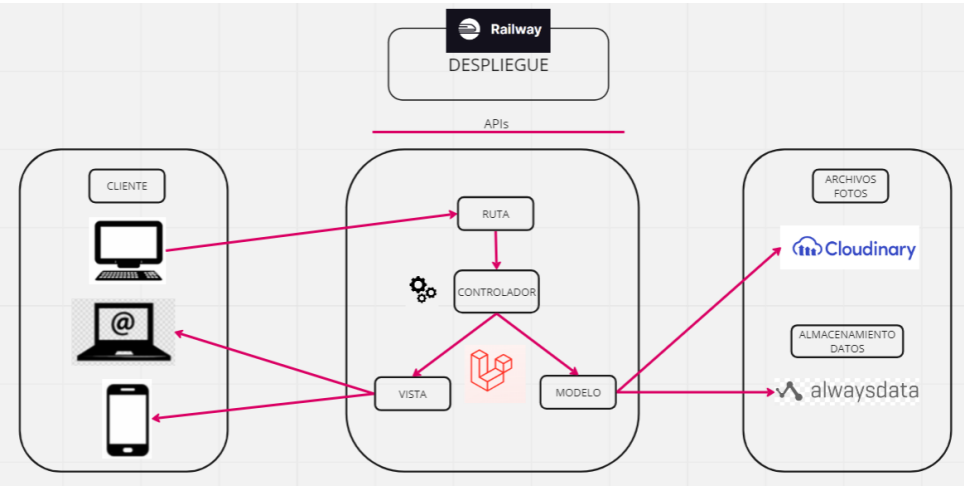

Figura 2.2 Modelo Arquitectónico - *Backend*

2.3 Herramientas de desarrollo

Ya especificado el patrón arquitectónico, en esta sección se presenta las herramientas aplicadas en el desarrollo de los *endpoints,* aprovechando al máximo el potencial de las tecnologías existentes para lograr la eficiencia y productividad del *backend*. La **Tabla 2.3** presenta el conjunto de herramientas que han aportado significativamente en el desarrollo del *backend*.

Tabla 2.3 Herramientas para el desarrollo - *Backend*

Herramienta	**Justificación**
Visual Studio Code	Es un entorno de desarrollo integrado amigable con los desarrolladores para el desarrollo de código. Usado para la edición y creación de la lógica del *backend* del sistema.
Git / GitHub	Son herramientas que permiten llevar un control de las versiones y cambio que va sufriendo el código a lo largo de su desarrollo. Usado para versionar y tener un control del código del *backend* del sistema.
Laravel	*Framework* con lenguaje principal PHP, proporciona una estructura MVC y un conjunto de componentes predefinidos, lo que ha facilitado el desarrollo rápido y eficiente del *backend* del sistema.
AlwaysData	Servicio en la nube que ha permitido desplegar la base de datos, *phpMyAdmin,* facilitando el acceso sin la necesidad de tenerla localmente.
Cloudinary	Es una herramienta en línea con características y funcionalidades específicas para almacenar, manipular y entregar contenido multimedia. Usado para la gestión del contenido multimedia de imágenes perfil de usuarios.
Railway	Es una herramienta que simplifica el despliegue de aplicaciones web, permite el alojamiento en la nube. Usado para el despliegue y porque automatiza el despliegue constante con la ayuda de *GitHub*.
Apache Bench	Es una herramienta que permite realizar pruebas de carga al servidor de Protocolo de transferencia de hipertexto (HTTP). De fácil uso mediante línea de comandos. Esta herramienta ha permitido la realización de pruebas de carga de manera muy sencilla.

3 RESULTADOS

A continuación, en este apartado se presenta los resultados del desarrollo del componente *backend.* Siguiendo lo planificado en el **ANEXO II-*Sprint Backlog***.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo

Para este *sprint* se tiene los siguientes resultados:

- Definición de roles
- Recopilación de requerimientos
- Diseño de la Base de Datos

Definición de roles

En la **Figura 3.1** se muestra los usuarios con roles establecidos que usen el componente *backend*, y el acceso a las funcionalidades de acuerdo con el rol establecido.

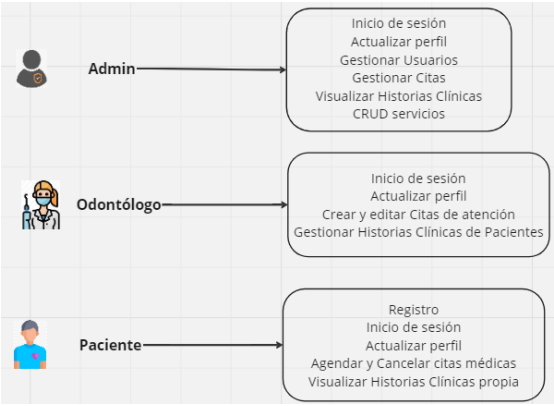

Figura 3.1 Roles de Usuarios

Recopilación de requerimientos

***Endpoint* para el registro de usuario**

Este *endpoint* permite registrar a un usuario con rol paciente, además se crea una historia clínica.

***Endpoints* de autenticación, cierre de sesión y restablecimiento de contraseña**

El *endpoint* de autenticación, permite validar los datos de email y contraseña de un usuario ya registrado además de otorgarle al usuario un *token Bearer*.

El *endpoint* de cierre de sesión es para borrar el *token Bearer* otorgado al usuario una vez que inicio sesión.

En el apartado de restablecimiento contraseña, existen *endpoints* que permiten enviar un email al correo del usuario con un código que le permite usar y actualizar a una nueva contraseña.

***Endpoints* para obtener y actualizar datos del usuario logeado**

Los siguientes *endpoints* son usados con el *token Bearer,* otorgado por el sistema al usuario autenticado:

- El *endpoint* permite obtener todos los datos del usuario que inicio sesión en el sistema.
- El *endpoint* permite actualizar los datos del perfil del usuario autenticado.
- El *endpoint* realiza una actualización de la contraseña del usuario logeado al sistema; se debe ingresar la contraseña actual junto a la nueva contraseña y confirmación de la nueva contraseña.

***Endpoints* para gestionar odontólogos**

Los siguientes *endpoints* son usados con el *token Bearer,* únicamente con rol de administrador:

- El *endpoint* para crear usuario con rol odontólogo es permitido únicamente a los usuarios con rol administrador.
- El sistema cuenta con el *endpoint* que permite crear usuarios con rol de Administrador, si el consultorio lo requiere.
- *Endpoints* que permiten obtener todos los usuarios, un usuario especificado por el número de cédula y usuarios según rol.
- *Endpoint* que realiza una actualización de los datos de un usuario mediante cédula del usuario.
- *Endpoint*s que permiten actualizar el estado de los usuarios de habilitado a deshabilitado, y viceversa.

***Endpoints* para gestionar pacientes**

Se cuenta con *endpoints* tales como, crear usuario paciente, actualizar datos de usuario con uso de la cédula, cambio de estado de usuario, y la obtención de usuarios.

***Endpoint* para visualizar citas odontológicas**

Endpoint que permite obtener las citas médicas según el rol del usuario, en el caso del administrador puede visualizar todas las citas; para el odontólogo solo puede visualizar aquellas que ha creado de atención; y para el paciente aquellas citas disponibles, así como las que ha agendado.

***Endpoints* para agendar citas odontológicas**

Endpoint que permita a un paciente cambiar el estado de la cita para agendarla.

Endpoint que permite al usuario administrador agendar una cita a un paciente (que se acerque al consultorio directamente – parte de la lógica del negocio).

***Endpoint* para actualizar datos de la cita médica**

Endpoint que permite actualizar los datos de la cita médica sea este usuario odontólogo o administrador.

***Endpoints* para registrar horario de atención**

Endpoint que permite al usuario odontólogo crear una cita médica según su hora de disponibilidad.

Endpoint que permite al usuario administrador crear una cita médica según la hora de disponibilidad de un odontólogo específico.

***Endpoint* para cancelar cita**

Endpoint que permite cancelar una cita médica.

***Endpoint* para seleccionar cita odontológica**

Endpoint que permite obtener los datos de una cita médica especifica.

Endpoint para visualizar citas según odontólogo para una mejor selección de citas para el usuario.

***Endpoints* para gestionar historias clínicas**

Endpoint que permite crear una historia clínica, en caso exclusivo, el administrador puede crear una historia clínica a usuarios que estén registrados pero que no cuenten con una historia clínica. **Importante:** cada paciente registrado se crea en conjunto con su historia clínica.

Endpoint que permite actualizar los antecedentes del paciente en la historia clínica de un paciente.

Endpoint que permite crear un nuevo detalle en la historia clínica de un paciente.

Endpoint que permite actualizar el detalle en la historia clínica de un paciente.

***Endpoints* para visualizar historias clínicas**

Endpoint que permite visualizar historia clínica según la cedula del paciente.

***Endpoints* para visualizar historias clínicas propia**

Endpoint que permite visualizar al paciente su propia historia clínica.

***Endpoints* para visualizar información del consultorio**

Endpoint que permite visualizar información importante sobre el consultorio.

***Endpoints* gestionar los servicios**

Endpoints que permite registrar servicios que se brindan en el consultorio; consultar servicios; actualizar servicios y eliminar servicios.

Servicio de recordatorio de citas a pacientes y odontólogos.

Para cumplir con el requerimiento de envío de recordatorio de citas, se ha implementado *Task Schedule* de *Laravel* permitiendo automatizar la tarea de enviar recordatorios de citas médicas a los dispositivos móviles de los usuarios registrados. Cabe mencionar que cada API del sistema se encuentra documentada en el **ANEXO IV-Documentación de las APIs**.

Diseño de la Base de Datos

Alwaysdata, permite el almacenamiento de los datos del sistema se logra la gestión correcta de los datos con MySQL. El diseño se la base de datos se lo muestra en la sección **2.2 Diseño de la arquitectura** apartado Arquitectura de Datos del presente documento.

3.2 *Sprint* 1. Autenticación y registro

Para este *sprint* se tiene los siguientes resultados:

- *Endpoints* para el registro de Usuario
- *Endpoints* de Autenticación, cierre de sesión y recuperar contraseña
- *Endpoints* para obtener y actualizar datos del usuario autenticado

Endpoint para el registro de Usuario

En la funcionalidad de registro de usuario, crea un usuario con rol “paciente” con el consumo de *endpoint* sin necesidad de usar un *token,* debido a que se trata de un *endpoint* público. Permite enviar los datos del usuario a registrar, ya que se trata de un *endpoint* de tipo *POST*.

A continuación, la **Figura 3.2** muestra el resultado exitoso de la creación de un paciente.

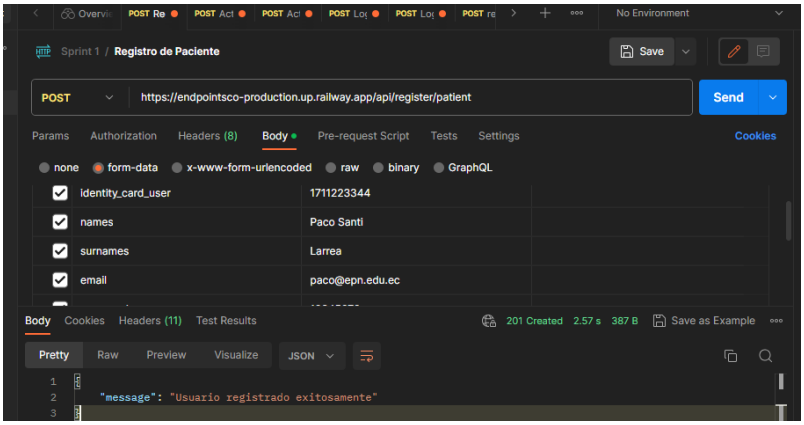

Figura 3.2 Método *POST* registrar paciente

Endpoints de Autenticación, cierre de sesión y recuperar contraseña

Para la funcionalidad de autenticación se envía las credenciales del usuario para obtener acceso a las funcionalidades, en este apartado de inicio de sesión desde la aplicación móvil se envía el *token* del dispositivo para acceder a la funcionalidad de recordatorio de citas.

A continuación, la **Figura 3.3** muestra el resultado exitoso de la autenticación.

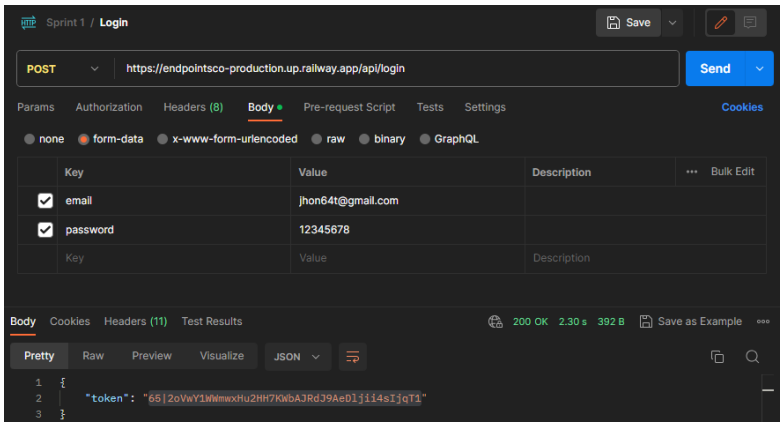

Figura 3.3 Método *POST* inicio de sesión

Para el cierre de sesión se usa el *token Bearer* que junto al *endpoint* de autenticación son de tipo *POST*.

A continuación, la **Figura 3.4** muestra el resultado exitoso del cierre de sesión.

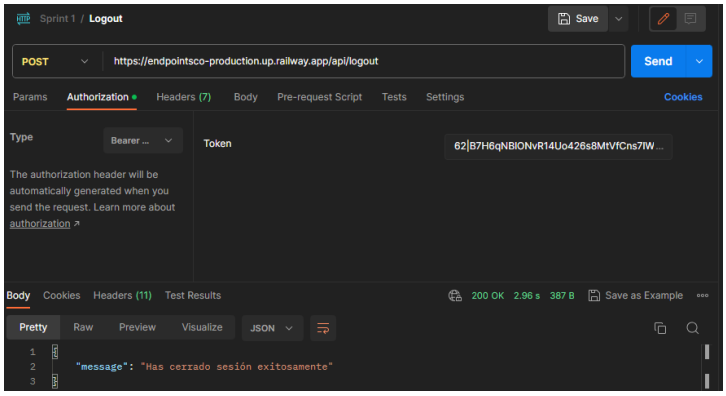

Figura 3.4 Método *POST* cerrar sesión

Para el restablecimiento de contraseña se cuenta con tres *endpoints,* el primer *endpoint* es para la creación de un código y envío de email al correo electrónico especificado, el segundo *endpoint* es para verificar que el código enviado es válido para el restablecimiento de la contraseña y el tercer *endpoint* permite la actualización de la contraseña.

A continuación, la **Figura 3.5**, **Figura 3.6** y **Figura 3.7** muestran el resultado exitoso del cada *endpoint* descrito anteriormente.

