# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## DESARROLLO DE UN SISTEMA WEB QUE SIRVA COMO GUÍA DE ESTUDIO PARA ESTUDIANTES DE LA ESFOT.

## DESARROLLO DE UN *BACKEND*

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

**ALEXANDER FERNANDO TUPIZA CARRERA**

**DIRECTOR: MALDONADO SOLIZ IVONNE FERNANDA**

**DMQ, marzo 2023**

# CERTIFICACIONES

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


**ALEXANDER FERNANDO TUPIZA CARRERA**

**alexander.tupiza@epn.edu.ec**

**fernandotupiza10@hotmail.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por ALEXANDER FERNANDO TUPIZA CARRERA, bajo mi supervisión.

**Ing. Ivonne Fernanda Maldonado Soliz**

**DIRECTOR**

**ivonne.maldonadof@epn.edu.ec**

# DECLARACIÓN DE AUTORÍA

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


ALEXANDER FERNANDO TUPIZA CARRERA

# DEDICATORIA

Este trabajo está dedicado a Dios, por la salud, vida y fortaleza que me brindo para seguir con mi carrera profesional y principalmente dedico este trabajado a mis padre y hermano que estuvieron en mis derrotas apoyándome a seguir con lo que más amo y segundo a mis profesores que inculcaron grandes conocimientos, apoyándome en inquietudes adquiridas durante cada materia brindada.

**Alexander**

# AGRADECIMIENTO

Agradezco a Dios y a mis padres por el entendimiento que me brindaron durante este duro camino para adquirir mi carrera tan anhelada, dado que mis padres son el pilar fundamental de mi educación tanto de valores como de conocimiento, puesto que gracias al apoyo logré ingresar a la Universidad motivándome a seguir en lo que más amo, Así mismo, un agradecimiento sincero a mis compañeros de curso, dado que logre llevarme con todos compartiendo risas, momentos y estudios, los cuales me han permitido seguir día a día con las mismas ganas de seguir estudiando, puesto que son grandes personas de ayuda y apoyo en momentos difíciles. Finalmente agradezco de corazón a mis padres y hermano que me vieron derramar lágrimas y aun así me lograron entender y apoyar para que siga adelante diciéndome “Tu si puedes solo es cuestión de dedicarte y esforzarte”

Agradezco a los profesores, en especial a la Ing. Ivonne Maldonado, quien fue la persona que estuvo ahí apoyándome y guiándome en el desarrollo de mi proyecto de titulación sin importar el tiempo en el que le escriba, siempre apoyándome, brindándome un poquito de tiempo para solventar ciertas dudas. Finalmente agradezco a todos mis Ingenieros que fueron mi pilar fundamental de los conocimientos y valores.

**Alexander**

# ÍNDICE DE CONTENIDO

# RESUMEN

La pandemia que se vivió hace poco demostró no solo la necesidad sino la importancia de la autoeducación, muchas instituciones tuvieron que migrar a nuevas tecnologías, sistemas, y plataformas para lograr la comunicación entre docentes y estudiantes, optando por las ventajas de la educación en línea que por detrás trae consigo un gran compromiso de autoeducación por parte tanto de estudiantes como de profesores. La autoeducación es un proceso que involucra la toma de conciencia sobre uno mismo, las necesidades y las luchas que cada uno pueda tener. Este proceso de educación en línea ha permitido detectar ciertos inconvenientes que las instituciones educativas deben cubrir para apoyar a la autoeducación de los estudiantes, por medio del desarrollo de herramientas que se adapten a este tipo de necesidad.

Optando por la necesidad de recursos de estudio para toda la “Comunidad ESFOT”, se ha desarrollado un sistema web, específicamente el componente *backend*, que sirva como una guía de autoeducación de las materias conforme a la malla curricular de las diferentes carreras de tecnología que actualmente se ofertan; contando con material, información o documentación para un estudio previo o como refuerzo.

El desarrollo del componente *backend* ha seguido la metodología SCRUM, permitiendo que se lleve una estructura, flexibilidad en los cambios realizados y manteniendo una mayor calidad y productividad a lo largo del desarrollo. La verificación de la funcionalidad correcta del *backend* se la ha realizado por medio de *Thunder Client*, mientras que la documentación de la API por medio de Swagger.

El presente documento está conformado por 4 secciones: la sección 1 abarca la descripción del componentes, objetivos, alcance y marco teórico, la sección 2 está conformada por el uso de la metodología con sus diferentes artefactos, herramientas utilizadas y patrón arquitectónico, la sección 3 muestra los resultados obtenidos en cada *sprint* y finalmente la sección 4 y 5 presentan las conclusiones y recomendación que se han ido obteniendo a lo largo de este trabajo de titulación.

**PALABRAS CLAVE:** *Backend*, SCRUM, Thunder Client, Sistema web, Swagger.

# ABSTRACT

The recent pandemic demonstrated not only the need but also the importance of self-education, many institutions had to migrate to new technologies, systems, and platforms to achieve communication between teachers and students, opting for the advantages of online education that brings with it a great commitment to self-education on the part of both students and teachers. Self-education is a process that involves becoming aware of oneself, one's needs and the struggles one may have. This process of online education has made it possible to detect certain drawbacks that educational institutions must cover in order to support students' self-education, by developing tools that adapt to this type of need.

Opting for the need of study resources for the entire "ESFOT Community", a web system has been developed, specifically the backend component, which serves as a guide for self-education of the subjects according to the curriculum of the different technology careers that are currently offered; having material, information or documentation for a previous study or as reinforcement.

The development of the backend component has followed the SCRUM methodology, allowing a structure, flexibility in the changes made and maintaining a higher quality and productivity throughout the development. The verification of the correct functionality of the backend has been done through Thunder Client, while the API documentation has been done through Swagger.

This document is made up of 4 sections: section 1 covers the description of the components, objectives, scope and theoretical framework, section 2 is made up of the use of the methodology with its different artifacts, tools used and architectural pattern, section 3 shows the results obtained in each sprint and finally section 4 and 5 presents the conclusions and recommendations that have been obtained throughout this degree work.

**KEYWORDS:** *Backend*, SCRUM, Thunder Client, Sistema web, Swagger.

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La cuarentena frente al resguardo por el COVID19 obligo a que las instituciones educativas hagan uso de herramientas tecnológicas y aprendizaje virtual improvisado, debido a que no se estaba preparado para una educación en tiempos de pandemia.

Las instituciones superiores tienen como obligación el estar preparadas para continuar con la impartición de clases aun en la presencia de casos fortuitos, de ahí el hecho que como medida se haya optado por la virtualidad, suponiendo no solo un reto para el quehacer docente sino para los múltiples desafíos como estudiantes, por el hecho de haber estado familiarizados únicamente con el modelo educativo tradicional de enseñanza-aprendizaje a través de clases magistrales; generando sentimientos de angustia, desconfianza e incertidumbre por el hecho de tener que migrar al modelo virtual [1].

La educación virtual abre todo tipo de posibilidades para el aprendizaje; adquirir conocimientos en todo tipo de áreas es uno de ellos, pero también requiere y exige mayores ingredientes de pedagogía y tecnología de especial disposición. Aquí, el docente debe ser un diseñador de ambientes de aprendizaje y el estudiante el conductor del autoaprendizaje [2]. Es decir, el estudiante tiene una mayor autonomía e independencia para el desarrollo del proceso de aprendizaje, marcando su ritmo de trabajo, sus objetivos y sus motivaciones intrínsecas [3].

Durante los dos últimos años se vivió el desafío de la educación virtual, dejando como resultado que hoy por hoy los estudiantes tengan conciencia de la importancia de la autoeducación y el autoaprendizaje. Entendiendo que es una forma potente para adquirir conocimientos sin depender de la guía de un profesor [4].

Este antecedente, ha permitido idear el presente proyecto de integración curricular en el cual se desarrolla un sistema web para la guía interactiva de los estudiantes de la ESFOT, para este caso en específico el componente *backend* probado desde una API REST. Este sistema web pretende ser una herramienta para la autoeducación al contar con información (material de clase, ejercicios resueltos, guías de laboratorio, deberes, talleres, etc.) de todas las materias, por semestre, de cada una de las carreras que actualmente ofrece la ESFOT.

El *backend* está basado en una API REST que se ha desarrollado con el lenguaje programación PHP, creando diferentes *endpoints* para que el usuario final pueda consumirlos, utilizando métodos de petición del protocolo HTTP. Alcanzando de esta manera la interacción de los estudiantes de la ESFOT por medio de documentos y

comentarios, con la visión principal de proveer material educativo para que los estudiantes de la ESFOT potencien la autoeducación y el autoaprendizaje.

## 1.1 Objetivo general

Desarrollar el *backend* de un sistema web de guía de estudio para estudiantes de la ESFOT.

## 1.2 Objetivos específicos

1. Levantar los requerimientos para el *backend* del sistema web.
2. Diseñar la base de datos SQL para el sistema web.
3. Implementar la arquitectura de *software* para el desarrollo del componente *backend.*
4. Comprobar el componente *backend*.
5. Desplegar el componente *backend* en un servidor web.

## 1.3 Alcance

Actualmente se tiene una fuente inagotable de nuevos conocimientos, por ello el desarrollar el lado autodidacta para ejercer el autoaprendizaje abre un mundo de posibilidades laborales o de emprendimiento [5]. La tecnología avanza a pasos agigantados; la facilidad con la que hoy se producen y distribuyen la información es abrumadora.

El sistema web que se desarrolla en el presente proyecto de integración curricular está destinado a todos quienes les interese auto educarse en las diferentes carreras que oferta la ESFOT. Para cumplir con ello se ha desarrollado un *backend* basado en la tecnología API REST, con el objetivo de establecer un sistema eficiente y con funcionalidades que benefician al usuario final.

En definitiva, el sistema web cuenta con recursos educativos (material de clase, ejercicios resueltos, guías de laboratorio, deberes, talleres, etc.) de las carreras la ESFOT, para potenciar el aprendizaje autónomo. La interacción con este contenido se basa en peticiones a la API REST, contando con los respectivos CRUD (Crear, Leer, Actualizar y Borrar) y tomando en cuenta restricciones según el perfil con el que se esté autenticado.

**El usuario con rol Administrador puede:**

- Autenticarse.
- Crear carreras y semestres.

- Crear materias y recursos educativos.
- Gestionar un CRUD en el apartado de preguntas e inquietudes.
- Actualizar perfil.
- Eliminar carreras, semestres, materias, recursos educativos y comentarios.

**El usuario con rol Estudiante puede:**

- Autenticarse.
- Registrarse.
- Visualizar las carreras y semestres ya creados en la primera vista.
- Visualizar materias y recursos educativos ya creados en la segunda vista.
- Aplicar un CRUD completo a la parte de preguntas e inquietudes.

## 1.4 Marco Teórico

La educación autónoma en la actualidad, desempeña un papel fundamental para cada uno de los estudiantes; en le educación superior ecuatoriana esto es clave para el éxito de la vida estudiantil debido a que el tiempo que tienen los docentes para la enseñanza es muy corto, obligando a que los estudiantes de manera independiente se acojan a la dinámica de un estudio autónomo, cubriendo ciertas inquietudes de nuevos temas a aprender, determinando un proceso educativo independiente [6].

### Metodología

En el campo del desarrollo de *software*, para crear un sistema exitoso se optar por ciertas normalizaciones que permitan que el desarrollo fluya de manera correcta. La metodología consta de procedimientos jerarquizados, haciendo uso de técnicas y métodos que permiten avanzar de manera satisfactoria para alcanzar los objetivos propuestos; permitiendo llevar al proyecto de manera mucho más estructurado [7]. En definitiva, la metodología es la ciencia amplia, concreta y eficiente, para el estudio que se quiere llevar a cabo, generando estrategias y estructuras para llevar la lógica en el entorno que se encuentre [8].

### Metodología ágil

El contenido de metodología ágil se basa en adecuarse al cliente, generando estrategias de trabajo para validar la rapidez y flexibilidad en cada tarea designada, siendo un entorno en el cual no se adelanta la planificación, sino que se avanza conforme a las

retroalimentaciones pertinentes. De ahí el hecho de que las metodologías agiles permitan llevar un desarrollo de *software* de manera estructurado, definiendo tareas y roles a las personas que conforman el equipo de trabajo para cumplir el tiempo propuesto, permitiendo que el usuario final conozca avances y posibilitando el cambio de manera oportuna [9].

### *Backend*

El *backend* se puede definir como la lógica interna de un sistema, la cual no es visible para el usuario final, en este caso es donde se gestiona todos los procesos y acciones requeridas por el propietario del desarrollo de *software*. Este componente permite ir generando funcionalidad de la lógica interna, la cual es un pilar fundamental para generar un interfaz para que el usuario final pueda acceder a su respetivo uso.

El desarrollo de *backend* es necesario, dado que es la parte fundamental en el ámbito de la funcionalidad, siendo el componente que primero accede a la base de datos y a la ejecución de código. El *backend* genera una comunicación entre base de datos, permite disminuir los procesos en el desarrollo por medio del uso de librerías y logra cumplir con las peticiones del usuario final [10].

### *API REST*

Se define como una estructura basada en el estándar HTTP para APIs que se logran intercomunicar entre sí, permitiendo acceder a la funcionalidad mediante identificados únicos (URI), logrando intercambiar la información entre el componente *backend* y *frontend*, permitiendo que un cliente pueda gestionar el uso de cierta lógica. Esta enlaza con reglas, las cuales deben ser cumplidas en el diseño de la arquitectura [11]. La API REST, mantiene su usabilidad con métodos para gestionar el proceso asignada a cada una de las APIs, en este caso los métodos sustentados al protocolo HTTP son los siguientes.

- *POST*: Función de crear.
- *PUT*: Modificar recurso que ya existe.
- *GET*: Consultar información.
- *DELETE*: Eliminar información.
- *PATCH*: Modificación de una sola información.

# 2 METODOLOGÍA

El implementar un nuevo *software* utilizando la guía de una metodología es clave, ya que se basa en un conjunto de métodos que se deben seguir para llevar una investigación eficiente y llegar a resultados esperados, estructurando los procedimientos a seguir para lograr el objetivo propuesto [12].

El estudio de caso se define como un estudio detallado sobre alguna temática propuestas, en diferentes campos como la educación, salud, tecnología y/o el campo empresarial; abarca métodos cualitativos y en ciertos aspectos métodos cuantitativos. El estudio de caso permite llevar un proyecto de manera centrada y manejable con la exploración de un tema, basado en peticiones que se necesita para satisfacer la investigación [13].

Este proyecto de integración curricular se basa en un estudio de caso, teniendo por un lado como problemática el hecho de que los estudiantes no siempre tienen a la mano recursos educativos o que en las herramientas como bibliotecas y aula virtual no se cuenta con información necesarias para tener una autoeducación completa, y por otro lado como solución se tiene el poder contar con un sitio de acceso a los recursos según la carrera y la materia que deseen fomentando el autoaprendizaje.

## 2.1 Metodología de Desarrollo

SCRUM es una de las metodologías ágiles que se destaca por la rapidez al crear proyectos, en lapsos de tiempos cortos establecido por cada sprint, basándose siempre en la lógica del negocio. Permite llevar el desarrollo de manera iterativa e incremental en función a la prioridad de asignación de acuerdo con la lógica del negoció [14].

### Roles

La metodología SCRUM permite llevar una buena organización dentro del proyecto, esto debido a que define de manera clara los roles de cada integrante del equipo, lo que genera que cada uno sepa las responsabilidades que tiene para el éxito del proyecto. Permitiendo un trabajo organizado, eficiente y sobre todo en grupo, logrando adaptarse a la función que cada uno desempeña [15] .

#### *Product Owner*

Tiene la función de la optimización del desarrollo de *software*, en ese caso es la persona propietaria del desarrollo, dado que debe verificar el equipo de trabajo para lograr un desempeño autenticado al proyecto. Es la persona responsable con el desarrollo final y la

encargada de preocuparse por la calidad que se le otorga al proyecto [16]. La **TABLA I** presenta la persona encargada para este rol.

