

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**DESARROLLO DE SISTEMA PARA LA GESTIÓN DE LOS
LABORATORIOS DE INFORMÁTICA - ESFOT**

DESARROLLO DE UN BACKEND

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

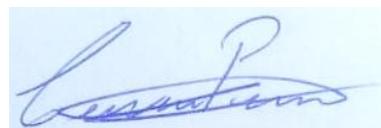
CHRISTIAN ABRAHAN PALACIOS PADILLA

DIRECTOR: ING. BYRON LOARTE

DMQ, febrero 2023

CERTIFICACIONES

Yo, Christian Abrahan Palacios Padilla 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.



CHRISTIAN ABRAHAN PALACIOS PADILLA

christian.palacios@epn.edu.ec

christianpalacios99@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Christian Abrahan Palacios Padilla, 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.

Christian Abrahan Palacios Padilla

DEDICATORIA

El presente trabajo de integración curricular está dedicado a mi familia, quienes son y han sido un fuerte apoyo tanto emocional como económico, a mi hermano Javier que siempre está a mi lado, a mi hermana Vannesa que siempre me trae alegría, a mi madre Jady que siempre está pendiente de que me encuentre bien, a mi padre Abrahan que siempre me ha guiado, a mi cuñado Yeison y a mis sobrinos James y Dylan quienes siempre me han motivado y a Poleth mi mejor amiga quien siempre estuvo conmigo en las buenas y en las malas.

Christian Abrahan Palacios Padilla

AGRADECIMIENTO

Agradezco al Ing. Byron Loarte, MSc. por su gran labor de enseñanza, por el cariño especial que tiene por sus estudiantes y por confiar en nuestro esfuerzo. Agradezco al cuerpo docente de la ESFOT en especial a los docentes de la carrera de Desarrollo de Software: Ing. Ivonne Maldonado, Ing. Juan Pablo Zaldumbide, Ing. Yadira Franco e Ing. Mayra Álvarez; por todo el conocimiento y valores profesionales impartidos en el transcurso de sus cátedras y por la cercanía con sus estudiantes.

Un agradecimiento especial al cuerpo administrativo de la ESFOT quienes además de compañeros de trabajo fueron amigos con quienes compartí muchos momentos siendo quienes motivaron el desarrollo del presente Trabajo de Integración Curricular.

Christian Abrahan Palacios Padilla

ÍNDICE DE CONTENIDO

1	INTRODUCCIÓN	1
1.1	Objetivo general.....	2
1.2	Objetivos específicos	2
1.3	Alcance.....	2
1.4	Marco teórico	4
2	METODOLOGÍA	7
2.1	Metodología de Desarrollo.....	7
	Roles.....	8
	Artefactos	9
2.2	Diseño de arquitectura	11
	Arquitectura Modelo Vista Controlador.....	12
2.3	Herramientas de desarrollo	12
	Librerías.....	13
3	RESULTADOS.....	15
3.1	Sprint 0 Configuración del ambiente de desarrollo	15
	Definición de requerimientos	15
	Estructura del proyecto	18
	Elaboración de la base de datos.....	18
	Definición de roles de usuario.....	19
3.2	Sprint 1 Resultados del desarrollo de <i>endpoints</i> para el perfil administrador	20
	Generar varios <i>endpoints</i> que permiten iniciar sesión, cerrar sesión y modificar contraseña.....	20
	Generar varios <i>endpoints</i> que permiten visualizar y modificar el perfil de usuario	22
	Generar varios <i>endpoints</i> que permiten gestionar pasantes	23
	Generar varios <i>endpoints</i> que permiten gestionar inventarios	24
	Generar varios <i>endpoints</i> que permiten visualizar reporte de inventarios.....	26
3.3	Sprint 2. Resultados del desarrollo de <i>endpoints</i> para el perfil personal administrativo	27
	Generar varios <i>endpoints</i> que permiten solicitar <i>tickets</i> de asistencia	27
	Generar varios <i>endpoints</i> que permiten gestionar comentarios y/o sugerencias.....	28
3.4	Sprint 3. Resultados del desarrollo de <i>endpoints</i> para el perfil profesor .	29
	Generar varios <i>endpoints</i> que permiten registrar comentarios y/o sugerencias	30

3.5	<i>Sprint 4. Resultados del desarrollo de endpoints para el perfil pasante..</i>	31
	Generar varios <i>endpoints</i> que permiten gestionar <i>tickets</i> de asistencia	31
	Generar varios <i>endpoints</i> que permiten gestionar reservas	32
3.6	<i>Sprint 5. Pruebas y Despliegue del backend.....</i>	33
	Aplicación de pruebas unitarias	33
	Aplicación de pruebas de compatibilidad	34
	Aplicación de pruebas de estrés.....	35
	Despliegue del <i>backend</i> en Azure	36
4	CONCLUSIONES.....	37
5	RECOMENDACIONES.....	38
6	REFERENCIAS BIBLIOGRÁFICAS.....	39
7	ANEXOS.....	42
	ANEXO I	43
	ANEXO II	44
	ANEXO III	68
	ANEXO IV	69

RESUMEN

Actualmente el manejo y administración de los laboratorios de informática de la Escuela de Formación de Tecnólogos es una tarea que requiere de un mayor esfuerzo, siendo que estos son usados diariamente. Además, cada laboratorio tiene asignado un número determinado de computadoras los cuales son registrados de forma manual en medios impresos o en raras ocasiones haciendo uso de medios ofimáticos lo que ocasiona confusiones entre el jefe de laboratorio, personal administrativo, pasantes, estudiantes y docentes; ya que no existe un correcto manejo de la información y esto diariamente dificulta el desarrollo de las actividades académicas y de mantenimiento.

Para simplificar las tareas de administración y acceso a los laboratorios en el presente Trabajo de Integración Curricular se ha desarrollado un *backend* para la gestión de los laboratorios de informática de la ESFOT, mediante el cual se simplifiquen las tareas para la gestión de inventarios de computadoras, facilitar el proceso de comunicación entre el cuerpo administrativo, docente y pasantes mediante el envío de comentario y/o sugerencias, *tickets* de asistencia, etc. Logrando de esta manera centralizar toda la información y mejorando así el desarrollo de las actividades referentes al uso de los laboratorios de informática.

Por último, este documento está estructurado siguiendo las siguientes secciones: para la primera sección se detallan los antecedentes, alcance, objetivos y el marco teórico respectivo. En la segunda sección, se detalla la integración de la metodología ágil *Scrum* para el presente proyecto, el modelo de datos para el manejo de la información, la arquitectura MVC, herramientas de codificación y pruebas. En la tercera sección, se detallan las actividades que se han realizado en base a las iteraciones que se han definido y los resultados que se han obtenido. En la cuarta sección, se detalla las conclusiones y recomendaciones que se han obtenido tras el desarrollo e implementación a producción del *backend*.

PALABRAS CLAVE: Inventarios, *Scrum*, Java, Spring Boot, Docker.

ABSTRACT

Currently, the management and administration of the computer laboratories of the School for the Training of Technologists is a task that requires more effort, since they are used on a daily basis. In addition, each laboratory is assigned a certain number of computers which are registered manually in printed media or in rare occasions using office automation, which causes confusion among the laboratory chief, administrative personnel, interns, students and teachers, since there is no correct management of the information and this hinders the development of academic and maintenance activities on a daily basis.

To simplify the tasks of administration and access to the laboratories in this Curricular Integration Work, a backend has been developed for the management of the computer laboratories of the ESFOT, which simplifies the tasks for the management of computer inventories, facilitates the communication process between the administrative staff, teachers and interns by sending comments and / or suggestions, attendance tickets, etc. Thus centralizing all information and improving the development of activities related to the use of computer labs.

Finally, this document is structured according to the following sections: the first section details the background, scope, objectives and the respective theoretical framework. In the second section, the integration of the agile Scrum methodology for the present project, the data model for information management, the MVC architecture, coding and testing tools are detailed. The third section details the activities that have been carried out based on the iterations that have been defined and the results that have been obtained. The fourth section details the conclusions and recommendations obtained after the development and production implementation of the backend.

KEYWORDS: Inventory, *Scrum*, Java, Spring Boot, Docker.

1 INTRODUCCIÓN

En la actualidad, el contar con un sistema de gestión de inventarios y laboratorios en instituciones de educación es crucial ya que permite gestionar de manera ordenada y digital la información generada en los laboratorios de informática [1]. Por otra parte, una incorrecta gestión de la información e inventario deriva en una serie de problemáticas que dificultan el llevar a cabo de forma correcta, las actividades académicas y administrativas tales como: falta de mecanismos preventivos y de acción para la protección de la información, la reserva de laboratorios se gestiona en medios impresos, información desactualizada, duplicidad de información, no existe información centralizada, entre otros. [2].

La Escuela de Formación de Tecnólogos (ESFOT) de la Escuela Politécnica Nacional, dispone de 6 laboratorios de informática en donde se encuentran equipos de cómputo para que los estudiantes puedan hacer uso durante su formación académica. Siendo así que el manejo de seriales, códigos de bien y características propias de cada equipo es una tarea que demanda mucho tiempo y dificultades, teniendo que realizar los registros en medios impresos y en raras ocasiones el uso de archivos de ofimática. Por otro lado, los laboratorios son requeridos para realizar actividades extracurriculares, siendo así que el manejo de reservas constituye una tarea tediosa ya que los docentes deben recurrir a enviar correos electrónicos, mensajes de *WhatsApp*, realizar llamadas o contactarse personalmente con algún miembro del cuerpo administrativo para realizar reserva de laboratorios, asistencia técnica, requerir algún *software* para sus clases, entre otras.

La importancia de contar con sistemas informáticos en la actualidad mejora en gran medida los distintos procesos operativos y aporta automatización a los mismos [3]. Además, otorga varias ventajas en el manejo de la información para las tareas de mantenimiento, actualización de equipos, instalación de *software*, etc., y por otra parte, favorece la toma de decisiones por parte del cuerpo administrativo [4].

Por lo citado anterior y en base a la problemática que existe en los laboratorios de informática de la Escuela de Formación de Tecnólogos, el presente Trabajo de Integración Curricular describe el desarrollo de un *backend* para la gestión de laboratorios e inventario informático denominado “BOTICS”, el cual dispone de varios *endpoints* para que el consumo de la información que se encuentra registrada por cada laboratorio sea fácilmente visualizada del lado del cliente. Además, dispone de una serie de roles para una mejor protección de la información, así como varios módulos para la gestión de tickets de asistencia para el cuerpo docente y administrativo de la ESFOT.

1.1 Objetivo general

Desarrollar un sistema para la gestión y administración de los laboratorios de informática - ESFOT.

1.2 Objetivos específicos

1. Identificar las funcionalidades necesarias para el desarrollo del *backend*.
2. Establecer la estructura y arquitectura de la base de datos para el desarrollo del *backend* según la recopilación de requerimientos.
3. Codificar los *endpoints* y módulos para el *backend* según la recopilación de requerimientos.
4. Testear cada uno de los *endpoints* verificando su correcta funcionalidad.
5. Desplegar el *backend* a producción.

1.3 Alcance

Dentro de lo que constituye una aplicación web, el *backend* implementa toda la lógica según la recopilación de requerimientos. Además, es un componente de gran importancia el cual implementa diversas tecnologías para brindar acceso a dicha información de forma segura en base a roles o niveles de acceso, establecer niveles de abstracción o integridad de los datos, mantener un servicio sin interrupciones y una optimización adecuada para garantizar una escalabilidad a futuro [5].

Dentro de las instituciones educativas los sistemas informáticos han otorgado una mejora al desarrollo de actividades académicas y docentes [6]. Siendo así que, en los laboratorios de informática de la ESFOT la implementación del *backend* permite que los administradores puedan manejar adecuadamente la información de cada uno de los laboratorios, gestionar de forma adecuada los tickets de asistencia por parte de los pasantes y una correcta gestión en lo que respecta a comentarios y sugerencias por parte del personal administrativo y docente. Utilizando para ello, varias tecnologías de desarrollo modernas y escalables del lado del servidor, un modelo arquitectónico para una correcta organización a nivel de código, una metodología de desarrollo ágil para la realización y cumplimiento de los objetivos del presente proyecto y una serie de pruebas para garantizar la calidad del producto final. Por último, el *backend* dispone de 4 tipos de perfiles de

usuarios que se describen a continuación para que cada uno pueda visualizar diferentes módulos según el rol asignado.

Perfiles que se establecen:

- Administrador.
- Personal administrativo.
- Profesor.
- Pasante.

Para el perfil administrador se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Endpoints* que permiten gestionar pasantes.
- *Endpoints* que permiten gestionar inventarios.
- *Endpoints* que permiten visualizar reporte de inventarios.

Para el perfil personal administrativo se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Endpoints* que permiten solicitar *tickets* de asistencia.

Para el perfil profesor se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Endpoints* que permiten solicitar *tickets* de asistencia.
- *Endpoints* que permiten registrar comentarios y sugerencias.

Para el perfil pasante se generan:

- *Endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Endpoints* que permiten visualizar y modificar el perfil de usuario.

- *Endpoints* que permiten gestionar *tickets* de asistencia.
- *Endpoints* que permiten gestionar comentarios y sugerencias.

1.4 Marco teórico

El *software* es un conjunto de instrucciones que cumplen con una serie de funciones establecidas para dar solución a un problema [7], estos se encuentran clasificados en varios tipos y entre los cuales se dispone:

- *Software* de sistema: los cuales son un conjunto de programas que dan servicio a otros programas.
- *Software* de aplicación: los cuales son programas aislados destinados a resolver una necesidad o problemática en específico.
- Aplicaciones *web*: las cuales son conocidas como “*webapps*” y que las mismas implementan soluciones en tiempo real con conexiones a distintas bases de datos.

La calidad de *software* consiste de forma general en una serie de estándares los cuales miden el grado en el que un sistema informático satisface las necesidades establecidas para cada uno de los usuarios que interactúan con el mismo. Se emplean mecanismos de medición interna y externa, siendo que los parámetros internos hacen referencia a las propiedades inherentes del sistema y los parámetros externos hacen referencia al comportamiento del sistema con propiedades externas [8].

Se denomina *backend* a la capa que permite el acceso a los datos definidos para una aplicación los cuales no son accesibles por el usuario final sin una interfaz adecuada, a su vez esta capa controla la lógica que se ha definido para la aplicación según la recopilación de requerimientos, siendo así que el *backend* trabaja por el lado del servidor haciendo uso de sistemas gestores de almacenamiento para Bases de datos relaciones y no relaciones y la implementación de algún lenguaje de programación como: Java, PHP, C# o Node.JS [9].

Una base de datos relacional es un sistema de almacenamiento de datos estructurados basado en registros el cual emplea el modelo basado en objetos, estos son gestionados mediante los “DBMS” o sistemas de manejo de bases de datos los cuales son un conjunto de elementos de *software* y programas los cuales permiten la consulta o actualización de

archivos por parte del usuario que los administra. Una base de datos de tipo relacional está basada en modelos los cuales son abstracciones del mundo físico aportando así una mayor comprensión e integridad de la información [10].

Siendo una de las tecnologías que han cambiado radicalmente el funcionamiento de los servicios *web*, las API's o también denominadas Interfaces de Programación de Aplicaciones son elementos o paquetes de *software* los cuales permite la interacción entre aplicaciones, siendo así que una API establece la conexión necesaria para que varios componentes de *software* trabajen en conjunto y compartan recursos con un fin determinado [11].

Java se caracteriza como un lenguaje sencillo el cual está orientado a objetos y tiene características llamativas como: robustez, interpretado, multihilo, dinámico, portable, eficaz, etc. Además, este lenguaje es considerado como un lenguaje distribuido y portable, ya que cuenta con una “máquina virtual Java” la cual consiste en una plataforma virtual que habilita la ejecución de aplicaciones java en un gran número de entornos y Sistemas Operativos de todo tipo [12].

Spring es un *framework*, el cual está orientado al desarrollo de *software* en entornos empresariales, este es uno de los frameworks de Java más utilizados ya que ofrece una gran variedad de características que agilizan y facilitan la implementación de soluciones de *software*, entre las cuales se tiene: inyección de dependencias, aplicación de *testing* automatizado, gestión de transacciones, entre otros. Adicional a ello, Spring cuenta con varios complementos multipropósito como: [13].

- *Spring MVC*: destinado para el desarrollar aplicaciones *web*.
- *Spring Security*: destinado para la integración de funcionalidades de autorización y autenticación.
- *Spring Data*: destinado para gestionar Bases de datos relaciones, no relacionales y servicios en la nube.
- *Spring Web Services*: destinado para la implementación de servicios *web* SOAP.

