ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UN SISTEMA PARA LA GESTIÓN ESTUDIANTIL DEL I.E.F MIGUEL DE SANTIAGO

DESARROLLO DE UN *BACKEND*

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

RAÚL ALEJANDRO TENORIO MENDIETA

DIRECTOR: MALDONADO SOLIZ IVONNE FERNANDA

DMQ, febrero 2023

CERTIFICACIONES

Yo, Raúl Alejandro Tenorio Mendieta 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.

RAÚL ALEJANDRO TENORIO MENDIETA

raul.tenorio@epn.edu.ec

tenorio0967@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Raúl Alejandro Tenorio Mendieta, bajo mi supervisión.

Ing. Ivonne Maldonado, MSc.

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.

Raúl Alejandro Tenorio Mendieta

DEDICATORIA

Quiero expresar mi más profunda gratitud por el apoyo incondicional que me han brindado durante este largo camino hacia la culminación de mi proyecto de titulación. Su ayuda y ánimo han sido esenciales para alcanzar este logro. A mis padres, Raúl Tenorio e Inés Mendieta, les debo todo lo que soy. Gracias por haberme guiado, apoyado y enseñado en cada momento de mi vida. Su dedicación, sacrificio y amor han sido la mayor inspiración para alcanzar este objetivo. Siempre estaré agradecido por todo lo que han hecho por mí.

A mis hermanos, Anthony Tenorio y Guillermo Tenorio, les agradezco por ser mis cómplices, amigos y confidentes. Gracias por haberme alentado en los momentos difíciles y por compartir conmigo los momentos felices. Sin ustedes, este camino no habría sido tan agradable. A mis amigos, Leonel Molina, Ariel Calderón, Manuel Auqui y Ernesto Salazar, les doy las gracias por estar siempre presentes en mi vida, por apoyarme y animarme en los momentos más difíciles. Agradezco las risas, las charlas y los buenos momentos que hemos compartido. Gracias por hacer de esta etapa una experiencia inolvidable.

Espero que este proyecto sea un tributo a su esfuerzo, dedicación y amor. Este logro no solo es mío, sino también de todos ustedes. Gracias por creer en mí y por estar a mi lado en todo momento.

Raúl Alejandro Tenorio Mendieta

AGRADECIMIENTO

En este momento tan importante de mi vida, quiero expresar mi más sincero agradecimiento por el apoyo incondicional durante la realización de mi proyecto de titulación. A mis padres, Raúl Tenorio e Inés Mendieta, les doy las gracias por su amor, paciencia y dedicación. Su confianza y apoyo han sido fundamentales en todo momento. Gracias por haberme brindado las herramientas necesarias para lograr mis metas y por haber estado a mi lado en todo momento.

A mis hermanos, Anthony Tenorio y Guillermo Tenorio, les agradezco por haberme acompañado y apoyado en todo momento. Gracias por las risas, las charlas y los buenos momentos que hemos compartido. Su presencia ha sido esencial en la realización de este logro. A mis amigos, Leonel Molina, Ariel Calderón, Manuel Auqui y Ernesto Salazar, les doy las gracias por haberme apoyado y animado en los momentos más difíciles. Su amistad, confianza y cariño han sido un gran estímulo en todo momento. Gracias por haberme acompañado en este camino y por haber compartido conmigo esta experiencia.

A mi tutora, Ivonne Maldonado, le agradezco por su guía, dedicación y paciencia. Gracias por haberme brindado las herramientas necesarias para lograr este objetivo. Sin su ayuda y orientación, este logro no habría sido posible.

Raúl Alejandro Tenorio Mendieta

ÍNDICE DE CONTENIDO

RESUMEN

Un sistema *web* de gestión escolar es una plataforma en línea que permite a las instituciones educativas administrar de manera más eficiente sus procesos académicos y administrativos. Algunas de sus características principales son que pueden incluir la automatización de tareas como la matrícula de estudiantes, la generación de reportes dinámicos y el control sobre las calificaciones de los estudiantes. La implementación de un sistema *web* de gestión escolar puede tener beneficios para una institución educativa como mejorar la eficiencia y la precisión en la gestión de los datos, lo que reduce la carga de trabajo de los administradores escolares y permite una toma de decisiones más informada y rápida.

El I.E.F Miguel de Santiago se encuentra en un dilema debido a que el sistema que mantienen actualmente no cumple con los requerimientos necesarios para poder llevar a cabo una gestión académica adecuada. En este sentido, con el fin de apoyar a la institución se ha desarrollar un sistema *web* que busca cumplir con las actividades de gestión académicas necesarias. Para ello se han generado varios *endpoints* con el fin de que el proyecto pueda ser utilizado tanto en un cliente *web* como un móvil, de esta manera la institución gracias a la tecnología puede contar con un medio digital adecuado.

El Trabajo de Integración Curricular actual está organizado de la siguiente manera: en primer lugar, se proporcionan detalles sobre los antecedentes, objetivos y el alcance del proyecto, junto con el marco teórico. A continuación, se describe la implementación de la metodología *Scrum*, así como sus artefactos, arquitectura y herramientas utilizados para desarrollar los *endpoints*. En la tercera sección, se presentan las actividades por iteración y los resultados de cada *Sprint*. Por último, en la cuarta sección se especifican las conclusiones y recomendaciones que se han obtenido a lo largo de este trabajo.

PALABRAS CLAVE: Institución educativa, sistema *web*, *endpoints,* gestión, *Scrum*.

ABSTRACT

A school management web system is an online platform that allows educational institutions to more efficiently manage their academic and administrative processes. Some of its main features may include the automation of tasks such as student enrollment, the generation of dynamic reports, and control over student grades. Implementing a school management web system can have benefits for an educational institution such as improving efficiency and accuracy in data management, reducing the workload of school administrators, and enabling faster, more informed decision-making.

The I.E.F Miguel de Santiago finds itself in a dilemma because the system they currently maintain does not meet the necessary requirements to carry out adequate academic management. In this sense, in order to support the institution, a web system has been developed that seeks to meet the necessary academic management activities. To achieve this, several endpoints have been generated so that the project can be used both on a web client and on a mobile device, thus enabling the institution to have an appropriate digital medium thanks to technology.

The current Curricular Integration Work is organized as follows: firstly, details are provided on the background, objectives, and scope of the project, along with the theoretical framework. Next, the implementation of the Scrum methodology is described, as well as its artifacts, architecture, and tools used to develop the endpoints. In the third section, the activities per iteration and the results of each Sprint are presented. Finally, in the fourth section, the conclusions and recommendations obtained throughout this work are specified.

KEYWORDS: Educational institution, web system, endpoints, management, Scrum.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Una adecuada gestión escolar en los centros educativos permite mejorar la calidad de todas las acciones y procesos administrativos que se llevan a cabo diariamente. Su importancia radica en que influye directamente en los resultados y flujos de trabajo. Sin embargo, varias de estas instituciones pasan por alto las ventajas y beneficios que trae consigo el tener un sistema automatizado que les permita ejecutar sus actividades de manera óptima y sencilla [1]. Desaprovechando la posibilidad del máximo rendimiento administrativo del centro educativo y en el peor de lo casos en entorpecimiento de las diferentes actividades que pudiesen realizarse en un menor tiempo y sin tanto esfuerzo. Actividades diarias como: generación de reportes de calificaciones y de comportamiento, consultas de información estudiantil, matrículas, certificados, permisos, entre otros [2]. En este sentido es lógico pensar que la automatización en la gestión estudiantil es indispensable, puesto a que los avances tecnológicos traen consigo una variedad de herramientas de desarrollo capaces de cumplir esta labor [3].

El desarrollo de cualquier sistema de *software* que utilice adecuadas prácticas de programación, metodologías y organización tiende a ser menos propenso a perder su escalabilidad, ya que, se le puede dar un soporte que le permita seguir creciendo en una magnitud considerable, añadiendo que un sistema de gestión estudiantil debe ser capaz de llevar el control de todas las actividades que comprende este proceso, manteniendo su integridad, legibilidad, seguridad y permanencia [4]. Así como también su mantenibilidad, debido a que con el pasar de los años el ciclo de vida de un *software* puede verse afectado cuando no se aplica ningún tipo de actualización y/o control. Es decir, una vez lanzado el producto nunca más se vuelve a trabajar sobre este causando repercusiones como la generación de vulnerabilidades o interrupciones en las funcionalidades que forman parte del negocio [5].

Con este antecedente, el presente trabajo de integración curricular tiene como fin el desarrollo de un sistema para la gestión estudiantil del I.E.F “Miguel de Santiago” con ubicación en Quito; con el objetivo de generar una herramienta que optimice y mejore los procesos administrativos como; generación de reportes, matrículas y consultas de información, por medio de la creación de diversos *endpoints* que son consumidos por el sistema *web* que controla gráficamente las peticiones de los recursos que sean requeridos.

1.1 Objetivo general

Desarrollar el *backend* para un sistema de gestión estudiantil del I.E.F “Miguel de Santiago”.

1.2 Objetivos específicos

1. Definir los requerimientos necesarios para el desarrollo del sistema *web*.
2. Implementar la arquitectura de datos para el sistema de *web* en base a la recopilación de requerimientos.
3. Codificar *endpoints* para el sistema *web* en base a la recopilación de requerimientos.
4. Realizar pruebas de los *endpoints* validando su funcionalidad.
5. Desplegar los *endpoints.*

1.3 Alcance

Actualmente con el avance de las tecnologías de la información, los centros educativos requieren contar con herramientas que les permitan asegurar un progreso en la educación de calidad. En este sentido, los sistemas más exitosos son aquellos que tienen disponibilidad inmediata de la información que manejan, aprovechando al máximo temas de procesos administrativos y organizacionales, es decir, contar con un *software* de gestión estudiantil permite mantener un control adecuado de las actividades que se llevan a cabo, asegurando la integridad y permanencia de los datos [6].

Con el objetivo de poner en marcha una solución tecnológica que genere beneficios al I.E.F “Miguel de Santiago”, en el presente trabajo de titulación se ha desarrollado un sistema *web* que proporciona las funcionalidades para una gestión estudiantil adecuada. Contando con módulos de autenticación por roles que validan el tipo de información a la que se tiene acceso. Además, existe un manejo organizado de los datos del rendimiento estudiantil con la apertura a la posibilidad de realizar análisis sobre mejoras en la calidad de la educación que se imparte. Por otro lado, existe la generación de reportes administrativos tales como los de calificaciones y matrículas, llevando a que las actividades dispuestas por usuario sean las siguientes:

El perfil secretaria permite:

- Iniciar y cerrar sesión.
- Gestionar su perfil personal.
- Gestionar información del colegio.
- Gestionar estudiantes.
- Gestionar profesores.
- Activar o desactivar usuarios.
- Gestionar periodos académicos, cursos y especialidades.
- Gestionar asignaturas y paralelos.
- Gestionar calificaciones.
- Generar reportes.

El perfil profesor permite:

- Iniciar y cerrar sesión.
- Gestionar su perfil personal.
- Gestionar calificaciones.
- Generar reporte de calificaciones.

El perfil estudiante permite:

- Iniciar y cerrar sesión.
- Gestionar su perfil personal.
- Visualizar calificaciones.
- Generar reporte de calificaciones.

Para lograr estas funcionalidades se han generado varios *endpoints RESTful* que son:

- *Endpoints* para iniciar sesión y cerrar sesión.
- *Endpoints* para gestionar el perfil personal del usuario.
- *Endpoints* para gestionar calificaciones.
- *Endpoints* para activar o desactivar usuarios.

- *Endpoints* para gestionar periodos académicos, cursos y especialidades.
- *Endpoints* para gestionar asignaturas y paralelos.
- *Endpoints* para generar reportes.

1.4 Marco teórico

La ingeniería de *software* surge como una rama de la ingeniería que basa su disciplina en la construcción de programas informáticos en base a una serie de técnicas y/o herramientas, que pretenden cubrir necesidades de una empresa y/o sociedad, por medio de la abstracción de ideas que permite remover las limitaciones dadas por algún medio físico [7].

La interpretación de calidad dentro de un proyecto de *software* suele ser variable en dependencia del autor, siendo entonces definido por Pressman como “la concordancia con los requisitos funcionales y de rendimientos explícitamente establecidos, estándares de desarrollo explícitamente documentados y características implícitas que se espera de todo *software* desarrollado profesionalmente” [8]. Partiendo de esta definición, un *software* de calidad puede establecerse como aquel que satisface las necesidades de un cliente y cumple los requerimientos para solventar la problemática que se haya planteado.

Indistintamente del ámbito en el cual se desarrolle un *software*, la utilización de una metodología permite la obtención de un producto de calidad, debido a que esta brinda el apoyo necesario para que los tiempos de entrega se cumplan mediante un control basado en determinados parámetros [9]. Así, pueden caracterizarse distintos tipos de metodologías tales como: *XP, KANBAN, Scrum*, etc., y cada una presenta cualidades únicas que apoyan al proyecto.

Un sistema *web* es un tipo de *software* que se maneja mediante el alojamiento en el Internet o una red local, su interfaz es idéntica a la de alguna página web que se visualiza comúnmente, la ventaja de estos tipos de *software* es que su accesibilidad es a nivel global, quiere decir, que se puede ingresar desde cualquier sitio remoto siendo únicamente necesario la conexión a la red y un navegador *web* [10].

Es esencial tener un espacio donde los datos obtenidos del sistema se almacenen adecuadamente y, por lo tanto, la implementación de una base de datos de tipo relacional (SQL) se considera la mejor opción, ya que, al poder relacionar los datos ente ellos se genera la posibilidad de obtener información de consultas de manera

rápida, además de que las tablas permiten identificar datos requeridos al instante mediante su valor único [11].

Una plataforma *web* que permite alojar bases de datos tanto SQL como NoSQL es conocida como *AlwaysData*, la cual mediante su Sistema Gestor de Bases de Datos Relacional (SGBD) puede gestionar y almacenar los datos a un nivel administrativo, teniendo en cuenta también que contiene un alto nivel de seguridad ante posibles ataques [12].

El lenguaje de programación *PHP* suele ser el más conocido y utilizado para el desarrollo de *software web*, debido a que puede ser incrustado con HTML que es donde se aloja el contenido visual de una página *web*, generándolo también de manera dinámica para que este sea cambiante y no estático. Generalmente el funcionamiento de este lenguaje se basa en peticiones desde la página hacia el servidor donde se alojan los datos, devolviendo la información requerida [13].

Laravel, es un *Framework* de código abierto que se maneja mediante *PHP*, por lo que mediante su sintaxis expresiva y elegante evita que se desarrolle código mezclado, es decir, que no tenga ningún tipo de orden lógico. Este *Framework* brinda un espacio base preconfigurado con las herramientas necesarias para que los desarrolladores puedan enfocarse en aquello que buscan generar [14].

Heroku es una plataforma pagada que brinda la posibilidad de poder desplegar uno o varios proyectos para la *web*, ofreciendo el alojamiento por medio de protocolos seguros de tipo HTTPS con el objetivo de generar confianza al usuario que haga uso de él [15].

Cloudinary es una plataforma que provee un servicio que permite el almacenamiento y gestión de imágenes y videos, estos se cargan en la nube y pueden ser accedidos mediante los recursos con los que cuenta, una de las ventajas que contiene es que se puede hacer uso de esta en conjunto con sistemas *web* mediante una *URL* [16].

Una API, es un conjunto de funciones y procedimientos que contienen capas integradas en sus sistemas como la abstracción de datos y es capaz de realizar transferencias de estos mediante el protocolo HTTP con una comunicación entre cliente-servidor, esto quiere decir que se envían peticiones desde el sistema *web* para poder obtener algún recurso desde el servidor donde se encuentra alojado, existiendo así los accesos de tipo privados y públicos [17].

El medio por el cual se obtiene el acceso a los datos de un servidor es conocido como *Backend* y es ejecutado desde el lado del servidor, por lo tanto, está encargado de gestionar los procesos de comunicación e integración, ya sea en aspectos como la base de datos y *hosting* respectivamente dependiendo de la lógica de negocio que se esté empleando [18].

2 METODOLOGÍA

El estudio de casos es conocido como un método de investigación que permite el análisis y recopilación de la información, que tiene como objetivo llegar a comprender un determinado hecho partiendo desde un estudio en particular. Se maneja de manera cualitativa por lo cual, en dependencia del caso que se siga, se pueden ir generando secuencialmente nuevas teorías y/o hipótesis que deben estar sujetas a una minuciosa investigación previa para que posteriormente sean comprobadas [19].

El presente proyecto de titulación se basa en un estudio de caso debido a que con una adecuada investigación acerca de los beneficios que un sistema *web* actualizado traer consigo, se puede mejorar de manera notable los procesos administrativos que se llevan a cabo diariamente en el I.E.F “Miguel de Santiago”. Dichos beneficios caen en cuenta para actividades tales como la generación de reportes de calificaciones, organización estudiantil, consultas de información personal, entre otros. El objetivo es encontrar la forma de optimizar cada proceso de modo ágil y eficiente.

1.1 Metodología de Desarrollo

Un producto de *software* puede derivarse de varias metodologías que dependen de la organización de trabajo que se vaya a implementar, cada una contiene estándares que ayudan a guiar el rumbo para emplear el desarrollo de un *software*, dividiendo todas las tareas y/o responsabilidades para obtener resultados favorables y productivos [20].

Para poder facilitar en gran medida el flujo de trabajo dentro del desarrollo de un *software* existen las metodologías ágiles, estas permiten en base a condiciones específicas organizar el ambiente con el fin de asegurar resultados proactivos, con la ventaja de que se encuentra presente la participación continua del cliente, en donde al realizar entregas progresivas se obtienen respuestas por su parte abriendo paso a mejoras en aspectos que no se observan en el instante del desarrollo [21]. *Scrum* es una metodología ágil de desarrollo de *software* que cumple con todas las características antes mencionadas, y de ahí el hecho que sea utilizada como guía para el desarrollo del presente trabajo de titulación.

Roles

Los roles permiten asegurar que la comunicación con el cliente se desenvuelva de manera adecuada y el proyecto de *software* marche correctamente [22]. Los miembros del equipo individualmente se encargan de manejar una sección específica del proyecto, manteniendo la coherencia y relación entre cada componente finalizado [23].

Product Owner

Se define como aquella persona que mantiene la mayor autoridad dentro del equipo, se encarga principalmente de la supervisión y de brindar la retroalimentación de datos sobre el desempeño del proyecto [23]. La **TABLA I** muestra quién es el que maneja este rol.

Scrum Master

Se asigna este rol únicamente a la persona quien dentro del equipo de desarrollo pueda manejar adecuadamente al resto con liderazgo, organización y una buena guía, generando un ambiente de confianza [23]. La persona que se encuentra a cargo de desempeñar este rol se puede visualizar en la **TABLA I.**

Development Team

Es el equipo conformado por uno o varios integrantes en el desarrollo del proyecto y son los responsables de finalizarlo, se caracterizan por ser multifuncionales y autoorganizados; son quienes se encargan de manejar las tareas designadas por el *Product Owner* [24]. La **TABLA I** muestra quién es el que maneja este rol.

TABLA I: Designación de roles para el proyecto.

Rol	Integrantes
Product Owner	Lcda. Jacqueline Carrera.
Scrum Master	Ing. Ivonne Maldonado, MSc.
Development Team	Raúl Tenorio.

Artefactos

Los artefactos dentro de *Scrum* son implementados para asegurar el sustento y fiabilidad dentro del equipo de desarrollo, manejando un adecuado control de la comunicación entre los mismos [24].

Recopilación de requerimientos

Es un pilar fundamental en el desarrollo de un *software* de calidad; cada requerimiento se toma como una funcionalidad que se requiere implementar, por lo que tiene una serie de características específicas para obtener la descripción detallada de la necesidad que se está planteando por parte del cliente [25]. El **ANEXO II** del presente documento contiene los aspectos necesarios para analizar la problemática abordada.

Historias de Usuario

Cuando los requerimientos ya se han recopilado de manera correcta, por cada uno de ellos se procede a fijar las funcionalidades respectivas, esto se conoce como Historias de Usuario las cuales, a manera de una descripción corta y simple da a entender aquella capacidad que el cliente desea que se implemente, su objetivo es establecer el orden de prioridades organizadamente [26]. De esta manera, el equipo está al tanto de cada parte del desarrollo del proyecto y el papel que cumplen en él. La **TABLA II** muestra un ejemplo de las historias de usuario, el **ANEXO II** del presente documento detalla las historias de usuario restantes.

TABLA II Ejemplo de Historia de usuario - HU005

HISTORIA DE USUARIO	
Identificador (ID): HU005	**Usuario**: Profesor.
Nombre Historia: Generar varios *endpoints* para gestionar calificaciones.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil profesor necesita varios *endpoints* para gestionar calificaciones de los estudiantes que pertenecen a su clase, cada estudiante se identifica mediante los datos de: ● Nombre y Apellido. ● Cédula. Además, por cada estudiante las notas que se registran son: ● Primer quimestre. o Primer parcial. o Segundo parcial. o Tercer parcial. ● Segundo quimestre. o Primer parcial. o Segundo parcial. o Tercer parcial. ● Nota del examen supletorio. ● Nota del examen remedial. ● Nota del examen de gracia. ● Promedio final por asignatura.	

● Promedio total.
Observación: Ninguna.

Product Backlog

Como resultado del análisis en la Recopilación de Requerimientos e implementación de las Historias de Usuario, surge el *Product Backlog* el cual maneja las tareas a realizar por cada requisito en forma de descripción sencilla a manera de la perspectiva del usuario, teniendo como objetivo un desarrollo ágil del sistema *web* [27]. En el **ANEXO II** del presente documento se detalla este elemento.

Sprint Backlog

En referencia al *Product Backlog*, cada uno de sus elementos se lista en iteraciones que deben completarse en un tiempo determinado para posteriormente generar entregables al cliente [28]. En el **ANEXO II** del presente documento se detalla este elemento.

1.2 Diseño de arquitectura

El patrón arquitectónico tiene el objetivo de brindar una solución óptima del problema que se ha planteado. A continuación, se detalla el modelo establecido para el desarrollo del sistema *web*.

Patrón arquitectónico Modelo Vista Controlador (MVC)

Es un modelo guía que tiene la finalidad de organizar y estructurar los componentes que comprende un *software*, sus tres capas (modelo, vista y controlador) se manejan a medida que se acoplan al mínimo entre sí [29].

- **Modelo**: se definen las entidades que almacenan los datos del sistema, por lo tanto, es aquella capa que maneja toda la lógica del negocio.
- **Vista**: es la encargada de generar las vistas del proyecto, siendo así la responsable de generar las interfaces del sistema *web*.
- **Controlador:** intermediario entre el usuario y el sistema, es decir, puede capturar acciones de la vista interpretando y realizando alguna función sobre aquello.

A continuación, la **Fig. 1** representa el diseño del patrón de arquitectura implementado junto con las herramientas para su despliegue a producción.

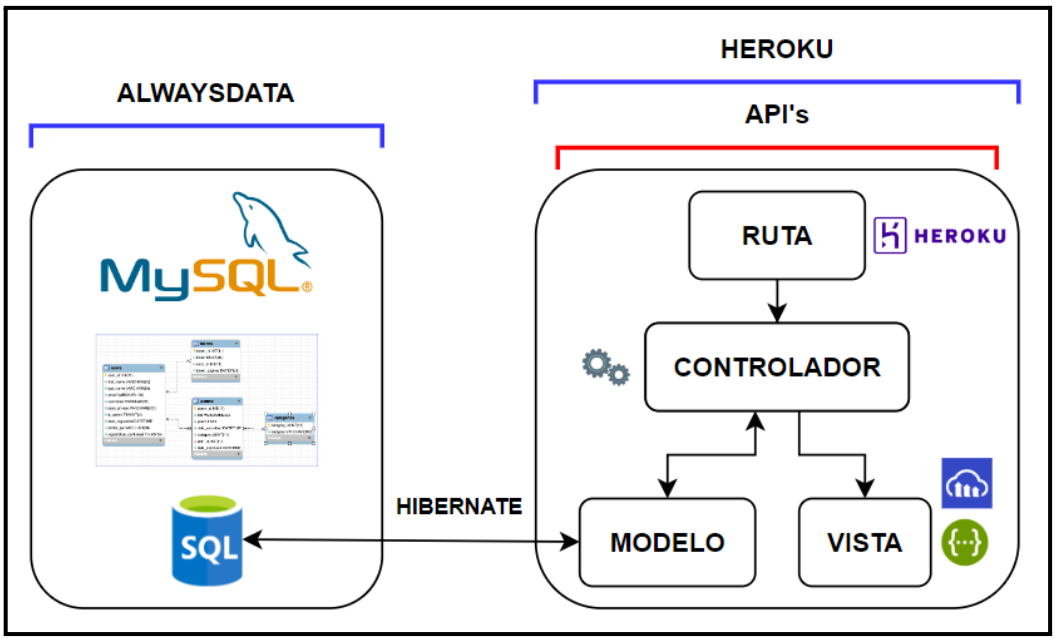

Fig. 1: Patrón Arquitectónico – Sistema *web*.