***Scrum Master***

Es la persona que lidera al equipo, generando asignación de tareas y procesos de manera organizada a cada uno de los que componen el equipo. Desempeña la función de garantizar la calidad en cada una de las tareas que asigna velando por que los obstáculos que se den sean resueltos de manera rápida [17]. La **TABLA I** presenta la persona encargada para este rol.

***Development Team***

Son las personas que se adaptan a las tareas designadas cumpliendo con los avances en un lapso de tiempo determinado; tienen atributos colaborativos y de responsabilidad con su tarea para lograr el objetivo satisfaciendo al usuario final. La **TABLA I** presenta la persona encargada para este rol.

**TABLA I:** Asignación de Roles.

| ROLES | NOMBRES |
|---|---|
| *Product Owner* | Sr. Kerly Delgado. |
| *Scrum Master* | Ing. Ivonne Maldonado. |
| *Development Team* | Sr. Alexander Tupiza. |

## Artefactos

Los artefactos potencian la transferencia de información, haciendo conocer de la lógica y entendimiento a todo el equipo para lograr una toma de decisiones correcta para el desarrollo [18]. Son de vital importancia al hacer comprender como avanza el proyecto sobre todo las acciones planificadas.

### Recopilación de Requerimientos

Los requisitos permiten conocer la necesidad del cliente, su recopilación se la realiza por medio de entrevistas y reuniones, con lo que se llega a determinar expectativas y necesidades que se quieren cubrir al implementar el desarrollo [19]. En el **ANEXO II** del presente documento se encuentra a detalle la tabla de la recopilación de requerimientos para el presente trabajo de titulación.

### Historias de Usuario

Permiten conocer los requerimientos y procesos a seguir, ayudando a que el equipo de trabajo conste de cada una de las tareas a trabajar. Son tarjetas que detallan el requerimiento con un apartado de las iteraciones que se va a realizar, en ese caso cada una de ellas es necesaria para el avance y organización de ejecutar el desarrollo. La **TABLA II** muestra un ejemplo de historia de usuario, mientras que en el **ANEXO II** del presente documento se encuentra las nueve historias de usuario restantes.

**TABLA II:** Historia de Usuario Nro. 8.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU008 | **Usuario:** Desarrollador Front-End |
| **Nombre historia:** Visualizar el contenido educativo de cada semestre | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 2 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, como perfil estudiante, tendrá la acción de solo visualizar el contenido educativo de cada semestre. | |
| **Observación:** En este caso al acceder al sistema como estudiante se le restringe la funcionalidad de solo poder ver la información carga anteriormente por un administrador. | |

### *Product Backlog*

Al utilizar una metodología ágil, se emplea una forma distinta de trabajo y organización. El *Producto Backlog* se define como una información ordenada de todas las actividades a realizarse, las cuales son divididas por *sprints,* logrando gestionar de manera correcta el proyecto. Siendo una lista de las actividades que se deben hacer y que deben estar visibles para todo el equipo para que todos entiendan la lógica del negocio [20]. En definitiva, es el orden con el que se debe trabajar, desglosando y definiendo cada requerimiento. En el **ANEXO II** del presente documento se encuentra a detalle el *Producto Backlog* para el presente trabajo de titulación.

### *Sprint Backlog*

Se define como la lista de tareas a cumplir, permite visualizar el avance realizado. Su objetivo es mantener la transparencia, revisión y análisis para conocer si las tareas tienen

un buen avance en el tiempo establecido [21]. En el **ANEXO II** del presente documento se encuentra a detalle el *Sprint Backlog* para el presente trabajo de titulación.

## 2.2 Diseño de la arquitectura

El diseño de la arquitectura ayuda a planificar el desarrollo y estructura que se va a llevar en el proyecto. Por otro lado, elegir el mejor conjunto de herramientas permite plantear una visión general del desarrollo de *software* de manera que a largo plazo se pueda generar nuevas funcionalidades satisfaciendo las necesidades del usuario final [22].

### Patrón arquitectónico

En el proyecto se ha implementado el patrón arquitectónico Modelo Vista Controlador (MVC), permitiendo jerarquizar la información, la lógica del sistema y la interfaz. Igualmente organiza cada uno de los componentes del sistema web, reduciendo el esfuerzo de programación y siendo útil para la escalabilidad en el futuro [23].

- **Modelo**: Componente donde existe interacción con los datos (consultas, búsquedas, filtros y actualizaciones), permite almacenar y definir qué datos va a contener en si el sistema web [23].
- **Vista**: Componente donde se almacena el código para generar la visualización de la interfaz gráfica, logrando crear pantallas, paginas o cualquier tipo de resultado que el usuario final solicite al interactuar con el sistema web [23].
- **Controlador**: Componente que ejerce como intercesor entre el usuario y el sistema web, respondiendo a los artefactos que puedan solicitar para llevar a cabo las necesidades del sistema web, enlazando vistas y modelos [23].

En la **Fig.1** se muestra la lógica que se ha llevado mediante una estructura jerarquizada y escalable para el desarrollo del componente *backend,* indicando el direccionamiento conforme a la lógica propuesta.

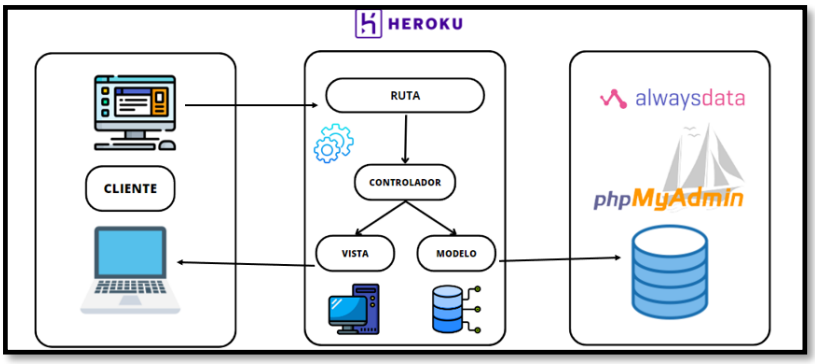



**Fig.1:** Patrón Arquitectónico – *backend.*

## 2.3 Herramientas de desarrollo

Una vez que se ha determinado el patrón arquitectónico, es importante realizar la elección del conjunto de herramientas para el éxito del desarrollo, con el objetivo de lograr resultados eficientes basados en requerimientos. La **TABLA III** muestra el conjunto de herramientas para el desarrollo del componente *backend*.

**TABLA III:** Herramientas para el desarrollo del Sistema Web.

| Herramienta | Justificación |
|---|---|
| ***Visual Studio Code*** | Es un editor de programación multiplataforma de código fuente, rápido y liviano, logrando la ejecución, depuración y ejecución de código [24]. Ha permitido la edición de código, haciendo uso de múltiples funciones únicas de la herramienta, posibilitando programar la lógica del componente *backend*. |
| ***Laravel*** | Es un f*ramework* PHP, de código abierto y gratis, que brinda herramientas útiles, poniendo énfasis en la calidad, mantenimientos y escalabilidad [25]. Ha permitido el desarrollo del componente *backend* de manera rápida y eficiente, por el conjunto de funcionalidades y recursos que facilitan el desarrollo de aplicaciones modernas. |
| ***Composer*** | Es una herramienta capaz de administrar dependencias en PHP, permitiendo la descarga e instalación de librerías y paquetes de terceros [26]. Por ello esta herramienta ha facilitado la instalación de dependencia y librerías, necesarias para la lógica del componente *backend.* |
| ***AlwaysData*** | Es un servicio en la nube (*phpMyAdmin*) que permite alojar la base de datos MySQL. Del mismo modo ha permitido tener una manipulación y constancia de datos, logrando recibir datos de entrada que operan entre ellos y devuelven un resultado de manera esperada. |

| | |
|---|---|
| ***ThunderClient*** | Es una herramienta liviana y eficiente, para probar las API de los clientes [27]. Por tanto, en el componente *backend* ha desempeñado una función indispensable, como es la ejecución de *endpoints* verificando su correcta funcionalidad. |
| ***Cloudinary*** | Es un servicio en la nube, la cual permite almacenar archivos de imagen [28]. En efecto la herramienta ha permitido almacenar imágenes y documentos que sube el role administrador, permitiendo al usuario estudiante visualizarlos. |
| ***Heroku*** | Es una plataforma en la nube, que sobrelleva diferentes lenguajes de programación, encargándose de la infraestructura que hay detrás [29]. Además, el componente *backend* ha consumido su mayor servicio perdiendo desplegar el sistema web mediante el registro de variables de entorno. |
| ***Swagger*** | Es una documentación online, que se basa sobre una API, siendo una herramienta con la cual se puede ver todos los *endpoints* de manera interactiva para verificar su funcionalidad. Se ha utilizado para tener la interactividad eficiente y entendible de la funcionalidad de cada método que se ha implementado. |

# 3 RESULTADOS

En esta sección, se procede a detallar los resultados que se han obtenido durante el desarrollo de los *endopoints* para el *backend*; actividades realizadas en cada uno de los *sprints* con sus respectivos objetivos alcanzados.

## 3.1 *Sprint* 0. Configuración del ambiente de desarrollo

En relación a la planificación del *Sprint Backlog*, el *Sprint* 0 tiene los siguientes resultados:

- Requerimientos necesarios para el sistema web.
- Elaboración y estructura del Modelo de Base de Datos.
- Roles de usuario.

### Requerimientos necesarios para el sistema web.

**Generar *endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.**

El *backend* cuenta con *endpoints* para que los usuarios (perfil administrador y estudiante) puedan hacer función de los tres métodos iniciar sesión, cerrar sesión y recuperar contraseña en el sistema web

**Generar *endpoints* para el registro de un estudiante.**

El *backend* cuenta con *endpoints* para que el usuario con perfil estudiante pueda registrarse conforme a los campos del formulario de registro. La información proporcionada es almacenada en la base de datos y es necesaria para acceder al sistema web.

**Generar *endpoints* para visualizar y editar perfil.**

El *backend* cuenta con *endpoints* para que los usuarios (perfil administrador y estudiante) visualice y edite su perfil. Se cuenta con validación de campos al momento que ingresa el nuevo nombre y apellido para actualizar, permitiendo mostrar al usuario que no debe excederse de 35 caracteres en ambos campos.

**Generar *endpoints* para gestionar (CRUD) carreras y semestres.**

El *backend* cuenta con *endpoints* para que el usuario con perfil administrador gestione las carreras y semestres, realizando acciones de: crear, leer, actualizar y activar/desactivar.

**Generar *endpoints* para visualizar carreras y semestres.**

El *backend* cuenta con *endpoints* para que el usuario con perfil estudiante pueda visualizar las carreras y semestres que oferta la ESFOT.

**Generar *endpoints* para gestionar (CRUD) materias y contenido educativo.**

El *backend* cuenta con *endpoints* para que el usuario con perfil administrador gestione (realizando acciones de: crear, leer, actualizar y activar/desactivar) las materias y sus contenidos, conforme a la carrera y semestre asignado.

**Generar *endpoints* para visualizar el contenido educativo.**

El *backend* cuenta con *endpoints* para que el usuario con perfil estudiante pueda visualizar el contenido educativo, que se encuentra cargado en la materia seleccionada.

**Generar *endpoints* para gestionar (CRUD) preguntas e inquietudes.**

El *backend* cuenta con *endpoints* para que los usuarios (perfil administrador y estudiante) puedan interactuar mediante preguntas e inquietudes, conforme a la carrera, semestres y materias.

A continuación, las **Fig. 2** y **Fig. 3** muestran los perfiles con la funcionalidad que pueden realizar en el sistema web.

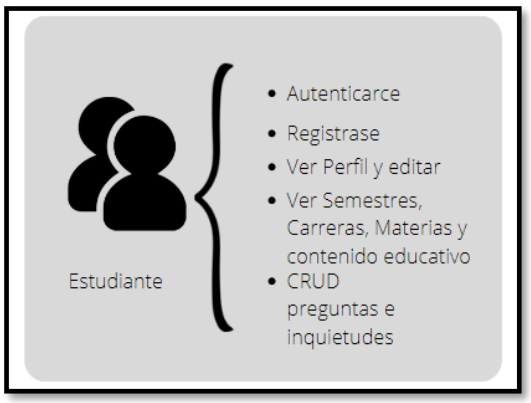



**Fig. 2:** Usuario con perfil estudiante.

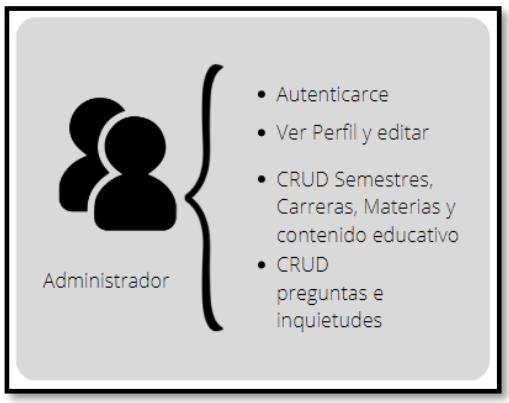



**Fig. 3:** Usuario con perfil administrador.

## Elaboración y estructura del Modelo de Base de Datos.

La creación y estructura del modelo de base de datos es un pilar fundamental para el desarrollo del sistema web, dado que es el lugar donde se almacena la información, tomando en cuenta los requerimientos planteados. Para la creación de la base de datos del presente proyecto se hace uso de Enloquent, definido como el mapeo relacional de objetos (ORM), logrando tener la facilidad y sencilles al momento de interactuar con la base de datos del sistema web.

La **Fig. 4** muestra el esquema de la base de datos que se ha utilizado para tener una conexión con el *backend,* teniendo integridad de los datos ingresados.

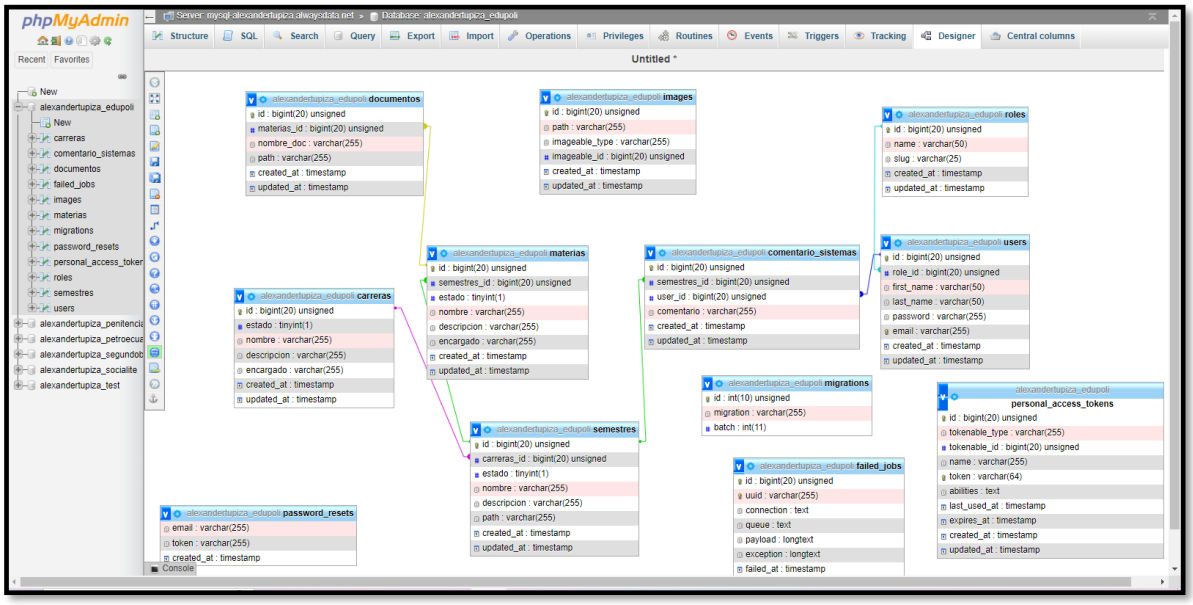



**Fig. 4:** Modelo de datos en *phpMyAdmin*.

**Roles de usuarios.**

La **Fig. 5** indica los diferentes perfiles de usuarios, cada uno con su respectivo módulo de funcionalidad dentro del sistema web.

