ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA DE GESTIÓN DE CITAS MÉDICAS PARA EL CENTRO DE TERAPIAS ALTERNATIVAS TERMO OASIS

BACKEND

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

EDUARDO SEBASTIAN ALMACHI MAISINCHO

DIRECTOR: JUAN PABLO ZALDUMBIDE PROAÑO

DMQ, agosto 2024

CERTIFICACIONES

Yo, Eduardo Sebastian Almachi Maisincho 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.

EDUARDO SEBASTIAN ALMACHI MAISINCHO

eduardo.almachi@epn.edu.ec

eduardoalmachi123@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Eduardo Sebastian Almachi Maisincho, bajo mi supervisión.

Ing. Juan Pablo Zaldumbide Proaño

DIRECTOR

juan.zaldumbide@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.

Eduardo Sebastian Almachi Maisincho

DEDICATORIA

Este trabajo está dedicado a mi familia directa Mónica, Medardo y Selena por el apoyo que me han dado desde un principio, también lo dedico a mi tía Sara por el apoyo que me ha dado en ciertos aspectos. Además, no me olvido de mi abuelita Inés, mi tío Santi y mi familia que está lejos de mi pero que siempre me han apoyado desde un principio. Sin duda, son valiosos en mi vida. Este proyecto y todo lo que he hecho hasta ahora no sería posible sin ustedes mil gracias.

AGRADECIMIENTO

Quiero dejar por escrito mi agradecimiento a mi familia por apoyarme en todos los sentidos a seguir y a no darme por vencido a pesar de cualquier cosa.

Quiero además agradecer a mi dios Jehová por no soltarme en los momentos difíciles que pase en esta temporada.

Asimismo, expresar mi agradecimiento a mi tutor por haber sido haber sido una guía en este proyecto.

Finalmente, quiero agradecer a todos los que me apoyaron de alguna manera desde los más pequeño hasta los más grande, mil gracias.

ÍNDICE DE CONTENIDO

RESUMEN

Tras la pandemia, el sector salud ha adoptado tecnologías avanzadas para solventar a nuevas demandas y enriquecer la interacción del usuario. En este contexto, TERMO OASIS, reconocido por sus servicios quiroprácticos en Quito, ha identificado la urgente necesidad de optimizar la gestión de citas y registros médicos. La carencia de un sistema adecuado ha causado sobrecarga de pacientes y conflictos en la agenda, afectando la calidad del servicio ofrecido.

Para abordar estos desafíos, se propone el desarrollo de un backend personalizado que gestione eficientemente la información de clientes y registros médicos. Este sistema garantiza alto rendimiento, escalabilidad y seguridad óptimos, permitiendo a TERMO OASIS mejorar significativamente su operatividad y posicionarse como líder en el sector de terapias alternativas.

El proyecto de integración curricular se estructura en cinco capítulos clave. El primero detalla la información inicial del proyecto. El segundo capítulo se centra los requerimientos previos, incluyendo la arquitectura diseñada para soportar las necesidades del centro. El tercer capítulo analiza detalladamente los sprints realizados, destacando el progreso y los resultados obtenidos. Las conclusiones derivadas del estudio se presentan en el cuarto capítulo, mientras que el quinto capítulo ofrece recomendaciones detalladas y retroalimentación para futuras mejoras del proyecto.

Este enfoque no solo fortalece la operativa diaria de TERMO OASIS, sino que también asegura una respuesta ágil y efectiva a los cambios tecnológicos y las expectativas del mercado, mejorando la sensación individual del servicio ofrecido.

PALABRAS CLAVE: Backend, JavaScript, Express, Gestión, NodeJs, API.

ABSTRACT

Following the pandemic, the health sector has adopted advanced technologies to adapt to new demands and optimize the customer’s journey . In this context, TERMO OASIS, recognized for its chiropractic services in Quito, has identified the urgent need to optimize Managing appointments and healthcare documentation . The lack of an adequate system has caused patient overload and conflicts in the agenda, affecting the quality of service offered.

To address these challenges, we propose the development of a customized backend that efficiently manages customer information and medical records. This system guarantees high performance, scalability and optimal safety, allowing TERMO OASIS to significantly improve its operability and position itself as a leader in the sector of alternative therapies.

The curricular integration project is structured in five key chapters. The opening section delineates the project's precise aims, scope, and theoretical framework . The next chapter focuses on the proposed approach for software development, including framework crafted to accommodate the center's demands . The third chapter analyses in detail the sprints performed, highlighting the progress and results obtained. The study's results are outlined in the fourth chapter, while the fifth chapter provides detailed recommendations and feedback for future project improvements.

This approach not only strengthens the daily operation of TERMO OASIS, but also ensures an agile and effective response to technological changes and market expectations, improving the overall customer experience and the efficiency of the service offered.

KEYWORDS: Backend, JavaScript, Express, Management, NodeJs, API.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En las circunstancias actuales muchos negocios han tenido que adaptarse a los cambios postpandemia. Para adaptarse a estos cambios y necesidades de los clientes, el ámbito de la salud ha tenido que ir a la par de las nuevas tecnologías. De hecho, es fundamental que lugares como centros dedicados a ofrecer servicios de salud implementen sistemas que le permitan gestionar eficientemente estos centros y servicios que ofrecen. Debido a que hay nuevas demandas, esto no solo beneficia al negocio si no que mejora la vivencia de los usuarios y la excelencia de los servicios que proporcionan [1].

El Centro de Terapias Alternativas TERMO OASIS ubicado en el Sur de Quito, reconocido por la calidad de sus servicios quiroprácticos, se encuentra ante la demanda de implementar una solución eficiente que optimice la organización de sus citas y registros médicos. La falta de un método adecuado para gestionar esta información ha generado inconvenientes como la saturación de clientes en el establecimiento, conflictos en el horario del especialista y cruces de citas. Estas situaciones afectan negativamente la calidad del servicio brindado, ocasionando molestias a los pacientes y dificultando la correcto ejecución de las actividades llevadas a cabo. Con base a estas problemáticas el especialista a cargo del centro de terapias alternativas TERMO OASIS se ha visto en la necesidad de poder encontrar soluciones a estas problemáticas. Para afrontar estos desafíos, se desarrolla una aplicación de gestión, la cual ofrece soluciones a las distintas problemáticas que presentan.

Por lo tanto, se ofrece como solución la creación de un backend, con el fin de abordar las problemáticas. Este sistema permite gestionar de manera integral la información de los clientes, las citas programadas y los registros médicos de cada paciente. El componente backend del sistema de gestión de citas para TERMO OASIS se desarrolla utilizando tecnologías de vanguardia, garantizando un alto rendimiento, escalabilidad y seguridad. Esta arquitectura permite la gestión eficiente de usuarios, citas y registros médicos. La elección estratégica de herramientas tecnológicas, junto con la implementación de las prácticas más efectivas del sector , asegura no solo la capacidad del sistema para atender las necesidades inmediatas del centro de terapias, sino también para adaptarse o evolucionar ante los desafíos y demandas cambiantes del mercado y los usuarios. Este enfoque posiciona a TERMO OASIS a una mejor ventaja en su sector al ofrecer una solución tecnológica innovadora y efectiva, capaz de mejorar significativamente a vivencia del cliente y la efectividad operativa del centro, alineándose así con los estándares más altos de la industria.

1.1 Objetivo general

Desarrollar un sistema de gestión de citas médicas para el centro de terapias alternativas Termo Oasis.

1.2 Objetivos específicos

1. Determinar los requerimientos para el desarrollo del backend.
2. Elaborar el modelo de base de datos.
3. Desarrollar los endpoints.
4. Realizar pruebas de funcionalidad.
5. Publicar el backend en el entorno de producción.

1.3 Alcance

La programación de este backend tiene como objetivo llegar a los clientes y ofrecer una mejor experiencia de gestión de citas en TERMO OASIS. Por tal motivo, el componente backend es un sistema completo que proporciona una API RESTful. Al cumplir con los objetivos específicos establecidos, desde la determinación de requisitos hasta la publicación en producción, se asegura que el backend de gestión de citas sea como se pretende [2]. El alcance de este proyecto incluye:

- Desarrollo del backend
    - Desde el análisis inicial de requerimientos hasta la implementación concluyente.
- Modelado de BD
    - Diseño y creación del esquema de datos, asegurando una arquitectura robusta y escalable. La base de datos no relacional a utilizar es MongoDB que almacena los datos de las citas, registros médicos y usuarios.
- Implementación de endpoints
    - Citas: Creación de rutas para crear, leer y actualizar citas,.
    - Usuarios: Creación de rutas para registrar, autenticar, leer, actualizar, y eliminar usuarios.
    - Registros: Creación para crear, leer, actualizar y listar registros.

- Asignación de roles
    - Paciente: este rol puede visualizar citas y cancelar citas.
    - Secretaria: este rol puede visualizar, actualizar, cancelar citas y administrar usuarios.
    - Medico: este rol puede visualizar citas; y visualizar, crear y editar registros médicos
    - Acciones en conjunto: los tres roles en conjunto pueden hacer login y recuperar credenciales de acceso.
- Pruebas de funcionalidad
    - Realización de pruebas unitarias, estrés y carga para asegurar el adecuado rendimiento de toda la infraestructura backend.
- Documentación
    - Creación de documentación completa y detallada del API, incluyendo guías para desarrolladores y usuarios finales.

Con este alcance, se pretende obtener un resultado final que proporcione a TERMO OASIS un backend completo y eficiente.

1.4 Marco Teórico

Es la parte donde se describe las bases para entender y abordar cualquier tema de manera fundamentada. Esta sección describe los conceptos claves relevantes relacionados al estudio, además es una base inicial para tener las ideas claras en un proyecto [3]. El desarrollo de un componente backend incluye herramientas, detalles y conocimientos a tomar en cuenta para que funcione correcta y eficientemente. Por eso en esta sección se detallan ideas fundamentales que se aplican en este Trabajo de Integración Curricular [4].

Node.js

NodeJs se caracteriza por ser una excelente herramienta al momento de desarrollar un componente backend escalable y eficiente. Node.js permite que los desarrolladores manejen múltiples tareas simultáneamente sin bloquear procesos principales. Esto hace que sea ideal para un componente backend puesto que ofrece una bastantes bibliotecas y frameworks que facilitan la creación de componentes backend. Estas facilidades la convierten en una opción viable para manejar proyectos escalables. [5]

MongoDB

Es una herramienta para manejar y de almacenar datos. Posee una arquitectura que permite manejar datos de manera veloz, escalable y eficiente. MongoDB almacena los datos por medio de JSON para manejar los datos de manera dinámica o constante cambio. Este gestor de datos ofrece, agilidad, velocidad, escalabilidad y facilidad. Por eso MongoDB es una opción viable. [6]

Endpoint

En términos generales es un punto de dirección la cual se permite recibir o mandar datos en una url. Cada endpoint tiene una dirección y un identificador único que lo distingue de otros. Los datos se suelen intercambiar por medio de JSON y para su comunicación se utiliza protocolos, en este caso HTTP. [7]

2 METODOLOGÍA

La metodología se basa en seguir pasos definidos previamente para cumplir con un objetivo en específico. Este enfoque permite desarrollar el componente de manera organizada, progresiva, priorizando que funcionalidades son necesarias. Además, facilita la realización de pruebas y ajustes para alcanzar objetivos predefinidos [8].

Dentro de la metodología se utiliza la herramienta llamada estudio de caso siendo esta es una herramienta fundamental. El estudio de caso permite identificar patrones, requerimientos y necesidades. Este enfoque proporciona una mejor vista al problema a resolver, permitiendo una toma de decisiones basadas y fundamentadas en el análisis previo de detalles o contextos [9].

2.1 Metodología de Desarrollo

El mundo tecnológico ha tenido un cambio increíble, lo cual se refleja en como se realizan actualmente las gestiones de proyectos de software. Debido a esto, siempre se debe tomar en cuenta que metodología utilizar y cual no. Por eso, este proyecto utiliza una metodología de desarrollo diseñada habitualmente para la creación de un software, con el objetivo de realizar este proyecto de manera eficiente. [10]

Ahora bien, tomando en cuenta lo mencionado Scrum se elige para esta componente. Esta metodología es una de las más utilizadas en la actualidad, ya que tiene como objetivo garantizar resultados y proporcionar lo que busca el cliente. Se enfoca en la cooperación, mejora continua y la entrega paso a paso. En Scrum, los equipos laboran basados en periodos breves conocidos como sprints, que suelen extenderse de dos a cuatro semanas, con el objetivo de crear un incremento del producto. El proceso incluye roles definidos previamente.

Roles

Esta metodología se basa en roles, cada uno de estos desempeña diferentes tareas para el resultado del producto final.

Product Owner

El Product Owner, o Propietario del Producto, es el representante del negocio o del producto. Su función principal es comunicar las tareas a realizar del producto al development team. Toda la elaboración del producto se basa en las directrices y requisitos proporcionados por el Product Owner, quien organiza y prioriza las tareas del proyecto de acuerdo con las necesidades del negocio o del producto. [11]

Scrum Master

Se encarga de la gestión del equipo, organizándolo y asegurándose que todos los miembros comprendan claramente los objetivos del proyecto [12]. Su función principal es garantizar que todos los sprints se completen sin problemas. Además, establece una planificación adecuada para que el equipo tenga una visión clara del producto y está en constante comunicación con lo que requiere el propietario para definir los requisitos del proyecto de manera precisa y detallada. [13]

Development Team

Son los individuos que trabajan desempeñando roles específicos dentro del equipo. Cada uno de ellos asume el rol correspondiente previamente mencionado [14].

La **Tabla 2.1** detalla la designación especifica de roles.

Tabla 2.1 Designación de roles

ROLES	NOMBRES
Product Owner	Oswaldo Avilez
Scrum Master	Ing. Juan Pablo Zaldumbide
Development Team	Eduardo Almachi

Artefactos

En Scrum, los artefactos no se limitan solo a documentos, sino que son herramientas cruciales para la planificación, seguimiento y ejecución del proyecto. Estos ayudan a gestionar de manera efectiva el progreso y los requerimientos del producto. Los principales artefactos que se abordan en este proyecto son: Sprint Backlog, Product Backlog, Recopilación de Requerimientos, por ende, cada artefacto se encarga de la organización del proyecto [15].

Recopilación de Requerimientos

Aquí se definen los requerimientos del producto solicitado que sirven como guía para la creación del proyecto. Esta recopilación sirve como guía para que el development team pueda realizar el producto de acuerdo con lo requerido [16]. En la **Tabla 2.2** se muestra un ejemplo del formato que tiene la recopilación de requerimientos.

Además, en la sección de **ANEXOS II** se encuentra el listado completo de los requerimientos.

Tabla 2.2 Formato de recopilación de requerimientos

Recopilación de requerimientos		
Tipo de Sistema	**ID-RR**	**ENUNCIADO DEL ÍTEM**
Nombre del componente	Numero identificador del requerimiento	Descripción del requerimiento

Historias de Usuario

Permiten tener en claro los objetivos y requisitos de lo que requiere el usuario de un manera eficaz y breve [17]. A continuación, en la **Tabla 2.3** detalla el formato de Historias de usuario. Además, en el **ANEXOS II** se encuentra listado todas las restantes.

Tabla 2.3 Formato de Historias de usuario

HISTORIA DE USUARIO	
Identificador(ID): Numero identificador	**Usuario:** Usuario de esta acción
Nombre Historia: Nombre de la historia	
Prioridad Negocio: Prioridad(alta/media/baja)	**Riesgo en desarrollo:** Riesgo(alta/media/baja)
Interacción Asignada: Número de interacción	
Responsable: Nombre del responsable	
Descripción: Descripción de la historia de usuario	
Observación: Observación de la historia de usuario	

Product Backlog

Se utiliza para saber que se debe priorizar y que parte del proyecto tiene más valor. Esto ayuda a determinar que debe entregar el equipo en primer lugar o a que darle mayor prioridad [18].

En la **Tabla 2.4** se puede visualizar un ejemplo de formato de este artefacto. Véase el **ANEXOS II** el listado de todo el Product Backlog.

Tabla 2.4 Ejemplo de formato - Product Backlog

ID-HU	Historia de Usuario	Prioridad	Iteración	Estado
Número identificador de la historia de usuario	Nombre de la historia de usuario	(Baja, Media, Alta)	Numero de iteración	(Pendiente, Terminada)

Sprint Backlog

Este artefacto permite que el development team visualice detalladamente que se debe hacer en cada sprint. Aquí se detalla las horas, tareas, tiempo estimado e historia de usuario de la cual proviene cada sprint. En general, el Sprint Backlog, lista específicamente cada avance [19]. En la **Tabla 2.5** se visualiza un ejemplo del formato. Además, en el **ANEXOS II** se visualiza la lista completa.

Tabla 2.5 Ejemplo de formato – Sprint Backlog

ID-SB	NOMBRE	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO
Número identificador	Nombre	Número de historia de usuario	Nombre de la historia de usuario	Numero de tareas	Tiempo especificado en horas

2.2 Diseño de la arquitectura

Es la estructura y diseño con la cual se basa todo el software. Este proceso no solo define la disposición física de los componentes y sus interacciones, sino que también establece las decisiones estratégicas que guían el desarrollo y la planificación del software [20]. Existen muchas arquitecturas de software que permiten crear y planificar un proyecto, por eso cada arquitectura depende del componente que se realiza y que característica cumple. [21].

Patrón Arquitectónico

Para la realización del componente Backend se utiliza el modelo de diseño de software llamado: modelo, vista, controlador (MVC) [22]. Esta arquitectura consiste en lo siguiente:

- **Modelo:** Aquí se especifican los datos que contiene la aplicación, incluyendo campos y tablas que maneja cada módulo.
- **Vista:** Aquí se define el resultado, es decir, la respuesta que se da al solicitar alguna petición del backend.
- **Controlador:** Aquí se implementan todos los servicios, configuraciones y lógicas que modifican tanto el modelo como la vista.

La implementación de esta arquitectura ayuda a que este componente se escalable y pueda manipularse fácilmente para nuevas implementaciones [23].

En la **Figura 2. 1** se explica la funcionalidad de este patrón arquitectónico de software de manera simple.

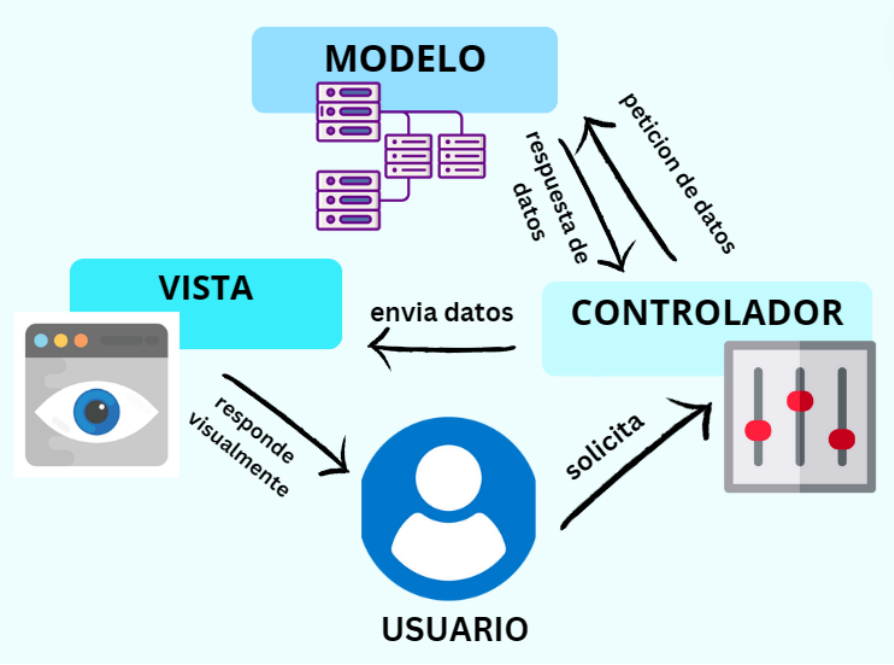

Figura 2. 1 Funcionalidad de MVC

2.3 Herramientas del desarrollo

Para realizar el componente backend se utilizan diferentes herramientas y tecnologías. Debido a esto, la elección de herramientas o tecnologías siempre depende del resultado

esperado, por eso, para este componente y en vista que se utiliza el patrón de arquitectura MVC se utiliza herramientas y librerías que van de acorde con el componente requerido.

Herramientas

Son programas que nos ayudan a realizar diversas tareas de manera más eficiente. La **Tabla 2.6** muestra a detalle las herramientas utilizadas.

