ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

SISTEMA WEB PARA REGISTRO Y PAGO EN LÍNEA DE CONFERENCIAS

DESARROLLO DEL *BACKEND*

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

JONATHAN ALEJANDRO ARMAS COLLAHUAZO

jonathan.armas@epn.edu.ec

DIRECTOR: IVONNE FERNANDA MALDONADO SOLIZ

ivonne.maldonado@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, Jonathan Alejandro Armas Collahuazo 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.

JONATHAN ALEJANDRO ARMAS COLLAHUAZO

jonathan.armas@epn.edu.ec

l_jandro@hotmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Jonathan Alejandro Armas Collahuazo, bajo mi supervisión.

IVONNE FERNANDA MALDONADO SOLIZ

DIRECTOR

ivonne.maldonado@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.

Jonathan Alejandro Armas Collahuazo

DEDICATORIA

Este proyecto de integración curricular se lo dedico a mis padres, que me han encaminado a lo largo de mi vida para ser una persona de bien con valores y principios. Se lo dedico a mi madre que ha sido una compañera fundamental en cada etapa, dedicando su vida a criarme y a enseñarme que las personas pueden llegar tan lejos como se propongan sin opacar a nadie y siendo integras a cada paso.

A mi padre, que es mi ejemplo de lucha ferviente ante las circunstancias de la vida, que me enseñó a no doblegarme ante los tiempos difíciles y a ser una persona que trabaje por sus sueños. Le dedico este logro que hubiera disfrutado como propio con total merecimiento. Que sepa que todos los días lo llevo en mis pensamientos y en mi corazón.

Jonathan Alejandro Armas Collahuazo

AGRADECIMIENTO

Agradezco a Dios que me ha encaminado en cada paso que he dado para cumplir con mi objetivo en la vida. A mis padres y a toda mi familia, que han sido mi fortaleza para seguir y persistir durante las circunstancias más complicadas, que han inculcado en mi persona valores que ligados de la mano con la parte técnica que me ha brindado la Escuela Politécnica Nacional han dado como resultado un buen profesional.

A mis amigos, aunque quisiera nombrarlos uno por uno me faltarían hojas enteras para agradecerles por todo, pero saben que siempre van a tener mi amistad, mi lealtad y agradecimiento. Pero en este punto quiero resaltar tres nombres que son a Byron, Majo y Andrés que han estado presentes en tiempos buenos y malos incondicionalmente, haciendo que nuestra amistad se vuelva un vínculo de hermandad en la vida.

A mi Andreita López, que fue la que inicio todo este proyecto de estudio con su gran apoyo. Que jamás en la vida voy a olvidar lo que hizo por mí, para que en este instante pueda disfrutar de este logro. Gracias por no dejar que me pierda del camino.

A cada uno de los ingenieros que me han impulsado transmitiéndome sus conocimientos y anécdotas. En especial a la Ing. Ivonne Maldonado que ha sido la persona que me ha apoyado y guiado durante todo este proceso universitario de principio a fin, que ha estado ahí para darme un consejo o resolver mis inquietudes. Ha sido un pilar fundamental para poder sacar este proyecto adelante. De todo corazón gracias.

Jonathan Alejandro Armas Collahuazo

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco Teórico 3
2 METODOLOGÍA 6
2.1 Metodología de Desarrollo 6
NOMBRES 7
Artefactos 8
2.2 Diseño de la Arquitectura 9
Arquitectura de Datos 10
Patrón arquitectónico 10
2.3 Herramientas de desarrollo 12
Librerías 13
3 RESULTADOS 14
3.1 *Sprint* 0. Configuración del ambiente de desarrollo 14
Recopilación y Definición de Requerimientos 14
Elaboración del Modelo de Datos 16
Estructura del Proyecto 16
3.2 *Sprint* 1. Módulo de Autenticación 16
Generar endpoints para el registro de usuarios 17
Generar endpoints para el inicio y cierre de sesión 19
Generar endpoints para verificación de email 21
Generar endpoints para la recuperación de cuenta 23
3.3 *Sprint* 2. Módulo de Ponentes 24
Generar endpoints para la información de los ponentes 24

3.4 *Sprint* 3. Módulo de Eventos 27
Generar endpoints para el registro de eventos 27
3.5 *Sprint* 4. Módulo de Pagos 29
Generar endpoints para los pagos en línea y consulta de información de asistentes y pagos realizado 29
3.6 *Sprint* 5. *Pruebas* y Despliegue de *Endpoints* 31
Pruebas unitarias 31
Pruebas de carga 32
Pruebas de rendimiento 33
Despliegue de *endpoints* a 000WebHost 34
4 Conclusiones 35
5 Recomendaciones 36
6 REFERENCIAS BIBLIOGRÁFICAS 37
7 ANEXOS 39

RESUMEN

El presente proyecto, planea dar una solución aplicable al entorno de publicación de eventos con pagos a través de canales electrónicos, que permitan aumentar la seguridad en las transacciones de los usuarios al obviar la necesidad de pagos en efectivo. Los pagos en línea ofrecen una forma rápida y conveniente de realizar transacciones desde cualquier lugar con acceso a internet, siendo un servicio 24/7.

Por otro lado, permite tener un modelo de negocio hasta cierto punto automatizado, permitiendo que los usuarios puedan obtener el servicio en este caso de conferencistas expertos en temas de tecnología con una solución al alcance de cualquier dispositivo.

La metodología *Scrum* ha posibilitado que se pueda obtener un enfoque más sencillo sobre las necesidades del cliente. Desarrollando el proyecto en pequeños módulos que han permitido un mejor control a nivel del equipo de desarrollo que minimiza los fallos y aumenta la productividad.

Es así como se ha desarrollado un *backend* con la creación de una base de datos que permita organizar y estructurar de una manera ágil la información de la plataforma. Y a su vez permita la comunicación con el *frontend* de manera efectiva tomando en cuenta las necesidades establecidas a través del *Product Owner* al momento de la planeación del proyecto.

Este documento está organizado de la siguiente forma: primero, se presenta la descripción del componente, los objetivos (general y específicos) del proyecto y el marco teórico. Segundo, se describe la metodología *Scrum*, la arquitectura y herramientas utilizadas en la implementación del componente *backend*. Tercero, se muestran los resultados por cada *Sprint*. Y finalmente se emiten conclusiones y recomendaciones obtenidas en el cumplimiento del proyecto.

PALABRAS CLAVE: *backend*, *Scrum*, *endpoint,* gestión, sistema web.

ABSTRACT

This project plans to provide a solution applicable to the event publishing environment with payments through electronic channels, which will increase the security of user transactions by obviating the need for cash payments. Online payments offer a fast and convenient way to make transactions from any place with internet access, being a full-time service.

In addition, it will allow a business model to some extent automated, allowing users to obtain the service in this case of expert speakers on technology issues with a solution within reach of any mobile device or computer.

The Scrum methodology has made it possible to obtain a simpler approach to the client's needs. Developing the project in small modules has allowed a better control at the development team level that minimizes failures and increases productivity.

This is how a backend has been developed with the creation of a database that allows to organize and structure in an agile way the information of the platform. And in allows communication with the frontend in an effective way taking into account the needs established through the Product Owner at the time of project planning.

This document is organized as follows: first, the description of the component, the objectives (general and specific) of the project and the theoretical framework are presented. Second, the Scrum methodology, architecture and tools used in the implementation of the backend component are described. Third, the results for each Sprint are shown. And finally, conclusions and recommendations obtained in the fulfillment of the project are issued.

KEYWORDS: *backend*, *Scrum*, *endpoint,* management, web application.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La importancia que ha ganado la tecnología en cada aspecto de la vida ha hecho que se disponga de un aparato con conexión a internet al alcance de la mano. En el Ecuador existen 10.17 millones de usuarios de internet representando el 78.8% de la población reduciendo la brecha digital que ha existido en el país durante años [1].

La tecnología ha revolucionado la forma de vida y ha tenido un gran impacto en todos los aspectos de la sociedad contemporánea. Desde la comunicación instantánea a larga distancia hasta la automatización de tareas, ampliando capacidades y transformando la manera en que se interactúa con el mundo. Ha mejorado la eficiencia en sectores como la medicina, la industria, la educación y el transporte, facilitando y acelerando la precisión de tareas complejas. Además, la tecnología ha fomentado la conectividad global, permitiendo a las personas compartir información, ideas y culturas de manera instantánea. Sin embargo, también es importante reconocer los desafíos que surgen con los avances tecnológicos, como la necesidad de la protección de datos.

Las plataformas que permiten la gestión de usuarios son herramientas esenciales para empresas y organizaciones que buscan administrar eficientemente las identidades y accesos de sus usuarios. Estas plataformas permiten centralizar y automatizar tareas, así como la solicitud de permisos y roles. Por otro lado, facilitan la autenticación y el inicio de sesión seguro, a través de métodos como contraseñas y autenticación en dos pasos. Al utilizar una plataforma de gestión de usuarios, las empresas pueden mejorar la seguridad de sus sistemas y datos, reducir los recursos de administración hacia los usuarios individualmente y garantizar su conformidad.

El presente componente tiene como objetivo desarrollar el *backend* de un sistema web que permita almacenar y manejar mediante MySQL los datos de conferencias, usuarios y administradores. Con la finalidad de alimentar el *frontend* para poder publicar en el sistema la información requerida, además de salvaguardar la confidencialidad de los datos siguiendo la lógica del negocio.

1.1 Objetivo general

Desarrollar un *backend* de un sistema web para registro y pago en línea de conferencias.

1.2 Objetivos específicos

1. Determinar los requerimientos del *backend* para el sistema web para registro y pago en línea de conferencias.
2. Diseñar el modelo de base de datos para el sistema web de acuerdo con los requerimientos levantados.
3. Desarrollar los *endpoints* de acuerdo con los requerimientos.
4. Realizar las pruebas al *backend* para poder comprobar su funcionamiento.
5. Desplegar el *backend*.

1.3 Alcance

El proyecto incluye conceptualización, desarrollo y pruebas del *backend* de un sistema web destinado a publicar conferencias de diversos temas, donde los usuarios pueden inscribirse y realizar sus pagos en línea.

El *backend* cuenta con *endpoints* que permiten un *login* al sistema, manejando dos perfiles: usuario y administrador. De manera general, el perfil usuario cuenta con varios *endpoints* que le permiten: consultar las conferencias ofertadas en el sistema web, realizar inscripción y pagos en los temas de interés. Mientras que el perfil administrador cuenta con varios *endpoints* que le permiten gestionar oferta de conferencias.

El desarrollo garantiza la seguridad de los datos por parte del *backend*, para tener un mejor control de los datos que son presentados por parte del *frontend*.

Se ha utilizado como guía de desarrollo a *Scrum*, una metodología que permite lograr los objetivos propuestos en los tiempos determinados; además se ha hecho uso de un modelo arquitectónico y una serie de pruebas para verificar la correcta funcionalidad del *backend*. Y finalmente, se ha realizado el despliegue de producción.

A continuación, se listan las diferentes acciones de cada rol.

Rol Administrador:

- Iniciar o cerrar sesión.
- Crear cuenta.
- Crear, editar y eliminar publicaciones.
- Crear, editar y eliminar información de ponentes.
- Controlar la capacidad de asistentes.

Rol Usuario:

- Iniciar o cerrar sesión.
- Crear cuenta.
- Navegar en la plataforma.
- Realizar pagos en línea.

1.4 Marco Teórico

El desarrollo de software es el proceso de diseñar y elaborar programas informáticos que cumplan con requisitos específicos. Este proceso involucra la planificación, el diseño, la codificación, las pruebas y despliegue del software, así como el mantenimiento y la mejora continua [2]. El desarrollo de software requiere una combinación de habilidades técnicas y creativas, donde los desarrolladores trabajan en estrecha colaboración con los analistas de sistemas, diseñadores de interfaz y otros profesionales para cumplir los requisitos del cliente en soluciones de software efectivas. Durante todo el proceso, se enfoca en la calidad, la eficiencia y la satisfacción del usuario, con el objetivo de crear software confiable, funcional y escalable.

Las metodologías ágiles son enfoques flexibles y colaborativos utilizados en el desarrollo de proyectos para adaptarse a los cambios y entregar valor de manera rápida y continua. En contraste con las metodologías tradicionales, que siguen un enfoque secuencial y planificado, las metodologías ágiles valoran la interacción y la retroalimentación constante, fomentando la autoorganización y la toma de decisiones colaborativas. Scrum, Kanban y XP son ejemplos de metodologías ágiles muy utilizadas [3]. Estas metodologías promueven la entrega de incrementos de software funcional en períodos cortos, conocidas como iteraciones o *sprints*, permitiendo que los equipos respondan rápidamente a los cambios y priorizando las necesidades del cliente.

Un sistema web es un conjunto de componentes interconectados y funcionales que trabajan en conjunto para brindar funcionalidad a través de internet. Estos sistemas utilizan tecnologías web, como servidores, navegadores web y protocolos de comunicación, para permitir la interacción de los usuarios con la aplicación a través de una interfaz basada en la web.

Un sistema web consta de diferentes capas, que incluyen el *frontend* (la interfaz de usuario con la que interactúan los usuarios), el *backend* (que procesa las solicitudes del usuario) y la capa de datos (donde se almacena la información utilizada por la aplicación) [4].

Los sistemas web pueden variar en complejidad y funcionalidad. Pueden ser desde simples sitios web estáticos que brindan información básica hasta aplicaciones web dinámicas y sofisticadas que permiten la interacción en tiempo real, el procesamiento de transacciones y la gestión de bases de datos.

El *backend*, también conocido como el lado del servidor, es una parte fundamental de cualquier aplicación web o plataforma digital. Se encarga de gestionar la lógica y la funcionalidad detrás de escena. El *backend* entonces maneja las solicitudes del usuario, procesa datos, interactúa con bases de datos, autentica y autoriza usuarios, implementando la lógica empresarial y proporcionando respuestas adecuadas al *frontend* [5]. Se basa en tecnologías como: lenguajes de programación como Python, Java o Node.js, *frameworks* como Django o Express.js, y bases de datos como MySQL o MongoDB. Un *backend* bien diseñado y eficiente es esencial para garantizar un rendimiento óptimo, seguridad robusta y una experiencia fluida para los usuarios de una aplicación o sitio web.

Una API (*Application Programming Interface*) es un conjunto de reglas y protocolos que permiten la comunicación entre diferentes sistemas de software. Sirve como un intermediario que define cómo los componentes de software deben interactuar y compartir información entre sí [6]. Una API proporciona una interfaz consistente y estructurada que permite a los desarrolladores acceder a funciones y datos específicos de un sistema o servicio, sin necesidad de conocer los detalles de su implementación. Las API se utilizan ampliamente en el desarrollo de software, permitiendo la integración de servicios complejos y poderosos al aprovechar las funcionalidades de otros sistemas.

Un API REST (*Application Programming Interface- Representational State Transfer*) es una interfaz de programación basada en los principios del estilo arquitectónico REST. Se utiliza para facilitar intercambio de datos entre diferentes aplicaciones y sistemas a través de internet [7]. Una API REST utiliza los métodos y protocolos del protocolo HTTP para permitir

que los clientes realicen solicitudes y obtengan respuestas en un formato estándar, como JSON o XML. Proporciona una manera eficiente y escalable de acceder y manipular recursos en la web, permitiendo la interoperabilidad y la integración de sistemas de manera sencilla.

El Protocolo de Transferencia de Hipertexto (HTTP) es el protocolo utilizado en la World Wide Web (WWW) para transferir información. HTTP sigue el modelo cliente-servidor, donde el cliente realiza solicitudes y el servidor responde con las respuestas correspondientes. Este protocolo se basa en una estructura de solicitud y respuesta, donde las solicitudes del cliente contienen métodos como GET (obtener un recurso), POST (crea un recurso), PUT (actualizar un recurso) o DELETE (eliminar un recurso), y las respuestas del servidor incluyen códigos de estado para indicar el resultado de la operación [8]. HTTP utiliza un enfoque sin estado, lo que significa que cada solicitud se procesa de forma independiente y no se mantiene información sobre conexiones anteriores. Sin embargo, se pueden utilizar cookies para mantener cierta persistencia en las interacciones entre el cliente y el servidor. HTTP es un protocolo ampliamente utilizado y fundamental para el funcionamiento de la web, permitiendo la transferencia de recursos como páginas HTML, imágenes, videos y otros tipos de contenido.

Las bases de datos son sistemas organizados y estructurados para almacenar, administrar y recuperar grandes cantidades de información de manera eficiente. Se utilizan para almacenar datos relacionados de manera lógica y coherente, lo que facilita su acceso y manipulación. Las bases de datos se componen de tablas o colecciones de datos interrelacionados que representan entidades y sus atributos. Estas pueden ser de dos tipos, bases de datos relacionales y no relacionales. Además, las bases de datos ofrecen mecanismos para realizar consultas y modificaciones de datos, garantizando la integridad y la seguridad de la información. Son ampliamente utilizadas en diversas aplicaciones y sectores, como comercio electrónico, sistemas de gestión empresarial, sitios web y muchas otras áreas donde se requiere una gestión eficiente y confiable de datos

2 METODOLOGÍA

La metodología se refiere al enfoque, conjunto de prácticas y procesos utilizados para gestionar y organizar el desarrollo de software de manera eficiente y efectiva. Estas metodologías establecen pautas y estructuras que ayudan a los equipos de desarrollo a planificar, diseñar, implementar y mantener el software.

El estudio de casos es una metodología de investigación que consiste en un análisis detallado y exhaustivo de una situación o fenómeno particular, con el objetivo de comprenderlo en profundidad, identificar patrones, factores causales y posibles soluciones.

En un estudio de casos, se recopilan y examinan múltiples fuentes de datos, como entrevistas, observaciones, registros históricos y documentos, con el fin de obtener una visión completa y contextualizada del tema de estudio. Este enfoque permite una comprensión más completa y a menudo se utiliza para generar conocimientos prácticos y teóricos que pueden aplicarse en situaciones similares en el futuro [9].

2.1 Metodología de Desarrollo

El producto de software es el resultado tangible de un proceso de desarrollo que combina diseño, programación y pruebas para crear una solución tecnológica. Puede ser una aplicación móvil, un sistema web o cualquier otro tipo de software destinado a satisfacer las necesidades de los usuarios. Un producto de software exitoso debe ser funcional, confiable, fácil de usar y satisfacer los requisitos específicos del cliente. Para lograrlo, se emplean metodologías de desarrollo, como Scrum, que permiten una iteración continua y la adaptación a medida que se obtiene mediante el *feedback*. Además, el producto de software debe estar respaldado por un proceso sólido de control de calidad para garantizar su rendimiento, seguridad y estabilidad.

Scrum es un marco de trabajo ágil utilizado en el desarrollo de software y en la gestión de proyectos. Se basa en la colaboración, la transparencia y la adaptabilidad para permitir a los equipos entregar valor de manera incremental y continua. En Scrum, el proyecto se divide en ciclos de trabajo llamados "*sprints*". Durante cada *sprint*, el equipo se enfoca en una serie de objetivos acordados y realiza un seguimiento regular de su progreso. Al final de cada *sprint*, se realiza una revisión para demostrar el trabajo completado y recibir retroalimentación. Scrum se basa en roles claros, como el *Scrum Master*, que facilita el proceso, el *Product Owner*, responsable de la visión y priorización del producto, y el *Development Team*, que realiza el trabajo real. Este enfoque iterativo y colaborativo de

Scrum permite una mayor flexibilidad, adaptabilidad y satisfacción del cliente a lo largo del proyecto [10].

