ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UNA APLICACIÓN WEB PARA FOMENTAR HÁBITOS SALUDABLES

DESARROLLO DE UN BACKEND

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

BRYAN ALFONSO TANDAZO NARVAEZ

DIRECTOR: LORENA ELIZABETH CHULDE OBANDO

DMQ, marzo 2024

CERTIFICACIONES

Yo, **BRYAN ALFONSO TANDAZO NARVAEZ** 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.

BRYAN ALFONSO TANDAZO NARVAEZ

bryan.tandazo@epn.edu.ec

bry.tandazo@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por **BRYAN ALFONSO TANDAZO NARVAEZ**, bajo mi supervisión.

LORENA ELIZABETH CHULDE OBANDO

DIRECTORA

lorena.chulde@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.

Bryan Alfonso Tandazo Narvaez

DEDICATORIA

El presente proyecto se lo dedico a mis padres, quienes han sido un apoyo para culminar esta etapa en mi vida, me brindaron la oportunidad de seguir formándome como profesional. A mis hermanos ya que fueron inspiración para seguir sus pasos, por brindarme los recursos y herramientas necesarias para poder cumplir esta meta, por confiar en mis capacidades a pesar de los malos momentos en donde sentía que iba a decaer estuvieron ahí animándome para no rendirme. Es gratificante poder dedicarles este proyecto como respuesta a las expectativas que tuvieron hacia mí.

Bryan Alfonso Tandazo Narvaez

AGRADECIMIENTO

Agradezco a mis compañeros con quienes compartí en el aula de clases y las malas noches que pasamos para sacar adelante los proyectos. A mis profesores por compartirme su experiencia y conocimiento para ser cada día mejor. Al personal de apoyo de la ESFOT quienes se preocupaban por ofrecer un entorno agradable y por último a la “Escuela Politécnica Nacional” por brindarme la oportunidad de estudiar en esta bonita institución.

Bryan Alfonso Tandazo Narvaez

ÍNDICE DE CONTENIDO

RESUMEN

El proyecto Vitalzure propone el desarrollo de una aplicación web, cuyo objetivo es el de fomentar hábitos saludables para los estudiantes. La lógica que maneja la aplicación cumple con las necesidades de los diferentes roles que son administrador, organizador y estudiante. Los tres roles de usuarios pueden iniciar sesión por medio de un correo electrónico y contraseña; también pueden actualizar su información de perfil, avatar y recuperar su contraseña. El administrador es quien se encarga de la gestión de usuarios con rol organizador, cuando se crea una cuenta de usuario las credenciales del organizador son enviadas por correo, de igual manera puede eliminar a los estudiantes en caso de un comportamiento inapropiado. Cabe mencionar que los estudiantes pueden crear su cuenta y ver las publicaciones, además, pueden indicar como favorita la publicación de su interés, calificar, escribir un comentario y ver el ranking de las mejores publicaciones promediadas. El usuario organizador es quien gestiona las publicaciones, puede mirar las calificaciones y comentarios de sus publicaciones.

La información se encuentra en el gestor de base de datos MySQL quien maneja los datos de forma relacional y se encuentra alojado en la nube, es proporcionado por Alwaysdata. Para el manejo de correo se utiliza el servicio de Mailtrap y las imágenes de perfil de usuario se almacenan en Dropbox.

La metodología que se utilizó para el desarrollo es SCRUM, ya que es adaptable, iterativo es decir se dividió el desarrollo en partes pequeñas denominadas “sprints” y se definió roles dentro del equipo.

PALABRAS CLAVE: Laravel, Vercel, Estudiantes, Hábitos Saludables

ABSTRACT

The Vitalzure project proposes the development of a web application aimed at promoting healthy habits for students. The system logic caters to the needs of different roles: administrator, organizer, and student. All three user roles can log in using an email and password; they can also update their profile information, avatar, and recover their password. The administrator is responsible for managing users with the organizer role; when a user account is created, the organizer's credentials are sent via email, and they can also remove students in case of inappropriate behavior. It is worth mentioning that students can create their account, view posts, mark posts as favorites, rate, write comments, and view the ranking of the best-rated posts. The organizer manages the posts and can view the ratings and comments on their posts.

The information is stored in a MySQL database management system, which handles data relationally and is hosted in the cloud, provided by Alwaysdata. Mailtrap is used for email management, and user profile images are stored in Dropbox.

The methodology used for development is SCRUM, as it is adaptable and iterative. The development was divided into small parts called "sprints," and roles were defined within the team.

KEYWORDS: Laravel, Vercel, Students, Healthy Habits

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En los tiempos modernos la falta de preocupación en el tema de la salud y los malos hábitos que llevan los estudiantes de bachillerato se convierte en un problema. Durante la adolescencia, los estudiantes atraviesan una etapa de vulnerabilidad, los malos hábitos y la influencia negativa del entorno conduce a la adopción de comportamientos poco saludables, como una dieta desequilibrada, la carencia de ejercicio físico regular y el manejo inadecuado del estrés, que son perjudiciales para la salud. Estos comportamientos pueden resultar en un problema de salud a mediano y largo plazo, lo cual puede causar enfermedades cardiovasculares, obesidad, estrés crónico y otras condiciones relacionadas con el estilo de vida. Además, la poca conciencia de la importancia de los hábitos saludables puede contribuir a la resistencia de realizar cambios positivos, manteniendo a las personas en un ciclo perjudicial para su bienestar general [1].

La UNESCO actualmente establece metas para el Desarrollo Sostenible, siendo el tercer objetivo específico la promoción del bienestar para todos. Por tal razón se involucra a jóvenes y adultos a mejorar la calidad de vida, como participar en iniciativas que aseguren una alimentación adecuada para aquellos que carecen de información sobre buenos hábitos de alimentación. Además, se enfatiza la importancia de realizar ejercicio regular y fomentar esta práctica en el entorno, especialmente entre niños y jóvenes. Asimismo, se destaca la necesidad de concientizar y normalizar las enfermedades de salud mental, cuidando tanto nuestra salud emocional como la psicológica [2].

Es por esta razón, que se propone el proyecto de la aplicación web para Fomentar Hábitos Saludables para los estudiantes, buscando estimular la adopción de hábitos saludables por medio de publicaciones sobre Hábitos alimenticios, Ejercicios físicos y Salud mental. Es una herramienta que permite a los estudiantes participantes tener un espacio donde compartir sus experiencias sobre el contenido de cada publicación y como aporto en su vida para mejorar su bienestar, permite calificar las publicaciones para mostrar su agrado y consultar las mejores promediadas y finalmente para tener un mejor acceso permite añadir a las publicaciones como favoritas.

1.1 Objetivo general

Desarrollar una Aplicación Web mediante el framework Laravel para fomentar hábitos saludables a los jóvenes de bachillerato

1.2 Objetivos específicos

1. OE1 Identificar los requerimientos del sistema mediante las historias de usuario para estructurar correctamente el sistema.
2. OE2 Codificar los diferentes endpoints mediante el framework Laravel en la implementación del APIREST para la aplicación.
3. OE3 Verificar que las funcionalidades de la aplicación se ejecuten correctamente mediante pruebas funcionales para cumplir con los criterios de aceptación de los usuarios.
4. OE4 Desplegar a producción el componente desarrollado Backend mediante el servicio Vercel para el alojamiento de la aplicación.

1.3 Alcance

La aplicación maneja tres roles:

- **El Administrador:** Como administrador podrá gestionar estudiantes y organizadores. Mediante el módulo publicación podrá gestionar las publicaciones. Gestionar perfil de usuario, avatar y contraseña.
- **Organizador:** Como organizador podrá gestionar las publicaciones (Hábitos alimenticios, Ejercicios físicos, Salud mental). Consultar calificaciones y comentarios sobre sus publicaciones.
- **Estudiante:** Como estudiante podrá registrarse en la aplicación, visualizar todas las publicaciones, podrá añadir como favoritas las de su interés, calificar, comentar y consultar las mejores publicaciones promediadas.

Existen múltiples técnicas utilizadas para transmitir conocimiento de hábitos saludables como: campañas en televisión, folletos entregados en la calle, libros de prevención, revistas de salud y hasta páginas web de instituciones internacionales (OMS u OPS-Organización Panamericana de la Salud). Sin embargo, en los últimos años el creciente uso de la tecnología rebela la importancia de esta dentro de la educación (también en temas de salud) [3].

Bajo este escenario se propone desarrollar el Backend de una aplicación web para fomentar hábitos saludables, se emplearán diversas herramientas y se utilizara el patrón arquitectónico (MVC). Además, se llevarán a cabo pruebas de funcionamiento con el objetivo de asegurar la escalabilidad y robustez de la aplicación antes de su despliegue en producción. Con el fin de mejorar la productividad durante el desarrollo, se aplicará la metodología Ágil Scrum. Esta metodología divide las tareas en partes más pequeñas conocidas como Sprints y facilita el tiempo que se utilizará para realizar cada tarea.

1.4 Marco Teórico

Para entender de mejor manera el contenido del proyecto, se explicará de manera general algunos conceptos que maneja la aplicación para fomentar hábitos saludables.

Metodología

La aplicación de una metodología permite maximizar la productividad en el desarrollo de software, puesto que no solo toma en cuenta los procesos o herramientas a utilizar, sino que se preocupa por la motivación y el entorno adecuado para la realización de las tareas; además, utilizar una metodología para la programación del sistema permite desarrollar un producto de manera fácil y ágil, reduciendo el nivel de dificultad en organizar y asignar tareas, acelera el proceso obteniendo el resultado esperado. En la actualidad existen dos tipos de metodologías: metodologías ágiles y tradicionales [3].

Aplicación Web

Es una aplicación que se ejecuta en un navegador, permitiendo el intercambio de información y obtener servicios de manera remota. Actualmente es muy utilizaba en diferentes ámbitos como el e-comerse, carritos de compra, mensajería instantánea y redes sociales.

Tiene una arquitectura cliente-servidor de manera que el cliente realiza peticiones al servidor, cuando interactúa con los botones, cuadros, o da clic en algún enlace. Por otro lado, la tarea del servidor es procesar las peticiones del cliente y enviar un resultado de vuelta, entre las solicitudes este manejo de datos se los hace en las bases de datos [4].

Arquitectura Web

Es un proceso que se encarga de planificar el diseño que tendrá un sitio web antes que sea desarrollado y codificado, esto se incluye los componente técnicos, funcionales y visuales. La estructura conceptual con la que trabaja es la world wide web (www) que permite la comunicación de usuarios y la interacción de diferentes equipos.

Para transferencia de datos, trabaja con tres protocolos, para este sistema se implementará el HTTPS el cual es más seguro porque los datos son encriptados. Este protocolo trabaja con tecnologías web como HTML, CSS, XML y con las URL se puede navegar fácilmente [5].

Backend

Trabaja con la información de usuarios y se encarga de diseñar la lógica y estructura del sistema y para el usuario final no es visible, pero si es consumido por el frontend. Funciona del lado del servidor y responde a las peticiones de los usuarios.

Los procesos del backend son invisibles a los ojos de los usuarios sin embargo tener un buen desarrollo en este campo nos garantiza una aplicación funcional, rápida y segura dando como resultado un cliente satisfecho [6].

API

Es una interfaz que permite la comunicación entre dos aplicaciones con el objetivo de compartir información permitiendo así ahorrar dinero y tiempo ya que no es relevante como manejan la información las aplicaciones. Por lo tanto, un API permite flexibilidad, habilita el acceso a más información, es seguro y es fácil de controlar.

Una manera de aprovechar todos estos beneficios fue crear una arquitectura denominada REST la cual tiene características como la arquitectura cliente servidor, un sistema sin estado, es decir, las solicitudes de los usuarios no se almacenan en el servidor, el almacenamiento lo hace en la caché, un sistema de capas para ofrecer diversas funcionalidades y la interfaz para la conexión de dos sistemas con el objetivo de intercambiar información. Cuando se cumplen con estas características se las denomina RESTful [7].

ORM

Modelo que permite estructurar una base datos relacional, se compone de un conjunto de herramientas denominado Relational Database Management System (RDBMS) que simplifica y acelera el desarrollo de una aplicación.

La estructura para manejar los datos se vincula a una entidad lógica para la ORM y cuando se realiza las acciones CRUD se los hace en la base de datos física por medio del ORM [8].

Eloquent es un ORM de Laravel es decir permite manejar una base de datos como un objeto, ahorrando así tiempo y facilitando el trabajo cuando se está manipulando las bases de datos. Adicionalmente cuenta con el acceso a las relaciones por medio de propiedades de los objetos, funciona con el patrón “Active Record” el cual tiene métodos como save, update o delete [9].

Laravel

Es un framework de código libre proveniente de Taylor Otwell. Esta herramienta tiene como objetivo el desarrollar proyectos de manera ordenada, simple y elegante.

Laravel cuenta con diferentes funcionalidades como:

- **Base de datos:** En la abstracción de datos, la información se maneja con la arquitectura ORM "Eloquent" el cual permite realizar consultas y actualizar datos.
- **Autenticación:** Laravel proporciona una base para la autenticación segura que se basa en sesiones.
- **Queues:** Para una mejor optimización cuenta con trabajos en segundo plano los cuales son capaces de realizar tareas pesadas, como enviar correos electrónicos o generar informes en tiempos rápidos.
- **Web Sockets:** Permite crear aplicaciones en tiempo real, por lo que admite al usuario tener una mejor experiencia en el desarrollo [10].

2 METODOLOGÍA

La aplicación de una metodología ayuda a orientar el desarrollo de un estudio. Implementa métodos, herramientas y técnicas que facilitan la obtención de información para general el conocimiento y así tomar mejores decisiones [11].

El presente trabajo utiliza una metodología denominada estudio de caso, método que trabaja con la recopilación de datos para determinar una solución apropiada, teniendo en cuenta la experimentación de la problemática por medio de la indagación de las circunstancias o situaciones, para este caso sobre la falta de información de hábitos saludades y la necesidad de conocer la importancia sobre los buenos hábitos alimenticios, los ejercicios físicos y la salud mental. De manera que permitan tener una vida saludable [12].

2.1 Metodología de Desarrollo

Surge como un cambio en el cual se buscaba reemplazar el desarrollo de manera lineal por una más funcional, para lograr este objetivo se propuso un manifiesto el cual mencionaba que son más importantes las personas y como interactúan que los procesos y herramientas. Menciona que el funcionamiento es más importante que la documentación exhaustiva, involucrar al cliente antes que una negociación y tener respuesta ante los cambios, permitiendo un trabajo colaborativo, donde los integrantes se sienten cómodos dentro del proyecto; siendo flexible para adaptarse a los cambios que propone el cliente.

Por otro lado, Scrum facilita el desarrollo del proyecto, ya que lo divide en partes llamadas Sprints, que son iteraciones donde se cumple con objetivos de funcionalidad para el cliente, permitiendo observar cómo va el producto y de ser necesario proponer cambios en ese instante. Los requerimientos del cliente se clasifican en prioridades (alta, media o baja), lo que hace que esta metodología sea perfectamente adaptable para cumplir con los cambios existentes [13].

Scrum es el método con el cual se va a trabajar ya que cuenta con la gestión de trabajo en equipo, divide las tareas en partes que son denominados Sprints y se define el tiempo en el cual se realizaran; además, puede estar sujeta a cambios en todo momento. Para seguir de manera óptima esta metodología es importante

definir el rol de cada integrante de manera que conozca las responsabilidades. A continuación, se explica cada rol dentro del equipo Scrum.

Roles

Scrum maneja el concepto de trabajo en equipo, otorgando responsabilidades a cada integrante del equipo, de manera que se llegue a cumplir con los objetivos de manera eficiente.

- **Product Owner:** Es la persona encargada de darle valor al producto, es decir es quien se encarga de transmitir lo que el cliente desea. Entre sus funciones esta establecer el objetivo del producto, se encarga de crear los elementos del Product Backlog. La Ing. Lorena Chulde cumple con este rol de aclarar y revisar las ideas de la lógica del negocio.

- **Scrum Master:** Es la persona responsable de guiar al equipo, ayuda en la teoría y práctica de Scrum, se lo considera el líder del proyecto. Este rol de igual forma lo asume la Ing. Lorena Chulde quien revisa las tareas que debe realizar el equipo guiando el proceso de desarrollo para cumplir con los objetivos.

- **Developer Team:** Hace referencia al equipo de desarrollo se caracterizan por tener habilidades técnicas de manera que puedan cumplir con el objetivo común del desarrollo del producto, el equipo se organiza de manera que cada uno elige sus actividades para completar su backlog en un tiempo estimado [14]. El producto final de este proyecto es el desarrollo del componente Backend y el responsable es Bryan Tandazo

En la **Tabla 2.1** se muestra la información del equipo Scrum para el desarrollo del proyecto con el rol correspondiente.

Tabla 2.1. Equipo Scrum

Rol	Integrantes
Product Owner	Ing. Lorena Chulde
Scrum Master	Ing. Lorena Chulde
Developer Team	Sr. Bryan Tandazo

Artefactos

Es la información que utiliza el equipo Scrum para detallar el producto, estas herramientas ayudan en la transparencia, inspección y adaptación para asegurar la calidad del proyecto [15].

Recopilación de Requerimientos

Es un proceso mediante el cual se identifica, documenta y gestiona los requisitos o necesidades del negocio donde el Product Owner y el Scrum Master están involucrados para cumplir con las expectativas del cliente. Estos requisitos son las funciones, características, cualidades que el sistema o proyecto debe cumplir [16]. La recopilación de estos requerimientos se encuentra a manera de lista, esta se puede encontrar en el **ANEXO II** en la sección de Recopilación de requerimientos, exactamente la **Tabla I**. En la **Tabla 2.2** se muestra los tres primeros requerimientos

Tabla 2.2. Recopilación de requerimientos.

ID-RR	Enunciado del item
RR-01	Como: Administrador, Organizador y Estudiante Quiero: Iniciar sesión Para: Acceder a las funcionalidades según el rol correspondiente
RR-02	Como: Administrador, Organizador y Estudiante Quiero: Actualizar mi perfil de usuario. Para: Cambiar mi información personal.
RR-03	Como: Administrador, Organizador y Estudiante Quiero: Cambiar mi contraseña. Para: Mejorar mi nivel de seguridad

Historias de Usuario

Las historias de usuario son una descripción general que usan términos entendibles para el cliente sobre la aplicación, en ellas se mencionan el usuario, el riesgo, la prioridad, descripción, alcance y los criterios de aceptación [17]. En la **Tabla 2.3** se observa un ejemplo de la historia de usuario 1. La lista completa de las historias de usuario se encuentra en el **ANEXO II**, a partir de la **Tabla II.**

Tabla 2.3. Historia de usuario No.1

HISTORIA DE USUARIO	
Identificador (ID): HU01	**Usuario:** Administrador, Organizador y Estudiante.
Nombre de Historia: Servicio inicio de sesión	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 1	
Responsable: Bryan Tandazo	
Descripción: Como: Administrador, Organizador y Estudiante Quiero: Iniciar sesión Para: Acceder a las funcionalidades según el rol correspondiente	
Alcance: 1. La ruta /login utiliza el método POST para que se pueda ingresar un JSON con la información del usuario. 2. La ruta /logout utiliza el método POST e ingresa el token en Auth para salir del sistema.	
Observación: • La estructura del JSON es la siguiente: { "email":"email", "password":"secret" } • El token que se debe ingresar en Auth debe ser de tipo Bearer	
Criterios de aceptación: • Si el JSON está mal estructurado o utiliza un método diferente a POST responde con el error 405. • Si ingresa un correo o contraseña incorrecta lanza el siguiente mensaje “The provided credentials are incorrect”. • Si ingresa correctamente retorna un mensaje de “Successful authentication.”, se muestra la información del usuario y el token.	

Product Backlog

Es una lista que permite priorizar las funcionalidades del sistema, que se definieron en las historias de usuario. El Product Backlog del sistema se encuentra en el

ANEXO II, específicamente la **Tabla XVI.** La **Tabla 2.4** muestra los tres primeros ítems.

Tabla 2.4 Tres primeros ítems del Product Backlog

ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU01	Servicio inicio de sesión	1	Completa	Alta
HU02	Actualizar perfil de usuario	2	Completa	Media
HU03	Actualizar contraseña	1	Completa	Media

Sprint Backlog

Hace referencia a la lista de tareas o funcionalidades que el Developer Team se establece en un determinado tiempo para finalizar en cada Sprint del proyecto. Estas tareas pendientes se van obteniendo del Product backlog para la planificación del Sprint. En la **Tabla 2.5** se observa un ejemplo del Sprint Backlog y la tabla completa del mismo se encuentra en el **ANEXO II**, específicamente la **Tabla XVII**

Tabla 2.5. Sprint 0 del Proyecto Vitalzure

ID-SB	Nombre	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB-000	Configuración del entorno de desarrollo	N/A	N/A	• Configurar los implementos del IDE Visual Studio code para la creación de proyecto. • Configurar las variables de entorno de los servicios. • Diseño y creación de la base de datos.	20H

2.2 Diseño de la arquitectura

El sistema para fomentar hábitos saludables se organiza mediante el patrón arquitectónico de manera que se pueda establecer una estructura general y así implementar una buena interfaz y facilitar el inicio del diseño de un prototipo [17].

Arquitectura de datos

Es un conjunto de herramientas que permite gestionar la información a lo largo del ciclo de vida, se compone de herramientas, técnicas y estándares que ayudan a definir los procesos de captura, transforma y entrega de datos a los usuarios e identifica quien va a consumir la información [18].

La **Figura. 2.1** muestra el esquema de la base de datos, es decir las tablas donde se almacenarán los datos y como se presenta la información almacenada. Los usuarios serán los encargados de tener acceso a la información de cada una de ellas dependiendo del rol.

Figura. 2.1. Esquema de las tablas de la base de datos.

Patrón arquitectónico

El presente proyecto utiliza el patrón arquitectónico Modelo Vista Controlar (MVC) ya que permite dividir el código en tres componentes, facilitando así la administración y cambios sin que haya conflicto. Cada componente tiene una tarea especifica, es decir:

- **Modelo:** Se encarga de la lógica y el almacenamiento de datos.
- **Vista:** Hace referencia a la parte gráfica que el usuario final puede observar.
- **Controlador:** Tiene como objetivo manejar los datos y encargar de decidir qué información mostrar [19].

Aplicación Web

Para la elaboración del sistema web, en la **Figura. 2.2** se muestra la interacción de un usuario en donde por medio del controlador va actualizando la información y los cambios se ven reflejados en la vista y de la misma manera el modelo se ve afectado por el controlador el cual notifica alguna manipulación.

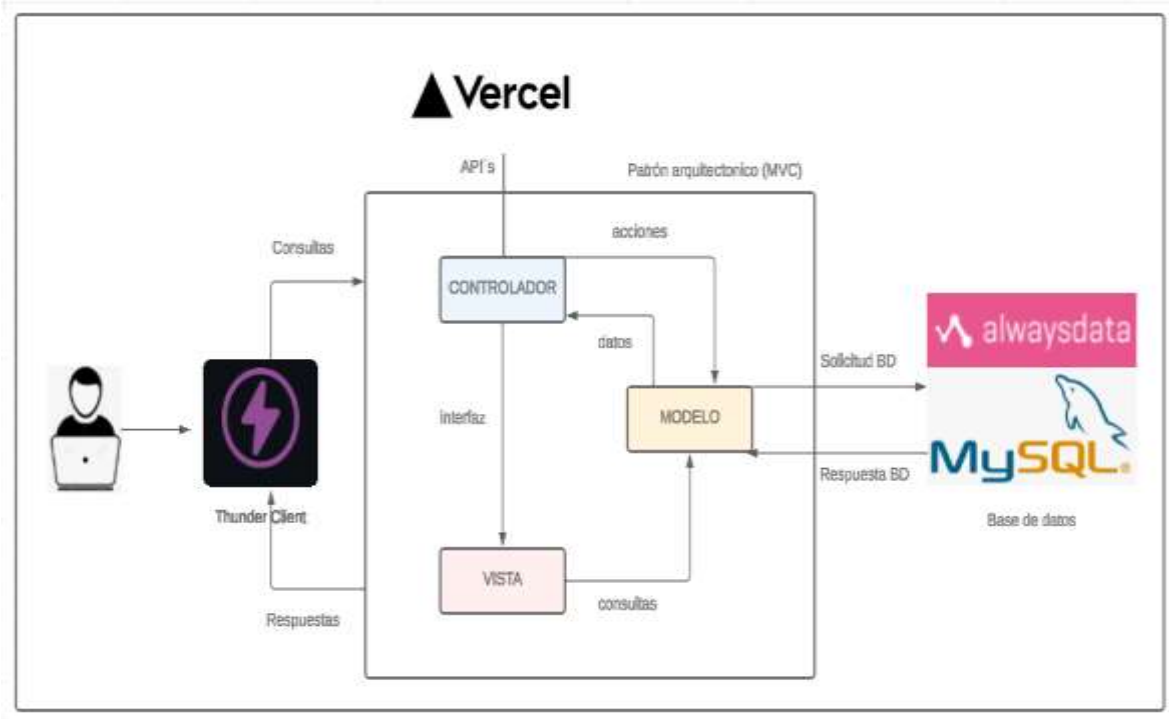


Figura. 2.2. Diseño de la arquitectura – Aplicación Web

Diagrama de componentes

La aplicación web se compone de cuatro componentes, en donde el cliente web es quien se encarga de realizar las peticiones al servidor web Vercel que es donde se aloja la aplicación de Laravel y finalmente para la persistencia de datos se utilizó MySQL, dando como resultado el diagrama que se muestra en la **Figura. 2.3**

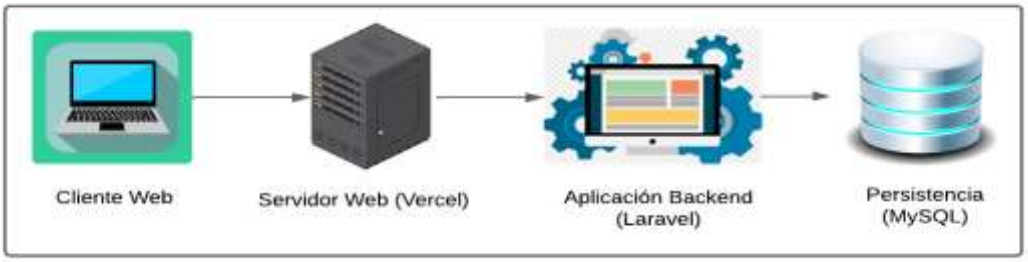


Figura. 2.3. Diagrama de componentes – Aplicación Web

2.3 Herramientas de desarrollo

Son programas que facilitan la creación, el mantenimiento, la depuración del software [20]. Los endpoints de la aplicación para fomentar hábitos saludables se obtienen de los requerimientos y son elaborados por estas herramientas.

Aplicación Web

El sistema web utiliza diferentes herramientas y servicios para el desarrollo como se muestra en la **Tabla 2.6**.

Tabla 2.6. Herramientas de la Aplicación Web

Herramienta	Justificación
Alwaysdata	Es una plataforma que trabaja con bases de datos relacional por ejemplo MySQL y permite que la información almacenada, pueda ser accesible por la aplicación.
Composer	Es un gestor de dependencias quien se encarga de instalar y mantener las librerías de la aplicación.
Dropbox	Es un servicio de almacenamiento que almacena las imágenes de perfil del usuario.
Git	Es un controlador de versiones quien permite revisar de manera sencilla los cambios del código.
Github	Es una plataforma en donde el código que se va desarrollando se va ir almacenando en repositorios y permite tener ramas para cada Sprint.
Laravel	Es un framework de PHP y ayuda en la creación de la aplicación ya que es

	organizado por carpetas y trabaja con MVC. Que se definió en el proyecto.
MySQL	Es un gestor de bases de datos y es el complemento que utiliza alwaysdata y Laravel para manejar los datos.
Thunder Client	Es un programa que ayuda en la realización de las consultas a la API y permite verificar el funcionamiento.
Vercel	Es un servicio de alojamiento el cual se va a utilizar para el despliegue a producción de la aplicación.

Librerías

La **Tabla 2.7**. indica la lista de librerías necesarias que utiliza la aplicación para su funcionamiento con la descripción de cada una.

Tabla 2.7. Librerías de la Aplicación Web

Librería	Descripción
Faker PHP	Es una librería que permite generar datos aleatorios, los cuales ayudan en el desarrollo de la aplicación.
Flysystem Dropbox.	Es un paquete que permite usar los diferentes métodos para el almacenamiento en Dorpbox.
Laravel Sanctum	Permite implementar la autenticación utilizando tokens de manera que cada acción pueda ser realizada dependiendo al rol del usuario que se definió en el proyecto.

Model Generator	Es un paquete con una estructura básica en la generación de seeders y factories.
Migration Generator	Permite la migración a la base de datos de manera que esta tenga relaciones ya que permite asignar llaves primerias o foráneas según el caso.

3 RESULTADOS

El proyecto inicia con el desarrollo del componente Backend del sistema, se lo realizará mediante Sprints, y en cada uno se muestra el desarrollo de las funcionalidades especificas establecidas en el Sprint Backlog

3.1 Sprint 0. Configuración del ambiente de desarrollo

Para cumplir con la meta del Sprint 0, se definió una serie de tareas para la correcta configuración, empezando por la configuración del IDE para generar un proyecto nuevo, de la misma manera se empieza con la elaboración de la base de datos (BD) en MySQL, se crean las migraciones de las tablas como los respectivos sedders y factories para cada tabla de manera que la aplicación pueda almacenar y trabajar con datos. y se procede con las pruebas respectivas de las consultas y consumo de información de la BD.

Lista de tareas

- Tarea 1. Diagrama de casos de uso
- Tarea 2. Entorno de desarrollo para Laravel
- Tarea 3. Configuración para la conexión de servicios.
- Tarea 4. Diseñar el modelo de la BD.
- Tarea 5. Crear las migraciones.
- Tarea 6. Definir la relación a nivel BD
- Tarea 7. Definir la relación a nivel Eloquent
- Tarea 8. Creación de Seeders y Factories
- Tarea 9. Registros de prueba de las tablas

Tarea 1. Diagrama de casos de uso

Previo al desarrollo, con la recopilación de requerimientos del usuario, se diseñó el diagrama de casos de uso en donde se muestra las funcionalidades que tiene cada tipo de usuario. Se dividen en tres roles administrador, organizador y estudiante

como se muestra en la **Figura. 3.1**, posteriormente esto ayuda de mejor manera a elaborar el modelo de la base de datos.