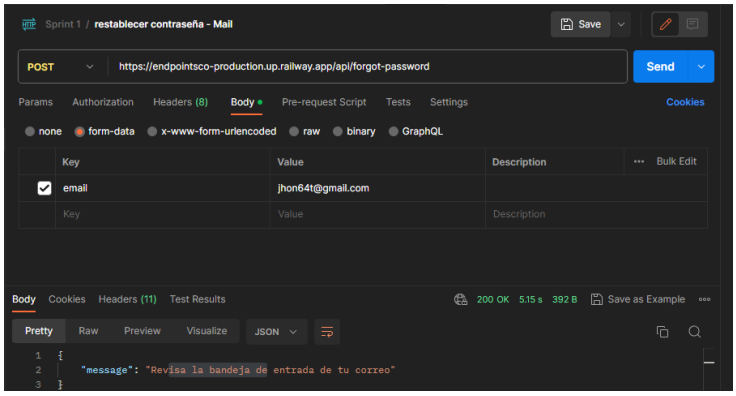

Figura 3.5 Método *POST* envió de correo recuperar contraseña

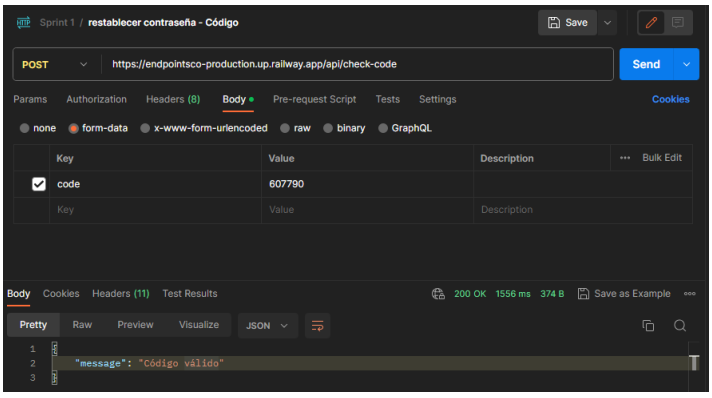

Figura 3.6 Método *POST* verificar código recuperar contraseña

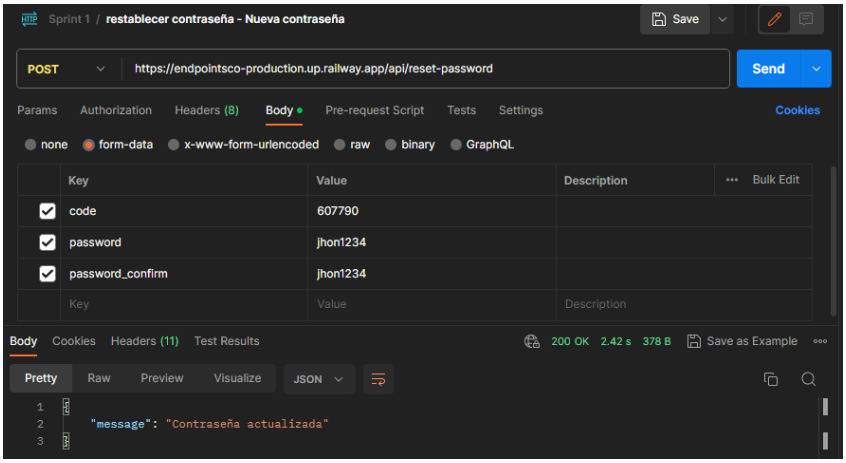

Figura 3.7 Método *POST* cambio de contraseña

Endpoints **para obtener y actualizar datos del usuario autenticado**

Para la gestión de perfil del usuario autenticado, con uso del *token*, se cuenta con los siguientes *endpoints:*

Endpoint privado de tipo *GET* que permite obtener los datos del usuario autenticado (ver en la **Figura 3.8** el resultado exitoso de la implementación de este *endpoint*).

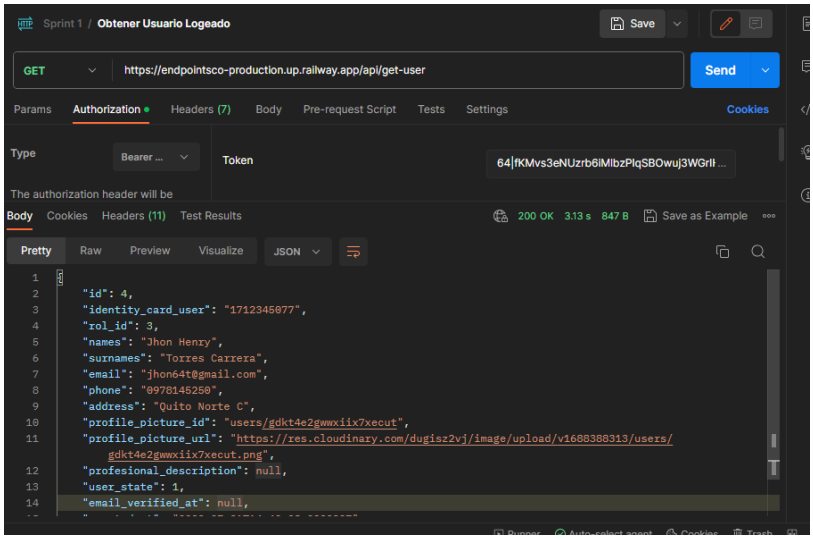

Figura 3.8 Método *GET* obtener datos del usuario autenticado

Endpoint privado de tipo *POST* que permite actualizar los datos personales del perfil del usuario autenticado (ver en la Figura 3.9 el resultado exitoso de la implementación de este *endpoint*), en este apartado la imagen para el perfil no es **obligatorio**.

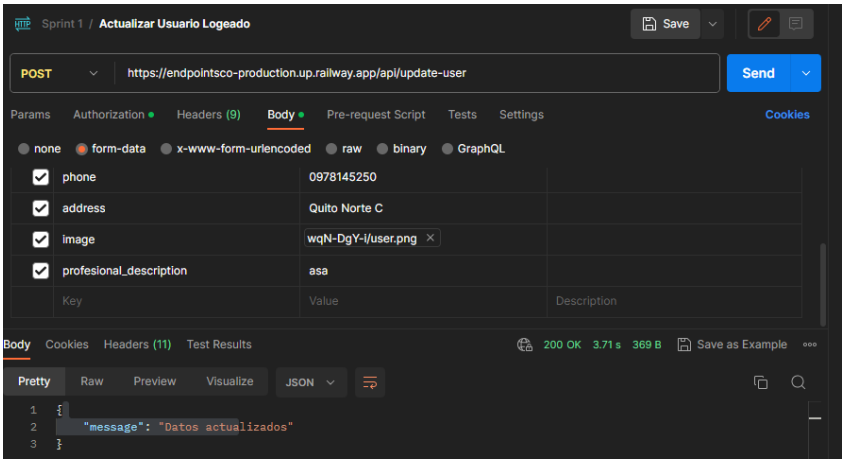

Figura 3.9 Método *POST* actualizar datos de usuario autenticado

Endpoint privado de tipo *POST* que permite actualizar la contraseña del usuario autenticado (ver en la **Figura 3.10** el resultado exitoso de la implementación de este *endpoint*).

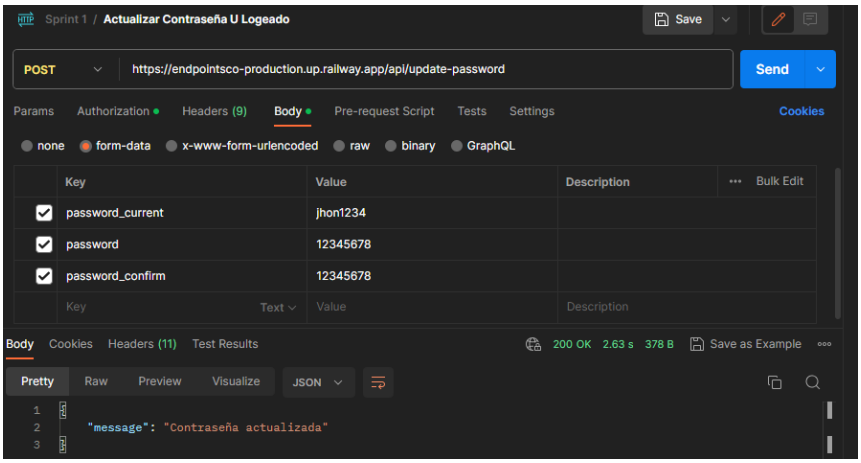

Figura 3.10 Método *POST* actualizar contraseña de usuario autenticado

3.3 *Sprint* 2. Gestión de usuarios y visualización de citas.

Para este *sprint* se tiene los siguientes resultados:

- *Endpoints* para gestionar odontólogos
- *Endpoints* para gestionar pacientes
- *Endpoints* para visualizar citas odontológicas

***Endpoints* para gestionar odontólogos**

Los siguientes *endpoints* han sido usados con el *token Bearer,* las siguientes funcionalidades del sistema son únicamente para usuarios con rol de administrador:

Endpoint privado de tipo *POST* para crear usuario con rol de odontólogo, permite al usuario Administrador crear un nuevo odontólogo con los datos (número de cédula de identidad, nombres, apellidos, correo, teléfono, dirección, descripción profesional e imagen de perfil) respectivos (ver en la **Figura 3.11** el resultado exitoso de la implementación de este *endpoint*).

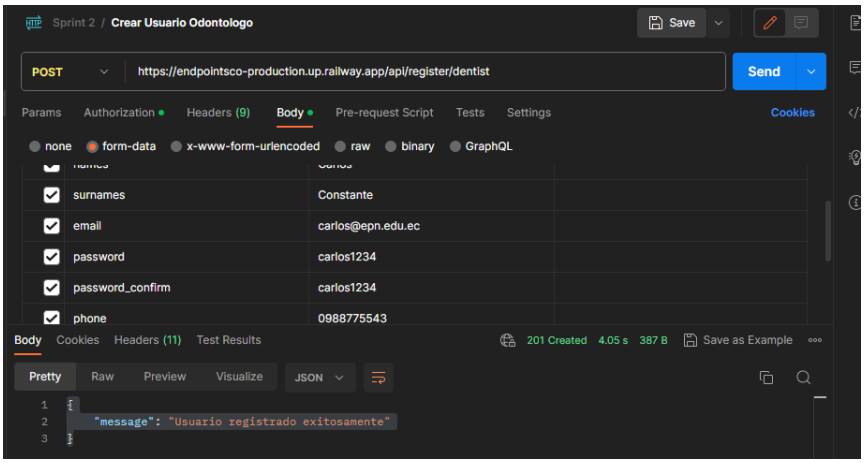

Figura 3.11 Método *POST* crear usuario Odontólogo

Endpoint privado de tipo *POST* que permite crear usuarios con rol “Administrador”, esta funcionalidad es para usuarios con rol de Administrador (ver en la **Figura 3.12** el resultado exitoso de la implementación de este *endpoint*).

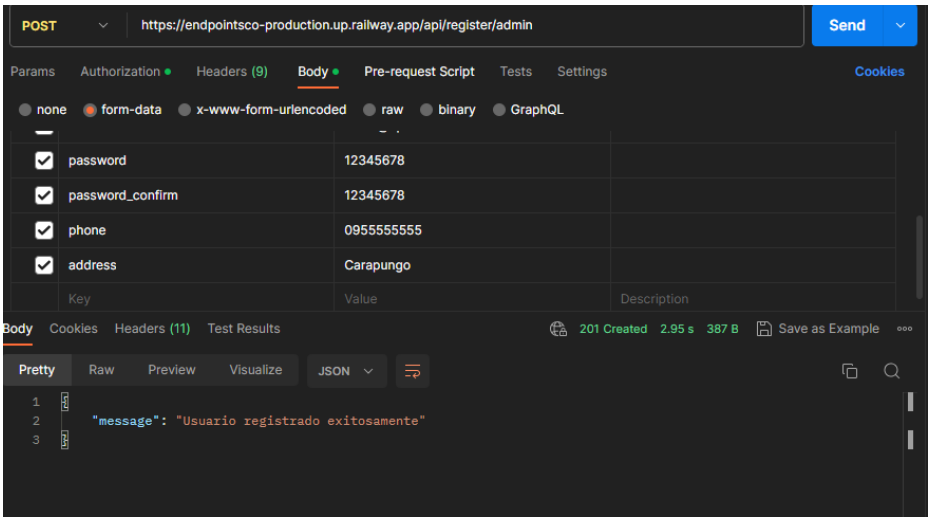

Figura 3.12 Método *POST* crear usuario administrador

Endpoints privados de tipo *GET* que permiten obtener todos los usuarios que se encuentran en la base de datos, *endpoint* para obtener un usuario especifico por el número de cédula y *endpoint* para obtener usuarios según rol.

A continuación, la **Figura 3.13**, **Figura 3.14** y **Figura 3.15** muestran el resultado exitoso del cada *endpoint* descrito anteriormente.

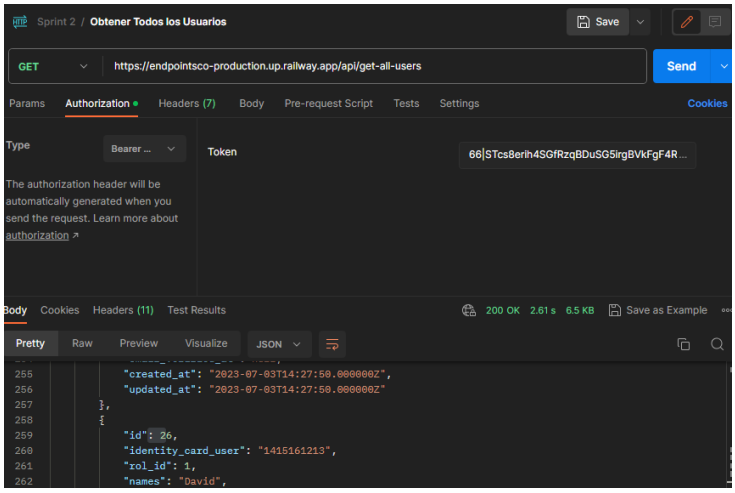

Figura 3.13 Método *GET* obtener todos los usuarios

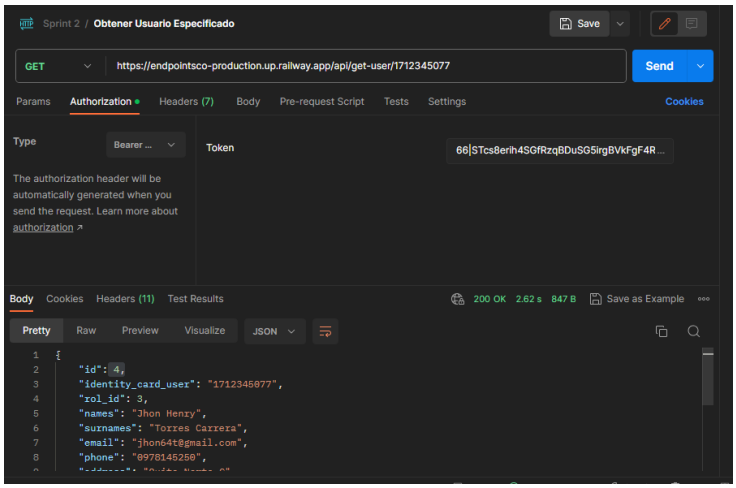

Figura 3.14 Método *GET* obtener usuario específico

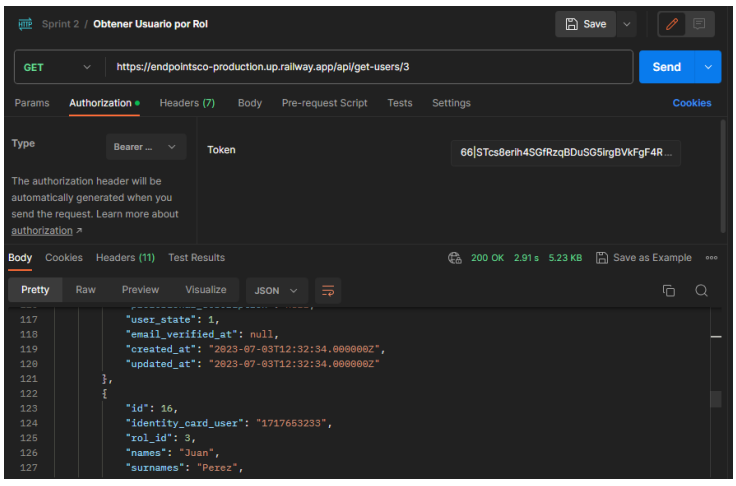

Figura 3.15 Método *GET* obtener usuarios según rol

Endpoint privado de tipo *POST* que realiza una actualización de los datos de un usuario mediante la cédula (ver en la **Figura 3.16** el resultado exitoso de la implementación de este *endpoint*), para este apartado no es **obligatorio** la imagen de perfil del usuario.