1.3 Herramientas de desarrollo

Las herramientas seleccionadas para el desarrollo del sistema *web* y creación de los *endpoints* se basan en los requerimientos obtenidos y los conocimientos del equipo de desarrollo, estas herramientas son empleadas para optimizar cada uno de los procesos de desarrollo, asegurando la productividad del proyecto [30]. La **TABLA III** presenta las herramientas seleccionadas para el desarrollo en conjunto con una breve justificación.

TABLA III: Herramientas para el desarrollo del sistema *web* y *endpoints*.

HERRAMIENTA	JUSTIFICACIÓN
Laravel	Es el *framework* manejado por el lenguaje de programación *PHP* en el cual se estructura el modelo MVC que contiene el presente proyecto [31].
Alwaysdata	Es el servicio que provee almacenamiento en la nube mediante una base de datos para realizar la gestión de la información desde el sistema *web* [12].
MySQL	Es un sistema de gestión para una base de datos relacional que se complementa adecuadamente con *Alwaysdata* y el framework de *Laravel* [32].

Heroku	Es el servicio que permite almacenar al sistema *web* en la nube para que se pueda acceder desde cualquier sitio y se evite el inconveniente de tenerlo localmente [15].
Cloudinary	Es el servicio de almacenamiento de archivos en la nube donde se encuentran todas las imágenes del sistema *web*, además de que contiene las librerías para que *Laravel* se conecte directamente y se puedan gestionar sus recursos [16].

Librerías

En la **TABLA IV** se describe cada una de las librerías empleadas para el desarrollo y codificación del sistema de *web*, así como la generación de los *endpoints*.

TABLA IV: Librerías para el desarrollo del sistema *web* y *endpoints*.

LIBRERÍA	DESCRIPCIÓN
Tinker	Se encarga de generar una consola de comandos para el lenguaje de programación PHP [33].
Model Generator	Paquete que permite dar la estructura básica a un objeto PHP junto con *seeders* y *factories* [33].
Migration Generator	Es un paquete que permite generar migraciones desde una base de datos con todos los índices, llaves primarias y/o foráneas que posea [33].
Laravel Sanctum	Proporciona un sistema de autenticación por *tokens* para que en la *API* se puedan establecer acciones en dependencia de cómo se defina [34].
Cloudinary Laravel	Funciona como un paquete para desarrolladores que permite gestionar archivos en la nube [35].
Laravel Send Email	Es un paquete de tipo SMTP que habilita el envío de correos electrónicos desde la aplicación *web* [36].

3 RESULTADOS

En esta sección se muestra los resultados de las tareas para la implementación de los *endpoints* del sistema *web*, las pruebas y el procedimiento realizado para el despliegue a producción. Hay que recalcar que los resultados se presentan por *Sprints* mismo que se detallan en el **ANEXO II** de este documento.

2.1 Sprint 0. Configuración del Ambiente de Desarrollo

Para el presente *Sprint* se comprende las siguientes tareas:

- Recopilación de requerimientos.
- Diseño e implementación de la Base de Datos.
- Estructura del proyecto.
- Roles de usuario para el sistema *web* y *endpoints*.

Recopilación de requerimientos

Iniciar y cerrar sesión

Se han implementado métodos para que los diferentes perfiles de usuarios (secretario/a, profesor/a y estudiante) puedan iniciar y cerrar sesión. Esto con credenciales de acceso para los tres tipos de usuarios (número de identificación y contraseña), cabe mencionar que las credenciales iniciales son creadas por el equipo de desarrollo. Además, mediante un módulo se generan los *endpoints* respectivos que permiten realizar esta funcionalidad para que posteriormente puedan ser consumidas por el sistema *web*.

Gestionar perfil personal

El sistema *web* a través de métodos tiene a su disponibilidad un módulo que permite a los usuarios gestionar su propio perfil personal, en dependencia del rol que posean tienen permisos para crear, visualizar, editar y eliminar la información respectiva, además de que tienen la posibilidad de establecerse una foto de perfil y realizar el cambio de contraseña para la cuenta.

Gestionar usuarios

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar a los usuarios con roles

profesor/a y estudiante. Cabe destacar que con decir “eliminar” se hace referencia a que el usuario queda inhabilitado, sin embargo, no se borra de la base de datos.

Gestionar periodos académicos

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar (inhabilitar) a los periodos académicos del colegio.

Gestionar cursos

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar (inhabilitar) a los cursos del colegio.

Gestionar paralelos

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar (inhabilitar) a los paralelos del colegio.

Gestionar especialidades

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar (inhabilitar) a las especialidades del colegio.

Gestionar asignaturas

El sistema *web* a través de métodos tiene a su disposición un módulo que permite al usuario con rol secretario/a crear, visualizar, editar y eliminar (inhabilitar) a las asignaturas del colegio.

Gestionar calificaciones

El sistema *web* a través de métodos tiene a su disponibilidad un módulo que permite a los usuarios gestionar las calificaciones obtenidas a lo largo de un periodo académico, en dependencia del rol que posean tienen permisos para crear, visualizar, editar y eliminar (inhabilitar) las calificaciones.

Generar reportes

El sistema *web* a través de métodos tiene a su disponibilidad un módulo que permite a los usuarios generar reportes de calificaciones, los cuales se crean por primer

quimestre, segundo quimestre y un esquema final. Además, los reportes pueden generarse por curso y paralelo o individualmente por estudiante.

La **Fig. 2** muestra los usuarios y acciones que pueden realizar dentro del sistema *web*.

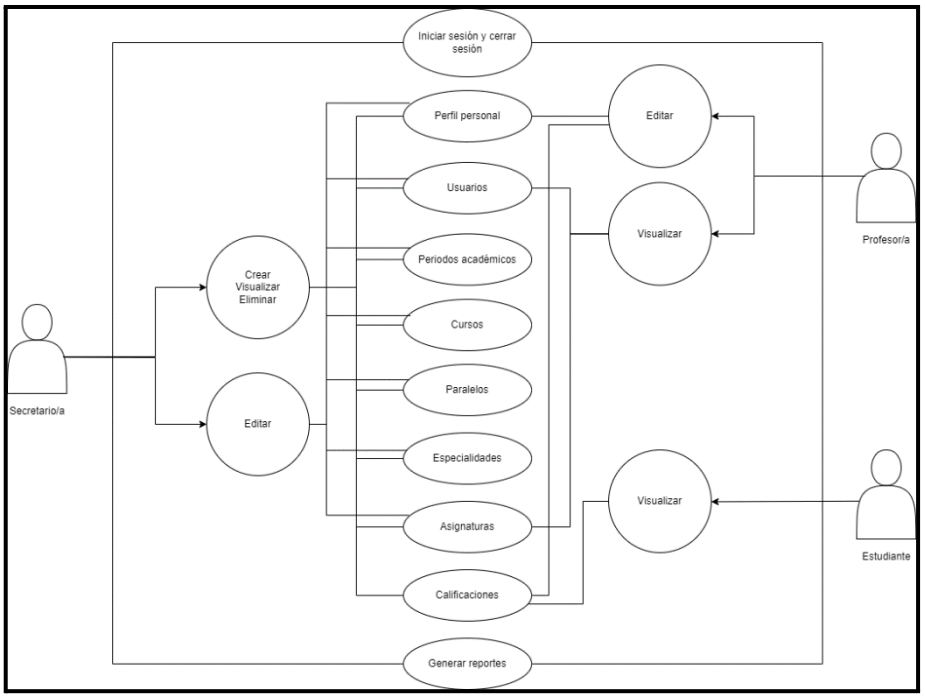

Fig. 2: Usuarios y funcionalidades dentro del sistema *web*

Generar *endpoints* para la visualización de la información del colegio

Se han desarrollado varios métodos y rutas las cuales permiten obtener la información necesaria para visualizar la información básica del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar el perfil personal

Se han desarrollado varios métodos y rutas las cuales permiten gestionar toda la información del perfil que posea un usuario, estos *endpoints* se encuentran disponibles para todos los usuarios, sin embargo, los datos pueden ser modificados en dependencia de los roles secretario/a, profesor/a y estudiante.

Generar *endpoints* para gestionar usuarios

Se han desarrollado varios métodos y rutas las cuales permiten gestionar las cuentas de los usuarios, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar periodos académicos

Se han desarrollado varios métodos y rutas las cuales permiten gestionar los periodos académicos del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar cursos

Se han desarrollado varios métodos y rutas las cuales permiten gestionar los cursos del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar paralelos

Se han desarrollado varios métodos y rutas las cuales permiten gestionar los paralelos del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar especialidades

Se han desarrollado varios métodos y rutas las cuales permiten gestionar las especialidades del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar asignaturas

Se han desarrollado varios métodos y rutas las cuales permiten gestionar las asignaturas del colegio, estos *endpoints* se encuentran disponibles únicamente para el usuario con rol secretario/a.

Generar *endpoints* para gestionar calificaciones

Se han desarrollado varios métodos y rutas las cuales permiten gestionar las calificaciones obtenidas por los estudiantes a lo largo del periodo académico actual del colegio, estos *endpoints* se encuentran disponibles para todos los usuarios, sin embargo, existen restricciones en dependencia de los roles de usuarios secretario/a, profesor/a y estudiante.

Generar *endpoints* para visualizar reportes

Se han desarrollado varios métodos y rutas privadas las cuales permiten generar reportes de las calificaciones, es decir, recibe datos individuales del estudiante o de un curso y paralelo específico para generarse por quimestre o un reporte final con toda la información del registro de calificaciones, por lo tanto, genera un archivo en

formato .pdf con la información requerida, estos *endpoints* se encuentran disponibles para todos los usuarios, pero por cada rol se utiliza un esquema distinto.

La **Fig. 3** muestra los usuarios y métodos que se realiza en los diferentes *endpoints*.

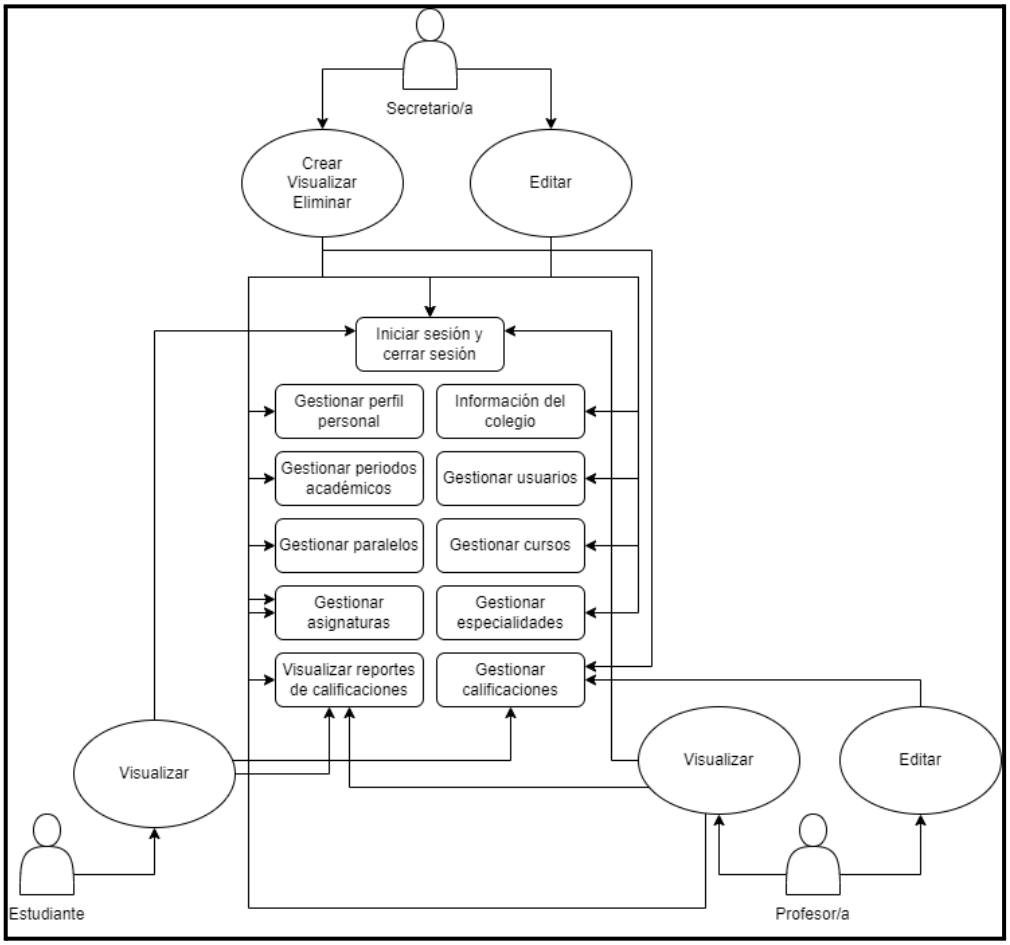

Fig. 3: Usuarios y funcionalidades para los *endpoints*

Diseño e implementación de la Base de Datos

Para el almacenamiento de datos se utiliza MySQL, un sistema gestor de bases de datos relacionales SQL que permite el almacenamiento de datos en forma de tablas que están relacionadas.

La **Fig. 4** muestra el diseño de base de datos, es decir las tablas y sus relaciones implementadas para la transferencia de datos.

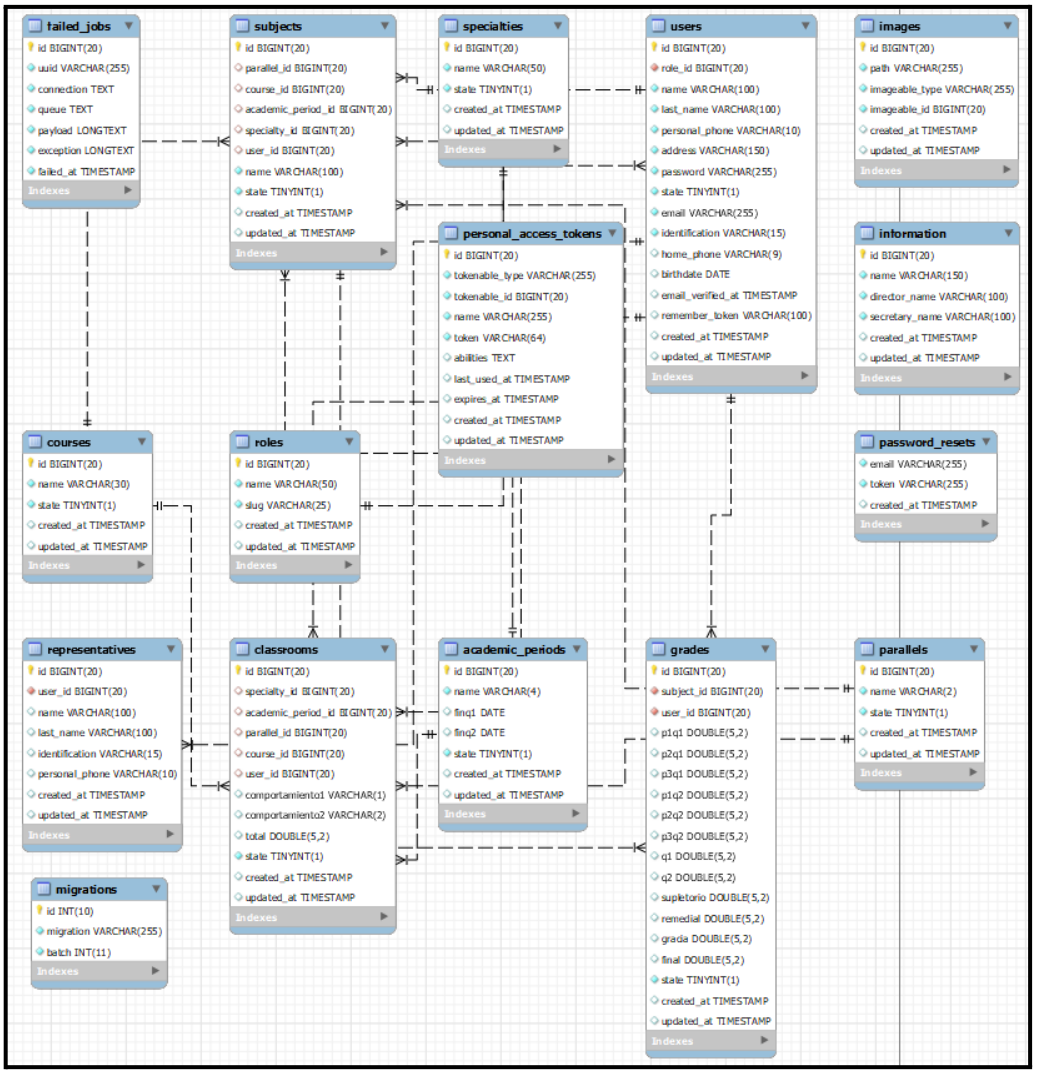

Fig. 4: Base de datos relacional

Estructura del proyecto

Para la generación de *endpoints* se ha empleado el *framework* de *Laravel* como entorno de desarrollo, en el cual mediante la ejecución del comando "*laravel new sismds_v1.0*" se crea el proyecto base con la estructura definida por el mismo *framework*, es así como este puede ser visualizado en la **Fig. 5**.

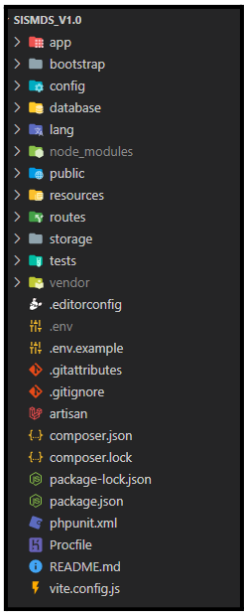

Fig. 5: Estructura del proyecto – *endpoints*.

Roles de usuario para el sistema web y *endpoints*

A continuación, para los *endpoints* en la **Fig. 6** se presentan todos los módulos a los cuales tienen acceso cada uno de los usuarios: secretario/a, profesor/a y estudiante, iniciando sesión en caso de ser necesario.

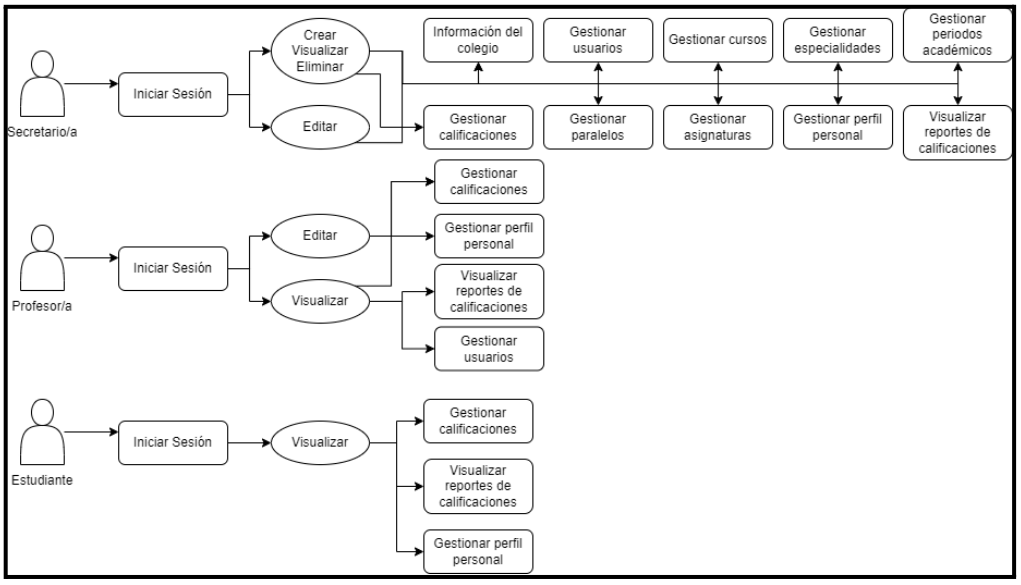

Fig. 6: Usuarios y roles para uso de *endpoints*.

2.2 Sprint 1. Implementación de *endpoints* para el módulo "secretaria".

Este *Sprint* comprende las siguientes tareas:

- Generar *endpoints* para inicio y cierre de sesión.
- Generar *endpoints* para gestionar el perfil personal de la secretaria.
- Generar *endpoints* para gestionar calificaciones.
- Generar *endpoint* para visualizar el reporte de calificaciones.
- Generar *endpoints* para gestionar la información del colegio.
- Generar *endpoints* para gestionar estudiantes.
- Generar *endpoints* para gestionar profesores.
- Generar *endpoints* para activar o desactivar usuarios.
- Generar *endpoints* para gestionar periodos académicos.
- Generar *endpoints* para gestionar cursos.
- Generar *endpoints* para gestionar paralelos.
- Generar *endpoints* para gestionar especialidades.
- Generar *endpoints* para gestionar asignaturas.

Generar *endpoints* para inicio y cierre de sesión

La funcionalidad de inicio y cierre de sesión es posible gracias a métodos y rutas que son consumidos en *endpoints* sin restricción alguna.

La implementación de un método *POST* permite el envío de credenciales e inicio de sesión, por medio de un *token* de acceso para los *endpoints* privados como se presenta en la **Fig. 7**.

El consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

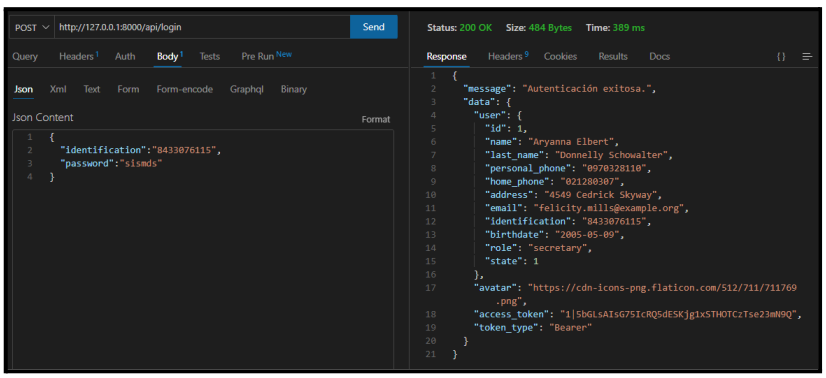

Fig. 7: Método *POST* para inicio de sesión.

Generar *endpoints* para gestionar el perfil personal de la secretaria

Para la gestión del perfil personal de la secretaria se han implementado métodos y rutas que permiten al usuario con rol secretaria consumir estos *endpoints* sin restricciones. Estos son de tipo *POST* y *GET* e incluyen la funcionalidad de poder mostrar y editar la información, así como deshabilitar la cuenta de un usuario con el mismo rol, tal y como se presenta en las figuras **Fig. 8**, **Fig. 9** y **Fig. 10**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

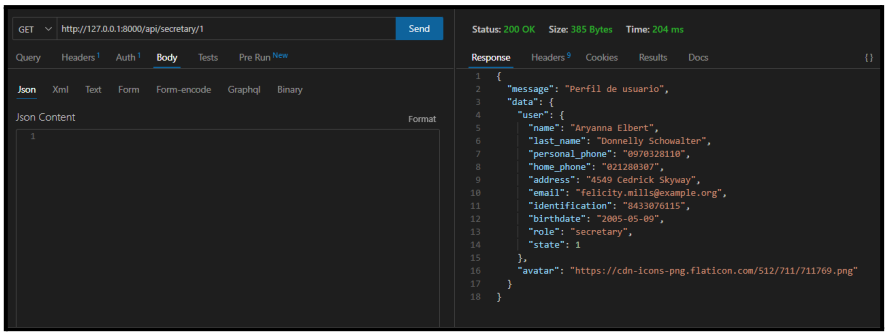

Fig. 8: Método *GET* mostrar el perfil de la secretaria.

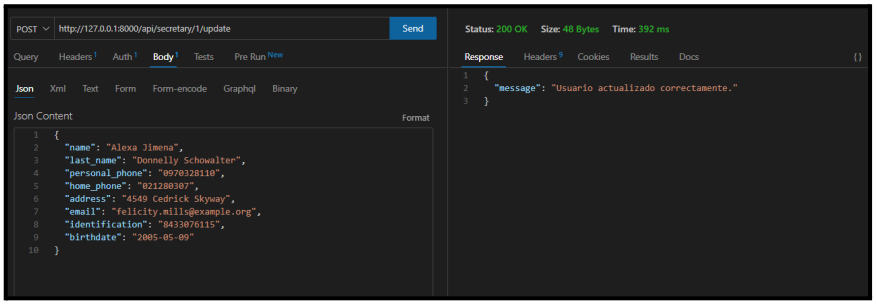

Fig. 9: Método *POST* actualizar el perfil de la secretaria.

