

ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA PARA BRINDAR SERVICIOS INFORMÁTICOS EN QUITO

DESARROLLO DE BACKEND

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

MANUEL JOSUE AUQUI SANCHEZ

manuel.auqui@epn.edu.ec

DIRECTOR: ING. BYRON LOARTE

DMQ, febrero 2023

CERTIFICACIONES

Yo, Manuel Josue Auqui Sánchez 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.

MANUEL JOSUE AUQUI SÁNCHEZ

manuel.auqui@epn.edu.ec

manuelauqui19@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Manuel Josue Auqui Sánchez, 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.

Manuel Josue Auqui Sánchez

DEDICATORIA

Queridos padres, con mucho orgullo y satisfacción hoy puedo decir que he terminado mi proyecto de Integración Curricular y con ello, mi carrera como desarrollador de software. Pero esto no habría sido posible sin el apoyo y la guía de ustedes.

A mis padres, quiero decirles que no tengo palabras para expresar lo agradecido que tengo por su amor y apoyo incondicional. Han estado a mi lado en cada paso del camino y siempre han creído en mí, incluso cuando yo mismo dudaba. Gracias por ser el pilar que me ha permitido llegar hasta aquí.

Con todo mi cariño y gratitud, dedico este proyecto de Integración Curricular a ustedes, mis padres. Espero haber hecho honor a su confianza en mí y haber cumplido con las expectativas que depositaron en mí.

Manuel Josue Auqui Sánchez

AGRADECIMIENTO

A mis profesores, quiero expresarles mi más sincero agradecimiento por su dedicación y paciencia en enseñarme todo lo que necesitaba para convertirme en el profesional que soy hoy. Sus enseñanzas y consejos me han sido invaluables y siempre llevaré su sabiduría conmigo en mi carrera.

Por último, pero no menos importante, quiero agradecer a la Escuela Politécnica Nacional por brindarme la oportunidad de completar mi carrera en un entorno de excelencia académica. Me siento muy afortunado de haber podido formar parte de esta institución y estoy seguro de que esta experiencia me será muy útil en el futuro.

Con todo mi cariño y gratitud, dedico este proyecto de Integración Curricular a ustedes, mis profesores y Escuela Politécnica Nacional. Espero haber hecho honor a su confianza en mí y haber cumplido con las expectativas que depositaron en mí.

Manuel Josue Auqui Sánchez

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 la arquitectura	11
	Patrón arquitectónico	11
2.3	Herramientas de desarrollo	12
	Librerías.....	13
3	RESULTADOS	14
3.1	<i>Sprint 0.</i> Configuración del ambiente de desarrollo	14
	Definición de requerimientos y restricciones para el <i>backend</i>	14
	Diseño e implementación de tablas y relaciones para la Base de datos SQL.....	17
	Definición de archivos, directorios y configuraciones para el proyecto <i>backend</i>	18
	Definición de módulos por cada rol de usuario	19
3.2	<i>Sprint 1.</i> Perfil administrador – implementación de <i>endpoints</i>	20
	Inicio de sesión, cierre de sesión y recuperación de contraseña	20
	Modificación de perfil de usuario	22
	Gestión de solicitudes de afiliación	24
	Visualización de comentarios, sugerencias y calificación de los técnicos	26
3.3	<i>Sprint 2.</i> Perfil técnico – implementación de <i>endpoints</i>	28
	Registro de usuario para el perfil técnico	29
	Solicitud de afiliación	30
	Gestión de servicios	32
	Aprobación de contratos	34
	Visualización de comentarios, sugerencias y calificación de los servicios	36
3.4	<i>Sprint 3.</i> Perfil cliente – implementación de <i>endpoints</i>	37
	Inicio de sesión, cierre de sesión y recuperación de contraseña para el perfil cliente	37
	Registro de usuario para el perfil cliente	38

Visualización de servicios	40
Contratación de servicios	41
Gestión de solicitudes de contratación	42
Comentar, sugerir y calificar servicios	45
3.5 Sprint 4. Ejecución de pruebas para el componente <i>backend</i>.	46
Resultados de la elaboración de las pruebas unitarias en el <i>backend</i>	46
Resultados de la elaboración de las pruebas de compatibilidad en el <i>backend</i>	48
Resultados de la elaboración de las pruebas de carga en el <i>backend</i>	48
3.6 Sprint 5. Despliegue del <i>backend</i> a un ambiente de producción.	49
4 CONCLUSIONES.....	50
5 RECOMENDACIONES.....	51
6 REFERENCIAS BIBLIOGRÁFICAS.....	52
7 ANEXOS	55
ANEXO I	56
ANEXO II	57
ANEXO III	86
ANEXO IV.....	87

RESUMEN

Actualmente, existe una falta de aplicaciones móviles que permitan a los ciudadanos encontrar servicios especializados en reparación de equipos de informática, al igual que un catálogo de técnicos especializados para ofertar dichos servicios. Por otra parte, existe otro inconveniente y es que tienen acercarse de manera presencial a dejar sus dispositivos tecnológicos en tiendas físicas y ser vulnerables a situaciones como: pérdida de tiempo, robos, precios altos, locales cerrados y lidiar con personal no capacitado lo que ocasiona que no exista garantías en el servicio ofertado.

La integración de aplicaciones web y móviles han otorgado varias ventajas para que los servicios en línea tengan un gran crecimiento, un impacto positivo en la sociedad y una mayor visibilidad para negocios en crecimiento y tiendas físicas, además permiten ofertar catálogos digitales, accesibilidad en cualquier lugar y momento, comodidad al buscar servicios y pagos más rápidos y seguros.

Con el objetivo de ayudar a la ciudadanía en general, en este proyecto de Integración Curricular presenta el desarrollo de un *backend* denominado “Tecnony”, que otorga una serie de *endpoints* para la creación de servicios técnicos especializados en informática a través de un *frontend* y que los servicios puedan ser contratados por los ciudadanos a través de una aplicación móvil. Además, el *backend* permite que los técnicos especializados puedan brindar sus servicios de manera segura con las condiciones y garantías apropiadas para el negocio.

El presente documento se compone de cinco capítulos, los cuales se componen de la descripción de componente, seguido de la metodología de desarrollo para luego presentar los resultados derivados del proyecto y terminar con las conclusiones y recomendaciones del proyecto, junto con las referencias bibliográficas y anexos.

PALABRAS CLAVE: *Backend*, Laravel, Asistencia técnica, *Endpoint*, Informática, Servicios informáticos.

ABSTRACT

Currently, there is a lack of mobile applications that allow citizens to find specialized computer repair services, as well as a catalog of specialized technicians to offer those services. Furthermore, there is another inconvenience and that is that they have to approach in a face-to-face manner to leave their technological devices in physical stores and be vulnerable to situations such as: loss of time, thefts, high prices, closed stores and dealing with untrained personnel which causes there to be no guarantees in the offered service.

The integration of web and mobile applications has provided several advantages for online services to have a great growth, a positive impact on society, and greater visibility for growing businesses and physical stores, in addition to allowing the offer of digital catalogs, accessibility anywhere and anytime, convenience in searching for services, and faster and safer payments.

With the goal of helping the general citizenry, this Curricular Integration project presents the development of a backend called "Tecnony", which provides a series of endpoints for the creation of specialized technical services in computer science through a frontend and that the services can be contracted by citizens through a mobile application. Additionally, the backend allows specialized technicians to provide their services safely with the appropriate conditions and guarantees for the business.

The document consists of seven chapters, which are composed of the description of the component, followed by the development methodology and the results derived from the project. Finally, it ends with the conclusions and recommendations of the project, along with the bibliographical references and appendices.

KEYWORDS: Backend, Laravel, Technical Assistance, Endpoint, Computing, Computer Services.

1 INTRODUCCIÓN

Debido al aislamiento provocado por el Covid-19 el uso del Internet en los hogares aumento en un 63% en el Ecuador, según los monitoreos realizados por Arcotel [1]. Así mismo, el equipamiento de dispositivos tecnológicos en hogares como computadoras de escritorio y portátiles también presentaron un incremento del 11.5% a nivel nacional según datos recopilados por el INEC [2].

En la última encuesta realizado a nivel nacional por INEC de las tecnologías de la información, determinó que las personas que tienen acceso a Internet desde sus hogares creció en un 86,1% para el año 2020 a nivel nacional. Es evidente el gran incremento que tuvo el uso de Internet en comparación al 68,1% del posterior año a la pandemia [2], ya que los servicios en línea también aumentaron y cobraron un gran impacto facilitando de esta manera las relaciones comerciales en todo el país [3].

Con base a las estadísticas mencionadas se puede deducir que cada hogar dispone de al menos 2 dispositivos tecnológicos con conexión a Internet los cuales son propensos a presentar fallas por su constante uso. Además, estos dispositivos tecnológicos necesitan una asistencia técnica para conservar su buen funcionamiento. Sin embargo, es evidente la falta de una aplicación tecnológica que le permita, por una parte, a la ciudadanía encontrar servicios especializados en reparación de equipos de informática y por otra, que los técnicos especializados puedan ofrecer sus servicios de mantenimiento en tiempo real. Adicional a ello, existe otra serie de inconvenientes cuando los usuarios tienen que dejar los equipos en negocios y tiendas físicas como, por ejemplo: pérdida de tiempo, robo del equipo, negocios cerrados, precios indebidos, personal no capacitado, falta de garantías en el servicio realizado, entre otras.

En la actualidad, el uso de aplicaciones digitales ha logrado que los servicios en línea obtengan un gran crecimiento, un impacto positivo en la sociedad y sobre todo un mayor posicionamiento para negocios y tiendas físicas, ya que brindan una serie de ventajas tales como: catálogos digitales, accesibilidad desde cualquier lugar y en cualquier momento, comodidad en la búsqueda de servicios, pagos más rápidos y seguros, entre otras [3].

Por lo dicho anteriormente y con el objetivo de ayudar a la ciudadanía en general, el presente proyecto de Integración Curricular propone el desarrollo de un *backend* el cual otorgue una serie de *endpoints* que permitan la creación servicios técnicos especializados en informática y que los mismos puedan ser contratados por la ciudadanía desde una aplicación móvil. La implementación y puesta en funcionamiento del backend en producción permite, por un lado, que las aplicaciones web o móviles puedan acceder a

todos los endpoints y por otra, permite que los técnicos especializados tengan la posibilidad de brindar sus servicios de una manera segura con las condiciones y garantías que corresponda al negocio.

1.1 Objetivo general

Desarrollar un sistema para brindar servicios informáticos en Quito.

1.2 Objetivos específicos

1. Levantar los requerimientos necesarios para el desarrollo del *backend*.
2. Diseñar la base de datos para el *backend* de acuerdo con los requerimientos que se han adquirido.
3. Aplicar el patrón arquitectónico (Modelo, Vista y Controlador) en la codificación de los *endpoints*
4. Probar el funcionamiento de los *endpoints*.
5. Desplegar el *backend* a producción para su consumo.

1.3 Alcance

Gracias al uso del internet y de sistemas informáticos el posicionamiento de pequeños negocios y tiendas físicas es mucho más fácil, permitiendo encontrar a nuevos clientes y colaboradores, además permite dar confianza y respaldo al negocio, generando un mayor crecimiento de ventas gracias a la contratación de servicios *online* [4].

En el desarrollo de *software*, la importancia de un *backend*, reside en el intercambio de información con una intensificación en la seguridad y accesibilidad a la información solicitada por los usuarios, de igual modo permite tener un control más seguro y optimizado para comunicarse e intercambiar información con otros sistemas [5]. Por otra parte, para adquirir el máximo beneficio de las herramientas y librerías que existen en el mercado, el desarrollo del *backend* otorga una serie de *endpoints* para que la ciudadanía a través de una aplicación móvil pueda contratar servicios especializados en el área de informática y que la gestión de los servicios, solicitudes de afiliación, gestión de comentarios y sugerencias por parte de los técnicos se lo pueda realizar desde un *frontend* previamente desarrollado. Manejando para ello, se utiliza una amplia variedad de tecnologías del lado del servidor, un enfoque de desarrollo ágil para cumplir los objetivos y una serie de pruebas

para garantizar la calidad del producto final. Posteriormente, el *backend* establece 3 tipos de perfiles de usuarios que se describen a continuación

Usuarios que dispone el *backend*:

- Administrador
- Técnico
- Cliente

El *backend* para el usuario administrador implementa:

- *Endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.
- *Endpoints* para modificar el perfil de usuario.
- *Endpoints* para gestionar solicitudes de afiliación.
- *Endpoints* para visualizar comentarios, sugerencias y calificación de los técnicos.

El *backend* para el usuario técnico implementa:

- *Endpoints* para registrarse.
- *Endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.
- *Endpoints* para modificar el perfil de usuario.
- *Endpoints* para solicitar afiliación.
- *Endpoints* para gestionar servicios.
- *Endpoints* para aprobar contrataciones.
- *Endpoints* para visualizar los comentarios, sugerencias y calificación del servicio.

El *backend* para el cliente implementa:

- *Endpoints* para registrarse.
- *Endpoints* para iniciar sesión, cerrar sesión y recuperar contraseña.
- *Endpoints* para modificar el perfil de usuario.
- *Endpoints* para visualizar servicios.
- *Endpoints* para contratar un servicio.

- *Endpoints* para gestionar solicitudes.
- *Endpoints* para comentar, sugerir y calificar servicios.

1.4 Marco teórico

El componente que administra toda la información en un sistema de *software* se designa como *backend*. Es el encargado de implementar toda la lógica del negocio, manejo de la información y la comunicación con otras aplicaciones a través de solicitudes HTTP [6]. En el *backend* se encuentran elementos que los usuarios finales no pueden interactuar directamente, como lo son: API's, librerías, lenguajes de programación, base de datos, entre otros. Adicional a ello, existen muchos lenguajes de programación y cada uno de ellos poseen diferentes *Frameworks* lo que posibilita que el desarrollo del *backend* sea mucho más rápido y sencillo [7].

Los *frameworks* surgieron para normalizar la estructura del código de cualquier sistema o aplicación *software* y que el código sea escalable a futuro. Además, un *framework* otorga una serie de ventajas como: automatización de patrones de programación, código más entendible y la separación de la lógica en tres capas que son: lógica de la presentación, lógica de datos y lógica de negocio [8].

Un régimen de codificación que se basa en la creación e interacción de objetos del mundo real se designa como programación orientada a objetos [9]. Además, dispone de elementos y conceptos propios los cuales son claves durante el desarrollo de cualquier sistema o *software*. En ese sentido, los elementos más importantes son las clases que es un modelo para la creación de objetos, los objetos son una entidad provista de comportamientos y valores de estado; por último, los métodos son comportamientos que realizan los objetos [9].

Las bases de datos relacionales provienen de un modelo estándar para representar y consultar datos. Su principal virtud es el uso de tablas que se relacionan entre sí para crear grupos de datos mejor estructurados y ordenados. Intrínsecamente en una base de datos relacional, un registro posee un identificador único nombrado *primary key* o clave primaria [10]. Esto permite a los sistemas *software* conservar datos de manera íntegra, facilitar el acceso a la información y evitar la duplicidad de los mismos.

Las relaciones polimórficas es un concepto ampliamente utilizado por el *Framework* Laravel, el cual es un método para relacionar un comportamiento o atributo que provienen

de distintos modelos [11]. Es decir, una relación polimórfica permite que el modelo de destino pueda almacenar información de uno o varios modelos a la vez utilizando para ello una sola asociación. Además, admite la reutilización de tablas, mejorando la rapidez en las consultas y disminuyendo la cantidad de información almacenada [12].

Una *API* es un grupo de definiciones y protocolos implementados para el intercambio de información entre varias aplicaciones, cuyo contenido se transmite a través del protocolo HTTP en formato JSON o XML. Por otro lado, *REST* es un conjunto de reglas de arquitectura que se implementa para construir una API [13]. Por último, una API RESTful es un medio de comunicación e intercambio de información para aplicaciones *software* que se ajustan a los términos de la arquitectura *REST*.

Un *ORM* es conocido por ser un “Mapeo de objetos relacional”, el cual es una herramienta que posibilita modelar una Base de datos relacional y asociarlo a una entidad lógica dentro del desarrollo del *backend*. Además, permite realizar consultas mucho más fáciles, ya que gestiona la persistencia de datos accediendo indirectamente a la base ignorando la escritura código SQL [14].

El lenguaje más comúnmente utilizado para crear aplicaciones *web* es PHP, el cual trabaja substancialmente del lado del servidor por medio de *scripts*, por lo que posee diversas ventajas para recopilación de datos a través de formularios, forjar contenido dinámico, gestionar *cookies*, etc. Además, PHP contiene soporte para trabajar con protocolos de comunicaciones como HTTP [15].

Un *web services* es una tecnología multiplataforma de tipo cliente y servidor, que es distribuida, es decir posibilita acceder a su contenido desde cualquier lugar a través de diferentes dispositivos que dispongan una conexión a internet [16].

HTTP es un protocolo de trasferencia de hipertexto, que permite realizar peticiones de datos o recursos desde el cliente hacia un servidor remoto, en donde el cliente se define como el navegador *web* que se ejecuta a través de un dispositivo tecnológico utilizando el puerto 80 para su conexión remota y el puerto 443 para una conexión más segura, mientras que el servidor sirve todo el contenido al cliente a través de un documento HTML [17].

Una peculiaridad de las comunicaciones HTTP es que posee códigos de estado que ayudan a entender lo que pasa durante las peticiones y respuesta de una conexión. Los códigos de estados que utiliza son de clase 100 que indica que las peticiones realizadas por el cliente aún están siendo procesadas, las de clase 200 indica que las peticiones tienen una respuesta exitosa, las de clase 300 indica el re-direcccionamiento de una respuesta, las

de clase 400 indica errores realizados en la solicitud del cliente y las de clase 500 indica errores de respuesta del lado del servidor [17].

2 METODOLOGÍA

El método de investigación que aborda un tema o acontecimiento en particular de manera profunda y busca una mayor compresión de su complejidad es conocido como estudio de casos [18]. De la misma manera, posee varias ventajas sobre otros métodos de investigación por ser un tipo de investigación cualitativa, es decir, explora de forma rigurosa el contexto que rodea la investigación con el objetivo de desarrollar nuevas hipótesis ya que la investigación recae sobre el investigador [19].

Por este motivo, el presente proyecto efectúa el estudio de casos como principal método investigativo en el desarrollo del componente *backend* ya que se parte inicialmente sobre una investigación en las principales problemáticas que existe al momento de ofertar servicios especializados en informática y lograr con ello dotar a los técnicos especializados un medio para que consigan publicar sus servicios informáticos y que la ciudadanía en general logre acceder a los mismos, gracias al uso del internet y dispositivos tecnológicos.

2.1 Metodología de desarrollo

Es una serie de regulaciones o estándares, donde se define la planificación, gestión, procedimientos, herramientas y documentos, que se presentan a lo largo de la realización de un proyecto de *software*. Por otra parte, como un nuevo modo de desarrollar *software*, nació las metodologías de desarrollo ágil, las cuales permiten implementar un ciclo de vida al desarrollo y que el mismo sea iterativo, adaptable y flexible [20].

La metodología de desarrollo ágil con mayor popularidad es *Scrum*, la cual es utilizada como una guía para el desarrollo de actividades que intervienen en un proyecto de *software*. Posee una serie de fases, roles y artefactos que permiten lograr el objetivo del proyecto propuesto, es por este motivo que *Scrum* es altamente eficaz en proyectos con plazos de entregas relativamente cortos y cambios imprevistos en los requerimientos [21].

Por tal motivo, este proyecto se ha llevado a cabo mediante la implementación de *Scrum*, ya que otorga al proyecto una serie de roles con tareas definidas un conjunto de artefactos para gestionar adecuadamente la información y una serie de iteraciones o *Sprints* para llevar a cabo todas las funcionalidades del *backend*, muestra de ello en las siguientes secciones se describe la implementación de cada una de las etapas de esta metodología.

Roles

Scrum es una metodología bien estructurada ya que permite el trabajo organizado y colaborativo a través de roles que se designan al equipo de desarrollo [21]. Los cuales son descritos a continuación.

Product Owner

Es el propietario del *software* y la persona encargada de definir los requisitos del producto a ser desarrollado, ya que él conoce todo el entorno del negocio y la visión del producto, además, sus decisiones deben ser respetadas y obedecidas por el resto del equipo [20]. Por consiguiente, en la **TABLA I** se demuestra la asignación de roles para la ejecución de las actividades en el desarrollo del *backend*.

Scrum Master

Se encarga de capacitar y guiar al equipo de desarrollo, además es el responsable de garantizar que el equipo trabaje bajo las prácticas y normas de *Scrum*. Además, este rol debe en todo momento mantener el desempeño del equipo y la calidad del *software* que se desarrolle [20]. Por esta razón, en la **TABLA I** se demuestra la designación de roles para la ejecución de las actividades en el desarrollo del *backend*.

Development Team

Es un grupo de personas encargadas de desarrollar el producto final a partir de los requerimientos que se han definido en el *Sprint Backlog*. Por lo general, es un equipo que desempeñan habilidades multifuncionales y que a la vez se organizan por sí mismos con el objetivo de compartir el conocimiento entre todos los implicados [20]. Por esta razón, en la **TABLA I** se demuestra la designación de roles para la ejecución de las actividades en el desarrollo del *backend*.

TABLA I: Asignación de Roles.

ROLES	NOMBRES
<i>Product Owner</i>	Ing. Byron Loarte, MSc.
<i>Scrum Master</i>	Ing. Byron Loarte, MSc.
<i>Development Team</i>	Manuel Auqui

Artefactos

Los artefactos son elementos que permiten visualizar la información de una forma mucho más detallada y que a la vez facilitan el correcto desempeño de las actividades dentro de un proyecto *software* [21]. A continuación, se presentan los cuatro artefactos esenciales de *Scrum*.

