ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE APLICACIÓN WEB Y MÓVIL PARA GESTIÓN DE RECETAS DE COCINA

COMPONENTE BACKEND

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

SINGAÑA VERGARA JOSUE ALEXANDER

DIRECTORA: ING. MAYRA ISABEL ALVAREZ JIMENEZ

DMQ, febrero 2023

CERTIFICACIONES

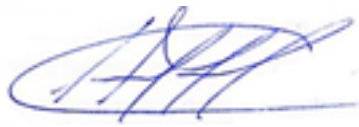
Yo, Josue Alexander Singaña Vergara declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Josue Alexander Singaña Vergara

Josue.Singana@epn.edu.ec

Josue.singtec@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Josue Alexander Singaña Vergara, bajo mi supervisión.



ING. Mayra Isabel Alvarez Jiménez MSc.

DIRECTORA

mayra.alvarez@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.

JOSUE ALEXANDER SINGAÑA VERGARA

DEDICATORIA

El presente proyecto es dedicado a mis padres, quienes desde pequeño me apoyaron y alentaron a seguir adelante con mis sueños. A mis abuelos quienes fueron parte importante en mis diferentes etapas educativas.

Finalmente, es dedicado a las personas que conocí a lo largo de los años y me ayudaron a mejorar en diferentes aspectos académicos y sociales.

JOSUE ALEXANDER SINGAÑA VERGARA

AGRADECIMIENTO

Agradezco principalmente a mis padres, que juntos me inculcaron valores como el respeto y la dedicación, los cuales formaron mi carácter. Agradezco a mi madre que siempre me cuido y alentó a no rendirme cuando tuve problemas, agradezco a mi padre que fue un pilar en mi educación al no limitarme nunca cuando tenía alguna oportunidad de estudio, también agradezco que fuera un ejemplo en el que pude basarme y poder entender que el único límite soy yo mismo.

A mis amigos, que nunca me abandonaron y compartimos momentos inigualables de felicidad y tristeza, además debo agradecer a una persona que recientemente formo parte de mi vida que junto a sus consejos y a sus pláticas me ayudo a concentrarme en el desarrollo del presente proyecto.

A mis hermanas que en los momentos difíciles nos supimos unir y ayudarnos a continuar con nuestras metas mientras afrontábamos las dificultades. Agradezco a mi mascota, que, en ningún momento, ya sea las noches de desvelo o salidas importantes siempre estuvo a mi lado acompañándome.

Finalmente, agradezco a los docentes quienes nos impartieron y ayudaron a comprender las diferentes materias que impartían, quienes nunca se negaron a corregirnos cuando nos equivocábamos en las tareas.

JOSUE ALEXANDER SINGAÑA VERGARA

ÍNDICE DE CONTENIDO

RESUMEN

La pandemia del COVID-19 puso en evidencia que una adecuada alimentación de las personas es de suma importancia para una buena salud, fundamentalmente en las personas con obesidad, sobrepeso, diabetes e hipertensión las cuales conforman gran parte del grupo de riesgo de contagio. Algunos de los factores que aumento la mala alimentación en las personas fueron el encierro y el bajo interés por la comida sana. Este mismo problema trae consigo enfermedades que afectan a la población en general.

Dado este antecedente se planteó el desarrollo de un recetario de cocina enfocado en la interacción entre usuarios el cual incluye un chat en vivo. El proyecto permite compartir recetas creadas por los usuarios, brindar retroalimentación de las mismas, realización de comentarios y calificación de recetas publicadas. Cabe mencionar que nos centramos específicamente en el desarrollo del componente *backend*,

La estructura del presente documento se presenta de la siguiente forma. En la sección I, se detalla los antecedentes, objetivos y el alcance del proyecto. En la sección II, se menciona la implementación de la metodología *SCRUM,* los artefactos, la recopilación de requerimientos y el diseño de la arquitectura propuesta (MVC). La sección III, contiene el desarrollo de los requerimientos con sus respectivas pruebas unitarias e integrales. Por último, se encuentras las conclusiones y recomendación una vez finalizado las secciones anteriores.

PALABRAS CLAVE: Alimentación, *backend*, interacción, recetario, salud, web.

ABSTRACT

The COVID-19 pandemic revealed that adequate nutrition for people is of the utmost importance for good health, especially in people with obesity, overweight, diabetes and hypertension, which make up a large part of the contagion risk group. Some of the factors that increase poor nutrition in people were confinement and low interest in healthy food. This same problem brings with it diseases that descend on the general population.

Given this background, the development of a cookbook focused on the interaction between users, which includes a live chat, was proposed. The project allows you to share recipes created by users, provide feedback on them, make comments and rate published recipes. It is worth mentioning that we specifically focus on the development of the backend component,

The structure of this document is presented as follows. Section I details the background, objectives and scope of the project. In section ll, the implementation of the SCRUM methodology, the artifacts, the gathering of requirements and the design of the proposed architecture (MVC) are mentioned. Section III contains the development of the requirements with their respective unit and integral tests. Finally, the conclusions are found and it is recommended once the previous sections have been completed.

KEYWORDS: Food, backend, interaction, recipe book, health, web.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En el Ecuador el sobrepeso es un constante problema de salud, que se ha presentado principalmente en los jóvenes y adultos, los mismos que representan un 64% de la población del país, es decir, la mayor parte de la población adulta no tiene un peso adecuado según su altura y una de las causas es la alimentación tradicional de los ecuatorianos que consta principalmente en carbohidratos [1]. Según la Organización Mundial de la Salud (OMS) las enfermedades cardiovasculares se desarrollan por un alto y constante consumo de carbohidratos, por lo que recomiendan mantener una dieta saludable y variada, que reducirían los riegos de estas enfermedades [2].

La pandemia del COVID-19 puso en evidencia que una adecuada nutrición de las personas es de suma importancia para una buena salud, fundamentalmente en las personas con obesidad, sobrepeso, diabetes e hipertensión las cuales conforman gran parte del grupo de riesgo de contagio, para darnos una idea los primeros reportes de hospitales que salieron desde China informan que el 22% de pacientes fallecidos en las primeras sepas del SARS-CoV-2 padecían diabetes, asimismo la producción de alimentos se vio reducida provocando el aumento de los precios y los escases de productos [3].

Especialistas sugieren que comer bien y evitar bebidas azucaradas beneficia a la estimulación del sistema inmunológico, además recomiendan aumentar el consumo de vegetales, frutas, cereales integrales y grasas saludables para así permitir reducir los síntomas levemente [4] . Por las medidas y recomendaciones sobre nutrición en tiempos de pandemia las personas optan por preparar sus propios alimentos. Según un estudio realizado por MasterCard en España, el 70% de los ciudadanos empezaron a dedicar más tiempo a la cocina y mejorar sus habilidades culinarias, reflejando que un 54% de los españoles paso más tiempo en la cocina a consecuencia de desarrollar hábitos más saludables [5].

Debido al encierro, aprender de forma online se volvió muy común en este tiempo, por ende, la cocina también tomo esta normalidad, lo cual trajo múltiples beneficios ya seas o no experto en la cocina, de los cuales encontramos; El aprender a tu ritmo, permite la optimización de la economía individual, permite la relación con la tecnología y mejora las habilidades de cocina [6].

En cuanto a aplicaciones que ofrezcan listados de recetas de cocina en sitios web, existen varias, por ejemplo: *Diario, Cook, Recipe keeper o Recette Tek* que tienen como principal

función la creación, modificación y eliminación de recetas, sin embargo, estas no permiten una interacción personalizada entre usuarios [7].

Por tanto, el enfoque del presente proyecto es que la interacción y comunicación entre usuarios sea más interactiva, ofreciendo recetas saludables que permitan mejorar la salud de las personas. Para ello, se desarrollará el componente *backend,* el mismo que tendrá opciones de búsqueda de los platos favoritos de los usuarios, también podrá realizar comentarios y calificar las recetas, además al ser intuitiva y enfocada como una red social los mismos usuarios pueden interactuar entre sí mediante un chat en vivo, con el fin de compartir recetas o archivos.

1.1 Objetivo general

Desarrollar un *backend* que brinde un recetario gastronómico de forma pública que se encuentre basada en la interacción de los usuarios.

1.2 Objetivos específicos

1. Determinar los requerimientos del sistema para desarrollar las diferentes funcionalidades del *backend*.
2. Diseñar la base de datos y arquitectura de sistema, a raíz de los requerimientos obtenidos.
3. Implementar los módulos del sistema para la posterior verificación de funcionalidades.
4. Probar el funcionamiento del sistema para verificar que los requerimientos cumplan con el objetivo.

1.3 Alcance

Una de las principales causas del desarrollo de esta aplicación, es que en la actualidad los organismos como la Unicef y la OMS indican que los sistemas inmunológicos fuerte y sano ayuda a los cuerpos a luchar con el virus del COVID-19, por ende indica que se debe consumir múltiples alimentos como; el pescado, la carne, futra, verduras, hortalizas y cereales, estos mismos mezclados con alimentos antioxidantes permiten combatir de forma adecuada al virus fortaleciendo el sistema inmunológico [8].

El conocer cómo preparar alimentos para enfrentar al virus, permite que se mantenga un ambiente de positivismo en la situación actual, además que fomenta la creatividad junto al

refuerzo de la autoestima, que es fundamental para la salud física y mental en nosotros [9]. Por ello la propuesta de un recetario está enfocado para que los personas que deseen mejorar sus habilidades de cocina, salud o simplemente quieren explorar nuevas formas de preparación de alimentos, teniendo también la posibilidad de dar una opinión respecto a la preparación de su alimento o comunicarse entre usuarios, existe dos perfiles principales, usuario y administrador:

Perfil Administrador

- Gestión del sitio web
- Visualizar y buscar información del sitio web
- Gestión de usuario
- Gestión de recetas

Perfil usuario:

- Registro de usuario
- Recuperación de contraseñar
- Creación, modificación y eliminación de sus recetas
- Visualizar y buscar las diferentes recetas y usuarios
- Seguir los diferentes usuarios
- Comentar u opinar sobre una receta

1.4 Marco Teórico

Metodología

En el *software* la importancia de las metodologías destaca principalmente porque son un conjunto de técnicas y métodos organizativos que permiten diseñar soluciones en el programa, es decir, las diferentes metodologías intentan organizar los equipos de trabajo de una manera óptima para que el programa salga los mejor posibles y en el tiempo establecido. Uno de los mayores benéficos que da el desarrollo con metodologías es la organización de trabajo, siendo el caso que permite tener en cuenta los costes, planificación y la dificultad de proyecto, siendo el caso que al ser utilizadas permite reducir la dificultas del *software* y permite agilizar procesos junto al establecimiento de tareas, que mejoran en si el resultado del software que desarrollemos [10].

Metodologías Agiles

Entre las metodologías existentes tenemos las metodologías agiles, que son de las más utilizadas actualmente debido a su alta flexibilidad y agilidad. Al trabajar con equipos de trabajo permite que los mismos sean más productivos y eficientes, debido a que ya tiene establecido en parte las tareas que se realizaran para el desarrollo del *software*. Una ventaja del uso de metodologías agiles es que permite adaptar el *software* a las necesidades que pueden ir surgiendo en el camino que facilitan su construcción [10].

Este tipo de metodologías se basan en un modelo incremental, que en cada ciclo de desarrollo permite el ir agregando nuevas funcionalidades al sistema final, al ser una metodología de ciclos cortos una desventaja es el aumento de grandes funcionalidades que podrían afectar el tiempo de entrega del *software*. Al permitir construir equipos de trabajo autosuficientes, permite optimizar en gran medida las reuniones con el equipo de trabajo y permite que se construya el producto final de forma constante, también permite que el cliente aporte con nuevos requerimientos y correcciones, así como estar al tanto con el desarrollo de su sistema [10]**.** Existen múltiples metodologías que ofrecen distintos tipos de desarrollo de un proyecto, por ello en la **TABLA I**, se realiza una comparativa entre metodologías más usadas en la actualidad.

TABLA I. Comparativa de metodologías

Metodologías	**¿De qué trata?**	**Características**	**Ventajas**	**Desventajas**
Scrum	Esta metodología permite llevar a cabo las tareas de forma regular teniendo como principal objetivo trabajar de manera colaborativa.	• Desarrollo iterativo. • Se prioriza lo que tiene más valor para el cliente. • Permite la sincronización de los desarrolladores y permite adaptarlas.	• Resultados anticipados. • Flexibilidad y adaptación a los proyectos. • Gestión sistemática de riesgos.	• Funciona de mejor manera en equipos reducidos. • Tiene una exhaustiva definición de las tareas.

Kanban	Permite visualizar los flujos y la carga de trabajo, principalmente muestra el proyecto organizado por columnas en un tablero.	• Incluir lista de control. • Establece límites para el avance del proyecto. • Colocar fechas de vencimiento de tarjetas.	• No sé de produce en exceso. • Se acortan tiempo de entrega. • Se optimiza el espacio de almacenamiento.	• No se implementan ciclos muy largos. • Solo sirve en procesos repetitivos. • Se presenta variaciones en las jornadas de trabajo.
Programación extrema (XP)	Se encuentra basada en reglas y buenas prácticas del desarrollo de *software* en ambientes cambiantes, se enfoca en la comunicación entre el equipo de desarrollo y el cliente.	• Desarrollo incremental. • Programación en parejas. • Pruebas unitarias continuas. • Corrección periódica de errores.	• Relación con el cliente. • *Software* más estable por las pruebas continuas. • Aplicación rápida de cambios. • Menos errores por la programación pareja.	• Mayor esfuerzo de trabajo. • Requiere mayor tiempo. • Precio elevado. • Requiere control de versiones.

Scrum, permite principalmente tener un desarrollo controlado, que principalmente beneficia al proyecto que consta de múltiples componentes y fases, al tener Sprint también permite tener control de versiones, o de avances, también pudiendo ver como fue el avance del proyecto, también beneficia al control de tiempos de entrega del proyecto.

Backend

El desarrollo *backend* al ser una parte del desarrollo *web*, el mismo se encarga de la lógica de la página, en otras palabras, se encarga de la comunicación con el servidor. Al ser el esqueleto del *software* implica que el desarrollador debe ser meticuloso y cuidados, el trabajo del *backend* debe ser lógico, racional y/o menos creativos, debido a que el mismo tiene que seguir funcionalidades requeridas por el usuario [11]. Existen múltiples *framework* y herramientas para el desarrollo *backend* que combinan u ofrecen complementos para diferentes tipos de desarrolladores y sus necesidades, en la **TABLA II**¡Error! No se encuentra el origen de la referencia.**,** se realiza una comparativa entre dos herramientas utilizadas para el desarrollo del proyecto.

TABLA II. Comparativa entre herramientas

Framework	**¿Qué es?**	**Características**
Inertia.js	Permite la creación de aplicaciones de una sola página sin tener que construir una API. No se encuentra estrictamente ligado con Laravel o Vue, pero permite la unión entre los dos *frameworks* [12].	• Las rutas se encuentran contenidas en un archivo solitario. • Es posible configurarla con *React.* • Permite el desarrollo *backend* y el *frontend* a la par.
Laravel	Es un *framework* PHP de los más utilizados en la actualidad, que permite el desarrollo web de forma ágil. Principalmente se enfoca en el lado del desarrollador *backend* [13].	• Al tener el motor de plantilla Blade, permite visualmente más potentes. • Tiene una arquitectura bastante eficaz MVC.

Al utilizar inertial.js permite el trabajo de forma independiente de un sistema, esto se debe principalmente en la combinación de componentes de backend y frontend, es decir, utiliza herramientas de desarrollo como Laravel para el esqueleto y lo complementa con un framework visual como Vue o React. Al trabajar a la par con los diferentes componentes, permite un desarrollo más eficaz de cualquier sistema sin la necesidad de un API.

2 METODOLOGÍA

En la siguiente sección se menciona sobre la metodología de desarrollo que se utiliza junto a sus diferentes componentes y las herramientas que permiten el desarrollo del proyecto.

2.1 Metodología de Desarrollo

Al tener múltiples funcionalidades y tiempos de entrega cortos, *Scrum* implementa un conjunto de buenas prácticas que posibilitan trabajar colaborativamente para obtener resultados de un proyecto, para ello lo realiza a través de entregas parciales y constantes, también al existir entornos complejos y/o donde los requisitos están en constante cambio, el uso de *Scrum* beneficia la adaptación a estos y evita retrasos en la entrega del proyecto [14]. Una característica importante de esta metodología es el uso de ciclos temporales (iteraciones o Sprint s) que normalmente duran 2 semanas, las iteraciones ayudan a tener un mayor control de desarrollo del proyecto con la finalidad de ser entregado en los plazos establecidos y el mismo se realice con el mínimo de errores [14].

2.1.1 Roles

Cuando usamos esta metodología identificamos 3 principales roles: *Product Owner, Scrum Master y Devoloper team.* La asignación de roles nos ayuda a dar seguimiento a cada iteración o *Sprint* y cada uno posee distintas responsabilidades que permiten centralizar la atención en la construcción de un *software* de calidad [15].

Product Owner

La principal función es de comunicarse con el cliente y así garantizar una comunicación clara sobre el producto al resto de miembro del equipo, al actuar como enlace entre el cliente y el equipo, posee responsabilidades como [15]:

- Definir una visión o idear el proyecto a desarrollar.
- Ayuda a determinar los miembros del *devoloper team.*
- Explicar las historias de usuario al *devoloper team.*
- Aceptar/rechazar los entregables.
- Asegurar los recursos financieros del proyecto.

Scrum Master

Se le considera como el líder del *Devoloper team*, principalmente se encarga de la supervisión del cumplimento de reglas y la metodología, también debe mantener un ambiente de trabajo productivo para los desarrolladores, al ser el líder del equipo el mismo debe cumplir algunas responsabilidades como [15]:

- Seleccionar al *Developer team.*
- Ayuda en la actualización del tablero *Scrum* y el registro de implementaciones.
- Garantizar un ambiente ideal para el equipo de desarrollo.
- Determinará la duración de los *Sprint s*.

Developer team

Scrum team o Devoloper team son los encargados del desarrollo del proyecto, este grupo de personas trabajan junto las historias de usuario y una lista de pendientes de los Sprint para desarrollar los entregables, el equipo se caracteriza por ser autogestionado y multifuncional cuando se realiza el proyecto, normalmente consta de responsabilidades diferentes a los anteriores roles como [15]:

- Crear entregables.
- Asignar las historias de usuarios a los *Sprint.*
- Desarrollar la lista de tareas en base a las historias de usuario.
- Actualizar la lista de trabajo y del tablero S*crum*.

La **TABLA III,** se visualiza las asignaciones correspondientes del equipo *Scrum*, con sus respectivos nombres y cargos.

TABLA III. Asignación de roles

Rol	**Integrantes**
Product Owner	Ing. Mayra Isabel Alvarez
Scrum Master	Ing. Mayra Isabel Alvarez
Scrum Team	Sr. Josue Singaña

2.1.2 Artefactos

Al utilizar *Scrum* también trabajamos con los artefactos que posee, tienen como principal objetivo garantizar la transparencia y realizar el registro del proceso de desarrollo*,* es decir, es los recursos que garantizan la productividad y la calidad de los proyectos [16].

Recopilación de Requerimientos

Cuando iniciamos un nuevo proyecto debemos obtener los requerimientos del sistema, para ello hacemos una recopilación que se encarga de la identificación y documentación de las funcionalidades deseadas o indispensables para comenzar con el desarrollo de proyecto, esta actividad busca prevenir los errores que muchas veces significa el fracaso del proyecto. [17]. El levantamiento de requerimientos tienes dos encargados principales los cuales son: *Product Owner y Developer team*.

En la **TABLA IV,** se muestra un ejemplo de los primeros ítems de la recopilación de requerimientos. En el **ANEXO II,** estos se presentan de forma más detallada.

TABLA IV. Recopilación de requerimientos

RECOPILACION DE REQUERIMIENTOS			
ID-RR	Nombre del Requerimiento	ENUNCIADO DEL ITEM	Usuario
RR01	Iniciar sesión	Como usuario registrado: Necesito tener un perfil para tener acceso a las funcionalidades del *backend*. Como usuario administrador: Necesito tener un perfil para iniciar sesión y realizar la gestión del *backend*.	Registrado/Administrador

Historias de Usuario

La historia de usuario consiste en una técnica para realizar el levantamiento de requerimientos, consiste en la explicación general de una funcionalidad del *software* y la misma se encuentra escrita desde la perspectiva del usuario final o cliente, se caracteriza por estar escrita con pocas frases del lenguaje sencillo [18]. La **TABLA V,** la representación de una estructura de una historia de usuario. El resto de esta se encuentra en el presente documento, en el **ANEXO II**, sección *Historias de Usuario*

TABLA V Historia de usuario inicio de sesion

Historia de usuario	
Identificador (ID): HU01	**Usuario**: Registrado/Administrador
Nombre historia: Iniciar sesión	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Baja
Interacción asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios deberán poseer un perfil para ingresar a las funcionalidades del *backend* mediante el uso de correo y contraseña, dependiendo del tipo de usuario ingresara con el rol perteneciente.	
Objetivo: Ingresar al sistema con sus credenciales creadas o asignadas.	

Product Backlog

Consta de un documento que recopila en una lista ordenada los requisitos transformados en las historias de usuario con el fin de cumplir las necesidades de los clientes, al ser una lista la misma nunca se encontrara completa, provocando que su entorno se encuentre en constante cambio a lo largo del desarrollo. Este artefacto tiene como responsable el *Product Owner*, que se encarga principalmente en añadir y ordenar los ítems [16]. En la **TABLA VI,** se visualiza un ejemplo del *Product Backlog*. En el **ANEXO II**, se encuentra la correspondiente al *Product Backlog* del proyecto.