Tabla 2.6 Herramientas utilizadas en el backend

Herramienta	Justificación
VSCode	Es un IDE que se puede personalizar ampliamente que brinda la posibilidad de escribir y depurar código [24].
ThunderClient	Es una extensión para Visual Studio Code que facilita la realización de pruebas de integración al consumir los endpoints de una API directamente desde propio entorno [25].
Vercel	Es servicio en la nube (PaaS) donde se puede subir y desplegar cualquier tipo de software web [26].
MongoDB	Es un plataforma de almacenamiento de información que de manera flexible y escalable guarda los datos en formato de objetos o JSON, facilitando su implementación a través de documentos [6].
JavaScript	Se trata de un lenguaje de desarrollo orientado a la creación de todo tipo de aplicaciones, conocido por su versatilidad en el manejo de diversos paradigmas y esquemas de desarrollo web [27].
Swagger	Es una utilidad que posibilita la creación documentación de un API de manera eficiente y rápida [28].
GitHub	Es una plataforma que facilita el control de versiones de proyectos de software [29].

Librerías

Las librerías ofrecen servicios, ayudas, implementaciones y funciones adicionales para la mejor funcionalidad del código. En la **Tabla 2.7** se detalla que librerías que se han utilizado.

Tabla 2.7 Librerías utilizadas

Librería	Justificación
nodemailer	Esta biblioteca proporciona la capacidad de enviar correos electrónicos de forma automatizada y en diversos formatos [30].
dotenv	Esta librería ofrece la funcionalidad de trabajar con variables de entorno en el código [31].
mongoose	Se trata de una librería ODM (Object-Document Mapping) que permite realizar consultas específicas sin necesidad de el lenguaje de consulta estructurado (SQL) [32].
jsonwebtoken	Esta librería permite crear tokens personalizados y seguros para la autenticación de cualquier componente en una lógica del código [33].
express	Express es un framework para el desarrollo del lado del servidor en JavaScript que facilita la creación y organización de APIs (Application Programming Interface) [34].
cors	Se trata de una biblioteca que permite compartir recursos entre dos sitios web o interacciones mediante el manejo de políticas de acceso CORS [35].
date-fns	Es una librería para formatear fechas en español y en la estructura deseada [36].
swagger-jsdoc	Es una librería que permite generar documentación de una API [37].

swagger-ui-express	Es un middleware que permite mostrar la documentación de una API en una interfaz web [38].
node-cron	Es una librería de JavaScript que permite programar tareas para que se ejecutan en momentos específicos [39].
jest	Jest es un marco de pruebas diseñado específicamente para facilitar el proceso de pruebas en JavaScript, enfocado a partes del código o funcionalidades en específico del software [40].
supertest	Es una librería para Node.js que ayuda a testear APIs. Esta mismo verifica las solicitudes HTTP que se hacen por medio de la API [41].

3 RESULTADOS

Este apartado del informe, se presentan los logros alcanzados en cada respectivo sprint junto con su respectiva evidencia, proporcionando así una visión clara y progresiva de los logros alcanzados [42]. Cada sprint se ha descrito con base a lo mencionado, gracias a este enfoque se puede tener una comprensión detallada de la evolución del proyecto en cada etapa, así como una evaluación precisa del progreso alcanzado en cada sprint.

Sprint 0. Configuración del ambiente de desarrollo

Esta sección se basa en estos propósitos a cumplir:

- Creación del diseño y la estructura del modelo de base de datos.
- Establecer la estructura del entorno de desarrollo.
- Determinación de roles.

Creación del diseño y la estructura del modelo de base de datos

Para la implementación de esta parte se utiliza MongoDB, creando la estructura del proyecto. Al utilizar MongoDB, una base de datos no relacional se implementa un enfoque que permite almacenar y gestionar los datos a través de campos y valores, es por eso que permite tener un backend escalable y eficiente [43]. En la **Figura 3.1** se muestra las colecciones que se utilizan en este componente.

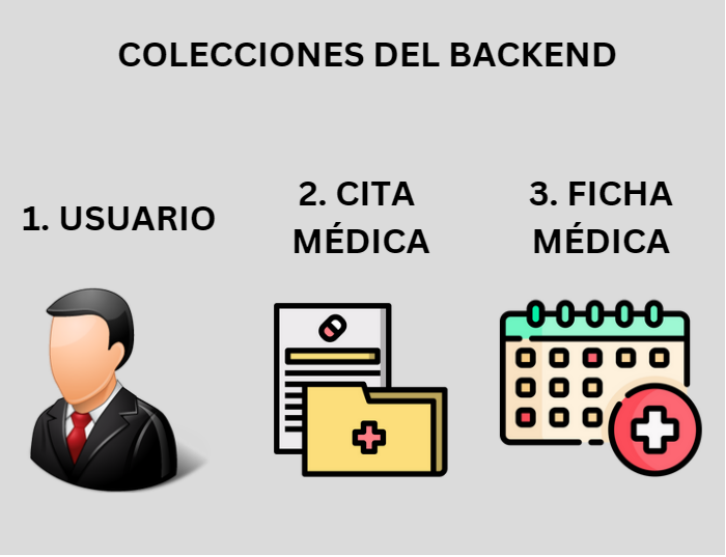

Figura 3.1 Colecciones de datos

Con base al modelo de datos, se ha procedido a crear las colecciones que están diseñadas de acuerdo con la estructura definida, lo que permite una integración entre la lógica y el

acceso a los datos. Esto asegura que el componente backend pueda solicitar o responder a las peticiones siguiendo así las mejores prácticas de desarrollo y manteniendo la consistencia de los datos a lo largo de la aplicación. En el **ANEXO II** se evidencia la estructura de las colecciones de datos.

Establecer la estructura del entorno de desarrollo

MVC (Modelo-Vista-Controlador), se ha establecido como estructura para el componente backend. Esta estructura incluye las carpetas correspondientes a modelos, controladores, rutas, así como archivos iniciales necesarios para iniciar el desarrollo de un componente backend utilizando Node.js y Express.

Estos archivos iniciales suelen incluir configuraciones importantes, como la configuración del servidor, conexión a la base de datos y a los archivos de seguridad necesarios para el manejo de solicitudes HTTP [44]. Esta estructura proporciona un marco organizativo sólido que facilita la escalabilidad, mantenibilidad y colaboración en el desarrollo del proyecto [45]. Esto se puede ver a más a detalle en la **Figura 3.2.**

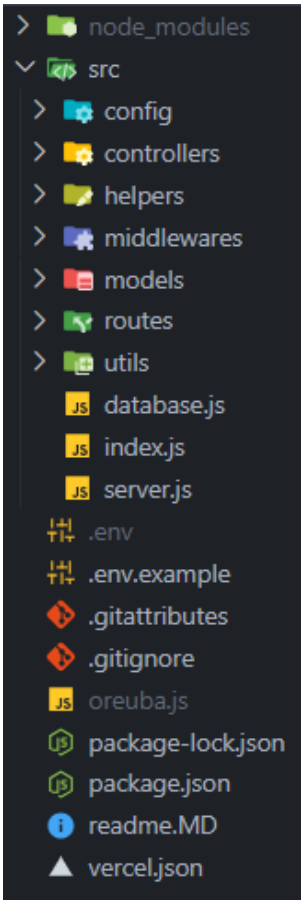

Figura 3.2 Estructura del entorno del desarrollo del componente backend

Determinación de roles

Para el desarrollo de este componente, se tiene en cuenta la gestión del centro de terapias, lo cual implica la participación de diferentes roles. Cabe destacar que cada rol puede acceder previamente funcionalidades como el login y recuperación de contraseña. Estos roles tienen distintas funcionalidades y se dividen en: paciente, secretaria y doctor. En la **Figura 3.3** se muestra más a detalle estos roles.

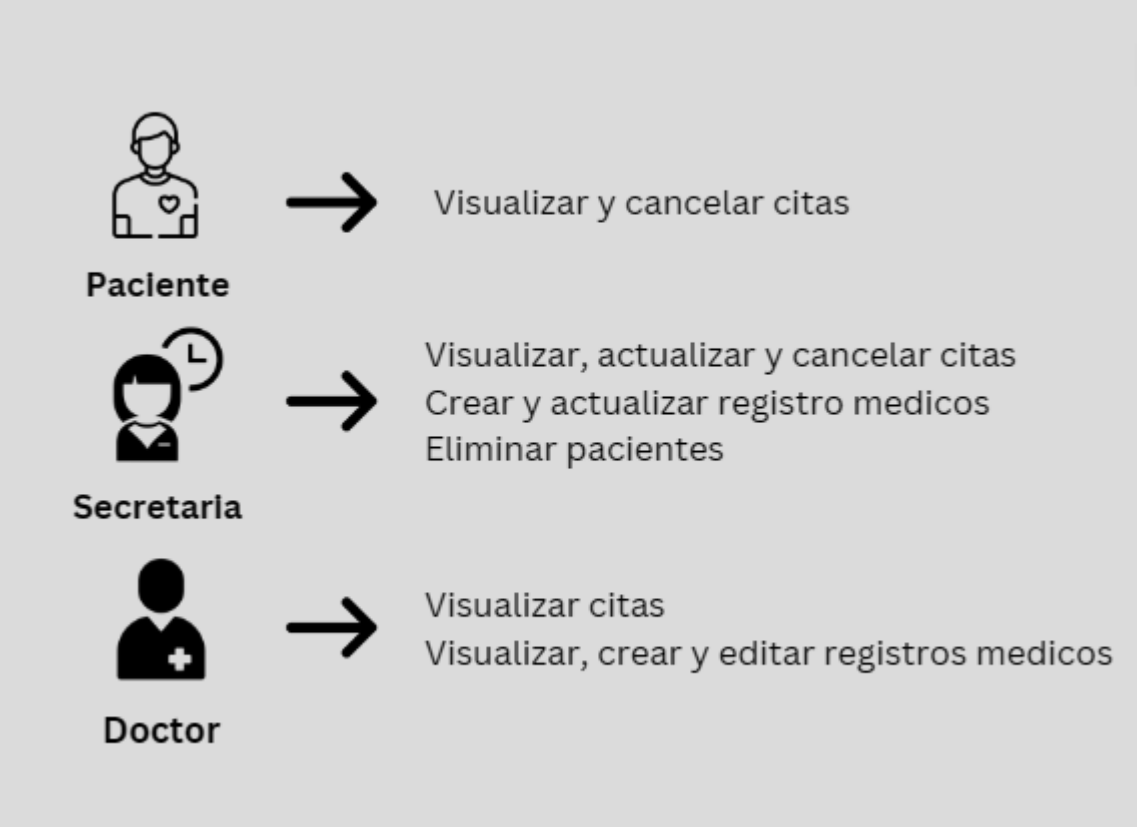

Figura 3.3 Roles de usuarios

Sprint 1. Autenticación

Para este sprint, se desarrolla las siguientes funcionalidades necesarias:

- Protección de rutas
- Endpoint para el registro de usuario
- Endpoint para el inicio de sesión
- Endpoints para la recuperación de contraseña

Protección de rutas

Para asegurar una buena autenticación y seguridad en cada endpoint, es fundamental protegerlos adecuadamente. Esto se logra mediante la creación de una función que actúa como middleware en la ruta. Por ende, el método que configura todo esto se encuentra en la **Figura 3.4**. Este middleware se aplica en cada ruta respectiva según sea necesario (véase la **Figura 3.5)**.

```
const verificarAutenticacion = async (req,res,next)=>{

if(!req.headers.authorization) return res.status(404)
    .json({msg:"Lo sentimos, debes proprocionar un token"})
    const {authorization} = req.headers
    try {
        const {id,rol} = jwt.verify(authorization.split(' ')[1],process.env.JWT_SECRET)
        if (rol==="usuario"){
            req.usuarioBDD = await Usuario.findById(id).lean().select("-password")
            next()
        }
    } catch (error) {
        const e = new Error("Formato del token no válido")
        return res.status(404).json({msg:e.message})
    }
}
```

Figura 3.4 Método para autenticación

```
registroRouter.post("/crear", verificarAutenticacion, crearRegistro);
```

Figura 3.5 Ejemplo de implementación del middleware

Además, para aumentar la seguridad, en cada ruta se implementa una condición que verifica si el usuario tiene los permisos necesarios mediante los encabezados “isdoctor”, “issecre” o “ispaciente”. Dependiendo del rol especificado en estos encabezados, se otorga o deniega el acceso a la ruta correspondiente. Estos valores se envían como cadenas de texto en lugar de valores booleanos (“true”) para maximizar la seguridad en la verificación de permisos. Esta práctica asegura que las funcionalidades de la aplicación sean solo accedidas por personas autorizadas, ya que, en caso intentar acceder sin estar autorizado no se permite el acceso a la funcionalidad.

La **Figura 3.6** se evidencia en parte este sistema de seguridad.

```
const isSecre = req.headers['issecre'] === 'true';
 const isDoctor = req.headers['isdoctor'] === 'true';
 const isPaciente = req.headers['ispaciente'] === 'true';

```

Figura 3.6 Ejemplo de condición de permisos

Endpoint para el registro de usuario

Este endpoint registra nuevos usuarios en la plataforma. Verifica que se completen todos los campos requeridos y que el correo electrónico no esté registrado previamente. Si los datos son válidos, se guarda el nuevo usuario en BD después de encriptar la contraseña. Este endpoint es utilizado mayormente por la secretaria. Para la verificaciones del funcionamiento en la **Figura 3.7** se evidencia la petición para este endpoint, resultando exitosa. La **Figura 3.8** detalla la prueba unitaria que verifica el registro exitoso, devolviendo un mensaje de éxito. Este enfoque asegura la integridad y fiabilidad del backend, al tiempo que proporciona una capa adicional de validación antes de su implementación en producción.

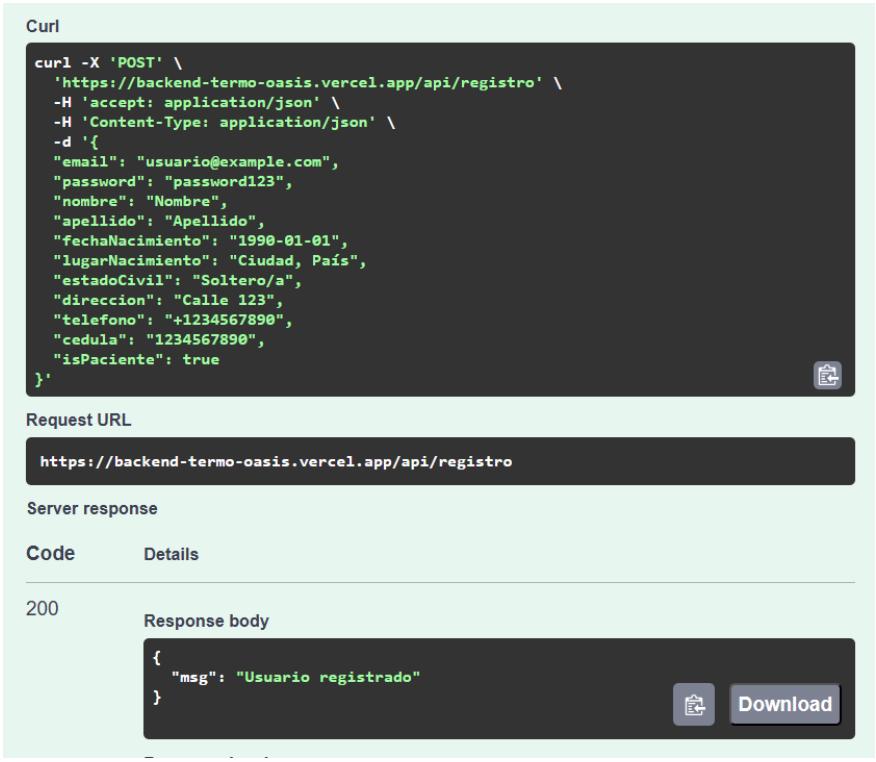

Figura 3.7 Petición para el endpoint de registro de usuario

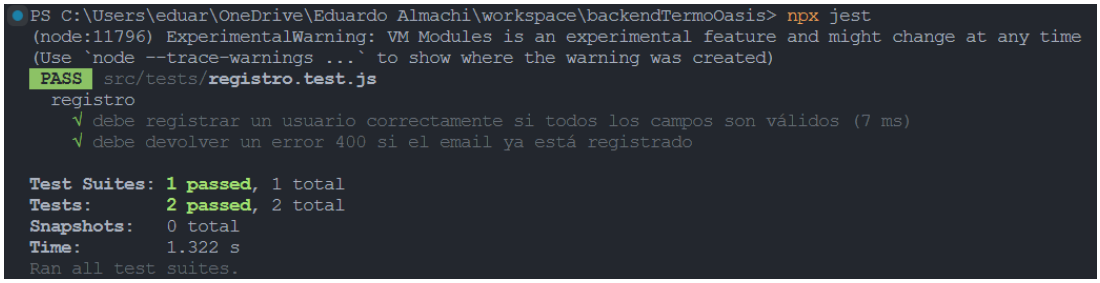

Figura 3.8 Prueba unitaria para el endpoint de registro de usuario

Endpoint para el inicio de sesión

Este endpoint gestiona el login. Primero, verifica que se completen todos los campos requeridos. Luego, si el usuario está registrado. Si no se encuentra registrado, se devuelve un mensaje de error, sino devuelve la información necesaria del usuario, esta respuesta devuelve todo lo necesario para quienes consuman este endpoint. En la **Figura 3.9** se evidencia la petición de este endpoint y en la **Figura 3.10** se evidencia la prueba unitaria.

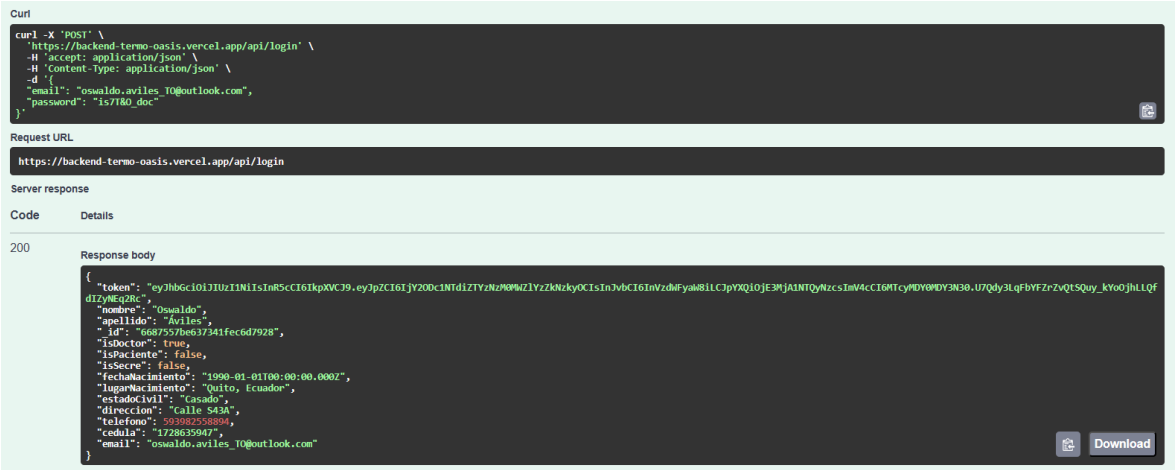

Figura 3.9 Petición para el endpoint de inicio de sesión

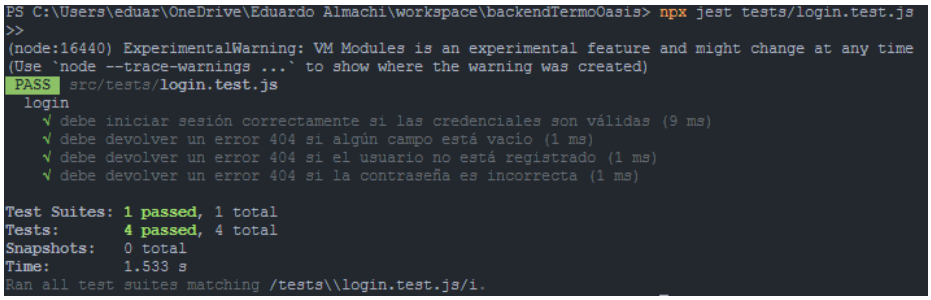

Figura 3.10 Prueba unitaria del endpoint de inicio de sesión

Endpoints para la recuperación de contraseña

Esta funcionalidad consta de varios endpoints para ser consumidos por diferentes interfaces de usuario. Se utilizan tres endpoints para ser consumidos por la parte frontend de la aplicación web y un endpoint diseñado para ser consumido por la parte móvil de la plataforma.