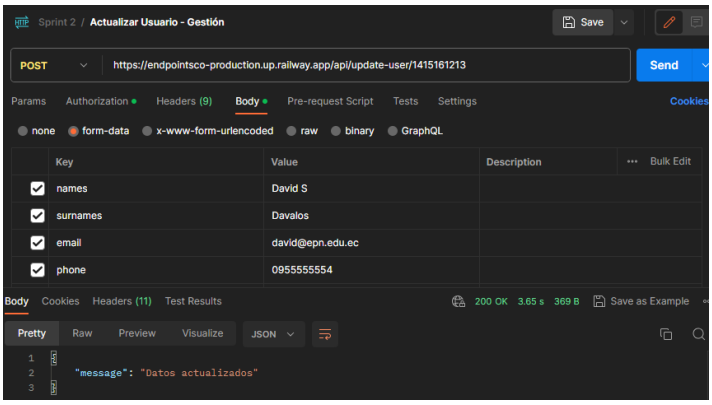

Figura 3.16 Método *POST* actualizar datos de usuario por el administrador

Endpoints privados de tipo POST que permiten actualizar el estado (habilitar y deshabilitar) de los usuarios, para el caso de que un usuario sea deshabilitado no podrá obtener el acceso al sistema (ver en la **Figura 3.17** y **Figura 3.18** el resultado exitoso de la implementación de este *endpoint*).

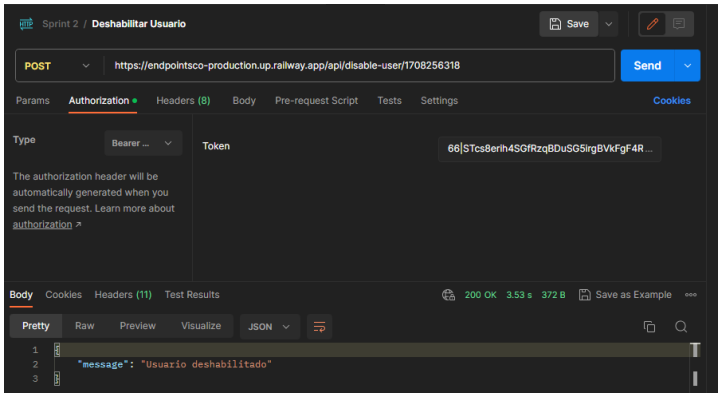

Figura 3.17 Método *POST* deshabilitar usuario por el administrador

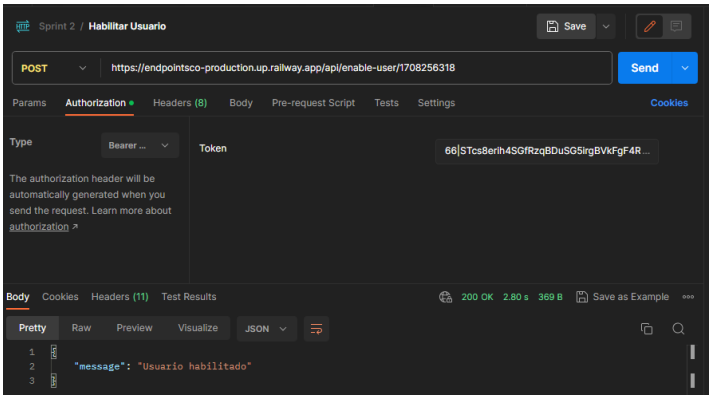

Figura 3.18 Método *POST* habilitar usuario por el administrador

Endpoints para gestionar pacientes

Para estos *endpoints* se reutiliza la funcionalidad creada en el *sprint 1*, para crear pacientes (ver **Figura 3.2**), para actualizar con uso de la cédula (ver **Figura 3.16**), los *endpoints* cambio de estado (ver **Figura 3.17** y **Figura 3.18**), y la obtención de usuarios (ver **Figura 3.15**). Estos *endpoints* se han mencionado en el resultado anterior de este *sprint*.

Endpoint para visualizar citas odontológicas

Endpoint privado de tipo *GET* que permite obtener las citas médicas según el rol del usuario, para el administrador devuelve todas las citas de la base de datos; para el odontólogo retorna las citas que el odontólogo ha creado de atención, y para el paciente se retorna todas las citas disponibles y las que ha agendado (ver en la **Figura 3.19** el resultado exitoso de la implementación de este *endpoint*).

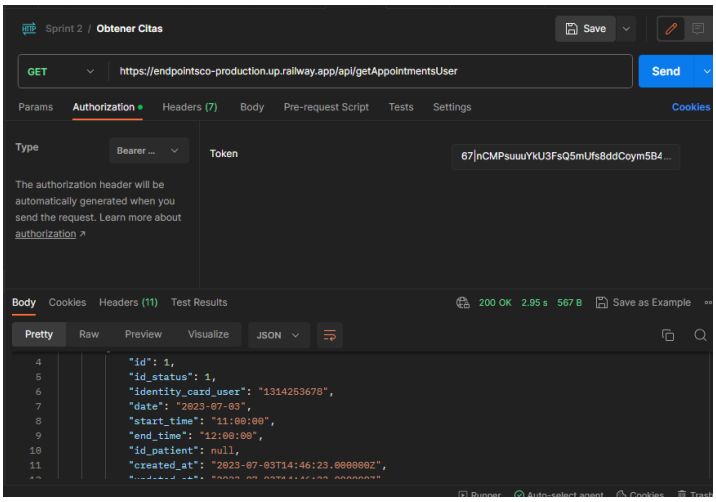

Figura 3.19 Método *GET* obtener citas según el usuario autenticado

3.4 *Sprint* 3. Citas Odontológicas.

Para este *sprint* se tiene los siguientes resultados:

- *Endpoints* para registrar horario de atención
- *Endpoints* para actualizar datos de la cita médica
- *Endpoints* para agendar citas odontológicas

Endpoints para registrar horario de atención

Endpoint privado que permite al usuario odontólogo crear una cita médica según su hora de disponibilidad, esta funcionalidad valida que no conste una cita del mismo odontólogo

en una misma hora de inicio el mismo día, antes al nuevo registro (ver en la **Figura 3.20** el resultado exitoso de la implementación de este *endpoint*).

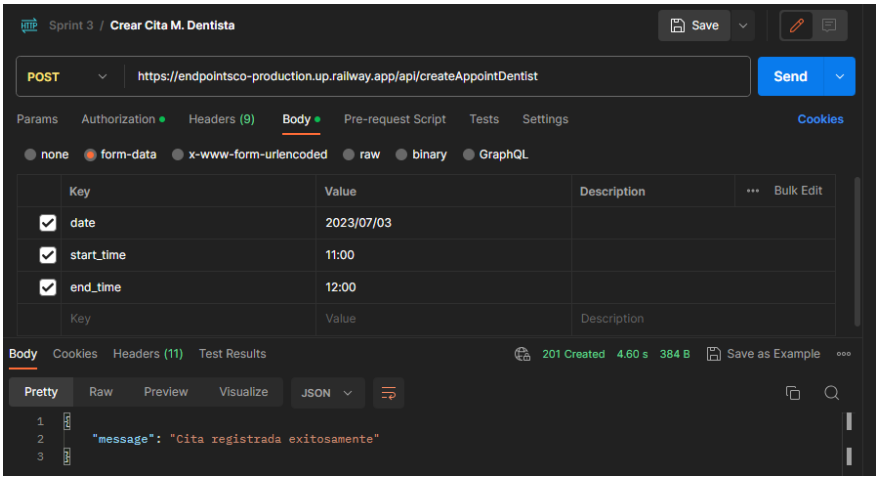

Figura 3.20 Método *POST* crear cita médica desde el odontólogo

Endpoint privado que permite al usuario administrador crear una cita médica según la hora de disponibilidad de un odontólogo específico (ver en la **Figura 3.21** el resultado exitoso de la implementación de este *endpoint*), validando que no se repita una cita con el mismo odontólogo con mismo horario y día.

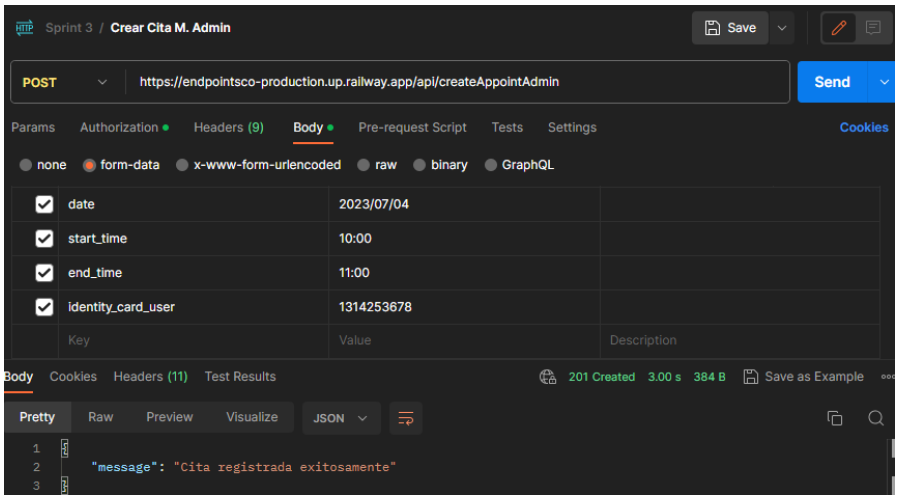

Figura 3.21 Método *POST* crear cita médica desde el administrador

Endpoint para actualizar datos de la cita médica

Endpoint privado que permite actualizar los datos de la cita médica, sea este usuario odontólogo o administrador (ver en la **Figura 3.22** el resultado exitoso de la implementación de este *endpoint*), al igual que al crear se valida que el odontólogo no cuente con una cita de las mismas características.

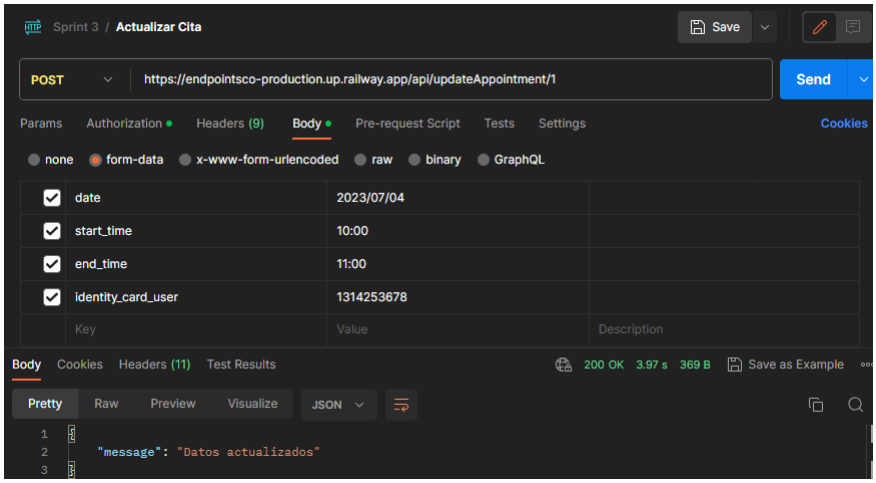

Figura 3.22 Método *POST* actualizar cita médica desde el administrador

Endpoints para agendar citas odontológicas

Endpoint privado que permite a un paciente cambiar el estado de la cita a “agendada” para poder apartar una cita médica (ver en la **Figura 3.23** el resultado exitoso de la implementación de este *endpoint*).

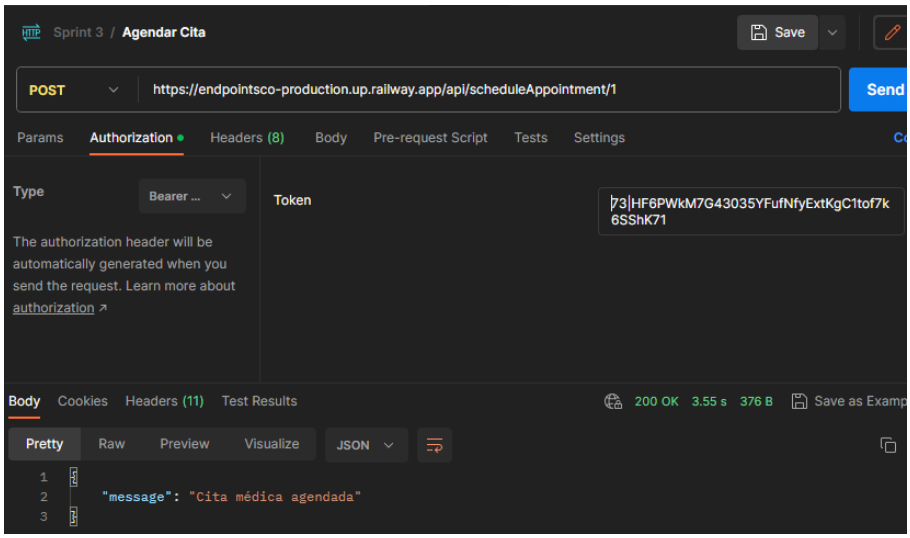

Figura 3.23 Método *POST* agendar cita médica desde el Paciente

Endpoint privado que permite al usuario administrador agendar una cita a un paciente que se acerque al consultorio directamente (ver en la **Figura 3.24** el resultado exitoso de la implementación de este *endpoint*), cabe mencionar que si bien así es la lógica del negocio existe la restricción de que el paciente debe estar registrado en el sistema.