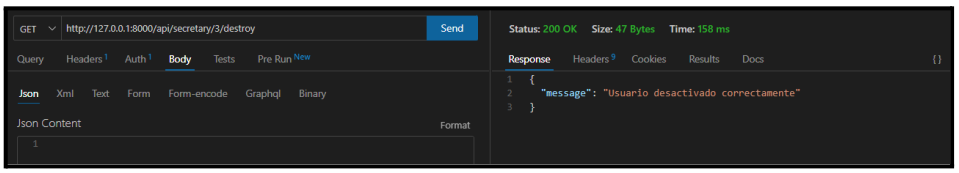

Fig. 10: Método GET deshabilitar/habilitar un usuario con rol de secretaria.

Generar *endpoints* para gestionar calificaciones

Para la gestión de calificaciones por parte de un usuario que contenga el rol de secretaria, se han implementado tres rutas. La primera, es de tipo *GET* y es aquella que trae las calificaciones de un estudiante en específico recibiendo como parámetros el identificador único del estudiante y el periodo académico, como se muestra en la **Fig. 11**. La segunda, es de tipo *POST* y tiene la función de modificar el comportamiento que ha tenido el estudiante en el periodo académico vigente,

internamente se calcula el promedio final, tal como se muestra en la **Fig. 12**. La tercera, también es de tipo *POST* y tiene como finalidad terminar el periodo académico vigente, esto se muestra en la **Fig. 13**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

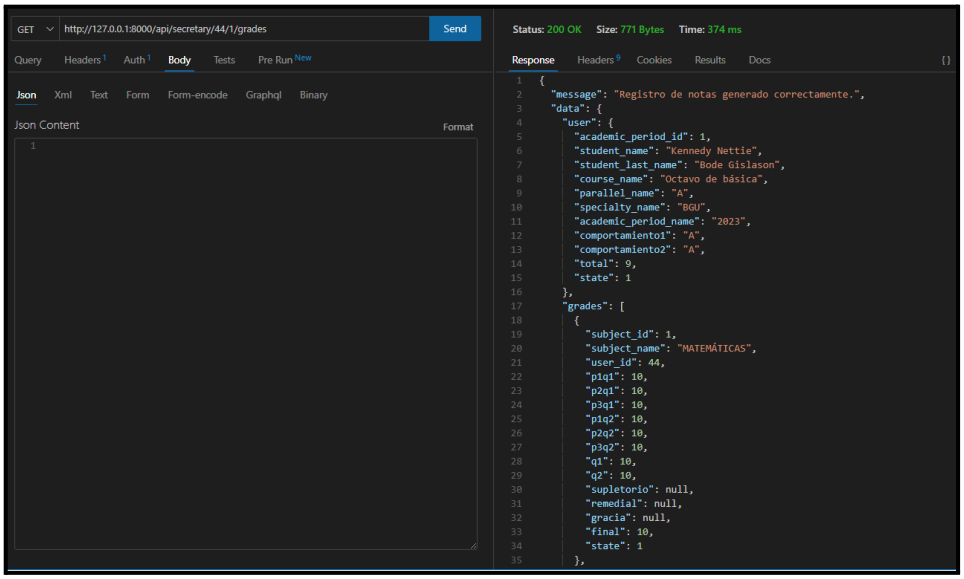

Fig. 11: Método *GET* obtener las calificaciones de un estudiante.

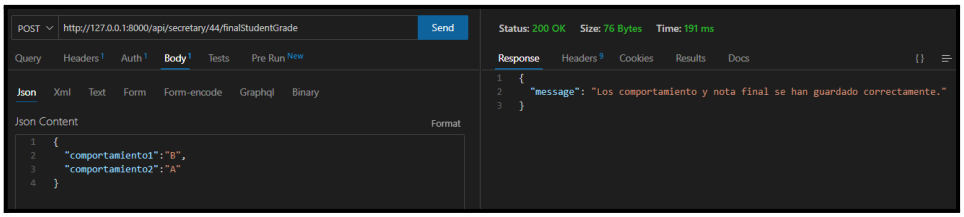

Fig. 12: Método *POST* para ingresar el comportamiento y nota final de estudiante.

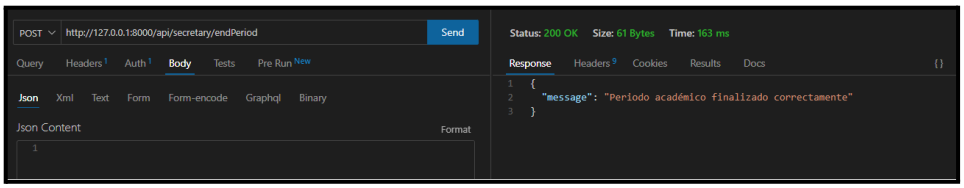

Fig. 13: Método *POST* para finalizar el periodo académico vigente.

Generar *endpoint* para visualizar el reporte de calificaciones

Para la obtención de reportes se ha creado un método tipo *GET* con una ruta que permite a un usuario con perfil secretario/a generarlo por curso, paralelo, especialidad y periodo académico, se extrae la información en un archivo JSON que es consumido por el sistema *web* con el fin de generar la vista de un reporte que pueda ser visualizado e impreso, como se presenta en la **Fig. 14**. En este aspecto,

el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

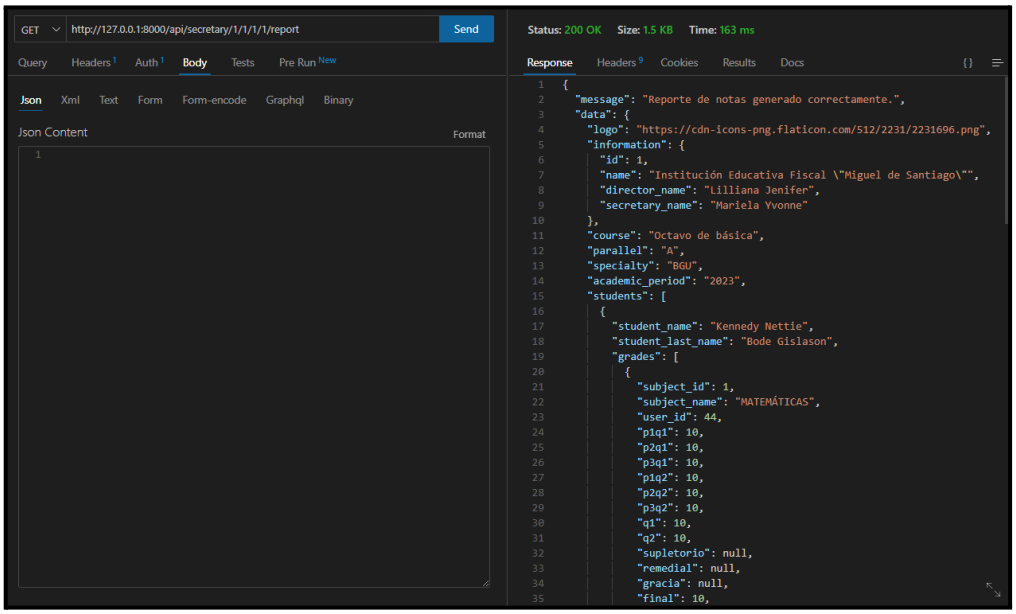

Fig. 14: Método *GET* para obtener el reporte con rol de secretario/a.

Generar *endpoints* para gestionar la información del colegio

Para la gestión de la información del colegio se han implementado métodos y rutas que solo pueden ser utilizadas por un usuario con el rol de secretario/a.

Se han implementado métodos tipo *POST* para realizar la actualización de la información y un método tipo *GET* para visualizar la misma, como se puede observar en las figuras **Fig. 15**, **Fig. 16** y **Fig. 17**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

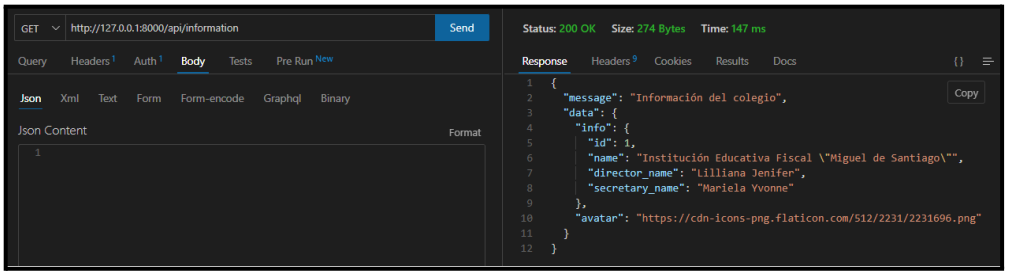

Fig. 15: Método *GET* para obtener la información del colegio.

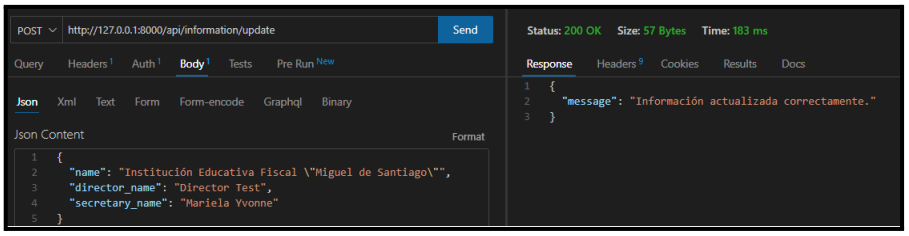

Fig. 16: Método *POST* para actualizar la información del colegio.

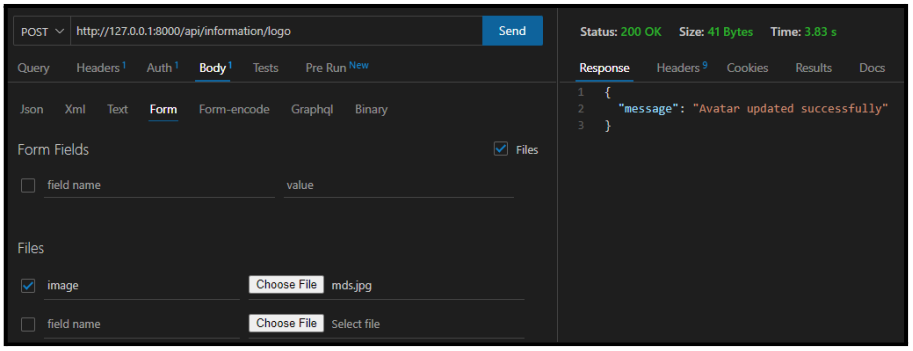

Fig. 17: Método *POST* actualizar el logo del colegio.

Generar *endpoints* para gestionar estudiantes

Para la gestión de estudiantes se han implementado métodos y rutas a los cuales el acceso de los *endpoints* solo se le permite a un usuario que posea el rol de secretario/a.

Se han implementado métodos tipo *POST* y *GET* que cumplen la funcionalidad de crear, visualizar, editar y buscar un estudiante en específico, como se puede visualizar en las figuras **Fig. 18**, **Fig. 19**, **Fig. 20** y **Fig. 21**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

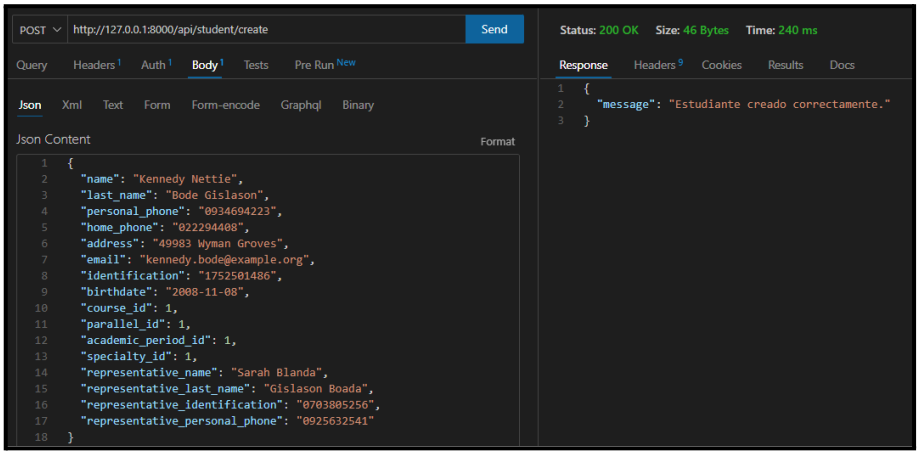

Fig. 18: Método *POST* para crear un estudiante.

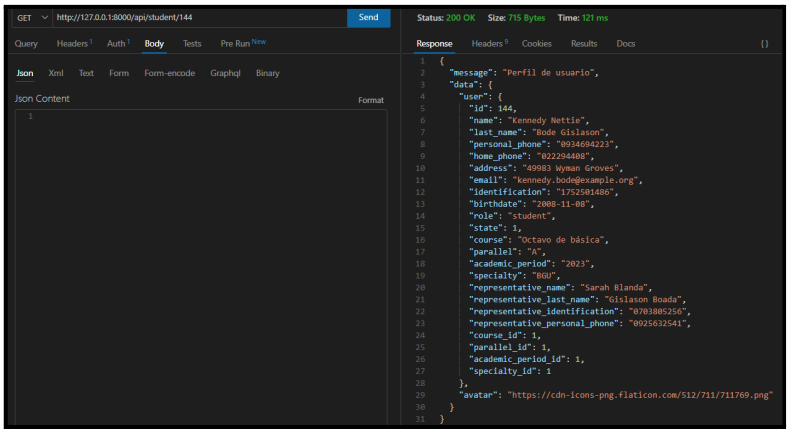

Fig. 19: Método *GET* para obtener los datos de un estudiante.

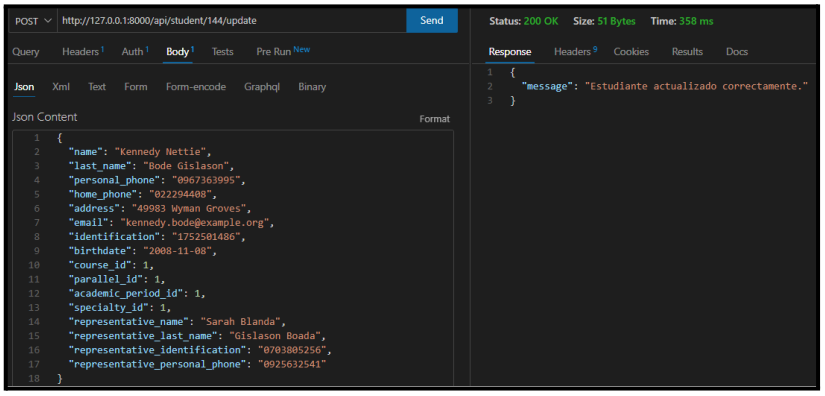

Fig. 20: Método *POST* para actualizar un estudiante.

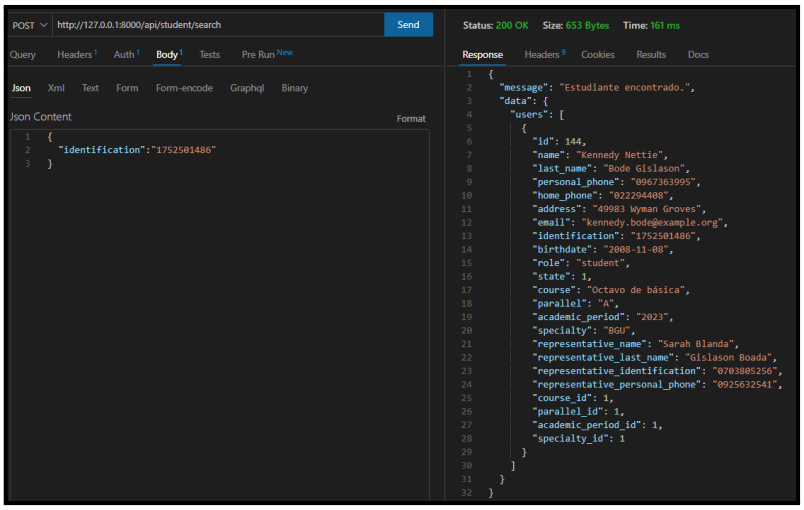

Fig. 21: Método *POST* para buscar estudiante.

Generar *endpoints* para gestionar profesores

Para la gestión de profesores se han implementado métodos y rutas a los cuales el acceso de los *endpoints* solo se le permite a un usuario que posea el rol de secretario/a.

Se han implementado métodos tipo *POST* y *GET* que cumplen la funcionalidad de crear, visualizar, editar y buscar un profesor en específico como se puede visualizar en las figuras **Fig. 22**, **Fig. 23, Fig. 24** y **Fig. 25**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

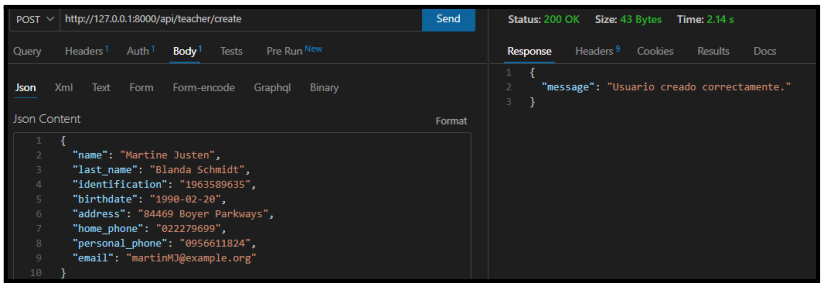

Fig. 22: Método *POST* para crear profesor.

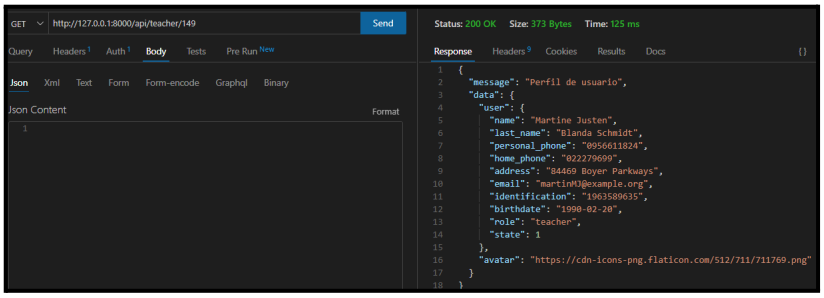

Fig. 23: Método *GET* para obtener los datos de un profesor.

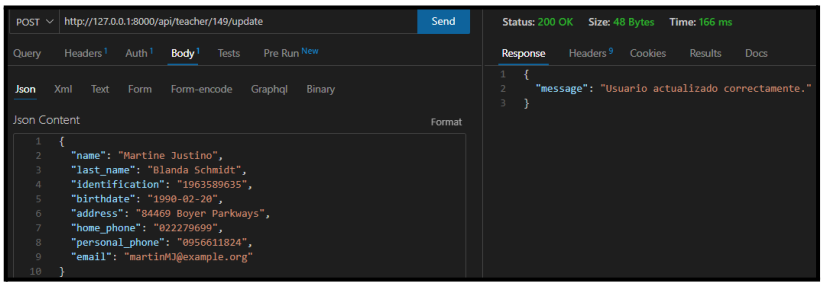

Fig. 24: Método *POST* para actualizar profesor.

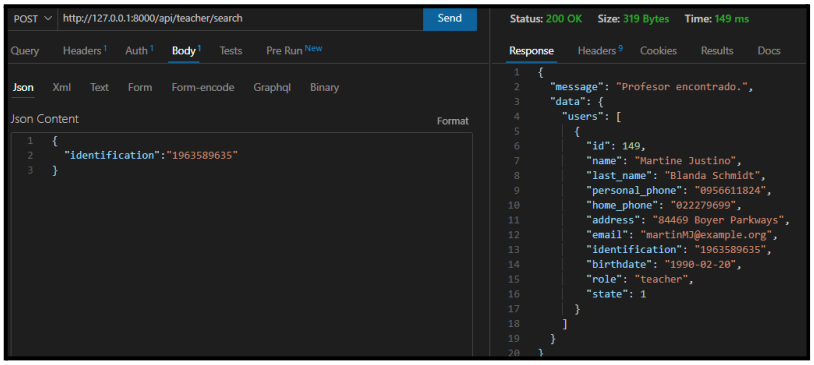

Fig. 25: Método *POST* para buscar profesor.

Generar *endpoints* para activar o desactivar usuarios

Para la funcionalidad de activar o desactivar usuarios se han implementado métodos y rutas a los cuales solo se puede acceder a los *endpoints* mediante un usuario con el rol de secretario/a.

Se han implementado varios métodos tipo *GET*, los cuales permiten deshabilitar/habilitar un usuario, como se puede observar en las figuras **Fig. 26**, **Fig. 27** y **Fig. 28**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

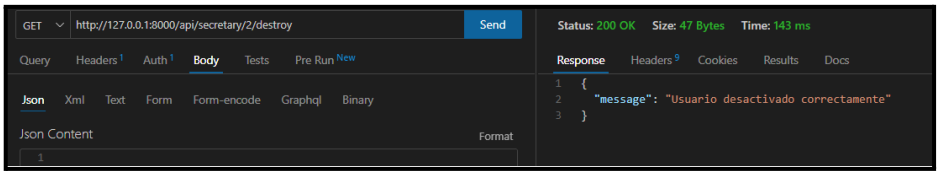

Fig. 26: Método *GET* para deshabilitar secretaria.

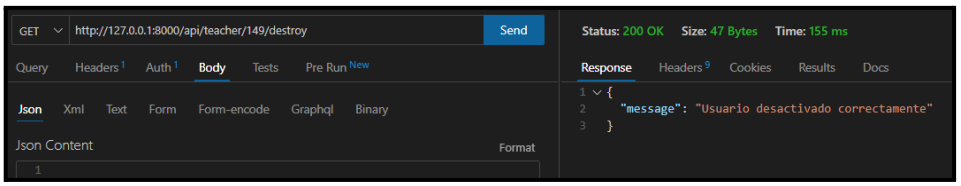

Fig. 27: Método *GET* para deshabilitar profesor.

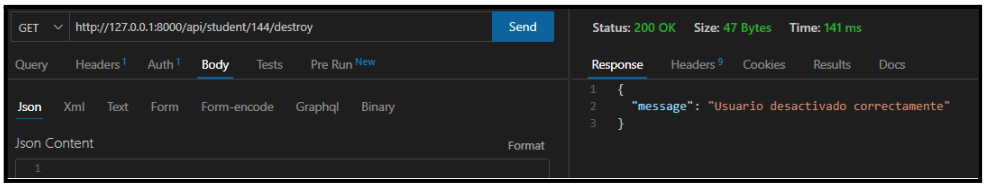

Fig. 28: Método *GET* para deshabilitar un estudiante.

Generar *endpoints* para gestionar periodos académicos

Para la gestión de los periodos académicos se ha creado varios métodos y rutas los cuales permiten al usuario con rol de secretario/a gestionar toda la información relacionada con los periodos académicos.

Se ha implementado un método tipo *GET* con una ruta para la obtención de la información de un periodo académico y métodos tipo *POST* con rutas para la creación, actualización e inhabilitación de la información respectivamente, como se presenta en las figuras **Fig. 29**, **Fig. 30**, **Fig. 31** y **Fig. 32**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

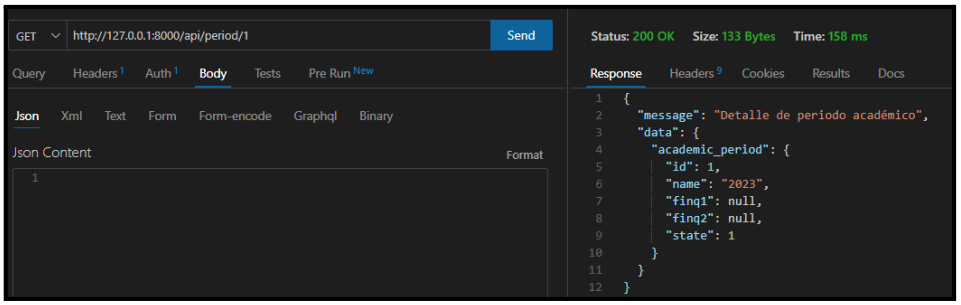

Fig. 29: Método *GET* para visualizar un periodo académico.

