ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA WEB PARA LA GESTIÓN DE INVENTARIOS EN PYMES

DESARROLLO DE UN BACKEND

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

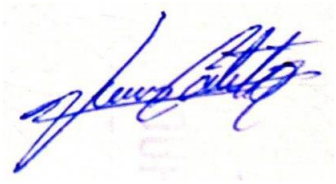
LUIS SEBASTIAN CATOTA CAIZALUISA

DIRECTOR: ING BYRON LOARTE

Quito, febrero 2023

CERTIFICACIONES

Yo, Luis Sebastian Catota Caizaluisa 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.



Luis Catota

luis.catota@epn.edu.ec

luissebastian11032014@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Luis Sebastian Catota Caizaluisa, bajo mi supervisión.

Ing. Byron Loarte, MSc.

DIRECTOR

byron.loarteb@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.

Luis Sebastian Catota Caizaluisa

DEDICATORIA

A mis padres, por sus palabras de aliento las cuales lograron llevarme a este punto de mi formación académica.

A mi familia, que me acompaño durante esta hermosa travesía llena de altos y bajos.

A mi novia Estefania, que es mi compañera de vida y estuvo de principio a fin en este camino.

A mi mejor amigo Paul, que estuvo en los momentos más complejos de la carrera.

A mis amigos y compañeros, que juntos culminamos esta extraordinaria aventura, recuerdo con alegría todas las horas de trabajo en las que nos juntamos para salir adelante en cada materia a lo largo de nuestra carrera.

Llegar al final de un escalón más de nuestra historia de vida me emociona por pensar en el porvenir de nuestras vidas académicas y profesionales.

Luis Sebastian Catota Caizaluisa

AGRADECIMIENTO

A mi familia, por el cariño, paciencia y apoyo tanto físico como emocional en todo momento, sin ustedes no podría haber logrado este paso en mi vida académica.

A mis profesores, por compartir sus conocimientos, dedicación, perseverancia y tolerancia de manera precisa y rigurosa.

A mi tutor y profesor Byron Loarte quien sin su esfuerzo, paciencia y constancia a lo largo del desarrollo del proyecto no lo hubiese conseguido. Sus recomendaciones tanto académicas como personales fueron siempre acertadas para no decaer en este camino.

A mi novia gracias por sus palabras de aliento, cuando más las necesite.

Luis Sebastian Catota Caizaluisa

ÍNDICE DE CONTENIDO

RESUMEN

Actualmente, en el Ecuador la generación de nuevas ideas de negocio o emprendimientos ha tenido un crecimiento exponencial debido a la falta de empleos. Lo cual evidencia como resultado la creación de nuevas empresas o la expansión de las ya existentes, las cuales buscan establecer innovación en sus procesos de administración de inventarios ya que es la base de un buen crecimiento en general. Sin embargo, en el mercado existen sistemas gestores de inventarios muy completos, pero a la vez muy caros y complejos para una pequeña empresa o emprendimiento lo cual dificulta su adquisición y utilización dentro de la misma. Lo que provoca que utilicen medios tradicionales para llevar la gestión de su inventario como es medios impresos o en raras ocasiones el uso de medios ofimáticos lo que genera una falta de organización de la información cuando los datos comienzan a crecer diariamente.

Por lo citado anteriormente, se muestra que el trabajo de Integración Curricular desarrollado y puesto a producción es un *backend,* el cual permita realizar la gestión de inventarios de una manera mucho más fácil y que se acople a las necesidades y procesos internos de los nuevos emprendimientos y empresas ya existentes. Por otra parte, las tareas más destacadas del *backend* son: administración del inventario por tres roles de usuarios, gestión de productos, generación de reportes, generación de notificaciones, entre otros. Por último, todas las acciones citadas fácilmente son generadas por una serie de *endpoints* públicos y privados, los cuales pueden ser accedidos y consumidos por aplicaciones del lado del cliente o móviles.

El documento se encuentra estructurado de la siguiente manera: La primera sección detalla los antecedentes, objetivos, alcance y marco metodológico. La segunda sección describe la adecuada implementación de *Scrum* como metodología ágil, modelado de los datos, diseño de la arquitectura y el uso de herramientas para el *backend*. La tercera sección presenta las tareas que se han establecido dentro de cada iteración, así como sus resultados y pruebas respectivas. Finalmente, la cuarta y quinta parte presenta las conclusiones y recomendaciones que se han logrado obtener a lo largo de todo el proyecto.

PALABRAS CLAVE: Sistemas gestores de inventarios, *backend, endpoints, Scrum.*

ABSTRACT

Currently, in *Ecuador* the generation of new business ideas or ventures has had an exponential growth due to the lack of jobs. This has resulted in the creation of new companies or the expansion of existing ones, which seek to establish innovation in their inventory management processes as it is the basis for good growth in general. However, in the market there are very complete inventory management systems, but at the same time very expensive and complex for a small company or enterprise, which makes its acquisition and use within the same difficult. This causes them to use traditional means to manage their inventory, such as printed media or, on rare occasions, the use of office automation, which generates a lack of organization of the information when the data begins to grow on a daily basis.

For the above mentioned it shows that the Curricular Integration work developed and put into production is a backend which allows inventory management in a much easier way and that fits the needs and internal processes of new ventures and existing companies. On the other hand, the most important tasks of the backend are: inventory management by three user roles, product management, report generation, sending notifications, among others. Finally, all the aforementioned actions are easily generated by a series of public and private endpoints, which can be accessed and consumed by client-side or mobile applications.

The paper is structured as follows: The first section details the background, objectives, scope and methodological framework. The second section describes the proper implementation of Scrum as an agile methodology, data modeling, architecture design and the use of backend tools. The third section presents the tasks that have been established within each iteration, as well as their respective results and tests. Finally, the fourth and fifth part presents the conclusions and recommendations that have been obtained throughout the project.

KEY WORDS: Inventory management systems, backend, endpoints, Scrum.

1 INTRODUCCIÓN

Existen diversas organizaciones las cuales realizan importantes inversiones en sistemas *software* que sean modernos, fáciles de utilizar y que a la vez permitan administrar y automatizar sus procesos internos de una manera mucho más eficiente. Además, estos sistemas *software* cada vez están reemplazando a los sistemas empresariales de gestión tradicional [1].

Un sistema *software* para gestionar inventarios se encarga de aportar las normas operativas que permiten mantener y controlar los productos destinados a ser almacenados y gestionados. Además, el sistema es el encargado de ordenar y recibir los productos, coordinar la ubicación de pedidos y su respectivo seguimiento y control [2].

A través de una investigación se determina que Ecuador es uno de los países en proceso de transformación donde el uso de la tecnología y los sistemas electrónicos enfocados al desarrollo empresarial tiene una mayor demanda cada día [3]. Por otra parte, un sistema *software* empresarial aporta una serie de ventajas al negocio o empresa ya que mantiene un inventario eficiente, realiza un control de los productos durante su envío a distintas ubicaciones, permite la recepción de nuevos productos, permite gestionar el almacenaje (empaquetado, envío y clasificación) de nuevos productos, prevenir caducidad, entre otros [4]. En ese sentido, el uso de un adecuado sistema *software* de gestión de inventario permite manejar correctamente el control de bodega y la cadena de suministros llegando a optimizar recursos y disminuir pérdidas para el negocio o empresa que lo implemente [5].

Un sistema que gestione inventarios proporciona soluciones para la administración de datos que existen en los negocios latentes [6]. Un emprendimiento no siempre cuenta con un sistema informático independiente, el cual maneje los datos específicos para la gestión de inventario de sus productos [7]. La adquisición y gestión de productos de forma regular, sin un sistema, es ineficaz, ineficiente y desactualizado. Por lo tanto los datos en el almacenaje de productos son una prioridad si el enfoque de la empresa es crecer a largo plazo [8].

En base a lo citado anteriormente y con el propósito de fortalecer el área informática de las PYMES en el ecuador, se ha desarrollado el despliegue y paso a producción de un *backend*, el cual permita realizar la gestión de inventarios de una manera mucho más eficiente y que se acople a las necesidades y procesos internos de las PYMES.

1.1 Objetivo general

Desarrollar el *backend* para un sistema web para la gestión de inventarios en PYMES.

1.2 Objetivos específicos

1. Realizar el levantamiento de requerimientos y funcionalidades para el *backend*.
2. Trazar el esquema para el almacenamiento de los datos en función de los requerimientos que se han establecido.
3. Implementar el patrón arquitectónico y arquitectura REST para el *backend.*
4. Codificar cada uno de los módulos y *endpoints* para el *backend*.
5. Evaluar la funcionalidad del *backend* a través de una serie de pruebas.
6. Desplegar a producción el *backend*.

1.3 Alcance

Al considerar el trabajo que conlleva la implementación de un *backend* que permita la gestión de inventario para las PYME y en base a un proceso de investigación y recopilación de funcionalidades, se determinaran dos funcionalidades generales que son: control de bodega y cadena de suministros. Por otra parte, entre las acciones por defecto que integra el *backend* se encuentran: el registro de usuarios, *login* y gestión de perfiles (administrador, empleado y empleado sucursal) los cuales tienen métodos *CRUD (Create, Read, Update, Delete)* para una correcta administración de la información.

Cabe destacar que el usuario administrador tiene la capacidad de realizar acciones como: valoración de inventario, ajustes, notificaciones, distribución, reportes y productos (ingreso, salida, alertas). Mientras que los usuarios con rol empleado pueden realizar acciones como: notificaciones y productos (ingreso y salida), los usuarios con rol empleado sucursal pueden acceder únicamente a realizar acciones como: notificaciones y reportes con sus respectivas restricciones. Por último, se asegura la integridad de los datos por parte del *backend* y la base de datos gracias a una serie de perfiles y roles previamente establecidos.

El *backend* y cada uno de los objetivos que se han determinado se han cumplido gracias a la implementación de una metodología ágil, un *framework* para el *backend* como lo es *Express.js*, el cual permite crear diferentes *endpoints* de manera rápida y eficiente, un patrón arquitectónico MVC (Modelo-Vista-Controlador), una sucesión de *tests* que validen el correcto funcionamiento de los *endpoints* y posterior a ello un adecuado despliegue a producción sobre una infraestructura en la nube como lo es Railway.

Por último, el *backend* dispone de los siguientes perfiles de usuarios que se describen a continuación:

Perfiles establecidos:

- Administrador
- Empleado
- Empleado sucursal

Se establece para el perfil administrador:

- *Endpoints* que presenten una página informativa.
- *Endpoints* que permitan cerrar sesión, iniciar sesión y cambiar la contraseña.
- *Endpoints* que permitan modificar los datos personales.
- *Endpoints* que permitan la gestión de perfiles de usuarios.
- *Endpoints* que permitan crear los productos.
- *Endpoints* que permitan observar notificaciones.
- *Endpoints* que permitan generar reportes de productos.

Se establece para el perfil empleado:

- *Endpoints* para modificar información personal.
- *Endpoints* para el ingreso de productos.
- *Endpoints* para observar notificaciones.
- *Endpoints* para generar reportes.

Se establece para el perfil empleado sucursal:

- *Endpoints* para modificar información personal.
- *Endpoints* que permitan observar notificaciones.
- *Endpoints* para la distribución de los productos.
- *Endpoints* para crear reportes de productos.

1.4 Marco teórico

En el ámbito administrativo dentro de una empresa el inventario desempeña un rol importante, el cual es utilizado para llevar un control adecuado en un proceso de producción y que de esta manera permita reducir costos en la adquisición de materiales, procesamiento de productos y productos terminados [9].

Existen inconvenientes en el desarrollo de *software*, en el caso de que el producto no sea desarrollado con buenas prácticas de codificación, patrones de diseño, patrones de arquitectura, etc. Lo que ocasiona una serie de problemas que pueden saturan al servidor y que el tiempo de respuesta no sea el adecuado [10]. En ese sentido, se establece la implementación de un patrón de arquitectura de tres capas que se describe a continuación:

- **Capa de presentación:** responsable de verificar en su totalidad la interfaz del usuario.
- **Capa de procesos o lógica de negocio:** responsable de verificar en su totalidad los métodos y el desempeño de las reglas del negocio.
- **Capa de datos o almacenamiento de la información:** responsable de verificar en su totalidad aquellos datos que son ingresados por parte de la capa de procesos y almacenamiento.

JavaScript es un lenguaje de programación popular dentro del ecosistema de código abierto enfocado en el desarrollo *web*, el progreso de este lenguaje se debe a su asociación con motores de navegación como el V8 de *Google* y *SpiderMonkey* de *Mozilla* [11]. Por otra parte, este lenguaje no solo es utilizado en navegadores *web,* sino que también cumple un rol importante dentro de las bases de datos no relaciones (*NoSQL*) modernas como *MongoDB* y *CouchDB*, los cuales utilizan *JavaScript* para ejecutar comandos y desarrollar servidores con entornos escalables [12].

Existe un debate entre los estándares de intercambio de información como lo son: *XML* y *JSON* los cuales tienen sus puntos fuertes y débiles, los dos permiten almacenar y transmitir datos entre aplicaciones. *XML* comparte documentos con distintos datos y elementos complejos a diferencia de *JSON* el cual comparte datos más simples con mayor velocidad de rendimiento debido a que tiene una estructura sencilla y de fácil acceso a los datos [13].

Node.Js es el ambiente de ejecución para el lenguaje de programación *JavaScript* del lado del servidor, el cual incorpora eventos asíncronos para crear y diseñar aplicaciones escalables y el desarrollo de *API's RESTFul* [14].

Un *API* permite la comunicación entre dos programas *software*, mediante el protocolo de transferencia de hipertexto (*HTTP*) el cual permite la transferencia de información para que las aplicaciones *web* sean más robustas y escalables [15].

Una *API RESTFul* es una *API* que sigue el esquema *REST* para crear servicios *web*, en donde se comparte información a través de internet de manera segura y confiable [16]. Posee métodos establecidos para compartir información como: *GET, POST, PUT, DELETE*, entre otros [15].

MongoDB es un almacén de datos *NoSQL* que implementa ciertas peculiaridades equivalentes a una base de datos relacional (*SQL*) como son: clasificar, indexar secundariamente, consultar rangos y consultar documentos anidados. También permite operaciones como: crear, insertar, leer, actualizar y eliminar [17].

Las librerías son una serie de implementaciones funcionales compiladas y codificadas en un lenguaje de programación las cuales ofrecen métodos o interfaces que permiten ser implementadas e invocadas de una forma sencilla para optimizar ciertas tareas y que estas se ejecuten de manera independiente [18].

Las metodologías ágiles emergen como respuesta para llenar vacíos metodológicos presentes en las metodologías tradicionales, consisten en colaborar con el cliente de tal manera que el desarrollo del *software* sea incremental en base a intervalos cortos de tiempo y entrega [19].

Un *token* de seguridad es un instrumento de *software* o *hardware* que se encarga de generar una clave segura, en ese sentido se puede definir que *JSON Web Token* (JWT) es un modelo para crear *tokens* seguros [20].

La arquitectura de *software* es una construcción inteligente y planeada, para resolver un problema, donde se produce una descomposición de alto grado de abstracción en partes más pequeñas las cuales sean más fáciles de cambiar después de haber sido construido el sistema *software*. Por esta razón, MVC es un patrón arquitectónico que ayuda a definir una estructura dentro un proyecto *software* para garantizar de esta manera la calidad del código y que sea escalable a futuro [21].

En la actualidad, los proyectos de desarrollo de *software* que poseen un *backend* y un *frontend* por separado proporcionan una mejor experiencia de usuario ya que suelen conectarse a través de *API's*, las cuales pueden utilizar arquitecturas de tipo *SOAP* o *REST*, logrando con ello una alta compatibilidad entre diferentes aplicaciones de *software* [22].

Para lograr que un *software* sea de calidad se necesita evaluarlo bajo ciertos parámetros y estándares, para lo cual es necesario ejecutar una serie de pruebas [23]. Es decir,

comprobar que los módulos que se han desarrollado se evalúen después de ser programados, sin esperar a que se integren al proyecto, es por esta razón que las pruebas deben realizarse a través de herramientas, técnicas y métodos que evalúen al *software* y de ser el caso corregir los posibles problemas [24].

La fase de despliegue dentro del desarrollo de un producto *software* radica en que el usuario final pueda utilizar dicho *software* desde cualquier navegador *web*. Para lograr lo antes mencionado, la fase de despliegue atraviesa un proceso exhaustivo de actividades permitiendo a los usuarios tener acceso al proyecto a través de una conexión a internet [25]. En ese sentido, se ha utilizado la plataforma *Railway* la cual permite desplegar proyectos *web* a entornos de producción [26].

2 METODOLOGÍA

El estudio de casos se destaca por ser llamativa, minuciosa, estricta y se caracteriza por responder preguntas de clase “porqué” o “cómo”; este tipo de preguntas generan teorías que producen la lógica del estudio de casos, además se usa como herramienta investigativa para construir el objetivo, plan de estudio y lograr una solución adecuada al problema planteado [27].

El presente trabajo de integración curricular mantiene un estudio de casos debido a que inicia con un estudio investigativo en el que muestra la relevancia de actualizar los métodos tradicionales de gestión de inventario por un sistema moderno que ayude a cada una de las PYMES, utilizando para ello un *backend* que solvente todas las necesidades requeridas.

2.1 Metodología de Desarrollo

Una metodología de desarrollo de *software* se utiliza dentro de un proyecto *software* para que el equipo de trabajo logre cumplir los objetivos que se han propuesto [28].

Las metodologías de desarrollo ágil aportan una serie de lineamientos que permiten realizar las tareas planificadas de forma eficiente, iterativa y colaborativa [29]. *Scrum* es una metodología ágil que permite adaptarse rápidamente a los cambios y entre sus características principales se encuentra la capacidad de desarrollar un *software* mediante *Sprints,* los cuales tienden a durar de dos a cuatro semanas donde el resultado de cada *Sprint* es un adelanto evidente del proyecto presentado al cliente, también se debe tomar en cuenta las reuniones diarias que realiza el equipo encargado del desarrollo, donde se coordinan e integran los siguientes S*prints*. A continuación, se presenta la implementación de *Scrum* en la creación de *endpoints* del *backend*.

Roles

Cada rol dentro de la metodología ágil *Scrum* se encuentra compuesto por personas que se comprometen con el proyecto y las actividades planteadas dentro del proceso *Scrum*, es decir el *Development team, Product Owner y el Scrum Master* son roles fundamentalmente necesarios para el punto de partida del proyecto y de esta manera encaminar y planear cada *Sprint* de la mejor manera [30]. Considerando los puntos mencionados anteriormente se muestran los roles del *backend.*

Product Owner

Es la persona con mayor grado jerárquico dentro del proyecto porque es el encargado de tomar las decisiones cruciales para el correcto desarrollo. Además, es el responsable de conocer a fondo la lógica del negocio y su visión del producto [30]. Por tal razón, la **TABLA I** presenta al usuario que se ha designado para este rol, el cual dicta todos los requisitos que el producto *software* debe tener.

Scrum Master

Cumple el rol de asegurar el modelo y la metodología *Scrum,* por ende, garantiza un buen funcionamiento y ejecución, lo cual se logra al compartir sus conocimientos y técnicas con los gestores y su equipo de trabajo, de esta manera implementa la función de eliminación de los posibles problemas que surjan y provoquen que el proceso no fluya [30]. Por lo que en la **TABLA I** presenta al usuario que se ha designado para este rol, el cual cumple con los conocimientos, habilidades y experiencia adecuada. Con la finalidad de solventar las tareas requeridas por este cargo.

Development Team

Consiste en un pequeño equipo de 5 a 9 desarrolladores los cuales tienen la potestad de organizar y tomar decisiones que permitan cumplir los objetivos del proyecto en el tiempo estimado. Además, el equipo debe participar en la estimación del esfuerzo del *Product* y *Sprint Backlog* [30]. Por lo tanto, en la **TABLA I** se muestra al usuario que se ha designado para este rol, el cual cumple con todos los conocimientos necesarios para elaborar una adecuada estimación de las tareas a ser planificadas.

TABLA I: Determinación de Roles en el backend.

ROLES	NOMBRE
Product Owner	Ing. Byron Loarte, MSc.
Scrum Master	Ing. Byron Loarte, MSc.
Development Team	Luis Catota

Artefactos

Los artefactos dentro de *Scrum* permiten registrar toda la información como parte del desarrollo de un producto *software* [30]. Además, se encargan de garantizar la calidad, transparencia y la productibilidad del proyecto de tal forma que se mantiene el control y el tiempo de entrega de cada *Sprint* [31]. En consecuencia, se presenta los artefactos que han sido útiles para el registro y control de los avances del *backend.*

Recopilación de requerimientos

Es la etapa encargada de definir las funciones del producto *software*, la estimación de tiempo y los cambios que se realizan en las entradas de información para su posterior salida en relación con las necesidades del cliente [32]. De tal manera que, en la **TABLA II** se visualiza la plantilla para recopilar los requerimientos del cliente. Sin embargo, el detalle de la plantilla completa se encuentra en el **ANEXO II** del presente escrito.

TABLA II: Plantilla de requerimientos que han sido recopilados.

REQUERIMIENTOS COMPILADOS		
TIPO DEL SISTEMA	**ID – RR**	**ENUNCIADO DEL ÍTEM**
Backend	**RR011**	El cliente empleado sucursal tiene la posibilidad de utilizar *endpoints* para: • Administrar productos.
	RR012	Los usuarios empleados y empleados sucursal necesitan consumir *endpoints* que: • Observar notificación de productos.

Historias de Usuario

Es la etapa en la cual se presentan de forma general los requerimientos y funcionalidades que posee el proyecto a desarrollar, una historia de usuario se forma con el trabajo colaborativo entre el cliente y el equipo encargado del proceso *Scrum* [30]. Por este motivo, en la **TABLA III** se ejemplifica la plantilla de tal manera que se detallen las acciones a desarrollar en el *backend*, sin embargo, el detalle de las demás tablas completas se encuentra en el **ANEXO II** del escrito.

TABLA III: *Endpoints* encargados del *login*.

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario**: Administrador, empleado sucursal y empleado.
Nombre Historia: Generar varios *endpoints* para un login.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Luis Catota	
Descripción: Se pueden generar varios *endpoints* los cuales gestionan el *login* del perfil asignado; además tiene la posibilidad de cambiar la contraseña si lo solicita.	
Observación: El cliente administrador, empleado y empleado sucursal necesariamente tienen que iniciar sesión para realizar cualquier acción.	

Product Backlog

Consiste en una serie de funcionalidades y obligaciones en forma de lista priorizada, es similar a una cola priorizada de tareas con sus respectivas características. Además, la estructura que posee se basa en una descripción específica y una estimación de la carga de trabajo que posee [33]. Por consiguiente, la **TABLA IV,** presenta el formato del *Product Backlog*, además el detalle del formato completo se encuentra en el **ANEXO II** del escrito.

TABLA IV: *Product Backlog*.

PRODUCT BACKLOG				
ID – HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU009	Generar varios *endpoints* para crear reportes de salida.	2	Finalizada	Alta
HU010	Generar varios *endpoints* para modificar información personal.	1	Finalizada	Media

Sprint Backlog

Es la lista de tareas o elementos de mayor prioridad que el equipo construye como parte de la planificación de un *Sprint* [33]. Además, tiene la responsabilidad de proporcionar las tareas a cada integrante del grupo y el tiempo para terminar dichas tareas. Por tal motivo el

proyecto se fracciona en partes más pequeñas que permiten monitorear los avances [30]. En ese sentido, la **TABLA V** muestra un ejemplo con la estructura para la elaboración de cada uno de los 5 *Sprints* los cuales permiten: Construir el ambiente de desarrollo, diseño y ejecución de *endpoints* para distintos usuarios como administrador, empleado y empleado sucursal, pruebas en el *backend* y despliegue. Por último, el detalle de la plantilla completa se encuentra en el **ANEXO II** del escrito.

TABLA V: *Sprint Backlog.*

SPRINT BACKLOG						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB004	Testeos del *backend*	-------	-----	----------	• *Testeos* unitarios. • *Testeos* de compatibilidad. • *Testeos* de estrés.	20 horas

2.2 Diseño de la arquitectura

En lo que respecta al desarrollo de un producto *software* siempre es necesario recurrir a diversas disciplinas arquitectónicas bien definidas, por lo cual se presenta un modelo que determina tres componentes (Modelo, Vista y Controlador) que permita organizar de forma clara el código de un proyecto, el manejo adecuado de errores y satisfacer la calidad del proyecto [34]. Debido a esto el presente *backend* define un modelo de arquitectura idóneo, el cual facilita la codificación y unificación del *backend* con otras herramientas, librerías y tecnologías modernas.

Patrón arquitectónico Modelo Vista Controlador (MVC)

MVC plantea un procedimiento serio que se divide en una fase de entrada, su posterior proceso y una fase de salida de los datos previa a una relación entre todas las fases mencionadas las cuales se especifican de la siguiente manera [35]:

- **Modelo:** capa en la cual los datos del modelo de negocio se almacenan dando paso a que el proyecto en desarrollo los procese.