TABLA VI. Ejemplo de *Product Backlog*

ELABORACIÓN DE HISTORIAS DE USUARIO				
ID-HU	HISTORIA DE USUARIO	ITERACION	ESTADO	PRIORIDAD
HU01	Iniciar sesión	1	Pendiente	Alta
HU02	Gestionar recetas	2	Pendiente	Alta
HU03	Registrar usuarios	1	Pendiente	Alta

Sprint Backlog

En cuanto el trabajo de un desarrollador, consta en el desestructurar las historias de usuario es una práctica común, con el objetivo de obtener tareas más específicas y pequeñas que faciliten el desarrollo, para ello el Sprint *Backlog* consiste en seleccionar los requerimientos

del *Product Backlog* e ir reorganizando en subconjunto que permitan ser realizados en un periodo de tiempo determinado (*Sprint*) con el fin de continuar con el incremento del proyecto. Este pertenece a la parte del desarrollo el cual se encuentra incorporado por el *Scrum team,* los cuales se encargan de completar y así mismo deciden si se modifican o eliminan [16]. La **TABLA VII**, nos presenta un ejemplo de elaboración de un *Sprint Backlog*. En el **ANEXO II**, se encuentra la correspondiente al *Sprint Backlog*.

TABLA VII. Ejemplo de *Sprint Backlog*

ELABORACIÓN DE SPRINT BACKLOG					
ID-SB	NOMBRE	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO EN HORAS
SB00	Configurar el entorno de desarrollo.	N/A	N/A	Instalación y actualización de herramientas de desarrollo. Diseño y creación de la BDD. Creación del proyecto en Inertia.js Creación del proyecto jetstream.	10 H

2.2 Diseño de *backend*

Una vez que se obtuvieron los requerimientos del sistema, se procede a la implementación de las mismas, para el desarrollo funcional del recetario. Por lo tanto, se comenzó la implementación de cada Historia de usuario, con el propósito de cumplir los objetivos previamente definidos.

2.3 Diseño de la arquitectura

Al seleccionar una arquitectura eficiente para los diferentes componentes, se considera de importancia debido a las dificultades de desarrollo que posee un proyecto de *software*, el mismo varía dependiendo del sistema que se va a construir. Para el desarrollo del *backend*, se ha propuesto el patrón arquitectónico Modelo-Vista-Controlador como la base de la arquitectura de datos.

2.3.1 Patrón arquitectónico

El patrón de arquitectura MVC consiste en separar el código para determinar distintas responsabilidades, y así mantener diferentes secciones encargadas en la realización de una tarea en concreto. Cada una de la capa de la división del código se le denomina como Modelo Vista Controlador respectivamente, que tiene como parámetros deseables la estructuración y reutilización de código, el cual influye positivamente en el desarrollo y el mantenimiento del mismo [19] .

Como parte de la arquitectura tenemos el modelo que trabaja con los datos permitiendo acceso a la información y la actualización de sus estados. Normalmente se encuentran en la base de datos, por lo cual tenemos todas las funcionalidades para el manejo de las tablas como: *selects, update, inserts, etc* [19].

Por otro lado, las vistas contienen el código de la aplicación que produce la visualización de las interfaces de usuario, es decir, es la capa que interactúa directamente con el usuario que ayuda a interpretar los estados de la aplicación en lenguaje *HTML* y *PHP* permitiendo mostrar la salida de las mismas. Generalmente se trabaja con los datos, pero no se tiene acceso directo a estos [19].

Finalmente, los controladores del código son necesarios para responder a las peticiones que se solicita en la aplicación, por ejemplo las peticiones que se realiza cuando realizamos una búsqueda, el controlador se encarga de recibir la petición, evaluarlo y comunicarse con la capa de modelos para posteriormente devolverla a la interfaz de usuario, en resumen actúa como enlace de las vistas y los modelos lo cual provoca que no manipule directamente la entrada de los datos [19].

A continuación, se presenta la **Fig. 1,** misma que representa el patrón arquitectónico del *backend*, en el cual se explica cómo se complementa los *frameworks* de *Laravel* y *Vue. Laravel* se conecta a la base de datos para la obtención y almacenamiento de los mismo. En segundo aspecto contiene los controladores que se va a utilizar para la lógica del proyecto que trabaja junto inertia.js para realizar la comunicación de las vistas, la herramienta de Inertia.js permite realizar las vistas con *frameworks* de nuestro agrado, en el patrón arquitectónico presentado se utiliza *Vue* que actuara como la vista de nuestro proyecto, así devolviendo una respuesta visualmente atractiva para nuestro usuario final.

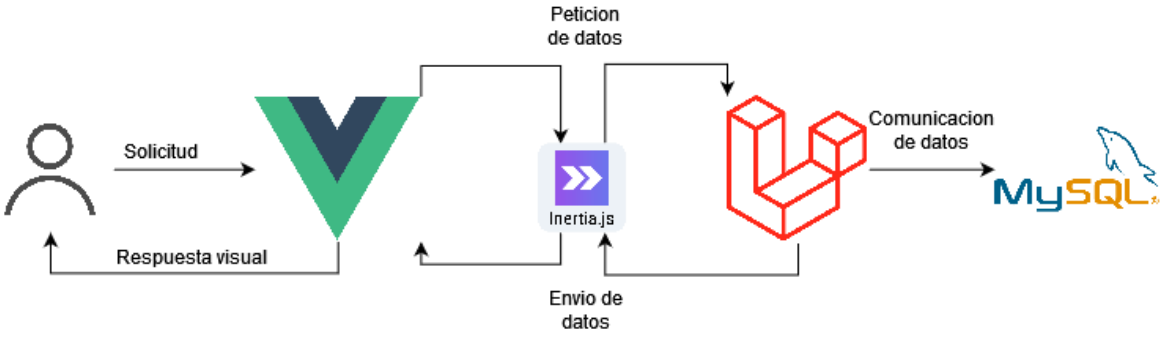


Fig. 1 Patrón arquitectónico

2.4 Herramientas de desarrollo

En el presente proyecto se utilizan las herramientas de desarrollo que se indican en la **TABLA VIII**, la cual detalla el funcionamiento de cada una y también se especifica en que parte del desarrollo *backend* se utiliza.

TABLA VIII. Herramientas para el desarrollo del *backend*

Herramienta	Descripción	Justificación
Visual Studio Code	Es un editor de código que permite trabajar con diferentes lenguajes de programación, además ofrece la posibilidad de instalar extensiones que faciliten el trabajo de desarrollo [20].	Se decido utilizar *Visual Studio Code* como ambiente de desarrollo, debido a que permite el uso de *plugins* para mejorar la dinámica de desarrollo, ayuda a obtener acceso rápido a los diferentes archivos y mayor familiaridad con la herramienta.
Mysql	Es un gestor de base de datos relacionales que se encuentra basado en el lenguaje de consulta SQL, se encuentra basado modelo cliente-servidor [21].	Al ser un sistema gestor de base de datos de código abierto, se vuelve perfecto para el proyecto, debido a que soporta la manipulación de datos a gran escala.
Laravel	Trabaja con una arquitectura de carpetas avanzadas, que permiten la separación de archivos en un orden definitivo, asi definiendo su estilo arquitectónico MVC [22].	Se utilizó Laravel para el desarrollo de *backend* por la facilidad del patrón arquitectónico MVC que posee.

Vue	*Vue* es un *framework* reactivo que permite construir interfaces de forma sencilla, debido a que se encuentra basado en JavaScript, al ser reactivo el mismo reacciona a los cambios facilitando la presentación en la aplicación [23].	Se usó Vue para el desarrollo de vistas el cual facilita la implementación de funcionalidades, debido a la combinación que tiene con el lenguaje JavaScript.
Laravel Jetstream	*Jetstreams* otorga una mesa prediseñada de trabajo cuando comenzamos un proyecto con Laravel. Se caracteriza por acomodarse a los tiempos de desarrollo debido a que posee una base sólida de inicio, también ofrece *stack* los cuales permiten integrar un *framework frontend* [24].	Se implementó *Jetstream* para implementar las pantallas de inicio que son prediseñadas, las cuales también son editables de forma sencilla, además que ayuda a implementar herramientas para mejorar el desarrollo.
Inertial.js	Permite diseñar aplicaciones singles page, sin la necesidad de construir una API, al no ser un *framework* no remplaza ninguno existente del lado del servidor o del cliente, pero se encuentra diseñado para trabajar junto a ellos [12].	Se usó inertial.js como enlace entre Laravel y Vue que permite el desarrollo a la par de los componentes de *backend* y *frontend*.
Pusher	Es un servicio online que encapsula la implementación de un *websockets* así permitiendo el manejo de funcionalidades en tiempo real de las aplicaciones en desarrollando, una de los servicios que permite implementar son: chats, paneles o notificaciones [25].	*Pusher* ayuda a implementar diferentes funcionalidades en vivo como: chat y notificaciones que se requieren para la comunicación entre usuarios en tiempo real.

xampp	Es una herramienta basada en PHP de desarrollo que permite probar una aplicación dentro de tu propio ordenador sin la necesidad de tener acceso a internet [26].	La herramienta de Inertia.js realiza las modificaciones en tiempo real, por ende, está constantemente conectado a una base de datos para realizar la modificación de los mismos.
tailwind css	Consiste en un *framework* para el diseño personalizado, que permite el desarrollar de forma ágil de la parte *frontend* y así optimizar los aplicativos webs [27].	Al utilizar Vue como *frontend*, utiliza las herramientas de *tailwind* para estilizar cada funcionalidad de la *backend*.
Composer	Esta herramienta se encarga del manejo de paquetes para PHP, es decir, el mismo permite administrar, descargas e instalar dependencias y librerías [28].	El poder instalar librerías, ayuda a optimizar y facilitar el desarrollo, principalmente del b*ackend.* Debido a reducen las líneas de código de un proyecto.
JMeter	*JMeter* es una herramienta de *testing*, que permite diseñar un *testplan* para generar un fichero el cual ayuda a ejecutar los diferentes componentes agregados en las funcionalidades del sistema [29].	*JMeter* permite realizar grabar un *script* con las peticiones que realicemos en nuestro sistema, para ejecutar y encontrar que funcionalidades presentan problemas al enviar solicitudes de forma constante mediante el informe que nos entrega.
Postman	*Postman* es una herramienta el cual permite realizar peticiones de manera simple a las *APIs* de tipo *REST,* es decir, permite verificar la información sé que distribuye mediante las *APIs* [30].	Al utilizar *postman*, ayuda a verificar la información que se solicita y también permite comprobar cómo se está esta información se está devolviendo para su posterior uso.

2.5 Librerías

Igualmente, el uso de librerías complementa el desarrollo ágil del proyecto, las cuales se presentan en la **TABLA IX**, que especifica el funcionamiento que tendrán estas librerías en el progreso de proyecto.

TABLA IX. Librerías para el desarrollo del progreso

Liberia	Justificación
spatie/laravel-permission	Librería de *spatie* que proporciona elementos de permisos y roles para los usuarios en una base de datos.
pusher/pusher-php-server	Librería que proporciona elementos de conexión al servidor de *pusher* para el desarrollo del chat en vivo.

3 RESULTADOS

A continuación, se detallará los resultados de cada *Sprint,* referente al desarrollo de la *backend* junto a las funcionalidades que se dividieron en 5 *Sprints*, el *Sprint* 0 que consiste en la configuración inicial y es el primero a ejecutar.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo

En este *Sprint* s necesario debido a que se lleva a cabo la configuración correspondiente del ambiente de desarrollo, permitiendo que el *backend* siga su flujo sin interrupciones. Los resultados obtenidos en esta iteración son los siguientes:

- Instalación de herramientas de desarrollo.
- Diseño y creación de la base de datos.
- Creación del proyecto en Laravel*J etstream* inertia.js.

Instalación de herramientas de desarrollo

Visual Studio Code

Visual Studio Code se encuentra en su página web oficial, una vez descargado, la instalación inicia automáticamente que al ser un software gratuito solo dependerá designar en donde instalaremos. Posteriormente se procede a iniciar el ambiente de desarrollo para el inicio de desarrollo.

Xampp

La herramienta Xampp es útil cuando se trabaja con base de datos debido a que cuenta con herramientas de MySQL, PHP, servicios y paquetes que se instalan de forma tradicional individualmente. Al ser un instalador se ejecuta de forma automática instalando los paquetes y servicios que se encuentren por defecto.

JMeter

Consta de un instalador que se descarga de la página oficial de Apache, por defecto y para el proyecto la configuración recomendada es la que se utiliza para realizar los *testing* o el plan de *testing*. La herramienta se utilizará una vez que finalice el proceso de desarrollo.

Diseño y creación de la base de datos.

La base de datos para el funcionamiento del *backend* es de tipo relacional, se encuentra definida con el fin que la misma tenga una tabla de Usuarios, Comentarios, Mensajes, *Post,*

Likes, Follower, *Chats*. El modelo sirve para el flujo de datos que se requiere en el desarrollo de los requerimientos. El diseño de la base de datos se presenta en la sección **ANEXOS** Estructura de la base de datos mientras que en la **Fig. 2,** se presenta una tabla de la misma base de datos.

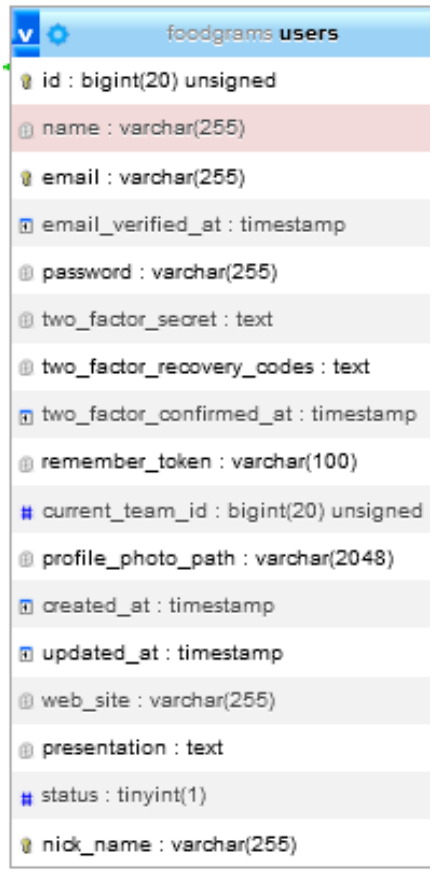


Fig. 2 Tabla de usuarios de la Base de datos

Creación del proyecto en Laravel *Jetstream* inertia.js.

En la instalación de *Laravel,* es requerido que se encuentre instalado el gestor de dependencias PHP llamado *composer*, debido a que *Laravel* se encuentra basado en PHP. En la **Fig. 3,** se presenta el comando para ejecutar la instalación de Laravel

```
C:\Users\Usuario>composer global require laravel/installer
```

Fig. 3 Comando de instalación de Laravel

Una vez instalado *Laravel* se procede a crear el proyecto, la **Fig. 4,** muestra el comando para comenzar el desarrollo utilizando *Laravel jetstream* el cual permite especificar más adelante con que *stack* vamos a trabajar.

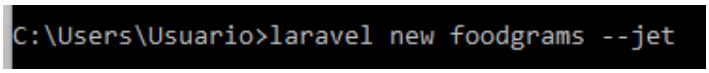


Fig. 4 Comando para empezar instalación de Jetstreams

Como se especificó anteriormente, *jetstream* permite trabajar con dos *stack*, en este caso se utiliza *inertial.js* como se muestra en la **Fig. 5,** el mismo permite la utilización del *framework* para el desarrollo visual.

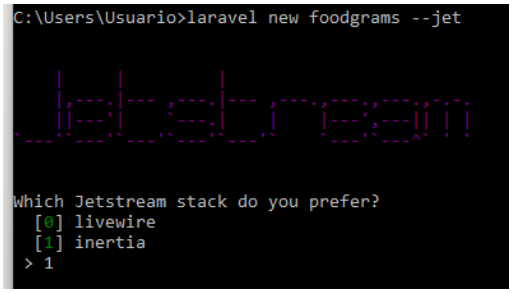


Fig. 5 Selección de stack Inertial.js

Cuando finalice la instalación del stack que se utilizara, se ejecuta el proyecto con Visual Studio Code. Al abrir el ambiente de desarrollo se mostrará la estructura de directorios inicial del proyecto, véase la **Fig. 6.**

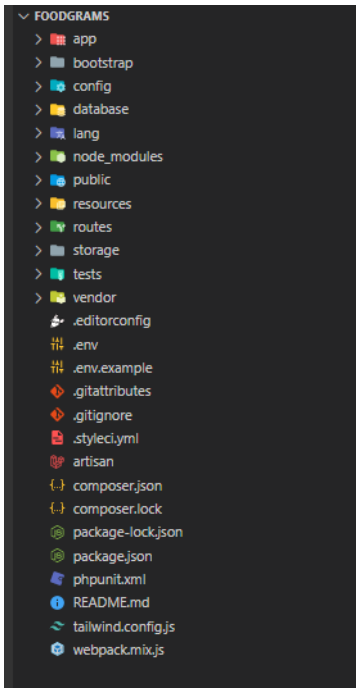


Fig. 6 Estructura inicial del proyecto

Luego de definir la relación entre los datos a través del diseño de la base de datos, se realiza la creación de las migraciones como se presenta en la **Fig. 7,** en cada una de estas migraciones se establecen las relaciones para la subida de información al gestor de base

de datos, un ejemplo de las relaciones establecidas se presenta en la **Fig. 8,** en donde se muestran las conexiones mediante el uso de claves foráneas.

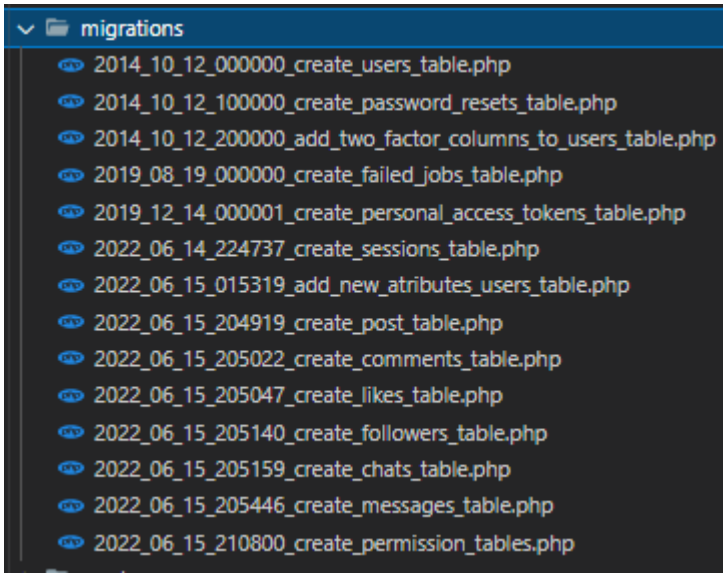


Fig. 7 Creación de migraciones

```php
Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->string('image_path');
    $table->text('description');
    $table->text('ingredients');
    $table->integer('portions')->default(0);
    $table->date('date_post');
    $table->foreignId('user_id');
    $table->timestamps();
}); <- #16-26 Schema::create
```

Fig. 8 Estructura de relaciones

Al realizar las migraciones se evidencia en el gestor de datos como se encuentren ya relacionadas cada una de las tablas, en la **Fig. 9**, se visualiza la creación de las tablas, junto a los elementos de la tabla *users*.

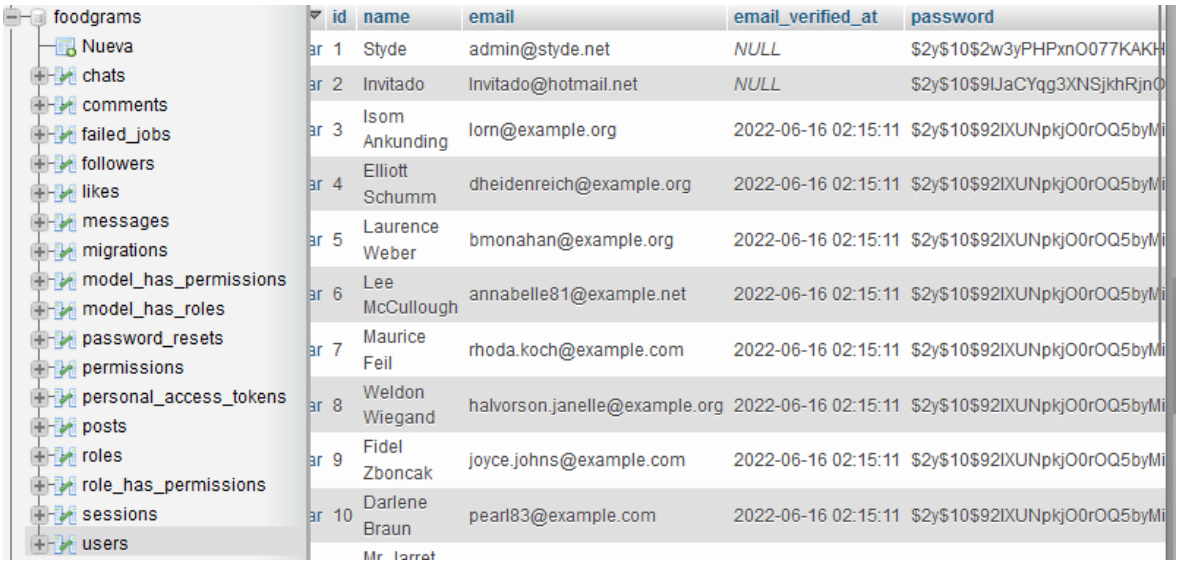


Fig. 9 Estructura de la Base de Datos

3.2 *Sprint* 1. Módulo de gestión de usuarios e inicio de sesión

El *Sprint* 1, se estableció de manera que el usuario ingrese en el proceso de registro y el mismo tenga acceso a las funcionalidades del *backend,* también se incluye el proceso de modificación de datos del perfil ingresado. A continuación, se listan los resultados alcanzados después de ejecutar las tareas designadas del Sprint.