Spring Boot es una extensión de *Spring* el cual tiene como principal objetivo la simplificación en el desarrollo gracias a sus librerías y herramientas, implementa la arquitectura Modelo Vista Controlador (MVC) para el desarrollo de servicios REST, API's REST, aplicaciones web estáticas, etc. Además, implementa servidores embebidos como

Tomcat o Jetty facilitando de esta manera el desarrollo de aplicaciones independientes y el despliegue de aplicaciones a un ambiente de producción [14].

El despliegue de una aplicación *web* es el proceso mediante el cual se pone a disposición del público un servicio o información asegurando de esta manera la estabilidad, disponibilidad y acceso a los recursos del mismo. En ese sentido, para un correcto despliegue se deben considerar distintas medidas tanto de seguridad, infraestructura y de rendimiento para el entorno y contexto que se establezca [15].

2 METODOLOGÍA

El estudio de casos es un modelo investigativo empleado generalmente para el avance científico ya sea en el área social, tecnológica o humanística. Además, impulsa el aprendizaje acerca de una situación específica mediante el análisis comprensivo y sistemático de dicha situación tomada en todo su conjunto y dentro de su propio contexto. Es por esta razón, que el estudio de casos permite dar respuesta a fenómenos a través de una serie de investigaciones [16].

Por lo descrito previamente, este proyecto de Integración Curricular implementa el estudio de casos, debido a que se investiga sobre las principales problemáticas existentes en la gestión de inventarios y en el registro de la información de laboratorios de informática de la ESFOT. Con ello, es posible llevar a cabo el desarrollo de un *backend* para la gestión de la información, *tickets* de asistencia, reservas e inventarios de los laboratorios de informática; mejorando la gestión de los laboratorios e información con el apoyo de nuevas tecnologías de desarrollo de *software*.

2.1 Metodología de Desarrollo

Dentro del desarrollo de sistemas y aplicaciones las metodologías de desarrollo son un conjunto de técnicas y métodos que permiten abordar las actividades definidas dentro del ciclo de vida referente a un proyecto de *software*. Además, dependiendo del tipo de metodología implementan casos de uso, artefactos, roles, buenas prácticas y diversas técnicas de tal forma que la administración del proyecto de *software* permita altas posibilidades de éxito [17].

Siendo que en la actualidad los sistemas *software* operan en un entorno global que se encuentra en constante cambio, el *software* debe responder a estos cambios y ofrecer funcionalidades operacionales que se adapten forma ágil a dichos cambios [41]. Las metodologías ágiles de desarrollo para sistemas y aplicaciones son el conjunto de técnicas y herramientas que permiten implementar proyectos de *software* flexibles, modulares y adaptables, además de promover un entorno colaborativo y una alta cercanía al cliente [18]. Es por esta razón, que en este Trabajo de Integración Curricular se desarrolla empleando la metodología ágil *Scrum* y muestra de ello en las siguientes secciones se describe la implementación de cada fase de esta metodología para el desarrollo del *backend*.

Roles

Para llevar a cabo las actividades que se han establecido para el desarrollo del *backend* cada uno de los integrantes del equipo cumple con un rol establecido mediante la metodología ágil Scrum, siendo así que se establece un *Product Owner*, *Scrum master* y el *Development Team* [19]. A continuación, se describe cada uno de los roles descritos anteriormente:

Product Owner

Es la persona responsable del ciclo de vida del proyecto, a su vez se encarga de proporcionar los objetivos y requerimientos como propiedades del mismo [20]. Por esta razón, en la **TABLA I** se muestra la persona responsable para cumplir el rol y las tareas asignadas en el desarrollo del *backend*.

Scrum Master

Es la persona encargada de orientar y dirigir al equipo de desarrollo (Development Team) de tal forma que se cumplan con todos los objetivos que se han definido para el proyecto de *software*. Además, ofrece asistencia al equipo, soluciones en tiempo de desarrollo, agilidad de procesos y adaptabilidad [21]. Por esta razón, en la **TABLA I** se muestra la persona responsable para cumplir el rol y las tareas asignadas en el desarrollo del *backend*.

Development Team

Es un grupo de personas altamente comprometidas que se encargan de la implementación del proyecto *software*, de esta forma, este rol se conforma con un conjunto de desarrolladores en el cual se encuentran *testers* y *designers*. El *Development Team* tiene como principal característica la auto-organización, la cual deriva en equipos independientes con procesos de toma de decisiones eficaces y acciones de respuesta eficientes [22]. Es por esto que, la **TABLA I** muestra la persona responsable para cumplir el rol y las tareas asignadas en el desarrollo del *backend*.

TABLA I: Designación de roles para el proyecto.

ROL	INTEGRANTES
<i>Product Owner</i>	Ing. Byron Loarte, MSc.
<i>Scrum Master</i>	Ing. Byron Loarte, MSc.
<i>Development Team</i>	Christian Palacios

Artefactos

Los artefactos dentro de la metodología *Scrum* son elementos que representan el valor o trabajo que se aporta durante el desarrollo de un proyecto, es decir, es la forma en la cual se maneja la información de tal forma que se prioriza la transparencia y permite el correcto desempeño de actividades [23]. A continuación, se presenta los artefactos que se han utilizado como parte de la metodología *Scrum*.

Recopilación de requerimientos

La recopilación de requerimientos en base a lo establecido por la metodología *Scrum*, es la base y el elemento con mayor importancia para el desarrollo *software*, ya que permite la recolección y esquematización de las funcionalidades con las que cuenta un sistema, además establece las operaciones que se realizan en el mismo [24]. Por esta razón, en la **TABLA II** se muestra el formato empleado para la recolección de requerimientos y el correcto ordenamiento de los mismos para el presente proyecto, por otro lado la recolección completa de requerimientos se la puede observar en el **ANEXO II** de este documento.

TABLA II: Tabla de recopilación de requerimientos.

Recopilación de requerimientos		
Tipo de sistema	ID-RR	Enunciado del ítem
<i>Backend</i>	RR008	Como usuario pasante necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">Gestionar <i>tickets</i> de asistencia
	RR009	Como usuario pasante necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">Gestionar reservas

Historias de Usuario

Las historias de usuario permiten describir detalladamente los requerimientos que se han recolectado previamente. Además, permiten determinar aspectos importantes para el desarrollo como: tipo de usuario, prioridad de dicha historia, riesgo, entre otros detalles [25]. Esto permite al equipo de desarrollo determinar la importancia de cada requerimiento y conocer cómo se deben implementar para cumplir con la funcionalidad establecida. Con ello, la **TABLA III** muestra el formato empleado para implementar las historias de usuario,

por otro lado la recolección completa de las demás historias de usuario se las puede observar en el **ANEXO II** de este documento.

TABLA III: Historia de usuario 1 – Registrarse.

HISTORIA DE USUARIO	
Identificador (ID): HU001	Usuario: Profesor y Administrativo.
Nombre Historia: Registrarse.	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Iteración Asignada: 1	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil profesor y administrativo puedan registrarse.	
Observación: Los usuarios con perfil profesor y administrativo necesitan registrarse para acceder a los demás <i>endpoints</i> asignados. Además, para el registro se requiere el nombre de usuario, correo y contraseña.	

Product Backlog

Establecido por la metodología *Scrum*, el *Product Backlog* es la recopilación final de los requerimientos que se han establecido para el desarrollo de una aplicación o sistema *software*. Además, permite recolectar información relevante como: resumen de las historias de usuario definidas, a qué iteración pertenecen dentro del desarrollo, prioridad y el estado en el cual se encuentran [26]. Para lo cual, la **TABLA IV** presenta el formato definido para la implementación del *Product Backlog*, por otro lado la recopilación completa se la puede observar en el **ANEXO II** de este documento.

TABLA IV: Formato para presentar el *Product Backlog*.

ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU009	Enviar comentarios y/o sugerencias.	4	Finalizado	Media
HU010	Solicitar reservas.	4	Finalizado	Alta

Sprint Backlog

El *Sprint Backlog* es el esquema con el cual se determina cuáles son las iteraciones o comúnmente llamados *Sprints*. Es por esta razón, que cada *Sprint* cuenta con un conjunto de historias de usuario para poder ordenadas de tal forma que se establezca los módulos funcionales y que estos a su vez permitan que el *Development Team* conozca que funcionalidades son prioritarias para el *Sprint* en el que se encuentra [27]. Para lo cual, la **TABLA V** muestra el formato aplicado para el desarrollo de cada uno de los *Sprints*, por otro lado la tabla completa se la puede observar en el **ANEXO II** de este documento.

TABLA V: Formato de *Sprint Backlog*.

ELABORACIÓN DE SPRINT BACKLOG						
ID – SB	NOMBRE	MÓDULO	ID-HU	HISTORIA DE USUARIO	TAREA	TIEMPO ESTIMADO
SB003	Diseño e implementación de <i>endpoints</i> para el usuario administrativo .	Módulo de solicitud de <i>tickets</i> de asistencia	HU008	Solicitar <i>tickets</i> de asistencia	<ul style="list-style-type: none">• Diseño e implementación de <i>endpoints</i> para solicitar <i>tickets de asistencia</i>.• Validación de los datos requeridos• Registro en la base de datos	30H

2.2 Diseño de arquitectura

Para diseñar la arquitectura se selecciona y trabaja sobre un patrón arquitectónico el cual permite modular la aplicación y por consiguiente presentar un *backend* ordenado y de fácil modificación [42]. Es por ello, que en la siguiente sección se detalla la arquitectura que se ha utilizado para el desarrollo del *backend*.

Arquitectura Modelo Vista Controlador

Esta es una arquitectura para el desarrollo de *software* el cual permite implementar un modelado por capas, este modelo por capas busca separar la lógica de la aplicación en controladores, una parte visual para la presentación del contenido visual y una parte para la gestión de la información a través del modelo [28]. En ese contexto, la **Fig. 1** muestra el diseño de la arquitectura empleada para el *backend* conjuntamente con una serie de herramientas de desarrollo.

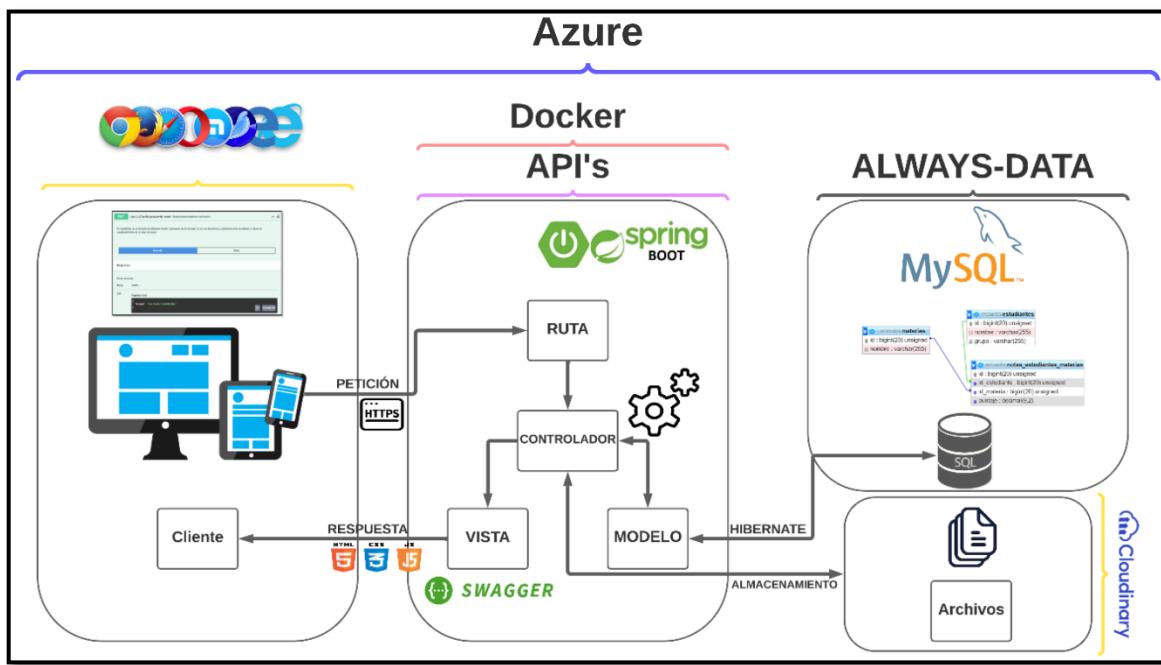


Fig. 1: Arquitectura que se ha implementado para el *backend*.

2.3 Herramientas de desarrollo

Las herramientas que han sido elegidas para la codificación del *backend* permiten el desarrollo de cada uno de los *endpoints* los cuales son accesibles gracias al conjunto de herramientas que son exclusivas para el despliegue de aplicaciones *software*, los cuales permiten gestionar datos mediante consultas e implementando sistemas de contenedores [29]. Por ello, la **TABLA VI** presenta el listado de herramientas que se han seleccionado y de qué forma aportan en el desarrollo del proyecto.

TABLA VI: Herramientas que se han empleado para el desarrollo del *backend*.

HERRAMIENTA	JUSTIFICACIÓN
Spring Boot	Es una extensión de <i>Spring</i> , que agiliza la configuración de <i>Beans</i> y clases para la inyección de dependencias [30].
MySQL	Permite almacenar toda la información de manera ordenada y la rápida ejecución de comandos SQL para gestionar consulta de datos [31].
Docker	Agiliza el despliegue del <i>backend</i> en cualquier sistema o entorno sin la necesidad de configuración del mismo [32].
LocalTunnel	Facilita la configuración de un servidor para el acceso https al sistema desde internet [33].

Librerías

En la **TABLA VII** se evidencian las librerías que se han utilizado para crear los diferentes *endpoints*.

TABLA VII: Librerías empleadas para el desarrollo del *backend*.

LIBRERÍA	DESCRIPCIÓN
Spring-Starter-Data-JPA	Simplifica la implementación y trabajo con sistemas de almacenamiento de datos y consultas [34].
Spring-Oauth2-Client	Aporta la capa de seguridad para trabajar con la arquitectura OAUTH2 en proyectos <i>Spring</i> [35].
Spring Security	Simplifica la configuración de clases para aplicar funciones de seguridad y <i>middlewares</i> [36].
Spring Web	Facilita el despliegue y ejecución de aplicaciones web implementando servidores embebidos [37].
Mysql-Connector	Permite la rápida conexión con bases de datos relaciones SQL e implementa un JDBC [38].

Spring-Starter-Mail	Aporta métodos para la implementación de funcionalidades y servicios con correo electrónico [39].
----------------------------	---

3 RESULTADOS

En este apartado del presente documento, se detallan los distintos resultados alcanzados tras la implementación y desarrollo de los *endpoints* que describen la lógica del *backend*, a su vez se detalla el resultado de cada prueba y el respectivo despliegue a producción. Además, el detalle de cada uno de los resultados que se han alcanzado se describen por medio de iteraciones o comúnmente llamado *Sprints*.

3.1 Sprint 0 Configuración del ambiente de desarrollo

A través de lo establecido dentro del Sprint Backlog el presente Sprint presenta estas tareas:

- Definición de requerimientos.
- Estructura del proyecto *backend*.
- Elaboración de la base de datos.
- Definición de roles de usuario.

Definición de requerimientos

Implementación de *endpoints* para el registro de usuarios

El *backend* implementa un *endpoint* que permite registrar a usuarios con perfil personal administrativo y profesor mediante los campos que se han establecido.

Implementación de *endpoints* para el inicio de sesión, cierre de sesión y modificar contraseña

El *backend* implementa un *endpoint* que permite a los usuarios con perfil administrador, personal administrativo, profesor y pasante inicien sesión mediante los campos que se han establecido. Además, de un *endpoint* que permite el cierre de sesión de los usuarios y de varios *endpoints* para la modificación de su contraseña.

Implementación de *endpoints* para modificar el perfil de usuario

El *backend* implementa un *endpoint* que permite a los usuarios con perfil administrador, personal administrativo, profesor y pasante visualicen el perfil personal del usuario.

Implementación de *endpoints* para gestionar pasantes

El *backend* implementa *endpoints* que autorizan al usuario con perfil administrador gestionar pasantes. Permitiendo de esta manera que el usuario pueda realizar varias

acciones como: visualizar el listado de pasantes que se han registrado, registrar pasantes, habilitar y deshabilitar pasantes de tal forma que sólo los usuarios con perfil pasante que cuenten con una cuenta habilitada pueden iniciar sesión.

Implementación de endpoints para gestionar inventarios

El *backend* implementa *endpoints* que autorizan al usuario con perfil administrador gestionar inventarios. Permitiendo de esta manera que el usuario pueda realizar varias acciones como: visualizar el listado de laboratorios con las características de cada uno, registrar computadoras con cada una de las características pertinentes, registrar el movimiento de las computadoras, etc.