- **Vista:** capa responsable de generar la interfaz la cual consiste en las pantallas, páginas o un modelo en el cual existen varias vistas asociadas al mismo modelo.
- **Controlador:** capa que tiene la funcionalidad de enlazar las comunicaciones como instrucciones y peticiones entre el usuario y el sistema, también tiene la posibilidad de acceder al modelo y las vistas.

La **Fig. 1**, presenta el modelo arquitectónico que se ha implementado como parte de la codificación del *backend*, en conjunto con las herramientas que han sido utilizadas para la etapa de desarrollo, testeo y paso a producción.

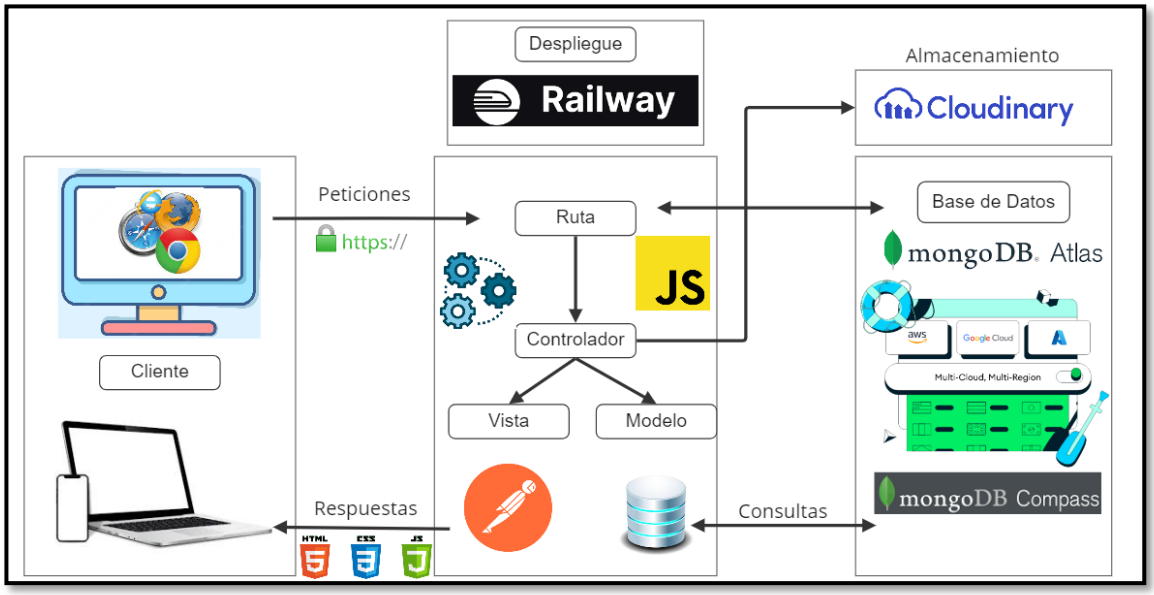


Fig. 1: Patrón Arquitectónico – *backend*.

2.3 Herramientas de desarrollo

El principal objetivo de los instrumentos o herramientas para desarrolladores es favorecer, agilitar y enriquecer la ejecución de un proyecto. Por ende, este tipo de herramientas se aplican en distintos campos y facilitan el desarrollo de funcionalidades más complejas [36]. De tal forma que la **TABLA VI** presenta una serie de herramientas que han sido útiles en la etapa de elaboración del código del *backend*.

TABLA VI: Instrumentos del *backend*.

HERRAMIENTA	JUSTIFICACIÓN
Node.js	Genera aplicaciones del lado del servidor escalables gracias a la ejecución de *JavaScript* asíncrono [37].

Express.js	Es un *Framework* que proporciona una serie de funcionalidades que ayudan a generar aplicaciones *web* [38].
MongoDB Atlas	Es un servicio que se ejecuta en la nube donde el aprovisionamiento, la configuración y la implementación de infraestructuras están totalmente automatizadas [39].
MongoDB Compass	Es una poderosa GUI (Interfaz gráfica de usuario) que consulta, agrega y analiza datos de *MongoDB* en un entorno visual más legible [40].
Railway	Es una plataforma de implementación o despliegue donde se puede subir la infraestructura localmente y luego implementar en la nube [41].
Visual Studio Code	Es el editor de código con mayor demanda actualmente ya que tiene afinidad con la mayor parte de *operating system* [42].
Cloudinary	Es un servicio de almacenamiento en la nube que se enfoca principalmente en el almacenaje de imagenes [43].

Librerías

Las librerías son archivos escritos en cualquier lenguaje de programación, que proporcionan funcionalidades adicionales las cuales facilitan los procesos dentro del código, por esta razón la **TABLA VII** presenta las librerías que han complementado la generación de *endpoints* en el *backend.*

TABLA VII: Librerías en el *backend*.

LIBRERÍA	DESCRIPCIÓN
Moongose	Modela los datos de un proyecto y permite acciones como: validación, creación de consultas, enlaces de lógica de negocios y más [44].
bcrypt	Biblioteca que permite cifrar contraseñas [45].
JWT	Son métodos estándar de la industria de código abierto para crear *tokens* seguros [46].
dotenv	Es un módulo de dependencias cero que carga variables de entorno desde un archivo "*env*" a un "*process.env*" [47].

nodemon	Se encarga de monitorear los cambios en el código y procede a reiniciar el servidor automáticamente [48].
cors	Se encarga de integrar cabeceras *HTTP* adicionales que permitan a los usuarios acceder a los recursos requeridos desde un servidor [49].
uuid	Es un identificador único que genera cadenas aleatorias y criptográficamente seguras [50].
Express-fileupload	Middleware de *express* que simplifica la cargar archivos [51].
node-fetch	Es una función de alto nivel y toma una *URL* para producir una promesa que se resuelve en la respuesta [52].

3 RESULTADOS

En el siguiente apartado se presenta la ejecución de los diferentes *endpoints* en el *backend*, además de los resultados que se han producido al momento de realizar los distintos testeos y el posterior pase a producción. En tal sentido, cada resultado se muestra por medio de *Sprints* los cuales se detallan en el **ANEXO II** del escrito.

3.1 *Sprint* 0. Preparación del entorno de desarrollo.

Dentro del *Sprint* inicial se establecen las siguientes tareas:

- Delimitación de requerimientos en el *backend*.
- Modelamiento de datos con *MongoDB*.
- Organización estructural del *backend*.
- Módulos para los usuarios.

Delimitación de requerimientos en el *backend*

Endpoints que presenten una página informativa

Dentro del *backend* se implementa *endpoints* para mostrar información del proyecto y sobre las PYMES.

Modelamiento de datos para el *backend*

El *backend* implementa *endpoints* que proporcionan capacidades especificas a los usuarios: administrador, empleado y empleado sucursal. De tal manera que pueden realizar un *login*. Además, se han generado *endpoints* con la función de devolver *tokens* los cuales garantizan que el inicio de sesión sea un proceso seguro.

Módulos para los usuarios

Se han originado *endpoints* encargados de administrar la información de los usuarios con perfil administrador, empleado y empleado sucursal, los cuales disponen de acciones como crear, visualizar, editar, y eliminar la información obtenida para la creación de un perfil.

Generar *endpoints* para gestionar los perfiles de usuarios

Se han originado *endpoints* destinados a la administración del perfil de cada usuario, los cuales disponen de acciones como crear, listar, modificar y habilitar o deshabilitar perfiles. Por otro lado, si se desea eliminar un perfil es necesario que el perfil se encuentre habilitado para realizar esta acción.

Generar *endpoints* para gestionar productos y visualizar *stock*

Se han originado *endpoints* encargados del correcto acceso al cliente con perfil administrador, a los datos relacionados con los productos, proporcionándole el permiso de ingresar, listar, modificar y eliminar los productos ingresados, almacenados y agotados. Dichas facultades se pueden llevar a cabo gracias a que cada producto se introduce con campos como: nombre, tipo de producto, cantidad, categoría, código y descripción. Se debe tomar en cuenta que esta funcionalidad también es accesible para el usuario con perfil empleado y empleado sucursal.

***Endpoints* que permitan observar notificaciones de productos**

Se han originado *endpoints* con la finalidad de permitir el acceso al usuario con perfil administrador, además de enviar y recibir notificaciones, de tal manera que se le proporcione la capacidad de gestionar los productos en *stock*, alertar la fecha de vencimiento de productos y de alertar los posibles errores en la gestión de productos. Se debe tomar en cuenta que esta funcionalidad también es accesible para el usuario con perfil empleado y empleado sucursal.

***Endpoints* para crear reportes de productos**

El *backend* implementa *endpoints* que permiten el acceso al usuario con perfil administrador a generar reportes semanales, quincenales o mensuales sobre la gestión, compra y salida de productos, según se lo requiera. En los cuales la estructura de la información del reporte de gestión de productos presenta un título, código, categoría, nombre del producto, *stock* del producto, tipo de bodega, fecha y hora de emisión del reporte, además de la fecha y hora de caducidad del producto si lo requiere. El reporte de compra de productos presenta un título, código, categoría, nombre del producto, descripción, cantidad de compra, además de la fecha y hora de emisión del reporte, finalmente el reporte de salida de productos presenta un título, código, categoría, nombre del producto, descripción, cantidad de salida, además de la fecha y hora de emisión del reporte. Se debe tomar en cuenta que la funcionalidad para generar reportes de gestión de productos también es accesible para el usuario con perfil empleado y empleado sucursal.

En las siguientes **Fig. 2, Fig. 3 y Fig. 4** se presentan los perfiles que dispone el *backend* con sus respectivas acciones.

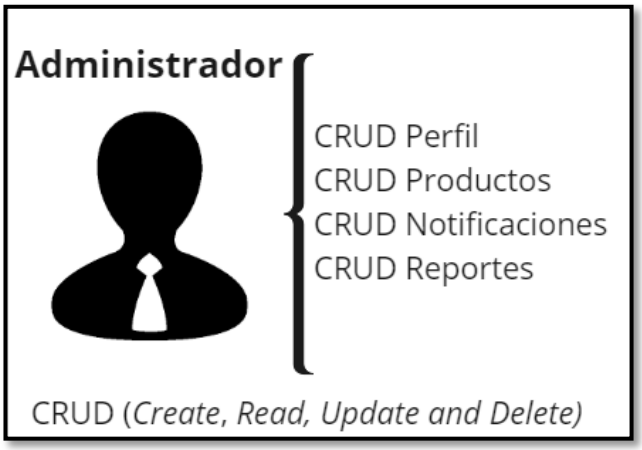


Fig. 2: Usuario administrador.

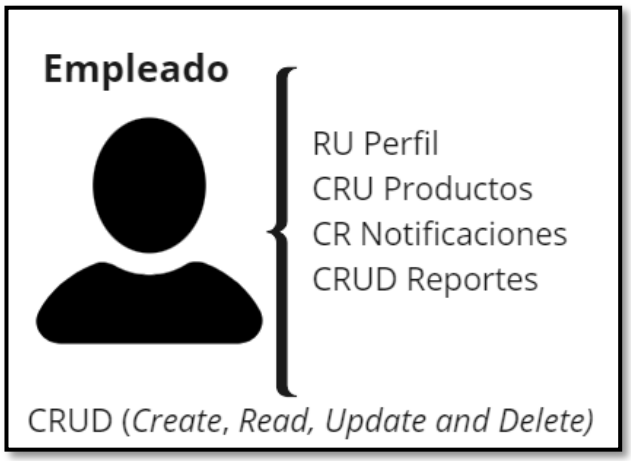


Fig. 3: Usuario empleado.

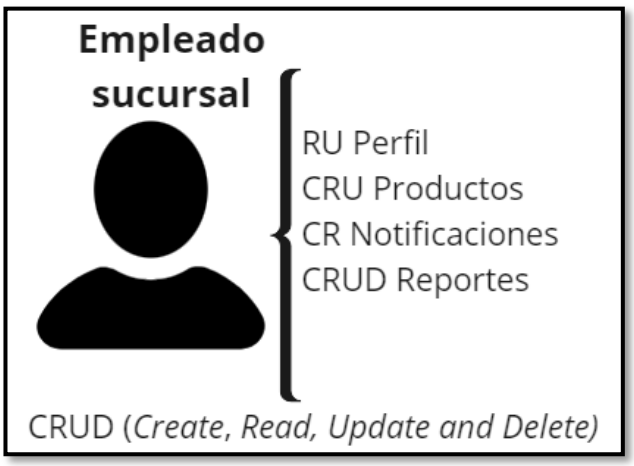


Fig. 4: Usuario empleado sucursal.

Modelamiento de datos con MongoDB

La información gestionada por el *backend* se almacena en la plataforma MongoDB, la cual mantiene, combina y retiene información simultáneamente mientras se realizan peticiones dentro de colecciones y documentos, proporcionando la versatilidad necesaria para soportar aplicaciones seguras de alto rendimiento [39]. Muestra de ello, en la **Fig. 5** se observan las colecciones principales que se han utilizado en el proyecto. Por último, el detalle de la base de datos completa se encuentra en el **ANEXO II** del escrito.

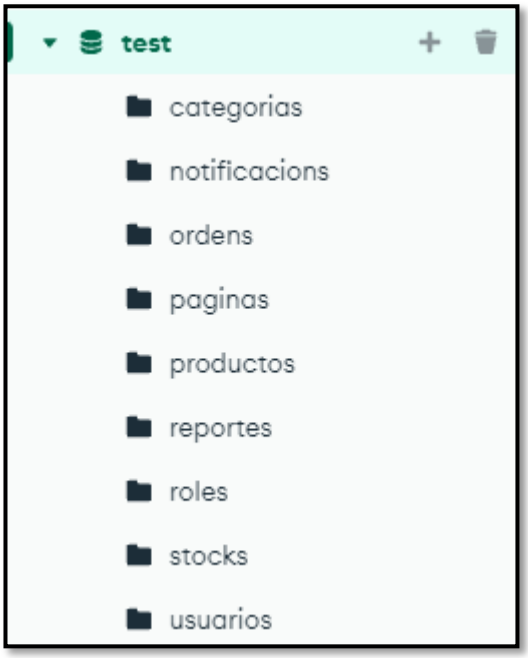


Fig. 5: Colecciones en MongoDB atlas.

Organización estructural del *backend*

El generar *endpoints* seguros, garantiza la fiabilidad y disponibilidad de los datos. Por lo cual el entorno de desarrollo que se ha utilizado es *Visual Studio Code*, además de bibliotecas que ayudan al desarrollo del *backend* en el marco de los lineamientos que proporciona una arquitectura *REST*. A continuidad la **Fig. 6** presenta la organización de archivos y directorios como parte de la codificación del *backend*.

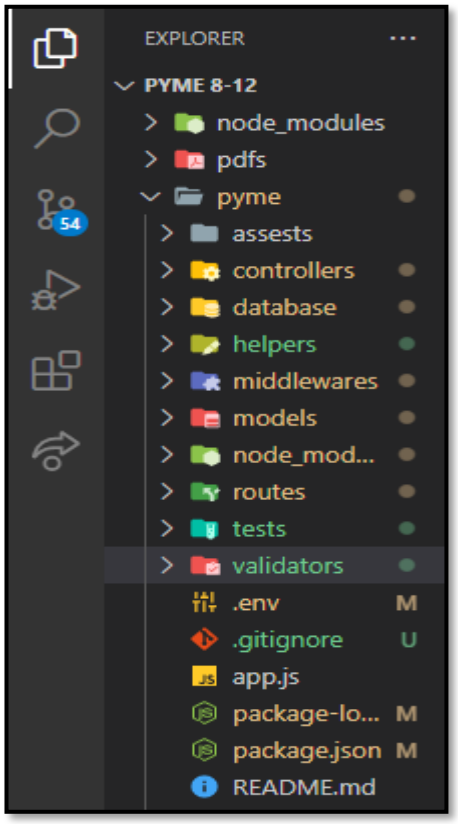


Fig. 6: Directorios y archivos como parte del proyecto *backend*.

Módulos para los usuarios

La **Fig. 7** exhibe los tipos de usuario con permisos correspondientes para acceder a sus respectivas acciones.

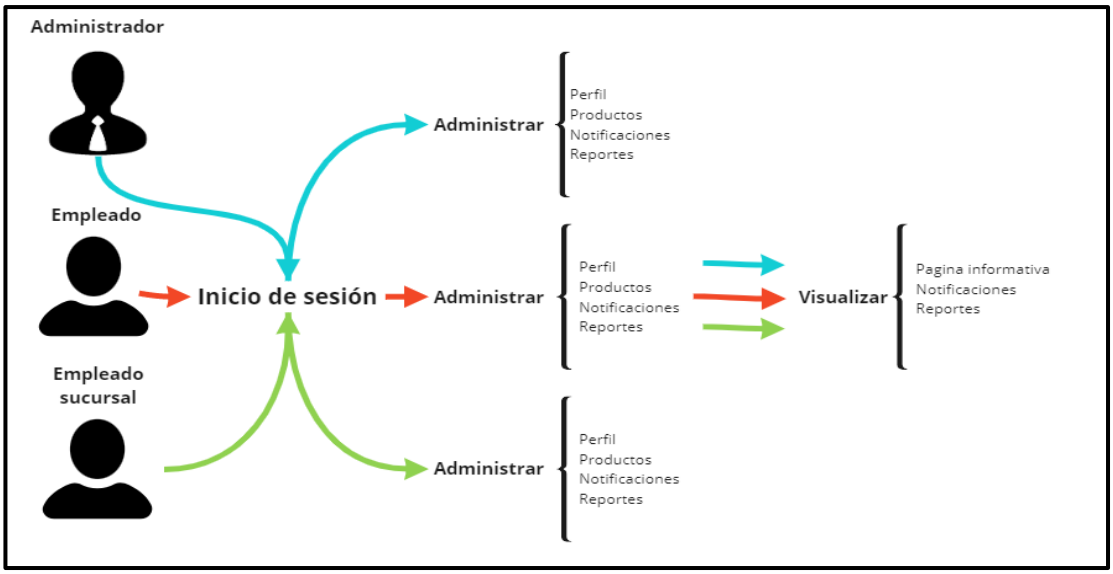


Fig. 7: Rol de cada usuario en el *backend*.

3.2 *Sprint* 1. Usuario administrador - Creación de *endpoints* y resultados

Como parte del *Sprint* 1 se establecen las siguientes tareas:

- Generar *endpoints* que presente una página informativa.
- Generar *endpoints* que permitan cerrar sesión, iniciar sesión y cambiar la contraseña.
- Generar *endpoints* que permitan modificar los datos personales.
- Generar *endpoints* que permitan la gestión de perfiles de usuarios.
- Generar *endpoints* que permitan crear productos.
- Generar *endpoints* que permitan observar notificaciones.
- Generar *endpoints* que permitan crear reportes de productos.

Generar *endpoints* que presente una página informativa.

En el *backend* se generan *endpoints* en conjunto con rutas *GET*, los cuales posibilitan el acceso a los datos al momento de realizar una petición previamente validada, dicha petición puede realizarse por un cliente *HTTP*, en el cual se presenta la información de la PYME como: nombre comercial de la empresa, misión, visión, entre otros como se presenta en la **Fig. 8**. Al igual que su respectiva prueba reflejada en la **Fig. 9**. Por otra parte, se inicia un análisis minucioso acerca de la forma en la que se ejecutan los *endpoints* y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del escrito.

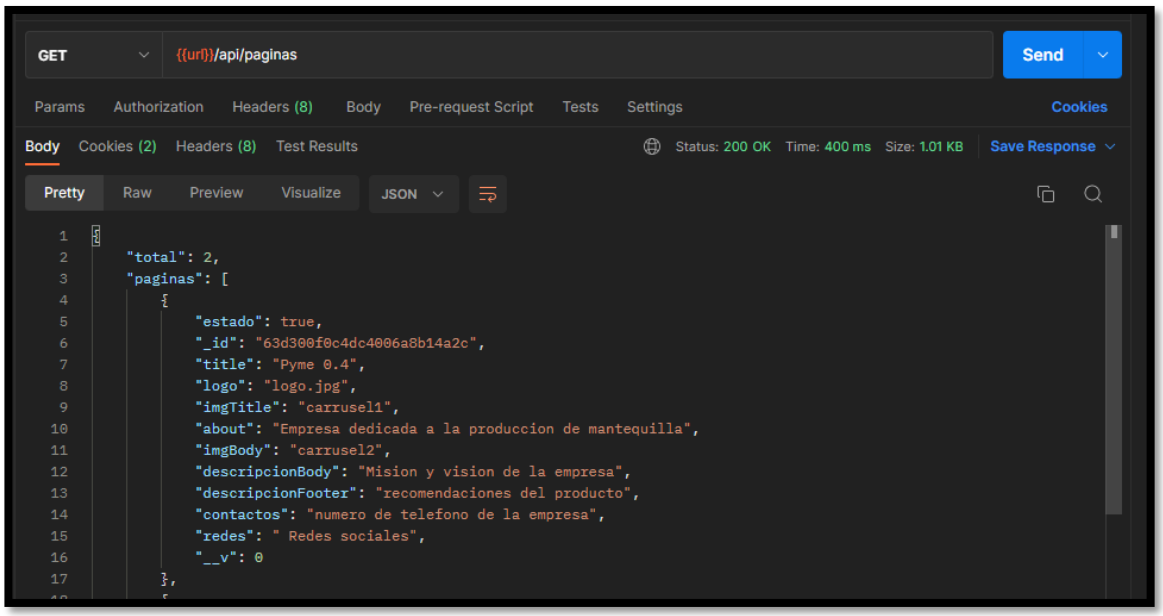


Fig. 8: Petición *GET* de página informativa.

```
C:\Windows\System32\cmd.exe
    at new Server (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:30:14)
    at Object.<anonymous> (C:\Users\lseb1\Documents\pyme 8-12\pyme\tests\task.js:4:14)
    at Module._compile (node:internal/modules/cjs/loader:1105:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:170:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:198:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:385:24)
    at async importModuleDynamicallyWrapper (node:internal/vm/module:437:15)
    at async formattedImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-utils.js:9:14)
    at async Object.exports.requireOrImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-u
tils.js:42:28)
    at async Object.exports.loadFilesAsync (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-ut
ils.js:100:20)
    at async singleRun (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run-helpers.js:125:3)
    at async Object.exports.handler (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run.js:370:5)


  Tasks API
    GET /api/paginas
      √ It should GET all home pages
      √ It should NOT GET all home pages


  2 passing (49ms)
```

Fig. 9: Testeo de la página informativa.

Generar *endpoints* que permitan cerrar sesión, iniciar sesión y cambiar la contraseña.

El *backend* posibilita un *login* completo donde se originan *endpoints* que permiten a los usuarios acceder a la información con mayor control y seguridad cuando se requiere consumir los datos almacenados en la base de datos. Además, la base de datos mencionada se encuentra previamente preparada para recibir y acumular la información requerida por los usuarios. Lo presentado anteriormente es posible gracias a que se crean *endpoints* en conjunto con una ruta *POST* la cual permite registrar datos en base a un formato establecido con campos obligatorios. A continuación, se ejecuta un *POST* encargado del *login* y la verificación de las credenciales del usuario de tal manera que permita el acceso a los recursos permitidos en base al rol que posea dicho usuario como se muestra en la **Fig. 10**, y en la **Fig. 11** se muestra el resultado de su respectivo testeo unitario. Finalmente se implementan un método *PUT* para realizar el cambio de contraseña validando que el usuario que realiza la acción tenga los permisos adecuados, como se muestra en la **Fig. 12**, además en la **Fig. 13** se muestra su respectivo testeo. Por otra parte, se ejecuta un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

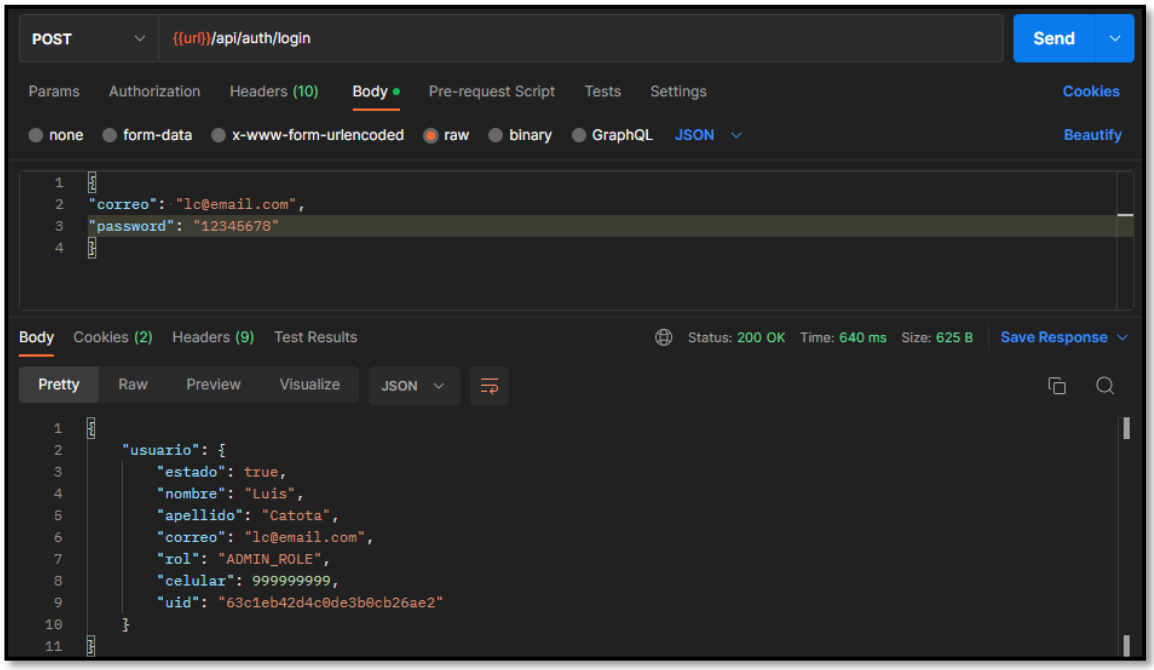


Fig. 10: *Login* completo.