- Registro de usuario.
- Iniciar sesión.
- Gestión de recetas administrador.
- Modificación de datos del perfil.

Registro de usuarios.

Otro formulario que implementa *Jetstream* es el de registro, que es fundamental para la creación de un perfil de un nuevo usuario, al no ser un formulario enfocado para un uso mayor es necesario implementar nuevos campos como lo es el *Nick_name*. La **Fig. 10,** indica el formulario modificado para el registro y la validación de datos.

```
<form @submit.prevent="submit">
    <div>
        <JetLabel for="name" value="Name" />
        <JetInput
            id="name"
            v-model="form.name"
            type="text"
            class="mt-1 block w-full"
            required
            autofocus
            autocomplete="name"
        />
    </div>

    <div class="mt-4">
        <JetLabel for="nick_name" value="Nick_name" />
        <JetInput
            id="name"
            v-model="form.nick_name"
            type="text"
            class="mt-1 block w-full"
            required
```

Fig. 10 Formulario de registro de usuario

Al tener la estructura de registro definida con los nuevos campos incorporados, se procede a implementa las validaciones y modificar las validaciones existentes en el formulario de registro. La **Fig. 11** muestra cómo se realiza las validaciones para el registro de nuevos usuarios.

```
public function create(array $input)
{
    Validator::make($input, [
        'name' => ['required', 'string', 'max:255'],
        'nick_name' => ['required', 'string', 'max:255', 'unique:users'],
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'password' => $this->passwordRules(),
        'terms' => Jetstream::hasTermsAndPrivacyPolicyFeature() ? ['accepted', 'required'] : '',
    ])->validate(); <- #23-29 Validator::make

    return User::create([
        'name' => $input['name'],
        'nick_name' => $input['nick_name'],
        'email' => $input['email'],
        'password' => Hash::make($input['password']),      JosueEPN, 6 months ago • Inicio del p
    ])->assignRole('Creador');; <- #31-36 return User::create
} <- #22-37 public function create(array $input)
```

Fig. 11 Validaciones para el registro y creación de usuarios

Iniciar sesión

Al utilizar *jetstream* ya incorpora lo necesario para realizar la autentificación para el inicio de sesión y un formulario de registro, el cual se modifica acorde a los requerimientos. La **Fig. 12,** muestra dicho formulario correspondiente para el acceso al sistema mediante el uso del correo electrónico y la contraseña. Tiene como propósito la protección de las demás vistas, con el fin de salvaguardar la información puesta por los usuarios y proteger las conversaciones privadas que los mismos contengan, la protección se realiza mediante un *middleware* como se muestra en la **Fig. 13** , esta función protege y verifica si un usuario se encuentra autentificado.

```
<form @submit.prevent="submit">
    <div>
        <JetLabel for="email" value="Email" />
        <JetInput
            id="email"
            v-model="form.email"
            type="email"
            class="mt-1 block w-full"
            required
            autofocus
        />
    </div>

    <div class="mt-4">
        <JetLabel for="password" value="Password" />
        <JetInput
            id="password"
            v-model="form.password"
            type="password"
            class="mt-1 block w-full"
            required
            autocomplete="current-password"
        />
```

Fig. 12 Formulario del inicio de sesión

```
Route::middleware([
    'auth:sanctum',
    config('jetstream.auth_session'),
    'verified',
```

Fig. 13 Protección de rutas mediante middleware

Perfil para iniciar sesión de administrador

El perfil administrador necesita acceder a las funcionalidades del sistema, por ello se generan automáticamente dentro del código especificando las credenciales junto al rol que tendrá mediante la utilización de *Laravel/perrmision* la cual permite asignar los roles y permisos a usuarios determinados. La **Fig. 14**¡Error! No se encuentra el origen de la referencia., muestra como es la asignación del rol para el usuario administrador que estamos registrando.

```php
User::create([        You, 10 seconds ago • Uncommi
    'name' => 'Josue Singaña',
    'email' => 'Josuesin01122015@gmail.com',
    'nick_name' => 'Josin15',
    'email_verified_at' => now(),
    'password' => bcrypt('secret'), // password
    'remember_token' => Str::random(10),
])->assignRole('Admin'); <- #22-29 User::create
```

Fig. 14 Creación de perfil administrador.

Modificación de datos del perfil.

Un aspecto importante para el funcionamiento del recetario con perfiles, es la modificación y edición de la información de los usuarios, para ello se procede a reutilizar y modificar los formularios los cuales deben incluir la modificación del web-site y descripción. La **Fig. 15**, presenta el formulario en la cual consta los nuevos campos de edición, mientras que la **Fig. 16**, muestra cómo se almacenan los nuevos datos y se validan los datos ingresados cuando se manda a modificar en la base.

```
<!-- Web site -->
<div class="col-span-6 sm:col-span-4">
    <JetLabel for="web_site" value="Web_site" />
    <JetInput
        id="web_site"
        v-model="form.web_site"
        type="text"
        class="mt-1 block w-full"
    />
    <JetInputError :message="form.errors.web_site" class="mt-2" />
</div>
 <!-- Presentation -->
<div class="col-span-6 sm:col-span-4">
    <JetLabel for="presentation" value="Presentation" />
    <JetInput
        id="presentation"
        v-model="form.presentation"
        type="text"
        class="mt-1 block w-full"
    />
    <JetInputError :message="form.errors.presentation" class="mt-2" />
</div>
```

Fig. 15 Formulario de Edicion de perfil

```php
public function update($user, array $input)
{
    Validator::make($input, [
        'nick_name' => ['required', 'string', 'max:255', 'unique:users'],
        'presentation' => ['required', 'string', 'max:255'],
        'web_site' => ['required', 'string', 'max:255'],
        'email' => ['required', 'email', 'max:255', Rule::unique('users')->ignore($user->id)],
        'photo' => ['nullable', 'mimes:jpg,jpeg,png', 'max:1024'],
    ])->validateWithBag('updateProfileInformation');    Undefined method 'validateWithBag'. <-

    if (isset($input['photo'])) {
        $user->updateProfilePhoto($input['photo']);
    }

    if ($input['email'] !== $user->email &&
        $user instanceof MustVerifyEmail) {
        $this->updateVerifiedUser($user, $input);
    } else {
        $user->forceFill([
            'nick_name' => $input['nick_name'],
            'presentation' => $input['presentation'],
            'web_site' => $input['web_site'],
            'email' => $input['email'],
        ])->save(); <- #38-43 $user->forceFill
    } <- #37-44 else
} <- #21-45 public function update($user, array $input)
```

Fig. 16 Modificación y validación de datos de usuario

Pruebas unitarias del *Sprint* 1

Finalizado el *Sprint 1,* se procede a realizar las comprobaciones necesarias, al crear un perfil administrador, el mismo tiene acceso a la modificación de datos y a visualizar los perfiles que han sido creados en el *backend*. La **Fig. 17**, muestra la tabla con los usuarios y los botones para la gestión de datos.

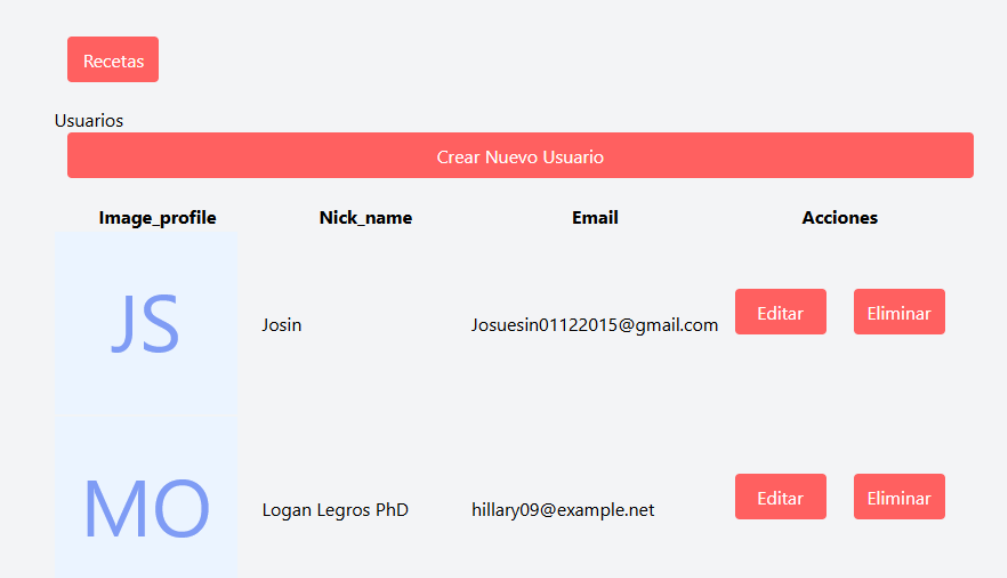


Fig. 17 Seccion de administracion de usuarios

3.3 Sprint 2. Módulo de gestión de recetas

El *Sprint* 2, se estableció de forma que se encargase de la gestión de recetas creadas por los usuarios, en donde se debe implementar los accesos especiales a los administradores para la gestión de la información que se comparte dentro del *backend*. Se presentan los resultados de este *Sprint* en la siguiente lista:

- Gestión de recetas por el usuario registrado.
- Perfil para iniciar sesión de administrador.
- Presentación de post recetas.

Gestión de recetas por el usuario registrado.

En la gestión de recetas se consideró que el usuario debe realizar el CRUD de sus recetas, para la creación se creó un formulario en donde se incluye la subida de imágenes, el título de la receta, porciones, ingredientes y preparación. La **Fig. 18**, indica el formulario para la creación de recetas, en cuanto la **Fig. 19**, se presenta la función que se encarga de la creación y almacenamiento en la base de datos.

```
<div class="grid grid-cols-2 basis-1/2">

    <div class="m-3">
        <label for="title" class="text-lx font-serif">Titulo:</label>
        <input v-model="text" type="text" placeholder="title" id="title" class="ml-2 outline-none p
        <input-error :message="errors.text"/>
    </div>

    <div class="m-3">
        <label for="title" class="text-lx font-serif">Porciones</label>
        <input v-model="number" type="number" class="ml-2 outline-none py-1 px-2 text-md border-2 r
        <input-error :message="errors.number"/>
    </div>

    <div class="m-3">
        <label class="block mb-2 text-lg font-serif">Ingredientes</label>
        <textarea v-model="text2" id="Ingredientes" cols="30" rows="10" placeholder="whrite here.."
        <input-error :message="errors.text2"/>
    </div>

    <div class="m-3">
        <label class="block mb-2 text-lg font-serif">Preparacion:</label>
        <textarea v-model="text3" id="description" cols="30" rows="10" placeholder="whrite here.." 
        <input-error :message="errors.text3"/>
    </div>


</div>
```

Fig. 18 Formulario de creacion de recetas

```php
public static function createPost($request){
    $file = $request->file('image');
    $name = uniqid().$file->getClientOriginalName();
    $url = null;

    $storage = Storage::disk('public')->put($name,$file);
    $url = asset('storage/'.$storage);

    $post = (new static)::create([
        'title' => $request->text,
        'image_path' => $url,
        'description' => $request->text3,
        'ingredients'=> $request->text2,
        'portions' => $request->number,
        'date_post' => Carbon::now(),
        'user_id' => Auth::id(),
    ]); <- #57-65 $post = (new static)::create

    return (new static)::with([
        'user',
        'comments',
        'likes'
    ])->find($post->id); <- #67-71 return (new static)::with
} <- #49-72 public static function createPost($request)
```

Fig. 19 Función de creación

Realizado el formulario de creación, se procede a comprobar si el mismo se puede editar para ello se utiliza el método *update* que se encuentra incorporado por defecto con la creación del controlador de post, la **Fig. 20**, presenta la función de edición o actualización de recetas.

```php
public function update(RequestEdit $request, Posts $posts)
{
    $editpost= $request->all();

    if($request->file('image')){
        Storage::delete('public'. $request->image_path);
        $file = $request->file('image');
        $name = uniqid().$file->getClientOriginalName();
        $storage = Storage::disk('public')->put($name,$file);
        $editpost['image'] = asset('storage/'.$storage);

    }else{
        unset($editpost['image']);
    }

    $posts->update($editpost);
    return $editpost;


}
```

Fig. 20 Función de edición de post.

Teniendo las funciones de edición y creación, también se debe eliminar una receta en caso de que el mismo ya no se desee compartir, para ello se utiliza el método *destroy* el cual debe eliminar todos los elementos correspondientes a ese post como lo es; los comentarios y los *likes*. La **Fig. 21**, se presenta como se encuentra estructurado el método mencionado.

```php
public function destroy(Posts $post, User $user)
{
        $likes = $post->likes;

        foreach($likes as $item) { //foreach element in $arr
            $deleteLikes = Likes::find($item['id']);
            $deleteLikes ->delete();
        } <- #119-123 foreach($likes as $item)

        $comments = $post->comments;

        foreach($comments as $item) {
            $deleteComments = Comments::find($item['id']);
            $deleteComments ->delete();
        }
        $url = str_replace('storage', 'public', "post");
        Storage::delete( $url,$post->image_path);
        $post->delete();
        return back();
} <- #115-135 public function destroy(Posts $post, User $user)
```

Fig. 21 Metodo destroy para eliminacion de post.

Gestión de recetas por el usuario administrador.

Al tener un perfil administrador, el mismo necesita tener acceso a la modificación de datos de las recetas, para tener un mayor control de las publicaciones y lo que se comparte en ellas. En la **Fig. 22**, se presenta la forma como se obtiene todos los *posts* que se encuentran en el *backend.* La **Fig. 23**, presenta una de las funcionalidades de la gestión de recetas, como lo es la actualización de datos.

```php
public function indexPost(){
    $posts = Posts::get();

    return Inertia::render('Admin/indexPost',[
        'posts' => $posts,

    ]);
} <- #36-43 public function indexPost()
```

Fig. 22 Presentación de post para el administrador

```php
public function update( Request $request)
{
    $editUser = User::find($request->id);
     $this->validate($request, [
        'nick_name' => 'string', 'max:255',  Rule::unique('users')->ignore($editUser->id),
        'presentation' =>  'nullable','string', 'max:255',
        'web_site' => 'nullable','string','max:255',
        'email' => 'required', 'email', 'max:255', Rule::unique('users')->ignore($editUser->id),
        'photo' => 'nullable', 'mimes:jpg,jpeg,png', 'max:1024',
    ]); <- #58-64 $this->validate

    if (isset($request->photo)) {
        $editUser->updateProfilePhoto($request->photo);
    } else {
        $editUser->forceFill([
            'nick_name' => $request->nick_name,
            'presentation' => $request->presentation,
            'web_site' => $request->web_site,
            'email' => $request->email,
        ])->save(); <- #69-74 $editUser->forceFill
    } <- #68-75 else
    return to_route('index.user.admin');
} <- #54-79 public function update( Request $request)
```

Fig. 23 Función de edición de *posts* del usuario administrador

Presentación de post recetas.

Una característica importante cuando se trabaja con perfiles de seguidores y seguidos, consiste en la presentación de las nuevas recetas en la pantalla principal, ya sean propios o de los usuarios a los cual se encuentran suscritos. En la **Fig. 24**, se presenta la función la cual consulta a la base de datos los *posts* de los usuarios, mientas que la **Fig. 25**, presenta como son impresos en la pantalla principal.

```php
public static function getPost($id,$profile = null){
    $querry = (new static)::with([
        'user',
        'comments' => function($query){
            $query->with('user:id,name,nick_name,profile_photo_path');
        },
        'likes'
    ]) <- #76-82 $querry = (new static)::with
    ->where('user_id',$id);
    if(is_null($profile)){
        $querry = $querry ->orWhereIn('user_id',Followers::select('user_id')->where('follower_id',$id)->get());
    }

   return $querry->orderBy('created_at','desc')
    ->get();

} <- #75-91 public static function getPost($id,$profile = null)
```

Fig. 24 Consulta de post en la base de datos.

```vue
<div v-if="posts.length > 0">
    <post-component v-for="post in posts" :key="post.id"
        :post="post"></post-component>
</div>
<div v-else class="text-3xl">No hay publicaciones</div>
```

Fig. 25 Presentación de posts en la pantalla principal.

Pruebas unitarias del *Sprint* 2

Una vez concluido el desarrollo del *Sprint* 2, se presenta la **Fig. 26**, la cual indica desde la perspectiva del administrador toda la gestión para las recetas que se encuentren creadas por los usuarios, mientras que la **Fig. 27**, se indica como las mismas recetas se presentan en la página principal de los usuarios seguidores.

Fig. 26 Gestión de recetas, perspectiva de administrador

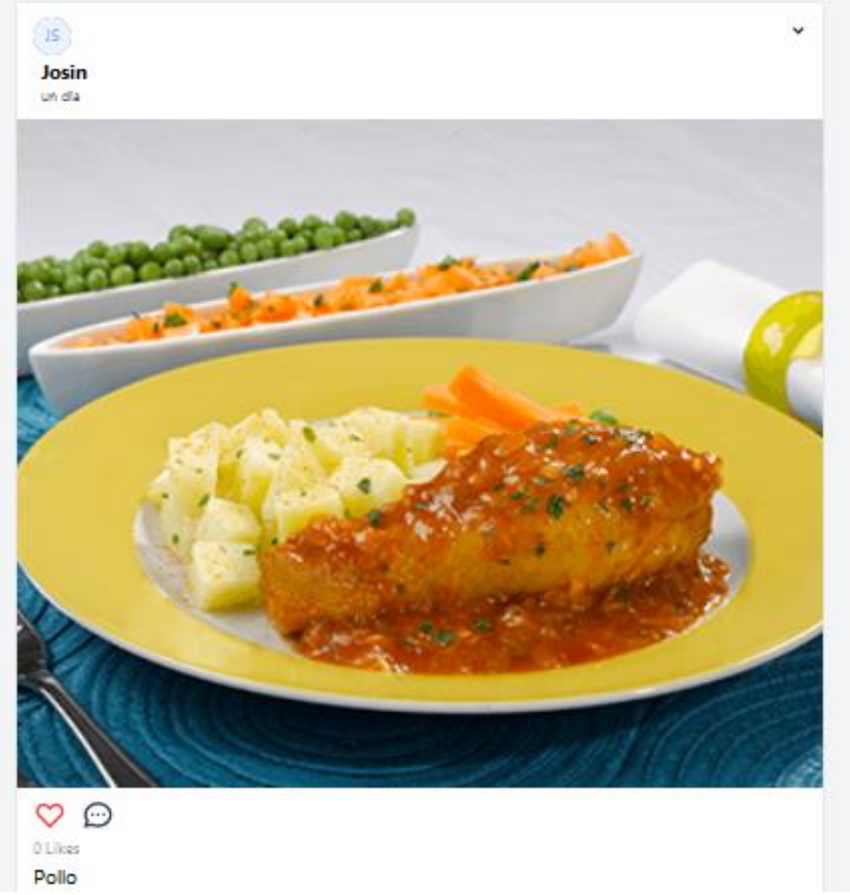


Fig. 27 Presentación de post en la pantalla principal

3.4 Sprint 3. Módulo de búsqueda y visualización de perfiles y recetas

El *Sprint 3,* se estableció de manera que el usuario pueda filtrar las recetas o perfiles a los cuales se encuentra interesado, también las mismas puedan ser visualizadas de manera que puedan ver las informaciones correspondientes como lo es el perfil con sus recetas creadas o la preparación de la receta consultada. A continuación, se listan los resultados alcanzados después de ejecutar las tareas designadas del *Sprint.*

- Filtrar recetas y usuarios.
- Visualizar perfil de usuarios.
- Visualizar recetas.

Filtrar recetas y usuarios registrados

Una vez terminada la sección de modificación de datos, es importante la implementación de *input* para recetas y perfiles, el cual permita filtrar de forma rápida la información del sistema, para ello se realiza un controlador que permita realizar este proceso, la **Fig. 28**, indica el funcionamiento del controlador y como filtra los datos.

```php
public function index()
{
    $posts = Posts::orderBy("id");
    $userRed = User::orderBy("id");

    if(request()->has("search")){
        $title=request("search");
        $posts = $posts->where('title' ,'like' ,'%'.$title."%");
        //$id = $posts->user_id;
        //$userRed = $userRed->where('id','like','%'.$id.'%');
    } <- #23-28 if(request()->has("search"))

    $posts= $posts->paginate(10);

    $users = User::orderBy("id");

    if(request()->has("search")){
        $nick_name=request("search");
        $users = $users->where('nick_name' ,'like' ,'%'.$nick_name."%");
    }

    $users = $users->paginate(10);
   // return $posts;

    return Inertia::render('Search', compact('users','posts'));
} <- #19-46 public function index()
```

Fig. 28 Filtro de recetas y perfiles.

Visualizar perfil de usuarios

Junto al filtro se implementa la visualización de perfiles de usuarios, mediante el redireccionamiento y el ID que se les otorga a los usuarios en cuanto son creado, se

procede a cargar la información de los usuarios, junto a los posts o recetas que los mismos compartieron dentro del sistema, La **Fig. 29**, indica como mediante la rutas se trasladan los datos a los controladores que procederá a presentar en las vistas, la **Fig. 30**, indica como es la interacción de datos con el controlador.

```
//Profile
Route::get('/profile/{nick_name}', [ProfileController::class,'index'])->name('profile');
```

Fig. 29 Ruta de visualización de perfiles.

```
public function index($nick_name){
    $user = $this->user->where('nick_name',$nick_name)->first();

    $followers = $user->followers()->count();
    $followed = $this->followers->where('follower_id',$user->id)->count();
    $postsCount = $user->post()->count();
    $posts = Posts::getPost($user->id,true);

    return Inertia::render('UserProfile/index',[
        'userProfile' => $user,
        'followers' => $followers,
        'followed' => $followed,
        'postsCount' => $postsCount,
        'posts' => $posts,
    ]); <- #32-38 return Inertia::render
} <- #24-39 public function index($nick_name)
```

Fig. 30 Función de controlador para resentacion de perfiles.