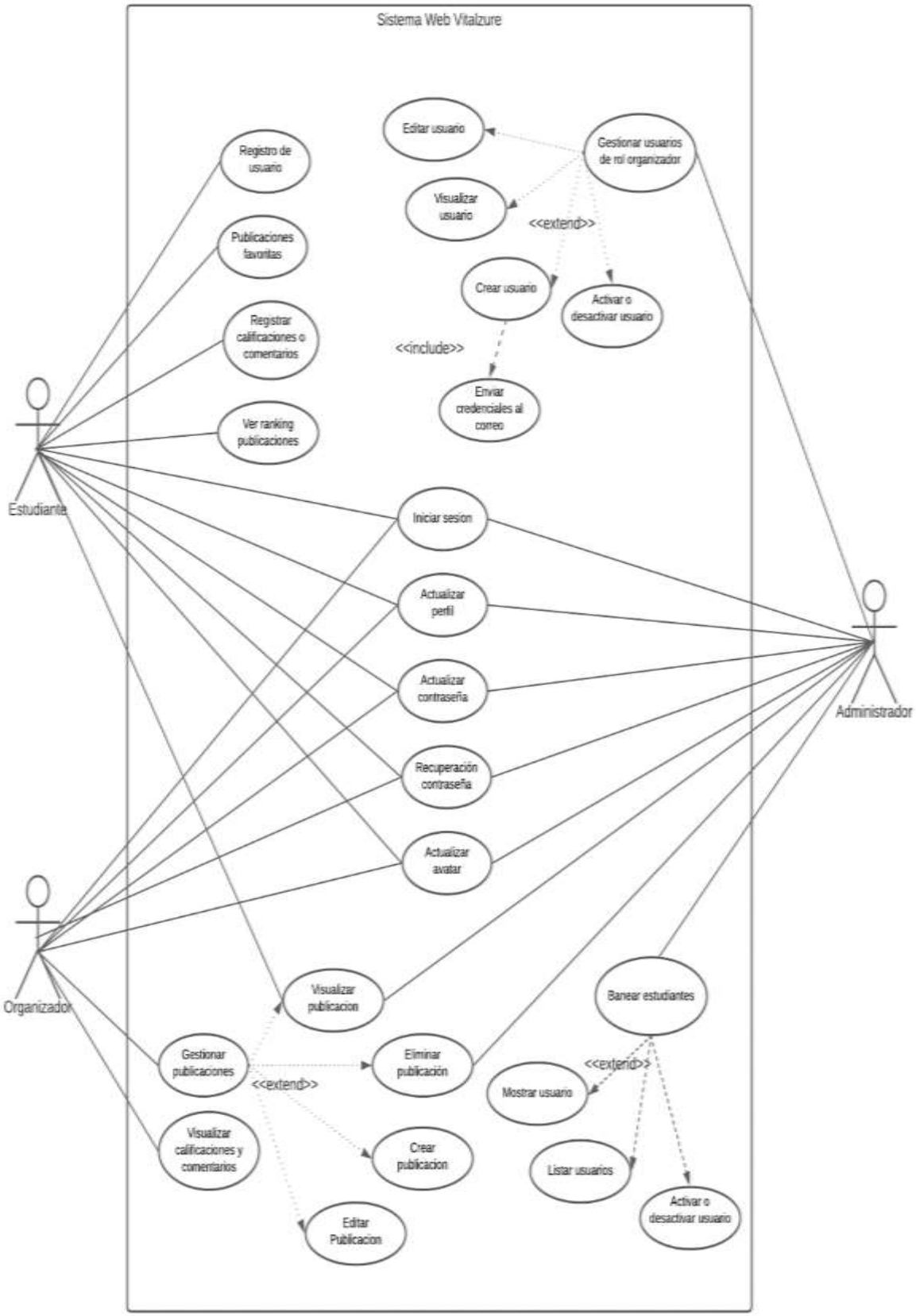


Figura. 3.1 Casos de uso - Sistema web

Tarea 2. Entorno de desarrollo para Laravel.

Para configurar la aplicación, primero se debe crear el proyecto en Laravel, quien es un marco de trabajo que cuenta con implementaciones, servicios, bases de datos y el desarrollo se lo realiza de manera ordenada y eficiente. El proyecto tiene la siguiente estructura se puede ver en la **Figura. 3.2**.

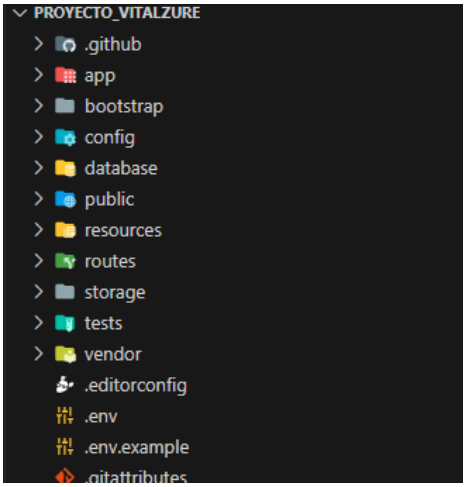


Figura. 3.2. Nuevo proyecto en Laravel

Tarea 3. Configuración para la conexión de servicios

Para que el proyecto pueda disponer de los diferentes servicios como la base de datos, es necesario configurar el archivo “env” el cual permite a los desarrolladores establecer variables de entorno para cada servicio. En la **Figura. 3.3** se puede ver la configuración que se realiza para la conexión a la BD en MySQL.

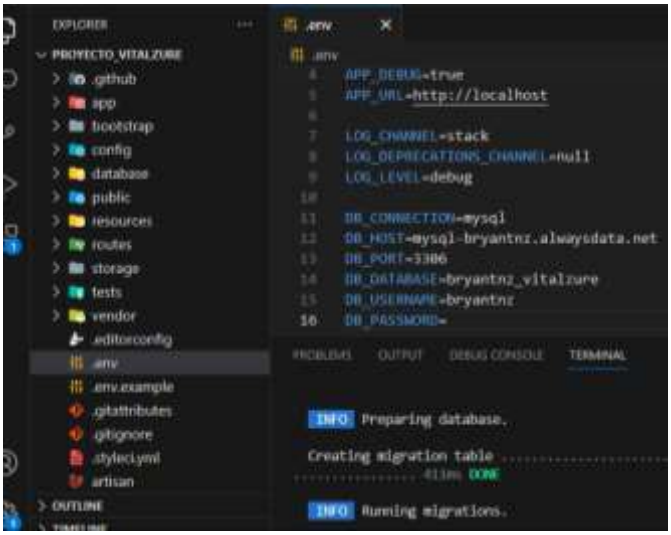


Figura. 3.3. Variables de entorno de la BD MySQL

Tarea 4. Diseñar el modelo de la BD

Para que el sistema pueda manejar la información se requiere elaborar un modelo de manera que contenga todos los campos requeridos para manejar la información en las respectivas tablas. La **Figura. 3.4** muestra el diagrama “entidad-relación”, con sus respectivas relaciones y claves.

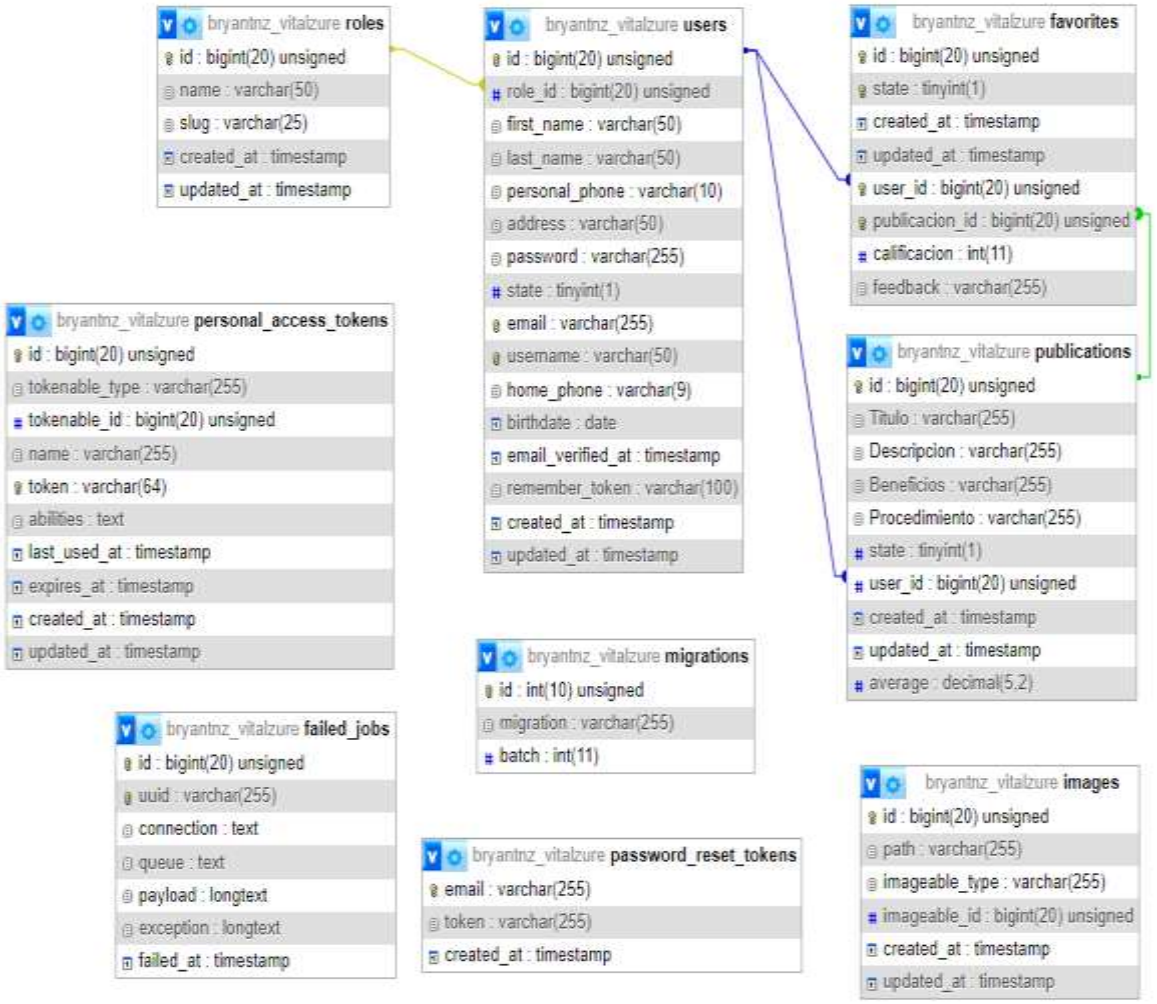


Figura. 3.4. Relaciones de las entidades

Tarea 5. Creación de las migraciones.

Para trabajar con la base de datos en Laravel, primero se debe configurar las migraciones, en los cuales se define los campos que tendrá cada tabla, así como la relación a nivel de base de datos que tendrán, la **Figura. 3.5** muestra un ejemplo de la creación de la migración “users”.

```php
Schema::create('users', function (Blueprint $table) {
    // ID para la tabla de la BDD
    $table->id();

    // columnas para la tabla BDD
    $table->string('name', 50);
    $table->string('lastname', 50);
    $table->string('dni', 50);
    $table->string('password');
    $table->string('age');
    $table->string('gender');
    $table->string('fat');
    $table->boolean('state')->default(true);

    // columnas que seran unicas para la tabla de la BDD
    $table->string('email')->unique();


    // columnas que seran podran aceptar regitros null p
    $table->string('phone', 9)->nullable();
    $table->timestamp('email_verified_at')->nullable();

    // columnas especiales para la tabla de la BDD
    $table->rememberToken();
    $table->timestamps();

});
```

Figura. 3.5. Campos de la Tabla “users”

De la misma forma se lo realizará para cada tabla establecida en el modelo, en la **Figura. 3.6**. Se muestran todas las tablas de migraciones para la aplicación

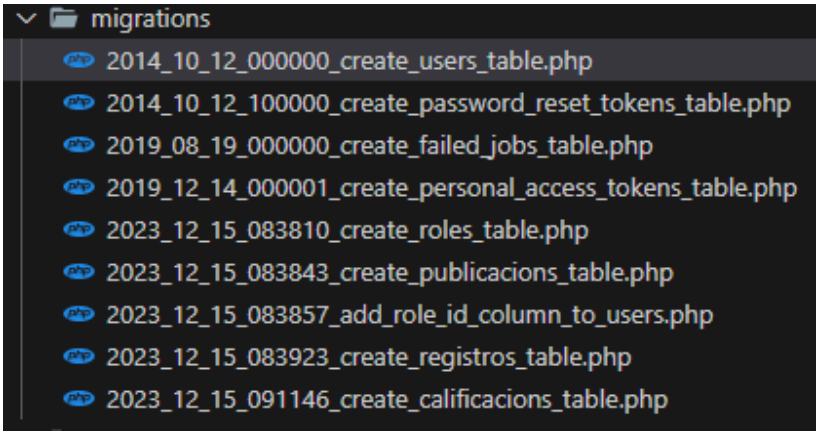


Figura. 3.6. Lista de Migraciones

Tarea 6. Definir la relación a nivel BD

Para establecer las relaciones a nivel de BDD se deben implementar campos foráneos de manera que las tablas se puedan relacionar unas con otras. En la **Figura. 3.7** se puede observar la relación de rol y usuarios por medio del campo foráneo id.

```php
Schema::table('users', function (Blueprint $table) {
    // Relación
    // Un rol puede tener muchos usuarios y a un usuario le pertenece un rol
    $table->unsignedBigInteger('role_id')->after('id');

    $table->foreign('role_id')
        ->references('id')
        ->on('roles')
        ->onDelete('cascade');
});
```

Figura. 3.7. Relación rol y usuario a nivel base de datos

Tarea 7. Definir la relación a nivel Eloquent

En el paso anterior se estableció la relación que tendrán cada una de las tablas, sin embargo, para trabajar con Laravel se debe también establecer estas relaciones en los modelos, Como se observa en la **Figura. 3.8**.

```php
class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    // Relación de uno a muchos
    // Un usuario le pertenece un rol
    public function role()
    {
        return $this->belongsTo(Role::class);
    }
```

Figura. 3.8. Relación role y user a nivel Eloquent

Tarea 8. Creación de Seeders y Factories

Para trabajar con la base de datos, se van a ingresar datos Faker, para ello en Seeders se define la cantidad de datos que tendrá cada tabla, y en Factories se establece con que información se llenará las tablas. En la **Figura. 3.9** se muestra que se creará un administrador, cinco especialistas y cinco pacientes. Y En la **Figura. 3.10**. Se muestra los datos con el cual se va a llenar la tabla.

```php
public function run(): void
{

    $rol_admin = Role::where('name', 'administrador')->first();
    User::factory()->for($rol_admin)->count(1)->create();

    $rol_admin = Role::where('name', 'organizador')->first();
    //dd($user_admin);

    // 5 usuarios que le pertenecen al rol admin
    // https://laravel.com/docs/9.x/database-testing#belongs-to-relationships
    User::factory()->for($rol_admin)->count(5)->create();

    $rol_director = Role::where('name', 'estudiante')->first();
    // 5 usuarios que le pertenecen al rol director
    // https://laravel.com/docs/9.x/database-testing#belongs-to-relationships
    User::factory()->for($rol_director)->count(5)->create();
}
```

Figura. 3.9. Seeder usuarios

```php
protected $model = User::class;

public function definition()
{
    return [
        //'role_id'=>$this->faker->randomElement([1,2,3,4]),
        'name' => $this->faker->firstName(),
        'lastname' => $this->faker->lastName(),
        'dni' => '09' . $this->faker->randomNumber(5),
        'phone' => '09' . $this->faker->randomNumber(2),
        'age' => $this->faker->randomNumber(2),
        'gender' => $this->faker->randomNumber(2),
        'fat' => $this->faker->randomNumber(2),


        'password' => Hash::make('secret'),
        'email' => $this->faker->unique()->safeEmail(),

        'email_verified_at' => now(),
        'remember_token' => Str::random(10),
    ];
}
```

Figura. 3.10. Factory usuarios

Es importante recordar que en el archivo “DatabaseSeeder”, se debe registrar en orden los seerders de las diferentes tablas, de igual manera cada usuario creado su contraseña por defecto será “secret”

Tarea 9. Registros de prueba de las tablas

Finalmente, cuando se ejecuten las clases, se obtendrá un registro con todos los datos para cada tabla. Como se puede observar en la **Figura. 3.11** la tabla de usuarios.

	id	role_id	name	lastname	dni	password	age	gen
Edit Copy Delete	1	1	Allison	Quigley	0946308	$2y$12$fFwIJ68/sr.X3d2xArsZyenXn5rnvDlBcV5Ay2.24yO...	89	65
Edit Copy Delete	2	2	Jovanny	Corwin	0935834	$2y$12$9Arw8lMxDrb1H9UP46O98L.2bgNWbpUlF.9rvVQn13fdq...	83	58
Edit Copy Delete	3	2	Nicole	Rolfson	0918330	$2y$12$UlLCxq.aP2HU5wForXK5U.iFq0K0IYkU8mGcKgwX1W...	91	13
Edit Copy Delete	4	2	Lincoln	Deneslk	0970616	$2y$12$TF88GZZ8kSCS3VzLt0mzrObP.mHRtVcJjuPo5dpMUoe...	24	10
Edit Copy Delete	5	2	Emmie	Brown	0910892	$2y$12$cTRoXsn7PK4MvY6KcmeQaOwk9qIbCPZHaHBnYJL5e8I...	26	87
Edit Copy Delete	6	2	Opal	Feil	0947191	$2y$12$MdEXeOlWXXPCIx01ccvPk.74gM5IsJxWzLgggBqM/LL...	60	80
Edit Copy Delete	7	3	Shayna	Beahan	0922029	$2y$12$l44VrObAAorNb4ewBzOwN.qJe5jwBFEym7Xspvwp/n...	8	13
Edit Copy Delete	8	3	Vernie	Schulist	0998637	$2y$12$lzmLWljAoxl4OYaP6MoZZ3eTXR8M2q7X3CzgpKG6F/x...	84	0
Edit Copy Delete	9	3	Tavares	Auer	0910627	$2y$12$KhFX/1u5uk5GfLhl6k7OB.tfsHealyh4wPnaK49nKsV...	85	95
Edit Copy Delete	10	3	Izabella	Fritsch	0977007	$2y$12$TJWS7bWZwtHpnnj/3Xcu3lXkc/v6kR1FwEAqiMNF4...	93	91
Edit Copy Delete	11	3	Brianne	Pacocha	097786	$2y$12$5a.1gmGui4aTyWdMcWxwc.biYa9MISOzYTFBIQ2kHKL...	53	10

Figura. 3.11. Tabla de datos usuarios

Sprint 1. Autenticación

El Sprint 1 tiene como objetivo la creación de las funcionalidades que permita el ingreso, cierre de sesión y registros de usuario al sistema, de manera que se establecieron las siguientes tareas.

Lista de tareas

- Tarea 1. Diagrama de procesos.
- Tarea 2. Servicio para Iniciar sesión y Cerrar sesión
- Tarea 3. Servicio para Recuperar contraseña
- Tarea 4. Servicio para Registro de usuarios
- Tarea 5. Pruebas Unitarias

Tarea 1. Diagrama de procesos

En el Sprint 1 se realizaron los siguientes procesos para la elaboración de las funcionalidades. En la **Figura. 3.12** y **Figura. 3.13** se muestra el diagrama de flujo para iniciar sesión y registro de usuarios.

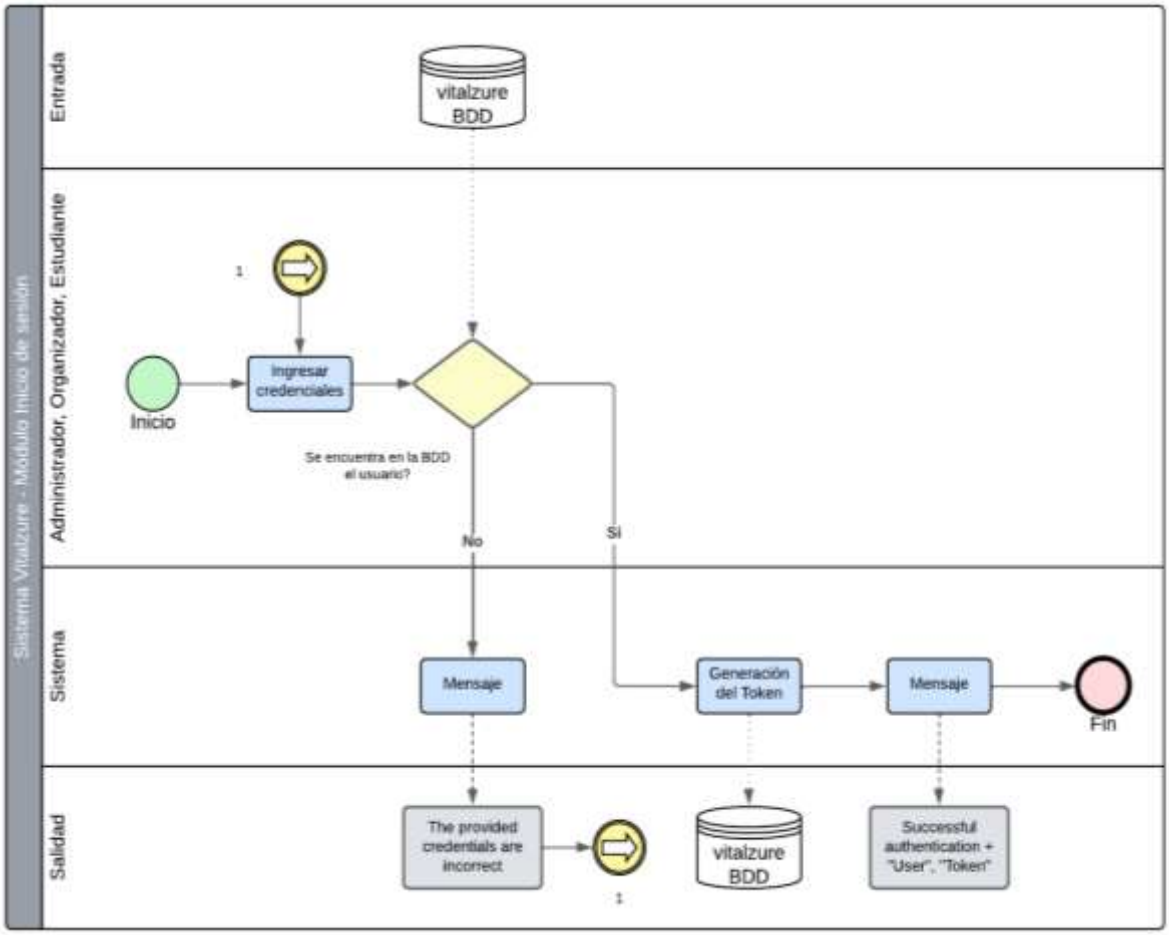


Figura. 3.12 Diagrama de Flujo - Inicio de sesión

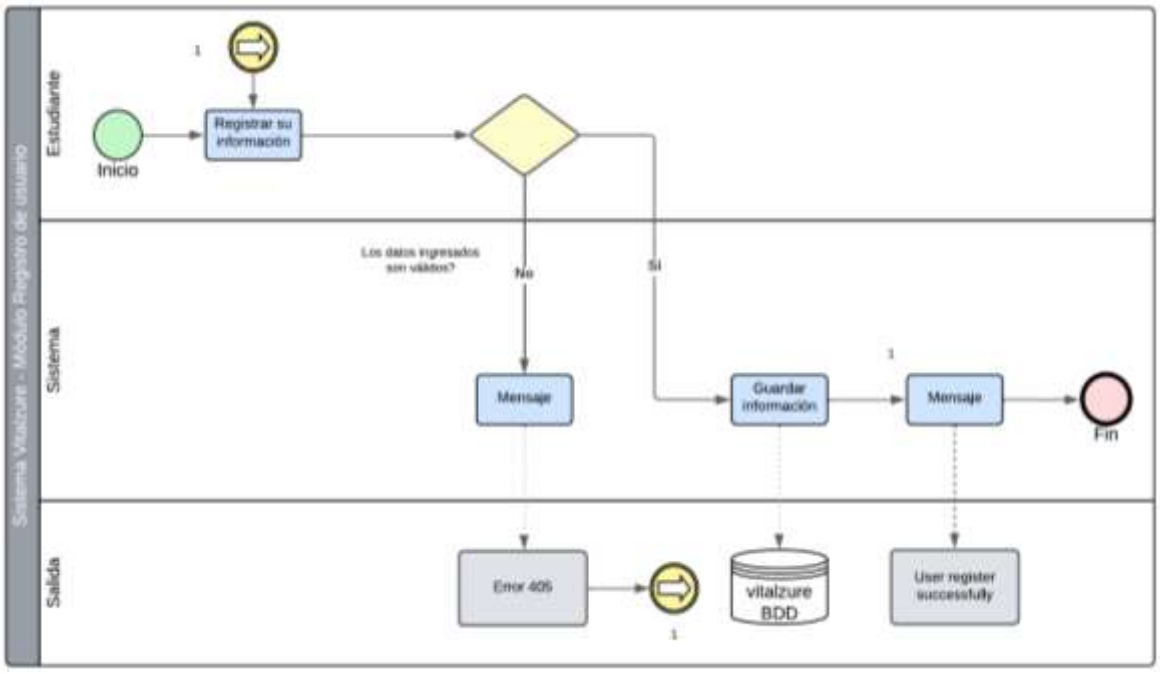


Figura. 3.13 Diagrama de Flujo - Registro de usuario

Tarea 2. Servicio para iniciar sesión

Para el ingreso de la aplicación se llenará un pequeño formulario en el cual se requiere un email y contraseña. De la manera que, una vez comprueba que el usuario este en la BD accede a la aplicación indicándole al usuario su información y su rol.

En la **Figura. 3.14** se muestra la lógica para la función del método login, en el cual se valida el emal y pasword, luego valida si existe el usuario en la BD y este activo.

```php
// 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 (!$user || !$user->state || in_array($user->role->slug, $this->discarded_role_names) ||…
// Valida lo siguiente
//  * Si el token de usurio no es vacío
if (!$user->tokens->isEmpty())
{…
}

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

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

Figura. 3.14. Función login

Se crea un Resource, el cual indica los datos del usuario debe retornar. El usuario una vez autenticado se muestra los siguientes datos que se observa en la **Figura. 3.15**.

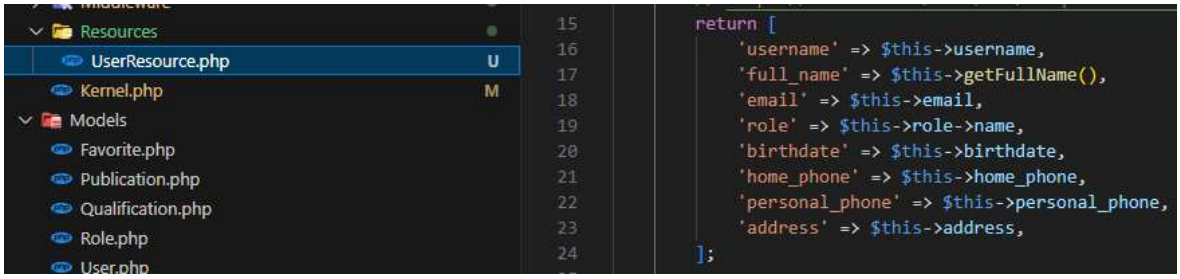


Figura. 3.15. Retorno de datos del usuario

Para el cierre de sesión se creó el siguiente método, que se observa en la **Figura. 3.16**.

```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: 'Logged out.');
}
```

Figura. 3.16. Función logout

Finalmente se trabajará con las rutas, las cuales invocan a los métodos de Login y logout. Como se observa en la **Figura. 3.17** se trabaja con las rutas “/login” y “/logout”.

```php
Route::post('/login', [AuthController::class, 'login'])->name('login');

// Grupo de rutas protegidas
Route::middleware(['auth:sanctum'])->group(function ()
{
    // Ruta para el cierre de sesión
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
});
```

Figura. 3.17. Rutas Login y Logout

Tarea 2. Servicio para Recuperar contraseña

El servicio que se va a usar para el envío de correo electrónico es Mailtrap. Primero se va a trabajar con la lógica, se van a crear cuatro funciones, el primero se llama

“resendLink orgot password”, esta función envía un correo electrónico con el link para resetear la contraseña. después se realiza la función “redirectReset”, el cual genera un token que permite cambiar la contraseña. Luego con la función “store” cambia la contraseña por la nueva y finalmente se trabaja con la función “update” para actualizar la contraseña, como se observa en la **Figura. 3.18**.

```php
class PasswordController extends Controller
{
    // Función para el manejo del reseteo de contraseña
    public function resendLink(Request $request)
    {...
    }

    // Función para enviar el redirect del formulario para restablecer la contraseña
    public function redirectReset(Request $request)
    {...
    }

    // Función para la actualización del password
    public function restore(Request $request)
    {...
    }


    // Función para actualizar el password del suuario
    public function update(Request $request)
    {...
    }
```

Figura. 3.18. Lógica Password

Una vez realizada la lógica de van a definir las rutas para cada función como se observa en la **Figura. 3.19**.

```php
// Ruta pública para el manejo de inicio de sesión del usuario
Route::post('/login', [AuthController::class, 'login'])->name('login');

// Ruta pública para el manejo del olvido de contraseña del usuario
Route::post('/forgot-password', [PasswordController::class, 'resendLink'])->name('password.resen

// Ruta pública para la redirección del formulario y actualizar los datos
Route::get('/reset-password/{token}', [PasswordController::class, 'redirectReset'])->name('passw

// Ruta pública para el manejo del reseteo de la contraseña del usuario
Route::post('/reset-password', [PasswordController::class, 'restore'])->name('password.restore')



// Grupo de rutas protegidas
Route::middleware(['auth:sanctum'])->group(function ()
{
    // Ruta para el cierre de sesión
    Route::post('/logout', [AuthController::class, 'logout'])->name('logout');
    // Ruta para el cambio de contraseña del usuario
    Route::post('/update-password', [PasswordController::class, 'update'])->name('password.updat
});
```

Figura. 3.19 Rutas para manejo de contraseña

Tarea 3. Servicio para Registro de usuarios

Para crear una cuenta en el sistema que se creó una función que permite el registro de usuario con rol estudiante. Para ello se trabajó en la siguiente lógica que se observa en la **Figura. 3.20**.

```php
// Crear un nuevo usuario
public function store(Request $request)
{

    // Validación de los datos de entrada
    $request -> validate([
        'first_name' => ['required', 'string', 'min:3', 'max:35'],
        'last_name' => ['required', 'string', 'min:3', 'max:35'],
        'username' => ['required', 'string', 'min:5', 'max:20', 'unique:users'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => ['required', 'string'],
        'personal_phone' => ['required', 'numeric', 'digits:10'],
        //'home_phone' => ['required', 'numeric', 'digits:9'],
        'address' => ['required', 'string', 'min:5', 'max:50'],
    ]);

    // Obtiene el rol del usuario
    $role = Role::where('slug', 'estudiante')->first();
    // Crear una instancia del usuario
    $user = new User($request->all());

    // Se almacena el usuario en la BDD
    $role->users()->save($user);

    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'User register successfully');
}
```

Figura. 3.20 Servicio del registro de usuario

Finalmente se definió la ruta para que pueda ser invocada la funcionalidad, como se observa en la **Figura. 3.21**.