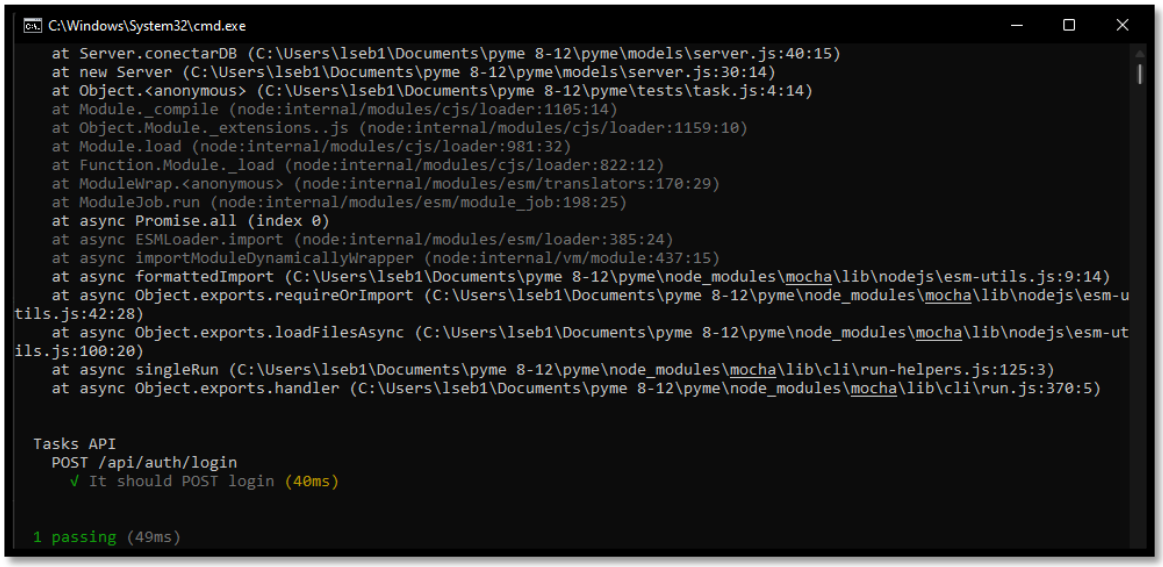


Fig. 11: Testeo del *Login*

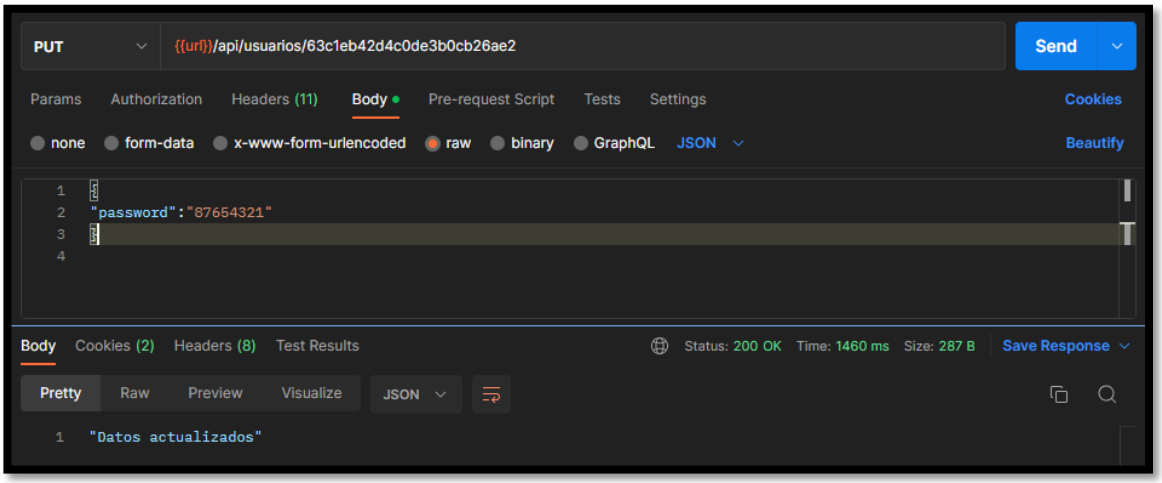


Fig. 12: Cambio de contraseña.

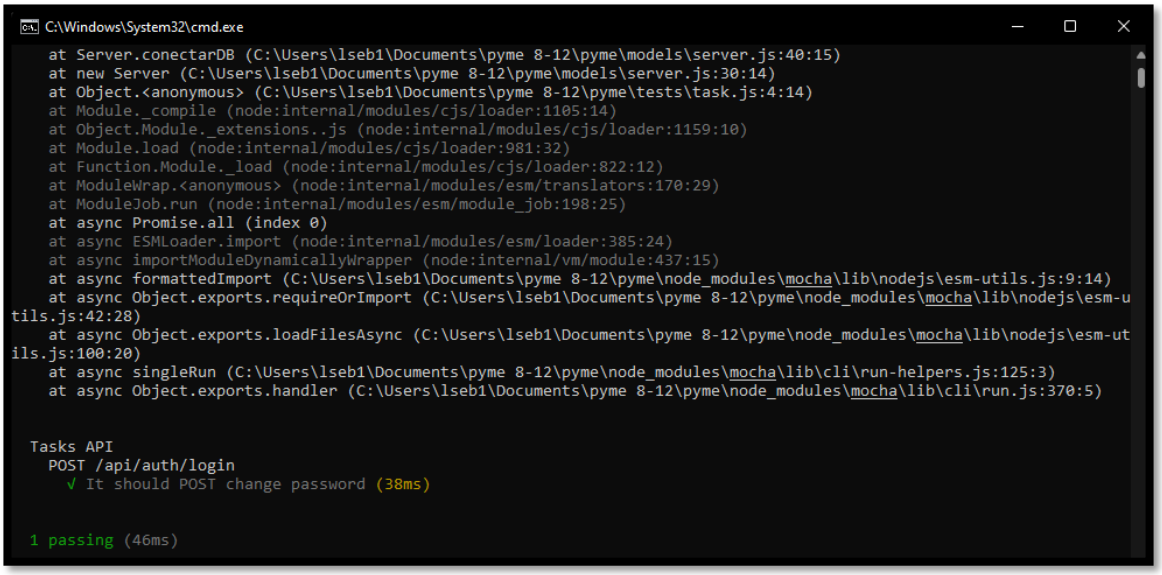


Fig. 13: Testeo de cambio de contraseña.

Generar varios *endpoints* que permita modificar los datos personales

Dentro del *backend* el usuario administrador es capaz de modificar su información personal por ende puede personalizar y cambiar los datos que ha registrado previamente. Estas acciones son posibles debido a métodos y rutas *PUT* las cuales permiten validar, editar, actualizar y observar los datos en función del ID y el *token* proporcionado, como se muestra en la **Fig. 14.** El resultado del testeo unitario se encuentra en la **Fig. 15**. Por otra parte, el detalle de la forma de ejecución de los *endpoint*s y sus respectivos testeos se detallan en el **ANEXO II** del presente escrito.

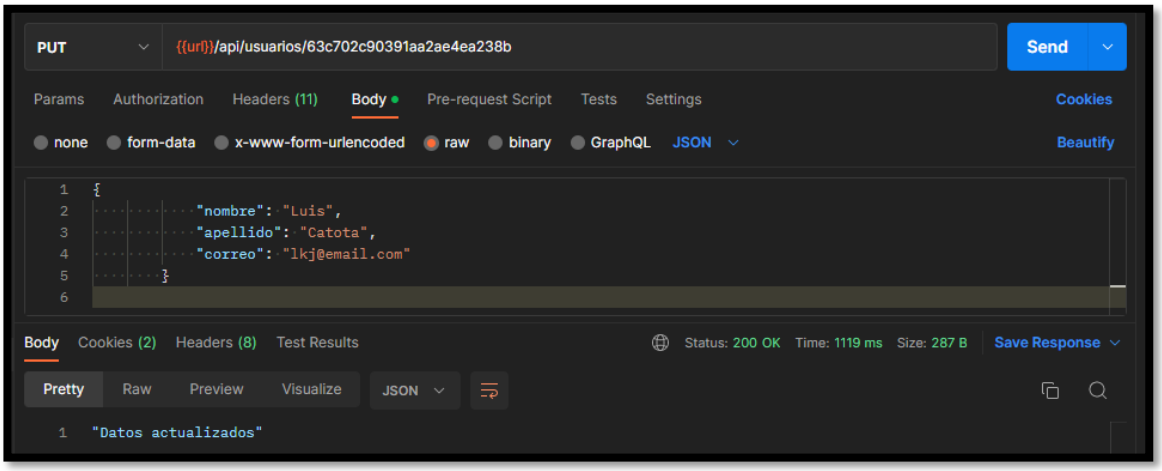


Fig. 14: Modificar información personal.

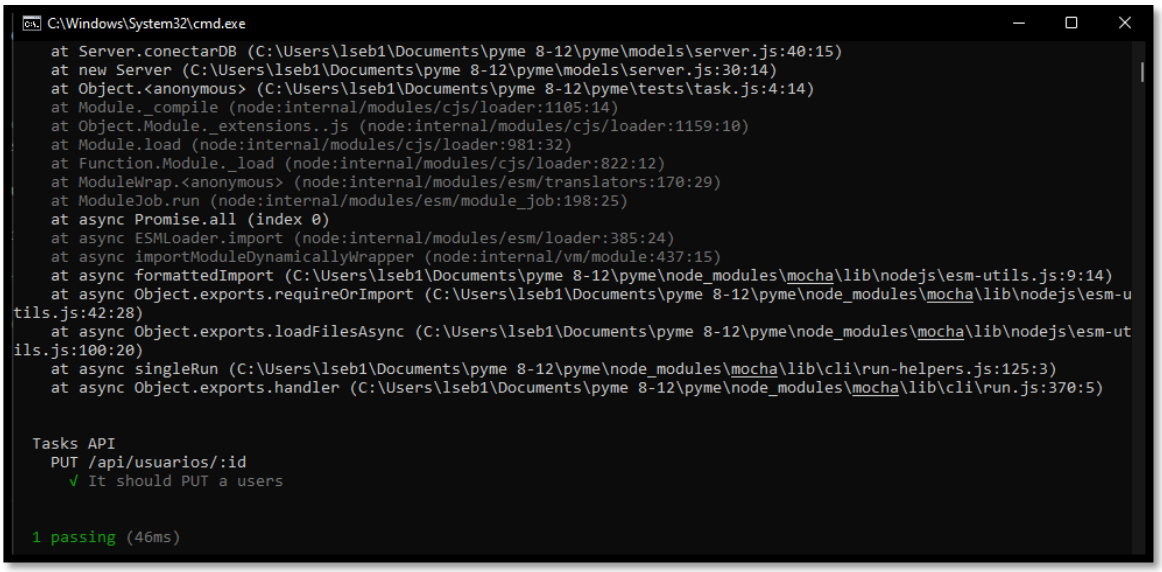


Fig. 15: Testeo de modificación de información personal.

Generar *endpoints* que permita la gestión de perfiles de usuarios

Dentro del *backend* el usuario con perfil administrador puede crear, listar, modificar, habilitar o deshabilitar y eliminar perfiles de usuario de acuerdo con sus necesidades. Por lo tanto, el usuario administrador no puede eliminar a un usuario si este se encuentra habilitado. Estas acciones son posibles debido a una ruta de tipo *GET* la cual obtiene información, una ruta de tipo *POST* con la misión de ingresar datos, una ruta de tipo *PUT* encargada de modificar los datos si son requeridos y finalmente una ruta *DELETE* con el propósito de cambiar de estado activo a inactivo a los usuarios, como se presenta en la **Fig. 16,** y el resultado del testeo unitario se presenta en la **Fig. 17**. Por otra parte, se ejecuta un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

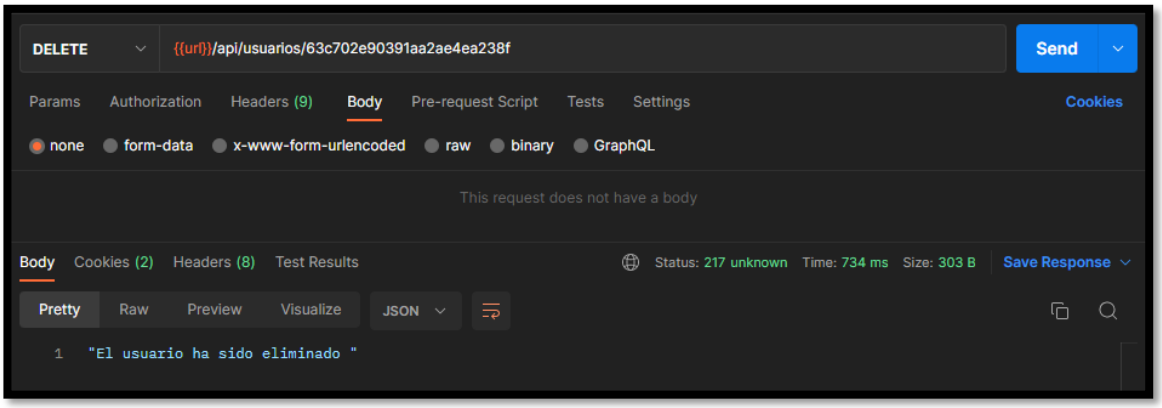


Fig. 16: Gestión de perfiles de usuario.

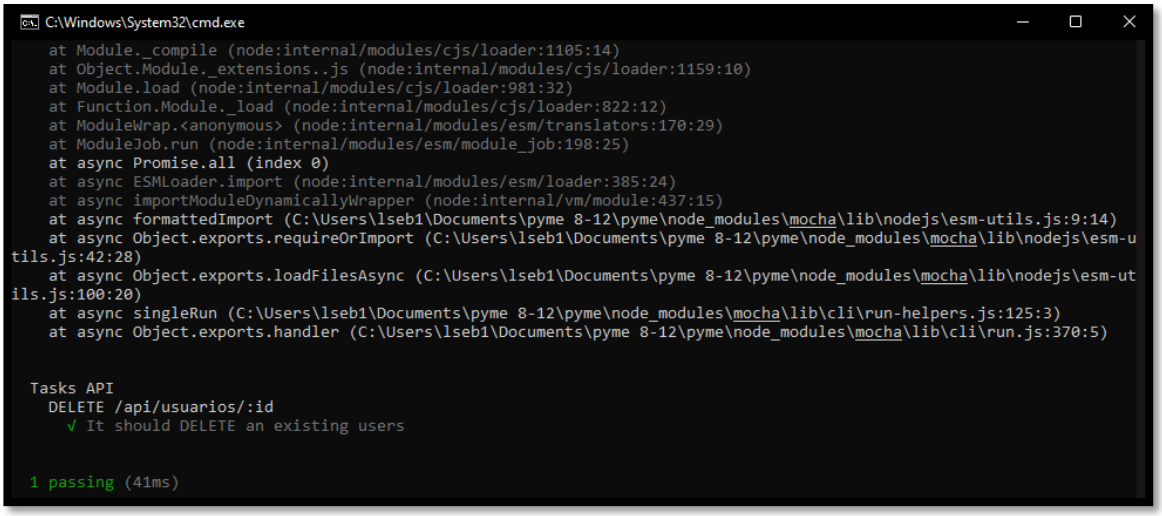


Fig. 17: Testeo de gestión de perfiles.

Generar *endpoints* que permitan crear productos

En el *backend* el usuario administrador tiene la potestad de utilizar *endpoints* que le permitan gestionar productos, en conjunto con las acciones de listar, ingresar, modificar y eliminar productos ingresados. Estas acciones son posibles debido a una ruta *GET* la cual obtiene información, una de tipo *POST* destinada a la recolección de datos, una ruta *PUT* encargada de cambiar los datos según la necesidad del usuario y finalmente una ruta *DELETE* con el propósito de eliminar los datos. Es importante mantener un adecuado ingreso de productos mediante una estructura con campos obligatorios como el nombre, categoría, precio fecha y hora de ingreso o salida, fecha de caducidad si es necesaria, código y descripción, los cuales aseguran la correcta ejecución de cada una de las peticiones descritas anteriormente, como se muestra en la **Fig. 18.** Su respectivo testeo unitario se muestra en la **Fig.19**. Por otra parte, se genera un análisis minucioso acerca de la forma en la que se ejecutan los *endpoints* y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

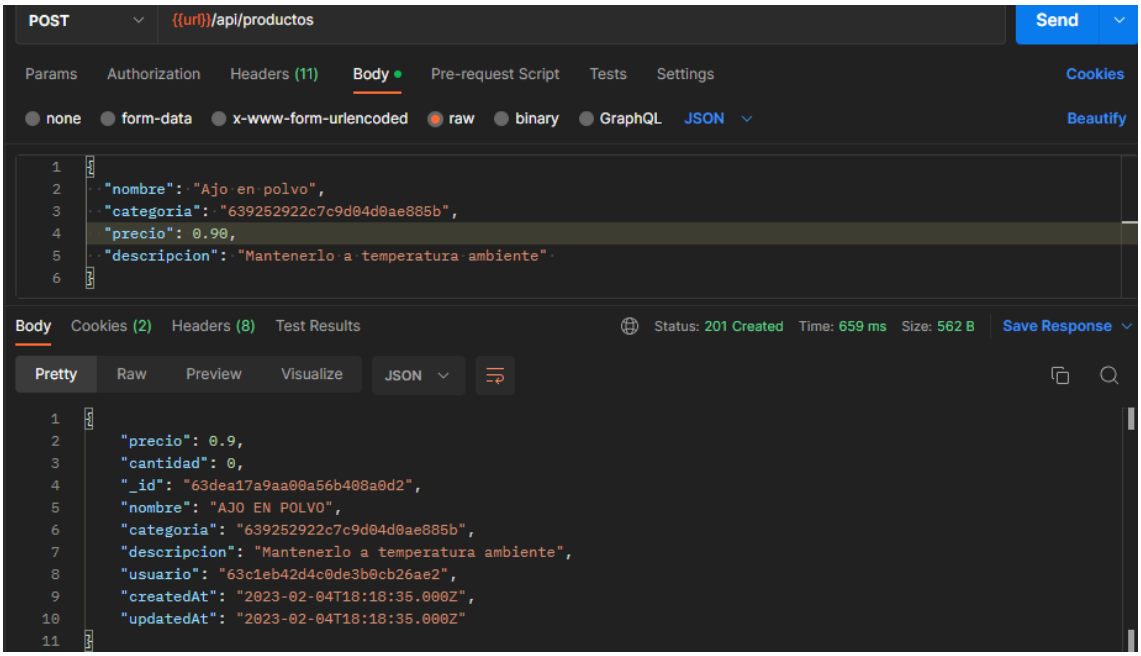


Fig. 18: Gestión de productos.

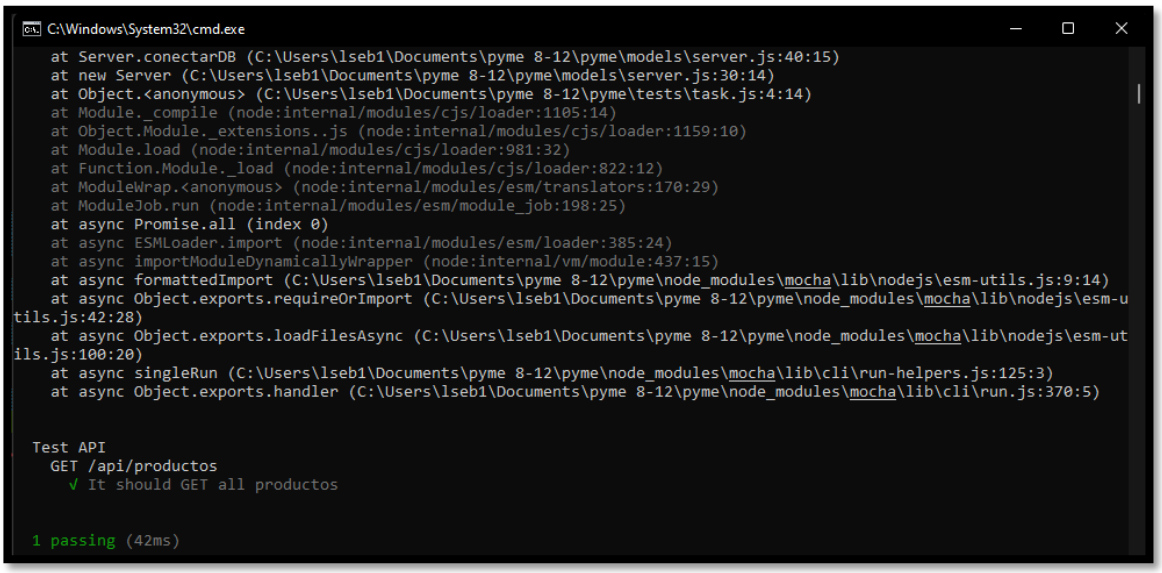


Fig.19: Testeo de gestión y surtido de productos.

Generar *endpoints* que permitan observar notificaciones

Dentro del *backend* el usuario administrador puede observar notificaciones con respecto a la gestión de productos en *stock*, alertas de productos próximos a su fecha de caducidad y errores en la gestión de productos de acuerdo con sus necesidades. Las acciones mencionadas se realizan a través de peticiones *GET*, como se muestra en la **Fig. 20,** con su respectivo testeo unitario como se observa en la **Fig. 21**. Por otra parte, se crea un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

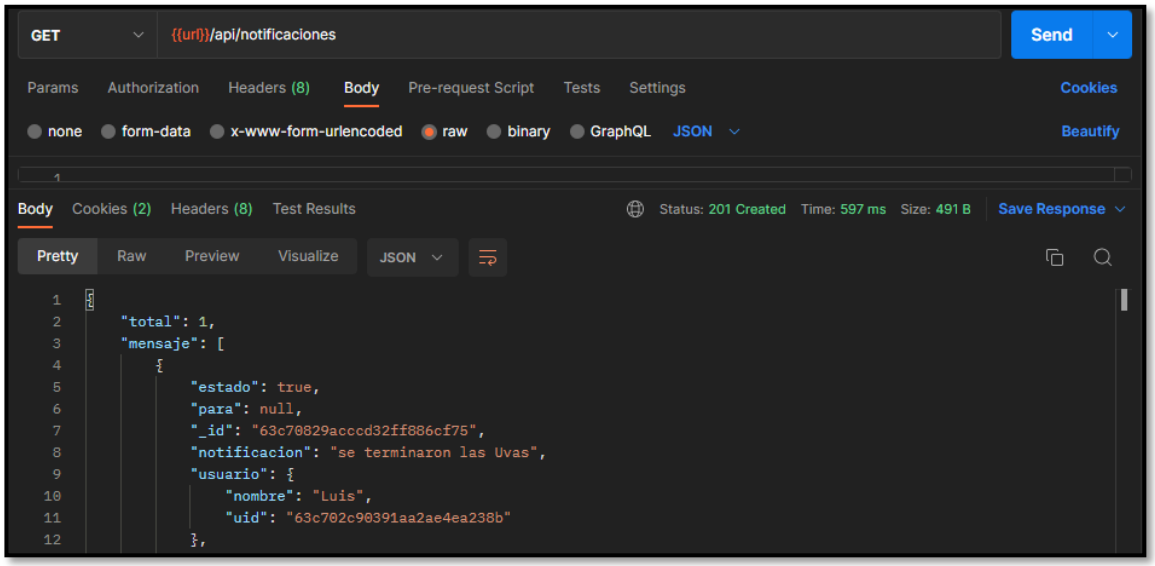


Fig. 20: Observación de notificaciones.