Roles

Los roles en la metodología Scrum determina la responsabilidad de cada individuo dentro del marco dentro del equipo de trabajo y están diseñados para promover la colaboración y la transparencia y la entrega a tiempo de los proyectos de manera eficiente [11].

Product Owner

Es el responsable de representar los intereses del cliente o usuario final. Su tarea consiste en gestionar y priorizar el *producto backlog*, definir los requisitos y asegurar que se entreguen las características de mayor valor para el negocio [12]. En la **Tabla 2.1** se puede observar la persona asignada a este rol, y es quien toma dichas responsabilidades.

Scrum Master

Es el facilitador del proceso, responsable de asegurar que el equipo siga las prácticas y principios de Scrum. Su función principal es eliminar obstáculos, promover la comunicación efectiva y garantizar que el equipo se mantenga enfocado en los objetivos del proyecto [13]. En la **Tabla 2.1** se puede observar la persona asignada a este rol, y es quien toma dichas responsabilidades.

Development Team

Es el grupo de profesionales multidisciplinarios que se encarga de realizar el trabajo real. Son responsables de convertir los elementos del *product backlog* en incrementos de producto, colaborando estrechamente entre sí y adaptándose a medida que avanzan en los *sprints* [14]. En la **Tabla 2.1** se puede observar la persona asignada a este rol, y es quien toma dichas responsabilidades.

Tabla 2.1 Asignación de Roles

ROLES	NOMBRES
Product Owner	Ing. Ivonne Maldonado, Msc.
Scrum Master	Ing. Ivonne Maldonado, Msc.
Development Team	Alejandro Armas

Artefactos

Scrum denomina a los artefactos como todo aquello que permite transparentar el proceso del desarrollo de software, es decir es la base para cumplir con los objetivos planteados.

Recopilación de Requerimientos

Es un proceso crucial en el desarrollo de software que implica identificar, analizar y documentar las necesidades y expectativas de los usuarios finales. Esta etapa se enfoca en comprender qué funcionalidades y características debe tener el producto final para satisfacer las necesidades del cliente[15]. El

ANEXO II apartado Recopilación de requerimientos del presente documento muestra la recopilación de requerimientos para analizar la problemática abordada en este proyecto.

Historias de Usuario

Es una técnica utilizada en el desarrollo de software que permite capturar y describir los requisitos funcionales de un sistema desde la perspectiva del usuario final. Se trata de una breve narrativa que describe una funcionalidad específica del software, generalmente escrita en un lenguaje sencillo y comprensible para todos los miembros del equipo de desarrollo. Consta de tres elementos principales: el título, que resume la funcionalidad deseada; la descripción, que detalla el contexto y los requisitos específicos; y los criterios de aceptación, que fundamentan las condiciones que deben cumplirse para considerar la historia completa [16]. La Tabla 2.2 muestra el ejemplo de una historia de usuario, dentro del

ANEXO II se detalla las restantes historias de usuario del proyecto.

Tabla 2.2 Ejemplo de Historia de Usuario - HU01

Historia de Usuario	
Identificador: HU01	**Usuario:** Usuario/Administrador
Nombre Historia: Generar *endpoints* para el registro de usuarios	
Prioridad en Negocio (Alta/Medio/Baja): Alta	**Riesgo en Desarrollo (Alta/Medio/Baja):** Baja
Iteración asignada: 1	
Responsable: Alejandro Armas	
Descripción: Se necesita crear *endpoints* para que el momento en que los usuarios deseen crear una cuenta en la plataforma se guarde la información en la base de datos.	

Observación:

- Se debe tener en cuenta que la cuenta no haya sido registrada previamente consultando a la base de datos.
- Se debe diferenciar al Administrador de los usuarios en la plataforma.
- Se debe de tomar en cuenta las validaciones en el frontend para la creación de campos dentro de la base de datos.

Product Backlog

Es una lista ordenada de todas las funcionalidades, mejoras y requisitos pendientes de un producto o proyecto. Representa el conjunto completo de elementos que se deben desarrollar para satisfacer las necesidades del cliente o usuario final. El *Product Backlog* es dinámico y evoluciona a lo largo del tiempo, ya que se va refinando y ajustando en función de los cambios en los requerimientos y las prioridades [17]. El mismo se encuentra detallado en el

ANEXO II **-** *Product Backlog* del presente documento.

Sprint Backlog

Es un elemento fundamental en la metodología ágil Scrum. Consiste en una lista detallada de todas las tareas que deben ser completadas durante un *sprint* específico. Este *backlog* se crea mediante la colaboración entre el equipo de desarrollo y el *Product Owner*, quien prioriza las tareas en función del valor que aportan al producto [18]. El *sprint backlog* proporciona una visión clara de los objetivos a alcanzar durante el *sprint*, junto con las historias de usuario y los criterios de aceptación asociados a cada tarea. El mismo se encuentra detallado en el

ANEXO II **-** *Sprint Backlog* del presente documento.

2.2 Diseño de la Arquitectura

El diseño de arquitectura es un proceso esencial en el desarrollo de software que implica la creación de una estructura y una planificación para el sistema en su conjunto [19]. Este proceso se enfoca en tomar decisiones estratégicas y de alto nivel que definen la organización de los componentes del software, sus interacciones y las principales funcionalidades del sistema.

Arquitectura de Datos

Es clave en el éxito del desarrollo de software contar con una arquitectura de datos robusta, pues de esta depende la gestión de la información. La arquitectura de datos es la encargada recopilar, transformar, distribuir y consumir de manera eficiente los datos dentro de sistema. Para lograr su diseño efectivo es necesario tener claro los requisitos del negocio.

En la **Figura 2.1** se muestra el esquema de la base de datos y la manera en que se relacionan las tablas creadas para las transferencias de datos.

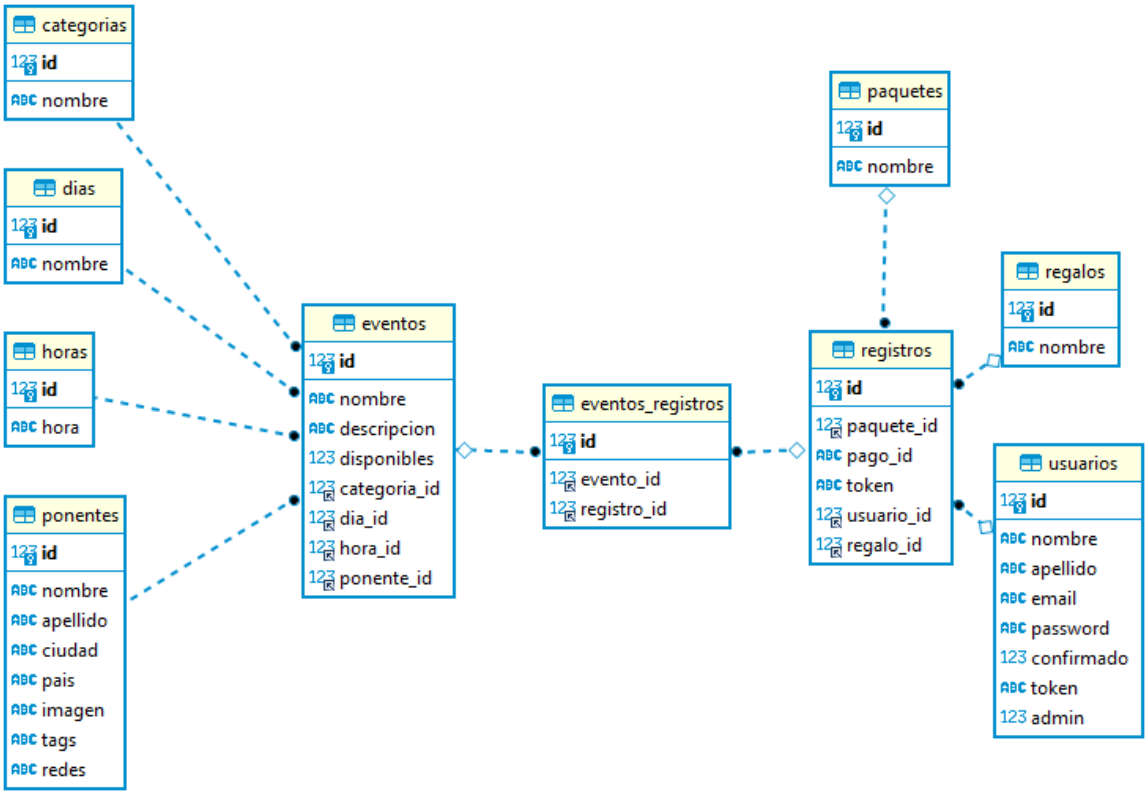

Figura 2.1 Base de Datos Relacional del Proyecto

Patrón arquitectónico

El Modelo-Vista-Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de software, especialmente en el ámbito de la programación orientada a objetos [20]. Su objetivo principal es dividir la aplicación en tres componentes interconectados pero independientes:

- **Modelo:** Representa los datos y la lógica de negocio. Es responsable de gestionar el acceso, la manipulación y el almacenamiento de los datos, así como de implementar las reglas y operaciones relacionadas con la lógica de negocio.
- **Vista:** Es la encargada de presentar los datos al usuario de una manera visualmente atractiva y comprensible. Es responsable de la interfaz gráfica y de la interacción con el usuario. La Vista obtiene los datos relevantes del Modelo y los muestra de forma adecuada, ya sea en forma de texto, imágenes, tablas u otros elementos visuales.
- **Controlador:** Interpreta las interacciones del usuario y realiza las operaciones necesarias en el Modelo, como agregar, eliminar o modificar datos. Además, el Controlador también puede realizar validaciones y aplicar reglas de negocio antes de actualizar el Modelo.

La **Figura 2.2** que se muestra presenta la arquitectura utilizada en el presente proyecto.

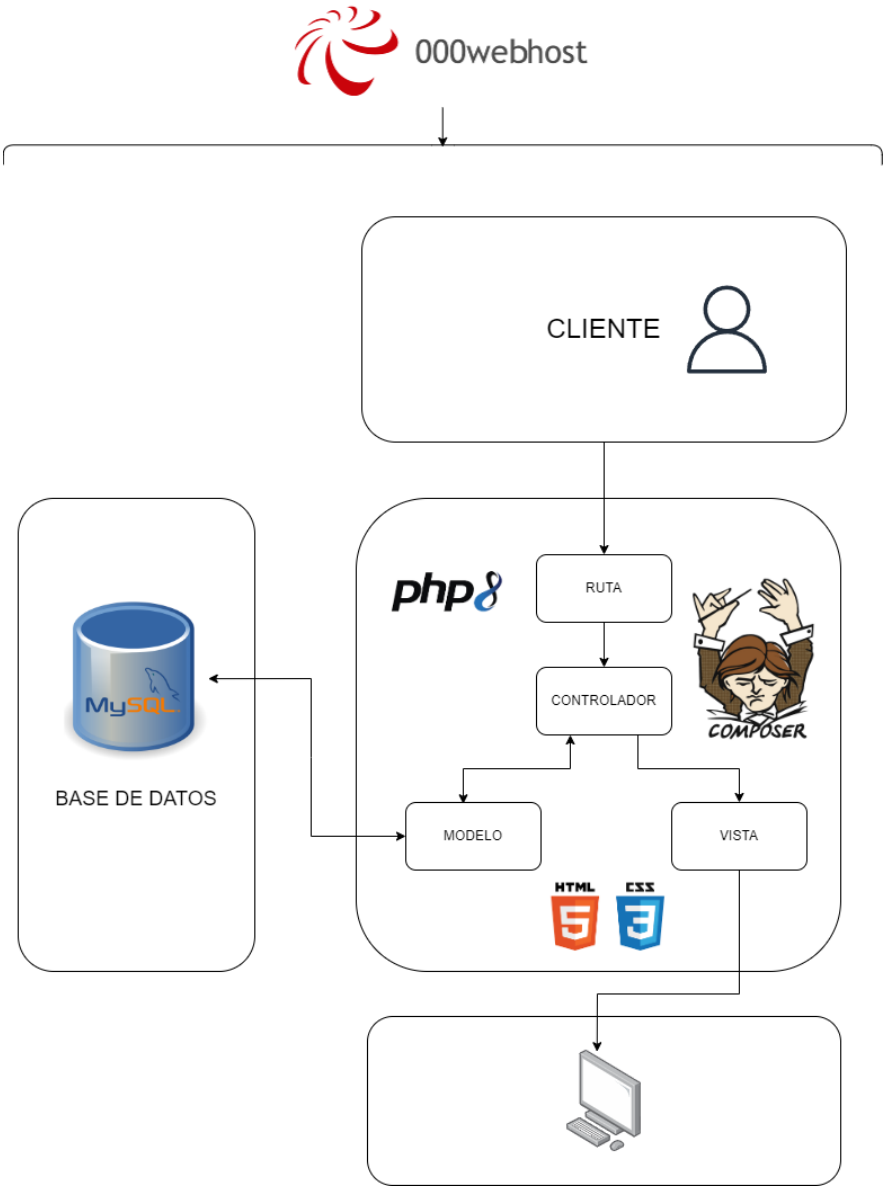

Figura 2.2 Patrón Arquitectónico-*backend*

2.3 Herramientas de desarrollo

Una vez que se ha plasmado el patrón arquitectónico, se muestra las herramientas seleccionadas para llevar a cabo el desarrollo del *backend;* tomando en cuenta los conocimientos del equipo de desarrollo y justificando su empleo para poder dar una resolución al presente problema de una manera más sencilla.

Tabla 2.3 Herramientas para el desarrollo del *backend*

HERRAMIENTA	JUSTIFICACIÓN
Composer	Es un administrador de paquetes para PHP, su uso ha facilitado la gestión de dependencias y bibliotecas de terceros [21].
PHP 8	Lenguaje de programación que se ha utilizado para el desarrollo del componente del presente proyecto. Es especialmente adecuado para la creación de sitios web interactivos y basados en bases de datos. PHP se ejecuta en el lado del servidor, lo que significa que el código PHP se procesa en el servidor antes de que se envíe la página web al cliente [22].
MySQL	Se utiliza para almacenar, administrar y recuperar datos en aplicaciones basadas en web y otros sistemas de software [23]. Es donde se almacena la información del presente proyecto como una base de datos relacional.
DBeaver	Se utiliza para realizar diversas tareas relacionadas con la base de datos, como diseñar y editar esquemas, ejecutar consultas SQL, visualizar y modificar datos, administrar usuarios y permisos, entre otras funcionalidades [24]. Ha proporcionado una interfaz gráfica intuitiva y potente, facilitando la navegación y el uso eficiente de la BDD.

Librerías

La **Tabla 2.4** presenta las librerías utilizadas para desarrollar el *backend* dando la facilidad de usar servicios que ya han sido creados por otros desarrolladores.

Tabla 2.4 Librerías para el desarrollo del *backend*

LIBRERÍA	DESCRIPCIÓN
PHPMailer	Se utiliza para enviar correos electrónicos desde una aplicación web. Proporciona una interfaz sencilla y completa para la creación y envío de mensajes de correo electrónico, lo que ha facilitado la integración de funcionalidades de correo electrónico [25].

3 RESULTADOS

A continuación, en esta sección se muestra de manera detallada los resultados que se han obtenido en cada uno de los *sprints* a lo largo del desarrollo del proyecto.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo

Dentro del *Sprint* 0 se encuentran los resultados de las siguientes tareas:

- Recopilación de Requerimientos
- Elaboración del Modelo de Datos
- Estructura del Proyecto

Recopilación y Definición de Requerimientos

Implementar *endpoints* para el registro de usuarios

El componente *backend* cuenta con un *endpoint* que permite el registro de usuarios para enviar la información hacia la base de datos.

Implementar *endpoints* para el inicio y cierre de sesión

El componente *backend* cuenta con un *endpoint* para el inicio de sesión de los usuarios al verificar la información en la base de datos de que existe la cuenta. A su vez el *endpoint*, cuenta con un método que permite el cierre de sesión por parte del usuario.

Implementar *endpoints* para verificación de email

El componente *backend* cuenta con un *endpoint* que permite verificar que el email que está haciendo uso el usuario existe, y permite activar la cuenta que ha sido creada mediante un token.

Implementar *endpoints* para la recuperación de cuenta

El componente *backend* cuenta con un *endpoint* que permite que el usuario pueda reestableces su contraseña mediante un email, que es enviado a la dirección de correo electrónico con la que se ha registrado la cuenta.

Implementar *endpoints* para la información de los ponentes

El componente *backend* posee un *endpoint* que permite añadir a la base de datos la información personal, las áreas de experiencia y las URL de las redes sociales que usa

cada ponente. Dentro de la funcionalidad también se puede editar y eliminar la información en caso de que lo necesite el administrador.

Implementar *endpoints* para el registro de eventos

El componente *backend* posee un *endpoint*, que permite añadir a la base de datos la información del día del evento y el ponente que ha realizar la charla. Dentro de la funcionalidad también se puede editar y eliminar la información en caso de que lo necesite el administrador.

Implementar *endpoints* para los pagos en línea dentro de la plataforma

El componente *backend* posee un *endpoint* que permite los pagos en línea de los eventos a los que el usuario desea registrarse, esto mediante la plataforma PayPal.

Implementar *endpoints* para la consulta de información de asistentes y pagos realizados

El componente *backend* posee un *endpoint* que permite consultar a la base de datos la información de los asistentes a cada conferencia y los pagos que han sido realizados a la misma.

La **Figura 3.1** y **Figura 3.2** muestran las operaciones que pueden realizar cada uno de los usuarios dentro del *backend.*

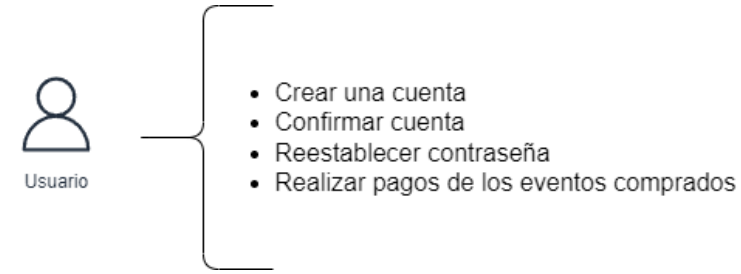

Figura 3.1 Acciones del Perfil de Usuario

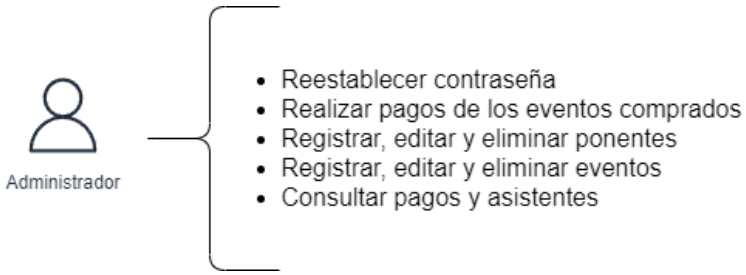

Figura 3.2 Acciones del Perfil Administrador

Elaboración del Modelo de Datos

La implementación de una base de datos es un proceso fundamental en el desarrollo de aplicaciones y sistemas de información. Implica la creación y configuración de una estructura que almacene, organice y gestione los datos de manera eficiente y segura. El diseño se la base de datos se encuentra en la sección 2.2 **Diseño de la Arquitectura** - Arquitectura de Datos de este documento.