Una vez el controlador presente la vista con el perfil del usuario que se visita, el mismo se divide en dos secciones, el primero se encarga de la presentación de la información del perfil al cual se ingresó como el *Nick_name* y la foto de perfil, en la **Fig. 31**, se presenta como se encuentra estructurado junto al botón de seguir. Junto a la vista de perfil se presenta un componente que se encarga de extraer todos los post o recetas que fueron creadas por el usuario y se presentan en el mismo perfil, la **Fig. 32**, se presenta la estructura del componente.

```
<div class="flex items-center">
    <h2 class="block leading-relaxed font-light text-gray-700 text-3xl">{{ userProfile.nick_name }}</h2>
    <Link v-if="userProfile.id !== $page.props.user.id" :href="'/direct-message/'+userProfile.id" class="cursor-pointer h-7 px-3 ml-3 outli
    <Link v-if="userProfile.id === $page.props.user.id" href="/user/profile" class="cursor-pointer h-7 px-3 ml-3 focus:outline-none hover:b
    <div v-else>
        <button v-if="!existsState" @click="follow" class="flex items-center ml-3 border border-blue-600 hover:bg-blue-600 hover:tex
            <span class="block">Seguir</span>
            <svg class="block h-5 w-5 pl-1" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M5 1314 4L19 7" />
            </svg>
        </button>
        <button v-else @click="unfollow" class="flex items-center ml-3 border border-blue-600 hover:bg-blue-600 hover:text-white rou
            <span class="block">Dejar de seguir</span>
            <svg class="block h-5 w-5 pl-1" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="currentColor">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M5 1314 4L19 7" />
            </svg>
        </button>
    </div>
</div>
```

Fig. 31 Sección de información principal.

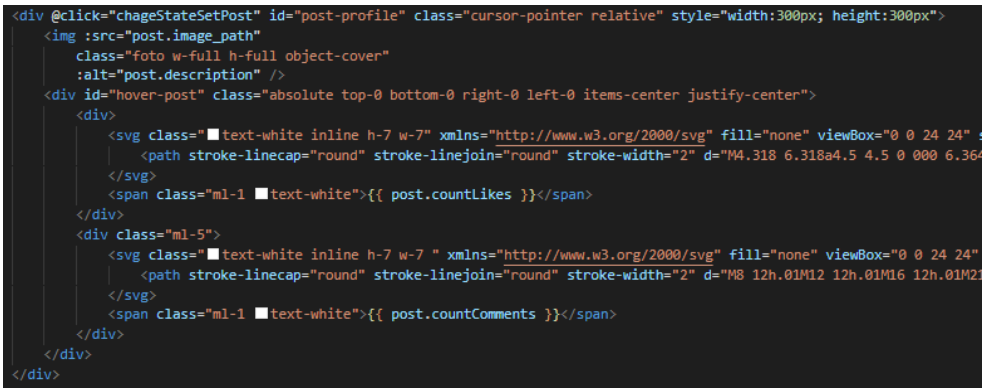


Fig. 32 Componente de carga de recetas.

Visualizar recetas

Al igual que la visualización de perfiles, las recetas son presentadas en una sección aparte, con los componentes correspondientes como lo son los *likes* y comentarios de la receta, esto se logra con el id del post el cual es enviado para su posterior consulta en la base de datos como se indica en la **Fig. 33**.

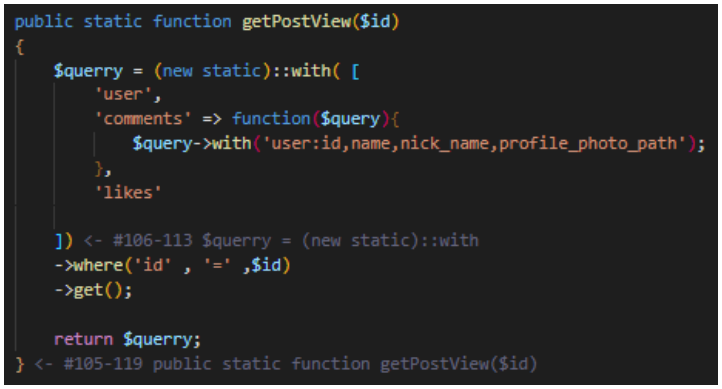


Fig. 33 Consulta de receta en la base de datos mediante el ID.

En la **Fig. 34**, hace referencia a la función del controlador de los *posts*, el cual se encarga de llevar el id del post y retorna a la vista de visualización los datos correspondientes a esa consulta, en la **Fig. 35**, se presenta la vista encargada de presentar la información.

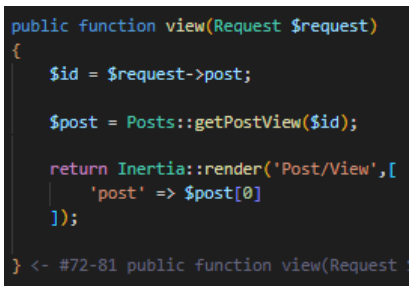


Fig. 34 Controlador de visualización de recetas.

```
    <Link  :href="'/profile/'+post.user.nick_name" class="block cursor-pointer py-4 flex items-center text-sm outline-
        <img :src="post.user.profile_photo_url"
        :alt="post.user.nick_name" class="h-8 w-8 rounded-full object-cover"/>
        <p class="block ml-2 font-bold">{{ post.user.nick_name }}</p>
        <span class="block ml-2 text-xs text-gray-600">Hace {{ getDifferenceTime(post.created_at) }}</span>
    </Link>

    <div v-if="$page.props.user.permission.includes('admin.recetas.create') || $page.props.user.id == post.user_id" >

        <button class="w-20 mx-3 mb-5 text-center bg-[#ff6060] rounded text-white py-2 outline-none focus:outline-
        <button  class="w-20 mx-3 mb-5 text-center bg-[#ff6060] rounded text-white py-2 outline-none focus:outline

    </div>



</header>
<div class="grid grid-rows-1 grid-flow-col ">
    <div>
        <h1 class="font-bold mb-3 ">{{post.title}}</h1>
        <h2 class="font-bold">Porciones</h2>
        <p> {{post.portions}}</p>
        <div class="grid grid-cols-2 ">
            <div>
                <h2 class="font-bold">Ingredientes</h2>
                <p>{{post.ingredients}}</p>
            </div>
            <div>
                <h2 class="font-bold">Preparación</h2>
                <p>{{post.description}}</p>
            </div>
```

Fig. 35 Vista de presentación de recetas.

Pruebas unitarias *Sprint* 3

El principal objetivo del *Sprint* 3 consta en la visualización de usuarios y recetas, en donde se incluye el filtro de búsqueda el cual presenta los resultados de las diferentes recetas y usuarios que se encuentren creadas con *Nick_names* o títulos similares. En la **Fig. 36**, se presenta el perfil de usuario en donde se pueden ver las recetas creadas por él y el filtro realizado en el *Sprint.*

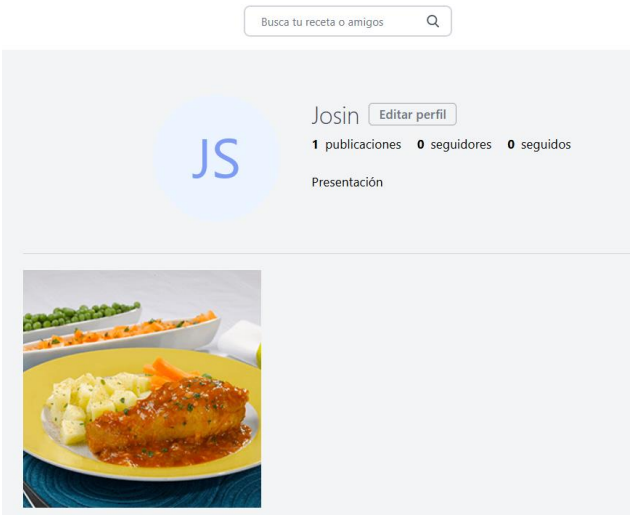


Fig. 36 Visualización de perfil

3.5 Sprint 4. Módulo de seguidores, seguidos e interacción con recetas

El *Sprint* 4, se estableció de manera que se pueda realizar la interacción con las recetas, en las cuales se incluye los comentarios y *likes*, junto a esta interacción se incluye el proceso de seguimiento de perfiles, el cual permite presentar a los usuarios las recetas recientemente creadas. Se presentan los resultados de este *Sprint* en la siguiente lista:

- Seguir perfiles.
- Visualizar cantidad de seguidores.
- Calificación de recetas.
- Comentarios.

Seguir perfiles

Cuando se trata de seguir perfiles, el mismo se realiza con el fin de obtener las nuevas recetas creadas de manera instantánea, en la Fig. 37, se presenta la función la cual permite crear un enlace entre usuario para establecer el seguimiento de perfiles, mientras que la Fig. 38, se indica la función la cual destruye esa relación cuando se quiere dejar de seguir a un usuario.

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

    $user = User::find($request->user_id);

    $user->notify(new NotifyFollow(Auth::user()));

    return $this->followers->follow($request->user_id);
} <- #57-64 public function followUser(Request $request)
```

Fig. 37 Función para seguir usuarios.

```php
public function unFollow(Request $request){
    $follow = $this->followers->where('user_id',$request->user_id)->where('follower_id','=',Auth::id())->first();

    return $follow->delete();
} <- #66-70 public function unFollow(Request $request)
```

Fig. 38 Función para dejar de seguir usuarios.

Visualizar cantidad de seguidores

Una vez establecido las funciones para el seguimiento de perfiles de usuario, es importante presentar la cantidad de seguidores que se tenga en se momento, para que los demás usuarios que visiten el perfil, los mismos puedan hacerse una idea de la popularidad y calidad de recetas que comparte. Como se muestra en la **Fig. 39**, cuando se realiza la consulta a la base de datos, se utiliza una función incorporada con *Laravel* el cual permite contar el número de elementos, devolviendo un número entero y el mismo se envía a la vista de perfil.

```
try
{
    $followers = $user->followers()->count();
    $followed = $this->followers->where('follower_id',$user->id)->count();
    $postsCount = $user->post()->count();
    $posts = Posts::getPost($user->id,true);
```

Fig. 39 Recuento de cantidad de seguidores.

Calificación de recetas

Otra función importante es la calificación de recetas, esto se logra mediante likes, que al igual que la cantidad de seguidores, se implementa principalmente para indicar si la receta les gusta a los usuarios, en la **Fig. 40**, se presenta la función para generar ese *like* y verificando al usuario perteneciente y el mismo, también verifica si existe anteriormente la interacción entre e usuario autentificado y el post.

```
public function likeOrDislike(Request $request){
    try {

        $postId = $request->post_id;
        $userId = auth()->user()->id;

        if($this->likes->where('post_id',$postId)->where('user_id',$userId)->exists()){
            $this->likes->deleteLike($postId,$userId);

            return response()->json(['like' => false,'likes' => $this->likes->where('post_id',$postId)->get()]);
        }else{
            $this->likes->like($postId,$userId);

            return response()->json(['like' => true,'likes' => $this->likes->where('post_id',$postId)->get()]);
        } <- #140-144 else
    } catch (\Exception $e) {
        return response()->json($e->getMessage(),500);
    }
} <- #130-148 public function likeOrDislike(Request $request)
```

Fig. 40 Función para dar *like* o *dislike*

Comentarios

Como última función que permite la interacción entre recetas se tiene los comentarios, el cual permite a los usuarios opinar o recomendar modificaciones en las recetas creadas por los usuarios. Como se presenta en la **Fig. 41**, se establece una función la cual permite crear un comentario en base un mensaje y el ID del usuario que se encuentra autentificado.

```php
public function store(CommentRequest $request, Posts $post, User $user)
{
    Comments::create([
     'user_id' => auth()->user()->id,
     'post_id' => $post->id,
     'comment' => $request->comment,
    ]); <- #44-48 Comments::create

    return back();
} <- #43-51 public function store(CommentRequest $request, Posts $post, U...
```

Fig. 41 Funcion para la creaccion de comentarios

Pruebas Unitarias del *Sprint 4*

Finalizado el *Sprint 4* se procede a comprobar los resultados esperados, en la **Fig. 42**, se evidencia como los comentarios y *likes* aparecen en la publicación de un usuario diferente, también se presenta como los mismo pueden eliminar el comentario en caso de que se cometiera un error a la hora de escribirla.

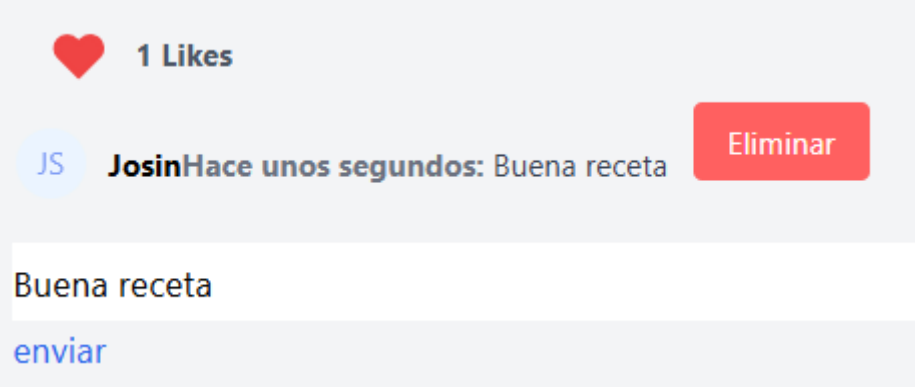


Fig. 42 Presentación de *likes* y comentarios.

3.6 Sprint 5. Módulo de notificaciones y comunicación

El *Sprint* 5, se estableció de manera que los usuarios obtengan una interacción directa entre ellos, para eso es necesario las notificaciones y el chat, en donde se pueden compartir recetas o notificar a los usuarios cuando alguien empezó a seguir su cuenta. A continuación, se listan los resultados alcanzados después de ejecutar las tareas designadas del Sprint.

- Notificaciones.
- Chat en vivo.

Notificaciones.

La sección de notificaciones permite que los usuarios visualicen las nuevas interacciones con su perfil, la cual consiste en alertar a un usuario cuando se obtiene un nuevo seguir en la página. Como se muestra en la **Fig. 43**, se utilizar *pusher* enlaces funciones para establecer un enlace de conexión con los usuarios mediante la nube, es decir, la notificación se envía una vez el usuario diera al botón se seguir.

```php
public function via($notifiable)
{
    return ['database','broadcast'];
}

public function toBroadcast($notifiable)
{


    return new BroadcastMessage([
        'date' => Carbon::now(),
        'message' => 'Comenzo a seguirte!',
        'user' => $this->user,
    ]); <- #33-37 return new BroadcastMessage
} <- #30-38 public function toBroadcast($notifiable)

public function toArray($notifiable)
{
    return [
        'message' => 'Comenzo a seguirte!',
        'user' => $this->user,
    ];
} <- #41-46 public function toArray($notifiable)
- #14-47 class NotifyFollow extends Notification
```

Fig. 43 Funciones de envio de notificacion.

Chat en vivo

Al obtener una interacción más directa entre usuario, se estableció la implementación de un chat en vivo, para que los usuarios puedan comunicarse entre sí. En la **Fig. 44**, se presenta las funciones que permiten la obtención del chat con el usuario deseado, las mismas funciones son las encargadas de redireccionar a las diferentes vistas con todos los mensajes de la conversación que se requiere.

```
public function index(){
    $chats = $this->chat->with([
        'usersent',
        'userrecive',
        'messages' => function($query){
            $query->latest();
        }
    ])->where('user_send',auth()->user()->id) <- #24-30 $chats = $this->chat->with
    ->orWhere('user_recive',auth()->user()->id)
    ->get();

    return Inertia::render('Chat/index',['chats' => $chats]);
} <- #23-35 public function index()

public function getChat($id){
    return $this->chat->with([
        'usersent',
        'userrecive',
        'messages'
    ])->where('id',$id) <- #38-42 return $this->chat->with
    ->first();
} <- #37-44 public function getChat($id)
```

Fig. 44 Controlador para él envió y recepción de mensajes

Obteniendo ya la información del chat mediante las funciones, se redirigió a la vista la cual se va a encargar de presentar los chats de los usuarios, la **Fig. 45**, muestra como es la obtención de datos de parte de la vista y el mismo se envía mediante funciones de *javascrip*.

```
var channel = pusher.subscribe('Foodgrams-channel');