Recopilación de Requerimientos

Es una tabla donde se definen todos los requerimientos esenciales que el *Product Owner* necesita como parte del *software* a desarrollar. Además, es importante su desarrollo ya que este artefacto es una guía fundamental para elaborar los demás artefactos. Adicional a ello, es indispensable recopilar dichos requisitos a través de reuniones o entrevistas ya que el objetivo principal es determinar todos los requerimientos de una forma clara y precisa [21]. Por tal motivo, en la **TABLA II** se muestra una parte de este artefacto, mientras que esta información se detalla a profundidad en el **ANEXO II**.

TABLA II: Fragmento de la recopilación de requerimientos.

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	ID - RR	ENUNCIADO DEL ÍTEM
Backend	RR09	<p>El usuario con perfil técnico necesita utilizar varios endpoints para:</p> <ul style="list-style-type: none">• Visualizar comentarios, sugerencias y calificación del servicio.
	RR010	<p>El usuario con perfil cliente necesita utilizar varios endpoints para:</p> <ul style="list-style-type: none">• Visualizar los servicios.

Historias de Usuario

Es una tarjeta donde se describe de manera mucho más detallada un requerimiento en función de lo que el usuario necesita realizar en un producto *software*. Además, esta tarjeta permite estimar la prioridad, el alcance y el tiempo que toma para el cumplimiento del requerimiento [22]. Por tal motivo, la **TABLA III** muestra una historia de usuario para el actual componente y la información completa se detalla a profundidad en el **ANEXO II**.

TABLA III: Fragmentos de las Historias de Usuario.

HISTORIA DE USUARIO	
Identificador: HU007	Usuario: Técnico
Nombre de historia: Gestionar servicios.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 3	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil técnico pueda gestionar servicios, implementando para ello un <i>endpoint</i> que le permita al técnico: <ul style="list-style-type: none"> • Visualizar servicios. • Crear servicios. • Actualizar servicios. • Cambiar el estado del servicio. 	
Observación: El <i>backend</i> debe verificar que el técnico este afiliado para que pueda gestionar sus servicios.	

Product Backlog

Es una tabla donde se describe la lista de requerimientos de un producto *software* y en donde cada una de sus filas refleja una pequeña descripción, el grado de prioridad, la estimación de esfuerzo y a que iteración corresponde. Por lo general, el *Product Backlog* presenta actualizaciones con el tiempo según los cambios de requisitos [20]. Por tal motivo, en la **TABLA IV** se muestra una parte de este artefacto y la información completa se detalla a profundidad en el **ANEXO II**.

TABLA IV: Fragmento del *Product Backlog*.

PRODUCT BACKLOG				
ID-HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU006	Solicitar afiliación.	2	Finalizado	Alto
HU007	Gestionar servicios.	3	Finalizado	Alto

Sprint Backlog

En este artefacto se concretan las tareas que se realizan durante el desarrollo del *software*. Dichas actividades y tareas suceden en un proceso llamado iteraciones o comúnmente llamado *Sprints*. Además, las tareas que se realizan durante un *Sprint* se concretan por orden de complejidad y prioridad los cuales están definidos en el *Product Backlog*. Así mismo, el *Sprint* se divide en tareas más pequeñas con el fin de obtener una mayor compresión de estas y asignarlas a cada uno de los miembros del *Development Team* para su implementación [20]. Por tal motivo, la **TABLA V** muestra una parte de este artefacto y la información completa se encuentra detallada en el **ANEXO II**.

TABLA V: Fragmento del *Sprint Backlog*.

SPRINT BACKLOG						
ID-SB	NOMBRE	MÓDULO	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB004	Pruebas en el <i>backend</i>			<ul style="list-style-type: none">• Pruebas unitarias.• Pruebas de compatibilidad.• Pruebas de aceptación.		40 horas

2.2 Diseño de la arquitectura

Patrón arquitectónico

Un patrón de arquitectura permite dividir la interfaz gráfica, la lógica y la capa de datos. Esto facilita que la funcionalidad y módulos de un producto *software* sea organizado, escalable con el pasar del tiempo y de fácil comprensión para el desarrollador o el equipo de trabajo [23]. A su vez, este patrón de arquitectura permite aplicar siempre las buenas prácticas y paradigmas de programación.

Para utilizar este patrón de forma eficiente se necesita comprender los tres niveles de abstracción:

- **Modelo:** incorpora la lógica y se faculta de dar acceso a los datos.
- **Vista:** se encarga de exponer la información obtenida desde el modelo y presentarla a través de una interfaz gráfica.

- **Controlador:** participa como mediador entre la capa de vista y la capa de modelo, además, controla las interacciones y filtra la información dependiendo de lo que solicita el usuario.

En la **Fig. 1** se expone el modelo arquitectónico que han sido utilizadas para la creación del *backend*.

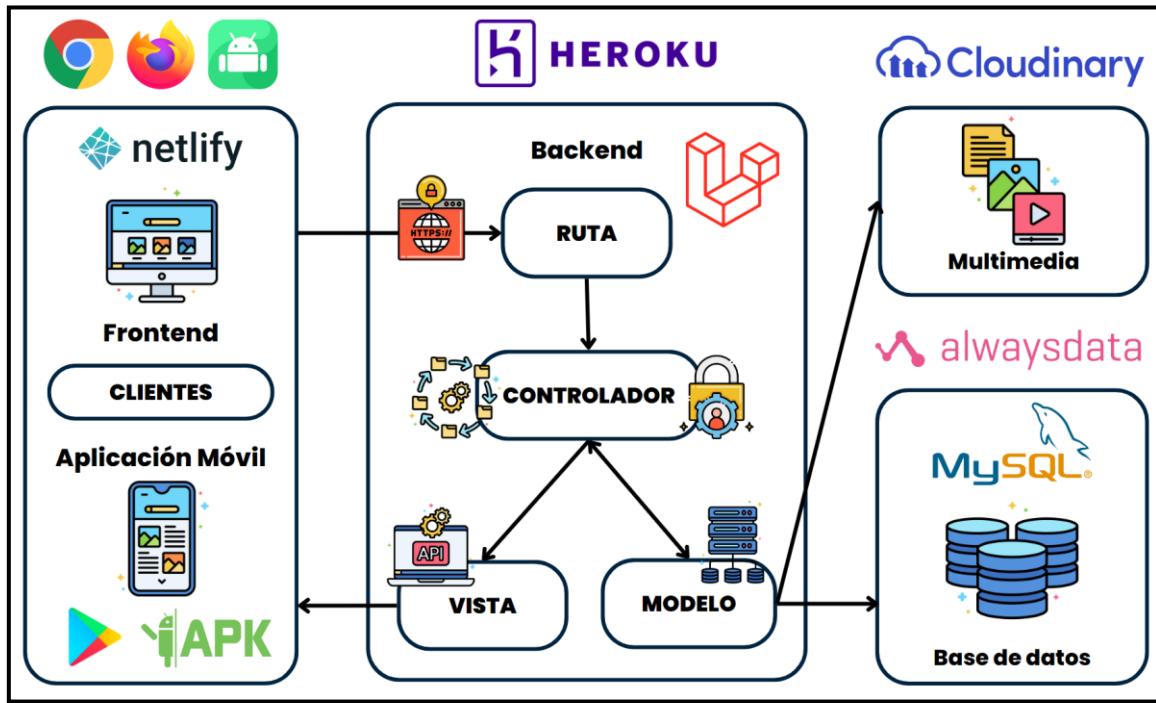


Fig. 1: Patrón arquitectónico.

2.3 Herramientas de desarrollo

En la **TABLA VI** se exponen las herramientas utilizadas para el desarrollo del *backend*, conjuntamente con los principales aportes que han otorgado en el desarrollo de cada uno de los *endpoints*.

TABLA VI: Herramientas para el desarrollo del *backend*.

HERRAMIENTAS	JUSTIFICACIÓN
Laravel	Es el <i>Framework</i> más completo para el perfeccionamiento de aplicaciones del lado del servidor y en este caso se utiliza para construir el <i>backend</i> [12].
Composer	Es un gestor de librerías para PHP, admite reutilizar paquetes que maneja la comunidad para la construcción de proyectos [24].

Alwaysdata	Es una plataforma que permite desplegar base de datos MySQL en la nube, además de poseer otros servicios parecidos [25].
Heroku	Es una plataforma para desplegar aplicaciones web a producción de forma gratuita [26].
Cloudinary	Es una plataforma para alojar archivos multimedia en la nube y admite el acceso sincrónico a la información [27].
Visual Studio Code	Es un editor de código que contiene extensiones y <i>plugins</i> para una mejor experiencia de codificación [28].
Apache jmeter	Es una plataforma para implementar pruebas de carga, rendimiento y estrés a recursos API Rest [29].

Librerías

En la **TABLA VII** se exponen las librerías utilizadas para el desarrollo del *backend*, conjuntamente con los principales aportes que han otorgado en el desarrollo de cada uno de los *endpoints*.

TABLA VII: librerías.

LIBRERÍAS	JUSTIFICACIÓN
cloudinary-laravel	Ofrece el aporte de conectar, optimizar y cargar archivos multimedia a través de los modelos de Laravel Eloquent a la nube [30].
laravel-lang	Optimiza el proceso de implementación de gestores de traducción, admitiendo el desarrollo de aplicaciones multilingüe [31].
sanctum	Aporta de un procedimiento de autenticación y protección de rutas basados en <i>tokens</i> [32].
faker	Generador de datos falsos optimizados para el <i>testing</i> de bases de datos [33].

3 RESULTADOS

En este apartado, se evidencia los resultados que se han completado durante el desarrollo del *backend*. Donde se detalla el diseño, codificación y la implementación de los *endpoints* indispensables para cada rol de usuario, teniendo en cuenta las respectivas pruebas y por último el despliegue a producción. Además, es importante recalcar que por cada *Sprint* se presentan las tareas que han sido fundamentales para cumplir dicha iteración.

3.1 Sprint 0. Configuración del ambiente de desarrollo

Este *Sprint* se compone de tareas indispensables para dejar listo el entorno de desarrollo, las cuales son:

- Definición de requerimientos y restricciones para el *backend*.
- Diseño e implementación de tablas y relaciones para la Base de datos SQL.
- Definición de archivos, directorios y configuraciones para el proyecto *backend*.
- Definición de módulos por cada rol de usuario.

Definición de requerimientos y restricciones para el *backend*.

Diseño e implementación de *endpoints* para el inicio de sesión, cierre de sesión y cambio de contraseña

El componente *backend* concede a los usuarios con perfil administrador, técnico y cliente ingresar al módulo de autenticación, para lo cual disponen de varios *endpoints* para el inicio sesión, cierre de sesión y cambio de contraseña. Además, la implementación de estos *endpoints* se divide en dos capas, la primera capa admite a los usuarios con perfil administrador y técnico acceder al módulo de autenticación a través de una serie de API's las cuales son consumidas por un *frontend* y la segunda capa permite al usuario con perfil cliente acceder al módulo de autenticación con otra serie de API's que son consumidas por una aplicación móvil.

Diseño e implementación de *endpoints* para visualizar y editar el perfil de usuario

El componente *backend* concede a los usuarios con perfil administrador, técnico y cliente acceder a su perfil de usuario, para lo cual disponen de varios *endpoints* que permiten a los usuarios visualizar y editar la información personal y el avatar.

Diseño e implementación de *endpoints* para gestionar solicitudes de afiliación

El componente *backend* concede al usuario con perfil administrador ingresar al módulo de gestión de solicitudes de afiliación, para lo cual disponen de varios *endpoints* que permite al administrador visualizar, aceptar y/o rechazar las solicitudes de afiliación que han sido realizadas y enviadas anticipadamente por los técnicos.

Diseño e implementación de *endpoints* para visualizar comentarios, sugerencias y calificación de técnicos

El componente *backend* concede al usuario con perfil administrador ingresar a un módulo que le permita visualizar los comentarios, sugerencias y calificación de los técnicos. Además, el módulo dispone de varios *endpoints* que le permiten al administrador activar y desactivar a los técnicos, por otra parte, a través de otros *endpoints* le permiten notificar al técnico en caso de que haya sido desactivado.

Diseño e implementación de *endpoints* para que los usuarios con perfil técnico y cliente puedan registrarse

El componente *backend* concede al usuario con perfil técnico y cliente acceder al módulo de registro, para lo cual disponen de varios *endpoints* que permiten al usuario con perfil técnico disponer de un método de registro distinto del usuario con perfil cliente, con el fin de evitar problemas en la asignación de roles.

Diseño e implementación de *endpoints* para solicitar afiliación

El componente *backend* concede al usuario con perfil técnico acceder al módulo de solicitud de afiliación, para lo cual disponen de varios *endpoints* que le permiten al técnico solicitar una afiliación con el ingreso de su información personal y laboral.

Diseño e implementación de *endpoints* para gestionar servicios

El componente *backend* concede al usuario con perfil técnico acceder al módulo de gestión de servicios, para lo cual disponen de varios *endpoints* que le permiten al técnico crear, visualizar, actualizar y eliminar servicios.

Diseño e implementación de *endpoints* para aprobar contrataciones

El componente *backend* concede al usuario con perfil técnico acceder al módulo de aprobación de contratos, para lo cual disponen de varios *endpoints* que le permiten al técnico visualizar las solicitudes de contrato enviadas previamente por el usuario con perfil cliente, para luego aprobar y/o rechazar las solitudes de contrato.

Diseño e implementación de *endpoints* para visualizar comentarios, sugerencias y calificación de los servicios

El componente *backend* concede al usuario con perfil técnico varios *endpoints* que le permitan visualizar comentarios, sugerencias y calificación de los servicios que ha realizado.

Diseño e implementación de *endpoints* para visualizar servicios

El componente *backend* concede al usuario con perfil cliente acceder al módulo de servicio, para lo cual disponen de varios *endpoints* que le permiten al cliente visualizar los servicios ofertados por los usuarios con perfil técnico. Estos *endpoints* tienen la peculiaridad de que son de acceso libre, a lo que se refiere es que el usuario no necesita iniciar sesión o estar registrado para visualizar dichos servicios.

Diseño e implementación de *endpoints* para contratar servicios

El componente *backend* concede al usuario con perfil cliente acceder al módulo de contratación de servicios, para lo cual disponen de varios *endpoints* que le permiten al cliente contratar servicios a través del ingreso de datos informativos sobre el dispositivo y los problemas que presenta.

Diseño e implementación de *endpoints* para gestionar solicitudes de contratación

El componente *backend* concede al usuario con perfil cliente acceder al módulo de servicios, para lo cual disponen de varios *endpoints* que le permiten al cliente visualizar los servicios que ha contratado y actualizar los parámetros de la contratación del servicio al igual de cancelar el servicio si lo requiere.

Diseño e implementación de *endpoints* para comentar, sugerir y calificar los servicios

El componente *backend* concede al usuario con perfil cliente varios *endpoints* que le permitan comentar, sugerir y/o calificar los servicios que ha contratado, siempre y cuando la contratación del servicio haya sido terminada.

Seguidamente, en la **Fig. 2**, **Fig. 3** y **Fig. 4** se expone los perfiles de usuarios y los *endpoints* a los que poseen acceso.

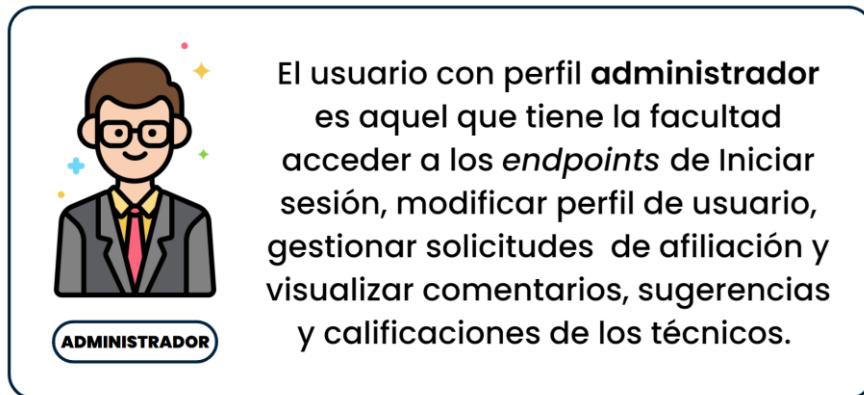


Fig. 2: Usuario Administrador.

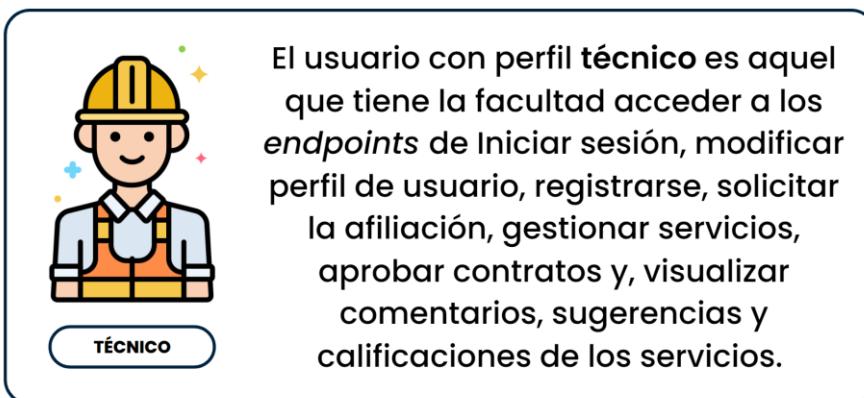


Fig. 3: Usuario Técnico.

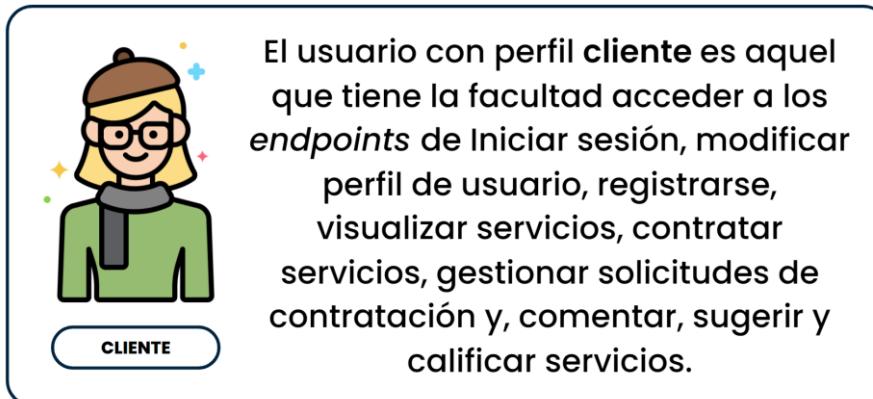


Fig. 4: Usuario Cliente.

Diseño e implementación de tablas y relaciones para la Base de datos SQL.

Para el desarrollo de este componente se ha seleccionado implementarlo en MySQL, el cual se encuentra desplegado en la plataforma alwaysdata, permitiendo de esta manera alojar la información sobre los usuarios y servicios. En la **Fig. 5**, se muestra a cada una de

las entidades de forma general que han sido determinadas, mientras que las relaciones entre entidades se las puede apreciar de mejor manera en el **ANEXO II**.

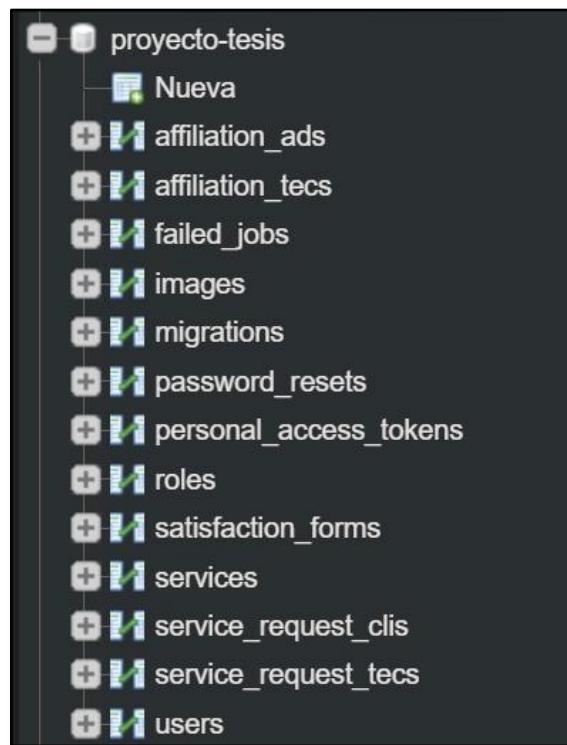


Fig. 5: Entidades de la Base de datos.

Definición de archivos, directorios y configuraciones para el proyecto *backend*

Visual Studio Code es la principal herramienta de codificación utilizada para el desarrollo de los *endpoints* que se han determinado para el *backend*. Además, esta herramienta conjuntamente con librerías y carpetas provista por el *framework Laravel* presenta una estructura ordenada y de fácil seguimiento. Muestra de ello, la **Fig. 6** presenta la estructura completa del proyecto.