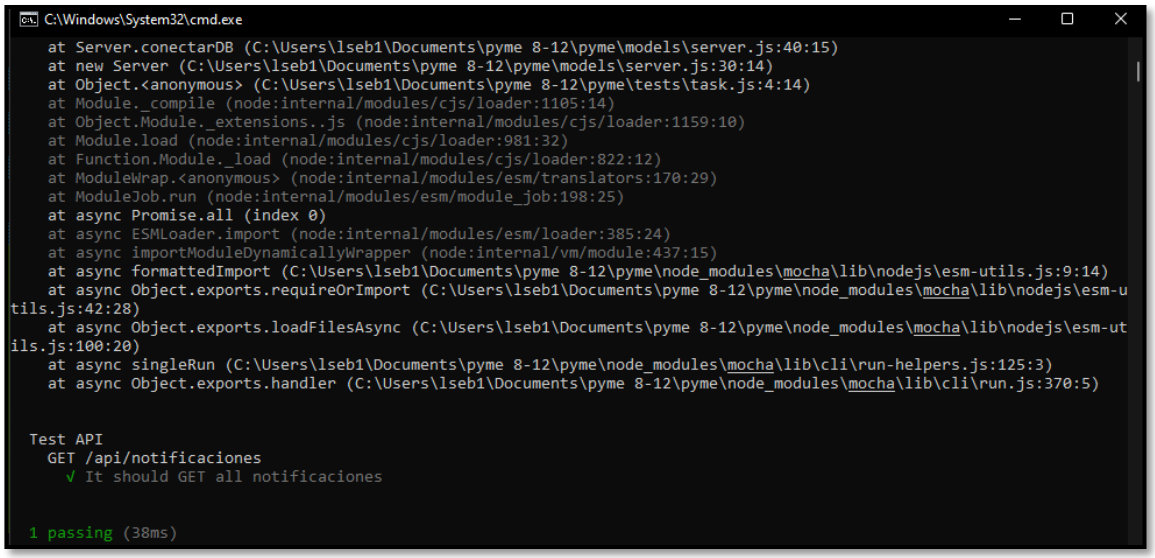


Fig. 21: Testeo para observar las notificaciones.

Generar *endpoints* que permitan crear reportes de productos

En el *backend* el usuario con perfil administrador puede crear reportes semanales, quincenales o mensuales sobre el *stock* de los productos, ya que existe un *endpoint* que le permite realizar dichos reportes, debido a una ruta de tipo *GET* la cual obtiene la información de la base de datos tal como se presenta en la **Fig. 22**. Su respectivo testeo unitario se muestra en la **Fig. 23**. Por otra parte, se realiza un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

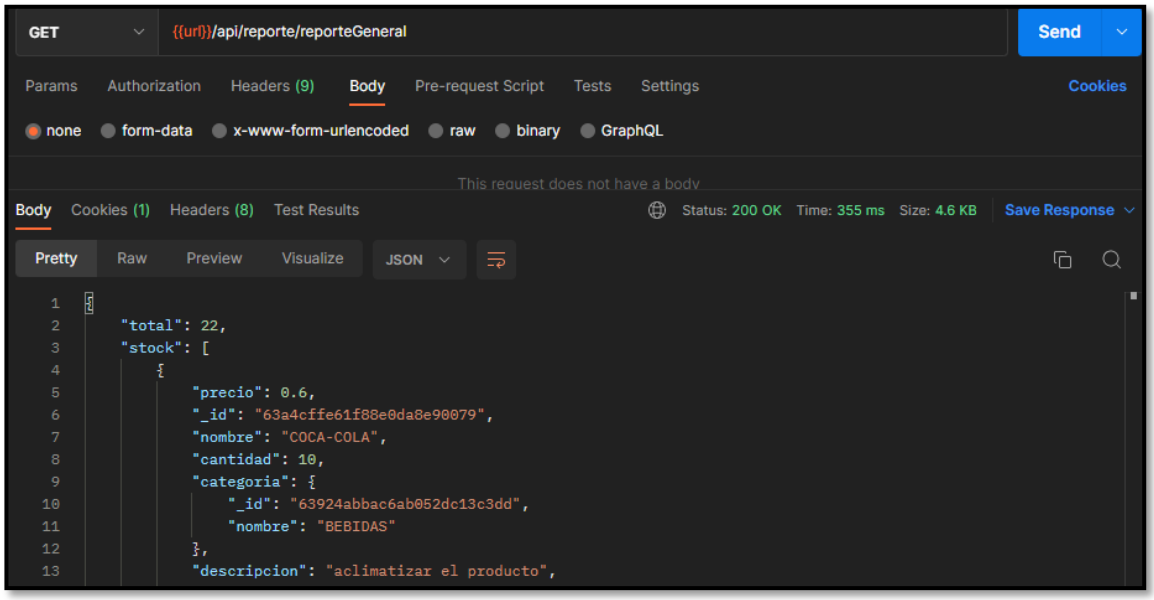


Fig. 22: Observación de reportes.

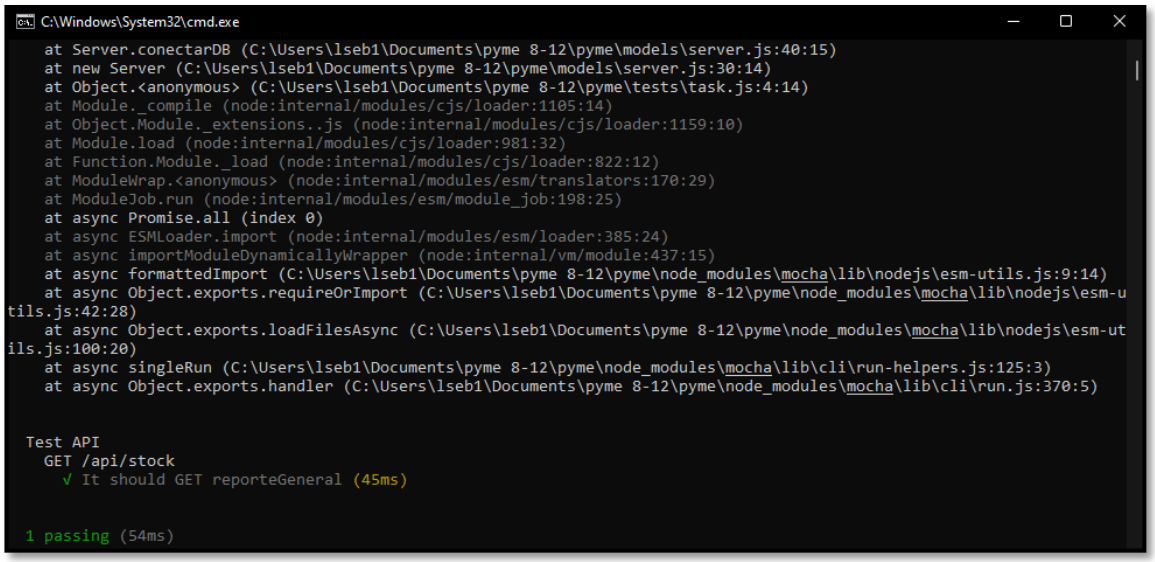


Fig. 23: Testeo para observar reportes.

3.3 *Sprint* 2. Creación y resultado de *endpoints* para el cliente empleado sucursal

Como parte del *Sprint* 2 se establecen las siguientes tareas:

- Generar *endpoints* para modificar información personal.
- Generar *endpoints* que permitan el ingreso de productos.
- Generar *endpoints* para observar notificaciones.
- Generar *endpoints* que permitan crear reportes.

Generar varios *endpoints* para modificar información personal

El cliente empleado sucursal es capaz de modificar su información personal por ende puede personalizar y cambiar los datos registrados previamente. Estas acciones son posibles debido a métodos y rutas de tipo *PUT* las cuales permiten validar, editar, actualizar los datos en función del ID y el *token* proporcionados, como se muestra en la **Fig. 24**, con su respectivo testeo unitario como se muestra en la **Fig. 25**. Por otra parte, se ejecuta un análisis minucioso acerca de la forma en la que se realizan los *endpoint*s y sus respectivos testeos como se detallan en el **ANEXO II** del presente escrito.

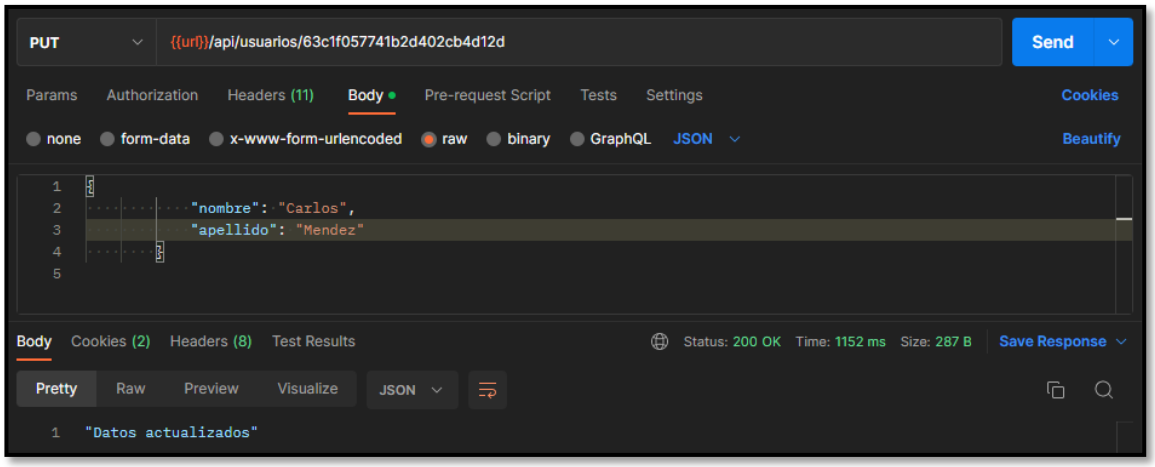


Fig. 24: Modificar datos del empleado sucursal.

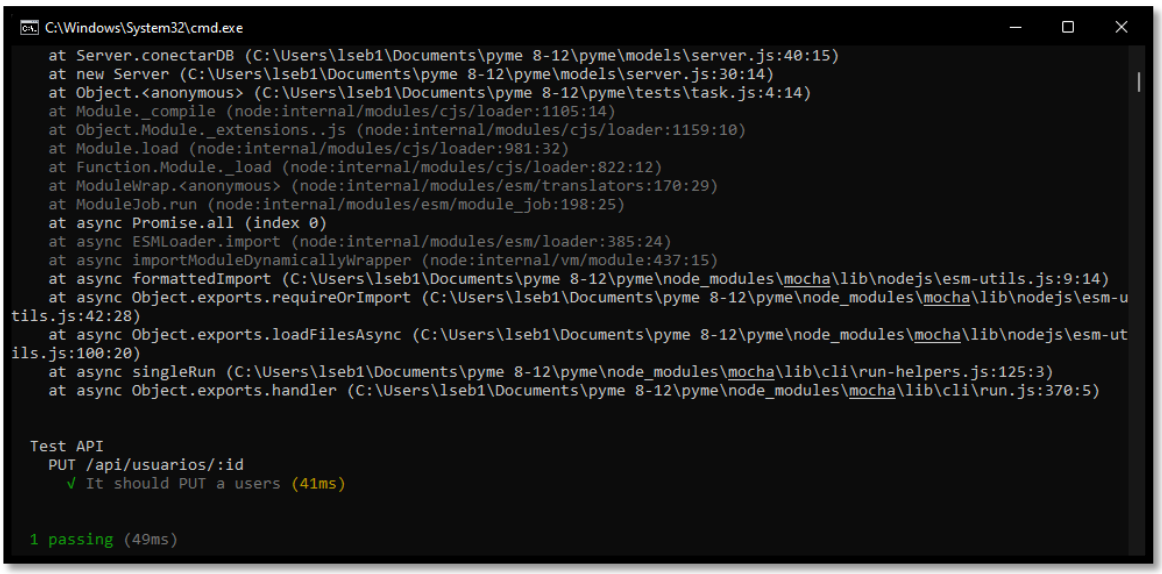


Fig. 25: Testeo para modificar datos del empleado sucursal.

Generar *endpoints* que permita el ingreso de productos

El usuario empleado sucursal tiene la potestad de utilizar los *endpoints* que le permitan crear productos, conjuntamente con las acciones de listar, modificar e ingresar productos al *Stock*. Estas acciones son posibles debido a una ruta de tipo *POST* destinada a la modificación de

la cantidad en existencia de los productos, tomando en cuenta las validaciones respectivas, como se muestra en la **Fig. 26.** El resultado de su respectivo testeo unitario se evidencia en la **Fig. 27**. Por otra parte, el detalle de la forma de ejecución de los *endpoint*s y sus respectivos testeos se detallan en el **ANEXO II** del presente escrito.

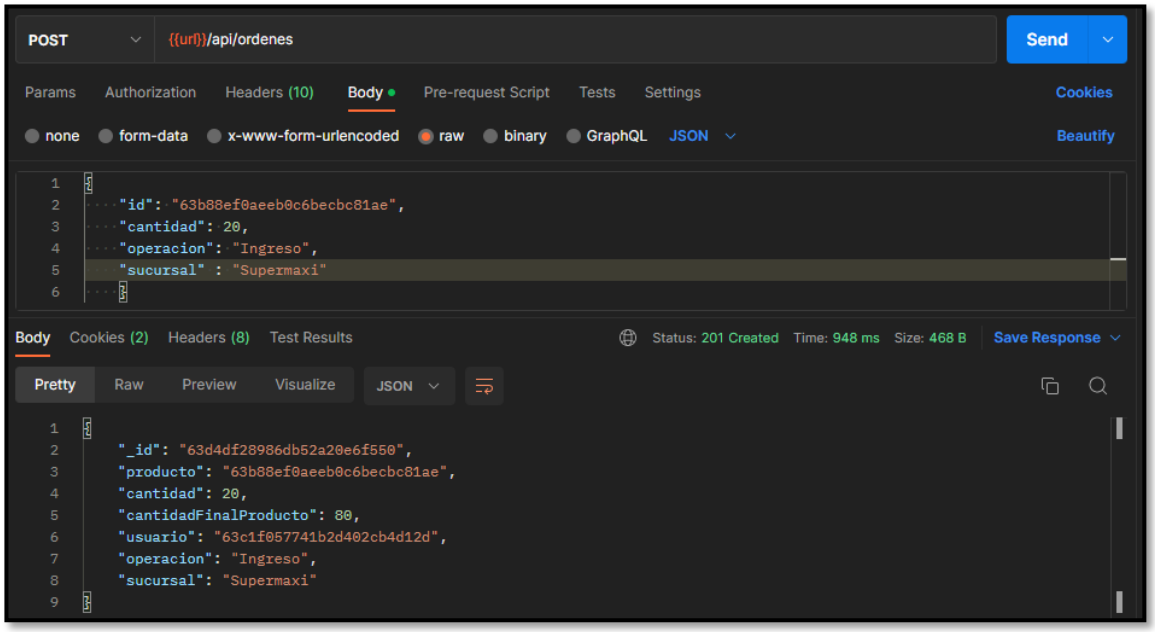


Fig. 26: Gestión e ingreso de productos.

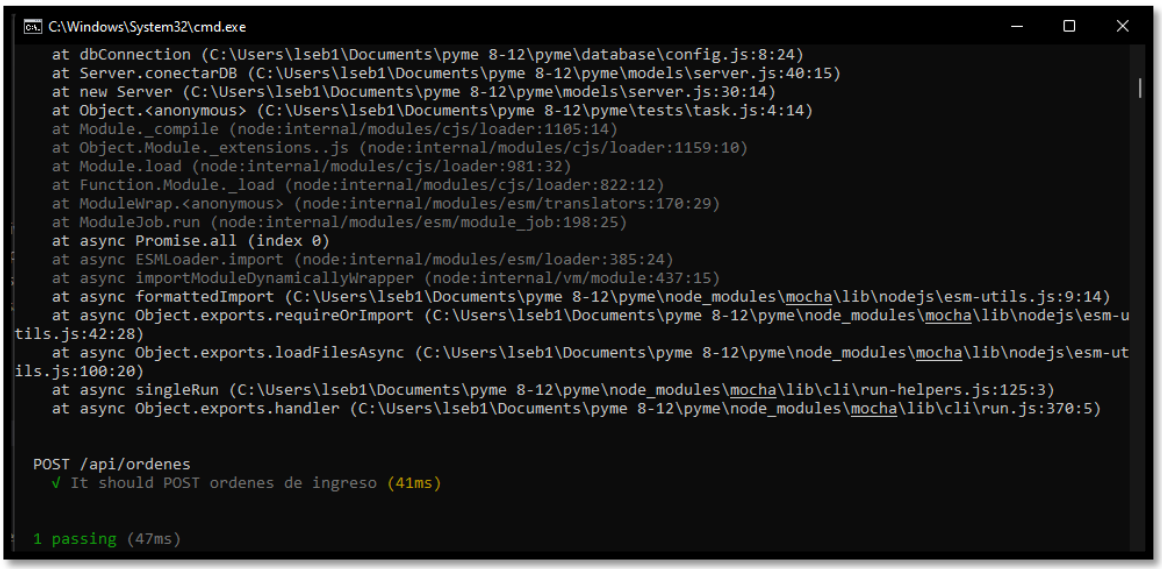


Fig. 27: Testeo para la gestión e ingreso de productos.

Generar *endpoints* para observar notificaciones

En el *backend* el usuario con perfil empleado sucursal puede observar notificaciones con respecto a la gestión de productos en *stock*, alertas de productos próximos a su fecha de caducidad y errores en la gestión de productos de acuerdo con sus necesidades. Las acciones mencionadas se realizan a través de peticiones *GET*, como se muestra en la **Fig.**

28, al igual que su respectivo testeo unitario presente en la **Fig. 29**. Por otra parte, el detalle de la forma de ejecución de los *endpoints* y sus respectivos testeos se detallan en el **ANEXO II** del presente escrito.

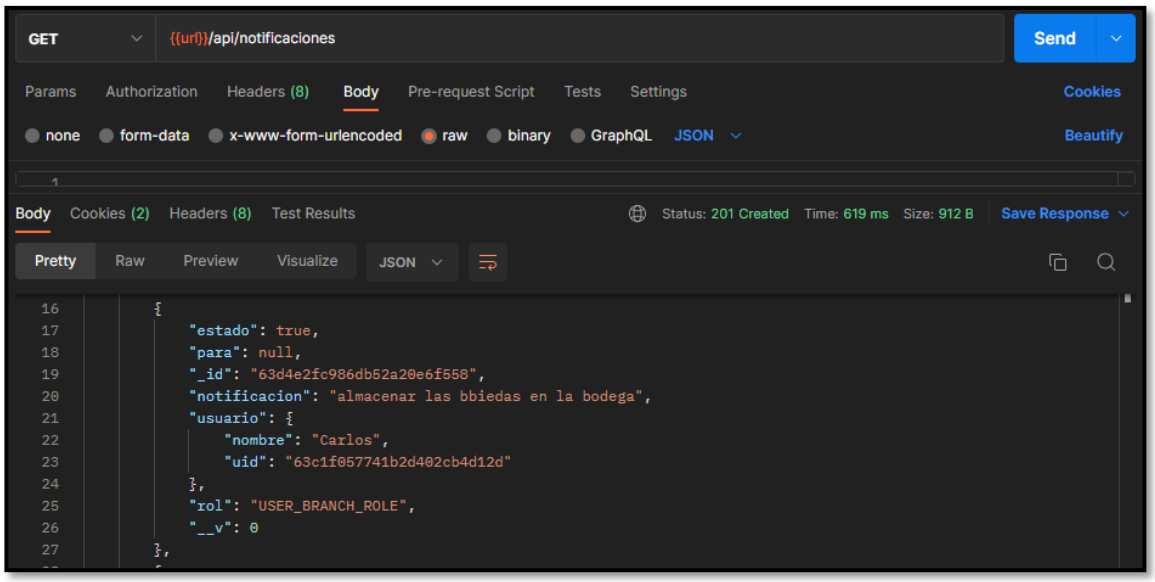


Fig. 28: Observar notificaciones.

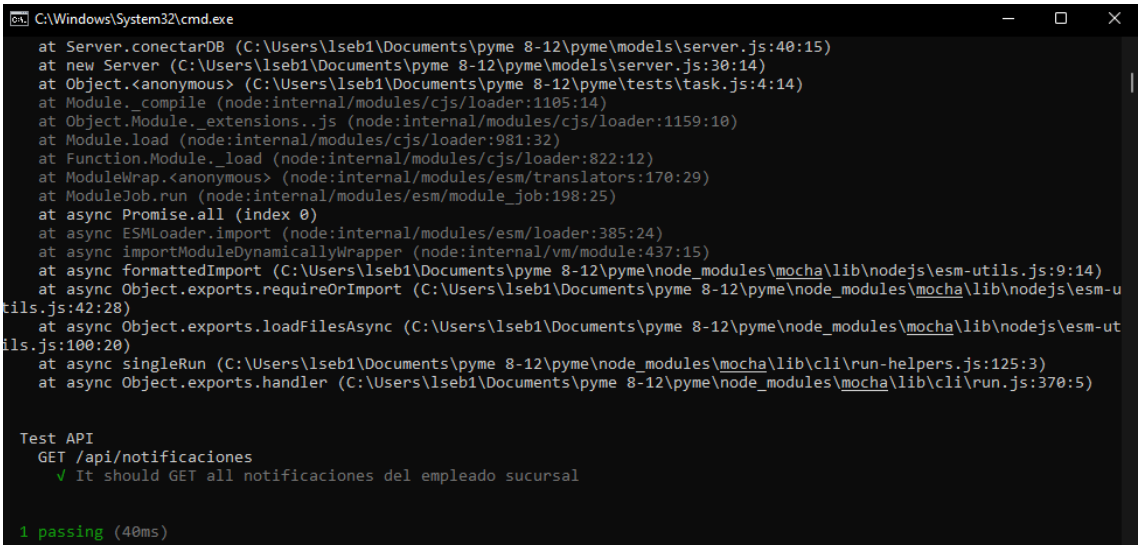


Fig. 29: Testeo de notificación.

Generar *endpoints* que permitan crear reportes de productos

En el *backend* el usuario con perfil empleado sucursal puede crear reportes semanales, quincenales o mensuales sobre el *stock* de los productos en existencia, ya que existe un *endpoint* que le permite realizar dichos reportes, debido a una ruta de tipo *GET* la cual obtiene la información de la base de datos tal como se presenta en la **Fig. 30,** con su respectivo testeo unitario como se muestra en la **Fig. 31**. Por otra parte, se ejecuta un

análisis minucioso acerca de la forma en la que se realizan los *endpoints* y sus respectivos testeos que a su vez se detallan en el **ANEXO II** del presente escrito.

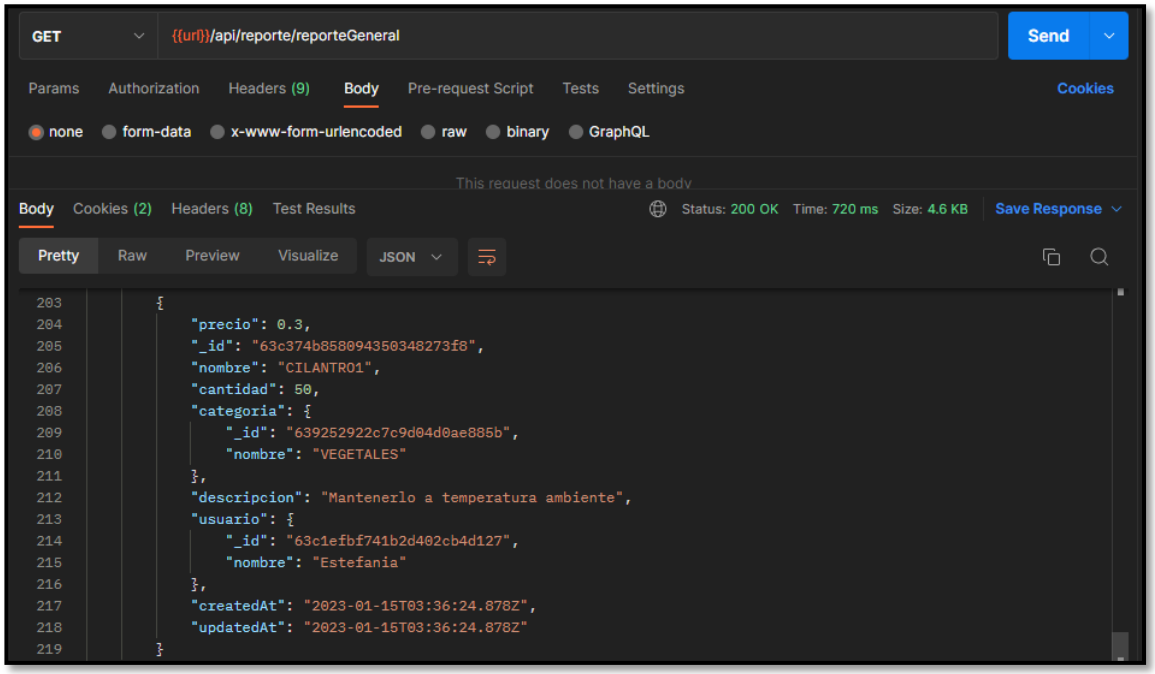


Fig. 30: Observar reporte.