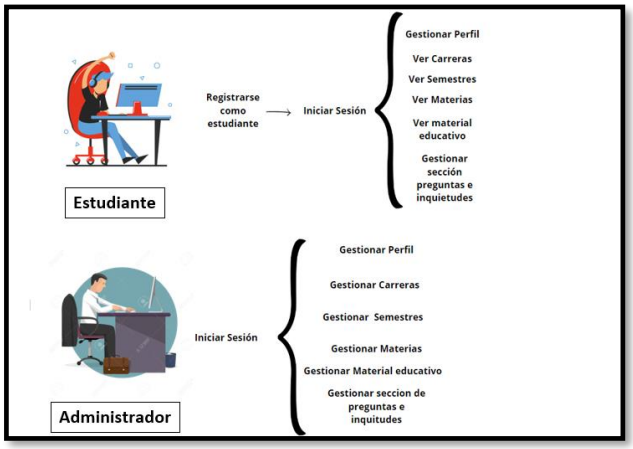



**Fig. 5:** Roles de usuario.

## 3.2 *Sprint* 1. Implementación del módulo usuarios con autenticación.

En relación a la planificación del *Sprint Backlog*, el *Sprint* 1 tiene los siguientes resultados:

- Implementar los *endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.

- Implementar los *endpoints* para visualizar y editar el perfil de usuario.
- Implementar el *endpoint* para registro (perfil estudiante) en el sistema web.

**Implementar los *endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.**

Para el inicio de sesión, cierre de sesión y cambio de contraseña se han implementado varios métodos útiles, estos son parte del módulo de autenticación.

El método “iniciar sesión” cuenta con dos campos, correo y contraseña, obligatorios para que el usuario pueda ingresar al sistema web, tal como lo muestra la **Fig. 6**. El método “cerrar sesión” elimina el token registrado, para que el usuario se desconecte del *backend,* como se muestra en la **Fig. 7**. Finalmente, el método “recuperar contraseña” requiere el ingreso del correo electrónico ya que a este correo se envía el *link* con los campos contraseña nueva y confirmación de nueva contraseña, como se muestra la

```php
// Función para el manejo del inicio de sesión
public function login(Request $request)
{
    // Validación de los datos de entrada
    $request->validate([
        'email' => ['required', 'string', 'email'],
        'password' => ['required', 'string'],
    ]);

    // Obtener un usuario
    $user = User::where('email', $request['email'])->first();

    // Valida lo siguiente
    //  * Si no existe un usuario
    //  * Si no tiene un estado
    //  * Verificar el rol del usuario existe en el array creado de roles descartados
    //  * Si no es el mismo password
    if (!$user || !Hash::check($request['password'], $user->password)) {
        // Se invoca a la función padre
        return $this->sendResponse(message: 'Las credenciales proporcionadas son incorrectas.', code: 404);
    }

    // Valida lo siguiente
    //  * Si el token de usurio no es vacío
    if (!$user->tokens->isEmpty()) {
        // Se invoca a la función padre
        return $this->sendResponse(message: 'User is already authenticated.', code: 403);
    }

    // Se procede a la creación de un token para el usuario
    $token = $user->createToken('auth-token')->plainTextToken;

    // Se invoca a la función padre
    return $this->sendResponse(message: 'Autenticación Satisfactoria.', result: [
        'user' => new UserResource($user),
        'access_token' => $token,
        'token_type' => 'Bearer',
    ]);
}
```

**Fig. 6:** Método para iniciar sesión.

```php
// Función para el manejo del cierre de sesión
public function logout(Request $request)
{
    // Se obtiene el token en el request y eliminar de la BDD
    // https://laravel.com/api/9.x/Illuminate/Http/Request.html

    $request->user()->tokens()->delete();

    // Se invoca a la función padre
    return $this->sendResponse(message: 'Deslogueado.');
}
```

**Fig. 7:** Método para cerrar sesión.

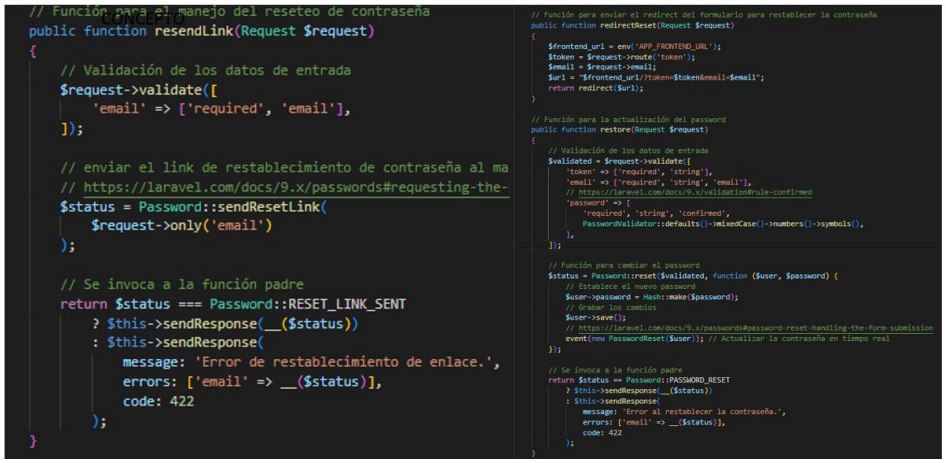


**Fig. 8:** Método para iniciar sesión.

### Implementar los *endpoints* para visualizar y editar el perfil de usuario.

Para visualizar y editar perfil (administrador y estudiante) se han implementado tres métodos. El primer método empleado para la visualización de los datos del perfil conforme al token de cada usuario (ver **Fig. 9**). El segundo método para la actualización de datos de cada perfil ingresado al sistema web (ver **Fig. 10**). Y el tercer método empleado para la actualización del avatar (ver **Fig. 11**).

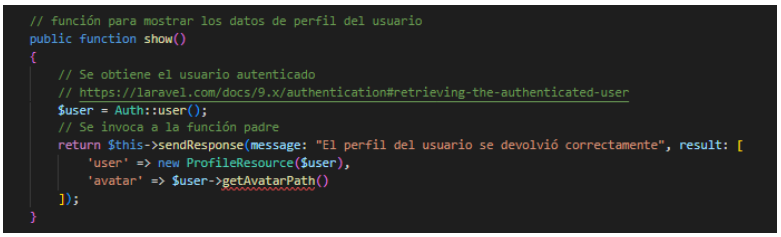


**Fig. 9:** Método ver perfil.

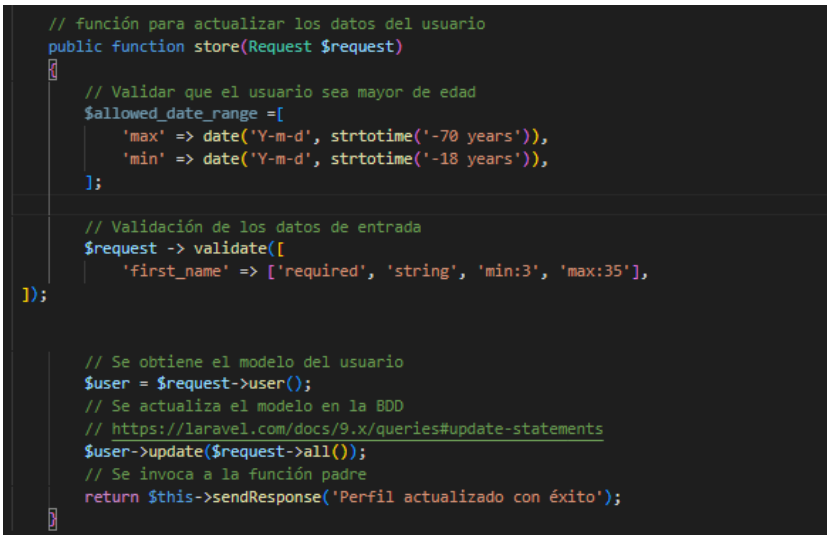
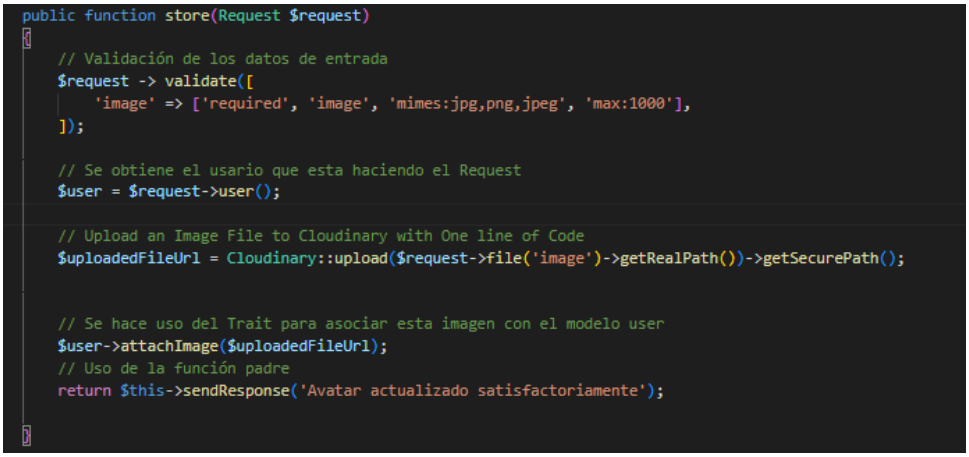


**Fig. 10:** Método para editar perfil.

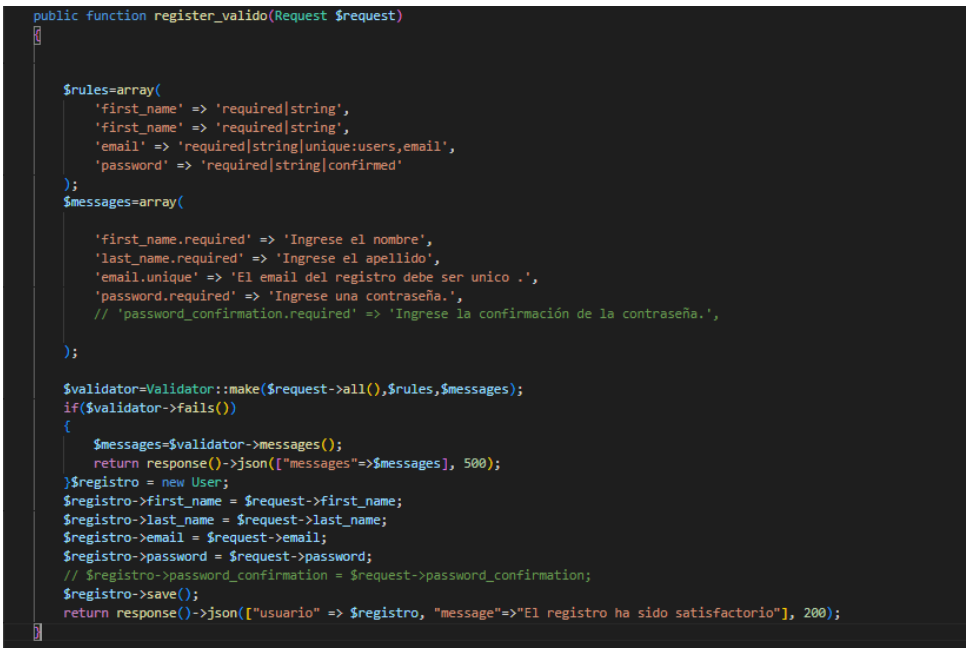


```php
public function store(Request $request)
{
    // Validación de los datos de entrada
    $request -> validate([
        'image' => ['required', 'image', 'mimes:jpg,png,jpeg', 'max:1000'],
    ]);

    // Se obtiene el usario que esta haciendo el Request
    $user = $request->user();

    // Upload an Image File to Cloudinary with One line of Code
    $uploadedFileUrl = Cloudinary::upload($request->file('image')->getRealPath())->getSecurePath();


    // Se hace uso del Trait para asociar esta imagen con el modelo user
    $user->attachImage($uploadedFileUrl);
    // Uso de la función padre
    return $this->sendResponse('Avatar actualizado satisfactoriamente');

}
```

**Fig. 11:** Método para actualizar avatar.

### Implementar el *endpoint* para registro (perfil estudiante) en el sistema web.

Para realizar el registro de un usuario con perfil estudiante se ha implementado un método que requiere el ingreso de cinco campos (nombre, apellido, correo, contraseña y confirmación de contraseña), generando un registro en la base de datos. La **Fig. 12** muestra la implementación de este método.



```php
public function register_valido(Request $request)
{


    $rules=array(
        'first_name' => 'required|string',
        'first_name' => 'required|string',
        'email' => 'required|string|unique:users,email',
        'password' => 'required|string|confirmed'
    );
    $messages=array(

        'first_name.required' => 'Ingrese el nombre',
        'last_name.required' => 'Ingrese el apellido',
        'email.unique' => 'El email del registro debe ser unico .',
        'password.required' => 'Ingrese una contraseña.',
        // 'password_confirmation.required' => 'Ingrese la confirmación de la contraseña.',

    );

    $validator=Validator::make($request->all(),$rules,$messages);
    if($validator->fails())
    {
        $messages=$validator->messages();
        return response()->json(["messages"=>$messages], 500);
    }$registro = new User;
    $registro->first_name = $request->first_name;
    $registro->last_name = $request->last_name;
    $registro->email = $request->email;
    $registro->password = $request->password;
    // $registro->password_confirmation = $request->password_confirmation;
    $registro->save();
    return response()->json(["usuario" => $registro, "message"=>"El registro ha sido satisfactorio"], 200);
}
```

**Fig. 12:** Método para registrar usuario.

## 3.3 *Sprint* 2. Gestión de Carreras y Semestres.

En relación a la planificación del *Sprint Backlog*, el *Sprint* 2 tiene los siguientes resultados:

- Implementar los *endpoints* para gestionar las carreras ofertadas por la ESFOT.
- Implementar los *endpoints* para gestionar los semestres por carrera.