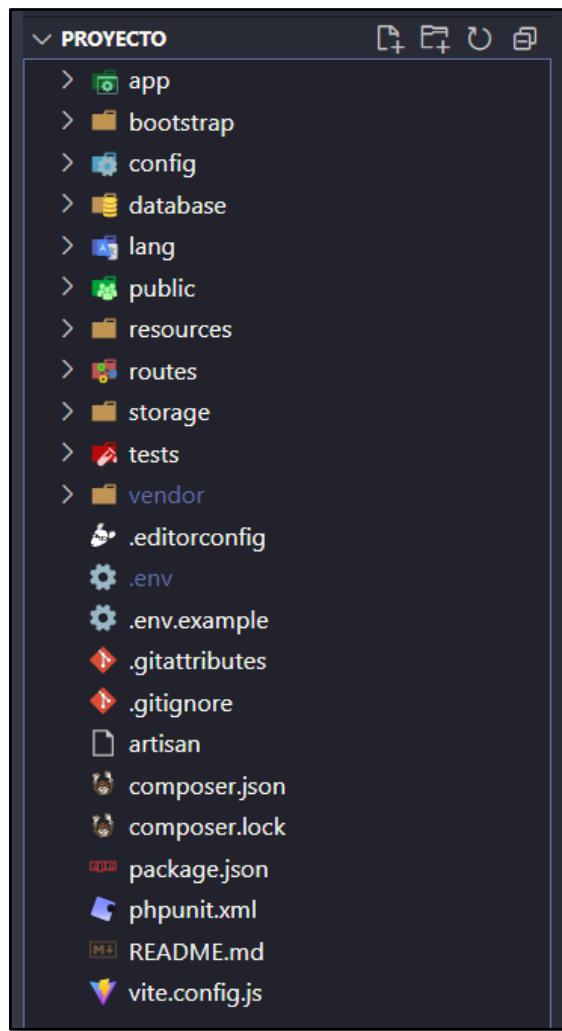


Fig. 6. Estructura del proyecto.

Definición de módulos por cada rol de usuario

En la **Fig. 7** se presenta los módulos a los cuales los usuarios con perfil administrador, técnico y cliente gozan de acceso después de iniciar sesión respectivamente en el *backend*.

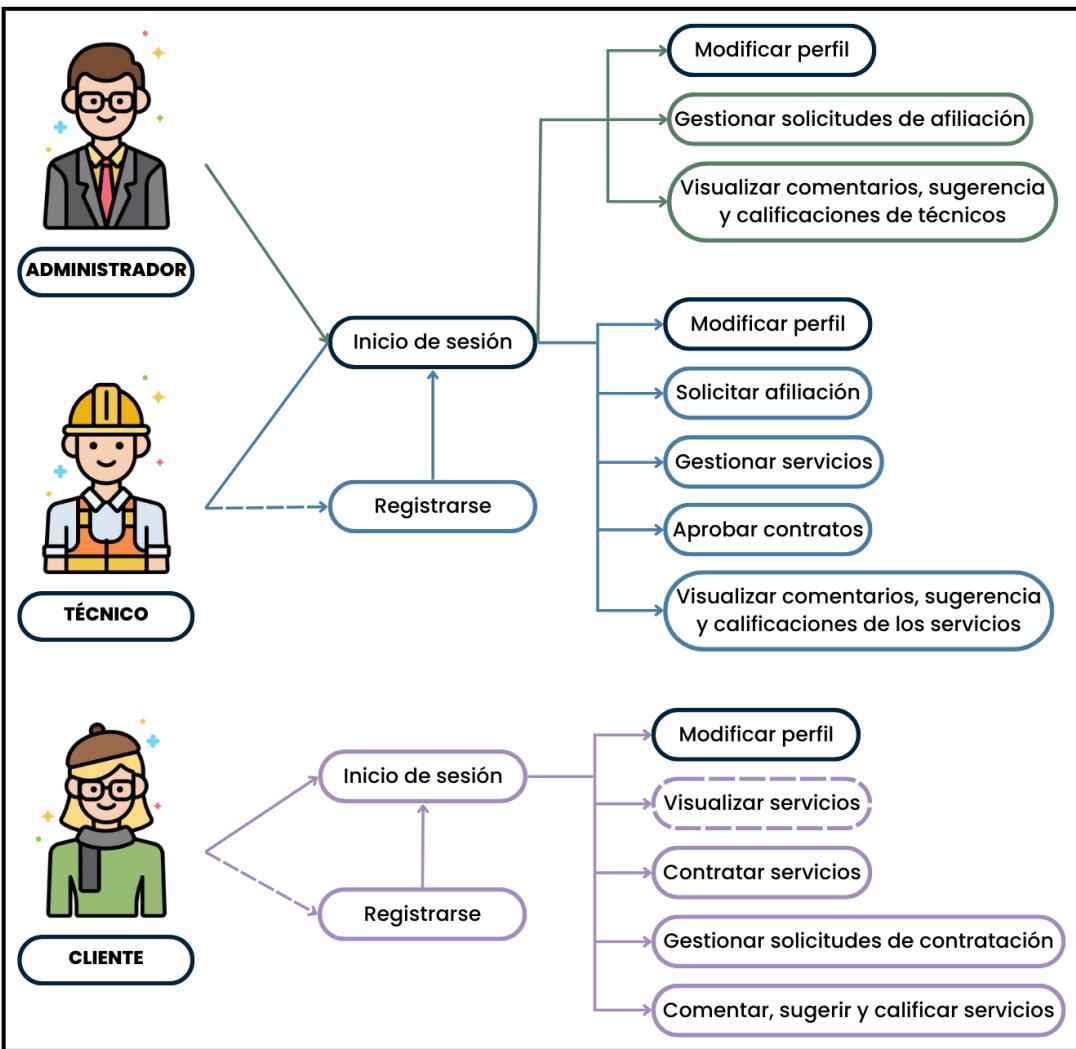


Fig. 7: Definición de módulos para los usuarios.

3.2 *Sprint 1. Perfil administrador – implementación de endpoints.*

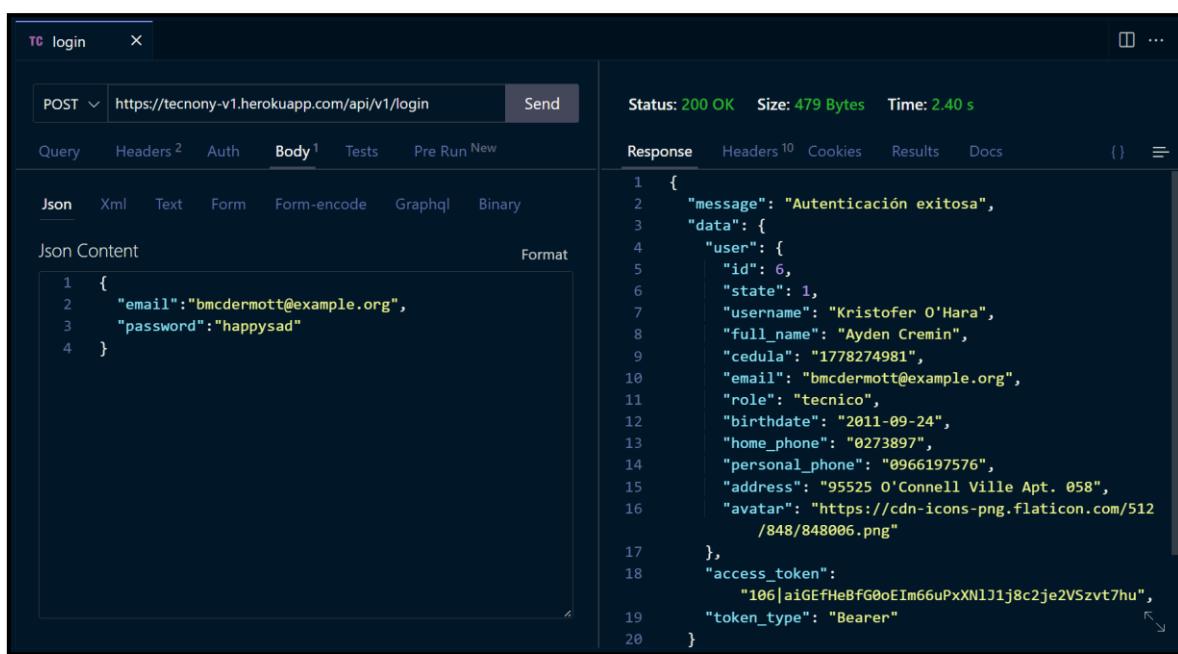
Este *Sprint* se compone de tareas indispensables para la codificación de cada uno de los *endpoints* del usuario administrador, los cuales son:

- Inicio de sesión, cierre de sesión y recuperación de contraseña.
- Modificación de perfil de usuario.
- Gestión de solicitudes de afiliación.
- Visualización de comentarios, sugerencias y calificación de los técnicos.

Inicio de sesión, cierre de sesión y recuperación de contraseña

El *backend* implementa diversos *endpoints* consignados para el inicio de sesión, cierre de sesión y recuperación de contraseñas, destinados para todos los usuarios de este

proyecto. Además, el *endpoint* para el *login* admite únicamente usuarios con perfil administrador y técnico, solicitando email y contraseña para ingresar a los módulos asignados como se observa en la **Fig. 8**. Mientras que, el *endpoint* para la recuperación de contraseña solicita email del usuario para él envió de un correo electrónico para restablecer la contraseña, como se observa en la **Fig. 9**. Por último, el *endpoint* para el cierre de sesión permite al usuario salir del sistema cuando lo deseé como se observa en la **Fig. 10**. Por último, en la **Fig. 11** se observa el código efectuado de una prueba unitaria aplicada al *endpoint* de login, junto a los resultados que se observa en la **Fig. 12**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.



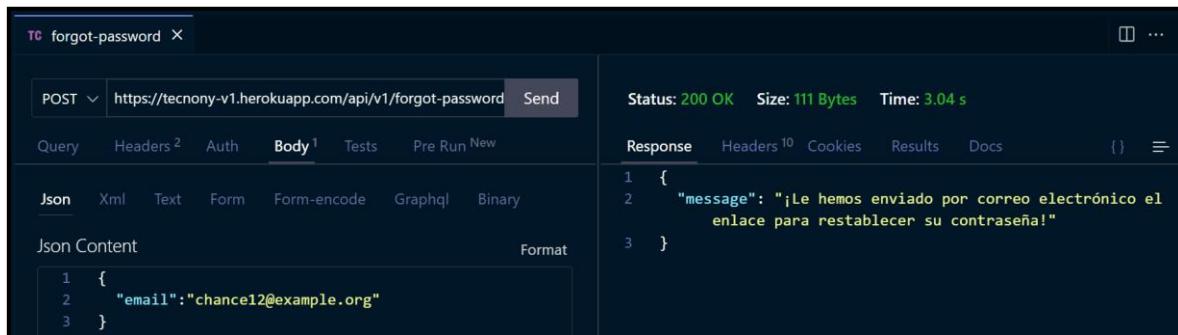
The screenshot shows a Postman request for 'TC login' with a POST method to 'https://tecnony-v1.herokuapp.com/api/v1/login'. The request body is JSON with fields 'email' and 'password'. The response status is 200 OK, size is 479 Bytes, and time is 2.40 s. The response body is a JSON object containing a message, user data, and an access token.

```

1 {
2   "message": "Autenticación exitosa",
3   "data": {
4     "user": {
5       "id": 6,
6       "state": 1,
7       "username": "Kristofer O'Hara",
8       "full_name": "Ayden Cremin",
9       "cedula": "1778274981",
10      "email": "bmcdermott@example.org",
11      "role": "técnico",
12      "birthdate": "2011-09-24",
13      "home_phone": "0273897",
14      "personal_phone": "0966197576",
15      "address": "95525 O'Connell Ville Apt. 058",
16      "avatar": "https://cdn-icons-png.flaticon.com/512/848/848006.png"
17    },
18    "access_token": "106|aiGEfHeBfg0oEIm66uPxXN1J1j8c2je2VSzvt7hu",
19    "token_type": "Bearer"
20  }

```

Fig. 8: Login para administrador y técnico.



The screenshot shows a Postman request for 'TC forgot-password' with a POST method to 'https://tecnony-v1.herokuapp.com/api/v1/forgot-password'. The request body is JSON with a single 'email' field. The response status is 200 OK, size is 111 Bytes, and time is 3.04 s. The response body is a JSON object with a message indicating a password recovery link has been sent.

```

1 {
2   "message": "¡Le hemos enviado por correo electrónico el enlace para restablecer su contraseña!"
3 }

```

Fig. 9: Recuperación de contraseña.

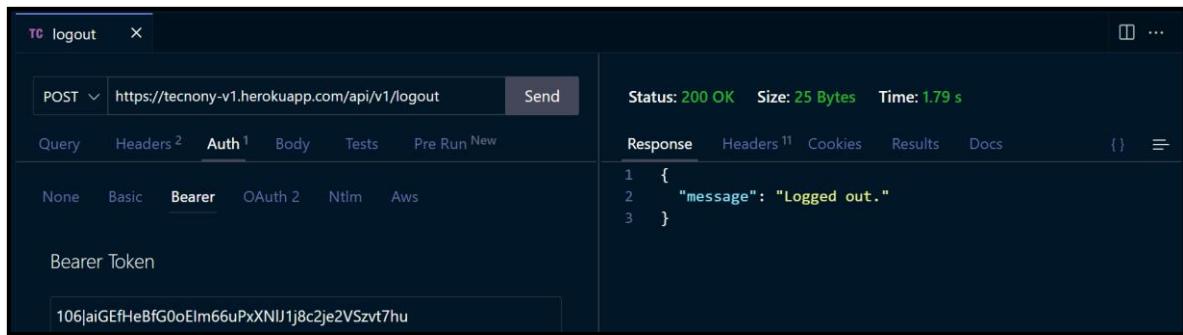


Fig. 10: Logout.

```

// ● 1. Iniciar sesión, cerrar sesión y recuperar contraseña
// - Iniciar sesión
public function test_inicio_de_sesion()
{
    $test_request = $this->post('/api/v1/login', [
        "email" => "chance12@example.org",
        "password" => "happySad1*"
    ]);
    $test_request->assertStatus(200);
}

```

Fig. 11: Test de inicio de sesión.

```

PASS Tests\Unit\S1AdminTest
✓ inicio de sesión

Tests: 1 passed
Time: 4.43s

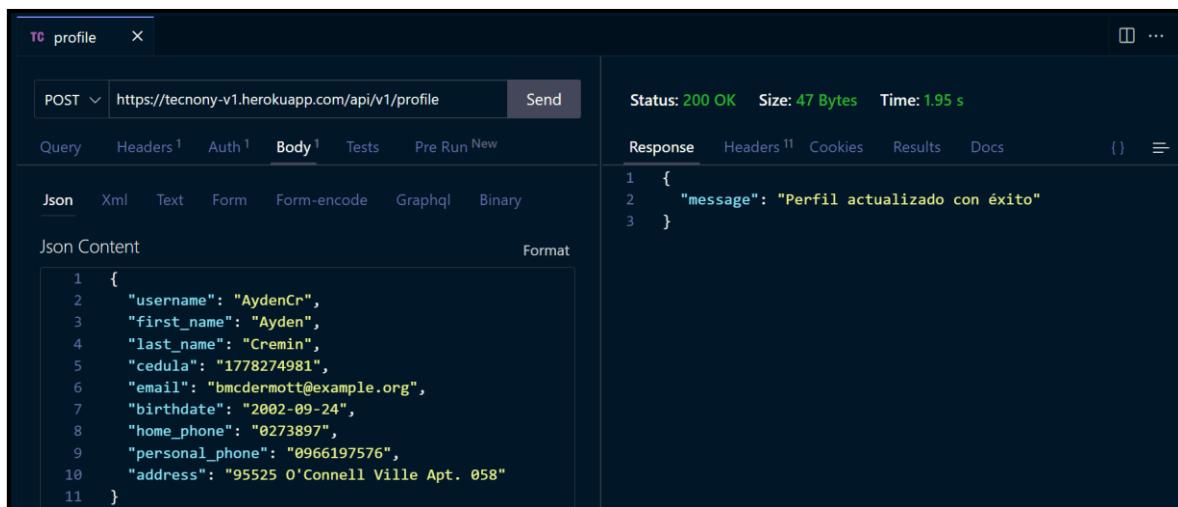
```

Fig. 12: Resultados de la prueba.

Modificación de perfil de usuario

El *backend* efectúa la implementación de varios *endpoints* consignados para la modificación de perfil de usuario, destinados para todos los usuarios de este proyecto. Además, el *endpoint* para modificar el perfil de usuario admite el cambio de datos personales del usuario, como se observa en la **Fig. 13**. Además, existe un *endpoint* que admite cambiar el avatar de usuario, como se observa en la **Fig. 14**. Por último, en la **Fig. 15** se observa el código efectuado de una prueba unitaria aplicada al *endpoint* de modificar

el perfil de usuario, junto a los resultados que se observa en la **Fig. 16**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.



The screenshot shows a Postman request for the endpoint `https://tecnony-v1.herokuapp.com/api/v1/profile`. The method is set to `POST`. The request body is in `JSON` format, containing the following data:

```

1  {
2    "username": "AydenCr",
3    "first_name": "Ayden",
4    "last_name": "Cremin",
5    "cedula": "1778274981",
6    "email": "bmcdermott@example.org",
7    "birthdate": "2002-09-24",
8    "home_phone": "0273897",
9    "personal_phone": "0966197576",
10   "address": "95525 O'Connell Ville Apt. 058"
11 }

```

The response status is `200 OK`, size is `47 Bytes`, and time is `1.95 s`. The response body is:

```

1  {
2    "message": "Perfil actualizado con éxito"
3  }

```

Fig. 13: Modificar perfil de usuario.

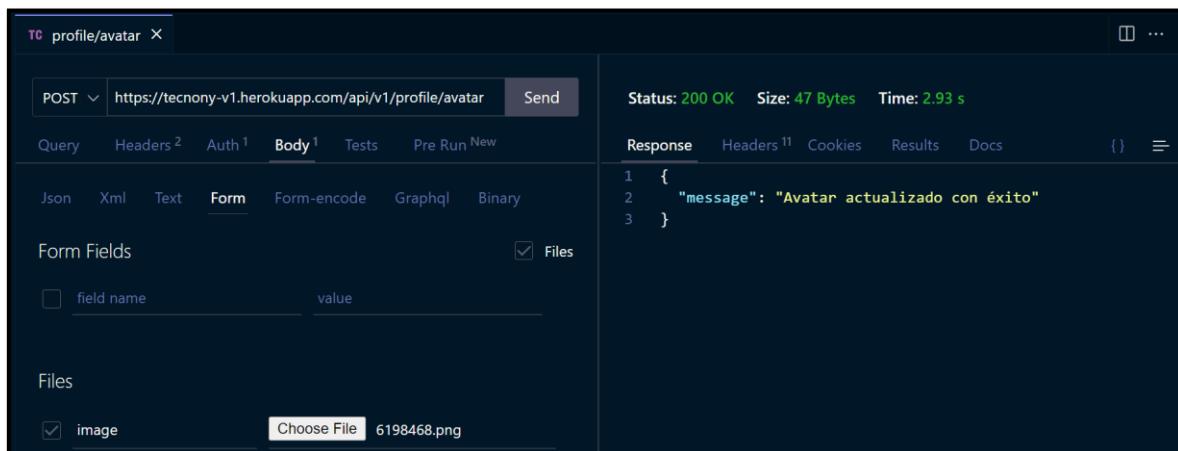


Fig. 14: Modificar avatar de usuario.

```

// ● 2. Modificar perfil de usuario
// - Modificar perfil
public function test_modificacion_de_perfil()
{
    $user = User::factory()->make(['role_id' => 2]);
    $profile = [
        "username" => "Juanca13",
        "first_name" => "Juan",
        "last_name" => "Carol",
        "cedula" => "1694698942",
        "email" => "juanca13@gmail.com",
        "birthdate" => "1999-08-15",
        "home_phone" => "0236379",
        "personal_phone" => "0992896598",
        "address" => "453 Guamani. 7"
    ];
    $test_request = $this->actingAs($user)->post('/api/v1/profile/', $profile);
    $test_request->assertStatus(200);
}

```

Fig. 15: Test de modificar perfil de usuario.

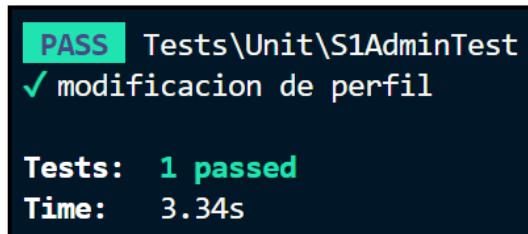


Fig. 16: Test de modificar perfil de usuario.

Gestión de solicitudes de afiliación

El *backend* efectúa la implementación de varios *endpoints* consignados para la gestión de afiliación. En ese sentido, existe un *endpoint* para la visualización de solicitudes de afiliaciones de los técnicos, como se muestra en la **Fig. 17**. El siguiente *endpoint* admite visualizar la solicitud de un técnico seleccionado, como se observa en la **Fig. 18**. Y el último *endpoint* admite aceptar o rechazar la solicitud de afiliación del técnico que se ha seleccionado, además el mismo *endpoint* permite escribir una observación para notificar al técnico si la solicitud de afiliación es rechazada, como se observa en la **Fig. 19**. Por último, en la **Fig. 20** se observa el código efectuado de una prueba unitaria aplicada al *endpoint* de visualización de solicitudes de afiliaciones de los técnicos, junto a los resultados que se observa en la **Fig. 21**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.

```

Status: 200 OK  Size: 1.51 KB  Time: 3.04 s
Response Headers [1] Cookies Results Docs
1 {
2   "message": "La lista de afiliados ha sido generada con éxito",
3   "data": [
4     "affiliations": [
5       {
6         "id": 10,
7         "state": 1,
8         "date_issue": "2022-12-15",
9         "full_name": "Juan Loja",
10        "profession": "Ingeniero en sistemas",
11        "specialization": "Informatica basica",
12        "work_phone": "0988675635",
13        "attention_schedule": "De 10 de la mañana hasta las 8 de la noche",
14        "local_name": "Los Tecnicos de aqui abajo",
15        "local_address": "67 chillogallo Oval Apt. 7",
16        "confirmation": 1,
17        "account_number": "4427061886",
18        "account_type": "Corriente",
19        "banking_entity": "BANCO DEL PACIFICO",
20        "documento": "No tiene archivo"
21      },
22      {
23        "id": 14,
24        "state": 1,
25        "date_issue": "2023-01-03",
26        "full_name": "Hector Perez",
27        "profession": "Ingeniero en sistemas",
28        "specialization": "Informatica basica",

```

Fig. 17: Visualizar solicitudes de afiliación.

```

Status: 200 OK  Size: 951 Bytes  Time: 5.29 s
Response Headers [1] Cookies Results Docs
1 {
2   "message": "La solicitud de afiliación fue devuelta correctamente",
3   "data": [
4     "affiliations": [
5       {
6         "id": 27,
7         "state": 1,
8         "date_issue": "2023-01-16",
9         "full_name": "Josue Perez",
10        "profession": "Ingeniero en sistemas",
11        "specialization": "En sistemas informaticos",
12        "work_phone": "0988675949",
13        "attention_schedule": "De 8 de la mañana hasta las 4 de la tarde",
14        "local_name": "Marquitos",
15        "local_address": "Chilibulo",
16        "confirmation": 1,
17        "account_number": "2974836748",
18        "account_type": "Corriente",
19        "banking_entity": "BANCO DEL PICHINCHA",
20        "documento": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1673879390/xms6stbd3fjdp4vhjxy.pdf"
21      },
22      "create_by": {
23        "id": 56,
24        "state": 1,
25        "username": "Josue24",
26        "first_name": "Josue",
27        "last_name": "Perez",
28        "cedula": "1722398942",

```

Fig. 18: Visualizar solicitud de afiliación seleccionado.