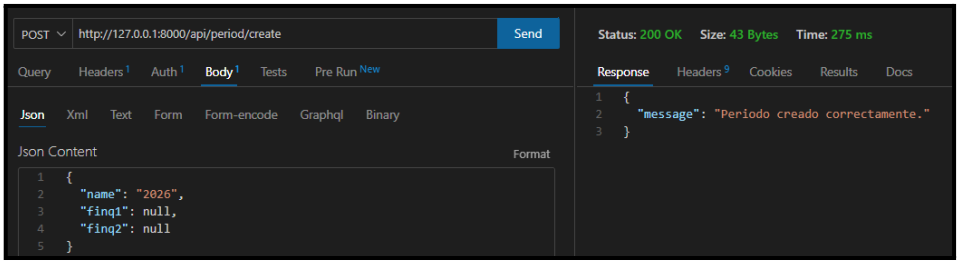

Fig. 30: Método *POST* para crear un periodo académico.

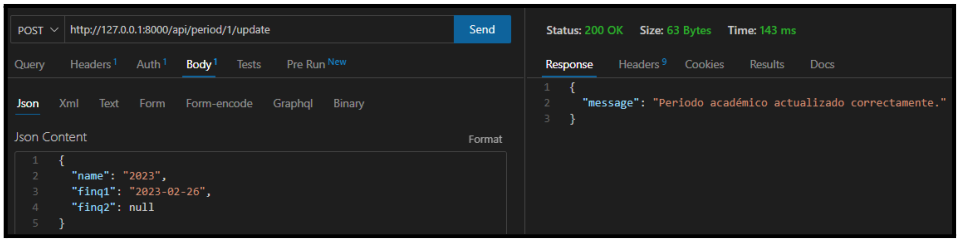

Fig. 31: Método *POST* para actualizar un periodo académico.

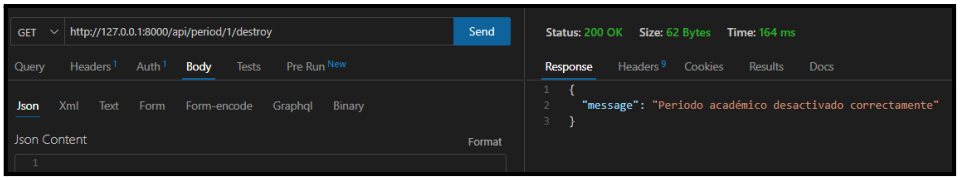

Fig. 32: Método *GET* para deshabilitar un periodo académico.

Generar *endpoints* para gestionar cursos

Para la gestión de los cursos se ha creado varios métodos y rutas los cuales permiten al usuario con rol de secretario/a gestionar toda la información relacionada con los cursos.

Se ha implementado un método tipo *GET* con una ruta para la obtención de la información de un curso y métodos tipo *POST* con rutas para la creación, actualización e inhabilitación de la información respectivamente, como se presenta en las figuras **Fig. 33**, **Fig. 34**, **Fig. 35** y **Fig. 36**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

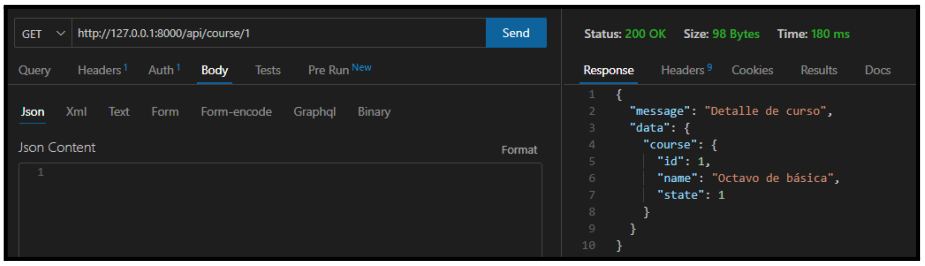

Fig. 33: Método *GET* para visualizar un curso.

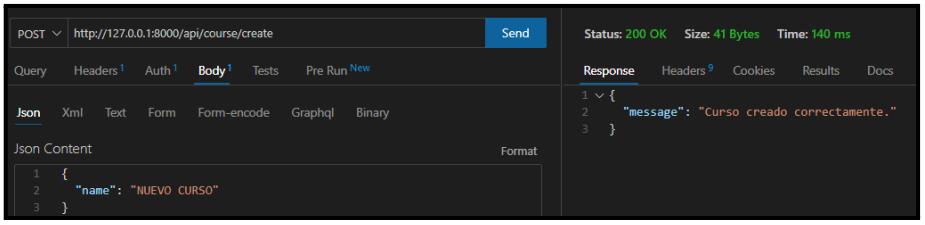

Fig. 34: Método *POST* para crear un curso.

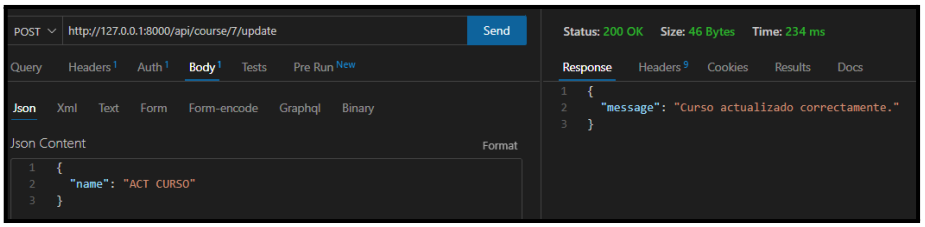

Fig. 35: Método *POST* para actualizar un curso.

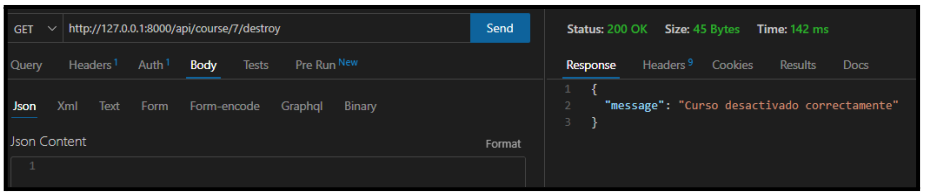

Fig. 36: Método *GET* para deshabilitar un curso.

Generar *endpoints* para gestionar paralelos

Para la gestión de los paralelos se ha creado varios métodos y rutas los cuales permiten al usuario con rol de secretario/a gestionar toda la información relacionada con los paralelos.

Se ha implementado un método tipo *GET* con una ruta para la obtención de la información de un paralelo y métodos de tipo *POST* con rutas para la creación, actualización e inhabilitación de la información respectivamente, como se presenta en las figuras **Fig. 37**, **Fig. 38**, **Fig. 39** y **Fig. 40**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

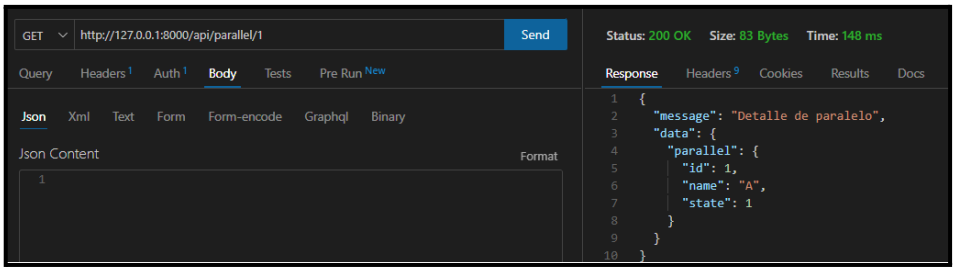

Fig. 37: Método *GET* para visualizar un paralelo.

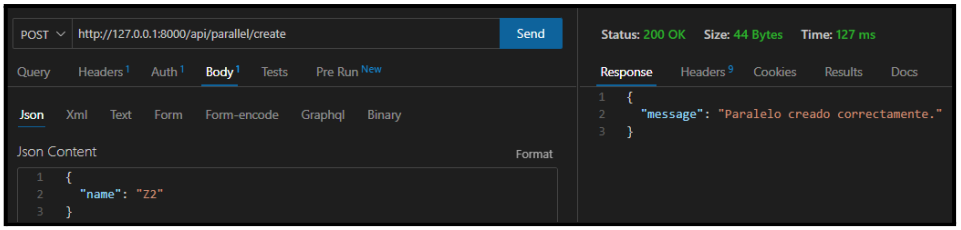

Fig. 38: Método *POST* para crear un paralelo.

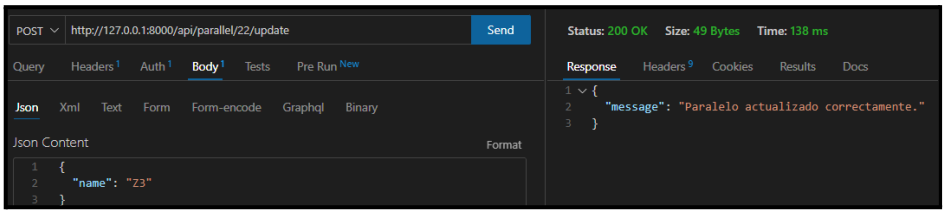

Fig. 39: Método *POST* para actualizar un paralelo.

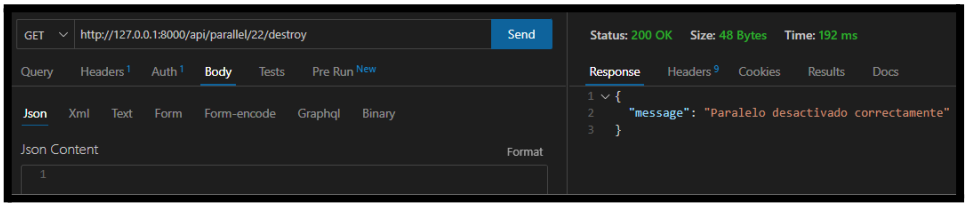

Fig. 40: Método *GET* para deshabilitar un paralelo.

Generar *endpoints* para gestionar especialidades

Para la gestión de las especialidades se ha creado varios métodos y rutas los cuales permiten al usuario con rol de secretario/a gestionar toda la información relacionada con las especialidades.

Se ha implementado un método tipo *GET* con una ruta para la obtención de la información de una especialidad y métodos de tipo *POST* con rutas para la creación, actualización e inhabilitación de la información respectivamente, como se presenta en las figuras **Fig. 41**, **Fig. 42**, **Fig. 43** y **Fig. 44**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

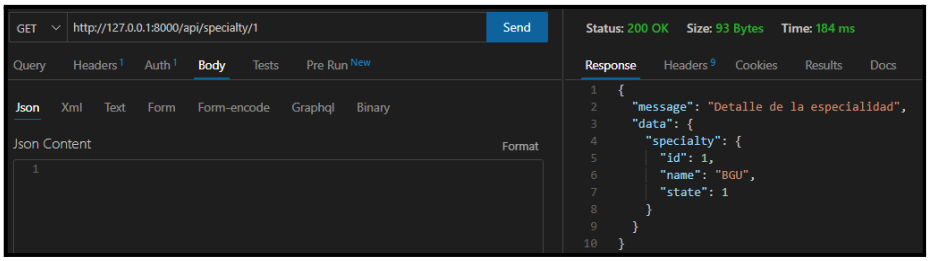

Fig. 41: Método *GET* para visualizar una especialidad.

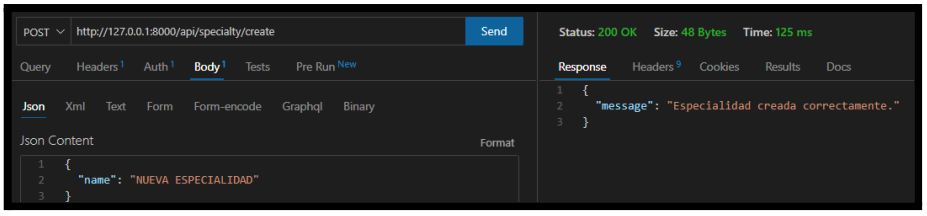

Fig. 42: Método *POST* para crear una especialidad.

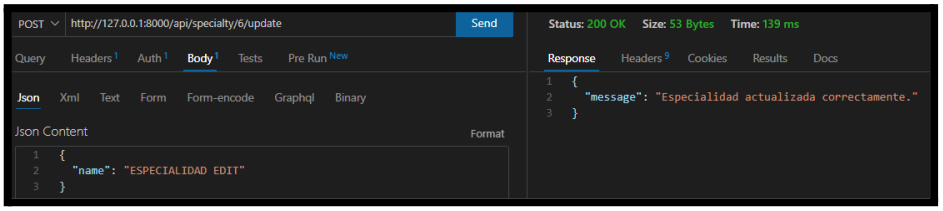

Fig. 43: Método *POST* para actualizar una especialidad.

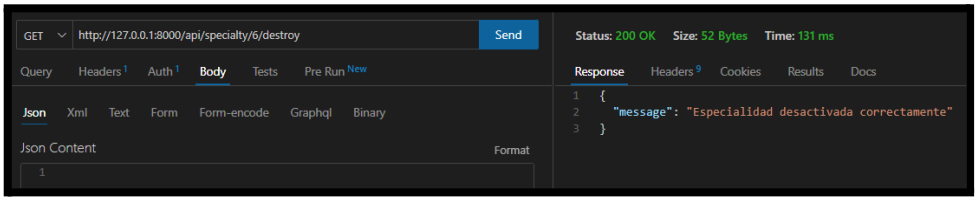

Fig. 44: Método *GET* para deshabilitar una especialidad.

Generar *endpoints* para gestionar asignaturas

Para la gestión de las asignaturas se ha creado varios métodos y rutas los cuales permiten al usuario con rol de secretario/a gestionar toda la información relacionada con las asignaturas.

Se ha implementado un método tipo *GET* con una ruta para la obtención de la información de una asignatura y métodos de tipo *POST* con rutas para la creación, actualización e inhabilitación de la información respectivamente, como se presenta en las figuras **Fig. 45**, **Fig. 46**, **Fig. 47** y **Fig. 48**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

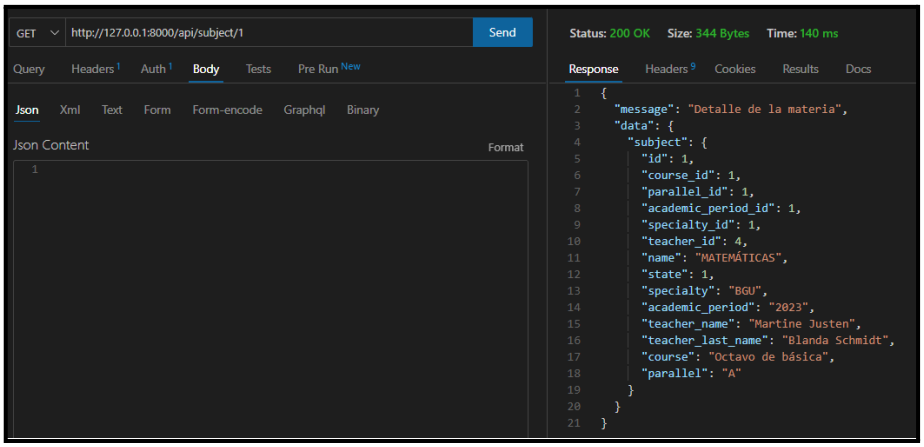

Fig. 45: Método *GET* para visualizar una asignatura.

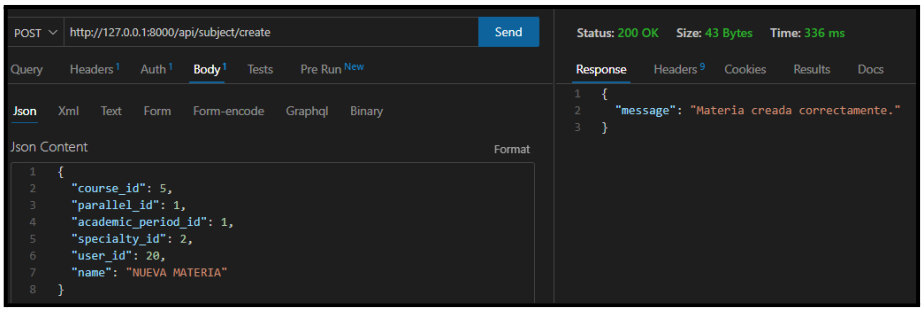

Fig. 46: Método *POST* para crear una asignatura.

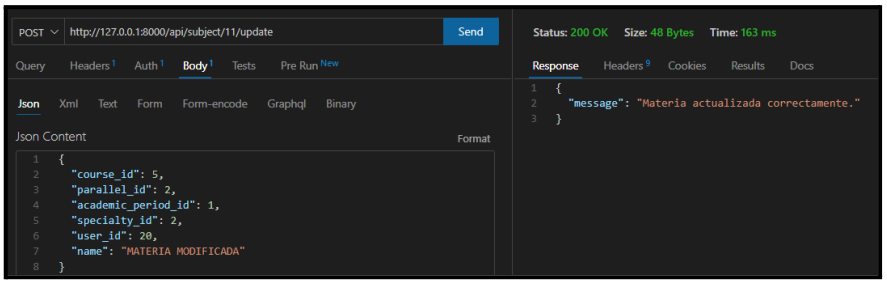

Fig. 47: Método *POST* para actualizar una asignatura.

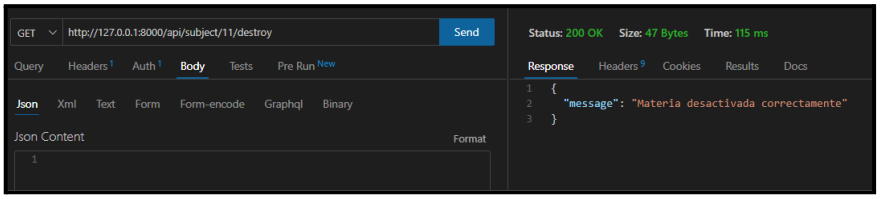

Fig. 48: Método *GET* para deshabilitar una asignatura.

2.3 Sprint 2. Implementación de *endpoints* para el módulo "profesor".

Este *Sprint* comprende las siguientes tareas:

- Generar *endpoints* para gestionar el perfil personal del profesor.

- Generar *endpoints* para gestionar calificaciones de estudiantes asignados a un profesor.
- Generar *endpoint* para visualizar el reporte de calificaciones de los estudiantes por asignatura.

Generar *endpoints* para gestionar el perfil personal del profesor

Para la gestión del perfil personal del profesor se tienen métodos y rutas que permiten al usuario con rol profesor consumir estos *endpoints* sin restricciones. Estos son de tipo *POST* y *GET* e incluyen la funcionalidad de poder mostrar y editar la información, tal y como se presenta en las figuras **Fig. 49** y **Fig. 50**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

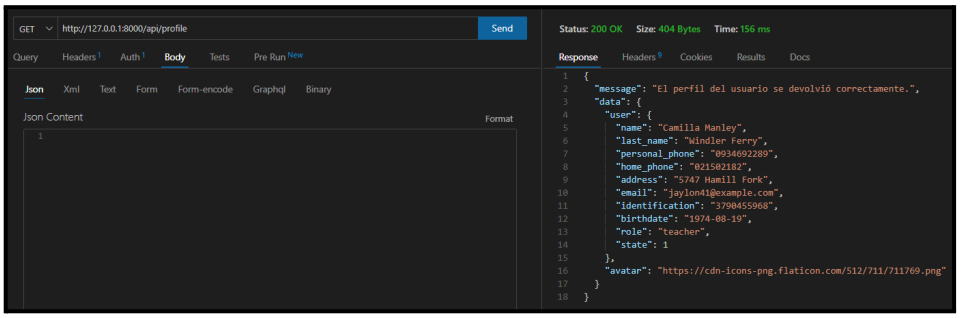

Fig. 49: Método *GET* para mostrar la información del perfil del profesor.

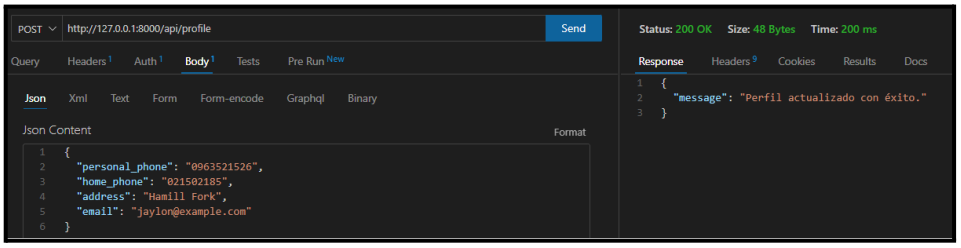

Fig. 50: Método *POST* para actualizar la información del perfil del profesor.

Generar *endpoints* para gestionar calificaciones de estudiantes asignados a un profesor

Para la gestión de calificaciones de los estudiantes se han implementado métodos y rutas que permiten al usuario con rol profesor administrar estos *endpoints* en base al tiempo asignado para su modificación en el periodo académico vigente. Estos son de tipo *POST* y *GET* e incluyen la funcionalidad de poder mostrar y editar las calificaciones, tal y como se presenta en las figuras **Fig. 51**, **Fig. 52**, **Fig. 53** y **Fig. 54**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

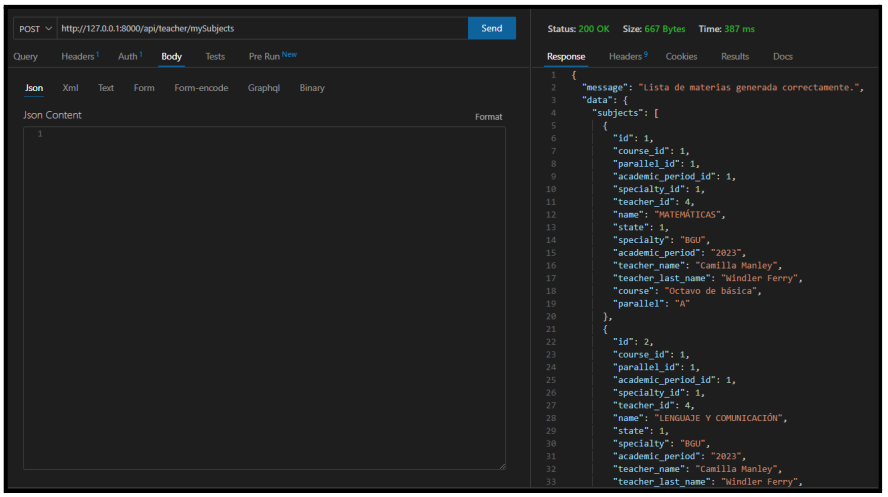

Fig. 51: Método *POST* para obtener las materias asignadas a un profesor.

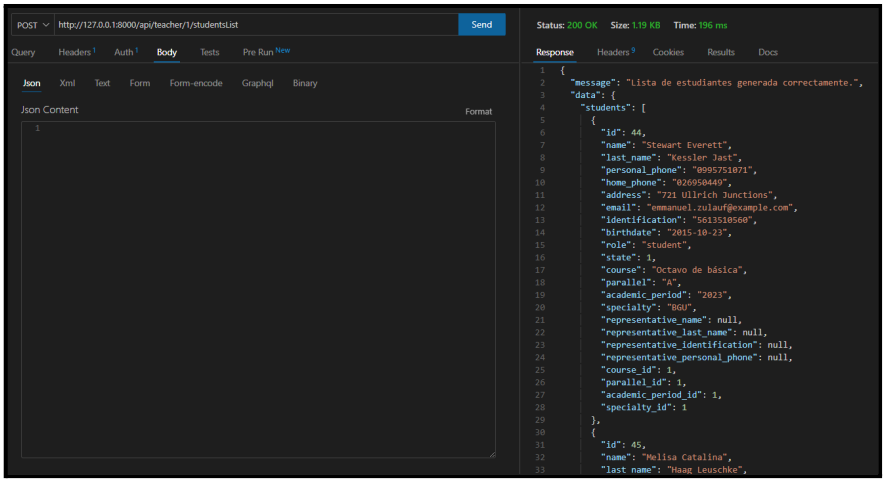

Fig. 52: Método *POST* para obtener la lista de estudiantes que cursan una materia.