Estructura del Proyecto

En la **Figura 3.3** se presenta la estructura del presente proyecto, misma que ha permitido llevar de una manera ordenada los paquetes manejados para la implementación exitosa del componente *backend.*

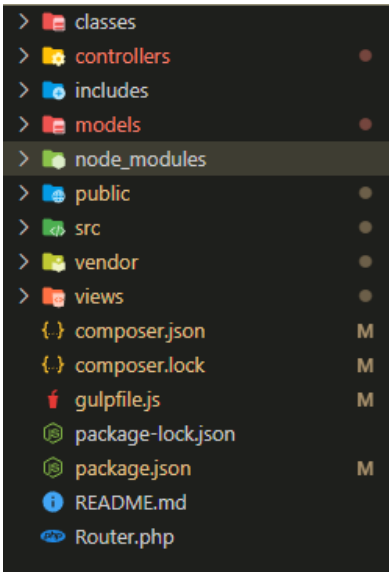

Figura 3.3 Estructura del Proyecto

3.2 *Sprint* 1. Módulo de Autenticación

Dentro del *Sprint* 1 se encuentran los resultados de las siguientes tareas:

- Generar *endpoints* para el registro de usuarios
- Generar *endpoints* para el inicio y cierre de sesión
- Generar *endpoints* para verificación de email
- Generar *endpoints* para la recuperación de cuenta

Generar endpoints para el registro de usuarios

Se debe tomar en cuenta que, dentro del presente proyecto existe un único usuario que cumple con el rol “administrador”. Por lo cual se ha creado un *endpoint* para el registro de los usuarios que desean adquirir una entrada presencial o virtual a las conferencias publicadas.

Para facilitar el trabajo del controlador que en este caso ha sido nombrado como *AuthController* se ha creado un modelo llamado *ActiveRecord*, el cual facilita las consultas necesarias a la base de datos para todo el proyecto mediante la utilización de *scripts* enviados desde el modelo hacia el controlador por medio de funciones. Es decir, implementando un CRUD (crear, leer, actualizar y eliminar) que puede ser reutilizado por todos los controladores dentro del proyecto. En la **Figura 3.4** se presenta las rutas GET y POST que se usan para presentar la vista al usuario y enviar la información del formulario respectivamente.

```php
// Crear Cuenta
$router->get('/registro', [AuthController::class, 'registro']);
$router->post('/registro', [AuthController::class, 'registro']);
```

Figura 3.4 Rutas para el Registro de Nuevos Usuarios

En la **Figura 3.5** se encuentra la función para la validación de datos, mientras que en la **Figura 3.6** se presenta el método usado para el registro de nuevos usuarios.

```php
// Validación para cuentas nuevas
public function validar_cuenta() {
    if(!$this->nombre) {
        self::$alertas['error'][] = 'El Nombre es Obligatorio';
    }
    if(!$this->apellido) {
        self::$alertas['error'][] = 'El Apellido es Obligatorio';
    }
    if(!$this->email) {
        self::$alertas['error'][] = 'El Email es Obligatorio';
    }
    if(!$this->password) {
        self::$alertas['error'][] = 'El Password no puede ir vacio';
    }
    if(strlen($this->password) < 6) {
        self::$alertas['error'][] = 'El password debe contener al menos 6 caracteres';
    }
    if($this->password !== $this->password2) {
        self::$alertas['error'][] = 'Los password son diferentes';
    }
    return self::$alertas;
}
```

Figura 3.5 Método de Validación de Datos para Registros Nuevos

```php
public static function registro(Router $router) {
    $alertas = [];
    $usuario = new Usuario;

    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        $usuario->sincronizar($_POST);
        $alertas = $usuario->validar_cuenta();
        if(empty($alertas)) {
            $existeUsuario = Usuario::where('email', $usuario->email);
            if($existeUsuario) {
                Usuario::setAlerta('error', 'El Usuario ya esta registrado');
                $alertas = Usuario::getAlertas();
            } else {
                // Hashear el password
                $usuario->hashPassword();
                // Eliminar password2
                unset($usuario->password2);
                // Generar el Token
                $usuario->crearToken();
                // Crear un nuevo usuario
                $resultado =  $usuario->guardar();
                // Enviar email
                $email = new Email($usuario->email, $usuario->nombre, $usuario->token);
                $email->enviarConfirmacion();
                if($resultado) {
                    header('Location: /mensaje');
                }
            }
        }
    }

    // Render a la vista
    $router->render('auth/registro', [
        'titulo' => 'Crea tu cuenta en EPN Conferencias',
        'usuario' => $usuario,
        'alertas' => $alertas
    ]);
}
```

Figura 3.6 Método para Registrar un Nuevo Usuario

Parte fundamental del proceso de desarrollo de software es realizar pruebas desde las fases tempranas de la codificación, es por ello que por cada *endpoints* implementado se presenta su respectiva prueba unitaria.

La **Figura 3.7** y **Figura 3.8** presentan los resultados de las pruebas unitarias para las rutas del registro. Que en el caso del método GET devuelve la vista en donde se deben ingresar los datos para crear la cuenta. Mientras que en el caso del método POST devuelve la vista con el mensaje de que se ha creado la cuenta exitosamente, pero debe ser confirmada mediante un email que ha sido enviado al correo utilizado por el usuario.

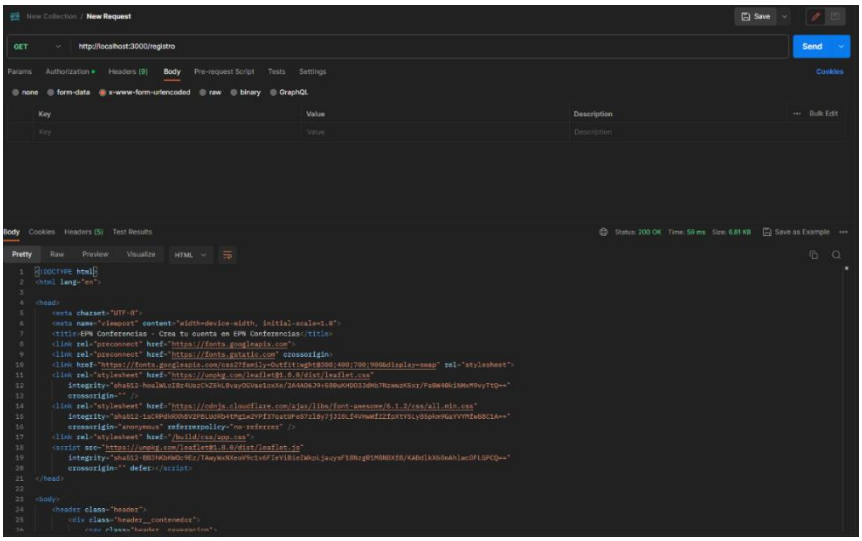

Figura 3.7 Prueba Unitaria método GET registro

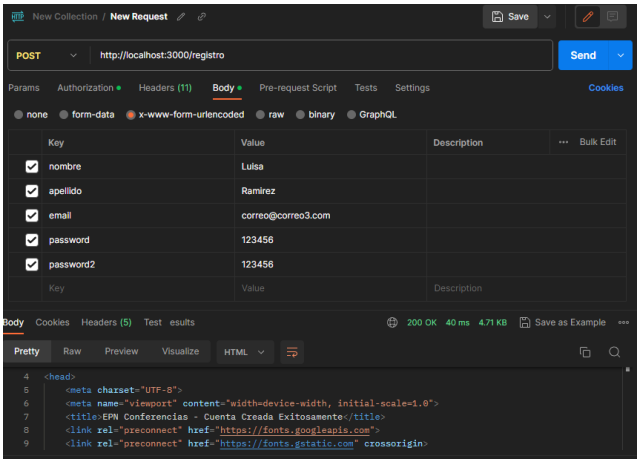

Figura 3.8 Prueba Unitaria Método POST

Generar endpoints para el inicio y cierre de sesión

Se ha implementado un *endpoint* que permite por medio de los métodos de *login* y *logout* el inicio y cierre de sesión dentro del proyecto. Además de un método que permite cerrar sesión si el usuario lo desea. En la **Figura 3.9** se presenta las rutas GET y POST que se usan para presentar la vista al usuario y enviar la información para el inicio de sesión.

```php
// Login
$router->get('/login', [AuthController::class, 'login']);
$router->post('/login', [AuthController::class, 'login']);
$router->post('/logout', [AuthController::class, 'logout']);
```

Figura 3.9 Rutas para el *Login* y *Logout*

En la **Figura 3.10** se encuentra el método usado para el cierre de sesión. Mientras que, en la **Figura 3.11** se presenta el método para el inicio de sesión.

```php
public static function login(Router $router) {
    $alertas = [];
    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        $usuario = new Usuario($_POST);
        $alertas = $usuario->validarLogin();
        if(empty($alertas)) {
            // Verificar quel el usuario exista
            $usuario = Usuario::where('email', $usuario->email);
            if(!$usuario || !$usuario->confirmado ) {
                Usuario::setAlerta('error', 'El Usuario No Existe o no esta confirmado');
            } else {
                // El Usuario existe
                if( password_verify($_POST['password'], $usuario->password) ) {
                    // Iniciar la sesión
                    session_start();
                    $_SESSION['id'] = $usuario->id;
                    $_SESSION['nombre'] = $usuario->nombre;
                    $_SESSION['apellido'] = $usuario->apellido;
                    $_SESSION['email'] = $usuario->email;
                    $_SESSION['admin'] = $usuario->admin ?? null;
                    //Redirección
                    if($usuario->admin){
                        header('Location: /admin/dashboard');
                        echo("sesion iniciada");
                    }else{
                        header('Location: /finalizar-registro');
                    }
                } else {
                    Usuario::setAlerta('error', 'Password Incorrecto');
                }
            }
        }
    }
    $alertas = Usuario::getAlertas();
    // Render a la vista
    $router->render('auth/login', [
        'titulo' => 'Iniciar Sesión',
        'alertas' => $alertas
    ]);
}
```

Figura 3.10 Método para Inicio de Sesión

```php
public static function logout() {
    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        session_start();
        $_SESSION = [];
        header('Location:/login');
    }

}
```

Figura 3.11 Método para el Cierre de Sesión

Mientras que, en la **Figura 3.12**, **Figura 3.13** y **Figura 3.14** se presenta el resultado de las pruebas unitarias para las rutas del *login* y *logout*. Como respuesta de GET se tiene la vista para el inicio de sesión mientras que para los métodos POST se devuelven las vistas del panel administrativo en caso de que sea un rol “administrador” o las opciones de pago para los diferentes paquetes que existen dentro de la plataforma en caso de que sea un rol “usuario”. Y a su vez se presenta la vista de inicio de sesión al momento de utilizar el método de *logout*.

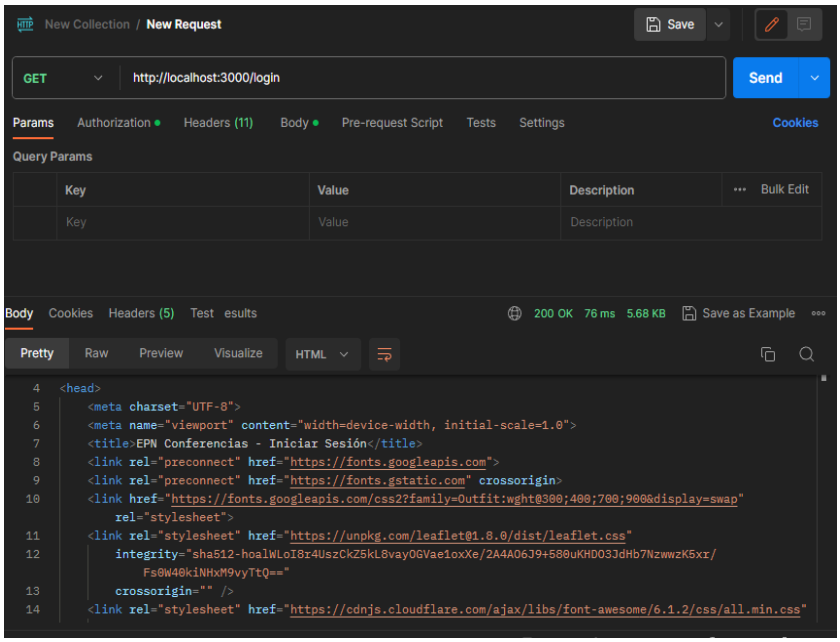

Figura 3.12 Prueba Unitaria GET para el Login

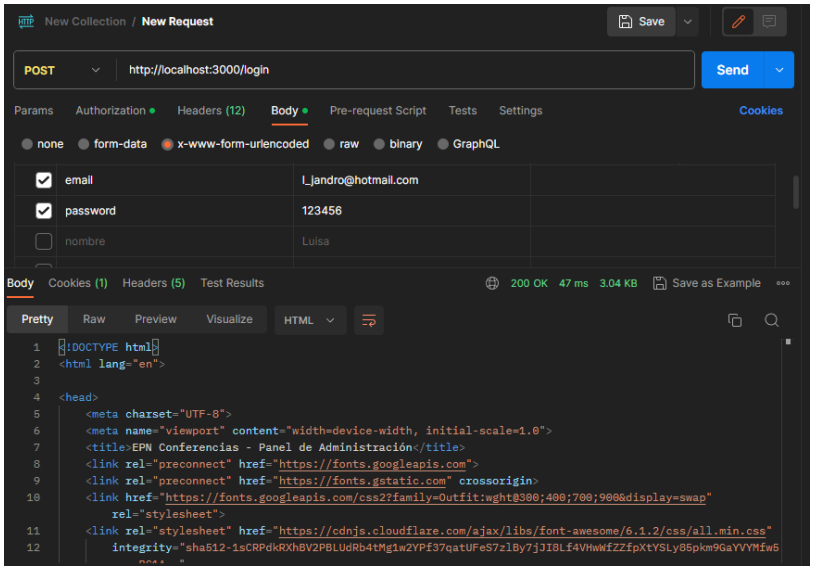

Figura 3.13 Prueba Unitaria POST para el Login

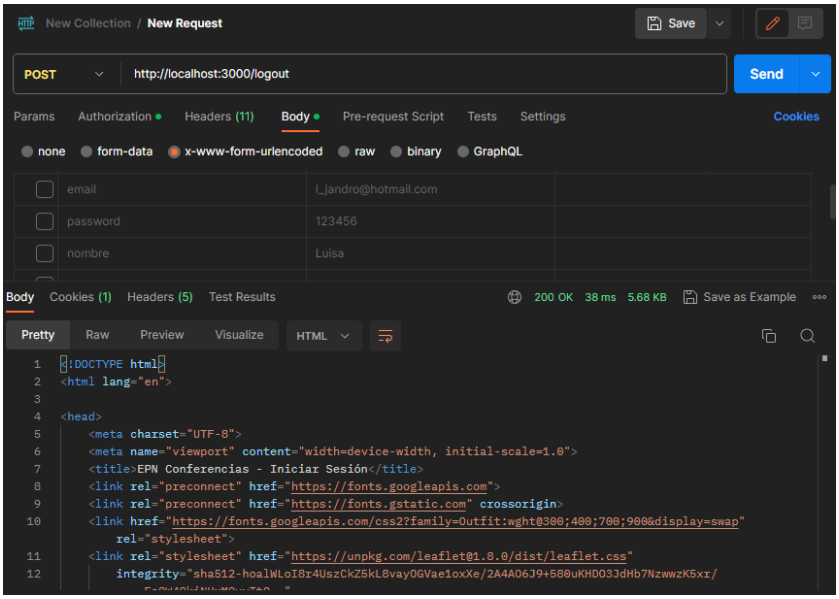

Figura 3.14 Prueba Unitaria POST *Logout*

Generar endpoints para verificación de email

El *AuthController* al momento de crear la cuenta dentro de la plataforma, mantiene un método para enviar un email al usuario y por medio de un token verifica la dirección electrónica con la que se ha creado la cuenta.

Por medio de la URL, se recupera el token con el cual se hace la verificación por medio de un método POST para determinar su existencia en la base de datos. Se realiza una búsqueda para empatar el token enviado en el email con el generado en la base de datos. En caso de que no se encuentre se envía un mensaje al usuario indicándole que la cuenta

no ha sido confirmada. Caso contrario se envía un mensaje diciendo que la cuenta ha sido comprobada exitosamente y puede iniciar sesión con la cuenta creada.