Implementación de endpoints para visualizar reporte de inventarios

El *backend* implementa *endpoints* que autorizan al usuario con perfil administrador pueda visualizar reportes de inventarios. Permitiendo de esta manera que el usuario pueda visualizar el reporte de inventario de computadoras por laboratorio y el historial del movimiento de computadores entre cada laboratorio.

Implementación de endpoints para gestionar comentarios y/o sugerencias

El *backend* implementa *endpoints* que permiten al usuario con perfil personal administrativo gestionar comentarios y/o sugerencias realizadas por los usuarios con perfil profesor. Permitiendo de esta manera que el usuario pueda visualizar y dar respuesta a los comentarios y/o sugerencias que se encuentra registradas.

Implementación de endpoints para solicitar tickets de asistencia

El *backend* implementa un *endpoint* que permite al usuario con perfil profesor solicitar *tickets* de asistencia y ser atendido por el usuario con perfil pasante.

Implementación de endpoints para enviar comentarios y/o sugerencias

El *backend* implementa un *endpoint* que permite al usuario con perfil profesor enviar comentarios y/o sugerencias en lo que respecta a los laboratorios y obtener respuesta por parte del usuario con perfil personal administrativo.

Implementación de endpoints para solicitar reservas

El *backend* implementa un *endpoint* que permite al usuario con perfil profesor solicitar reservas de algún laboratorio y obtener una respuesta detallada por parte del usuario con perfil pasante sobre la reserva.

Implementación de *endpoints* para gestionar *tickets* de asistencia

El *backend* implementa *endpoints* para que el usuario con perfil pasante pueda gestionar tickets de asistencia, para que de esta manera pueda visualizar todos los *tickets* de asistencia que se han generado por parte del usuario con perfil profesor, atender y dar una respuesta a cada uno de ellos.

Implementación de *endpoints* para gestionar reservas

El *backend* implementa *endpoints* que permiten al usuario con perfil pasante gestionar reservas por los usuarios con perfil profesor, para que puedan ser atendidos y dar respuesta a cada uno de ellos. Por último, en la **Fig. 2** se presentan las funcionalidades de los usuarios con perfil administrador, personal administrativo, profesor y pasante.

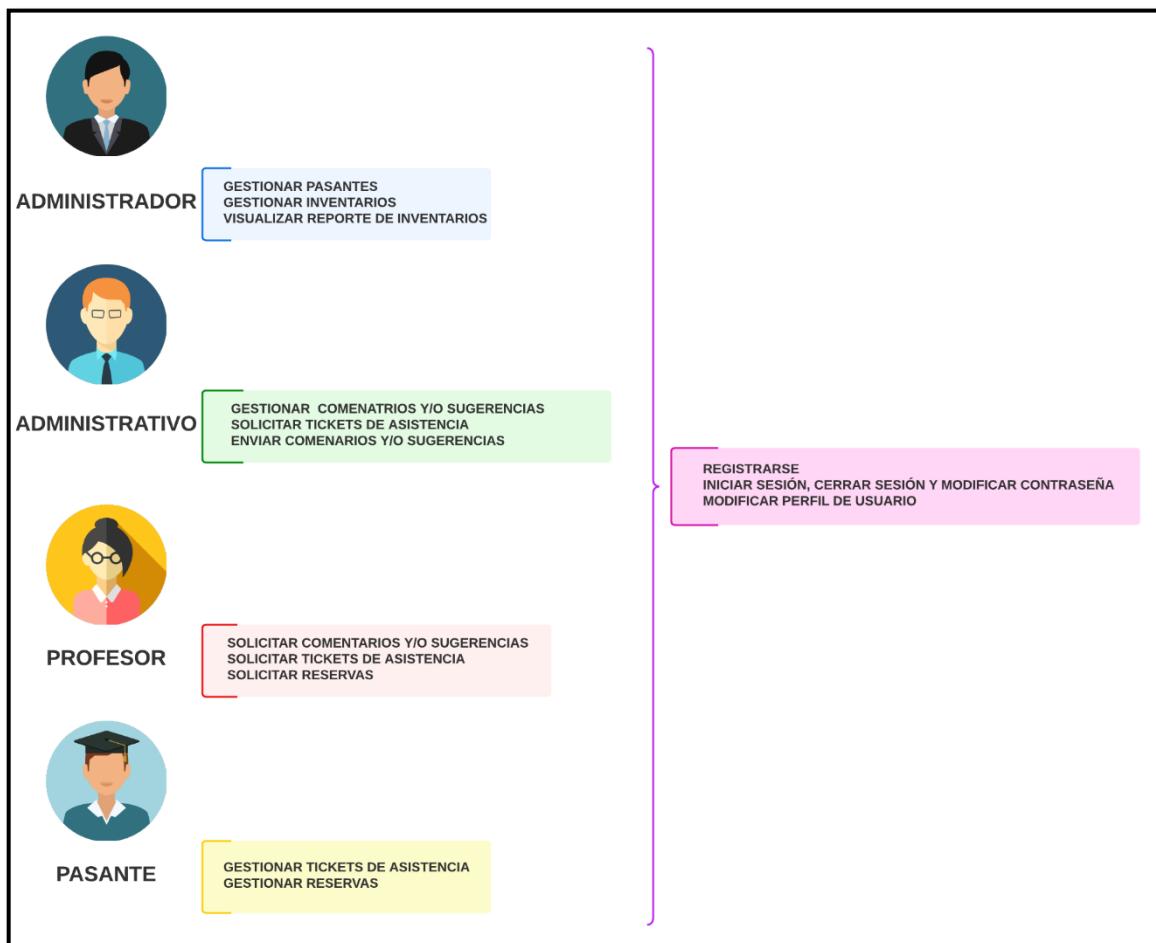


Fig. 2: Perfiles de usuario y funcionalidades dentro del *backend*.

Estructura del proyecto

Se ha utilizado el editor de código IntelliJ IDEA, el cual permite inicializar rápidamente un proyecto definiendo la estructura en base al patrón arquitectónico establecido y los respectivos archivos de configuración y directorios necesarios, para la implementación de los distintos *endpoints* del *backend*. Muestra de ello, la **Fig. 3** presenta la estructura del *backend* que se ha desarrollado.

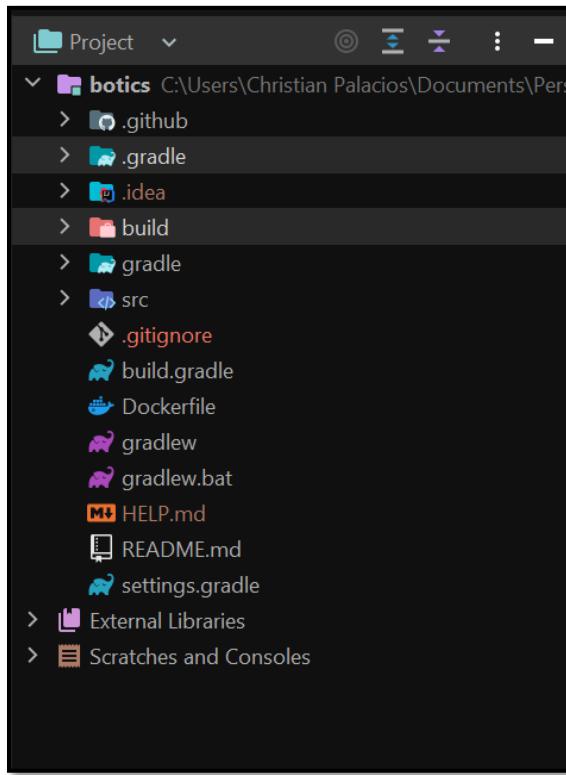


Fig. 3: Estructura del proyecto *backend*.

Elaboración de la base de datos

Se ha hecho uso de MySQL como sistema gestor de bases de datos para implementar una base de datos relacional para que toda la información de laboratorios y usuarios sea integrada gracias a una serie de relaciones que se han establecido, muestra de ello la **Fig. 4** presenta las 13 tablas que se han definido para el almacenamiento de la información, por otra parte, en el **ANEXO II** se encuentra el modelo completo con todas sus relaciones respectivas.

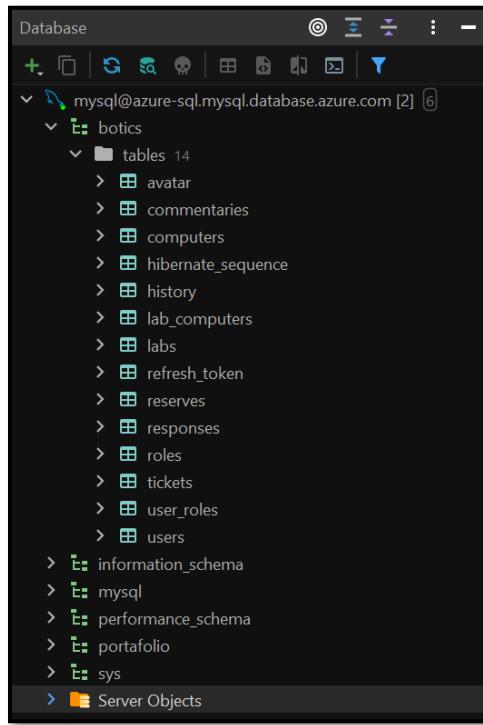


Fig. 4: Tablas implementadas en la base de datos del *backend*.

Definición de roles de usuario

Como parte del desarrollo del *backend* se han definido 4 roles de usuario, los cuales se encuentran en la **Fig. 5**. Además, se establece los niveles de acceso y las acciones que pueden realizar dentro del *backend*.

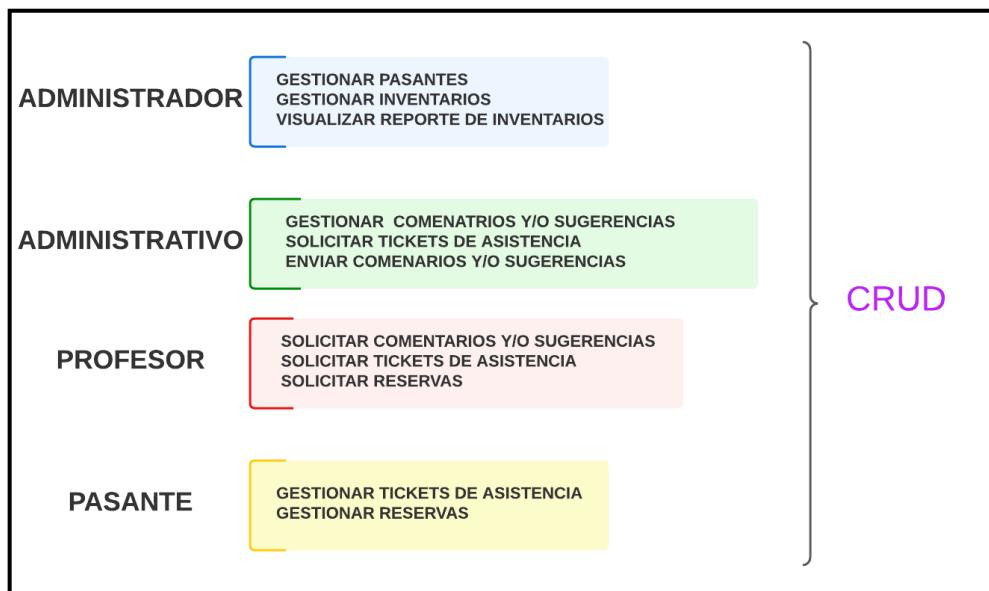


Fig. 5: Roles de usuario del proyecto *backend*.

3.2 Sprint 1 Resultados del desarrollo de *endpoints* para el perfil administrador

A través de lo establecido dentro del *Sprint Backlog* el presente *Sprint* presenta estas tareas:

- *Generar endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña.
- *Generar endpoints* que permiten visualizar y modificar el perfil de usuario.
- *Generar endpoints* que permiten gestionar pasantes.
- *Generar endpoints* que permiten gestionar inventarios.
- *Generar endpoints* que permiten visualizar reporte de inventarios.

Generar varios *endpoints* que permiten iniciar sesión, cerrar sesión y modificar contraseña

La información se encuentra almacenada y gestionada en la base de datos SQL. Para la lectura y acceso a los recursos, el *backend* implementa varios *endpoints* que definen las diferentes rutas privadas. Por consiguiente, el acceso a estos recursos privados por parte de los usuarios con perfil administrador, administrativo, profesor y pasante se implementa un *endpoint* de tipo *POST* para el registro de nuevos usuarios mediante un formulario, posteriormente se implementa un *endpoint* de tipo *POST* para el inicio de sesión el cual autentica al usuario y autoriza el acceso a los recursos en base a su rol de usuario como se muestra en la **Fig. 6**, y en la **Fig. 7** se muestra el resultado de su prueba unitaria. Por último, se implementan varios *endpoints* de tipo *POST* para modificar la contraseña en el caso de que un usuario olvide su contraseña este pueda restablecerla, haciendo uso del ingreso de un correo electrónico y un segundo campo para ingresar la nueva contraseña como se muestra en la **Fig. 8** y el resultado de la prueba unitaria respectiva en la **Fig. 9**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

POST /api/v1/auth/signin Endpoint para iniciar sesión.

Se otorgan los detalles del usuario junto con un token de autorización el cual caduca cada 15 minutos, a su vez almacena el token de reinicio en la base de datos.

Responses

Server response

Code	Details
200	Response body

```
{
  "id": 1,
  "username": "admin",
  "email": "cristhianpalacios3@gmail.com",
  "roles": [
    "ROLE_ADMIN"
  ],
  "jwtToken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbilsImIzcyI6ImJvdGljcyIsImIhdCI6MTY3MzQ3MDYyMCwiZXhwIjoxNjcZNDCxNTIwfQ.POD4ON1prZgu2NjK34iCytb-fV0wAYhBfgtABuCM5J6V9UpwhnfjUM0v1ydlsOhmL9wy7AQmjtt4ggZTPC8Q",
  "refreshToken": "fb183a3a-3fc7-4059-b0de-ba33f2d5de60"
}
```

Copy **Download**

Fig. 6: Endpoint para el inicio de sesión.

```
AuthTest.signinShouldReturnOk ✘
Tests passed: 1 of 1 test – 11 sec 697 ms
insert
into
  refresh_token
  (created_at, expiry_date, token, updated_at, user_id, id)
values
  (?, ?, ?, ?, ?, ?)
2023-01-11 16:15:28.504 INFO 2740 --- [ionShutdownHook] j.LocalContainerEntityMana
2023-01-11 16:15:28.539 INFO 2740 --- [ionShutdownHook] com.zaxxer.hikari.HikariDa
2023-01-11 16:15:28.611 INFO 2740 --- [ionShutdownHook] com.zaxxer.hikari.HikariDa
BUILD SUCCESSFUL in 3m 10s
4 actionable tasks: 2 executed, 2 up-to-date
16:15:29: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AuthTest.'
```

Fig. 7: Prueba unitaria del endpoint para el inicio de sesión.

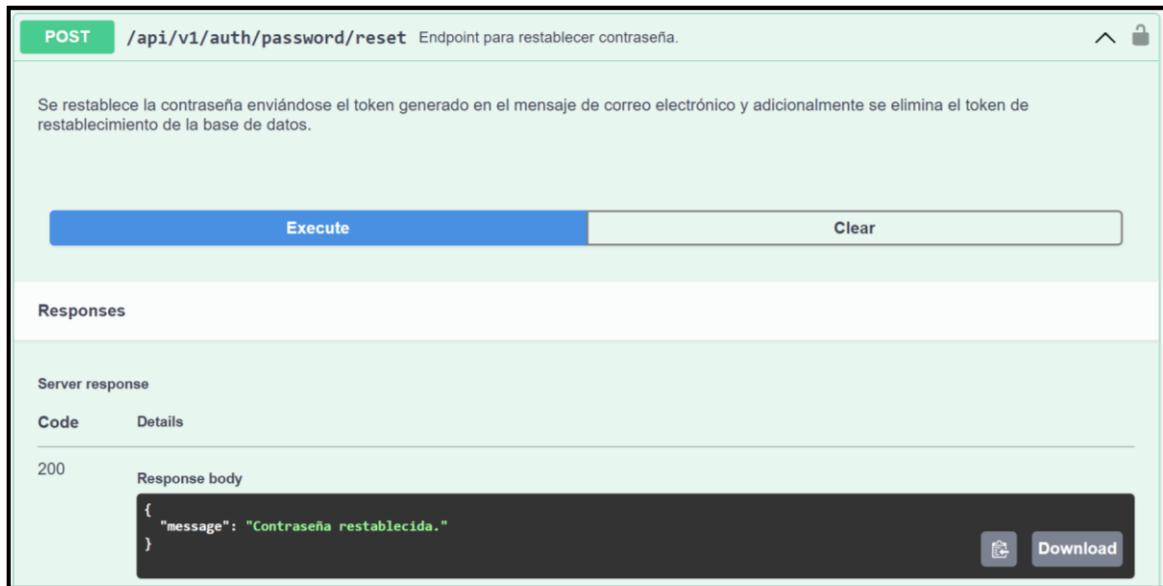


Fig. 8: *Endpoint* para el restablecimiento de contraseña.

```

AuthTest.resetPasswordShouldReturnOk ✘
Tests passed: 1 of 1 test – 5 sec 580 ms
user_name=?
where
id=?
2023-01-11 16:52:28.012 INFO 9968 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 16:52:28.023 INFO 9968 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 16:52:28.065 INFO 9968 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 1m 39s
4 actionable tasks: 2 executed, 2 up-to-date
16:52:28: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AuthTest.resetPasswordSh

```

Fig. 9: Prueba unitaria del *endpoint* para el restablecimiento de contraseña.