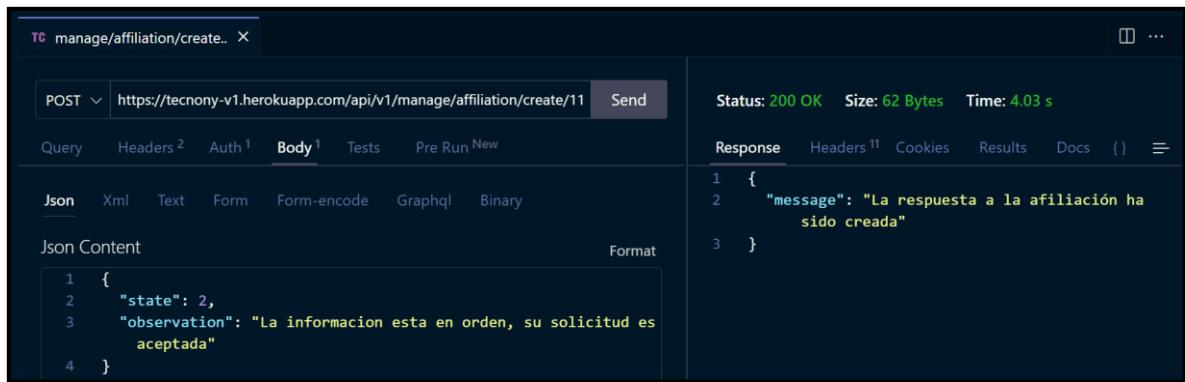


Fig. 19: Aceptación de solicitud de afiliación.

```

// 3. Gestión de solicitudes de afiliación
// - Visualizar solicitudes de afiliacion atendidas
public function test_visualizacion_de_solicitudes_de_afiliacion()
{
    $user = User::where('id', 1)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/manage/affiliations');
    $test_request->assertStatus(200);
}

```

Fig. 20: Test de visualizar solicitudes de afiliación.

```

PASS Tests\Unit\S1AdminTest
✓ visualizacion de solicitudes de afiliacion

Tests: 1 passed
Time: 5.68s

```

Fig. 21: Resultados de la prueba.

Visualización de comentarios, sugerencias y calificación de los técnicos

El *backend* efectúa la implementación de varios *endpoints* consignados para la visualización de comentarios, sugerencias y calificaciones de los técnicos. En ese sentido, existe un *endpoint* para la visualización de todos los técnicos, como se muestra en la **Fig. 22**. El siguiente *endpoint* admite visualizar los comentarios, sugerencias y calificaciones generadas por los formularios de satisfacción de las solicitudes de contratación atendidas por un técnico que se haya seleccionado, como se observa en la **Fig. 23**. Y el último *endpoint* admite activar o desactivar a un técnico que se ha seleccionado, además el mismo

endpoint permite escribir una observación para notificar al técnico que ha sido desactivado, como se observa en la **Fig. 24**. Por último, en la **Fig. 25** se observa el código efectuado de una prueba unitaria aplicada al *endpoint* de visualización de comentarios de un técnico seleccionado, junto a los resultados que se observa en la **Fig. 26**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III.**

```

1 {
2   "message": "Lista de técnicos generada con éxito",
3   "data": {
4     "users": [
5       {
6         "id": 6,
7         "state": 0,
8         "username": "AydenCr",
9         "full_name": "Ayden Cremin",
10        "cedula": "1778274981",
11        "email": "bmcdermott@example.org",
12        "role": "técnico",
13        "birthdate": "2002-09-24",
14        "home_phone": "0273897",
15        "personal_phone": "0966197576",
16        "address": "95525 O'Connell Ville Apt. 058",
17        "avatar": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1672105480/rfisxkezouainvzcryda.png"
18      },
19      {
20        "id": 7,
21      }
22    ]
23  }
  
```

Fig. 22: Visualización de técnicos.

```

1 {
2   "message": "La puntuación y los comentarios del técnico se ha devuelto con éxito",
3   "data": {
4     "score": 7.32,
5     "technical": {
6       "id": 6,
7       "state": 1,
8       "username": "AydenCr",
9       "first_name": "Ayden",
10      "last_name": "Cremin",
11      "cedula": "1778274981",
12      "email": "bmcdermott@example.org",
13      "birthdate": "2002-09-24",
14      "home_phone": "0273897",
15      "personal_phone": "0966197576",
16      "address": "95525 O'Connell Ville Apt. 058",
17      "avatar": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1672105480/rfisxkezouainvzcryda.png"
18    },
19    "satisfaction_forms": [
20      {
21        "id": 1,
22        "comment": "Dicta amet veritatis possimus quia minus. Quidem molestiae repudiandae aperiam asperiores reprehenderit beatae temporibus.",
23        "suggestion": "Qui ut que nisi labore voluptatibus tempora Officiis minus nisi sed fugiat repellat sed."
24      }
25    ]
26  }
  
```

Fig. 23: Visualización de comentarios de un técnico seleccionado.

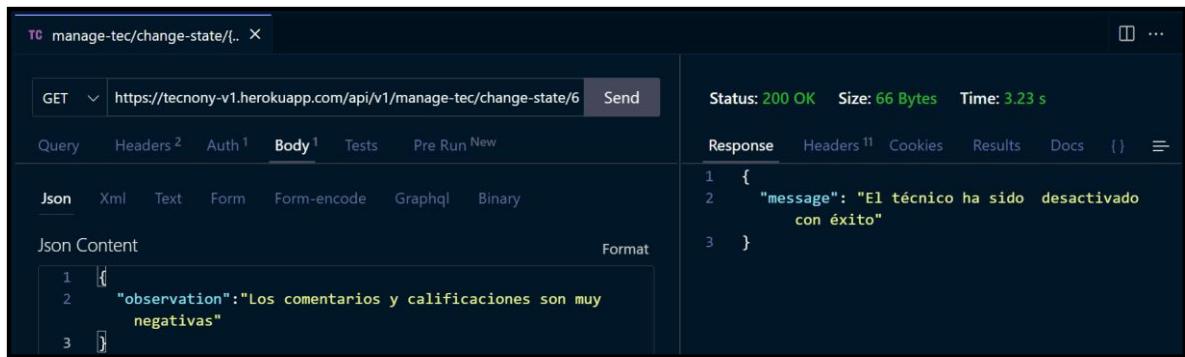


Fig. 24: Activación o desactivación de técnico.

```
// ● 4. Visualización de comentarios, sugerencias y calificación de los técnicos
// - Visualizar comentarios de un tecnico
public function test_visualizacion_de_comentarios_de_un_tecnico()
{
    $user = User::where('id', 1)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/manage-tec/show/%u', 6));
    $test_request->assertStatus(200);
}
```

Fig. 25: Test de visualización de comentarios de un técnico seleccionado.

```
PASS Tests\Unit\S1AdminTest
✓ visualizacion de comentarios de un tecnico

Tests: 1 passed
Time: 15.50s
```

Fig. 26: Resultados de la prueba.

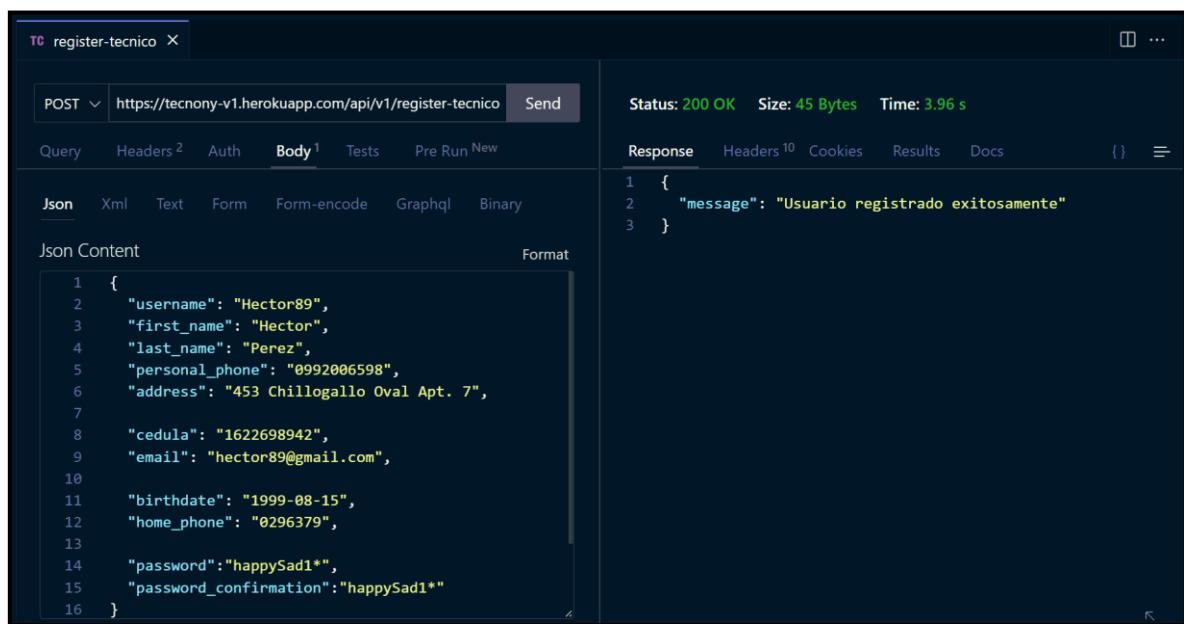
3.3 Sprint 2. Perfil técnico – implementación de endpoints.

Este *Sprint* se compone de tareas indispensables para la codificación de cada uno de los *endpoints* del usuario técnico, los cuales son:

- Registro de usuario para el perfil técnico.
- Solicitud de afiliación.
- Gestión de servicios.
- Aprobación de contratos.
- Visualización de comentarios, sugerencias y calificación de servicios.

Registro de usuario para el perfil técnico

El *backend* efectúa la implementación de un *endpoint* consignado para el registro de usuarios con perfil técnico. En ese sentido, el *endpoint* para el registro de usuario admite el ingreso de datos personales los cuales son validados para la creación de un nuevo usuario técnico en el *backend* como se muestra en la **Fig. 27**. Por último, en la **Fig. 28** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de registro de usuario técnico, junto a los resultados que se observa en la **Fig. 29**. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el **ANEXO III**.



The screenshot shows a Postman interface with the following details:

- Request URL:** https://tecnony-v1.herokuapp.com/api/v1/register-tecnico
- Method:** POST
- Body (JSON Content):**

```
1 {
2   "username": "Hector89",
3   "first_name": "Hector",
4   "last_name": "Perez",
5   "personal_phone": "0992006598",
6   "address": "453 Chillogallo Oval Apt. 7",
7
8   "cedula": "1622698942",
9   "email": "hector89@gmail.com",
10
11  "birthdate": "1999-08-15",
12  "home_phone": "0296379",
13
14  "password": "happySad1*",
15  "password_confirmation": "happySad1*"
16 }
```
- Response Status:** 200 OK
- Response Headers:** Size: 45 Bytes
- Response Body:**

```
1 {
2   "message": "Usuario registrado exitosamente"
3 }
```

Fig. 27: Registro de usuario técnico.

```

// ● 5. Registro de usuario para el perfil técnico
// - Registrar usuario técnico
public function test_registro_de_tecnico()
{
    $test_request = $this->post('/api/v1/register-tecnico', [
        "username" => "Hector89",
        "first_name" => "Hector",
        "last_name" => "Perez",
        "personal_phone" => "0992006598",
        "address" => "453 Chillogallo Oval Apt. 7",

        "cedula" => "1622698942",
        "email" => "hector89@gmail.com",

        "birthdate" => "1999-08-15",
        "home_phone" => "0296379",

        "password" => "happySad1*",
        "password_confirmation" => "happySad1*"
    ]);
    $test_request->assertStatus(200);
}

```

Fig. 28: Test de registro de técnico.

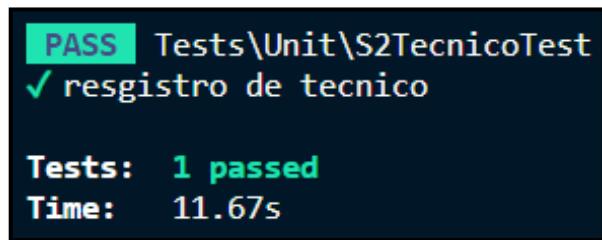


Fig. 29: Resultados de la prueba.

Solicitud de afiliación

El *backend* efectúa la implementación de un *endpoint* consignado para la solicitud de afiliación. En ese sentido, el *endpoint* solicitud de afiliación admite el ingreso de datos personales y laborales los cuales son validados para la aceptación o rechazo de un nuevo usuario técnico en el *backend*, como se muestra en la **Fig. 30**. Por último, en la **Fig. 31** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de solicitud de

afiliación, junto a los resultados que se observa en la **Fig. 32**. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el **ANEXO III**.

Fig. 30: Solicitud de afiliación.

```
// 6. Solicitud de afiliación.
// - Solicitud de afiliación
public function test_crear_solicitud_de_afiliacion()
{
    $user = User::where('id', 59)->first();
    $affiliation = [
        "profession" => "Ingeniero en sistemas",
        "specialization" => "Informatica computacional",
        "work_phone" => "0983657935",
        "attention_schedule" => "De 8 de la mañana hasta las 3 de la noche",
        "local_name" => "CompuLed",
        "local_address" => "67 Guamani Oval Apt. 7",
        "confirmation" => true,

        'account_number' => 2238491478,
        'account_type' => "Ahorros",
        'banking_entity' => "BANCO DE PICHINCHA"
    ];
    $test_request = $this->actingAs($user)->post('/api/v1/affiliation/create', $affiliation);
    $test_request->assertStatus(200);
}
```

Fig. 31: Test de solicitud de afiliación.

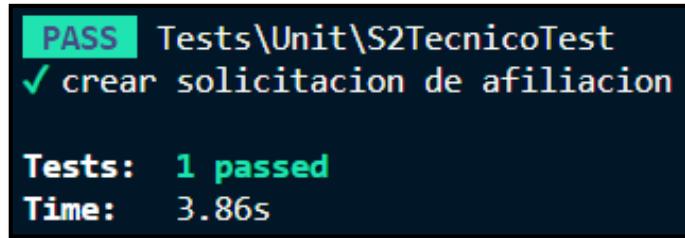


Fig. 32: Resultados de la prueba.

Gestión de servicios

El *backend* efectúa la implementación de varios *endpoints* consignados para la gestión de servicios. En ese sentido, existe un *endpoint* para la visualización de los servicios generados por el técnico, como se muestra en la **Fig. 33**. El siguiente *endpoint* admite la creación de un nuevo servicio a través del ingreso de un título, descripción y una imagen, como se observa en la **Fig. 34**. Y el último *endpoint* permite habilitar o inhabilitar un servicio, como se observa en la **Fig. 35**. Por último, en la **Fig. 36** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de visualización de servicios generados, junto a los resultados que se observa en la **Fig. 37**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.

TC service X

GET https://tecnony-v1.herokuapp.com/api/v1/service Send

Query Headers² Auth¹ Body Tests Pre Run New

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 3.14 KB Time: 2.98 s

Response Headers¹¹ Cookies Results Docs

```
1  {
2      "message": "Lista de servicios generada con éxito",
3      "data": {
4          "services": [
5              {
6                  "id": 20,
7                  "state": 0,
8                  "name": "Reparacion de computadoras",
9                  "categories": "reparación",
10                 "description": "Se repara computadoras",
11                 "price": 5.5,
12                 "image": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1670621120
13                     /opngjv5nwlsrwfolobn5.jpg",
14                 "attention_mode": 1,
15                 "attention_description": "In mollitia nihil quis. A ad sint repudiandae corrupti
16                     ipsam labore. Magni eos ipsum quaerat sequi iure quasi. Dolores sint
17                     voluptatibus aut quis earum aperiam quidem. Quibusdam sed et iusto ea.",
18                 "payment_method": 1,
19                 "payment_description": "In mollitia nihil quis. A ad sint repudiandae corrupti
20                     ipsam labore. Magni eos ipsum quaerat sequi iure quasi. Dolores sint
21                     voluptatibus aut quis earum aperiam quidem. Quibusdam sed et iusto ea.",
22                 "created_at": "2022-12-09 21:25:18"
23             },
24             {
25                 "id": 21,
26                 "state": 1,
27                 "name": "Instalación Microsoft office",
28                 "categories": "software",
29             }
30         ]
31     }
32 }
```

Fig. 33: Visualización de servicios generados.

Fig. 34: Creación de un servicio.

Fig. 35: Activación o desactivación de un servicio.

```
// 7. Gestión de servicios
// - Visualizar servicios generados
public function test_visualizacion_de_servicios_generados()
{
    $user = User::where('id', 7)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/service');
    $test_request->assertStatus(200);
}
```

Fig. 36: Test de visualización de servicios generados.

```

PASS Tests\Unit\S2TecnicoTest
✓ visualización de servicios generados

Tests: 1 passed
Time: 5.69s

```

Fig. 37: Resultados de la prueba.

Aprobación de contratos

El *backend* efectúa la implementación de varios *endpoints* consignados para la aprobación de contratos. En ese sentido, existe un *endpoint* para visualizar las solicitudes de contratación hechas por los usuarios clientes, como se muestra en la **Fig. 38**. El siguiente *endpoint* permite visualizar la información detallada de una solicitud de contratación seleccionada, junto a la información personal del cliente quien lo ha realizado, como se observa en la **Fig. 39**. Y el último *endpoint* permite aprobar la solicitud de contratación, como se observa en la **Fig. 40**. Por último, en la **Fig. 41** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de aprobación de contratos, junto a los resultados que se observa en la **Fig. 42**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.

```

Status: 200 OK  Size: 900 Bytes  Time: 2.84 s
Response Headers  Cookies  Results  Docs
1 {
2   "message": "La solicitudes de servicio a atender se devolvió con éxito",
3   "data": [
4     "service_requests": [
5       {
6         "id": 84,
7         "cliente": "Jhoana Aucancela",
8         "state": 3,
9         "device": "Celular",
10        "model": "Huawei",
11        "brand": "Huawei 10",
12        "serie": null,
13        "description_problem": "Instalar word",
14        "payment_method": 1,
15        "date_issue": "2023-01-18"
16      },
17      {
18        "id": 86,
19        "cliente": "Jhoana Aucancela",
20        "state": 3,
21        "device": "Fjkttxc",
22        "model": "Fbltfvn",
23        "brand": "Fbkitr",
24        "serie": null,
25        "description_problem": "Fjiexb",
26        "payment_method": 1,
27        "date_issue": "2023-01-18"

```

Fig. 38: Visualización de solicitudes de contratación.

The screenshot shows a Postman collection named "manage-hiring". A GET request is made to `https://tecnony-v1.herokuapp.com/api/v1/manage-hiring/showone/84`. The response status is 200 OK, size is 1.46 KB, and time taken is 3.03 s. The response body is a JSON object containing details of a service request:

```

1  {
2      "message": "La solicitud de servicio se devolvió con éxito",
3      "data": {
4          "service_requests": {
5              "id": 84,
6              "cliente": "Jhoana Aucancela",
7              "state": 3,
8              "device": "Celular",
9              "model": "Huawei",
10             "brand": "Huawei 10",
11             "serie": null,
12             "description_problem": "Instalar word",
13             "payment_method": 1,
14             "date_issue": "2023-01-18"
15         },
16         "created_by": {
17             "id": 33,
18             "state": 1,
19             "username": "Jhoana6413",
20             "first_name": "Jhoana",
21             "last_name": "Aucancela",
22             "cedula": "1711264091",
23             "email": "lupitamerced19@gmail.com",
24             "birthdate": "2001-12-28",
25             "home_phone": "7654321",
26             "personal_phone": "0979002546",
27             "address": "Chilibulo",
28             "avatar": "https://res.cloudinary.com/dlzylh5f6/image/upload"
}

```

Fig. 39: Visualización de solicitud de contratación seleccionado.

The screenshot shows a Postman collection named "manage-hiring". A GET request is made to `nony-v1.herokuapp.com/api/v1/manage-hiring/approve/35`. The response status is 200 OK, size is 55 Bytes, and time taken is 2.51 s. The response body is a JSON object:

```

1  {
2      "message": "La solicitud de servicio ha sido aprobada"
3  }

```

Fig. 40: Aprobación de solicitud de contratación.

```

// ● 8. Aprobación de contratos.
// - Aprobación de solicitud de contratación.
public function test_aprobacion_de_solicitud_de_contratacion()
{
    $user = User::where('id', 21)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/manage-hiring/approve/%u', 87));
    $test_request->assertStatus(200);
}

```

Fig. 41: Test de aprobación de solicitud de contratación.

```

PASS Tests\Unit\S2TecnicoTest
✓ aprobacion de solicitud de contratacion

Tests: 1 passed
Time: 3.58s

```

Fig. 42: Resultados de la prueba.