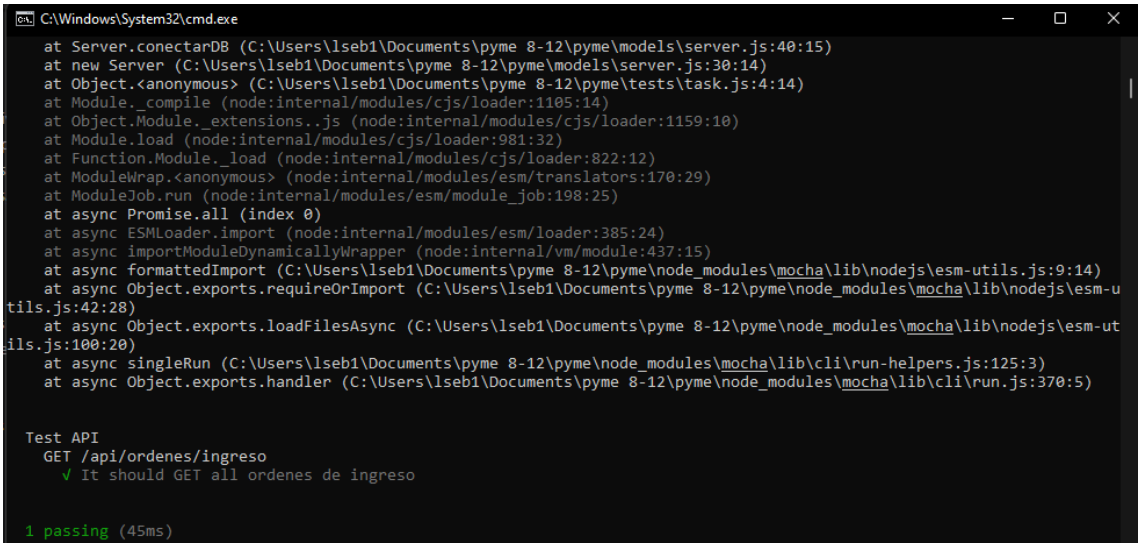


Fig. 31: Testeo para observar reporte.

3.4 *Sprint* 3. Creación y resultado de *endpoints* para el cliente con perfil empleado.

Como parte del *Sprint* 3 se establecen las siguientes tareas:

- Generar *endpoints* para modificar información personal.
- Generar *endpoints* que permitan generar notificaciones.
- Generar *endpoints* que permitan la distribución de los productos.
- Generar *endpoints* para crear reportes de productos.

Generar varios *endpoints* para modificar información personal

El cliente empleado es capaz de modificar su información personal por ende puede personalizar y cambiar los datos registrados previamente. Estas acciones son posibles debido a métodos y rutas de tipo *PUT* las cuales permiten validar, editar y actualizar los datos en función del ID y el *token* proporcionado, como se muestra en la **Fig. 32**. Con su respectivo testeo unitario como se muestra en la **Fig. 33**. Por otra parte, se inicia un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos, los cuales se detallan en el **ANEXO II** del presente escrito.

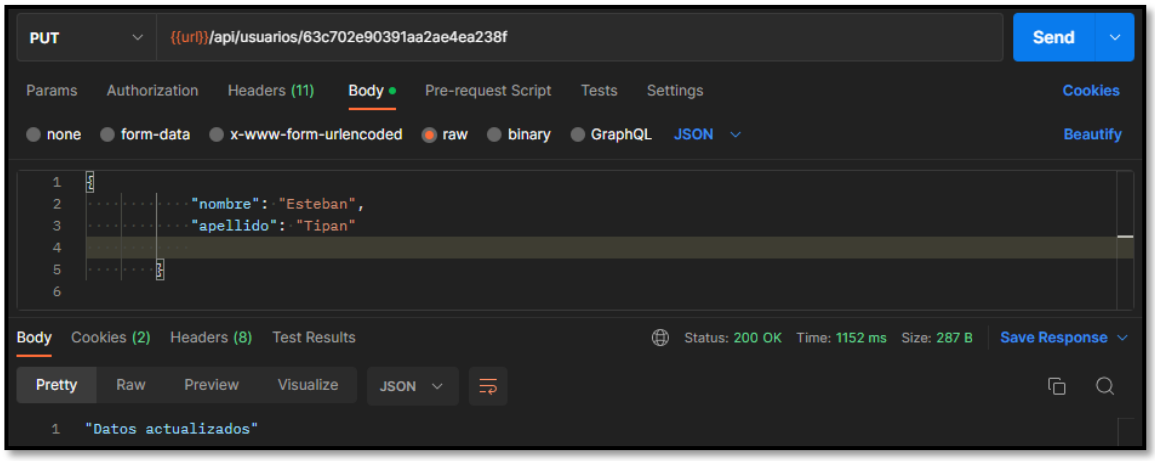


Fig. 32 Modificar información personal.

```
C:\Windows\System32\cmd.exe
    at dbConnection (C:\Users\lseb1\Documents\pyme 8-12\pyme\database\config.js:8:24)
    at Server.conectarDB (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:40:15)
    at new Server (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:30:14)
    at Object.<anonymous> (C:\Users\lseb1\Documents\pyme 8-12\pyme\tests\task.js:4:14)
    at Module._compile (node:internal/modules/cjs/loader:1105:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:170:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:198:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:385:24)
    at async importModuleDynamicallyWrapper (node:internal/vm/module:437:15)
    at async formattedImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-utils.js:9:14)
    at async Object.exports.requireOrImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-u
tils.js:42:28)
    at async Object.exports.loadFilesAsync (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-ut
ils.js:100:20)
    at async singleRun (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run-helpers.js:125:3)
    at async Object.exports.handler (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run.js:370:5)


  PUT /api/usuarios/:id
    √ It should PUT a users (50ms)


  1 passing (56ms)
```

Fig. 33: Testeo para modificar información personal**.**

Generar *endpoints* que permitan observar notificaciones

En el *backend* el usuario con perfil empleado puede recibir notificaciones con respecto a la gestión de productos en *stock* y errores en la gestión de productos de acuerdo con sus necesidades. Este método se implementa a través de peticiones de tipo *GET* tal como se presenta en la **Fig. 34,** además de su respectivo testeo unitario presente en la **Fig. *35***. Por otra parte, se inicia un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos, los cuales se detallan en el **ANEXO II** del presente escrito.

```
POST    {{url}}/api/notificaciones                                   Send
Params  Authorization  Headers (10)  Body •  Pre-request Script  Tests  Settings  Cookies
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON                Beautify
{
  "notificacion": "almacenar las bbiedas en la bodega",
  "para": "63c1efbf741b2d402cb4d127"
}
Body  Cookies (2)  Headers (8)  Test Results   Status: 201 Created  Time: 437 ms  Size: 477 B  Save Response
Pretty  Raw  Preview  Visualize  JSON
{
  "estado": true,
  "para": "63c1efbf741b2d402cb4d127",
  "_id": "63d4e2fc986db52a20e6f558",
  "notificacion": "almacenar las bbiedas en la bodega",
  "usuario": "63c1f057741b2d402cb4d12d",
  "rol": "USER_BRANCH_ROLE",
  "__v": 0
}
```

Fig. 34: Observación de notificaciones.

```
C:\Windows\System32\cmd.exe
    at Server.conectarDB (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:40:15)
    at new Server (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:30:14)
    at Object.<anonymous> (C:\Users\lseb1\Documents\pyme 8-12\pyme\tests\task.js:4:14)
    at Module._compile (node:internal/modules/cjs/loader:1105:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:170:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:198:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:385:24)
    at async importModuleDynamicallyWrapper (node:internal/vm/module:437:15)
    at async formattedImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-utils.js:9:14)
    at async Object.exports.requireOrImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-u
tils.js:42:28)
    at async Object.exports.loadFilesAsync (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-ut
ils.js:100:20)
    at async singleRun (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run-helpers.js:125:3)
    at async Object.exports.handler (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run.js:370:5)


  Test API
    GET /api/notificaciones
      √ It should GET all notificaciones del empleado


  1 passing (38ms)
```

Fig. 35: Testeo para observar notificaciones.

Generar *endpoints* que permitan la distribución de productos

El usuario empleado tiene la potestad de utilizar los *endpoints* que le permitan la salida productos, conjuntamente con las acciones de listar, modificar e ingresar productos al *Stock*. Estas acciones son posibles debido a una ruta de tipo *POST* destinada a la modificación de la cantidad en existencia de los productos, tomando en cuenta las validaciones respectivas, como se muestra en la **Fig. 36**. El resultado de su respectivo testeo unitario se presenta en la **Fig. 37**. Por otra parte, se inician un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos, los cuales se detallan en el **ANEXO II** del presente escrito.

```
POST    {{url}}/api/ordenes                                                    Send
Params  Authorization  Headers (10)  Body •  Pre-request Script  Tests  Settings  Cookies
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON                Beautify
1  {
2      "id": "63d30a2abf19000c84b218bf",
3      "cantidad": "10",
4
5      "operacion": "Salida",
6      "sucursal": "tiendita"
Body  Cookies (2)  Headers (8)  Test Results   Status: 201 Created  Time: 784 ms  Size: 466 B  Save Response
Pretty  Raw  Preview  Visualize  JSON
1  {
2      "_id": "63d4edb8986db52a20e6f57e",
3      "producto": "63d30a2abf19000c84b218bf",
4      "cantidad": 10,
5      "cantidadFinalProducto": 30,
6      "usuario": "63c1efbf741b2d402cb4d127",
7      "operacion": "Salida",
8      "sucursal": "tiendita"
```

Fig. 36: Salida de productos.

```
C:\Windows\System32\cmd.exe
    at Server.conectarDB (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:40:15)
    at new Server (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:30:14)
    at Object.<anonymous> (C:\Users\lseb1\Documents\pyme 8-12\pyme\tests\task.js:4:14)
    at Module._compile (node:internal/modules/cjs/loader:1105:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:170:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:198:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:385:24)
    at async importModuleDynamicallyWrapper (node:internal/vm/module:437:15)
    at async formattedImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-utils.js:9:14)
    at async Object.exports.requireOrImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-u
tils.js:42:28)
    at async Object.exports.loadFilesAsync (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-ut
ils.js:100:20)
    at async singleRun (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run-helpers.js:125:3)
    at async Object.exports.handler (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run.js:370:5)


  Test API
    POST /api/ordenes
      √ It should POST ordenes salida (44ms)


  1 passing (54ms)
```

Fig. 37: Testeo de salida de productos.

Generar *endpoints* que permitan crear reportes de productos

En el *backend* el usuario con perfil empleado puede crear reportes semanales, quincenales o mensuales sobre el *stock* de los productos en existencia, ya que existe un *endpoint* que le permite realizar dichos reportes, debido a una ruta de tipo *GET* la cual obtiene la información de la base de datos tal como se presenta en la **Fig. 38**. Con su respectivo testeo unitario como se muestra en la **Fig. 39**. Por otra parte, se inicia un análisis minucioso acerca de la forma en la que se ejecutan los *endpoint*s y sus respectivos testeos, los cuales se detallan en el **ANEXO II** del presente escrito.

```
GET    {{url}}/api/ordenes/salida                                   Send
Params  Authorization  Headers (8)  Body  Pre-request Script  Tests  Settings  Cookies
none  form-data  x-www-form-urlencoded  raw  binary  GraphQL
This request does not have a body
Body  Cookies (2)  Headers (8)  Test Results   Status: 201 Created  Time: 465 ms  Size: 2.59 KB  Save Response
Pretty  Raw  Preview  Visualize  JSON
{
    "total": 10,
    "ordenes": [
        {
            "_id": "63c7039c0391aa2ae4ea23a7",
            "productoNombre": "PERAS",
            "producto": "63c703280391aa2ae4ea2398",
            "cantidad": 30,
            "cantidadFinalProducto": 70,
            "usuario": {
                "_id": "63c7029f0391aa2ae4ea2387",
                "nombre": "peter"
            },
            "operacion": "Salida",
            "sucursal": "sucursal norte"
        },
```

Fig. 38: Observación de reporte de salidas.

```
C:\Windows\System32\cmd.exe
    at dbConnection (C:\Users\lseb1\Documents\pyme 8-12\pyme\database\config.js:8:24)
    at Server.conectarDB (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:40:15)
    at new Server (C:\Users\lseb1\Documents\pyme 8-12\pyme\models\server.js:30:14)
    at Object.<anonymous> (C:\Users\lseb1\Documents\pyme 8-12\pyme\tests\task.js:4:14)
    at Module._compile (node:internal/modules/cjs/loader:1105:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1159:10)
    at Module.load (node:internal/modules/cjs/loader:981:32)
    at Function.Module._load (node:internal/modules/cjs/loader:822:12)
    at ModuleWrap.<anonymous> (node:internal/modules/esm/translators:170:29)
    at ModuleJob.run (node:internal/modules/esm/module_job:198:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:385:24)
    at async importModuleDynamicallyWrapper (node:internal/vm/module:437:15)
    at async formattedImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-utils.js:9:14)
    at async Object.exports.requireOrImport (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-u
tils.js:42:28)
    at async Object.exports.loadFilesAsync (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\nodejs\esm-ut
ils.js:100:20)
    at async singleRun (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run-helpers.js:125:3)
    at async Object.exports.handler (C:\Users\lseb1\Documents\pyme 8-12\pyme\node_modules\mocha\lib\cli\run.js:370:5)


  GET /api/ordenes/salida
    √ It should GET all the salidas (43ms)


  1 passing (49ms)
```

Fig. 39: Testeo para observar reportes de salida.

3.5 *Sprint* 4. Pruebas del *backend*.

Como parte del *Sprint* 4 se establecen las siguientes tareas:

- Realización de testeos unitarios y conclusiones.
- Realización de testeos de compatibilidad y conclusiones.
- Realización de testeos de carga y conclusiones.

Realización de testeos unitarios y conclusiones

Al concluir con la etapa de implementación de cada uno de los *endpoints* se puede continuar con lo establecido en la planificación previa, por tal motivo se presenta la etapa de testeos unitarios. En ese sentido, los testeos unitarios son usados en la validación de elementos, clases y métodos dentro de un programa *software* [53]. Por ende, para llevar a cabo esta etapa se utilizan las herramientas *Mocha y Chai*, ya que facilitan la correcta ejecución de testeos unitarios asincrónicos y completos [54]. La **Fig. 40** presenta un ejemplo del *endpoint* encargado del *login* del usuario y la **Fig. 41** muestra la ejecución del testeo unitario con su respectivo resultado. Por otra parte, el desarrollo a detalle de la manera en que se ejecutan los testeos y sus respectivos resultados se detallan de mejor manera en el **ANEXO II** del documento actual.

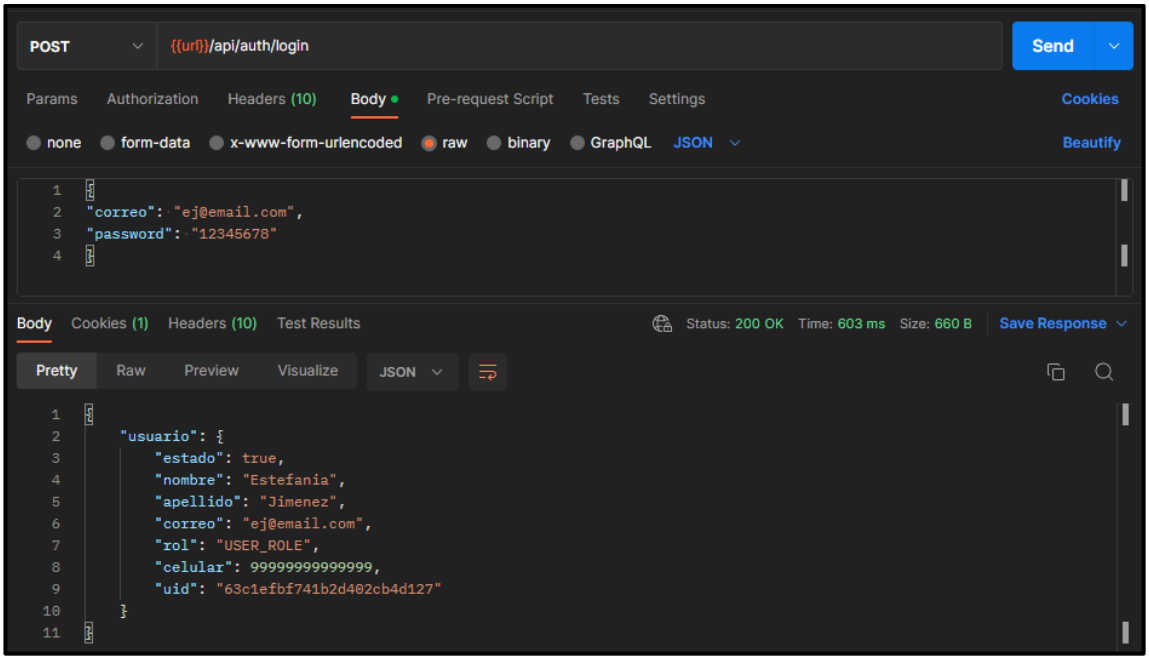


Fig. 40: Testeo del *Login*.

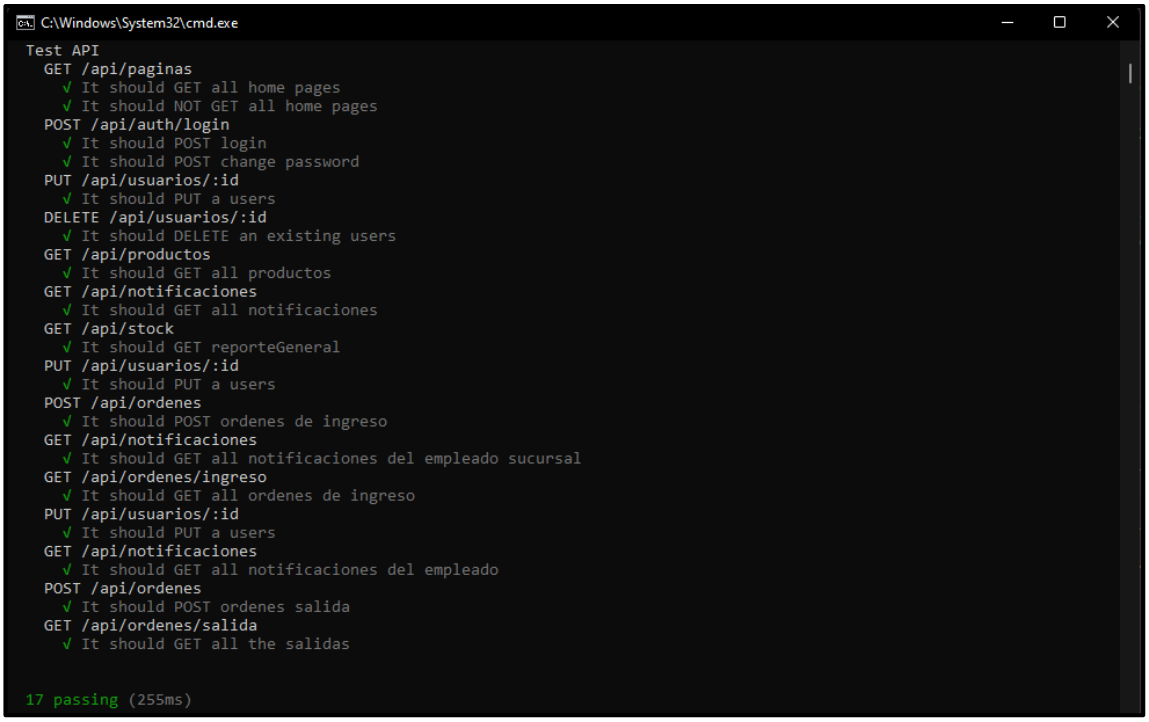


Fig. 41: Resultado del testeo en general.

Por último, tomando en cuenta los resultados de todos los testeos se determina que los *endpoints* no presenta errores ni fallas al momento de su ejecución o validación respectiva, posibilitando de esta manera el consumo e implementación en aplicaciones móviles o aplicaciones del lado del cliente si el caso lo requiere.

Realización de testeos de compatibilidad y conclusiones

La intención del testeo de compatibilidad consiste en verificar la correcta presentación de datos en una variedad de dispositivos o clientes *HTTP* [55]. Por lo tanto, se realizan testeos de compatibilidad a través de clientes HTTP como lo son: *Postman* y *HTTP Request client*, los cuales permiten comprobar el funcionamiento adecuado de cada *endpoint* como se muestra en la **TABLA VIII** la cual muestra la versión que se han utilizado. Por otra parte, el desarrollo a detalle de la manera en que se ejecutan los testeos y su resultado se encuentra en el **ANEXO II** del documento actual.

TABLA VIII: Testeo de compatibilidad.

NOMBRE	VERSIÓN
Postman	V10.5.7
Thunder Client	V2.0.2

Al finalizar la etapa de testeos de compatibilidad se determina que el desarrollo del *backend* no presenta errores en los tiempos de respuesta a una petición o errores al momento de presentar la información solicitada por parte del cliente.

Realización del testeo de carga y conclusiones

El testeo de carga consiste en un sistema global de inyectores de carga, lo que le permite medir rápidamente el rendimiento de páginas web, aplicaciones y *APIs*, se consideran muy relevantes para garantizar el rendimiento de las aplicaciones, dichos testeos se aseguran de que cada parte sea sólida y se ejecutan previamente a la fase de paso a producción del proyecto, lo cual posibilita formar una aplicación preparada para las demandas de los usuarios [56]. En la **Fig. 42**, se muestra la forma en la que se configura la prueba de carga dentro del entorno de pruebas de Apache *JMeter* y posterior a ello se encuentra la evidencia de la prueba aprobada exitosamente. Por último, se inicia un análisis minucioso acerca de la forma en la que se ejecutan los testeos de rendimiento con sus respectivos resultados, los cuales se detallan en el **ANEXO II** del escrito.

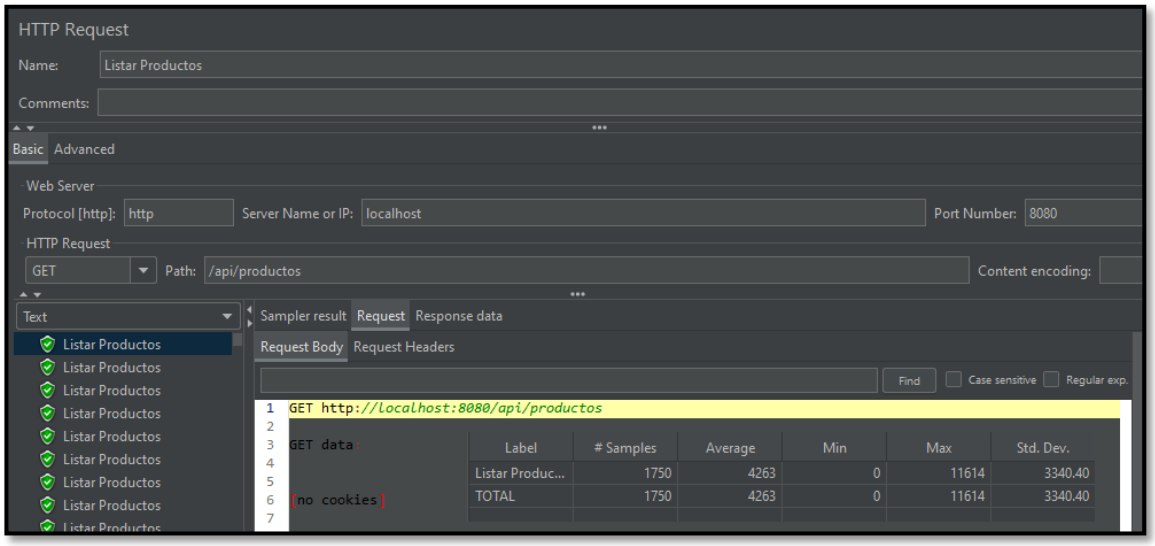


Fig. 42: Testeo de carga con Apache *JMeter* – Historia de usuario N°5.

Al finalizar la etapa de testeos de carga se determina que el desarrollo del *backend* cumple con un tiempo de respuesta mínimo para cada una de las peticiones.