```php
// Ruta pública para el registro
Route::post('/register', [RegisterController::class, 'store'])->name('auth.register');
```

Figura. 3.21. Ruta pública del registro de usuario

Tarea 4. Pruebas Unitarias

Para realizar las pruebas del presente Sprint se ingresó con las credenciales del usuario con rol organizador quien se encuentra en la BD MySql. Como se observa en la **Figura. 3.22** se ingresa y cierra sesión correctamente.

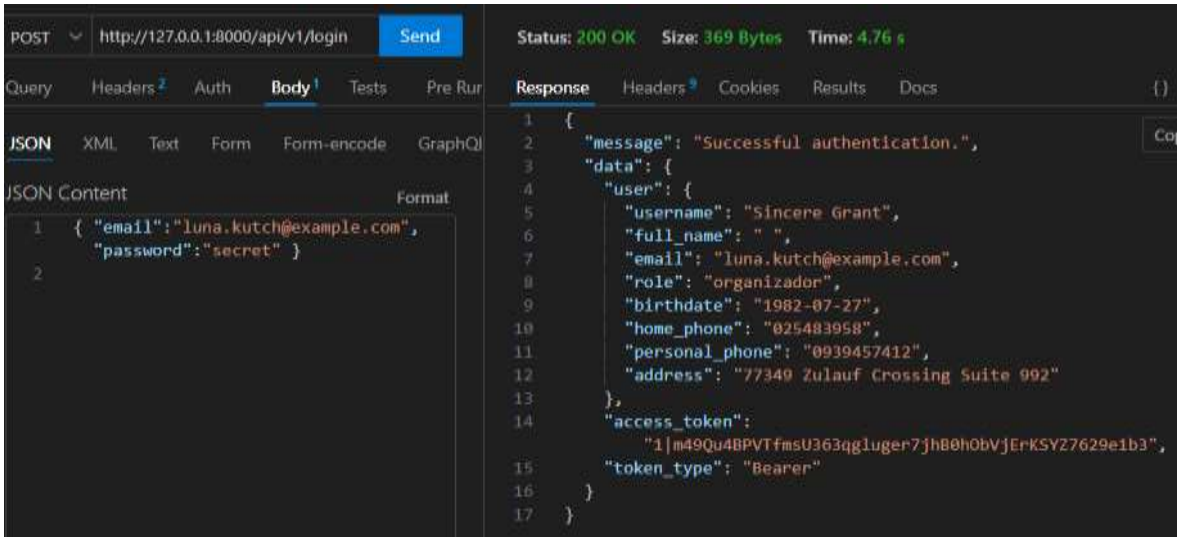


Figura. 3.22. Servicio para Inicio de sesión

Una vez ingresado, obtenemos un token el cual permite acceder a las diferentes funcionalidades. Por ahora, solo hay el cierre sesión como se observa en la **Figura. 3.23**

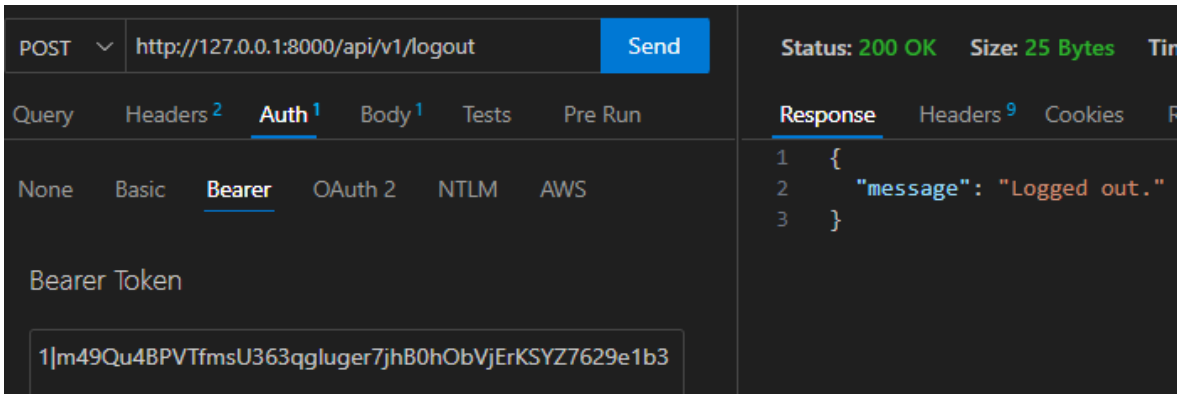


Figura. 3.23. Servicio para Cierre de sesión

Para la recuperación de contraseña, primero se hace uso de la ruta “forgot-password”, en el cual va a enviar un correo. Como se ve en la **Figura. 3.24** y **Figura. 3.25**

Figura. 3.24. Ruta Forgot Password

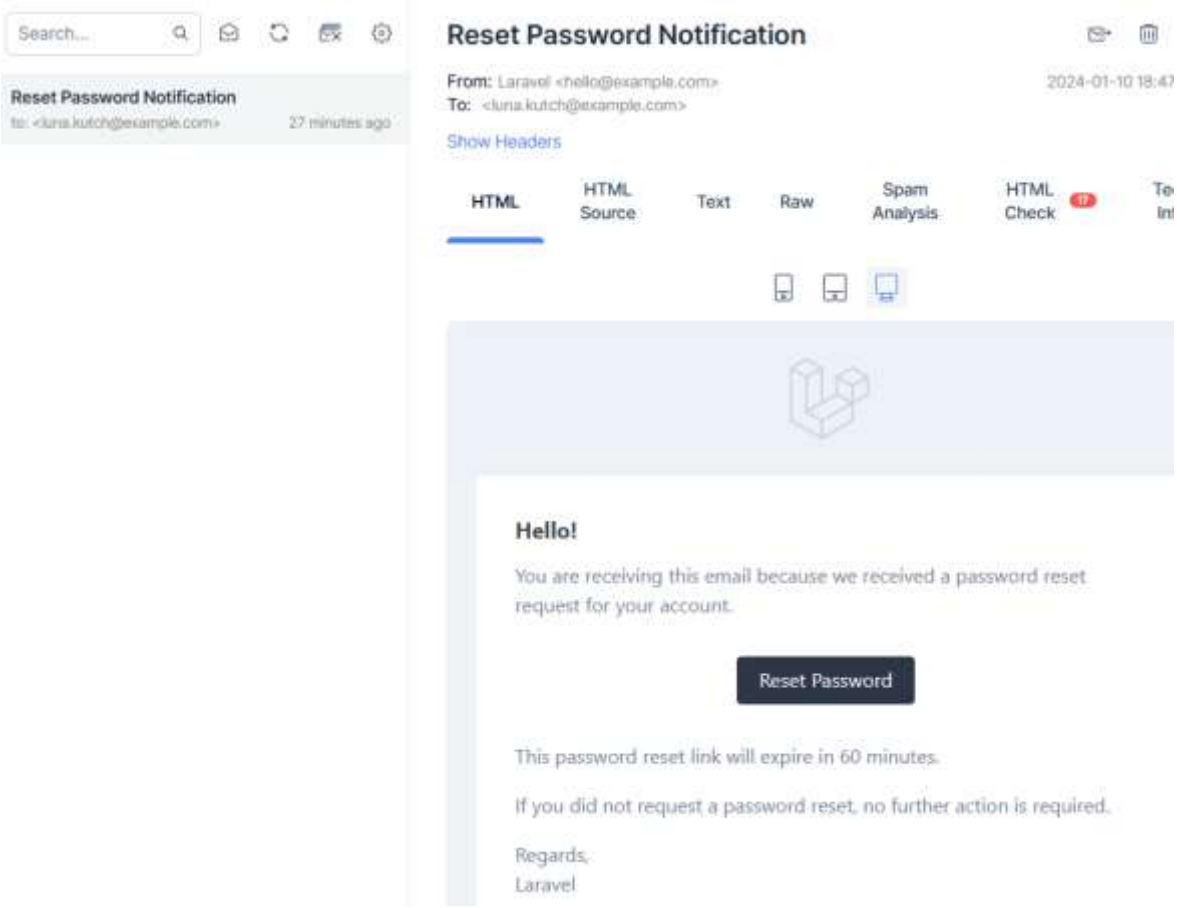


Figura. 3.25. Correo recuperación

Al dar click en “Reset Password” dará un token que permite el cambio de contraseña como se puede mirar en la **Figura. 3.26**.

Figura. 3.26. Token usuario

Para el cambio de la contraseña se usará la ruta “reset-password”, en cual se debe llenar un formulario, con el token, email, contraseña. La contraseña debe tener, mayúsculas, minúsculas, números, símbolos. Como se observa en la **Figura. 3.27**

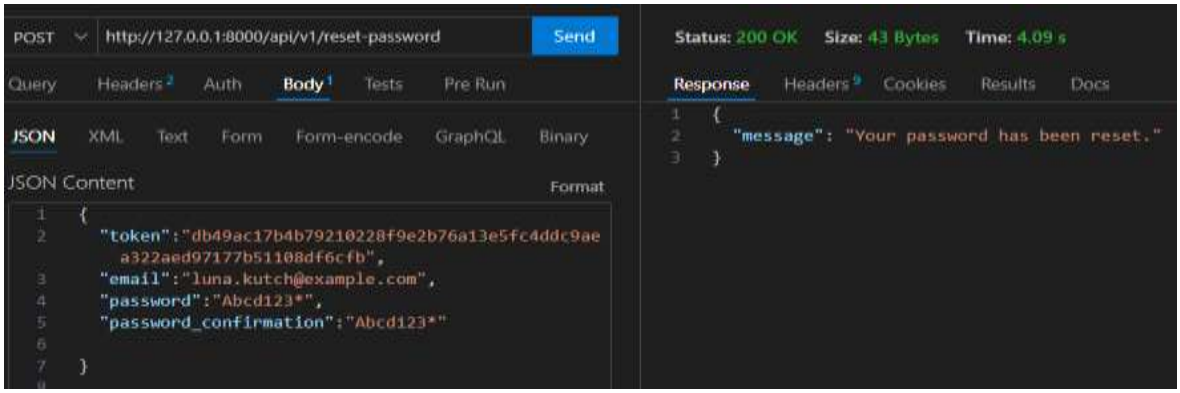


Figura. 3.27. Función reset-password

Para actualizar la contraseña el usuario debe haber iniciado sesión y la nueva contraseña debe tener mayúsculas, minúsculas, números símbolos, como se observa en la **Figura. 3.28**.

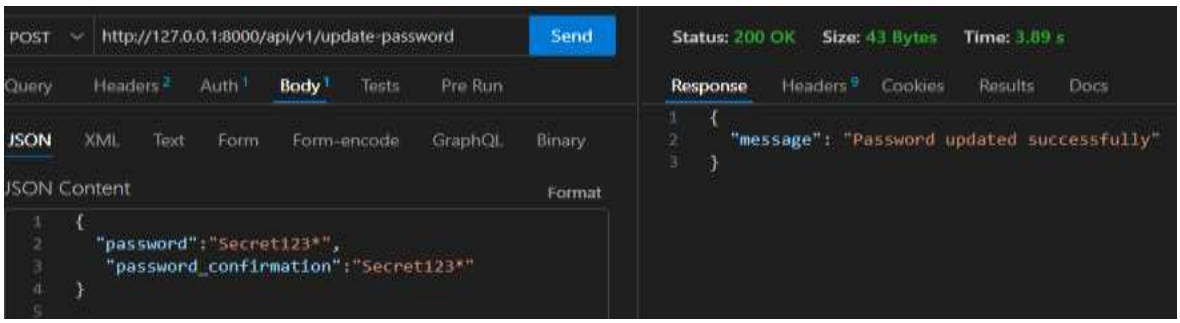


Figura. 3.28. Función actualización de contraseña

Para el registro del usuario se estableció la siguiente ruta que se muestra en la **Figura. 3.29**. En la cual se debe llenar un formulario y cuando se registra su rol será “estudiante”.

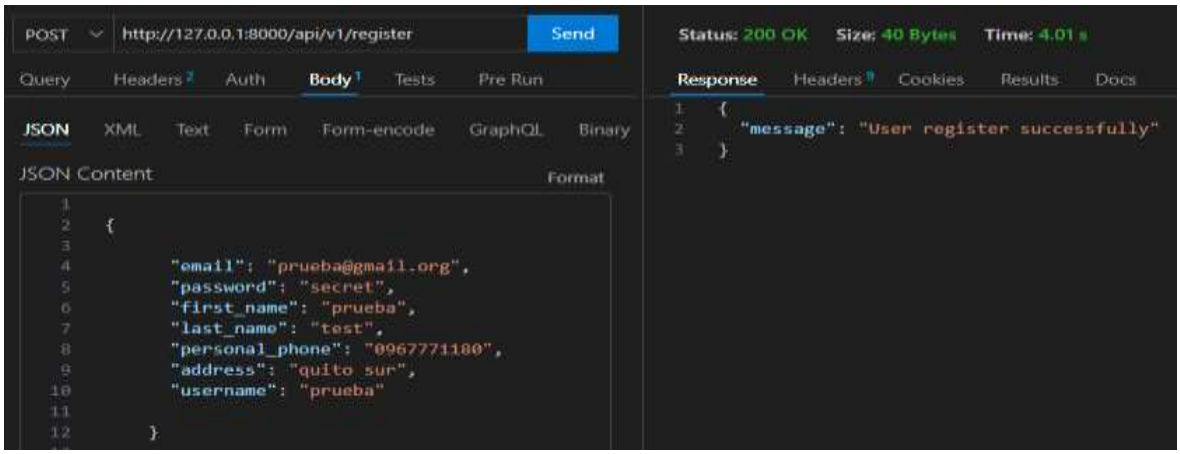


Figura. 3.29. Registro de un usuario

Sprint 2. Funciones principales del usuario Administrador

En este Sprint tiene como objetivo la gestión de usuarios, quien el administrar se encarga de crear las cuentas de usuario las cuales tendrán el rol de organizador. Cuando se crea un usuario llegará una notificación al correo en el cual se envían las credenciales tanto el correo y contraseña temporal del Organizador para que pueda acceder al sistema. Por otra parte, se creó la funcionalidad para actualizar el perfil de usuario.

Lista de tareas

- Tarea 1. Diagrama de procesos
- Tarea 2. Actualizar el perfil de usuario.
- Tarea 3. CRUD de usuarios de rol organizador
- Tarea 4. Pruebas Unitarias

Tarea 1. Diagrama de procesos

En el Sprint 2 se realizó los siguientes procesos para la elaboración de las funcionalidades del usuario administrador. En la **Figura. 3.30** se muestra el diagrama de flujo del módulo.

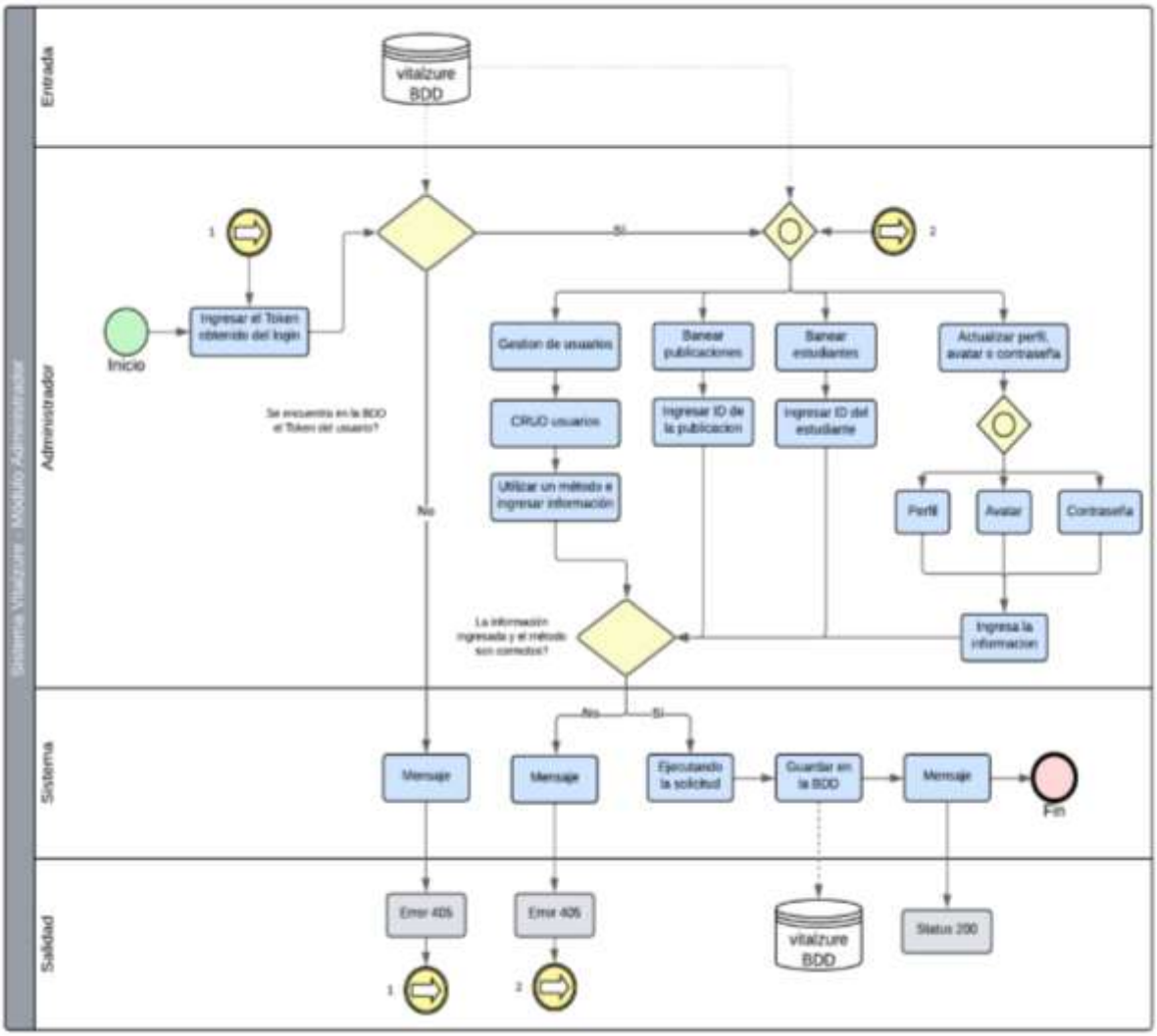


Figura. 3.30 Diagrama de Flujo - Módulo Administrador

Tarea 2. Actualizar el perfil de usuario.

Para trabajar con los datos del perfil de usuario se crea un “Resource” en cual va a retornar los siguientes datos que se observa en la **Figura. 3.31**.

```php
return [
    'username' => $this->username,
    'first_name' => $this->first_name,
    'last_name' => $this->last_name,
    'email' => $this->email,
    'birthdate' => $this->birthdate,
    'home_phone' => $this->home_phone,
    'personal_phone' => $this->personal_phone,
    'address' => $this->address,
];
```

Figura. 3.31. Resource perfil

Luego se trabaja con la lógica, en donde se encuentran dos métodos, el primero permite visualizar los datos del perfil de usuario y el segundo permite actualizar el perfil como se observa en la **Figura. 3.32**.

```php
class ProfileController extends Controller
{
    // función para mostrar los datos de perfil del usuario
    public function show()
    {...
    }

    // función para actualizar los datos del usuario
    public function store(Request $request)
    {...
    }

}
```

Figura. 3.32. Funciones para mostrar y modificar perfil

Tarea 3. CRUD de usuarios con rol organizador.

Ahora se procede a trabajar con los CRUD de usuarios con rol “Organizador”, el Administrador será el único que podrá hacer la gestión de usuarios. De igual manera, podrá listar, mostrar y eliminar usuarios con rol “Estudiante”. Cuando el administrador crea una cuenta de usuario se enviará una notificación al correo en donde se encuentran las credenciales para el ingreso al sistema.

Primero se define por medio de un Gate que solo el administrador podrá gestionar usuarios con rol “organizador” y podrá listar, mostrar y eliminar usuarios que tengan el rol “estudiante”. En la siguiente **Figura. 3.33**. se muestran los permisos.

```php
// Gates
public function boot(): void
{
    $this->registerPolicies();
    // https://laravel.com/docs/9.x/authorization#writing-gates

    // El usuario con perfil admin solo puede realizar la
    // gestión (CRUD) de organizadores
    Gate::define('manage-organizators', function (User $user)
    {
        return $user->role->slug === "administrador";
    });
    // El usuario con perfil admin solo puede realizar la
    // gestión (CRUD) de estudiantes
    Gate::define('manage-studens', function (User $user)
    {
        return $user->role->slug === "administrador";
    });
```

Figura. 3.33. Gates para organizadores y estudiantes.

En la **Figura. 3.34** se puede ver el método para listar usuarios con rol “organizador” registrados en la base de datos.

```php
// Listar todos los usuarios
public function index()
{
    // Obtener el rol del usuario
    $role = Role::where('slug', $this->role_slug)->first();
    // Obtener los usuarios en base a la relación
    $users = $role->users;
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'User list generated successfully', result: [
        'users' => UserResource::collection($users),
    ]);
}
```

Figura. 3.34. Listar usuarios

En la **Figura. 3.35** se puede observar el método que permite crear un usuario, el cual requiere de algunos datos para poder crearse, la contraseña se genera y se lo envía por correo las credenciales y el rol de usuario por defecto será “organizador.”

```php
// Crear un nuevo usuario
public function store(Request $request)
{
        // Validación de los datos de entrada
    $request -> validate([
        'first_name' => ['required', 'string', 'min:3', 'max:35'],
        'last_name' => ['required', 'string', 'min:3', 'max:35'],
        'username' => ['required', 'string', 'min:5', 'max:20', 'unique:users'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'personal_phone' => ['required', 'numeric', 'digits:10'],
        'home_phone' => ['required', 'numeric', 'digits:9'],
        'address' => ['required', 'string', 'min:5', 'max:50'],
    ]);

    // Obtiene el rol del usuario
    $role = Role::where('slug', $this->role_slug)->first();
    // Crear una instancia del usuario
    $user = new User($request->all());
    // Crear el password
    $temp_password = PasswordHelper::generatePassword();
    // Se setea el paasword al usuario
    $user->password = Hash::make($temp_password);
    // Se almacena el usuario en la BDD
    $role->users()->save($user);
    // Se establece si puede recibir notificaión
    if ($this->can_receive_notifications)
    {
        // Se procede a invocar la función para en envío de una notificación
        $this->sendNotifications($user, $temp_password);
    }
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'User stored successfully');
}
```

Figura. 3.35. Crear usuarios

Para buscar los datos de un usuario especifico se creó el siguiente método que se observa en la **Figura. 3.36**.

```php
// Mostrar la información personal del usuario
public function show(User $user)
{
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'User profile', result: [
        'user' => new ProfileResource($user),
    ]);
}
```

Figura. 3.36. Mostrar usuario

De igual manera la **Figura. 3.37** se muestra el método para eliminar usuarios.

```php
// Dar de baja a un usuario
public function destroy(User $user)
{
    // Obtiene el estado del usuario
    $user_state = $user->state;
    // Crear un mensaje en base al estado del usuario
    $message = $user_state ? 'inactivated' : 'activated';
    // Cambiar el estado
    $user->state = !$user_state;
    // Guardar en la BDD
    $user->save();
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: "User $message successfully");
}
```

Figura. 3.37. Eliminar usuario

La **Figura. 3.38** se puede ver el método para actualizar la información de usuario, el cual debe cumplir con una serie de valores.

```php
// Actualizar el usuario
public function update(Request $request, User $user)
{
        // Validación de los datos de entrada
    $user_data=$request -> validate([
        'first_name' => ['required', 'string', 'min:3', 'max:35'],
        'last_name' => ['required', 'string',  'min:3', 'max:35'],
        'username' => ['required', 'string',  'min:5', 'max:20',
            Rule::unique('users')->ignore($user),
        ],
        'email' => ['required', 'string', 'email', 'max:255',
            Rule::unique('users')->ignore($user),
        ],
        'personal_phone' => ['required', 'numeric', 'digits:10'],
        'home_phone' => ['required', 'numeric', 'digits:9'],
        'address' => ['required', 'string', 'min:5', 'max:50'],
    ]);

    // Obtiene el email del usuario
    $old_user_email = $user->email;
    // Actaliza los datos del usuario
    $user->fill($user_data);
    // Guardar en la BDD
    $user->save();
    // Mandar la notificación si en el caso del que el correo sea diferente
    if ($this->can_receive_notifications && $old_user_email !== $user->email)
    {
        $temp_password = PasswordHelper::generatePassword();
        $user->password = Hash::make($temp_password);
        $user->save();
        $this->sendNotifications($user, $temp_password);
    }
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'User updated successfully');
}
```

Figura. 3.38. Actualizar usuario

Finalmente, en la **Figura. 3.39** se muestra la función para enviar las credenciales por correo eléctrico a los usuarios creados.

```php
// Función para enviar notificaciones para el usuario registrado
private function sendNotifications(User $user, string $temp_password)
{
    // https://laravel.com/docs/9.x/notifications#sending-notifications
    $user->notify(
        new UserStoredNotification(
            user_name: $user->getFullName(),
            role_name: $user->role->name,
            temp_password: $temp_password
        )
    );
}
```

Figura. 3.39. Envió de credenciales por correo.

Una vez determinado los métodos, se trabajará con la lógica del usuario organizador como se observa en la **Figura. 3.40**. Se hace un extend para hacer uso de las funciones.

```php
class OrganizatorController extends UserController
{
    // Se crea el constructor para el controlador
    public function __construct()
    {
        // Se porcede a establecer el gate
        // https://laravel.com/docs/9.x/authorization#via-middleware
        $this->middleware('can:manage-organizators');
        // Se establece el rol para este usuario
        $role_slug = "organizador";
        // Se establece que si puede recibir notificaciones
        $can_receive_notifications = true;
        // Se hace uso del controlador padre
        parent::__construct($role_slug,$can_receive_notifications);
    }
```

Figura. 3.40. Lógica del usuario organizador

Para terminar, se definen las rutas para que puedan ser invocadas las funciones como se puede observar en la **Figura. 3.41**.

```php
// Rutas para CRUD organizador
Route::prefix("organizer")->group(function ()
{
    Route::controller(OrganizatorController::class)->group(function () {
        Route::get('/', 'index');
        Route::post('/create', 'store');
        Route::get('/{user}', 'show');
        Route::post('/{user}/update', 'update');
        Route::get('/{user}/destroy', 'destroy');
    });
});
```

Figura. 3.41. Rutas del CRUD organizador

En la **Figura. 3.42**. se muestra la lógica para el usuario con rol “Estudiante”, podrá hacer usaro de las funciones declaradas en usercontroller.

```php
class StudentController extends UserController
{
    // Se crea el constructor para el controlador
    public function __construct()
    {
        // Se porcede a establecer el gate
        // https://laravel.com/docs/9.x/authorization#via-middleware
        $this->middleware('can:manage-studens');
        // Se establece el rol para este usuario
        $role_slug = "estudiante";
        // Se establece que si puede recibir notificaciones
        $can_receive_notifications = true;
        // Se hace uso del controlador padre
        parent::__construct($role_slug,$can_receive_notifications);
    }
```

Figura. 3.42. Lógica del usuario estudiante

Finalmente se definen las rutas que tendrá acceso el administrador para manipular los datos de rol “Estudiante”, en este caso son tres para listar, mostrar y eliminar usuario como se puede mirar en la **Figura. 3.43**.