Generar varios *endpoints* que permiten visualizar y modificar el perfil de usuario

Para la interacción del usuario con perfil administrador, administrativo, profesor y pasante con su respectivo perfil de usuario se implementan varios *endpoints* con sus respectivas rutas públicas. En este sentido, se implementa un *endpoint* con ruta privada de tipo *GET* para obtener todos los datos del usuario que se ha autenticado previamente y un *endpoint* privado de tipo *POST* para la actualización de los datos como se muestra en la **Fig. 10** y el resultado de la prueba unitaria respectiva en la **Fig. 11**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra en el **ANEXO II** de este documento.

Fig. 10: *Endpoint* para actualizar el perfil de usuario.

```

ProfileTest.updateProfileInfoShouldReturnOk ✘
Tests passed: 1 of 1 test – 6 sec 378 ms
avatar0_.name as name3_0_0_,
avatar0_.path as path4_0_0_,
avatar0_.type as type5_0_0_,
avatar0_.updated_at as updated_6_0_0_
from
    avatar avatar0_
where
    avatar0_.id=?
2023-01-11 20:49:57.873  INFO 7744 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean :
2023-01-11 20:49:57.878  INFO 7744 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      :
2023-01-11 20:49:57.919  INFO 7744 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource      :
BUILD SUCCESSFUL in 3m 33s
4 actionable tasks: 2 executed, 2 up-to-date
20:49:59: Execution finished ':test --tests "esfot.thesis.botics.endpoints.ProfileTest.updateProfile"

```

Fig. 11: Prueba unitaria del *endpoint* para actualizar el perfil de usuario.

Generar varios *endpoints* que permiten gestionar pasantes

Para la gestión de pasantes de laboratorios se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrador gestionar pasantes de forma íntegra. En ese sentido, se implementa un *endpoint* privado de tipo *POST* para registrar a un usuario con perfil pasante, dos *endpoints* con rutas públicas de tipo *GET* uno para obtener el registro de un pasante por su nombre y otro para obtener la información de todos los pasantes que se han registrado y dos *endpoints* con ruta privada de tipo *GET* uno para deshabilitar la cuenta de un usuario pasante y otra para habilitarla ambas mediante el

id como se muestra en la **Fig. 12** y el resultado de la prueba unitaria respectiva en la **Fig. 13**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada campo se muestra el **ANEXO II**.



Fig. 12: *Endpoint* para habilitar una cuenta de usuario con perfil pasante.

```
AdminTest.enableInternShouldReturnOk ✘
Tests passed: 1 of 1 test - 15 sec 194 ms
  avatar0_.path as path4_0_0_,
  avatar0_.type as type5_0_0_,
  avatar0_.updated_at as updated_6_0_0_
from
  avatar avatar0_
where
  avatar0_.id=?
2023-01-11 21:03:43.764  INFO 13216 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 21:03:43.769  INFO 13216 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 21:03:44.096  INFO 13216 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 2m 12s
4 actionable tasks: 2 executed, 2 up-to-date
21:03:44: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AdminTest.enableInternSho
```

Fig. 13: Prueba unitaria del *endpoint* para habilitar una cuenta de usuario con perfil pasante.

Generar varios *endpoints* que permiten gestionar inventarios

Para la gestión de inventarios de los laboratorios se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrador gestionar inventarios de forma segura. En ese sentido, para la implementación de los *endpoints* se divide en dos grupos, laboratorios y computadoras. Para el grupo de laboratorios se implementan dos *endpoints* con rutas privadas de tipo *GET* uno para obtener el registro de un laboratorio por

su nombre y otro para obtener la información de todos los laboratorios que se han registrado. Mientras que para el grupo de computadoras se implementan varios *endpoints* con rutas privadas de tipo *GET* uno para acceder y obtener el registro de una computadora por su nombre de host y otro para obtener los datos de cada una de las computadoras que han sido registradas, un *endpoint* con ruta privada de tipo *PUT* para asignar una computadora a un laboratorio, un *endpoint* con ruta privada de tipo *POST* para registrar una computadora mediante un formulario y un *endpoint* con ruta privada de tipo *DELETE* para eliminar una computadora mediante el id como se muestra en la **Fig. 14** y el resultado de la prueba unitaria respectiva en la **Fig. 15**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

The screenshot shows a REST API endpoint for deleting a computer. The URL is `/api/v1/admin/computer/delete/{hostName}`. The description is "Endpoint para eliminar una computadora por su nombre de host." Below the URL, there is a note: "Elimina una computadora según su nombre de host." Under the "Responses" section, there is a "Server response" table with two columns: "Code" and "Details". The "Code" column has a single entry "200". The "Details" column contains "Response body" and "Response headers". The "Response body" is a JSON object:

```
{ "message": "Computadora borrada." }
```

 It includes "Copy" and "Download" buttons. The "Response headers" section lists: cache-control: no-cache,no-store,max-age=0,must-revalidate; connection: keep-alive; content-type: application/json; date: Thu,12 Jan 2023 02:07:46 GMT; expires: 0.

Fig. 14: *Endpoint* para eliminar una computadora.

```

AdminTest.deleteComputerShouldReturnOk ✘
Tests passed: 1 of 1 test – 9 sec 828 ms
  where
    computer0_.host_name=?
Hibernate:
  delete
  from
    computers
  where
    id=?
2023-01-11 21:21:00.650  INFO 8512 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 21:21:00.656  INFO 8512 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 21:21:00.705  INFO 8512 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 1m 35s
4 actionable tasks: 2 executed, 2 up-to-date
21:21:01: Execution finished ':test --tests "esfot.tesis.botics.endpoints.AdminTest.deleteComputer"

```

Fig. 15: Prueba unitaria del *endpoint* para eliminar una computadora.

Generar varios *endpoints* que permiten visualizar reporte de inventarios

Para la visualización de reportes de inventarios de laboratorios se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrador generar reportes de inventarios. En ese sentido, se implementa varios *endpoints* con rutas privadas de tipo GET uno para generar un reporte de inventario de laboratorios y otro para el historial de movimientos de las computadoras como se muestra en la **Fig. 16** y el resultado de la prueba unitaria respectiva en la **Fig. 17**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

GET /api/v1/admin/inventory/history/index Endpoint para listar el historial de movimiento de computadoras.

Retorna un arreglo de asignaciones, desasignaciones y cambios de computadoras entre laboratorios para generar el reporte en PDF.

Responses

Server response

Code	Details
200	Response body

```
[
  {
    "id": 5,
    "state": false,
    "changeDetails": "Prueba de cambio.",
    "labName": "LAB_SMD",
    "hostName": "1-21-01-E021-01",
    ...
  }
]
```

Fig. 16: *Endpoint* para generar historial de movimiento de computadoras.

```

AdminTest.indexHistoryShouldReturnOk ✘
Tests passed: 1 of 1 test - 1 min 21 sec
computer0_.serial_monitor as serial_17_2_0_,
computer0_.state as state18_2_0_,
computer0_.updated_at as updated19_2_0_
from
computers computer0_
where
computer0_.id=?
2023-01-11 21:45:46.407 INFO 12720 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 21:45:46.413 INFO 12720 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 21:45:46.438 INFO 12720 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 3m 9s
4 actionable tasks: 2 executed, 2 up-to-date
21:45:46: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AdminTest.indexHistorySho

```

Fig. 17: Prueba unitaria del *endpoint* para generar historial de movimiento de computadoras.

3.3 Sprint 2. Resultados del desarrollo de *endpoints* para el perfil personal administrativo

A través de lo establecido dentro del Sprint Backlog el presente Sprint presenta estas tareas:

- Generar *endpoints* que permiten solicitar *tickets* de asistencia.
- Generar *endpoints* que permiten gestionar comentarios y/o sugerencias.

Generar varios *endpoints* que permiten solicitar *tickets* de asistencia

Para la solicitud de tickets de asistencia se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrativo y profesor solicitar asistencia técnica. En ese sentido, se implementa un *endpoint* privado de tipo POST para registrar un ticket de asistencia mediante un formulario en el que se describen las razones de realizar dicha solicitud y un *endpoint* privado de tipo GET para obtener los registros de todos los tickets de asistencia que se han como se muestra en la **Fig. 18** y el resultado de la prueba unitaria respectiva en la **Fig. 19**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de campos se presenta en el **ANEXO II**.

GET /api/v1/administrative/ticket/index/{idUser} Endpoint para listar los tickets de asistencia registrados por el usuario especificado.

Retorna un arreglo de los tickets de asistencia realizados por el usuario especificado por su id, cual ha sido su respuesta y que usuario dio dicha respuesta.

Responses

Server response

Code	Details
200	Response body <pre>[{ "id": 12, "subject": "Computador sin internet lab 14.", "description": "La computadora 6 del laboratorio 14 no tiene conexión a internet, por favor realizar una revisión.", "state": true, "createdAt": null, "updatedAt": null, "firstName": "Carlos", "lastName": "Fuentes", "email": "adminis@epn.edu.ec", "status": "Pendiente" }]</pre>

Fig. 18: Endpoint para obtener *tickets* de asistencia.

```
AdministrativeTest.indexTicketsShouldReturnOk ✘
Tests passed: 1 of 1 test - 10 sec 713 ms
user_roles roles0_
inner join
  roles role1_
    on roles0_.role_id=role1_.id
where
  roles0_.user_id=?
2023-01-11 22:15:20.822 INFO 4620 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 22:15:20.827 INFO 4620 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 22:15:21.178 INFO 4620 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 2m 29s
4 actionable tasks: 2 executed, 2 up-to-date
22:15:23: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AdministrativeTest.index'
```

Fig. 19: Prueba unitaria del *endpoint* para obtener tickets de asistencia.

Generar varios *endpoints* que permiten gestionar comentarios y/o sugerencias

Para la gestión de comentarios y/o sugerencias se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrativo dar respuesta a comentario y/o sugerencia. En ese sentido, se implementa un *endpoint* privado de tipo POST para responder a un comentario y/o sugerencia y un endpoint de tipo GET para obtener los registros de todos los comentarios y/o sugerencias como se muestra en la **Fig. 20** y el resultado de la prueba unitaria respectiva en la **Fig. 21**. Adicionalmente, para el

consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

The screenshot shows a Swagger UI interface for a REST API. At the top, there is a blue header bar with the method "GET" and the endpoint path "/api/v1/administrative/manage/commentary/index/{idUser}". To the right of the path, there is a small lock icon and the text "Endpoint para listar los comentarios no registrados por el usuario especificado." Below the header, there is a brief description: "Retorna un arreglo de comentarios no realizados por el usuario especificado por su id, cual ha sido la respuesta a cada uno y que usuario dio dicha respuesta." Under the description, there is a section titled "Responses" which includes a "Server response" table with columns "Code" and "Details". A row for code 200 is selected, showing the "Response body" as a JSON array. The first item in the array is a JSON object with fields: "id": 19, "subject": "Comentario de prueba de profesor 1.", "message": "Comentario de prueba con descripción para profe 1.", "state": true, "createdAt": null, "updatedAt": null, "firstName": null, "lastName": null, "email": "profesor@epn.edu.ec".

Fig. 20: Endpoint para administrar comentarios y/o sugerencias.

The screenshot shows a terminal window with the title "AdministrativeTest.manageCommentaryShouldReturnOk". It displays the results of a test: "Tests passed: 1 of 1 test – 13 sec 82 ms". Below the test results, there is a SQL query for selecting roles based on user_id. The log output shows several INFO messages from the application server and database. At the bottom, it shows the build status: "BUILD SUCCESSFUL in 2m 17s" and the command executed: "22:24:38: Execution finished ':test --tests \"esfot.thesis.botics.endpoints.AdministrativeTest.manageCommentaryShouldReturnOk\"'".

Fig. 21: Prueba unitaria del endpoint para administrar comentarios y/o sugerencias.

3.4 Sprint 3. Resultados del desarrollo de endpoints para el perfil profesor

A través de lo establecido dentro del *Sprint Backlog* el presente *Sprint* presenta la siguiente tarea que es generar *endpoints* que permiten registrar comentarios y/o sugerencias.

Generar varios *endpoints* que permiten registrar comentarios y/o sugerencias

Para el registro de comentarios y/o sugerencias se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil administrativo y profesor registrar sus comentarios y/o sugerencias. En ese sentido, se implementa un *endpoint* con ruta privada de tipo POST para registrar un comentario y/o sugerencia mediante un formulario en el que se detalla la inquietud encontrada y un *endpoint* con ruta privada de tipo GET para obtener los registros de todos los comentarios y/o sugerencias que se han registrado por parte de los usuarios como se muestra en la **Fig. 22** y el resultado de la prueba unitaria respectiva en la **Fig. 23**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

The screenshot shows a Swagger UI interface for a REST API. The endpoint is `/api/v1/teacher/commentary/index/{idUser}`. The description states: "Endpoint para listar los comentarios registrados por el usuario especificado." Below the description, it says: "Retorna un arreglo de comentarios realizados por el usuario especificado por su id, cual ha sido su respuesta y que usuario dio dicha respuesta." Under the "Responses" section, there is a "Server response" table with two rows: one for code 200 and one for code 404. The 200 row has a "Response body" example:

```
[{"id": 19, "subject": "Comentario de prueba de profesor 1.", "message": "Comentario de prueba con descripción para profe 1.", "state": true, "response": {"id": 26, "subject": "Respuesta a Comentario de prueba de profe 1.", "details": "Se ha respondido al comentario de prueba de profe 1 con éxito."}}
```

Fig. 22: *Endpoint* para obtener comentarios y/o sugerencias.

```
TeacherTest.indexCommentaryShouldReturnOk ✘
Tests passed: 1 of 1 test - 10 sec 896 ms
  reserves reservez_
    on responsel_.id=reserve2_.response_id
  left outer join
    tickets ticket3_
      on responsel_.id=ticket3_.response_id
  where
    commentary0_.response_id=?
2023-01-11 22:38:24.088  INFO 7592 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 22:38:24.093  INFO 7592 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 22:38:24.228  INFO 7592 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 2m 6s
4 actionable tasks: 2 executed, 2 up-to-date
22:38:25: Execution finished ':test --tests "esfot.thesis.botics.endpoints.TeacherTest.indexCommentaryShouldReturnOk"'
```

Fig. 23: Prueba unitaria del *endpoint* para obtener comentarios y/o sugerencias.

3.5 Sprint 4. Resultados del desarrollo de *endpoints* para el perfil pasante

A través de lo establecido dentro del Sprint Backlog el presente Sprint presenta estas tareas:

- Generar *endpoints* que permiten gestionar *tickets* de asistencia.
- Generar *endpoints* que permiten gestionar reservas.

Generar varios *endpoints* que permiten gestionar *tickets* de asistencia

Para la gestión de tickets de asistencia se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil pasante atender a los tickets de asistencia. En ese sentido, se implementa un *endpoint* privado de tipo POST para responder a un ticket de asistencia y un *endpoint* con ruta privada de tipo GET para obtener los registros de todos los tickets de asistencia como se muestra en la **Fig. 24** y el resultado de la prueba unitaria respectiva en la **Fig. 25**. A su vez, el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

The screenshot shows a Swagger UI interface for a RESTful API. At the top, there is a blue header bar with the method "GET" and the endpoint "/api/v1/intern/manage/ticket/index". Below the header, the description reads: "Endpoint para listar los tickets de asistencia registrados." To the right of the header, there are icons for copy, lock, and refresh.