Visualización de comentarios, sugerencias y calificación de los servicios

El *backend* efectúa la implementación de un *endpoint* consignado para la visualización de comentarios, sugerencias y calificaciones de los servicios. En ese sentido, el *endpoint* admite visualizar comentarios, sugerencias y calificaciones generadas por los formularios de satisfacción que han sido realizadas por los clientes, como se observa en la **Fig. 43**. Por último, en la **Fig. 44** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de visualización de comentarios, junto a los resultados que se observa en la **Fig. 45**. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el **ANEXO III**.

The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: <https://tecnony-v1.herokuapp.com/api/v1/view-satisfaction-form>
- Status: 200 OK
- Size: 1.91 KB
- Time: 2.76 s
- Auth tab is selected, showing Bearer token input: 136BgUslcAnQf4bmkqhFvFX6KJrSZJl2tmAGjkm2Tf
- Response tab displays JSON data:

```
1  {
2    "message": "Las solicitudes de servicios comentados se han devuelto con éxito",
3    "data": [
4      {
5        "score": 4.57,
6        "satisfaction_forms": [
7          {
8            "id": 13,
9            "comment": "Ea voluptatibus qui fugiat qui beatae odit. Commodo hic rerum odio optio. Occaecati quia facilis quis harum reprehenderit in qui.",
10           "suggestion": "Voluptatem explicabo molestias architecto et et dolor vitae reiciendis. Minus cum et magni veniam qui exercitationem. Ducimus recusandae vitae harum.",
11           "qualification": 2.45
12         },
13         {
14           "id": 14,
15           "comment": "Officiis tempora est vero"
```

Fig. 43: Visualización de comentarios.

```
// 9. Visualización de comentarios, sugerencias y calificación de servicios.
// - Visualización de comentarios.
public function test_visualizacion_de_comentarios()
{
    $user = User::where('id', 7)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/view-satisfaction-form');
    $test_request->assertStatus(200);
}
```

Fig. 44: Test de visualización de comentarios de servicios.

```
PASS Tests\Unit\S2TecnicoTest
✓ visualizacion de comentarios

Tests: 1 passed
Time: 3.79s
```

Fig. 45: Resultados de la prueba.

3.4 Sprint 3. Perfil cliente – implementación de endpoints.

Este *Sprint* se compone de tareas indispensables para la codificación de cada uno de los *endpoints* del usuario cliente, los cuales son:

- Inicio de sesión, cierre de sesión y recuperación de contraseña para el perfil cliente.
- Registro de usuario para el perfil cliente.
- Visualización de servicios.
- Contratación de servicios.
- Gestión de solicitudes de contratación.
- Comentar, sugerir y calificar servicios.

Inicio de sesión, cierre de sesión y recuperación de contraseña para el perfil cliente

El *backend* efectúa la implementación de un *endpoint* consignado para el inicio de sesión, cierre de sesión y recuperación de contraseña. En ese sentido, el *endpoint* para el inicio de sesión admite únicamente usuarios con perfil cliente, el cual solicita email y contraseña para ingresar a los módulos asignados, como se observa en la **Fig. 46**. Por último, en la **Fig. 47** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* de inicio de sesión para clientes, junto a los resultados que se observa en la **Fig. 48**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.

POST https://tecnony-v1.herokuapp.com/api/v1/loginCli

Status: 200 OK Size: 459 Bytes Time: 3.95 s

Response Headers: Content-Type, Content-Length, Date, Connection, Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Headers, Set-Cookie

```

1  {
2      "message": "Autenticación exitosa",
3      "data": {
4          "user": {
5              "id": 35,
6              "state": 1,
7              "username": "Leoni23",
8              "full_name": "Leoni Guumbo",
9              "cedula": "1723299203",
10             "email": "leoni23@gmail.com",
11             "role": "cliente",
12             "birthdate": "1999-08-15",
13             "home_phone": "0289463",
14             "personal_phone": "0967748367",
15             "address": "4 Comite del pueblo. 78",
16             "avatar": "https://cdn-icons-png.flaticon.com/512/848/848006.png"
17         },
18         "access_token": "143|rs0uhWwf4aY7hsaxgEChf1Fi6GuqHKzsJ35zMv7j",
19         "token_type": "Bearer"
20     }

```

Fig. 46: Inicio de sesión para clientes.

```

// ● 10. Inicio de sesión, cierre de sesión y recuperación de contraseña para el perfil cliente.
// - inicio de sesión para clientes.
public function test_iniciar_sesion_para_clientes()
{
    $test_request = $this->post('/api/v1/loginCli', [
        "email" => "leoni23@gmail.com",
        "password" => "happySad1*"
    ]);
    $test_request->assertStatus(200);
}

```

Fig. 47: Test de inicio de sesión para clientes.

```

PASS Tests\Unit\S3ClienteTest
✓ iniciar sesión para clientes

Tests: 1 passed
Time: 4.02s

```

Fig. 48: Resultados de la prueba.

Registro de usuario para el perfil cliente

El *backend* efectúa la implementación de un *endpoint* consignado para el registro de usuarios con perfil cliente. En ese sentido, el *endpoint* para el registro de usuarios admite el ingreso de datos personales los cuales son validados para la creación de nuevo usuario cliente en el *backend*, como se muestra en la Fig. 49. Por último, en la Fig. 50 se observa

el código efectuado de la prueba unitaria aplicada al *endpoint* de registro de usuario cliente, junto a los resultados que se observa en la **Fig. 51**. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el **ANEXO III**.

The screenshot shows a Postman interface with the following details:

- Request Method:** POST
- URL:** https://tecnony-v1.herokuapp.com/api/v1/register-cliente
- Status:** 200 OK
- Size:** 45 Bytes
- Time:** 10.04 s
- Body Response:**

```

1  {
2      "message": "Usuario registrado exitosamente"
3  }

```

The JSON body sent in the request is:

```

1  {
2      "username": "Leoni23",
3      "first_name": "Leoni",
4      "last_name": "Guambo",
5      "personal_phone": "0967748367",
6      "address": "4 Comite del pueblo. 78",
7
8      "cedula": "1723299203",
9      "email": "leoni23@gmail.com",
10
11     "birthdate": "1999-08-15",
12     "home_phone": "0289463",
13
14     "password": "happySad1*",
15     "password_confirmation": "happySad1*"
16 }
```

Fig. 49: Registro de usuario cliente.

```

// ● 11. Registro de usuario para el perfil cliente
// - Registrar usuario cliente
public function test_resgistro_de_cliente()
{
    $test_request = $this->post('/api/v1/register-cliente', [
        "username" => "Manu17",
        "first_name" => "Manu",
        "last_name" => "Auqui",
        "personal_phone" => "0992094598",
        "address" => "453 Chillogallo Oval Apt. 7",

        "cedula" => "1784698942",
        "email" => "manu17@gmail.com",

        "birthdate" => "1999-08-15",
        "home_phone" => "0266279",

        "password" => "happySad1*",
        "password_confirmation" => "happySad1*"
    ]);
    $test_request->assertStatus(200);
}

```

Fig. 50: Test de registro de cliente.

```

PASS Tests\Unit\S3ClienteTest
✓ registro de cliente

Tests: 1 passed
Time: 4.59s

```

Fig. 51: Resultados de la prueba.

Visualización de servicios

El *backend* efectúa la implementación de un *endpoint* consignado para la visualización de servicios. En ese sentido, el *endpoint* admite visualizar los servicios que son ofertados por los usuarios técnicos, como se observa en la **Fig. 52**. Por último, en la **Fig. 53** se observa el código efectuado de la prueba unitaria aplicada al *endpoint* visualización de servicios, junto a los resultados que se observa en la **Fig. 54**. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el **ANEXO III**.

Query Parameters	
parameter	value

```

Status: 200 OK  Size: 11.13 KB  Time: 3.93 s
Response Headers 10 Cookies Results Docs
1 {
2   "message": "Lista de servicios generada con éxito",
3   "data": [
4     "services": [
5       {
6         "id": 1,
7         "state": 1,
8         "name": "Desarrollo de aplicaciones móviles",
9         "categories": "software",
10        "description": "In mollitia nihil quis. A ad sint repudiandae corrupti ipsam labore. Magni eos ipsum querat sequi iure quasi. Dolores sint voluptatibus aut quis earum aperiam quidem. Quibusdam sed et iusto ea.",
11        "price": 28.42,
12        "image": "https://picsum.photos/id/1/200/300",
13        "attention_mode": 1,
14        "attention_description": "In mollitia nihil quis. A ad sint repudiandae corrupti ipsam labore. Magni eos ipsum querat sequi iure quasi. Dolores sint voluptatibus aut quis earum aperiam quidem. Quibusdam sed et iusto ea.",
15        "payment_method": 1,
16        "payment_description": "In mollitia nihil quis. A ad sint repudiandae corrupti ipsam labore. Magni eos ipsum querat sequi iure quasi. Dolores sint voluptatibus aut quis earum aperiam quidem. Quibusdam sed et iusto ea.",
17        "created_at": "2022-12-02 04:00:37"
18      },
19      {
20        "id": 2,

```

Fig. 52: Visualización de servicios.

```
// ● 12. Visualización de servicios
// - Visualizar servicios
public function test_visualizacion_de_servicios()
{
    $test_request = $this->get('/api/v1/view-service');
    $test_request->assertStatus(200);
}
```

Fig. 53: Test de visualización de servicios.

```
PASS Tests\Unit\S3ClienteTest
✓ visualizacion de servicios

Tests: 1 passed
Time: 9.21s
```

Fig. 54: Resultados de la prueba.

Contratación de servicios

El *backend* efectúa la implementación de un *endpoint* consignado para la contratación de servicios. En ese sentido, el *endpoint* admite el ingreso de datos para realizar una contratación de un servicio que haya seleccionado, como se observa en la Fig. 55. Por último, en la Fig. 56 se observa el código efectuado de la prueba unitaria aplicada al *endpoint* contratación de servicios, junto a los resultados que se observa en la Fig. 57. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el ANEXO III.

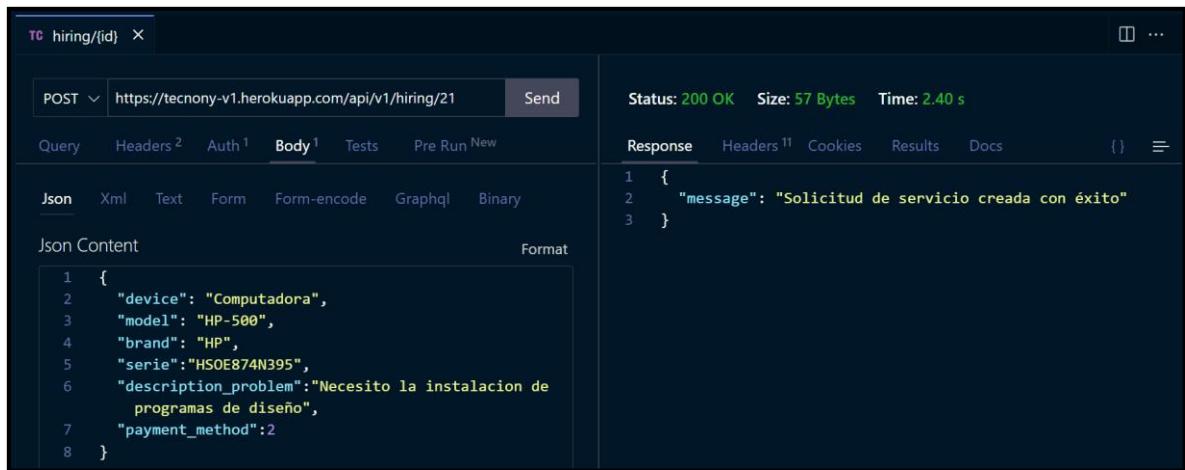


Fig. 55: Contratación de servicio.

```
// ● Contratación de servicios.
// - Contratación de servicio.
public function test_contratacion_de_servicio()
{
    $user = User::where('id', 35)->first();
    $hiring = [
        "device" => "Computadora",
        "model" => "HP-500",
        "brand" => "HP",
        "serie" => "H5OE874N395",
        "description_problem" => "Necesito la instalación de
            programas de diseño",
        'payment_method' => 2
    ];
    $test_request = $this->actingAs($user)->post(sprintf('/api/v1/hiring/%u', 8), $hiring);
    $test_request->assertStatus(200);
}
```

Fig. 56: Test de contratación de servicios.

```
PASS Tests\Unit\S3ClienteTest
✓ contratacion de servicio

Tests: 1 passed
Time: 3.93s
```

Fig. 57: Resultados de la prueba.

Gestión de solicitudes de contratación

El *backend* efectúa la implementación de varios *endpoints* consignados para gestión de solicitudes de contratación. En ese sentido, existe un *endpoint* para visualizar las solicitudes de contratación de servicio, como se muestra en la **Fig. 58**. El siguiente *endpoint*

permite visualizar la información detallada de una solicitud de contratación que se haya seleccionado, junto a la información personal del técnico quien lo atendió, como se muestra en la **Fig. 59**. El siguiente *endpoint* permite actualizar los datos de la solicitud de contratación, como se observa en la **Fig. 60**. Y el último *endpoint* permite cancelar o rehabilitar la solicitud de contratación, además, el *endpoint* admite cancelar la solicitud siempre y cuando no haya sido atendida por un técnico, como se observa en la **Fig. 61**. Por último, en la **Fig. 62** se observa el código efectuado de una prueba unitaria aplicada al *endpoint* visualización solicitudes de contratación, junto a los resultados que se observa en la **Fig. 63**. La funcionalidad de cada uno de los *endpoints* mencionados anteriormente se detallan a profundidad en el **ANEXO III**.

```

1  {
2    "message": "La lista de solicitudes de servicio se ha generado correctamente",
3    "data": [
4      "service_requests": [
5        {
6          "id": 3,
7          "cliente": "Genesis Mertz",
8          "state": 6,
9          "device": "quibusdam",
10         "model": "rerum",
11         "brand": "optio",
12         "serie": "LUOB13202",
13         "description_problem": "Dicta totam aut dicta enim quos et. Ea sint debitis
               molestiae ratione nobis. Consectetur eos doloribus at laborum nam
               consequatur. Et quis neque non neque ut doloribus dicta tenetur.",
14         "payment_method": 1,
15         "date_issue": "2022-11-23"
16       },
17       {
18         "id": 6,
19         "cliente": "Genesis Mertz",
20         "state": 6,
21         "device": "corrupti",
22         "model": "inventore",
23         "brand": "est",
24         "serie": "LQMUM8697",
25         "description_problem": "Dicta quisquam quos ratione rem voluptatum et. Animis
               quos qui corrupti ipsam. At fugiat aspernatur omnis."
26       }
27     ]
28   ]
29 }
```

Fig. 58: Visualización de solicitudes de contratación.

```

1  {
2      "message": "La solicitud de servicio fue devuelta con éxito",
3      "data": {
4          "comprobante": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1675032782
5              /rupzgfdtvjpjlntgabwn.jpg",
6          "service_request": {
7              "id": 6,
8              "clients": "Genesis Mertz",
9              "state": 6,
10             "device": "corrupti",
11             "model": "inventore",
12             "brand": "est",
13             "serie": "LOMUN8697",
14             "description_problem": "Dicta quisquam quos ratione rem voluptatum et. Anim
15                 quos qui corrupti ipsam. At fugiat aspernatur omnis.",
16             "payment_method": 1,
17             "date_issue": "2022-11-28"
18         },
19         "attention": {
20             "id": 6,
21             "cliente": "Genesis Mertz",
22             "device": "corrupti",
23             "state": 5,
24             "diagnosis": "Molestias qui qui vel aliquam delectus quia. Error laboriosam
25                 accusamus laboriosam enim consequatur doloribus. Adipisci exercitatem
26                 vel quis ut dolor reprehenderit iure.",
27             "incident_resolution": "Quo omnis sit fuga atque qui. Dolorem in saepe fugit
28                 sunt dolor. Et cumque sapiente maxime consequetur cumque occaecati quo."
29         }
30     }

```

Fig. 59: Visualizar contratación seleccionada.

```

1  {
2      "message": "Solicitud de servicio actualizada con éxito"
3  }

```

Fig. 60: Actualizar solicitud de contratación.

```

1  {
2      "message": "La solicitud de servicio ha sido cancelada
3          con éxito"
4  }

```

Fig. 61: Cancelar una solicitud de contratación.

```

// ● 14. Gestión de solicitudes de servicios
// - Visualizar contrataciones
public function test_visualizacion_de_contrataciones()
{
    $user = User::where('id', 35)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/hiring/show');
    $test_request->assertStatus(200);
}

```

Fig. 62: Test de gestión de solicitudes de contratación.

```

PASS Tests\Unit\S3ClienteTest
✓ visualizacion de contrataciones

Tests: 1 passed
Time: 2.95s

```

Fig. 63: Resultados de la prueba.

Comentar, sugerir y calificar servicios

El *backend* efectúa la implementación de un *endpoint* consignado para los comentarios, sugerencias y calificaciones de servicios. En ese sentido, el *endpoint* admite comentar, sugerir y calificar la atención de un contrato a través de un formulario de satisfacción, como se observa en la Fig. 64. Por último, en la Fig. 65 se observa el código efectuado de una prueba unitaria aplicada al *endpoint* comentar, sugerir y calificar un servicio, junto a los resultados que se observa en la Fig. 66. La funcionalidad del *endpoint* mencionado anteriormente se detalla a profundidad en el ANEXO III.

The screenshot shows a Postman interface with the following details:

- Request URL:** POST https://v1.herokuapp.com/api/v1/satisfaction-form/create/58
- Body:** JSON (with raw code shown below)
- Response Status:** 200 OK, Size: 79 Bytes, Time: 2.41 s
- Response Body:**

```

1 {
2     "message": "El formulario de satisfacción ha sido creado
            con éxito."
3 }
```

```

1 {
2     "comment": "El trabajo resultó satisfactorio",
3     "suggestion": "Nada que sugerir, todo bien",
4     "qualification": 9.60
5 }

```

Fig. 64: Comentar, sugerir y calificar servicio.

```

// ● 15. Comentar, sugerir y calificar los servicios
// - Comentar ... un servicio
public function test_comentar_un_servicio()
{
    $user = User::where('id', 16)->first();
    $comment = [
        "comment" => "El trabaja resulto satisfactorio",
        "suggestion" => "Nada que sugerir, todo bien",
        "qualification" => 9.85
    ];
    $test_request = $this->actingAs($user)->post(sprintf('/api/v1/satisfaction-form/create/%u', 34), $comment);
    $test_request->assertStatus(200);
}

```

Fig. 65: Test de comentar, sugerir y calificar los servicios.

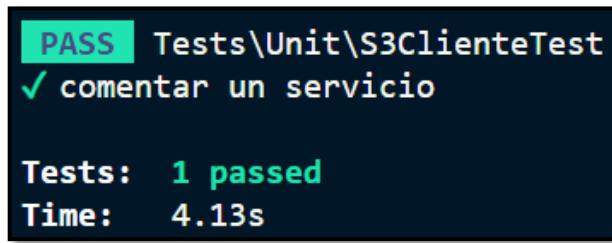


Fig. 66: Resultados de la prueba.

3.5 Sprint 4. Ejecución de pruebas para el componente *backend*.

Este *Sprint* se compone de tareas indispensables para la ejecución de pruebas tras haber finalizado la etapa de codificación de los *endpoints*:

- Resultados de la elaboración de las pruebas unitarias en el *backend*.
- Resultados de la elaboración de las pruebas de compatibilidad en el *backend*.
- Resultados de la elaboración de las pruebas de carga en el *backend*.

Resultados de la elaboración de las pruebas unitarias en el *backend*

Este tipo de prueba radica en la división del *software* en partes o secciones más pequeñas con el fin de realizar un proceso de pruebas a cada una de esas partes por separado. Además, este tipo de pruebas radica plenamente en el diseño de la arquitectura del proyecto y el lenguaje de programación en el que fue diseñado, ya que estos factores intervienen en el seccionamiento de las partes de *software* a probar [34].

En la **Fig. 67** se presenta una sección de código ejecutado para la prueba de unitaria de solicitud de afiliación. Mientras que en la **Fig. 68** se presenta los resultados que se han derivado de la implementación de todas las pruebas. Por último, en el **ANEXO II** se

encuentra los códigos completos de todas las pruebas unitarias restantes, aplicadas en el *backend*.

```
// - Aceptación de solicitud de afiliación
public function test_aceptacion_de_solicitud_de_afiliacion()
{
    $user = User::where('id', 1)->first();
    $answer = [
        "state" => 2,
        "observation" => "La información esta en orden",
    ];
    $test_request = $this->actingAs($user)->post(sprintf('/api/v1/manage/affiliation/create/%u', 14), $answer);
    $test_request->assertStatus(200);
}
```

Fig. 67: Test de aceptación de solicitud de afiliación.

```
PASS Tests\Unit\S1AdminTest
✓ recuperar contraseña
✓ cerrar sesion
✓ modificacion de avatar
✓ visualizacion de solicitudes de afiliacion seleccionada
✓ aceptacion de solicitud de afiliacion
✓ visualizar tecnicos
✓ activacion o desactivacion de tecnicos

PASS Tests\Unit\S2TecnicoTest
✓ creacion de un servicio
✓ activacion o desactivacion de un servicio
✓ visualizacion de solicitudes de contratacion
✓ visualizacion de solicitudes de contratacion seleccionado

PASS Tests\Unit\S3ClienteTest
✓ visualizacion contratacion seleccionada
✓ actualizar solicitud de contratacion
✓ cancelar una solicitud de contratacion

Tests: 14 passed
Time: 60.47s
```

Fig. 68: Resultados de las pruebas unitarias.