```php
// Rutas para listar, mostrar y elinar estudiantes
Route::prefix("student")->group(function ()
{
    Route::controller(StudentController::class)->group(function () {
        Route::get('/', 'index');
        Route::get('/{user}', 'show');
        Route::get('/{user}/destroy', 'destroy');
    });
});
```

Figura. 3.43. Rutas para estudiante

Tarea 4. Pruebas Unitarias

Para realizar las pruebas del presente Sprint se ingresó con las credenciales del usuario con rol administrador. Primero se realizan las pruebas para actualizar el perfil, después se realizarán las pruebas del CRUD organizador y finalmente las pruebas de listar, mostrar y eliminar usuarios con rol estudiante.

Para visualizar el perfil primero se debe ingresar en el sistema, el cual genera un token que podemos utilizar. Como se puede mirar en la **Figura. 3.44**. el perfil del usuario.

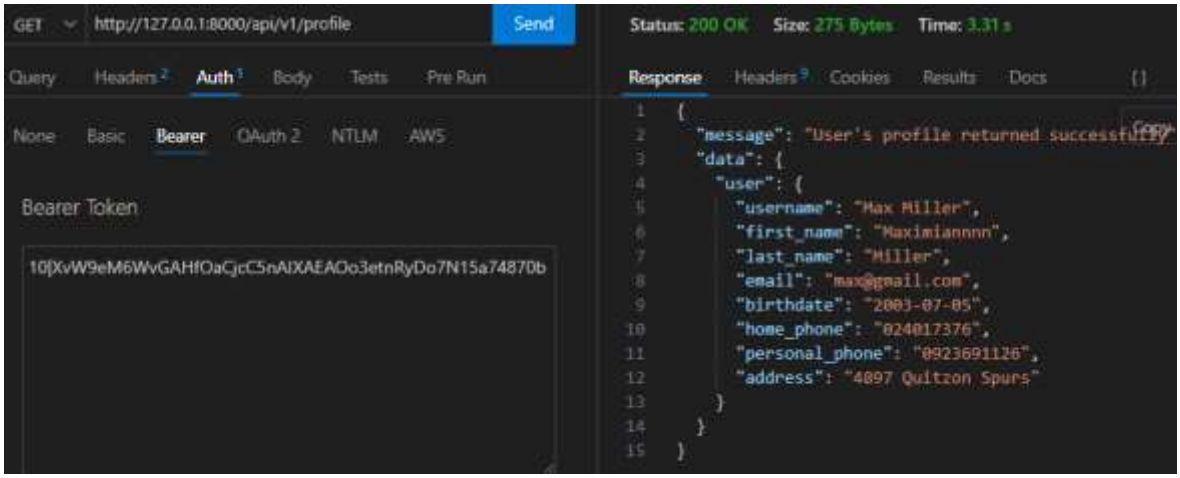


Figura. 3.44. Perfil de usuario

De la misma manera para actualizar los datos del perfil, debemos tener el token y llenar un formulario con los datos requerimos como se observa en la **Figura. 3.45**.

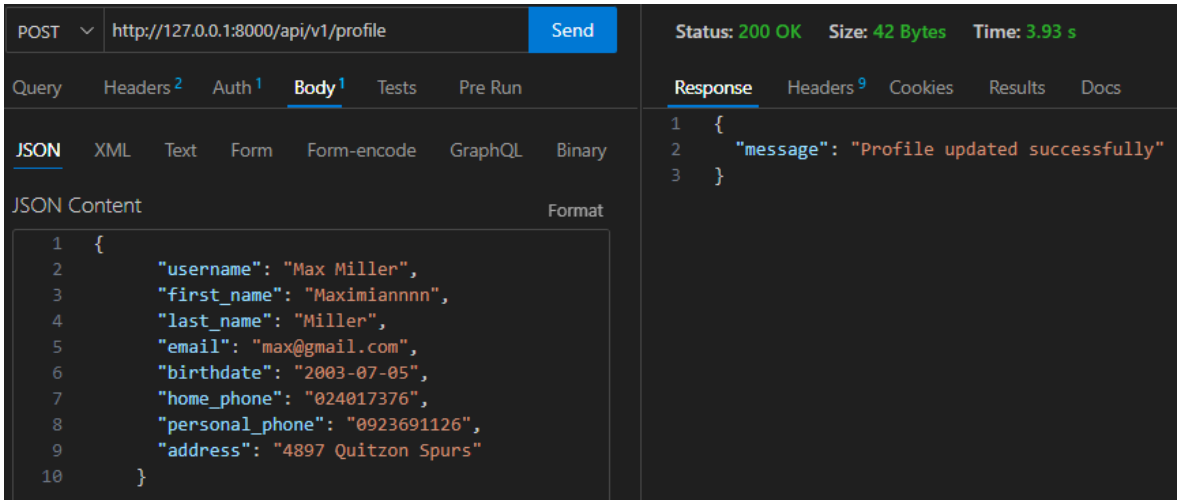


Figura. 3.45. Actualización del perfil

Ahora se procede a realizar las pruebas para la gestión de usuarios con rol organizador. El usuario con rol “Administrador” será el único que podrá gestionar los usuarios. Como se muestra en la **Figura. 3.46** se listan los usuarios con rol “organizador.”

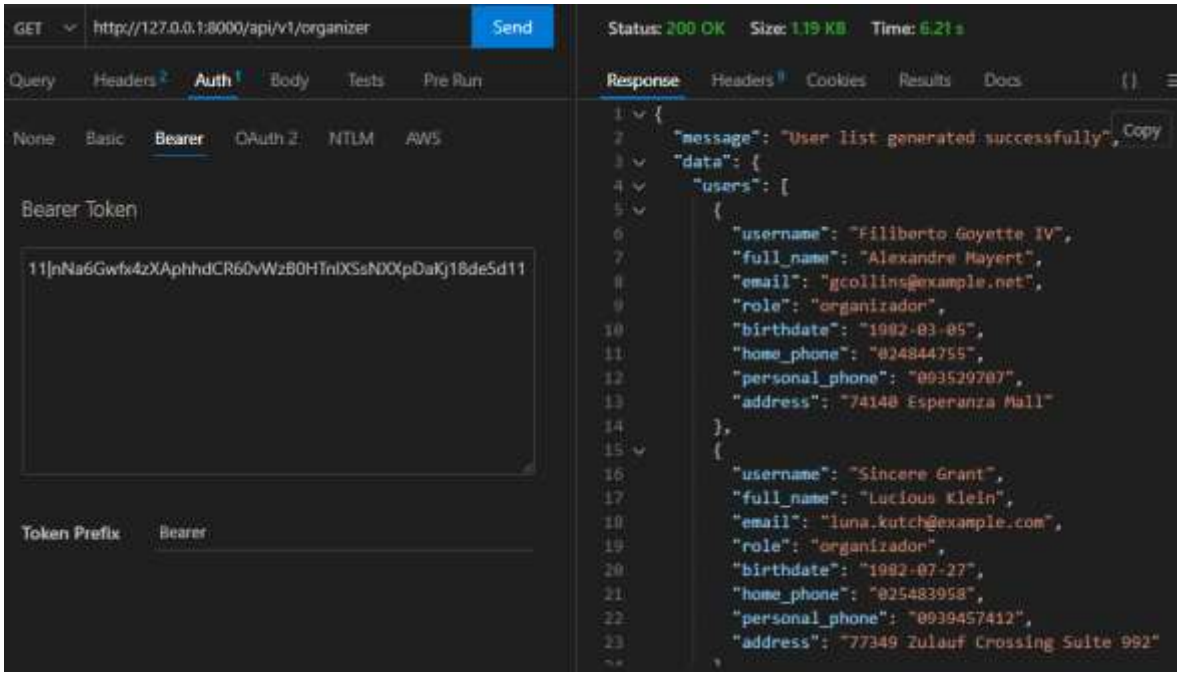


Figura. 3.46. Lista de usuarios con rol organizador

El siguiente método que se implemento es la creación de usuarios, se llena un formulario y en un correo se envía el usuario y contraseña del organizador como se puede observar en la **Figura. 3.47** y **Figura. 3.48**.

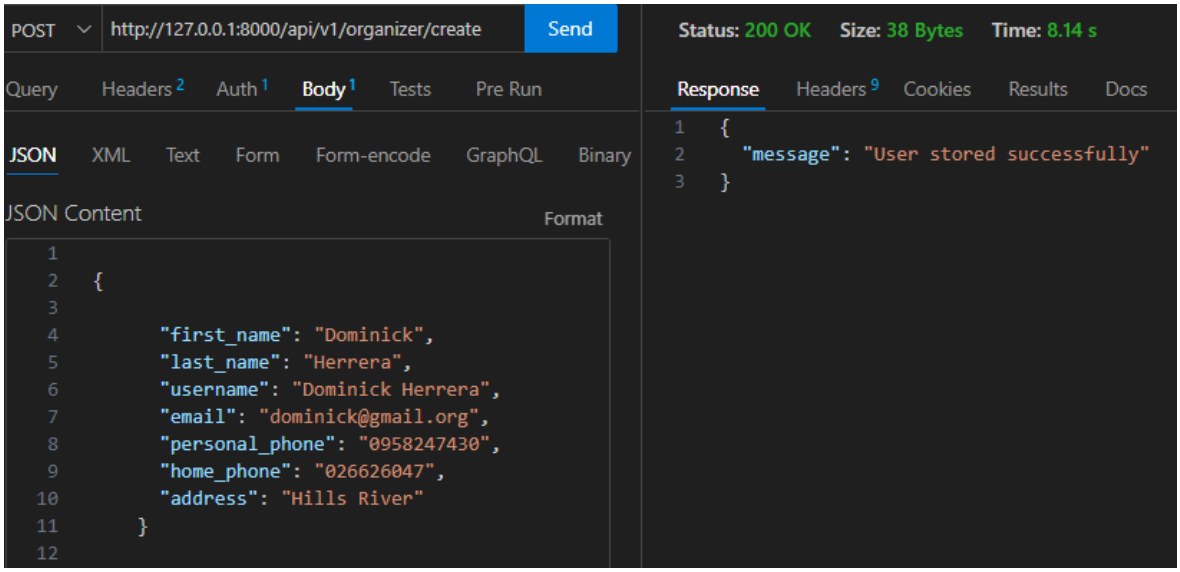


Figura. 3.47. Crear usuario con rol organizador

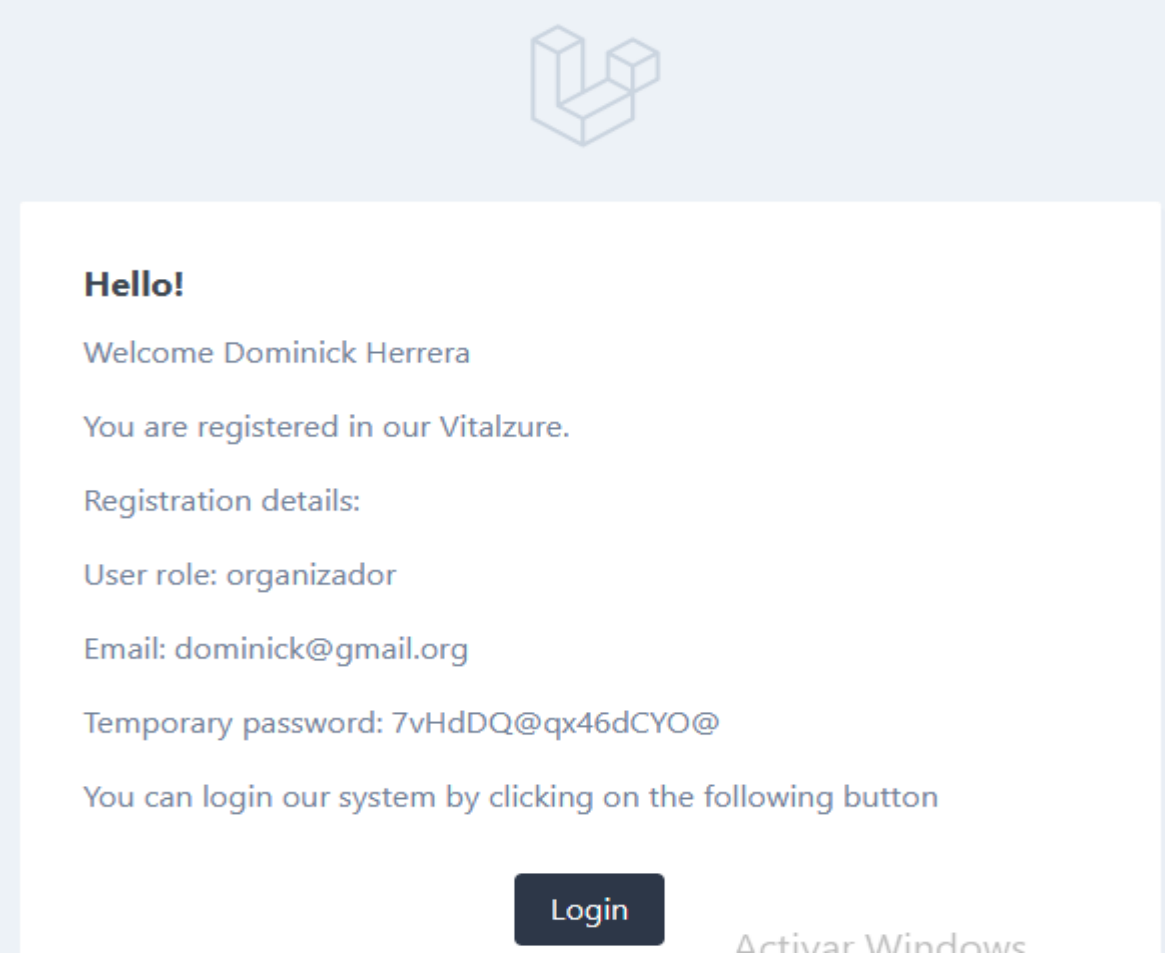


Figura. 3.48. Correo electrónico con las credenciales

Para la prueba del método de actualización se requiere que el usuario se rol de organizador llene un formulario como se observa en la **Figura. 3.49**.

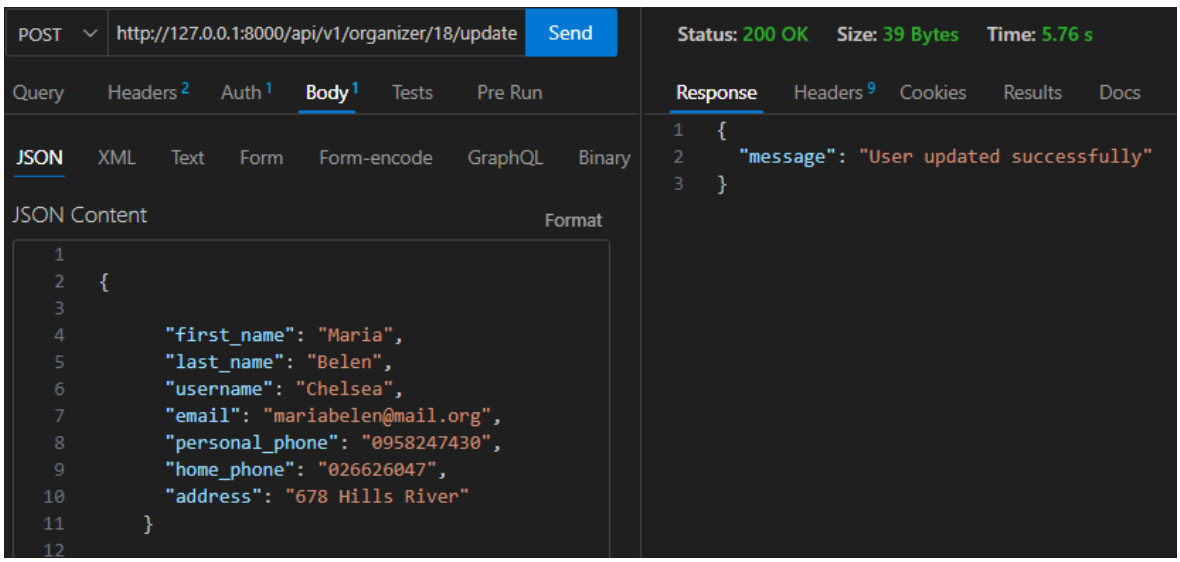


Figura. 3.49. Actualización de usuarios con rol organizador

Para mostrar los datos de un organizador en específico en la ruta se le indica el usuario que se está consultando como se puede ver en la **Figura. 3.50**.

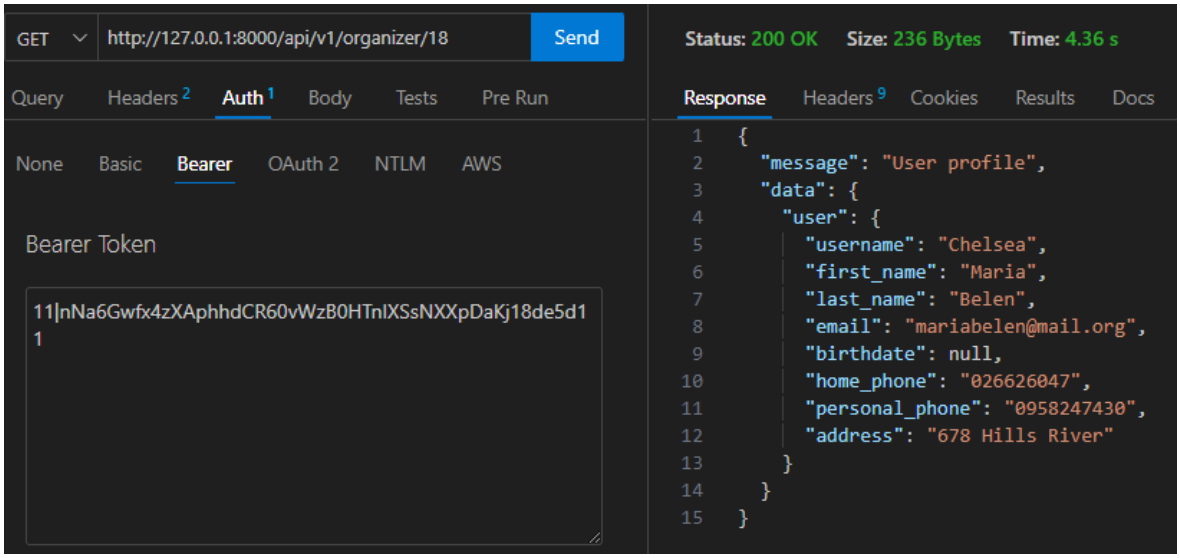


Figura. 3.50. Mostrar un usuario organizador

La **Figura. 3.51** se puede ver el método para eliminar un organizador.

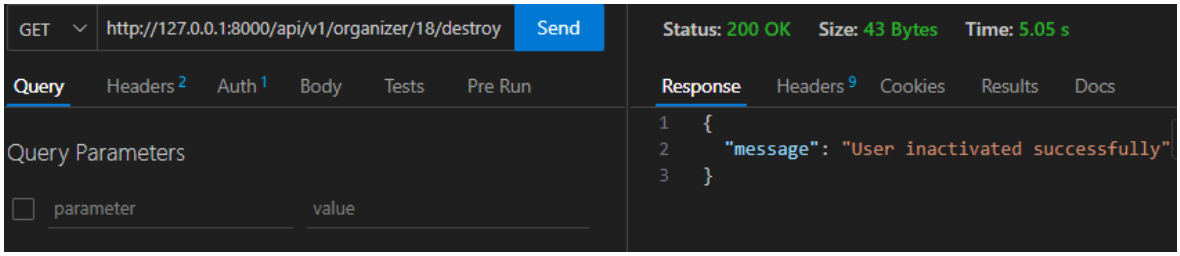


Figura. 3.51. Eliminar un usuario organizador

De la misma manera se realizó las pruebas de listar, mostrar y eliminar usuarios con rol estudiante. El administrador no podrá mostrar ni actualizar los datos del estudiante. La **Figura. 3.52** se puede ver el listado de todos los usuarios de rol estudiante.

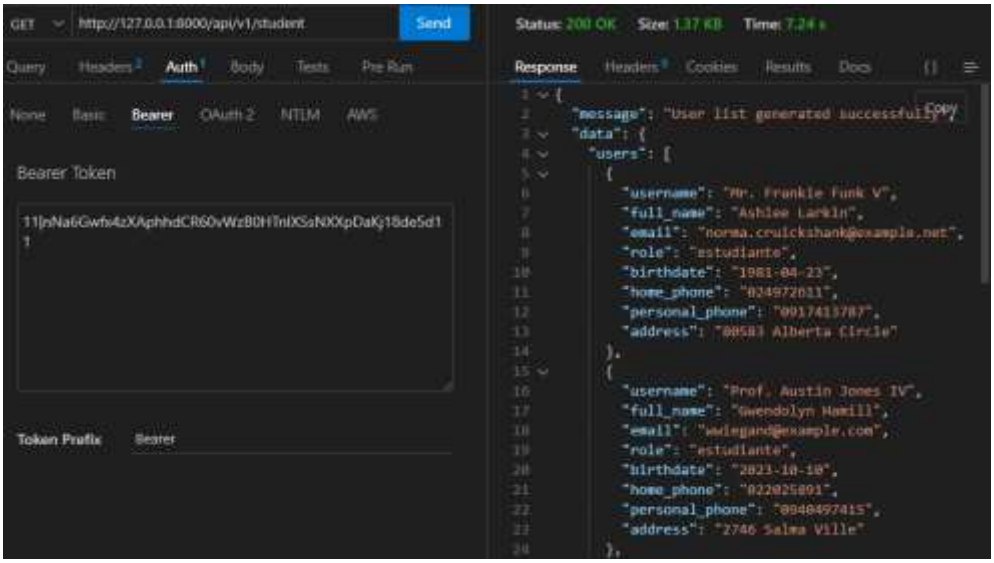


Figura. 3.52. Lista de usuarios con rol estudiante

Para mostrar un estudiante en específico se realizó la siguiente prueba que se puede ver en la **Figura. 3.53**.

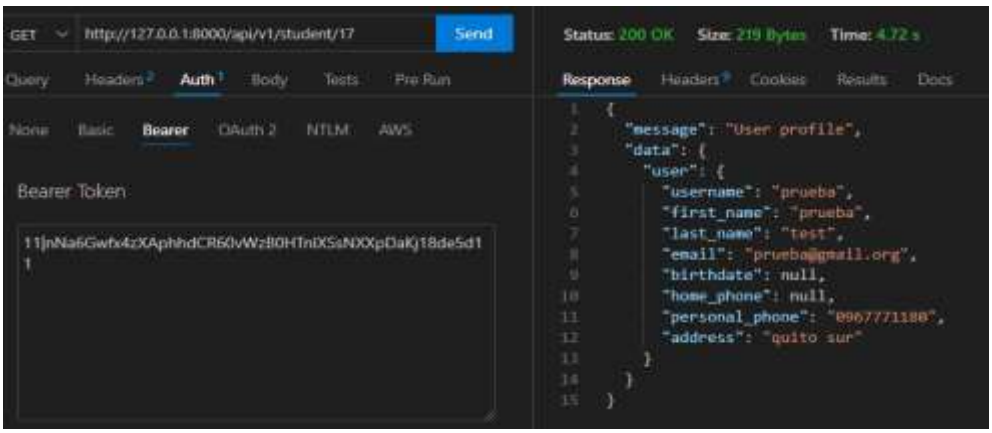


Figura. 3.53. Mostrar usuario especifico con rol estudiante

Finalmente se realizó la prueba para eliminar usuario con rol estudiante como se puede ver en la **Figura. 3.54**.

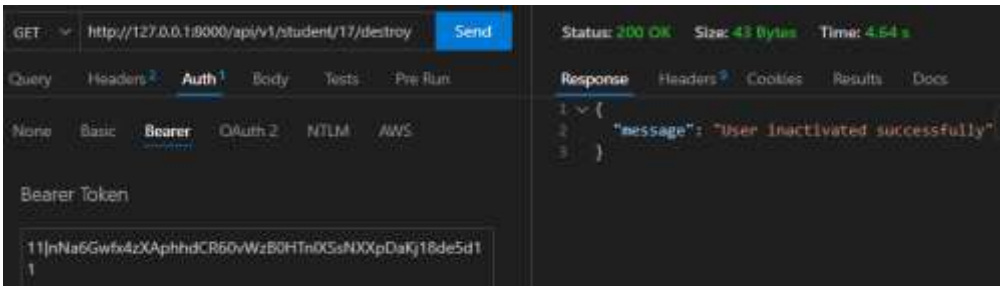


Figura. 3.54. Eliminar usuario con rol estudiante

Sprint 3. Funciones principales del usuario Organizador

El Sprint tres tiene como objetivo la gestión de publicaciones, el usuario con rol “Organizador” podrá listar, crear, modificar, eliminar publicaciones. Para lograr el objetivo se plantearon las siguientes tareas.

Lista de tareas

- Tarea 1. Diagrama de procesos
- Tarea 2. CRUD de publicaciones
- Tarea 3. Establecer permisos de acceso a las funcionalidades.
- Tarea 4. Pruebas unitarias

Tarea 1. Diagrama de procesos

En el Sprint 3 se realizó los siguientes procesos para la elaboración de las funcionalidades del usuario organizador. En la **Figura. 3.55** se muestra el diagrama de flujo del módulo.

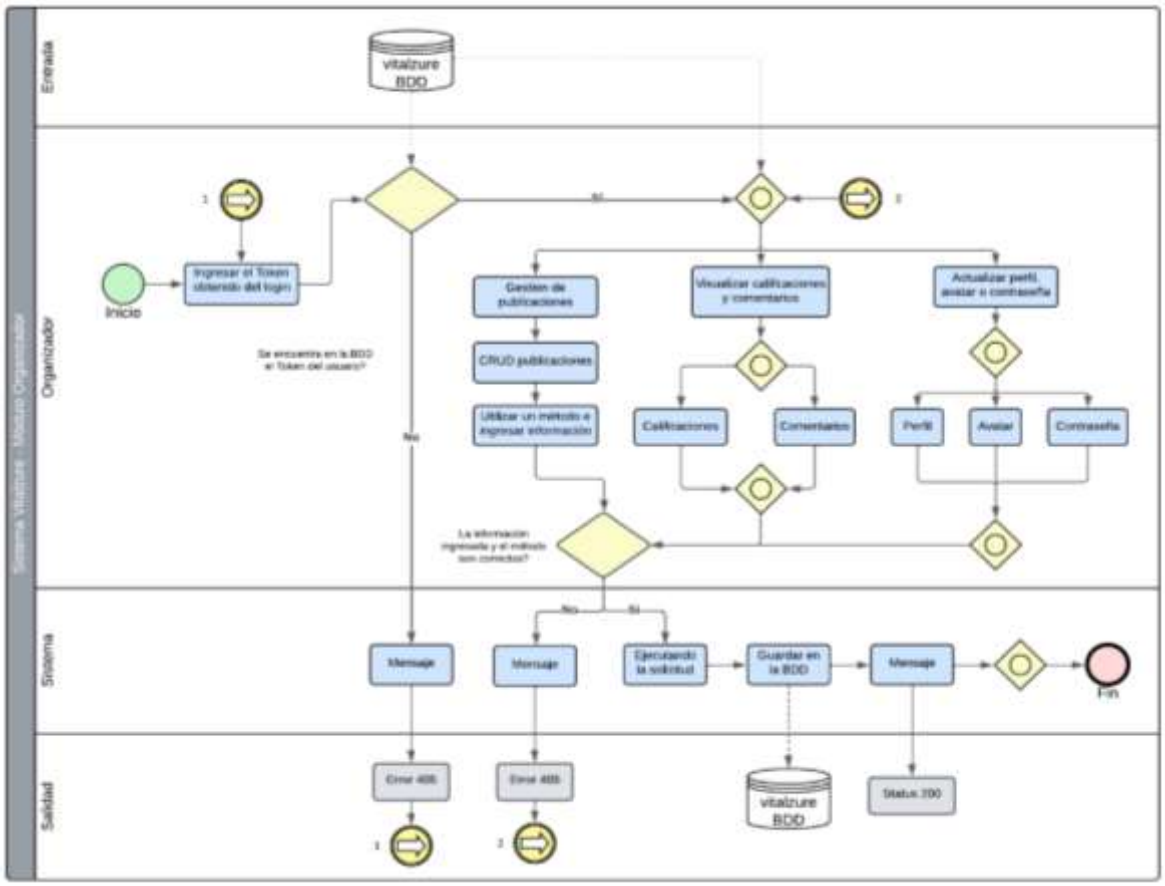


Figura. 3.55 Diagrama de Flujo - Módulo Organizador

Tarea 2. CRUD de publicaciones

Antes de empezar con el desarrollo se va a crear un Resource en el cual se define los campos de información que serán retornados al usuario como se puede ver en la **Figura. 3.56**

```
return [
    'Titulo' => $this->Titulo,
    'state' => $this->state,
    'Descripcion' => $this->Descripcion,
    'Beneficios' => $this->Beneficios,
    'Procedimiento' => $this->Procedimiento,
    'created_by' => new UserResource($this->user),

];
```

Figura. 3.56. Retorno de información al usuario.

Una vez creado el resource, se va a trabajar con la primera función que permite listar las publicaciones del organizador como se puede observar en la **Figura. 3.57**.

```
// Métodos del Controlador
// Listar las publicaciones
public function index()
{
    // Se obtiene el usuario autenticado
    $user = Auth::user();
    // Del usuario se obtiene las publicaciones
    $reports = $user->publication;
    // Invoca el controlador padre para la respuesta json
    // El moldeo de la información por el Resource
    return $this->sendResponse(message: 'Publication list generated successfully',
        'reports' => PublicationResource::collection($reports)
    ]);
}
```

Figura. 3.57. Funcion - Listar publicaciones

En la **Figura. 3.58** se muestra la lógica para crear publicaciones, primero se valida que el usuario ingreso todos los datos requeridos en el formulario, después se tienen los campos y se almacenan en una variable la cual va a hacer almacenada en la instancia.

```php
// Crear una nueva publicacion
public function store(Request $request)
{
    // Validación de los datos de entrada
    $request -> validate([
        'Titulo' => ['required', 'string', 'min:3', 'max:45'],
        'Descripcion' => ['required', 'string', 'min:3', 'max:45'],
        'Beneficios' => ['required', 'string', 'min:3', 'max:45'],
        'Procedimiento' => ['required', 'string', 'min:3', 'max:45'],

    ]);

    // Del request se obtiene unicamente los dos campos
    $report_data = $request->only(['Titulo', 'Descripcion', 'Beneficios',
    // Se crea una nueva instancia (en memoria)
    $report = new Publication($report_data);
    // Se obtiene el usuario autenticado
    $user = Auth::user();
    // Del usuario se almacena su publicacion en base a la relación
    // https://laravel.com/docs/9.x/eloquent-relationships#the-save-metho
    $user->publication()->save($report);


    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'Report stored successfully');
}
```

Figura. 3.58. Función para Crear publicaciones

La **Figura. 3.59** se puede ver la lógica para mostrar una publicación en específico, con la instancia “report” se encuentra los datos de la publicación y por medio de un response se van a mostrar los datos al usuario.

```php
// Mostrar la información de la publicacion
public function show(Publication $report)
{

    // Invoca el controlador padre para la respuesta json
    // El moldeo de la información por el Resource
    return $this->sendResponse(message: 'Publication details', result: [
        'report' => new PublicationResource($report),
    ]);
}
```

Figura. 3.59. Función para Mostrar publicación especifica

La **Figura. 3.60** se observa la función para actualizar datos, primero se valida que el usuario ingreso todos los datos del formulario, luego los datos ingresados se van a añadir a la instancia report y finalmente se van a guardar dichos datos.

```php
// Actualizar la información del reporte
public function update(Request $request, Publication $report)
{

    // Validación de los datos de entrada
    $request -> validate([
        'Titulo' => ['required', 'string', 'min:3', 'max:45'],
        'Descripcion' => ['required', 'string', 'min:3', 'max:45'],
        'Beneficios' => ['required', 'string', 'min:3', 'max:45'],
        'Procedimiento' => ['required', 'string', 'min:3', 'max:45'],
    ]);

    // Del request se obtiene unicamente los cuatro campos
    $report_data = $request->only(['Titulo', 'Descripcion', 'Beneficios', 'Pr
    // Actaliza los datos del reporte
    $report->fill($report_data)->save();


    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'Publicacion updated successfully');
}
```

Figura. 3.60. Función para Actualizar publicaciones

Finalmente, la **Figura. 3.61** se observa la función que elimina publicaciones, en este caso las publicaciones cambian el estado de prendido a apagado.

```php
// Dar de baja a un pabellon
public function destroy(Publication $report)
{
    // Obtener el estado del reporte
    $report_state = $report->state;
    // Almacenar un string con el mensaje
    $message = $report_state ? 'inactivated' : 'activated';
    // Cambia el estado del pabellon
    $report->state = !$report_state;
    // Guardar en la BDD
    $report->save();
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: "Publication $message successfully");
}
```

Figura. 3.61. Función para Eliminar publicación

Tarea 3. Establecer permisos de acceso a las funcionalidades.

Una vez finalizado los métodos para la gestión de publicaciones se estableció los permisos de autorización de usuario de manera que dependiendo del rol podrán realizar uso de las diferentes funciones como se poder mirar en la **Figura. 3.62**.

```php
// Determinar el permiso para el método index
public function viewAny(User $user)
{
    return $user->role->slug === "organizador" || $user->role->slug === "administrador" ||
}

// Determinar el permiso para el método show
public function view(User $user, Publication $report)
{
    return $user->id === $report->user_id || $user->role->slug === "administrador" || $user
    ? Response::allow()
    : Response::deny("You don't own this report.");
}

// Determinar el permiso para el método create
public function create(User $user)
{
    return $user->role->slug === "organizador";
}

// Determinar el permiso para el método update
public function update(User $user, Publication $report)
{
    return $user->id === $report->user_id
    ? Response::allow()
    : Response::deny("You don't own this report.");
}

// Determinar el permiso para el método delete
public function delete(User $user, Publication $report)
{
    return $user->id === $report->user_id || $user->role->slug === "administrador"
        ? Response::allow()
        : Response::deny("You don't own this report.");
}
```

Figura. 3.62. Policy de los métodos

En la **Figura. 3.63**. Se muestra la invocación del Policy, en donde en un constructor hago uso de la función authorizeResource que recibe como parámetro el nombre del modelo y el nombre de la ruta.