3.6 *Sprint 5.* Despliegue del *backend*

Tomando en cuenta las directrices del *Sprint Backlog* la siguiente etapa es:

Despliegue del *backend* en *Railway*

Al concluir con todas las fases delimitadas en el proyecto se procede a la fase de paso a producción mediante *Railway*, como se presenta en la **Fig. 43**, siendo así el acceso al *backend* se lo puede realizar accediendo a la siguiente *url* desde un navegador:

https://api-pyme-backend-production.up.railway.app

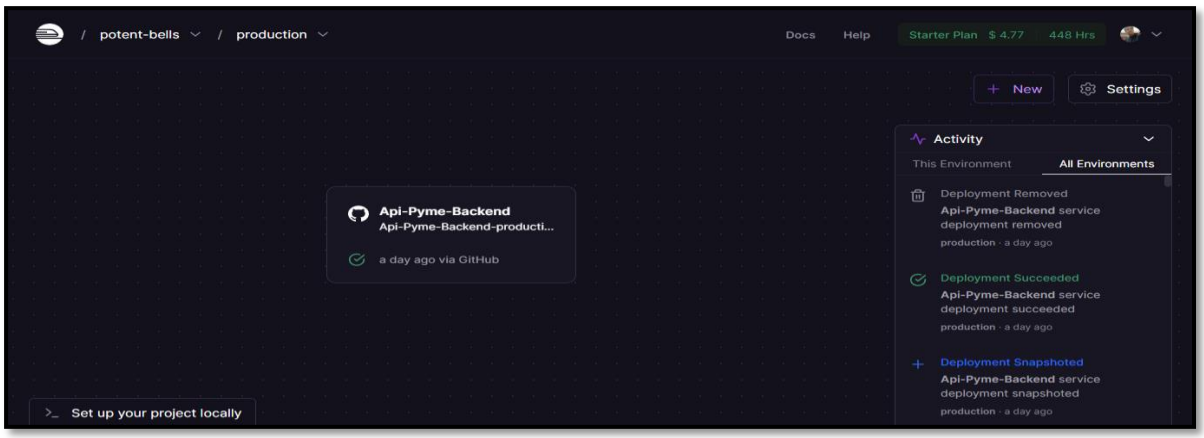


Fig. 43: Proyecto en producción en la plataforma *Railway.*

4 CONCLUSIONES

En este apartado se exhibe los resultados que se han logrado en el transcurso del presente Trabajo de Integración Curricular.

- Los requerimientos que se han planteado al inicio del proyecto han permitido encaminar a una correcta ejecución del *backend* en conjunto a la estructura de una *API* de tipo *RESTful* resultando un proyecto completamente estructurado y escalable, dando la apertura al incremento de módulos dinámicos y compatibles posteriormente.

- El uso de *Scrum* ha posibilitado que el proyecto se encamine de la mejor manera y que se logre un producto de calidad en los tiempos establecidos.

- El proyecto utiliza una arquitectura altamente eficiente como lo es el Modelo-Vista-Controlador, el cual posibilita un desarrollo del *backend* organizado, una fácil compresión de todas las funcionalidades y una estructura escalable para cualquier desarrollador que se integre posteriormente.

- Al utilizar un base de datos no relacional y un *SGB* como lo es *MongoDB Atlas* han producido un efecto positivo en la organización de los datos debido a que favorecen la gestión y adaptabilidad. Además, han permitido que la presentación de datos sea mucho más rápida al no existir relaciones como en modelos de datos tradicionales (SQL), permitiendo acceder a los datos en el instante que son solicitados.

- La etapa de testeos que se han elaborado en el *backend* presenta resultados favorables de tal manera que los testeos unitarios, de compatibilidad y de carga permiten asegurar el buen funcionamiento de cada uno de los *endpoints* y métodos que se han implementado, logrando de esta manera que el *backend* sea consumido fácilmente por aplicativos móviles o web.

- El presente *backend* se encuentra desplegado satisfactoriamente, lo que permite su respectivo uso por parte de una aplicación del lado del cliente, móvil u operador externo que esté interesado en consumir las rutas y *endpoints* que se han generado en el proyecto.

5 RECOMENDACIONES

Al finalizar el Trabajo de Integración Curricular se recomienda.

- Hay que confirmar que la versión de *Node*.js se encuentre actualizada debido a que presenta errores al momento de realizar el despliegue en cualquier plataforma como servicio (SaaS).
- Generar *backups* de la base de datos en función de la carga de trabajo que se maneje en la PYME, con el propósito de salvaguardar datos sensibles.
- Si hay un aumento de nuevas funcionalidades se debe revisar la documentación previa ya que existen métodos generales los cuales permiten su reutilización de tal manera que se agilite el proceso de codificación.
- Al existir un aumento de nuevos requerimientos se debe conservar el formato de artefactos que se ha implementado en el presente proyecto, de tal manera que el proceso de planificación, codificación y testeo se realice adecuadamente para todo el equipo de trabajo.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] A. Benvenuto, «dialnet.unirioja.es,» 01 01 2006. [En línea]. Available: https://dialnet.unirioja.es/descarga/articulo/2573348.pdf. [Último acceso: 12 05 2022].

[2] M. A. A. V. P. C. A. Vasconez Víctor H. Mayorca Myriam J.MORENO, «revistaespacios.com,» 02 07 2020. [En línea]. Available: http://www.revistaespacios.com/a20v41n03/a20v41n03p07.pdf. [Último acceso: 10 04 2022].

[3] W. O. Almeida Palacios, «repositorio.uasb.edu.ec,» 01 01 2017. [En línea]. Available: http://hdl.handle.net/10644/5566. [Último acceso: 02 02 2022].

[4] ©. M. 2022, «dynamics.microsoft.com,» 01 01 2021. [En línea]. Available: https://dynamics.microsoft.com/es-es/field-service/inventory-management-system/. [Último acceso: 02 02 2022].

[5] P. D. S. P. T. D. J. I. R. G. A. Ortega Marqués Ana, «revistas.unisimon.edu.co,» 04 05 2017. [En línea]. Available: https://revistas.unisimon.edu.co/index.php/liderazgo/article/view/3261. [Último acceso: 2022 01 20].

[6] Ekos, «ekosnegocios.com,» 02 01 2020. [En línea]. Available: https://www.ekosnegocios.com/articulo/erp-la-plataforma-del-futuro. [Último acceso: 29 01 2022].

[7] É. S. |. Q. P. L. |. T. C. M. Fernández, «sedici,» 01 10 2017. [En línea]. Available: http://sedici.unlp.edu.ar/handle/10915/63855. [Último acceso: 28 01 2022].

[8] R. Cenete, «tendencias,» 05 01 2019. [En línea]. Available: https://www.tendencias.kpmg.es/2019/09/cadena-suministro-digital/. [Último acceso: 28 01 2022].

[9] «https://www.researchgate.net/,» 30 01 2020. [En línea]. Available: https://www.researchgate.net/profile/Saleh-Hadidi/publication/339897797_COMPARISON_BETWEEN_CLOUD_ERP_AND_TRADITIONAL_ERP/links/5e6adc12a6fdccf321d9251d/COMPARISON-BETWEEN-CLOUD-ERP-AND-TRADITIONAL-ERP.pdf. [Último acceso: 02 02 2022].

[10] J. G. G. JOSE GUILLERMO VALLE, «ecotec.edu,» 01 01 2005. [En línea]. Available: https://www.ecotec.edu.ec/documentacion/investigaciones/docentes_y_directivos/articulos/5743_TRECALDE_00212.pdf. [Último acceso: 07 03 2022].

[11] B. G. Loarte, «Desarrollo de una aplicación web y móvil en tiempo real, una evolución de las aplicaciones actuales,» 2019. [En línea]. Available: https://cienciadigital.org/revistacienciadigital2/index.php/CienciaDigital/article/view/282/680. [Último acceso: 2023].

[12] A. V, «ebooksworld.ir,» 02 02 2016. [En línea]. Available: https://dl.ebooksworld.ir/motoman/Packt.Mastering.JavaScript.www.EBooksWorld.ir.pdf. [Último acceso: 03 02 2022].

[13] S. A. &. M. M, «search.proquest.com,» 2014. [En línea]. Available: https://search.proquest.com/openview/b2179529867e943aee2ed485dda9af5f/1?pq-origsite=gscholar&cbl=1986354. [Último acceso: 15 05 2022].

[14] O. Foundation, «nodejs.org,» 2022. [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: 05 02 2022].

[15] V. R. Anshu Soni, «https://www.researchgate.net/,» 01 07 2019. [En línea]. Available: https://www.researchgate.net/profile/Virender-Ranga/publication/335419384_API_Features_Individualizing_of_Web_Services_REST_and_SOAP/links/5d64960ea6fdccc32cd31171/API-Features-Individualizing-of-Web-Services-REST-and-SOAP.pdf. [Último acceso: 19 05 2022].

[16] B. G. Loarte, «Desarrollo de un backend para la gestión del sistema penitenciario del Ecuador,» 2022. [En línea]. Available: https://cienciadigital.org/revistacienciadigital2/index.php/ConcienciaDigital/article/view/2319/5614. [Último acceso: 2023].

[17] A. T. K. S. Makris, «link.springer.com,» 25 09 2020. [En línea]. Available: https://link.springer.com/article/10.1007/s10707-020-00407-w. [Último acceso: 19 05 2022].

[18] A. Zorita, «bytecode.es,» 14 10 2016. [En línea]. Available: https://bytecode.es/que-es-una-libreria-informatica/. [Último acceso: 12 02 2022].

[19] L. P. &. P. M. C. Canós J. H., «academia.edu,» Universidad Politécnica de Valencia, 01 01 2003. [En línea]. Available: https://www.academia.edu/download/34546906/XP_Agil.pdf. [Último acceso: 07 02 2022].

[20] A. d. Young, «support.zendesk.com,» 25 10 2021. [En línea]. Available: https://support.zendesk.com/hc/es/articles/4408881965722-Anatom%C3%ADa-de-una-solicitud-JWT. [Último acceso: 20 05 2022].

[21] V. H. A., «researchgate.net,» 01 01 2010. [En línea]. Available: https://www.researchgate.net/profile/Hector-Valdecantos/publication/308314622_Principios_y_patrones_de_diseno_de_software_en_torno_al_patron_compuesto_Modelo_Vista_Controlador_para_una_arquitectura_de_aplicaciones_interactivas/links/57e0683c08aec6ce9f28e7. [Último acceso: 07 02 2022].

[22] L. D. Arango Amaya, «bibliotecadigital.udea.edu.co,» 01 01 2021. [En línea]. Available: http://hdl.handle.net/10495/25340. [Último acceso: 18 02 2022].

[23] L. G. &. H. S., «fit.um.edu.mx,» 01 01 2008. [En línea]. Available: https://fit.um.edu.mx/CI3/publicaciones/Technical%20Report%20COMP-018-2008.pdf. [Último acceso: 29 02 2022].

24] V. E. D. R. &. V. E. E. G., «cybertesis.unmsm.edu.pe,» 01 01 2005. [En línea]. Available: https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/272/Valdivia_ee.pdf?sequence=1. [Último acceso: 09 03 2022].

[25] O. F. D., «ria.utn.edu.ar,» 02 02 2021. [En línea]. Available: https://ria.utn.edu.ar/bitstream/handle/20.500.12272/5913/Tesis%20Felipe%20Ortiz.pdf?sequence=1&isAllowed=y. [Último acceso: 26 02 2022].

[26] heroku, «www.heroku.com,» 2022. [En línea]. Available: https://www.heroku.com/platform. [Último acceso: 04 06 2022].

[27] E. Yacuzzi, «econstor.eu,» Universidad del Centro de Estudios Macroeconómicos de Argentina (UCEMA), Buenos Aires, 2005. [En línea]. Available: https://www.econstor.eu/handle/10419/84390. [Último acceso: 2022].

[28] J. Z. Gamboa, «dialnet.unirioja.es,» 02 01 2018. [En línea]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=6777227. [Último acceso: 20 03 2022].

[29] J. P. Esteban Gabriel Maida, «repositorio.uca.edu.ar,» 2015. [En línea]. Available: https://repositorio.uca.edu.ar/handle/123456789/522. [Último acceso: 10 04 2022].

[30] M. T. Gallego, «openaccess.uoc.edu,» 2012. [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf. [Último acceso: 19 05 2022].

[31] S. S., «www.academia.edu,» 02 09 2019. [En línea]. Available: https://www.academia.edu/download/46357609/11_ijecs.pdf. [Último acceso: 28 05 2022].

[32] G. J. Rodriguez, «northware.mx,» 15 01 2012. [En línea]. Available: https://www.northware.mx/blog/tecnicas-efectivas-para-la-toma-de-requerimientos/. [Último acceso: 19 05 2022].

[33] M. S. Kirsi Mikkonen, «aaltodoc.aalto.fi,» 2010. [En línea]. Available: https://aaltodoc.aalto.fi/bitstream/handle/123456789/3248/urn100230.pdf?sequence=1&isAllowed=y. [Último acceso: 10 06 2022].

[34] P. Á. Romero, «dialnet.,» 2006. [En línea]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=4786655. [Último acceso: 19 05 2022].

[35] E. B. Pantoja, «scielo.org.,» 2004. [En línea]. Available: http://www.scielo.org.bo/scielo.php?pid=S1683-07892004000100005&script=sci_arttext. [Último acceso: 16 05 2022].

[36] G. G. S. V. M. G. P. S. V. D. &. G. V. E. Martín A, «sedici.unlp.edu.ar,» 06 2013. [En línea]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/27190/T%C3%A9cnicas+y+Herramientas+para+Desarrollo+de+Sitios+Web.pdf?sequence=1. [Último acceso: 29 05 2022].

[37] L. PUCIARELLI, «Instalación-Arquitectura-node y npm,» 2020.. [En línea]. Available: https://books.google.com.ec/books?hl=es&lr=&id=GOfqDwAAQBAJ&oi=fnd&pg=PP1&dq=node.js+que+es&ots=uJmZT5PIbQ&sig=-C8g0gqGGa-8KMGmWMxq3q3fODg&redir_esc=y#v=onepage&q=node.js%20que%20es&f=false. [Último acceso: 23 10 2022].

[38] A. MARDAN, «Express. js Guide: The Comprehensive Book on Express. js.,» 2014. [En línea]. Available: https://books.google.com.ec/books?hl=es&lr=&id=5eGRAwAAQBAJ&oi=fnd&pg=PP6&dq=que+es+express.js%3F&ots=nksew3bqNG&sig=_IhWDd-Lpk9V5iyBL5zL8MZjkuI&redir_esc=y#v=onepage&q=que%20es%20express.js%3F&f=false. [Último acceso: 10 12 2022].

[39] mongodb, «mongodb,» 07 07 2022. [En línea]. Available: https://www.mongodb.com/docs/atlas/.

[40] filehorse, «filehorse.com,» 2023 01 12. [En línea]. Available: https://www.filehorse.com/es/descargar-mongodb-compass/. [Último acceso: 01 02 2023].

[41] Railway, «railway.app,» 01 01 2022. [En línea]. Available: https://docs.railway.app/. [Último acceso: 01 01 2023].

[42] F. Flores, «Openwebinars.net,» 22 06 2022. [En línea]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/. [Último acceso: 20 01 2023].

[43] Nisfe, «Nisfe,» 14 02 2014. [En línea]. Available: https://www.nisfe.com/almacenamiento/cloudinary-almacenamiento-de-imagenes-en-la-nube-para-utilizarlas-en-una-web/. [Último acceso: 14 11 2022].

[44] J. Munro, «code.tutsplus,» 09 09 2017. [En línea]. Available: https://code.tutsplus.com/es/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527. [Último acceso: 20 11 2022].

[45] npmjs, «npmjs,» 06 08 2022. [En línea]. Available: https://www.npmjs.com/package/bcrypt. [Último acceso: 11 11 2023].

[46] npmjs., «npmjs.com,» 21 12 2021. [En línea]. Available: https://www.npmjs.com/package/jsonwebtoken. [Último acceso: 11 12 2022].

[47] V. V. Rico, «victorvr,» 20 06 06. [En línea]. Available: https://www.victorvr.com/tutorial/variables-de-entorno-con-nodejs. [Último acceso: 05 05 2022].

[48] vortexbird, «vortexbird,» 19 07 2017. [En línea]. Available: https://vortexbird.com/nodemon/. [Último acceso: 05 06 2022].

[49] Mozilla, «developer.mozilla.org,» 2022. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

[50] npmjs, «uuid,» 05 08 2022. [En línea]. Available: https://www.npmjs.com/package/uuid. [Último acceso: 01 01 2023].

[51] npmjs, «npmjs,» 2022. [En línea]. Available: https://www.npmjs.com/package/express-fileupload. [Último acceso: 2023].

[52] npmjs, «node-fetch,» 02 11 2022. [En línea]. Available: https://www.npmjs.com/package/node-fetch. [Último acceso: 01 02 2023].

[53] N. M. P. &. M. P. P. Viteri Arias S. Mayorga Soria T, «cienciadigital.org,» 2008. [En línea]. Available: https://cienciadigital.org/revistacienciadigital2/index.php/CienciaDigital/article/view/140/385. [Último acceso: 20 07 2022].

[54] mochajs, «mochajs.org,» 2022. [En línea]. Available: https://mochajs.org/#installation. [Último acceso: 07 07 2022].

[55] M. A. M. C. L. G. E. Leandro N. Sabaren, «sedici.unlp.edu.ar,» 13 10 2017. [En línea]. Available: http://sedici.unlp.edu.ar/bitstream/handle/10915/63811/Documento_completo.pdf?sequence=1. [Último acceso: 20 07 2022].

[56] C. técnicos, «loadview-testing,» 10 16 2020. [En línea]. Available: https://www.loadview-testing.com/es/blog/pruebas-de-carga-de-node-js-por-que-loadview-lo-hace-facil/. [Último acceso: 15 01 2021].

7 ANEXOS

A continuación, se presenta cada uno de los Anexos que se ha utilizado para el desarrollo del *backend*, los cuales se encuentran detallados de la siguiente manera:

- **ANEXO I.** Resultado del programa antiplagio Turnitin.
- **ANEXO II.** Manual Técnico.
- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Credenciales de acceso y despliegue.

ANEXO I

A continuación, se presenta el certificado que el Director de Tesis ha emitido y en donde se evidencia el resultado que se ha obtenido en la herramienta antiplagio Turnitin.

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

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 11 de febrero de 2023

De mi consideración:

Yo, Loarte Cajamarca Byron Gustavo, en calidad de Director del Trabajo de Integración Curricular titulado Desarrollo de un backend asociado al DESARROLLO DE SISTEMA WEB PARA LA GESTIÓN DE INVENTARIOS EN PYMES elaborado por el estudiante Luis Sebastian Catota Caizaluisa de la carrera en Tecnología Superior en Desarrollo de Software, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones, producto del Trabajo de Integración Curricular indicado.

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

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

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

Atentamente,

Loarte Cajamarca Byron Gustavo
Profesor Ocasional a Tiempo Completo
Escuela de Formación de Tecnólogos

ANEXO II

Recopilación de Requerimientos

En la **Tabla IX** se muestra los requerimientos que han sido recopilados al inicio del proyecto en donde se evidencia lo solicitado por el *Product Owner*.

Tabla IX: Recopilación de requerimientos

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
Backend	**RR001**	Como usuario administrador necesita crear un *endpoint* para visualizar una página informativa.
	RR002	Como usuario administrador, empleado y empleado sucursal necesita generar varios *endpoints* para: • Iniciar sesión. • Cerrar sesión. • Cambiar de contraseña.
	RR003	Como usuario administrador necesita generar *endpoints* para: • Modificar información personal.
	RR004	Como usuario administrador necesita generar *endpoints* para: • Gestionar perfiles de usuarios.
	RR005	Como usuario administrador necesita generar *endpoints* para: • Gestionar productos. • Visualizar *stock* de productos.
	RR006	Como usuario administrador necesita generar *endpoints* para: • Visualizar notificaciones.

	RR007	Como usuario administrador necesita generar *endpoints* para: • Generar reportes del stock general productos.
	RR008	Como usuario administrador necesita generar *endpoints* para: • Generar reportes de compra de productos.
	RR009	Como usuario administrador necesita generar *endpoints* para: • Generar reportes de ingreso y salida de productos.
	RR010	Como usuario empleado sucursal necesita generar *endpoints* para: • Modificar información personal.
	RR013	Como usuario empleado necesita generar *endpoints* para: • Modificar información personal.
	RR014	Como usuario empleado necesita generar *endpoints* para: • Generar reportes de productos.

Historias de Usuario

Culminada la etapa de recopilación de requerimientos, se procede a desarrollar las Historias de Usuario, para el *backend*. A continuación, se presentan las 14 Historias de Usuario escritas en base a los requerimientos del proyecto que va desde la

Tabla X a la **Tabla XXII**.

Tabla X: Historia de usuario Nro.1: Generar *endpoints* para visualizar una página informativa.

HISTORIA DE USUARIO	
Identificador: HU001	**Usuario:** Administrador.
Nombre historia: Generar *endpoints* para visualizar una página informativa.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media

Iteración asignada: 1
Responsable (es): Luis Catota
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para ser visualizados en el *frontend*. Además, las secciones de la página informativa son: ● *Header* ● *Main* ● *Footer* ● Redes Sociales
Observación: La página informativa presenta información de la PYME como un *endpoint* público.

Tabla XI: Historia de usuario Nro.3: Generar varios *endpoints* para modificar información personal.

HISTORIA DE USUARIO	
Identificador: HU003	**Usuario:** Administrador.
Nombre historia: Generar *varios endpoints* para modificar información personal.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para modificar la información personal a través de un formulario con los siguientes campos: ● Nombre. ● Apellido. ● Número telefónico. ● Correo. ● Imagen. ● Rol.	

• Edad. • Genero.
Observación: El usuario administrador tiene la capacidad de modificar su información personal en elmomento que lo requiera.

Tabla XII: Historia de usuario Nro.4: Generar varios *endpoints* para la gestión de perfiles de usuarios.

HISTORIA DE USUARIO	
Identificador: HU004	**Usuario:** Administrador.
Nombre historia: Generar un *endpoint* para la gestión de perfiles de usuarios.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para la gestión de perfiles de usuario, el cual permite: • Crear perfiles. • Listar perfiles. • Modificar perfiles. • Habilitar o deshabilitar perfiles. Además, se dividen en tres perfiles distintos: • Perfil administrador. • Perfil empleado. • Perfil empleado sucursal.	
Observación: El usuario administrador no puede eliminar a un usuario si este se encuentra habilitado.	

Tabla XIII: Historia de usuario Nro.5: Consumir un *endpoint* para la gestión y *stock* de productos.

HISTORIA DE USUARIO	
Identificador: HU005	**Usuario:** Administrador y empleado sucursal
Nombre historia: Consumir un *endpoint* para la gestión y *stock* de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 2	
Responsable (es): Luis Catota	
Descripción: Los usuarios con el perfil designado en el *backend* tienen la posibilidad de generar varios *endpoints* para la gestionar productos con acciones como: • Listar, ingresar, modificar y eliminar los productos ingresados. • Listar, ingresar, modificar y eliminar los productos almacenados. • Listar, ingresar, modificar y eliminar los productos agotados. El proceso mencionado anteriormente se realiza mediante un formulario con los siguientes campos: • Nombre. • Tipo de producto. • Cantidad. • Categoría. • Código. • Descripción.	
Observación: Solo el usuario con perfil administrador puede realizar la creación de productos y el usuario empleado sucursal establece la cantidad del *stock* de productos.	

Tabla XIV: Historia de usuario Nro.6: Generar varios *endpoints* para notificación de productos.

HISTORIA DE USUARIO	
Identificador: HU006	**Usuario:** Administrador.

Nombre historia: Generar varios *endpoints* para notificación de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para enviar y recibir notificaciones acerca de: • Gestión de productos en stock. • Alerta de fecha de vencimiento de productos. • Errores en la gestión de productos.	
Observación: Dentro de la notificación se evidencia el mensaje y para quien la genero.	

Tabla XV: Historia de usuario Nro.7: Generar varios *endpoints* para crear reportes de productos.

HISTORIA DE USUARIO	
Identificador: HU007	**Usuario:** Administrador.
Nombre historia: Generar varios *endpoints* para generar reportes de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable (es): Luis Catota	

Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para crear reportes de *stock*, semanales, quincenales o mensuales. La estructura de la información del reporte está dada por: • Código del producto. • Categoría. • Nombre del producto. • Descripción. • *Stock* de productos o cantidad. • Precio. • Imagen si la posee. • Código del usuario. • Fecha y hora de creación.
Observación: El usuario con perfil administrador es el único que puede generar reportes de productos.

Tabla XVI: Historia de usuario Nro.8: Generar varios *endpoints* para crear reportes de compra.

HISTORIA DE USUARIO	
Identificador: HU008	**Usuario:** Administrador.
Nombre historia: Generar varios *endpoints* para crear reportes de compra.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable (es): Luis Catota	

Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para crear reportes de compra semanales, quincenales o mensuales. La estructura de la información del reporte está dada por: • Código del producto. • Categoría. • Nombre del producto. • Descripción. • *Stock* de productos o cantidad igual a cero. • Precio. • Imagen si la posee. • Código del usuario. • Fecha y hora de creación.
Observación: El usuario con perfil administrador es el único que puede generar reportes de compras.

Tabla XVII: Historia de usuario Nro.9: Generar varios *endpoints* para crear reportes de salida.

HISTORIA DE USUARIO	
Identificador: HU009	**Usuario:** Administrador, empleado sucursal y empleado.
Nombre historia: Generar varios *endpoints* para crear reportes de salida.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable (es): Luis Catota	
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para crear reportes de salida semanales, quincenales o mensuales. La estructura de la información del reporte está dada por: • Código del producto. • Nombre de producto. • Cantidad en stock inicial del producto. • Cantidad en stock final del producto. • Descripción.	