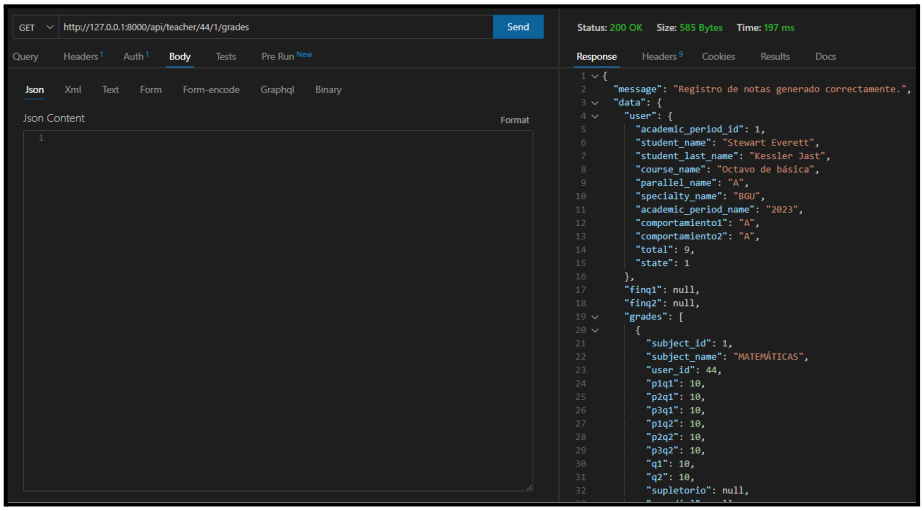

Fig. 53: Método *GET* para obtener las calificaciones de un estudiante.

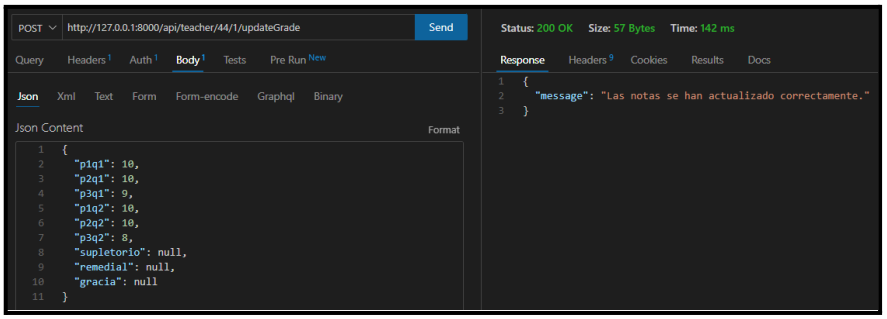

Fig. 54: Método *POST* para actualizar las calificaciones de un estudiante.

Generar *endpoint* para visualizar el reporte de calificaciones de los estudiantes por asignatura

Para la generación del reporte de calificaciones por materia para un profesor se ha implementado un método y ruta que permite al usuario con rol profesor consumir este *endpoint* sin restricciones. Este es de tipo *GET* e incluye la funcionalidad de generar los datos en un formato específico, tal y como se presenta en la figura **Fig. 55**. En este aspecto, el proceso que detalla el consumo de este *endpoint* se encuentra en el **ANEXO III** del presente documento.

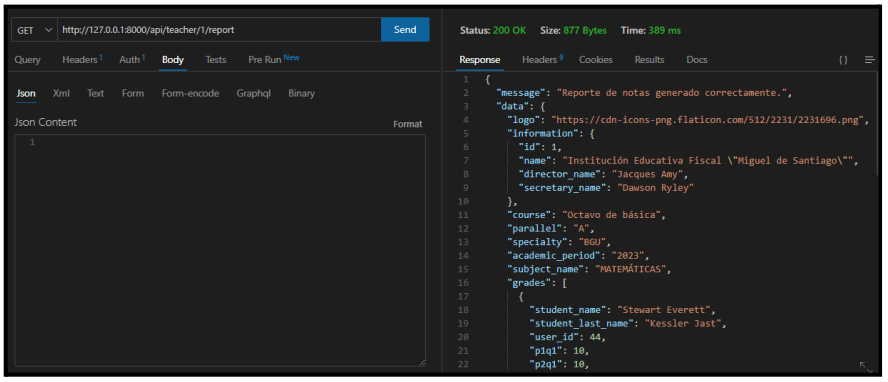

Fig. 55: Método *GET* para generar el reporte de calificaciones de un estudiante.

2.4 Sprint 3. Implementación de *endpoints* para el módulo "estudiante".

Este *Sprint* comprende las siguientes tareas:

- Generar *endpoint* para visualizar la información del perfil personal del estudiante.
- Generar *endpoints* para visualizar las calificaciones.
- Generar *endpoints* para visualizar el reporte de calificaciones individual.

Generar *endpoint* para visualizar la información del perfil personal del estudiante

Para la gestión del perfil personal del estudiante se ha implementado un método y ruta que permite al usuario con rol estudiante consumir este *endpoint* sin restricciones. Este es de tipo *GET* e incluye la funcionalidad de poder mostrar la información, tal y como se presenta en la figura **Fig. 56**. En este aspecto, el proceso que detalla el consumo de este *endpoint* se encuentra en el **ANEXO III** del presente documento.

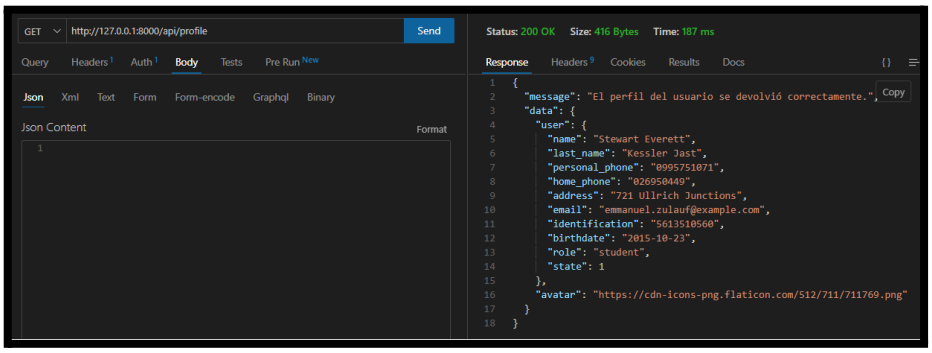

Fig. 56: Método *GET* para mostrar la información del perfil del estudiante.

Generar *endpoints* para visualizar calificaciones

Para la visualización de calificaciones de los estudiantes se ha implementado métodos y rutas que permiten al usuario con rol estudiante visualizar estos *endpoints*. Estos son de tipo *POST* y *GET* e incluyen la funcionalidad de poder mostrar las calificaciones realizando previamente una consulta a los cursos que tiene asignado un estudiante específico, tal y como se presenta en las figuras **Fig. 57** y **Fig. 58**. En este aspecto, el proceso que detalla el consumo de estos *endpoints* se encuentra en el **ANEXO III** del presente documento.

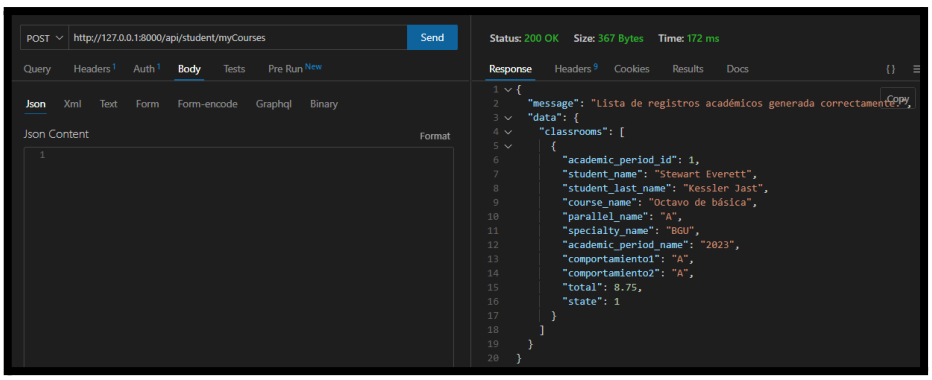

Fig. 57: Método *POST* para mostrar los cursos que tiene asignado un estudiante.

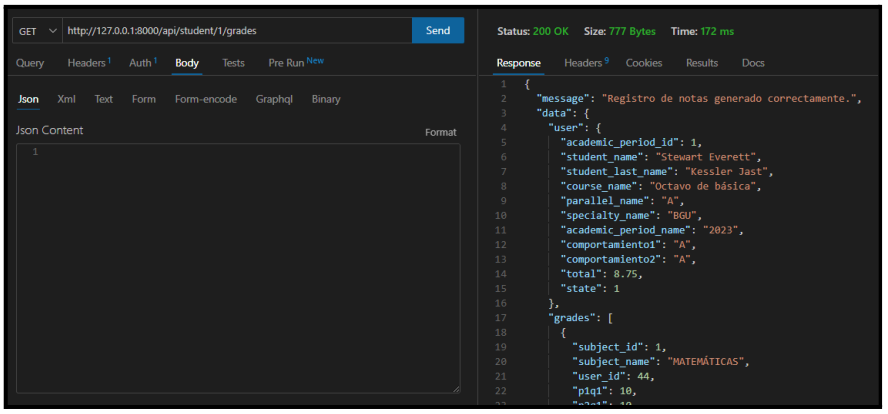

Fig. 58: Método *GET* para mostrar las calificaciones de un estudiante.

Generar *endpoints* para visualizar el reporte de calificaciones individual

Para la generación del reporte de calificaciones para un estudiante se ha implementado un método y ruta que permite al usuario con rol estudiante consumir este *endpoint* sin restricciones. Este es de tipo *POST* e incluye la funcionalidad de generar los datos en un formato específico, tal y como se presenta en la figura **Fig. 59**. En este aspecto, el proceso que detalla el consumo de este *endpoint* se encuentra en el **ANEXO III** del presente documento.

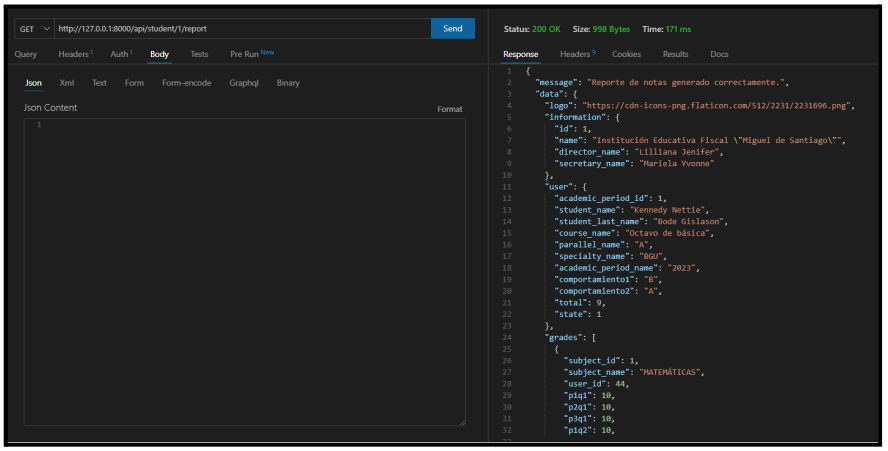

Fig. 59: Método *GET* para mostrar la información del perfil del estudiante.

2.5 Sprint 4. Pruebas de *endpoints*

Este *Sprint* comprende las siguientes tareas:

- Resultados de la ejecución de pruebas unitarias.
- Resultados de la ejecución de la prueba de carga.
- Resultad de la ejecución de la prueba de estrés.

Resultados de la ejecución de pruebas unitarias

Las pruebas unitarias son conocidas como una división de código de la cual se pone a prueba su funcionamiento, teniendo así el objetivo de validar el comportamiento en la lógica de un determinado objeto [37]. *Laravel* cuenta con librerías internas propias del *framework* que permiten realizar pruebas unitarias de manera sencilla mediante simulación de llamados, sin embargo, dichas pruebas alteran el funcionamiento del proyecto de *software* local.

La **Fig. 60** presenta una sección del código empleado para obtener la información del colegio, la cual se ha implementado para una de las pruebas del sistema *web*, adicionalmente en la **Fig. 61** se puede visualizar el resultado posterior a la ejecución de la prueba previamente mencionada. La ejecución del resto de pruebas, así como los resultados se encuentran en el **ANEXO II** del presente documento.

```php
// Prueba unitaria para obtener la información del colegio
public function test_obtener_informacion_del_colegio()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/information');
    $response->assertStatus(200);
}
```

Fig. 60: Fragmento de código para obtener la información del colegio.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ obtener informacion del colegio