The main content area has a light gray background. It contains a section titled "Responses" with a sub-section "Server response". Under "Server response", there are two tabs: "Code" and "Details", with "Code" selected. The "Code" tab shows the HTTP status code "200" and the label "Response body". Below these, there is a JSON snippet representing a list of tickets:

```
[  
  {  
    "id": 12,  
    "subject": "Computador sin internet lab 14.",  
    "description": "La computadora 6 del laboratorio 14 no tiene conexión a internet, por favor realizar una revisión.",  
    "state": true,  
    "createdAt": null,  
    "updatedAt": null,  
    "firstName": "Carlos",  
    "lastName": "Fuentes",  
    "email": "adminis@pn.edu.ec",  
    "role": "ROLE_ADMINISTRATIVO",  
    "user": {  
      "id": 1,  
      "username": "adminis@pn.edu.ec",  
      "password": "adminis123",  
      "name": "Administrador",  
      "email": "adminis@pn.edu.ec",  
      "role": "ROLE_ADMINISTRATIVO"  
    }  
  }  
]
```

Fig. 24: Endpoint para administrar *tickets* de asistencia.

```

InternTest.manageTicketsShouldReturnOk ✘
Tests passed: 1 of 1 test - 17 sec 986 ms

inner join
    roles role1_
        on roles0_.role_id=role1_.id
where
    roles0_.user_id=?
2023-01-11 22:51:28.504  INFO 7556 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 22:51:28.509  INFO 7556 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 22:51:28.544  INFO 7556 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 2m 17s
4 actionable tasks: 2 executed, 2 up-to-date
22:51:30: Execution finished ':test --tests "esfot.thesis.botics.endpoints.InternTest.manageTickets"

```

Fig. 25: Prueba unitaria del *endpoint* para administrar *tickets* de asistencia.

Generar varios *endpoints* que permiten gestionar reservas

Para la gestión de reservas se implementan varios *endpoints* con sus respectivas rutas que permiten al usuario con perfil pasante dar respuesta a comentario y/o sugerencia. En ese sentido, se implementa un *endpoint* privado de tipo POST para responder a una reserva y un *endpoint* de tipo GET para obtener los registros de todas las reservas como se muestra en la **Fig. 26** y el resultado de la prueba unitaria respectiva en la **Fig. 27**. Adicionalmente, para el consumo y almacenamiento de registros conjuntamente con las validaciones de cada uno de los campos se muestra el **ANEXO II**.

GET /api/v1/intern/manage/reserve/index Endpoint para listar las reservas registradas.

Retorna un arreglo de reservas con el detalle de que usuario las realizó, cual ha sido su respuesta y que usuario dio dicha respuesta.

Responses

Server response

Code	Details
200	Response body

```
[
  {
    "id": 18,
    "labName": "Reserva de prueba de profe 1.",
    "description": "Reserva de prueba de profe 1 de descripción.",
    "state": true,
    "createdAt": null,
    "updatedAt": null,
    "firstName": null,
    "lastName": null,
    "email": "profesor@epn.edu.ec",
    "role": "ROLE_PROFESOR",
    "responses": [
      {
        "user": "profesor@epn.edu.ec",
        "comment": "Reserva de prueba de profe 1",
        "date": null
      }
    ]
  }
]
```

Fig. 26: *Endpoint* para administrar reservas.

```

InternTest.manageReservesShouldReturnOk ✘
Tests passed: 1 of 1 test – 12 sec 106 ms
from
    user_roles roles0_
inner join
    roles role1_
        on roles0_.role_id=role1_.id
where
    roles0_.user_id=?
2023-01-11 23:00:00.328 INFO 8684 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 23:00:00.335 INFO 8684 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 23:00:00.418 INFO 8684 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 2m 11s
4 actionable tasks: 2 executed, 2 up-to-date
23:00:00: Execution finished ':test --tests "esfot.tesis.botics.endpoints.InternTest.manageReserve'

```

Fig. 27: Prueba unitaria del *endpoint* para administrar reservas.

3.6 Sprint 5. Pruebas y Despliegue del *backend*

A través de lo establecido dentro del *Sprint Backlog* el presente *Sprint* presenta estas tareas:

- Aplicación de pruebas unitarias.
- Aplicación de pruebas de compatibilidad.
- Aplicación de pruebas estrés.
- Despliegue del *backend* en *Azure*.

Aplicación de pruebas unitarias

Con la finalización de la codificación de cada uno de los módulos en este apartado se realizan las pruebas unitarias para el *backend*, este tipo de pruebas permiten el *testing* de funcionalidades específicas de un producto *software* de tal forma que se agilizan las actividades de detección y corrección de errores. Siendo así, se implementan las herramientas JUnit y Mockito que facilitan la aplicación de pruebas unitarias para proyectos con Java [40].

Con ello, en la **Fig. 28:** Prueba unitaria para el registro de una computadora. se evidencia una sección del código empleado registrar una computadora, adicionalmente, la **Fig. 29** presenta el resultado detallado de la implementación de dicha prueba. Por otro lado, todas las pruebas faltantes y sus respectivos resultados se muestran en el **ANEXO II**.

```

    @Test
    public void saveComputerShouldReturnOk() throws Exception {
        String request = "{" +
            "\"hostName\": \"1-21-01-E022-02\", " +
            "\"serialMonitor\": \"AU2426\", " +
            "\"serialKeyboard\": \"FXLM78J\", " +
            "\"serialCpu\": \"FXLM78JL\", " +
            "\"codeMonitor\": \"887195045\", " +
            "\"codeKeyboard\": \"887195045\", " +
            "\"codeCpu\": \"887195045\", " +
            "\"model\": \"HP optiplex 9020\", " +
            "\"hardDrive\": \"1TB\", " +
            "\"ram\": \"16GB\", " +
            "\"processor\": \"i 7700\", " +
            "\"operativeSystem\": \"Windows 7\", " +
            "\"details\": \"Computador nuevo.\", " +
            "\"observations\": \"Actualización de drivers.\", " +
            "\"labReference\": " + 0 +
            "}";
        this.mockMvc.perform( requestBuilder: post( urlTemplate: uri + "computer/save").contentType( "application/json") ).andExpect( status:isOk() ).andReturn();
    }
}

```

Fig. 28: Prueba unitaria para el registro de una computadora.

```

AdminTest.saveComputerShouldReturnOk ✘
Tests passed: 1 of 1 test - 14 sec 56 ms
insert
into
    computers
    (code_cpu, code_keyboard, code_monitor, created_at, details, hard_drive, host_name, lab_ref
values
    (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
2023-01-11 23:08:58.541  INFO 13108 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 23:08:58.547  INFO 13108 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 23:08:58.578  INFO 13108 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 1m 32s
4 actionable tasks: 2 executed, 2 up-to-date
23:08:59: Execution finished ':test --tests "esfot.thesis.botics.endpoints.AdminTest.saveComputerShouldReturnOk"'

```

Fig. 29: Resultado de la prueba unitaria para el registro de una computadora.

Con el resultado de la prueba, se evidencia el correcto funcionamiento de los módulos y se determina que se han aplicado todas las validaciones necesarias.

Aplicación de pruebas de compatibilidad

Con estas pruebas se verifica que se dé una correcta presentación de la información y contenido multimedia de un producto software en varios dispositivos y clientes HTTP de manera independiente [41]. En ese sentido, para la ejecución de la prueba de compatibilidad se ha empleado los siguientes clientes HTTP Swagger y ThunderClient

dando como resultado una correcta presentación del contenido respectivo a cada endpoint que se ha implementado.

Con ello, la **TABLA VIII** presenta el detalle de los clientes HTTP empleados descritos anteriormente y los resultados de dichas pruebas se las muestra en el **ANEXO II**.

TABLA VIII: Clientes utilizados para las pruebas de compatibilidad.

NOMBRE	VERSIÓN
Swagger	V 3.0.1
Thunder Client	V 1.10.0

Con el resultado de la prueba se evidencia el correcto funcionamiento de los *endpoints* tanto en el tiempo de respuesta como en la visualización de la información.

Aplicación de pruebas de estrés

Con estas pruebas se verifica el comportamiento que tiene una API cuando es sometida a un gran número de peticiones concurrentes de tal forma que se busca estresar al sistema para conocer cuáles son sus tiempos de respuesta, latencia y otros parámetros que brindan información útil [42].

Con ello, la **Fig. 30** presenta a detalle una prueba de estrés y su respectivo resultado aplicando con la herramienta Apache JMeter, por otro lado, el detalle de cada prueba se lo puede observar en el **ANEXO II** de este documento.

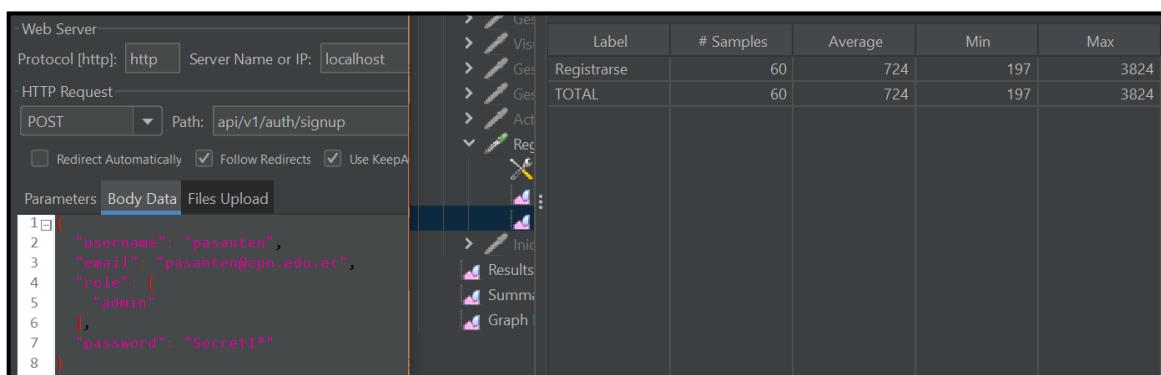


Fig. 30: Prueba de estrés para la historia de usuario N°1

Con el resultado de la prueba se establece que cada uno de los usuarios se encuentran conformes con las respuestas a las peticiones de cada una de las funcionalidades del

backend de tal forma que el mismo ha sido aprobado por el *Product Owner* de tal forma que se puede proseguir con la etapa posterior.

Despliegue del *backend* en Azure

Con la finalización de la codificación de cada uno de los módulos, mediante lo establecido en esta sección se despliega el *backend* a producción en *Azure* desde una imagen de docker del *backend* como se evidencia en la **Fig. 31**.

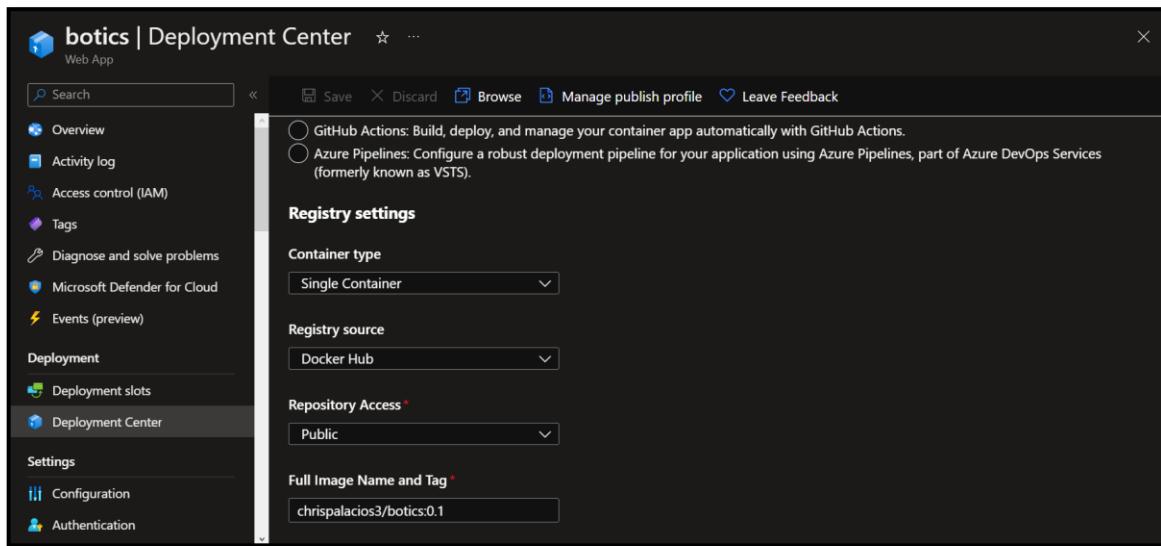


Fig. 31: Despliegue de imagen de Docker en Azure.

El detalle completo del proceso para el despliegue del *backend* se lo puede encontrar en el **ANEXO IV** siendo así el acceso al *backend* se lo puede realizar ingresando accediendo a la siguiente url desde un navegador:

<https://botics.azurewebsites.net/swagger-ui/index.html>

4 CONCLUSIONES

Esta parte del documento presenta cada una de las conclusiones a las que se han llegado durante el desarrollo del *backend* en el presente Trabajo de Integración Curricular:

- Con la correcta recopilación de requerimientos realizada al iniciar el proyecto, se ha logrado la implementación de los distintos *endpoints* del *backend* de tal forma que se ha obtenido una API RESTful con una estructura correctamente definida la cual habilita su escalabilidad para la implementación de nuevas funcionalidades a futuro.
- Con la implementación del modelo entidad relación se ha desarrollado la base de datos relacional SQL de tal manera que se dé un correcto manejo de la información, se obtenga un rápido acceso a los registros y se mantenga la integridad de los datos.
- Con la aplicación del patrón arquitectónico Modelo-Vista-Controlador se ha logrado estructurar el *backend* permitiendo la mantenibilidad del mismo con bloques de código que separan las distintas funcionalidades y una estructura de archivos altamente sencilla.
- Las ejecuciones de las distintas pruebas han permitido determinar con gran certeza que el *backend* cumple con las funcionalidades que se han establecido para el mismo.
- Gracias al uso de servicios PaaS el *backend* se encuentra disponible a través de internet mediante https otorgando una capa de seguridad adicional para cada usuario.

5 RECOMENDACIONES

Esta parte del documento se presenta cada una de las recomendaciones a las que se han llegado durante el desarrollo del *backend* en el presente trabajo de integración curricular:

- Es de gran importancia definir los comandos de docker necesarios para poder generar la imagen del *backend* esto en un dockerfile en la carpeta raíz del proyecto.
- Para brindar seguridad al *backend* es importante que se use el sistema dentro de la red de la institución o redes externas privadas.
- No compartir las credenciales las credenciales de acceso de administrador ya que con ellas un usuario mal intencionado puede comprometer los datos del sistema almacenados en la base de datos.
- Realizar respaldos de la base de datos periódicamente es necesario para no perder la información si ocurren ataques o problemas en el servicio de almacenamiento.

6 REFERENCIAS BIBLIOGRÁFICAS.

- [1] Mecalux.es. (2022, 31 marzo). Sistema de inventario: métodos para controlar el stock del almacén. Recuperado 20 de octubre de 2022, de <https://www.mecalux.es/blog/sistema-de-inventario>
- [2] Molina, M. (2015). IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE INVENTARIOS Y MANTENIMIENTO DE EQUIPOS INFORMÁTICOS MEDIANTE LA METODOLOGÍA SCRUM, EN LOS LABORATORIOS DE LA CARRERA DE INGENIERÍA EN INFORMÁTICA Y SISTEMAS COMPUTACIONALES DE LA UNIVERSIDAD TÉCNICA DE COTOPAXI DURANTE EL PERÍODO 2014-2015 (tesis de pregrado). Universidad técnica de Cotopaxi, Latacunga, Ecuador.
- [3] Fernández, B. F., & Sumoza, G. O. (2015). ¿ Por qué los sistemas de información son esenciales. Facultad de Ciencias Económicas y Sociales.
- [4] González Frutos, C. J. (2018). Sistema de Información para el Control de Equipos de Cómputo Utilizando la Técnica Benchmark para el Ministerio de Educación (Coordinación Zonal 3) (Bachelor's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Carrera de Ingeniería en Sistemas Computacionales e Informáticos).
- [5] Parrish, F. (2007). Front end/back end: the importance of communication. *Dermatology Nursing*, 19(4), 379-380.
- [6] Chiluisa Pallo, A. P., & Loarte Cajamarca, B. G. (2014). Desarrollo e implantación del sistema de control de inventarios y gestión de laboratorios para la de la facultad de Ciencias (Bachelor's thesis, Quito: EPN, 2014.).
- [7] Pressman. (2022). Ingeniería De Software (7.a ed.). MCGRaw HILL EDDUCATION.
- [8] Muñoz, C. C., Velthuis, M. G. P., & de la Rubia, M. Á. M. (2010). Calidad del producto y proceso software. Editorial Ra-Ma.
- [9] Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al Backend. In XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja).
- [10] Cabello, M. V. N. (2010). Introducción a las bases de datos relacionales. Vision Libros.
- [11] de Bogotá, C. D. C. (2019). El mundo conectado por las API.
- [12] Groussard, T. (2012). JAVA 7: Los fundamentos del lenguaje Java. Ediciones Eni.
- [13] González, G. M. (2016). Aprende a Desarrollar con Spring Framework: 2^a Edición. IT Campus Academy.