**Implementar los *endpoints* para gestionar las carreras ofertadas por la ESFOT.**

Para gestionar carreras se han implementado cuatro métodos mismos que son necesarios para el avance de la lógica de cada uno de los requerimientos, permitiendo ir cumpliendo con los objetivos del *backend.* Estos cuatro métodos son:

1. Método para crear carreras, se muestra en la **Fig. 13**,
2. Método para visualizar carreras, se muestra en la **Fig. 14**,
3. Método para actualizar carrera, se muestra en la **Fig. 15**, y
4. Método para activar o desactivar una carrera registrada, se muestra en la **Fig. 16**.

```php
//FUNCION PARA CREAR CARRERAS
public function store(Request $request)
{
    $rules=array(
        'nombre' => 'required|string|unique:carreras',
        'descripcion' => 'required|string',
        'encargado' => 'required|string|'
    );
    $messages=array(

        'nombre.unique' => 'La carrera debe ser unica',
        'descripcion.required' => 'Debe tener una descripción.',
        'encargado.required' => 'Ingrese una persona a cargo.',
        // 'password_confirmation.required' => 'Ingrese la confirmación de la contraseña.',

    );

    $validator=Validator::make($request->all(),$rules,$messages);
    if($validator->fails())
    {
        $messages=$validator->messages();
        return response()->json(["messages"=>$messages], 500);
    }$carreras = new Carreras();
    $carreras->nombre= $request->nombre;
    $carreras->descripcion = $request->descripcion;
    $carreras->encargado = $request->encargado;
    // $registro->password_confirmation = $request->password_confirmation;
    $carreras->save();
    return response()->json(["carreras" => $carreras, "message"=>"La carrera se ha creado satisfactoriamente"], 200);

}
```

**Fig. 13:** Método para crear carreras.

```php
//FUNCION PARA VER LAS CARRERAS CREADAS ACTIVAS
    public function index (Request $request){


        $carreras = Carreras::where('estado',1)->get();

        return response()->json([
            'data'=> $carreras

        ]);
    }

    //FUNCION PARA VER LAS CARRERAS CREADAS
    public function index_admin (Request $request){

        $carreras = Carreras::all();
        return response()->json([
            'data'=> $carreras

        ]);
    }
```

**Fig. 14:** Método para ver carreras.

```php
//FUNCION PARA ACTUALIZAR LA CARRERA
    public function update (Request $request, $id){

        $fields = $request->validate([

            'nombre' => 'required|string|unique:carreras',
            'descripcion' => 'required|string',
            'encargado' => 'required|string'

        ]);


        $carreras = Carreras::find($id);


        if($carreras){
            $carreras->update($fields);

            return response()->json([
                'message' => 'La carrera se actualizado satisfactoriamente.',
                'data'=> $carreras
            ]);
        }
        else{
            return response()->json([
                'message'=> 'No existe ninguna carrera con ese id.'

            ], 404);

        }


    }
```

**Fig. 15:** Método para actualizar carreras.

```php
//ACTIVAR E INACTIVAR CARRERAS


    public function destroy (Carreras $carreras){


        // $carreras = Carreras::find($id);
    $carreras_estado = $carreras->estado;
    $mensaje = $carreras_estado ? "inactiva.":"activa.";
    $carreras->estado = !$carreras_estado;
    $carreras->save();

    return $this->sendResponse(message: "La carrera esta $mensaje ");


    }
```

**Fig. 16:** Método para activar y desactivar carreras.

### Implementar los *endpoints* para gestionar los semestres por carrera.

Para gestionar semestres se han implementado cuatro métodos, los cuales son necesarios para llevar la lógica del almacenamiento de documentos educativos según la carrera. Estos cuatro métodos son:

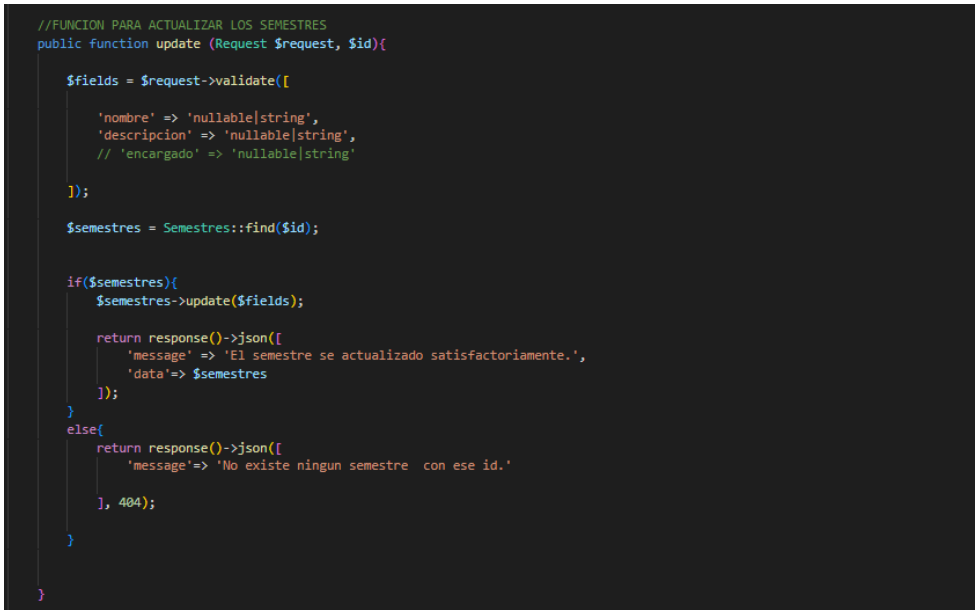
1. Método para crear semestres en relación con la carrera, se muestra en la **Fig. 17**,
2. Método para visualizar semestres, se muestra en la **Fig. 18**,
3. Método para actualizar datos de la carrera, se muestra en la **Fig. 19**, y
4. Método para activar y desactivar semestre, se muestra en la **Fig. 20**.

```php
//---------FUNCION PARA CREAR LOS SEMESTRES------------//
public function store(Request $request)
{
    $rules=array(
        'nombre' => 'required|string|unique:semestres',
        'descripcion' => 'required|string',

    );
    $messages=array(

        'nombre.unique' => 'El semestre debe ser unico',
        'descripcion.required' => 'Debe tener una descripción.',

    );

    $validator=Validator::make($request->all(),$rules,$messages);
    if($validator->fails())
    {
        $messages=$validator->messages();
        return response()->json(["messages"=>$messages], 500);
    }$semestres = new Semestres();
    $semestres ->nombre= $request->nombre;
    $semestres ->descripcion = $request->descripcion;
    // $registro->password_confirmation = $request->password_confirmation;
    $semestres ->save();
    return response()->json(["semestres" => $semestres , "message"=>"El semestre se ha creado satisfactoriamente"], 200);

}
```

**Fig. 17:** Método para crear semestres.

```php
//--------FUNCION PARA VER LOS SEMESTRES ACTIVOS--------//
public function index (Request $request){

    $semestres = Semestres::where('estado',1)->get();

    return response()->json([
        'data'=> $semestres

    ]);
}
//-------------------------------------------------------//


//---FUNCION PARA VER TODOS LOS SEMESTRES CREADOS-----//
public function index_admin (Request $request){

    $semestres = Semestres::all();

    return response()->json([
        'data'=> $semestres

    ]);
}
```

**Fig. 18:** Método ver semestres.



```php
//FUNCION PARA ACTUALIZAR LOS SEMESTRES
public function update (Request $request, $id){

    $fields = $request->validate([

        'nombre' => 'nullable|string',
        'descripcion' => 'nullable|string',
        // 'encargado' => 'nullable|string'

    ]);

    $semestres = Semestres::find($id);


    if($semestres){
        $semestres->update($fields);

        return response()->json([
            'message' => 'El semestre se actualizado satisfactoriamente.',
            'data'=> $semestres
        ]);
    }
    else{
        return response()->json([
            'message'=> 'No existe ningun semestre  con ese id.'

        ], 404);

    }


}
```

**Fig. 19:** Método para actualizar semestres.

```php
//ACTIVAR E INACTIVAR SEMESTRE


public function destroy (Semestres $semestres){


    // $carreras = Carreras::find($id);
$semestres_estado = $semestres->estado;
$mensaje = $semestres_estado ? "Inactiva":"Activa";
$semestres->estado = !$semestres_estado;
$semestres->save();

return $this->sendResponse(message: "El semestres esta $mensaje ");


}
```

**Fig. 20:** Método para activar y desactivar semestres.

## 3.4 *Sprint* 3. Gestión de Materias y Recursos

En relación a la planificación del *Sprint Backlog*, el *Sprint* 3 tiene los siguientes resultados:

- Implementar los *endpoints* para gestionar materias.
- Implementar los *endpoints* para gestionar recursos educativos.

### Implementar los *endpoints* para gestionar materias.

Para gestionar las materias conforme a la carrera y semestre se han creado cuatro métodos mismos que permiten que el estudiante visualice materias existentes en relación con la carrera y nivel de semestre elegido. Estos cuatro métodos son:

1. Método para crear materia, se muestra en la **Fig. 21**,

2. Método para visualizar carreras registradas, se muestran en la **Fig. 22**,

3. Método para actualizar materias, se muestra la **Fig. 23**, y

4. Método para activar y desactivar materias registradas, se muestra en la **Fig. 24**.

```php
//FUNCION PARA CREAR MATERIAS
public function store(Request $request)
{
    $rules=array(
        'nombre' => 'required|string|unique:carreras',
        'descripcion' => 'required|string',
        'encargado' => 'required|string|'
    );
    $messages=array(

        'nombre.unique' => 'La materia debe ser unica',
        'descripcion.required' => 'Debe tener una descripción.',
        'encargado.required' => 'Ingrese una persona a cargo.',
        // 'password_confirmation.required' => 'Ingrese la confirmación de la contraseña.',

    );

    $validator=Validator::make($request->all(),$rules,$messages);
    if($validator->fails())
    {
        $messages=$validator->messages();
        return response()->json(["messages"=>$messages], 500);
    }$materias = new Materias();
    $materias->nombre= $request->nombre;
    $materias->descripcion = $request->descripcion;
    $materias->encargado = $request->encargado;
    // $registro->password_confirmation = $request->password_confirmation;
    $materias->save();
    return response()->json(["materias" => $materias, "message"=>"La materia se ha creado satisfactoriamente"], 200);

}
```

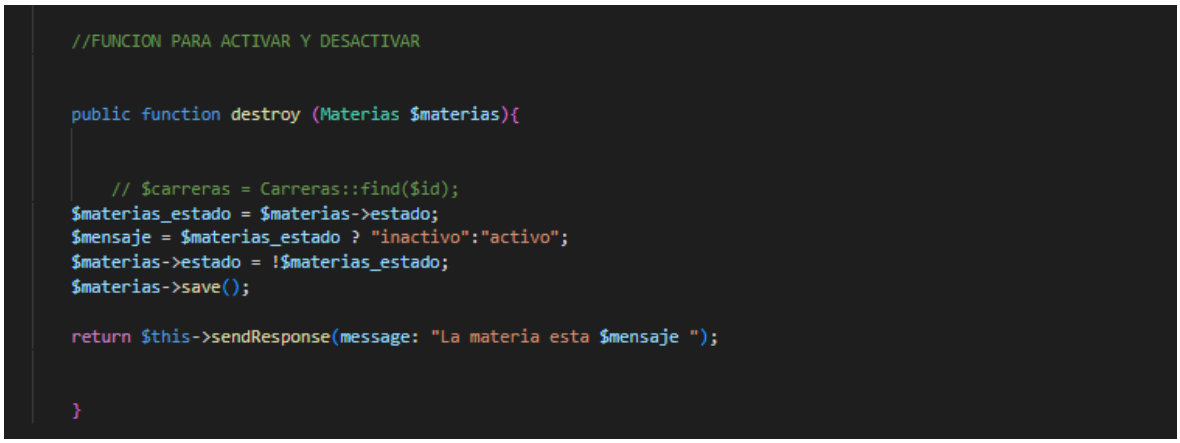
**Fig. 21:** Método para crear materias.

```php
    //FUNCION PARA VER LAS MATERIAS CREADAS ACTIVAS
    public function index (Request $request){


        $materias = Materias::where('estado',1)->get();
        return response()->json([
            'data'=> $materias

        ]);
    }

    //FUNCION PARA VER TODAS LA MATERIAS CREADAS
    public function index_admin (Request $request){

        $materias = Materias::all();
        return response()->json([
            'data'=> $materias

        ]);
    }
```

**Fig. 22:** Método para ver materias.

```php
//FUNCION PARA ACTUALIZAR LA MATERIA
public function update (Request $request, $id){

    $fields = $request->validate([

        'nombre' => 'nullable|string',
        'descripcion' => 'nullable|string',
        'encargado' => 'nullable|string'

    ]);

    $materias = Materias::find($id);


    if($materias){
        $materias->update($fields);

        return response()->json([
            'message' => 'La materia se actualizado satisfactoriamente.',
            'data'=> $materias
        ]);
    }
    else{
        return response()->json([
            'message'=> 'No existe ninguna carrera con ese id.'

        ], 404);

    }


}
```

**Fig. 23:** Método para actualizar materias.



```php
//FUNCION PARA ACTIVAR Y DESACTIVAR


public function destroy (Materias $materias){


    // $carreras = Carreras::find($id);
$materias_estado = $materias->estado;
$mensaje = $materias_estado ? "inactivo":"activo";
$materias->estado = !$materias_estado;
$materias->save();

return $this->sendResponse(message: "La materia esta $mensaje ");


}
```

**Fig. 24:** Método para activar y desactivar materias.

### Implementar los *endpoints* para gestionar recursos educativos.

Para gestionar los recursos educativos conforme a la materia, se han creado cinco métodos, logrando que el usuario estudiante pueda visualizar los documentos pdf con relación a la matera. Estos cinco métodos son:

1. Método para subir un documento pdf a la sección de recursos educativos, se muestra en la **Fig. 25.**
2. Método para visualizar el path del documento conforme al ID correspondiente, se muestra en la **Fig. 26**.

3. Método para visualizar todos los path de los documentos existentes, se muestra en la **Fig. 27**.
4. Método para actualizar el documento pdf, se muestra en la **Fig. 28**, y
5. Método para eliminar los documentos educativos, se muestra en la **Fig. 29**.

```php
public function store_admin (Request $request, Materias $materias)
{
    $response = Gate::inspect('gestion-documentos-admin');

    if($response->allowed())
    {
        $doc=$request -> validate([

            'documentos' => ['required', 'file', 'mimes:pdf', 'max:20000'],
        ]);
        $doc1 = $doc [ 'documentos'];
        $uploadedFileUrl = Cloudinary::uploadFile($doc1->getRealPath())->getSecurePath();
        $documentos = new Documentos($request->all());
        $documentos->materias_id = $materias->id;

        $documentos->path= $uploadedFileUrl;
        $documentos->save();

        return $this->sendResponse('Documento subido  satisfactoriamente');
    }else{
        echo $response->message();
    }
}
```

**Fig. 25:** Método para subir documentos.

```php
public function show_admin ($id){
    $response = Gate::inspect('gestion-documentos-admin');

    if($response->allowed())
    {
        $documentos = Documentos::find($id);
        if($documentos){
        return response()->json([
            'mesagge' => 'Path del documento a vizualizarse',
            'data'=> $documentos

        ]);
        }else{
            return response()->json([
                'message' => 'No existe ninguna path con ese id.',


            ], 404);

        }
    }else{
        echo $response->message();
    }
}
```

**Fig. 26:** Método para ver documentos según el ID.

```php
//FUNCION PARA VER TODOS LOS DOCUMENTOS CREADOS  CREADAS
public function index_admin (Request $request){
    $response = Gate::inspect('gestion-documentos-admin');

    if($response->allowed())
    {
        $documentos = Documentos::all();
        return response()->json([
            'data'=> $documentos

        ]);
    }else{
        echo $response->message();
    }

}
```

**Fig. 27:** Método para ver todos los documentos.

```php
//FUNCION PARA VER TODOS LOS DOCUMENTOS CREADOS  CREADAS
public function index_admin (Request $request){
    $response = Gate::inspect('gestion-documentos-admin');

    if($response->allowed())
    {
        $documentos = Documentos::all();
        return response()->json([
            'data'=> $documentos

        ]);
    }else{
        echo $response->message();
    }

}
```

**Fig. 28:** Método para actualizar documentos.

```php
// FUNCION PARA ELIMINAR DOCUMENTOS

public function delete_admin ($id){

    $response = Gate::inspect('gestion-documentos-admin');

    if($response->allowed())
    {
        $documentos = Documentos::find($id);


        if($documentos){
            $documentos->delete();

            return response()->json([
                'message'=> 'El documento se ha eliminado satisfactoriamente'
            ]);
        }
        else{
            return response()->json([
                'message'=> 'No existe ninguna documento con ese id.'

            ], 404);

        }

    }else{
        echo $response->message();
    }

}
```

**Fig. 29:** Método para eliminar documentos.

## 3.5 *Sprint* 4. Gestionar Comentarios

En relación a la planificación del *Sprint Backlog*, el *Sprint* 4 tiene el siguiente resultado:

- Implementar los *endpoints* para gestionar la caja de comentarios.