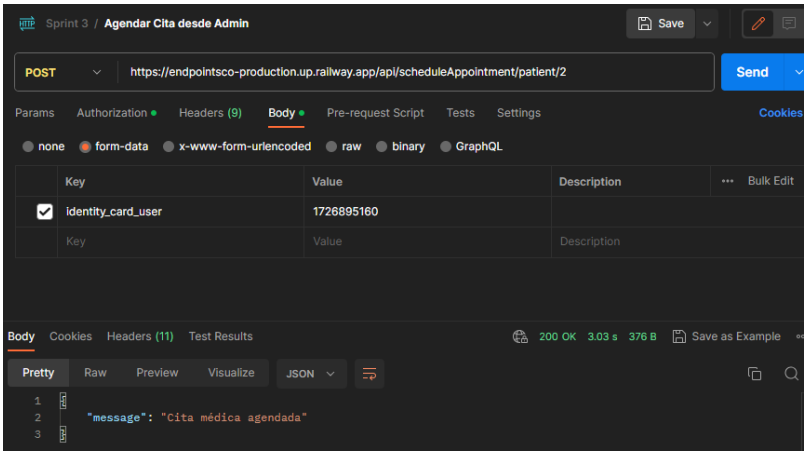

Figura 3.24 Método *POST* agendar cita médica desde el administrador

3.5 *Sprint* 4. Citas Odontológicas e Historias Clínicas

Para este *sprint* se tiene los siguientes resultados:

- *Endpoint* para seleccionar cita odontológica
- *Endpoints* para cancelar cita
- *Endpoints* para gestionar los servicios
- *Endpoints* para gestionar historias clínicas

Endpoint para seleccionar cita odontológica

Endpoint que permite obtener los datos de una cita médica especifica con el id de la cita médica (ver en la **Figura 3.25** el resultado exitoso de la implementación de este *endpoint*), este consumo de *endpoint* lo puede hacer cualquier usuario sin importar el rol que tenga.

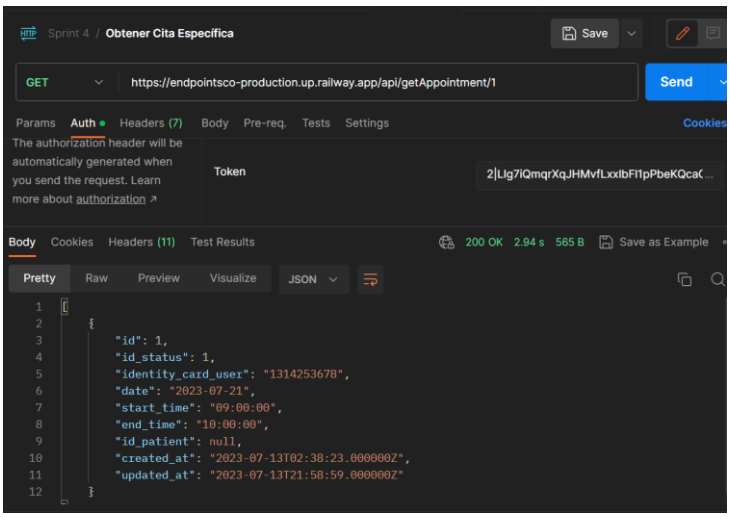

Figura 3.25 Método *GET* obtener cita específica

Endpoint para visualizar citas según odontólogo, esto con el finde mejor la selección de citas para el usuario, la consulta se realiza según la cédula del odontólogo (ver en la **Figura 3.26** el resultado exitoso de la implementación de este *endpoint*). Adicionalmente, no se toma en cuenta el rol del usuario para consumirlo.

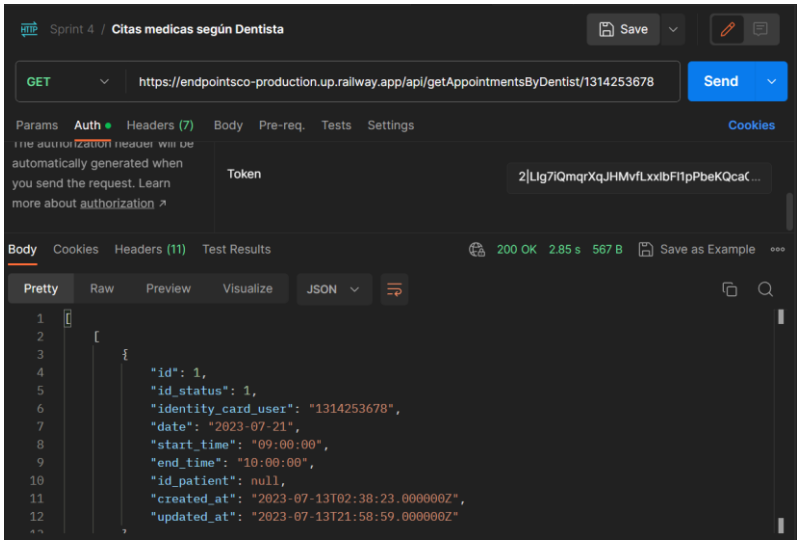

Figura 3.26 Método *GET* obtener citas según el odontólogo especificado

Endpoint para cancelar cita

Endpoint que permite cancelar una cita médica, los usuarios con rol paciente son los únicos que pueden cancelar sus propias citas médicas (ver en la **Figura 3.27** el resultado exitoso de la implementación de este *endpoint*).

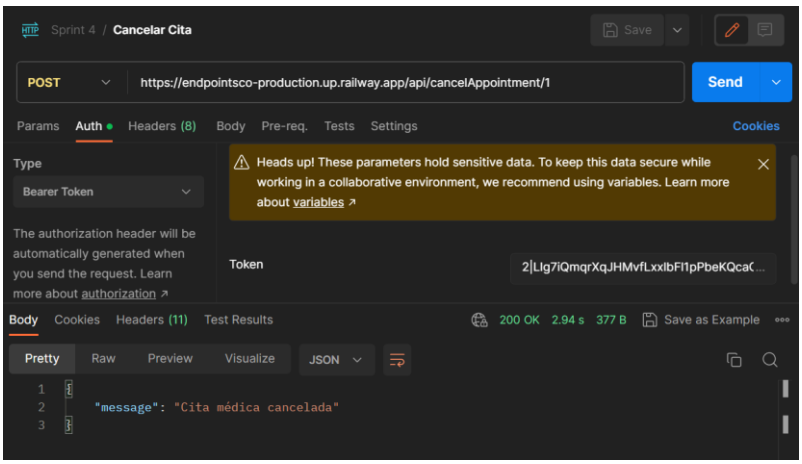

Figura 3.27 Método *POST* cancelar cita médica

Endpoints gestionar los servicios

Endpoints que permite registrar servicios que se brindan en el consultorio: consultar, actualizar y eliminar servicios.

El *endpoint* para registrar servicios únicamente es permitido para el Administrador (ver en la **Figura 3.28** el resultado exitoso de la implementación de este *endpoint*).

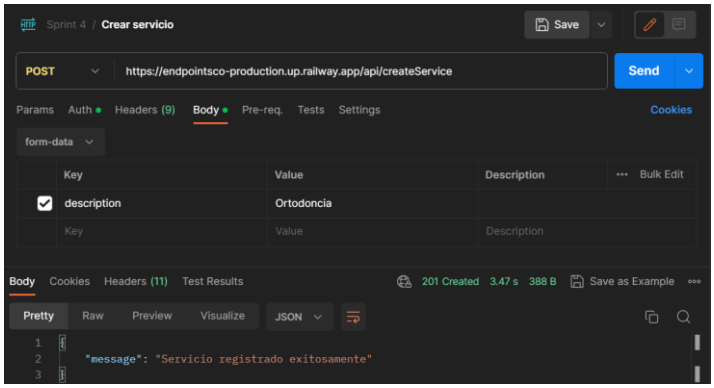

Figura 3.28 Método *POST* crear servicio

El *endpoint* para obtener servicios es permitido a todos los usuarios sin importar el rol (ver en la **Figura 3.29** el resultado exitoso de la implementación de este *endpoint*).

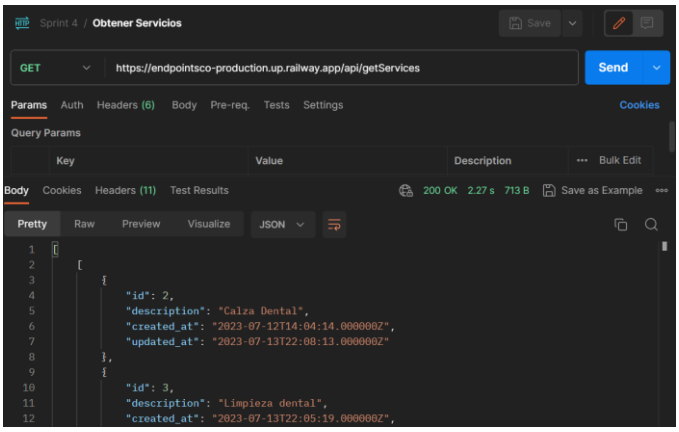

Figura 3.29 Método *GET* obtener los servicios

El *endpoint* para actualizar servicio especificando el id del servicio, únicamente es permitido el uso a los usuarios con rol de Administrador (ver en la **Figura 3.30** el resultado exitoso de la implementación de este *endpoint*).

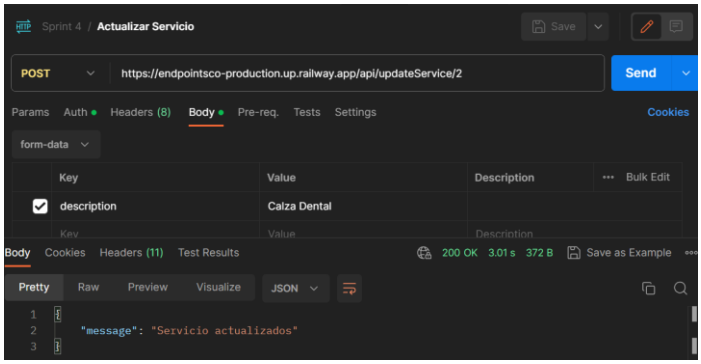

Figura 3.30 Método *POST* actualizar servicio

El *endpoint* eliminar servicio por id, únicamente es permitido para el rol de Administrador (ver en la **Figura 3.31** el resultado exitoso de la implementación de este *endpoint*).

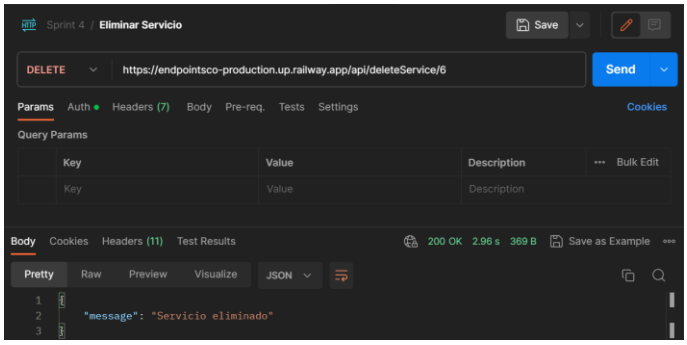

Figura 3.31 Método *DELETE* eliminar servicio

Endpoints para gestionar historias clínicas

Endpoint que permite crear una historia clínica, en caso exclusivo, el administrador crea una historia clínica a usuarios que estén registrados pero que no cuenten con una historia clínica (ver en la **Figura 3.32** el resultado exitoso de la implementación de este *endpoint*). **Importante:** cada paciente registrado se crea en conjunto con su historia clínica.

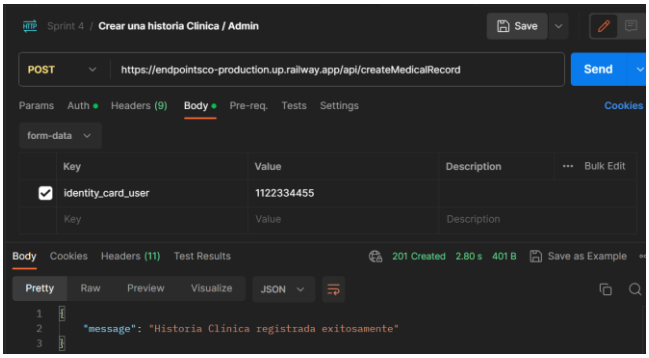

Figura 3.32 Método *POST* crear historia Clínica

Endpoint que permite actualizar los antecedentes del paciente la historia clínica de un paciente (ver en la **Figura 3.33** el resultado exitoso de la implementación de este *endpoint*).

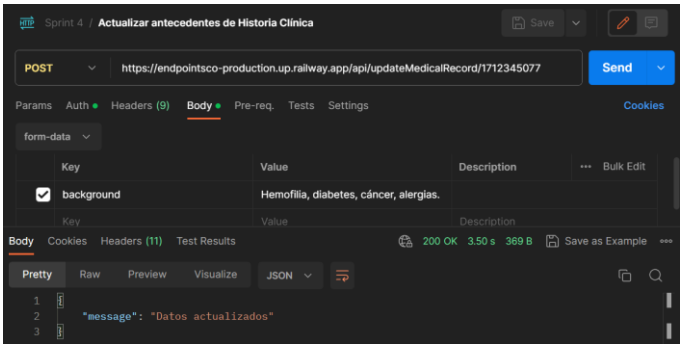

Figura 3.33 Método *POST* actualizar historia Clínica

Endpoint que permite crear un nuevo detalle en la historia clínica de un paciente (ver en la **Figura 3.34** el resultado exitoso de la implementación de este *endpoint*). El usuario con rol odontólogo puede registrar detalles de una historia clínica. Cabe mencionar que en el apartado de odontograma se valida el JSON en el *Frontend.*

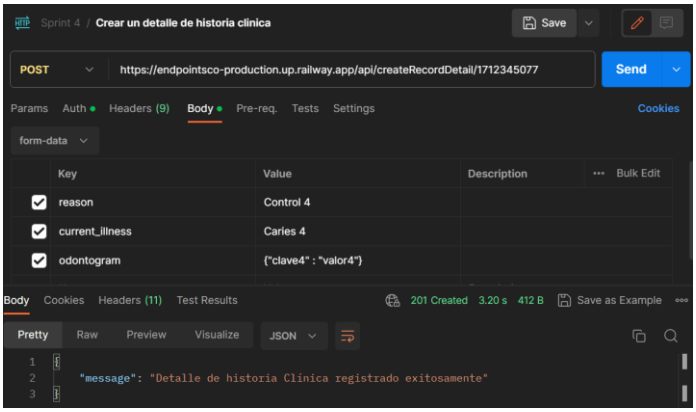

Figura 3.34 Método *POST* crear detalle de historia clínica

Endpoint que permite actualizar el detalle específico de una historia clínica de un paciente especifico (ver en la **Figura 3.35** el resultado exitoso de la implementación de este *endpoint*). El usuario con rol odontólogo puede registrar detalles de una historia clínica. Cabe mencionar que en el apartado de odontograma se valida el JSON en el *Frontend.*

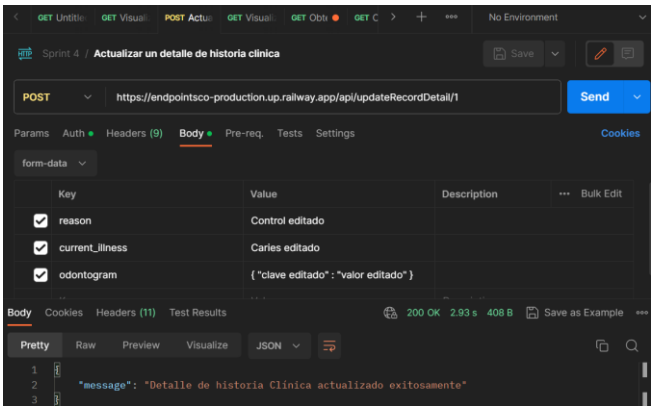

Figura 3.35 Método *POST* actualizar un detalle de historia clínica

3.6 *Sprint* 5. Historias clínicas y Landing Page

Para este *sprint* se tiene los siguientes resultados:

- *Endpoint* para visualizar historias clínicas
- *Endpoints* para visualizar información del Consultorio
- Enviar recordatorios a pacientes y odontólogos

Endpoints para visualizar historias clínicas

Endpoint que permite visualizar historia clínica según la cédula del paciente (ver en la **Figura 3.36** el resultado exitoso de la implementación de este *endpoint*), es utilizado por los usuarios con rol Administrador y Odontólogo.