- [14] Haro, E., Guarda, T., Peñaherrera, A. O. Z., & Quiña, G. N. (2019). Desarrollo backend para aplicaciones web, servicios web restful: Node. js vs spring boot. Revista Ibérica de Sistemas e Tecnologias de Informação, (E17), 309-321.
- [15] Hernández Yeja, A., & Porven Rubier, J. (2016). Procedimiento para la seguridad del proceso de despliegue de aplicaciones web. Revista Cubana de Ciencias Informáticas, 10(2), 42-56.
- [16] Murillo, F. J., Payeta, A. M., Martín, I. M., Lara, A. J., Gutiérrez, R. C., Sánchez, J. C. S., & Moreno, R. V. (2013). Estudio de casos.
- [17] Maida, E. G., & Pacienza, J. (2015). Metodologías de desarrollo de software.
- [18] Montero, B. M., Cevallos, H. V., & Cuesta, J. D. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. Espirales revista multidisciplinaria de investigación, 2(17), 114-121.
- [19] Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). Información básica de SCRUM. California: Scrum Training Institute.
- [20] Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The role of the product owner in scrum-comparison between theory and practices. Procedia-Social and Behavioral Sciences, 119, 257-267.
- [21] Gonçalves, L. (2021, 28 noviembre). ¿Qué es un Scrum Master, un resumen para los líderes! ADAPT METHODOLOGY®. <https://adaptmethodology.com/es/que-es-un-scrum-master/>
- [22] Sachdeva, S. (2016). Scrum Methodology. Internationa Journal Of Engineering and Computer Science, URL: https://www.academia.edu/26010951/Scrum_Methodology (2.9. 2019).
- [23] Schwaber, K., & Sutherland, J. (2011). The scrum guide. Scrum Alliance, 21(19), 1.
- [24] Gudiño Cañar, E. M., & Páez Mora, A. S. (2018). Diseño y desarrollo de una aplicación móvil aplicando la metodología SCRUM, que permita el reconocimiento de cantos de ranas. Caso de estudio: anuros de los Andes del Ecuador (Bachelor's thesis, PUCE-Quito).
- [25] Durán Rodríguez, M. J. (2020). Modelo Integral de Atributos Para la Estimación de Historias de Usuario en Scrum.
- [26] Sedano, T., Ralph, P., & Péreira, C. (2019, May). The product backlog. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 200-211). IEEE.
- [27] Fowler, F. M. (2019). The sprint backlog. In Navigating Hybrid Scrum Environments (pp. 67-70). Apress, Berkeley, CA.
- [28] Sagredo, J. G. C., Espinosa, A. T., Reyes, M. M., & García, M. D. L. L. (2012). Automatización de la codificación del patrón modelo vista controlador (MVC) en

proyectos orientados a la Web. CIENCIA ergo-sum, Revista Científica Multidisciplinaria de Prospectiva, 19(3), 239-250.

- [29] Caiza, J. C., Guamán, D. S., & López, G. R. (2015). Herramientas de desarrollo con soporte colaborativo en Ingeniería de Software. Enfoque UTE, 6(2), 102-116.
- [30] Walls, C. (2015). Spring Boot in action. Simon and Schuster.
- [31] Santillán, L. A. C., Ginestà, M. G., & Mora, Ó. P. (2014). Bases de datos en MySQL. Universitat oberta de Catalunya.
- [32] Anderson, C. (2015). Docker [software engineering]. Ieee Software, 32(3), 102-c3.
- [33] van Leijenhorst, T., Chin, K. W., & Lowe, D. (2008). On the viability and performance of DNS tunneling.
- [34] Davis, A. L. (2020). Spring data. In Spring Quick Reference Guide (pp. 43-59). Apress, Berkeley, CA.
- [35] Nascimento, A. E. (2017). OAuth 2.0 Cookbook: Protect Your Web Applications Using Spring Security. Packt Publishing Ltd.
- [36] G. Grefory, C. Bauer y G. King, Java Persistence with Hibernate, Manning, 2015.
- [37] Scarioni, C. (2013). Pro Spring Security. Apress.
- [38] Pereira, A. L., Raoufi, M., & Frost, J. C. (2010, July). Using MySQL and JDBC in new teaching methods for undergraduate database systems courses. In International Conference on Data Engineering and Management (pp. 245-248). Springer, Berlin, Heidelberg.
- [39] GeeksforGeeks. (2022, 28 marzo). Spring Boot - Sending Email via SMTP. <https://www.geeksforgeeks.org/spring-boot-sending-email-via-smtp/>
- [40] Rivero, A. (2019, 9 diciembre). Introducción a las Pruebas Unitarias (Unit Testing) con JUnit y Mockito. Java desde 0. <https://javadesde0.com/introduccion-a-las-pruebas-unitarias-con-junit-y-mockito/>
- [41] https://scholar.google.com.ec/scholar?hl=es&as_sdt=0%2C5&q=Desarrollo+de+un+backend+para+la+gesti%C3%B3n+del+sistema+penitenciario+del+Ecuador&btnG=
- [42] https://scholar.google.com.ec/scholar?hl=es&as_sdt=0%2C5&q=Desarrollo+de+una+aplicaci%C3%B3n+web+y+m%C3%B3vil+en+tiempo+real%2C+una+evolucion%C3%B3n+de+las+aplicaciones+actuales&btnG=

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 anti plagio *Turnitin*.
- **ANEXO II.** Manual de Usuario.
- **ANEXO III.** Manual de Instalación.
- **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.



CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 09 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 PARA LA GESTIÓN DE LOS LABORATORIOS DE INFORMÁTICA - ESFOT elaborado por el estudiante Christian Abrahan Palacios Padilla 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,

A handwritten signature in blue ink, appearing to read "B. Loarte".

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.

RECOPIACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	ID - RR	ENUNCIADO DEL ÍTEM
<i>Backend</i>	RR001	Como usuario profesor y administrativo necesitan generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Registrarse
	RR002	Como usuario administrador, pasante, profesor y administrativo necesitan generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Iniciar sesión• Cerrar sesión• Modificar contraseña
	RR003	Como usuario administrador, pasante, profesor y administrativo necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Modificar perfil de usuario
	RR004	Como usuario administrador necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Gestionar pasantes
	RR005	Como usuario administrador necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Gestionar inventarios
	RR006	Como usuario administrador y pasante necesitan generar varios <i>endpoints</i> para: <ul style="list-style-type: none">• Visualizar reporte de inventarios

	RR007	Como usuario administrativo necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">● Gestionar comentarios y/o sugerencias
	RR010	Como usuario profesor y administrativo necesitan generar varios <i>endpoints</i> para: <ul style="list-style-type: none">● Solicitar tickets de asistencia
	RR011	Como usuario profesor necesita generar varios <i>endpoints</i> para: <ul style="list-style-type: none">● Solicitar reservas
	RR012	Como usuario profesor y administrativo necesitan generar varios <i>endpoints</i> para: <ul style="list-style-type: none">● Enviar comentarios y/o sugerencias

Historias de Usuario

Una vez que se ha realizado y completado el proceso de recopilación de requerimientos, a continuación, se implementan las historias de usuario del *backend*. Con ello se muestran 11 de las historias de usuario en base a la recopilación de requerimientos las cuales representan desde la **TABLA X** que corresponde a la historia de usuario 2 hasta la **TABLA XX** que corresponde a la historia de usuario 12.

TABLA X: Historia de usuario 2 - Iniciar sesión, cerrar sesión y modificar contraseña.

HISTORIA DE USUARIO	
Identificador: HU002	Usuario: Administrador, pasante, profesor y administrativo
Nombre historia: Iniciar sesión, cerrar sesión y modificar contraseña.	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Iteración asignada: 1	
Responsable (es): Christian Palacios	

Descripción: El *backend* permite generar varios *endpoints* para que el perfil profesor, pasante y administrativo puedan:

- Iniciar sesión.
- Cerrar sesión.
- Modificar contraseña.

Observación: Los usuarios con perfil administrador, pasante, profesor y administrativo necesitan estar registrados para acceder a los *endpoints* asignados. Además, para modificar la contraseña el *backend* envía un correo al e-mail del usuario.

TABLA XI: Historia de usuario 3 - Modificar perfil de usuario.

HISTORIA DE USUARIO	
Identificador: HU003	Usuario: Administrador, pasante, profesor y administrativo
Nombre historia: Modificar perfil de usuario.	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Iteración asignada: 1	
Responsable (es): Christian Palacios	
<p>Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil profesor, pasante y administrativo puedan visualizar y editar su perfil por medio de los siguientes campos:</p> <ul style="list-style-type: none"> ● Nombres ● Apellidos ● Avatar ● Extensión 	
<p>Observación: Los usuarios con perfil administrador, pasante, profesor y administrativo son los únicos que pueden acceder a los <i>endpoints</i> mencionados anteriormente, es decir, necesitan iniciar sesión para modificar su perfil respectivo.</p>	

TABLA XII: Historia de usuario 4 - Gestionar pasantes.

HISTORIA DE USUARIO	
Identificador: HU004	Usuario: Administrador
Nombre historia: Gestionar pasantes	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Iteración asignada: 2	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil administrador pueda: <ul style="list-style-type: none"> • Registrar pasantes • Visualizar lista de pasantes • Habilitar y/o deshabilitar pasantes 	
Observación: El usuario con perfil administrador tiene a potestad de habilitar y/o deshabilitar pasantes según lo requiera.	

TABLA XIII: Historia de usuario 5 - Gestionar inventarios.

HISTORIA DE USUARIO	
Identificador: HU005	Usuario: Administrador
Nombre historia: Gestionar inventarios	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Iteración asignada: 2	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil administrador pueda: <ul style="list-style-type: none"> • Registrar inventarios • Modificar inventarios 	

- Visualizar inventarios
- Eliminar inventarios

Observación: El usuario con perfil administrador es el único que puede acceder a los *endpoints* anteriormente mencionados, es decir, necesita iniciar sesión para poder gestionar inventarios.

TABLA XIV: Historia de usuario 6 - Visualizar reporte de inventarios.

HISTORIA DE USUARIO	
Identificador: HU006	Usuario: Administrador
Nombre historia: Visualizar reporte de inventarios	
Prioridad en Negocio: Alta	Riesgo en desarrollo: Media
Iteración asignada: 2	
Responsable (es): Christian Palacios	
Descripción: El backend permite generar varios <i>endpoints</i> para que el perfil administrador pueda: <ul style="list-style-type: none"> • Visualizar reporte de inventario. 	
Observación: El usuario administrador es el único que puede acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesita iniciar sesión para poder gestionar inventarios.	

TABLA XV: Historia de usuario 7 - Gestionar comentarios y/o sugerencias.

HISTORIA DE USUARIO	
Identificador: HU007	Usuario: Administrativo
Nombre historia: Gestionar comentarios y/o sugerencias	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media

Iteración asignada: 3
Responsable (es): Christian Palacios
<p>Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil administrativo y pasante puedan:</p> <ul style="list-style-type: none"> • Visualizar comentarios y/o sugerencias. • Atender comentarios y/o sugerencias.
<p>Observación: Los usuarios administrativo y pasante son los únicos que pueden acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesitan iniciar sesión para poder gestionar comentarios y/o sugerencias.</p>

TABLA XVI: Historia de usuario 8 - Solicitar *tickets* de asistencia.

HISTORIA DE USUARIO	
Identificador: HU008	Usuario: Profesor y administrativo
Nombre historia: Solicitar <i>tickets</i> de asistencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Iteración asignada: 4	
Responsable (es): Christian Palacios	
<p>Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil profesor y administrativo puedan:</p> <ul style="list-style-type: none"> • Registrar solicitud de asistencia. 	
<p>Observación: Los usuarios profesor y administrativo son los únicos que pueden acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesitan iniciar sesión para poder solicitar <i>tickets</i> de asistencia.</p>	

TABLA XVII: Historia de usuario 9 - Enviar comentarios y/o sugerencias.

HISTORIA DE USUARIO	
Identificador: HU009	Usuario: Profesor y administrativo

Nombre historia: Enviar comentarios y/o sugerencias	
Prioridad en Negocio: Media	Riesgo en desarrollo: Media
Iteración asignada: 4	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil profesor y administrativo puedan: <ul style="list-style-type: none"> • Registrar comentarios y/o sugerencias. 	
Observación: Los usuarios profesor y administrativo son los únicos que pueden acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesitan iniciar sesión para poder enviar comentarios y/o sugerencias.	

TABLA XVIII: Historia de usuario 10 - Solicitar reservas.

HISTORIA DE USUARIO	
Identificador: HU0010	Usuario: Profesor
Nombre historia: Solicitar reservas	
Prioridad en Negocio: Alta	Riesgo en Negocio: Media
Iteración asignada: 4	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil profesor pueda: <ul style="list-style-type: none"> • Registrar solicitud de reservas. 	
Observación: El usuario profesor es el único que puede acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesitan iniciar sesión para poder solicitar reservas.	

TABLA XIX: Historia de usuario 11 - Gestionar tickets de asistencia.

HISTORIA DE USUARIO	
Identificador: HU0011	Usuario: Pasante
Nombre historia: Gestionar <i>tickets</i> de asistencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Iteración asignada: 5	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil pasante pueda: <ul style="list-style-type: none"> • Visualizar <i>tickets</i>. • Atender <i>tickets</i>. 	
Observación: El usuario pasante es el único que puede acceder a los <i>endpoints</i> anteriormente mencionados, es decir, necesita iniciar sesión para poder gestionar <i>tickets</i> de asistencia.	

TABLA XX: Historia de usuario 12 - Gestionar reservas.

HISTORIA DE USUARIO	
Identificador: HU0012	Usuario: Pasante
Nombre historia: Gestionar reservas	
Prioridad en negocio: Media	Riesgo en desarrollo: Media
Iteración asignada: 5	
Responsable (es): Christian Palacios	
Descripción: El <i>backend</i> permite generar varios <i>endpoints</i> para que el perfil pasante pueda: <ul style="list-style-type: none"> • Visualizar reservas. • Tramitar reservas. 	

Observación: El usuario pasante es el único que puede acceder a los *endpoints* anteriormente mencionados, es decir, necesita iniciar sesión para poder gestionar reservas.

Product Backlog

Este apartado presenta la definición completa del *Product Backlog* siendo así la **TABLA XXI** muestra cual es la prioridad que tiene cada historia de usuario en base a la importancia del mismo para el dueño del producto y su complejidad en el desarrollo.

TABLA XXI: *Product Backlog.*

ELABORACIÓN DEL PRODUCT BACKLOG				
ID – HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU001	Registrarse.	1	Finalizada	Media
HU002	Iniciar sesión, cerrar sesión y modificar contraseña.	1	Finalizada	Media
HU003	Modificar perfil de usuario.	1	Finalizada	Media
HU004	Gestionar pasantes.	2	Finalizada	Media
HU005	Gestionar inventarios.	2	Finalizada	Alta
HU006	Visualizar reporte de inventarios.	2	Finalizada	Alta

HU007	Gestionar comentarios y/o sugerencias	3	Finalizada	Alta
HU008	Solicitar <i>tickets</i> de asistencias.	3	Finalizada	Alta
HU011	Gestionar <i>tickets</i> de asistencia.	5	Finalizada	Alta
HU012	Gestionar reservas.	5	Finalizada	Media

Sprint Backlog

La **TABLA XXII** presenta los *Sprints* que se han desarrollado por parte del *backend*, describiendo las actividades y el tiempo para cumplirlos respectivamente establecidos por el dueño del producto.