Esto se hace porque las interfaces de usuario en aplicaciones web y móviles suelen diferir en los flujos y patrones de interacción. Al desarrollar endpoints específicos para cada plataforma, se puede optimizar la disponibilidad al adaptarse mejor a las particularidades y restricciones de cada tipo de dispositivo.

De hecho, implementar diferentes tipos de endpoints para el frontend y el móvil no solo mejora la eficiencia y la seguridad, sino que también proporciona una experiencia de usuario más fluida y adaptada a cada plataforma.

Recuperación de contraseña para la parte frontend

Los tres endpoints diseñados para el frontend web permiten a los usuarios iniciar sesión, mientras que el endpoint diseñado para la parte móvil de la plataforma proporciona la misma funcionalidad, adaptada para su consumo en dispositivos móviles.

A continuación, en las que se muestra el endpoint recuperar contraseña, comprobar token contraseña y nueva contraseña que forman parte de los endpoints que consuman la parte frontend web.

Previamente en la **Figura 3.11** se visualiza las rutas definidas de las siguiente manera.

```
1  //Frontend
2  usuarioRouter.post("/recuperar-password", recuperarPassword)
3  usuarioRouter.get("/recuperar-password/:token", comprobarTokenPassword);
4  usuarioRouter.post("/nueva-password/:token", nuevaPassword)
5
6  //Movil
7  usuarioRouter.post("/recuperar-password-movil", recuperarPasswordMovil)
```

Figura 3.11 Rutas para los endpoints de recuperar contraseña

El primer endpoint recibe el email para enviar un correo en la cual el usuario tiene que confirmar que quiere recuperar las credenciales como se ve en la **Figura 3.12** donde se muestra la petición a este endpoint y la prueba unitaria en la **Figura 3.15**

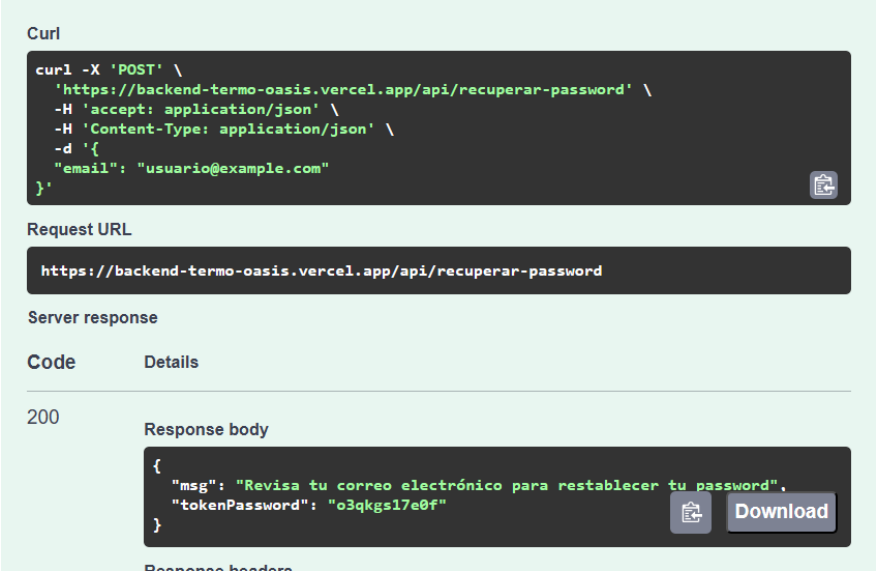

Figura 3.12 Petición para el endpoint de recuperar contraseña | Frontend

El segundo endpoint confirma el token que se recibe del primero para validar la cuenta y que el usuario sea autenticado y así pueda recuperar su contraseña. En la **Figura 3.13** se ve la petición a este endpoint y en la **Figura 3.15** se muestra la prueba unitaria.

Figura 3.13 Petición para el endpoint comprobar el token contraseña | Frontend

El tercer endpoint, utilizado para la recuperación de contraseñas, permite establecer una nueva contraseña. Este proceso incluye la validación de la cuenta del usuario y verifica la nueva contraseña a través de una confirmación de token único. De esta manera, permite que la contraseña se cambie de manera segura y que solo el usuario autorizado pueda realizar esta acción. En la **Figura 3.14** se visualiza la petición a este endpoint y la prueba unitaria en la **Figura 3.15**

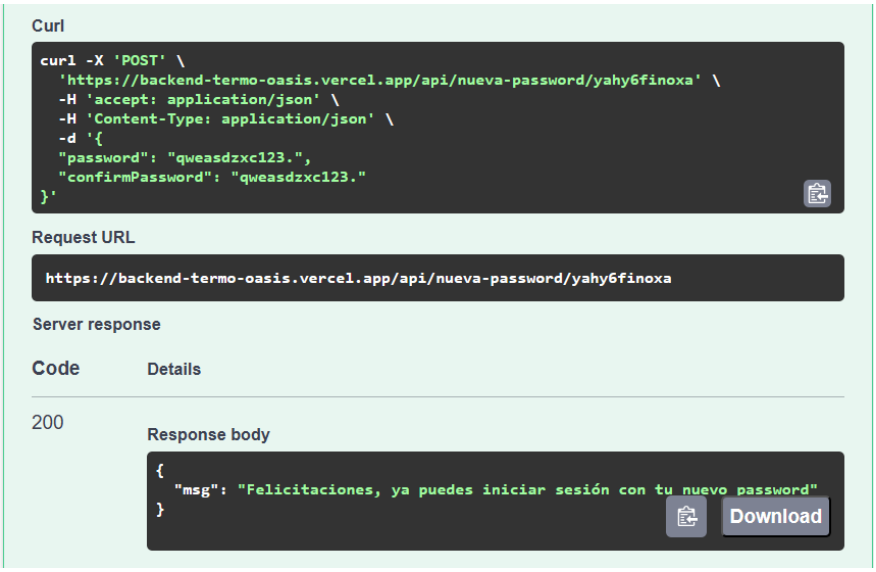

Figura 3.14 Petición para el endpoint de nueva contraseña | Frontend

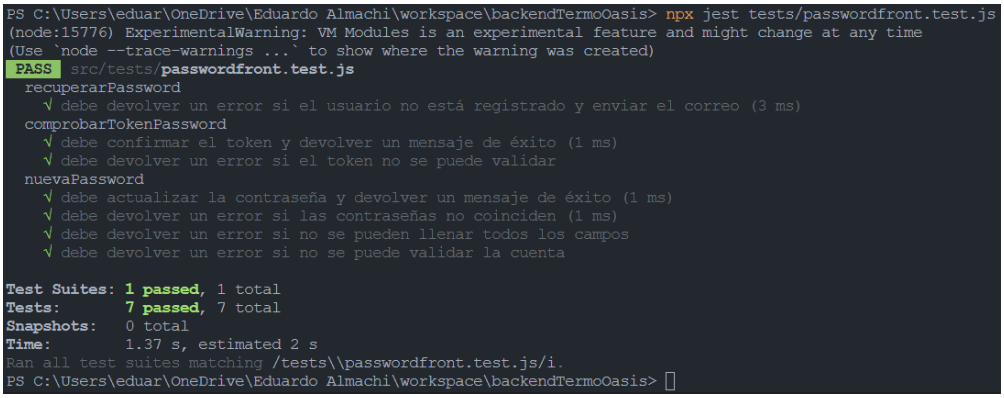

Figura 3.15 Prueba unitaria para los endpoints de recuperar contraseña | Frontend

Recuperación de contraseña de la parte móvil

Por otro lado, en la **Figura 3.16** se presenta el endpoint para la recuperación de contraseña que va a ser consumida por la parte móvil, esta solamente recibe los campos necesarios para validarlos, esto se hace para dar una mejor experiencia de usuario que utilizan plataformas móviles. Respectivamente, en la **Figura 3.17** se visualiza la prueba unitaria.

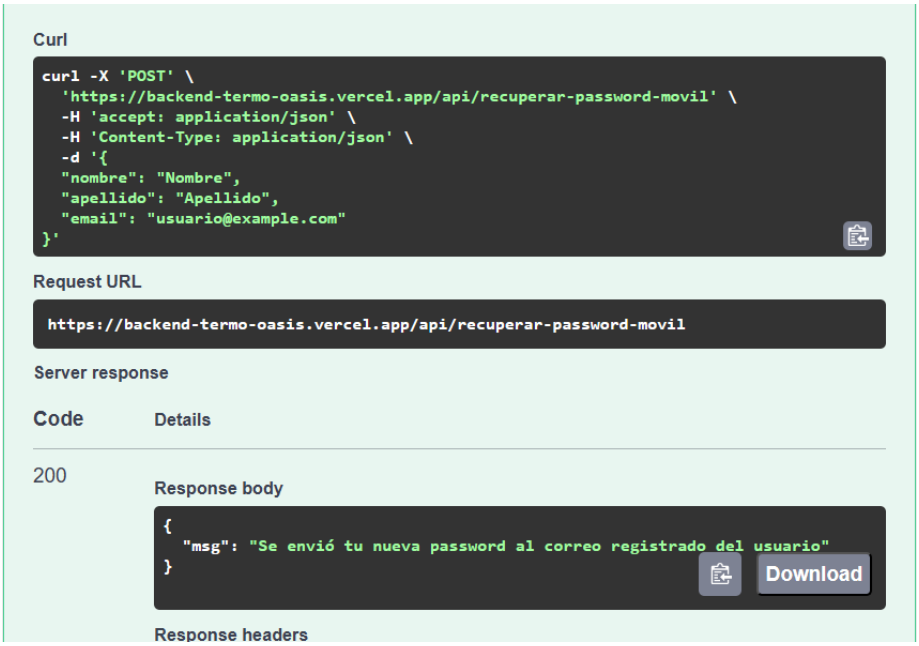

Figura 3.16 Petición para el endpoint de recuperar contraseña | Móvil

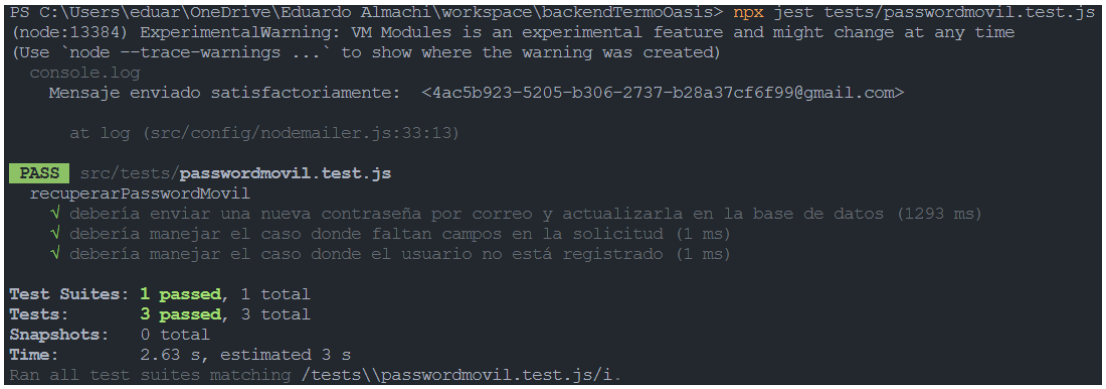

Figura 3.17 Prueba unitaria para el endpoint de recuperar contraseña | Móvil

Sprint.2 Módulo usuarios

Este sprint se basa en la gestión de usuarios, constando con las siguientes tareas:

- Creación de usuario por rol
- Endpoint para obtener los usuarios pacientes
- Endpoint para eliminar usuario
- Endpoint para detalle de usuario
- Endpoint para el perfil del usuario validado

Creación de usuario por rol

Para la acción de crear usuarios con rol especifico, se ha desarrollado y documentado un endpoint. El controlador maneja la lógica de negocio para crear nuevos pacientes. Aquí se valida la entrada para poder crear un paciente, para ello se utiliza el endpoint creado en el sprint número uno de registrar usuarios, solamente que aquí se pone como True cualquiera de las variables de rol para registrarlo como tal. En la **Figura 3.18** puede visualizar como se implementa.

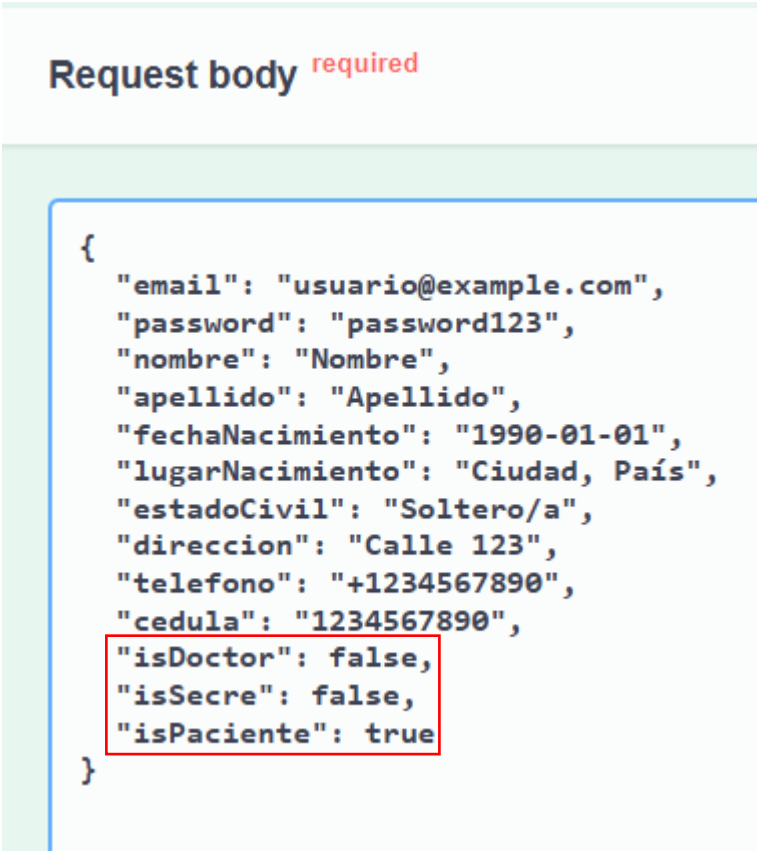

Figura 3.18 Asignación rol de usuario

Endpoint para obtener los usuarios pacientes

Por otro lado, para la visualización de todo los pacientes, se ha desarrollado y documentado un endpoint. El endpoint utiliza el controlador para obtener todos los usuarios, esta función otorga la posibilidad de poder desplegar toda la información de los pacientes registrados, como se puede ver en la **Figura 3.19**. Además, en la **Figura 3.20** se muestra la prueba unitaria que da constancia del funcionamiento.

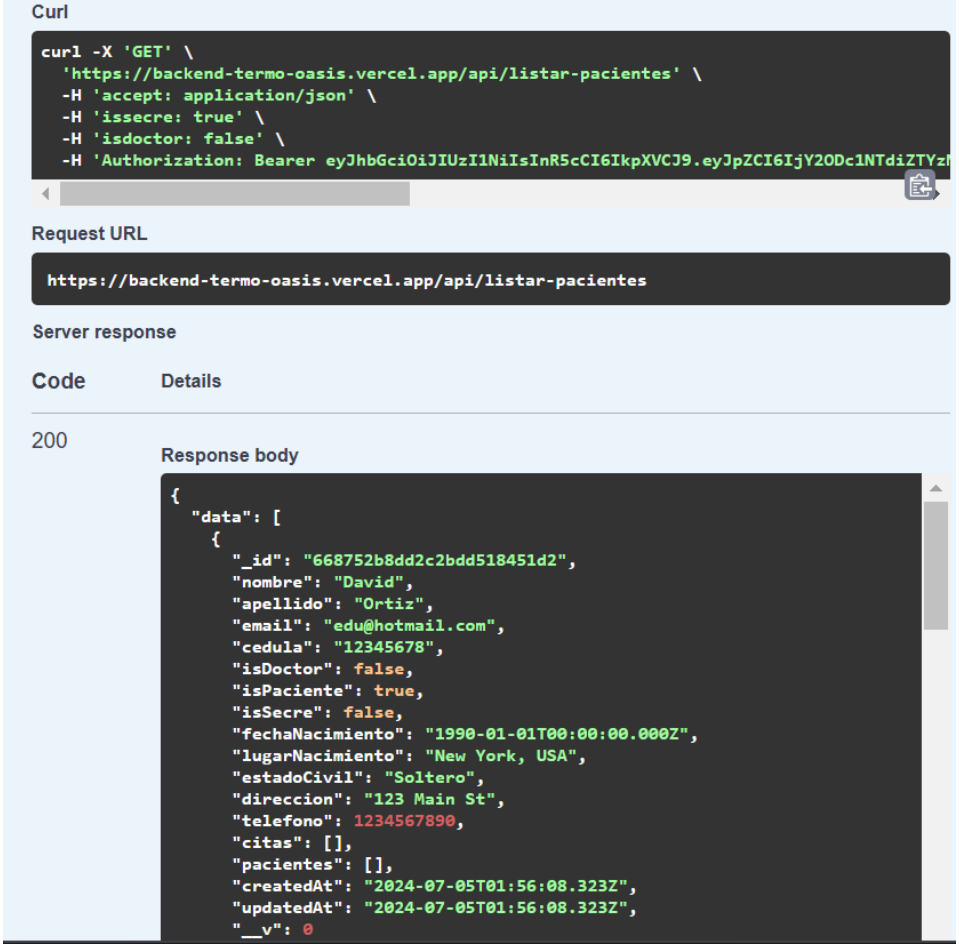

Figura 3.19 Petición para el endpoint de mostrar pacientes

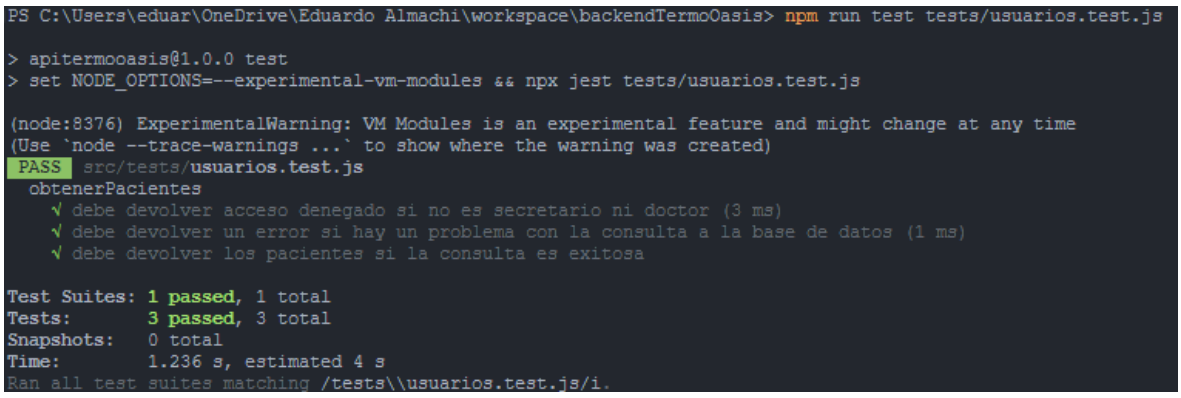

Figura 3.20 Prueba unitaria para el endpoint de mostrar pacientes

Endpoint para eliminar usuario

Para la eliminación de usuarios, se implementa un endpoint que utiliza un controlador, este se encarga de manejar la lógica de eliminación de un usuario en específico. Este controlador recibe el ID del usuario a eliminar a través de la solicitud y realiza las validaciones necesarias para asegurar que el usuario existe antes de proceder con la

eliminación. Además, se asegura de manejar posibles errores y devolver una respuesta adecuada, indicando si la operación fue exitosa o si ocurre algún problema durante el proceso, como se puede evidenciar en la **Figura 3.21**. Adicionalmente, se ha desarrollado una prueba unitaria para asegurar que el endpoint funcione correctamente como se evidencia en la **Figura 3.22**. Esta implementación garantiza una gestión segura y eficiente de los usuarios en el backend.