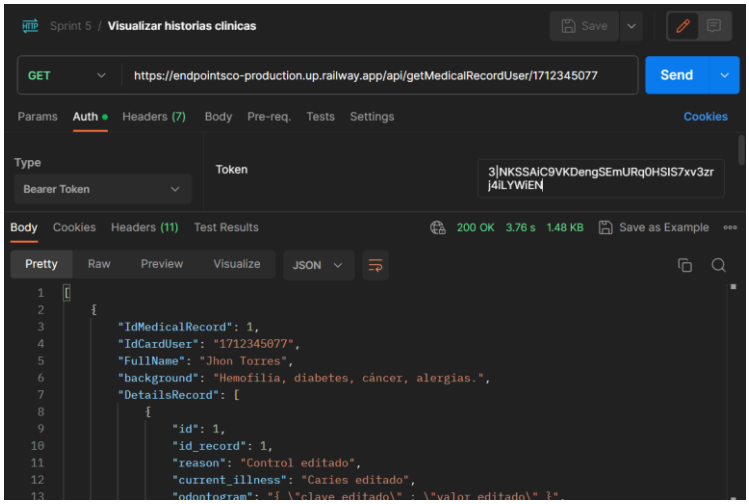

Figura 3.36 Método *GET* visualizar historia clínica de paciente

Además, se cuenta con un *endpoint* que permite visualizar al paciente su propia historia clínica (ver en la **Figura 3.37** el resultado exitoso de la implementación de este *endpoint*), es utilizado para obtener historia clínica del usuario autenticado.

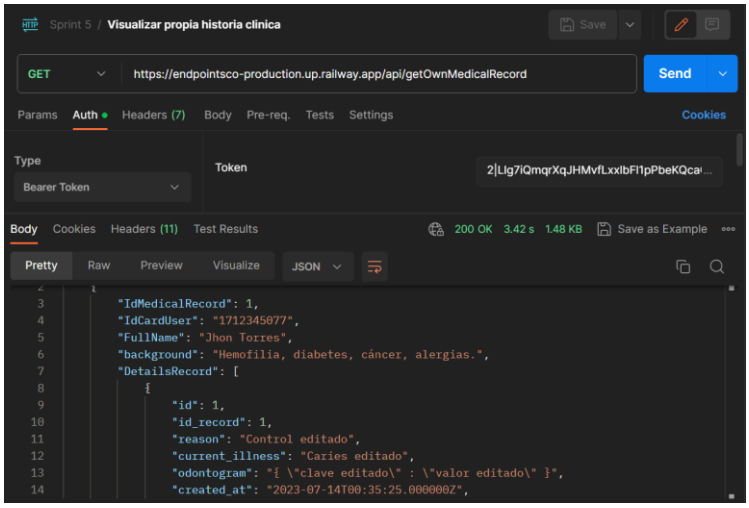

Figura 3.37 Método *GET* visualizar propia historia clínica

Endpoints para visualizar información del Consultorio

Endpoint que permite visualizar información importante sobre el consultorio, listado de los odontólogos y los servicios que tiene el consultorio (ver en la **Figura 3.38** el resultado exitoso de la implementación de este *endpoint*).

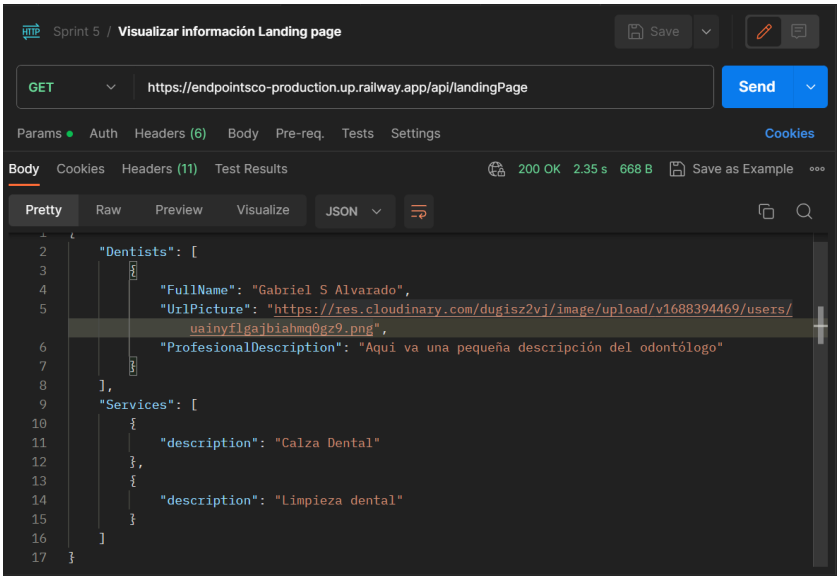

Figura 3.38 Método *GET* visualizar información para Landing Page

Enviar recordatorios a pacientes y odontólogos.

Para lograr este requerimiento se ha hecho uso de *Task Schedule* de *Laravel*, para automatizar los recordatorios de citas médicas de los usuarios, teniendo en cuenta la fecha de las citas que estén en el rango de 2 días hasta llegar al día de la cita, se envía el recordatorio diariamente a los dispositivos móviles de los usuarios (ver en la **Figura 3.39** la implementación de la tarea de envío de recordatorios).

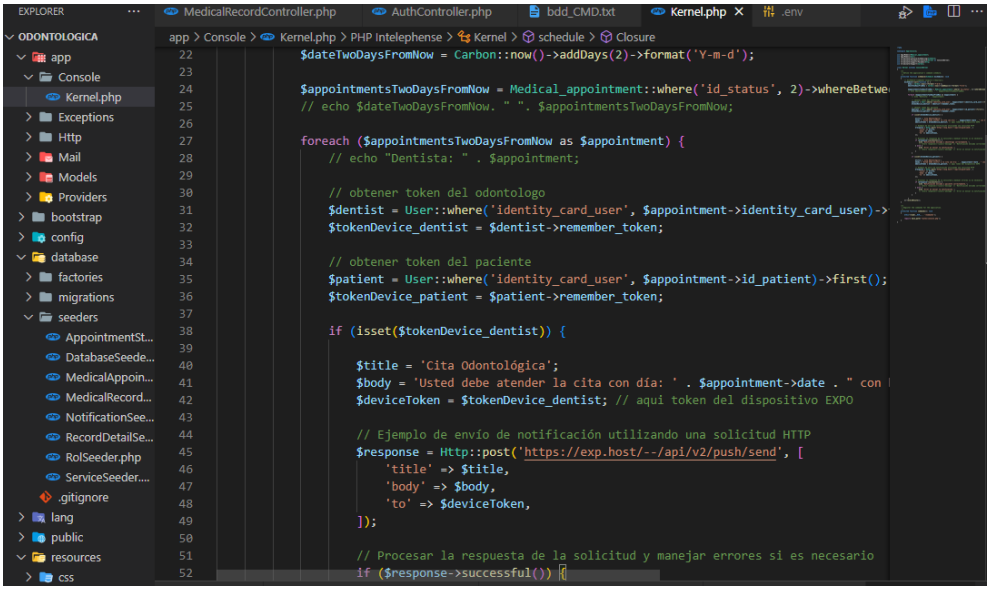

Figura 3.39 Tarea de envío de recordatorios

3.7 *Sprint* 6. Pruebas y Despliegue

Para este *sprint* se tiene los siguientes resultados:

- Pruebas unitarias
- Prueba de carga
- Prueba de performance
- Despliegue del *backend*

Pruebas unitarias

Este tipo de pruebas permiten comprobar si un fragmento de código en específico funciona de la manera correcta y esperada. Es uno de los procedimientos que no deben faltar antes de la integración de todo un sistema, puesto que al aislar una parte del código y probarlo se logra validar el comportamiento lógico y objetivo [20]. Es importante mencionar que este tipo de pruebas se realiza durante la fase de desarrollo, por ello cada *endpoint* implementado se ha ido probando como se puede observar en cada uno de los *sprints* anteriores.

Además, con el objetivo de asegurar que cada unidad de código funcione correctamente y produzca los resultados esperados, se han realizado pruebas unitarias especificas a aquellos requerimientos que manejan la “base” del negocio (CORE del negocio). A continuación, la **Figura 3.40** muestra la prueba unitaria al método “crear historia clínica desde el administrador”, mientras que la **Figura 3.41** el resultado obtenido de dicha prueba. En el **ANEXO II-Pruebas** se puede visualizar el resto de las pruebas unitarias realizadas.

```php
public function test_crear_historia_clinica_adm(): void
{
    $user = User::factory() ->make([
        'rol_id' => 1
    ]);
    $Cedula_id = [
        "identity_card_user" => "1314253678",
    ];
    $request = $this->actingAs($user)->post(sprintf('/api/createMedicalRecord'), $Cedula_id);
    $request -> assertStatus(201);
}
```