En la **Figura 3.15** se presenta las rutas usadas para la confirmación de cuenta.

```
// Confirmación de Cuenta
$router->get('/mensaje', [AuthController::class, 'mensaje']);
$router->get('/confirmar-cuenta', [AuthController::class, 'confirmar']);
```

Figura 3.15 Rutas para la confirmación de cuenta

En la **Figura 3.16** se encuentra el método para enviar el email de confirmación. En la **Figura 3.17** y **Figura 3.18** se presenta el resultado de las pruebas unitarias para las rutas de confirmación.

```
public function enviarConfirmacion() {

    // create a new object
    $mail = new PHPMailer();
    $mail->isSMTP();
    $mail->Host = $_ENV['EMAIL_HOST'];
    $mail->SMTPAuth = true;
    $mail->Port = $_ENV['EMAIL_PORT'];
    $mail->Username = $_ENV['EMAIL_USER'];
    $mail->Password = $_ENV['EMAIL_PASS'];

    $mail->setFrom('cuentas@devwebcamp.com');
    $mail->addAddress($this->email, $this->nombre);
    $mail->Subject = 'Confirma tu Cuenta';

    // Set HTML
    $mail->isHTML(TRUE);
    $mail->CharSet = 'UTF-8';

    $contenido = '<html>';
    $contenido .= "<p><strong>Hola " . $this->nombre . "</strong> Has Registrado Correctamente tu cuenta en DevWebCamp; pero es necesario confirmarla</p>";
    $contenido .= "<p>Presiona aqui: <a href='" . $_ENV['HOST'] . "/confirmar-cuenta?token=" . $this->token . "'>Confirmar Cuenta</a>";
    $contenido .= "<p>Si tu no creaste esta cuenta; puedes ignorar el mensaje</p>";
    $contenido .= '</html>';
    $mail->Body = $contenido;

    //Enviar el mail
    $mail->send();
}
```

Figura 3.16 Método para Enviar Mail de Confirmación

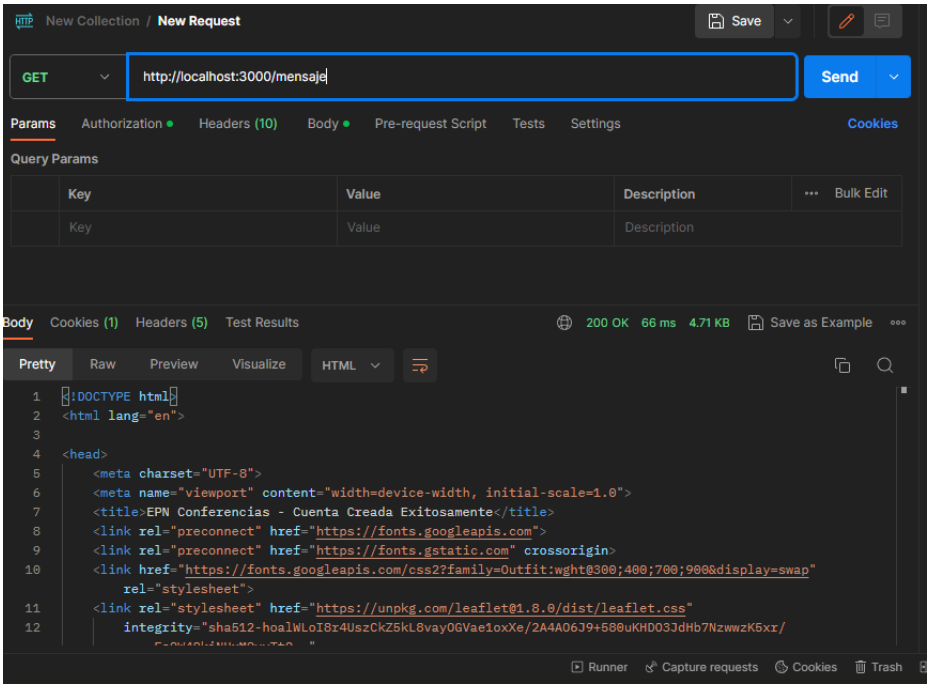

Figura 3.17 Prueba Unitaria GET Mensaje de Confirmación

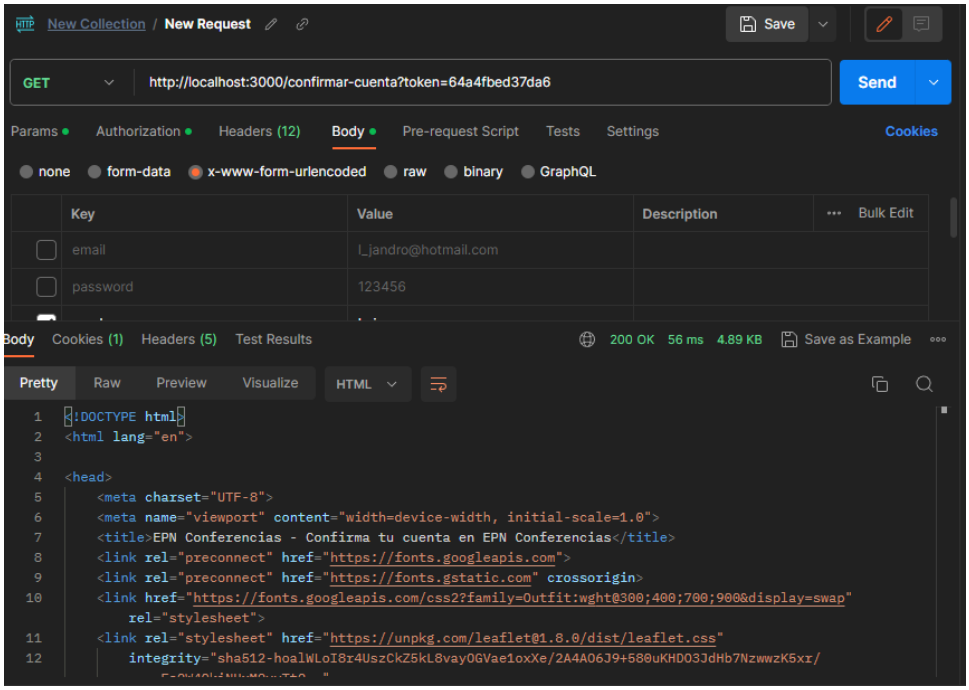

Figura 3.18 Prueba Unitaria para Confirmación de Cuentas Mediante Token

Generar endpoints para la recuperación de cuenta

Además, se ha encontrado la necesidad de reestablecer la contraseña en caso de que el usuario olvide sus credenciales de acceso. Utilizando la misma lógica que para poder confirmar la cuenta en la plataforma, se ha creado dentro del *endpoint AuthController* una función por la cual el usuario pueda ingresar una nueva contraseña para actualizarla en la base de datos al momento de verificar el token que enviado a su email.

En la **Figura 3.19** se presenta las rutas usadas para reestablecer la cuenta.

```
// Formulario de olvide mi password
$router->get('/olvide', [AuthController::class, 'olvide']);
$router->post('/olvide', [AuthController::class, 'olvide']);

// Colocar el nuevo password
$router->get('/reestablecer', [AuthController::class, 'reestablecer']);
$router->post('/reestablecer', [AuthController::class, 'reestablecer']);
```

Figura 3.19 Rutas para reestablecer la cuenta

En la **Figura 3.20** se encuentra el método para enviar el email que permite reestablecer la contraseña.

```php
public function enviarInstrucciones() {

    // create a new object
    $mail = new PHPMailer();
    $mail->isSMTP();
    $mail->Host = $_ENV['EMAIL_HOST'];
    $mail->SMTPAuth = true;
    $mail->Port = $_ENV['EMAIL_PORT'];
    $mail->Username = $_ENV['EMAIL_USER'];
    $mail->Password = $_ENV['EMAIL_PASS'];
    $mail->setFrom('cuentas@EPNConferencias.com');
    $mail->addAddress($this->email, $this->nombre);
    $mail->Subject = 'Reestablece tu password';
    // Set HTML
    $mail->isHTML(TRUE);
    $mail->CharSet = 'UTF-8';

    $contenido = '<html>';
    $contenido .= "<p><strong>Hola " . $this->nombre .  "</strong> Has solicitado reestablecer tu password, sigue el siguiente enlace para hacerlo.</p>";
    $contenido .= "<p>Presiona aquí: <a href='" . $_ENV['HOST'] . "/reestablecer?token=" . $this->token . "'>Reestablecer Password</a>";
    $contenido .= "<p>Si tu no solicitaste este cambio, puedes ignorar el mensaje</p>";
    $contenido .= '</html>';
    $mail->Body = $contenido;

    //Enviar el mail
    $mail->send();
}
}
```

Figura 3.20 Método para Envío de Email para Reestableces Contraseña

3.3 *Sprint* 2. Módulo de Ponentes

Dentro del *Sprint* 2 se encuentra el resultado de la siguiente tarea:

- Generar *endpoints* para la información de los ponentes

Generar endpoints para la información de los ponentes

Para poder desarrollar el módulo de ponentes, se ha procedido a crear un modelo que tiene en su constructor los campos que se han creado en la base de datos. Permitiendo que sea más sencillo mapear la información a ser enviada por medio del *endpoint* hacia la tabla de ponentes.

De la misma manera se ha validado que la información que sea de carácter obligatoria sea confirmada antes de ser enviada hacia la base de datos. Además, se ha creado un controlador con el nombre de *PonentesController* que permite que el *endpoint* pueda crear, editar o eliminar un ponente que ha sido creado por el administrador.

Por medio de un método POST, el e*ndpoint* permite enviar los campos como el nombre, el apellido, la ciudad de residencia entre otra información del ponente que a través de la sincronización en el *ActiveRecord* permite mapear los datos para ser enviados a sus respectivas columnas dentro de la tabla de ponentes.

Además, permite que la información del ponente sea consultada desde la base de datos y actualizada en caso de ser necesario por el administrador a través de un método GET por

medio del id del ponente. Utilizando la misma lógica, se elimina la información del ponente de la base de datos a través de su id.

En la **Figura 3.21** se muestran las rutas a ser usadas. Mientras de la **Figura 3.22** a la **Figura 3.24** presentan los métodos utilizados para el CRUD.

```php
$router->get('/admin/ponentes', [PonentesController::class, 'index']);
$router->get('/admin/ponentes/crear', [PonentesController::class, 'crear']);
$router->post('/admin/ponentes/crear', [PonentesController::class, 'crear']);
$router->get('/admin/ponentes/editar', [PonentesController::class, 'editar']);
$router->post('/admin/ponentes/editar', [PonentesController::class, 'editar']);
$router->post('/admin/ponentes/eliminar', [PonentesController::class, 'eliminar']);
```

Figura 3.21 Rutas Usadas para el *Endpoint* de Ponentes

```php
public static function crear(Router $router) {
    if(!is_admin()) {
        header('Location: /login');
    }
    $alertas = [];
    $ponente = new Ponente;
    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_admin()) {
            header('Location: /login');
        }
        // Leer imagen
        if(!empty($_FILES['imagen']['tmp_name'])) {
            $carpeta_imagenes = '../public/img/speakers';
            // Crear la carpeta si no existe
            if(!is_dir($carpeta_imagenes)) {
                mkdir($carpeta_imagenes, 0755, true);
            }
            $imagen_png = Image::make($_FILES['imagen']['tmp_name'])->fit(800,800)->encode('png', 80);
            $imagen_webp = Image::make($_FILES['imagen']['tmp_name'])->fit(800,800)->encode('webp', 80);
            $nombre_imagen = md5( uniqid( rand(), true) );
            $_POST['imagen'] = $nombre_imagen;
        }
        $_POST['redes'] = json_encode( $_POST['redes'], JSON_UNESCAPED_SLASHES );
        $ponente->sincronizar($_POST);
        // validar
        $alertas = $ponente->validar();
        // Guardar el registro
        if(empty($alertas)) {
            // Guardar las imagenes
            $imagen_png->save($carpeta_imagenes . '/' . $nombre_imagen . ".png" );
            $imagen_webp->save($carpeta_imagenes . '/' . $nombre_imagen . ".webp" );
            // Guardar en la BD
            $resultado = $ponente->guardar();
            if($resultado) {
                header('Location: /admin/ponentes');
            }
        }
    }

    $router->render('admin/ponentes/crear', [
        'titulo' => 'Registrar Ponente',
        'alertas' => $alertas,
        'ponente' => $ponente,
        'redes' => json_decode($ponente->redes)
    ]);
}
```

Figura 3.22 Método Crear Ponente

```php
public static function editar(Router $router) {
    if(!is_admin()) {
        header('Location: /login');
    }
    $alertas = [];
    // Validar el ID
    $id = $_GET['id'];
    $id = filter_var($id, FILTER_VALIDATE_INT);
     if(!$id) {
        header('Location: /admin/ponentes');
    }
    // Obtener ponente a Editar
    $ponente = Ponente::find($id);
    if(!$ponente) {
        header('Location: /admin/ponentes');
    }
    $ponente->imagen_actual = $ponente->imagen;
    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_admin()) {
            header('Location: /login');
        }
    if(!empty($_FILES['imagen']['tmp_name'])) {
            $carpeta_imagenes = '../public/img/speakers';
            // Crear la carpeta si no existe
            if(!is_dir($carpeta_imagenes)) {
                mkdir($carpeta_imagenes, 0755, true);
            }
            $imagen_png = Image::make($_FILES['imagen']['tmp_name'])->fit(800,800)->encode('png', 80);
            $imagen_webp = Image::make($_FILES['imagen']['tmp_name'])->fit(800,800)->encode('webp', 80);
            $nombre_imagen = md5( uniqid( rand(), true) );
            $_POST['imagen'] = $nombre_imagen;
        } else {
            $_POST['imagen'] = $ponente->imagen_actual;
        }
        $_POST['redes'] = json_encode( $_POST['redes'], JSON_UNESCAPED_SLASHES );
        $ponente->sincronizar($_POST);
        $alertas = $ponente->validar();
        if(empty($alertas)) {
            if(isset($nombre_imagen)) {
                $imagen_png->save($carpeta_imagenes . '/' . $nombre_imagen . ".png" );
                $imagen_webp->save($carpeta_imagenes . '/' . $nombre_imagen . ".webp" );
            }
            $resultado = $ponente->guardar();
            if($resultado) {
                header('Location: /admin/ponentes');
            }}}
$router->render('admin/ponentes/editar', [
        'titulo' => 'Actualizar Ponente',
        'alertas' => $alertas,
        'ponente' => $ponente,
        'redes' => json_decode($ponente->redes)
    ]);
```

Figura 3.23 Método Editar Ponente

```php
public static function eliminar() {

    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_admin()) {
            header('Location: /login');
        }

        $id = $_POST['id'];
        $ponente = Ponente::find($id);
        if(!isset($ponente) ) {
            header('Location: /admin/ponentes');
        }
        $resultado = $ponente->eliminar();
        if($resultado) {
            header('Location: /admin/ponentes');
        }
    }

}
}
```

Figura 3.24 Método Eliminar Ponente

3.4 *Sprint* 3. Módulo de Eventos

Dentro del *Sprint* 3 se encuentra el resultado de la siguiente tarea:

- Generar *endpoints* para el registro de eventos

Generar endpoints para el registro de eventos

Para poder desarrollar el módulo de eventos, se ha procedido a crear un modelo que tiene en su constructor los campos que se han creado en la base de datos. Permitiendo que sea más sencillo mapear la información a ser enviada por medio del *endpoint* hacia la tabla de eventos.

De la misma manera se ha validado que la información que sea de carácter obligatoria sea confirmada antes de ser enviada hacia la base de datos. Además, se ha creado un controlador con el nombre de *EventosController* que permite que el *endpoint* pueda crear, editar o eliminar un evento que ha sido creado por el administrador.

Por medio de un método POST, el e*ndpoint* permite enviar los campos como el nombre, la descripción, el tipo sea una conferencia o un workshop, el día y la hora, entre otra información del evento que a través de la sincronización en el *ActiveRecord* permite mapear los datos para ser enviados a sus respectivas columnas dentro de la tabla de eventos.

Además, permite que la información del evento sea consultada desde la base de datos y actualizada en caso de ser necesario por el administrador a través de un método GET por medio del id del evento. Utilizando la misma lógica, se elimina la información del evento de la base de datos a través de su id.

En la **Figura 3.25** se muestran las rutas a ser usadas. Mientras que la **Figura 3.26**, **Figura 3.27** y **Figura 3.28** presentan los métodos utilizados para el CRUD.

```
$router->get('/admin/eventos/crear', [EventosController::class, 'crear']);
$router->post('/admin/eventos/crear', [EventosController::class, 'crear']);
$router->get('/admin/eventos/editar', [EventosController::class, 'editar']);
$router->post('/admin/eventos/editar', [EventosController::class, 'editar']);
$router->post('/admin/eventos/eliminar', [EventosController::class, 'eliminar']);
```

Figura 3.25 Rutas Usadas para el *Endpoint* de Eventos

```php
public static function crear(Router $router) {
    if(!is_admin()) {
        header('Location: /login');
        return;
    }

    $alertas = [];

    $categorias = Categoria::all('ASC');
    $dias = Dia::all('ASC');
    $horas = Hora::all('ASC');

    $evento = new Evento;

    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_admin()) {
            header('Location: /login');
            return;
        }

        $evento->sincronizar($_POST);

        $alertas = $evento->validar();

        if(empty($alertas)) {
            $resultado = $evento->guardar();
            if($resultado) {
                header('Location: /admin/eventos');
                return;
            }
        }
    }

    $router->render('admin/eventos/crear', [
        'titulo' => 'Registrar Evento',
        'alertas' => $alertas,
        'categorias' => $categorias,
        'dias' => $dias,
        'horas' => $horas,
        'evento' => $evento
    ]);
}
```

Figura 3.26 Método para crear un nuevo Evento

```php
public static function editar(Router $router) {

    if(!is_admin()) {
        header('Location: /login');
        return;
    }
    $alertas = [];
    $id = $_GET['id'];
    $id = filter_var($id, FILTER_VALIDATE_INT);
    if(!$id) {
        header('Location: /admin/eventos');
        return;
    }
    $categorias = Categoria::all('ASC');
    $dias = Dia::all('ASC');
    $horas = Hora::all('ASC');
    $evento = Evento::find($id);
    if(!$evento) {
        header('Location: /admin/eventos');
        return;
    }
    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_admin()) {
            header('Location: /login');
            return;
        }
        $evento->sincronizar($_POST);
        $alertas = $evento->validar();
        if(empty($alertas)) {
            $resultado = $evento->guardar();
            if($resultado) {
                header('Location: /admin/eventos');
                return;
            }
        }
    }

    $router->render('admin/eventos/editar', [
        'titulo' => 'Editar Evento',
        'alertas' => $alertas,
        'categorias' => $categorias,
        'dias' => $dias,
        'horas' => $horas,
        'evento' => $evento
    ]);
}
```

Figura 3.27 Método para Editar un Evento

```php
    public static function eliminar() {

        if($_SERVER['REQUEST_METHOD'] === 'POST') {
            if(!is_admin()) {
                header('Location: /login');
                return;
            }

            $id = $_POST['id'];
            $evento = Evento::find($id);
            if(!isset($evento) ) {
                header('Location: /admin/eventos');
                return;
            }
            $resultado = $evento->eliminar();
            if($resultado) {
                header('Location: /admin/eventos');
                return;
            }
        }

    }
}
```

Figura 3.28 Método para Eliminar un Evento

3.5 *Sprint* 4. Módulo de Pagos

Dentro del *Sprint* 4 se encuentra el resultado de la siguiente tarea:

- Generar *endpoints* para los pagos en línea y consulta de información de asistentes y pagos realizado.

  Generar endpoints para los pagos en línea y consulta de información de asistentes y pagos realizado.

Para el módulo de pagos se ha considerado tres tipos de suscripción, una gratuita, otra presencial y por último la de tipo virtual. Cada una al momento de ser registrada crea un token con el cual el usuario puede asistir o ingresar a los eventos dependiendo del tipo de paquete que decida obtener.

Siguiendo con la lógica dentro del proyecto, se ha creado un modelo de registro que tiene en su constructor los campos que se han creado en la base de datos. Permitiendo que sea más sencillo mapear la información a ser enviada por medio del *endpoint* hacia la tabla de registros.

Al mismo tiempo, el presente módulo permite enviar por medio de un método POST, la información de los pagos realizados, en este caso a través de la página de PayPal por medio de la información que permite recolectar la API de esta plataforma de pagos; haciendo que se pueda llevar un control de las transacciones realizadas por los usuarios.

Tanto la información del usuario, como la información del pago es enviada hacia la base de datos para tener un registro de todas las transacciones que han sido realizadas y así puedan ser consultadas mediante el panel de administrador de ser necesario.

Además, se lleva un control acerca de la disponibilidad en cada uno de los eventos para que no puedan ingresar más usuarios de los que son posibles. Por medio de un método POST se crea los registros en la base de datos para la tabla eventos_registro y así se pueda relacionar los eventos con los registros de la plataforma y controlar las asistencias a los eventos con los registros ya realizados.