• Código del usuario • Rectificación de la acción que se realizó (ingreso o salida). • Sucursal hacia donde se dirige el producto.
Observación: Solo los usuarios con el perfil permitido pueden generar reportes de compras.

Tabla XVIII: Historia de usuario Nro.10: Generar varios *endpoints* para crear reportes de salida.

HISTORIA DE USUARIO	
Identificador: HU010	**Usuario:** Empleado
Nombre historia: Generar varios *endpoints* para modificar información personal.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario empleado en el *backend* tiene la posibilidad de generar varios *endpoints* paramodificar la información personal a través de un formulario con los siguientes campos: • Nombre. • Dirección. • Imagen. • Fecha de nacimiento. • Edad. • Genero.	
Observación: El usuario tiene la capacidad de modificar su información personal en el momento que lo requiera.	

Tabla XIX: Historia de usuario Nro.11: Generar un *endpoint* para la gestión y *stock* de productos.

HISTORIA DE USUARIO	
Identificador: HU011	**Usuario:** Empleado y empleado sucursal
Nombre historia: Generar un *endpoint* para la gestión y *stock* de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable (es): Luis Catota	
Descripción: El usuario administrador en el *backend* tiene la posibilidad de generar varios *endpoints* para la gestión de: • Listar, ingresar, modificar los productos ingresados. • Listar, ingresar, modificar los productos almacenados. • Listar, ingresar, modificar los productos agotados. El proceso mencionado anteriormente se realiza mediante un formulario con los siguientes campos: • Nombre. • Tipo de producto. • Cantidad. • Categoría. • Tipo de bodega. • Código. • Descripción. • Fecha y hora de ingreso.	
Observación: El usuario con perfil empleado no puede eliminar los productos.	

Tabla XX: Historia de usuario Nro.12: Generar varios *endpoints* para notificación de productos.

HISTORIA DE USUARIO	
Identificador: HU012	**Usuario:** Empleado sucursal
Nombre historia: Generar varios *endpoints* para notificación de productos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario empleado y empleado sucursal en el *backend* tienen la posibilidad de generarvarios *endpoints* para enviar y recibir notificaciones acerca de: • Gestión de productos en *stock*. • Alerta de fecha de vencimiento de productos. • Errores en la gestión de productos.	
Observación: Solo los usuarios con perfil empleado y empleado sucursal pueden enviar y recibir notificaciones.	

Tabla XXI: Historia de usuario Nro.13: Generar varios *endpoints* para modificar información personal.

HISTORIA DE USUARIO	
Identificador: HU013	**Usuario:** Administrador.
Nombre historia: Generar varios *endpoints* para modificar información personal.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 1	
Responsable (es): Luis Catota	
Descripción: El usuario empleado sucursal en el *backend* tiene la posibilidad de generar varios *endpoints* para modificar la información personal a través de un formulario con los siguientes campos: • Nombre. • Dirección.	

• Imagen. • Fecha de nacimiento. • Edad. • Genero.
Observación: El usuario tiene la capacidad de modificar su información personal en el momento que lo requiera.

Tabla XXII: Historia de usuario Nro.14: Generar varios *endpoints* para generar reportes.

HISTORIA DE USUARIO	
Identificador: HU014	**Usuario:** Empleado
Nombre historia: Generar varios *endpoints* para crear reportes.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Iteración asignada: 2	
Responsable (es): Luis Catota	
Descripción: El usuario empleado sucursal en el *backend* tiene la posibilidad de generar varios *endpoints* para crear reportes de productos faltantes. La estructura de la información del reporte está dada por: • Código del producto. • Categoría. • Nombre del producto. • Descripción. • *Stock* de productos igual a cero • Precio. • Imagen si la posee. • Código del usuario. • Fecha y hora de creación.	
Observación: El usuario con perfil empleado sucursal es el único que puede generar reportes de productos faltantes.	

Product Backlog

La

Tabla XXIII enumera la prioridad de cada requisito que se ha implementado en el *backend*. Estos requisitos se clasifican de acuerdo con las necesidades del dueño del producto y la complejidad del desarrollo.

Tabla XXIII: *Product Backlog.*

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID – HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU001	Generar *endpoints* para visualizar una página informativa.	1	Finalizado	Media
HU002	Generar varios *endpoints* para iniciar sesión, cerrar sesión y cambiar la contraseña.	1	Finalizado	Alta
HU003	Generar varios *endpoints* para modificar información personal.	1	Finalizado	Media
HU004	Generar un *endpoint* para la gestión de perfiles de usuarios.	1	Finalizado	Alta
HU005	Generar varios *endpoint* para la gestión y *stock* de productos	2	Finalizado	Alta
HU006	Generar varios *endpoints* para notificación de productos.	1	Finalizado	Media
HU007	Generar varios *endpoints* para generar reportes de productos.	2	Finalizado	Alta
HU008	Generar varios *endpoints* para crear reportes de compra.	2	Finalizado	Alta
HU011	Generar varios *endpoints* para la gestióny *stock* de productos.	2	Terminado	Alta
HU012	Generar varios *endpoints* para notificación de productos.	1	Terminado	Alta

HU013	Generar varios *endpoints* para modificar información personal.	1	Terminado	Media
HU014	Generar varios *endpoints* para crear reportes de productos faltantes.	2	Terminado	Alta

Sprint Backlog

La **Tabla XXIV** presenta los seis *Sprints* en los que se ha desarrollo el *backend*, listando las actividades y el tiempo determinado para cumplir con los entregables que se han establecido en la recopilación de requerimientos.

Tabla XXIV: *Sprint Backlog.*

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID – SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Diseño e implementación del *endpoint* para los usuarios con perfil administrador, empleado y empleado sucursal	Sección informativa	HU001	Generar *endpoints* para visualizar una página informativa.	• Implementación de *endpoints* de tipo público que presentan información relevante del a empresa, como: Publicidad y contactos.	30H
		Módulo acceso	HU002	Generar varios *endpoints* para iniciar sesión, cerrarsesión y cambiar la contraseña.	• Diseño e implementación del *endpoint* para el inicio de sesión. • Diseño e implementación del *endpoint* para el cierre de sesión. • Diseño e implementación del *endpoint* para el cambio de contraseña. • Verificación del consumo por parte del cliente REST. • Consulta a la base de datos y	

					autorización. • Cargar los módulos asignados. • Testeos.	
SB001	Diseño e implementación de los *endpoints* para el usuario con perfil administrador	Módulo - perfil	HU003	Generar varios *endpoints* para modificar información personal	• Diseño e implementación de *endpoints* para registrar, visualizar y eliminar su información personal. • Consulta en la Base de datos. • Validación de los datos. • Testeos.	90H
		Módulo - productos	HU005	Generar varios *endpoints* para la gestión y *stock* de productos.	• Diseño e implementación de *endpoints* para la gestión de *endpoint* para la gestión y *stock* de productos. • Consulta en la Base de datos. • Validación de los datos requeridos. • Verificación que el código del producto sea único. • Testeos.	

		Módulo - notificaciones	HU006	Generar varios *endpoints* para notificación de productos.	• Diseño e implementación de *endpoints* para notificación de productos. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	
		Módulo – reportes de productos	HU007	Generar varios *endpoints* para generar reportes deproductos.	• Diseño e implementación de *endpoints* para generar reportes de productos. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	
		Módulo - reportes de compras	HU008	Generar varios *endpoints* para crear reportes de compra.	• Diseño e implementación de *endpoints* para generar reportes de compra. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	

		Módulo - reporte de salidas	HU009	Generar varios *endpoints* para crear reportes de salida.	• Diseño e implementación de *endpoints* para generar reportes de salida. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	
SB002	Diseño e implementación de los *endpoints* para el usuario con perfil empleado sucursal	Módulo perfil	HU0010	Generar varios *endpoints* para modificar información personal.	• Diseño e implementación de *endpoints* para modificar, visualizar y eliminar su información personal. • Consulta en la Base de datos. • Validación de los datos. • Testeos.	40H
		Módulo - productos	HU0011	Generar varios *endpoints* para la gestión y *stock* de productos.	• Diseño e implementación de *endpoints* para la gestión y *stock* de productos. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	

		Módulo - notificaciones	HU0012	Generar varios *endpoints* para notificación de productos.	• Diseño e implementación de *endpoints* para notificación de productos. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	
SB003	Diseño e implementación de los *endpoints* parael usuario con perfil empleado	Módulo perfil	HU0013	Generar varios *endpoints* para modificar información personal	• Diseño e implementación de *endpoints* para modificar, visualizar y eliminar su información personal. • Consulta en la Base de datos. • Validación de los datos. • Testeos.	40H
		Módulo reporte	HU0014	Generar varios *endpoints* para generar reportes deproductos faltantes.	• Diseño e implementación de *endpoints para* generar reportes de productos faltantes. • Consulta en la Base de datos. • Validación de los datos requeridos. • Testeos.	
SB004	Pruebas en el *backend*	• Testeos unitarios. • Testeos de compatibilidad. • Testeos de carga.				20H

SB005	Despliegue del *backend* y la Base de datos No SQL	• Despliegue del *backend* en Railway. • Despliegue de la base de datos NoSQL en MongoDB Atlas.	10H
Documentación		• Integración Curricular. • Anexos.	50H
TOTAL			**240 H**

Diseño de la Base de datos NoSQL

La **Fig. 44**, presenta las nueve colecciones principales que han sido necesarias en el desarrollo de cada uno de los *endpoints* del *backend*, manteniendo la información mejor organizada y permitiendo que las consultas sean más eficientes.

PAGINAS

```
{"titulo":"string",
"estado":"boolean",
"logo": "string",
"img": "string",
"about": "string",
"contactos":"string",
"descriptionBody": "string",
"descriptionFooter":"string",
"contactos":"string",
"redes":"string"}
```

ADMIN_ROLE

```
{"id":"string",
"estado":"boolean",
"nombre": "string",
"apellido": "string",
"correo" : "string",
"rol" : "number",
"celular":"number",
"password" : "varchart",
"genero" : "string",
"edad" : Date,
"img" : "string",}
```

USER_ROLE

```
{"id":"string",
"estado":"boolean",
"nombre": "string",
"apellido": "string",
"correo" : "string",
"rol" : "number",
"celular":"number",
"password" : "varchart",
"genero" : "string",
"edad" : Date,
"img" : "string",}
```

USER_BRANCH_ROLE

```
{"id":"string",
"estado":"boolean",
"nombre": "string",
"apellido": "string",
"correo" : "string",
"rol" : "number",
"celular":"number",
"password" : "varchart",
"genero" : "string",
"edad" : Date,
"img" : "string",}
```

PRODUCTOS

```
{"id":"string",
"estado":"boolena",
"nombre": "string",
"precio": "number",
"cantidad":"number",
"descripcion" : "string",
"fecha de creacion" :
Date,
"img" : "string",}
```

REPORTES

```
{ "id": "string",
"nombre" : "string",
"cantidad": "number",
"categoria",
"descripcion": "string",
"fecha de ingreso": Date,
"fecha de
caducidad":Date}
```

CATEGORIA

```
{"id":"string",
"estado":"boolean",
"nombre": "string"}
```

ADMIN_ROLE

```
{"id":"string",
"estado":"boolean",
"rol": "string"}
```

NOTIFICACIONES

```
{"id":"string",
"estado":"boolean",
"para":"string",
"notificacion" : "string",
"Date" : Date}
```

Fig. 44: Colecciones de la base de datos.

Pruebas

Dentro del marco de las buenas prácticas de programación se encuentra el constante testeo unitario, testeo de compatibilidad y el testeo de carga, de tal manera que dichos testeos aseguren la calidad del software en conjuntos con cada *endpoint* desarrollado.

Testeo unitario

Dentro del rango de las figuras desde la **Fig. 45** hasta la **Fig. 60** se presenta el código en el cual se establecen los testeos unitarios.

```
17      describe("GET /api/paginas", () => {
18          it("It should GET all home pages", (done) => {
19              chai.request(server.app)
20                  .get("/api/paginas")
21                  .end((err, response) => {
22                      response.should.have.status(200);
23                  done();
24                  });
25          });
```

Fig. 45: Porción de código para visualizar el home pague.

```
40      describe("POST /api/auth/login", () => {
41          it("It should POST login", (done) => {
42          const task = {
43              correo: "lc@email.com",
44              password: "12345678",
45          };
46          chai.request(server.app)
47              .post("/api/auth/login")
48              .send(task)
49              .end((err, response) => {
50                  response.should.have.status(201);
51                  response.body.should.be.a('object');
52                  response.body.should.have.property('id').eq("63c1eb42d4c0de3b0cb26ae2");
53                  response.body.should.have.property('correo').eq("lc@email.com");
54                  response.body.should.have.property('password').eq(12345678);
55              done();
56              });
57      });
```

Fig. 46: Porción de código para el *login*.

```
60          it("It should POST change password ", (done) => {
61              const task = {
62                  password: "12345678",
63              };
64              chai.request(server.app)
65                  .post("/api/auth/login")
66                  .send(task)
67                  .end((err, response) => {
68                      response.should.have.status(200);
69                      response.body.should.be.a('object');
70                      response.body.should.have.property('id').eq("63c1eb42d4c0de3b0cb26ae2");
71                      response.body.should.have.property('password').eq(12345678);
```

Fig. 47: Porción de código para cambiar de contraseña.

```
197     describe("PUT /api/usuarios/:id", () => {
198         it("It should PUT a users", (done) => {
199             const taskId = "63c702c90391aa2ae4ea238b";
200             const task = {
201                 name: "Luis",
202                 apellido: "Catota",
203                 correo: "lkj@email.com",
204             };
205             chai.request(server.app)
206                 .put("/api/tasks/" + taskId)
207                 .send(task)
208                 .end((err, response) => {
209                     response.should.have.status(200);
210                     response.body.should.have.property('id').eq("63c702c90391aa2ae4ea238b");
211                     response.body.should.have.property('name').eq("Luis");
212                     response.body.should.have.property('apellido').eq("Catota");
213                     response.body.should.have.property('correo').eq("lkj@email.com");
214                 done();
215                 });
216         });
```

Fig. 48: Porción de código para modificar información personal.

```
103     describe("DELETE /api/usuarios/:id", () => {
104         it("It should DELETE an existing users", (done) => {
105         const taskId = "63c702e90391aa2ae4ea238f";
106         chai.request(server.app)
107             .delete("/api/usuarios/:id" + taskId)
108             .end((err, response) => {
109                 response.should.have.status(202);
110             done();
111             });
112     });
113 });
```

Fig. 49: Porción de código para gestionar perfiles.

```
118     describe("GET /api/productos", () => {
119         it("It should GET all productos", (done) => {
120         chai.request(server.app)
121             .get("/api/productos")
122             .end((err, response) => {
123                 response.should.have.status(200);
124             done();
125             });
126         });
127     });
128
```

Fig. 50: Porción de código para gestionar productos.

```
131     describe("GET /api/notificaciones", () => {
132         it("It should GET all notificaciones", (done) => {
133             chai.request(server.app)
134                 .get("/api/notificaciones")
135                 .end((err, response) => {
136                     response.should.have.status(200);
137
138                 done();
139                 });
140         });
141     });
```

Fig. 51: Porción de código para observar las notificaciones.

```
144     describe("GET /api/stock", () => {
145         it("It should GET reporteGeneral", (done) => {
146             chai.request(server.app)
147                 .get("/api/notificaciones")
148                 .end((err, response) => {
149                     response.should.have.status(200);
150
151                 done();
152                 });
153         });
154 });
```

Fig. 52: Porción de código para observar un reporte general.

```
384     describe("PUT /api/usuarios/:id", () => {
385         it("It should PUT a users", (done) => {
386             const taskId = "63c1f057741b2d402cb4d12d";
387             const task = {
388                 name: "Carlos",
389                 apellido: "Mendez",
390             };
391             chai.request(server.app)
392                 .put("/api/usuarios/:id" + taskId)
393                 .send(task)
394                 .end((err, response) => {
395                     response.should.have.status(200);
396                     response.body.should.have.property('id').eq("63c1f057741b2d402cb4d12d");
397                     response.body.should.have.property('name').eq("Carlos");
398                     response.body.should.have.property('apellido').eq("Mendez");
399                 done();
400                 });
401         });
```

Fig. 53: Porción de código para modificar datos del empleado sucursal.

```
182     describe("POST /api/ordenes", () => {
183         it("It should POST ordenes de ingreso", (done) => {
184         const task = {
185             id: "63b88ef0aeeb0c6becbc81ae",
186             cantidad: 20,
187             operacion: "Ingreso",
188             sucursal: "Supermaxi"
189         };
190         chai.request(server.app)
191             .post("/api/ordenes")
192             .send(task)
193             .end((err, response) => {
194                 response.should.have.status(201);
195                 response.body.should.be.a('object');
196                 response.body.should.have.property('id').eq("163b88ef0aeeb0c6becbc81ae");
197                 response.body.should.have.property('cantidad').eq(20);
198                 response.body.should.have.property('operacion').eq("Ingreso");
199                 response.body.should.have.property('sucursal').eq("Supermaxi");
200             done();
201             });
202     });
203 });
```

Fig. 54: Porción de código para la gestión y surtido de productos.

```
// Test get notificaciones del empleado sucursal

    describe("GET /api/notificaciones", () => {
        it("It should GET all notificaciones del empleado sucursal", (done) => {
        chai.request(server.app)
            .get("/api/notificaciones")
            .end((err, response) => {
                response.should.have.status(200);

            done();
            });
    });
});
```

Fig. 55: Porción de código para recibir notificaciones.

```
218   // Test get de ordenes de ingreso
219
220       describe("GET /api/ordenes/ingreso", () => {
221           it("It should GET all ordenes de ingreso", (done) => {
222           chai.request(server.app)
223               .get("/api/notificaciones")
224               .end((err, response) => {
225                   response.should.have.status(200);
226
227               done();
228               });
229       });
230   });
```

Fig. 56: Porción de código para observar reporte.

```
236     describe("PUT /api/usuarios/:id", () => {
237         it("It should PUT a users", (done) => {
238         const taskId = "63c702e90391aa2ae4ea238f";
239         const task = {
240             name: "Esteban",
241             apellido: "Tipan",
242         };
243         chai.request(server.app)
244             .put("/api/usuarios/:id" + taskId)
245             .send(task)
246             .end((err, response) => {
247                 response.should.have.status(200);
248                 response.body.should.have.property('id').eq("63c702e90391aa2ae4ea238f");
249                 esponse.body.should.have.property('name').eq("Esteban");
250                 response.body.should.have.property('apellido').eq("Tipan");
251             done();
252             });
253     });
254 });
```

Fig. 57: Porción de código para modificar información personal del empleado.

```
256     describe("GET /api/notificaciones", () => {
257         it("It should GET all notificaciones del empleado", (done) => {
258         chai.request(server.app)
259             .get("/api/notificaciones")
260             .end((err, response) => {
261                 response.should.have.status(200);
262
263             done();
264             });
265         });
266
267     });
```

Fig. 58: Porción de código para visualizar notificaciones del empleado.

```
272     describe("POST /api/ordenes", () => {
273         it("It should POST ordenes salida", (done) => {
274             const task = {
275                 id: "63b88ef0aeeb0c6becbc81ae",
276                 cantidad: 20,
277                 operacion: "Salida",
278                 sucursal: "Aki"
279             };
280             chai.request(server.app)
281                 .post("/api/ordenes")
282                 .send(task)
283                 .end((err, response) => {
284                     response.should.have.status(201);
285                     response.body.should.be.a('object');
286                     response.body.should.have.property('id').eq("163b88ef0aeeb0c6becbc81ae");
287                     response.body.should.have.property('cantidad').eq(20);
288                     response.body.should.have.property('operacion').eq("Salida");
289                     response.body.should.have.property('operacion').eq("Aki");
290                 done();
291                 });
292         });
293     });
```

Fig. 59: Porción de código para distribución de productos.

```
294    describe("GET /api/ordenes/salida", () => {
295        it("It should GET all the salidas", (done) => {
296            chai.request(server.app)
297                .get("/api/notificaciones")
298                .end((err, response) => {
299                    response.should.have.status(200);
300
301                done();
302                });
303        });
```

Fig. 60: Porción de código para reportes de salida de productos.

Testeos de compatibilidad

Para la implementación de testeos de compatibilidad se utilizan varios clientes HTTP, en este caso se utilizan los clientes *Postman* y *Thunder Client* con los cuales se verifica la correcta funcionalidad de la API en diferentes clientes, los cuales corroboran que los resultados sean los mismos, por lo tanto se presenta en la **Fig. 61** el resultado en *Postman*, donde se evidencia el método *GET* para mostrar un producto en específico y de la misma manera en la **Fig. 62**, muestra el uso de *Thunder Client* para evaluar la misma petición , siendo este un cliente HTTP que funciona como una extensión para el editor *Visual Studio Code* el cual presenta la misma respuesta.

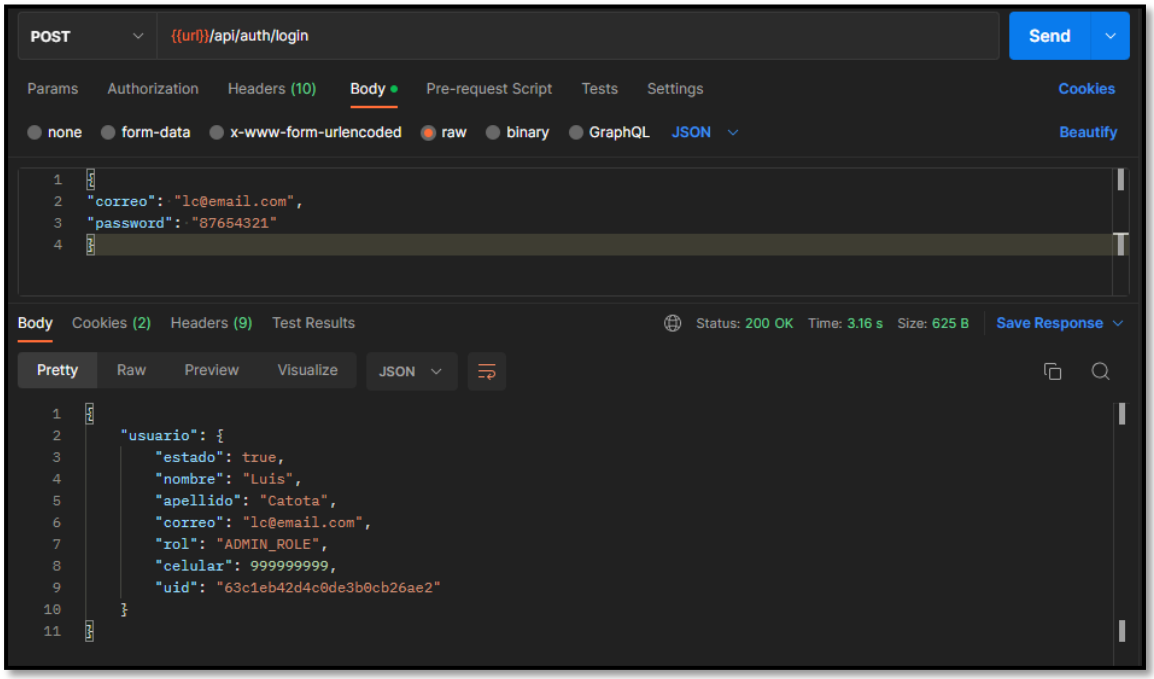


Fig. 61: Petición *POST* en *Postman.*

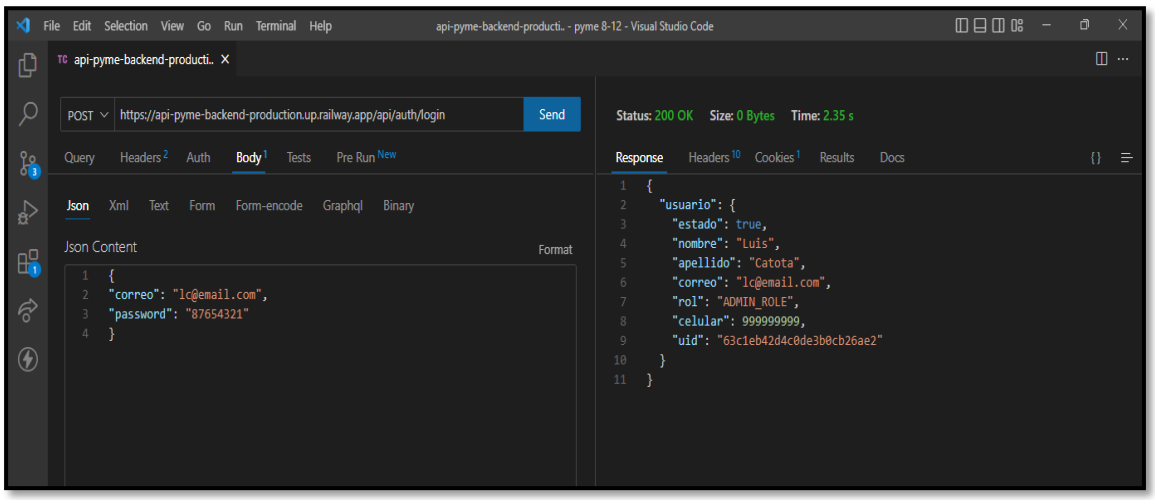


Fig. 62*:* Petición *POST* en *Thunder Client.*

Testeos de carga

En este apartado se presentan 14 testeos de carga, las cuales ejemplifican el comportamiento adecuado de los *endpoints* más notables, ante un flujo continuo de peticiones, de tal manera que se demuestre que los *endpoints* funcionen de la forma en la que se espera en un tiempo determinado. Por tal motivo, desde la **Fig. 63** hasta la **Fig. 75** se evidencian los testeos de carga realizados con sus respectivos resultados.