Figura 3.40 Prueba unitaria crear historias

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

 PASS  Tests\Unit\MedicalRecordTest
 ✓ crear historia clinica adm

  Tests:    1 passed (1 assertions)
  Duration: 2.67s
```

Figura 3.41 Resultado de la prueba unitaria crear historias

Prueba de carga

El objetivo principal de estas pruebas es identificar posibles cuellos de botella, evaluar la capacidad de respuesta y rendimiento del sistema, y asegurar que pueda manejar la cantidad esperada de usuarios y transacciones sin degradar su desempeño.

Para el *endpoint* de obtener las citas médicas según el usuario autenticado, se ha realizado una estimación de 600 usuarios realizando 20 solicitudes en simultaneo para obtener la información de las citas médicas. Obteniendo que los tiempos durante las peticiones, el menor tiempo ha sido de 1559 ms junto al mayor tiempo de 2693 ms, teniendo un tiempo de 51.107 segundos total para completar la prueba (ver **Figura 3.42**). En el **ANEXO II-Pruebas** se puede visualizar el resto de las pruebas de carga realizadas.

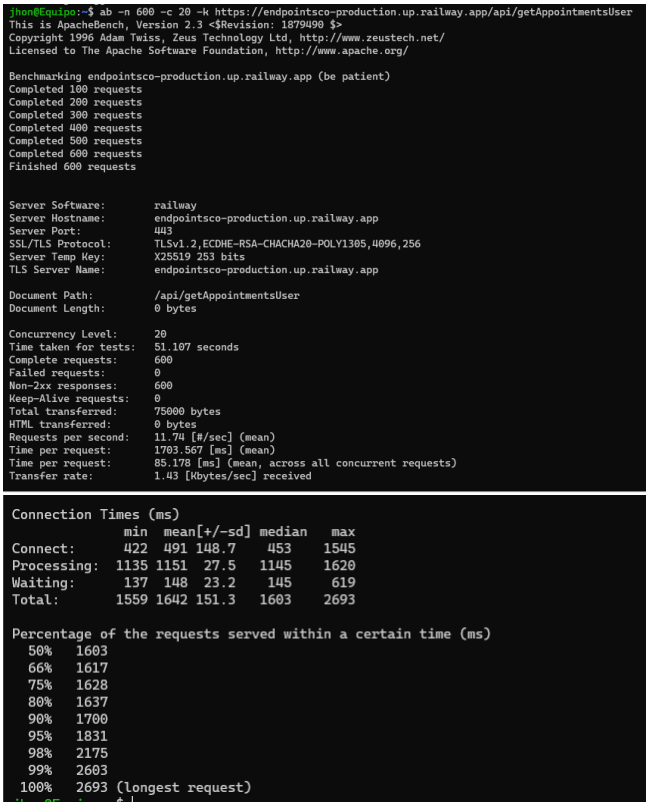

Figura 3.42 Prueba de carga 1

Prueba de performance

El propósito principal de estas pruebas es verificar cómo el *software* se comporta en términos de velocidad, tiempo de respuesta, consumo de recursos y estabilidad, con el objetivo de identificar y solucionar posibles problemas de rendimiento.

Railway proporciona un panel de métricas sobre los recursos que son usados al hacer uso de los servicios de la aplicación. Para esto se cuenta con los recursos de CPU, Memoria y Red.

A continuación, la **Figura 3.43** muestra que al realizar un número de peticiones de hasta 5000 peticiones a la API, el CPU no obtiene variación en la métrica “recursos del CPU”). En el **ANEXO II-Pruebas** se puede visualizar el resultado de la prueba para el resto de las métricas.

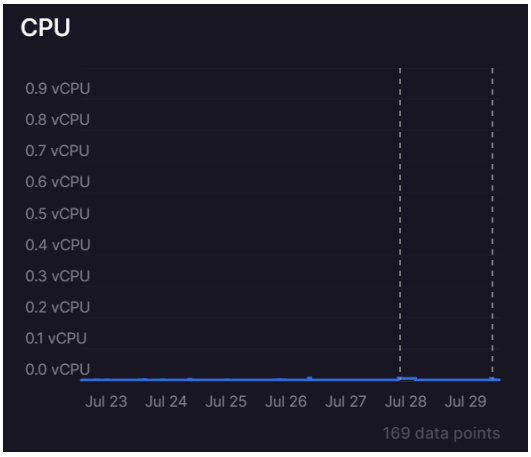

Figura 3.43 Prueba Performace - Métrica de CPU

Despliegue del *backend*

La importancia del despliegue radica en que una vez que se lo logra realizar todos los servicios están disponibles (de manera pública o privada) para el consumo sencillo por parte ya sea del *frontend* o de la aplicación móvil.

Para el despliegue del componente *backend* se ha hecho uso de ***Railway,*** una plataforma de infraestructura que permite el despliegue de manera sencilla, de ahí que se ha hecho uso de esta para desplegar el componente *backend* del presente trabajo de titulación.

Cabe mencionar, que con respecto al proveedor *Railway* el servicio no permite ejecutar *Task Schedule*, por lo que como solución se ha implementado dicho servicio en una opción diferente y que otorga *AlwaysData.* Tanto el error como la solución se exponen a detalle en el **ANEXO II-Errores sobre el uso de Task Schedule en Railway**..

4 CONCLUSIONES

- El uso de la metodología Scrum ha permitido gestionar el proyecto ante los cambios y aspectos que en el desarrollo del *backend* fueron surgiendo para obtener un estado óptimo de los requerimientos para el sistema.
- El *framework* Laravel ha permitido una mayor estructuración del componente *backend*, permitiendo manejar la lógica del negocio mediante los controladores, una organización de las rutas de los *endpoints* del sistema y un manejo de la base de datos SQL por medio de Eloquent.
- El servicio otorgado por *Railway* es muy útil al automatizar los despliegues en cada cambio, esta herramienta facilita los despliegues conforme a los avances del sistema y el versionado del código en *GitHub*.
- Las pruebas que se han implementado tales como, unitarias, de carga y de *performance* sobre los recursos, han evidenciado que el uso de los *endpoints*, presentan un buen rendimiento de respuesta y de funcionalidad.

5 RECOMENDACIONES

- Se recomienda realizar que las credenciales otorgadas en el presente trabajo no sean compartidas con personas que puedan ser un riesgo ante la integridad de los datos generado y almacenados por el sistema.
- Debido a que la lógica de los recordatorios citas corresponde a ser enviados dos días antes de la cita, es recomendable que esta funcionalidad sea desplegada o ejecutada de manera local o no, pero es importante recordar que debe estar en constante funcionamiento algo que corresponde a uso de recursos constantemente.
- El componente *backend* no cuenta con lógica de notificaciones en el sistema web, se recomienda a futuro tomar la implementación de este apartado.
- El dilema que presenta el uso de la herramienta de *Railway* fue al intentar implementar el envío de los recordatorios de las citas médicas a los dispositivos móviles de los usuarios. Ya que este servicio no permite trabajar con Task Schedule que proporciona Laravel para realizar tareas programadas en el servidor.
- La solución aplicada en la problemática de los recordatorios de citas, *alwaysdata* mantiene el sistema de horario algo confuso ya que en la primera configuración se le otorgo una hora de 10:00, siendo que a las 4:48 am en horario ecuatoriano llegó la notificación al dispositivo móvil, su última configuración fue en *alwaysdata* con hora 12:20, llegando los recordatorios de las citas a los respectivos usuarios alrededor de las 7:17 de la mañana.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] Universidad de Murcia, «Introducción a HTML y CSS. Desarrollo de Aplicaciones Web,» 11 Febrero 2019. [En línea]. Available: https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-1-introduccion-html-css.html. [Último acceso: 17 Mayo 2023].

[2] Arimetrics, «Qué es Backend,» 2023 Febrero 2023. [En línea]. Available: https://web.archive.org/web/20230000000000*/https://www.arimetrics.com/glosario-digital/backend. [Último acceso: 17 Mayo 2023].

[3] U. Hernández, «Arquitectura MVC,» Código Facilito, 22 Febrero 2015. [En línea]. Available: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado. [Último acceso: 16 Mayo 2023].

[4] O. Fernández, «Bases de Datos Relacionales y SQL – Introducción,» Aprender Big Data, 25 Diciembre 2022. [En línea]. Available: https://aprenderbigdata.com/bases-de-datos-relacionales/. [Último acceso: 16 Mayo 2023].

[5] AlwaysData, AlwaysData, [En línea]. Available: https://www.alwaysdata.com/en/. [Último acceso: 16 Mayo 2023].

[6] Amazon Web Services, «¿Qué es la API RESTful?,» AWS, 2023. [En línea]. Available: https://aws.amazon.com/es/what-is/restful-api/. [Último acceso: 16 Mayo 2023].

[7] T. Satpathy, «Una guía para el CONOCIMIENTO DE SCRUM,» [En línea]. Available: https://www.tenstep.ec/portal/images/pdfs/Suscripciones_TenStep/Silver/SCRUMstudy_GUIA_SBOK_espanol.pdf. [Último acceso: 16 Mayo 2023].

[8] LoadView de Dotcom-Monitor, «Tutorial de pruebas de carga de postman: WDSL, JSON y API,» LoadView Testing, 16 Octubre 2020. [En línea]. Available: https://www.loadview-testing.com/es/blog/tutorial-de-pruebas-de-carga-de-postman-wdsl-json-y-api/. [Último acceso: 16 Mayo 2023].

[9] Laravel, «Laravel,» 2023. [En línea]. Available: https://laravel.com/docs/10.x. [Último acceso: 16 Mayo 2023].

[10] Kinsta, «El Framework PHP Laravel – Construcción de Aplicaciones Web Para Todos,» Kinsta, 29 Noviembre 2022. [En línea]. Available: https://kinsta.com/es/base-de-conocimiento/que-es-laravel/. [Último acceso: 16 Mayo 2023].

[11] Jeff, «Aprende a usar Eloquent el ORM de Laravel,» Styde, 15 Mayo 2015. [En línea]. Available: https://styde.net/aprende-a-usar-eloquent-el-orm-de-laravel/. [Último acceso: 16 Mayo 2023].

[12] Railway, «Railway,» Railway, [En línea]. Available: https://railway.app/. [Último acceso: 08 Agosto 2023].

[13] D. A., «ApacheBench (ab), realiza pruebas de carga de tu página web,» Ubunlog, 27 Marzo 2023. [En línea]. Available: https://ubunlog.com/apachebench-carga-pagina-web/. [Último acceso: 08 Agosto 2023].

[14] R. E. Stake, Investigación con estudio, Madrid: Morata S. L., 1998.

[15] M. Á. D. DIOS, «Scrum: qué es y cómo funciona este marco de trabajo,» WAM Global, 09 Mayo 2022. [En línea]. Available: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html. [Último acceso: 24 Mayo 2023].

[16] J. Roche, «Artefactos Scrum: las 3 herramientas clave de gestión,» Deloitte, 1 Octubre 2022. [En línea]. Available: https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html. [Último acceso: 24 Mayo 2023].

[17] Team Asana, «Guía de 6 pasos para la recopilación de requisitos para asegurar el éxito de tu proyecto,» Asana, 13 Noviembre 2022. [En línea]. Available: https://asana.com/es/resources/requirements-gathering. [Último acceso: 13 Mayo 2023].

[18] H. Cervantes, «Arquitectura de Software,» Software Guru, 07 07 2022. [En línea]. Available: https://sg.com.mx/revista/27/arquitectura-software. [Último acceso: 29 06 2023].

[19] IBM, «¿Qué es una arquitectura de datos?,» IBM, 06 Agosto 2023. [En línea]. Available: https://www.ibm.com/es-es/topics/data-architecture. [Último acceso: 08 Agosto 2023].

[20] Yeeply, «¿Qué son las pruebas unitarias y cómo llevar una a cabo?,» Yeeply, 28 Marzo 2023. [En línea]. Available: https://www.yeeply.com/blog/que-son-pruebas-unitarias/. [Último acceso: 08 Agosto 2023].

7 ANEXOS

Los anexos son toda la información relevante que transparenta el desarrollo de este proyecto de integración curricular.

- **ANEXO I.** Certificado de Originalidad
- **ANEXO II.** Manual Técnico
- **ANEXO III.** Manual de Usuario
- **ANEXO IV** Manual de Instalación

ANEXO I

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
CAMPUS POLITÉCNICO "ING. JOSÉ RUBÉN ORELLANA"

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 21 de agosto de 2023

De mi consideración:

Yo, IVONNE FERNANDA MALDONADO SOLIZ, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DE UN BACKEND asociado al DESARROLLO DE SISTEMA DE GESTIÓN DE CITAS MÉDICAS PARA CONSULTORIO ODONTOARIAS elaborado por el estudiante JHON HENRY TORRES CARRERA de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones (sin anexos), producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 10%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin.

Atentamente,

Firmado electrónicamente por: IVONNE FERNANDA MALDONADO SOLIZ

Ivonne Maldonado
Docente Ocasional a Tiempo Completo
ESFOT

ANEXO II

Este anexo presenta información necesaria para complementar cada una de las fases del desarrollo de *software* del presente proyecto de integración curricular.

Recopilación de requerimientos

En la **Tabla 4**, se observa la Recopilación de Requerimientos obtenida para el desarrollo del sistema.

Tabla 4 Recopilación de Requerimientos del sistema

Recopilación de requerimientos		
Tipo de sistema	**ID-RR**	**Enunciado del Ítem**
Backend	**RR001**	Como usuario con rol paciente necesita generar *endpoints* para: - Registrarse.
	RR002	Como usuario con rol administrador, doctor o paciente necesita generar *endpoints* para: - Iniciar y cerrar sesión. - Reestablecer contraseña.
	RR003	Como usuario con rol administrador, doctor o paciente necesita generar *endpoints* para: - Gestionar su perfil personal.
	RR004	Como usuario con rol administrador necesita generar *endpoints* para: - Gestionar doctores (agregar, editar, deshabilitar, mostrar).
	RR005	Como usuario con rol administrador necesita generar *endpoints* para: - Gestionar pacientes (agregar, editar, deshabilitar, mostrar).
	RR006	Como usuario con cualquier rol necesita generar *endpoints* para: - Visualizar citas odontológicas.
	RR007	Como usuario con rol administrador necesita generar *endpoints* para: - Agendar citas odontológicas (en caso de que el paciente se acerque al consultorio).
	RR008	Como usuario con rol administrador necesita generar *endpoints* para: - Visualizar historias clínicas.
	RR009	Como usuario con rol doctor necesita generar *endpoints* para: - Gestionar citas odontológicas (no se elimina cita odontológica solo se actualiza)

	RR010	Como usuario con rol doctor necesita generar *endpoints* para: - Registrar horario de atención en el consultorio (horario de disponibilidad).
	RR011	Como usuario con rol doctor necesita generar *endpoints* para: - Gestionar historias clínicas (la acción eliminar no le está permitido).
	RR012	Como usuario con rol paciente necesita generar *endpoints* para: - Seleccionar cita odontológica (agendamiento de cita).
	RR013	Como usuario con rol paciente necesita generar *endpoints* para: - Visualizar la información del consultorio (tipo *landing page*).
	RR014	Como usuario con rol paciente necesita generar *endpoints* para: - Cancelar cita odontológica.
	RR015	Como usuario con rol paciente necesita generar *endpoints* para: - Visualizar su historia clínica.
	RR016	Como usuario con rol administrador necesita generar *endpoints* para: - Gestionar los servicios
	RR017	Servicio de recordatorio de citas: - Recordar a los pacientes y odontólogos a través de la aplicación móvil

Historias de Usuario

Desde la **Tabla 5** hasta **Tabla 20**, se observa las historias de usuario una vez finalizada la recopilación de requerimientos, historias de usuario que se han utilizado para el desarrollo del sistema.

Tabla 5 Historia de Usuario N°1

HISTORIA DE USUARIO	
Identificador (ID): HUB001	**Usuario:** Administrador, paciente y odontólogo
Nombre Historia: *Endpoint* para registrar Usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Jhon Torres	
Descripción: El *backend* permite crear nuevos usuarios en el sistema.	

Observación: Mediante la interfaz de registro de usuario, se puede consumir un *endpoint* para registrar un usuario, pero con rol por defecto de Paciente por defecto.

Tabla 6 Historia de Usuario N°3

HISTORIA DE USUARIO	
Identificador (ID): HUB003	**Usuario:** Administrador, paciente y odontólogo
Nombre Historia: *Endpoint* para obtener y actualizar datos del usuario logeado.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir el *endpoint* para obtener la información del usuario que inicie sesión.	
Observación: Se usa para mostrar los datos en una interfaz de perfil además de actualizar datos desde una interfaz de perfil de usuario.	

Tabla 7 Historia de Usuario N°4

HISTORIA DE USUARIO	
Identificador (ID): HUB004	**Usuario:** Administrador
Nombre Historia: *Endpoints* para gestionar odontólogos (agregar, editar, deshabilitar, mostrar).	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 2	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoints* para el CRUD de odontólogos.	
Observación: El usuario con rol administrador puede administrar los registros sobre odontólogos, tales como crear, leer, actualizar y deshabilitar.	

Tabla 8 Historia de Usuario N°5

HISTORIA DE USUARIO	
Identificador (ID): HUB005	**Usuario:** Administrador
Nombre Historia: *Endpoints* para gestionar pacientes (agregar, editar, deshabilitar, mostrar).	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 2	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoints* para el CRUD de pacientes.	

Observación: El usuario con rol administrador puede administrar los registros sobre pacientes, tales como crear, leer, actualizar y deshabilitar.

Tabla 9 Historia de Usuario N°6

HISTORIA DE USUARIO	
Identificador (ID): HUB006	**Usuario:** Administrador, paciente y odontólogo
Nombre Historia: *Endpoints* para visualizar citas odontológicas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 2	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se ha consumido *endpoint* para mostrar datos necesarios sobre las citas odontológicas.	
Observación: Todos los usuarios dentro del sistema pueden ver las citas odontológicas existentes.	

Tabla 10 Historia de Usuario N°7

HISTORIA DE USUARIO	
Identificador (ID): HUB007	**Usuario:** Administrador
Nombre Historia: *Endpoints* para agendar citas odontológicas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 3	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoint* que permita agendar una cita odontológica.	
Observación: El usuario con rol Administrador puede registrar una cita odontología para un paciente que se acerque personalmente al consultorio.	

Tabla 11 Historia de Usuario N°8

HISTORIA DE USUARIO	
Identificador (ID): HUB008	**Usuario:** Administrador y Odontólogo
Nombre Historia: *Endpoints* para visualizar historia clínica de un paciente.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 5	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoint* que permita mostrar las historias clínicas de los pacientes.	

Observación: El usuario con rol Administrador puede visualizar las historias clínicas de cada paciente que exista en el sistema del consultorio.

Tabla 12 Historia de Usuario N°9

HISTORIA DE USUARIO	
Identificador (ID): HUB009	**Usuario:** Odontólogo
Nombre Historia: *Endpoints* para actualizar datos de cita odontológica.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 3	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoint* que permita actualizar una cita odontológica.	
Observación: El usuario con rol Odontólogo puede actualizar campos de una cita odontología. Esta historia requiere de la existencia de la cita odontológica.	

Tabla 13 Historia de Usuario N°10

HISTORIA DE USUARIO	
Identificador (ID): HUB010	**Usuario:** Odontólogo
Nombre Historia: *Endpoints* para registrar horario de atención en el consultorio (horario de disponibilidad).	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 3	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoint* que permita registrar una cita odontológica.	
Observación: El usuario con rol Odontólogo puede registrar una cita odontología, en la que se evidencia la disponibilidad del odontólogo para la atención a pacientes.	

Tabla 14 Historia de Usuario N°11

HISTORIA DE USUARIO	
Identificador (ID): HUB011	**Usuario:** Odontólogo
Nombre Historia: *Endpoints* para gestionar historias clínicas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 4	
Responsable (es): Jhon Torres	
Descripción: Desde el *Frontend* se puede consumir *endpoint* que permita crear, mostrar y actualizar el registro de historias clínicas.	
Observación: El usuario con rol Odontólogo no puede eliminar una historia clínica de un paciente.	

Tabla 15 Historia de Usuario N°12

HISTORIA DE USUARIO	
Identificador (ID): HUB012	**Usuario:** Paciente
Nombre Historia: *Endpoints* para seleccionar cita odontológica.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 4	
Responsable (es): Jhon Torres	
Descripción: El usuario paciente puede seleccionar la cita odontológica que mejor le convenga para el agendamiento de la cita.	
Observación: Ninguna.	

Tabla 16 Historia de Usuario N°13

HISTORIA DE USUARIO	
Identificador (ID): HUB013	**Usuario:** Paciente
Nombre Historia: *Endpoints* para visualizar la información del consultorio.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Iteración Asignada: 5	
Responsable (es): Jhon Torres	
Descripción: El usuario paciente se le muestra información relevante del consultorio que permita convertir a los visitantes de la página web en potenciales clientes.	
Observación: Ninguna.	

Tabla 17 Historia de Usuario N°14

HISTORIA DE USUARIO	
Identificador (ID): HUB014	**Usuario:** Paciente
Nombre Historia: *Endpoints* para cancelar cita odontológica.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 4	
Responsable (es): Jhon Torres	
Descripción: El usuario paciente puede cancelar una cita odontológica si en caso, el paciente no puede asistir el día agendado.	
Observación: Para cancelar una cita se debe estar agendada por el usuario paciente.	

Tabla 18 Historia de Usuario N°15

HISTORIA DE USUARIO	
Identificador (ID): HUB015	**Usuario:** Paciente
Nombre Historia: *Endpoints* para visualizar su historia clínica.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Iteración Asignada: 5	
Responsable (es): Jhon Torres	

Descripción: El usuario paciente puede visualizar todo su historial clínico dentro del Consultorio.
Observación: Ninguna.

Tabla 19 Historia de Usuario N°16

HISTORIA DE USUARIO	
Identificador (ID): HUB016	**Usuario:** Administrador
Nombre Historia: *Endpoints* para gestionar servicios.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Iteración Asignada: 4	
Responsable (es): Jhon Torres	
Descripción: El usuario administrador puede registrar, leer, actualizar y eliminar los servicios dentro del Consultorio.	
Observación: Forma parte del registro y actualización de historias clínicas.	

Tabla 20 Historia de Usuario N°17

HISTORIA DE USUARIO	
Identificador (ID): HUB017	**Usuario:** N/A
Nombre Historia: Enviar recordatorios a pacientes y odontólogos.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 5	
Responsable (es): Jhon Torres	
Descripción: El servicio de recordatorio envía notificaciones a los dispositivos móviles de del paciente y del odontólogo, cuando una cita está a 2 días de llegar a la fecha de la cita.	
Observación: El recordatorio se envía diariamente iniciando a 2 días antes hasta el día de la cita.	

Product Backlog

En la **Tabla 21**, se observa la tabla completa de *Product Backlog* derivada de las historias de usuario, donde se evidencia la relevancia de los elementos.

Tabla 21 *Product Backlog*

ID-SB	HISTORIA DE USUARIO	ITERACIÓN	PRIORIDAD
HUB001	*Endpoint* para registro de Usuario	1	Media
HUB002	*Endpoint* de Autenticación, Cierre de sesión y Recuperar contraseña.	1	Alta

HUB003	*Endpoint* para obtener y actualizar datos del usuario autenticado	1	Media
HUB004	*Endpoints* para gestionar doctores (agregar, editar, eliminar, mostrar).	2	Alta
HUB005	*Endpoints* para gestionar pacientes (agregar, editar, eliminar, mostrar).	2	Alta
HUB006	*Endpoints* para visualizar citas odontológicas.	2	Alta
HUB007	*Endpoints* para agendar citas odontológicas.	3	Alta
HUB008	*Endpoints* para visualizar historias clínicas.	5	Media
HUB009	*Endpoints* para actualizar datos de cita odontológica	3	Alta
HUB010	*Endpoints* para registrar horario de atención en el consultorio (horario de disponibilidad).	3	Alta
HUB011	*Endpoints* para gestionar historias clínicas.	4	Alta
HUB012	*Endpoints* para seleccionar cita odontológica	4	Alta
HUB013	*Endpoints* para visualizar la información del consultorio.	5	Media
HUB014	*Endpoints* para cancelar cita odontológica	4	Alta
HUB015	*Endpoints* para visualizar su historia clínica	5	Media
HUB016	*Endpoints* para gestionar los servicios	4	Baja
HUB017	Enviar recordatorios a pacientes y odontólogos.	5	Media

Sprint Backlog

En la **Tabla 22**, se observa la tabla completa de *Sprint Backlog*, donde se evidencia los procesos de cada sprint.

Tabla 22 *Sprint Backlog*

ID-SB	NOMBRE	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
S0	Configuración del ambiente del desarrollo	N/A	• Definición de roles • Configuración de herramientas necesarias • Diseño de la Base de Datos	20 h
S1	Autenticación y registro	HUB001: *Endpoint* para registro de Usuario	• Creación de un formulario donde solamente se va a registrar los pacientes • Inicio de sesión • Cierre de sesión • Recuperación de contraseña	40 h
		HUB002: *Endpoint* de Autenticación, Cierre de sesión y Recuperar contraseña.		
		HUB003: *Endpoint* para obtener y actualizar datos del usuario autenticado		
S2	Gestión de usuarios	HUB004: *Endpoints* para gestionar doctores (agregar, editar, eliminar, mostrar).	• Registro de usuarios en el sistema, dentro del panel • Visualización de usuarios del sistema • Deshabilitación de usuarios	30 h
		HUB005: *Endpoints* para gestionar pacientes (agregar, editar, eliminar, mostrar)		
	Citas odontológicas	HUB006: *Endpoints* para visualizar citas odontológicas.	• visualizar citas odontológicas • agendar citas odontológicas	
S3	Citas odontológicas	HUB007: *Endpoints* para agendar citas odontológicas.		30 h
		HUB009: *Endpoints* para actualizar datos de cita odontológica	Odontólogos • actualizar datos de cita odontológica • registrar horario de atención en el consultorio	
		HUB010: *Endpoints* para registrar horario de atención en el consultorio (horario de disponibilidad).		

S4	Citas odontológicas	HUB014: *Endpoints* para cancelar cita odontológica	• Cancelación de cita odontológica	35 h
		HUB012: *Endpoints* para seleccionar cita odontológica	Pacientes • Seleccionar cita odontológica	
	Historias Clínicas	HUB016: *Endpoints* para gestionar los servicios	• crear, leer, actualizar y eliminar servicios	
		HUB011: *Endpoints* para gestionar historias clínicas.	• Crear nueva historia Clínica • Visualizar historias Clínicas • Actualizar historias Clínicas	
S5	Historias Clínicas	HUB008: *Endpoints* para visualizar historias clínicas.	• visualizar historias clínicas	40 h
		HUB015: *Endpoints* para visualizar su historia clínica	• visualizar su historia clínica	
	Landing Page (informativo)	HUB013: *Endpoints* para visualizar la información del consultorio.	• Visualizar información importante sobre el Consultorio	
	Recordatorios a la aplicación móvil	HUB017: Enviar recordatorios a pacientes y odontólogos.	• Implementar servicio de notificación	20 h
S6	Pruebas y Despliegue	N/A	• Pruebas unitarias. • Prueba de carga. • Prueba de *performance*.	5 h
Documentación		N/A	• Trabajo de Integración Curricular. • Anexos.	20 h
			Total =	240 h

Pruebas

A continuación, se presenta las diferentes pruebas que se han realizado. Para la realización de las pruebas se han tomado las funcionalidades “CORE” del negocio.

Pruebas unitarias

A continuación, se muestra las pruebas unitarias que permiten verificar el funcionamiento de los métodos más significativos dentro del sistema.

Método “actualizar historia clínica desde el Odontólogo”