### Implementar los *endpoints* para gestionar la caja de comentarios.

Para gestionar los comentarios de preguntas e inquietudes en relación con la materia, se han creado cinco métodos que permiten que el usuario estudiante conozca distintas inquietudes y soluciones referente a distintas opiniones del resto de usuarios. Estos cinco métodos son:

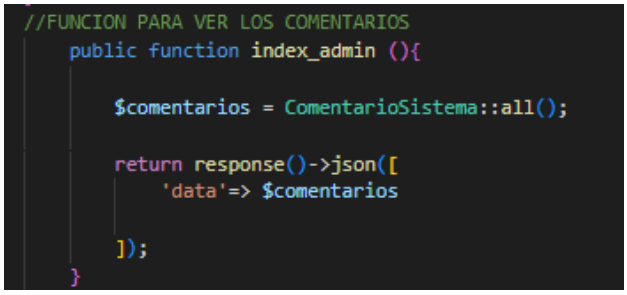
1. Método para crear comentarios e interactuar, se muestra en la **Fig. 30**,
2. Método para visualizar el comentario mediante un ID, se muestra en la **Fig. 31**,
3. Método para visualizar todos los comentarios, se muestra en la **Fig. 32**,
4. Método para actualizar un comentario, se muestra en la **Fig. 33**, y
5. Método para eliminar comentario, se muestra en la **Fig. 34**.

```php
//FUNCION PARA CREAR COMENTARIOS
public function store_admin (Request $request, Materias $materias)
{
    $rules=array(

        'comentario' => 'required|string',
    );
    $messages=array(
        'comentario.required' => 'Debe tener un comentario.',
    );

    $validator=Validator::make($request->all(),$rules,$messages);
    if($validator->fails())
    {
        $messages=$validator->messages();
        return response()->json(["messages"=>$messages], 500);
    }

    $comentarios = new ComentarioSistema($request->all());
    $comentarios ->materias_id = $materias->id;
    $user = Auth::user();
    $comentarios->user_id = $user->id;
    $comentarios->save();
    return response()->json(["comentarios" => $comentarios, "message"=>"El comentario se ha creado satisfactoriamente"], 200);

}
```

**Fig. 30:** Método para crear comentarios.

```php
//FUNCION PARA VER UN COMENTARIOS
    public function show_admin ( $id){

        $comentarios = ComentarioSistema::find($id);
        if($comentarios){
            return response()->json([
                'mesagge' => 'Comentario a vizualizarse',
                'data'=> $comentarios

            ]);
        }else{
            return response()->json([
            'message' => 'No existe ninguna carrera con ese id.',


            ], 404);

        }
    }
```

**Fig. 31:** Método para ver comentarios mediante un ID.



```php
//FUNCION PARA VER LOS COMENTARIOS
    public function index_admin (){

        $comentarios = ComentarioSistema::all();

        return response()->json([
            'data'=> $comentarios

        ]);
    }
```

**Fig. 32:** Método para ver todos comentarios.

```php
//FUNCION PARA ACTUALIZAR COMENTARIOS
    public function update_admin (Request $request, $id){

        $fields = $request->validate([

            'comentario' => 'nullable|string',

        ]);

        $comentarios = ComentarioSistema::find($id);


        if($comentarios){
            $comentarios->update($fields);

            return response()->json([
                'message' => 'El comentario se actualizado satisfactoriamente.',
                'data'=> $comentarios
            ]);
        }
        else{
            return response()->json([
                'message'=> 'No existe ningun comentario con ese id.'

            ], 404);

        }


    }
```

**Fig. 33:** Método para actualizar comentarios.

```php
// FUNCION PARA ELIMINAR COMENTARIOS
public function delete_admin (Request $request, $id){


    $comentarios = ComentarioSistema::find($id);


    if($comentarios){
        $comentarios->delete();

        return response()->json([
            'message'=> 'El comentario se ha eliminado satisfactoriamente'
        ]);
    }
    else{
        return response()->json([
            'message'=> 'No existe ninguna comentario con ese id.'

        ], 404);

    }


}
```

**Fig. 34:** Método para eliminar comentarios.

## 3.6 *Sprint* 5. Pruebas del *backend*

En relación a la planificación del *Sprint Backlog*, el *Sprint* 5 tiene los siguientes resultados:

- Ejecución de pruebas unitarias.
- Ejecución de pruebas de carga.
- Ejecución de pruebas de estrés.
- Despliegue del *backend* en Heroku y documentación api en Swagger.

### Ejecución de pruebas unitarias.

El objetivo de las pruebas unitarias es verificar la funcionalidad de los métodos construidos mediante funciones, lo que permite detectar ciertos errores y así solucionarlos a tiempo antes de continuar con el desarrollo. A continuación, se muestra este tipo de pruebas que han sido realizadas por medio phpUnit (herramienta de *testing* unitarios perteneciente de Laravel), a los métodos que se consideran los más importantes dentro del *backend*, esto debido a que forman parte del *core* del negocio que se está resolviendo.

En la **Fig. 35** se muestra la función que permite hacer la prueba unitaria al *endpoint* "visualizar carreras", de esta manera se puede detectar ciertos errores si existieran en un fragmento de código. El resultado de esta prueba se puede observar en la **Fig. 36.** Las restantes pruebas unitarias realizadas se pueden observar con mayor detalle en el **ANEXO II.**

```php
//Prueba Unitaras- Metodo ver Carreras
public function test_ver_carreras()
{
    $user = User::Factory() -> make([
        'role_id' => 2
    ]);
    $request = $this->actingAs($user)->get('/api/v1/carreras/adminE');
    $request -> assertStatus(200);
}
```

**Fig. 35:** Prueba Unitaria del Método para visualizar Carreras.

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS C:\Users\Alexander\Desktop\TESIS BACKEND Y FRONT END\backendedupoli>php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Unit\PruebasTest
  ✓ ver carreras