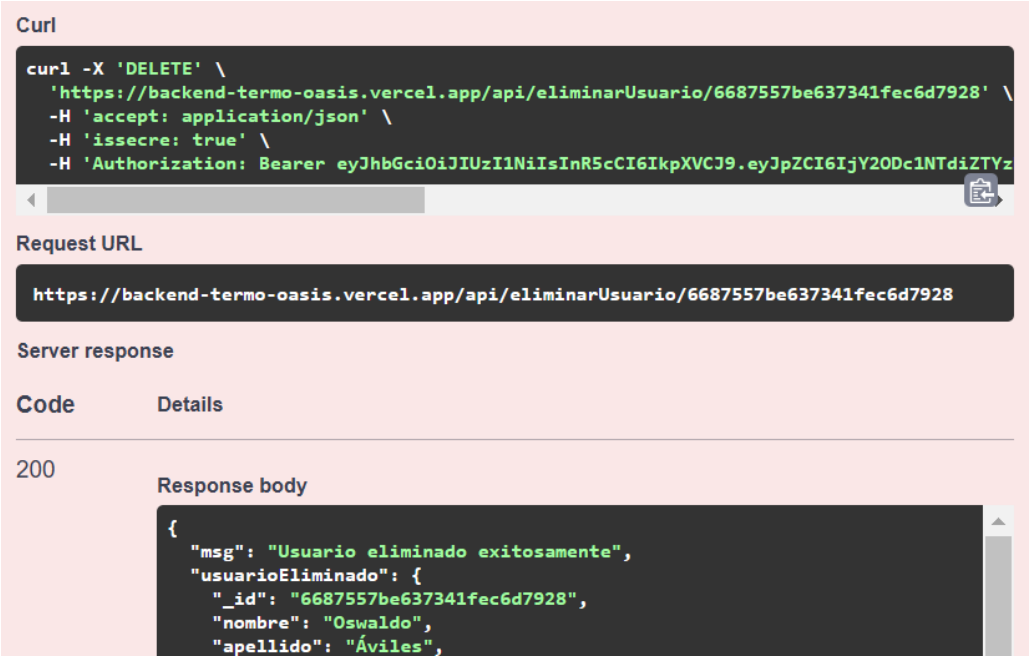

Figura 3.21 Petición para el endpoint de eliminar usuario

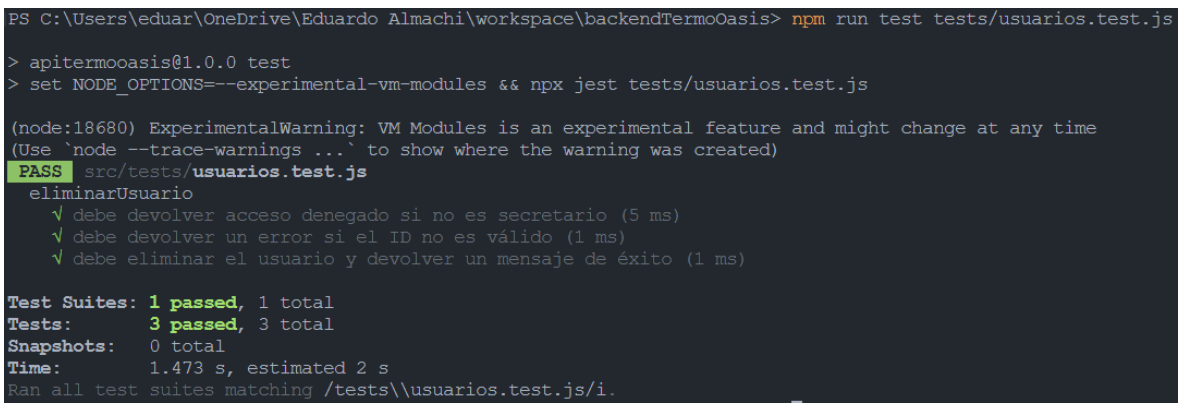

Figura 3.22 Prueba unitaria para el endpoint de eliminar usuario

Endpoint para detalle de usuario

Este endpoint se encarga de desplegar toda la información sobre un paciente en específico, que es utilizado para cualquier rol. Para la creación de este endpoint se utiliza un controlador que obtiene estos valores. Además, al momento de realizar la petición a esta

ruta se verifica cualquier error relacionado con la funcionalidad. En caso de éxito, el controlador devuelve una respuesta con toda la información del paciente como se puede ver en la **Figura 3.23.** Además, en la **Figura 3.24** se evidencia la prueba unitaria que asegura el correcto funcionamiento de cada lógica.

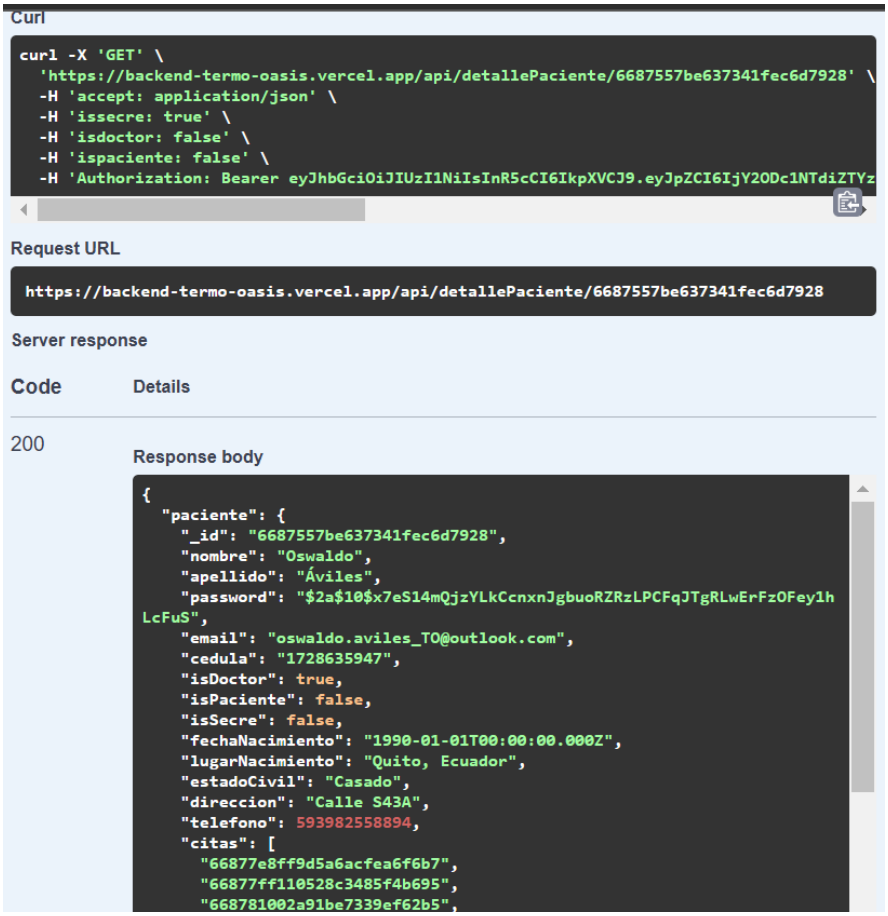

Figura 3.23 Controlador para el endpoint de detalle de usuario

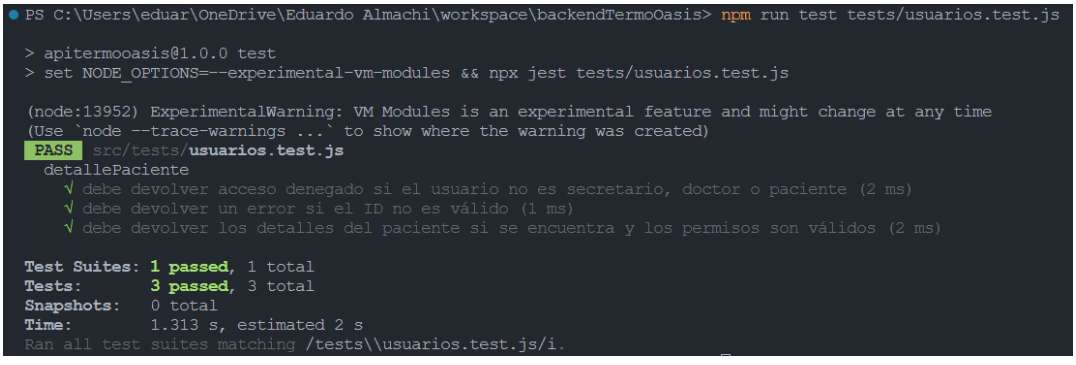

Figura 3.24 Prueba unitaria para el endpoint de detalle de paciente

Endpoint para el perfil del usuario validado

Este controlador a diferencia del que muestra los detalles de un paciente en específico, este presenta los datos del usuario actualmente autenticado, para que el controlador perfil sepa cual usuario está validado y traiga sus valores, utiliza el middleware de autenticación. Este middleware verifica el token de autenticación y extrae la información del usuario, almacenándola. Es endpoint está diseñado para que el backend lo consuma cada vez que un usuario inicie sesión. De esta manera, se eliminan valores innecesarios como el token, las marcas de tiempo y la versión del documento, asegurando que solo se envíe al frontend la información relevante y no sensible del usuario. Esto optimiza la seguridad y eficiencia del manejo de datos en la interfaz de usuario. En la **Figura 3.25** Petición para el endpoint del perfil y en la **Figura 3.26**

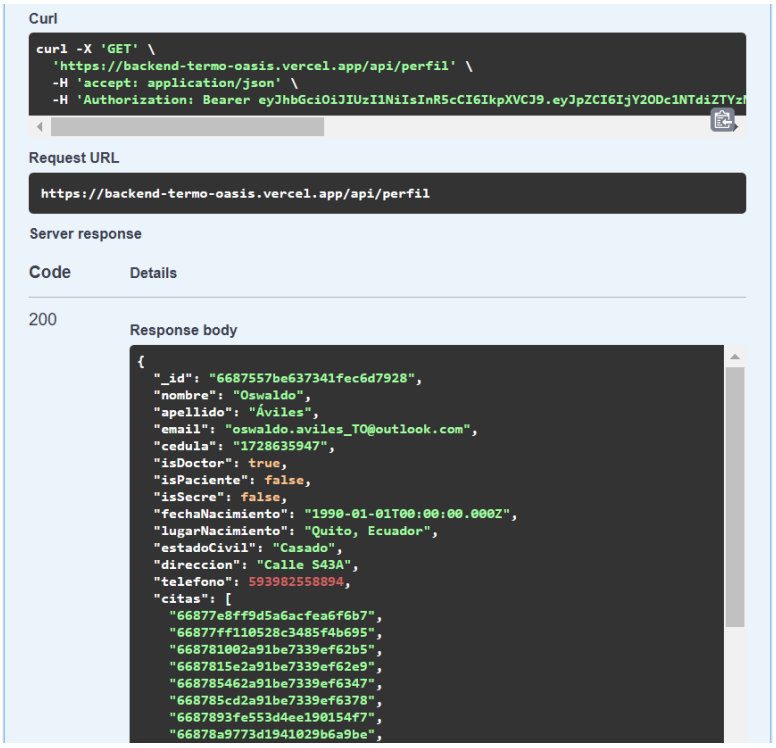

Figura 3.25 Petición para el endpoint del perfil validado

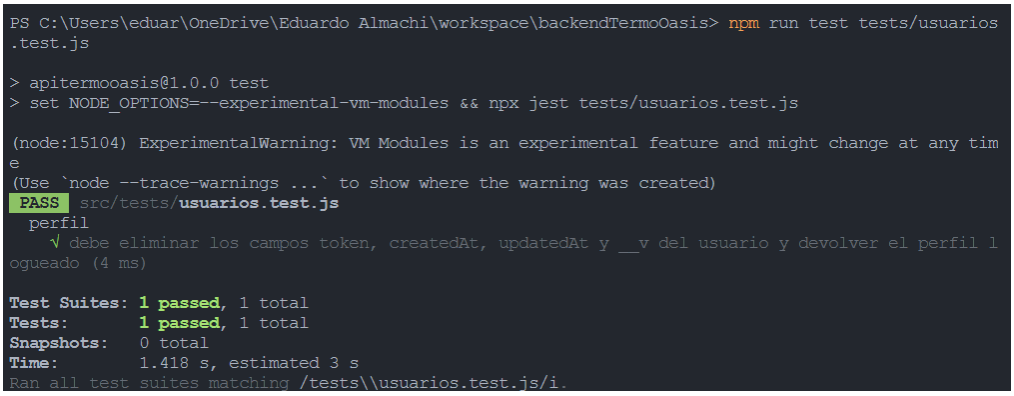

Figura 3.26 Prueba unitaria para el endpoint del perfil validado

Sprint 3. Módulo citas

Este sprint se basa en los requerimientos para la gestión de citas, constando como tareas las siguientes:

- Endpoint para crear cita
- Endpoint para editar cita
- Endpoint para mostrar todas las citas de un paciente
- Endpoint para cancelar cita
- Endpoint para mostrar todas las citas
- Endpoint para mostrar una cita en especifico

Endpoint crear cita

Para permitir la creación de citas en la aplicación, se ha desarrollado un endpoint. Este consta de un controlador para el rol de secretaria, que gestiona la creación de una nueva cita extrayendo y validando los datos necesarios, verificando la existencia del paciente y el doctor en la base de datos, verificando fechas y finalmente, creando la cita. En la **Figura 3.27** se evidencia la petición y en la **Figura 3.28** se detalla el resultado exitoso con base a la prueba unitaria.

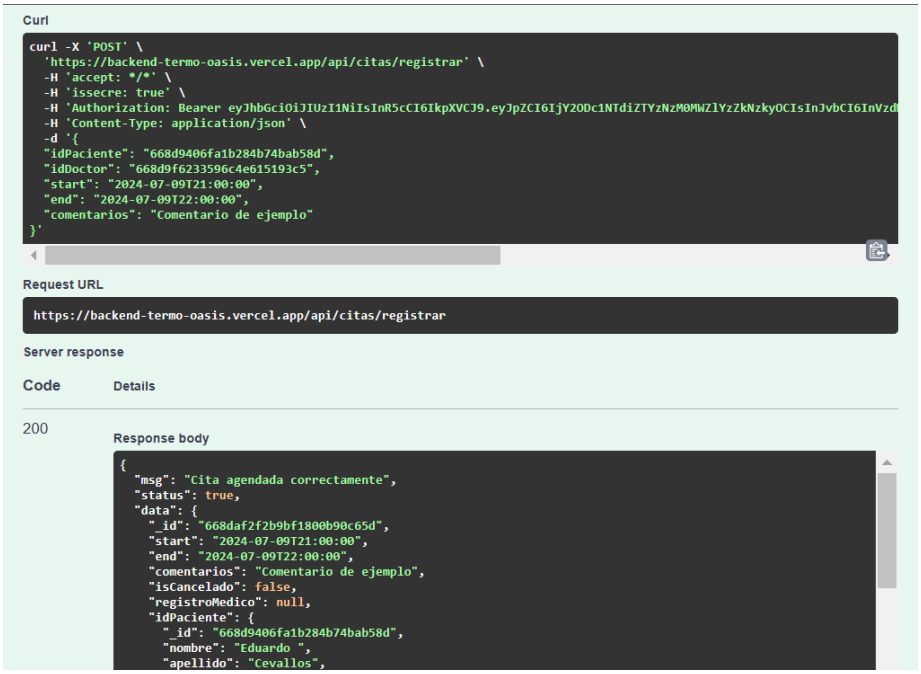

Figura 3.27 Controlador para el endpoint de crear cita

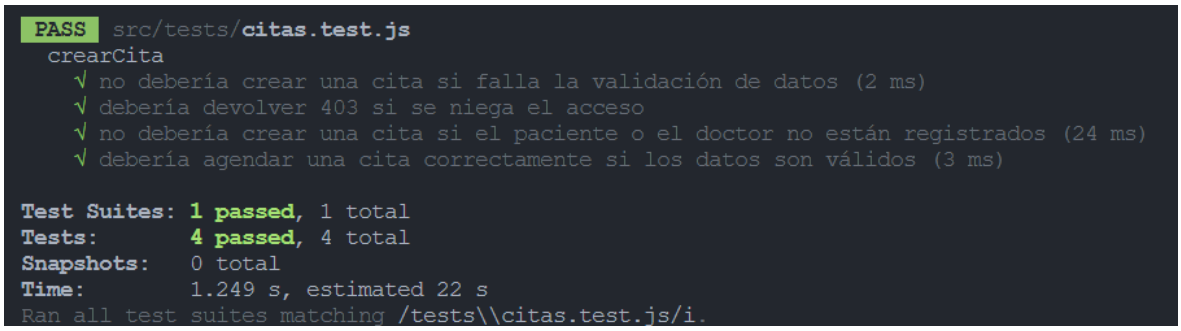

Figura 3.28 Prueba unitaria para el endpoint de creación de cita

Endpoint para editar cita

Para permitir la actualización de citas en la aplicación, se ha desarrollado un endpoint específico. Este endpoint consta de un controlador para el rol de secretaria que gestiona la actualización de una cita extrayendo los datos nuevos si es el caso si no se mantiene los mismos para después guardarlos en la base. Además, dispone de funcionalidades que verifican las fechas de la reagendación de la cita. A través de las **Figura 3.29** se ve la petición del endpoint y en la **Figura 3.30** se observa la prueba unitaria del endpoint.

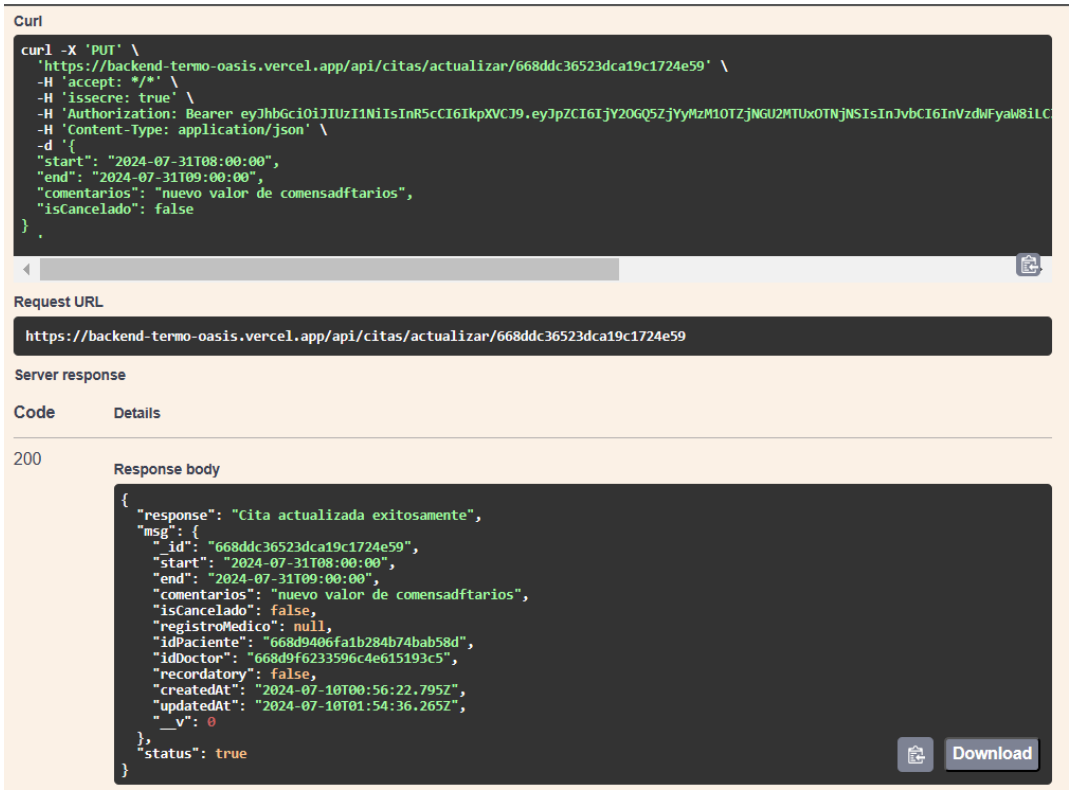

Figura 3.29 Petición para el endpoint de actualizar cita

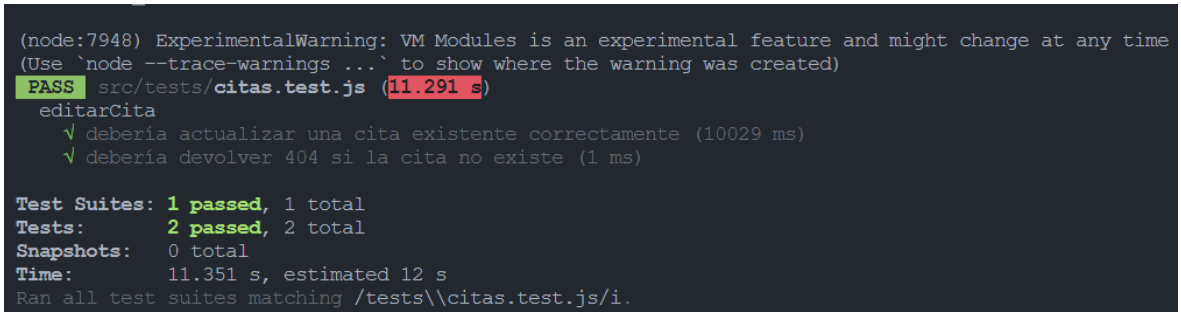

Figura 3.30 Prueba unitaria para el endpoint de actualizar cita

Endpoint para mostrar las citas de un paciente

Una vez que un usuario se registra con el rol correspondiente, puede visualizar las citas que tiene, este mismo es utilizado por todos los roles. Para hacer esto, se utiliza un endpoint que busca las citas de un paciente específico en la base de datos. Este endpoint recibe el ID del paciente como parámetro en la solicitud. Luego, devuelve todas las citas encontradas, incluyendo toda la información relevante. Ante posibles contratiempos , el endpoint devuelve un mensaje de error. La **Figura 3.31** permite ver la petición que muestra los datos que se obtienen**.** Finalmente, en la **Figura 3.32** se visualiza la prueba unitaria respectiva.