En la **Figura 3.29** se presenta el modelo para cada uno de los registros. Mientras que en la **Figura 3.30** y **Figura 3.31** se presentan las funciones dentro del *endpoint* en caso de que el usuario escoja un paquete gratuito o de paga.

```php
<?php

namespace Model;

class Registro extends ActiveRecord {
    protected static $tabla = 'registros';
    protected static $columnasDB = ['id', 'paquete_id', 'pago_id', 'token', 'usuario_id', 'regalo_id'];


    public function __construct($args = [])
    {
        $this->id = $args['id'] ?? null;
        $this->paquete_id = $args['paquete_id'] ?? '';
        $this->pago_id = $args['pago_id'] ?? '';
        $this->token = $args['token'] ?? '';
        $this->usuario_id = $args['usuario_id'] ?? '';
        $this->regalo_id = $args['regalo_id'] ?? 1;
    }

}
```

Figura 3.29 Modelo de Registro

```php
public static function gratis(Router $router) {

    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_auth()) {
            header('Location: /login');
            return;
        }

        // Verificar si el usuario ya esta registrado
        $registro = Registro::where('usuario_id', $_SESSION['id']);
        if(isset($registro) && $registro->paquete_id === "3") {
            header('Location: /boleto?id=' . urlencode($registro->token));
            return;
        }

        $token = substr( md5(uniqid( rand(), true )), 0, 8);

        // Crear registro
        $datos = [
            'paquete_id' => 3,
            'pago_id' => '',
            'token' => $token,
            'usuario_id' => $_SESSION['id']
        ];

        $registro = new Registro($datos);
        $resultado = $registro->guardar();

        if($resultado) {
            header('Location: /boleto?id=' . urlencode($registro->token));
            return;
        }

    }
}
```

Figura 3.30 Función para el paquete gratuito

```php
public static function pagar(Router $router) {

    if($_SERVER['REQUEST_METHOD'] === 'POST') {
        if(!is_auth()) {
            header('Location: /login');
            return;
        }

        // Validar que Post no venga vacio
        if(empty($_POST)) {
            echo json_encode([]);
            return;
        }

        // Crear el registro
        $datos = $_POST;
        $datos['token'] = substr( md5(uniqid( rand(), true )), 0, 8);
        $datos['usuario_id'] = $_SESSION['id'];

        try {
            $registro = new Registro($datos);

            $resultado = $registro->guardar();
            echo json_encode($resultado);
        } catch (\Throwable $th) {
            echo json_encode([
                'resultado' => 'error'
            ]);
        }

    }
}
```

Figura 3.31 Función para el paquete de pago

3.6 *Sprint* 5. *Pruebas* y Despliegue de *Endpoints*

Al finalizar la etapa de desarrollo, antes de realizar el despliegue de un aplicativo se deben realizar pruebas para analizar el correcto funcionamiento de todos los componentes. Este *sprint* tiene como resultados las diferentes pruebas realizadas al *backend,* así como el despliegue de este.

Dentro del *Sprint* 5 se encuentra los resultados de las siguientes tareas:

- Pruebas unitarias
- Pruebas de carga
- Pruebas de rendimiento
- Despliegue de *endpoints* a 000WebHost

Pruebas unitarias

Este tipo de pruebas permiten verificar la funcionalidad del código implementado durante y después del desarrollo. Permitiendo encontrar posibles fallos en el código de manera que se pueda asegurar el éxito del proyecto, demostrando el correcto funcionamiento de partes concretas del código. La **Figura 3.32** muestra el resultado exitoso de la prueba unitaria realizada al método GET de reestablecer contraseña, mientras que la **Figura 3.33** muestra el resultado exitoso de la prueba unitaria realizada al método POST de reestablecer

contraseña. La evidencia del resto de los métodos probados de forma unitaria se presenta en el

ANEXO II en el apartado de Pruebas.

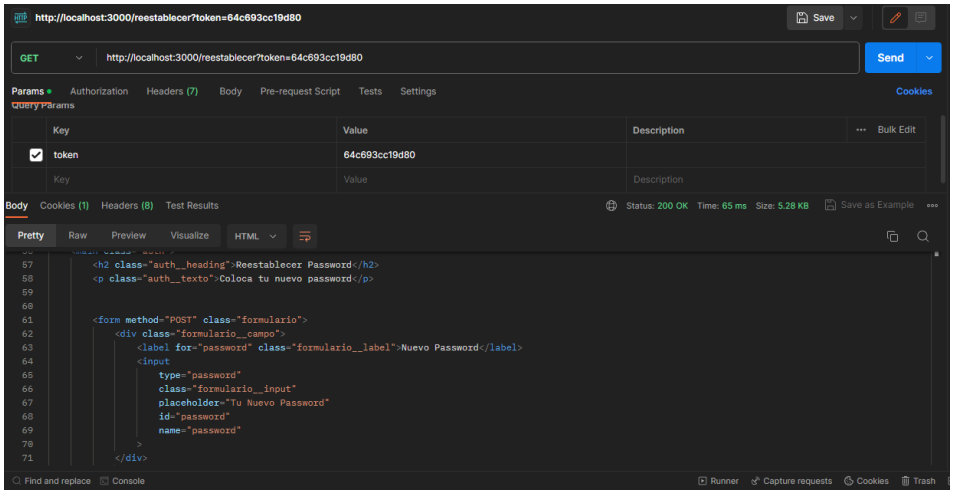

Figura 3.32 Prueba Unitaria Método GET de la ruta /reestablecer

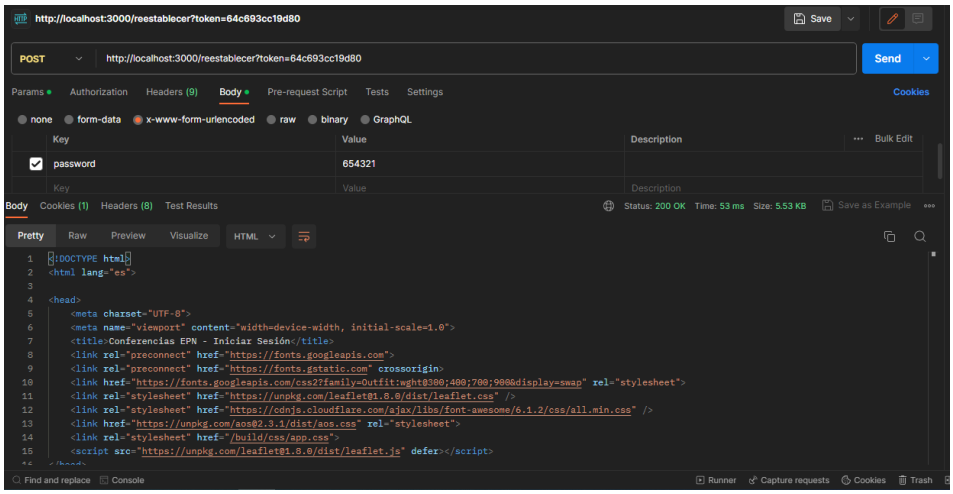

Figura 3.33 Prueba Unitaria Método POST de la ruta /reestablecer

Una vez realizadas estas pruebas se ha llegado a la conclusión de que el *backend* cumple con los requerimientos, que el código sigue la lógica del negocio y que los *endpoints* están listo para su publicación y consumo.

Pruebas de carga

Una vez culminado el desarrollo se han realizado las respectivas pruebas de carga, para demostrar el comportamiento del código cuando se encuentra bajo una gran demanda de transacciones. Estas prueban han permitido analizar el sistema sometiéndolo a una carga significativa de peticiones para determinar su estabilidad. La Figura 3.34 muestra el

resultado de la prueba realizada con 50 hilos (peticiones). La evidencia de esta prueba se presenta en el

ANEXO II en el apartado de **Pruebas**.

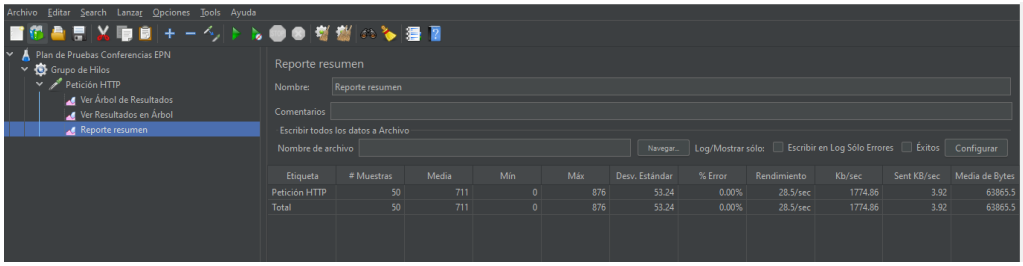

Figura 3.34 Reporte de la prueba en JMeter

Al culminar las pruebas y esperando que la página tenga una media de 50 visitantes que podrían realizar peticiones al mismo tiempo (de acuerdo con la lógica del negocio), se ha llegado a la conclusión de que el sistema soporta la carga esperada para ese número. Lo que permite una salida a producción sin ningún problema.

Pruebas de rendimiento

Se han realizado las pruebas de rendimiento para conocer la velocidad y eficiencia del código a ser desplegado. A continuación, la **Figura 3.35** muestra los resultados de las pruebas de rendimiento realizadas para comprobar los tiempos de respuesta y el manejo de la carga de trabajo. Cabe mencionar que para este tipo de pruebas se ha considerado mostrar un pequeño *Frontend* que consuma de manera eficiente los *endpoints* que se ha implementado como parte de este proyecto de titulación.

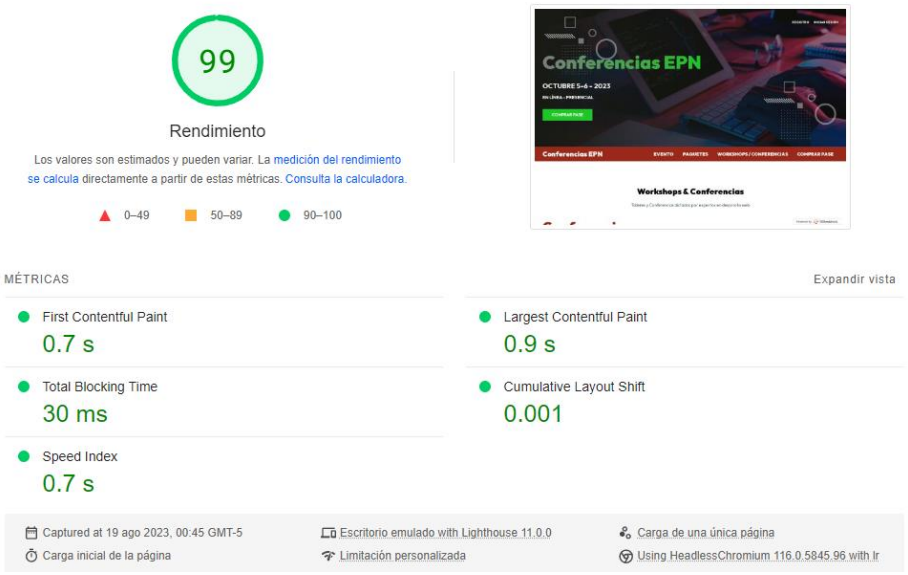

Figura 3.35 Reporte de la prueba de rendimiento

Al termino de las pruebas de rendimiento se ha verificado que cumple con los estándares pedidos en el levantamiento de requerimientos, que las herramientas usadas hacen que el código cumpla con la visión y eficiencia que requiere el modelo de negocio. Por lo tanto, el paso a producción ha sido considerado como punto final a realizarse.

Despliegue de *endpoints* a 000WebHost

Para el despliegue de los *endpoints* se ha utilizado la plataforma de 000WebHost misma que brinda el servicio de alojamiento web gratuito con ciertas restricciones. La razón por la cual se ha escogido esta plataforma es por su compatibilidad con el lenguaje de programación y la base de datos escogida para este proyecto.

En la **Figura 3.36** se presentan los *endpoints* desplegados en la plataforma.

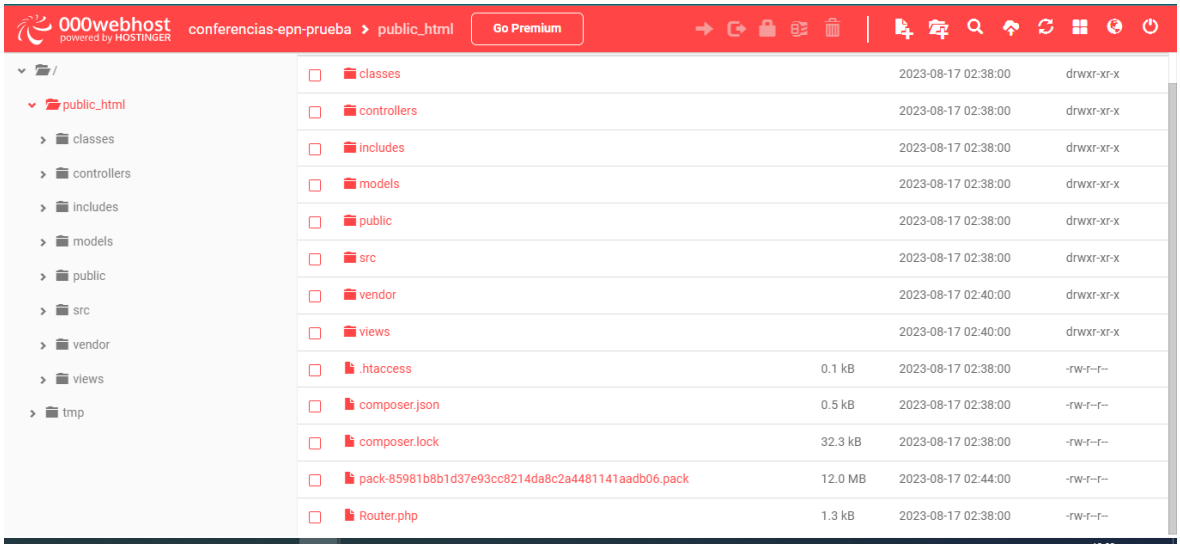

Figura 3.36 Despliegue de *endpoints* en 000WebHost

4 CONCLUSIONES

En este punto se presentan las conclusiones que se han podido sacar del presente proyecto de integración curricular:

- Los *endpoints* que han sido desarrollados en el presente proyecto cumplen con la funcionalidad para la que han sido creados, permitiendo la gestión dentro del sistema web, y que la información existente pueda ser respaldada dentro de una base de datos SQL.
- El uso de las metodologías ágiles como SCRUM permiten el desarrollo de proyectos en intervalos de tiempo más cortos. Permitiendo a los desarrolladores en caso de fallas poder regresar a un sprint anterior con la menor pérdida de tiempo y recursos dentro del proyecto.
- El desarrollar un proyecto mediante el paradigma de la programación orientada a objetos, permite llevar una mejor organización a nivel de código haciendo que se convierta en un framework que pueda ser usado para propósitos futuros.
- El patrón Modelo-Vista-Controlador ha demostrado ser una herramienta valiosa para el desarrollo de aplicaciones robustas, mantenibles y escalables, y su adopción sigue siendo ampliamente recomendada en el desarrollo de software.
- Los sistemas de bases de datos relacionales garantizan la integridad de los datos mediante la imposición de restricciones y reglas en la definición de las tablas. Esto ayuda a mantener la consistencia y la precisión de la información almacenada.
- La principal función de las pruebas es identificar y corregir errores, fallos o bugs en el software. Esto ayuda a mejorar la estabilidad y robustez del producto final.

5 RECOMENDACIONES

A continuación, se relatarán las recomendaciones que se pueden dar a partir del desarrollo del presente trabajo de integración curricular:

- Realizar evaluaciones periódicas acerca de la experiencia del usuario para poder mejorar la funcionalidad a partir de la experiencia que se tenga con el manejo de la plataforma.
- Realizar backups de manera regular que permitan recuperar la información de la base de datos y tenerla respaldada para poder así aumentar la disponibilidad en caso de fallos.
- Establecer un proceso para manejar cambios y solicitudes adicionales hacia el equipo de desarrollo en caso de requerirlo.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] C. Alvino, «branch,» 05 Mayo 2021. [En línea]. Available: https://branch.com.co/marketing-digital/estadisticas-de-la-situacion-digital-de-ecuador-en-el-2020-2021/#:~:text=El%20n%C3%BAmero%20de%20dispositivos%20m%C3%B3viles,%2C8%25%20de%20la%20poblaci%C3%B3n. [Último acceso: 21 Mayo 2023].

[2] S. G. Sotomayor, «IEBS,» 09 Diciembre 2021. [En línea]. Available: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/#:~:text=Por%20definici%C3%B3n%2C%20las%20metodolog%C3%ADas%20%C3%A1giles,las%20circunstancias%20espec%C3%ADficas%20del%20entorno.. [Último acceso: 24 05 2023].

[3] M. Á. Álvarez, «desarrolloweb.com,» 07 Diciembre 2022. [En línea]. Available: https://desarrolloweb.com/articulos/procesos-desarrollo-software. [Último acceso: 24 Mayo 2023].

[4] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[5] J. Nieto Rodrigo, «riunet,» 2016. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/75120/NIETO%20-%20Desarrollo%20de%20una%20aplicaci%C3%B3n%20web%2C%20con%20Front-end%20y%20Back-end%20para%20compraventa%20de%20segunda%20mano.pdf?sequence=1. [Último acceso: 31 Mayo 2023].

[6] Y. Fernández, «Xataka,» 23 Agosto 2019. [En línea]. Available: https://www.xataka.com/basics/api-que-sirve. [Último acceso: 12 Junio 2023].

[7] I. d. Souza, «rockcontent,» 17 Marzo 2020. [En línea]. Available: https://rockcontent.com/es/blog/api-rest/. [Último acceso: 12 Junio 2023].

[8] J. A. Fisteus, «Universidad Carlos III de Madrid,» 2022. [En línea]. Available: http://www.it.uc3m.es/jaf/aw/transparencias/http.pdf. [Último acceso: 31 Mayo 2023].

[9] F. J. Murillo, «academia.edu,» 2013. [En línea]. Available: https://d1wqtxts1xzle7.cloudfront.net/32017666/Estudio_de_Casos_Trabajo-libre.pdf?1391461259=&response-content-disposition=inline%3B+filename%3DEstudio_de_casos.pdf&Expires=1685573445&Signature=Lfr8lTpY4AkJmXS7Aiys-vmfEX9-GsfmxD~UOII9ujqb~n8NTUQt6HpxUEcYG. [Último acceso: 31 Mayo 2023].

[10] H. Kniberg, scrum y xp desde las trincheras, Estados Unidos: InfoQ, 2007.

[11] S. E. M. B. P. P. Ariel Pasini, Q-Scrum: una fusión de Scrum y el estándar ISO/IEC, Buenos Aires: RedUNCI, 2013.

[12] T. Satpathy, Una Guía para el cuerpo de conocimiento de scrum, Arizona: SCRUMstudy, 2016.

[13] A. Flores, «crehana,» Future of People, 21 Octubre 2021. [En línea]. Available: https://www.crehana.com/blog/negocios/roles-de-scrum/. [Último acceso: 01 Junio 2023].

[14] D. West, «Atlassian,» Atlassian, 2022. [En línea]. Available: https://www.atlassian.com/es/agile/scrum/roles. [Último acceso: 01 Junio 2023].

[15] G. B. C. L. B. V. Pete Deemer, Información Básica de Scrum, Agile: Scrum Training Institute, 2009.

[16] G. L. J. P. M. Á. S. Alexander Menzinsky, Historias de Usuario, Ingeniería de requisitos ágil, 2022.

[17] D. Molina, «IEBS,» IEBS, 09 Diciembre 2021. [En línea]. Available: https://www.iebschool.com/blog/que-es-un-product-backlog-y-como-hacer-uno-guia-scrum-agile-scrum/. [Último acceso: 02 Junio 2023].