```php
public function test_actualizar_historia_clinica(): void
{
    $user = User::where('identity_card_user', 1314253678)->first();
    $background = [
        "background" => "Hemofilia, diabetes, cáncer, alergias."
    ];
    $request = $this->actingAs($user)->post(sprintf('/api/updateMedicalRecord/1314253678'), $background);
    $request->assertStatus(200);
}
```

Figura 44 Prueba unitaria actualizar historia clínica

```
PASS  Tests\Unit\MedicalRecordTest
✓ actualizar historia clinica

Tests:    1 passed (1 assertions)
Duration: 6.42s
```

Figura 45 Resultado de la prueba unitaria actualizar historia clínica

Método “crear detalle historia clínica desde el Odontólogo”

```php
public function test_crear_detalle_historia_clinica(): void
{
    $user = User::where('identity_card_user', 1314253678)->first();
    $odontograma = [
        46 =>[
            "part_up" => "diagnostico"
        ]
    ];
    $details = [
        "reason" => "Control 1",
        "current_illness" => "Caries",
        "odontogram" => json_encode($odontograma)
    ];
    $request = $this->actingAs($user)->post(sprintf('/api/createRecordDetail/1314253678'), $details);
    $request->assertStatus(201);
}
```

Figura 46 Prueba unitaria de crear detalle de historia clínica

```
PS C:\xampp\htdocs\Odontologica> php artisan test

  PASS  Tests\Unit\MedicalRecordTest
  ✓ crear detalle historia clinica

  Tests:    1 passed (1 assertions)
  Duration: 3.07s
```

Figura 47 Resultado de la prueba unitaria de crear detalle de historia clínica

Método “obtener historia clínica desde el Odontólogo”

```php
public function test_obtener_historia_clinica() : void
{
    $user = User::where('identity_card_user', 1314253678)->first();
    $request = $this->actingAs($user)->get('/api/getMedicalRecordUser/1314253678');
    $request->assertStatus(200);
}
```

Figura 48 Prueba unitaria de obtener historia clínica

```
PS C:\xampp\htdocs\Odontologica> php artisan test

  PASS  Tests\Unit\MedicalRecordTest
  ✓ obtener historia clinica

  Tests:    1 passed (1 assertions)
  Duration: 4.33s
```

Figura 49 Resultado de la prueba unitaria de obtener historia clínica

Pruebas de carga

Para el *endpoint* de crear cita médica con un dentista, el consultorio cuenta con 3 odontólogos, las pruebas se han realizado con un número de aproximación a triplicar el número de los odontólogos existentes. Obteniendo que por 8 odontólogos que usen el *endpoint* de crear cita médica, 5 peticiones en simultaneo, el tiempo menor en carga es de 1574 ms y el tiempo máximo es de 1622 ms, mostrado en la **Figura 50**. Y un tiempo de 3.765 segundos tomado para completar la prueba.

```
jhon@Equipo:~$ ab -n 8 -c 5 -k https://endpointsco-production.up.railway.app/api/createAppointDentist
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking endpointsco-production.up.railway.app (be patient).....done


Server Software:        railway
Server Hostname:        endpointsco-production.up.railway.app
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-CHACHA20-POLY1305,4096,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        endpointsco-production.up.railway.app

Document Path:          /api/createAppointDentist
Document Length:        0 bytes

Concurrency Level:      5
Time taken for tests:   3.765 seconds
Complete requests:      8
Failed requests:        0
Non-2xx responses:      8
Keep-Alive requests:    0
Total transferred:      1000 bytes
HTML transferred:       0 bytes
Requests per second:    2.12 [#/sec] (mean)
Time per request:       2353.213 [ms] (mean)
Time per request:       470.643 [ms] (mean, across all concurrent requests)
Transfer rate:          0.26 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:      436  445  10.4    444     466
Processing:  1137 1145   5.9   1144    1157
Waiting:      139  143   3.9    143     151
Total:       1574 1589  15.9   1586    1622
```