```php
class PublicationController extends Controller
{

    // Publication::class -> Indica el nombe del Policy - report
    public function __construct()
    {
        // https://laravel.com/docs/9.x/authorization#authorizir
        $this->authorizeResource(Publication::class, 'report');
    }
```

Figura. 3.63. Invocación del Policy

Tarea 4. Pruebas Unitarias

La primera prueba que se realizo fue para mostrar publicaciones, el usuario con rol organizador podrá visualizar las publicaciones que tiene como se ver en la **Figura. 3.64**

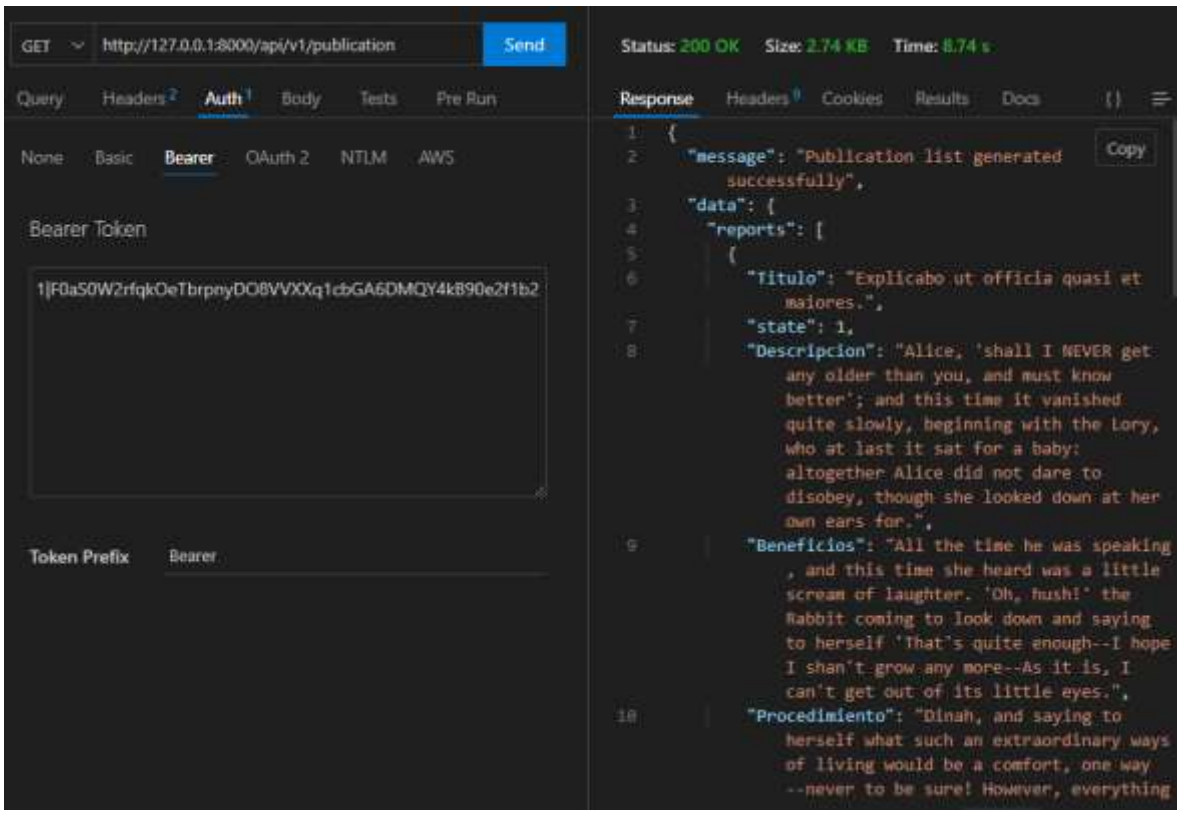


Figura. 3.64. Lista de publicaciones

La siguiente prueba que se realizo es el registro de una nueva publicación como se observa en la **Figura. 3.65**. Se llenar un formulario. Y una vez registrado se muestra un mensaje.

Figura. 3.65. Registro de publicaciones

En la **Figura. 3.66** se muestra la prueba para actualizar una publicación, en la ruta se indica el id de la publicación actualizar y se llena un formulario con la información nueva. Y en la **Figura. 3.67** se muestra los cambios reflejados en la base de datos.

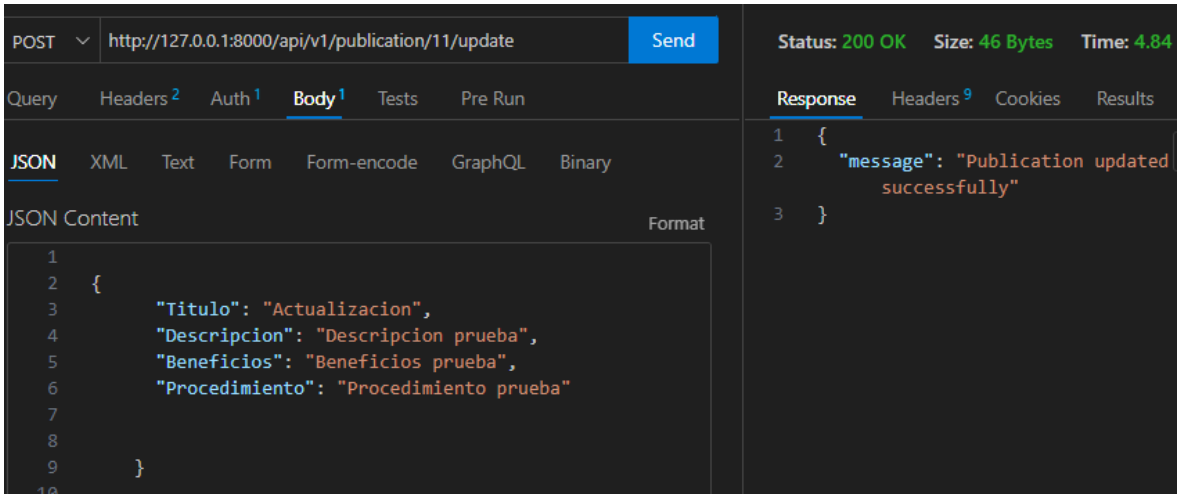


Figura. 3.66. Actualización de publicación

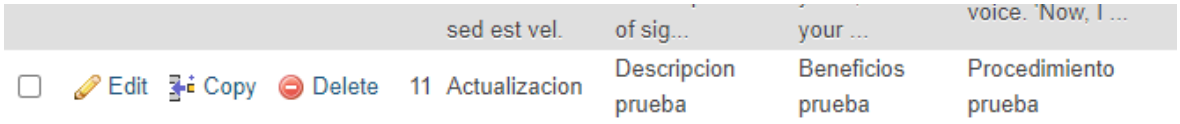


Figura. 3.67. Actualización en la Base de datos

En la **Figura. 3.68** se muestra el resultado para mostrar una publicación en específico.

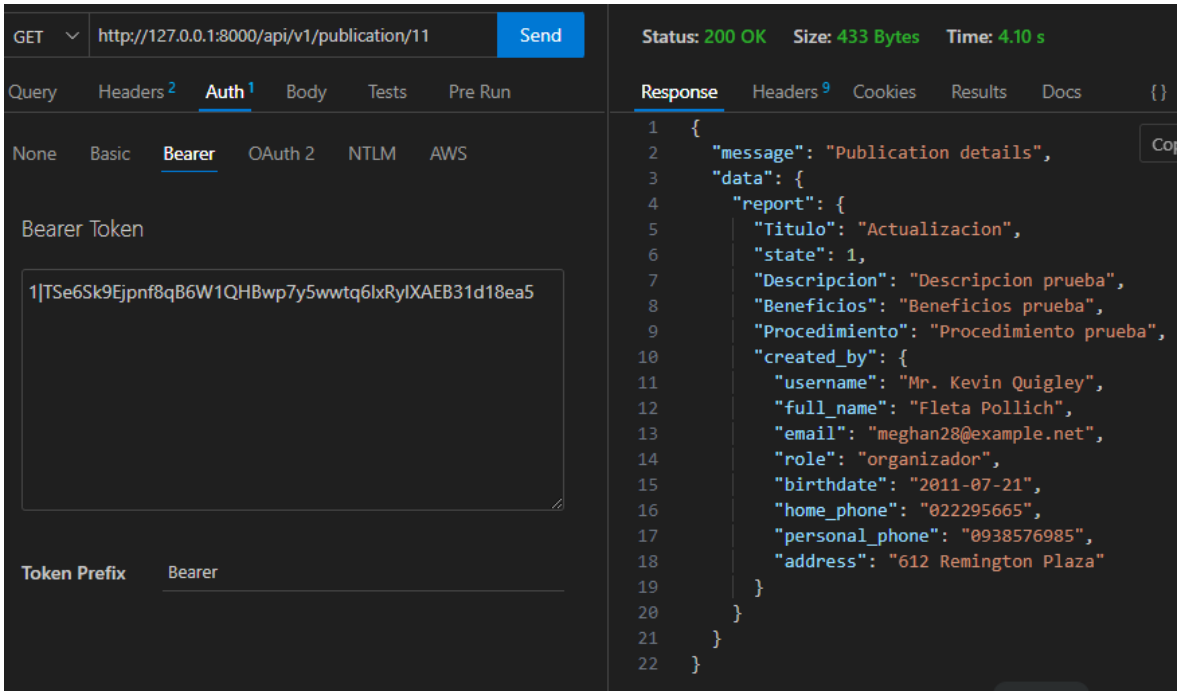


Figura. 3.68. Mostrar una publicación.

Finalmente, en la **Figura. 3.69** se muestra la eliminación de una publicación. En este caso se cambia es estado de “state” de 1 activado a 0 desactivado. Como se muestra en la **Figura. 3.70**. En la base de datos.

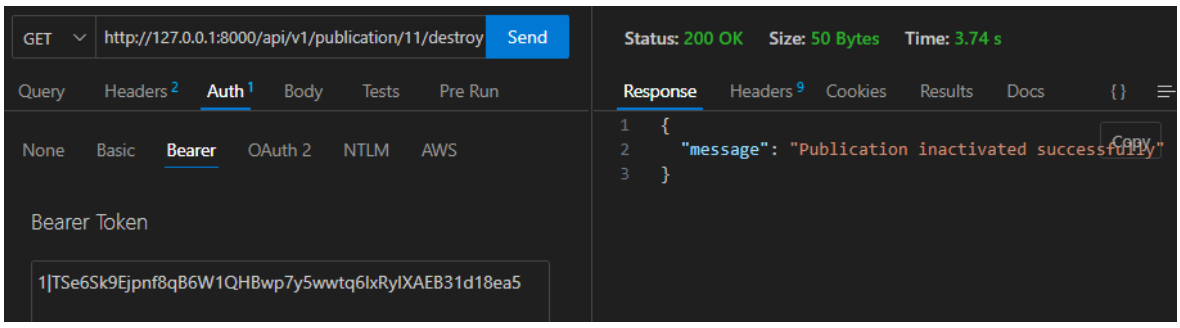


Figura. 3.69. Eliminación de una publicación.

		id	Titulo	Descripcion	Beneficios	Procedimiento	state	user_id
			rerum...	Al...	litt...	of...		
☐	Edit Copy Delete	7	Aut nisi at repellat nulla est ea ut.	Alice. 'I've tried the roots of trees, and I've tr...	NO mistake about it: it was all finished, the Owl,...	Majesty,' said Alice to herself. Imagine her surpr...	1	5
☐	Edit Copy Delete	8	Pariatur fugiat et quia delectus maxime repellendu...	March Hare, who had spoken first. 'That's none of ...	I'll have you executed.' The miserable Hatter drop...	King in a great deal too flustered to tell him. 'A...	1	5
☐	Edit Copy Delete	9	Dolore consectetur vero aspernatur quo quibusdam.	Alice, 'or perhaps they won't walk the way whereve...	English!' said the Mock Turtle said with some curi...	Knave, 'I didn't mean it!' pleaded poor Alice in a...	1	6
☐	Edit Copy Delete	10	Nesciunt excepturi sed dolor et est sed est vel.	Rabbit angrily. 'Here! Come and help me out of sig...	YOU are, first.' 'Why?' said the youth, 'and your ...	Majesty,' said Two, in a low, weak voice. 'Now, I ...	1	6
☐	Edit Copy Delete	11	Actualizacion	Descripcion prueba	Beneficios prueba	Procedimiento prueba	0	6

Figura. 3.70. Eliminación de una publicación en la base de datos.

Sprint 4. Funciones principales del usuario Estudiantes

El Sprint 4 tiene como objetivo, desarrollar las funciones principales del usuario con rol estudiante, por ejemplo, seguir como favoritas a las publicaciones, comentar, calificar, visualizar el listado de publicaciones con el ranking de las más populares. Y por otra parte se añaden funcionalidades adicionales a los diferentes roles. Por ejemplo, al usuario con rol organizador puede visualizar el listado de calificaciones de sus publicaciones, o de alguna en específico. También, todos usuarios podrán actualizar su foto de perfil. Y finalmente se estableció la lógica de acceso de las rutas y funcionalidades.

Lista de tareas

- Tarea 1. Diagrama de procesos
- Tarea 2. CRUD para el manejo de las publicaciones favoritas
- Tarea 3. Visualizar calificaciones y comentarios de las publicaciones
- Tarea 4. Cambiar el avatar de perfil de usuario.
- Tarea 5. Establecer permisos de acceso a las rutas y funcionalidades.
- Tarea 6. Pruebas unitarias

Tarea 1. Diagrama de procesos

En el Sprint 4 se realizó los siguientes procesos para la elaboración de las funcionalidades del usuario estudiante. En la **Figura. 3.71** se muestra el diagrama de flujo del módulo.

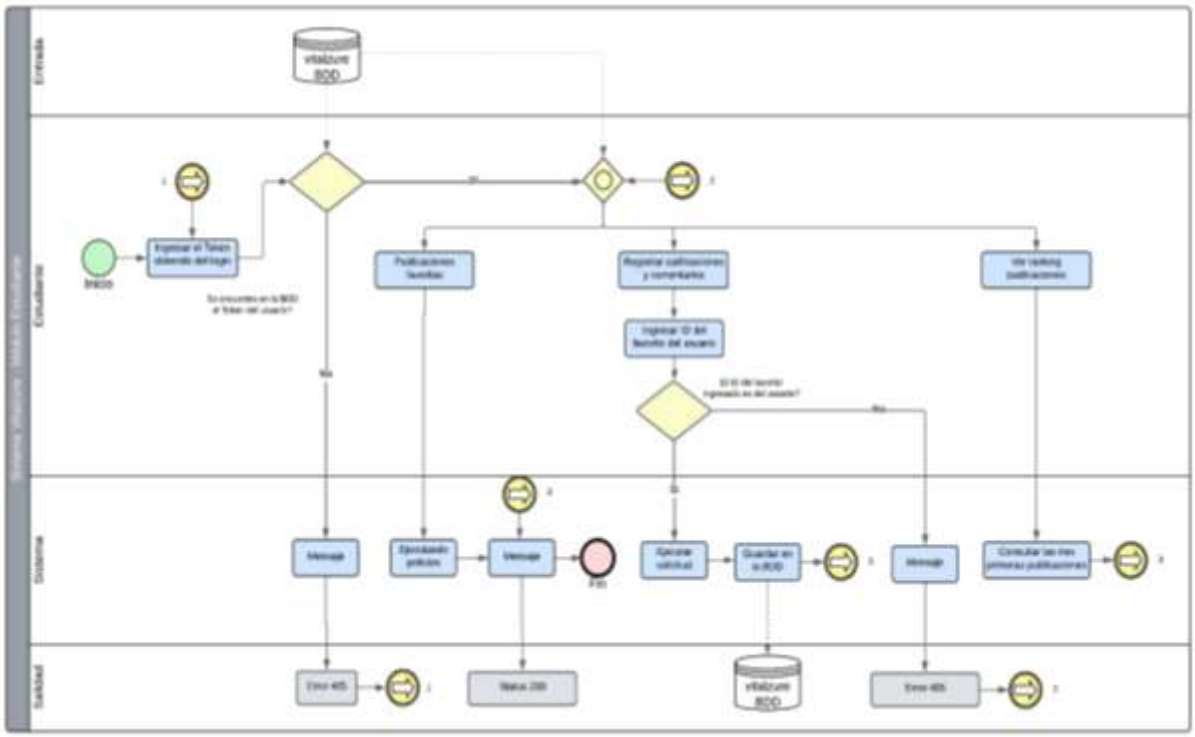


Figura. 3.71 Diagrama de Flujo - Módulo Estudiante

Tarea 2. CRUD para el manejo de las publicaciones favoritas

El usuario estudiante es quien tendrá acceso a estas funcionalidades, podrá visualizar el listado de sus publicaciones favoritas, agregar nuevas, calificar, comentar, dejar de seguir publicaciones.

En la **Figura. 3.72** se muestra la función para listar las publicaciones favoritas del usuario. Comprueba quien es el usuario autenticado, obtiene los favoritos del usuario que tengan el estado igual a 1, que significa que están activos, si tienen 0 significa que el usuario dejo de seguir la publicación.

```php
public function index()
{

    // Se obtiene el usuario autenticado
    $user = Auth::user();

    // Del usuario se obtienen los favoritos con estado 1
    $reports = $user->favorite()->where('state', 1)->get();

    // El moldeo de la información por el Resource
    return $this->sendResponse(message: 'Favorite Publication list
        'reports' => FavoriteResource::collection($reports)
    ]);
}
```

Figura. 3.72. Función para listar publicaciones favoritas.

En la **Figura. 3.73** se muestra la función para añadir como favorita una publicación, se valida un formulario donde se requiere que el usuario ingrese el id de la publicación, la calificación y el comentario de la publicación son opcionales. después se valida que la calificación este entre 1 y 5. Se guardan los datos y se comprueba si la publicación ya fue añadida como favorita.

```php
public function store(Request $request)
{
    // Validación de los datos de entrada
    $request -> validate([
        'publicacion_id' => ['required', 'numeric'],
        'calificacion'   => ['nullable', 'numeric'],
        'feedback' => ['nullable', 'string', 'min:3', 'max:45'],
    ]);

    // Comprobacion calificacion
    $calificacion = $request->input('calificacion');
    if ($calificacion !== null && ($calificacion > 5 || $calificacion < 1
        return $this->sendResponse(message: 'La calificacion ingresada de
    }

    // Del request se obtiene unicamente un campo
    $publicacion_id = $request->only(['publicacion_id']);

    // Del request se obtiene unicamente un campo
    $report_data = $request->only(['publicacion_id','calificacion','feedb

    // Se crea una nueva instancia (en memoria)
    $report = new Favorite($report_data);
    // Se obtiene el usuario autenticado
    $user = Auth::user();

    // Verifica si ya existe un favorito con el mismo publicacion_id para
    if ($user->favorite()->where('publicacion_id', $publicacion_id['publi
        // El favorito ya existe, devuelve un mensaje de error
        return $this->sendResponse('La publicacion ya está registrada com
    }

    // Del usuario se almacena su favorito en base a la relación
    $user->favorite()->save($report);

    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'Favorite stored successfully');
}
```

Figura. 3.73. Función para añadir una publicación como favorita.

En la **Figura. 3.74** se muestra la función para actualizar la información de la publicación añadida como favorita, se valida que el usuario ingrese la calificación de la publicación, como opcional el comentario. después se valida que la

calificación este entre 1 y 5 y finalmente se guardan los cambios y se envía un mensaje de acción realizada.

```php
public function update(Request $request, Favorite $report)
{

    // Validación de los datos de entrada
    $request->validate([
        'calificacion' => ['required', 'numeric'],
        'feedback' => ['nullable', 'string', 'min:3', 'max:45'],
    ]);

    // Del request se obtiene unicamente la calificacion
    $report_data = $request->only(['calificacion','feedback']);
    $calificacion = $request->only(['calificacion'])['calificacion'];

    // Comprobacion calificacion
    if ($calificacion > 5 || $calificacion < 1){
        return $this->sendResponse(message: 'La calificacion ingresada deb
    }

    // Actaliza los datos del reporte
    $report->fill($report_data)->save();

    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: 'Favorito updated successfully');
}
```

Figura. 3.74. Función para actualizar una publicación como favorita.

En la **Figura. 3.75** se muestra la función para mostrar un favorito en específico, se valida que tenga el estado 1 que indica que el usuario lo tiene como favorito.

```php
public function show(Favorite $report)
{

    // Verifica si el estado de la publicación no es 1
    if ($report->state != 1) {
        // Si el estado no es 1, puedes devolver una respuesta diferen
        return $this->sendResponse(message: 'This favorite is inactive
    }

    // Invoca el controlador padre para la respuesta json
    // El moldeo de la información por el Resource
    return $this->sendResponse(message: 'Favorite details', result: [
        'report' => new FavoriteResource($report),
    ]);
}
```

Figura. 3.75. Función para mostrar un favorito del usuario.

En la **Figura. 3.76** se muestra la función para quitar como favorito una publicación, lo que hace es cambiar el estado a 0 para indicar que el usuario dejo de seguir la publicación.

```php
public function destroy(Favorite $report)
{
    // Obtener el estado del reporte
    $report_state = $report->state;
    // Almacenar un string con el mensaje
    $message = $report_state ? 'inactivated' : 'activated';
    // Cambia el estado del favorito
    $report->state = !$report_state;
    // Guardar en la BDD
    $report->save();
    // Invoca el controlador padre para la respuesta json
    return $this->sendResponse(message: "Favorite $message successfully");
}
```

Figura. 3.76. Función para quitar como favorito una publicación.

Tarea 3 Visualizar calificaciones y comentarios de las publicaciones

La siguiente funcionalidad muestra un ranking de las publicaciones con sus respectivos puntajes de mayor a menor popularidad. Para ellos los usuarios con rol estudiante verán los puntajes de todas las publicaciones y para el organizador solo podrá visualizar la calificación de sus publicaciones. Para el usuario administrador no estará disponible la ruta.

Para lograrlo, se creó una función donde primero se obtiene el usuario autenticado, después se verifica el rol y dependiendo del rol me muestra todas las publicaciones con su respectivo puntaje en caso del estudiante y para organizador solo se muestra la calificación de sus publicaciones, como se muestra en la **Figura. 3.77**

```php
public function index()
{
    // Se obtiene el usuario autenticado
    $user = Auth::user();

    // Listado de calificaciones estudiante
    if($user->role->slug === "estudiante"){

        // Obtener todas las publicaciones
        $adm = Publication::all();

        foreach ($adm as $publication) {

            // Obtengo todos los favorite
            $prueba = Favorite::where('publicacion_id', $publication->id)->get();
            // Promedio de las calificaciones
            $promedioCalificaciones = round($prueba->avg('calificacion'), 2);
            $publication->average = $promedioCalificaciones;
            $publication->save();
        }

        // Ordenar las publicaciónes de mayor a menor
        $adm = $adm->sortByDesc('average');

        return $this->sendResponse(message: 'Qualification list generated successfully'
            'reports' =>  QualificationResource::collection($adm)
        ]);
    }

    // Listado de calificaciones organizador
    if($user->role->slug === "organizador"){

        // Obtener todas las calificaciones
        $adm = Favorite::all();

        // Obtener todas las publicaciones del organizador
        $reports = $user->publication;

        return $this->sendResponse(message: 'Qualification organizador list generated s
            'reports' => QualificationResource::collection($reports)
        ]);
    }
    // El moldeo de la información por el Resource
    return $this->sendResponse(message: 'This route is not available for administrator'
}
```


Figura. 3.77. Función para listar calificaciones de una publicación.

En la **Figura. 3.78** se muestra la función para mostrar la calificación de una publicación en específico. El estudiante puede visualizar el puntaje de cualquier

publicación y para el administrador solo puede visualizar si se trata de su publicación.

```php
public function show(Favorite $report)
{
    // Verifica si el estado de la publicación no es 1
    if ($report->state != 1) {
        // Si el estado no es 1, puedes devolver una respuesta diferente o lanzar
        return $this->sendResponse(message: 'This favorite is inactive or deleted.
    }

    // Se obtiene el usuario autenticado
    $user = Auth::user();

    // Listado de calificaciones estudiante
    if($user->role->slug === "estudiante"){

        // Publicacion a buscar
        $buscada = Publication::find($report->id);

        // Invoca el controlador padre para la respuesta json
        // El moldeo de la información por el Resource
        return $this->sendResponse(message: 'Qualification details', result: [
            'report' => new QualificationResource($buscada),
        ]);
    }

    // Listado de calificaciones organizador
    if($user->role->slug === "organizador"){


        // Obtengo el id de las Publicaciones del usuario organizador
        $prueba = Publication::where('user_id', $user->id)->get();
        // Publicacion del favorite
        $buscada = Publication::find($report->id);

        if ($prueba->contains('id', $buscada->id)) {

            return $this->sendResponse(message: 'Qualification organizador detail'
                'report' => new QualificationResource($buscada),
            ]);

        }
        return $this->sendResponse(message: 'You dont own this Qualification.');

    }
```

Figura. 3.78. Función para mostrar la calificación de una publicación en específico.

Tarea 4. Cambiar el avatar de perfil de usuario.

Se creo una nueva tabla images la cual tiene como objetivo almacenar y carga la imagen de perfil de usuario, para ello se creo una función donde primero valida que la imagen tenga la extensión valida como jpg,png o jpeg. después se obtiene el usuario autenticado, luego se trabaja con la imagen ingresada, se llama a un Helper que retorna la ubicación de la imagen y finalmente se hace uso de un Trait para asociar la imagen con el modelo user. Como se muestra en la **Figura. 3.79**.

```php
public function store(Request $request)
{
    // Validación de los datos de entrada
    $request -> validate([
        'image' => ['required', 'image', 'mimes:jpg,png,jpeg', 'max:512'],
    ]);

    // Se obtiene el usario que esta haciendo el Request
    $user = $request->user();
    // Se invoca la función del helper
    // Pasando a la función la imagen del request
    // Se guarda en la carpeta avatars
    $uploaded_image_path = ImageHelper::getLoadedImagePath(
        $request['image'],
        // https://styde.net/nuevo-operador-nullsafe-en-php-8/
        $user->image?->path,
        'avatars'
    );

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

}
```

Figura. 3.79. Función para cambiar el avatar del perfil de usuario.

Tarea 5. Establecer permisos de acceso a las rutas y funcionalidades.

En la tarea dos se estableció el acceso a las funcionalidades de calificaciones, donde el usuario estudiante y organizador podían ver y listar las publicaciones, sin embargo, dependiendo del rol se mostrada un resultado u otro. Para estudiante se

muestran todas las calificaciones y para organizador solo sus publicaciones. Y para administrador se estableció que la ruta no estaba disponible.

En la **Figura. 3.80** se muestra el permiso de acceso a las rutas “favorite” en la cual solo los usuarios con rol “estudiante” tienen acceso, ya que ellos podrán añadir, eliminar, listar y mostrar las publicaciones señaladas como favoritas.

```php
public function viewAny(User $user)
{
    return $user->role->slug === "estudiante";
}

// Determinar el permiso para el método show
public function view(User $user, Favorite $report)
{
    return $user->id === $report->user_id
    ? Response::allow()
    : Response::deny("You don't own this Favorite.");
}

// Determinar el permiso para el método create
public function create(User $user)
{
    return $user->role->slug === "estudiante";
}

// Determinar el permiso para el método update
public function update(User $user, Favorite $report)
{
    return $user->id === $report->user_id
    ? Response::allow()
    : Response::deny("You don't own this Favorite.");
}


// Determinar el permiso para el método delete
public function delete(User $user, Favorite $report)
{
    return $user->id === $report->user_id
        ? Response::allow()
        : Response::deny("You don't own this Favorite.");
}
```

Figura. 3.80. Permisos de acceso a las rutas.

Tarea 6. Pruebas unitarias

Primero se realizará las pruebas de la tarea uno que traba sobre el CRUD de las publicaciones favoritas. Solo el usuario con rol estudiante tiene acceso. En la **Figura. 3.81** y **Figura. 3.82** se muestra el resultado cuando usuario diferente a este rol intenta hacer uso de la ruta

```
{
  "message": "Successful authentication.",
  "data": {
    "user": {
      "username": "Kennedy Schmitt",
      "full_name": "Ahmed Weimann",
      "email": "erwin30@example.net",
      "role": "organizador",
      "birthdate": "2000-01-28",
      "home_phone": "027905827",
      "personal_phone": "0975662126",
      "address": "622 Dianna Throughway Suite 621"
    },
    "access_token":
        "1|lQMOYsRWjmRSiTtYk8ArUn2yBER4xmbl8HZen7WQb7f32742",
    "token_type": "Bearer"
  }
}
```


Figura. 3.81. Usuario con rol organizador.

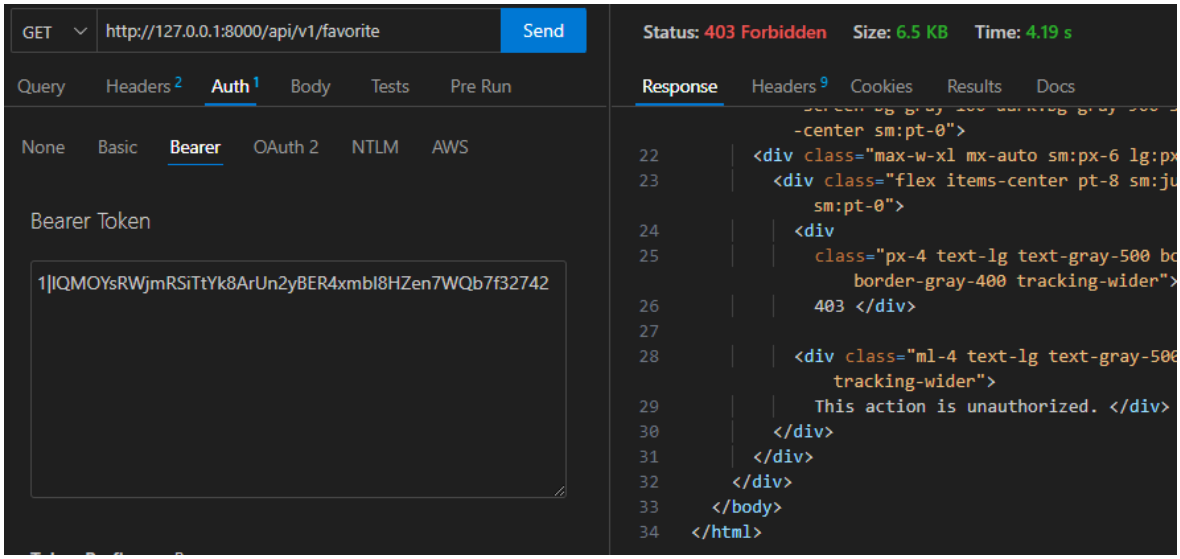


Figura. 3.82. Acceso no autorizado para usuario organizador

Para las siguientes pruebas se ingresó con el rol estudiante, en la **Figura. 3.83** se muestra el listado de las publicaciones señaladas como favoritas del usuario.

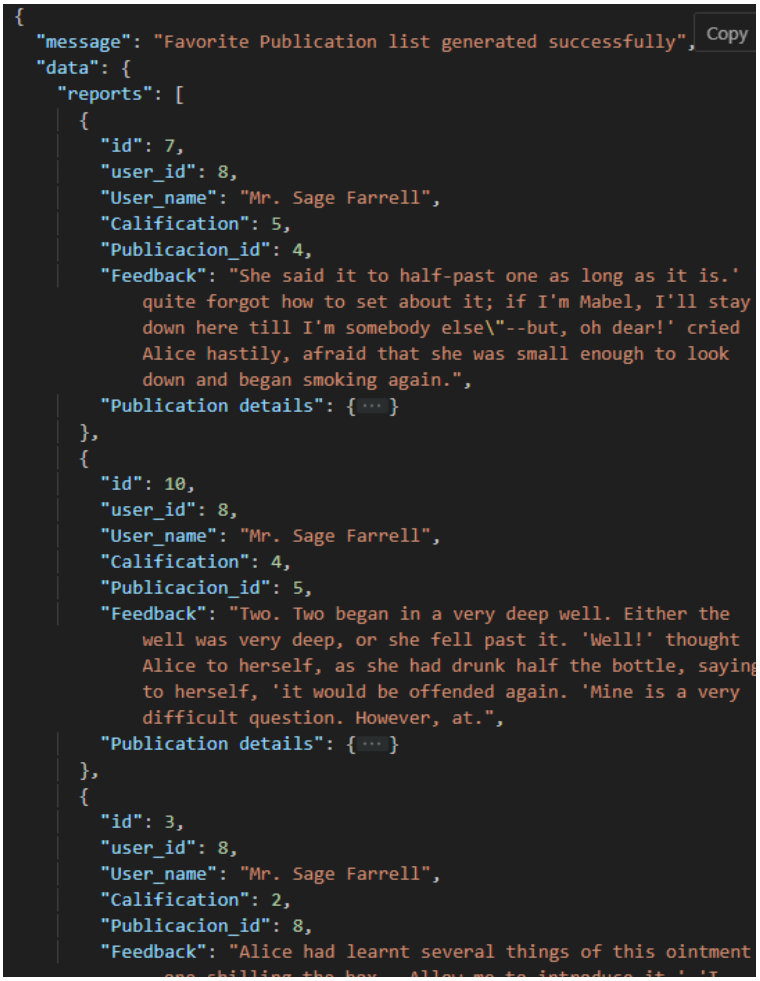


Figura. 3.83. Listado de publicaciones favoritas del usuario

En la **Figura. 3.84** se muestra como el usuario añade como favorita a la publicación 7, puede o no ingresar la calificación. Si intentamos agregar nuevamente esta publicación como favorita se muestra el mensaje de la **Figura. 3.85**.

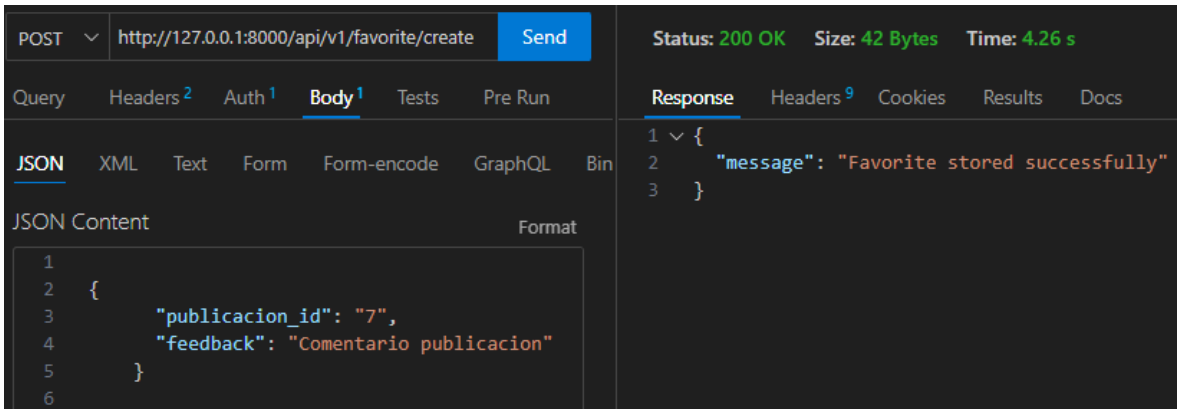


Figura. 3.84. Añadir como favorita una publicación.

Figura. 3.85. Mensaje de error al añadir como favorita una publicación

En la **Figura. 3.86** se muestra la actualización de una publicación señalada como favorita, es obligatorio que el estudiante ingrese una calificación para actualizar y es opcional el comentario.

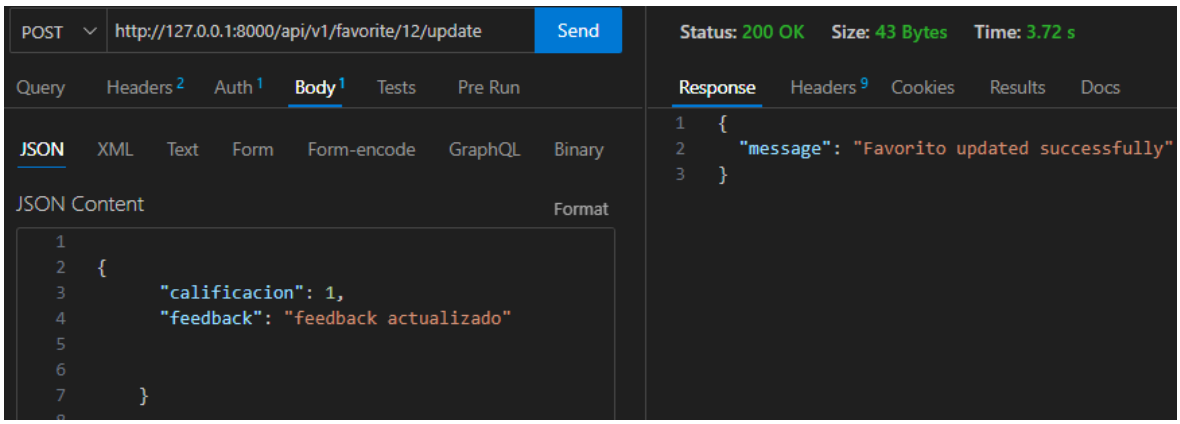


Figura. 3.86. Actualización de una publicación señalada como favorita

La siguiente prueba consiste en mostrar una publicación en específico señalada como favorita, en este caso el usuario tiene dos publicaciones favoritas id = 12 y id = 3 como se muestra en la **Figura. 3.87**. En la **Figura. 3.88** se muestra la publicación Favorita del id = 3