[18] J. S. Ken Schwaber, La Guía de Scrum, Creative Commons, 2020.

[19] P. Huet, «OpenWebinars,» 24 Agosto 2022. [En línea]. Available: https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/. [Último acceso: 01 Julio 2023].

[20] R. D. Hernández, «freeCodeCamp,» 28 Junio 2021. [En línea]. Available: https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/. [Último acceso: 04 Julio 2023].

[21] Yair, «Styde,» 23 Diciembre 2019. [En línea]. Available: https://styde.net/que-es-composer-y-como-usarlo/. [Último acceso: 04 Julio 2023].

[22] I. d. Souza, «rockcontent,» 09 Septiembre 2020. [En línea]. Available: https://rockcontent.com/es/blog/php/. [Último acceso: 04 Julio 2023].

[23] P. Londoño, «Hubspot,» 19 Enero 2023. [En línea]. Available: https://blog.hubspot.es/website/que-es-mysql. [Último acceso: 04 Julio 2023].

[24] M. A. Alvarez, «desarrolloweb6.vom,» 27 Septiembre 2022. [En línea]. Available: https://desarrolloweb.com/articulos/dbeaver. [Último acceso: 04 Julio 2023].

[25] M. A. Alvarez, «desarrolloweb.com,» 11 Mayo 2023. [En línea]. Available: https://desarrolloweb.com/articulos/phpmailer.html. [Último acceso: 04 Julio 2023].

7 ANEXOS

En este apartado se listan los diferentes Anexos que se han utilizado a lo largo del presente proyecto.

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

ANEXO I

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 23 de agosto de 2023

De mi consideración:

Yo, IVONNE FERNANDA MALDONADO SOLIZ, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DE UN BACKEND asociado al SISTEMA WEB PARA REGISTRO Y PAGO EN LÍNEA DE CONFERENCIAS elaborado por el estudiante JONATHAN ALEJANDRO ARMAS COLLAHUAZO de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones (sin anexos), producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

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

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

Atentamente,

Firmado electrónicamente por:
IVONNE FERNANDA MALDONADO SOLIS

Ivonne Maldonado
Docente Ocasional a Tiempo Completo
ESFOT

ANEXO II

En este anexo se presenta toda la evidencia del trabajo que se ha realizado para el cumplimiento de los objetivos de proyecto de titulación.

Recopilación de requerimientos

En la **Tabla 1** se presentan los requerimientos que se han logrado obtener a partir de las solicitudes presentadas por el *Product Owner.*

Tabla 1: Recopilación de Requerimientos

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DE SISTEMA	**ID RR**	**ENUNCIADO DEL ITEM**
Backend	**RR01**	Como usuario, necesita generar *endpoints* el registro de usuarios.
	RR02	Como usuario, necesita generar *endpoints* para iniciar y cerrar sesión.
	RR03	Como usuario, necesita generar *endpoints* para que por medio de un método de verificación de email se pueda activar la cuenta.
	RR04	Como usuario, necesita generar *endpoints* para reestablecer la contraseña de la cuenta.
	RR05	Como usuario administrador, necesita generar *endpoints* para gestionar (CRUD) ponentes.
	RR06	Como usuario administrador, necesita generar *endpoints* para gestionar (CRUD) eventos.
	RR07	Como usuario administrador, necesita generar *endpoints* para el registro de conferencias.
	RR08	Como usuario administrador necesita generar *endpoints* para los pagos en línea dentro de la plataforma.
	RR09	Como usuario administrador, necesita generar *endpoints* para la información de asistentes y pagos realizados.

Historias de Usuario

Una vez finalizada la etapa de recopilación de requerimientos, se procede a realizar las respectivas Historias de Usuario donde se detalla al equipo de manera sencilla la funcionalidad esperada. A continuación, desde la **Tabla 2** a la **Tabla 8** se presentan las historias de usuario en base a los requerimientos establecidos.

Tabla 2: Historia de usuario 02- Generar *endpoints* para el inicio y cierre de sesión

Historia de Usuario	
Identificador: HU02	**Usuario:** Usuario/Administrador
Nombre Historia: Generar *endpoints* para el inicio y cierre de sesión	
Prioridad en Negocio (Alta/Medio/Baja): Alta	**Riesgo en Desarrollo (Alta/Medio/Baja):** Baja
Iteración asignada: 1	
Responsable: Alejandro Armas	
Descripción: Se necesita crear *endpoints* para que los usuarios puedan iniciar y cerrar sesión dentro de la aplicación.	
Observación: • Se debe diferenciar los usuarios de los usuarios administradores, para poder presentar las funcionalidades de cada uno en la plataforma. • Se debe validar el email de acceso y la contraseña en la base de datos para poder iniciar sesión.	

Tabla 3: Historia de usuario 03-Generar *endpoints* para verificación de email

Historia de Usuario	
Identificador: HU03	**Usuario:** Usuario
Nombre Historia: Generar *endpoints* para verificación de email	
Prioridad en Negocio (Alta/Medio/Baja): Media	**Riesgo en Desarrollo (Alta/Medio/Baja):** Baja
Iteración asignada: 1	
Responsable: Alejandro Armas	
Descripción: Se necesita generar *endpoints* para verificar el mail de los usuarios al momento de registrarse en la plataforma.	
Observación: Ninguna	

Tabla 4: Historia de usuario 04-Generar *endpoints* para la recuperación de cuenta

Historia de Usuario	
Identificador: HU04	**Usuario:** Usuario/Administrador
Nombre Historia: Generar *endpoints* para la recuperación de cuenta.	
Prioridad en Negocio (Alta/Medio/Baja): Media	**Riesgo en Desarrollo (Alta/Medio/Baja):** Baja
Iteración asignada: 1	
Responsable: Alejandro Armas	
Descripción: Se debe generar *endpoints* para que en caso de que el usuario olvide la contraseña de su cuenta pueda volver a recuperarla mediante el email que usa en la plataforma.	
Observación: • Se debe verificar que el email que se esté usando para iniciar sesión esté registrado en la plataforma mediante la base de datos.	

Tabla 5: Historia de usuario 05-Generar *endpoints* para la información de los ponentes

Historia de Usuario	
Identificador: HU05	**Usuario:** Administrador
Nombre Historia: Generar *endpoints* para la información de los ponentes.	
Prioridad en Negocio (Alta/Medio/Baja): Media	**Riesgo en Desarrollo (Alta/Medio/Baja):** Media
Iteración asignada: 2	
Responsable: Alejandro Armas	
Descripción: Se necesita crear *endpoints,* que permitan enviar la información de cada ponente hacia la base de datos para poder almacenarla.	
Observación: • Se debe de tomar en cuenta las validaciones del *frontend* para poder crear los campos dentro de la base de datos. • Se debe crear un modelo que permita enviar los datos para su almacenamiento y se puedan consultar los mismos desde el *frontend.* • Se debe crear un método de almacenamiento de imágenes para usar la fotografía del ponente en caso de requerirlo.	

Tabla 6: Historia de usuario 06-Generar *endpoints* para la información de eventos

Historia de Usuario	
Identificador: HU06	**Usuario:** Administrador
Nombre Historia: Generar *endpoints* para el registro de eventos.	
Prioridad en Negocio (Alta/Medio/Baja): Alta	**Riesgo en Desarrollo (Alta/Medio/Baja):** Media
Iteración asignada: 2	
Responsable: Alejandro Armas	
Descripción: Se debe crear *endpoints,* que permitan enviar la información de cada evento hacia la base de datos para poder almacenarlos.	
Observación: • Se debe tomar en cuenta las validaciones del *frontend* para poder crear los campos en la base de datos. • Se debe crear un modelo que permita enviar los datos para su almacenamiento y se puedan consultar los mismos desde el *frontend.*	

Tabla 7: Historia de usuario 07-Generar varios *endpoints* para el registro de usuarios

Historia de Usuario	
Identificador: HU07	**Usuario:** Administrador
Nombre Historia: Generar *endpoints* para los pagos en línea dentro de la plataforma.	
Prioridad en Negocio (Alta/Medio/Baja): Media	**Riesgo en Desarrollo (Alta/Medio/Baja):** Media
Iteración asignada: 4	
Responsable: Alejandro Armas	
Descripción: Se debe crear *endpoints,* que permita a los usuarios realizar los pagos en la plataforma, dicha información se debe enviar hacia la base de datos.	
Observación: • Se debe crear un modelo que permita enviar los datos para su almacenamiento y se puedan consultar los mismos desde el *frontend.*	

Tabla 8: Historia de usuario 08-Generar *endpoints* para la consulta de información de asistentes y pagos realizados

Historia de Usuario	
Identificador: HU08	**Usuario:** Administrador
Nombre Historia: Generar *endpoints* para la consulta de información de asistentes y pagos realizados.	
Prioridad en Negocio (Alta/Medio/Baja): Baja	**Riesgo en Desarrollo (Alta/Medio/Baja):** Media
Iteración asignada: 4	
Responsable: Alejandro Armas	
Descripción: Se debe crear *endpoints,* que permitan consultar la información de los usuarios que se han registrado a una conferencia y de los pagos realizados en la plataforma.	
Observación: • Se debe crear un modelo que permita consultar los datos de la plataforma respecto a asistentes y pagos desde el *frontend.*	

Product Backlog

En la **Tabla 9** se detalla la prioridad de los requerimientos de acuerdo con las necesidades del *Product Owner* y la complejidad que presenta cada uno.

Tabla 9: *Product Backlog*

ID-PB	ID-HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
PB-001	HU01	Generar *endpoints* para el registro de usuarios	1	Finalizado	Alta
PB-002	HU02	Generar *endpoints* para el inicio y cierre de sesión	1	Finalizado	Alta
PB-003	HU03	Generar *endpoints* para verificación de email	1	Finalizado	Media
PB-004	HU04	Generar *endpoints* para la recuperación de cuenta.	1	Finalizado	Media
PB-005	HU05	Generar *endpoints* para la información de los ponentes	2	Finalizado	Media
PB-006	HU06	Generar *endpoints* para el registro de eventos	3	Finalizado	Alta
PB-007	HU07	Generar *endpoints* para los pagos en línea dentro de la plataforma	4	Finalizado	Medio
PB-008	HU08	Generar *endpoints* para la consulta de información de asistentes y pagos realizados	4	Finalizado	Baja

Sprint Backlog

En la **Tabla 10** se detalla los cinco *sprints* que se han llevado a cabo a lo largo del desarrollo del componente *backend*, en estos se listan las tareas que se han realizado con el fin de entregar el producto a tiempo.

Tabla 10: *Sprint Backlog*

ID - SB	NOMBRE	ID - HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB-000	Configuración del entorno de desarrollo	N/A	N/A	• Recopilación de Requerimientos • Elaboración del Modelo de Datos • Estructura del Proyecto	40H
SB-001	Módulo de Autenticación	HU01	Generar *endpoints* para el registro de usuarios	• Definir los campos para crear la tabla en la base de datos • Diseñar el modelo para el CRUD dentro de la base de datos • Implementar los respectivos *endpoints*	50 H
		HU02	Generar *endpoints* para el inicio y cierre de sesión	• Diseñar el modelo para el inicio y cierre de sesión de los usuarios. • Diseñar el modelo de validación para el inicio de sesión. • Implementar autenticación mediante correo electrónico y contraseña. • Validación de que el usuario se encuentra registrado en la plataforma	

		HU03	Generar *endpoints* para verificación de email	• Diseñar el modelo para verificar que el email con el que se registra el usuario es real. • Crear el modelo para activar la cuenta una vez el usuario haya verificado su cuenta de email.	
		HU04	Generar *endpoints* para la recuperación de cuenta	• Verificar que la cuenta que se está recuperando exista en la base de datos. • Crear el modelo para actualizar la nueva contraseña dentro de la base de datos.	
SB-002	Módulo de Ponentes	HU05	Generar *endpoints* para la información de ponentes	• Definir los campos para crear la tabla en la base de datos • Diseñar el modelo para el CRUD dentro de la base de datos. • Implementar los respectivos *endpoints.*	40H
SB-003	Módulo de Eventos	HU06	Generar *endpoints* para el registro de eventos	• Definir los campos para crear la tabla en la base de datos • Diseñar el modelo para el CRUD dentro de la base de datos • Implementar los respectivos *endpoints.*	
SB-004	Módulo de Pagos	HU07	Generar *endpoints* para los pagos en línea dentro del plataforma	• Definir los campos para crear la tabla en la base de datos • Diseñar el modelo para el CRUD dentro de la base de datos • Implementar los respectivos *endpoints*.	40H

		HU08	Generar *endpoints* para la consulta de información de asistentes y pagos realizados	• Diseñar el modelo para la consulta dentro de la base de datos	
SB-005	Pruebas de *endpoints* y despliegue en 000WebHost	• Pruebas unitarias • Pruebas de carga • Pruebas de rendimiento • Despliegue de *endpoints* a 000WebHost			40H
DOCUMENTACIÓN		• Informe Técnico • Anexos			30H
TOTAL					240H

Pruebas

A continuación, se muestra el resultado de las pruebas que se han llevado a cabo.

Pruebas Unitarias

Las pruebas unitarias se presentan desde la **Figura 1** hasta la **Figura 23**.