  Tests:  1 passed
  Time:   0.18s
```

Fig. 61: Resultado de la prueba.

Con la obtención del resultado de las pruebas unitarias se afirma que los módulos del sistema *web* y los métodos de los *endpoints* funcionan adecuadamente, sin incidencias a nivel de código de por medio.

Resultados de la ejecución de la prueba de carga

La prueba de carga permite verificar el comportamiento que el sistema *web* tiene al momento de recibir varias peticiones hacia los *endpoints* y/o rutas definidas con sus respectivas funcionalidades. Así, si un sistema tiene la capacidad de ejecutarse correctamente con el número de peticiones establecidas para los usuarios a nivel de *software* quiere decir que el sistema es adecuado para dicha cantidad.

La **TABLA V** presenta la cantidad con la cual el sistema puede recibir peticiones HTTPS sin verse sometido a sus límites de respuesta, estas pruebas se han realizado utilizando *Apache JMeter* versión 5.5, una herramienta que permite enviar múltiples peticiones a los *endpoints* en tiempo real. La descripción de la ejecución de la prueba de carga, así como sus resultados, se encuentran en el **ANEXO II** del presente documento.

TABLA V: Ejecución de la prueba de carga para los *endpoints*.

CANTIDAD	TIEMPO	OBSERVACIÓN
2040	1 segundo	Completamente funcional

Con la obtención del resultado de las pruebas de carga se demuestra que el sistema *web* permite realizar las peticiones HTTPS adecuadas para los usuarios finales, además de no presentar ningún fallo en sus tiempos de ejecución, receptando cada petición correctamente.

Resultados de la ejecución de la prueba de estrés

La prueba de estrés se utiliza cuando se requiere verificar la capacidad que tiene el sistema *web* para responder a peticiones HTTPS a un nivel duplicado al que debería tener con normalidad.

La **TABLA VI** presenta la cantidad con la que el sistema *web* puede recibir peticiones HTTPS viéndose sometido a sus límites de respuesta, estas pruebas se han realizado utilizando la misma herramienta que para las pruebas de carga, *Apache JMeter*. La descripción tanto de la ejecución, así como los resultados de la prueba de estrés se encuentran en el **ANEXO II** del presente documento.

TABLA VI: Ejecución de la prueba de estrés para los *endpoints*.

CANTIDAD	TIEMPO	OBSERVACIÓN
4080	1 segundo	Completamente funcional

Con la obtención del resultado de las pruebas de estrés se contempla una adaptabilidad adecuada al doble de peticiones que el sistema *web* soporta normalmente, además de no presentar ningún fallo en sus tiempos de ejecución, receptando cada petición correctamente

2.6 Sprint 5. Despliegue de *endpoints*

Una vez finalizada la codificación y pruebas de los *endpoints*, el *sprint* 5 tiene como objetivo el despliegue a producción en la plataforma de *Heroku* de los *endpoints* desarrollados.

El detalle se lo puede encontrar en el **ANEXO IV**. Para la etapa inicial como se identifica en la **Fig. 62** se puede visualizar el proyecto creado y desplegado en el panel de control principal.

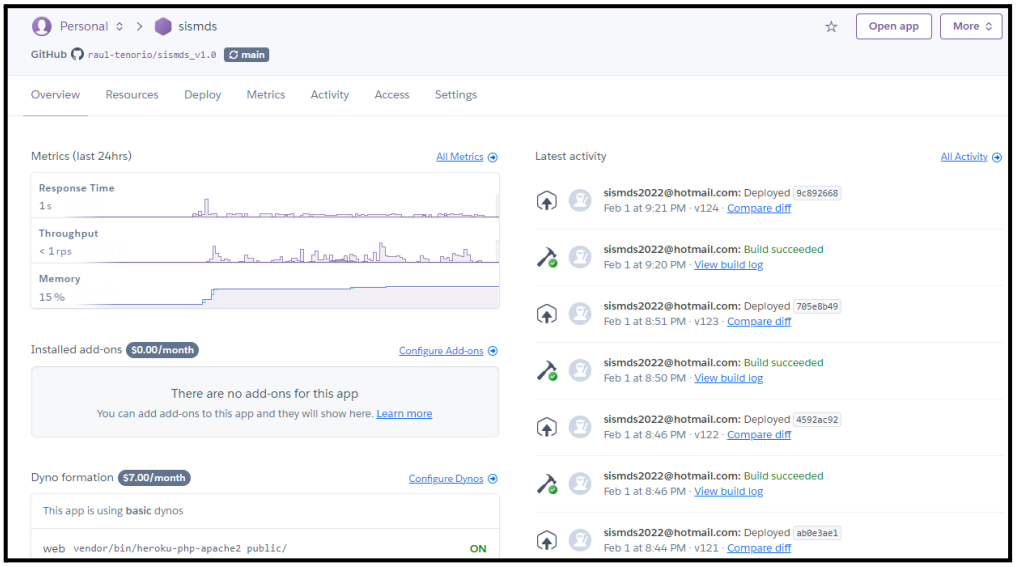

Fig. 62: Despliegue de *endpoints* a *Heroku*.

4 CONCLUSIONES

A continuación, se presentan las conclusiones que se han obtenido a lo largo del desarrollo del actual trabajo de integración curricular.

- El sistema *web* y los *endpoints* desarrollados cumplen en su totalidad con los alcances, objetivos y requerimientos definidos para el proyecto, permitiendo al I.E.F Miguel de Santiago contar con un sistema *web* para la gestión académica, automatizando los procesos para tener una mayor agilidad y eficiencia para controlar los aspectos que sean necesarios respecto a la gestión académica.

- El hacer uso de la metodología ágil *Scrum* en el proceso de desarrollo del presente trabajo de integración curricular, ha brindado la posibilidad de tener avances que sean entregables para cada *Sprint*, permitiendo así tener un sistema completo validando todas sus funcionalidades en los periodos de tiempo determinados.

- El emplear la arquitectura Modelo-Vista-Controlador en el desarrollo para la generación de *endpoints,* ha permitido estructurar y manejar el proyecto bajo un estándar específico que es entendible para un desarrollador, en caso de integrar un nuevo miembro al equipo de desarrollo, así como de un mantenimiento mucho más sencillo.

- La utilización de una base de datos SQL, ha brindado la posibilidad de gestionar adecuadamente la información permitiendo garantizar la integridad de los datos almacenados, además de que las relaciones establecidas también se gestionan por este medio.

- En el transcurso del proceso de codificación se ha empleado tanto herramientas como librerías para la generación de los *endpoints,* mediante los cuales se ha obtenido un resultado satisfactorio debido a su alta compatibilidad y un proceso de desarrollo sencillo.

- En la fase de pruebas aplicada a los *endpoints* ha posibilitado comprobar todas sus funcionalidades, además de determinar su resistencia ante una carga exigente de peticiones.

5 RECOMENDACIONES

A continuación, se presentan las recomendaciones obtenidas a lo largo del desarrollo del actual trabajo de integración curricular.

- Con el objetivo de proteger y mantener la información almacenada, se recomienda realizar cada cierto periodo de tiempo *backups* de la Base de Datos para poder garantizar la existencia de un respaldo en caso de perdidas.
- En el caso de que se requiera añadir un nuevo módulo al sistema *web*, se recomienda contactar al equipo de desarrollo para informar sobre nuevas necesidades y/o requerimientos con el fin mantener el sistema actualizado.

6 REFERENCIAS BIBLIOGRÁFICAS.

[1] SYDLE, «SYDLE,» 04 Mayo 2022. [En línea]. Available: https://www.sydle.com/es/blog/gestion-de-procesos-en-la-educacion-626c2a0361423f655c608217/. [Último acceso: 19 Noviembre 2022].

[2] K. Bielefeld, «Boxlight,» 11 Febrero 2020. [En línea]. Available: https://lablog.boxlight.com/automatizacion-de-procesos-en-el-aula-para-una-mayor-eficiencia. [Último acceso: 19 Noviembre 2022].

[3] Deloitte, «Deloitte,» Septiembre 2016. [En línea]. Available: https://www2.deloitte.com/content/dam/Deloitte/mx/Documents/strategy/pov_robotics.pdf. [Último acceso: 19 Noviembre 2022].

[4] R. Mujica, «issuu,» 27 Marzo 2016. [En línea]. Available: https://issuu.com/ruthmujica/docs/ruthmujica_revista_8. [Último acceso: 19 Noviembre 2022].

[5] Twinlan, «Twinlan,» 2022. [En línea]. Available: https://twinlan.com/blog-consultoria-tecnologica/ultimas-publicaciones/riesgos-de-trabajar-con-sistemas-sin-actualizar-o-sin-soporte. [Último acceso: 19 Noviembre 2022].

[6] A. Pareja Glass y E. Näslund-Hadley, «BID,» 27 Febrero 2015. [En línea]. Available: https://n9.cl/bh0i9s. [Último acceso: 20 Noviembre 2022].

[7] UNIR, «UNIR,» 06 Abril 2021. [En línea]. Available: https://n9.cl/rkiso. [Último acceso: 20 Noviembre 2022].

[8] R. Pressman, Ingenieria de Software. Un enfoque práctico, Mexico: MCGRAW-HILL, 2005.

[9] M. A. Mascheroni, C. Greiner, R. Petris, G. Dapozo y M. Estayno, «SEDICI Repositorio Institucional de la UNPL,» 06 Agosto 2012. [En línea]. Available: http://sedici.unlp.edu.ar/handle/10915/19202. [Último acceso: 20 Noviembre 2022].

[10] Admin, «Addappto,» 21 Agosto 2015. [En línea]. Available: http://www.addappto.com/que-es-un-sistema-web/. [Último acceso: 20 Noviembre 2022].

[11] Amazon Web Services, «AWS,» 2022. [En línea]. Available: https://aws.amazon.com/es/relational-database/. [Último acceso: 20 Noviembre 2022].

[12] AlwaysData, «AlwaysData,» 2022. [En línea]. Available: https://www.alwaysdata.com/en/. [Último acceso: 20 Noviembre 2022].

[13] KeepCoding, «KeepCoding,» 09 Septiembre 2022. [En línea]. Available: https://n9.cl/91g2b. [Último acceso: 20 Noviembre 2022].

[14] Laravel, «Laravel,» 2022. [En línea]. Available: https://laravel.com/docs/9.x. [Último acceso: 20 Noviembre 2022].

[15] Heroku, «Heroku Dev Center,» 2022. [En línea]. Available: https://devcenter.heroku.com/. [Último acceso: 20 Noviembre 2022].

[16] Cloudinary, «Cloudinary,» 2022. [En línea]. Available: https://cloudinary.com/. [Último acceso: 20 Noviembre 2022].

[17] V. R. Anshu Soni, «API Features Individualizing of Web Services: REST and SOAP,» *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 8, nº 9S, pp. 664-671, 2019.

[18] Epitech España, «Epitech-it,» 3 Junio 2021. [En línea]. Available: https://www.epitech-it.es/backend/. [Último acceso: 20 Noviembre 2022].

[19] E. Yacuzzi, «El estudio de caso como metodología de investigación: Teoría, mecanismos casuales, validación,» 2005. [En línea]. Available: https://www.econstor.eu/bitstream/10419/84390/1/496805126.pdf. [Último acceso: 20 Noviembre 2022].

[20] E. G. Maida y J. Pacienzia, «Repositorio Institucional UCA,» 2015. [En línea]. Available: https://repositorio.uca.edu.ar/handle/123456789/522. [Último acceso: 20 Noviembre 2022].

[21] Red Hat, «Red Hat,» 19 Julio 2022. [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 20 Noviembre 2022].

[22] M. A. Alvarez, «Desarrolloweb.com,» 28 Julio 2020. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 20 Noviembre 2022].

[23] E. Abellán, «We are marketing,» Mayo 2020. [En línea]. Available: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html. [Último acceso: 20 Noviembre 2022].

[24] J. M. de Agar Tirado, «Mamá... ¿Qué es Scrum?,» 29 Abril 2020. [En línea]. Available: https://mamaqueesscrum.com/2020/04/29/que-es-un-development-team-os-proponemos-una-dinamica/. [Último acceso: 20 Noviembre 2022].

[25] C. A. Guerra, «Sg.com,» 28 Diciembre 2007. [En línea]. Available: https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia. [Último acceso: 15 Junio 2022].

[26] M. Rehkopf, «Atlassian Agile coach,» s.f. [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: 15 07 2021].

[27] EALDE, «EALDE,» 07 Agosto 2019. [En línea]. Available: https://www.ealde.es/product-backlog-sprint-backlog/. [Último acceso: 30 Junio 2022].

[28] M. Garcia, «ITtude,» 17 Julio 2020. [En línea]. Available: https://ittude.com.ar/b/scrum/que-es-el-sprint-backlog/. [Último acceso: 18 Diciembre 2022].

[29] J. M. Aguilar, «campusmvp.es,» 15 Octubre 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx. [Último acceso: 18 Diciembre 2022].

[30] Euroinnova, «Euroinnova,» [En línea]. Available: https://www.euroinnova.edu.es/blog/herramientas-de-programacion#:~:text=Las%20herramientas%20de%20programaci%C3%B3n%2C%20o,o%20apoyar%20programas%20y%20aplicaciones.. [Último acceso: 05 Enero 2023].

[31] R. Altube Vera, «OpenWebinars,» 31 Marzo 2021. [En línea]. Available: https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/. [Último acceso: 05 Enero 2022].

[32] A. Robledano, «OpenWebinars,» 24 Septiembre 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-mysql/. [Último acceso: 05 Enero 2023].

[33] Programación en Castellano, «Programacion.net,» 2021. [En línea]. Available: https://programacion.net/articulo/10_paquetes_imprescindibles_para_laravel_que_no_debes_dejar_pasar_1914. [Último acceso: 05 Enero 2023].

[34] Laravel LLC., «Laravel,» 2023. [En línea]. Available: https://laravel.com/docs/9.x/sanctum. [Último acceso: 05 Enero 2023].

[35] Cloudinary, «Cloudinary,» 2023. [En línea]. Available: https://cloudinary.com/blog/laravel-cloudinary-v2-release-update. [Último acceso: 05 Enero 2023].

[36] J. Ur Rehman, «AIIPHPTricks,» 6 Mayo 2022. [En línea]. Available: https://www.allphptricks.com/how-to-send-email-in-laravel-9-using-smtp/. [Último acceso: 05 Enero 2023].

7 ANEXOS

A continuación, se muestra la división de los Anexos que se han utilizado para el desarrollo del sistema *web* y *endpoints*.

- **ANEXO I.** Certificado de Originalidad
- **ANEXO II.** Manual Técnico.

- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Manual de Instalación.

ANEXO I

ANEXO II

Recopilación de requerimientos

A continuación, la **TABLA I** se hacen presentes los requerimientos obtenidos en base a las solicitudes realizadas por el *Product Owner*.

TABLA I: Recopilación de requerimientos.

Recopilación de requerimientos		
Tipo de sistema	**ID-RR**	**Enunciado del Ítem**
Sistema *web*	**RR001**	Como usuario con perfil profesor, secretaria o estudiante necesita generar varios *endpoints* para: ● Iniciar y cerrar sesión.
	RR002	Como usuario con perfil profesor necesita generar varios *endpoints* para: ● Gestionar su perfil personal.
	RR004	Como usuario con perfil secretaria necesita generar varios *endpoints* para: ● Gestionar su perfil personal.
	RR005	Como usuario con perfil estudiante necesita generar varios *endpoints* para: ● Gestionar su perfil personal.
	RR006	Como usuario profesor necesita generar varios *endpoints* para: ● Gestionar calificaciones.
	RR007	Como usuario profesor necesita generar un *endpoint* para: ● Crear reporte de calificaciones.
	RR008	Como usuario estudiante necesita generar un *endpoint* para: ● Obtener calificaciones.

	RR019	Como usuario estudiante necesita generar un *endpoint* para: ● Crear reporte de calificaciones.
	RR010	Como usuario secretaria necesita generar un *endpoint* para: ● Gestionar información del colegio.
	RR011	Como usuario secretaria necesita generar varios *endpoints* para: ● Gestionar estudiantes.
	RR012	Como usuario secretaria necesita generar varios *endpoints* para: ● Gestionar profesores.
	RR013	Como usuario secretaria necesita generar varios *endpoints* para: ● Activar o desactivar usuarios con perfil profesor, secretaria y estudiante.
	RR014	Como usuario secretaria necesita generar varios *endpoints* para: ● Gestionar periodos académicos, cursos y especialidades.
	RR015	Como usuario secretaria necesita generar varios *endpoints* para: ● Gestionar asignaturas y paralelos.
	RR016	Como usuario secretaria necesita generar varios *endpoints* para: ● Gestionar calificaciones.

Historias de Usuario

Finalizada la etapa de Recopilación de requerimientos, se desarrolla cada Historia de Usuario para el sistema *web* individualmente. En ese sentido, se presentan las 16 Historias de Usuario descritas en base a los requerimientos establecidos por el *Product Owner*.

TABLA II: Historia de usuario 001 – Generar varios *endpoints* para iniciar y cerrar sesión.

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario**: Profesor, secretaria, estudiante.
Nombre Historia: Generar varios *endpoints* para iniciar y cerrar sesión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil profesor, secretaria o estudiante, necesita generar varios *endpoints* para iniciar y cerrar sesión. Por lo tanto, se definen las rutas respectivas para la autenticación.	
Observación: Ninguna.	

TABLA III: Historia de usuario 002 – Generar varios *endpoints* para gestionar el perfil personal del profesor.

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario**: Profesor.
Nombre Historia: Generar varios *endpoints* para gestionar el perfil personal del profesor.	
Prioridad en negocio: Bajo	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil profesor necesita generar varios *endpoints* para obtener la información de su perfil que se compone de los siguientes datos: ● Nombres.	

• Apellidos. • Cédula. • Fecha de nacimiento. • Dirección de domicilio. • Teléfono convencional. • Teléfono celular. • Correo electrónico. Además, se establece un control para que el usuario con perfil profesor solo pueda modificar los datos de: • Dirección de domicilio. • Teléfono convencional. • Teléfono celular. • Correo electrónico.
Observación: Ninguna.

TABLA IV: Historia de usuario 003 - Generar varios *endpoints* para gestionar el perfil personal de la secretaria.

HISTORIA DE USUARIO	
Identificador (ID): HU003	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar el perfil personal de la secretaria.	
Prioridad en negocio: Bajo	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para obtener y modificar la información de su perfil enviando y recibiendo los siguientes datos: • Nombres. • Apellidos. • Cédula.	

● Fecha de nacimiento. ● Dirección de domicilio. ● Teléfono convencional. ● Teléfono celular. ● Correo electrónico.
Observación: Ninguna.

TABLA V: Historia de usuario 004 – Generar varios *endpoints* para gestionar el perfil personal del estudiante.

HISTORIA DE USUARIO	
Identificador (ID): HU004	**Usuario**: Estudiante
Nombre Historia: Generar varios *endpoints* para gestionar el perfil personal del estudiante.	
Prioridad en negocio: Bajo	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil estudiante necesita generar varios *endpoints* para obtener la información de su perfil que se compone de los siguientes datos: ● Nombres. ● Apellidos. ● Cédula. ● Fecha de nacimiento. ● Dirección de domicilio. ● Teléfono convencional. ● Teléfono celular. ● Correo electrónico.	

Observación: Ninguna.

TABLA VI Historia de usuario 006 – Generar varios *endpoints* para crear reporte de calificaciones por materia.

HISTORIA DE USUARIO	
Identificador (ID): HU006	**Usuario**: Profesor.
Nombre Historia: Generar varios *endpoints* para crear reporte de calificaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil profesor necesita generar varios *endpoints* para crear reportes de las calificaciones de los estudiantes que pertenecen a su clase por asignatura, curso y paralelo.	
Observación: Ninguna.	

TABLA VII Historia de usuario 007 – Generar un endpoint para obtener calificaciones.

HISTORIA DE USUARIO	
Identificador (ID): HU007	**Usuario**: Estudiante.
Nombre Historia: Generar un *endpoint* para obtener calificaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 2	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil estudiante necesita generar un *endpoint* para obtener las calificaciones a lo largo del periodo académico, los datos enviados son: ● Primer quimestre. o Primer parcial.	

o Segundo parcial. o Tercer parcial. ● Segundo quimestre. o Primer parcial. o Segundo parcial. o Tercer parcial. ● Nota del examen supletorio. ● Nota del examen remedial. ● Nota del examen de gracia. ● Promedio final por asignatura. ● Promedio total.
Observación: Ninguna.

TABLA VIII Historia de usuario 008 - Generar un *endpoint* para crear un reporte de calificaciones por curso.

HISTORIA DE USUARIO	
Identificador (ID): HU008	**Usuario**: Estudiante.
Nombre Historia: Generar un *endpoint* para crear un reporte de calificaciones por curso.	
Prioridad en negocio: Bajo	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 2	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil estudiante necesita generar un *endpoint* para crear un reporte de calificaciones.	
Observación: Ninguna.	

TABLA IX Historia de usuario 009 - Generar un *endpoint* para gestionar información del colegio.

HISTORIA DE USUARIO	
Identificador (ID): HU009	**Usuario**: Secretaria.
Nombre Historia: Generar un *endpoint* para gestionar información del colegio.	
Prioridad en negocio: Bajo	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 2	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar un *endpoint* para consultar, ingresar y/o actualizar el nombre de la secretaria y la rectora de la institución.	
Observación: Ninguna.	

TABLA X Historia de usuario 010 - Generar varios *endpoints* para gestionar estudiantes.

HISTORIA DE USUARIO	
Identificador (ID): HU010	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar estudiantes.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alto
Iteración Asignada: 2	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para crear usuarios con perfil estudiante, así como consultar o actualizar la información de estos. Además, los datos que se generan son: ● Nombres. ● Apellidos. ● Cédula. ● Fecha de nacimiento. ● Dirección de domicilio.	

● Teléfono convencional. ● Teléfono celular. ● Correo electrónico. ● Datos de lo(s) representante(s).
Observación: Ninguna.

TABLA XI Historia de usuario 011 - Generar varios *endpoints* para gestionar profesores.

HISTORIA DE USUARIO	
Identificador (ID): HU011	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar profesores.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para crear usuarios con perfil profesor o secretaria, así como consultar o actualizar la información de estos. Además, los datos que se generan son: ● Nombres. ● Apellidos. ● Cédula. ● Fecha de nacimiento. ● Dirección de domicilio. ● Teléfono convencional. ● Teléfono celular. ● Correo electrónico.	
Observación:	

Ninguna.

TABLA XII Historia de usuario 012 - Generar varios *endpoints* para activar o desactivar usuarios.

HISTORIA DE USUARIO	
Identificador (ID): HU012	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para activar o desactivar usuarios.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para activar o desactivar usuarios con perfil profesor, secretaria o estudiante.	
Observación: Ninguna.	

TABLA XIII Historia de usuario 013 - Generar varios *endpoints* para gestionar periodos académicos, cursos y especialidades.

HISTORIA DE USUARIO	
Identificador (ID): HU013	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar periodos académicos, cursos y especialidades.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para gestionar periodos académicos, cursos y especialidades.	
Observación: Ninguna.	

TABLA XIV Historia de usuario 014 - Generar varios *endpoints* para gestionar asignaturas y paralelos

HISTORIA DE USUARIO

Identificador (ID): HU014	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar asignaturas y paralelos.	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para gestionar asignaturas y paralelos.	
Observación: Ninguna.	

TABLA XV Historia de usuario 015 - Generar varios *endpoints* para gestionar calificaciones.

HISTORIA DE USUARIO	
Identificador (ID): HU015	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para gestionar calificaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para gestionar calificaciones de los estudiantes en donde cada uno de ellos será identificado por: ● Nombre y Apellido. ● Cédula. Los datos que se pueden modificar por cada estudiante son: ● Primer quimestre. o Primer parcial. o Segundo parcial. o Tercer parcial.	

• Segundo quimestre.
  o Primer parcial.
  o Segundo parcial.
  o Tercer parcial.
• Nota del examen supletorio.
• Nota del examen remedial.
• Nota del examen de gracia.
• Promedio final por asignatura.
• Promedio total.

Observación: Ninguna.

TABLA XVI Historia de usuario 16 - Generar varios *endpoints* para crear reporte de calificaciones.

HISTORIA DE USUARIO	
Identificador (ID): HU016	**Usuario**: Secretaria.
Nombre Historia: Generar varios *endpoints* para crear reporte de calificaciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Raúl Tenorio	
Descripción: El usuario con perfil secretaria necesita generar varios *endpoints* para crear reportes de las calificaciones de los estudiantes.	
Observación: Ninguna.	

Product Backlog

La **TABLA XVII** se maneja mediante el orden prioritario de los requerimientos para el sistema *web*. Por esta razón, cada uno es clasificado en base a las necesidades del *Product Owner* además del grado de complejidad de los ítems.

TABLA XVII: ***Product Backlog.***

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID – HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU001	Generar varios *endpoints* para iniciar sesión y cerrar sesión.	1	Finalizado	Alta
HU002	Generar varios *endpoints* para gestionar el perfil personal del profesor.	1	Finalizado	Baja
HU003	Generar varios *endpoints* para gestionar el perfil personal de la secretaria.	1	Finalizado	Baja
HU004	Generar varios *endpoints* para gestionar el perfil personal del estudiante.	1	Finalizado	Baja
HU005	Generar varios *endpoints* para gestionar calificaciones.	1	Finalizado	Alta
HU006	Generar varios *endpoints* para crear reportes de calificaciones.	2	Finalizado	Media
HU007	Generar un *endpoint* para obtener calificaciones.	2	Finalizado	Media
HU008	Generar un *endpoint* para crear un reporte de calificaciones por materia.	2	Finalizado	Baja
HU009	Generar un *endpoint* para gestionar información del colegio.	2	Finalizado	Baja
HU010	Generar varios *endpoints* para gestionar estudiantes.	2	Finalizado	Alta
HU011	Generar varios *endpoints* para gestionar profesores y secretarias.	3	Finalizado	Alta
HU012	Generar varios *endpoints* para activar o desactivar usuarios.	3	Finalizado	Media
HU013	Generar varios *endpoints* para gestionar periodos académicos, cursos y especialidades.	3	Finalizado	Alta
HU014	Generar varios *endpoints* para gestionar asignaturas y paralelos.	3	Finalizado	Alta
HU015	Generar varios *endpoints* para gestionar calificaciones por materia.	3	Finalizado	Alta

HU016	Generar varios *endpoints* para crear reportes de calificaciones por curso.	3	Finalizado	Media

Sprint Backlog

En la **TABLA XVIII** se presentan los cinco Sprints definidos para el desarrollo completo del sistema *web*, se listan las actividades y el tiempo requerido para cada una de ellas con el fin de cumplir con los entregables designados por el *Product Owner*.

TABLA XVIII: *Sprint Backlog*.

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID - SB	**NOMBRE**	**MÓDULO**	**ID - HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	MÓDULO DE PREPARACIÓN Y CONFIGURACIÓN	Módulo de elaboración de la base de datos y configuración de las herramientas de trabajo.	HU000	Creación de la base de datos y configuración de las herramientas de trabajo.	● Diseño e implementación de la base de datos en *MySQL.* ● Preparación del entorno de desarrollo en *Visual Studio Code.* ● Importación y configuración de recursos desde *Composer.*	20H
SB001	MÓDULOS PARA LA SECRETARIA	Módulo de inicio de sesión.	HU001	Iniciar y cerrar sesión.	● Diseño e implementación de *endpoints* para el inicio y cierre de sesión.	90H
		Módulo perfil personal secretaria.	HU003	Gestionar el perfil personal de la secretaria.	● Diseño e implementación de *endpoints* para gestionar el perfil personal de la secretaria.	
		Módulo calificaciones.	HU015	Gestionar calificaciones.	● Diseño e implementación de *endpoints* para gestionar calificaciones.	
		Módulo reportes.	HU016	Crear reporte de calificaciones.	● Diseño e implementación de *endpoints* para generar el reporte de calificaciones.	
		Módulo información del colegio.	HU009	Gestionar información del colegio.	● Diseño e implemetación de *endpoints* para gestionar información del colegio.	
		Módulo gestión de estudiantes.	HU010	Gestionar estudiantes.	● Diseño e implementación de *endpoints* para gestionar estudiantes.	

		Módulo gestión de profesores y sercretarias.	HU011	Gestionar profesores.	• Diseño e implementación de *endpoints* para gestionar profesores.	
		Módulo habilitación y deshabilitación de usuarios.	HU012	Activar o desactivar usuarios.	• Diseño e implementación de *endpoints* para activar o desactivar usuarios.	
		Módulo de gestión de periodos académicos, cursos y especialidades.	HU013	Gestionar periodos académicos, cursos y especialidades.	• Diseño e implementación de *endpoints* para gestionar periodos académicos. • Diseño e implementación de endpoints para gestionar cursos. • Diseño e implementación de endpoints para gestionar especialidades.	
		Módulo de gestión de asignaturas y paralelos.	HU014	Gestionar asignaturas y paralelos.	• Diseño e implementación de endpoints para gestionar asignaturas. • Diseño e implementación de endpoints para gestionar paralelos.	
SB002	MÓDULOS PARA EL PROFESOR	Módulo perfil personal profesor.	HU002	Gestionar el perfil personal del profesor.	• Diseño e implementación de *endpoints* para gestionar el perfil personal del profesor.	40H
		Módulo calificaciones.	HU005	Gestionar calificaciones por materia.	• Diseño e implementación de *endpoints* para gestionar calificaciones de estudiantes asignados a un profesor.	
		Módulo reportes.	HU006	Crear reporte de calificaciones por materia.	• Diseño e implementación de *endpoints* para generar el reporte de calificaciones de los estudiantes que están asignados a un profesor en específico.	
SB003	MÓDULOS PARA EL ESTUDIANTE	Módulo perfil personal estudiante.	HU004	Gestionar el perfil personal del estudiante.	• Diseño e implementación de *endpoints* para gestionar el perfil personal del estudiante.	40H

		Módulo obtener calificaciones.	HU007	Visualizar calificaciones.	• Diseño e implementación de *endpoints* para obtener el reporte de calificaciones del estudiante.	
		Módulo reportes.	HU008	Crear reporte de calificaciones por curso.	• Diseño e implementación de *endpoints* para generar el reporte de calificaciones de un estudiante en específico.	
SB004	PRUEBAS DE ENDPOINTS	• Pruebas unitarias. • Pruebas de carga. • Pruebas de estrés.				20H
SB005	DESPLIEGUE DE ENDPOINTS	• Despliegue de *endpoints* a *Heroku*.				10H
DOCUMENTACIÓN		• Informe Técnico • Anexos				20H
TOTAL						**240H**

Pruebas

Una vez finalizada la etapa de codificación se han implementado pruebas unitarias, de carga y de estrés para corroborar la correcta ejecución del código en los *endpoints* que se han desarrollado. Estas pruebas se han ejecutado para los módulos disponibles.

Pruebas unitarias

La figura muestra la ejecución de los *endpoints* que se han desarrollado desde la **Fig. 1** a la **Fig. 100** presentando las pruebas realizadas en los *endpoints* generados con sus respectivos resultados.

```php
// Prueba unitaria para el inicio de sesión
public function test_inicio_de_sesion()
{
    $response = $this->post('/api/login', [
        "identification" => "3776382111",
        "password" => "sismds"
    ]);
    $response->assertStatus(200);
}
```

Fig. 1: Prueba unitaria #1 Inicio de sesión.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\GeneralTest
  ✓ inicio de sesion

  Tests:  1 passed
  Time:   0.20s

PS C:\Users\USER\sismds_v1.0>
```

Fig. 2: Resultado prueba unitaria #1.

```php
// Prueba unitaria para mostrar la información de una secretaria
public function test_mostrar_secretaria()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/secretary/1');
    $response->assertStatus(200);
}
```

Fig. 3: Prueba unitaria #2 Mostrar información de una secretaria.

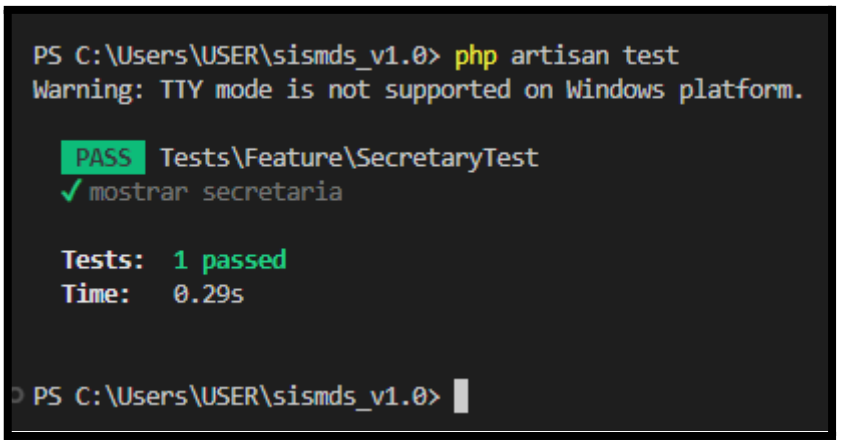

Fig. 4: Resultado prueba unitaria #2.

```php
// Prueba unitaria para actualizar a un usuario con rol secretaria
public function test_actualizar_secretaria()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Alexa Jimena",
        "last_name" => "Donnelly Schwalter",
        "personal_phone" => "0986352654",
        "home_phone" => "021280307",
        "address" => "4549 Quito",
        "email" => "juanca13@gmail.com",
        "identification" => "8055916104",
        "birthday" => "1995-05-09"
    ];
    $response = $this->actingAs($user)->post('/api/secretary/1/update', $data);
    $response->assertStatus(200);
}
```

Fig. 5: Prueba unitaria #3 Actualizar secretaria.

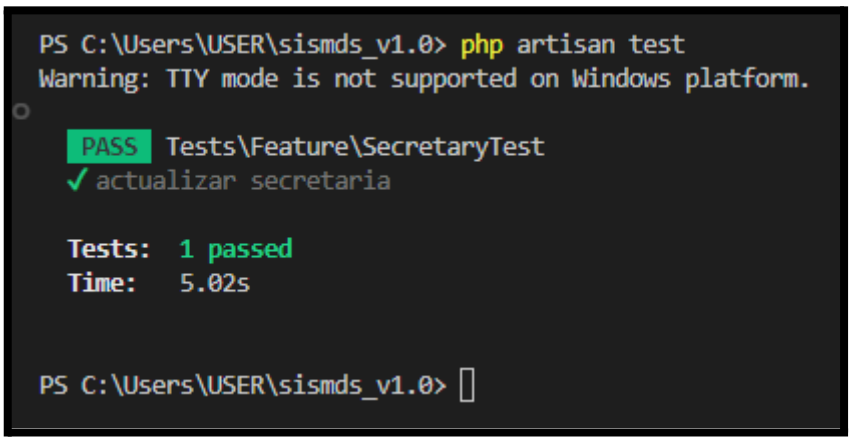

Fig. 6: Resultado prueba unitaria #3.

```php
// Prueba unitaria para obtener las calificaciones de un estudiante
public function test_obtener_calificaciones_de_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/secretary/44/1/grades');
    $response->assertStatus(200);
}
```

Fig. 7: Prueba unitaria #4 Obtener calificaciones de estudiante.

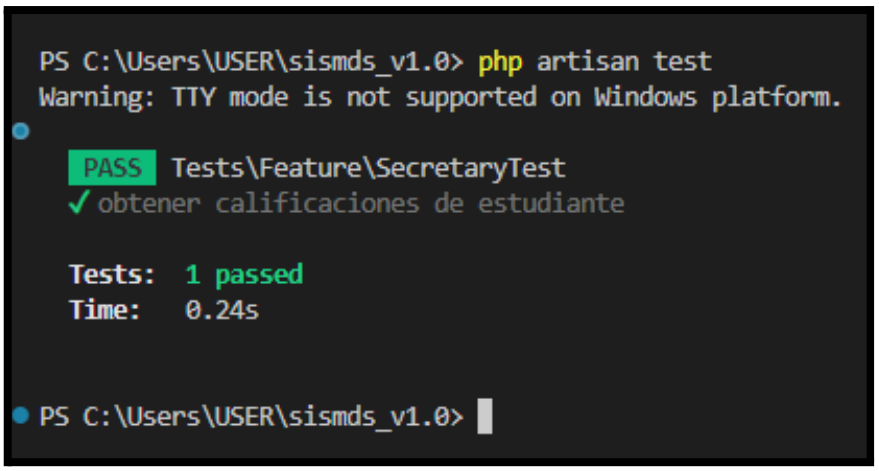

Fig. 8: Resultado prueba unitaria #4.

```php
// Prueba unitaria para asignar los comportamientos de un estudiante
public function test_asignar_comportamientos_a_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "comportamiento1" => "A",
        "comportamiento2" => "B"
    ];
    $response = $this->actingAs($user)->post('/api/secretary/44/finalStudentGrade', $data);
    $response->assertStatus(200);
}
```

Fig. 9: Prueba unitaria #5 Asignar comportamientos a estudiante.