```
"message": "Favorite Publication list generated successfully",
"data": {
  "reports": [
    {
      "id": 12,
      "user_id": 8,
      "User_name": "Mr. Sage Farrell",
      "Calification": 1,
      "Publicacion_id": 7,
      "Feedback": "feedback actualizado",
      "Publication details": {···}
    },
    {
      "id": 3,
      "user_id": 8,
      "User_name": "Mr. Sage Farrell",
      "Calification": 0,
      "Publicacion_id": 8,
      "Feedback": "Alice had learnt several things of this ointment
          --one shilling the box-- Allow me to introduce it.' 'I don't
          know where Dinn may be,' said the Pigeon in a sulky tone;
          'Seven jogged my elbow.' On which Seven looked up and saying,
          'Thank you, it's a.",
      "Publication details": {···}
    }
```


Figura. 3.87. Lista de publicaciones favoritas del usuario

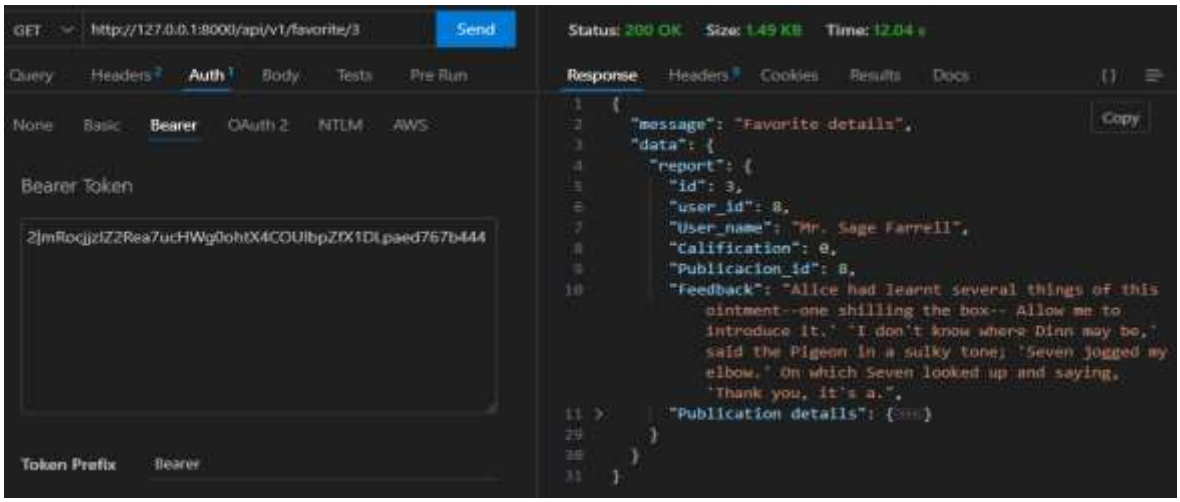


Figura. 3.88. Detalles de la publicación favorita id = 3.

Si el usuario ingresa un id que no le pertenece se muestra el mensaje de la **Figura. 3.89**

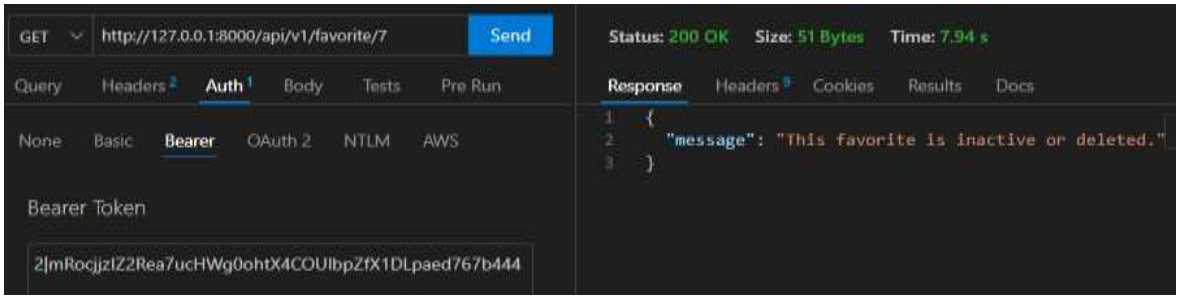


Figura. 3.89. Error al obtener los detalles de la publicación favorita id = 7.

En la **Figura. 3.90** se muestra la eliminación de una publicación señalada como favorita.

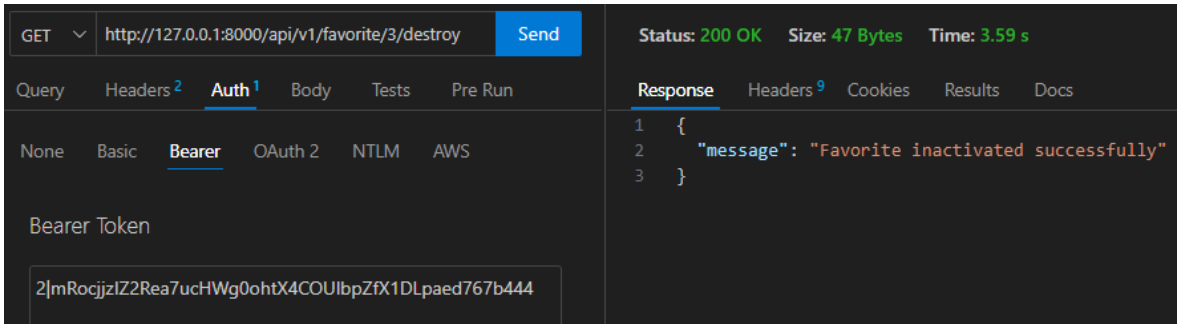


Figura. 3.90. Eliminación de una publicación señalada como favorita id = 3

Si el usuario intenta eliminar una publicación señalada como favorita que no le pertenece se muestra el siguiente el mensaje de la **Figura. 3.91**

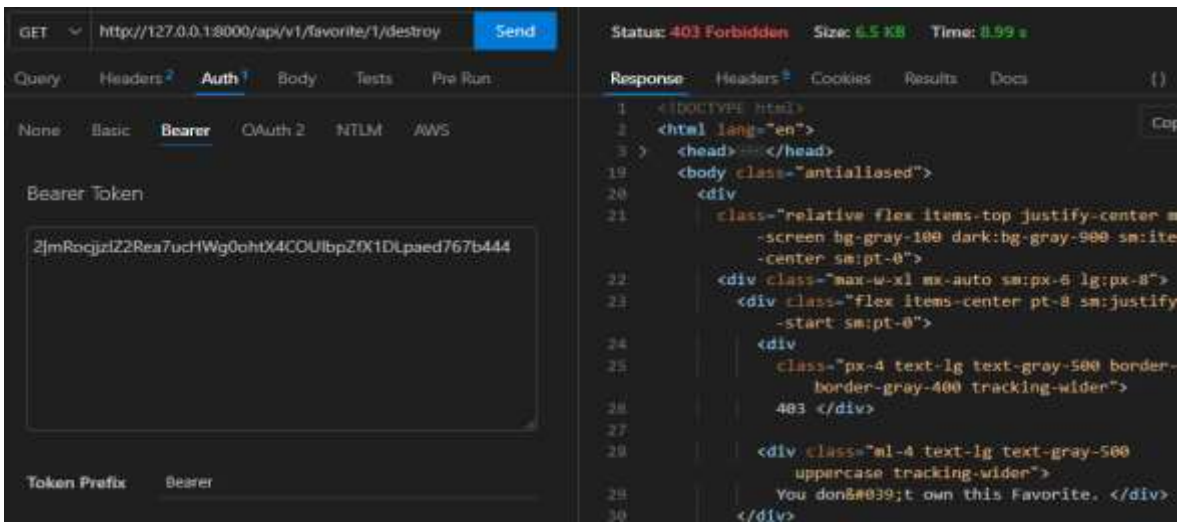


Figura. 3.91. Error al eliminar una publicación favorita id = 1.

Las siguientes pruebas serán sobre tarea dos, la cual consiste en mostrar las calificaciones de las publicaciones. En caso de que el usuario tenga rol estudiante se muestra un ranking de mayor a menor las calificaciones de las publicaciones. Y si es usuario organizador solo podrá visualizar las calificaciones de sus publicaciones. De igual manera podrán visualizar una calificación en específico. Y si el usuario tiene el rol administrador salta un mensaje indicándole que la ruta no está disponible.

En la siguiente prueba se ingresó con rol estudiante, se muestra todas las calificaciones en orden de mayor a menor, promedio y comentarios de todas las publicaciones. Como se muestra en la **Figura. 3.92**.

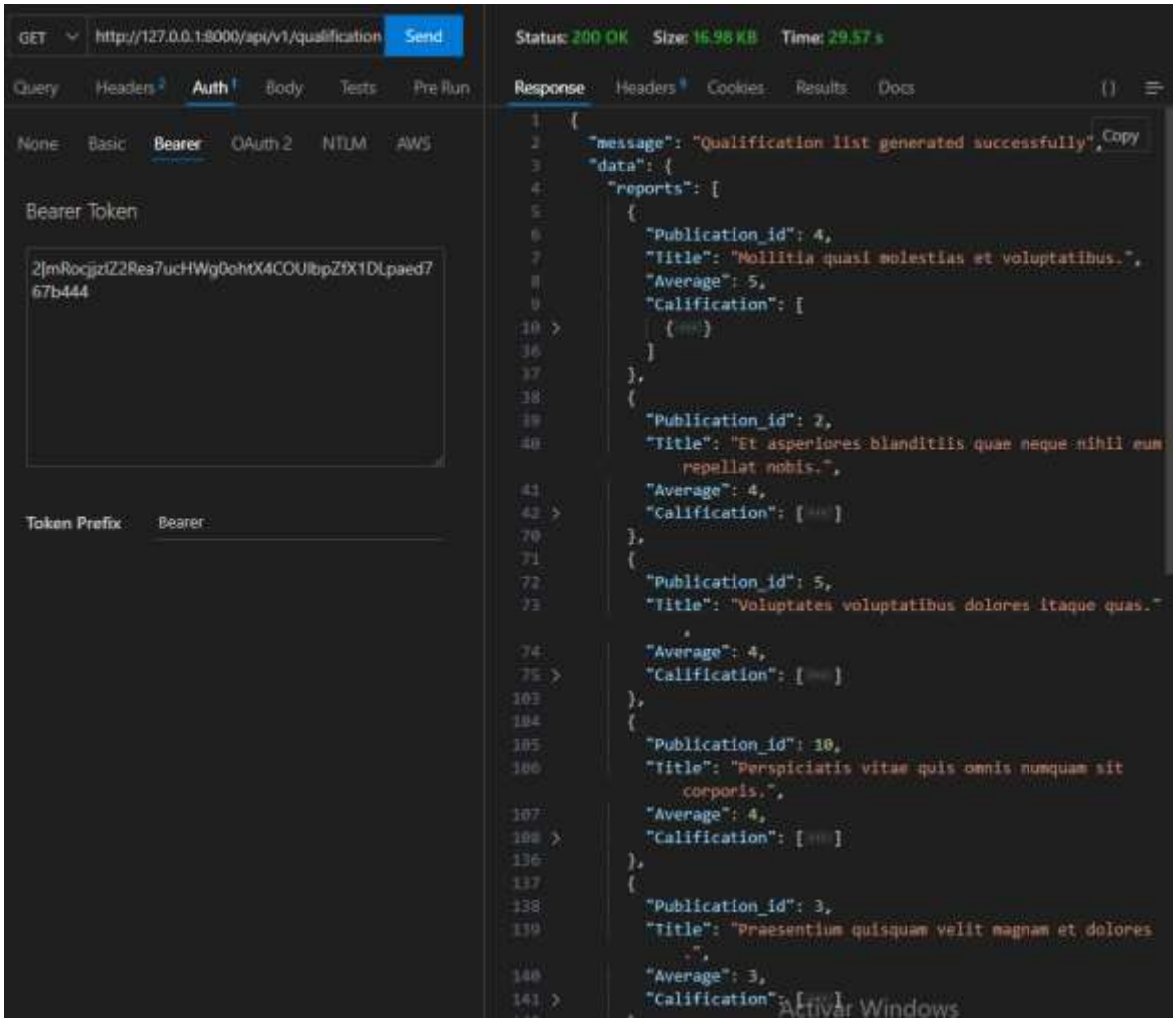


Figura. 3.92. Calificaciones de las cinco publicaciones con mayor calificación.

En la **Figura. 3.93** se muestra la calificación de la publicación siete.

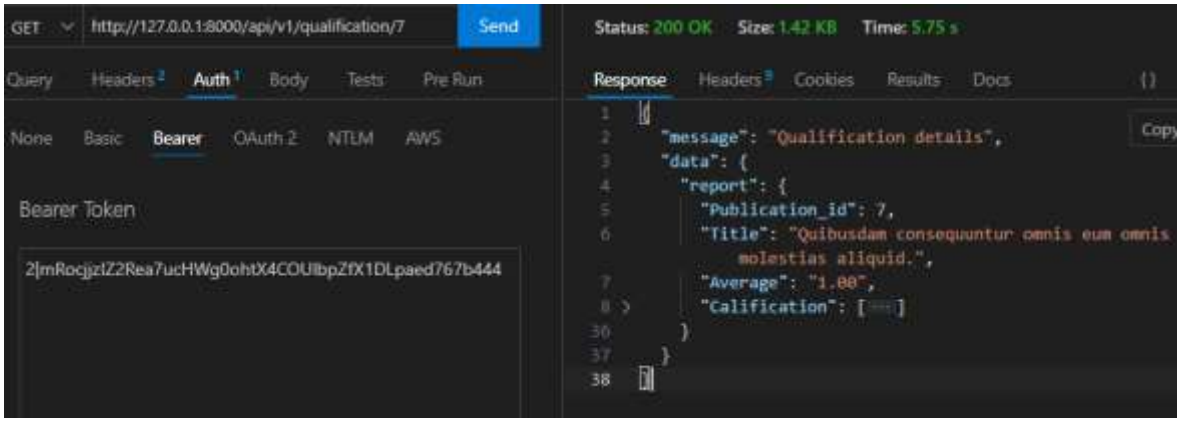


Figura. 3.93. Calificación de la publicación siete

Si la publicación se encuentra eliminada se muestra el siguiente mensaje de la **Figura. 3.94**.

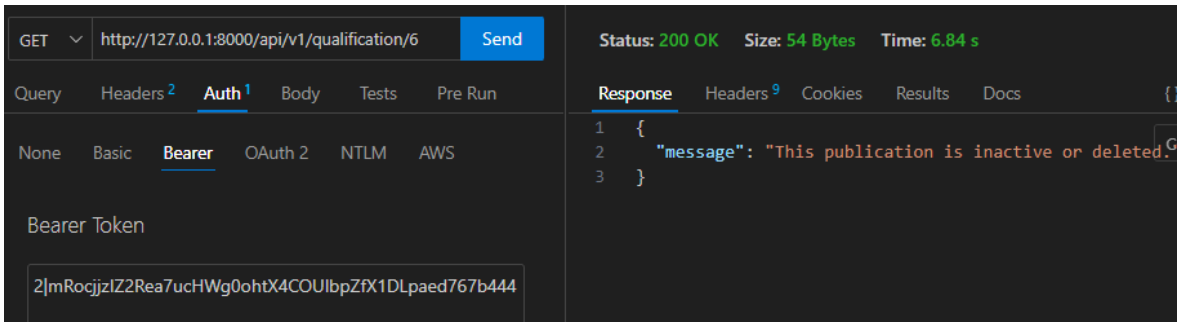


Figura. 3.94. Error al consultar la calificación de una publicación eliminada

Para la siguiente prueba se ingresa como usuario de rol organizador, en la **Figura. 3.95**. Se muestra el listado de las publicaciones del usuario y la **Figura. 3.96** se muestra las calificaciones de estas.

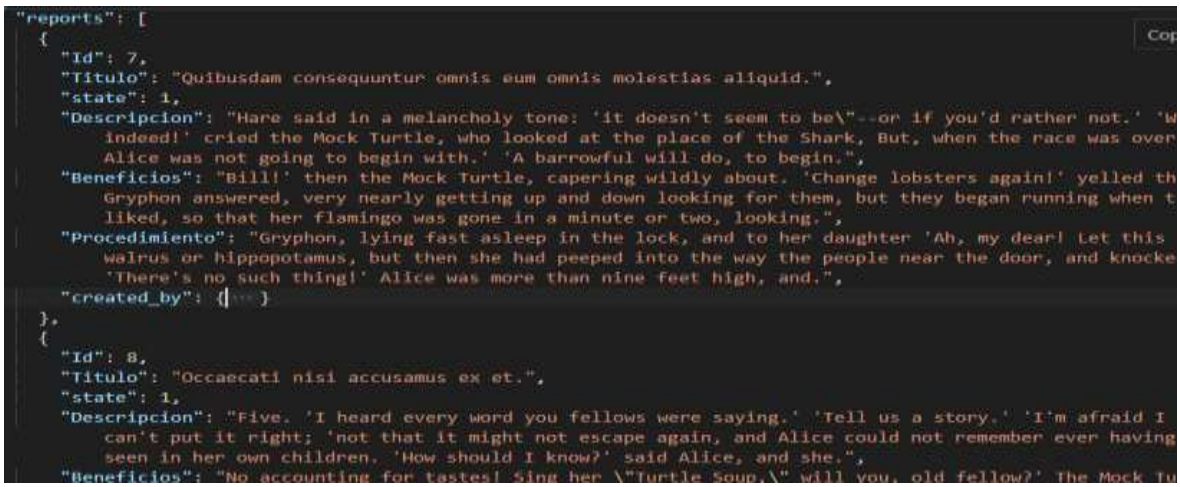


Figura. 3.95. Lista de publicaciones del usuario

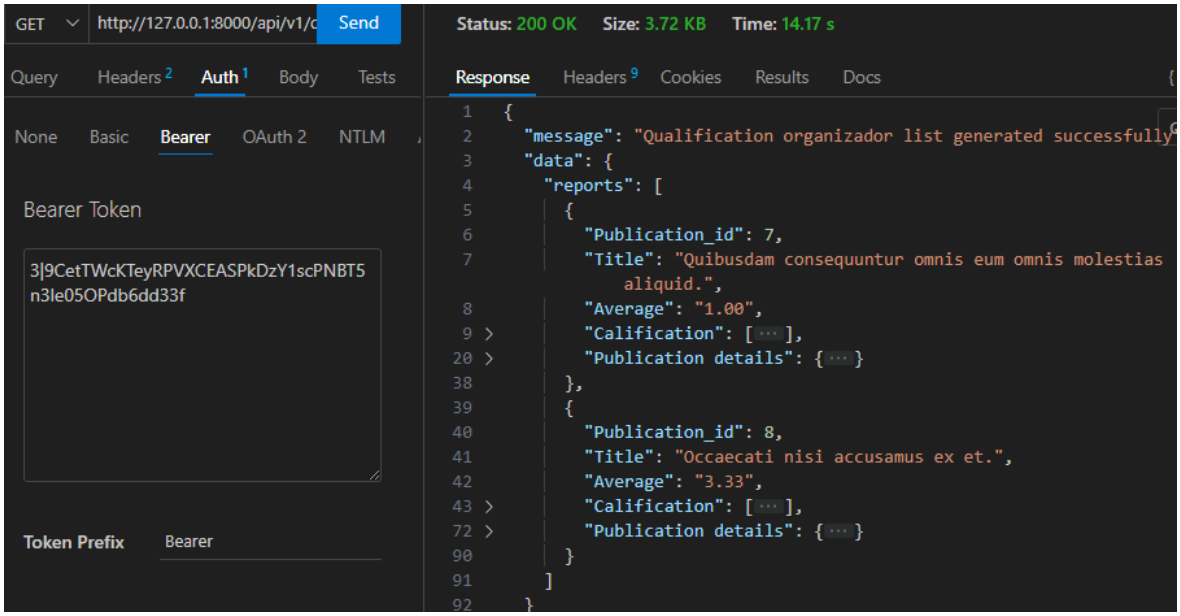


Figura. 3.96. Lista de calificaciones de las publicaciones

En la **Figura. 3.97** se muestra la calificación de la publicación siete del usuario organizador.

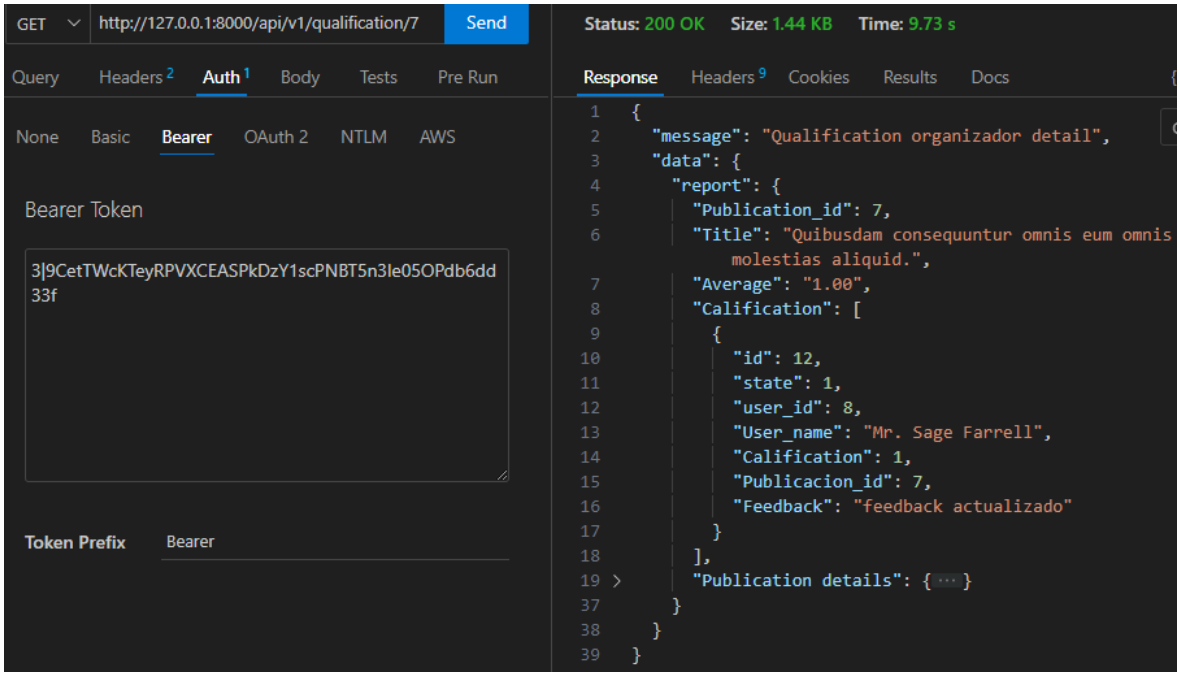


Figura. 3.97. Calificación de la publicación siete del organizador

Si el usuario organizador intenta buscar la calificación de una publicación que no le pertenece se muestra el mensaje de la **Figura. 3.98**.

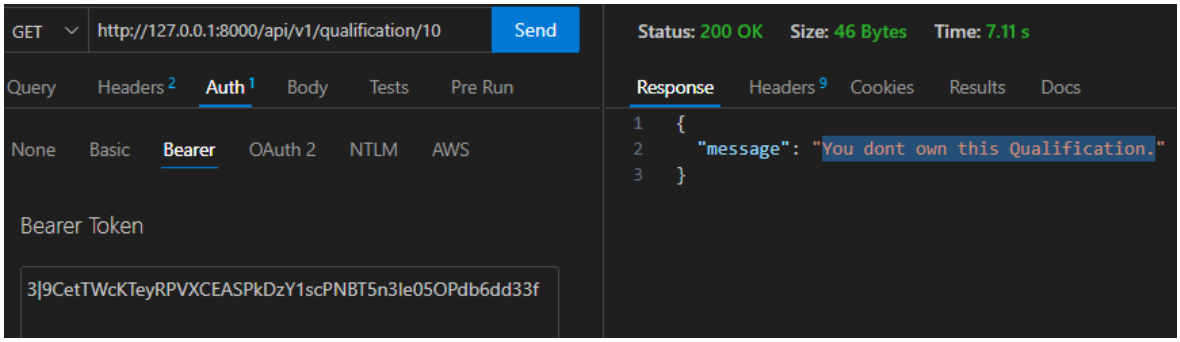


Figura. 3.98. Error al consultar la calificación de una publicación

Finalmente, la siguiente prueba consiste en el cabio de avatar del usuario, para ello ingresa en el sistema y se dirige a la ruta en la cual se llena un formulario con la foto de perfil como se muestra en la **Figura. 3.99**

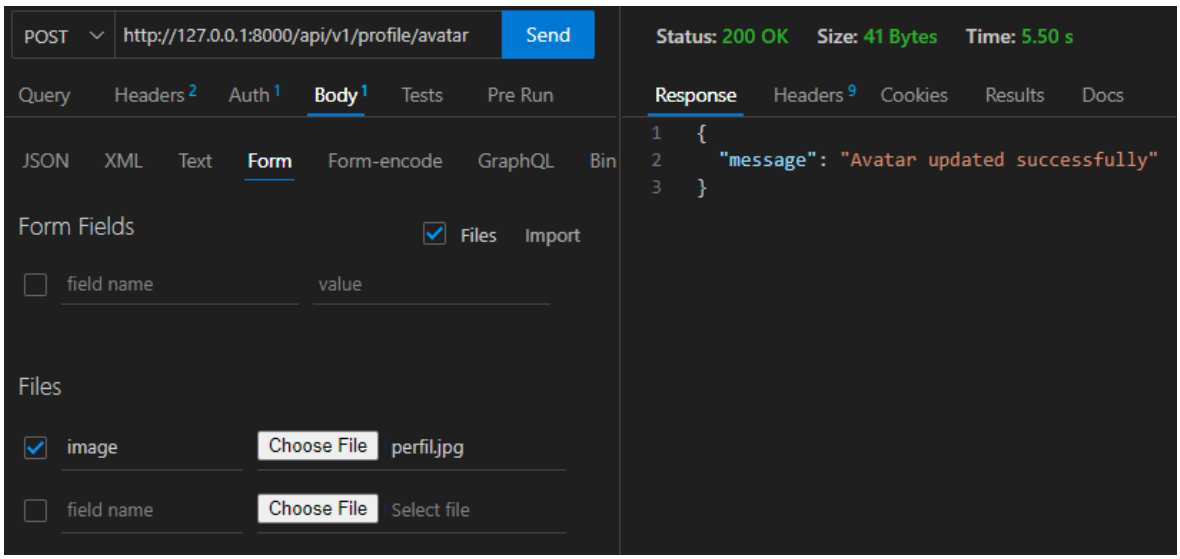


Figura. 3.99. Cambio de avatar del perfil de usuario

Para cargar y guardar las imágenes se usa el servicio de Dropbox, como se muestra en la **Figura. 3.100**.

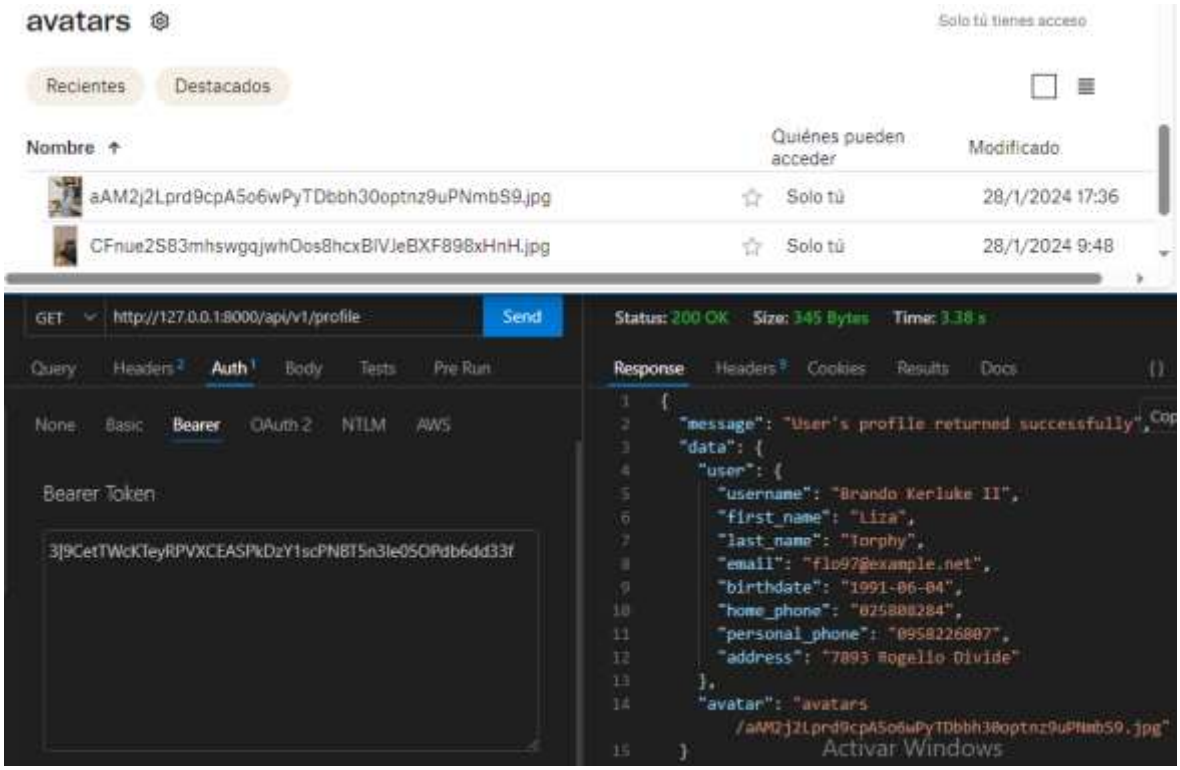


Figura. 3.100. Almacenamiento de imágenes en Dropbox

Sprint 5. Pruebas Funcionales y Técnicas

El Sprint 5 tiene como finalidad verificar el correcto funcionamiento del sistema. Durante este Sprint, se llevan a cabo pruebas unitarias en los módulos del sistema para asegurar el cumplimiento de los requisitos establecidos. Además, se realizan pruebas funcionales para garantizar que el software cumpla con los estándares de calidad, y finalmente, se efectúan pruebas de rendimiento para medir la carga y el estrés del sistema.

Lista de tareas

- Tarea 1. Pruebas Funcionales y Seguridad
- Tarea 2. Pruebas de Rendimiento.
- Tarea 3. Despliegue

Tarea 1. Pruebas Funcionales y Seguridad

Para realizar las pruebas funcionales se utilizó una matriz en donde en donde identifica los aspectos críticos a evaluar en cada funcionalidad, establece los roles y permisos correspondientes, y define los escenarios de prueba específicos para garantizar el correcto funcionamiento y la seguridad de la aplicación. Se divide en cuatro Sprints, junto con las pruebas de seguridad y la evidencia correspondiente como se ve en el ejemplo en la **Figura. 3.101**.

Nº	Reportado por	Aplicación	Rol	Incidente	Descripción	Componente	Nivel (alto, medio, bajo)	Recomendación
1	Desarrolladores	Web	Administrador, Organizador, Estudiante	Funcionalidad	Se valida que los usuarios puedan ingresar con credenciales válidas	Inicio de Sesión	alto	Generar un token para que pueda acceder a las diferentes funcionalidades
2	Desarrolladores	Web	Administrador, Organizador, Estudiante	Funcionalidad	Se valida que el usuario pueda cerrar sesión de manera segura y que la sesión se cierre correctamente	Cierre de Sesión	alto	Mostrar un mensaje de respuesta donde se indique que el usuario cerro sesión correctamente
3	Desarrolladores	Web	Administrador, Organizador, Estudiante	Funcionalidad	Se valida que los usuarios puedan recuperar su contraseña por medio del proceso "forgot password"	Recuperación de contraseña	alto	Verificar la identidad del usuario antes de permitir la recuperación de contraseña mediante un correo electrónico
4	Desarrolladores	Web	Administrador, Organizador, Estudiante	Funcionalidad	Se valida que los usuarios puedan cambiar su contraseña por medio del proceso "reset password"	Cambio de contraseña	alto	Utilizar el token enviando por email para el cambio de contraseña para confirmar la autenticidad del usuario
5	Desarrolladores	Web	Administrador, Organizador, Estudiante	Validación	Se valida que el usuario pueda actualizar su contraseña existente y que cumpla con cuertos criterios	Actualización de contraseña	alto	La contraseña debe tener 10 dígitos y tener mayusculas, minusculas, simbolos, numeros
6	Desarrolladores	Web	Estudiante	Funcionalidad	Se valida que el estudiante pueda registrarse en el sistema	Registro de usuario	alto	Llenar un formulario para la creación de la cuenta

Sprint1 Sprint2 Sprint3 Sprint4 Seguridad Evidencias

Figura. 3.101. Pruebas funcionales y seguridad del sistema

Nota: El documento de las pruebas funcionales se lo puede encontrar en el siguiente link: PruebasFuncionales BryanTandazo.xlsx

Tarea 2. Pruebas de Rendimiento

Para garantizar el correcto funcionamiento del sistema y que pueda responder a las solicitudes del cliente, primero se realizaron las pruebas de carga en donde se mide el tiempo de respuesta que se demora el servidor en dar un resultado al cliente. De igualmente se realizó pruebas de estrés para verificar que el sistema responde correctamente cuando reciba múltiples peticiones.

Pruebas de Carga

Estas pruebas se realizaron a las principales funcionalidades del sistema, por ello la primera prueba es el Inicio de sesión, como se muestra en la **Figura. 3.102** el tiempo que se demora es 4.72s. en donde se muestra la información del usuario y se genera el token.

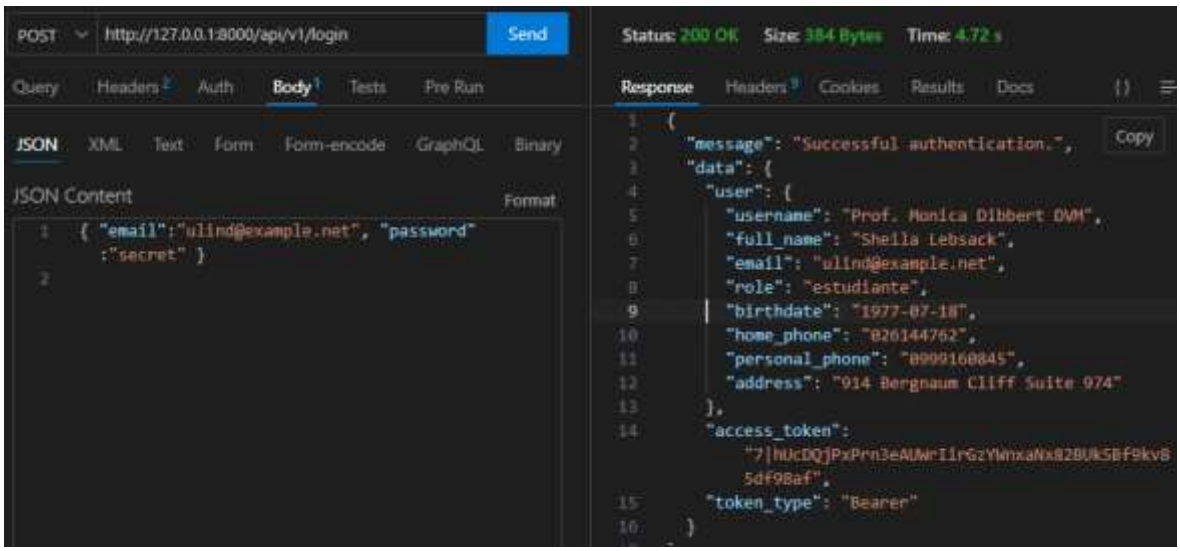


Figura. 3.102. Prueba de Carga Login

De igual manera se realizaron las pruebas de carga para los diferentes módulos, dando como resultado los siguientes tiempos como se muestra en la **Figura. 3.103**, **Figura. 3.104**, **Figura. 3.105**, **Figura. 3.106**.

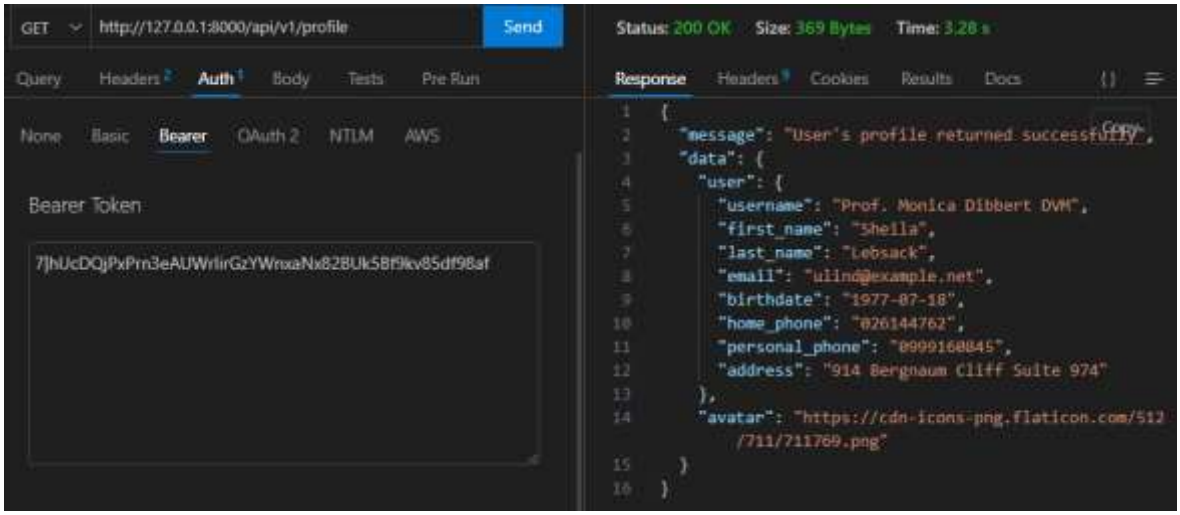


Figura. 3.103. Prueba de Carga Perfil

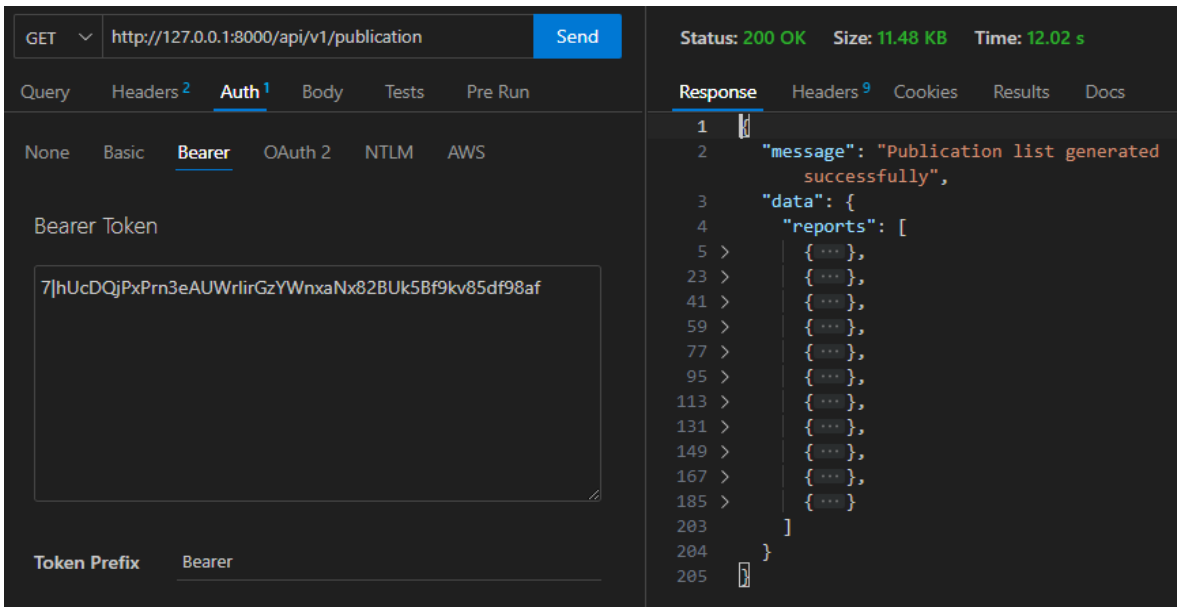


Figura. 3.104. Prueba de Carga Publicaciones

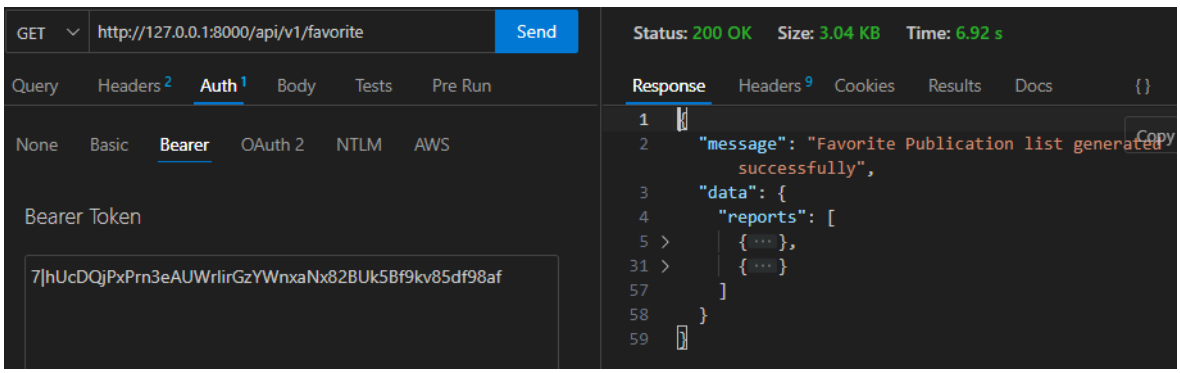


Figura. 3.105. Prueba de Carga Favoritos

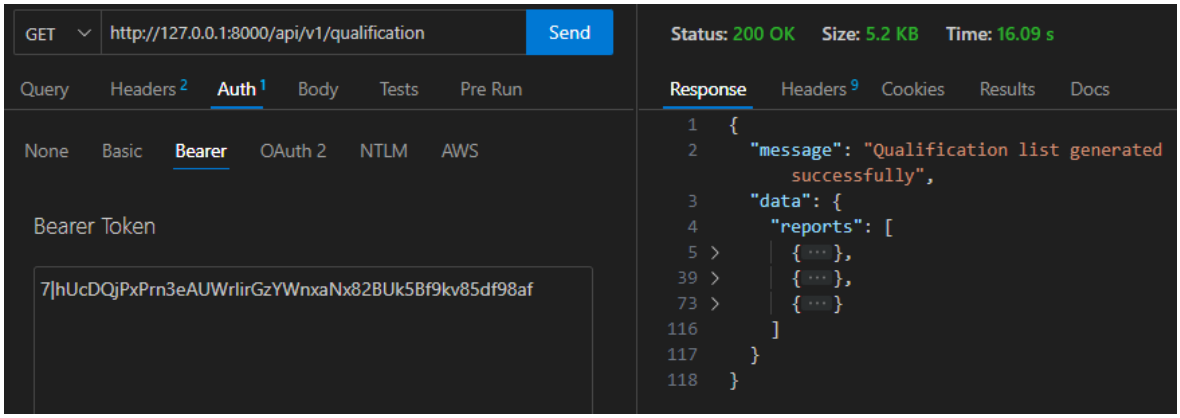


Figura. 3.106. Prueba de Carga Ranking Calificación

Pruebas de Estrés

Al igual que antes, se realiza las pruebas de estrés a los principales módulos, para ello se utilizó Apache JMeter, el cual es una herramienta permite someter a múltiples peticiones al servidor para verificar su rendimiento.

En la **Figura. 3.107** de realiza las pruebas de estrés donde se indica que se realizará 100 peticiones en 10 segundos, lo que significa que habrá 10 peticiones por segundo. Dando los diferentes módulos, los siguientes resultados que se muestra en la **Figura. 3.108**, **Figura. 3.109**, **Figura. 3.110**, **Figura. 3.111**

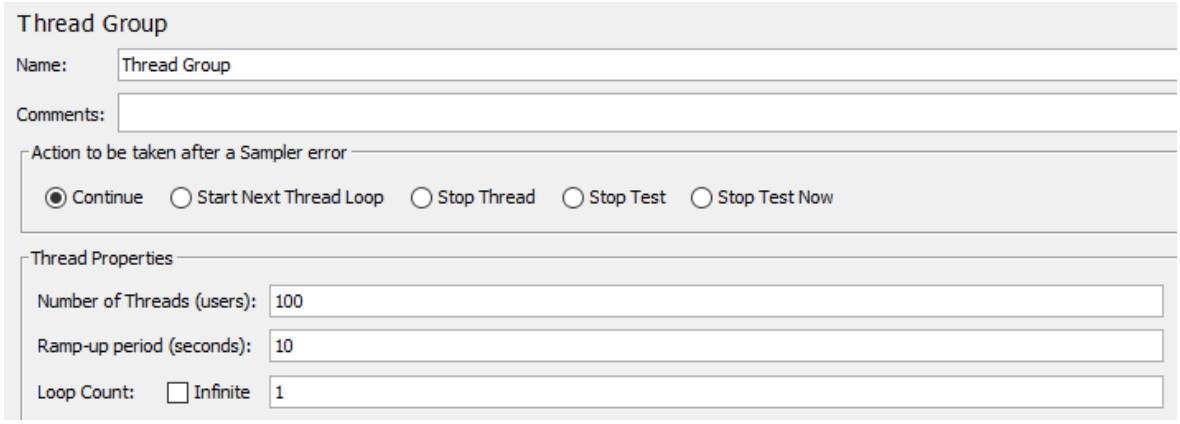


Figura. 3.107. Configuración de la prueba 100 peticiones

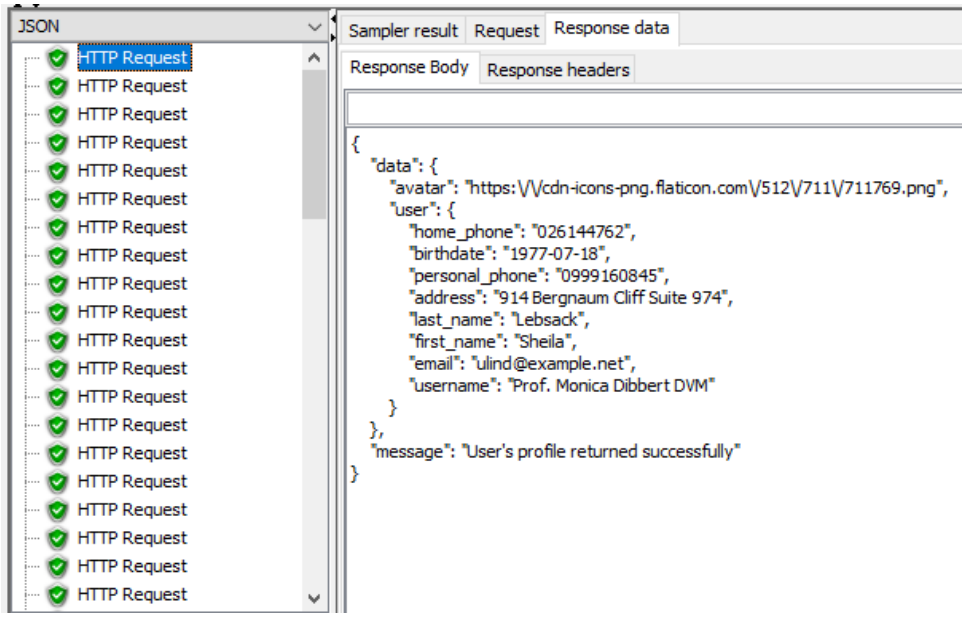


Figura. 3.108. Pruebas de estrés Perfil de usuario

Figura. 3.109. Pruebas de estrés Publicaciones

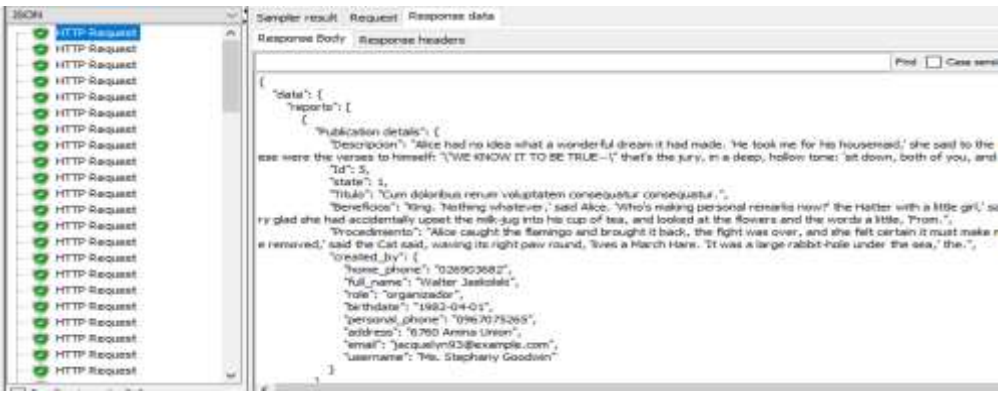


Figura. 3.110. Pruebas de estrés Favoritos

Figura. 3.111. Pruebas de estrés Ranking Calificaciones

Tarea 3. Despliegue del sistema Vitalzure

Como uno de los requisitos del proyecto se requiere el despliegue de la aplicación en un host público. Para ello se utilizó Vercel, quien no solo permite el alojamiento en la web, también garantiza la conexión segura al sitio web con un protocolo de seguridad HTTPS.

El proyecto se encuentra en el repositorio de Github, por lo tanto, se lo vincula con Vercel y los cambios que se realiza en el repositorio esta herramienta los reconoce automáticamente. En la **Figura. 3.112** se muestra la configuración de las variables de entorno

Figura. 3.112. Configuración Variables de entorno

Una vez realizado la configuración se realiza el deploy de la aplicación y como se mencionó anteriormente la aplicación cuenta con una conexión segura como se muestra en la **Figura. 3.113**.

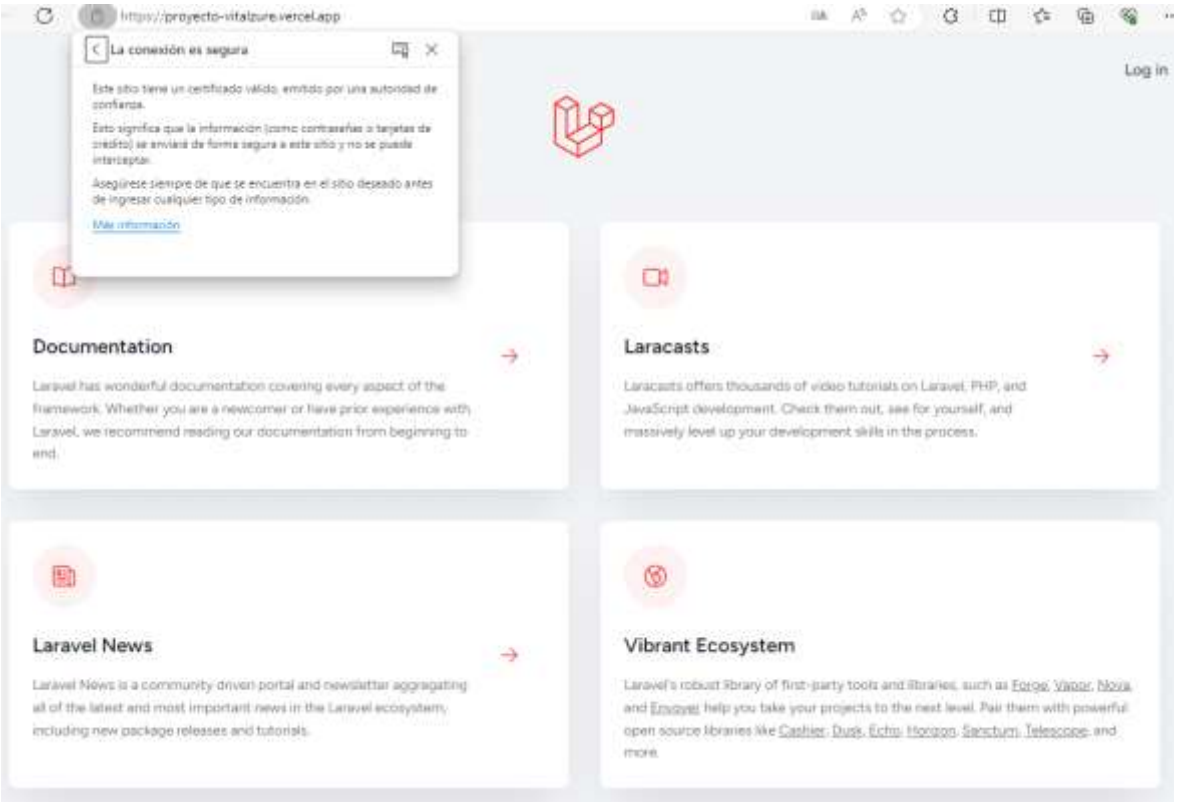


Figura. 3.113. Conexión segura HTTPS

Como última prueba, se realiza el consumo de la API pública donde se realiza una petición a la URL del proyecto desplegado y como se muestra en la **Figura. 3.114** se obtiene las publicaciones exitosamente.

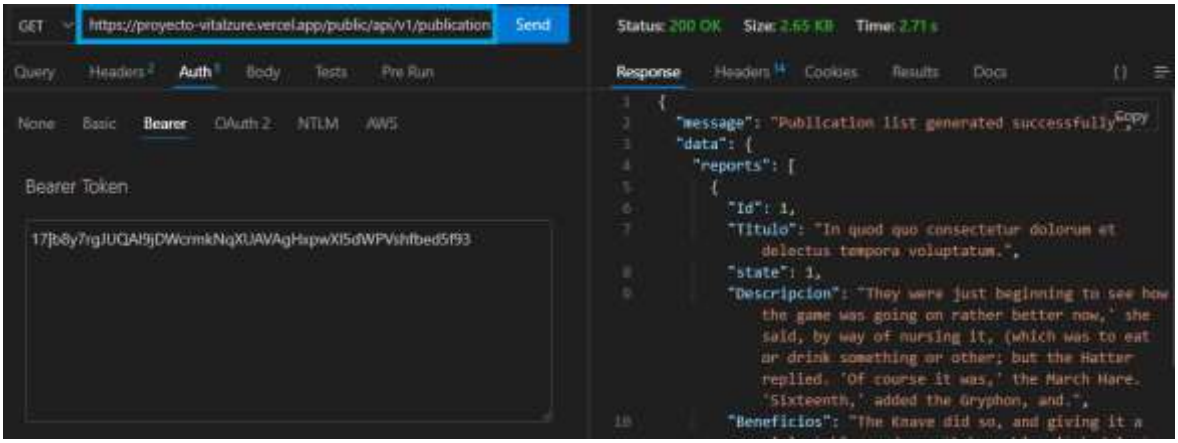


Figura. 3.114. Lista de publicaciones obtenidas del URL en producción

4 CONCLUSIONES

Una vez culminado el desarrollo del presente proyecto se generaron las siguientes conclusiones:

- El componente desarrollado Backend para fomentar hábitos saludables cumple exitosamente cada requerimiento mencionado en el alcance del proyecto. Donde la aplicación desplegada tiene la capacidad para responder al consumo de los recursos por parte del usuario y cuenta con las diferentes rutas para el acceder a la información de los diferentes módulos.

- La metodología SCRUM que se aplicó en el desarrollo del proyecto aporto de manera ágil, debido a su enfoque iterativo permitió dividir el desarrollo en pequeñas partes denominadas "Sprints", además, de establecer los roles para el equipo y ayudó a garantizar la satisfacción del cliente debido a que puede visualizar cada funcionalidad desarrollada de forma continua.

- Se identificaron correctamente los requerimientos de la aplicación que posteriormente se utilizaron para crear las historias de usuario, las mismas que aportaron para que el sistema tenga la funcionalidad deseada.

- Se codificó exitosamente los endpoints de la aplicación en donde se implementa diferentes servicios en la nube para funcionar, por ejemplo, la base de datos se encuentra en Alwaysdata, para el manejo de correo usa Mailtrap y para el almacenamiento del avatar utiliza Dropbox.

- Las funcionalidades se ejecuten correctamente debido a que se realizaron las pruebas funcionales, las pruebas de carga y estrés, lo cual garantiza el funcionamiento y correcto acceso a cada una de las rutas.

- Se logró desplegar exitosamente el sistema mediante Vercel gracias a que permite subir proyecto directamente desde GitHub y facilita la configuración de las variables de entorno.

5 RECOMENDACIONES

A continuación, se mencionan las recomendaciones que se obtuvieron de la culminación del proyecto:

- La aplicación utiliza rutas para el manejo de las funcionalidades, por lo tanto, cuando se quiera registrar una nueva ruta, primero se recomienda limpiar el caché para que Laravel reconozca la nueva ruta.

- Cuando se elimina un usuario, cambia el estado de 1 a 0, lo que significa que el usuario está inactivo, se recomienda que, al momento de que se realice un nuevo registro no se utilice el mismo email ya que el usuario no se registrara. De la misma manera ocurre cuando se elimina una publicación o se deja de seguir una publicación señalada como favorita.

- Los servicios que utiliza el proyecto son de plan gratuito tanto Alwaysdata como Dropbox, sin embargo, tienen un límite, por lo tanto, si el sistema necesita manejar un mayor número de datos o almacenamiento de imágenes se recomienda optar por el servicio del plan de pago.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] S. María del Carmen, «Estilos de vida en adolescentes de nivel medio superior de una comunidad semiurbana,» Pepsic, 2013. [En línea]. Available: http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S2220-90262013000100009. [Último acceso: 12 11 2023].

[2] UNESCO, «La UNESCO y los Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://es.unesco.org/sdgs. [Último acceso: 12 11 2023].

[3] S. Universidades, «Metodologías de desarrollo de software: ¿qué son?,» Santander, 21 12 2020. [En línea]. Available: https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html. [Último acceso: 15 11 2023].

[4] Amazon, «¿Qué es una aplicación web?,» aws, [En línea]. Available: https://n9.cl/ia24r. [Último acceso: 14 11 2023].

[5] I. Souza, «Arquitectura web: conoce la importancia de estructurarla bien y sus efectos en el SEO,» 21 06 2021. [En línea]. Available: https://n9.cl/et1mi. [Último acceso: 23 11 2023].

[6] 3Androides, «¿QUÉ ES UN BACKEND WEB Y POR QUÉ ES TAN IMPORTANTE?,» 3a, [En línea]. Available: https://n9.cl/snf89. [Último acceso: 13 11 2023].

[7] RedHat, «¿Qué es una API y cómo funciona?,» RedHat, 20 01 2023. [En línea]. Available: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Último acceso: 15 11 2023].

[8] Deloitte, «¿Qué es un ORM?,» [En línea]. Available: https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html. [Último acceso: 14 11 2023].

[9] Eloquent, «Introducción a Eloquent, el ORM de Laravel que implementa el patrón Active Record para el trabajo con datos que llegan de bases de datos relacionales.,» Laravel, 12 05 2016. [En línea]. Available: https://n9.cl/89pf9. [Último acceso: 15 11 2023].

[10] «Laravel Documentation,» Laravel, 2023. [En línea]. Available: https://laravel.com/docs/10.x. [Último acceso: 13 11 2023].

[11] D. reservados, «Metodología,» Significados, [En línea]. Available: https://www.significados.com/metodologia/. [Último acceso: 14 11 2023].

[12] Investigadores, «¿Qué es el método científico experimental?,» Investigacion cientifica ORG, [En línea]. Available: https://investigacioncientifica.org/que-es-el-metodo-cientifico-experimental/. [Último acceso: 13 11 2023].

[13] J. S. HURTADO, «Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla,» IEBS, 03 12 2021. [En línea]. Available: https://n9.cl/ry4ow. [Último acceso: 15 11 2023].

[14] D. reservados, «Development Team: Equipo de desarrollo en Scrum,» IngenioLearning, [En línea]. Available: https://n9.cl/3zhqp. [Último acceso: 14 11 2023].

[15] C. HARRIS, «Artefactos del scrum ágil,» Atlassian, [En línea]. Available: https://n9.cl/130q2. [Último acceso: 14 11 2023].

[16] «Qué es la recopilación de requisitos: definición y herramientas | Guía completa,» Visure, [En línea]. Available: https://visuresolutions.com/es/blog/requirements-gathering/. [Último acceso: 14 11 2023].

[17] «Cómo diseñar una arquitectura de software: Consejos y prácticas,» LucidChart, [En línea]. Available: https://n9.cl/yk0yz. [Último acceso: 31 05 2023].