TABLA XXII: *Sprint Backlog.*

ELABORACIÓN DEL SPRINT BACKLOG						
ID – SB	NOMBRE	MÓDULO	ID-HU	HISTORIAS DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB001	Diseño e implementación de <i>endpoints</i> para el registro, inicio de sesión, cierre de sesión y modificar contraseña.	Módulo de registro	HU001	Registrarse	<ul style="list-style-type: none">● Diseño e implementación de los <i>endpoints</i> para el registro de los usuarios con perfil administrativo y profesor.● Registro en la base de datos.● Validación de datos requeridos.	40H

		Módulo de inicio de sesión	HU002	Iniciar sesión, cerrar sesión y modificar contraseña.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para el inicio de sesión, cerrar sesión y modificar contraseña para todos los usuarios. ● Consulta a la base de datos y autorización. ● Validación de los datos requeridos. 	
		Módulo de perfil de usuario	HU003	Modificar perfil de usuario.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para visualizar y modificar el perfil de usuario para todos los usuarios. ● Consulta en la base de datos. ● Validación de los datos requeridos. 	
SB002	Diseño e implementación de <i>endpoints</i> para el usuario administrador.	Módulo de gestión de pasantes	HU004	Gestión de pasantes.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para registrar y visualizar pasantes. 	40H

				<ul style="list-style-type: none"> ● Validación de los datos requeridos. ● Registro en la base de datos. ● Verificación que el registro sea único. 	
	Módulo de gestión de inventarios	HU005	Gestionar inventarios.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para visualizar registrar, modificar, visualizar y eliminar inventarios. ● Validación de los datos requeridos. ● Consulta en la base de datos 	
	Módulo de visualización de reportes de inventarios	HU006	Visualizar inventarios.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para visualizar reporte de inventarios. ● Consulta en la base de datos. 	

SB003	Diseño e implementación de <i>endpoints</i> para el usuario administrativo.	Módulo de gestión de comentarios y/o sugerencias	HU007	Gestionar comentarios y/o sugerencias.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar y atender comentarios y sugerencias. 	30H
		Módulo de solicitud de <i>tickets</i> de asistencia	HU008	Solicitar <i>tickets</i> de asistencia.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para solicitar <i>tickets</i> de asistencia. Validación de los datos requeridos. Registro en la base de datos. 	
		Módulo de envío de comentarios y/o sugerencias	HU009	Enviar comentarios y/o sugerencias.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para enviar comentarios y/o sugerencias. Validación de los datos requeridos. Registro en la base de datos. 	

SB004	Diseño e implementación de endpoints para el usuario profesor	Módulo de envío de comentarios y/o sugerencias	HU008	Solicitar comentarios y/o sugerencias.	<ul style="list-style-type: none"> Diseño e implementación de endpoints para visualizar y atender comentarios y sugerencias. 	30H
		Módulo de solicitud de tickets de asistencia	HU009	Solicitar tickets de asistencia.	<ul style="list-style-type: none"> Diseño e implementación de endpoints para solicitar tickets de asistencia. Validación de los datos requeridos. Registro en la base de datos. 	
		Módulo de solicitud de reservas	HU0010	Solicitar reservas.	<ul style="list-style-type: none"> Diseño e implementación de endpoints para solicitar reservas. Validación de los datos requeridos. Registro en la base de datos. 	
	Diseño e implementación de endpoints para el	Módulo de gestión de tickets de asistencia	HU0011	Gestionar tickets de asistencia.	<ul style="list-style-type: none"> Diseño e implementación de endpoints para visualizar y atender tickets. 	30H

SB005	usuario pasante	Módulo de gestión de reservas	HU0012	Gestionar reservas.	<ul style="list-style-type: none"> ● Diseño e implementación de <i>endpoints</i> para visualizar y tramitar reservas. ● Validación de los datos requeridos. ● Registro en la base de datos. 	
SB006	Pruebas del <i>backend</i>			<ul style="list-style-type: none"> ● Pruebas unitarias. ● Pruebas de compatibilidad. ● Pruebas de estrés. 		10H
SB007	Despliegue del <i>backend</i>			<ul style="list-style-type: none"> ● Despliegue del <i>backend</i> en Azure. 		10H
Documentación				<ul style="list-style-type: none"> ● Trabajo de Integración Curricular. ● Anexos. 		30H
TOTAL						240 H

Elaboración de la base de datos relacional

Este apartado presenta las 14 tablas de la base de datos relacional SQL que se han implementado para el manejo de la información del *backend*, siendo así la **Fig. 32** muestra los detalles de cada tabla y sus respectivas relaciones manteniendo una correcta organización de la información.

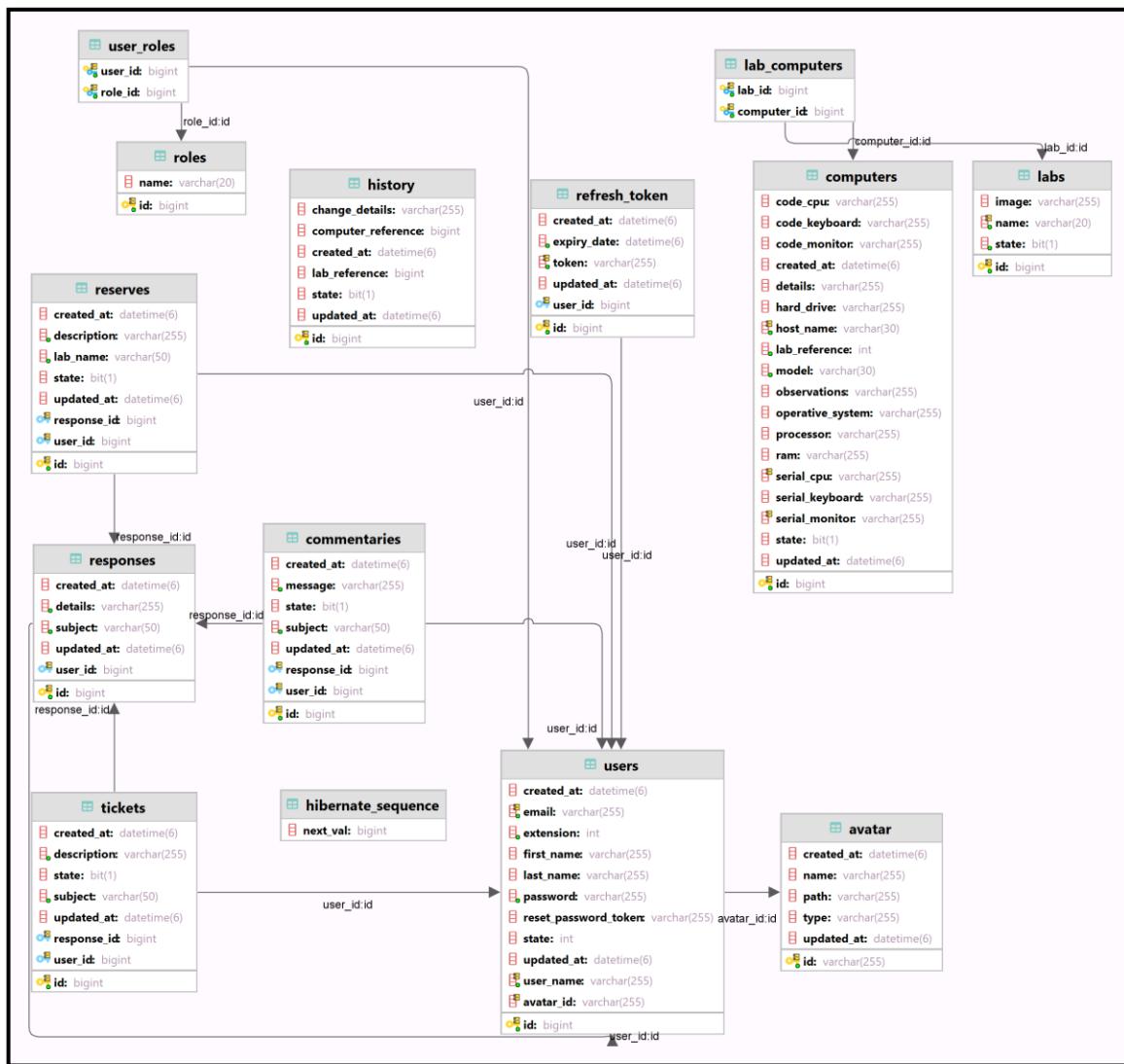


Fig. 32: Estructura de la base de datos relacional.

Pruebas

Al completarse la etapa de desarrollo se aplican las pruebas pertinentes para asegurar que el funcionamiento del *backend* sea el deseado en base a lo estipulado por la recopilación de requerimientos y el dueño del producto.

Pruebas unitarias

Para la implementación de varias pruebas unitarias se utilizan una serie de bloques de código descritos de la siguiente forma: la **Fig. 33** muestra el bloque de código empleado para probar el listado de todas las computadoras registradas mediante un método *GET* el cual retorna un código 200 si es correcto con ello, la **Fig. 34** muestra el resultado de la prueba; la **Fig. 35** muestra el bloque de código empleado para probar el cambio de una computadora de un laboratorio a otro mediante un método *PUT* siendo que si el cambio es realizado con éxito se devuelve un código 200 con ello, la **Fig. 36** muestra el resultado de la prueba; por último la **Fig. 37** muestra el bloque de código empleado para probar la obtención de un pasante por su nombre mediante un método *GET* el cual retorna un código 200 si se ha encontrado el pasante especificado con ello, la **Fig. 38** muestra el resultado de la prueba.

```
@Test
public void indexComputersShouldReturnOk() throws Exception {
    this.mockMvc.perform(requestBuilder: get(urlTemplate: uri + "computer/index")
        .contentType(contentType: MediaType.APPLICATION_JSON)
        .header(name: "Authorization",
            ...values: "Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbkiIs]
        .andExpect(matcher: status().isOk()));
}
```

Fig. 33: Bloque de código para probar el listado de computadoras.

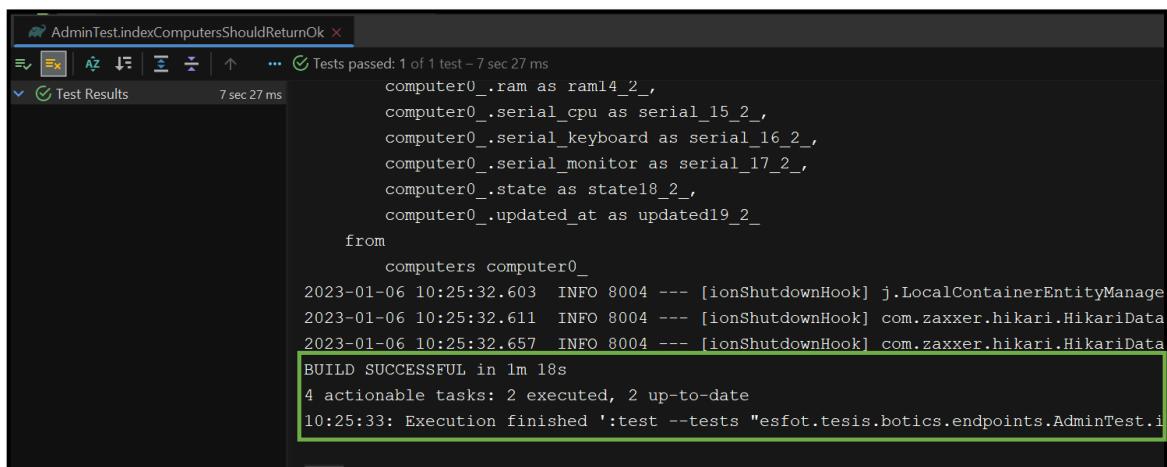


Fig. 34: Resultado de la prueba para el listado de computadoras.

```

@Test
public void reassignComputerShouldReturnOk() throws Exception {
    this.mockMvc.perform(
        requestBuilder: put( urlTemplate: uri + "/computer/reassign/2/3/130")
        .contentType( contentType: MediaType.APPLICATION_JSON)
        .header( name: "Authorization", ...values: "eyJhbGciOiJIUzUxMiJ9.eyJzdWI")
        .andExpect( matcher: status().isOk()));
}

```

Fig. 35: Bloque de código para probar el cambio de una computadora.

```

AdminTest.reassignComputerShouldReturnOk ✘
Tests passed: 1 of 1 test - 4 sec 787 ms
Test Results      4 sec 787 ms
from
    user_roles roles0_
    inner join
        roles role1_
        on roles0_.role_id=role1_.id
    where
        roles0_.user_id=?
2023-01-06 11:23:43.589  INFO 8084 --- [ionShutdownHook] j.LocalContainerEntityManagerFactory
2023-01-06 11:23:43.593  INFO 8084 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-06 11:23:44.912  INFO 8084 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 1m 31s
4 actionable tasks: 1 executed, 3 up-to-date
11:23:45: Execution finished ':test --tests "esfot.tesis.botics.endpoints.AdminTest.reassign'

```

Fig. 36: Resultado de la prueba para el cambio de una computadora.

```

@Test
public void showInternByNameShouldReturnOk() throws Exception {
    this.mockMvc.perform( requestBuilder: get( urlTemplate: uri + "intern/" + "Chris")
        .contentType( contentType: MediaType.APPLICATION_JSON)
        .header( name: "Authorization", ...values: "Bearer eyJhbGciOiJIUzUxMiJ9.e")
        .andExpect( matcher: status().isOk()));
}

```

Fig. 37: Bloque de código para probar la obtención de un pasante por su nombre.

```

AdminTest.showInternByNameShouldReturnOk ✘
Tests passed: 1 of 1 test - 11 sec 718 ms
Test Results      11 sec 718 ms
on response0_.id=ticket3_.response_id
left outer join
    tickets ticket3_
    on response0_.id=ticket3_.response_id
where
    response0_.user_id=?
2023-01-11 23:27:40.188  INFO 5308 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2023-01-11 23:27:40.193  INFO 5308 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2023-01-11 23:27:40.241  INFO 5308 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
BUILD SUCCESSFUL in 3m 3s
4 actionable tasks: 3 executed, 1 up-to-date
23:27:41: Execution finished ':test --tests "esfot.tesis.botics.endpoints.AdminTest.showInternByName'

```

Fig. 38: Resultado de la prueba para la obtención de un pasante por su nombre.

Pruebas de compatibilidad

Para la implementación de las pruebas de aceptación se utilizan varios clientes HTTP, en este caso se utilizan los clientes *Swagger* y *Thunder Client* con los cuales se verifica que el funcionamiento de la API en diferentes clientes sea el correcto y la información sea la misma. Con ello se evalúa el *endpoint* para el inicio de sesión en *Swagger* siendo que la **Fig. 39** muestra el resultado que se ha obtenido tras la petición.

Por otro lado, se utiliza *Thunder Client* para evaluar el inicio de sesión, siendo este un cliente HTTP que funciona como una extensión para el editor *Visual Studio Code* se muestra la **Fig. 40** el resultado que se ha obtenido desde el mismo comprobándose que se obtiene el mismo resultado en ambos clientes.

The screenshot shows the Swagger UI interface with the following details:

- Curl:** A code block containing a curl command to sign in to the API endpoint `https://botics.loca.lt/api/v1/auth/signin` with username "admin" and password "Secret1\$".
- Request URL:** The URL `https://botics.loca.lt/api/v1/auth/signin`.
- Server response:** A table with columns "Code" and "Details". The "Code" column shows "200".
- Response body:** A JSON object representing a user profile:

```
{  "id": 1,  "username": "admin",  "email": "cristhianpalacios3@gmail.com",  "roles": [    "ROLE_ADMIN"  ],  "jwtToken": "eyJhbGciOiJIUzIwMjQ9.eyJzdWJlciI6ImJvdGljcyIsInlhdcI6MTY3MzAyNjg5NywiZXhwIjoxNjczMDIzNzk3fQ.gTX6_XeemDYiB0UTcPYHQUf1XSVxlnF31E-nHzdnIs6dzMcxYnlyOzm38meurZ6ognO2ljfw60CjwYr01wWsQ"}
```
- Response headers:** A section showing various header fields.

Fig. 39: Resultado de la prueba de inicio de sesión con *Swagger*.