  Tests:  1 passed
  Time:   8.92s

PS C:\Users\Alexander\Desktop\TESIS BACKEND Y FRONT END\backendedupoli>
```

**Fig. 36:** Resultados Prueba Unitaria del Método para visualizar Carreras.

Realizadas cada una de las pruebas unitarias se puede afirmar que este paso ha desempeñado un papel fundamental puesto que no se han detectado inconvenientes en los métodos desarrollados, por lo que cada uno está listo para ser consumido sin ningún problema.

### Ejecución de pruebas de carga.

Las pruebas de carga han permitido verificar la funcionalidad bajo una cantidad de usuarios y tiempo específico. Este tipo de pruebas debe ser llevada a cabo bajo un previo análisis de carga máxima que puede soportar ciertos *endpoints.* En ese sentido para ejecutar dichas pruebas se ha optado por utilizar la herramienta Apache Beanch, la cual ha permitido comprobar el rendimiento a través del ingreso masivo de usuarios a ciertos métodos.

En la **Fig. 37** se muestra el comando que permite realizar la prueba, para este caso se verifica el comportamiento que existe al ingresar 100 peticiones con 10 usuario para el método “visualizar carreras”, por otro lado, en la **Fig. 38.** muestra el resultado que se ha obtenido al ejecutar dicha prueba de carga. Las restantes pruebas de carga realizadas se pueden observar con mayor detalle en el **ANEXO II.**

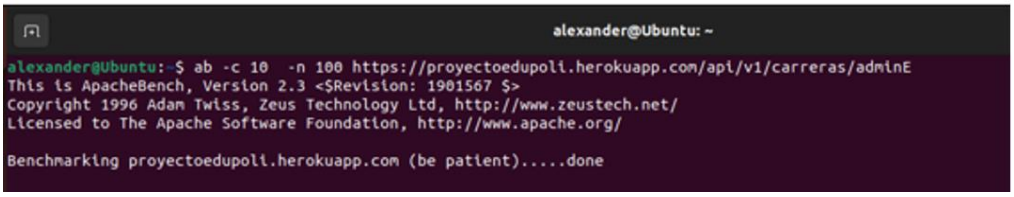



**Fig. 37**: Comando para la prueba de carga del método visualizar carreras.

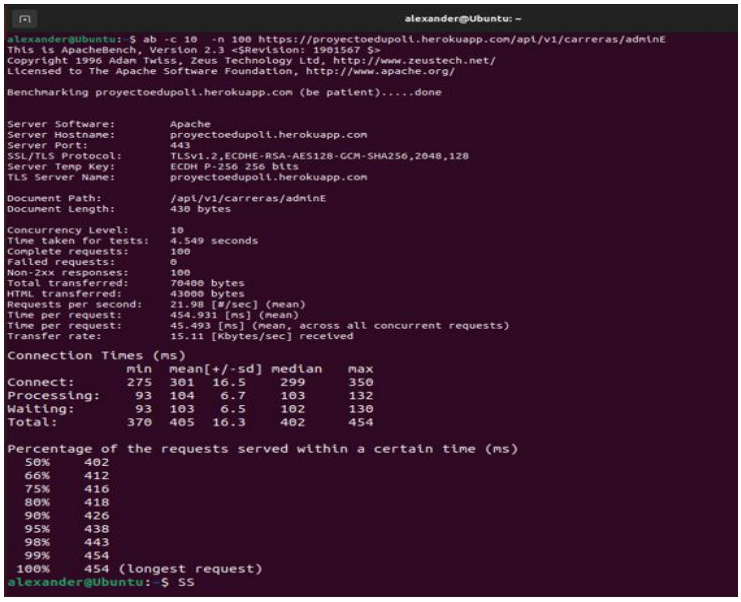



**Fig. 38:** Resultados Prueba de Carga del Método para visualizar Carreras.

Cabe mencionar que las pruebas de carga se han realizado tomando en cuenta que actualmente la ESFOT tiene 664 número de estudiantes matriculados (ver **ANEXO II**), de ahí el hecho de tomar 800 (20% más al número de estudiantes actuales) como número máximo de peticiones que se pueden realizar al mismo tiempo. Por ello una vez realizadas las diferentes pruebas se puede afirmar que soporta la carga esperada.

### Ejecución de pruebas de estrés.

El objetivo de este tipo de pruebas es ejecutar un ingreso masivo de peticiones y usuarios, logrando estresar al sistema, permitiendo conocer el ingreso tope de usuarios y peticiones con relación a cada *endpoint* que se ha implementado en el *backend*.

En la **Fig. 39** se muestra el comando y resultado de la prueba de estrés con relación al *endpoint "*visualizar carreras", concluyendo que al ingresar 1021 usuario y 1021 peticiones el método se estresa es decir no logra responder a tantas peticiones. Las restantes pruebas de estrés realizadas se pueden observar con mayor detalle en el **ANEXO II.**


**Fig. 39:** Resultados Prueba de Estrés del Método para ver Carreras.

Sabiendo el número máximo de estudiantes que actualmente tiene la ESFOT y tomando en cuenta que 1020 si responde se ha realizado diferentes pruebas para conocer en qué número de petición exactamente el sistema llega a estresarse, teniendo como resultado que al intentar 1021 número de usuarios y 1021 número de peticiones ya no puede responder de manera eficiente. Por ello se concluye que este número de peticiones estresa al sistema web, lo que significa que de llegar a tener este número de estudiantes el sistema no puede responder a todos al mismo tiempo.

### Despliegue del *backend* en Heroku y su documentación api en Swagger.

Una vez que se ha finalizado el proceso de desarrollo e implementación de métodos y las pruebas respectivas, se procede a desplegar el *backend* a producción por medio de *Heroku* y su api por *Swagger.* El resultado de se muestra en la **Fig. 40** y **Fig. 41**.

Acceder al *backend* se lo puede realizar mediante la siguiente URL.

https://proyectoedupoli.herokuapp.com/api/documentation

El detalle más específico del despliegue se encuentra en el **ANEXO IV** del presente documento.

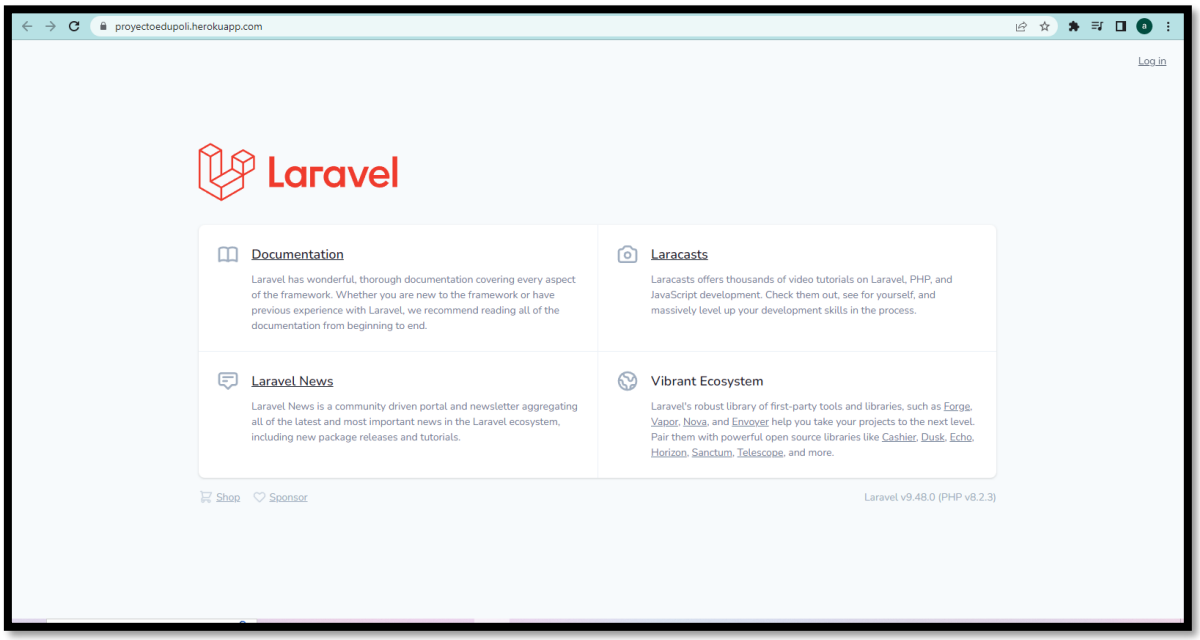



**Fig. 40:** Despliegue *Backend Heroku*.

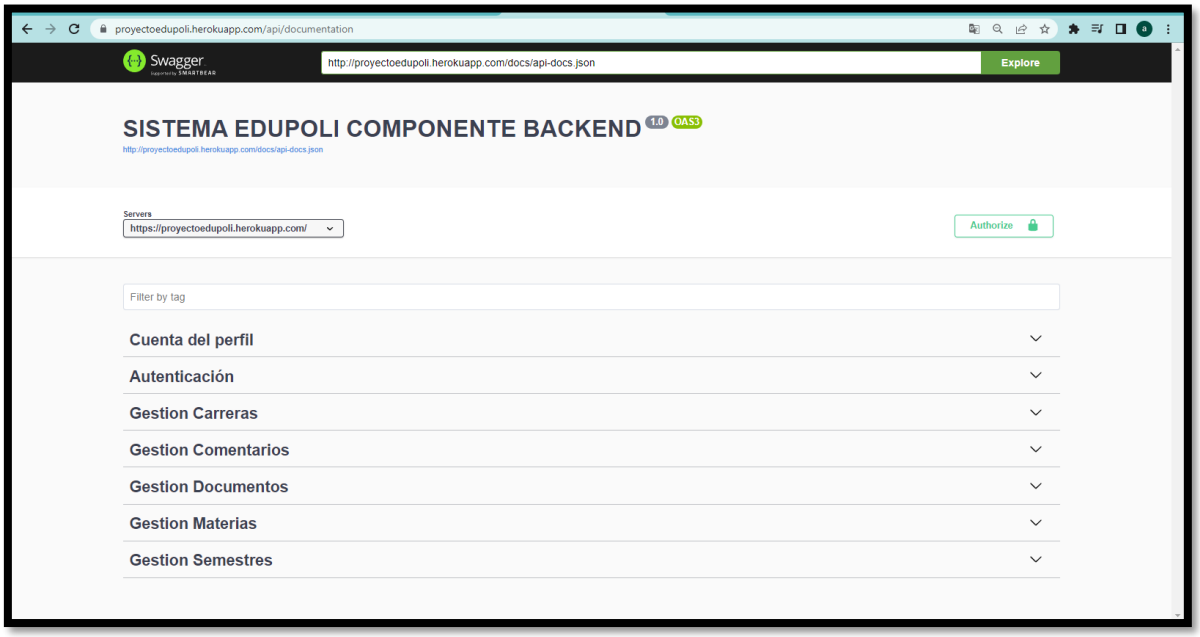



**Fig. 41:** Despliegue *Api Swagger*.

# 4 CONCLUSIONES

En esta sección se presenta las conclusiones obtenidas en el transcurso de este trabajo de integración curricular.

- La organización y estructura que se planteó antes de empezar con el proyecto fue fundamental para conocer la lógica de negocio que se quiere implementar, para eso se ha impulsado requerimientos e historias de usuario, permitiendo gestionar los métodos de manera eficiente.
- La indagación de metodologías ha permitido que este proyecto se realice de manera estructura con lineamientos, en este caso la metodología SCRUM permitió seguir las condiciones y organización para que el avance del proyecto sea satisfactorio.
- Optar por lineamientos y arquitecturas ha facilitado el desarrollo del componente *backend*, dado que gracias a la arquitectura MVC, el *backend* ha desempeñado estructurar el código de manera sencilla y eficiente permitiendo hacer entender la lógica para su posible mantenimiento o mejora en el futuro.
- La implementación de una base de datos SQL, ha facilitado la manipulación de datos de manera eficiente, permitiendo trabajar con Enloquent, lo que ha facilitado el mapear las estructuras de una base de datos relacional, logrando hacer uso de migraciones para colocar datos falsos para pruebas realizadas.
- Actualmente el *backend* se encuentra documentado en swagger, documentación que tiene la facilidad e interfaz de hacer uso de los métodos, Así es fácil y eficiente para que logren entender las rutas a ser consumidas por el *frontend*.
- Las pruebas que se ha implementado como son la unitarias, carga y estrés, han garantizado la funcionalidad de los *endpoints* creados, gracias a las pruebas se ha verificado funcionalidades y errores que se han solucionado a tiempo.

# 5 RECOMENDACIONES

En este apartado se presentan las recomendaciones, las cuales se han obtenido en el transcurso del desarrollo de este trabajo de integración curricular.

- Se recomienda indagar las herramientas que van a hacer útiles para el proyecto, para verificar hasta que limite su servicio es gratuito, lo que permite saber si su uso es eficiente en el proceso a corto y largo plazo.
- Se recomienda hacer varias pruebas independientes, para ir verificando la funcionalidad de cada uno de los métodos, logrando detectar algún fallo que puede ingresar el usuario final, puesto que el avance tanto de *frontend* y *backend* es igualitario.
- Se recomienda que, si se va a implementar nuevas funcionalidades, seguir la arquitectura y metodología escogida para seguir con el nuevo desarrollo.
- Se recomienda que las credenciales de servicios y sistema web no sean publicadas ni compartidas, por el motivo de manipulación de información o acceso no autorizado.

# 6 REFERENCIAS BIBLIOGRÁFICAS


[1] H. A. Oliva, «ResearchGate,» 03 2020. [En línea]. Available: https://www.researchgate.net/publication/340270478_La_Educacion_en_tiempos_de_pandemias_vision_desde_la_gestion_de_la_educacion_superior. [Último acceso: 21 12 2022].

[2] F. U. C. d. Norte, «Eduvirtual,» 08 2005. [En línea]. Available: https://eduvirtual.cuc.edu.co/moodle/pluginfile.php/640398/mod_resource/content/2/AVA.pdf. [Último acceso: 21 12 2022].

[3] P. Politico, «PortalPolitico.tv,» 22 04 2020. [En línea]. Available: https://www.portalpolitico.tv/negocios/educacion-en-tiempos-de-pandemiaun-reto-para-estudiantes-y-docentes. [Último acceso: 21 12 2022].

[4] S. Universidades, «Santander,» 16 11 2022. [En línea]. Available: https://www.becas-santander.com/es/blog/autoaprendizaje-una-soft-skill-imprescindible.html#:~:text=El%20autoaprendizaje%20es%20la%20habilidad,cuenta%20y%20a%20su%20propio%20ritmo.. [Último acceso: 21 12 2022].

[5] AliatUniversidades, «AliatUniversidades,» 2018. [En línea]. Available: https://aliatuniversidades.com.mx/blog/index.php/autoaprendizaje/. [Último acceso: 22 12 2022].

[6] Universia, «Universia,» 1 12 2020. [En línea]. Available: https://www.universia.net/co/actualidad/orientacion-academica/que-es-el-aprendizaje-autonomo.html. [Último acceso: 21 11 2022].

[7] Etecé, «Concepto,» 05 08 2021. [En línea]. Available: https://concepto.de/metodologia/. [Último acceso: 21 11 2022].

[8] V. Morles, «Scielo Analytics,» 01 2002. [En línea]. Available: http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0798-97922002000100006. [Último acceso: 21 11 2022].

[9] A. Encarna, «Global Growth Agents,» 06 02 2020. [En línea]. Available: https://www.wearemarketing.com/es/blog/que-es-la-metodologia-agile-y-que-beneficios-tiene-para-tu-empresa.html. [Último acceso: 21 11 2022].

[10] M. Fernando, «Crehana,» 18 05 2022. [En línea]. Available: https://www.crehana.com/blog/transformacion-digital/que-es-el-backend-y-como-usarlo/. [Último acceso: 21 11 2022].

[11] Maria, «Tribalyte Technologies,» 24 05 2021. [En línea]. Available: https://tech.tribalyte.eu/blog-que-es-una-api-rest. [Último acceso: 21 11 2022].

[12] S. DELSOL, «DELSOL,» [En línea]. Available: https://www.sdelsol.com/glosario/metodologia/. [Último acceso: 19 11 2022].

[13] QuestionPro, «QuestionPro,» [En línea]. Available: https://www.questionpro.com/blog/es/que-es-un-estudio-de-caso/. [Último acceso: 19 11 2022].

[14] Digité, «Digité,» [En línea]. Available: https://www.digite.com/es/agile/que-es-scrum/. [Último acceso: 20 11 2022].

[15] J. S. Hurtado, «IEBS,» 03 12 2021. [En línea]. Available: https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. [Último acceso: 22 11 2022].

[16] D. Molina, «IEBS,» 15 12 2021. [En línea]. Available: https://www.iebschool.com/blog/diferencias-product-owner-product-manager-marketing-marketing-estrategico/. [Último acceso: 21 11 2022].

[17] Lucichart, «Blog,» [En línea]. Available: https://www.lucidchart.com/blog/es/roles-y-responsabilidades-del-scrum-master. [Último acceso: 20 21 2022].

[18] C. Ramos, 02 02 2017. [En línea]. Available: https://cristinaramosvega.com/z-los-artefactos-scrum/. [Último acceso: 20 11 2022].

[19] I. Zabala, «Enredando Proyectos,» 07 06 2019. [En línea]. Available: https://enredandoproyectos.com/recopilar-los-requisitos-de-un-proyecto/. [Último acceso: 20 11 2022].

[20] D. Molina, «IEBS,» 09 12 2021. [En línea]. Available: https://www.iebschool.com/blog/que-es-un-product-backlog-y-como-hacer-uno-guia-scrum-agile-scrum/. [Último acceso: 21 11 2022].

[21] E. B. School, «EALDE Business School,» 2019 27 08. [En línea]. Available: https://www.ealde.es/product-backlog-sprint-backlog/. [Último acceso: 21 12 2022].

[22] P. Huet, «OpenWebinars,» 24 08 2022. [En línea]. Available: https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/. [Último acceso: 21 12 2022].

[23] M. García, «Coding or not,» 05 10 2017. [En línea]. Available: https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve. [Último acceso: 21 12 2022].

[24] R. Velasco, «SZ Soft Zone,» 25 05 2021. [En línea]. Available: https://www.softzone.es/programas/utilidades/visual-studio-code/. [Último acceso: 21 12 2022].

[25] Q. DEVS, «QUALITY DEVS,» 23 06 2021. [En línea]. Available: https://www.qualitydevs.com/2021/06/23/que-es-laravel/. [Último acceso: 21 12 2022].

[26] S. Soni, «Envatotuts,» 27 02 2020. [En línea]. Available: https://code.tutsplus.com/es/tutorials/what-is-composer-for-php-and-how-to-install-it--cms-35160. [Último acceso: 21 12 2022].

[27] C. Santiago, «SantiagoCarlos,» 20 05 2021. [En línea]. Available: http://santiagocarlos.com/thunder-client-client-http-para-probar-rest-api-desde-vs-code. [Último acceso: 21 12 2022].

[28] NISFE, «NISFE,» 14 02 2013. [En línea]. Available: https://www.nisfe.com/almacenamiento/cloudinary-almacenamiento-de-imagenes-en-la-nube-para-utilizarlas-en-una-web/. [Último acceso: 21 12 2022].

[29] CognosOnline, «CognosOnline,» 23 11 2022. [En línea]. Available: https://cognosonline.com/co/blog/que-es-heroku/. [Último acceso: 21 12 2022].

# 7 ANEXOS

En este apartado, se presentan los anexos que son información relevante y que transparenta la elaboración del presente proyecto de titulación.

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

# ANEXO I

A continuación, se presenta el certificado que la directora del proyecto de titulación ha emitido, donde se evidencia el resultado que se ha obtenido por la herramienta anti-plagio Turnitin.

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

**CERTIFICADO DE ORIGINALIDAD**

Quito, D.M. 28 de febrero de 2023

De mi consideración:

Yo, IVONNE FERNANDA MALDONADO SOLIZ, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DE UN BACKEND asociado al DESARROLLO DE UN SISTEMA WEB QUE SIRVA COMO GUÍA DE ESTUDIO PARA ESTUDIANTES DE LA ESFOT elaborado por el estudiante ALEXANDER FERNANDO TUPIZA CARRERA de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones (sin anexos), producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 8%.

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

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

Atentamente,

Firmado electrónicamente por:
IVONNE FERNANDA
MALDONADO SOLIZ

**Ivonne Maldonado**
**Docente Ocasional a Tiempo Completo**
**ESFOT**

# ANEXO II

## Recopilación de Requerimientos

La **TABLA I** muestro la Recopilación de requerimientos necesarios para el equipo de trabajo, que se obtuvo mediante peticiones del *Product Owner.*

**TABLA I:** Recopilación de requerimientos.

| **RECOPILACIÓN DE REQUERIMIENTOS** | | |
|---|---|---|
| **TIPO DEL SISTEMA** | **ID - RR** | **ENUNCIADO DEL ÍTEM** |
| ***BACKEND*** | **RR001** | Como usuario estudiante y administrador necesito generar *endpoints* para:<br>• Iniciar sesión<br>• Cerrará sesión<br>• Recuperar Contraseña |
| | **RR002** | Como usuario estudiante, debo registrarme a través de un formulario para poder utilizar el sistema web. |
| | **RR003** | Como usuario estudiante y administrado, pueden editar mi perfil |
| | **RR004** | Como usuario administrador puede realizar un CRUD completo para la gestión de carreras y semestres |
| | **RR005** | Como usuario estudiante solo puedo ver las carreras y semestres ya creados por el administrador. |
| | **RR006** | Como usuario estudiante y administrador, puedo visualizar mi perfil. |
| | **RR07** | Como usuario estudiante puedo visualizar el contenido educativo de cada materia que tiene el semestre. |
| | **RR08** | Como usuario administrador, puedo crear, leer, actualizar y borrar materias y contenido educativo. |
| | **RR09** | Como usuario estudiante y administrador, puedo tener un CRUD completo en el apartado de sugerencias e inquietudes. |

## Historias de Usuario

Después de haber obtenido los requerimientos necesarios, las historias de usuarios permiten tener una información corta pero detallada del proceso que se va a realizar, En este caso existen 10 historias de usuario, relacionando los requerimientos propuestos anteriormente que van desde la **TABLA II** hasta la **TABLA IX.**

**TABLA II:** Historia de usuario Nro.1.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU001 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Iniciar sesión, cerrar sesión y recuperar contraseña. | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 1 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma de la API, deber ingresar sus credenciales, como email y contraseña para:<br>• Iniciar sesión<br>• Cerrar sesión<br>• Recuperar contraseña | |
| **Observación:**<br>• En el apartado de la autenticación, el sistema web debe de validar que las credenciales ingresadas ya existan en la base de datos haciendo uso de un registro, dado que si no es así no se podrá hacer uso de sus respectivas funcionalidades. | |

**TABLA III:** Historia de usuario Nro.2.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU002 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Registrar usuario. | |
| **Prioridad en Negocio:** Media | **Riesgo en Negocio:** Media |
| **Iteración asignada:** 1 | |
| **Responsable:** Alexander Tupiza | |

| |
|---|
| **Descripción:** El desarrollador que consuma los *endpoints* de la API, para realizar un respectivo registro debe llenar un formulario compuesto por 5 inputs que son los siguientes:<br>• Nombre<br>• Apellido<br>• Correo electrónico<br>• Contraseña<br>• Confirmar Contraseña |
| **Observación:** Los campos a llenar son varchart, logrando tener una validación en la contraseña que debe coincidir con la contraseña de confirmación. |

**TABLA IV:** Historia de usuario Nro.3.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU003 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Modificar perfil. | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 2 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, tendrá la posibilidad de actualizar cualquier campo ingresado en el registro, asimismo podrá actualizar el avatar, con eso me baso que cualquier usuario final puedo realizar esta función.<br>• Nombre<br>• Apellido<br>• Email<br>• Imagen de perfil | |
| **Observación:** La actualización del perfil se da cuando el usuario lo desee, en ese caso lo que siempre quedara permanente es el id que se le otorga y el perfil con el cual se encuentra si es un administrado o un estudiante. | |

**TABLA V:** Historia de usuario Nro.4.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU004 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Gestionar carreras y semestres | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 2 | |
| **Responsable:** Alexander Tupiza | |

| |
|---|
| **Descripción:** El desarrollador que consuma la API, tendrá la posibilidad para gestionar las carreras y semestres, haciendo uso de un CRUD:<br>• Crear<br>• Eliminar<br>• Editar<br>• Leer |
| **Observación:** Al ingresar al sistema web como un perfil administrador, será la única persona que podrá hacer uso de un CRUD completo la gestión de carreras y semestres. |

**TABLA VI:** Historia de usuario Nro.5.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU005 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Visualizar las carreras y semestres | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 1 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, con usuario estudiante, tendrá solo acceso a la visualización de las carreras y semestres. | |
| **Observación:** Al usuario estudiante se le restringe la funcionalidad dentro del sistema logrando solo visualizar las carreras y semestres, a comparación de un administrador que puede tener acceso a un CRUD completo. | |

**TABLA VII:** Historia de usuario Nro.6.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU006 | **Usuario:** Desarrollador *Front-End* |
| **Nombre historia:** Visualizar perfil. | |
| **Prioridad en Negocio:** Alta | **Riesgo en Negocio:** Alta |
| **Iteración asignada:** 2 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, tiene la posibilidad, tanto administrado como estudiante el acceso a visualizar su perfil al momento de que este dentro del sistema web. | |

| |
|---|
| **Observación:** En este caso los dos perfiles existentes tienen la misma función de ver su perfil. |

**TABLA VIII:** Historia de usuario Nro.7.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU007 | **Usuario:** Desarrollador Front-End |
| **Nombre historia:** Gestionar contenido educativo. | |
| **Prioridad en Negocio:** Alto | **Riesgo en Negocio:** Alto |
| **Iteración asignada:** 2 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, con perfil de administrador, tiene la posibilidad de accionar a una CRUD completo para la gestión de información educativa:<br>• Crear<br>• Eliminar<br>• Editar<br>• Leer | |
| **Observación:** El usuario administrador en este caso es el único que puede hacer cambios en la información educativa cargada. | |

**TABLA IX:** Historia de usuario Nro.9.

| **HISTORIA DE USUARIO** | |
|---|---|
| **Identificador:** HU009 | **Usuario:** Desarrollador Front-End |
| **Nombre historia:** Gestionar las sugerencias e inquietudes. | |
| **Prioridad en Negocio:** Medio | **Riesgo en Negocio:** Medio |
| **Iteración asignada:** 1 | |
| **Responsable:** Alexander Tupiza | |
| **Descripción:** El desarrollador que consuma la API, tanto estudiante como administrador, tendrá la posibilidad de dar una funcionalidad completa en el apartado de inquietudes y sugerencias, logrando accionar un CRUD:<br>• Crear<br>• Eliminar<br>• Editar<br>• Leer | |

**Observación:** El usuario administrador y estudiante tiene la misma funcionalidad en el apartado de sugerencias, logrando interactuar con nuevas peticiones y conocimientos.

### *Product Backlog*

La **TABLA X** muestra detalladamente la estipulación de cada requisito que se ha implementado en el *backend* conforme a las iteraciones necesarias para cada historia de usuario, cumpliendo los requisitos necesarios del *Product Owner.*

**TABLA X:** Product Backlog.

| **ELABORACIÓN DEL *PRODUCT BACKLOG*** | | | | |
|---|---|---|---|---|
| **ID–HU** | **HISTORIA DE USUARIO** | **ITERACIÓN** | **ESTADO** | **PRIORIDAD** |
| HU001 | Iniciar sesión, cerrar sesión y recuperar contraseña. | 1 | Finalizada | Alta |
| HU002 | Registrar usuario. | 1 | Finalizada | Media |
| HU003 | Modificar perfil. | 2 | Finalizada | Alta |
| HU004 | Gestionar carreras y semestres | 2 | Finalizada | Alta |
| HU005 | Visualizar las carreras y semestres. | 1 | Finalizada | Alta |
| HU006 | Visualizar perfil | 2 | Finalizada | Alta |
| HU007 | Visualizar el contenido educativo de cada semestre. | 2 | Proceso | Alta |
| HU008 | Gestionar contenido educativo. | 2 | Proceso | Alta |
| HU09 | Gestionar las sugerencias e inquietudes. | 1 | Finalizada | Medio |

### *Sprint Backlog*

La **TABLA XI** se muestra la ejecución de los *Sprints* que se han desarrollado por parte del *backend*, describiendo las tareas asignadas a cada uno verificando el tiempo para cumplirlos respectivamente establecidos por el *Product Owner*.

**TABLA XI:** Sprint Backlog.

| **ELABORACIÓN DEL *SPRINT BACKLOG*** | | | | | | |
|---|---|---|---|---|---|---|
| **ID – SB** | **NOMBRE** | **MÓDULO** | **ID-HU** | **HISTORIAS DE USUARIO** | **TAREAS** | **TIEMPO ESTIMADO** |
| SB000 | Configuración del ambiente de desarrollo | **----** | **----** | **-------** | • Requerimientos necesarios para el sistema web.<br>• Elaboración y estructura del modelo de base de datos.<br>• Roles de usuarios. | 40 H |

| | | | | | | |
|---|---|---|---|---|---|---|
| SB001 | Autenticación y registro. | Modulo-Inicio de sesión rol estudiante y administrador | HU001 | Iniciar sesión, cerrar sesión y cambiar contraseña. | • Implementación de método para el inicio de sesión, cierre de sesión y recuperación de contraseña.<br>• Análisis de información registrada en la base de datos sobre el usuario para la autenticación verificando el rol con el que se encuentra.<br>• Validación de campos requeridos. | 40H |
| | | Modulo-Registro de usuario estudiante. | HU002 | Registro de estudiante | • Ejecución de método para el registro de un nuevo usuario.<br>• Registro de campos necesarios en relación con la base de datos.<br>• Ingreso de información conforme a los campos requeridos y sus respectivas validaciones. | |
| | | Modulo- Perfil usuario | HUU3 | Editar perfil | • Implementar métodos para visualizar el perfil, actualizar los campos necesarios y actualizar el avatar.<br>• Implementar conexión con el servicio Cloudinary para el almacenamiento de imágenes.<br>• Validación de usuarios existentes en la base de datos. | |
| SB002 | Gestión de carreras y semestres | Modulo- Gestión de carreras | HU004 | Gestionar | • Diseño e implementación de métodos para crear, ver, actualizar, activar o desactivar las carreras. | 40H |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | carreras | • Validación de los campos, para su respectiva creación y actualización.<br>• Verificación del estado en el que se encuentra dicha carrera (activado-desactivado). | |
| | | Modulo- Gestión de semestres. | HU005 | Gestionar semestres | • Diseño e implementación de métodos para crear, ver, actualizar, activar o desactivar semestres.<br>• Validación de los campos, para su respectiva creación y actualización.<br>• Verificación del estado en el que se encuentra dicho semestre (activado-desactivado). | |
| SB003 | Gestión de Materias y recursos educativos | Modulo-Gestión Materas | HU006 | Gestionar Materias | • Ejecución de métodos para crear, ver, actualizar, activar o desactivar materias en el caso que se den de baja del semestre.<br>• Validación de los datos requeridos.<br>• Verificación que el registro sea único. | 40H |
| | | Modulo-Gestión recursos educativos | HU007 | Gestionar recursos educativos | • Diseño e implementación del método que permita subir y descargar archivos pdf.<br>• Validación en el tamaño de la documentación al subir. | |

| | | | | | | |
|---|---|---|---|---|---|---|
| SB004 | Gestión de preguntas | Modulo-Preguntas | HU009 | Gestionar las preguntas e inquietudes de usuarios. | • Generar varios métodos que permita crear, ver, actualizar y eliminar comentarios.<br>• Validar los campos que se requiere al momento de crear y actualizar. | 20 H |
| SB005 | Pruebas del *backend* | • Pruebas unitarias.<br>• Prueba de rendimiento.<br>• Prueba de aceptación. | | | | 30H |
| SB006 | Despliegue del *backend* | • Despliegue del *backend* en Heroku. | | | | 10H |
| Documentación | | • Trabajo de Integración Curricular.<br>• Anexos. | | | | 20H |
| **TOTAL** | | | | | | **240 H** |

## Pruebas

Una vez finalizado la fase de codificación, se ha implementado la ejecución de pruebas unitarias, carga y estrés, con el objetivo de verificar calidad y rendimiento del *backend*.

### Pruebas unitarias

A continuación, se muestra varios apartados, cado uno con los métodos propuesto, a los cuales se ha realizado las pruebas unitarias verificando si no existe algún error.

Las figuras que van desde la **Fig.** a la **Fig. 8**: muestra las funciones codificadas, las cuales permiten verificar si existe algún error aislando una parte del código para comprobar que funcione perfectamente.

- Método “visualizar semestres”.