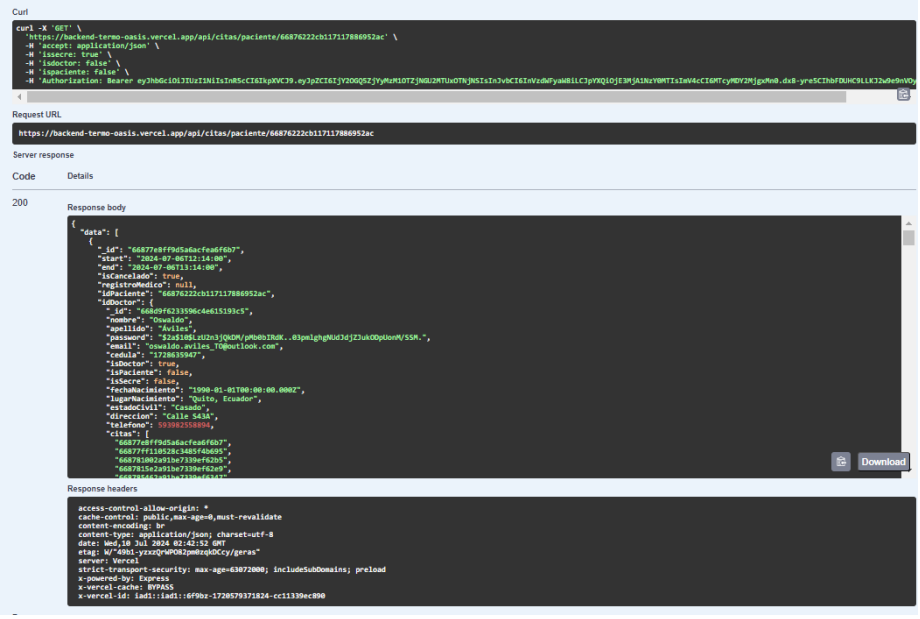

Figura 3.31 Petición para el endpoint de visualización de citas por paciente

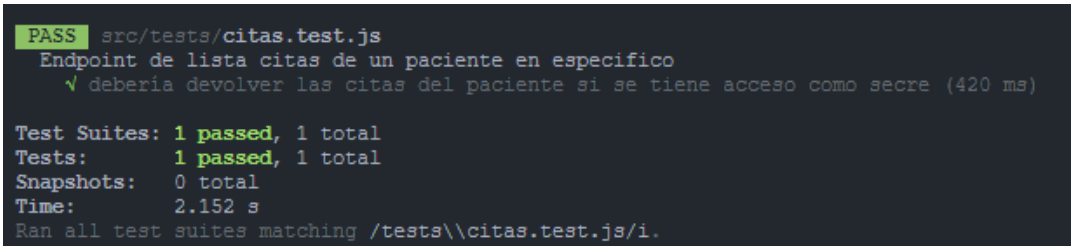

Figura 3.32 Prueba unitaria del endpoint de visualización de citas por paciente

Endpoint para cancelar cita

Este endpoint está diseñado para permitir la cancelación de citas únicamente cuando el usuario tenga una cita agendada, el mismo es utilizado por la secretaria y el paciente. Es importante destacar que la cancelación también puede ser realizada por otro usuario con un rol administrativo distinto. Para ello este endpoint tiene la funcionalidad de buscar entre las citas creadas y actualiza el campo de cancelado a true y como consecuencia la cita consta como cancelada. En base a esto la **Figura 3.33** detalla la petición y la **Figura 3.34** evidencia de manera clara la correcta implementación del endpoint a través de una prueba unitaria.

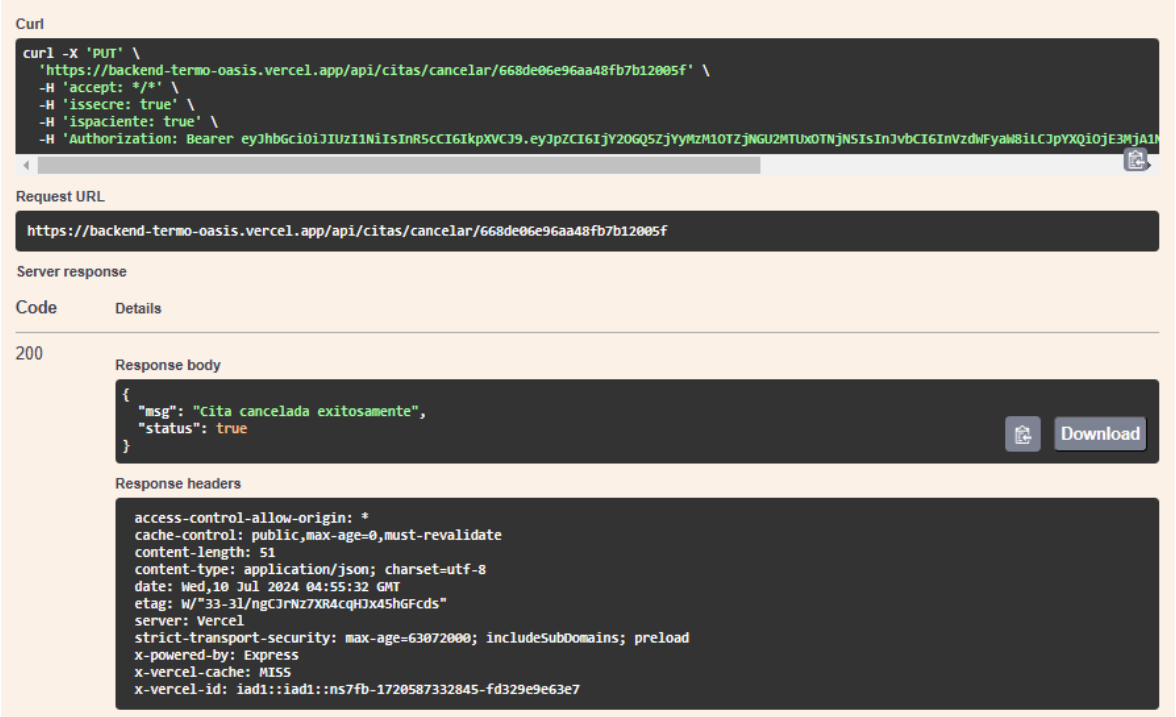

Figura 3.33 Petición para el endpoint de cancelación de cita

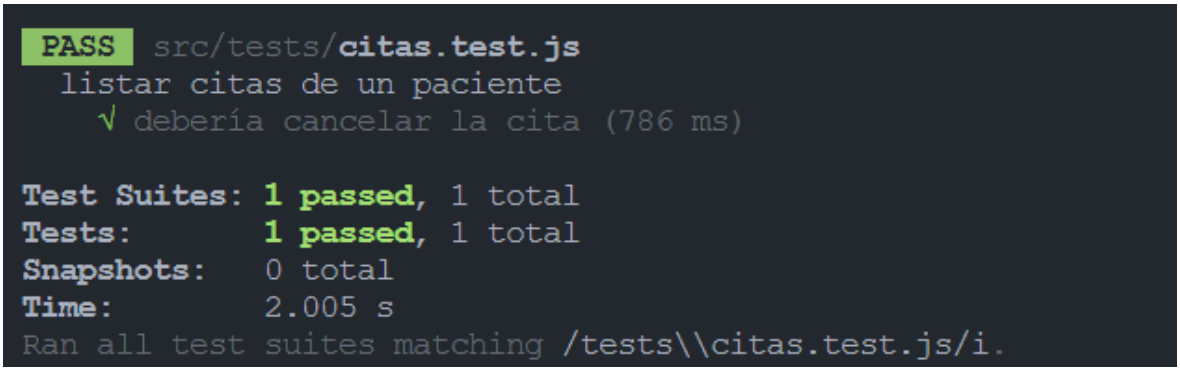

Figura 3.34 Prueba unitaria para el endpoint de cancelación de cita

Endpoint para mostrar todas las citas

Para mostrar todas las citas en la aplicación, se ha desarrollado un endpoint específico, que es utilizado por todos los roles. Este endpoint consta de un controlador que gestiona la obtención de todas las citas sin excepción y por ende obtiene todos los datos. Además, al momento de realizar las consultas se verifica que el rol sea el requerido, en este caso cualquier rol puede acceder a esa funcionalidad. Esto se evidencia en la **Figura 3.35** Finalmente, en la **Figura 3.36** se refleja el éxito alcanzado con la prueba unitaria.

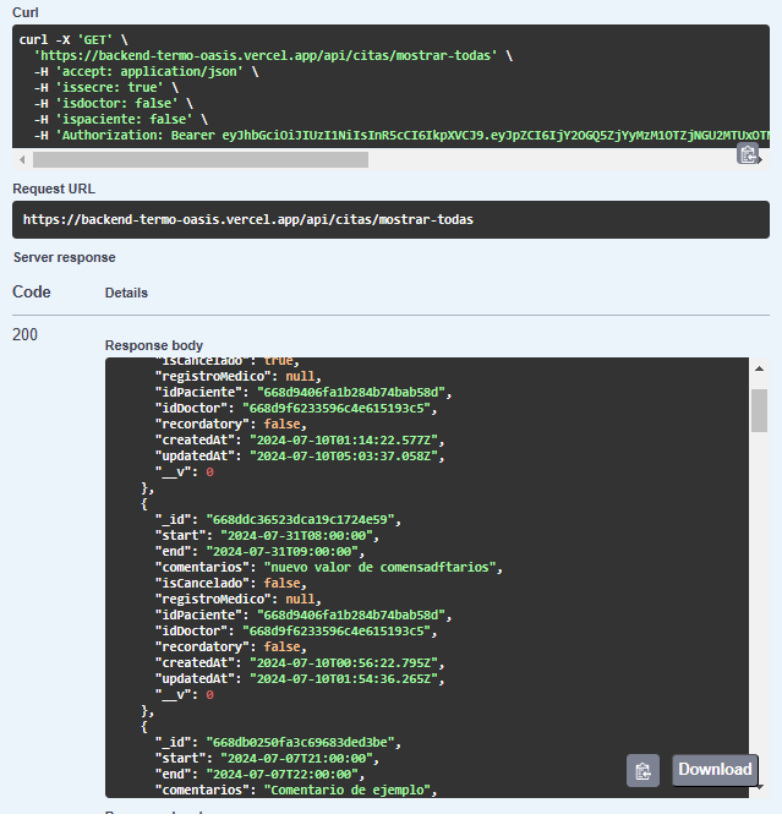

Figura 3.35 Petición para el endpoint de mostrar citas

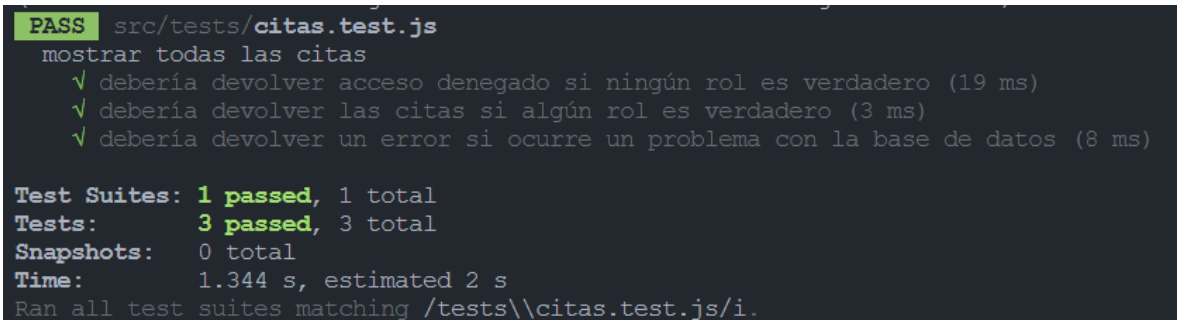

Figura 3.36 Prueba unitaria para el endpoint de mostrar citas

Endpoint para mostrar una cita en especifico

Para mostrar una cita en específico, se ha desarrollado un endpoint, que es utilizado por todos los roles. Este endpoint consta de un controlador que se encarga de toda la lógica. Este endpoint gestiona la obtención de una cita en concreto a partir del número de identificación de la cita. De esa manera se obtiene todos los datos de una cita incluyendo los datos del paciente y el doctor, además, al momento de realizar las peticiones se verifica el rol para la petición. La petición muestra el correcto funcionamiento en la **Figura 3.37**. Finalmente, la **Figura 3.38** evidencia el desarrollo de la prueba unitaria.

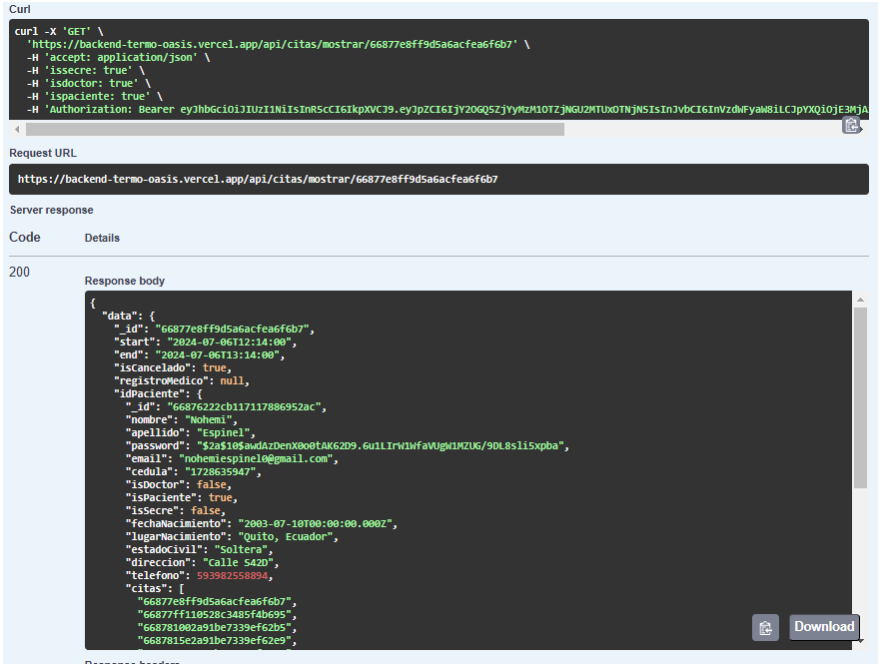

Figura 3.37 Petición para el endpoint de mostrar cita en específico

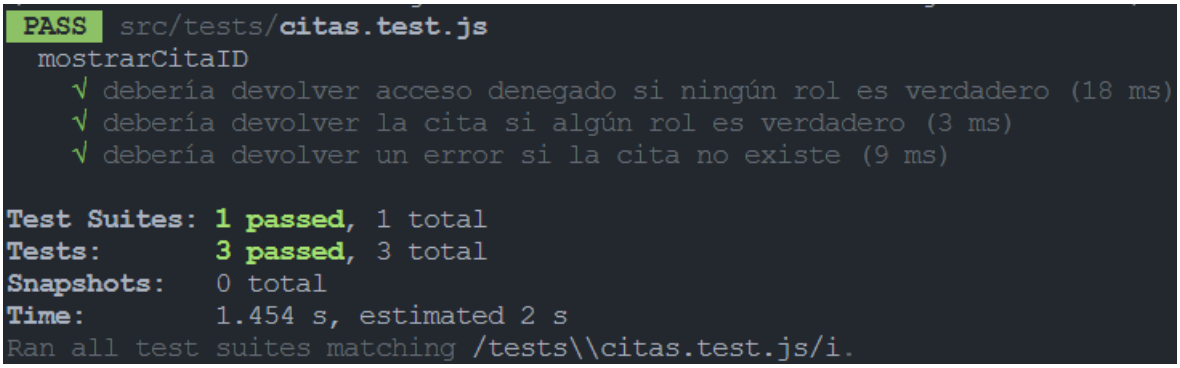

Figura 3.38 Prueba unitaria para el endpoint de mostrar cita en especifico

Sprint 4. Módulo ficha medica

Este sprint se basa en los requerimientos para administrar fichas médicas, constando como tareas las siguientes:

- Endpoint para crear una ficha médica.
- Endpoint para mostrar fichas medicas de un paciente.
- Endpoint para actualizar una ficha médica

Endpoint para crear una ficha médica

Se ha creado un endpoint dedicado a crear registro médicos, que es manejado únicamente por el rol doctor. Este endpoint concede funcionalidades al rol doctor. Este endpoint se encarga de registrar una ficha médica, además, contiene distintas funcionalidades para verificar los campos y registrar una sola ficha médica por cita, esto se puede ver implementado gracias a la petición de la **Figura 3.39**. Finalmente, la **Figura 3.40** muestra la prueba unitaria que verifica que todas las partes se cumplen como es previsto.

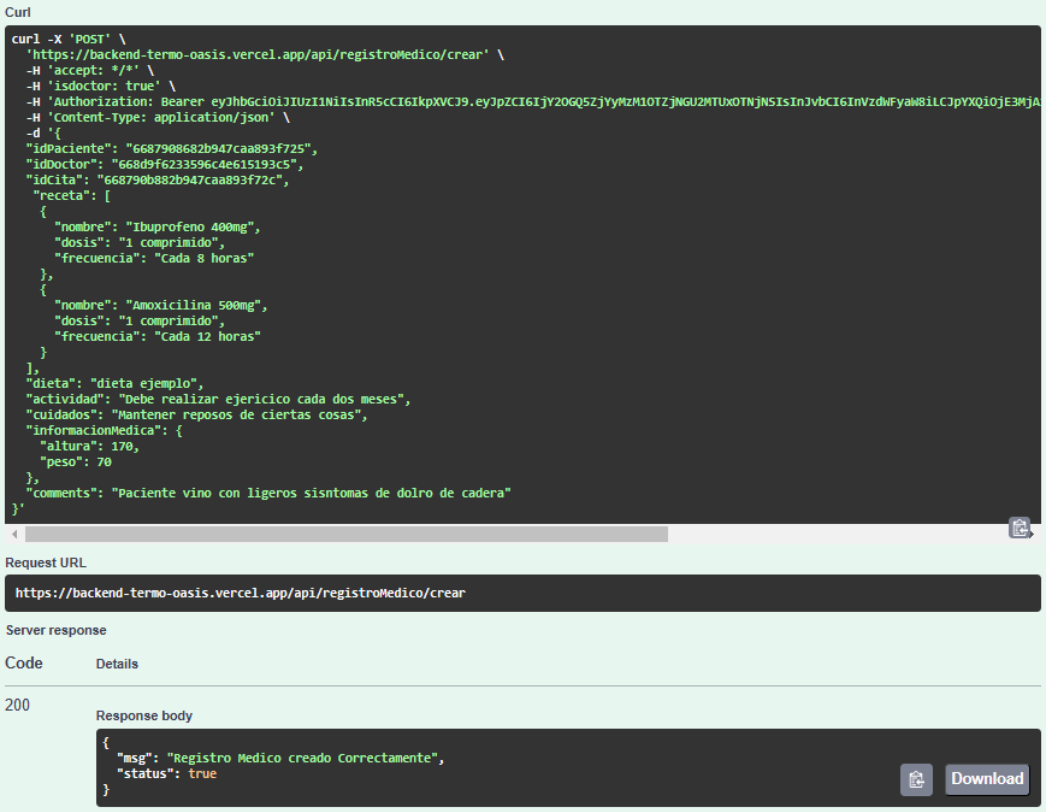

Figura 3.39 Controlador para el endpoint de crear ficha medica

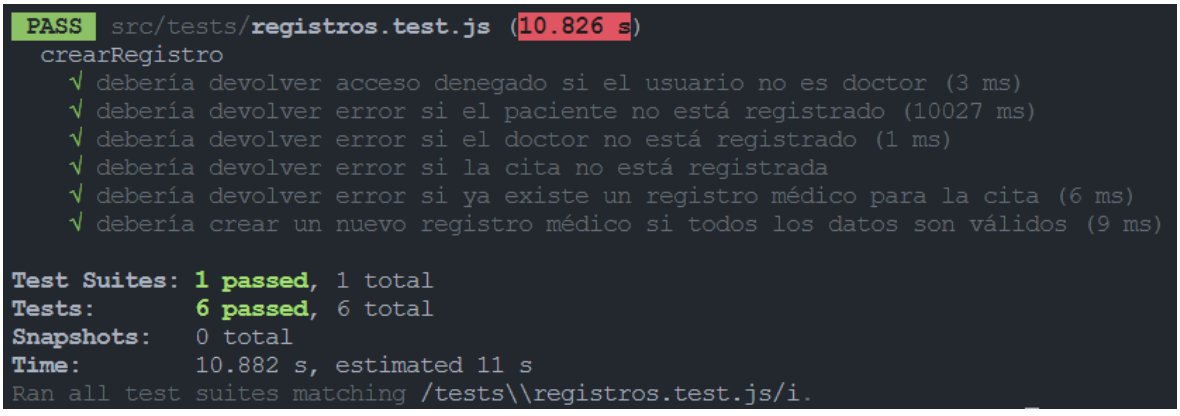

Figura 3.40 Prueba unitaria para el endpoint de crear ficha medica

Endpoint para mostrar fichas medicas de un paciente

Se ha establecido un endpoint para listar las citas médicas específicas de un paciente que es utilizado por el rol doctor y secretaria. Este endpoint filtra los registros con base al id de una cita y muestra la ficha medica respectiva. La funcionalidad asociada a este endpoint maneja condiciones importantes como verificar que exista la cita y la verificación de permisos. La **Figura 3.41** evidencian el esto y la **Figura 3.42** muestra la prueba unitaria.