```
Percentage of the requests served within a certain time (ms)
  50%   1586
  66%   1597
  75%   1597
  80%   1597
  90%   1622
  95%   1622
  98%   1622
  99%   1622
 100%   1622 (longest request)
```

Figura 50 Prueba de carga 2

Para el *endpoint* de obtener cita médica especifica, se ha realizado una estimación de 400 usuarios tratando de recuperar 2 veces una cita especifica en simultaneo. Obteniendo lo siguiente, en tiempos, como tiempo mínimo el valor de 1564 ms y como tiempo máximo el valor de 3369 ms, se muestra en la **Figura 51**. Y un tiempo de 329.123 segundos tomado para completar la prueba.

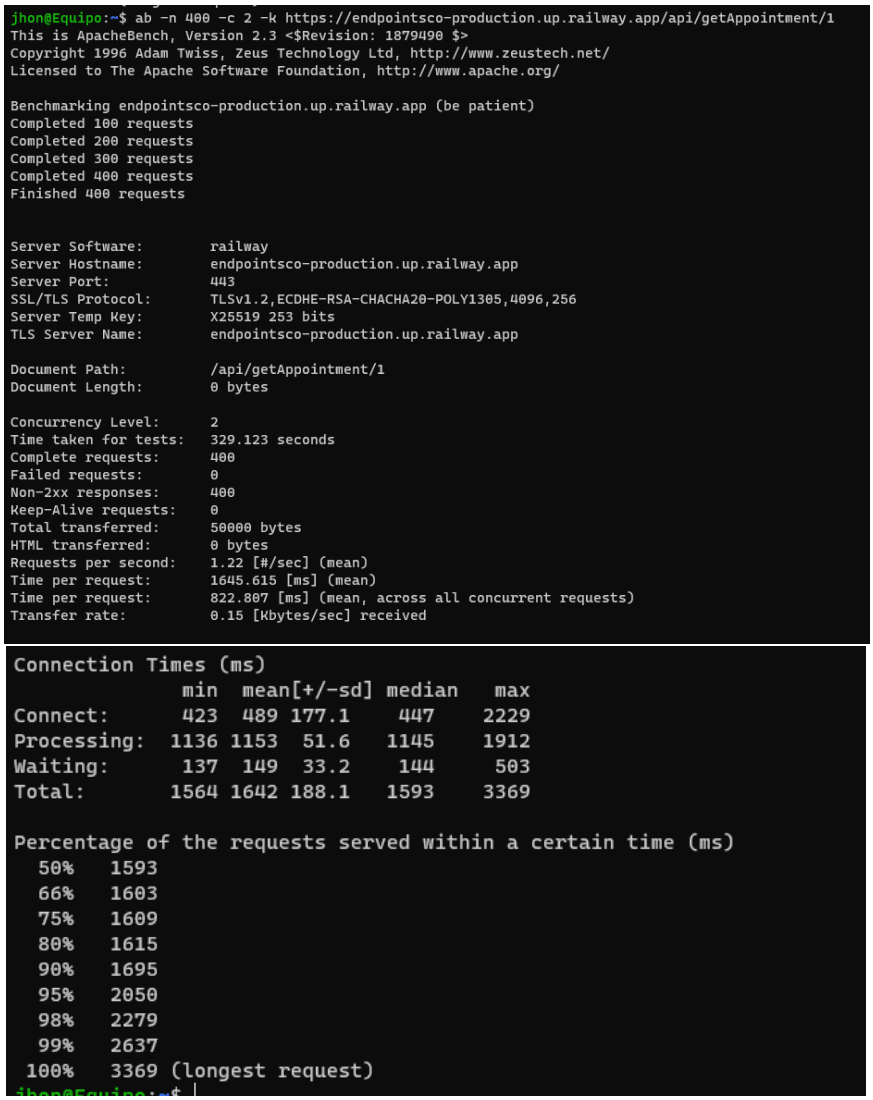

Figura 51 Prueba de carga 3

Para el *endpoint* de obtener citas médicas según dentista, se ha realizado con una estimación de 600 usuarios realizando una petición. Obteniendo que, el tiempo mínimo de 1560 ms y un tiempo máximo de 4426 ms, **Figura 52**. Adicional a el tiempo de 1010.545 segundos tomado para completar la prueba.

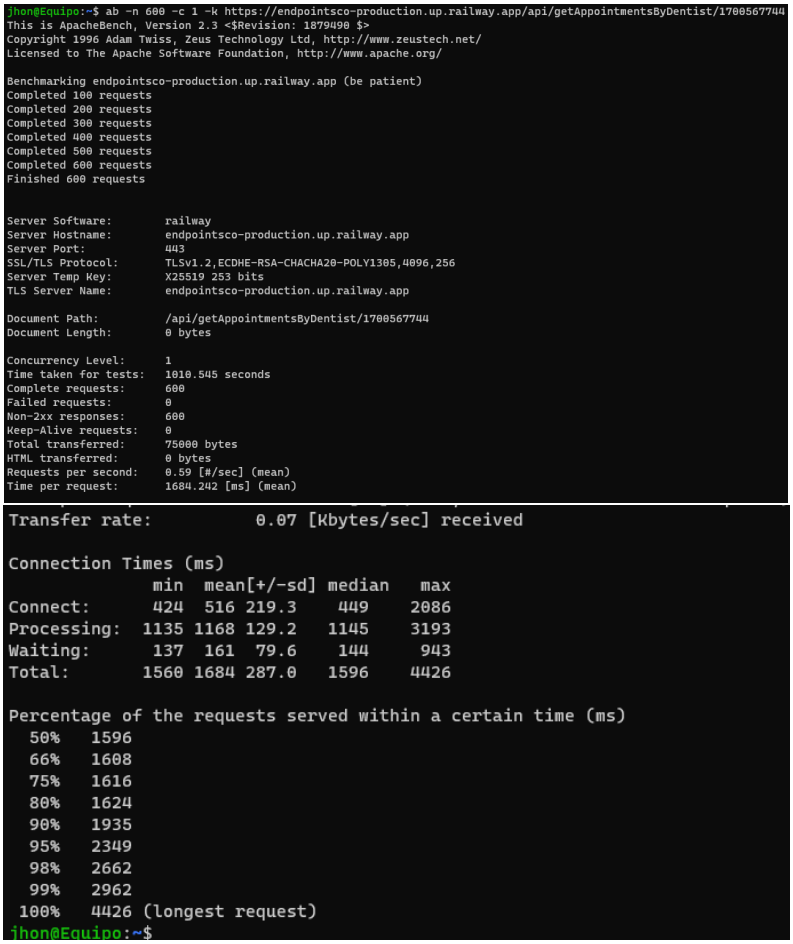

Figura 52 Prueba de carga 4

Pruebas de performance

Para el recurso de memoria del sistema existe una variación considerable entre un rango mínimo de 50 MB hasta un rango máximo de 300MB de un total de 450 MB. Ver la **Figura 53**.

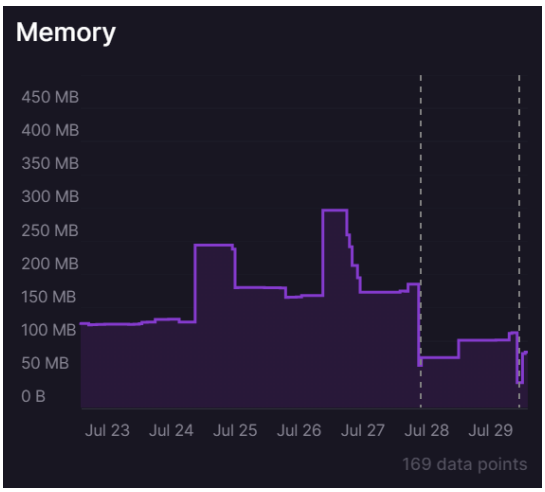

Figura 53 Métrica de memoria

El recurso de Red es el que más varía, debido a que es dependiendo de la conexión al momento de realizar la petición. *Railway* muestra la siguiente variación de la red en la **Figura 54**.

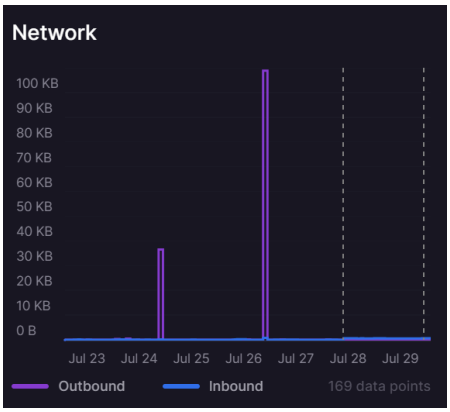

Figura 54 Métrica de Red

Errores sobre el uso de Task Schedule en Railway.

En base a la implementación de Task Schedule en el proyecto de forma local se ejecuta el comando "**php artisan schedule:work"** sin ninguna afectación, al contrario del despliegue a producción.

Se presenta que al desplegar el proyecto y luego usar el comando, "**php artisan schedule:work",** necesario para ejecutar la tarea de envío de recordatorios en el servidor del proveedor *Railway,* ver **Figura 55.**

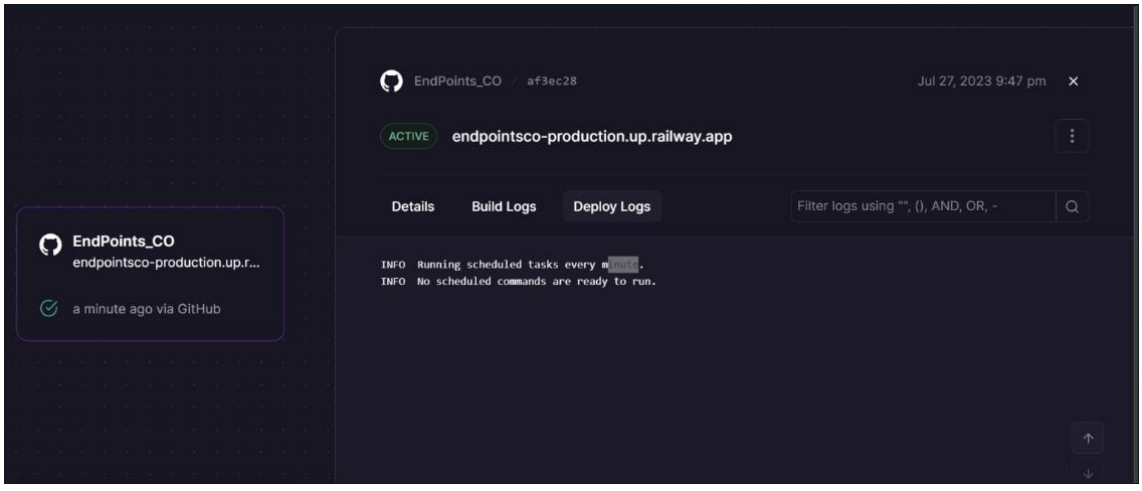

Figura 55 Despliegue y comando de Task Schedule

Se obtiene un error 503 en el servicio, **Figura 55**, dificultando el correcto funcionamiento de las APIs del proyecto, ver la **Figura 56.**

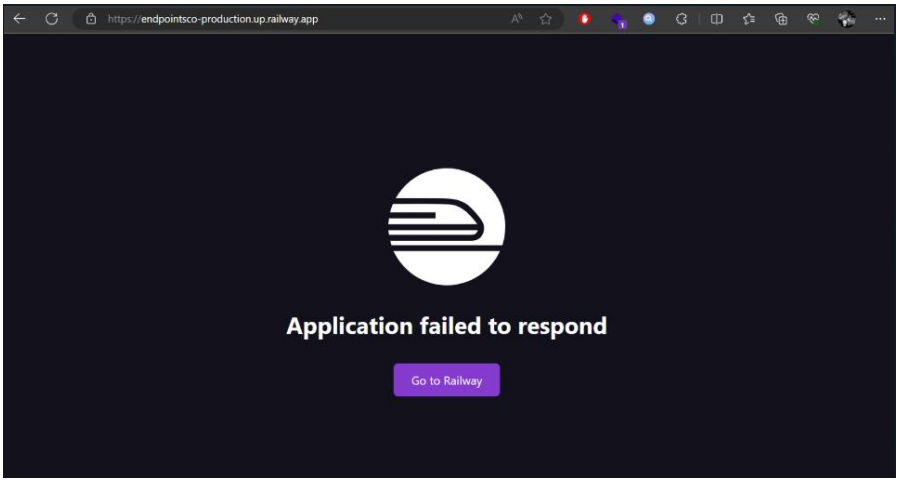

Figura 56 Error en despliegue

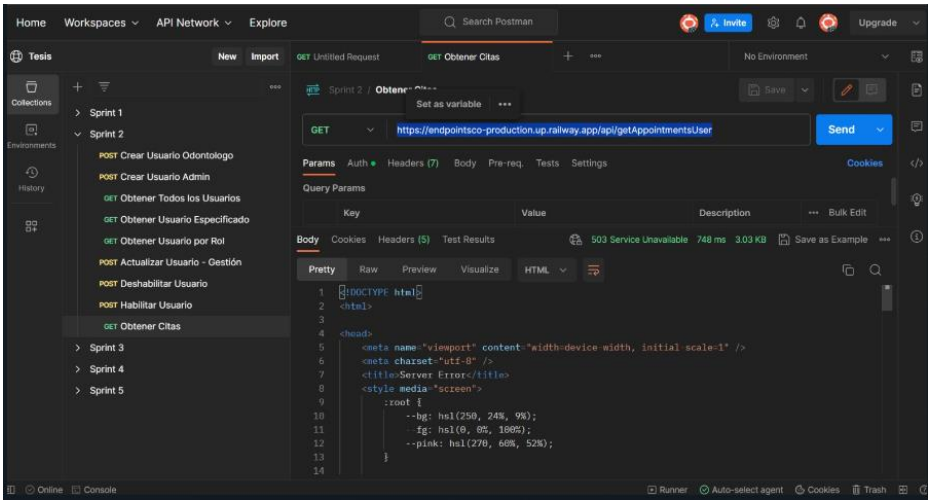

Figura 57 Error en el consumo de *endpoints*

Solución al envío de recordatorios de citas en *AlwaysData*.

Se propuso la siguiente solución para el envío de recordatorios de citas a los dispositivos móviles de los usuarios. Que la lógica ya creada en el cron job, sea colocada en un *endpoint*, con característica de ser ruta pública.

AlwaysData permite crear tareas que se ejecuten dado la frecuencia de ejecutar algún comando o acceder a una URL.

Permitiendo crear una tarea automatizada para acceder al *endpoint* de envío de recordatorios, como se muestra a continuación en la **Figura 58**

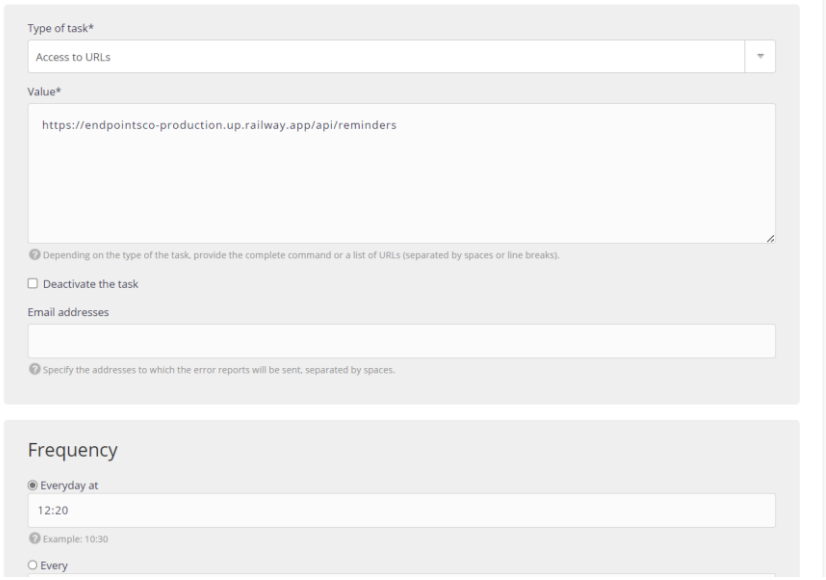

Figura 58 Tarea programada en *AlwaysData*

ANEXO III

El enlace que se presenta a continuación, contine el video del manual de usuario del componente *backend* del sistema web.

https://www.youtube.com/watch?v=zHoP9nJ1GBY

ANEXO IV

A continuación, se presentan las credenciales “tipo prueba” de cada rol existente en el *backend*, el enlace al código fuente (repositorio *GitHub*, donde se encuentra el README que indica los pasos para desplegar de manera efectiva el *backend*).

Documentación de las APIs

A continuación, se comparte el enlace en donde se organiza las APIs según funciones de usuario, citas médicas, historias clínicas, servicios y landing page.

https://miro.com/app/board/uXjVM-SE2Ss=/?share_link_id=894202684543

Adicionalmente, se presenta el enlace en donde se evidencia el consumo de cada API del sistema organizados por *Sprint,* en la herramienta *POSTMAN.*

https://www.postman.com/orange-escape-919790/workspace/tesis

Finalmente, el enlace al repositorio del código fuente de la aplicación, alojado en *Github.*

jhon-torres/EndPoints_CO: APIs Consultorio (github.com)

Credenciales

Credenciales usuario administrador:

- Correo: admin@epn.edu.ec
- Contraseña: password

Credenciales usuario odontólogo:

- Correo: gabo@epn.edu.ec
- Contraseña: password

Credenciales usuario administrador:

- Correo: jhon64t@gmail.com
- Contraseña: password