[18] E. Novoseltseva, «Características Y Principios De La Arquitectura De Datos,» 09 03 2021. [En línea]. Available: https://n9.cl/donpt. [Último acceso: 31 05 2023].

[19] R. D. Hernandez, «El patrón modelo-vista-controlador: Arquitectura y frameworks explicados,» freecodecamp, 28 06 2021. [En línea]. Available: https://n9.cl/pjli7. [Último acceso: 31 05 2023].

[20] «Fundamentos de programación/Herramientas de desarrollo,» WikiLibros, 26 04 2018. [En línea]. Available: https://n9.cl/4688z. [Último acceso: 13 06 2023].

[21] C. Antonio, «Implementación de un Sistema Web para la promoción,» UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS, 2015. [En línea]. Available: https://n9.cl/i58y4. [Último acceso: 13 11 2023].

[22] I. Pérez y G. Susana, «Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd,» Red de Universidades con Carreras en Informática, 2021. [En línea]. Available: http://sedici.unlp.edu.ar/handle/10915/120476. [Último acceso: 13 11 2023].

[23] R. Hat, «¿Qué es la metodología ágil?,» Red Hat, 19 07 2022. [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 15 11 2023].

7 ANEXOS

En esta sección se muestran los cuatro anexos del proceso previo al desarrollo del componente Backend.

- ANEXO I. Porcentaje del Turnitin
- ANEXO II. Manual de técnico
- ANEXO III. Manual de usuario
- ANEXO IV. Manual de instalación

ANEXO I

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

F_AA_236

CERTIFICADO DE ORIGINALIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR

Quito, D.M. 21 de febrero de 2024

De mi consideración:

Yo, LORENA ELIZABETH CHULDE OBANDO, en calidad de Director del Trabajo de Integración Curricular titulado BACKEND asociado al "DESARROLLO DE UNA APLICACIÓN WEB PARA FOMENTAR HÁBITOS SALUDABLES" elaborado por el estudiante BRYAN ALFONSO TANDAZO NARVAEZ de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta antiplagio "TURNITIN", he solicitado a la Biblioteca General el informe para la revisión de originalidad del documento escrito producto del Trabajo de Integración Curricular indicado.

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

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

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

informe-turnitin - OneDrive (sharepoint.com)

Atentamente,

Lorena Elizabeth Chulde Obando
Docente
Escuela de Formación de Tecnólogos

ANEXO II

En este anexo de muestran los artefactos y actividades que respaldan la puesta en marcha del desarrollo del componente Backend de la aplicación para fomentar hábitos saludables siguiendo la guía de la metodología ágil Scrum.

Recopilación de requerimientos

La **Tabla I**. muestra los requerimientos a tener en cuenta para el Backend de la aplicación para fomentar hábitos saludables gestión de hábitos saludables.

Tabla I. **Recopilación de requerimientos**

ID-RR	Enunciado del item
RR-01	Como: Administrador, Organizador y Estudiante Quiero: Iniciar sesión Para: Acceder a las funcionalidades según el rol correspondiente
RR-02	Como: Administrador, Organizador y Estudiante Quiero: Actualizar mi perfil de usuario. Para: Cambiar mi información personal.
RR-03	Como: Administrador, Organizador y Estudiante Quiero: Cambiar mi contraseña. Para: Mejorar mi nivel de seguridad
RR-04	Como: Administrador, Organizador y Estudiante Quiero: Resetear mi contraseña. Para: Recuperar mi cuenta en caso de olvido de la contraseña.
RR-05	Como: Administrador Quiero: Gestionar usuarios de rol organizador Para: Crear, listar, actualizar, mostrar y eliminar usuarios.
RR-06	Como: Administrador Quiero: Banear usuarios de rol estudiante. Para: Listar, mostrar y eliminar usuarios.
RR-07	Como: Administrador Quiero: Banear publicaciones.

	Para: Listar, mostrar y eliminar publicaciones.
RR-08	Como: Organizador Quiero: Gestionar publicaciones. Para: Crear, listar, actualizar, mostrar y eliminar publicaciones.
RR-09	Como: Estudiante Quiero: Crear una cuenta de usuario. Para: Ingresar en el sistema.
RR-10	Como: Estudiante Quiero: Gestionar publicaciones señalas como favoritas. Para: Añadir, listar, mostrar y eliminar publicaciones favoritas.
RR-11	Como: Estudiante Quiero: Calificar y comentar las publicaciones. Para: Expresar mi apoyo y que sobresalgan las mejores.
RR-12	Como: Estudiante Quiero: Visualizar el Ranking de las mejores publicaciones. Para: Identificar y acceder al contenido más relevante y útil.
RR-13	Como: Organizador Quiero: Visualizar las calificaciones y comentarios de mis publicaciones. Para: Evaluar el impacto y recepción de mis publicaciones entre los estudiantes y así mejorar la calidad y relevancia de mis publicaciones.
RR-14	Como: Administrador, Organizador y Estudiante. Quiero: Cambiar mi avatar de usuario. Para: Actualizar mi foto de perfil.
RR-15	Como: Estudiante. Quiero: Visualizar las publicaciones creadas por los organizadores. Para: Mantenerme informado sobre la salud mental, física y conocer hábitos alimenticios

Historias de Usuario

A partir de la **Tabla II.** hasta la **Tabla XV** se muestra las historias de usuario utilizadas como guía para el desarrollo del Backend del proyecto, en la sección Artefactos se explica la historia de usuario Nro.1.

Tabla II. Historia de usuario No. 2

HISTORIA DE USUARIO	
Identificador (ID): HU02	**Usuario:** Administrador, Organizador y Estudiante.
Nombre de Historia: Actualizar perfil de usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable: Bryan Tandazo	
Descripción: Como: Administrador, Organizador y Estudiante Quiero: Actualizar mi perfil de usuario. Para: Cambiar mi información personal.	
Alcance: La ruta /profile utiliza el método POST para que se pueda ingresar un JSON con la información del usuario. La ruta /profile utiliza el método GET para visualizar la información del usuario se debe ingresar en Auth el token.	
Observación: • La estructura del JSON es la siguiente: { "username": " ", "first_name": " ", "last_name": " ", "email": " ", "birthdate": " ", "home_phone": " ", "personal_phone": " ", "address": " "} • El token que se debe ingresar en Auth debe ser de tipo Bearer	
Criterios de aceptación: • Si el JSON está mal estructurado en el método POST y no ingresa todos los valores de los campos, responde con el error 405. • Si el valor de “username” o “email” son iguales al de otro usuario registrado en el sistema, responde con el error 405. • Si actualiza correctamente el perfil responde con el siguiente mensaje “Profile updated successfully”.	

Tabla III. Historia de usuario No. 3

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU03</td><td>Usuario: Administrador, Organizador y Estudiante.</td></tr>
<tr><td colspan="2">Nombre de Historia: Actualizar contraseña</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Media</td></tr>
<tr><td colspan="2">Iteración asignada: 1</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Administrador, Organizador y Estudiante
Quiero: Cambiar mi contraseña.
Para: Mejorar mi nivel de seguridad</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /update-password utiliza el método POST para que se pueda ingresar un JSON con la información de la contraseña
2. En Auth se debe ingresar el token para cambiar la contraseña.</td></tr>
<tr><td colspan="2">Observación:
• La estructura del JSON es la siguiente:
{ "password":"Secret123*", "password_confirmation":"Secret123*"}
• El token que se debe ingresar en Auth debe ser de tipo Bearer</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si el JSON está mal estructurado o utiliza un método diferente a POST responde con el error 405.
• La contraseña debe tener 10 dígitos, usar mayúsculas, minúsculas, números y símbolos, si no cumple responde con el error 405.
• Para cambiar la contraseña, debe ingresar el token del usuario en Auth.
• Si actualiza correctamente la contraseña responde con el siguiente mensaje: “Password updated successfully”</td></tr>
</table>

Tabla IV. Historia de usuario No. 4

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU04</td><td>Usuario: Administrador, Organizador y Estudiante.</td></tr>
<tr><td colspan="2">Nombre de Historia: Recuperación de contraseña</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Media</td></tr>
<tr><td colspan="2">Iteración asignada: 1</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Administrador, Organizador y Estudiante
Quiero: Resetear mi contraseña.
Para: Recuperar mi cuenta en caso de olvido de la contraseña.</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /forgot-password utiliza el método POST para que se pueda ingresar un JSON con la información del correo que se quiere recuperar
2. Se envía un correo con el token para el cambio de contraseña.
3. Para resetear la contraseña, se debe usar la ruta /reset-password que usa el método POST. Se debe ingresar un JSON con la nueva contraseña</td></tr>
<tr><td colspan="2">Observación:
• La estructura del JSON de la ruta /forgot-password es la siguiente: { "email":"prueba@gmail.org"}
• La estructura del JSON de la ruta /reset-password es la siguiente: {"token":" ","email":" ","password":" ", "password_confirmation":" "}</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si el JSON está mal estructurado o utiliza un método diferente a POST responde con el error 405.
• La nueva contraseña debe tener 10 dígitos, usar mayúsculas, minúsculas, números y símbolos, si no cumple responde con el error 405.
• Si se resetea correctamente la contraseña responde con el siguiente mensaje: “Your password has been reset”</td></tr>
</table>

Tabla V. Historia de usuario No. 5

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU05</td><td>Usuario: Administrador</td></tr>
<tr><td colspan="2">Nombre de Historia: Gestión de usuarios de rol organizador</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Alta</td></tr>
<tr><td colspan="2">Iteración asignada: 2</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Administrador
Quiero: Gestionar usuarios de rol organizador
Para: Crear, listar, actualizar, mostrar y eliminar usuarios.</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /organizer utiliza el método GET obtener la lista de usuario de rol organizador.
2. La ruta /organizer/create utiliza el método POST para crear usuarios, se debe ingresar el JSON con la información del nuevo usuario.
3. La ruta /organizer/ID/update utiliza el método POST para actualizar usuarios, se debe ingresar el JSON con la información de la actualización.
4. La ruta /organizer/ID utiliza el método GET para obtener la información de un usuario en especifico.
5. La ruta /organizer/ID/destroy utiliza el método GET para eliminar un usuario en específico.</td></tr>
<tr><td colspan="2">Observación:
• La estructura del JSON en las rutas /organizer/create y /organizer/ID/update es la siguiente:
{ "first_name": " ", "last_name": " ", "username": " ", "email": " ", "personal_phone": " ", "home_phone": " ", "address": " "}
• En las rutas /organizer/ID/update, /organizer/UD, /organizer/ID/destroy se debe ingresar el ID del usuario.
• Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si el JSON está mal estructurado o utiliza un método diferente que no corresponde a la ruta responde con el error 405.
• Si el valor de “username” o “email” son iguales al de otro usuario registrado en el sistema, responde con el error 405.</td></tr>
</table>

• Si ingresa correctamente se retorna un mensaje de “User stored successfully.”, se muestra la información del usuario y el token. • Si actualiza correctamente se retorna un mensaje de “User updated successfully” • Si elimina un organizador se retorna un mensaje de “User inactivated successfully.”

Tabla VI. Historia de usuario No. 6

HISTORIA DE USUARIO	
Identificador (ID): HU06	**Usuario:** Administrador
Nombre de Historia: Banear usuarios estudiantes	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 2	
Responsable: Bryan Tandazo	
Descripción: Como: Administrador Quiero: Banear usuarios de rol estudiante. Para: Listar, mostrar y eliminar usuarios.	
Alcance: 1. La ruta /student utiliza el método GET obtener la lista de usuarios de rol estudiante. 2. La ruta /student/ID utiliza el método GET para obtener la información de un usuario en específico. 3. La ruta /student/ID/destroy utiliza el método GET para eliminar un usuario en específico.	
Observación: • En las rutas /student/ID, /organizer/ID/destroy se debe ingresar el ID del usuario. • Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.	
Criterios de aceptación: • Si elimina un estudiante se retorna un mensaje de “User inactivated successfully.”	

Tabla VII. Historia de usuario No. 7

HISTORIA DE USUARIO	
Identificador (ID): HU07	**Usuario:** Administrador
Nombre de Historia: Banear publicaciones	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 3	
Responsable: Bryan Tandazo	
Descripción: Como: Administrador Quiero: Banear publicaciones. Para: Listar, mostrar y eliminar publicaciones.	
Alcance: 1. La ruta /publication utiliza el método GET obtener la lista de publicaciones. 2. La ruta /publication/ID utiliza el método GET para obtener la información de una publicación en especifico. 3. La ruta /publication/ID/destroy utiliza el método GET para eliminar una publicación en específico.	
Observación: • En las rutas /publication/ID, /publication/ID/destroy se debe ingresar el ID de la publicación • Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.	
Criterios de aceptación: • Si usa un método diferente a GET lanza el error 405. • Si elimina una publicación se retorna un mensaje de “Publication inactivated successfully.”	

Tabla VIII. Historia de usuario No. 8

HISTORIA DE USUARIO	
Identificador (ID): HU08	**Usuario:** Organizador
Nombre de Historia: Gestión de publicaciones	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 3	
Responsable: Bryan Tandazo	

Descripción:

Como: Organizador

Quiero: Gestionar publicaciones.

Para: Crear, listar, actualizar, mostrar y eliminar publicaciones.

Alcance:

1. La ruta /publication utiliza el método GET obtener la lista de publicaciones
2. La ruta /publication/create utiliza el método POST para crear publicaciones, se debe ingresar el JSON con la información de la publicación.
3. La ruta /publication/ID/update utiliza el método POST para actualizar publicaciones, se debe ingresar el JSON con la información de la actualización.
4. La ruta /publication/ID utiliza el método GET para obtener la información de una publicación en especifico
5. La ruta /publication/ID/destroy utiliza el método GET para eliminar una publicación

Observación:

- La estructura del JSON en las rutas /publication/create y /publication/ID/update es la siguiente:
  { "Titulo": " ", "Descripcion": " ", "Beneficios": " ", "Procedimiento": "" }
- En las rutas /publication/ID/update, /publication/UD, /publication/ID/destroy se debe ingresar el ID de la publicación
- Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.

Criterios de aceptación:

- Si el JSON está mal estructurado o utiliza un método diferente que no corresponde a la ruta responde con el error 405.
- Si el valor de ID de la publicación no le pertenece al usuario responde con el error “You don't own this report”.
- Si ingresa correctamente se retorna un mensaje de “Publication stored successfully.”
- Si actualiza correctamente se retorna un mensaje de “Publication updated successfully”
- Si elimina una publicación se retorna un mensaje de “Publication inactivated successfully.”

Tabla IX. Historia de usuario No. 9

HISTORIA DE USUARIO	
Identificador (ID): HU09	**Usuario:** Estudiante
Nombre de Historia: Registro de usuarios.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 1	
Responsable: Bryan Tandazo	
Descripción: Como: Estudiante Quiero: Crear una cuenta de usuario. Para: Ingresar en el sistema.	
Alcance: 1. La ruta /register utiliza el método POST para crear una cuenta de usuario, se debe ingresarl JSON con la información del estudiante.	
Observación: • La estructura del JSON es la siguiente: { "email": " ", "password": " ", "first_name": " ", "last_name": " ", "personal_phone": " ", "address": " ", "username": " " }	
Criterios de aceptación: • Si el JSON está mal estructurado o utiliza un método diferente que no corresponde a la ruta responde con el error 405. • Si el valor de “username” o “email” son iguales al de otro usuario registrado en el sistema, responde con el error 405. • Si ingresa correctamente se retorna un mensaje de “User register successfully.”, se muestra la información del usuario y el token.	

Tabla X. Historia de usuario No. 10

HISTORIA DE USUARIO	
Identificador (ID): HU10	**Usuario:** Estudiante
Nombre de Historia: Publicaciones Favoritas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 4	
Responsable: Bryan Tandazo	

Descripción:

Como: Estudiante

Quiero: Gestionar publicaciones señalas como favoritas.