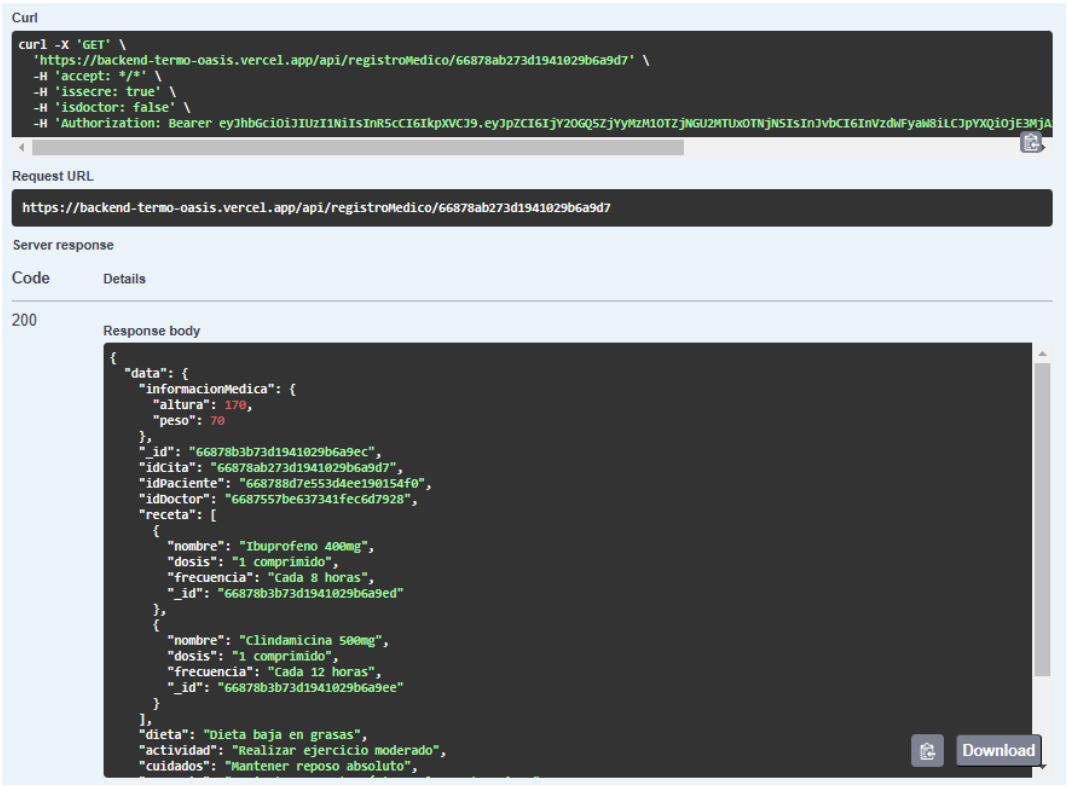

Figura 3.41 Petición para el endpoint de mostrar fichas medicas

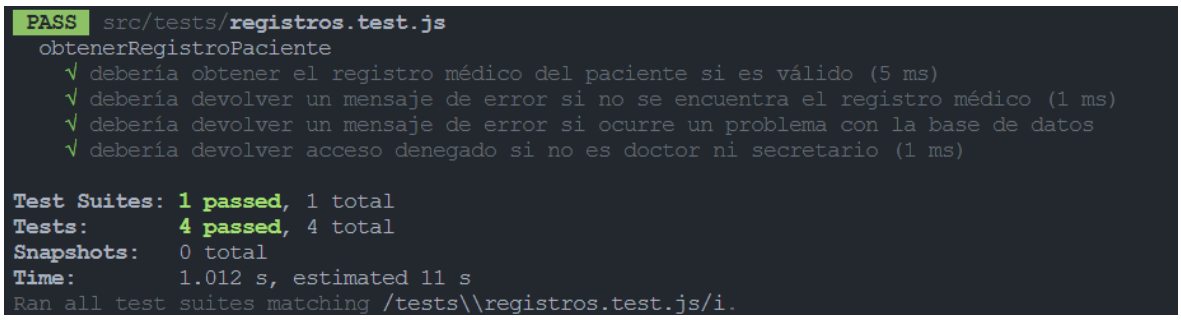

Figura 3.42 Prueba unitaria para el endpoint de mostrar fichas

Endpoint para actualizar una ficha médica

Para realizar esta lógica, se ha implementado un endpoint que es utilizado por el rol de doctor. Este endpoint involucra la edición de una ficha medica en específico. Además,

incluye más funcionalidades como la validaciones de id y el control de errores permitiendo así la actualización efectiva de la ficha médica asociada a una cita específica. Respectivamente esto se visualiza a través de la petición en la **Figura 3.43** y en la **Figura 3.44** la prueba unitaria de las partes del endpoint.

Figura 3.43 Petición para el endpoint de editar registros médicos

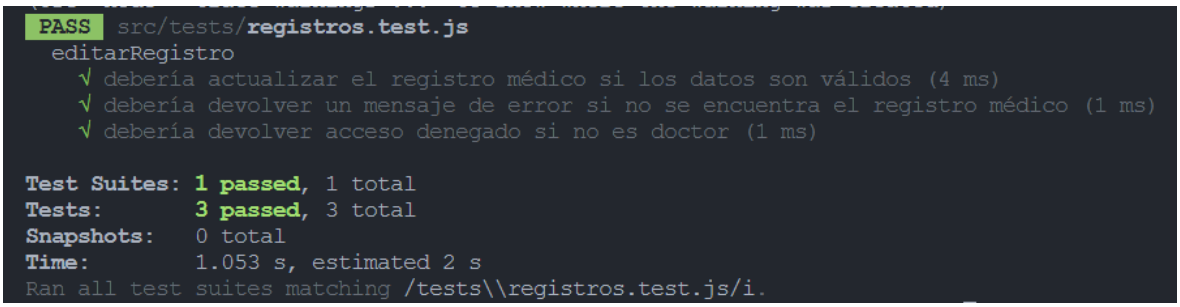

Figura 3.44 Prueba unitaria para el endpoint de editar registros médicos

Sprint 5. Implementación de lógica de notificación de emails

En esta sección se desarrolla el envío de emails, siendo una funcionalidad que depende de los requisitos, por ende, se desarrollan las siguientes tareas:

- Envío de email al recuperar contraseña (frontend y móvil)
- Envío de email al crear cita
- Envío de email al cancelar cita
- Envío de email al actualizar cita

Envío de email al recuperar contraseña (frontend y móvil)

Se implemento la funcionalidad de envío de correos electrónicos para la recuperación de contraseñas, tanto para web (frontend) como para móvil. El objetivo es proporcionar a los usuarios una manera sencilla y segura de restablecer su contraseña en caso de que la olviden.

Para la configuración del servicio de envío de emails, hemos utilizado NodeMailer. Las funciones correspondientes se integran dentro de los controladores pertinentes de recuperación de contraseña, activándose cada vez que se consume el endpoint correspondiente del controlador. Esto garantiza que los emails de recuperación de contraseña se envíen de manera eficiente y segura.

Email de recuperación de contraseña para la parte frontend

Para el envío de emails en la parte frontend, se utiliza un template que incluye un enlace para la confirmación del token. Este proceso se implementa para verificar que el usuario desea actualizar su contraseña, proporcionando así una capa adicional de seguridad. Cada vez que un usuario solicita la recuperación de su contraseña, se envía un email con el enlace de confirmación, asegurando que solo el usuario legítimo pueda cambiar la contraseña, en la **Figura 3.45** se detalla el correo como resultado.

Figura 3.45 Email para le recuperación de contraseña (frontend)

Email de recuperación de contraseña para la parte móvil

Se ha desarrollado al lógica para el envío de emails que utiliza los usuarios de móviles, se ha utilizado un template que incluye directamente la nueva contraseña. Esta decisión se toma a solicitud del equipo de desarrollo para mejorar el componente encargado de la parte móvil. Este enfoque asegura que los usuarios de la aplicación móvil reciban su nueva

contraseña de manera rápida y eficiente, facilitando un proceso de recuperación de contraseña más ágil. En la **Figura 3.46** se evidencia la vista del correo.

Figura 3.46 Email para la recuperación de contraseña

Envío de email al crear cita

Se ha desarrollado el envío de emails cuando se crea una cita, se utiliza un template que incluye el día, la fecha exacta en que debe asistir, así como la confirmación de la cita asignada. Esta implementación asegura que los usuarios reciban toda la información necesaria sobre su cita de manera clara y oportuna, mejorando la comunicación y reduciendo la probabilidad de confusiones o ausencias. En la **Figura 3.47** se evidencian la vista del correo mencionado.

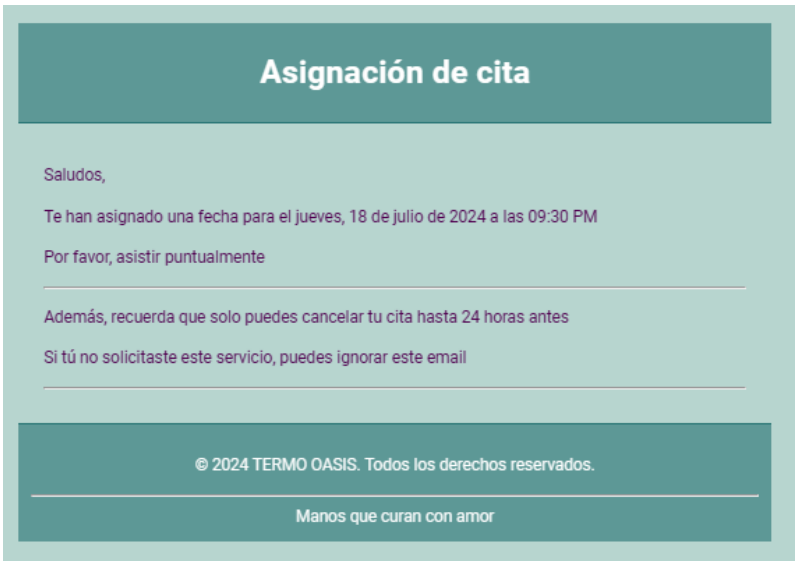

Figura 3.47 Email para el envío de creación de cita

Envío de email al cancelar cita

Para el envío de emails cuando se cancela una cita, se utiliza un template que incluye la fecha de la cita que ha sido cancelada. Esta implementación garantiza que los usuarios estén debidamente informados sobre las cancelaciones, proporcionando claridad y permitiéndoles reorganizar su agenda con la debida antelación. Véase la **Figura 3.48** donde se muestra la vista del correo.

Figura 3.48 Email para el envío de cancelación de cita

Envío de email al actualizar cita

Para el envío de emails cuando se actualiza una cita, se utiliza un template que incluye la nueva fecha y hora de la cita actualizada. Esta implementación asegura que reciban actualizaciones, garantizando que estén al tanto de cualquier cambio en sus citas y puedan ajustarse en consecuencia. En la **Figura 3.49** se evidencia el ejemplo del correo.

Figura 3.49 Email para el envío de actualización de cita

Envío de email para recordatorio para citas

En la implementación actual, la aplicación programa tareas automáticas para verificar periódicamente las citas , esta mismo consulta la base de datos en busca de citas que estén programadas para las próximas 12 horas. Para cada cita encontrada, se genera automáticamente un email, incluyendo la fecha y la hora. Este correo electrónico se envía utilizando un servicio configurado para garantizar un envío seguro y confiable. En la **Figura 3.50** se ve el formato de email que se envía.

Figura 3.50 Email para recordatorio de cita

Sprint 6. Despliegue y pruebas del backend

En este último Sprint se hacen pruebas y el despliegue del componente backend. Esto tiene el propósito de tener un componente que pueda ser utilizado por otros desarrolladores para su consumo y para asegurar su exacto funcionamiento. Durante este sprint se ha realizado las siguientes tareas:

- Pruebas de estrés
- Pruebas de carga
- Pruebas unitarias
- Despliegue en producción

Pruebas de estrés

Por otro lado, las pruebas de estrés llevan al backend más allá de su capacidad normal, sometiéndolo a condiciones extremas de carga para evaluar su resistencia y estabilidad bajo presión máxima. Estas pruebas son fundamentales para descubrir los límites del backend, identificar áreas de vulnerabilidad y preparar medidas preventivas para garantizar su funcionamiento óptimo en situaciones críticas [46].

Se ha realizado una prueba de estrés únicamente en los endpoints que se espera tengan una mayor demanda en el uso del componente. Mientras tanto, los otros endpoints son verificados mediante pruebas unitarias respectivas, comprobando que funcionan adecuadamente bajo una carga normal. Dado que el backend no recibe un alto volumen de usuarios, se espera que el flujo de uso normal no cause problemas de sobrecarga.

Se ha efectuado una prueba de estrés al método de visualización de pacientes con una carga simulada de 200 pacientes, que corresponde al límite máximo para TERMO OASIS según las especificaciones recibidas (véase la **Figura 3.51**). Los resultados revelan que el backend gestiona esta carga sin problemas significativos, demostrando su capacidad para manejar de manera efectiva y estable el volumen máximo esperado de usuarios. En el **ANEXO** II

II se encuentran las pruebas de estrés restantes.

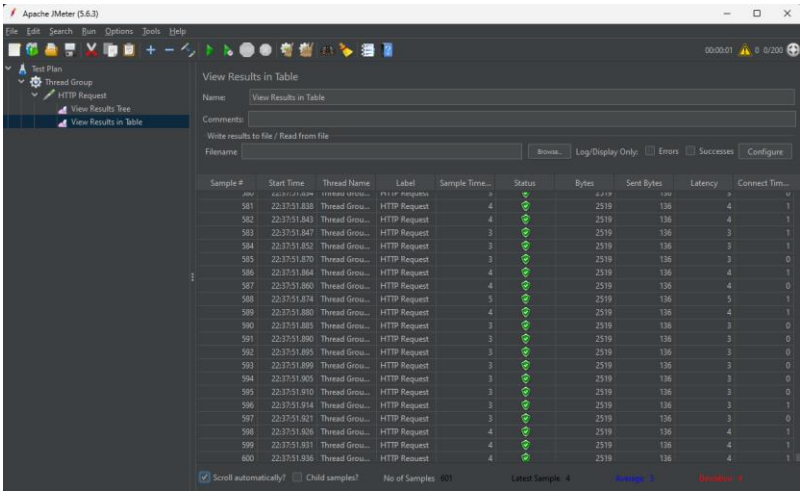

Figura 3.51 Prueba de carga para visualización de pacientes

Pruebas de carga

Permiten comprobar el rendimiento, identificar cuellos de botella, evaluar la escalabilidad y asegurar la capacidad de una aplicación bajo la carga de un número específico de usuarios definido por el numero habitual que se espera que tenga. Al enviar múltiples solicitudes que utilizan la aplicación al mismo tiempo, se puede observar el rendimiento y determinar si la aplicación tiene la capacidad adecuada para manejar la carga prevista [47].

Se realizan pruebas de carga en los endpoints del backend que se anticipa sean los más frecuentados. Se eligen específicamente aquellos con mayor demanda esperada. Los otros endpoints se evalúan mediante pruebas unitarias, confirmando su correcto funcionamiento bajo carga normal. Dado que el backend no maneja un gran volumen de usuarios, el flujo de uso proyectado no causa sobrecarga.

Se ha realizado una prueba de carga para el endpoint de visualizar pacientes, enviando 50 peticiones con 50 usuarios simultáneamente. La prueba ha dado un resultado positivo, ya que todas las peticiones se ejecutan con éxito y no se presenta ningún error (véase la **Figura 3.52**).

Se debe tomar en cuenta que el número de peticiones se basa en los usuarios que comúnmente dispone TERMO OASIS. En el **ANEXO II**

se visualiza las pruebas de carga restantes.

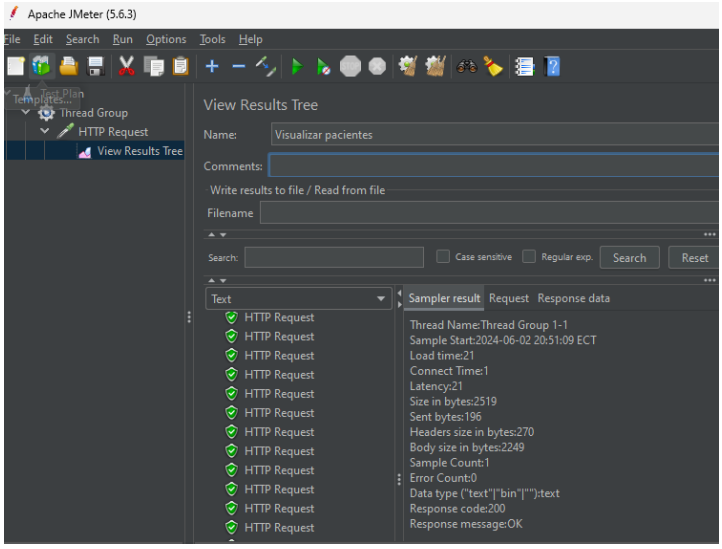

Figura 3.52 Prueba de carga para el método de visualización de pacientes

Pruebas unitarias

Son secciones de código que se invocan para validar su funcionamiento, validando el comportamiento en la unidad del código [48]. Esto significa que prueba específicamente condiciones, entradas, salidas o casos especiales dentro del código, por ende, la función principal es validar el código de manera aislada sin que intervenga el backend completo [49].

Se han realizado pruebas unitarias en cada endpoint y se han detallado en cada sprint respectivamente. Los resultados exitosos de estas pruebas indican que el código es eficiente y funciona correctamente de manera independiente del backend. Esto asegura que cada componente cumple con los requisitos establecidos y se comporta según lo esperado.