```php
public function test_ver_semestres()
{
    $user = User::Factory() -> make([
        'role_id' => 2
    ]);
    $request = $this->actingAs($user)->get('/api/v1/semestres/admin');
    $request -> assertStatus(200);
}
```

**Fig. 1:** Prueba unitaria para el Método de visualizar semestres.

```
PS C:\Users\Alexander\Desktop\TESIS BACKEND Y FRONT END\backendedupoli> php artisan test
Warning: TTY mode is not supported on Windows platform.

  PASS  Tests\Unit\PruebasTest
  ✓ ver semestres

  Tests:  1 passed
  Time:   5.89s

PS C:\Users\Alexander\Desktop\TESIS BACKEND Y FRONT END\backendedupoli>
```

**Fig. 2:** Resultados de la Prueba unitaria para el Método de visualizar semestres.

- Método “visualizar materias”.

```php
public function test_ver_materias()
{
    $user = User::Factory() -> make([
        'role_id' => 2
    ]);
    $request = $this->actingAs($user)->get('/api/v1/materias/adminE');
    $request -> assertStatus(200);
}
```

**Fig. 3:** Prueba unitaria para el Método de visualizar materias.

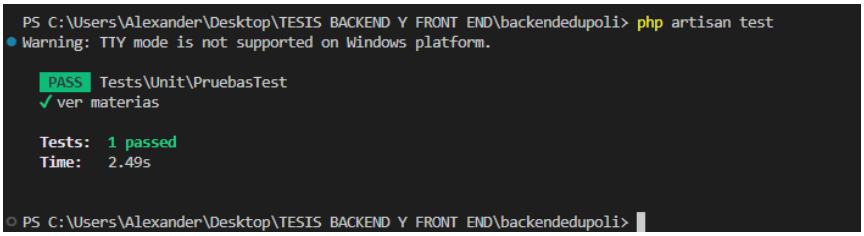



**Fig. 4:** Resultados de la Prueba unitaria para el Método de visualizar materias.

- Método “visualizar documentos”.

```php
public function test_ver_documentos()
{
    $user = User::Factory() -> make([
        'role_id' => 2
    ]);
    $request = $this->actingAs($user)->get('/api/v1/documentos/admin');
    $request -> assertStatus(200);
}
```

**Fig. 5:** Prueba unitaria para el Método de visualizar documentos.

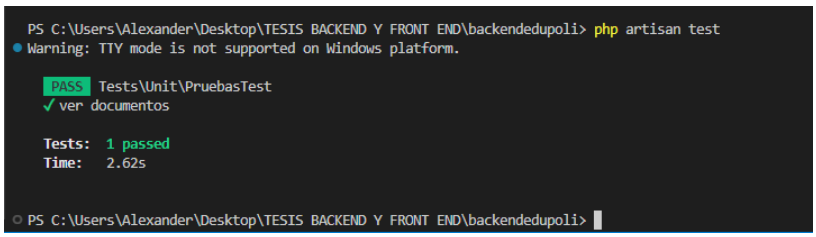
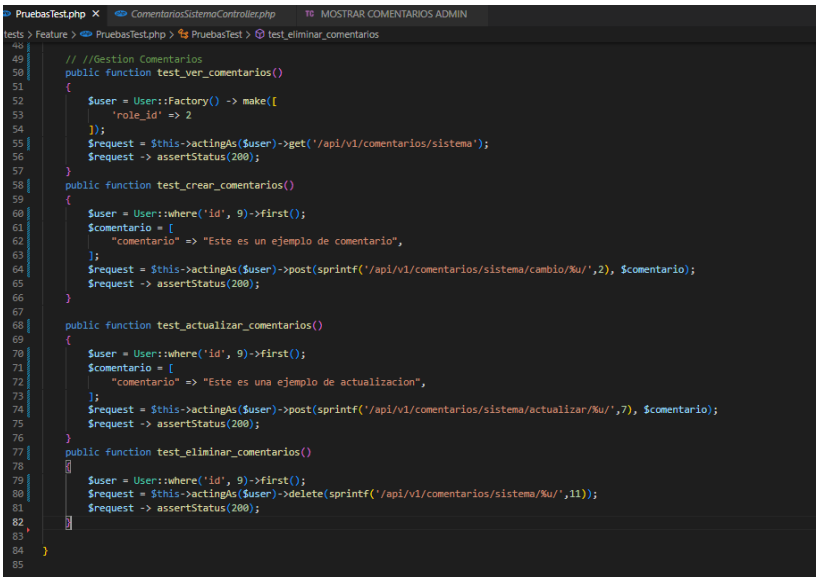



**Fig. 6:** Resultados de la Prueba unitaria para el Método de visualizar materias.

- Resultados “gestión de comentarios”.



```php
// //Gestion Comentarios
public function test_ver_comentarios()
{
    $user = User::Factory() -> make([
        'role_id' => 2
    ]);
    $request = $this->actingAs($user)->get('/api/v1/comentarios/sistema');
    $request -> assertStatus(200);
}
public function test_crear_comentarios()
{
    $user = User::where('id', 9)->first();
    $comentario = [
        "comentario" => "Este es un ejemplo de comentario",
    ];
    $request = $this->actingAs($user)->post(sprintf('/api/v1/comentarios/sistema/cambio/%u/',2), $comentario);
    $request -> assertStatus(200);
}