Una vez culminada las pruebas unitarias, se ha alcanzado resultados satisfactorios para las 14 API's restantes implementados en los 13 *endpoints* principales, con un tiempo de respuesta de 60.47 segundos.

Resultados de la elaboración de las pruebas de compatibilidad en el *backend*

Este tipo de prueba radica en probar el funcionamiento de un *software* en diferentes entornos como dispositivos, sistemas operativos, navegadores, entre otros. Cada entorno donde se ejecuta el *software* tiene la posibilidad de presentar diferentes resultados como: velocidad, rendimiento y funcionalidad. Además, el objetivo de estas pruebas es minimizar esos pequeños factores que puede ocasionar errores de ejecución en los diferentes entornos donde se ejecuta el *software* [21].

En la **TABLA VIII** se encuentran los clientes HTTP que han sido utilizados para probar la compatibilidad de las API's generadas por parte del *backend*. Por último, en el **ANEXO II** se presenta los resultados completos de todas las pruebas de compatibilidad que se han aplicado.

TABLA VIII. Clientes HTTP.

NOMBRE	VERSIÓN
<i>Thunder Client</i>	V2.0.2
<i>Postman</i>	V10.5.7

Una vez culminadas las pruebas de compatibilidad, se ha alcanzado resultados satisfactorios del consumo de las API's por medio de clientes HTTP, obteniendo respuestas semejantes al momento de ejecutar las peticiones.

Resultados de la elaboración de las pruebas de carga en el *backend*

Las pruebas de carga son una técnica utilizada para evaluar el comportamiento de un sistema, como una API REST, bajo una carga de trabajo simulada. Esta carga de trabajo puede ser generada por un gran número de solicitudes simultáneas, o por una cantidad elevada de datos que deben ser procesados. El objetivo es determinar el rendimiento del sistema en condiciones normales y extremas, y detectar cuellos de botella o problemas de escalabilidad [35].

En la **Fig. 69** se exponen una prueba de carga ejecutada en *apache jmeter* que se ha implementado para el *endpoint* de visualización de solicitud de afiliación. Por último, en el

ANEXO II se localizan los resultados completos de todas las pruebas de carga que se han aplicado.

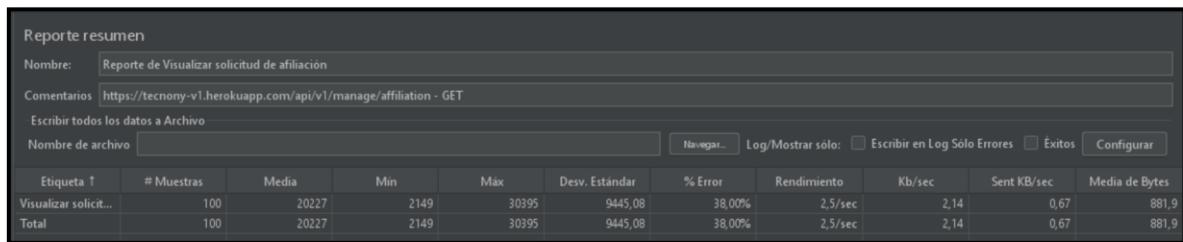


Fig. 69: Prueba de carga a visualizar solicitud de afiliación.

Una vez culminada la ejecución de las pruebas carga, se ha alcanzado resultados satisfactorios para las API's implementadas en los 13 *endpoints* principales, con un rendimiento de carga promedia de 2,5 segundos al ejecutar de 50 a 100 peticiones.

3.6 Sprint 5. Despliegue del backend a un ambiente de producción.

Este *Sprint* se compone de una sola tarea, la cual es el despliegue del *backend* a un ambiente de producción mediante *Heroku*, el cual es una plataforma para el despliegue de toda lógica directamente a producción siendo accesible desde cualquier lugar, como se observa en la **Fig. 70**. Por último, en el **ANEXO IV** se localiza el proceso de despliegue resultante.

<https://tecnony-v1.herokuapp.com/>

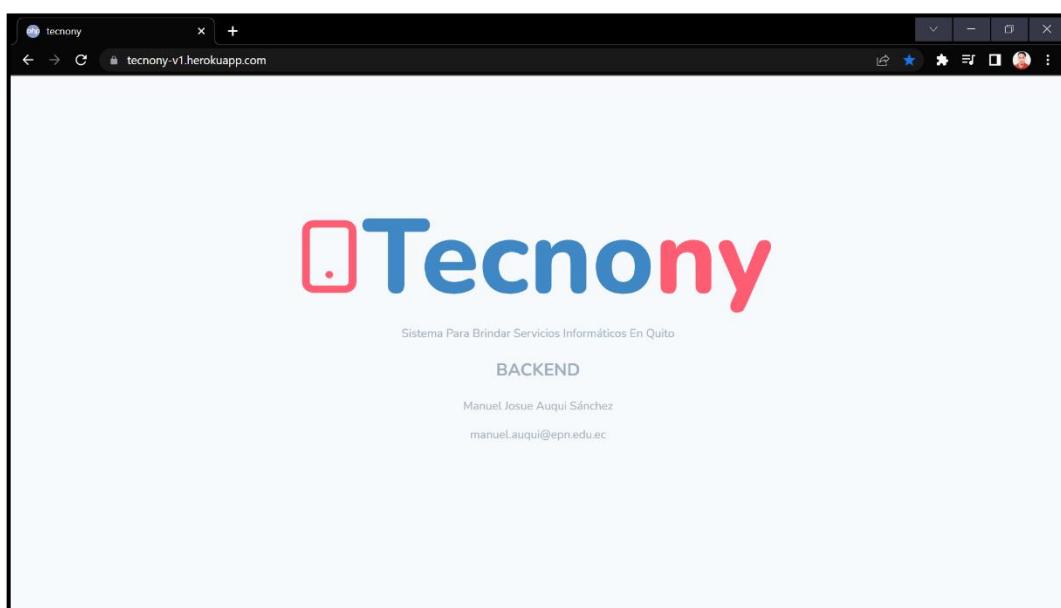


Fig. 70: Despliegue del backend.

4 CONCLUSIONES

Una vez culminado con las pruebas y despliegue, es indispensable destacar las conclusiones derivadas del desarrollo del *backend*.

- El *backend* logra solventar todas las necesidades para el exitoso funcionamiento de la aplicación “Tecnony”, el cual ofrece servicios técnicos especializados en el área de informática, admitiendo el consumo de *endpoints* para el área administrativa y técnica que se desenvuelve a través de un *frontend* y, por otro lado, permitiendo a la ciudadanía visualizar y contratar servicios a través de una aplicación móvil.
- Con ayuda de la metodología de desarrollo ágil “Scrum” ha sido posible levantar los requerimientos necesarios para el desarrollo del *backend* y organizarlos en diversos artefactos, por lo cual ha sido viable culminar el presente proyecto con anticipación.
- La base de datos, se presta como pilar fundamental para la lógica del proyecto, gracias a una base de datos relacional alojada en MySQL e implementada a través de *Eloquent* ha facilitado el almacenamiento y consulta de la información.
- Durante el desarrollo, aplicar el modelo arquitectónico modelo, vista y controlador, ha sido esencial para organizar el proyecto y presentar la información de manera estructurada a través de API's.
- Al aplicar las distintas pruebas en cada uno de los *endpoints* se ha logrado obtener resultados favorables en cada una de las funcionalidades del *backend*, su compatibilidad con distintos clientes HTTP y la satisfacción de los usuarios finales.
- Finalmente, el despliegue del *backend* a producción en *Heroku* se presta para el consumo satisfactorio de la información tanto para el *frontend*, como la aplicación móvil.

5 RECOMENDACIONES

Para el siguiente apartado, se lista algunas recomendaciones dignas de mencionar.

- Se recomienda efectuar algún método de organización para el desarrollo de cualquier proyecto, con el fin de documentar y gestionar el progreso de diseño e implementación de *endpoints*.
- Es preciso no alejarse de las actividades definidas a través de la recopilación de requerimientos, con el único objetivo de desarrollar cualquier proyecto en función de las necesidades del *Product Owner* y de los usuarios finales.
- Se recomienda documentar las API's originadas a través del desarrollo de los *endpoints*, con el fin identificar el funcionamiento y ocuparlos de un modo más rápido.
- Para el mayor entendimiento de los *endpoints* generados por el *backend*, es recomendable explicar la funcionalidades y restricciones de los *endpoints* a los desarrolladores del *frontend* y aplicación móvil, para que logren consumir los *endpoints* de manera efectiva.

6 REFERENCIAS BIBLIOGRÁFICAS

- [1] El universo, 26 jun 2020. [En línea]. Available: <https://www.eluniverso.com/noticias/2020/06/23/nota/7881924/internet-fijo-servicio-operadoras-demanda-cuarentena-covid-19/>. [Último acceso: 24 oct 2022].
- [2] INEC, «INEC. Buenas cifras, mejores vidas,» Abril 2021. [En línea]. Available: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2020/202012_Principales_resultados_Multiproposito_TIC.pdf. [Último acceso: 26 Junio 2022].
- [3] ESPE, «Universidad de las Fuerzas Armadas,» 2020. [En línea]. Available: <https://roa.cedia.edu.ec/webappscode/43/index.html>. [Último acceso: 25 oct 2022].
- [4] Useit, 4 jun 2018. [En línea]. Available: <https://www.useit.es/blog/beneficios-de-tener-una-pagina-web>. [Último acceso: 17 nov 2022].
- [5] udima, «Naxter,» 10 jun 2021. [En línea]. Available: <https://www.naxter.es/noticias/que-es-el-backend/>. [Último acceso: 17 NOV 2022].
- [6] N. Chapaval, «platzi,» 2018. [En línea]. Available: <https://platzi.com/blog/que-es-frontend-y-backend/>. [Último acceso: 31 Jul 2022].
- [7] D. Arias, «ENTER.CO,» 19 May 2021. [En línea]. Available: <https://www.enter.co/especiales/dev/empleos-dev/front-end-vs-back-end-cual-es-la-diferencia/>. [Último acceso: 31 Jul 2022].
- [8] G. villalobos, G. Camacho y D. Biancha, «Universidad Tecnológica de Pereira,» abr 2010. [En línea]. Available: <https://www.redalyc.org/pdf/849/84917316032.pdf>. [Último acceso: 27 oct 2022].
- [9] E. Bahit, «Elementos de POO,» de *POO Y MVC en PHP*, Creative Commons, 2011, p. 11.
- [10] Oracle, «OCI,» 2020. [En línea]. Available: <https://www.oracle.com/ar/database/what-is-a-relational-database/>. [Último acceso: 4 sep 2022].
- [11] J. P. Guardado, «Virtudes informáticas,» 10 abr 2020. [En línea]. Available: <https://virtumedia.wordpress.com/2020/04/10/relaciones-polimorficas-en-laravel/#:~:text=Las%20relaciones%20polim%C3%B3rficas%20nos%20permiten,modulos%20que%20tengan%20ese%20comportamiento.&text=Ahora%20en%20vez%20de%20tener,cada%20modelo%2C%20tenemos%20un>. [Último acceso: 4 sep 2022].
- [12] Laravel, «Laravel docs,» 2022. [En línea]. Available: <https://laravel.com/docs/7.x/eloquent-relationships#polymorphic-relationships>. [Último acceso: 4 sep 2022].
- [13] Red Hat, «Red Hat,» 6 may 2020. [En línea]. Available: <https://www.redhat.com/es/topics/api/what-is-a-rest-api>. [Último acceso: 4 sep 2022].

- [14] A. B. Martínez, «Platzi,» [En línea]. Available: <https://platzi.com/clases/1920-eloquent-laravel/28515-que-es-un-orm-y-para-que-sirve-eloquent/>. [Último acceso: 17 nov 2022].
- [15] php, [En línea]. Available: <https://www.php.net/manual/es/intro-whatis.php>. [Último acceso: 18 nov 2022].
- [16] G. E. D. Vega, «Arquitectura para diseñar e implementar Web Services,» Universidad Nacional de Colombia, Bogota, 2015.
- [17] Mozilla, «mdn web docs_,» [En línea]. Available: <https://developer.mozilla.org/es/docs/Web/HTTP/Overview>. [Último acceso: 11 nov 2022].
- [18] M. M. Durán, «El estudio de caso en la investigación cualitativa,» de *Revista nacional de admonistración*, Costa Rica, Escuela de Ciencias de la Administración, 2012, p. 121.
- [19] R. STAKE, «Case Study,» de *World Yearbook of Education*, London, Kogan Page, 1985, p. 277.
- [20] A. E. Meza, Manual para elejir una metodología de desarrollo de software dentro de un proyecto informático, Piura: Universidad de Piura, 2013.
- [21] R. S. Pressman, Ingeniería del software, un enfoque práctico, Mexico: McGraw-Hill, 2010.
- [22] S. Rivadeneira, G. Vilanova y M. Miranda, «El modelado de requerimientos en las metodologías ágiles,» SEDICI, Parana, 2013.
- [23] E. Bahit, «¿Comó funciona el patrón MVC?,» de *POO Y MVC en PHP*, Creative Commons, 2011, pp. 36-38.
- [24] Composer.org, «Documentacion,» 2022. [En línea]. Available: <https://getcomposer.org/doc/00-intro.md>. [Último acceso: 13 nov 2022].
- [25] alwaysdata, [En línea]. Available: <https://www.alwaysdata.com/en/>. [Último acceso: 13 nov 2022].
- [26] Heroku, «Heroku Dev Center,» [En línea]. Available: <https://devcenter.heroku.com/categories/reference>. [Último acceso: 13 nov 2022].
- [27] Cloudinary, «Cloudinary,» [En línea]. Available: <https://cloudinary.com/>. [Último acceso: 2023 ene 03].
- [28] visualstudio, «code.visualstudio.com,» [En línea]. Available: <https://code.visualstudio.com/docs>. [Último acceso: 13 nov 2022].
- [29] T. A. S. Foundation. [En línea]. Available: <https://jmeter.apache.org/>. [Último acceso: 11 ene 2023].

- [30] Cloudinary, «GitHub,» [En línea]. Available: <https://github.com/cloudinary-devs/cloudinary-laravel>. [Último acceso: 2023 ene 03].
- [31] Laravel, «Laravel Lang,» [En línea]. Available: <https://laravel-lang.com/>. [Último acceso: 2023 ene 03].
- [32] Laravel, «Laravel-santum,» [En línea]. Available: <https://laravel.com/docs/9.x/sanctum#introduction>. [Último acceso: 2023 ene 03].
- [33] FakerPHP, «FakerPHP / Faker,» [En línea]. Available: <https://fakerphp.github.io/>. [Último acceso: 2023 ene 03].
- [34] D. Delgado, «EVALUACIÓN DEL IMPACTO DE LAS PRUEBAS UNITARIAS,» UNIVERSIDAD DE COSTA RICA, Costa Rica, 2013.
- [35] Z. J. C. Mario, C. Velásquez y Jesús, Comparación de las características de algunas herramientas de software para pruebas de carga, Medellín, Colombia: Universidad Nacional de Colombia, 2011.

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 anti plagio Turnitin.



CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 12 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 BRINDAR SERVICIOS INFORMÁTICOS EN QUITO elaborado por el estudiante Manuel Josue Auqui Sanchez 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: Tabla de requerimientos.

RECOPIACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	ID - RR	ENUNCIADO DEL ÍTEM
Backend	RR001	Los usuarios con perfil administrador, técnico y cliente necesitan utilizar varios endpoints para: <ul style="list-style-type: none">• Iniciar sesión• Cerrar sesión• Recuperar contraseña
	RR002	El usuario con perfil administrador, técnico y cliente necesitan utilizar varios endpoints para: <ul style="list-style-type: none">• Modificar perfil de usuario
	RR003	El usuario con perfil administrador necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Gestionar solicitudes de afiliación
	RR004	El usuario con perfil administrador necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Visualizar comentarios, sugerencias y calificación de los técnicos.
	RR005	El usuario con perfil técnico y cliente necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Registrarse
	RR006	El usuario con perfil técnico necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Solicitar afiliación
	RR007	El usuario con perfil técnico necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Gestionar servicios

	RR008	El usuario con perfil técnico necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Aprobar y/o rechazar servicios
	RR011	El usuario con perfil cliente necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Solicitar contratación.
	RR012	El usuario con perfil cliente necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Gestionar contrataciones.
	RR013	El usuario con perfil cliente necesita utilizar varios endpoints para: <ul style="list-style-type: none">• Comentar, sugerir y calificar servicio

Historias de Usuario

Concluida la etapa de recopilación de requerimientos, se ha procedido a crear las Historias de usuario para el *backend*. A continuación, se presenta las 13 historias de usuarios que han sido elaboradas en base a los requerimientos que han sido recopilados previamente, las cuales inician desde **TABLA X** hasta **TABLA XXI**.

TABLA X: Iniciar sesión, cerrar sesión y recuperar contraseña.

HISTORIA DE USUARIO	
Identificador: HU001	Usuario: Administrador, Técnico y Cliente
Nombre de Historia: Iniciar sesión, cerrar sesión y recuperar contraseña.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 1	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios endpoints para que los usuarios con perfil administrador, técnico y cliente puedan: <ul style="list-style-type: none">• Iniciar sesión.• Cerrar sesión.• Recuperar contraseña.	
Observación: El <i>backend</i> debe verificar que los datos ingresados en el formulario sean validados, además, el <i>backend</i> debe enviar un correo electrónico al usuario que desee recuperar su contraseña.	

TABLA XI: Modificar perfil de usuario.

HISTORIA DE USUARIO	
Identificador: HU002	Usuario: Administrador, Técnico y Cliente
Nombre de historia: Modificar perfil de usuario.	
Prioridad en Negocio: Media	Riesgo en Negocio: Media
Iteración asignada: 1	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que los usuarios con perfil administrador, técnico y cliente puedan visualizar y editar su perfil llenando los siguientes campos del formulario: <ul style="list-style-type: none"> • Datos personales. • Imagen del avatar. 	
Observación: El <i>backend</i> permite que los usuarios puedan modificar su perfil siempre y cuando hayan iniciado sesión previamente y dispongan de un <i>token</i> de acceso.	

TABLA XII: Gestionar usuarios.

HISTORIA DE USUARIO	
Identificador: HU003	Usuario: Administrador.
Nombre de historia: Gestionar solicitudes de afiliación.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 2	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil administrador pueda realizar las siguientes acciones: <ul style="list-style-type: none"> • Visualizar solicitudes de afiliaciones de los nuevos técnicos. • Aceptar o rechazar las afiliaciones de los técnicos. 	
Observación: El <i>backend</i> debe verificar la validación de los campos al momento de rechazar una afiliación, además, es responsabilidad del administrador los criterios para aceptar o rechazar una solicitud de afiliación.	

TABLA XIII: Visualizar comentarios de los técnicos.

HISTORIA DE USUARIO	
Identificador: HU004	Usuario: Administrador.
Nombre de historia: Visualizar comentarios, sugerencias y calificación de los técnicos.	
Prioridad en Negocio: Medio	Riesgo en Negocio: Medio
Iteración asignada: 2	
Responsable: Manuel Auqui	
<p>Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil administrador pueda realizar las siguientes acciones:</p> <ul style="list-style-type: none"> • Visualizar el listado de los técnicos. • Visualizar el promedio de las calificaciones de un técnico. • Visualizar comentarios, sugerencias enviadas al técnico. • Activar o inactivar un técnico. 	
<p>Observación: El <i>backend</i> debe verificar la validación de campos al momento de inactivar a un técnico por malas calificaciones que haya recibido, además el técnico recibe un correo para informarle sobre el motivo por el cual ha sido inactivado.</p>	

TABLA XIV: Registrarse.

HISTORIA DE USUARIO	
Identificador: HU005	Usuario: Técnico y Cliente.
Nombre de historia: Registrarse.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 2	
Responsable: Manuel Auqui	
<p>Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que los usuarios con perfil técnico y cliente puedan registrarse, implementando un formulario que permitan a los usuarios:</p> <ul style="list-style-type: none"> • Ingresar sus datos personales. • Creación de una contraseña. 	
<p>Observación: El <i>backend</i> debe verificar que el usuario llene todos campos correctamente para que pueda ingresar a los módulos de la aplicación.</p>	

TABLA XV: Solicitar afiliación.

HISTORIA DE USUARIO	
Identificador: HU006	Usuario: Técnico.
Nombre de historia: Solicitar afiliación.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 2	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil técnico pueda solicitar la afiliación, implementando un formulario de registro de: <ul style="list-style-type: none"> • Datos laborales. 	
Observación:	
El <i>backend</i> debe verificar que el técnico este previamente registrado para que pueda hacer una solicitud de afiliación.	

TABLA XVI: Aprobar contrataciones.

HISTORIA DE USUARIO	
Identificador: HU008	Usuario: Técnico.
Nombre de historia: Aprobar contrataciones.	
Prioridad en Negocio: Media	Riesgo en Negocio: Media
Iteración asignada: 3	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil técnico pueda aprobar contrataciones, implementando para ello un <i>endpoint</i> que le permita al técnico: <ul style="list-style-type: none"> • Visualizar las solicitudes de contratación enviados por los clientes. • Aprobar o rechazar la solicitud de contratación. 	
Observación:	
El <i>backend</i> debe validar que cuando el técnico acepte la solicitud de contratación el estado de la solicitud cambia a en proceso, una vez haya atendido la solicitud el estado cambiara a finalizado. Además, el cliente puede ver el estado de su solicitud de contratación.	

TABLA XVII: Visualizar comentarios del servicio.