channel.bind('new-message', function(data) {
    if(component.chatid === data.message.chat_id){
        component.newMessage(data.message)
    }
}); <- #221-225 channel.bind
```

Fig. 45 Funcion de envio para nuevos.

Pruebas unitarias Sprint 5

Completado el *Sprint 5*, se procede a comprobar los resultados esperados dentro del desarrollo del mismo, en la **Fig. 46**, se presenta la sección de chat, la cual se puede evidenciar él envió de mensajes en tiempo real. También se implementó un mensaje que permite visualizar cuando el usuario se encuentra escribiendo un mensaje para su envío.

Fig. 46 Visualizacion de chat en vivo

3.7 Sprint 6. Pruebas y despliegue a producción

Al finalizar la codificación de las diferentes funcionalidades del *backend*, se procede a continuar con las tareas previamente establecidas en el Sprint backlog, las cuales consisten en la sección de pruebas. Como tareas principales se tienen:

- Pruebas de carga.
- Pruebas de estrés.
- Despliegue en Amazon Web Services (AWS).

Pruebas de carga

En la **Fig. 47,** se evidencia la utilización de *postman* para la utilización de *apis*, al utilizar *larave*l como *framework* de *backend* permite la generación de rutas para la utilización de api permite obtener la información solicitada por los usuarios, en cada petición y manipulación que se realiza a la base de datos es necesario que me devuelva una respuesta de la ejecución correcta, por ello al utilizar las rutas con *postman*, muestra de mejor manera que información nos devuelve y como nos devuelve.

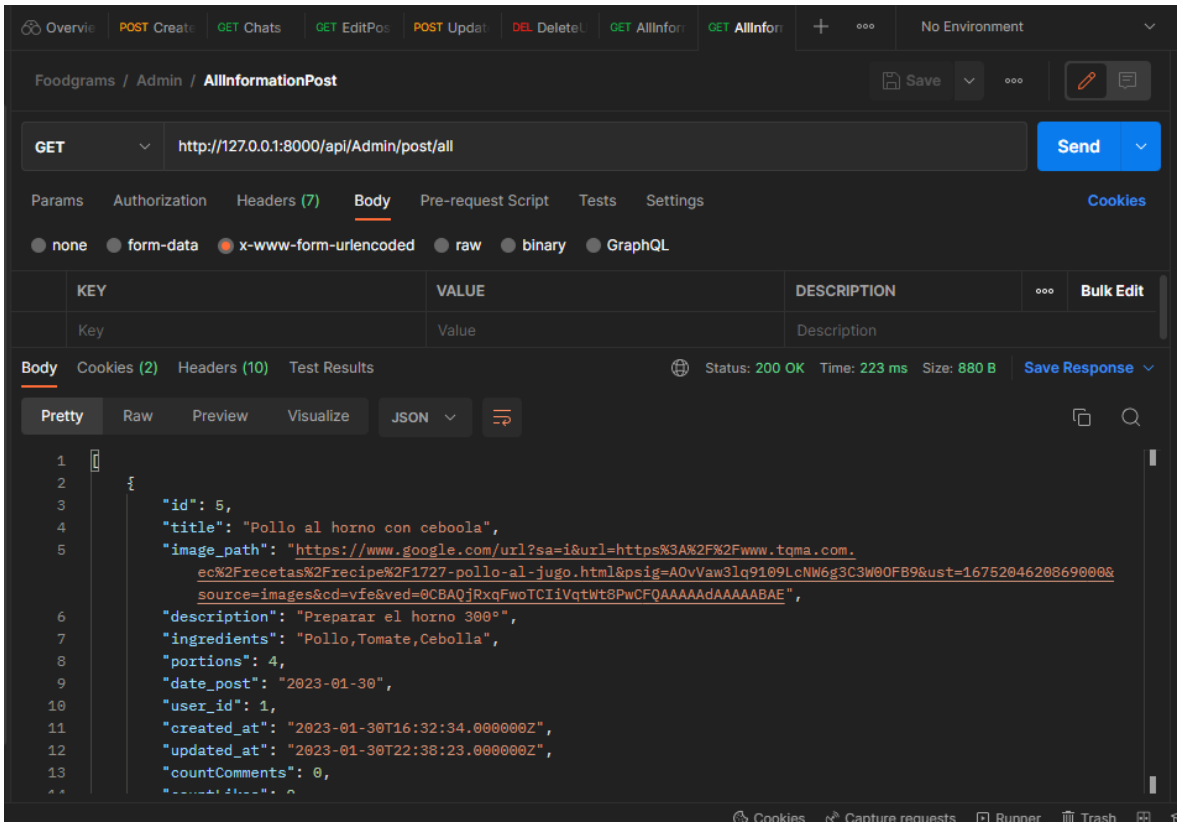


Fig. 47 Pruebas de carga

Pruebas de estrés

En las pruebas de estrés se busca verificar la capacidad de peticiones que se pueden realizar en un tiempo determinado, en la **Fig. 48**, se presenta las pruebas realizadas mediante *Jmeter*, la cual repite la solicitud indicada y nos tira el porcentaje de error en cuanto a número de solicitudes soporta. Al ser un *backend* en tiempo real necesita soportar grandes cantidades de solicitudes.

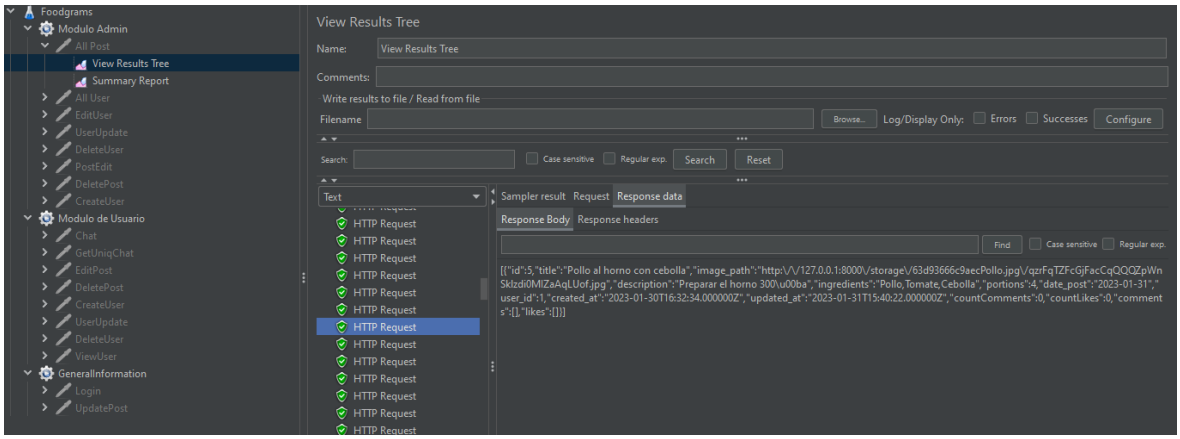


Fig. 48 Pruebas de estrés

Despliegue en Amazon Web Services (AWS)

El *deploy* del sistema a producción en AWS se realiza mediante línea de comandos, debido a la facilidad y eficacia de rendimiento que tiene la plataforma para el alojamiento de proyectos en diferentes *frameworks*, para ello es necesario tener una cuenta de AWS, una vez dentro se busca “EC2” que permite la creación de instancias, en la **Fig. 49**, se presenta la vista de instancia de nuestro hosting en la cual se puede lanzar una instancia para la carga del proyecto.

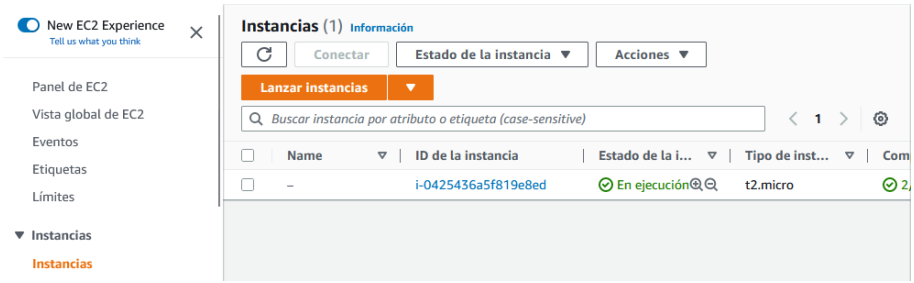


Fig. 49 Creación de instancia de ejecución.

Al lanzar una instancia, esta viene por defecto en las opciones gratuitas, pero existen más opciones gratuitas las cuales se puede modificar para facilitar el despliegue, en la **Fig. 50**, se muestra las opciones de sistemas operativos con las cuales podemos trabajar, en el caso del desarrollo del presente proyecto se trabajó con el sistema operativo Ubuntu. A continuación, se presenta unos pasos extra para asegurar la protección de la instancia, en la **Fig. 51**, se crean un par de claves para la conexión segura, esta se descargará automáticamente, y se deben colar en una carpeta segura para poder realizar la conexión

Fig. 50 Sistemas operativos para la instancia.

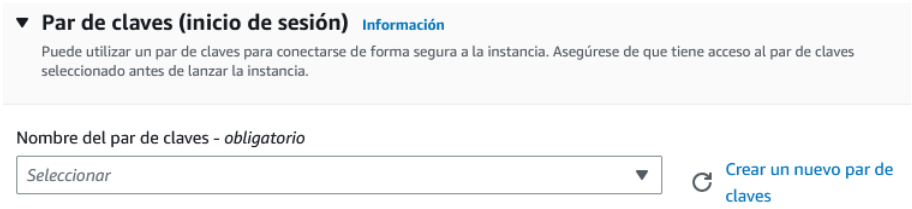


Fig. 51 Generación de claves para el ingreso a la máquina virtual.

Colocado las claves en una carpeta segura, se procede a cambiar los permisos de usuario y se limitan al perfil en el cual se esté trabajado, se logra modificando en las propiedades del archivo de permisos, en la **Fig. 52**, se presenta un ejemplo de cómo debe quedar los permisos modificados.

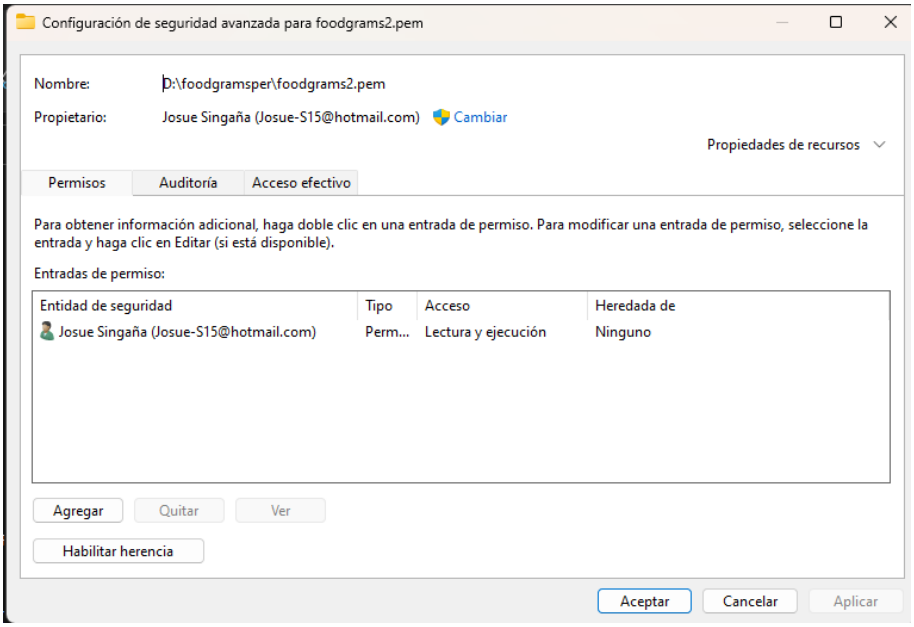


Fig. 52 Modificación de permisos en Windows.

Una vez lanzada la instancia, se procede a conectar mediante un terminal compatible, en el caso del proyecto se utiliza Git bash, en la **Fig. 53**, se presenta los pasos y comandos que recomienda el servicio para su conexión.

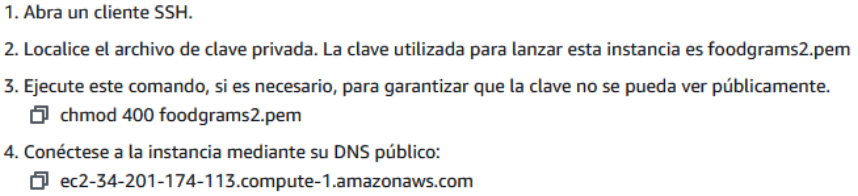


Fig. 53 Pasos para la vinculación con la instancia.

Al seguir los pasos que indica el servicio de AWS, procedemos a la instalación de los servicios que necesita el servidor para la ejecución del proyecto, las principales instalaciones que necesita tener son:

- PHP
- Apache2
- Composer
- Node.js

En la **Fig. 54**, se presentan las versiones de las herramientas instaladas y requeridas, continuando con el despliegue se procede clonar el repositorio que contiene el proyecto, siguiendo los pasos que se tiene en ANEXOS sección lV, siguiendo se procede modificar el archivo apache2.conf esto para que apunte a nuestro índex del proyecto y el mismo se ejecute de forma automática.

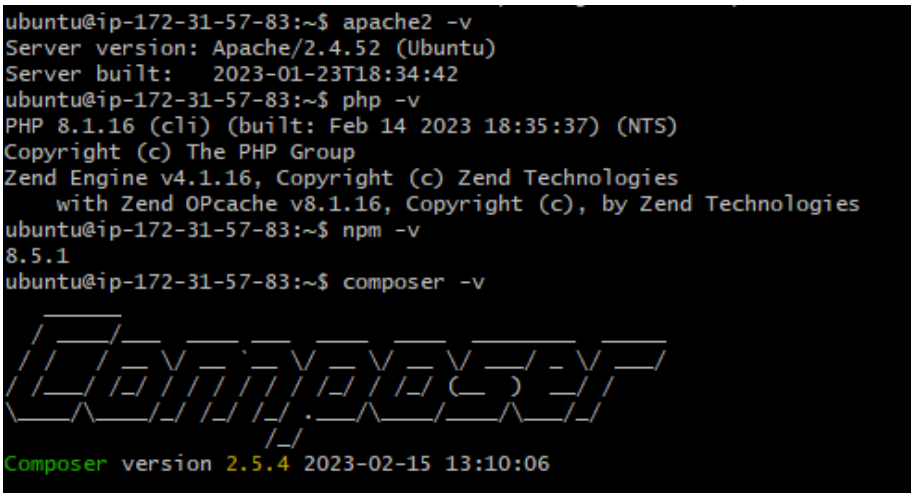


Fig. 54 Verificación de herramientas instaladas.

Finalizando se procede a verificar la ejecución del proyecto en la instancia, se procede a ingresar a la instancia que aloja el proyecto y podemos verificar toda la información de la misa, en la **Fig. 55**, se presenta la Ip publica la cual nos permite ingresar y verificar de forma rápida el funcionamiento del proyecto.

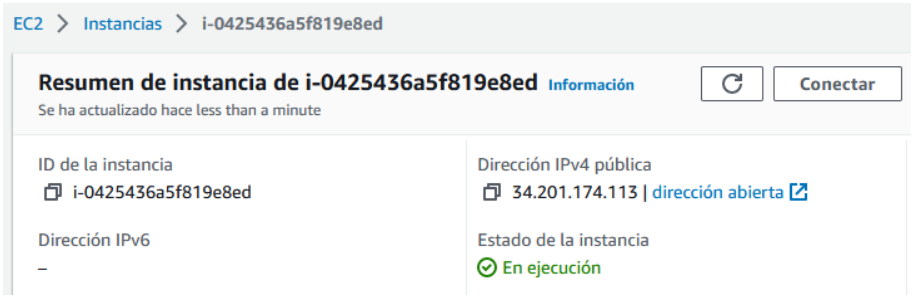


Fig. 55 IP publica para verificación de ejecución.

4 CONCLUSIONES

Una vez finalizado el desarrollo del proyecto, se presentan a continuación las conclusiones en base a los objetivos planteados previamente:

- Se completaron todos los objetivos y alcance planteados referentes al desarrollo, gestión y diseño del recetario.
- El tener en cuenta los requerimientos necesarios para el funcionamiento del *backend,* permitió el desarrollo correcto del mismo, de manera que ayuda a implementar la metodología *SCRUM* para facilitar la salida a producción.
- El diseño de la base de datos, permitió el almacenamiento correcto de la información ingresada por los usuarios, también al tener diferentes tablas ayuda a la extracción de mensajes de los diferentes usuarios, además al conectarse a un servicio especializado en la nube como *pusher* ayudo al manejo de información en tiempo real
- La implementación de los módulos establecidos, se realizó de manera correcta y en los tiempos estimados, los mismos fueron comprobados de forma individual con el fin de verificar el cumplimiento de las tareas propuestas.
- La verificación de funcionalidades se realizó de dos diferentes formas, el diseño simple de interfaces para la comprobación de carga de datos y la comprobación mediante *postman,* el cual nos devuelve de una forma más técnica la información solicitada.

5 RECOMENDACIONES

A continuación, se presentan las recomendaciones obtenidas durante el desarrollo del presente proyecto.

- Al implementar un proyecto *laravel* con *Jetstream* e *Inertia.js*, es recomendable tener un buen conocimiento de *javascript* y *php*, además revisar la documentación propia de *inertia*, debido a que algunas interacciones no se realizan de forma tradicional o a la acostumbrada.

- Antes de retornar una petición la base de datos, revisar la estructura en la cual se envía, debido a que pueden surgir inconvenientes en cuando a las estructuras de arrays u objetos y puede provocar la presentación de forma diferente en las vistas. Se recomienda revisar en cada parte de la petición.

- Al ser un proyecto que puede escalar en gran medida, revisar constantemente el estrés que soporta, con el fin de evitar lentitud en el servicio, se puede solucionar si se implementa servidores, debido al chat que se encuentra en tiempo real y existe una gran carga de solicitudes.

- Al implementar un chat en vivo o algún servicio en tiempo real, se recomienda utilizar un servicio especializado como lo es *pusher,* debido a la a la facilidad de instalación y manejo que estos tienen enfocándose en situaciones en concreto.

- Si se trabaja con email para la recuperación de contraseñas e ingreso al sistema, para hacer las comprobaciones se deben revisar los diferentes puertos compatibles y ver si los mismos se encuentran habilitados.

6 REFERENCIAS BIBLIOGRAFÍCAS

[1] K. Lucero, «Multiplica Ediciones: "El ecuatoriano transita entre la desnutrición y el sobrepeso ",» Revista Gestión, 16 Septiembre 2020. [En línea]. Available: https://acortar.link/PpfPsT. [Último acceso: 15 Mayo 2022].

[2] G. García, «UTE: Propuesta de elaboración de menus nutritivos para personas con problemas cardiovasculares que habitan en el sector norte de la ciudad de Quito",» 12 Septiembre 2018. [En línea]. Available: http://repositorio.ute.edu.ec/bitstream/123456789/11733/1/50257_1.pdf. [Último acceso: 15 Mayo 2022].

[3] C. Sanches, «Escuela de Nutriciòn: "Alimentación y su impacto en tiempos de COVID-19",» 23 julio 2020. [En línea]. Available: https://merida.anahuac.mx/noticias/alimentacion-y-su-impacto-en-tiempos-de-covid19. [Último acceso: 15 mayo 2022].

[4] S. G. Martinez y E. S. Sopeña, «Revista medica: "Importancia de una buena nutrición ante el COVID-19",» 25 Abril 2020. [En línea]. Available: https://revistamedica.com/importancia-buena-nutricion-covid-19/. [Último acceso: 16 Mayo 2022].

[5] MasterCard, «La pandemia ha acentuado el interés por la gastronomía: casi el 70% de los españoles dedica más tiempo a la cocina y ha mejorado sus habilidades culinarias,» 18 Marzo 2021. [En línea]. Available: https://acortar.link/pnvNk6. [Último acceso: 16 Mayo 2022].

[6] C. Gandarilla, «Aprende Institute: "8 beneficios de aprender a cocinar online",» [En línea]. Available: https://acortar.link/TCmjTC. [Último acceso: 16 Mayo 2022].

[7] J. Romero, «TreceBits: "6 apps para crear y guardar tus recetas de cocina",» 06 Mayo 2020. [En línea]. Available: https://www.trecebits.com/2020/05/06/6-apps-para-crear-y-guardar-tus-recetas-de-cocina/. [Último acceso: 30 Mayo 2022].

[8] Cuideo, «Coronavirus: Los alimentos que ayudan en nuestra recuperación,» [En línea]. Available: https://cuideo.com/blog/coronavirus-alimentos-recuperacion/. [Último acceso: 15 Mayo 2022].

[9] «Teixido: "Talleres de Cocina a Teixidó",» [En línea]. Available: https://www.angelteixido.com/es/la-importancia-de-saber-cocinar/. [Último acceso: 16 Mayo 2022].