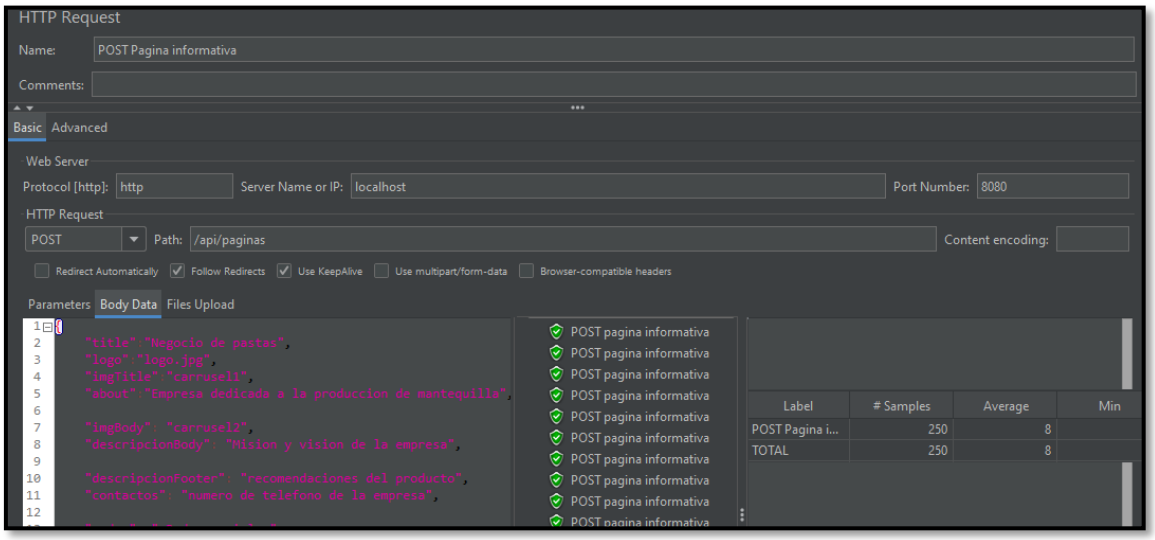


Fig. 63: Testeo de carga con *Apache JMeter* – Historia de usuario N°1.

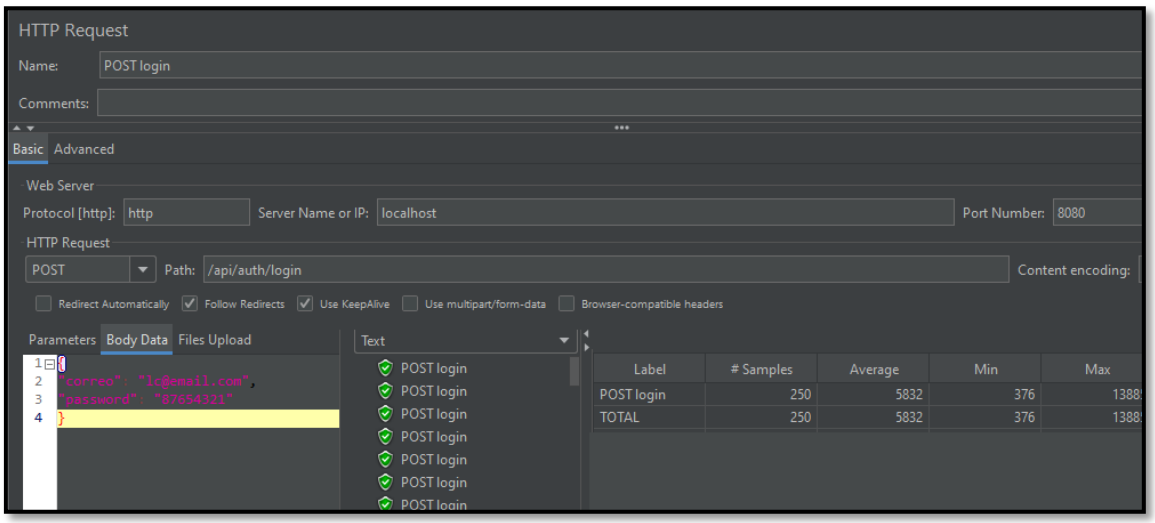


Fig. 64: Testeo de carga con *Apache JMeter* – Historia de usuario N°2.

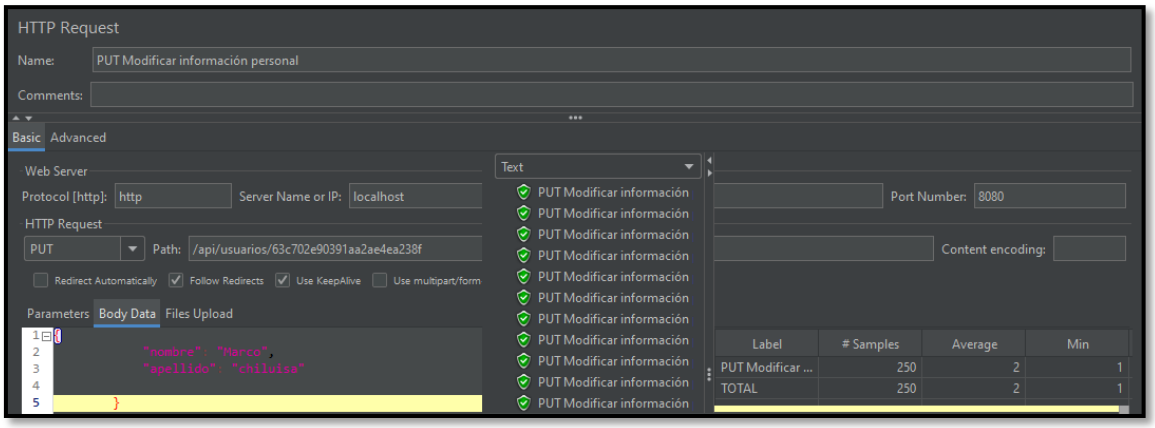


Fig. 65: Testeo de carga con *Apache JMeter* – Historia de usuario N°3.

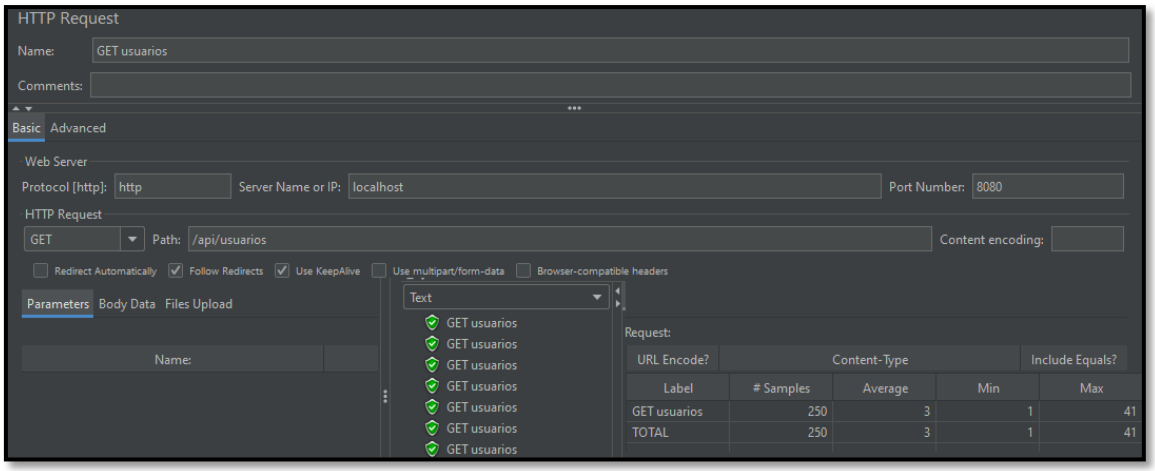


Fig. 66: Testeo de carga con *Apache JMeter* – Historia de usuario N°4.

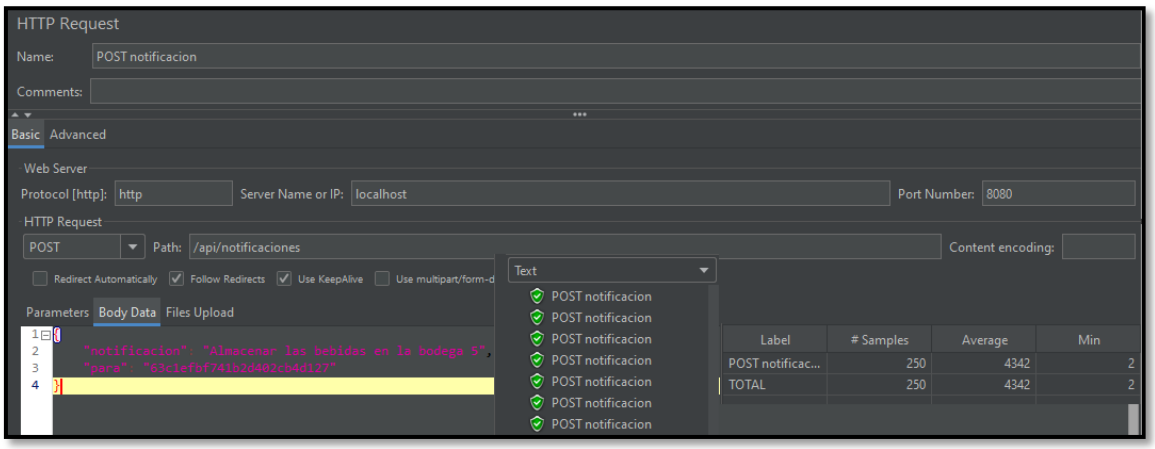


Fig. 67: Testeo de carga con *Apache JMeter* – Historia de usuario N°6.

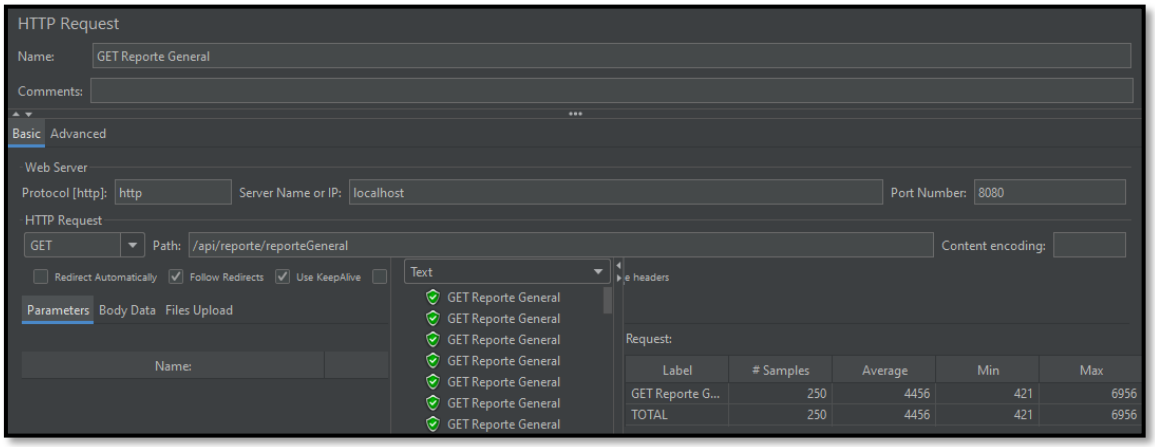


Fig. 68: Testeo de carga con *Apache JMeter* – Historia de usuario N°7.

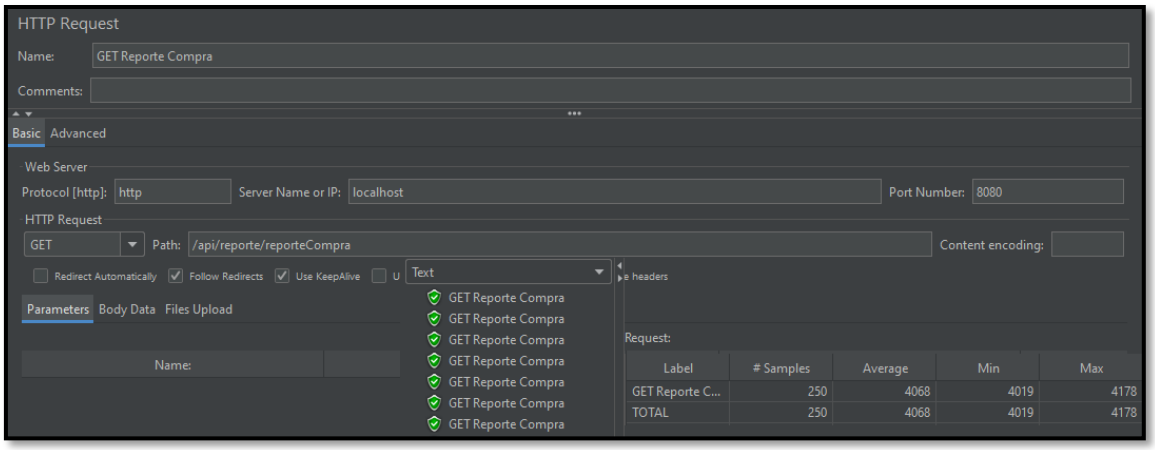


Fig. 69: Testeo de carga con *Apache JMeter* – Historia de usuario N°8.

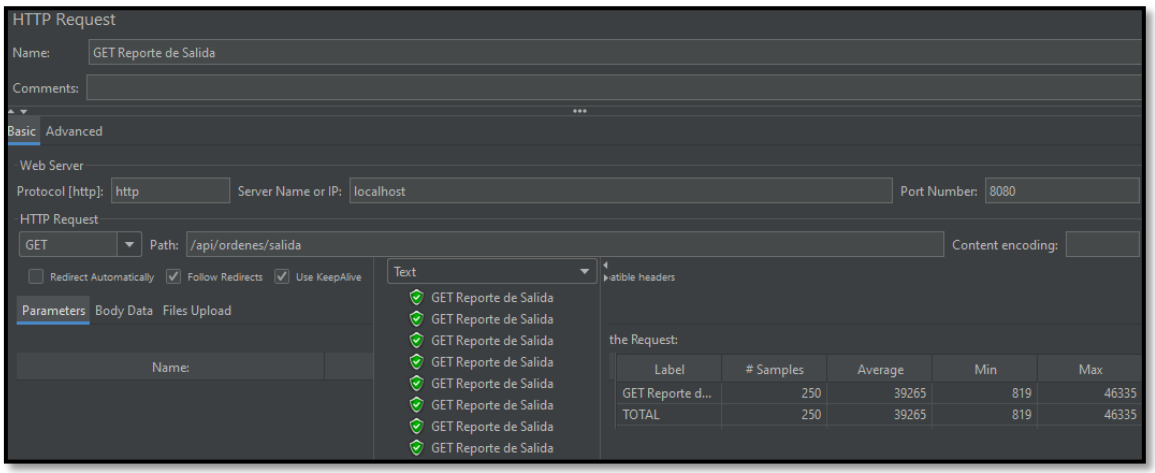


Fig. 70: Testeo de carga con *Apache JMeter* – Historia de usuario N°9.

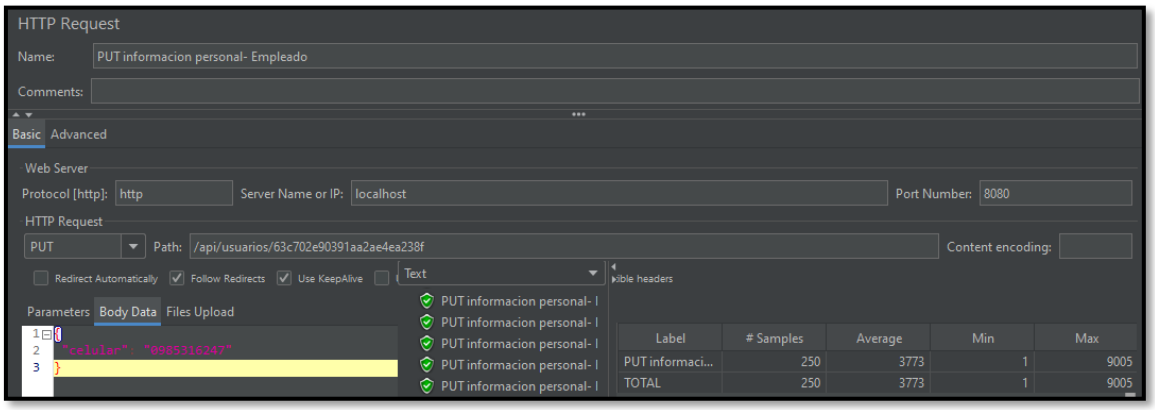


Fig. 71: Testeo de carga con *Apache JMeter* – Historia de usuario N°10.

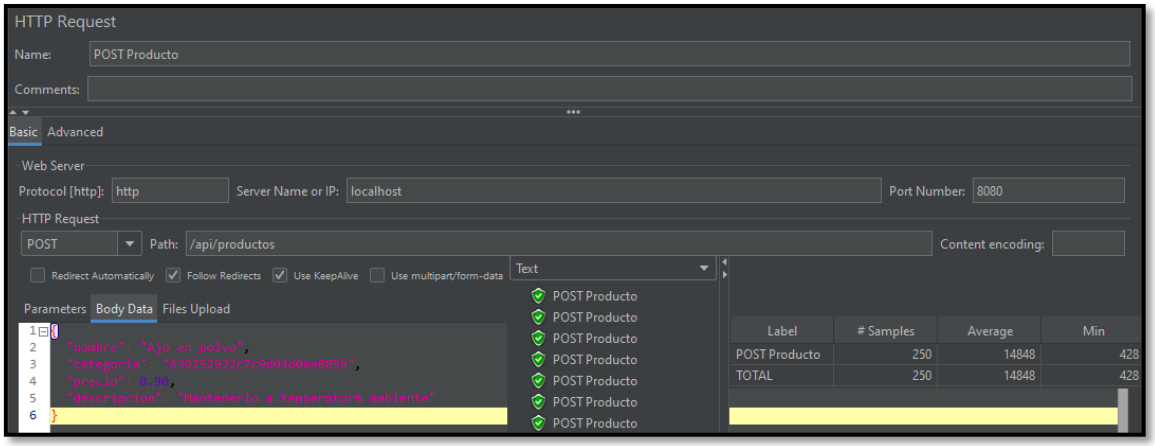


Fig. 72: Testeo de carga con *Apache JMeter* – Historia de usuario N°11.

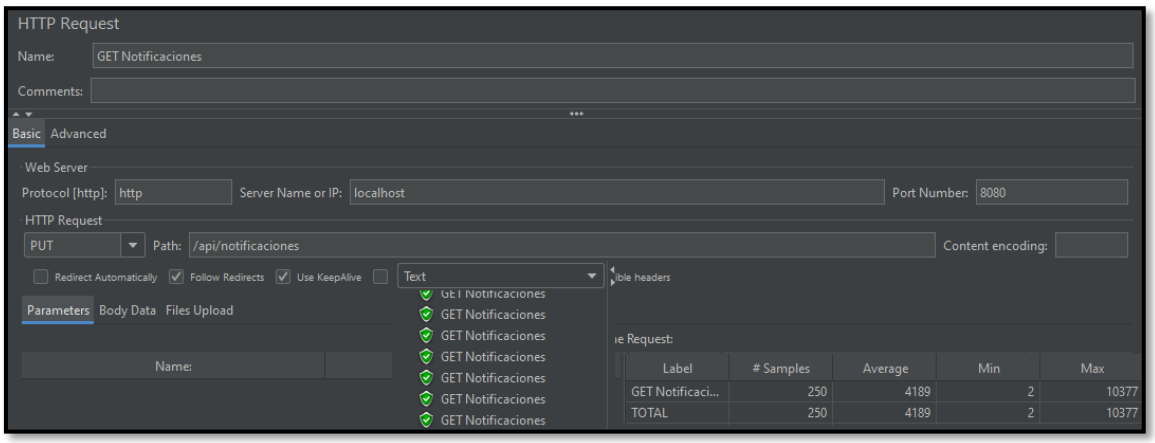


Fig. 73: Testeo de carga con *Apache JMeter* – Historia de usuario N°12.

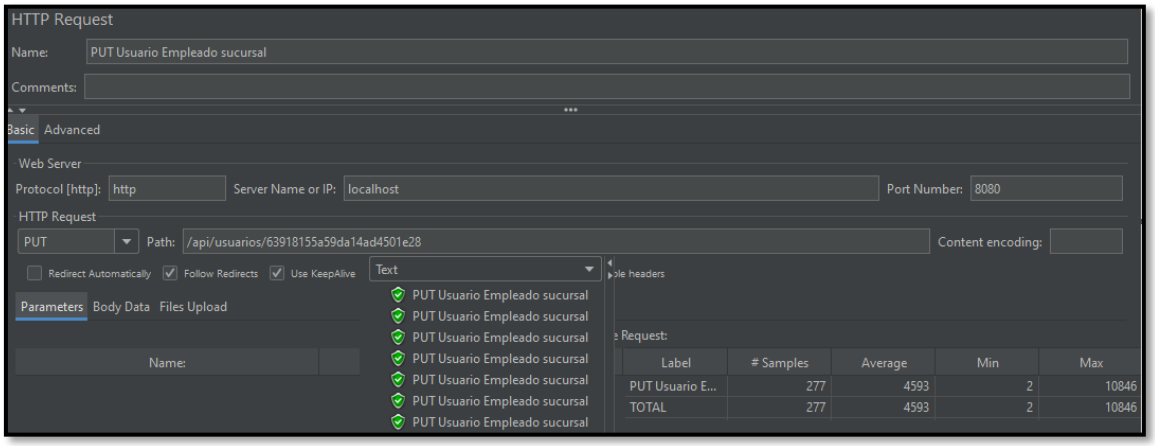


Fig. 74: Testeo de carga con *Apache JMeter* – Historia de usuario N°13.

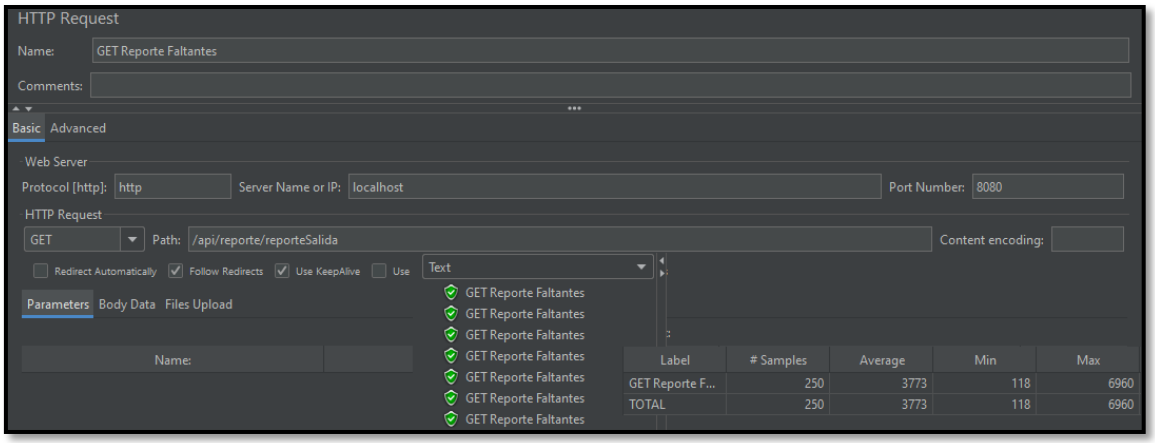


Fig. 75: Testeo de carga con *Apache JMeter* – Historia de usuario N°14.

ANEXO III

A continuación, para visualizar el Manual de Usuario del *backend* se debe digitar la siguiente URL:

https://youtu.be/Ho_ghv5wFBs

En donde se explican de forma clara y precisa los diversos *endpoints* que forman parte del *backend*, así como cada uno de los roles que forman parte de este componente.

ANEXO IV

A continuación, se presenta el enlace de acceso del *backend*, además del repositorio de GitHub, donde se encuentra todo el código del *backend*, además de los pasos a seguir para su instalación en el apartado del README.

Credenciales para el ingreso al *backend.*

Para ingresar al *backend* ya en producción, se ingresa mediante la URL:

https://api-pyme-backend-production.up.railway.app***/***

Repositorio del código fuente del *backend*.

El proyecto se encuentra en un repositorio de *GitHub,* que se accede mediante la siguiente URL:

https://github.com/Lseb1103/Pyme.git