HISTORIA DE USUARIO	
Identificador: HU009	Usuario: Técnico.
Nombre de historia: Visualizar comentarios, sugerencias y calificación del servicio.	
Prioridad en Negocio: Medio	Riesgo en Negocio: Medio
Iteración asignada: 3	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil técnico pueda: <ul style="list-style-type: none"> • Visualizar los comentarios, sugerencias y calificación del servicio. 	
Observación:	
El <i>backend</i> debe verificar comentarios, sugerencias y calificaciones que hacen los clientes que han sido atendidos por el técnico.	

TABLA XVIII: Visualizar servicios.

HISTORIA DE USUARIO	
Identificador: HU010	Usuario: Cliente.
Nombre de historia: Visualizar servicios.	
Prioridad en Negocio: Medio	Riesgo en Negocio: Medio
Iteración asignada: 2	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil cliente puedan: <ul style="list-style-type: none"> • Visualizar el listado de servicios disponibles. 	
Observación:	
El <i>backend</i> debe validar que no es necesario que un cliente inicie sesión para que pueda visualizar los servicios que se ofertan.	

TABLA XIX: Solicitar contratación.

HISTORIA DE USUARIO	
Identificador: HU011	Usuario: Cliente.
Nombre de historia: Solicitar contratación.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 3	

Responsable: Manuel Auqui
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil cliente pueda: <ul style="list-style-type: none"> • Realizar una solicitud de contratación.
Observación:
El <i>backend</i> debe verificar que el cliente haya iniciado sesión para que pueda hacer una solicitud de contratación.

TABLA XX: Gestionar contrataciones.

HISTORIA DE USUARIO	
Identificador: HU012	Usuario: Cliente.
Nombre de historia: Gestionar contrataciones.	
Prioridad en Negocio: Alta	Riesgo en Negocio: Alta
Iteración asignada: 3	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil cliente pueda gestionar solicitudes de contratación, implementando para ello un <i>endpoint</i> que le permita al cliente: <ul style="list-style-type: none"> • Visualizar contrataciones. • Cancelar contrataciones. 	
Observación:	
El <i>backend</i> debe verificar que el cliente pueda cancelar una contratación siempre y cuando la contratación aun no haya sido aprobada por el técnico.	

TABLA XXI: Comentar, sugerir y calificar servicio.

HISTORIA DE USUARIO	
Identificador: HU013	Usuario: Cliente.
Nombre de historia: Comentar, sugerir y calificar servicio.	
Prioridad en Negocio: Media	Riesgo en Negocio: Media
Iteración asignada: 3	
Responsable: Manuel Auqui	
Descripción: El <i>backend</i> debe generar varios <i>endpoints</i> para que el usuario con perfil cliente pueda: <ul style="list-style-type: none"> • Comentar, sugerir y calificar el servicio contratado. 	

Observación:

La contratación debe haber finalizado para que el cliente pueda hacer la calificación respectiva.

Product Backlog

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

TABLA XXII: Product Backlog.

PRODUCT BACKLOG				
ID-HU	HISTORIA DE USUARIO	ITERACIÓN	ESTADO	PRIORIDAD
HU001	Iniciar sesión, cerrar sesión y recuperar contraseña.	1	Finalizado	Alta
HU002	Modificar perfil de usuario.	1	Finalizado	Medio
HU003	Gestionar solicitudes de afiliación.	2	Finalizado	Alto
HU004	Visualizar comentarios, sugerencias y calificación de los técnicos.	2	Finalizado	Medio
HU005	Registrarse.	2	Finalizado	Alto
HU008	Aprobar contrataciones.	3	Finalizado	Medio
HU009	Visualizar comentarios, sugerencias y calificación del servicio.	3	Finalizado	Medio
HU010	Visualizar servicios.	2	Finalizado	Medio
HU011	Solicitar contratación.	3	Finalizado	Alta
HU012	Gestionar contrataciones.	3	Finalizado	Alta

HU013	Comentar, sugerir y calificar servicio.	3	Finalizado	Medio
-------	---	---	------------	-------

Sprint Backlog

La **TABLA XXIII** muestra todos los *Sprints* que se han planificado en el proyecto los cuales se dividen en 5 *Sprints* principales, cada uno de ellos detalla las tareas que se han realizado conjuntamente con el tiempo para completar los entregables que se han definido con el *Product Owner*.

TABLA XXIII: *Sprint Backlog.*

SPRINT BACKLOG						
ID-SB	NOMBRE	MÓDULO	ID-HU	HISTORIA DE USUARIO	TAREAS	TIEMPO ESTIMADO
SB001	Diseño e implementación de <i>endpoints</i> para el perfil administrador	Módulo de autenticación y recuperación de contraseña	HU001	Iniciar sesión, cerrar sesión y recuperar contraseña.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para el inicio de sesión, cierre de sesión y cambio de contraseña. Implementación consulta a la base de datos y autorización. Validación de los datos requeridos. 	40 Horas
		Módulo - Perfil	HU002	Modificar perfil de usuario.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar y editar el perfil de usuario. Implementación de <i>endpoints</i> de consultas en la base de datos. Validación de los datos requeridos. 	

		Módulo – Usuarios	HU003	Gestionar solicitudes de afiliación.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar, aceptar o rechazar las solicitudes de técnicos. Implementación de <i>endpoints</i> de consultas en la base de datos. 	
		Módulo – Comentarios, sugerencias y calificación	HU004	Visualizar comentarios, sugerencias y calificación de los técnicos.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar comentarios, sugerencias y calificación de los técnicos. Implementación de <i>endpoints</i> de consultas en la base de datos. Diseño e implementación de <i>endpoints</i> para activar y desactivar técnicos. Implementa <i>endpoints</i> para notificar al técnico en caso de que haya sido desactivado. 	
SB002	Diseño e implementación de <i>endpoints</i> para el perfil técnico	Módulo de autenticación y recuperación de contraseña	HU001	Iniciar sesión, cerrar sesión y recuperar contraseña.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para el inicio de sesión, cierre de sesión y cambio de contraseña. Implementación consulta a la base de datos y autorización. Validación de los datos requeridos. 	40 Horas
		Módulo - Perfil	HU002	Modificar perfil de usuario.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar y editar el perfil de usuario. 	

					<ul style="list-style-type: none"> • Implementación de <i>endpoints</i> de consultas en la base de datos. • Validación de los datos requeridos. 	
	Módulo – Registro	HU005	Registrarse.		<ul style="list-style-type: none"> • Diseño e implementación de <i>endpoints</i> para el usuario con perfil técnico pueda registrarse. • Implementación de <i>endpoints</i> de ingreso de datos en la base de datos. • Validación de los datos requeridos. 	
	Módulo – Afiliación	HU006	Solicitar afiliación.		<ul style="list-style-type: none"> • Diseño e implementación de <i>endpoints</i> para solicitar afiliación, por medio de un formulario de datos personales y laborales. • Implementación de <i>endpoints</i> de ingreso de datos en la base de datos. • Validación de los datos requeridos. 	
	Módulo – Servicios	HU007	Gestionar servicios.		<ul style="list-style-type: none"> • Diseño e implementación de <i>endpoints</i> para crear, visualizar, actualizar y eliminar servicios. • Implementación de <i>endpoints</i> de consultas en la base de datos. • Validación de los datos requeridos. 	
		HU008	Aprobar contratación.		<ul style="list-style-type: none"> • Diseño e implementación de <i>endpoints</i> para visualizar las solicitudes de contratación enviadas por el cliente. 	

					<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para aprobar o rechazar la solicitud de contratación. Implementación de <i>endpoints</i> de consultas en la base de datos. Validación de los datos requeridos. 	
		Módulo – Comentarios, sugerencias y calificación	HU009	Visualizar la comentarios, sugerencias y calificación del servicio.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar los comentarios, sugerencias y calificación de los servicios. Implementación de <i>endpoints</i> de consultas en la base de datos. 	
SB003	Diseño e implementación de <i>endpoints</i> para el perfil cliente	Módulo de autenticación y recuperación de contraseña	HU001	Iniciar sesión, cerrar sesión y recuperar contraseña.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para el inicio de sesión, cierre de sesión y cambio de contraseña. Implementación consulta a la base de datos y autorización. Validación de los datos requeridos. 	40 Horas
		Módulo - Perfil	HU002	Modificar perfil de usuario.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar y editar el perfil de usuario. Implementación de <i>endpoints</i> de consultas en la base de datos. Validación de los datos requeridos. 	

		Módulo – Registro	HU005	Registrarse.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para el usuario con perfil cliente pueda registrarse. Implementación de <i>endpoints</i> de ingreso de datos en la base de datos. Validación de los datos requeridos. 	
		Módulo – Servicios	HU010	Visualizar los servicios.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para visualizar los servicios. Implementación de <i>endpoints</i> de consultas en la base de datos. 	
			HU011	Solicitar contratación.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para solicitar un contrato de algún servicio seleccionado, por medio de un formulario de datos. Implementación de <i>endpoints</i> de ingreso de datos en la base de datos. Validación de los datos requeridos. 	
			HU012	Gestionar contrataciones.	<ul style="list-style-type: none"> Diseño e implementación de <i>endpoints</i> para, visualizar, actualizar y cancelar contrataciones. Implementación de <i>endpoints</i> de consultas en la base de datos. Validación de los datos requeridos. 	

		Módulo – Comentarios, sugerencias y calificación	HU013	Comentar, sugerir y calificar los servicios	<ul style="list-style-type: none"> • Diseño e implementación de <i>endpoints</i> para hacer comentarios, sugerencias y calificar el servicio contratado. • Implementación de <i>endpoints</i> de ingreso de datos en la base de datos. • Validación de los datos requeridos. 	
SB005	Despliegue del <i>backend</i>	<ul style="list-style-type: none"> • Despliegue del <i>backend</i> en Heroku. 				10 Horas
SB006	Documentación	<ul style="list-style-type: none"> • Documento del Trabajo de Integración Curricular. • Anexos. 				30 Horas
TOTAL						240H

Diseño de la Base de Datos

En la **Fig. 71** se muestra las tablas necesarias para el almacenamiento y relacionamiento de la información que se ha usado para el desarrollo de la base de datos del *backend*.

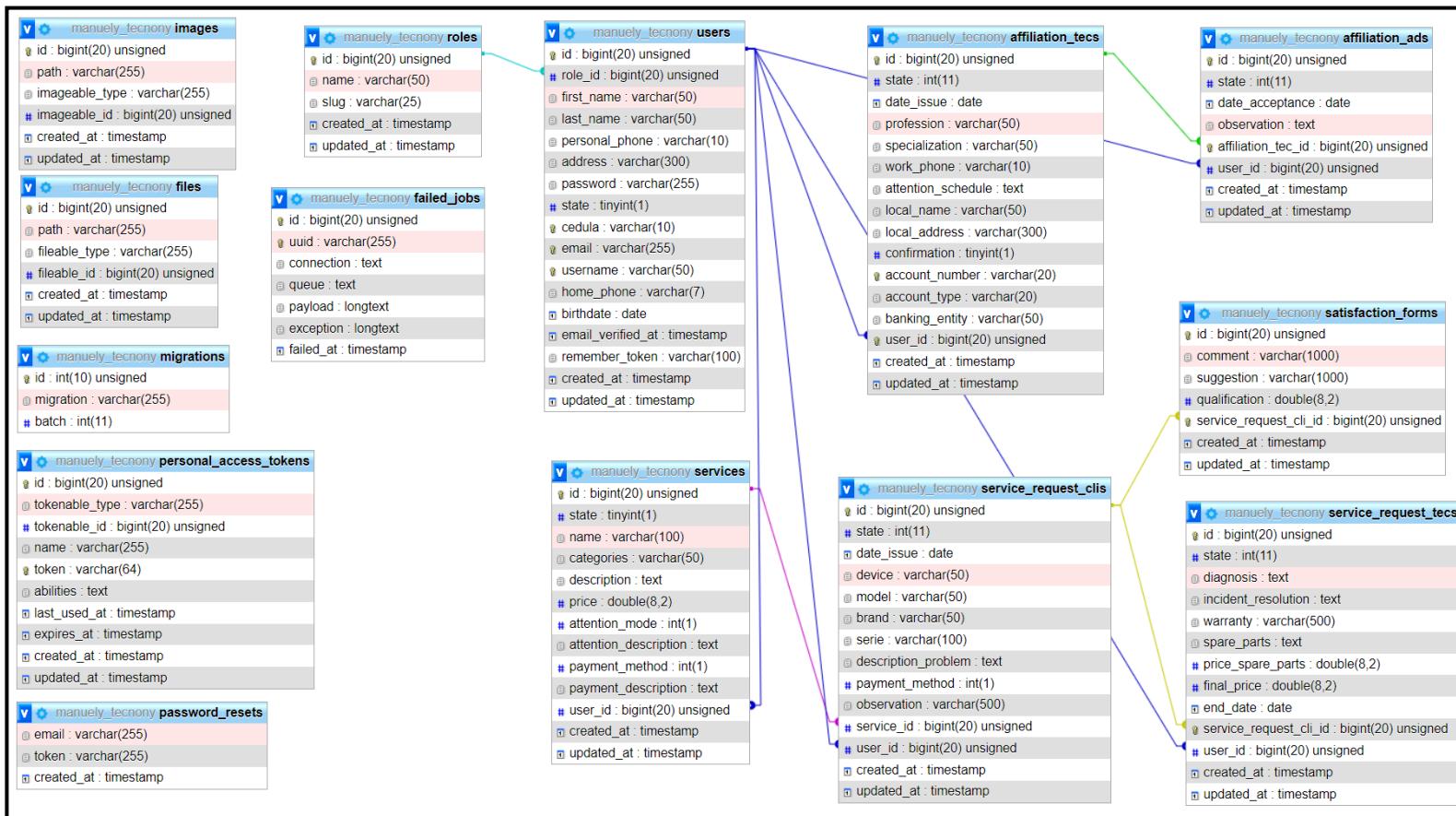


Fig. 71: Diseño de la base de datos.

Pruebas

Después de culminar con el desarrollo de los *endpoints* es indispensable probar el funcionamiento satisfactorio de los mismo. De esta manera, las pruebas se han ejecutado de los siguientes modos.

Pruebas Unitarias

En este apartado se presenta las 13 pruebas unitarias restantes que han sido elaboradas en base a los *endpoints* que han sido desarrollados previamente, las cuales se muestran desde **Fig. 72** hasta **Fig. 84**.

```
// - Recuperación de contraseña
public function test_recuperar_contraseña()
{
    $test_request = $this->post('/api/v1/forgot-password', [
        "email" => "chance12@example.org",
    ]);
    $test_request->assertStatus(200);
}
```

Fig. 72: Test de recuperación de contraseña.

```
// - Cerrar sesión
public function test_cerrar_sesion()
{
    $user = User::factory()->make(['role_id' => 1]);
    $test_request = $this->actingAs($user)->post('/api/v1/logout');
    $test_request->assertStatus(200);
}
```

Fig. 73: Test de Cierre de sesión.

```
// - Modificar avatar
public function test_modificacion_de_avatar()
{
    $user = User::factory()->make(['role_id' => 1]);
    $avatar = [
    ];
    $test_request = $this->actingAs($user)->post('/api/v1/profile/avatar', $avatar);
    $test_request->assertStatus(200);
}
```

Fig. 74: Test de modificar avatar.

```

// - Visualizar solicitud de afiliación seleccionada
public function test_visualizacion_de_solicitudes_de_afiliacion_seleccionada()
{
    $user = User::where('id', 1)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/manage/affiliation/%u', 14));
    $test_request->assertStatus(200);
}

```

Fig. 75: Test de visualizar solicitud de afiliación seleccionada.

```

// - Visualizar técnicos
public function test_visualizar_tecnicos()
{
    $user = User::where('id', 1)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/manage-tec');
    $test_request->assertStatus(200);
}

```

Fig. 76: Test de visualizar técnicos.

```

// - Activación o desactivación de técnico
public function test_activacion_o_desactivacion_de_tecnicos()
{
    $user = User::where('id', 1)->first();
    $change = [
        "observation" => "Tecnico desactivado",
    ];
    $test_request = $this->actingAs($user)->post(sprintf('/api/v1/manage-tec/change-state/%u', 24), $change);
    $test_request->assertStatus(200);
}

```

Fig. 77: Test de activación o desactivación de técnicos.

```

// - Creación de un servicio
public function test_creacion_de_un_servicio()
{
    $user = User::factory()->make(['role_id' => 2]);
    $service = [
        "name" => "Mantenimientos de computadoras",
        "categories" => "mantenimiento",
        "description" => "Se realiza mantenimiento a computadora de todo tipo",
        "price" => 7.00,
        "attention_mode" => 1,
        "attention_description" => "Solo poseemos atencion en el mismo local, necesita",
        "payment_method" => 2,
        "payment_description" => "Poseemos pagos en efectivo y deposito o trasferencia"
    ];
    $test_request = $this->actingAs($user)->post('/api/v1/service/create', $service);
    $test_request->assertStatus(200);
}

```

Fig. 78: Test de creación de un servicio.

```

// - Activación o desactivación de un servicio
public function test_activacion_o_desactivacion_de_un_servicio()
{
    $user = User::where('id', 8)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/service/destroy/%u', 5));
    $test_request->assertStatus(200);
}

```

Fig. 79: Test de activación o desactivación de un servicio.

```

// - Visualización de solicitudes de contratación.
public function test_visualizacion_de_solicitudes_de_contratacion()
{
    $user = User::where('id', 21)->first();
    $test_request = $this->actingAs($user)->get('/api/v1/manage-hiring/shownew');
    $test_request->assertStatus(200);
}

```

Fig. 80: Test de visualización de solicitudes de contratación.

```

// - Visualización de solicitud de contratación seleccionado.
public function test_visualizacion_de_solicitudes_de_contratacion_seleccionado()
{
    $user = User::where('id', 8)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/manage-hiring/showone/%u', 5));
    $test_request->assertStatus(200);
}

```

Fig. 81: Test de visualización de solicitud de contratación seleccionada.

```

// - Visualizar contratación seleccionada.
public function test_visualizacion_contratacion_seleccionada()
{
    $user = User::where('id', 33)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/hiring/show/%u', 61));
    $test_request->assertStatus(200);
}

```

Fig. 82: Test de visualizar contratación seleccionada.

```
// - Actualizar solicitud de contratación.
public function test_actualizar_solicitud_de_contratacion()
{
    $user = User::where('id', 33)->first();
    $hiring = [
        "device" => "Computadora",
        "model" => "HP-500",
        "brand" => "HP",
        "serie" => "H50E874N395",
        "description_problem" => "Necesito el formateo completo",
        'payment_method' => 1
    ];
    $test_request = $this->actingAs($user)->post(sprintf('/api/v1/hiring/update/%u', 61), $hiring);
    $test_request->assertStatus(200);
}
```

Fig. 83: Test de actualizar solicitud de contratación.

```
// - Cancelar una solicitud de contratación.
public function test_cancelar_una_solicitud_de_contratacion ()
{
    $user = User::where('id', 33)->first();
    $test_request = $this->actingAs($user)->get(sprintf('/api/v1/hiring/cancel/%u', 61));
    $test_request->assertStatus(200);
}
```

Fig. 84: Test de cancelar una solicitud de contratación.

Pruebas de compatibilidad

En este apartado se presenta las 15 pruebas de compatibilidad que han sido elaboradas en base a los *endpoints* que han sido desarrollados previamente, las cuales se muestran desde **Fig. 85** hasta **Fig. 99**.

The screenshot shows a POST request to the endpoint `https://tecnony-v1.herokuapp.com/api/v1/login`. The request body is a JSON object with fields `email` and `password`. The response is a 200 OK status with a response time of 2.35 s and a size of 752 B. The response body is a JSON object containing a message, user data, and tokens.

```

{
  "message": "Autenticación exitosa",
  "data": {
    "user": {
      "id": 7,
      "state": 1,
      "username": "Prof. Elody Beer",
      "full_name": "Santa Schimmel",
      "cedula": "176678879",
      "email": "chance12@example.org",
      "role": "técnico",
      "birthdate": "1987-12-12",
      "home_phone": "0237530",
      "personal_phone": "0980539785",
      "address": "434 Block Skyway",
      "avatar": "https://cdn-icons-png.flaticon.com/512/848/848006.png"
    },
    "access_token": "1641ZFQgD0GxfV3vAf3zxIdhkKYsd3gkRD78BR0s8za8",
    "token_type": "Bearer"
  }
}

```

Fig. 85: Prueba de inicio de sesión con Postman.

The screenshot shows the Postman interface with a POST request to `https://tecnony-v1.herokuapp.com/api/v1/profile`. The request body contains a JSON object with profile details. The response status is 200 OK, and the message "Perfil actualizado con éxito" is displayed.

```

1 {
2   "username": "santa12",
3   "first_name": "Santa",
4   "last_name": "Schimmel",
5   "cedula": "1766768879",
6   "email": "chance12@example.org",
7   "birthdate": "1987-12-12",
8   "home_phone": "0237530",
9   "personal_phone": "0980539785",
10  "address": "434 Block Skyway"
11 }

```

```

1 "message": "Perfil actualizado con éxito"
2
3

```

Fig. 86: Prueba de modificar perfil con Postman.

The screenshot shows the Postman interface with a GET request to `https://tecnony-v1.herokuapp.com/api/v1/manage/affiliations`. The request includes an Authorization header set to Bearer Token. The response status is 200 OK, and the message "La lista de afiliaciones respondidas se ha generado con éxito" is displayed, along with a list of affiliations.

```

1 "message": "La lista de afiliaciones respondidas se ha generado con éxito",
2 "data": [
3     {
4         "affiliations": [
5             {
6                 "id": 1,
7                 "state": 3,
8                 "date_acceptance": "2009-02-11",
9                 "observation": "Solicitud de afiliacion actualizada"
10            },
11            {
12                "id": 2,
13                "state": 2,
14                "date_acceptance": "2022-01-20",
15                "observation": "Recusandae perferendis sint eos et. Ratione consequatur est laboriosam ratione alias. Eos non quasi adipisci omnis voluptas. Et ierum et numquam est et magni voluptatibus vel."
16            }
17        ]
18    }
19 ]

```

Fig. 87: Prueba de gestión de afiliaciones con Postman.