[10] Santander Universidades, «Metodologías de desarrollo de software: ¿qué son?,» 27 Diciembre 2020. [En línea]. Available: https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html. [Último acceso: 16 Mayo 2022].

[11] R. Arjonilla, «BackEnd,» [En línea]. Available: https://rafarjonilla.com/que-es/backend/. [Último acceso: 16 Mayo 2022].

[12] RootStack, «nertia.js y sus ventajas para los desarrolladores de Laravel,» 20 Diciembre 2021. [En línea]. Available: https://www.rootstack.com/es/blog/laravel-inertia/. [Último acceso: 9 Junio 2022].

[13] Quality Devs, «Qué es Laravel y para qué sirve,» 23 Junio 2021. [En línea]. Available: https://www.qualitydevs.com/2021/06/23/que-es-laravel/. [Último acceso: 9 Junio 2022].

[14] «Proyectos agiles: "Qué es SCRUM",» [En línea]. Available: https://proyectosagiles.org/que-es-scrum/. [Último acceso: 30 Mayo 2022].

[15] «Proyectum: "Los tres principales roles en Scrum",» 19 Octubre 2016. [En línea]. Available: https://acortar.link/UFPlTE. [Último acceso: 30 Mayo 2022].

[16] «ViewNext: "Artefactos Scrum ¿Qué son y para qué sirven?",» [En línea]. Available: https://www.viewnext.com/artefactos-scrum/. [Último acceso: 30 Mayo 2022].

[17] «PMOinformatica.com: "7 Técnicas de levantamiento de requerimientos software",» 7 Agosto 2016. [En línea]. Available: http://www.pmoinformatica.com/2016/08/tecnicas-levantamiento-requerimientos.html. [Último acceso: 30 Mayo 2022].

[18] M. Rehkopf, «Atlassian: "Historias de usuario con ejemplos y plantilla ",» [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: 30 Mayo 2022].

[19] M. Alvarez, «desarrolloweb.com: "Qué es MVC",» 28 julio 2020. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html. [Último acceso: 8 Junio 2022].

[20] Aitana Soluciones, «Aitana: "Visual Studio Code: Funcionalidades y extensiones",» 16 Octubre 2018. [En línea]. Available: https://acortar.link/z6nZQA. [Último acceso: 9 Junio 2022].

[21] TechTarget, «ComputerWeekly: "MySQL",» Abril 2021. [En línea]. Available: https://www.computerweekly.com/es/definicion/MySQL. [Último acceso: 9 Junio 2022].

[22] DesarrolloWeb, «Laravel,» [En línea]. Available: https://desarrolloweb.com/home/laravel. [Último acceso: 9 Junio 2022].

[23] E. García, «códigofacilito: "¿Qué es Vue.JS?",» 1 abril 2019. [En línea]. Available: https://codigofacilito.com/articulos/que-es-vue. [Último acceso: 9 Junio 2022].

[24] G. Chávez, «¿Qué es Laravel Jetstream?,» [En línea]. Available: https://gabrielchavez.me/que-es-laravel-jetstream/. [Último acceso: 9 Junio 2022].

[25] T. Rodríguez, «Genbeta: "Pusher, servicio en la nube para gestionar las conexiones y envío de mensajes mediante Websockets ",» 31 Marzo 2013. [En línea]. Available: https://acortar.link/pY3fv0. [Último acceso: 9 Junio 2022].

[26] M. Garcia, «Nettix: "¿Qué es Xampp y Comó puedo usarlo?",» 30 Mayo 2020. [En línea]. Available: https://www.nettix.com.pe/blog/web-blog/que-es-xampp-y-como-puedo-usarlo/. [Último acceso: 9 Junio 2022].

[27] L. Holgado, «atsistemas: "Qué es Tailwind y por qué usarlo",» 21 Enero 2021. [En línea]. Available: https://www.atsistemas.com/es/blog/que-es-tailwind. [Último acceso: 9 Junio 2022].

[28] Yair, «Styde "¿Qué es Composer y cómo usarlo",» 23 Diciembre 2019. [En línea]. Available: https://styde.net/que-es-composer-y-como-usarlo/. [Último acceso: 19 Enero 2023].

[29] MADEJA, «Desarrollo de la junta de Andalucía: "Introducción a JMeter: Conceptos Básicos ",» [En línea]. Available: https://www.juntadeandalucia.es/servicios/madeja/printpdf/872. [Último acceso: 14 Junio 2022].

[30] Y. Muradas, «OpenWebinar: " Qué es Postman y primeros pasos",» 03 Junio 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-postman/. [Último acceso: 31 Enero 2023].

7 ANEXOS

A continuación, se presentan los anexos mencionados en el proceso de desarrollo del presente proyecto.

- Certificado de autenticidad de trabajo de integración curricular.
- Manual técnico.
- Manual de usuario.
- Manual de instalación.

ANEXO I

Finalizado el desarrollo, se procede a comprobar la autenticidad del presente proyecto, para ello se utiliza la herramienta de Turnitin el cual analiza y nos lanza un reporte de las secciones que tienen problemas por plagio.

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 23 de Febrero de 2023

De mi consideración:

Yo, MAYRA ISABEL ALVAREZ JIMÉNEZ, en calidad de directora del Trabajo de Integración Curricular componente **BACKEND**, titulado "**DESARROLLO DE APLICACIÓN WEB Y MÓVIL PARA GESTIÓN DE RECETAS DE COCINA**", elaborado por el estudiante **JOSUE ALEXANDER SINGAÑA VERGARA** de la carrera en DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, 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,

Ing. Mayra Alvarez MSc.
Técnico docente EPN - ESFOT

ANEXO II

Al utilizar la metodología *SCRUM* se deben tener en cuenta la utilización de sus artefactos como lo son: *Sprint Backlog, Product Backlog*, interfaces de usuario y pruebas realizadas. Juntos a los artefactos mencionados se encuentran también complementos a la información presentada en el documento. En esta sección se recopila la información correspondiente a cada artefacto.

Recopilación de Requerimientos

El primer artefacto que se define para la metodología scrum, es la recopilación de requerimientos, para el presente proyecto en **TABLA X**, se presenta todos los requerimientos obtenidos previos al desarrollo.

TABLA X Recopilacion de requerimeintos

RECOPILACION DE REQUERIMIENTOS			
ID-RR	Nombre del Requerimiento	ENUNCIADO DEL ITEM	Usuario
RR01	Iniciar sesión	Como usuario registrado: • Necesito tener un perfil para tener acceso a las funcionalidades del *backend*. Como usuario administrador: • Necesito tener un perfil para iniciar sesión y realizar la gestión del *backend*.	Registrado/Administrador
RR02	Gestionar recetas	Como usuario registrado: • Necesito poder crear, editar y eliminar las recetas que yo añada al *backend*. Como usuario administrador: • Necesito poder crear, editar y eliminar todas las recetas que se encuentren dentro del b*ackend.*	Registrado/Administrador

RR03	Registrar usuarios	Como usuario invitado: • Necesito poder crear un perfil para tener accesos a las funcionalidades del *backend*. Como usuario administrador: • Necesito poder registrar a nuevos usuarios.	Invitado/administrador
RR04	Filtrar recetas y usuarios	Como usuario registrado y administrador: • Necesito poder realizar la búsqueda de recetas y usuarios dentro del *backend*.	Registrado/Administrador
RR05	Seguir perfiles	Como usuario registrado y administrador: • Necesito una opción que me permita seguir los diferentes perfiles para que aparezcan los nuevos posts en mi perfil.	Registrado/Administrador
RR06	Presentar post de recetas	Como usuario registrado y administrador: • Necesito visualizar las diferentes publicaciones en mi pantalla principal para no perderme de las nuevas recetas.	Registrado/Administrador
RR07	Visualizar y modificar datos de perfil	Como usuario registrado: • Necesito modificar mi información personal para mantener la información actualizada. Como usuario administrador: • Necesito visualizar la información de los perfiles y	Registrado/Administrador

		poder modificar los datos en caso de que lo requiera.	
RR08	Visualizar perfil de usuarios	Como usuario registrado y administrador: • Necesito ver mi perfil personal y de otros usuarios con el fin de poder ver las diferentes recetas creadas por los mismos.	Registrado/Administrador
RR09	Visualizar recetas	Como usuario registrado y administrador: • Necesito visualizar mis recetas y de otros usuarios para poder ver la información en cada una de ellas.	Registrado/Administrador
RR10	Visualizar cantidad seguidores	Como usuario registrado y administrador • Necesito ver mi número de seguidores para interpretar el número de personas que les interesa mis recetas.	Registrado/Administrador
RR11	Visualizar notificaciones	Como usuario registrado y administrador: • Necesito ver las notificaciones para saber cuándo una persona le interesa mis recetas.	Registrado/Administrador
RR12	Visualizar comentarios	Como usuario registrado y administrador: • Necesito comentar las recetas para poder realizar una retro alimentación de las recetas.	Registrado/Administrador

RR13	Reaccionar recetas	Como usuario registrado y administrador: • Necesito indicar si me gusta una receta con el fin de saber que tan popular es una receta.	Registrado/Administrador
RR14	Implementar chat en vivo	Como usuario registrado y administrador: • Necesito enviar y recibir mensajes, para comunicarme con otros usuarios y compartir recetas.	Registrado/Administrador
RR15	Probar y desplegar el *backend*	La verificación del correcto funcionamiento del componente *backend* es verificada mediante la realización de pruebas y despliegue del sistema.	Todos los usuarios

Historias de Usuario

El siguiente artefacto son las historias de usuario, las cuales se basan en los requerimientos obtenidos y los mismo se listan de la **TABLA XI** a la **TABLA XXIV**, en donde se especifica la función que se debe cumplir de los requerimientos.

TABLA XI Getionar recetas

Historia de usuario	
Identificador (ID): HU02	**Usuario**: Registrado/Administrador
Nombre historia: Gestionar recetas	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Media
Iteración asiganada:2	
Responsable: Josue Singaña	
Descripción: Los usuarios tendrán la posibilidad de crear, editar y eliminar recetas, con la diferencia de que el administrador podrá gestionar las recetas de todo el sitio y el usuario registrado realizar la gestión de sus recetas pertenecientes.	

Objetivo: Realizar la creación, edición y eliminación de recetas dentro de la base de datos según los permisos de cada usuario.

TABLA XII Registrar usuarios

Historia de usuario	
Identificador (ID): HU03	**Usuario**: Invitado/Administrador
Nombre historia: Registrar usuarios	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Baja
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: El usuario invitado podrá registrarse dentro del sistema con sus datos, para posteriormente acceder a las funcionalidades del sistema. El usuario administrador tendrá la posibilidad de crear nuevos usuarios desde su perfil.	