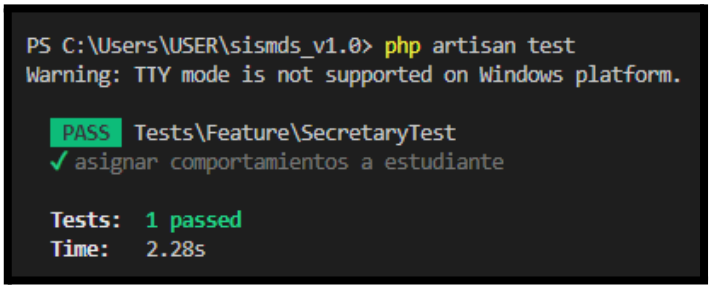

Fig. 10: Resultado prueba unitaria #5.

```php
// Prueba unitaria para finalizar el periodo académico vigente
public function test_terminar_periodo_academico()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->post('/api/secretary/endPeriod');
    $response->assertStatus(200);
}
```

Fig. 11: Prueba unitaria #6 Finalizar periodo académico vigente.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ terminar periodo academico

  Tests:  1 passed
  Time:   0.21s
```

Fig. 12: Resultado prueba unitaria #6.

```php
// Prueba unitaria para generar el reporte de califaciones para la secretaria
public function test_generar_reporte_de_calificaciones_de_estudiantes()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/secretary/1/1/1/1/report');
    $response->assertStatus(200);
}
```

Fig. 13: Prueba unitaria #7 Generar reporte de calificaciones de estudiantes.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ generar reporte de calificaciones de estudiantes

  Tests:  1 passed
  Time:   0.18s
```

Fig. 14: Resultado prueba unitaria #7.

```php
// Prueba unitaria para actualizar la información del colegio
public function test_actualizar_informacion_del_colegio()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "A",
        "director_name" => "B",
        "secretary_name" => "B"
    ];
    $response = $this->actingAs($user)->post('/api/information/update', $data);
    $response->assertStatus(200);
}
```

Fig. 15: Prueba unitaria #8 Actualizar información del colegio.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar informacion del colegio

  Tests:  1 passed
  Time:   0.17s
```

Fig. 16: Resultado prueba unitaria #8.

```php
// Prueba unitaria para crear un usuario con rol de estudiante
public function test_crear_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Alexa Jimena",
        "last_name" => "Donnelly Schwalter",
        "personal_phone" => "0986352654",
        "home_phone" => "021280307",
        "address" => "4549 Quito",
        "email" => "juanca@gmail.com",
        "identification" => "1752563258",
        "birthday" => "1995-05-09",
        "course_id" => 1,
        "parallel_id" => 1,
        "academic_period_id" => 1,
        "specialty_id" => 1,
        "representative_name" => "Sarah Blanda",
        "representative_last_name" => "Gislason Boada",
        "representative_identification" => "0703852625",
        "representative_personal_phone" => "0985636524"
    ];
    $response = $this->actingAs($user)->post('/api/student/create', $data);
    $response->assertStatus(200);
}
```

Fig. 17: Prueba unitaria #9 Crear estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear estudiante

  Tests:  1 passed
  Time:   5.78s
```

Fig. 18: Resultado prueba unitaria #9.

```php
// Prueba unitaria para obtener los datos de un estudiante
public function test_obtener_datos_de_un_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/student/144');
    $response->assertStatus(200);
}
```

Fig. 19: Prueba unitaria #10 Obtener datos de un estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ obtener datos de un estudiante

  Tests:  1 passed
  Time:   0.19s
```

Fig. 20: Resultado prueba unitaria #10.

```php
// Prueba unitaria para actualizar un usuario con rol de estudiante
public function test_actualizar_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Alexa Jimena",
        "last_name" => "Donnelly Schwalter",
        "personal_phone" => "0986352654",
        "home_phone" => "021280307",
        "address" => "4549 Quito",
        "email" => "juanca@gmail.com",
        "identification" => "1752563258",
        "birthday" => "1995-05-09",
        "course_id" => 1,
        "parallel_id" => 1,
        "academic_period_id" => 1,
        "specialty_id" => 1,
        "representative_name" => "Sarah Blanda",
        "representative_last_name" => "Gislason Boada",
        "representative_identification" => "0703852625",
        "representative_personal_phone" => "0985636524"
    ];
    $response = $this->actingAs($user)->post('/api/student/144/update', $data);
    $response->assertStatus(200);
}
```

Fig. 21: Prueba unitaria #11 Actualizar estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar estudiante

  Tests:  1 passed
  Time:   0.27s
```

Fig. 22: Resultado prueba unitaria #11.

```php
// Prueba unitaria para buscar un estudiante
public function test_buscar_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "identification" => "1752563258"
    ];
    $response = $this->actingAs($user)->post('/api/student/search', $data);
    $response->assertStatus(200);
}
```

Fig. 23: Prueba unitaria #12 Buscar estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ buscar estudiante

  Tests:  1 passed
  Time:   0.18s
```

Fig. 24: Resultado prueba unitaria #12.

```php
// Prueba unitaria para crear un usuario con rol profesor
public function test_crear_profesor()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Alexa Jimena",
        "last_name" => "Donnelly Schwalter",
        "identification" => "1568526359",
        "birthday" => "1995-05-09",
        "address" => "4549 Quito",
        "home_phone" => "021280307",
        "personal_phone" => "0986352654",
        "email" => "alexajmn@gmail.com",
    ];
    $response = $this->actingAs($user)->post('/api/teacher/create', $data);
    $response->assertStatus(200);
}
```

Fig. 25: Prueba unitaria #13 Crear profesor.

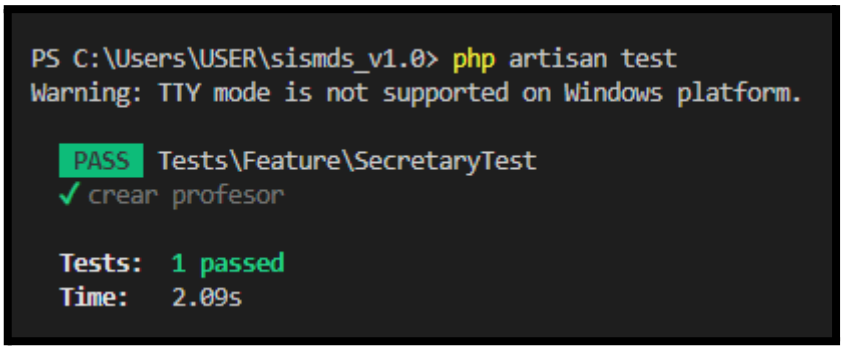

Fig. 26: Resultado prueba unitaria #13.

```php
// Prueba unitaria para obtener los datos de un profesor
public function test_obtener_datos_de_un_profesor()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/teacher/4');
    $response->assertStatus(200);
}
```

Fig. 27: Prueba unitaria #14 Obtener datos de un profesor.

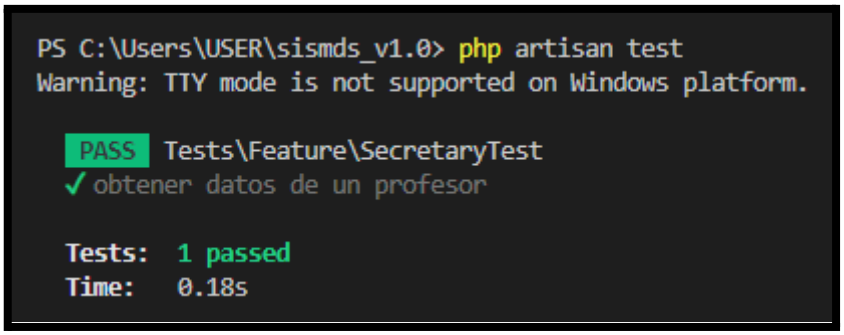

Fig. 28: Resultado prueba unitaria #14.

```php
// Prueba unitaria para actualizar un usuario con rol profesor
public function test_actualizar_profesor()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Alexa Jimena",
        "last_name" => "Donnelly Schwalter",
        "identification" => "1652569856",
        "birthday" => "1995-05-09",
        "address" => "4549 Quito",
        "home_phone" => "021280307",
        "personal_phone" => "0986352654",
        "email" => "alexatestn@gmail.com",
    ];
    $response = $this->actingAs($user)->post('/api/teacher/4/update', $data);
    $response->assertStatus(200);
}
```

Fig. 29: Prueba unitaria #15 Actualizar profesor.

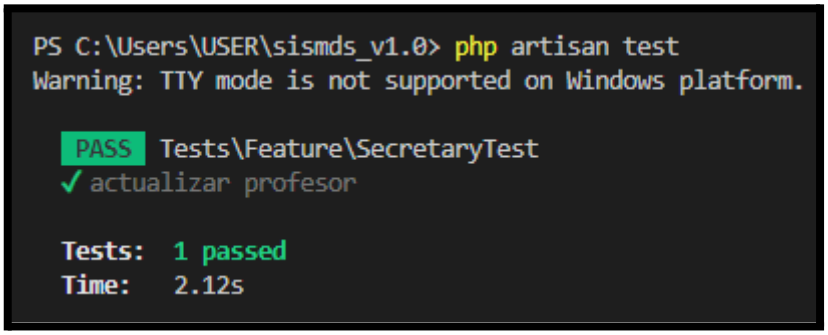

Fig. 30: Resultado prueba unitaria #15.

```php
// Prueba unitaria para buscar un profesor
public function test_buscar_profesor()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "identification" => "1652569856"
    ];
    $response = $this->actingAs($user)->post('/api/teacher/search', $data);
    $response->assertStatus(200);
}
```

Fig. 31: Prueba unitaria #16 Buscar profesor.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ buscar profesor

  Tests:  1 passed
  Time:   0.18s
```

Fig. 32: Resultado prueba unitaria #16.

```php
// Prueba unitaria para activar/desactivar un usuario con rol secretario/a
public function test_activar_y_desactivar_secretaria()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/secretary/2/destroy');
    $response->assertStatus(200);
}
```

Fig. 33: Prueba unitaria #17 Activar y desactivar secretaria.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar secretaria

  Tests:  1 passed
  Time:   0.17s
```

Fig. 34: Resultado prueba unitaria #17.

```php
// Prueba unitaria para activar/desactivar un usuario con rol profesor/a
public function test_activar_y_desactivar_profesor()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/teacher/5/destroy');
    $response->assertStatus(200);
}
```

Fig. 35: Prueba unitaria #18 Activar y desactivar profesor.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar profesor

  Tests:  1 passed
  Time:   0.17s
```

Fig. 36: Resultado prueba unitaria #18.

```php
// Prueba unitaria para activar/desactivar un usuario con rol estudiante
public function test_activar_y_desactivar_estudiante()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/student/45/destroy');
    $response->assertStatus(200);
}
```

Fig. 37: Prueba unitaria #19 Activar y desactivar estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar estudiante

  Tests:  1 passed
  Time:   0.16s
```

Fig. 38: Resultado prueba unitaria #19.

```php
// Prueba unitaria para visualizar un periodo académico
public function test_visualizar_periodo_academico()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/period/1');
    $response->assertStatus(200);
}
```

Fig. 39: Prueba unitaria #20 Visualizar periodo académico.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ visualizar periodo academico

  Tests:  1 passed
  Time:   0.18s
```

Fig. 40: Resultado prueba unitaria #20.

```php
// Prueba unitaria para crear un periodo académico
public function test_crear_periodo_academico()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "2030",
        "finq1" => null,
        "finq2" => null
    ];
    $response = $this->actingAs($user)->post('/api/period/create', $data);
    $response->assertStatus(200);
}
```

Fig. 41: Prueba unitaria #21 Crear periodo académico.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear periodo academico

  Tests:  1 passed
  Time:   0.18s
```

Fig. 42: Resultado prueba unitaria #21.

```php
// Prueba unitaria para actualizar un periodo académico
public function test_actualizar_periodo_academico()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "2023",
        "finq1" => null,
        "finq2" => null
    ];
    $response = $this->actingAs($user)->post('/api/period/1/update', $data);
    $response->assertStatus(200);
}
```

Fig. 43: Prueba unitaria #22 Actualizar periodo académico.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar periodo academico

  Tests:  1 passed
  Time:   0.18s
```

Fig. 44: Resultado prueba unitaria #22.

```php
// Prueba unitaria para deshabilitar/habilitar un periodo académico
public function test_activar_y_desactivar_periodo_academico()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/period/1/destroy');
    $response->assertStatus(200);
}
```

Fig. 45: Prueba unitaria #23 Activar y desactivar periodo académico.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar periodo academico

  Tests:  1 passed
  Time:   0.17s
```

Fig. 46: Resultado prueba unitaria #23.

```php
// Prueba unitaria para visualizar un curso
public function test_visualizar_curso()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/course/1');
    $response->assertStatus(200);
}
```

Fig. 47: Prueba unitaria #24 Visualizar curso.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ visualizar curso

  Tests:  1 passed
  Time:   0.19s
```

Fig. 48: Resultado prueba unitaria #24.

```php
// Prueba unitaria para crear un curso
public function test_crear_curso()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "NUEVO CURSO"
    ];
    $response = $this->actingAs($user)->post('/api/course/create', $data);
    $response->assertStatus(200);
}
```

Fig. 49: Prueba unitaria #25 Crear curso.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear curso

  Tests:  1 passed
  Time:   0.18s
```

Fig. 50: Resultado prueba unitaria #25.

```php
// Prueba unitaria para actualizar un curso
public function test_actualizar_curso()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "ACT CURSO"
    ];
    $response = $this->actingAs($user)->post('/api/course/7/update', $data);
    $response->assertStatus(200);
}
```

Fig. 51: Prueba unitaria #26 Actualizar curso.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar curso

  Tests:  1 passed
  Time:   0.18s
```

Fig. 52: Resultado prueba unitaria #26.

```php
// Prueba unitaria para deshabilitar/habilitar un curso
public function test_activar_y_desactivar_curso()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/course/7/destroy');
    $response->assertStatus(200);
}
```

Fig. 53: Prueba unitaria #27 Activar y desactivar curso.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar curso

  Tests:  1 passed
  Time:   0.17s
```

Fig. 54: Resultado prueba unitaria #27.

```php
// Prueba unitaria para visualizar un paralelo
public function test_visualizar_paralelo()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/parallel/1');
    $response->assertStatus(200);
}
```

Fig. 55: Prueba unitaria #28 Visualizar paralelo.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ visualizar paralelo

  Tests:  1 passed
  Time:   0.20s
```

Fig. 56: Resultado prueba unitaria #28.

```php
// Prueba unitaria para crear un paralelo
public function test_crear_paralelo()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Z2"
    ];
    $response = $this->actingAs($user)->post('/api/parallel/create', $data);
    $response->assertStatus(200);
}
```

Fig. 57: Prueba unitaria #29 Crear paralelo.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear paralelo

  Tests:  1 passed
  Time:   0.18s
```

Fig. 58: Resultado prueba unitaria #29.

```php
// Prueba unitaria para actualizar un paralelo
public function test_actualizar_paralelo()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "Z3"
    ];
    $response = $this->actingAs($user)->post('/api/parallel/22/update', $data);
    $response->assertStatus(200);
}
```

Fig. 59: Prueba unitaria #30 Actualizar paralelo.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar paralelo

  Tests:  1 passed
  Time:   0.23s
```

Fig. 60: Resultado prueba unitaria #30.

```php
// Prueba unitaria para deshabilitar/habilitar un paralelo
public function test_activar_y_desactivar_paralelo()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/parallel/22/destroy');
    $response->assertStatus(200);
}
```

Fig. 61: Prueba unitaria #31 Activar y desactivar paralelo.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar paralelo

  Tests:  1 passed
  Time:   0.16s
```

Fig. 62: Resultado prueba unitaria #31.

```php
// Prueba unitaria para visualizar una epecialidad
public function test_visualizar_especialidad()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/specialty/1');
    $response->assertStatus(200);
}
```

Fig. 63: Prueba unitaria #32 visualizar especialidad.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ visualizar especialidad

  Tests:  1 passed
  Time:   0.19s
```

Fig. 64: Resultado prueba unitaria #32.

```php
// Prueba unitaria para crear una especialidad
public function test_crear_especialidad()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "NUEVA ESPECIALIDAD"
    ];
    $response = $this->actingAs($user)->post('/api/specialty/create', $data);
    $response->assertStatus(200);
}
```

Fig. 65: Prueba unitaria #33 Crear especialidad.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear especialidad

  Tests:  1 passed
  Time:   0.18s
```

Fig. 66: Resultado prueba unitaria #33.

```php
// Prueba unitaria para actualizar una especialidad
public function test_actualizar_especialidad()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "name" => "NEUVA ESPECIALIDAD"
    ];
    $response = $this->actingAs($user)->post('/api/specialty/6/update', $data);
    $response->assertStatus(200);
}
```

Fig. 67: Prueba unitaria #34 Actualizar especialidad.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar especialidad

  Tests:  1 passed
  Time:   0.18s
```

Fig. 68: Resultado prueba unitaria #34.

```php
// Prueba unitaria para deshabilitar/habilitar una especialidad
public function test_activar_y_desactivar_especialidad()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/specialty/6/destroy');
    $response->assertStatus(200);
}
```

Fig. 69: Prueba unitaria #35 Activar y desactivar especialidad.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar especialidad

  Tests:  1 passed
  Time:   0.16s
```

Fig. 70: Resultado prueba unitaria #35.

```php
// Prueba unitaria para visualizar una asignatura
public function test_visualizar_asignatura()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/subject/1');
    $response->assertStatus(200);
}
```

Fig. 71: Prueba unitaria #36 Visualizar asignatura.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ visualizar asignatura

  Tests:  1 passed
  Time:   0.20s
```

Fig. 72: Resultado prueba unitaria #36.

```php
// Prueba unitaria para crear una asignatura
public function test_crear_asignatura()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "course_id" => 1,
        "parallel_id" => 1,
        "academic_period_id" => 1,
        "specialty_id" => 1,
        "user_id" => 20,
        "name" => "NUEVA MATERIA"
    ];
    $response = $this->actingAs($user)->post('/api/subject/create', $data);
    $response->assertStatus(200);
}
```

Fig. 73: Prueba unitaria #37 Crear asignatura.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ crear asignatura

  Tests:  1 passed
  Time:   0.19s
```

Fig. 74: Resultado prueba unitaria #37.

```php
// Prueba unitaria para actualizar una asignatura
public function test_actualizar_asignatura()
{
    $user = User::where('role_id', 1)->first();
    $data = [
        "course_id" => 1,
        "parallel_id" => 1,
        "academic_period_id" => 1,
        "specialty_id" => 1,
        "user_id" => 20,
        "name" => "MATERIA MODIFICADA"
    ];
    $response = $this->actingAs($user)->post('/api/subject/11/update', $data);
    $response->assertStatus(200);
}
```

Fig. 75: Prueba unitaria #38 Actualizar asignatura.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ actualizar asignatura

  Tests:  1 passed
  Time:   0.18s
```

Fig. 76: Resultado prueba unitaria #38.

```php
// Prueba unitaria para deshabilitar/habilitar una asignatura
public function test_activar_y_desactivar_asignatura()
{
    $user = User::where('role_id', 1)->first();
    $response = $this->actingAs($user)->get('/api/subject/11/destroy');
    $response->assertStatus(200);
}
```

Fig. 77: Prueba unitaria #39 Activar y desactivar asignatura.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\SecretaryTest
  ✓ activar y desactivar asignatura

  Tests:  1 passed
  Time:   2.14s
```

Fig. 78: Resultado prueba unitaria #39.

```php
// Prueba unitaria para mostrar la información del perfil del profesor
public function test_mostrar_perfil_profesor()
{
    $user = User::where('role_id', 2)->first();
    $response = $this->actingAs($user)->get('/api/profile');
    $response->assertStatus(200);
}
```

Fig. 79: Prueba unitaria #40 Mostrar el perfil del profesor.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\TeacherTest
  ✓ mostrar perfil profesor

  Tests:  1 passed
  Time:   0.28s
```

Fig. 80: Resultado prueba unitaria #40.

```php
// Prueba unitaria para actualizar el perfil como un usuario profesor
public function test_actualizar_perfil_profesor()
{
    $user = User::where('role_id', 2)->first();
    $data = [
        "personal_phone" => "0986352654",
        "home_phone" => "021280307",
        "address" => "4549 Quito",
        "email" => "test12@gmail.com"
    ];
    $response = $this->actingAs($user)->post('/api/profile', $data);
    $response->assertStatus(200);
}
```

Fig. 81: Prueba unitaria #41 Actualizar el perfil del profesor.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\TeacherTest
  ✓ actualizar perfil profesor

  Tests:  1 passed
  Time:   0.36s
```

Fig. 82: Resultado prueba unitaria #41.

```php
// Prueba unitaria para obtener las materias asignadas a un profesor
public function test_mostrar_materias_asignadas()
{
    $user = User::where('role_id', 2)->first();
    $response = $this->actingAs($user)->post('/api/teacher/mySubjects');
    $response->assertStatus(200);
}
```

Fig. 83: Prueba unitaria #42 Obtener materias asignadas a un profesor.

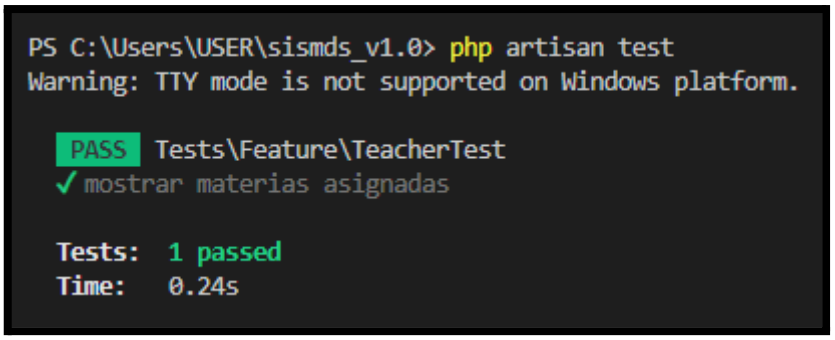

Fig. 84: Resultado prueba unitaria #42.

```php
// Prueba unitaria para obtener la lista de estudiantes asignados a una materia
public function test_mostrar_estudiantes_asignados_a_materia()
{
    $user = User::where('role_id', 2)->first();
    $response = $this->actingAs($user)->post('/api/teacher/1/studentsList');
    $response->assertStatus(200);
}
```

Fig. 85: Prueba unitaria #43 Mostrar lista de estudiantes asignados a una materia.

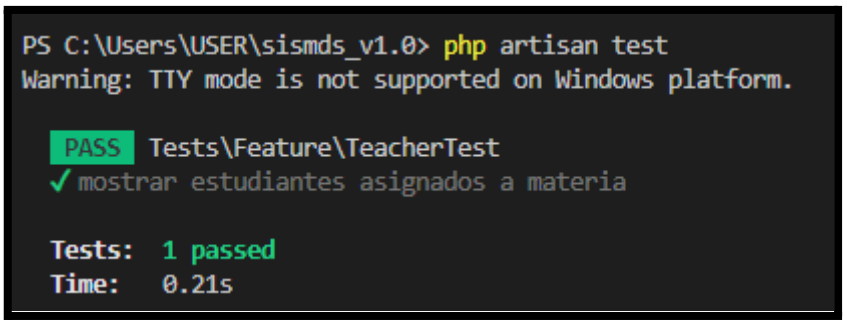

Fig. 86: Resultado prueba unitaria #43.

```php
// Prueba unitaria para obtener las calificaciones de un estudiante
public function test_obtener_notas_de_estudiante()
{
    $user = User::where('role_id', 2)->first();
    $response = $this->actingAs($user)->get('/api/teacher/44/1/grades');
    $response->assertStatus(200);
}
```

Fig. 87: Prueba unitaria #44 Obtener las calificaciones de un estudiante como profesor.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\TeacherTest
  ✓ obtener notas de estudiante

  Tests:  1 passed
  Time:   0.21s
```

Fig. 88: Resultado prueba unitaria #44.

```php
// Prueba unitaria para actualizar las calificaciones de un estudiante
public function test_actualizar_calificaciones_de_estudiante()
{
    $user = User::where('role_id', 2)->first();
    $data = [
        "p1q1" => 10,
        "p2q1" => 10,
        "p3q1" => 9,
        "p1q2" => 10,
        "p2q2" => 10,
        "p3q2" => 8,
        "supletorio" => null,
        "remedial" => null,
        "gracia" => null
    ];
    $response = $this->actingAs($user)->post('/api/teacher/44/1/updateGrade', $data);
    $response->assertStatus(200);
}
```

Fig. 89: Prueba unitaria #45 Actualizar las calificaciones de un estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\TeacherTest
  ✓ actualizar calificaciones de estudiante

  Tests:  1 passed
  Time:   0.20s
```

Fig. 90: Resultado prueba unitaria #45.