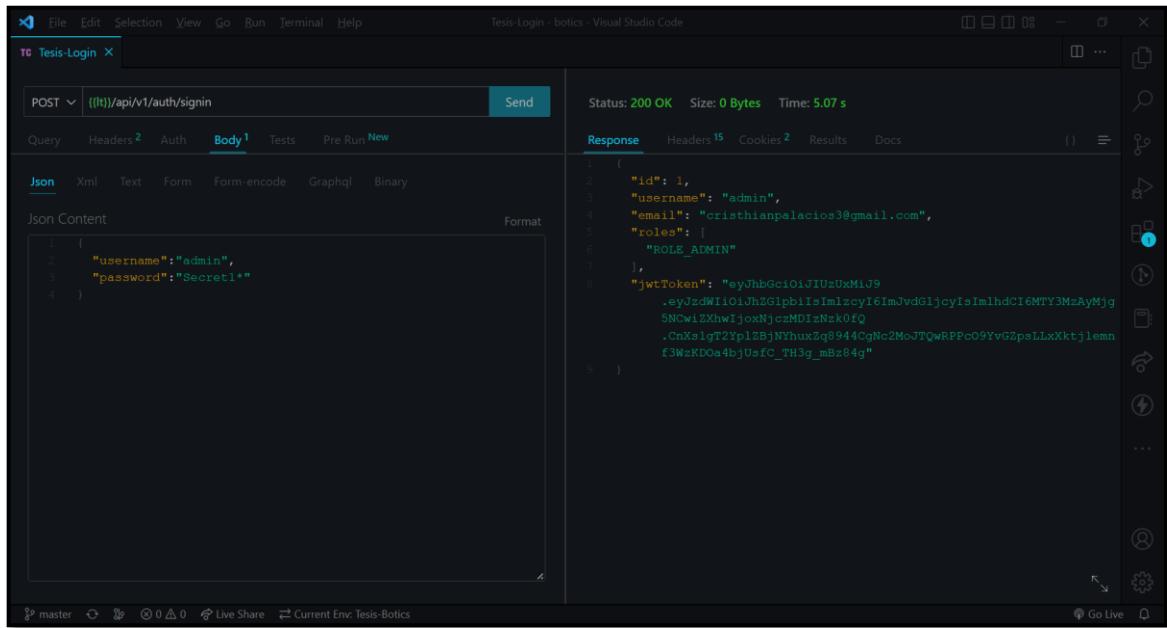


Fig. 40: Resultado de la prueba de inicio de sesión con *Thunder Client*.

Prueba de estrés

En este apartado se presentan 11 pruebas de estrés mostrando cual es el comportamiento de los *endpoints* ante un cierto flujo de peticiones, verificando que estos funcionen de la forma en la que se espera en un tiempo determinado. Con ello, desde la **Fig. 41** que corresponde a la prueba de estrés número 2 hasta la **Fig. 51** que corresponde a la prueba de estrés número 12, se presentan los resultados que se han obtenido.

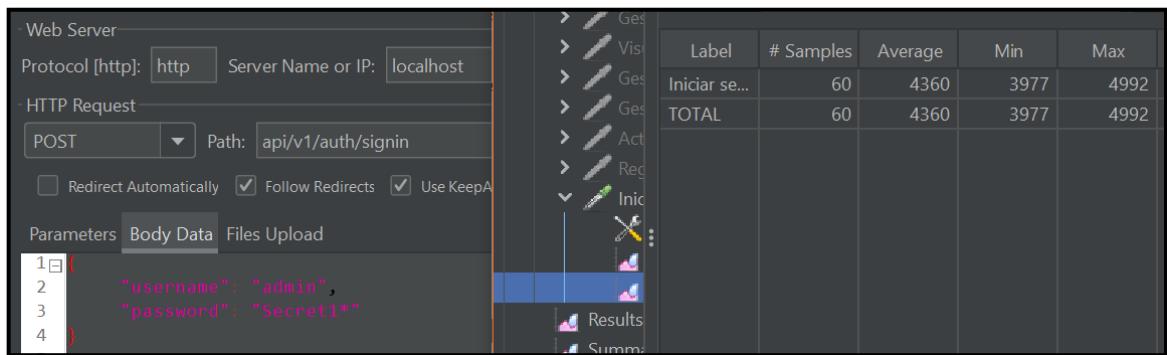


Fig. 41: Prueba de estrés para la historia de usuario N°2.

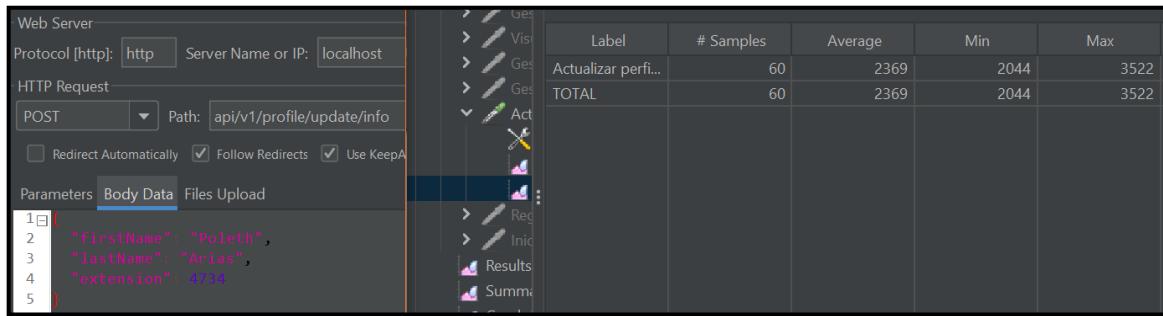


Fig. 42: Prueba de estrés para la historia de usuario N°3.

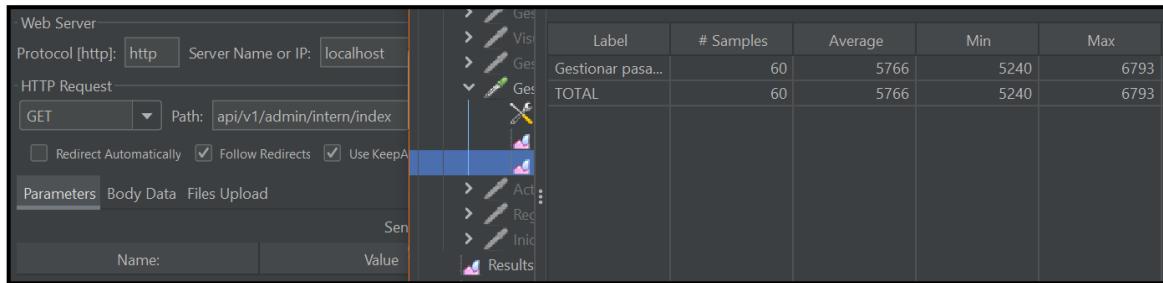


Fig. 43: Prueba de estrés para la historia de usuario N°4.

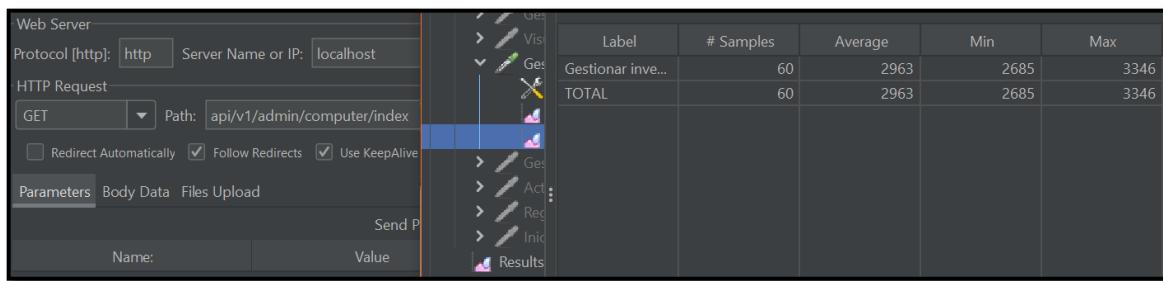


Fig. 44: Prueba de estrés para la historia de usuario N°5.

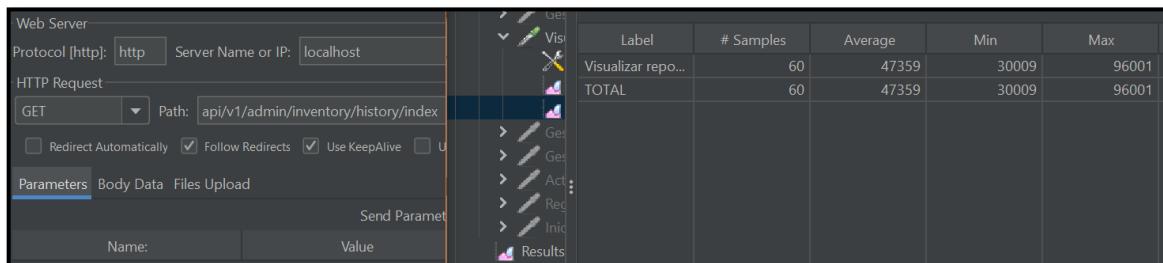


Fig. 45: Prueba de estrés para la historia de usuario N°6.

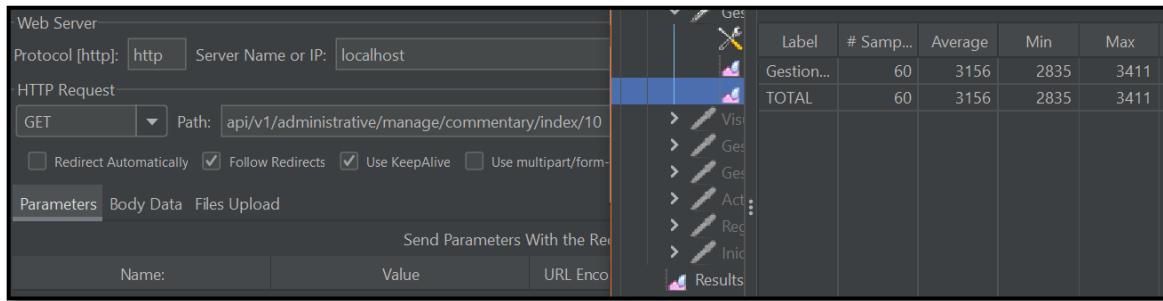


Fig. 46: Prueba de estrés para la historia de usuario Nº7.

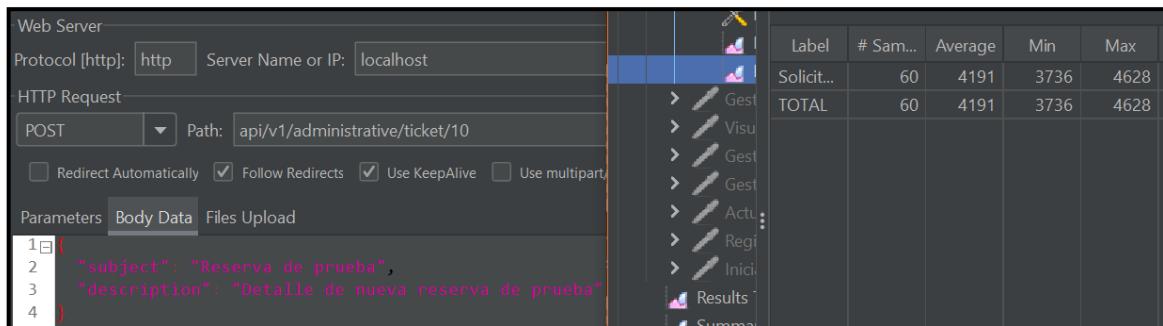


Fig. 47: Prueba de estrés para la historia de usuario Nº8.

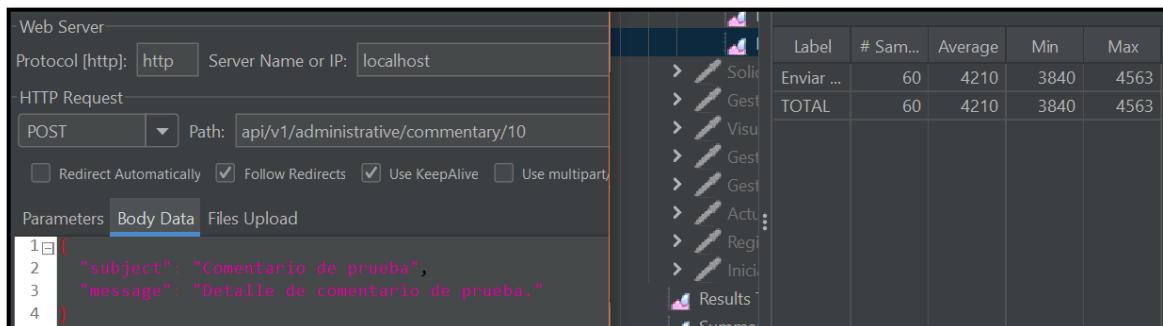


Fig. 48: Prueba de estrés para la historia de usuario Nº9.

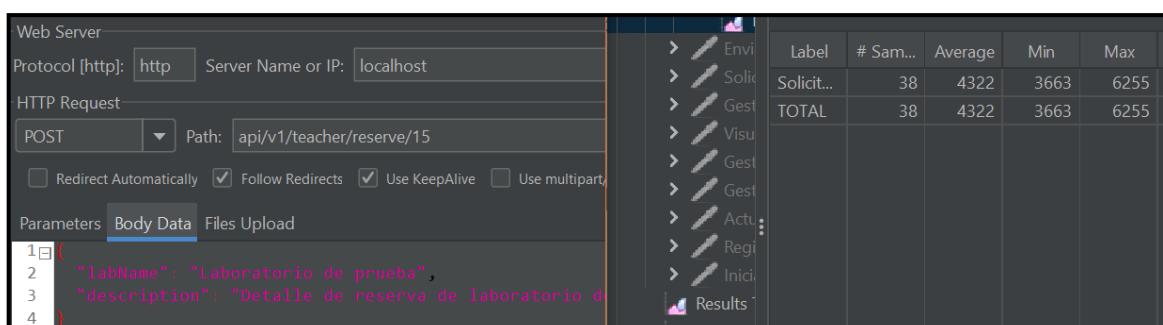


Fig. 49: Prueba de estrés para la historia de usuario Nº10.

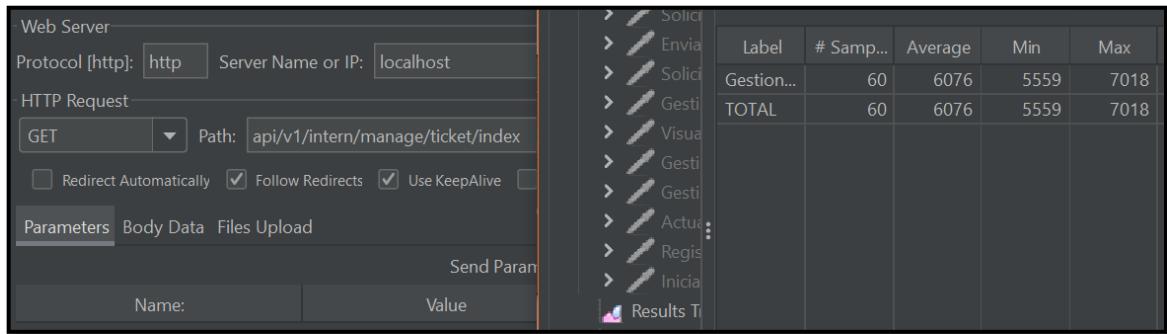


Fig. 50: Prueba de estrés para la historia de usuario N°11.

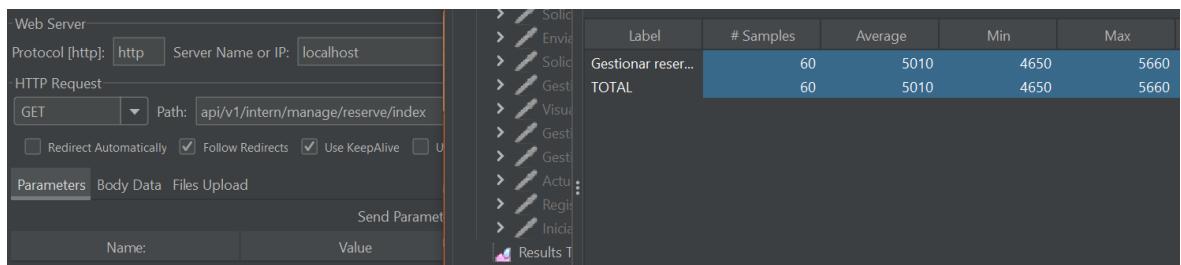


Fig. 51: Prueba de estrés para la historia de usuario N°12.

ANEXO III

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

<https://youtu.be/tyE6poeRdYI>

En donde se explica de forma clara y sencilla las diversas funcionalidades del *backend*, así como cada uno de los perfiles que forman parte de este componente.

ANEXO IV

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

<https://github.com/ChristianPPP/botics>

Credenciales para el acceso del *backend*

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

<https://botics.azurewebsites.net/swagger-ui/index.html>

Credenciales del perfil administrador:

- Correo del administrador: admin@epn.edu.ec
- Contraseña: **Secret1***

Credenciales del perfil administrativo:

- Correo del administrador: administrativo@epn.edu.ec
- Contraseña: **Secret1***

Credenciales para el perfil pasante:

- Correo de pasante: pasante@epn.edu.ec
- Contraseña: **Secret1***

Credenciales para el perfil docente:

- Correo de docente: profesor@epn.edu.ec
- Contraseña: **Secret1***

Repositorio del *backend*

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

URL:

<https://github.com/ChristianPPP/botics>