Objetivo: Presentar un formulario donde el usuario que no se encuentre registrado, pueda hacerlo para acceder a las funcionalidades del sistema.	

TABLA XIII Filtrar recetas y usuarios

Historia de usuario	
Identificador (ID): HU04	**Usuario**: Registrado/Administrador
Nombre historia: Filtrar recetas y usuarios	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Media
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: El usuario con acceso al *backend* podrá filtrar los títulos de las recetas y los *nick_name* de los usuarios, para desplegar una lista con información acorde al filtro realizado.	
Objetivo: Filtrar el usuario o receta que él requiera con el fin de ver la información correspondiente.	

TABLA XIV Seguir perfiles

Historia de usuario	
Identificador (ID): HU05	**Usuario**: Registrado/Administrador
Nombre historia: Seguir perfiles	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Alta

Iteración asiganada:3
Responsable: Josue Singaña
Descripción: Los usuarios dentro del sistema, realizaran el seguimiento a los perfiles de su preferencia.
Objetivo: Incluir la opción de seguir perfiles de su preferencia.

TABLA XV Presentar post de recetas

Historia de usuario	
Identificador (ID): HU06	**Usuario**: Registrado/Administrador
Nombre historia: Presentar post de recetas	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Media
Iteración asiganada:2	
Responsable: Josue Singaña	
Descripción: Los usuarios, visualizaran las publicaciones que realicen los perfiles que este siguiendo en su pantalla principal.	
Objetivo: Visualizar en su inicio las recetas del perfil simpatizado en orden de creación de los usuarios.	

TABLA XVI Visualizar y modificar datos de perfil

Historia de usuario	
Identificador (ID): HU07	**Usuario**: Registrado/Administrador
Nombre historia: Visualizar y modificar datos de perfil	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Baja
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: El usuario registrado tendrá acceso a su perfil en donde podrá visualizar su información personal y de ser el caso modificará sus datos personales. El usuario administrador, tendrá la posibilidad de visualizar y editar la información de los usuarios dentro del backend.	
Objetivo: Editar la información del perfil según el rol y permisos.	

TABLA XVII Visualizar perfil de usuarios

Historia de usuario	
Identificador (ID): HU08	**Usuario**: Registrado/Administrador
Nombre historia: Visualizar perfil de usuarios	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Media
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema podrán ver los perfiles de los usuarios junto a su información y recientes post de recetas.	
Objetivo: Presentar los perfiles de usuario junto a la información del perfil y recetas.	

TABLA XVIII Visualizar recetas

Historia de usuario	
Identificador (ID): HU09	**Usuario**: Registrado
Nombre historia: Visualizar recetas	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Media
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema visualizaran las recetas, para facilitar encontrar información se implementa filtros de búsqueda junto de recetas.	
Objetivo: Visualizar la información de recetas.	

TABLA XIX Visualizar cantidad seguidores

Historia de usuario	
Identificador (ID): HU10	**Usuario**: Registrado/Administrador
Nombre historia: Visualizar cantidad seguidores	
Prioridad en negocio: Baja	**Riesgo de desarrollo**: Media
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema podrán visualizar la cantidad de seguidores que tenga en ese momento.	
Objetivo: Visualizar los el número de seguidores del momento.	

TABLA XX Visualizar notificaciones

Historia de usuario	
Identificador (ID): HU11	**Usuario**: Registrado/Administrador
Nombre historia: Visualizar notificaciones	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Media
Iteración asiganada:2	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema recibirán notificaciones de nuevos seguidores en tiempo real.	
Objetivo: Presentar la interacción de diferentes usuarios hacia su perfil.	

TABLA XXI Realizar comentarios

Historia de usuario	
Identificador (ID): HU12	**Usuario**: Registrado/Administrador
Nombre historia: Realizar comentarios	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Alta
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema realizaran comentarios a las recetas de otros usuarios y podrá visualizar los comentarios que se realicen a sus recetas.	
Objetivo: Comentar los diferentes posts de recetas de diferentes usuarios y comentar sus posts propios.	

TABLA XXII Reaccionar recetas

Historia de usuario	
Identificador (ID): HU13	**Usuario**: Registrado/Administrador
Nombre historia: Reaccionar recetas	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Baja
Iteración asiganada:1	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema indicaran si la receta vista es de su agrado.	
Objetivo: Calificar las recetas de cada usuario deseado	

TABLA XXIII Implementar chat en vivo

Historia de usuario	
Identificador (ID): HU14	**Usuario**: Registrado/Administrador
Nombre historia: Implementar chat en vivo	
Prioridad en negocio: Media	**Riesgo de desarrollo**: Alta
Iteración asiganada:3	
Responsable: Josue Singaña	
Descripción: Los usuarios dentro del sistema podrán interactuar entre usuarios mediante un chat, en donde podrán enviar archivos fotos o mensajes.	
Objetivo: Realizar la interacción entre usuarios mediante un chat en vivo.	

TABLA XXIV Probar y desplegar el backend

Historia de usuario	
Identificador (ID): HU17	**Usuario**: Todos
Nombre historia: Probar y desplegar el *backend*	
Prioridad en negocio: Alta	**Riesgo de desarrollo**: Alta
Iteración asignada: 3	
Responsable: Josue Singaña	
Descripción: Es necesario realizar la verificación del funcionamiento del sistema acorde a requerimientos obtenidos, mediante la realización de pruebas y despliegue del componente desarrollado.	
Objetivo: Verificar la integración de y funcionamiento del componente desarrollado.	

Product Backlog

El siguiente artefacto se presenta en la **TABLA XXV**, la cual nos indica el estado de las historias de usuario, en donde se especifica si las mismas fueron finalizadas o se encuentran pendiente.

TABLA XXV Product Backlog

ELABORACIÓN DE HISTORIAS DE USUARIO				
ID-HU	HISTORIA DE USUARIO	ITERACION	ESTADO	PRIORIDAD
HU01	Iniciar sesión	1	Finalizado	Alta

HU02	Gestionar recetas	2	Finalizado	Alta
HU03	Registrar usuarios	1	Finalizado	Alta
HU04	Filtrar recetas y usuarios	1	Finalizado	Alta
HU05	Seguir perfiles	1	Finalizado	Alta
HU06	Presentar post de recetas	2	Finalizado	Media
HU07	Visualizar y modificar datos del perfil	1	Finalizado	Media
HU08	Visualizar perfil de usuarios	1	Finalizado	Media
HU09	Visualizar recetas	1	Finalizado	Media
HU10	Visualizar cantidad de seguidores	1	Finalizado	Baja
HU11	Visualizar notificaciones	2	Finalizado	Media
HU12	Realizar Comentarios	1	Finalizado	Media
HU13	Reaccionar recetas	1	Finalizado	Media
HU14	Implementar chat en vivo	3	Finalizado	Media
HU15	Probar y desplegar el sistema	1,2, 3	Finalizado	Alto

Sprint Backlog

Como último artefacto que se presenta, el Sprint Backlog el cual se presenta en **TABLA XXVI**, aquí se especifica las tareas y requerimientos que se van a cumplir en cada Sprint indicado para el presente proyecto.

TABLA XXVI Sprint Backlog

ELABORACIÓN DE SPRINT BACKLOG					
ID-SB	NOMBRE	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO EN HORAS
SB00	Configurar el entorno de desarrollo	N/A	N/A	• Instalación y actualización de herramientas de desarrollo	10
				• Diseño y creación de la BDD	
				• Creación del proyecto en *jetstream Inertia.js*	
SB01	Módulo de gestión de usuarios e inicio de sesión	HU03	Registro de usuario	• Crear un formulario para el registro de usuario	30
				• Validar la información previa al registro al BD	
				• Implementar el usuario en el BD	
		HU07	Modificación de datos del perfil	• Implementar un menú secundario en donde el usuario puede cerrar sesión o acceder a su perfil	
				• Implementar una interfaz en donde permita modificar la información personal del usuario	
				• Verificar que los cambiones se realicen en la interfaz del programa	

				• Validar los campos para posteriormente almacenarlos en la BD	
				• Implementar una interfaz en donde se presente la información del usuario	
				• Implementar una función de regreso a al *dashboard* desde la interfaz perfil	
		HU01	Iniciar sesión	• Validar los campos de entidad de usuario de la BD	
				• Crear usuario administrativo con sus credenciales	
				• Implementar formulario de inicio de sesión	
				• Verificar si el usuario puede acceder a las funcionalidades del sistema	
SB06	Módulo de gestión de recetas	HU02	Gestión de recetas	• Implementar un formulario para la creación de recetas, para posteriormente almacenarlas en la BD	40
				• Permitir guardar la información registrada para la regresa	
				• Presentar la receta en el perfil del usuario creador	
				• Implementar una sección en donde se presenten las recetas creadas	
				• Implementar un acceso para modificar la información de una receta propia	
				• Validar los cambios de la receta, para realizar los cambios en la BD	
				• Guardar los cambios realizados en la receta	

				• Implementar un botón para eliminar la receta del perfil personal	
				• Eliminar la receta de la base de datos	
				• Verificar si la receta eliminada se deja de presentar en la sección de recetas	
		HU06	Presentación de post recetas	• Implementar un recuadro de resumen para su presentación	
				• Verificar si la receta se presenta en el *dashboard* personal o seguidores	
SB03	Módulo de búsqueda y visualización de perfiles y recetas	HU04	Filtrar recetas y usuarios	• Implementar una barra de búsqueda	40
				• Validar los campos de búsqueda de usuarios y recetas para consultar en la base de datos	
				• Presentar una lista de los perfiles y recetas similares	
		HU08	Visualizar perfil de usuarios	• Verificar si se puede acceder al perfil de usuario que se requiera por la búsqueda	
				• Verificar que ningún usuario modifique la información de otro usuario	
		HU09	Visualizar recetas	• Implementar una interfaz en donde se pueda visualizar de mejor manera la receta junto a su información y el creador de la receta	