public function test_actualizar_comentarios()
{
    $user = User::where('id', 9)->first();
    $comentario = [
        "comentario" => "Este es una ejemplo de actualizacion",
    ];
    $request = $this->actingAs($user)->post(sprintf('/api/v1/comentarios/sistema/actualizar/%u/',7), $comentario);
    $request -> assertStatus(200);
}
public function test_eliminar_comentarios()
{
    $user = User::where('id', 9)->first();
    $request = $this->actingAs($user)->delete(sprintf('/api/v1/comentarios/sistema/%u/',11));
    $request -> assertStatus(200);
}
}
```

**Fig. 7:** Pruebas unitarias para la gestión de comentarios.

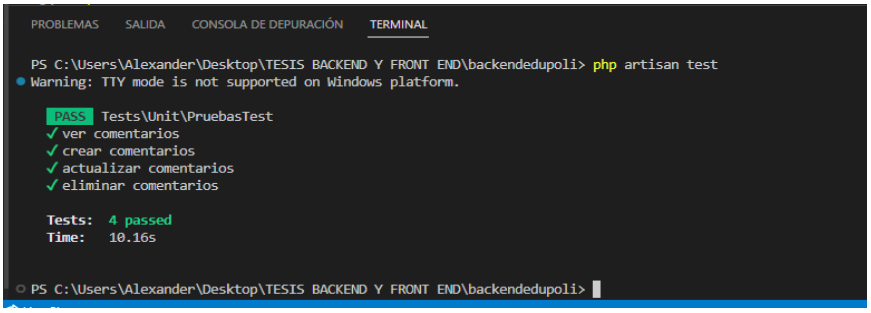



**Fig. 8:** Resultado de las Pruebas unitarias para la gestión de comentarios.

**Pruebas de carga**

A continuación, se muestra las pruebas de carga realizadas a los métodos considerados los más importantes.

Las figuras que van desde la **Fig. 9:** a la **Fig.** muestran los comandos y resultados, con los cuales se verifica el comportamiento cuando hay 800 peticiones con 800 usuarios.

- Método “visualizar semestres”.


**Fig. 9:** Comando para verificar la prueba de carga en el método de visualizar semestres.

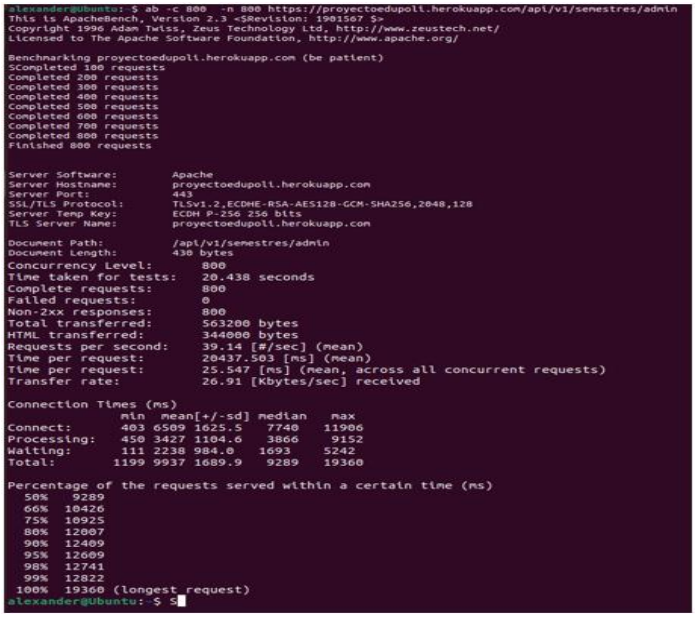



**Fig. 10:** Resultado de la prueba de carga en el método de visualizar semestres.

- Método “visualizar materias”.

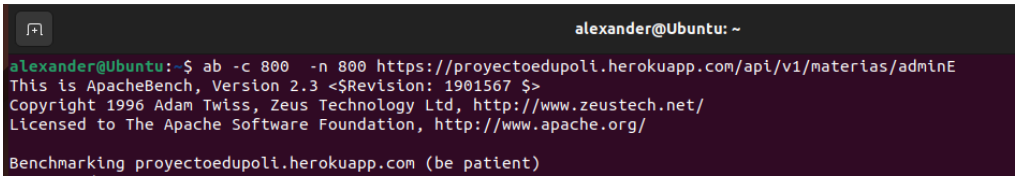



**Fig. 11:** Comando para verificar la prueba de carga en el método de visualizar materias.

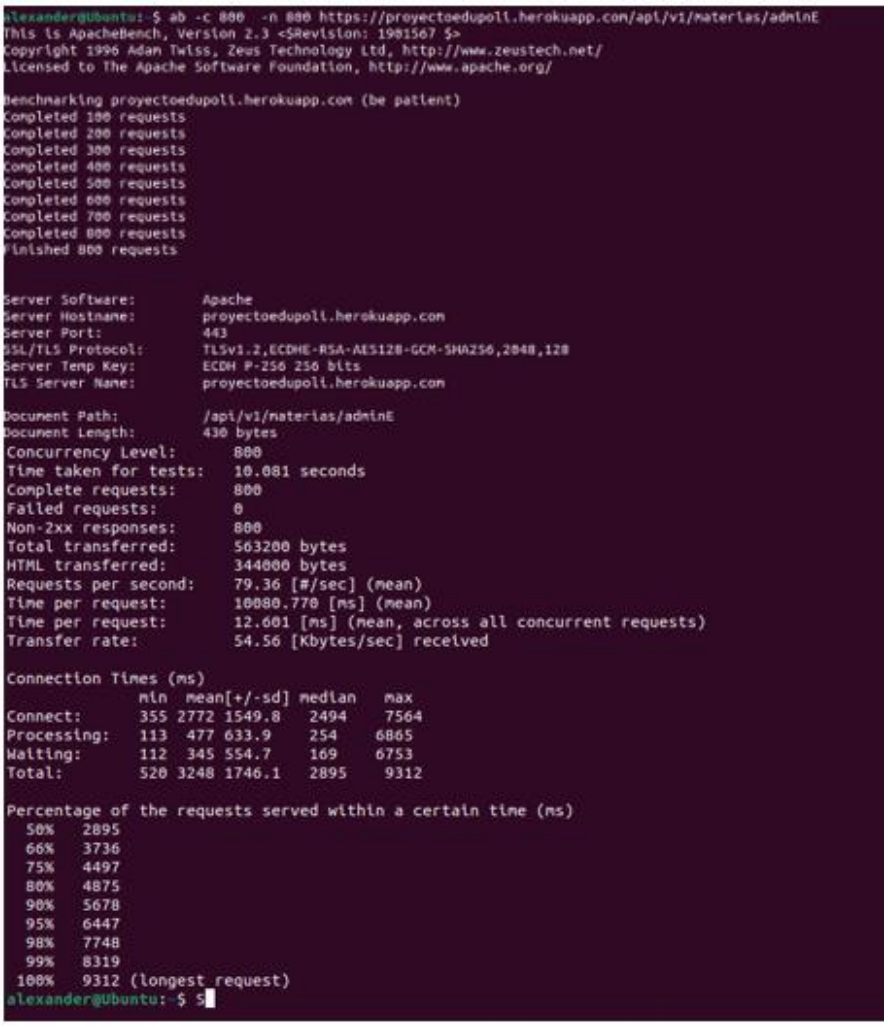



**Fig. 12:** Resultado de la prueba de carga en el método de visualizar semestres.

- Método “visualizar documentos”.

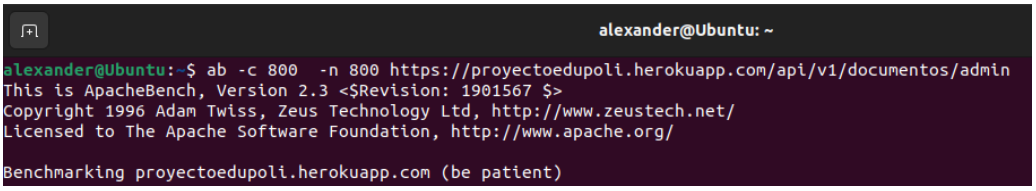



**Fig. 13:** Comando para verificar la prueba de carga en el método de visualizar documentos.

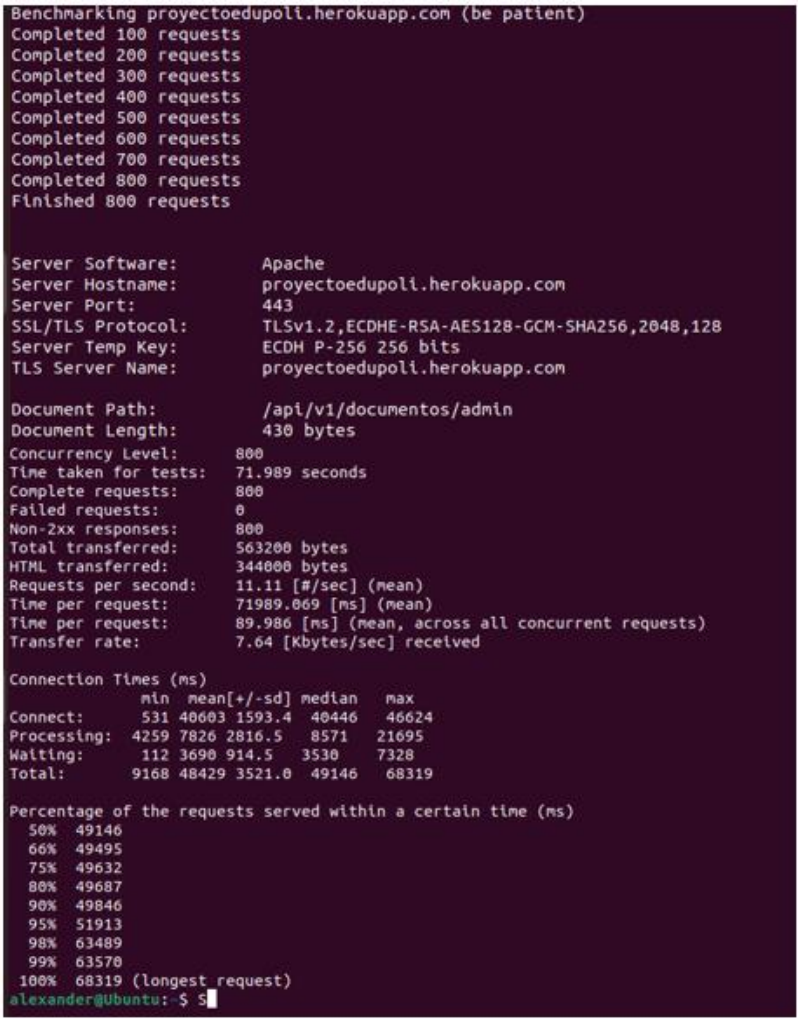



**Fig. 14:** Resultado de la prueba de carga en el método de visualizar documentos.

- Método “visualizar comentarios”.


**Fig. 15:** Comando para verificar la prueba de carga en el método de visualizar comentarios.

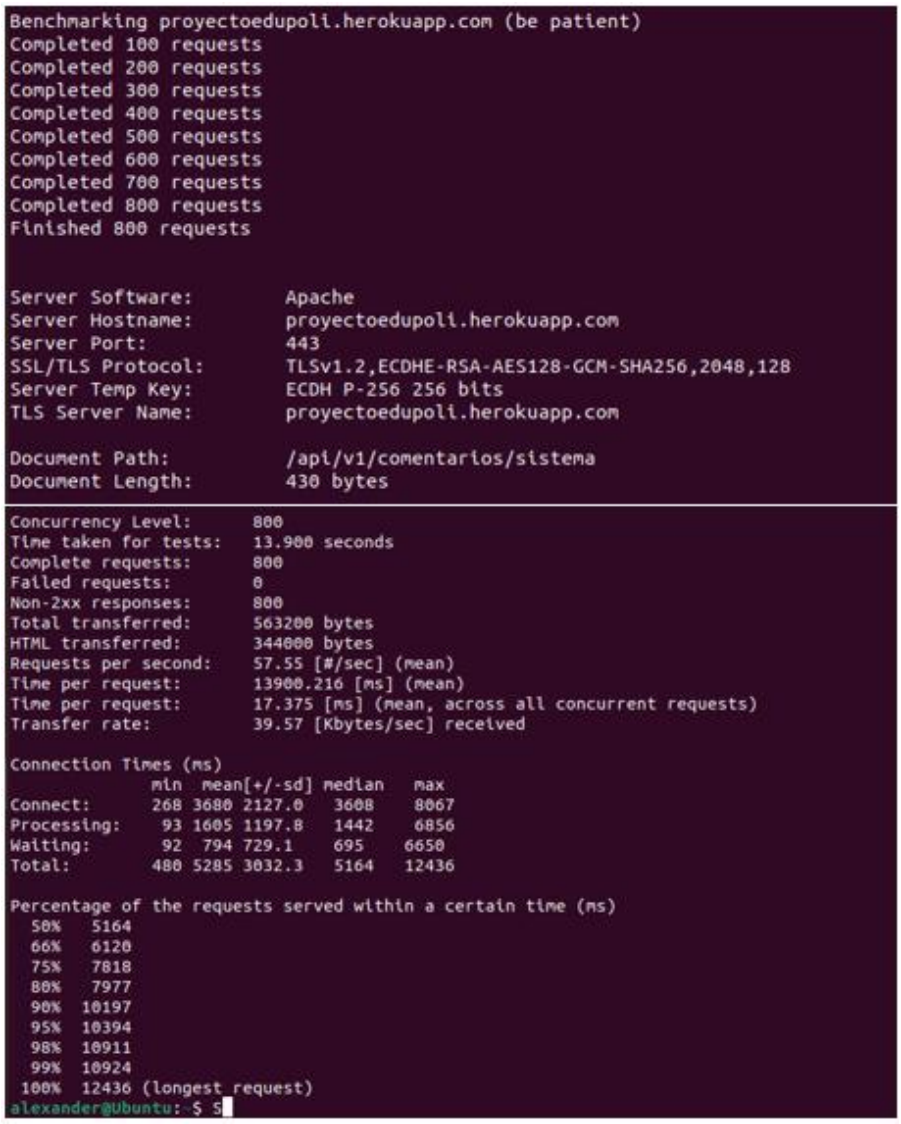



**Fig. 16:** Resultado de la prueba de carga en el método de visualizar comentarios.

**Pruebas de estrés**

A continuación se muestra las pruebas realizadas para lograr verificar el nivel máximo de ingreso de usuarios y peticiones que logra soportar el sistema web.

En la **Fig. 17** se muestra el número de estudiantes actualmente matriculados, No obstantes las figuras que van desde la **Fig. 18** hasta la **Fig. 21** muestran los comando y resultados y evidencias, que permiten estresar el método, logrando detectar el alcance de ingreso de usuarios y solicitudes con relación a cada uno de los métodos propuestos

- Estudiantes actualmente Matriculados.

| Nro. | Codcar | Carrera | AspCarAcep | AspCarMatr | TotalIns | TotalMatr | TotalMatrH | AnulaMatric | TotalMatrProv |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 236 | (RRA20) AGUA Y SANEAMIENTO AMBIENTAL | 30 | 29 | 97 | 97 | 0 | 1 | 0 |
| 2 | 234 | (RRA20) DESARROLLO DE SOFTWARE | 84 | 82 | 220 | 220 | 0 | 1 | 0 |
| 3 | 237 | (RRA20) ELECTROMECÁNICA | 44 | 43 | 167 | 167 | 0 | 2 | 0 |
| 4 | 249 | (RRA20) PROCESAMIENTO DE ALIMENTOS | 0 | 0 | 10 | 10 | 0 | 0 | 0 |
| 5 | 250 | (RRA20) PROCESAMIENTO INDUSTRIAL DE LA MADERA | 0 | 0 | 6 | 6 | 0 | 0 | 0 |
| 6 | 230 | (RRA20) REDES Y TELECOMUNICACIONES | 39 | 38 | 164 | 164 | 1 | 1 | 0 |
| 7 | | TOTAL TECNOLOGICA | 197 | 192 | 664 | 664 | 1 | 5 | 0 |

**Fig. 17:** Estudiantes actualmente matriculados.

- Método “visualizar semestres”.

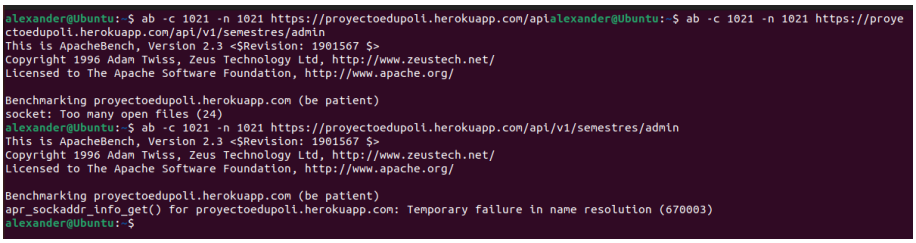



**Fig. 18:** Comando para verificar la prueba de estrés en el método de visualizar semestres.

- Método “visualizar materias”.

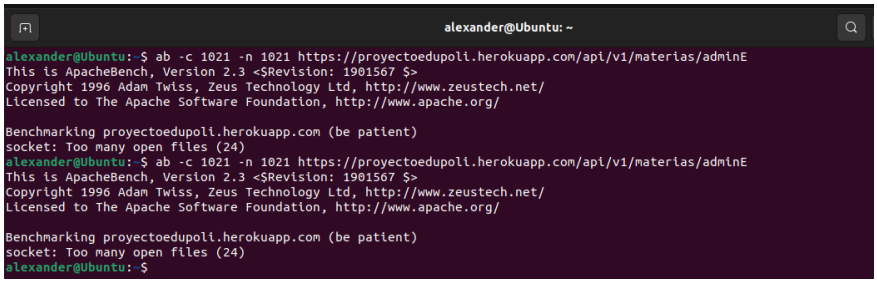



**Fig. 19:** Comando para verificar la prueba de estrés en el método de visualizar comentarios.

- Método “visualizar documentos”.

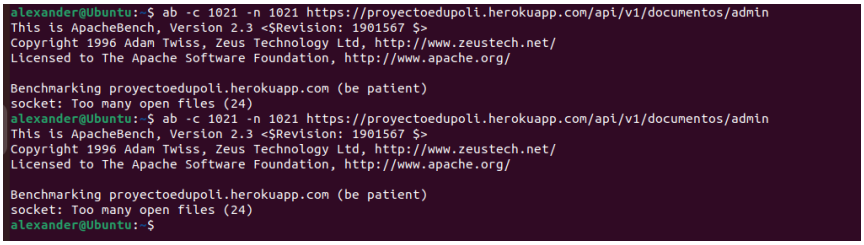



**Fig. 20:** Comando para verificar la prueba de estrés en el método de visualizar documentos.

- Método “visualizar comentarios”.


**Fig. 21:** Comando para verificar la prueba de estrés en el método de visualizar comentarios.

# ANEXO III

A continuación, se muestra el enlace con el video del manual de usuario del componente *backend* del sistema web.

https://youtu.be/nDEoT-Be5H8

# ANEXO IV

A continuación, se muestra las credenciales según los roles que asigna el *backend*, así como el enlace de GitHub en donde se encuentra alojado el código fuente y en la parte del README, se indica los pasos a seguir para desplegar el *backend* en heroku.

## Credenciales para el ingreso haci el *backend*

Para tener la accesibilidad al *backend*, ingresar por la URL que se indica a continuación.

https://proyectoedupoli.herokuapp.com/api/documentation

Credenciales para el perfil administrador:

- Correo del usuario 1: adrianchicaiza3@gmail.com
- Contraseña 1: Secret123*
- Correo del usuario 2: fernandotupiza10@hotmail.com
- Contraseña 2: Secret1*

Credenciales para el perfil estudiante:

- Correo del usuario: fernandotupiza03@gmail.com
- Contraseña: Secret123*

## Repositorio del código fuente del *backend*

El código fuente del componente *backend* se encuentra alojado en el repositorio de GitHub, para acceder al código fuente se lo puede realizar ingresando a la siguiente URL.

https://github.com/FernandoTupiza/Componente_Backend_Tesis.git