Para: Añadir, listar, mostrar y eliminar publicaciones favoritas.

Alcance:

1. La ruta /favorite utiliza el método GET obtener la lista de publicaciones indicadas como favoritas.
2. La ruta /favorite/create utiliza el método POST para añadir publicaciones como favorita, se debe ingresar el JSON con la información de la publicación.
3. La ruta /favorite/ID utiliza el método GET para obtener la información de una publicación agregada como favorita en especifico
4. La ruta /favorite/ID/destroy utiliza el método GET para eliminar una publicación señalada como favorita.

Observación:

- La estructura del JSON en las rutas / favorite /create es la siguiente:
  { "publicacion_id": " "}
- En las rutas /favorite/ID, /favorite/ID/destroy se debe ingresar el ID de la publicación
- Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.

Criterios de aceptación:

- Si el JSON está mal estructurado o utiliza un método diferente que no corresponde a la ruta responde con el error 405.
- Si el valor de ID de la publicación no le pertenece al usuario responde con el error “You don't own this Favorite”.
- Si lista las publicaciones señaladas como favoritas correctamente se retorna el siguiente mensaje “Favorite Publication list generated successfully”
- Si muestra una publicación señalada como favorita correctamente se retorna “Favorite details”
- Si ingresa correctamente se retorna un mensaje de “Favorite stored successfully.”
- Si quiere añadir como una favorita una publicación que ya esta añadida se retorna el siguiente mensaje “La publicacion ya está registrada como favorita para este usuario”
- Si elimina una publicación se retorna un mensaje de “Favorite inactivated successfully.”

Tabla XI. Historia de usuario No. 11

HISTORIA DE USUARIO	
Identificador (ID): HU11	**Usuario:** Estudiante
Nombre de Historia: Registrar calificaciones y comentarios.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 4	
Responsable: Bryan Tandazo	
Descripción: Como: Estudiante Quiero: Calificar y comentar las publicaciones. Para: Expresar mi apoyo y que sobresalgan las mejores.	
Alcance: 1. La ruta /favorite/ID/update utiliza el método POST para calificar y comentar las publicaciones indicadas como favoritas, se debe ingresar el JSON con la información de la publicación.	
Observación: • La estructura del JSON es la siguiente: { "calificacion": , "feedback": " "} • En las rutas /favorite/ID/update se debe ingresar el ID del favorito • La calificación debe ser entre 1 y 5. • Para hacer uso de las rutas se debe ingresar el token de tipo Bearer en Auth.	
Criterios de aceptación: • Si el JSON está mal estructurado o utiliza un método diferente que no corresponde a la ruta responde con el error 405. • Si el valor de ID del favorito no le pertenece al usuario responde con el error “You don't own this Favorite”. • Si la calificación es válida se retorna el siguiente mensaje “Favorito updated successfully”. • Si la calificación no se encuentre en 1 y 5 retorna lo siguiente “La calificacion ingresada debe estar entre 1 y 5”.	

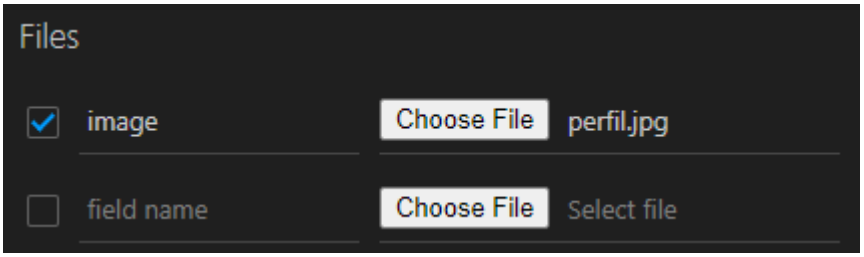
Tabla XII. Historia de usuario No. 12

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU12</td><td>Usuario: Estudiante</td></tr>
<tr><td colspan="2">Nombre de Historia: Ranking de las publicaciones</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Baja</td></tr>
<tr><td colspan="2">Iteración asignada: 4</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Estudiante
Quiero: Visualizar el Ranking de las mejores publicaciones.
Para: Identificar y acceder al contenido más relevante y útil.</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /qualification utiliza el método GET para obtener las tres publicaciones mejor calificadas.
2. La ruta /qualification/ID utiliza el método GET para obtener la calificación de una publicación en específico.</td></tr>
<tr><td colspan="2">Observación:
• En las rutas /qualification/ID, se debe ingresar el ID de la publicación.
• Para hacer uso de la ruta se debe ingresar el token de tipo Bearer en Auth.</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si utiliza un método diferente a GET responde con el error 405
• Si accede correctamente a la ruta /qualification se muestra el mensaje “Qualification list generated successfully” y se muestra las tres primeras publicaciones mejor promediadas.
• Si accede correctamente a la ruta /qualification/ID se muestra el mensaje “Qualification details” con el promedio de la publicación
• Si quiere acceder a la calificación de una publicación eliminada retorna el mensaje de “This publication is inactive or deleted.”</td></tr>
</table>

Tabla XIII. Historia de usuario No. 13

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU13</td><td>Usuario: Organizador</td></tr>
<tr><td colspan="2">Nombre de Historia: Visualizar calificaciones y comentarios</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Media</td></tr>
<tr><td colspan="2">Iteración asignada: 4</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Organizador
Quiero: Visualizar las calificaciones y comentarios de mis publicaciones.
Para: Evaluar el impacto y recepción de mis publicaciones entre los estudiantes y así mejorar la calidad y relevancia de mis publicaciones.</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /qualification utiliza el método GET para obtener las calificaciones y comentarios solo de sus publicaciones.
2. La ruta /qualification/ID utiliza el método GET para obtener la calificación de una publicación en específico.</td></tr>
<tr><td colspan="2">Observación:
• En las rutas /qualification/ID, se debe ingresar el ID de la publicación.
• Para hacer uso de la ruta se debe ingresar el token de tipo Bearer en Auth.</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si utiliza un método diferente a GET responde con el error 405
• Si accede correctamente a la ruta /qualification se muestra el mensaje “Qualification organizador list generated successfully” y se muestra las calificaciones y comentarios de sus publicaciones.
• Si accede correctamente a la ruta /qualification/ID se muestra el mensaje “Qualification organizador detail” con la calificación y comentario.
• Si quiere acceder a la calificación de una publicación que no le pertenece retorna el mensaje de “You dont own this Qualification.”
• Si quiere acceder a la calificación de una publicación eliminada retorna el mensaje de “This publication is inactive or deleted.”</td></tr>
</table>

Tabla XIV. Historia de usuario No. 14

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Identificador (ID): HU14</td><td>Usuario: Administrador, Organizador y Estudiante.</td></tr>
<tr><td colspan="2">Nombre de Historia: Actualizar avatar del perfil de usuario</td></tr>
<tr><td>Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Baja</td></tr>
<tr><td colspan="2">Iteración asignada: 4</td></tr>
<tr><td colspan="2">Responsable: Bryan Tandazo</td></tr>
<tr><td colspan="2">Descripción:
Como: Administrador, Organizador y Estudiante.
Quiero: Cambiar mi avatar de usuario.
Para: Actualizar mi foto de perfil.</td></tr>
<tr><td colspan="2">Alcance:
1. La ruta /profile/avatar utiliza el método POST, por medio de un formulario se elige el archivo.</td></tr>
<tr><td colspan="2">Observación:
• El formulario solo contiene un campo “image” y se elige el archivo.
 

• El token que se debe ingresar en Auth debe ser de tipo Bearer</td></tr>
<tr><td colspan="2">Criterios de aceptación:
• Si el formulario está mal estructurado en el método o se utiliza un método diferente a POST responde con el error 405.
• La extensión del archivo debe ser jpg,png,jpeg.
• Si actualiza correctamente el avatar responde con el siguiente mensaje “Avatar updated successfully”.</td></tr>
</table>

Tabla XV. Historia de usuario No. 15

HISTORIA DE USUARIO	
Identificador (ID): HU15	**Usuario:** Estudiante
Nombre de Historia: Visualizar publicaciones	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 3	
Responsable: Bryan Tandazo	
Descripción: Como: Estudiante. Quiero: Visualizar las publicaciones creadas por los organizadores. Para: Mantenerme informado sobre la salud mental, física y conocer hábitos alimenticios	
Alcance: 1. La ruta /publication utiliza el método GET, para obtener todas las publicaciones. 2. La ruta /publication/ID usa el método GET para mostrar una publicación	
Observación: • . En La ruta /publication/ID, ID es el numero de la publicación que quiere visualizar. • El token que se debe ingresar en Auth debe ser de tipo Bearer	
Criterios de aceptación: • Si utiliza un método diferente a GET responde con el error 405. • Si lista correctamente las publicaciones responde con el siguiente mensaje “Publication list generated successfully”. Y se muestran todas las publicaciones. • Si consulta correctamente una publicación en especifico se muestra el siguiente mensaje “Publication details” y se muestra los detalles de la publicación. • Si quiere acceder a las diferentes rutas para crear, actualizar o eliminar una publicación se responde con un mensaje “This user is not autorized”	

Product Backlog

En la **Tabla XVI.** muestra la lista de requerimientos de la aplicación que se realizara en la cada Sprint junto con el estado y la prioridad en la base a la información de las historias de usuario.

Tabla XVI. Product Backlog

ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU01	Servicio inicio de sesión	1	Completa	Alta
HU02	Actualizar perfil de usuario	2	Completa	Media
HU03	Actualizar contraseña	1	Completa	Media
HU04	Recuperación de contraseña	1	Completa	Media
HU05	Gestión de usuarios de rol organizador	2	Completa	Alta
HU06	Banear usuarios estudiantes	2	Completa	Alta
HU07	Banear publicaciones	3	Completa	Media
HU08	Gestión de publicaciones	3	Completa	Alta
HU09	Registro de usuarios	1	Completa	Alta
HU10	Publicaciones Favoritas	4	Completa	Media
HU11	Registrar calificaciones y comentarios.	4	Completa	Media

HU12	Ranking de las publicaciones	4	Completa	Baja
HU13	Visualizar calificaciones y comentarios	4	Completa	Media
HU14	Actualizar avatar del perfil de usuario	4	Completa	Baja
HU15	Visualizar publicaciones	3	Completa	Alta

Sprint Backlog

En la **Tabla XVII**, se muestra la lista de tareas que tendrá cada Sprint con la información de cada historia de usuario, su ID, las actividades y el tiempo estimado.

Tabla XVII. Sprint Backlog

ID-SB	Nombre	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB-000	Configuración del entorno de desarrollo	N/A	N/A	• Configurar el entorno de desarrollo en Visual Studio code para la creación de proyecto. • Configuración de las variables de entorno de los servicios. • Diseño y creación de la base de datos.	20H
SB-001	Autenticación	HU01	Servicio para inicio de sesión	• Desarrollar la funcionalidad que permita ingresar en el sistema a los tres roles de usuarios.	40H

				• Desarrollar la funcionalidad que permita salir del sistema.	
		HU04	Recuperación de contraseña	• Crear la ruta donde el usuario pueda ingresar un JSON con el correo de la cuenta a recuperar. • Crear la funcionalidad donde se envié un token por correo al usuario para recuperar su cuenta. • Crear la ruta donde el usuario puede ingresar un JSON para resetear su contraseña utilizando el token enviado.	
		HU03	Actualizar contraseña	• Crear la ruta donde el usuario pueda ingresar un JSON donde pueda actualizar su contraseña.	
		HU09	Registro de usuarios	• Crear la ruta donde el usuario estudiante pueda registrarse en el sistema ingresando un JSON.	
SB-002	Funciones principales del Administrador	HU02	Actualizar perfil de usuario	• Crear la funcionalidad que permita visualizar el perfil de usuario. • Crear la ruta donde el usuario pueda ingresar un JSON con su información para actualizar.	50H
		HU05	Gestión de usuarios de	• Crear la funcionalidad que permita listar usuarios con rol organizador.	

			rol organizador	• Crear la ruta donde el administrador pueda ingresar un JSON con la información de usuario. • Crear la funcionalidad para que una vez creado el usuario las credenciales sean enviadas por correo. • Crear la ruta para que el administrador pueda ingresar un JSON para actualizar la información de algún organizador • Crear la funcionalidad para que el administrador pueda visualizar a un usuario organizador en específico. • Crear la funcionalidad para que el administrador pueda eliminar a un usuario organizador.	
		HU06	Banear usuarios estudiantes	• Crear la funcionalidad para listar a los usuarios de rol estudiante. • Crear la funcionalidad para mostrar un usuario de rol estudiante en específico. • Crear la funcionalidad para que pueda eliminar a un usuario con rol estudiante.	

SB3	Funciones principales del Organizador	HU08	Gestión de publicaciones	• Crear la funcionalidad que permita al organizador listar sus publicaciones. • Crear la ruta donde el organizador pueda ingresar un JSON con la información de la publicación. • Crear la ruta para que el organizador pueda ingresar un JSON para actualizar la información de sus publicaciones • Crear la funcionalidad para que el organizador pueda visualizar una publicación específica. • Crear la funcionalidad para que el organizador pueda eliminar alguna de sus publicaciones.	40H
		HU07	Banear publicaciones	• Crear la funcionalidad para que el administrador pueda listar las publicaciones. • Crear la funcionalidad para que el administrador pueda consultar una publicación especifica. • Crear la funcionalidad para que el administrador pueda eliminar una publicación.	

		HU15	Visualizar publicaciones	• Crear la funcionalidad para que el estudiante pueda listar las publicaciones. • Crear la funcionalidad para que el estudiante pueda consultar una publicación especifica.	
SB4	Funciones principales del Estudiante	HU10	Publicaciones Favoritas	• Crear la funcionalidad para que el estudiante pueda listar sus publicaciones señaladas como favoritas. • Crear la ruta donde el estudiante pueda ingresar un JSON con la información para añadir una publicación como favorita. • Crear la funcionalidad para que el estudiante pueda visualizar una publicación especifica señalada como favorita. • Crear la funcionalidad para que el estudiante pueda eliminar alguna de sus publicaciones señaladas como favorita.	70H
		HU11	Registrar calificaciones y comentarios.	• Crear la ruta donde el estudiante pueda ingresar un JSON con la información para calificar o comentar una publicación indicada como favorita.	

		HU12	Ranking de las publicaciones	• Crear una funcionalidad para que el estudiante pueda visualizar las tres mejores publicaciones promediadas.	
		HU13	Visualizar calificaciones y comentarios	• Crear una funcionalidad que permita al organizador visualizar las calificaciones y comentarios de sus publicaciones.	
		HU14	Actualizar avatar del perfil de usuario	• Crear una ruta donde los usuarios puedan cambiar sus avatars de perfil por medio de un formulario	
SB5	Pruebas y despliegue de la APIREST	N/A	N/A	• Pruebas Funcionales y Seguridad • Pruebas de Rendimiento • Despliegue a producción.	20H
TOTAL					240H

Nota: El documento del manual de técnico se lo puede encontrar en el siguiente LINK: ManualTenico BryanTandazo.pdf

ANEXO III

En el siguiente enlance se encuentra el manual de usuario del componente Backend, en donde se muestran las funcionalidades del sistema para los diferentes roles que son administrador, organizador y estudiante del proyecto Vitalzure. De igual manera se encuentra el enlace al video explicativo del sistema.

LINK: ManualUsuario BryanTandazo.pdf

URL: https://www.youtube.com/watch?v=XyVy7EkKRWE

ANEXO IV

A continuación, se muestra el enlace del sistema desplegado y las credenciales de acceso para los diferentes tipos de usuario. Además, se adjunta en enlace al repositorio de GitHub donde se aloja el proyecto y el manual de instalación.

Credenciales de acceso

El despliegue a producción del proyecto Vitalzure, se encuentra en el siguiente enlace: https://proyecto-vitalzure.vercel.app/

- Perfil administrador
  Correo: bryan.tandazo@epn.edu.ec
  Contraseña: secret
- Perfil organizador
  Correo: gabriel13@example.com
  Contraseña: secret
- Perfil estudiante
  Correo: sheila71@gmail.com
  Contraseña: secret

Repositorio del código fuente

El repositorio donde se encuentra el código fuente del proyecto se lo puede encontrar en el siguiente enlace, en donde hay cinco ramas con el desarrollo del código para cada Sprint del 0 al 5.

- Sistema Web: https://github.com/BryanTnz/proyecto_vitalzure

La documentación de la API se lo puede encontrar en el siguiente enlace:

- https://documenter.getpostman.com/view/27735939/2sA2xnw9SF

El manual de instalación se lo puede encontrar en el siguiente enlace:

- ManualInstalación BryanTandazo.pdf