				• Verificar que ningún usuario exacto el creador y administrador puedan modificar la receta	
				• Implementar una sección para la presentación de me gusta y comentarios	
SB04	Módulo de seguidores, seguidos e interacción con recetas	HU05	Seguir perfiles	• Implementar un botón que permita seguir los perfiles	40
				• Validar los campos de seguimiento para realizar la implementación en la BD	
				• Indicar si el usuario ya se encuentra siguiendo ese perfil	
				• Verificar que las publicaciones del perfil seguido se presenten en el Dashboard	
		HU10	Visualizar cantidad de seguidores	• Implementar un apartado en donde se presente el número de seguidores del momento	
		HU13	Calificación de recetas	• Implementar un botón para indicar si al usuario le gusto la receta	
				• Validar los campos y almacenar la cantidad de me gusta en la BD	
				• Permitir quitar el me gusta de una receta y actualizar la BD con la nueva información.	

		HU12	Comentarios	• Implementar una sección en las recetas de los usuarios para realizar comentarios de la mismas	
				• Validar los campos del comentario y registrarlo en la BD	
				• Verificar si el comentario se presenta de forma correcta en las recetas	
SB05	Módulo de notificaciones y comunicación	HU11	Notificaciones	• Imprentar una sección en el *dashboar* para las notificaciones	30
				• Avisar cuando llegue una notificación al perfil del usuario	
				• Desplegar una lista con las notificaciones recientes y el tipo de notificación	
		HU14	Chat en vivo	• Implementar una interfaz para presentar los chats que se tenga con los diferentes usuarios	
				• Permitir él envió de archivos, fotos o mensajes a través del chat con el usuario	
				• Validar los campos de envío para almacenar en la base de datos.	
SB06	Pruebas y despliegue a producción	HU15	Pruebas y despliegue del *backend*	• Realizar las pruebas correspondientes para el *backend*	20
				• Desplegar el proyecto *backend* a producción	
				• Verificar el funcionamiento de cada apartado del proyecto	
Documentación					30

Total, de Horas	240

Estructura de base de datos

Al tener en cuenta toda la información que se debe almacenar en la base de datos se procede a presentar en la **Fig. 56.**La estructura final de la base de datos con todos sus enlaces realizados para un correcto flujo de la información.

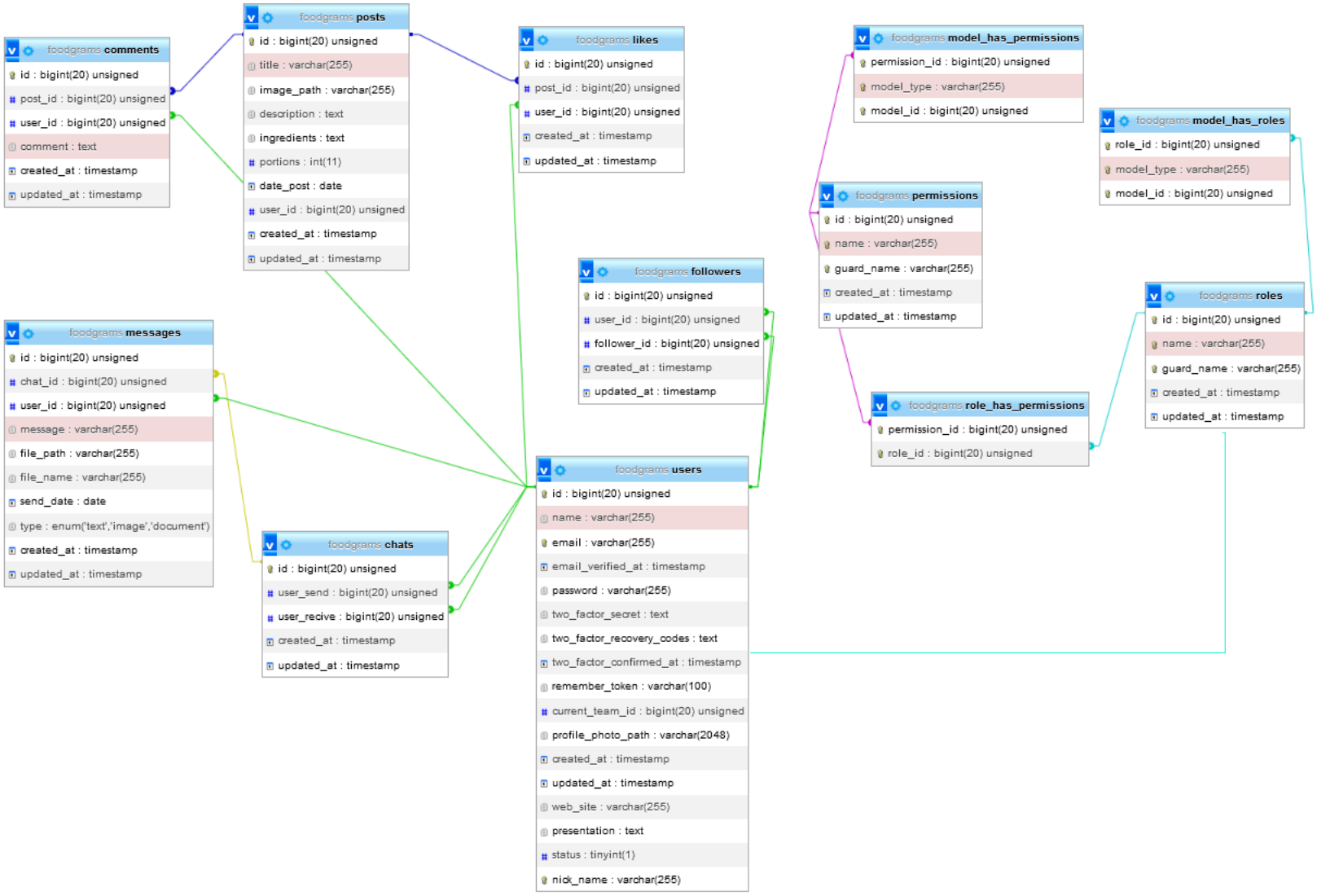


Fig. 56 Estructura de base de datos

Pruebas de carga

Al realizar las pruebas de carga, se presenta principalmente las dos funciones principales de administrador, la **Fig. 57**, y la **Fig. 58**, nos presenta todos los datos que se encuentra el sistema como lo es el llamo de todos los usuarios y el llamado a todos los posts recetas para su posterior revisión o *crud*.

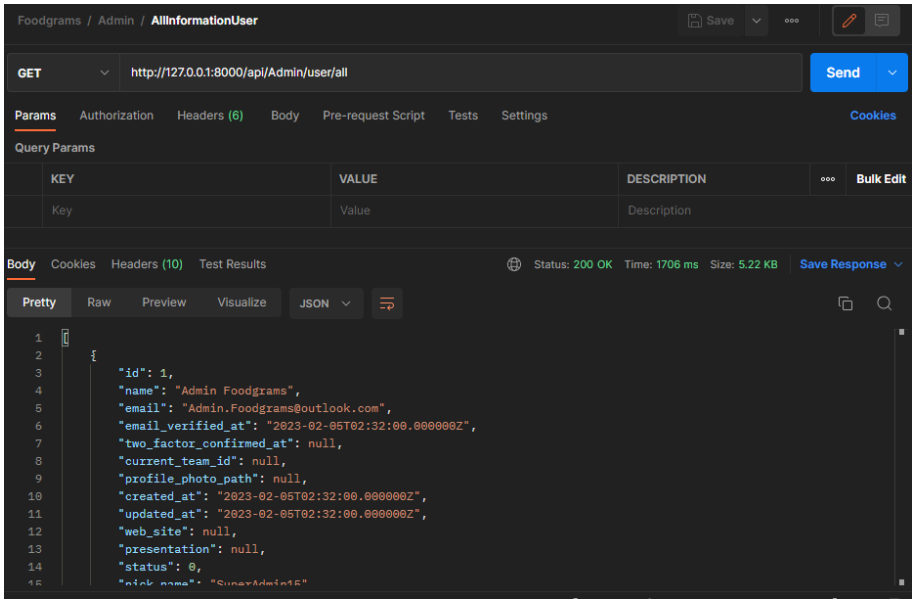


Fig. 57 Llamada de todos los usuarios registrador.

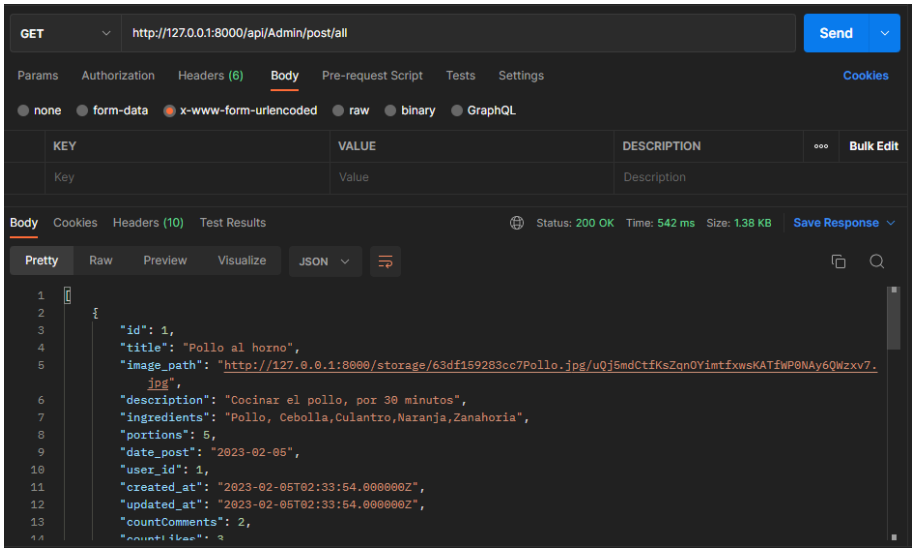


Fig. 58 Llamada de toso los post o recetas ingresadas.

Al igual que el Admin puede realizar la gestión de usuarios, los mismos usuarios utilizan funciones similares para realizar la edición de sus datos personales, la única diferencia es la protección de rutas y de accesos que tiene unos y otros. La **Fig. 59**, muestra la

información de un perfil en específico, la **Fig. 60,** presenta funcionalidad de actualizar datos, la **Fig. 61**, nos indica la función de eliminar usuario y por último la **Fig. 62**, presenta la función de creación de un usuario.

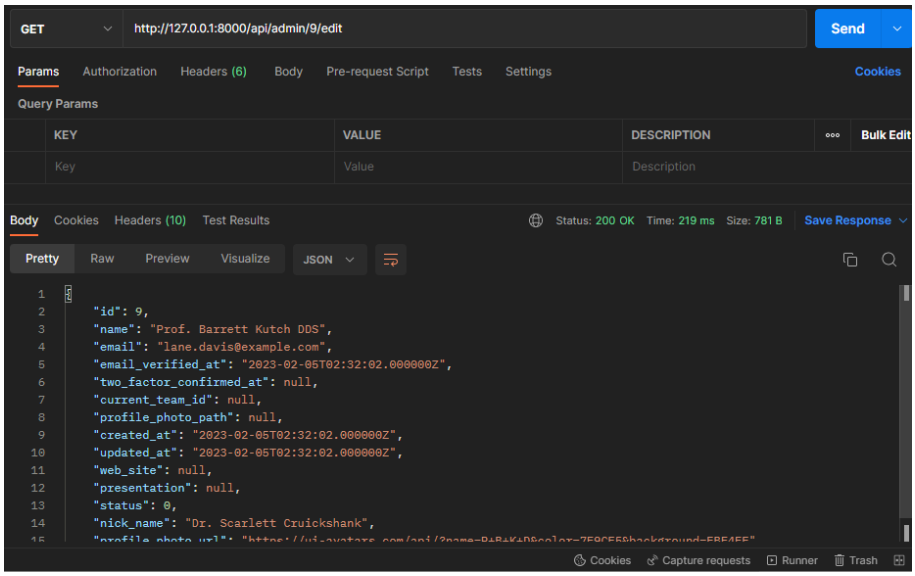


Fig. 59 Llamada de la información de un usuario por postman.

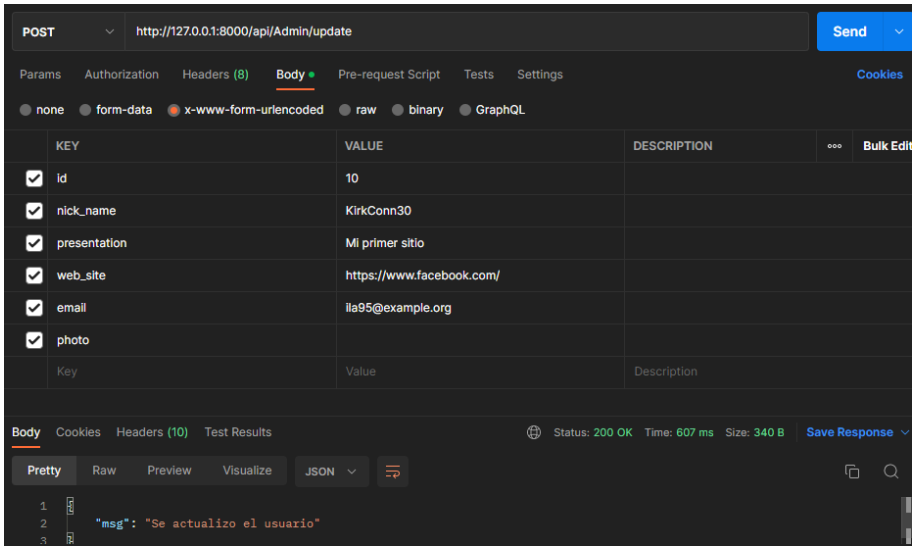


Fig. 60 Comprobación de la función de actualización de datos de usuario.

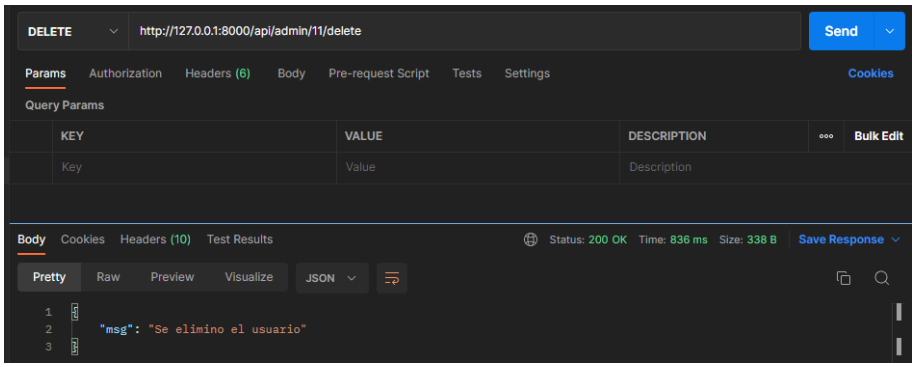


Fig. 61 Comprobación de función para la eliminación de usuario.

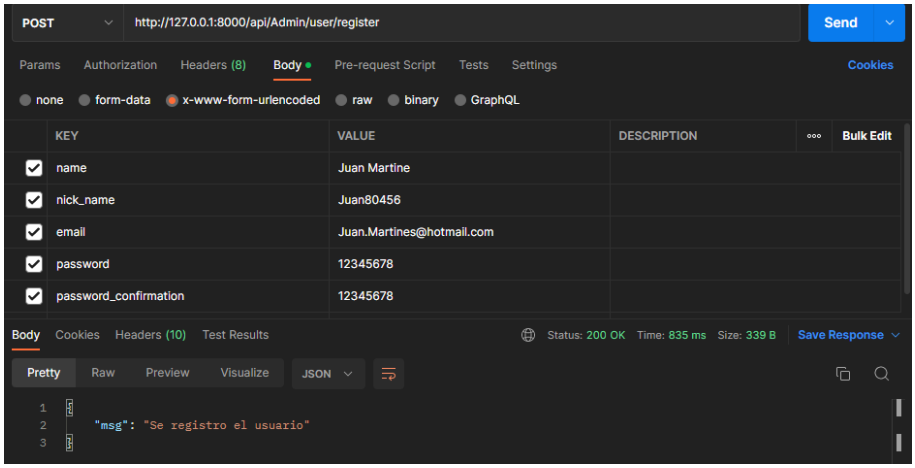


Fig. 62 Comprobación de la función para la creación de usuarios.

Similar al *CRUD* de los usuarios, los posts recetas comparten funcionalidades similares entre usuarios, que al igual poseen rutas diferentes para poder acceder a cada una de ella, dependiendo si es el Admin o un creador. La **Fig. 63**, nos trae toda la información de un post receta, mediante la el id del post y del usuario, la **Fig. 64**, permite realizar la actualización de los datos de post que vamos a editar, la **Fig. 65**, nos permite realizar la eliminación de todo el post recetas y al final la **Fig. 66**, muestra la creación de post, que se encuentra habilitado en para que cualquier usuario pueda realizarlo.

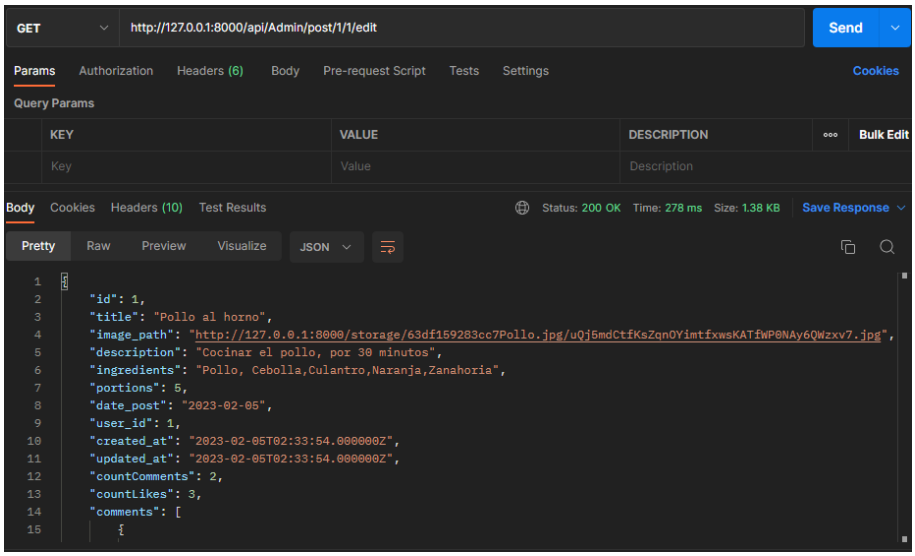


Fig. 63 Comprobación de llamada de datos post recetas.

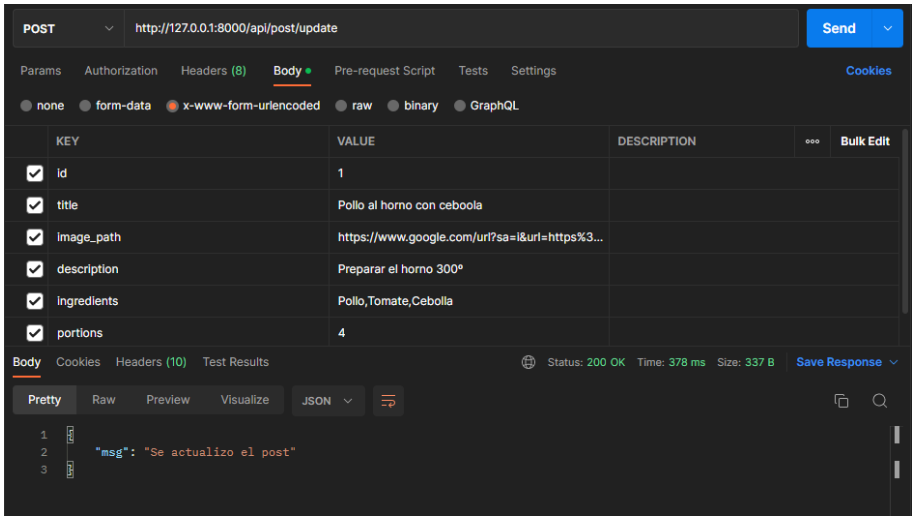


Fig. 64 Comprobación de actualización de recetas.

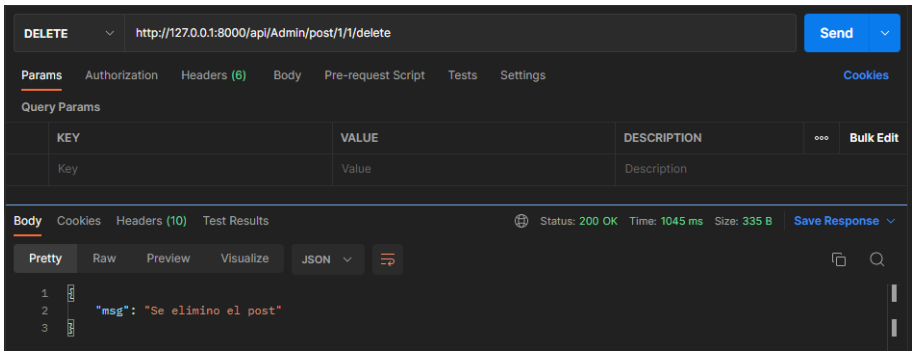


Fig. 65 Comprobación de post de elimiancion de recetas.

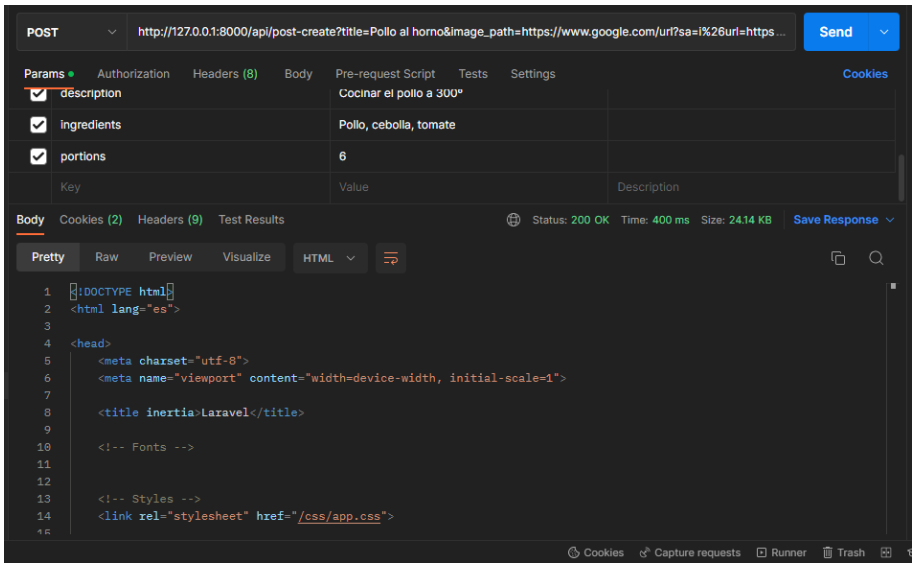


Fig. 66 Comprobación de creación de un post recetas.

Finalizando con las pruebas de carga se procede a comprobar el uso del chat en vivo, para ello se realiza la llamada de los mensajes y chat de la base de datos, y el chat en vivo se realiza mediante interfaces para comprobar el funcionamiento, la **Fig. 67**, nos trae todo los chat que se encuentran creado entre los usuarios, mientras que la **Fig. 68**, no presenta un chat creado junto a sus mensajes, al finalizar la **Fig. 69**, presenta el funcionamiento del chat en vivo mediante la interfaz creada.

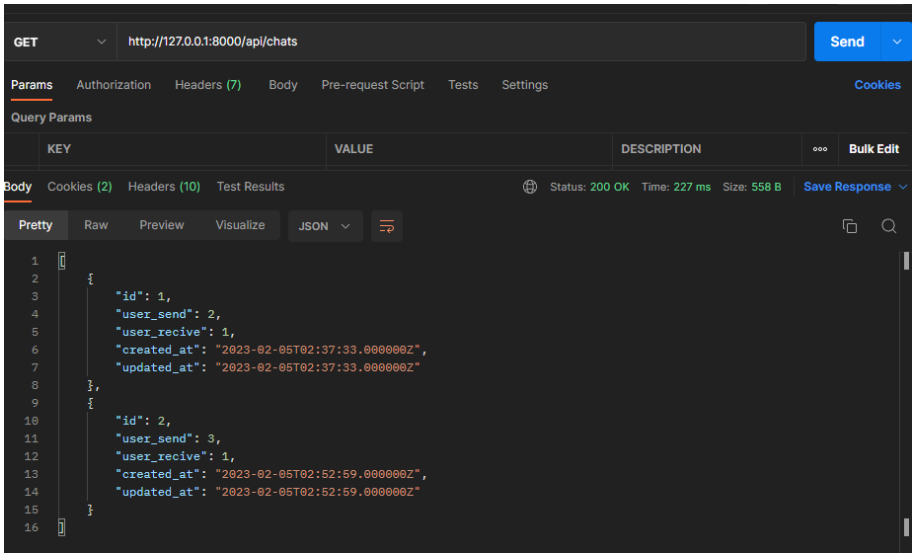


Fig. 67 Comprobación de llamado de enlaces entre chats.

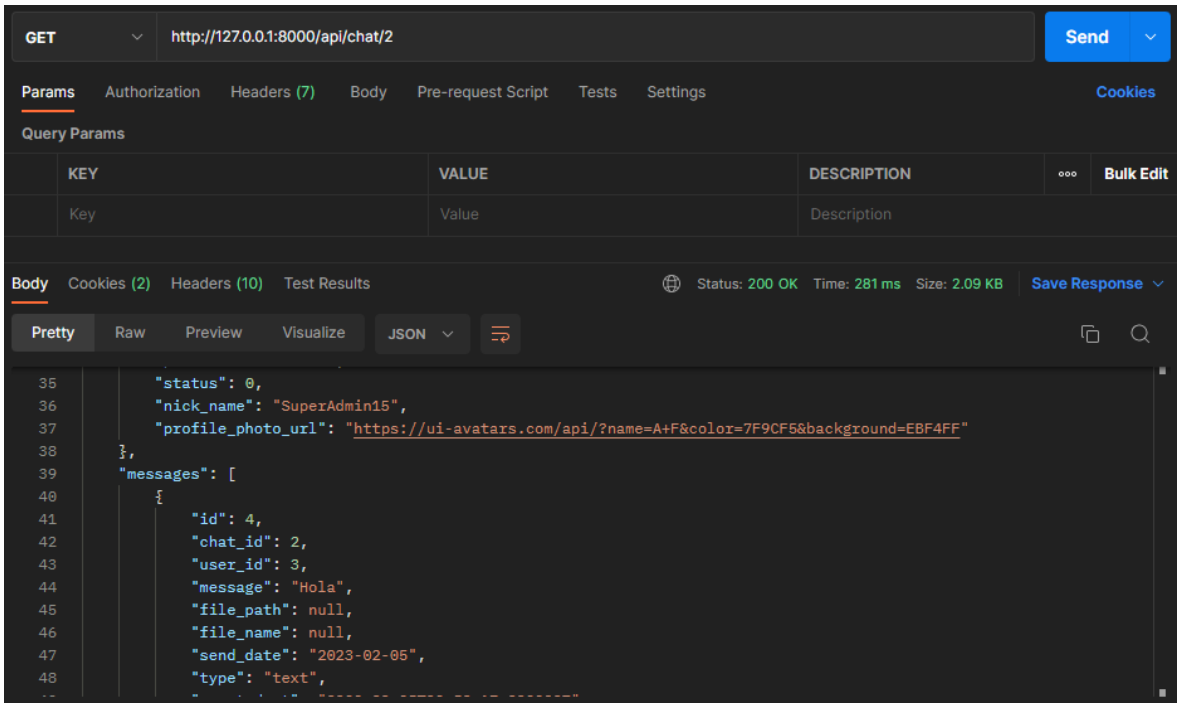


Fig. 68 Comprobación de llamada de chat con sus mensajes.

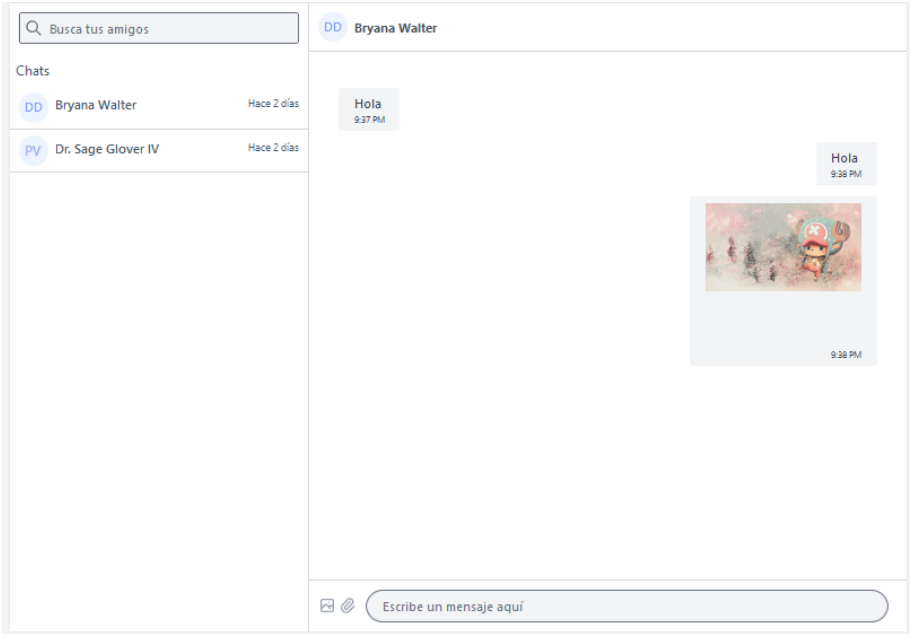


Fig. 69 Comprobación visual de chat en vivo

Pruebas de estrés

Al igual que las pruebas de carga, a continuación, se anexan las principales pruebas realizas que de igual forma se encuentra en conjunto a los perfiles demostrado, pero las mismas se encuentran protegidas según el rol. Cabe mencionar que las pruebas pueden variar dependiendo del servidor que se utilice en este caso el PC, que al ser una web que funciona constantemente en tiempo real, debe soportar cantidades de solicitudes mayores en poco tiempo.

En la **Fig. 69**, y en la **Fig. 70**, se presentan las principales peticiones de parte del administrador, como es el llamado de toda la información de *posts* y usuarios, al no existir muchos administradores, se realiza las pruebas de estrés con un mínimo de 60 peticiones en 10 segundos, es decir, 6 peticiones por segundo

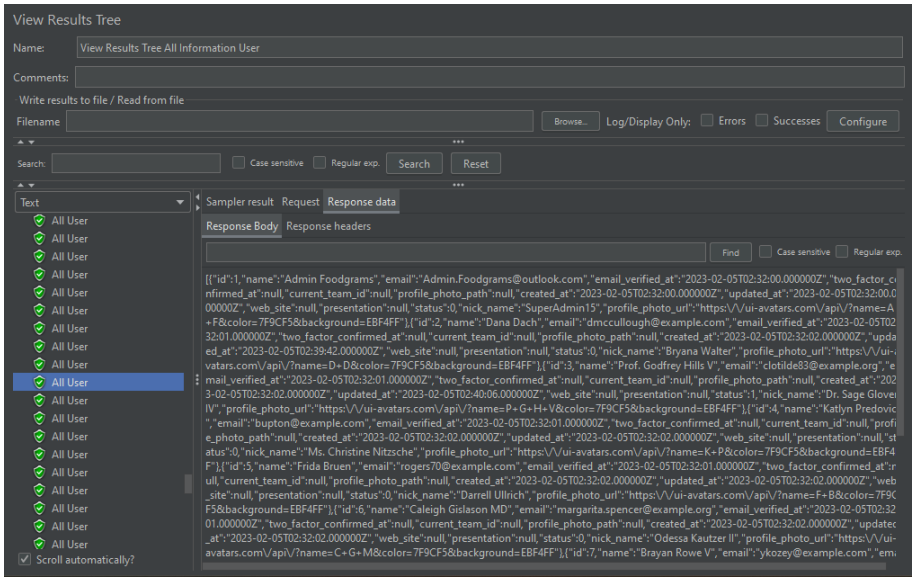


Fig. 70 Prueba de estrés para la petición de usuarios.

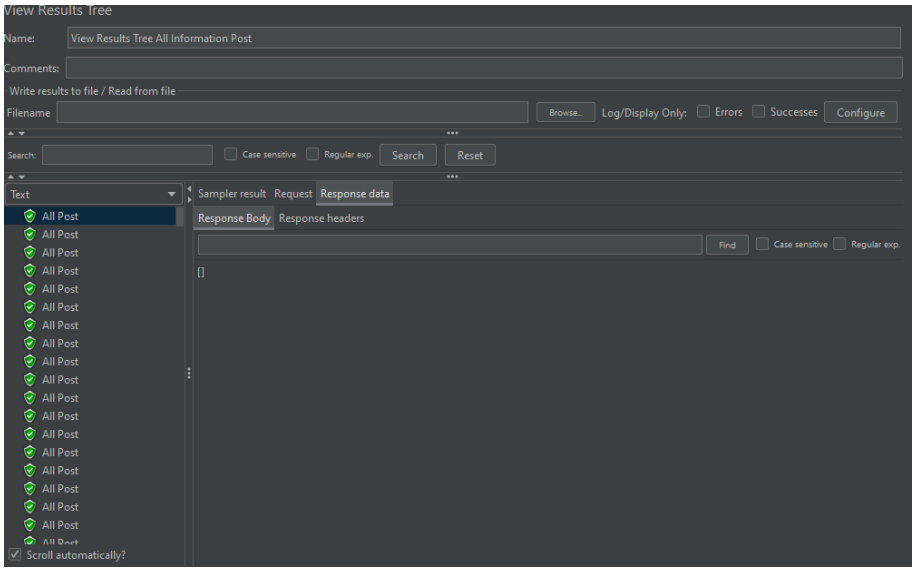


Fig. 71 Pruebas de estrés para peticiones de post recetas

Para el CRUD de usuarios se utilizan las mismas funciones para el *Admin* y los usuarios, la **Fig. 72**, nos trae la información del usuario solicitado, la **Fig. 73,** muestra la actualización de datos del usuario al cual lo solicitamos, la **Fig. 73**, indica la creación satisfactoria de un nuevo usuario y la **Fig. 74**, nos indica la eliminación del usuario, al ser eliminado las siguientes peticiones aparecen como error de usuario o página no encontrada la cual no afecta al rendimiento de la pagina

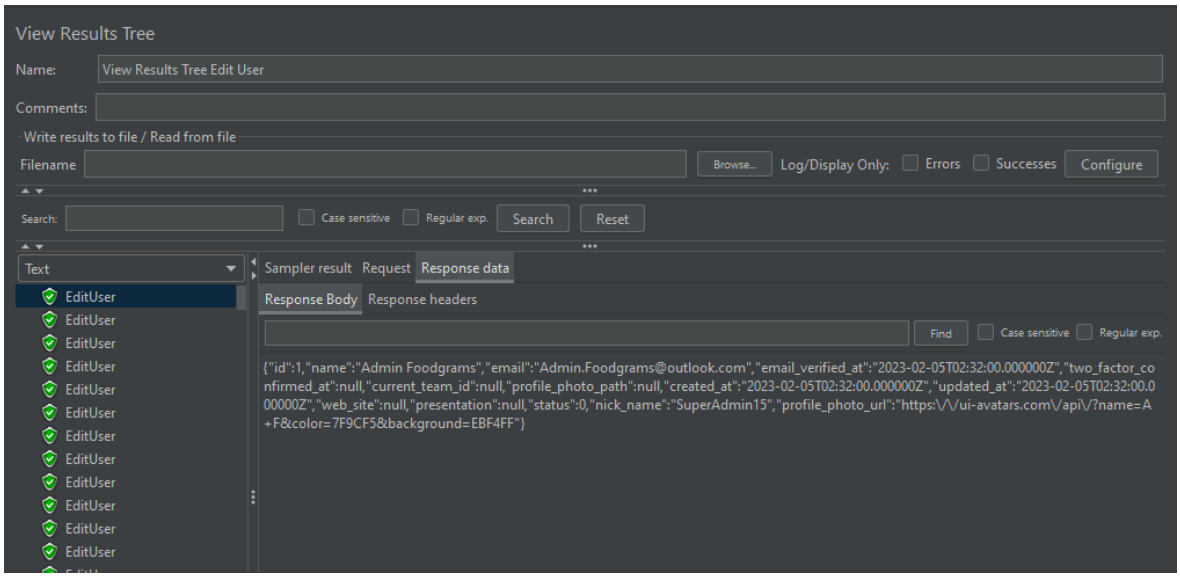


Fig. 72 Prueba de estrés de llamada de datos de usuario.

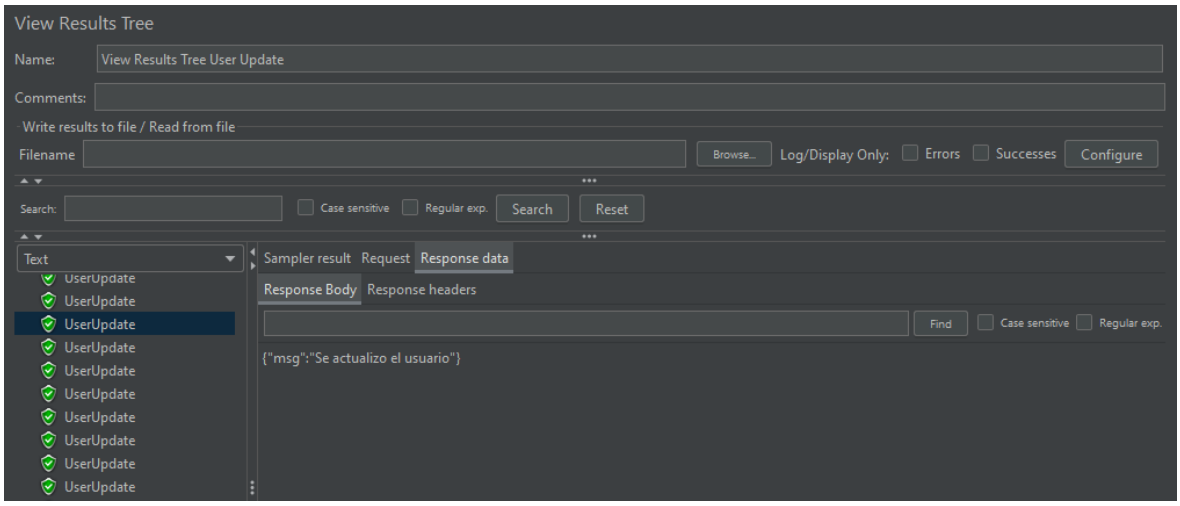


Fig. 73 Prueba de estrés para la actualización de datos de usuario.

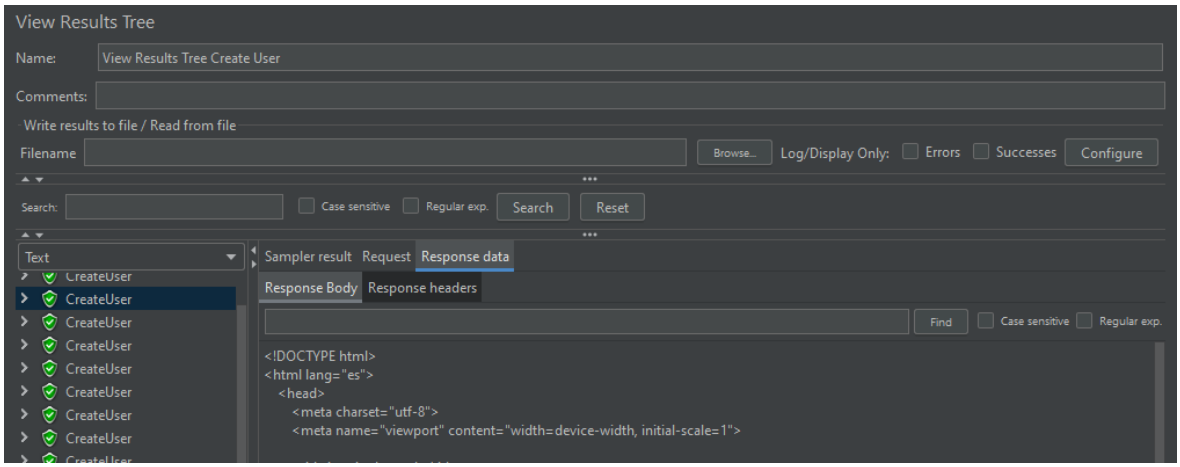


Fig. 74 Pruebas de estrés para la llamada para la creación de un nuevo usuario

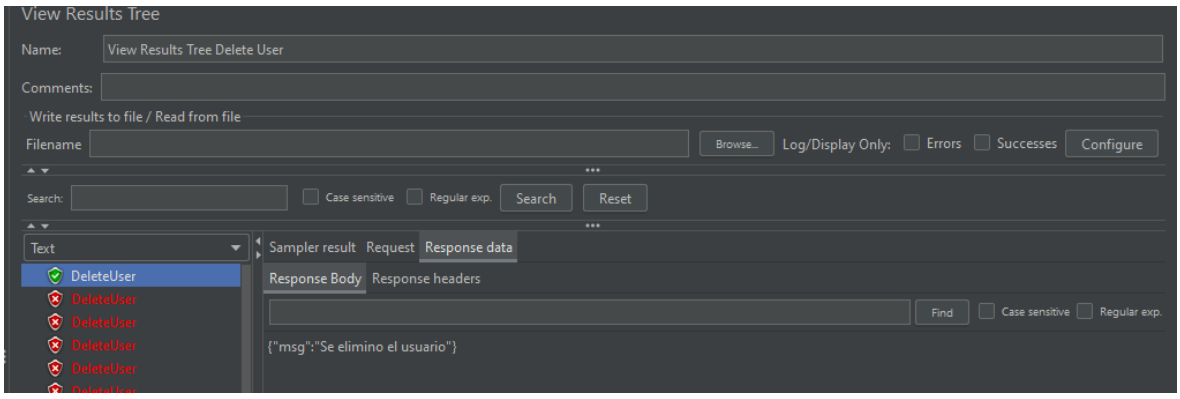


Fig. 75 Prueba de estrés para la eliminación de usuario

A continuación, se presenta el *CRUD* para los posts de tipos recetas, en la **Fig. 76**, se muestra la creación de un post múltiples veces de un mismo post, en la **Fig. 77**, se muestra la actualización de datos que por problemas con *jmeter* y la limitante de formas de ingreso

de datos que tiene el mismo ocasiona el error de actualización los post, en la **Fig. 787**, se presenta la llamada de la información que contiene el post, en la cual consta los likes y los comentarios y al final en la **Fig. 798**, se presenta la eliminación del post, que al igual que los usuarios una vez eliminado salta el error de no se encontró ese post.

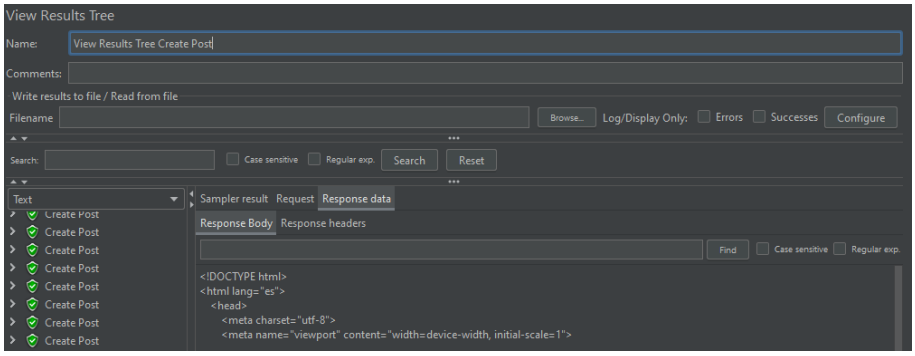


Fig. 76 Pruebas de estrés para la creación de post.

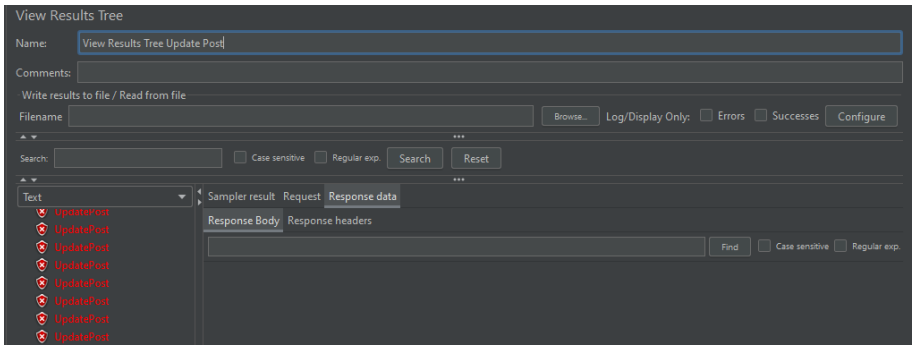


Fig. 77 Prueba de estrés para la actualización de post.

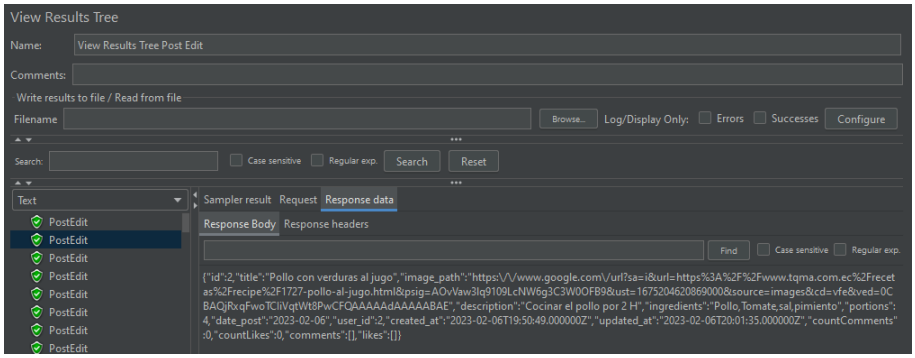


Fig. 78 Prueba de estrés para la llamada de un post.

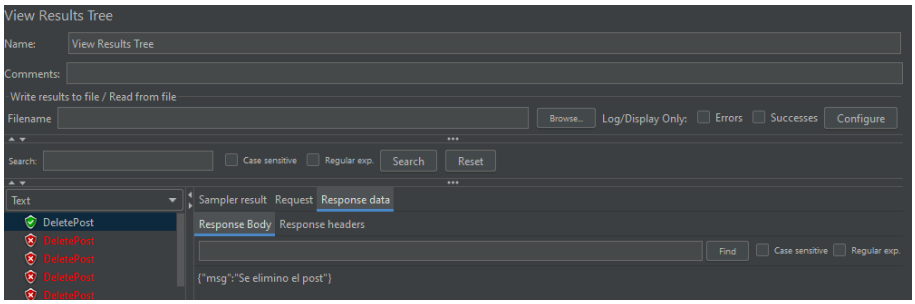


Fig. 79 Prueba de estrés para la eliminación de un post.

Al finalizar las pruebas de estrés correspondientes, se presentan las dos últimas de pruebas, al usar chat en vivo *jmetter*, no permite el envio de mensajes en tiempo real de diferentes chats, lo cual no permite evaluar la cantidad de datos que soporta esta función. Mientras tanto en la **Fig. 80**, se presenta el llamado constate de todos los chats que se han creado y la **Fig. 81**, se presenta toda la información de un chat que se realizado.

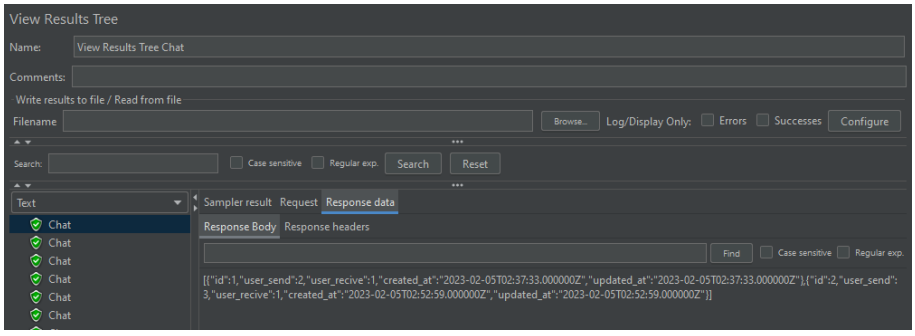


Fig. 80 Pruebas de estrés para la llamada de chats.

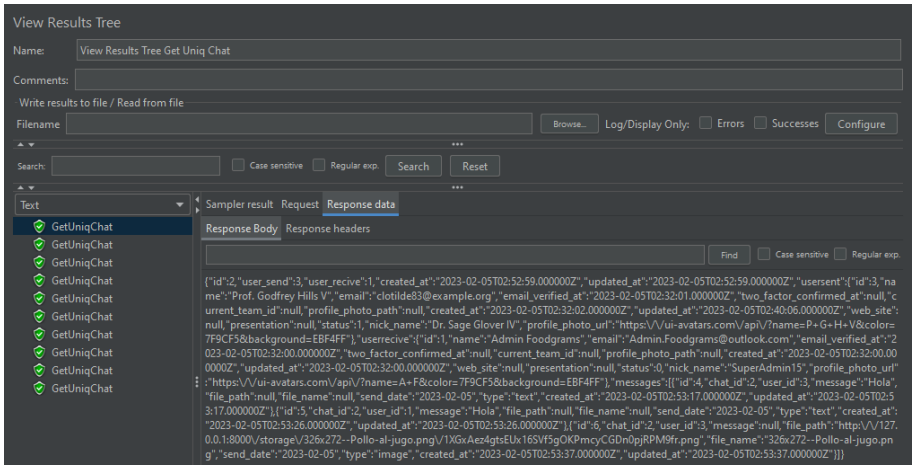


Fig. 81 Pruebas de estrés para la llamada de información de un chat.

ANEXO III

En esta sección se presenta el manual de usuario, el cual se encuentra redactado en el presente link, en donde se indica las principales características que se obtuvieron en el desarrollo del *backend*.

https://www.youtube.com/watch?v=mGh0zCBhJQY

ANEXO IV

A continuación, se presenta las credenciales que otorgan acceso al sistema de la parte administrativa, si se desea ingresar como usuario normal, se recomienda crear una cuenta propia. Además, se adjunta el link del github en el cual se presenta el manual de instalación.

Repositorio del Github

https://github.com/JosueEPN/Foodgrams.git

Link del despliegue a producción

http://foodgrams.click/

Credenciales de acceso

- **Administrador**

  Email = supera15proyect@gmail.com

  Contraseña = admin.15@2023

- **Usuario**

  Correo = User15Proyect@outlook.com

  Contraseña= user.15@2023