Despliegue en producción

El despliegue de nuestro backend se ha creado en Vercel, aprovechando sus múltiples beneficios como la facilidad de integración con nuestro flujo de trabajo, la escalabilidad automática, y la capacidad de manejar peticiones de manera eficiente. Vercel nos proporciona un entorno optimizado para el desarrollo y despliegue continuo, garantizando alta disponibilidad y rendimiento. Gracias a estas ventajas, se logra un proceso de despliegue simplificado y fiable como se ve en la **Figura 3.53**, culminando en la siguiente API RESTful disponible con la siguiente url:

https://backend-termo-oasis.vercel.app/

Además, para utilizar esta API RESTful, se ha creado la documentación respectiva con Swagger. Esta documentación está disponible en el siguiente enlace:

https://backend-termo-oasis.vercel.app/api-docs

Además, las instrucciones sobre el uso y el manual lo encontrara en el **ANEXO III**

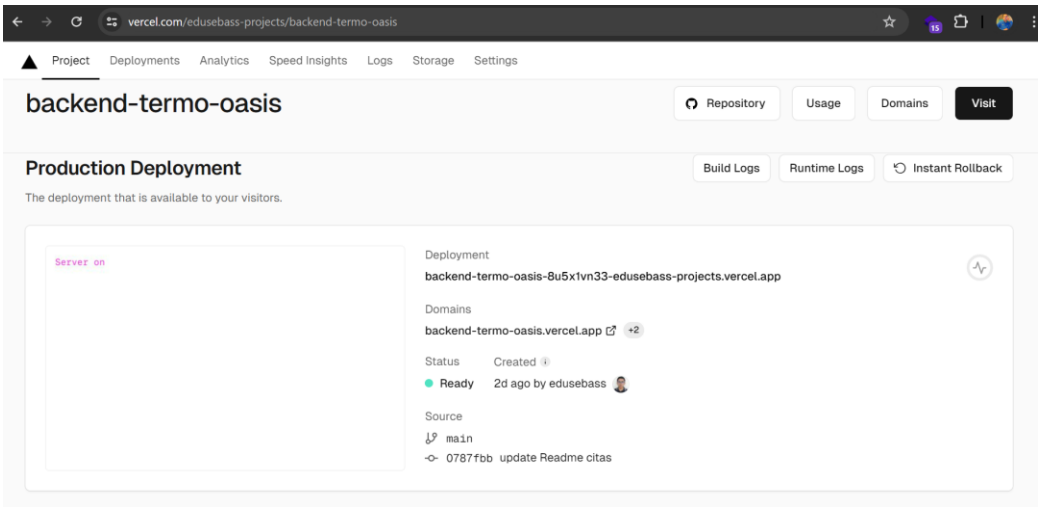

Figura 3.53 Despliegue en Vercel

4 CONCLUSIONES

- Al determinar los requerimientos, se ha logrado establecer una base sólida para el proyecto. Este proceso ha involucrado la identificación de todas las funcionalidades necesarias, las interacciones del usuario, y las integraciones con otros sistemas.
- La elaboración de la base de datos y la implementación de su arquitectura han sido claves para asegurar una gestión de datos eficiente y escalable, además la adopción de la arquitectura (MVC) establece backend organizado y robusto.
- El desarrollo de los endpoints ha proporcionado una interfaz accesible para la interacción con el componente. Además, el desarrollo de estos endpoints ha sido fundamental para asegurar la comunicación entre el frontend y el backend
- Las pruebas de carga, estrés y unitarias realizadas han demostrado que el backend actual maneja eficientemente las solicitudes bajo la carga prevista, sin errores y con éxito. Estos resultados aseguran el buen funcionamiento continuo del componente en condiciones de uso normales y garantizan una experiencia de usuario fluida y confiable.
- La publicación del backend en el ambiente de producción marca la finalización del de desarrollo y las pruebas. Este paso ha implicado la configuración de un entorno de producción seguro, escalable y eficiente.

5 RECOMENDACIONES

- Para asegurar la preparación del backend para futuras expansiones y mejorar su escalabilidad, se recomienda adoptar una arquitectura basada en microservicios, implementar estrategias flexibles de gestión de datos utilizando tecnologías NoSQL, y establecer más mecanismos robustos de seguridad.

- Para consideraciones futuras, es importante tener en cuenta que MongoDB, en su versión gratuita, tiene limitaciones en cuanto a almacenamiento y rendimiento. A medida que el backend crece y se expande, se debe evaluar la transición a una versión comercial de MongoDB o explorar alternativas de bases de datos que puedan ofrecer mayores capacidades y soporte.

- Se recomienda revisar regularmente el código para identificar áreas que puedan optimizarse, como buscar mejores maneras de seguridad para el control de roles, mejorar la eficiencia de consultas a la base y aplicar patrones adecuados como la reutilización de código.

- Con base al despliegue del backend, se recomienda considerar otro sitio para alojarlo, ya que, aunque Vercel es una excelente herramienta, puede no ser la mejor opción a largo plazo si el número de usuarios incrementa.

6 REFERENCIAS

[1] OPS, «OPS/OMS | Organización Panamericana de la Salud,» 14 Julio 2023. [En línea]. Available: https://www.paho.org/es/temas/evaluacion-tecnologias-salud. [Último acceso: 15 Julio 2024].

[2] F. Doglio, REST API Development with Node. js, Apress, 2018.

[3] C. A. Bernal Torres, Metodología de la investigación, Pearson Educación, 2006.

[4] O. A. Zapata, Metodología de la investigación, PAX MEXICO, 2005.

[5] OpenJS Foundation, «NodeJs,» [En línea]. Available: https://nodejs.org. [Último acceso: 30 abril 2024].

[6] K. Chodorow, MONGO DB The Definitive Guide, California: O`Reilly Media, Inc, 2014.

[7] F. Doglio, REST API Development with Node.js, La Paz: Apress, 2015.

[8] J. Pérez y A. Gardey, «Metodología - Qué es, definición, tipos y teoría,» Definicion.DE, 19 08 2021. [En línea]. Available: https://definicion.de/metodologia/. [Último acceso: 01 mayo 2024].

[9] W. O. L. González, El estudio de casos:, Mérida: educere, 2013.

[10] PZT by Posizionarte, «PZT,» 3 6 2021. [En línea]. Available: https://pzt.es/metodologias-de-desarrollo-de-software/. [Último acceso: 02 mayo 2024].

[11] Scrum.org, «What is a Product Owner?,» [En línea]. Available: https://www.scrum.org/resources/what-is-a-product-owner. [Último acceso: 03 mayo 2024].

[12] T. Dimes, Conceptos Básicos de Scrum: Desarrollo de software Agile y manejo de proyectos Agile, Babelcube, Inc, 2015.

[13] Scrum.org, «What is a Scrum Master?,» [En línea]. Available: https://www.scrum.org/resources/what-is-a-scrum-master. [Último acceso: 03 mayo 2024].

[14] J. Sutherland, Scrum El arte de hacer el doble de trabajo en la mitad de tiempo, Océano, 2016.

[15] A. J. Canosa Ferreiro, SCRUM. Teoría e Implementación práctica, RA-MA S.A. Editorial y Publicaciones, 2024.

[16] R. Gómez Blanes, El Libro Práctico del Programador Ágil, Amazon Digital Services LLC - KDP Print US, 2019.

[17] J. O. Hernandez Lopez, Scrum Master: Fundamentos Actuales, Scrumbee, 2024.

[18] D. Radigan, «Product backlog grooming,» [En línea]. Available: https://www.atlassian.com/agile/scrum/backlogs#:~:text=A%20product%20backlog%20is%20a,knows%20what%20to%20deliver%20first.. [Último acceso: 06 mayo 2024].

[19] Asana, «What is a Sprint Backlog? Create With Examples [2024] • Asana,» [En línea]. Available: https://asana.com/resources/sprint-backlog. [Último acceso: 06 mayo 2024].

[20] P. Clements, L. Bass y R. Kazman, Software Architecture in Practice, Addison-Wesley, 2003.

[21] F. Valdes, «La guía definitiva de la arquitectura de software,» [En línea]. Available: https://medium.com/@ktufernando/la-gu%C3%ADa-definitiva-de-la-arquitectura-del-software-f419db9c6bf7. [Último acceso: 06 mayo 2024].

[22] E. Teniente López, A. Olivé Ramon, E. Mayol Sarroca y C. Gómez Seone, Diseño de sistemas software en UML, Universitat Politecnica de Catalunya. Iniciativa Digital Politecnica, 2004.

[23] D. Voorhees, Guide to Efficient Software Design, Springer International Publishing, 2020.

[24] Microsoft, «Visual Studio Code - Code editing. Redefined,» 11 Noviembre 2021. [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 11 mayo 2024].

[25] Thunder Client, «Thunder Client - Rest API client extension for VS code,» [En línea]. Available: https://www.thunderclient.com/. [Último acceso: 11 mayo 2024].

[26] Vercel, «Vercel,» [En línea]. Available: https://vercel.com. [Último acceso: 11 mayo 2024].

[27] Pluralsight, «Learn JavaScript online - Courses for beginners - Javascript.com,» [En línea]. Available: https://www.javascript.com/. [Último acceso: 11 mayo 2024].

[28] SmartBear Software, «API documentation & design tools for teams | Swagger,» [En línea]. Available: https://swagger.io/. [Último acceso: 11 mayo 2024].

[29] GitHub, Inc., «GitHub: Let’s build from here,» [En línea]. Available: https://github.com/. [Último acceso: 11 mayo 2024].

[30] A. Reinman, «Nodemailer :: Nodemailer,» [En línea]. Available: https://www.nodemailer.com/. [Último acceso: 11 mayo 2024].

[31] motdotla, «npm: dotenv,» [En línea]. Available: https://www.npmjs.com/package/dotenv. [Último acceso: 11 mayo 2024].

[32] mongoose, «Mongoose ODM V8.3.4,» [En línea]. Available: https://mongoosejs.com/. [Último acceso: 11 mayo 2024].

[33] Auth0, «NPM: JsonwebToken,» [En línea]. Available: https://www.npmjs.com/package/jsonwebtoken. [Último acceso: 11 mayo 2024].

[34] OpenJS Foundation, «Express - Node.js Web Application Framework,» [En línea]. Available: https://expressjs.com/. [Último acceso: 11 mayo 2024].

[35] Troy Goode, «NPM: Cors,» [En línea]. Available: https://www.npmjs.com/package/cors#author. [Último acceso: 11 mayo 2024].

[36] S. Koss, «Modern JavaScript Date Utility Library,» [En línea]. Available: https://date-fns.org/. [Último acceso: 21 mayo 2024].

[37] Surnet, «npmjs,» [En línea]. Available: https://www.npmjs.com/package/swagger-jsdoc. [Último acceso: 29 junio 2024].

[38] S. Scott, «npmjs,» [En línea]. Available: https://www.npmjs.com/package/swagger-ui-express. [Último acceso: 29 junio 2024].

[39] Node Cron, «Node Cron,» [En línea]. Available: https://www.npmjs.com/package/node-cron. [Último acceso: 11 julio 2024].

[40] OpenJS Foundation, «JEST,» [En línea]. Available: https://jestjs.io/. [Último acceso: 11 julio 2024].

[41] ladjs, «SuperTest,» [En línea]. Available: https://www.npmjs.com/package/supertest. [Último acceso: 11 julio 2024].

[42] H. Ñaupas Paitán, M. R. Valdivia Dueñas, J. J. Palacios Vilela y H. E. Romero Delgado, Metodología de la Investigación cuantitativa-cualitativa y redacción de la tesis, Ediciones de la U, 2019.

[43] A. Giamas, Mastering MongoDB 3.x An Expert's Guide to Building Fault-tolerant MongoDB Applications, Packt Publishing, 2017.

[44] . N. Sebhastian , Node.js Web Development For Beginners, Amazon Digital Services LLC - Kdp, 2024.

[45] G. Sharma, NodeJS and Asynchronous Backend - Rapid API Development Guide with NodeJS, Gunjan Sharma, 2024.

[46] J. M. Ortega Candel, Desarrollo seguro en ingeniería del software, Alpha Editorial, 2020, p. 354.

[47] M. L. Peláez Recios, UF1889 - Desarrollo de componente software en sistemas ERP-CRM, Editorial Elearning, S.L., 2015.

[48] Natsys, Testing Convertite en un experto en QA, Natsys, 2017.

[49] J. Dolado Cosín y I. Ramos Román, Técnicas Cuantitativas para la Gestión en la Ingeniería del Software., Netbiblo, 2007.

[50] The Blokehead, The blokehead series SCRUM !Guía Definitiva de Prácticas Ágiles Esenciales de Scrum, Babelcube, Inc, 2016.

7 ANEXOS

ANEXO I. Certificado de originalidad

ANEXO II. Manual técnico

ANEXO III. Manual de usuario

ANEXO IV. Manual de instalación

ANEXO I

F_AA_236

CERTIFICADO DE ORIGINALIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR

Quito, D.M. 02 de agosto de 2024

De mi consideración:

Yo, Juan Pablo Zaldumbide Proaño, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DE BACKEND asociado al DESARROLLO DE SISTEMA DE GESTIÓN DE CITAS MÉDICAS PARA EL CENTRO DE TERAPIAS ALTERNATIVAS TERMO OASIS elaborado por el estudiante ALMACHI MAISINCHO EDUARDO SEBASTIAN de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta antiplagio "TURNITIN" para la revisión de originalidad del documento escrito producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 9%.

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.

Juan Pablo Zaldumbide Proaño
Docente Ocasional a Tiempo Completo
ESFOT

ANEXO II

Recopilación de requerimientos

Se definen todos los requisitos de antemano antes de iniciar con el componente y todos los mismos se encuentran en la **Tabla 1**.

Tabla 1 Recopilación de requerimientos

Recopilación de requerimientos		
TIPO DE SISTEMA	**ID-RR**	**ENUNCIADO DEL ÍTEM**
Backend	RR001	El paciente, doctor y secretaria va a poder iniciar sesión, cerrar sesión, recuperar contraseña.
	RR002	Como paciente puede visualizar sus citas .
	RR003	Como paciente puede cancelar su cita solo una día antes de la misma.
	RR004	Como doctor puede visualizar las citas.
	RR005	Como doctor puede visualizar cada cita de paciente.
	RR006	Como doctor puede actualizar, visualizar y crear la ficha medica de la cita solo en el tiempo establecido de la misma.
	RR007	Como secretaria puede crear, eliminar, y visualizar pacientes.
	RR008	Como secretaria puede visualizar, actualizar y cancelar citas.
	RR009	Al momento de crear una cita se debe enviar un correo de información cuando: • La cita es creada • La cita es cancelada • La cita es editada • La cita esta próxima
	RR010	La ficha medica tiene los campos que el cliente solicito.

Historias de Usuario

Las historias de usuario se han creado después de documentar los requisitos previamente y están documentados desde **Tabla 2** hasta la **Tabla 7**.

Tabla 2 Historia de usuario #1

HISTORIA DE USUARIO	
Identificador(ID): HU001	**Usuario:** Todos los roles
Nombre Historia: Gestión de usuarios	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Baja
Interacción Asignada: 1	
Responsable: Eduardo Sebastián Almachi Maisincho	
Descripción: En este módulo de usuarios la API se encargar de gestionar los usuarios y darles el rol correspondiente.	
Observación: Cada rol tiene diferentes funcionalidades las cuales dependen de los requerimientos previamente definidos.	

Tabla 3 Historia de usuario #2

HISTORIA DE USUARIO	
Identificador(ID): HU002	**Usuario:** Paciente
Nombre Historia: Rol paciente	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Media
Interacción Asignada: 1	
Responsable: Eduardo Sebastián Almachi Maisincho	
Descripción: Endpoints que permitan registrar un paciente y dar recordatorios respectivos de sus citas a cada paciente en específico.	
Observación: Cada paciente puede ver sus citas y tener acceso a sus datos.	

Tabla 4 Historia de usuario #3

HISTORIA DE USUARIO	
Identificador(ID): HU003	**Usuario:** Secretaria
Nombre Historia: Rol Secretaria	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Media
Interacción Asignada: 1	
Responsable: Eduardo Sebastián Almachi Maisincho	
Descripción: Endpoints que permitan todos los requerimientos de que debe realizar la secretaria.	
Observación: La secretaria puede gestionar las citas y usuarios.	

Tabla 5 Historia de usuario #4

HISTORIA DE USUARIO	
Identificador(ID): HU004	**Usuario:** Todos los roles
Nombre Historia: Rol Doctor	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Media
Interacción Asignada: 1	
Responsable: Eduardo Sebastián Almachi Maisincho	
Descripción: Endpoints que cumplan con lo requerido del rol doctor.	
Observación: El doctor puede visualizar las citas que tiene y por paciente además de poder gestionar las fichas medicas	

Tabla 6 Historia de usuario #5

HISTORIA DE USUARIO	
Identificador(ID): HU005	**Usuario:** Todos los roles
Nombre Historia: Envío de emails	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Media

Interacción Asignada: 1
Responsable: Eduardo Sebastián Almachi Maisincho
Descripción: Endpoints para enviar emails de notificación cada vez que se cumpla con los requerimientos previamente indicados.
Observación: En este caso solo hay casos específicos para mandar emails de notificación.

Tabla 7 Historia de usuario #6

HISTORIA DE USUARIO	
Identificador(ID): HU006	**Usuario:** Todos los roles
Nombre Historia: Pruebas y despliegue en producción	
Prioridad Negocio: Baja	**Riesgo en desarrollo:** Media
Interacción Asignada: 1	
Responsable: Eduardo Sebastián Almachi Maisincho	
Descripción: Para el consumo de la api en producción se necesita que se la despliegue para el uso de los desarrolladores además de realizar pruebas para verificar su funcionamiento.	
Observación: El despliegue en producción se hace en un sitio donde no cueste y sea eficaz para la rapidez de respuesta.	

Product Backlog

El Product Backlog ayuda a determinar y visualizar que historias de usuario tienen prioridad. Esto se encuentra detallado en la **Tabla 8**.

Tabla 8 Product Backlog

ID-PB	Historia de Usuario	Prioridad	Iteración	Estado
HUB001	Gestión de usuarios	Baja	1	Terminada
HUB002	Rol paciente	Media	2	Terminada
HUB003	Rol Secretaria	Alta	3	Terminada
HUB004	Rol Doctor	Media	4	Terminada

HUB005	Envío de emails	Media	5	Terminada
HUB006	Pruebas y despliegue en producción	Alta	6	Terminada

Sprint Backlog

Este proyecto consta de 5 sprints, el sprint backlog organiza las historias de usuario en sprints, estos se detallan claramente en la **Tabla 9**

Tabla 9 Sprint Backlog

ID-SB	NOMBRE	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB000	Configuración del ambiente de desarrollo	-	-	• Creación del diseño y estructura del modelo de base de datos. • Establecer la estructura del entorno de desarrollo. • Determinación de roles.	20H
SB001	Autenticación	HU001 HU002 HU003 HU004	Gestión de usuarios Rol Paciente Rol Secretaria Rol Doctor	• Endpoint para el registro de usuario • Endpoint para el inicio de sesión • Endpoints para recuperación de contraseña • Protección de rutas	40H

SB002	Módulo usuarios	HU002 HU003 HU004	Rol Paciente Rol Secretaria Rol Doctor	• Endpoint para crear usuario • Endpoint para eliminar usuario • Endpoint para detalle de usuario	20H
SB003	Módulo citas	HU002 HU003 HU004	Rol Paciente Rol Secretaria Rol Doctor	• Endpoint para crear cita • Endpoint para editar cita • Endpoint para cancelar cita • Endpoint para editar cita • Endpoint para mostrar todas las citas • Endpoint para mostrar una cita en especifico • Endpoint para mostrar todas las citas de un paciente	60H
SB004	Módulo fichas medicas	HU004	Rol Doctor	• Endpoint para actualizar fichas medicas • Endpoint crear fichas medicas	40H

SB005	Email de notificaciones	HU005	Implementación de Lógica de Notificación de Emails	• Envío de email al crear cita • Envío de email al cancelar cita • Envío de email al actualizar cita • Envío de email al actualizar contraseña (frontend y móvil) • Envío de recordatorio	40h
SB006	Despliegue y pruebas del backend	HU006	Despliegue en producción	• Despliegue del backend en Vercel • Pruebas de estrés • Pruebas de carga • Pruebas unitarias	20H
TOTAL					240H

Diseño de BDD

En esta sección se anexan las colecciones utilizadas. Estas se emplean como conjuntos de documentos en MongoDB, una base de datos NoSQL, y se detallan a continuación desde la **Figura 1** hasta la **Figura 3**.