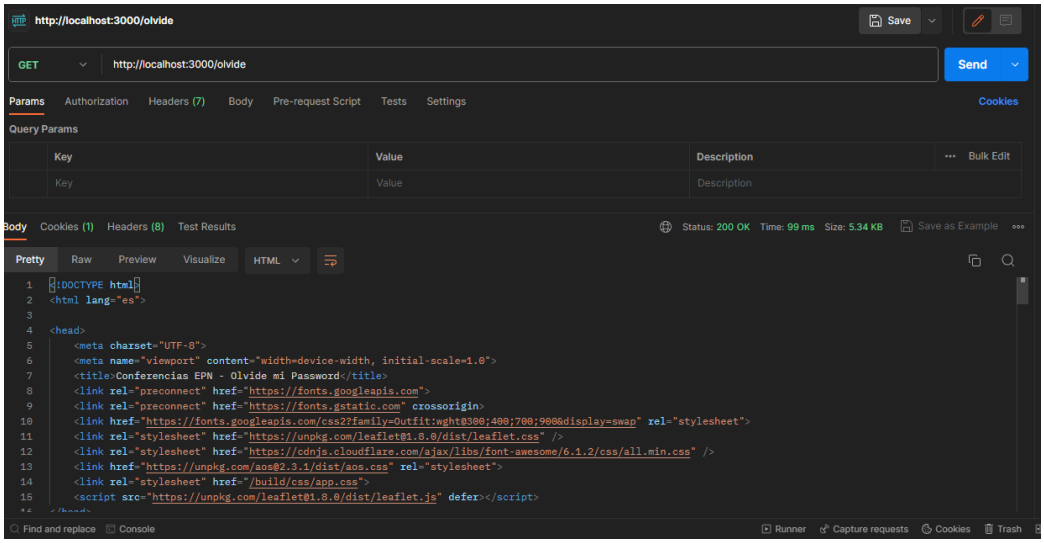

Figura 1 Prueba Unitaria Método GET de la ruta /olvide

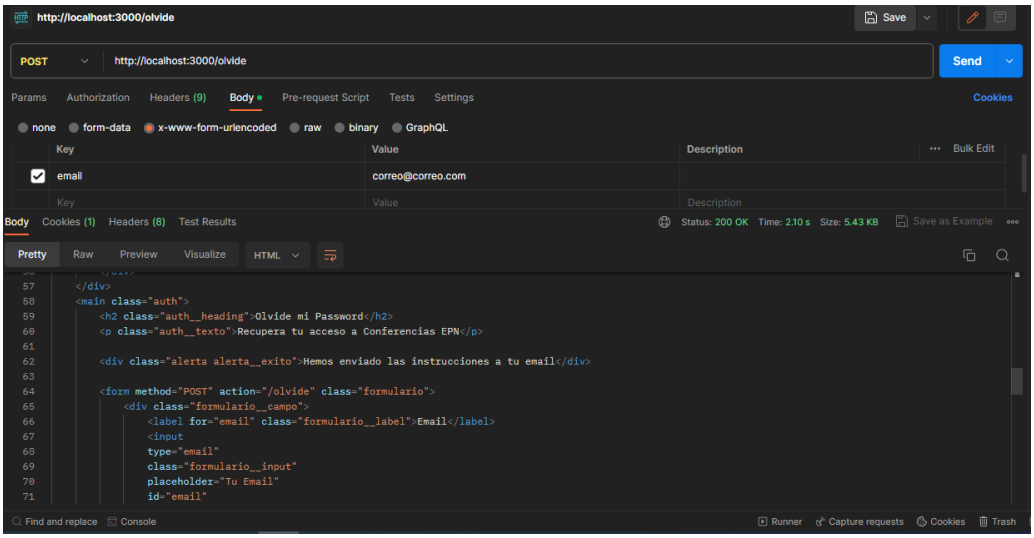

Figura 2 Prueba Unitaria Método POST de la ruta /olvide

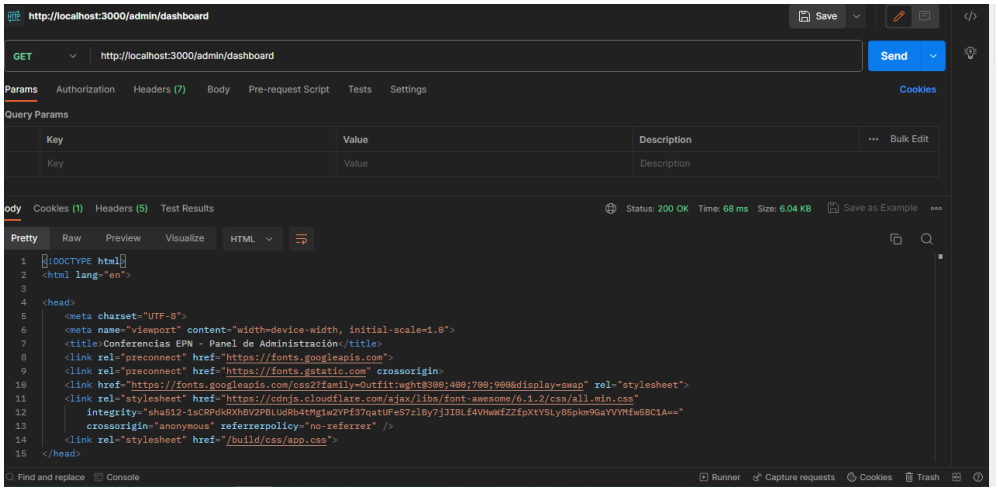

Figura 3 Prueba Unitaria Método GET de la ruta /admin/dashboard

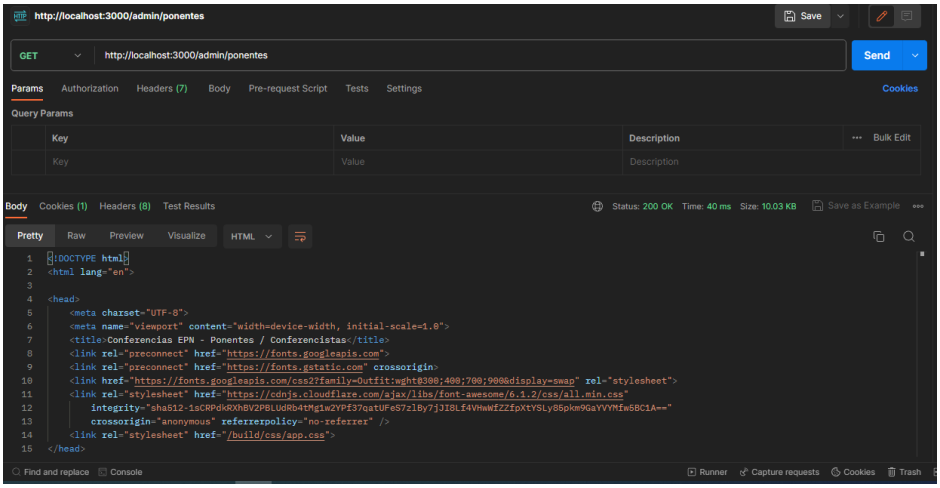

Figura 4 Prueba Unitaria Método GET de la ruta /admin/ponentes

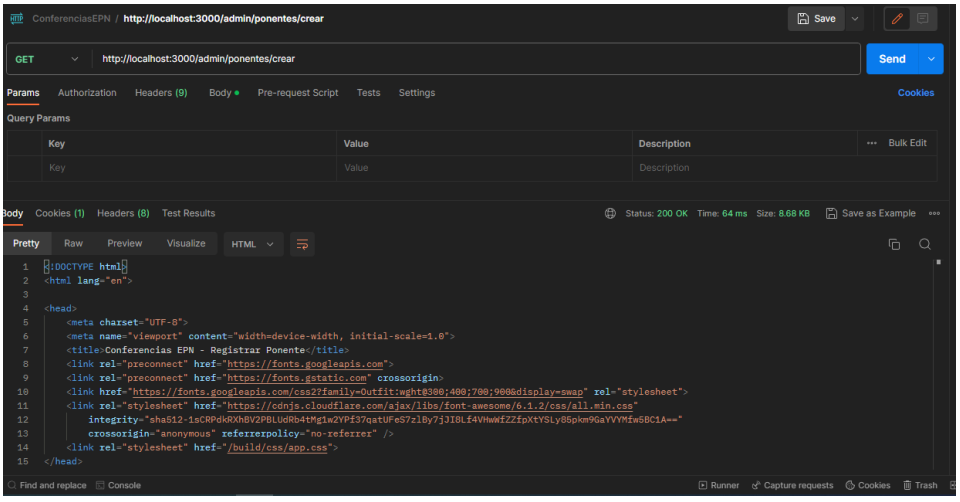

Figura 5 Prueba Unitaria Método GET de la ruta /admin/ponentes/crear

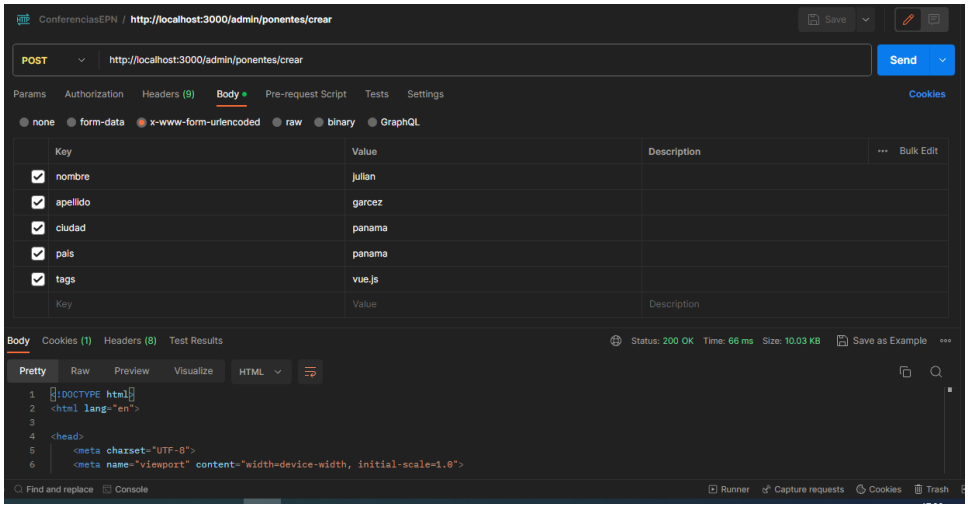

Figura 6 Prueba Unitaria Método POST de la ruta /admin/ponentes/crear

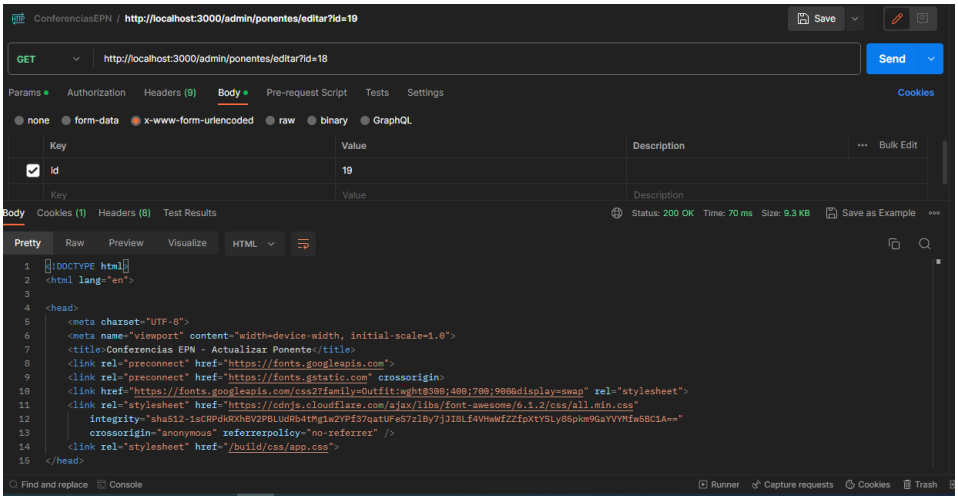

Figura 7 Prueba Unitaria Método GET de la ruta /admin/ponentes/editar

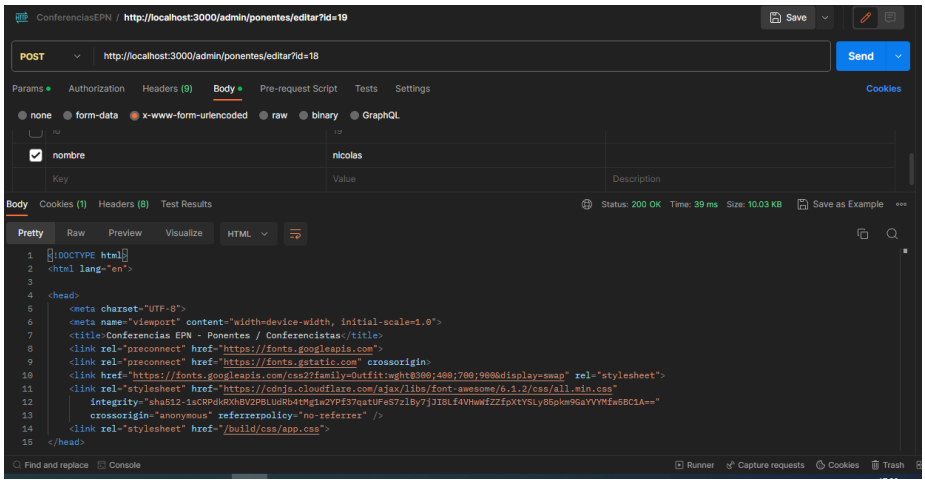

Figura 8 Prueba Unitaria Método POST de la ruta /admin/ponentes/editar

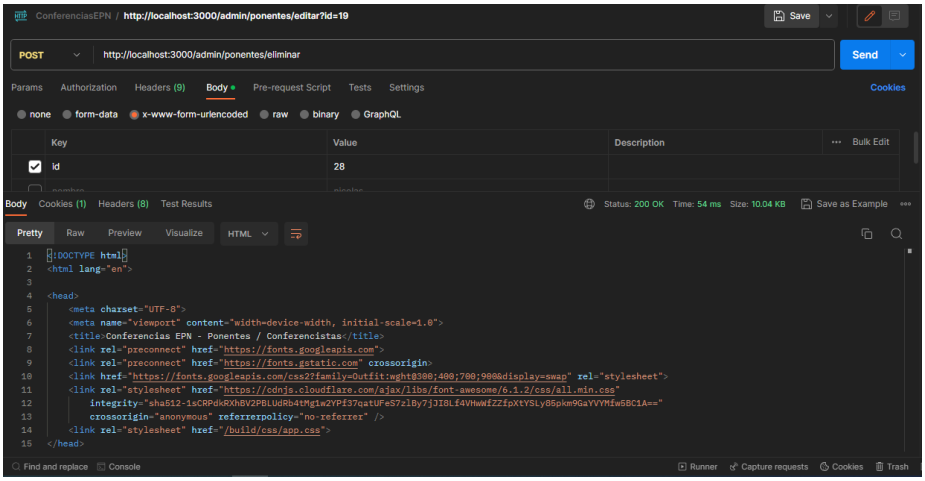

Figura 9 Prueba Unitaria Método POST de la ruta /admin/ponentes/eliminar

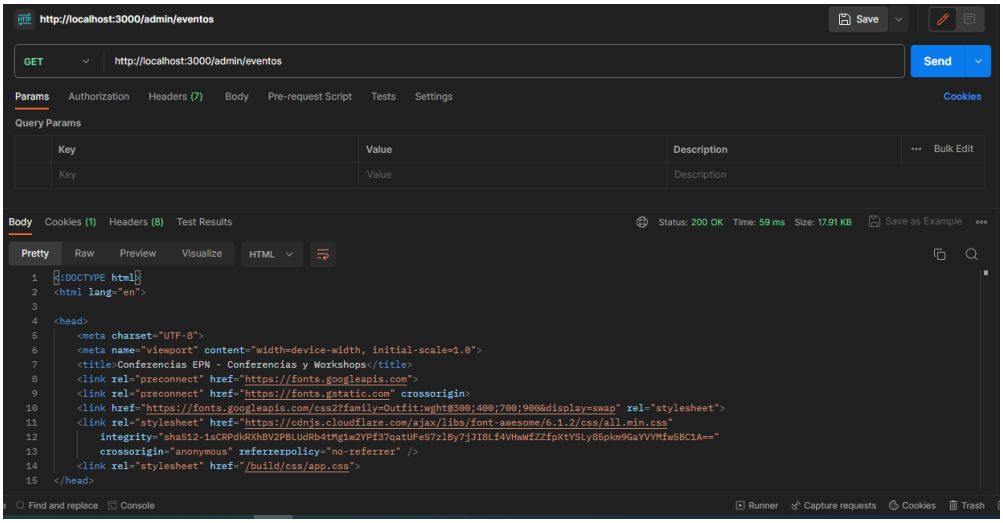

Figura 10 Prueba Unitaria Método GET de la ruta /admin/eventos

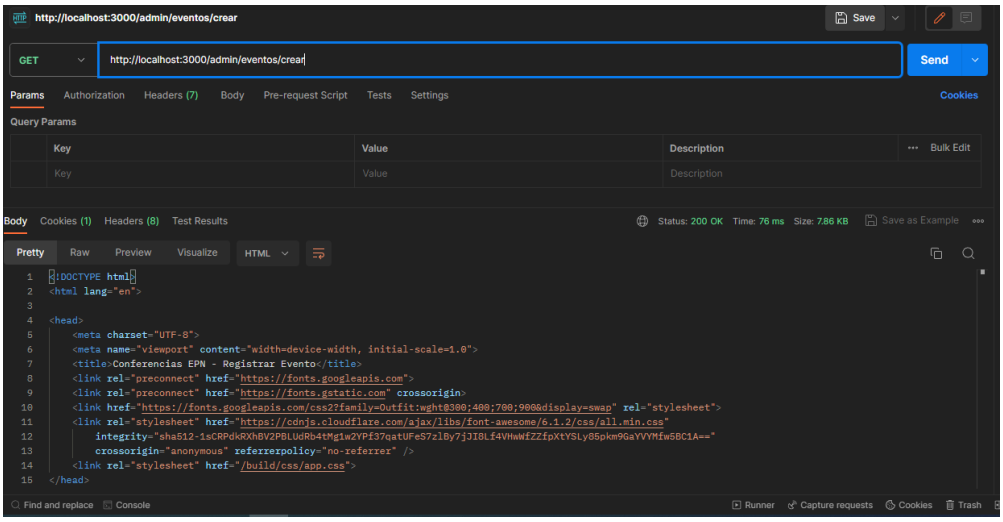

Figura 11 Prueba Unitaria Método GET de la ruta /admin/eventos/crear

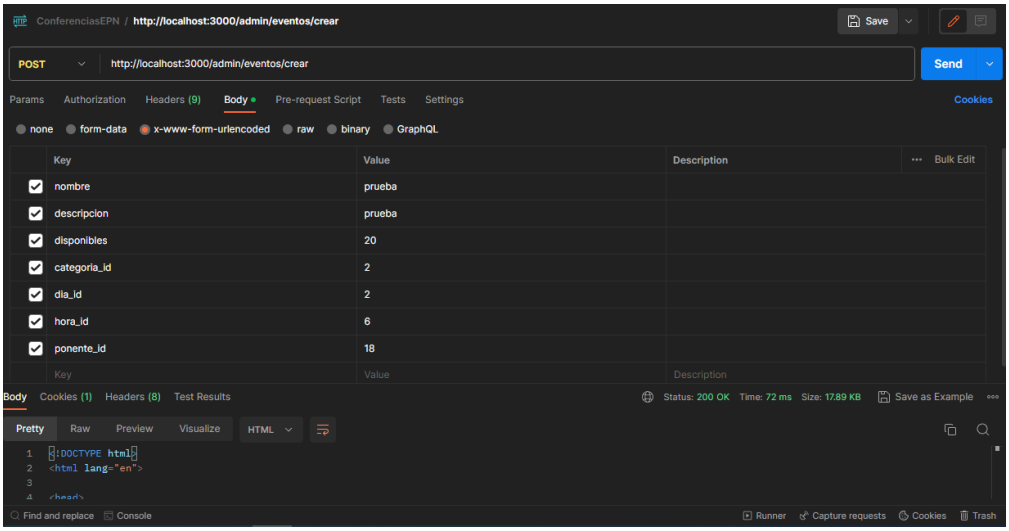

Figura 12 Prueba Unitaria Método POST de la ruta /admin/eventos/crear

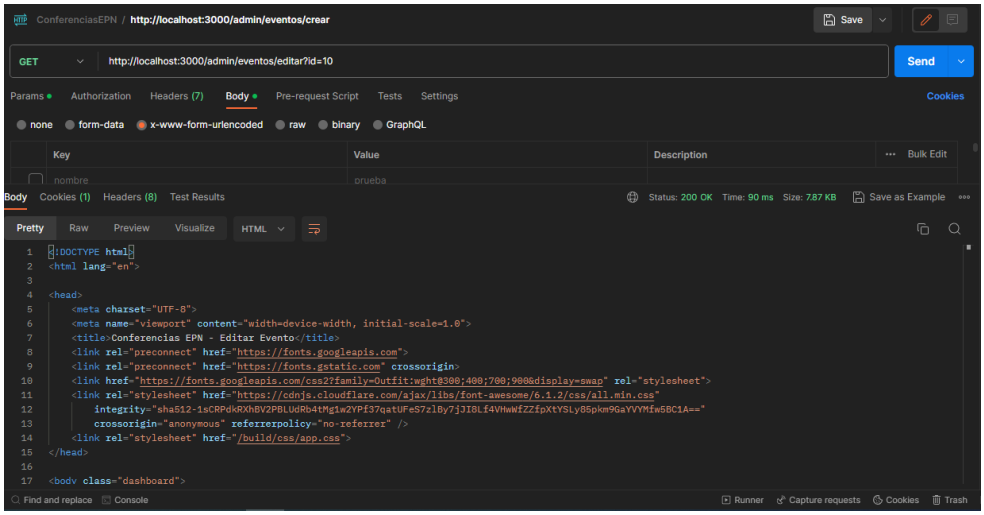

Figura 13 Prueba Unitaria Método GET de la ruta /admin/eventos/editar

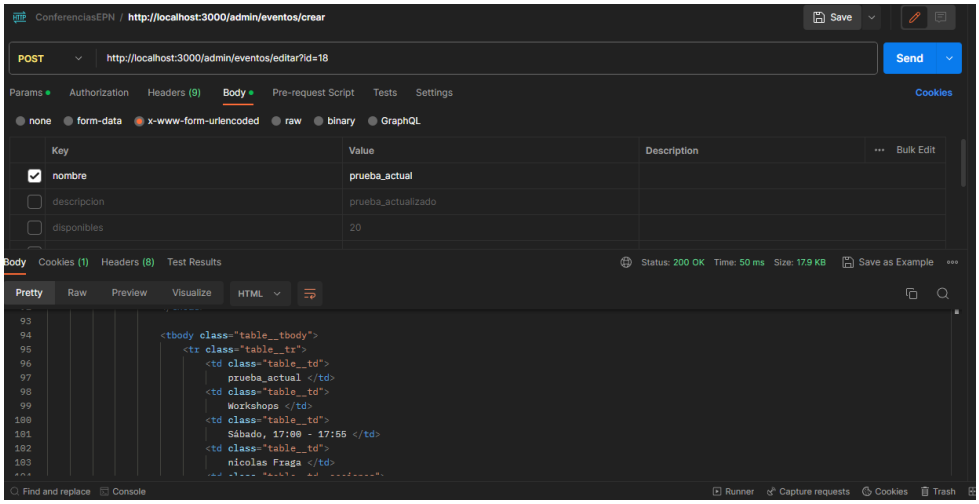

Figura 14 Prueba Unitaria Método POST de la ruta /admin/eventos/editar

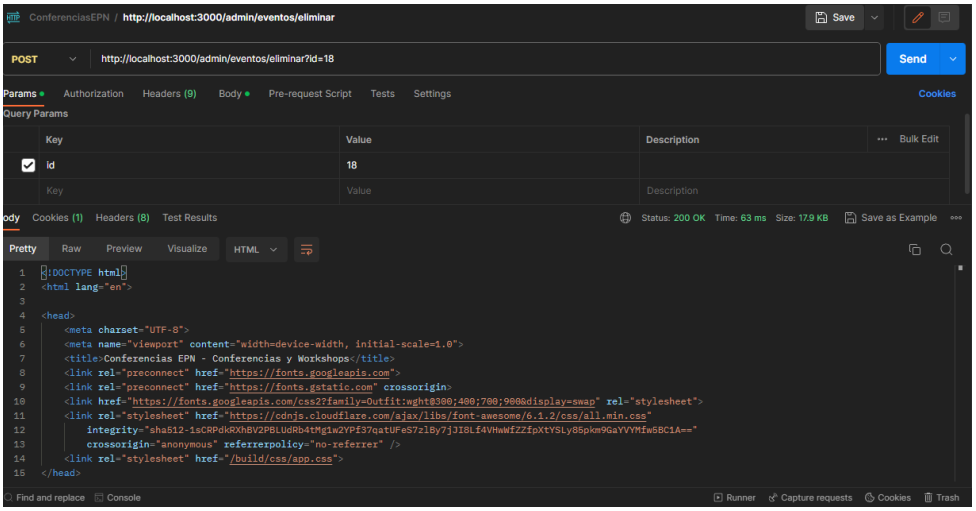

Figura 15 Prueba Unitaria Método POST de la ruta /admin/eventos/eliminar

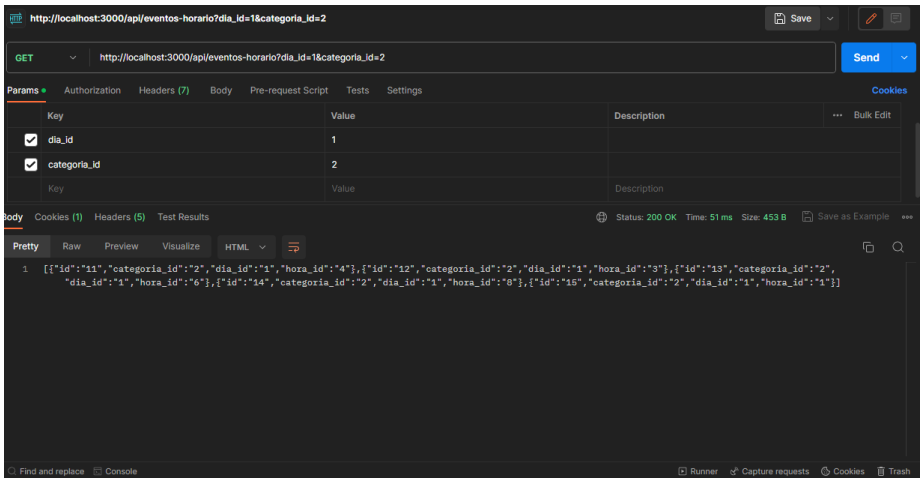

Figura 16 Prueba Unitaria Método GET de la ruta /api/eventos-horario

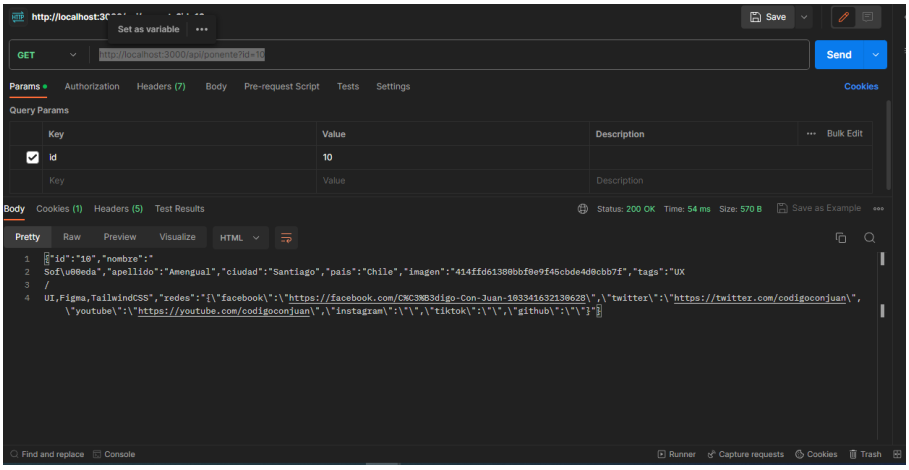

Figura 17 Prueba Unitaria Método GET de la ruta /api/ponentes

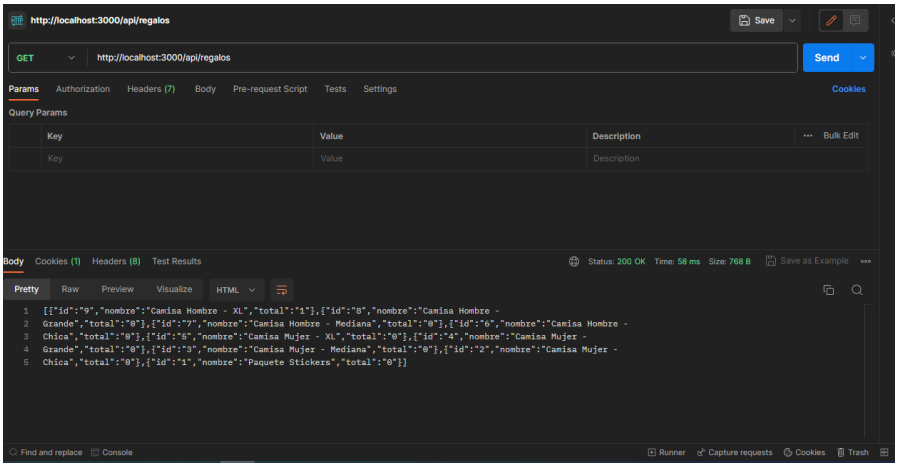

Figura 18 Prueba Unitaria Método GET de la ruta /api/regalos