```php
// Prueba unitaria para obtener el reporte de calificaciones por materia
public function test_generar_reporte_por_materia()
{
    $user = User::where('role_id', 2)->first();
    $response = $this->actingAs($user)->get('/api/teacher/1/report');
    $response->assertStatus(200);
}
```

Fig. 91: Prueba unitaria #46 Obtener reporte de calificaciones por materia.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\TeacherTest
  ✓ generar reporte por materia

  Tests:  1 passed
  Time:   0.17s
```

Fig. 92: Resultado prueba unitaria #46.

```php
// Prueba unitaria para mostrar la información del perfil del estudiante
public function test_mostrar_perfil_estudiante()
{
    $user = User::where('role_id', 3)->first();
    $response = $this->actingAs($user)->get('/api/profile');
    $response->assertStatus(200);
}
```

Fig. 93: Prueba unitaria #47 Mostrar perfil del estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\StudentTest
  ✓ mostrar perfil estudiante

  Tests:  1 passed
  Time:   0.15s
```

Fig. 94: Resultado prueba unitaria #47.

```php
// Prueba unitaria para mostrar los cursos que tiene el estudiante
public function test_mostrar_cursos_estudiante()
{
    $user = User::where('role_id', 3)->first();
    $response = $this->actingAs($user)->post('/api/student/myCourses');
    $response->assertStatus(200);
}
```

Fig. 95: Prueba unitaria #48 Mostrar cursos que tiene el estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\StudentTest
  ✓ mostrar cursos estudiante

  Tests:  1 passed
  Time:   0.17s
```

Fig. 96: Resultado prueba unitaria #48.

```php
// Prueba unitaria para que el estudiante pueda visualizar sus calificaciones
public function test_visualizar_calificaciones_estudiante()
{
    $user = User::where('role_id', 3)->first();
    $response = $this->actingAs($user)->get('/api/student/1/grades');
    $response->assertStatus(200);
}
```

Fig. 97: Prueba unitaria #49 Visualizar calificaciones para estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\StudentTest
  ✓ visualizar calificaciones estudiante

  Tests:  1 passed
  Time:   0.15s
```

Fig. 98: Resultado prueba unitaria #49.

```php
// Prueba unitaria para que el estudiante pueda generar su reporte de calificaciones
public function test_generar_reporte_de_calificaciones_estudiante()
{
    $user = User::where('role_id', 3)->first();
    $response = $this->actingAs($user)->get('/api/student/1/report');
    $response->assertStatus(200);
}
```

Fig. 99: Prueba unitaria #50 Generar reporte de calificaciones para estudiante.

```
PS C:\Users\USER\sismds_v1.0> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Feature\StudentTest
  ✓ generar reporte de calificaciones estudiante

  Tests:  1 passed
  Time:   0.16s
```

Fig. 100: Resultado prueba unitaria #50.

Prueba de carga

Para la presente sección se presenta la funcionalidad del consumo de los *endpoints* desarrollados sometidos a una prueba de carga mediante un servicio de peticiones HTTPS como lo es *Apache JMeter,* verificando su correcto funcionamiento, cabe destacar que cada ruta se somete a 40 usuarios en 1 segundo, dando como resultado en ingreso total de 2040 peticiones para las 51 rutas disponibles. A continuación, desde la **Fig. 101** hasta la **Fig. 106** se presentan los resultados obtenidos de los *endpoints*.

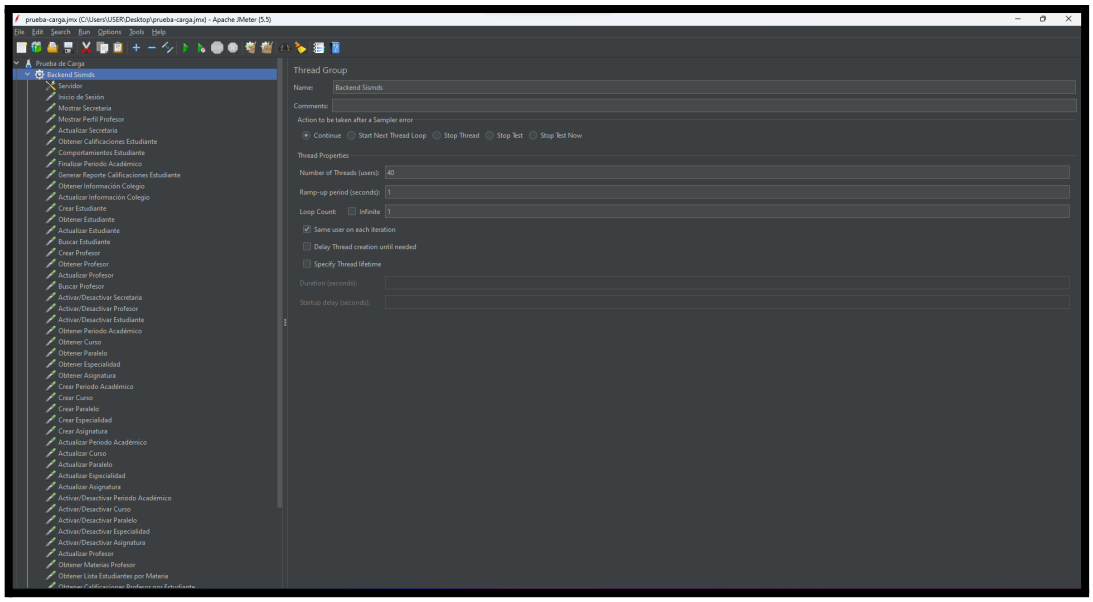

Fig. 101: Configuración del número de peticiones a ejecutar.

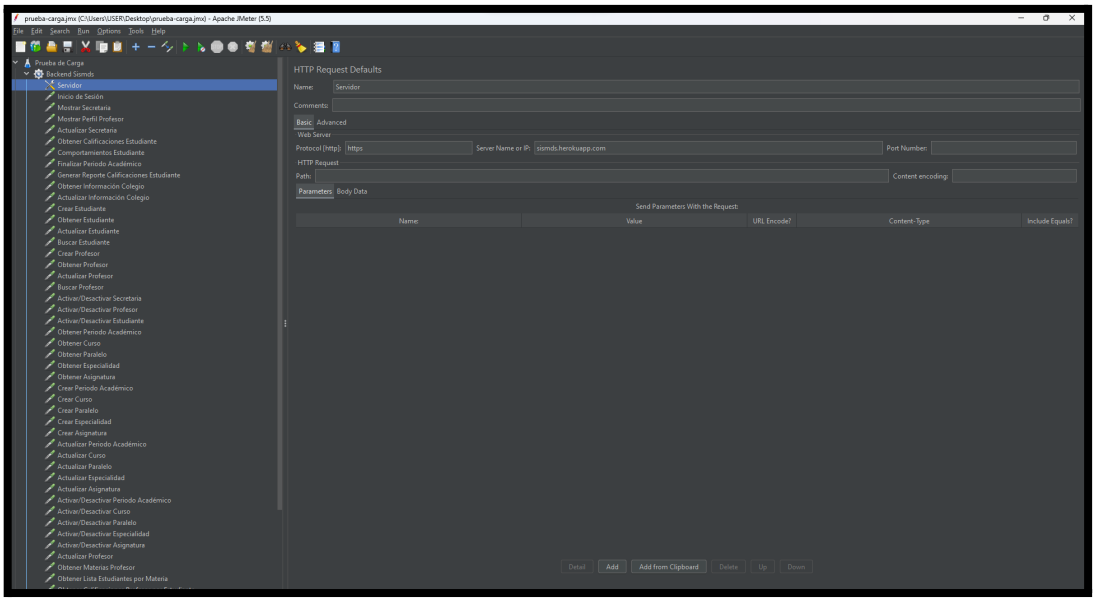

Fig. 102: Configuración de la aplicación desplegada.

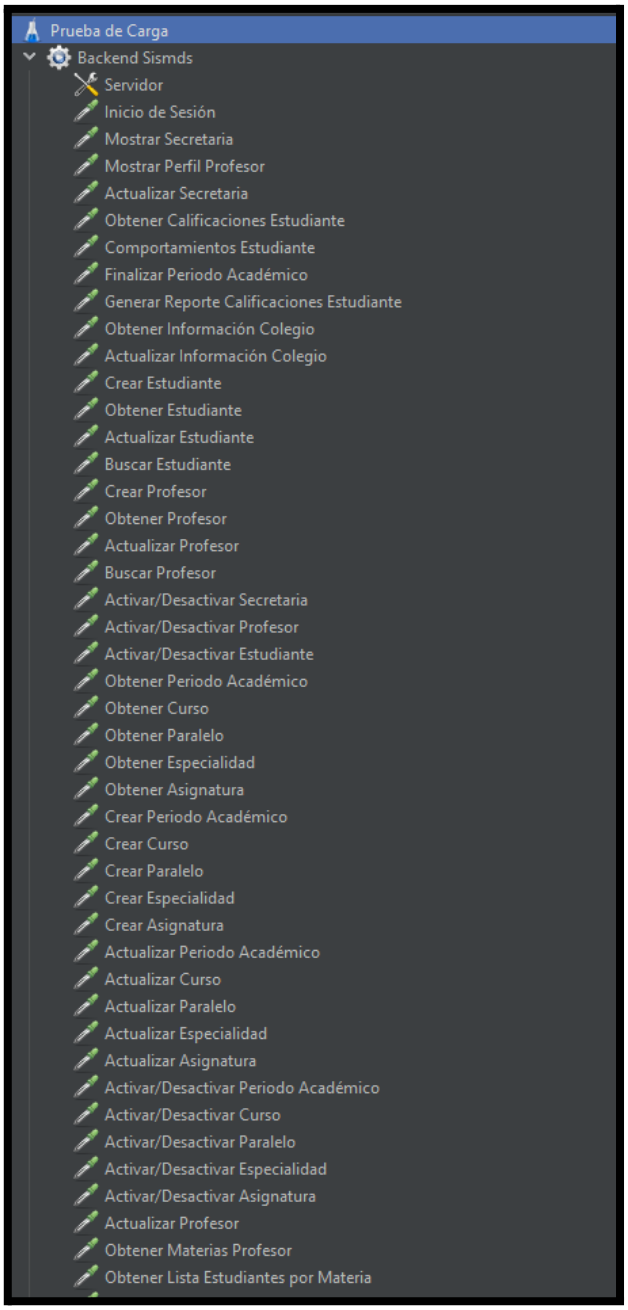

Fig. 103: Configuración de rutas primera parte.

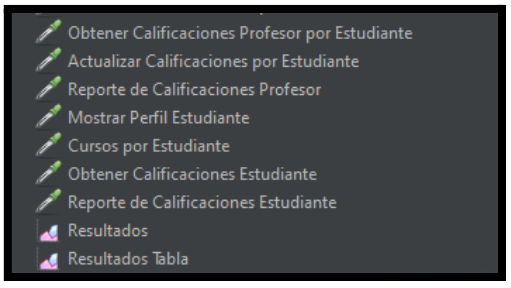

Fig. 104: Configuración de rutas segunda parte.

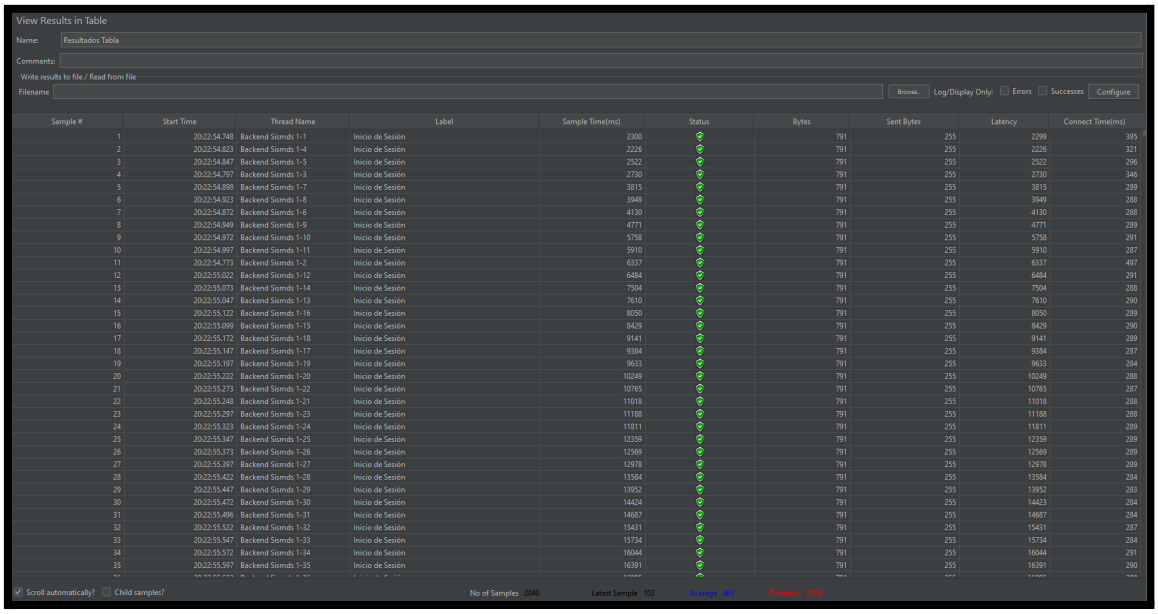

Fig. 105: Resultado de la prueba de carga inicio.

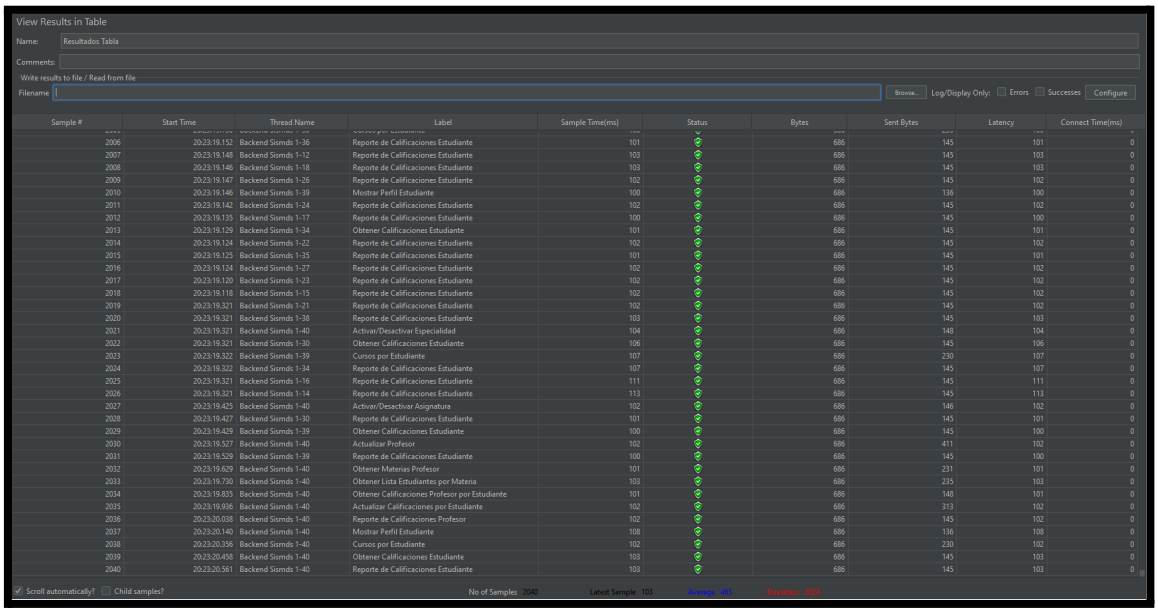

Fig. 106: Resultado la prueba de carga final.

Prueba de estrés

Para la presente sección se presenta la funcionalidad del consumo de los *endpoints* desarrollados sometidos a una prueba de estrés mediante un servicio de peticiones HTTPS como lo es *Apache JMeter,* verificando su correcto funcionamiento, cabe destacar que cada ruta se somete a 80 usuarios en 1 segundo, dando como resultado en ingreso total de 4080 peticiones para las 51 rutas disponibles, lo cual es el doble de peticiones que se ingresarían normalmente. A continuación, desde la **Fig. 107** hasta la **Fig. 109** se presentan los resultados obtenidos de los *endpoints*.

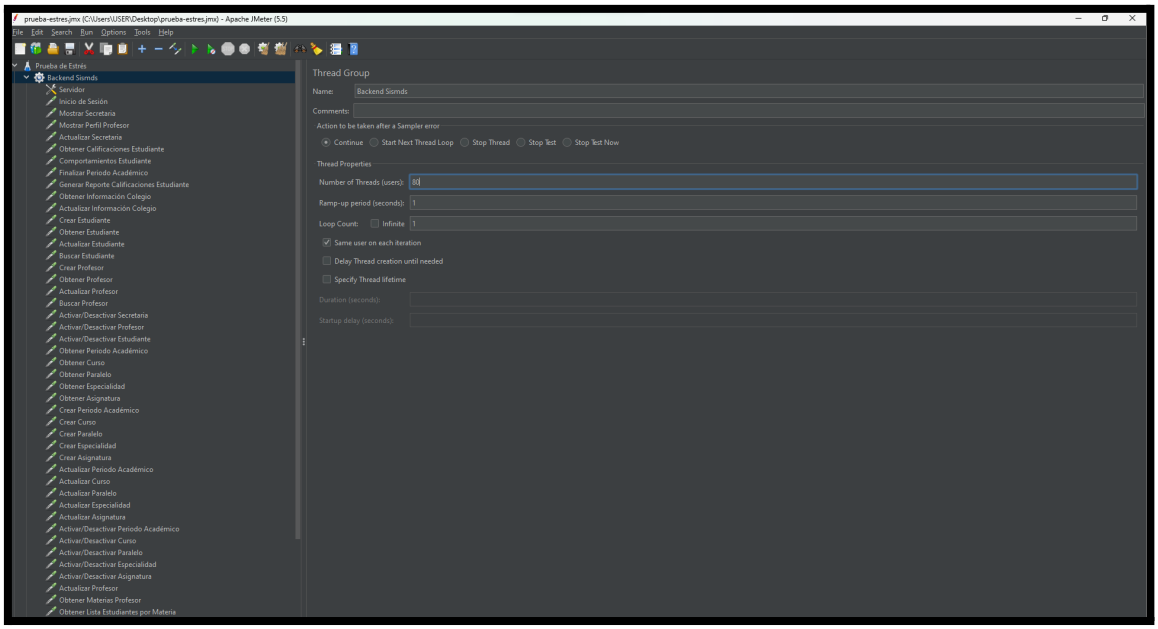

Fig. 107: Configuración del número de peticiones a ejecutar para la prueba de estrés.

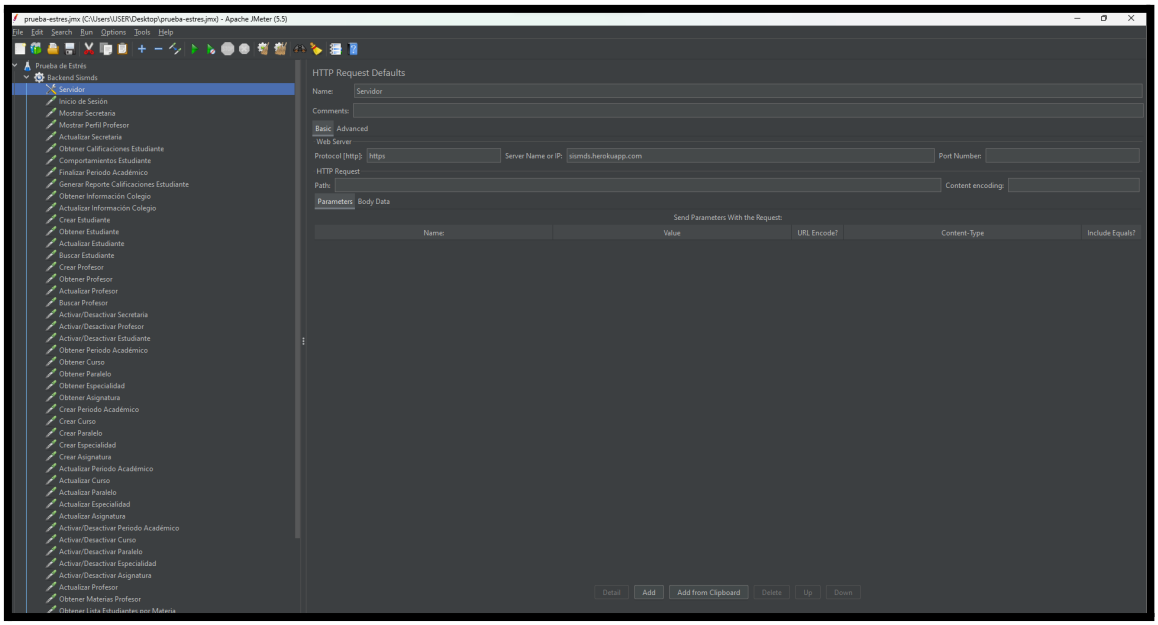

Fig. 108: Configuración de la aplicación desplegada para la prueba de estrés.

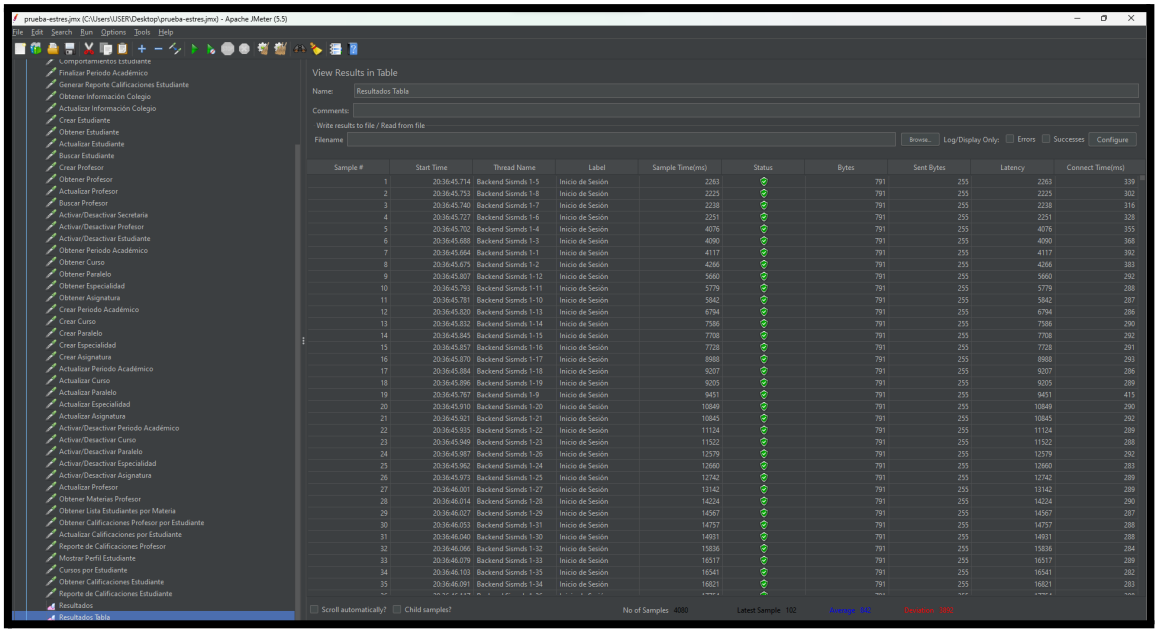

Fig. 109: Resultado de la prueba de estrés.

ANEXO III

El siguiente enlace, permite observar el Manual de Usuario, donde esta detallada la información con respecto a todas las funcionalidades de los *endpoints,* así como la participación de los roles en la interacción.

https://youtu.be/bHx427G7ZhE

ANEXO IV

A continuación, se especifica las credenciales de acceso para los *endpoints* por roles de usuario, así como el enlace al repositorio en *GitHub* en donde se encuentra el código fuente.

Credenciales de acceso para sistema *web* y *endpoints*

Para acceder a los *endpoints* desplegados en producción, ingresar a la siguiente URL:

https://cuddly-art-655.notion.site/APIs-c513db2e607f4b81bb6c3c2444bf76d4

Credenciales de acceso para un usuario con rol secretario/a del sistema *web*:

- **Identificación:** 11111
- **Contraseña:** sismds

Credenciales de acceso para un usuario con rol profesor/a del sistema *web*:

- **Identificación:** 44444
- **Contraseña:** sismds

Credenciales de acceso para un usuario con rol estudiante del sistema *web*:

- **Identificación:** 4444444444
- **Contraseña:** sismds

Repositorio del código fuente del sistema de escritorio y *endpoints*

El código fuente de todo el proyecto, se encuentra alojado en el repositorio *GitHub* y se puede acceder a través de la siguiente URL:

https://github.com/raul-tenorio/sismds_v1.0.git