Figura 1 Colección modelo Usuario

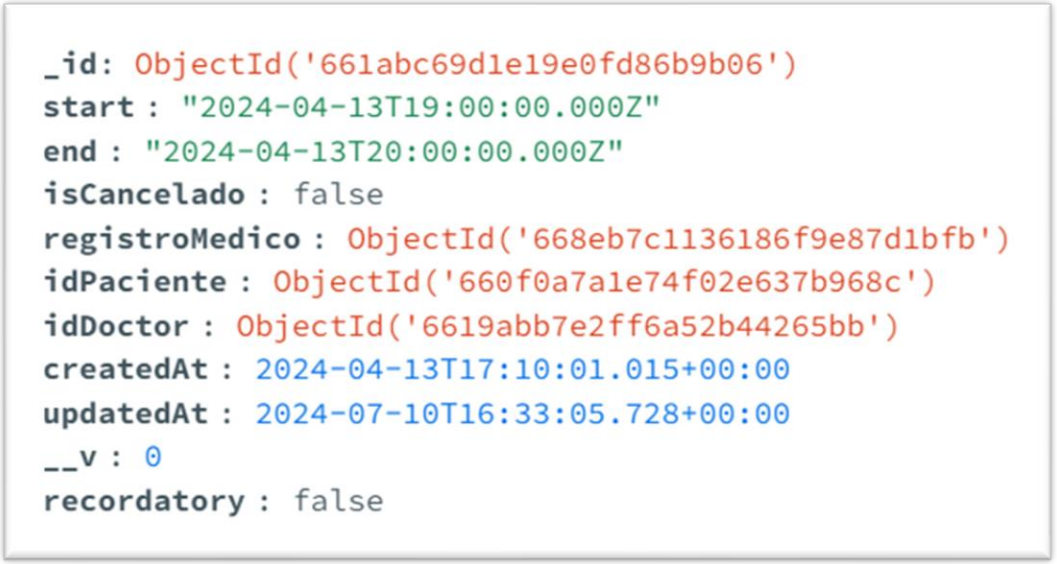

Figura 2 Colección modelo Cita

Figura 3 Colección modelo Registro médico

Pruebas

En este apartado del anexo, se detallan todas las pruebas de carga y estrés llevadas a cabo, con especial consideración a los módulos que se prevé tienen un mayor consumo y relevancia, como se mencionó previamente.

Pruebas de estrés

Cada prueba se la realizo con una simulación de 200 peticiones por 200 usuarios.

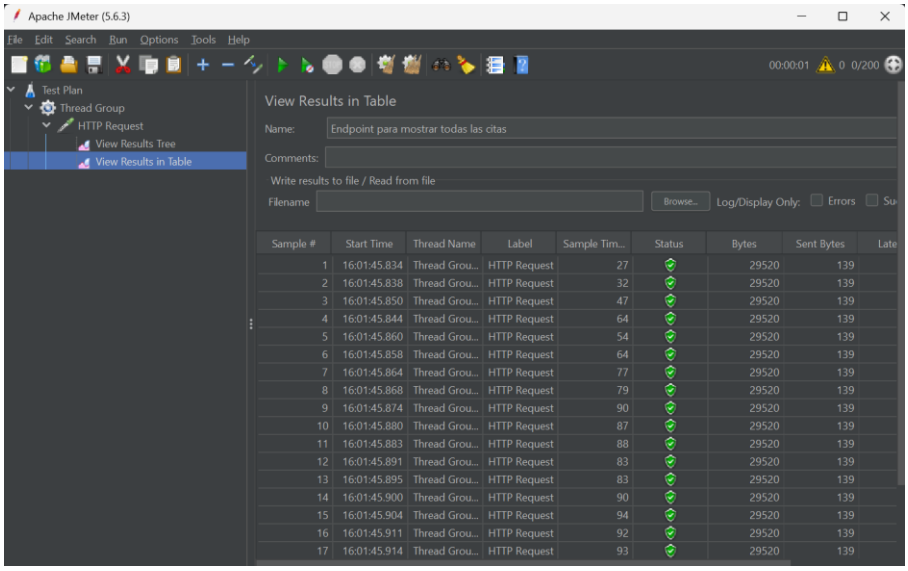

Figura 4 Prueba de estrés para el endpoint de mostrar citas

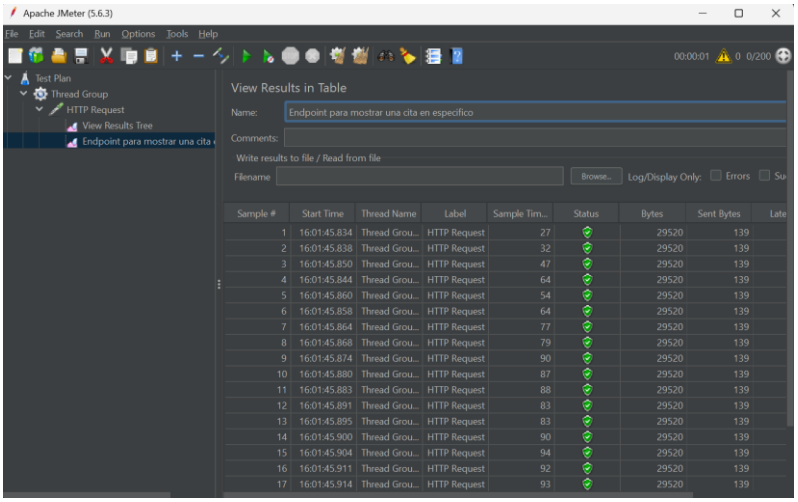

Figura 5 Prueba de estrés para el endpoint de mostrar una cita en especifico

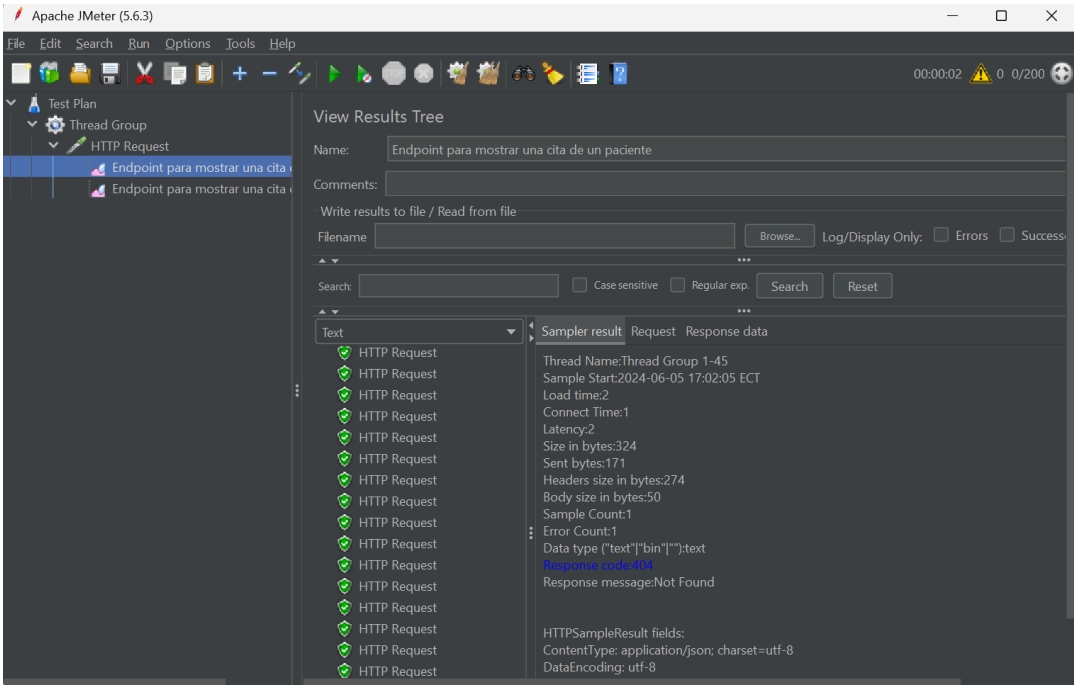

Figura 6 Prueba de estrés para el endpoint de mostrar una cita de un paciente

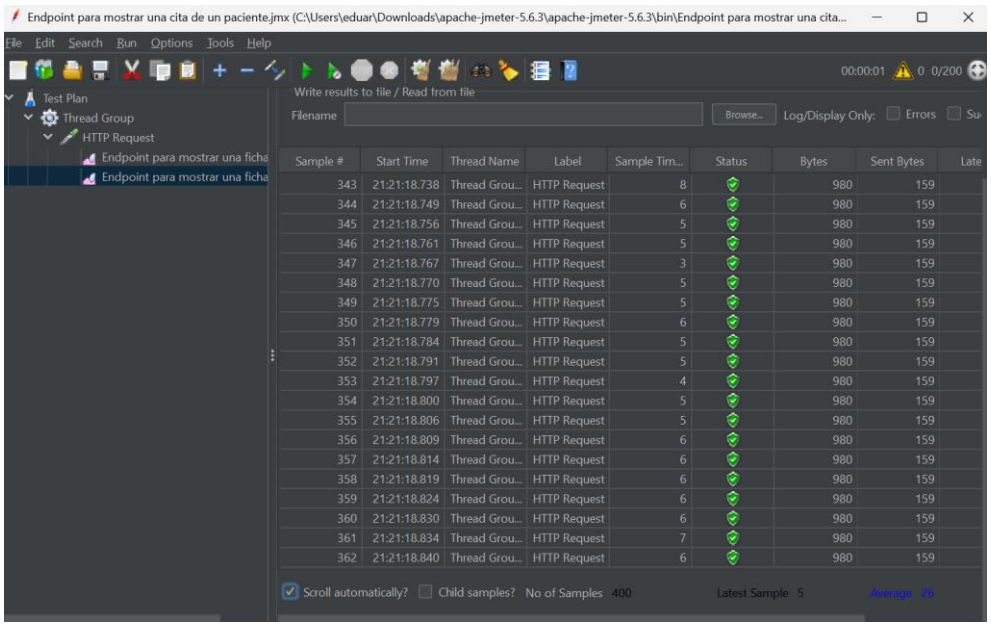

Figura 7 Prueba de estrés para el endpoint de mostrar ficha médica de un paciente

Pruebas de carga

Cada prueba se la realizo con una simulación de 50 peticiones por 50 usuarios

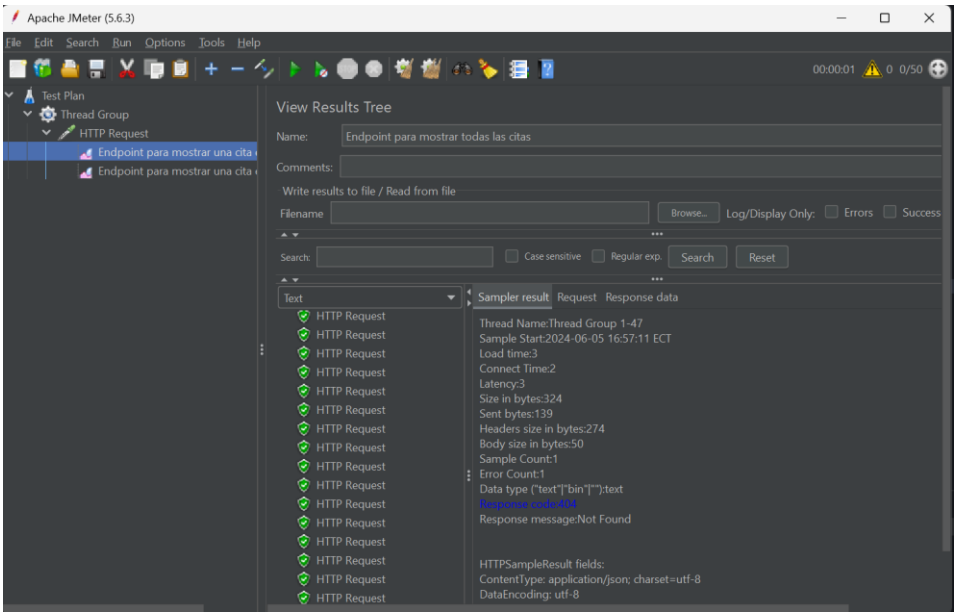

Figura 8 Prueba de carga para el endpoint de mostrar todas las citas

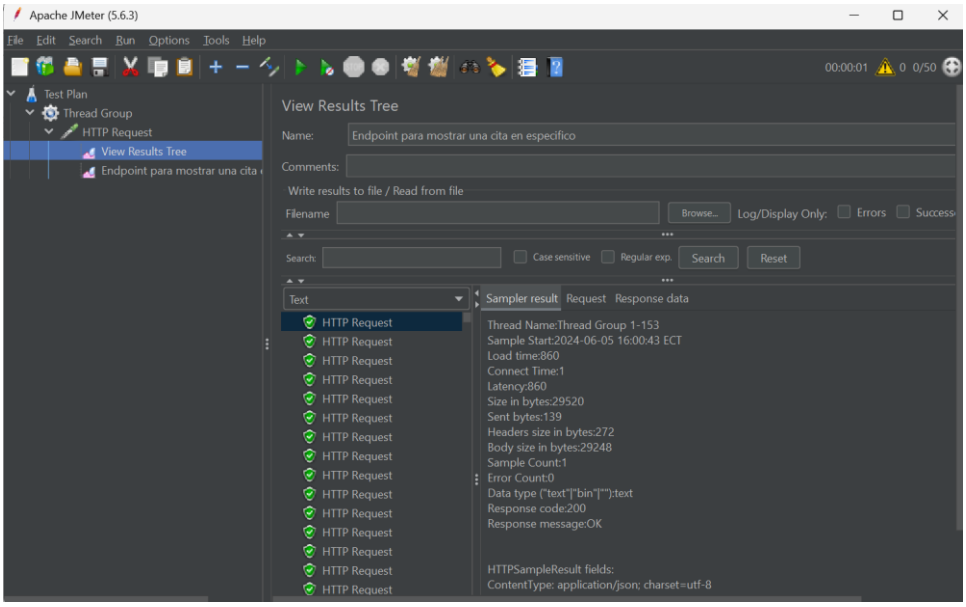

Figura 9 Prueba de carga el endpoint de mostrar una cita en especifico

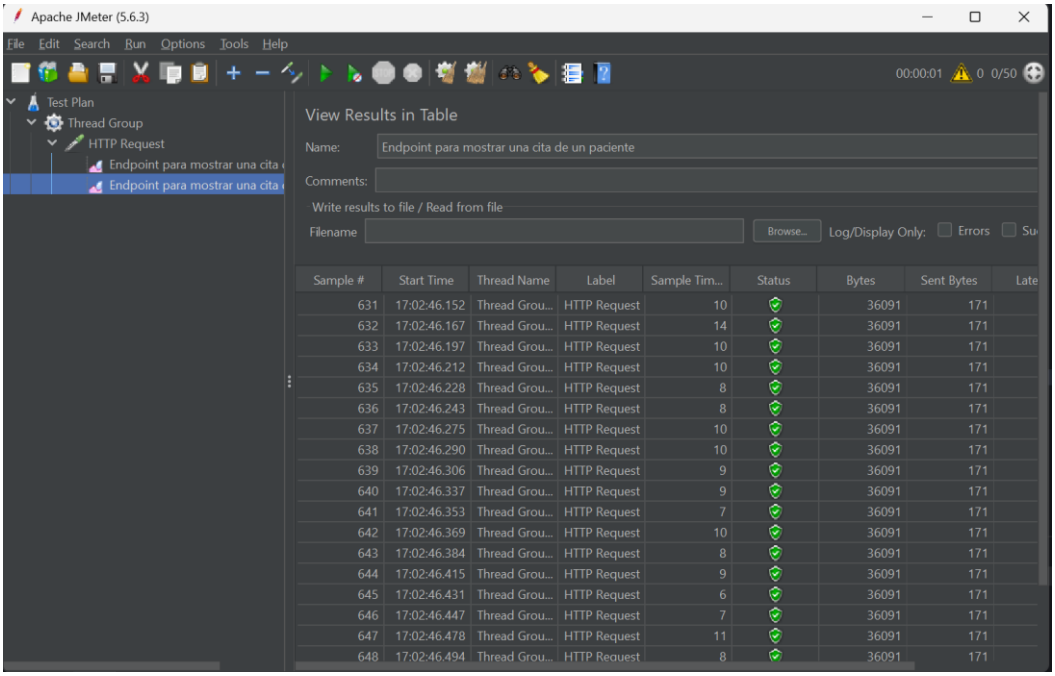

Figura 10 Prueba de carga para el endpoint de mostrar una cita de un paciente

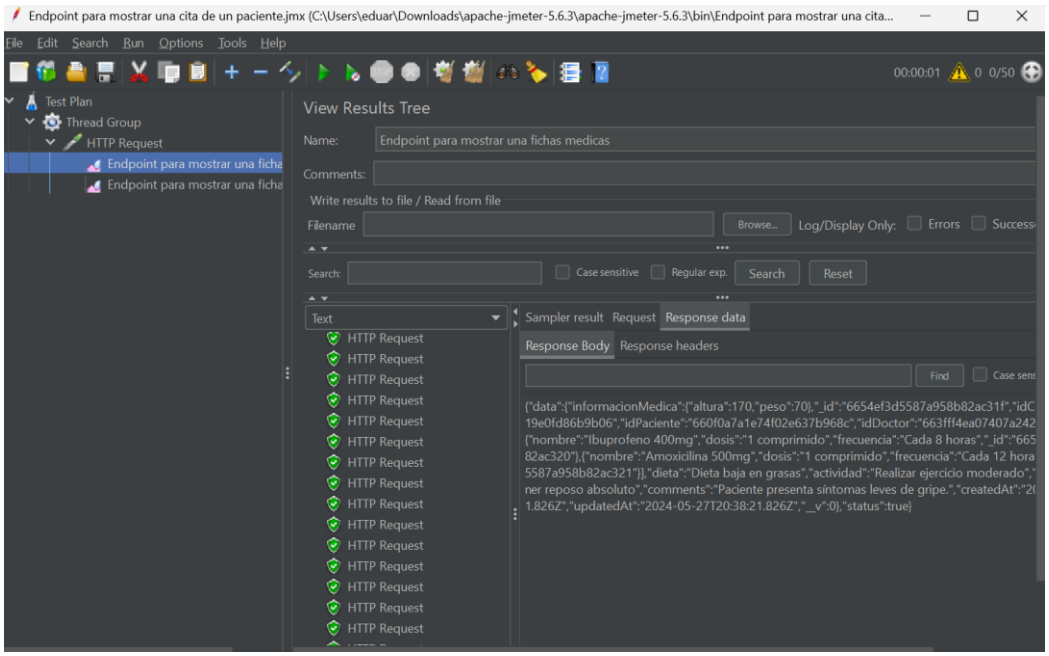

Figura 11 Prueba de carga para el endpoint de mostrar ficha medica de un paciente

Esta parte del anexo evidencia el certificado de la empresa en cuanto a la aplicación de sistema de gestión de citas

TERMO OASIS

Quito, 11 de Julio de 2024

CERTIFICADO

Yo, **Ángel Oswaldo Avilés Merino**, con cédula de identidad **0604001784**, en mi calidad de Gerente de TERMO OASIS, certifico:

Que el Sr. **Eduardo Sebastián Almachi Maisincho**, con cédula de identidad **1751395623**, de la carrera de Tecnología Superior en Desarrollo de Software, ha cumplido con todos los requisitos establecidos en su totalidad. Durante el desarrollo de este proyecto, ha demostrado un alto nivel de competencia técnica y dedicación en la creación de una aplicación conforme a las especificaciones solicitadas.

El proyecto incluye la implementación de funcionalidades específicas que van de acorde con lo que ofrecemos. Además, se ha mantenido una comunicación efectiva respecto a los objetivos y alcances de su trabajo, mostrando un claro entendimiento de las necesidades del proyecto y asegurando la calidad y funcionalidad del sistema desarrollado.

Atentamente,

Ángel Oswaldo Avilés Merino

C.I. 0604001784

Oswaldo Aviles
Termo Oasis - Quiropraxia

"Contribuyente Negocio
Popular Régimen RIMPE"

ANEXO III

Manual de usuario

En esta sección se presenta el manual de usuario, la cual explica la funcionalidad del componente backend que para visualizarlo puede entrar al siguiente link:

https://youtu.be/KoxwnB7w88Y

ANEXO IV

En esta sección se detallan, las credenciales de acceso y el link del repositorio de GitHub.

Credenciales de acceso

Para acceder con las credenciales se adjunta el backend en producción:

https://backend-termo-oasis.vercel.app/

Usuario doctor:

- oswaldo.aviles_TO@outlook.com
- is7T&O_doc

Usuario secretaria:

- alexandra.quin_TO@outlook.com
- T&Osecre_is6

Usuario paciente:

- pacienteTermoOasis@outlook.com
- pacienteTermo123_

Link del repositorio:

Para acceder al código se adjunta el link del repositorio de GitHub:

https://github.com/edusebass/backendTermoOasis