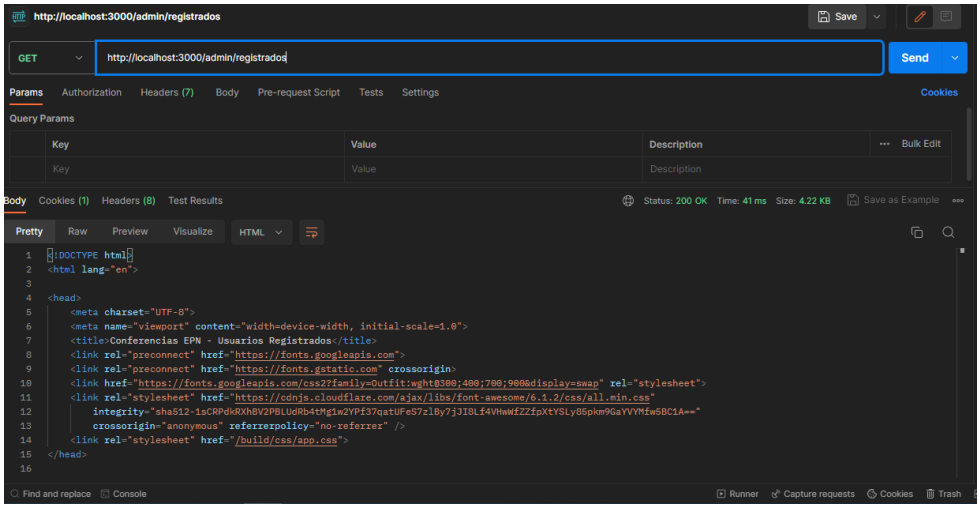

Figura 19 Prueba Unitaria Método GET de la ruta /admin/registrados

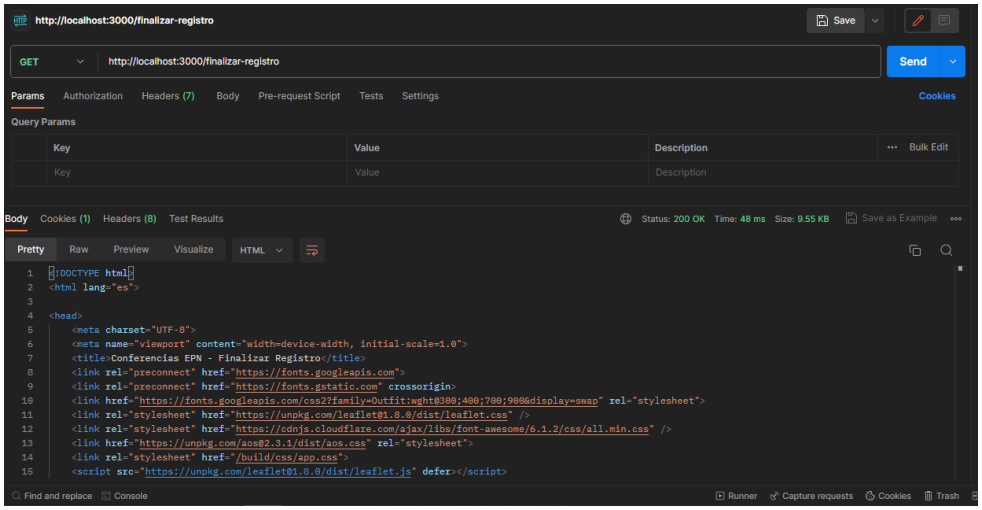

Figura 20 Prueba Unitaria Método GET de la ruta /finalizar-registro

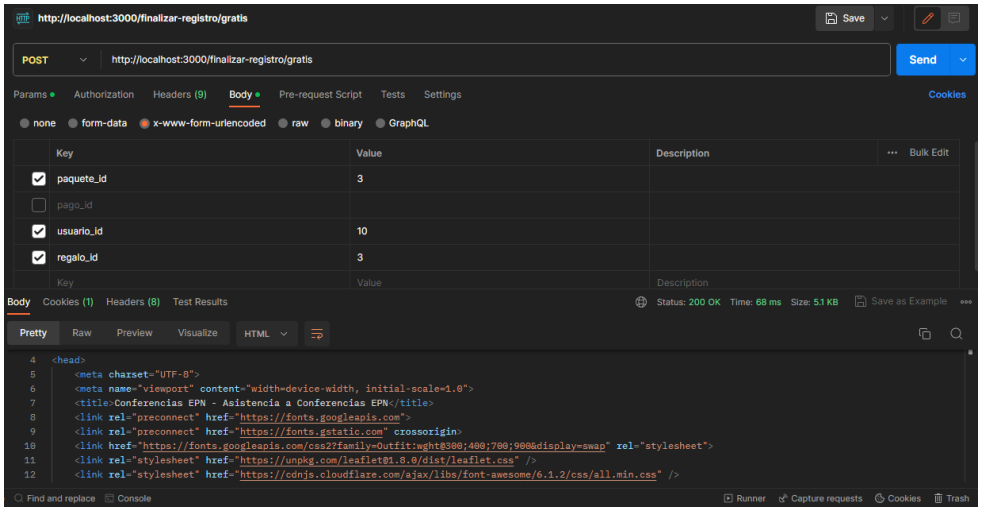

Figura 21 Prueba Unitaria Método POST de la ruta /finalizar-registro/gratis

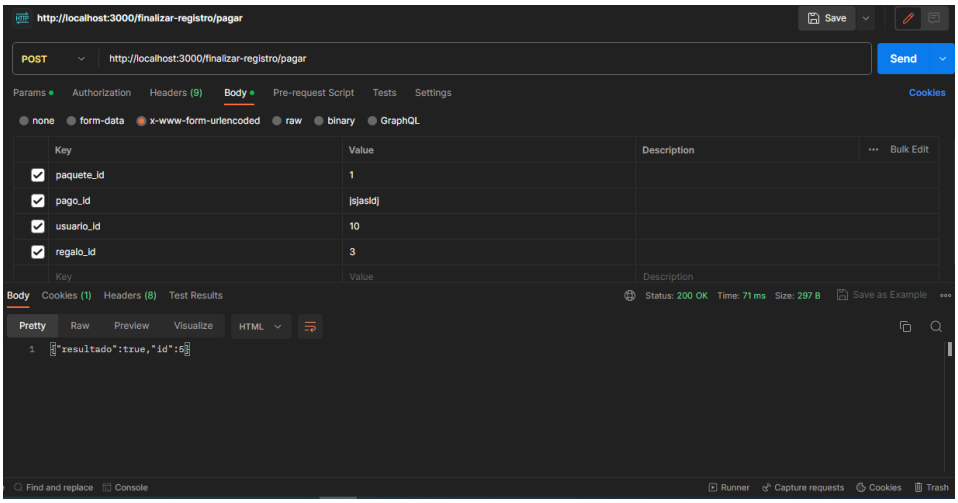

Figura 22 Prueba Unitaria Método POST de la ruta /finalizar-registro/pagar

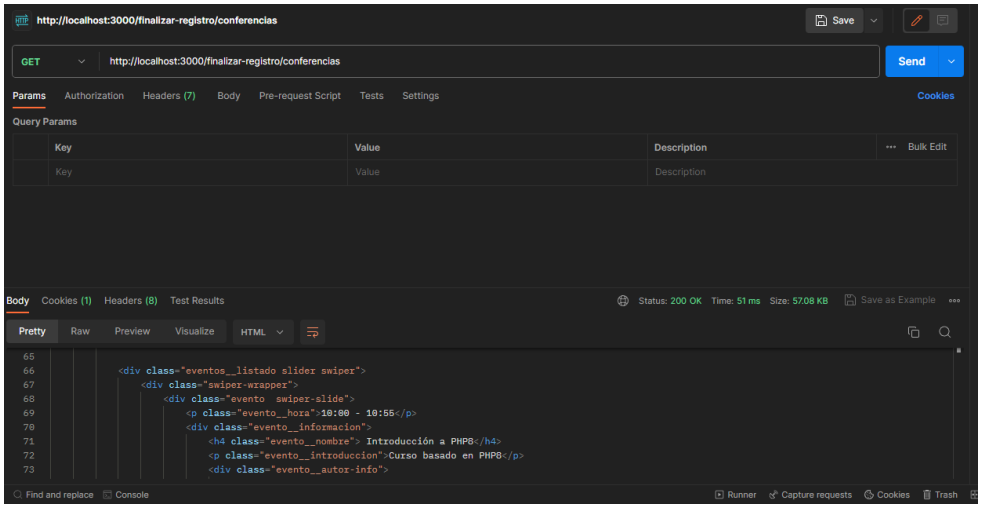

Figura 23 Prueba Unitaria Método GET de la ruta /finalizar-registro/conferencias

Pruebas de Carga

Estas pruebas se muestran desde la **Figura 24** hasta la **Figura 26.**

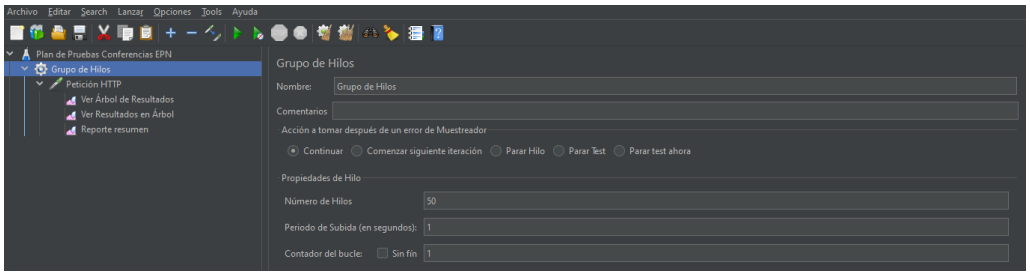

Figura 24 Preparación de la prueba de carga por hilos

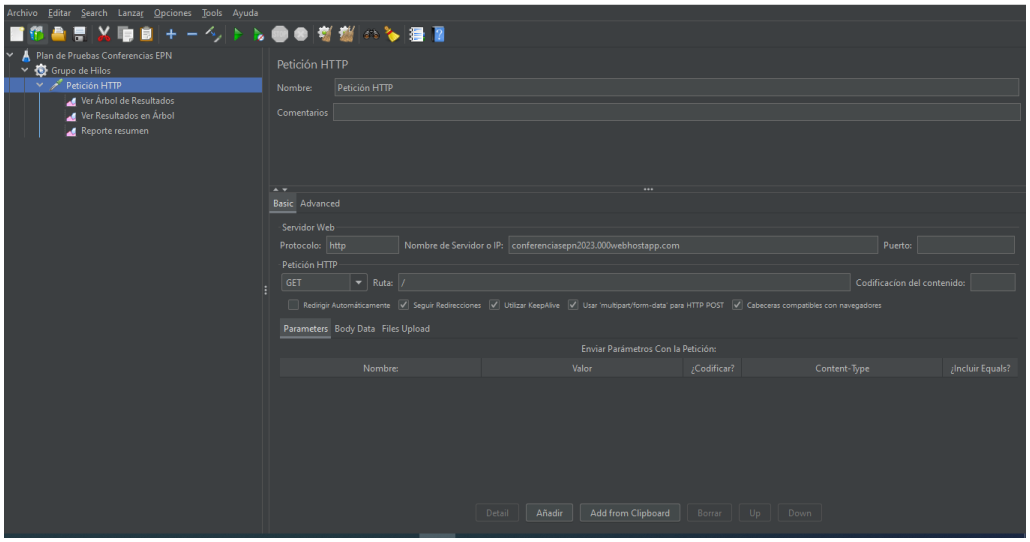

Figura 25 Información acerca de la página a realizar la prueba

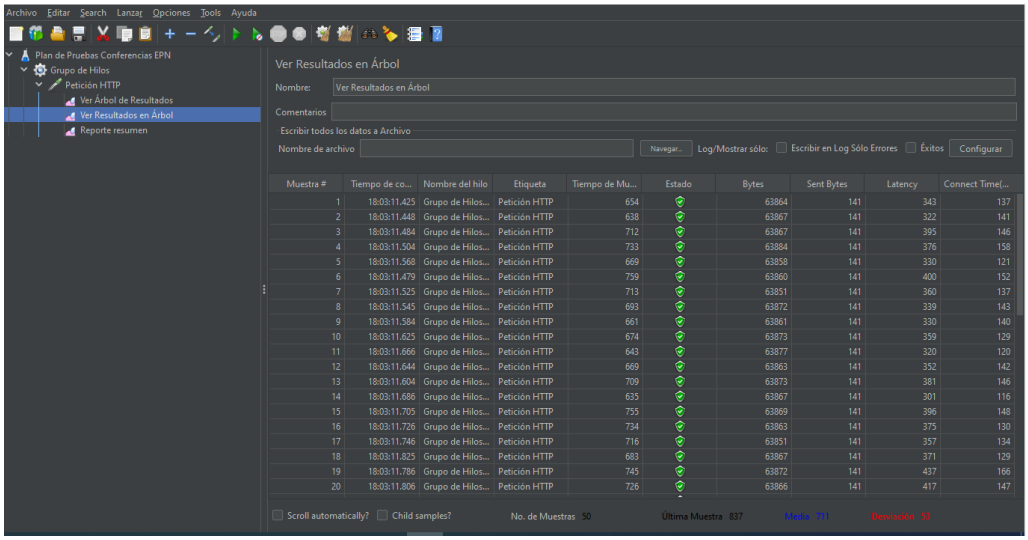

Figura 26 Respuesta del servidor a la prueba de JMeter

ANEXO III

El siguiente enlace, se detalla la información sobre el funcionamiento de los *endpoints,* así como la interacción y manejo de los datos por parte del *backend*:

https://youtu.be/yEiysCd6FwA

ANEXO IV

A continuación, se presentan las credenciales de acceso al backend, así como el acceso al código fuente en el repositorio de GitHub.

Credenciales de acceso para el *Backend*

El siguiente enlace permite evidenciar la aplicación en ambiente de producción.

https://app.swaggerhub.com/apis-docs/LJANDRO/ConferenciasEPN/1.0.0

Credenciales para el ingreso al sistema web con permisos de Administrador de ConferenciasEPN:

- Usuario: correo_admin@correo.com
- Password: 123456

Credenciales para el perfil de usuario:

- Usuario: correo@correo.com
- Password: 123456

Repositorio del código fuente del *Backend*

En el siguiente enlace se puede encontrar el repositorio de GitHub donde se encuentra almacenado el código fuente del proyecto:

https://github.com/alejandroarmas1992/conferenciasepn.git