The screenshot shows a POST request to `https://tecnony-v1.herokuapp.com/api/v1/manage-tec/show/6`. The response body is a JSON object:

```

{
  "message": "La puntuación y los comentarios del técnico se ha devuelto con éxito",
  "data": [
    {
      "score": 7.64,
      "technical": {
        "id": 6,
        "state": 1,
        "username": "AydenCt",
        "first_name": "Ayden",
        "last_name": "Cremin",
        "cedula": "1778274981",
        "email": "bmcdermott@example.org",
        "birthdate": "2002-09-24",
        "home_phone": "0273897",
        "personal_phone": "0966197576",
        "address": "95525 O'Connell Ville Apt. 058",
        "avatar": "https://res.cloudinary.com/dlzylh5f6/image/upload/v1672105480/rfisxkezouainvzcryda.png"
      },
      "satisfaction_forms": [
        {
          "id": 1,
          "comment": "Dicta amet veritatis possimus quia minus. Quidem molestiae repudiandae aperiam asperiores reprehenderit beatae temporibus.",
          "suggestion": "Oui ut quae nisi labore voluptatibus tempora. Officiis minus nisi sed fugiat"
        }
      ]
    }
  ]
}

```

Fig. 88: Prueba de visualización de comentarios con Postman.

The screenshot shows a POST request to `https://tecnony-v1.herokuapp.com/api/v1/register-tecnico`. The response body is a JSON object:

```

{
  "message": "Usuario registrado exitosamente"
}

```

Fig. 89: Prueba de registro de técnico con Postman.

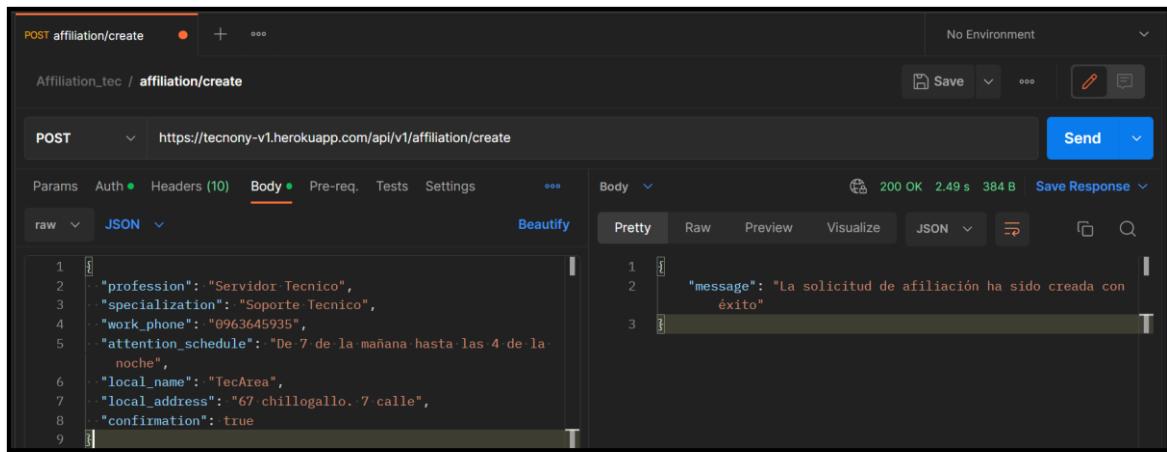


Fig. 90: Prueba de solicitar afiliación con Postman.

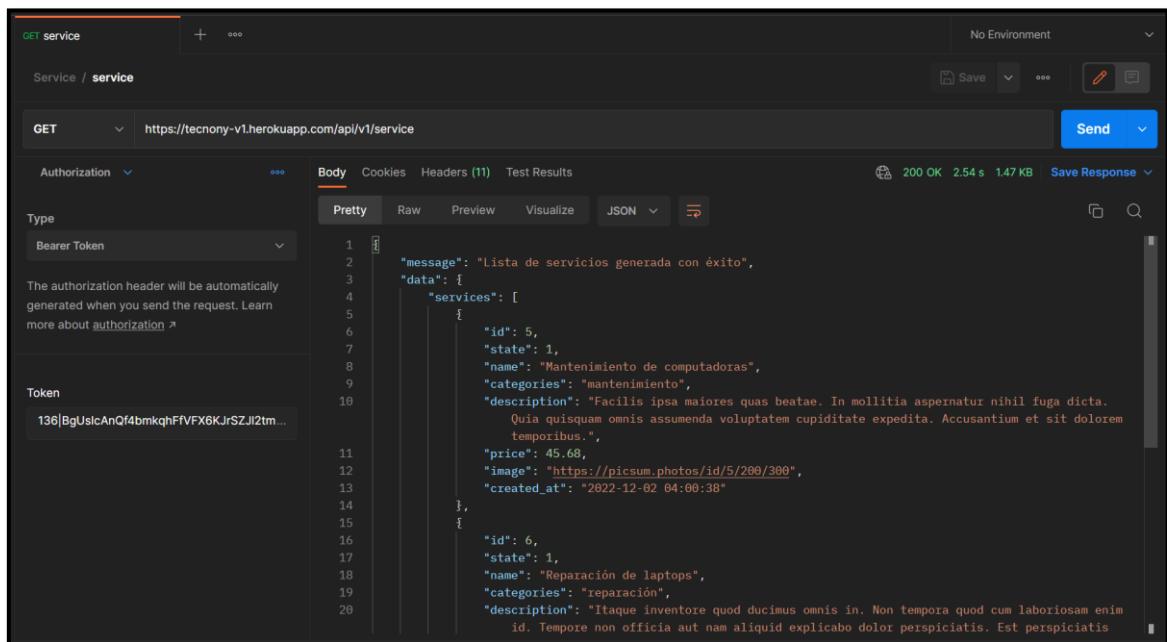


Fig. 91: Prueba de gestión de contrataciones con Postman.

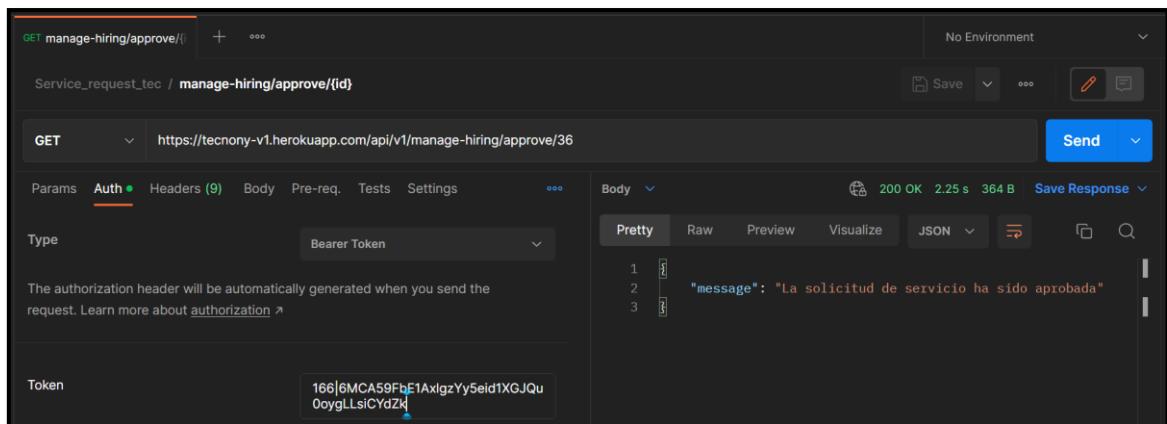


Fig. 92: Prueba de aprobación de contratación con Postman.

```

{
  "message": "Las solicitudes de servicios comentados se han devuelto con éxito",
  "data": [
    {
      "score": 4.57,
      "satisfaction_forms": [
        {
          "id": 13,
          "comment": "Ea voluptatibus qui fugiat qui beatae edit. Commodo hic rerum odio optio. Occaecati quia facilis quis harum reprehenderit in qui.",
          "suggestion": "Voluptatem explicabo molestias architecto et et dolor vitae reiciendis. Minus cum et magni veniam qui exercitationem. Ducimus recusandae vitae harum.",
          "qualification": 2.45
        },
        {
          "id": 14,
          "comment": "Officiis tempora est vero laborum. Quos eos inventore voluptatem. Nostrum magni ipsum distinctio veniam dolor officiis.",
          "suggestion": "Id ad repellendus et voluptas. Vitae exercitationem qui quo et suscipit veritatis.",
          "qualification": 5.76
        }
      ],
      "id": 15
    }
  ]
}

```

Fig. 93: Prueba de visualización de comentarios de los servicios con Postman.

```

{
  "message": "Autenticación exitosa",
  "data": {
    "user": {
      "id": 23,
      "state": 1,
      "username": "Mano1",
      "full_name": "Manolo Perez",
      "cedula": "1723945003",
      "email": "mano1@example.net",
      "role": "cliente",
      "birthdate": "1997-08-15",
      "home_phone": "0296263",
      "personal_phone": "0967002567",
      "address": "4 chilligallo Oval Apt. 78",
      "avatar": "https://cdn-icons-png.flaticon.com/512/848/848006.png"
    },
    "access_token": "1671nDJEYkACvjs3CQIDmafTOoJVLx1bHP3N3DQ2d15D",
    "token_type": "Bearer"
  }
}

```

Fig. 94: Prueba de inicio de sesión para clientes con Postman.

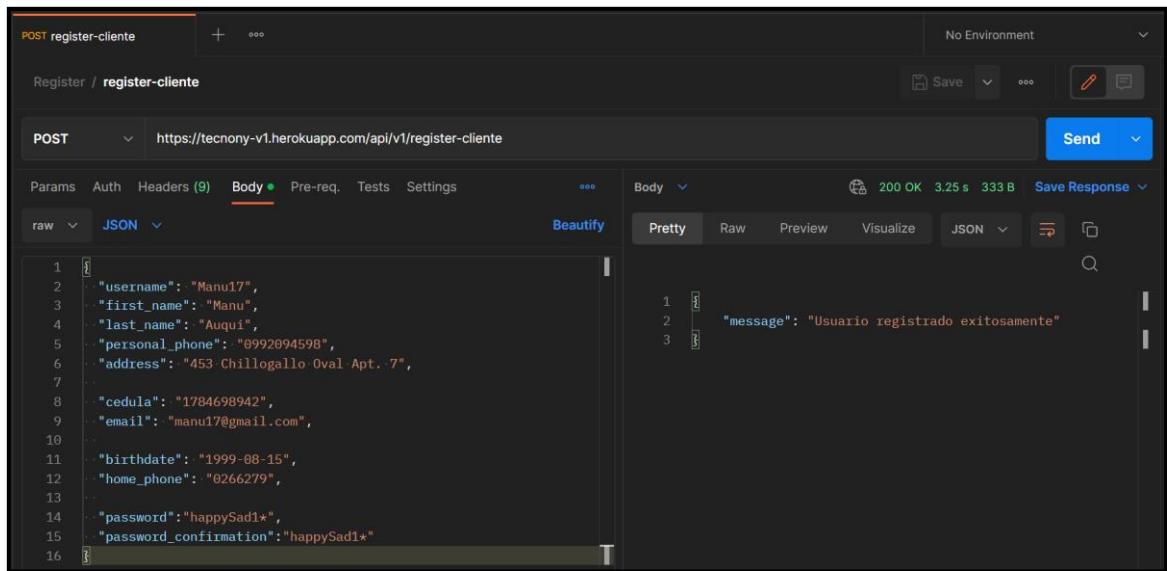


Fig. 95: Prueba de registro de clientes con Postman.

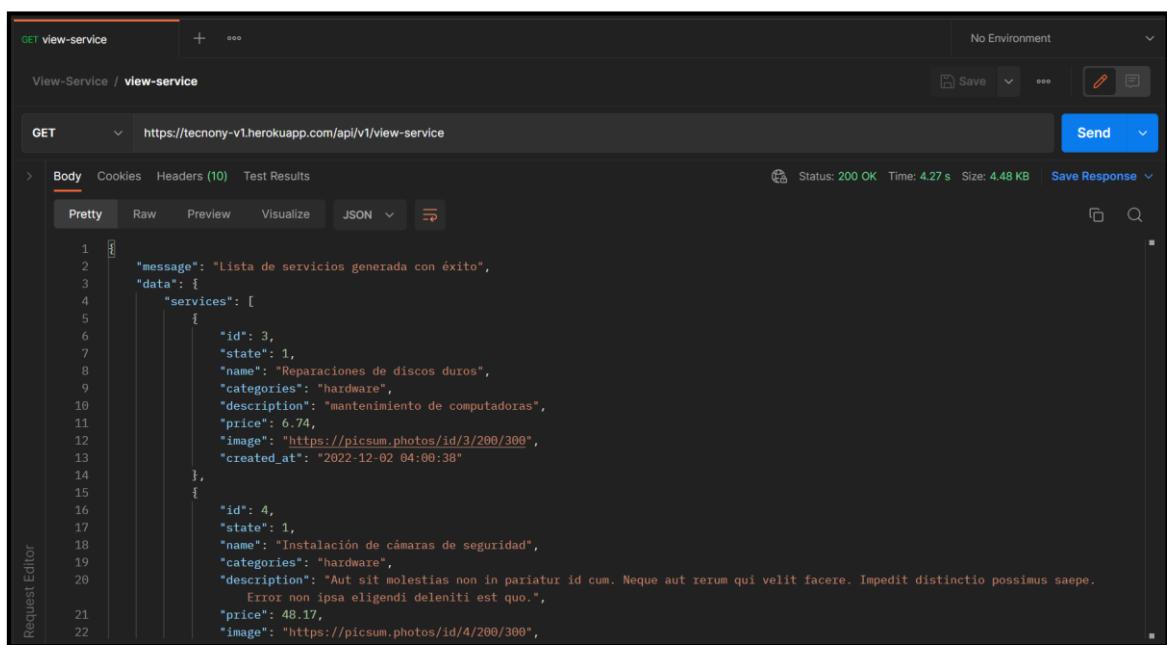


Fig. 96: Prueba de visualización de servicios con Postman.

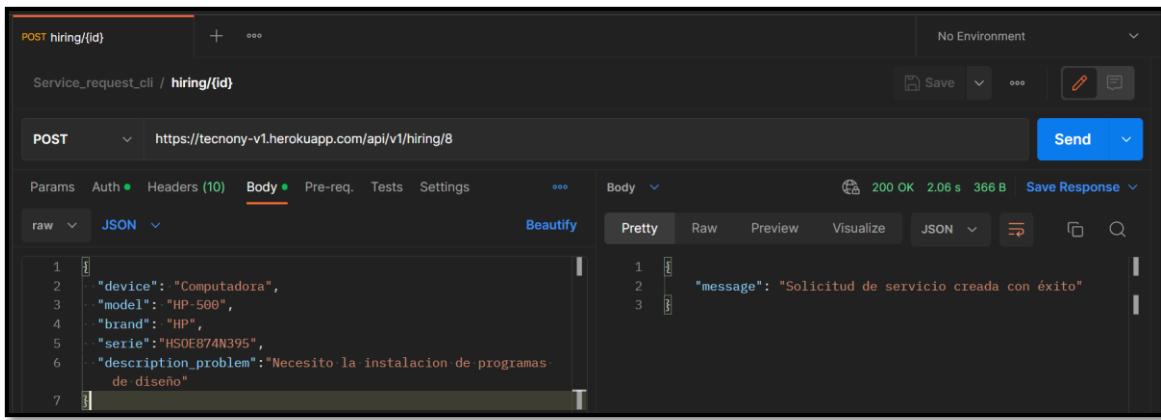


Fig. 97: Prueba de contratación de servicios con Postman.

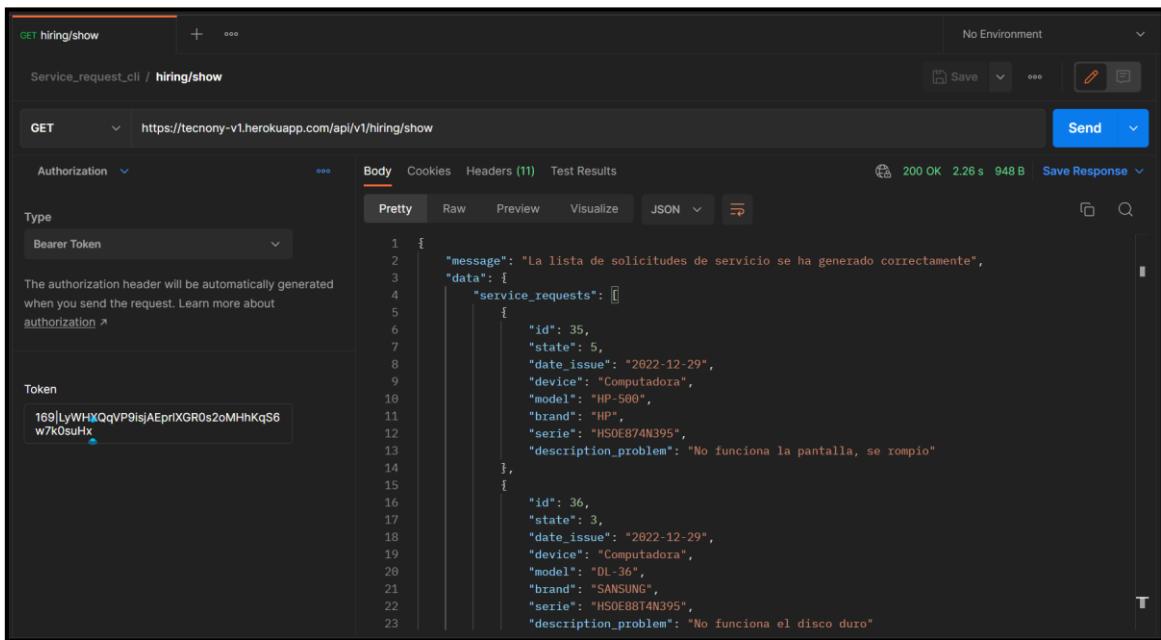


Fig. 98: Prueba de gestión de contratación con Postman.

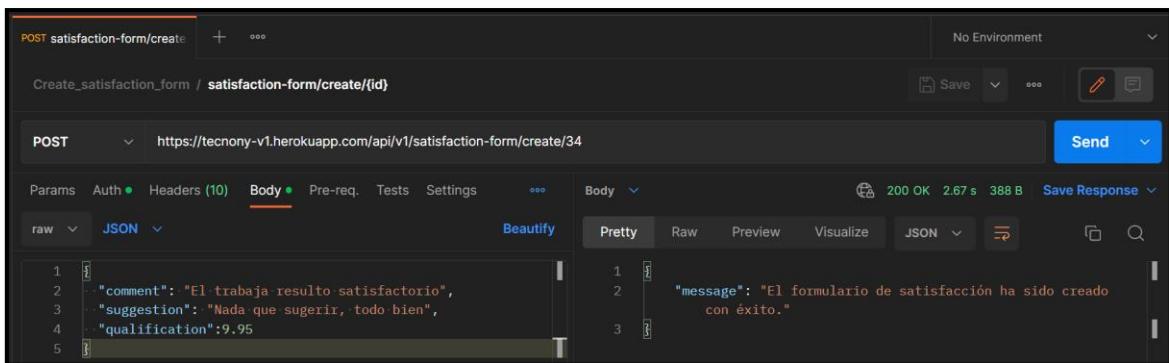


Fig. 99: Prueba de comentar, sugerir y calificar servicios con Postman.

Prueba de carga

En este apartado se presenta las 13 pruebas de carga que han sido elaboradas en base a las historias de usuario que han sido desarrollados previamente, las cuales se muestran desde **Fig. 100** hasta **Fig. 111**.

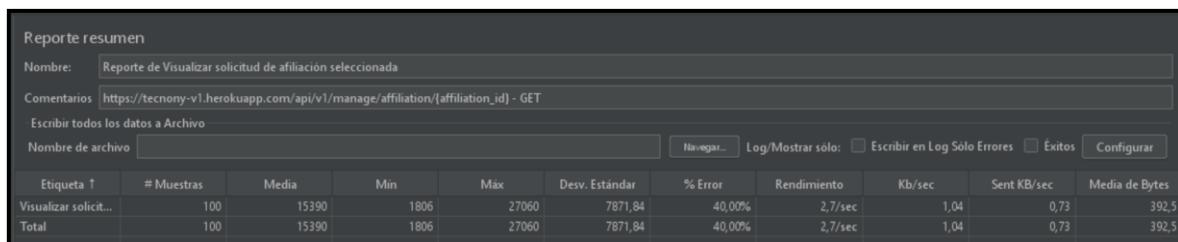


Fig. 100: Prueba de carga a visualizar solicitud de afiliación seleccionada.

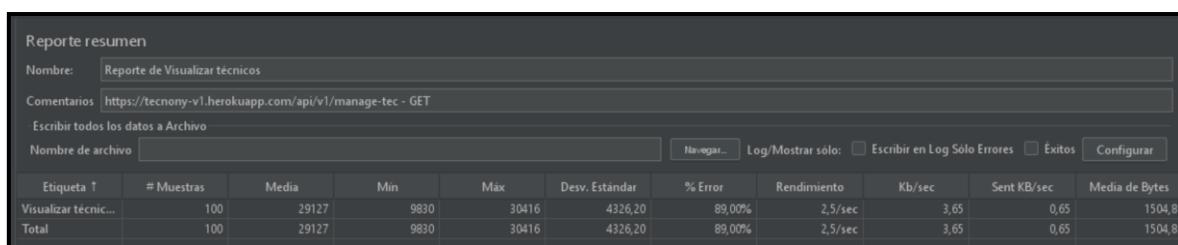


Fig. 101: Prueba de carga a visualizar técnicos.

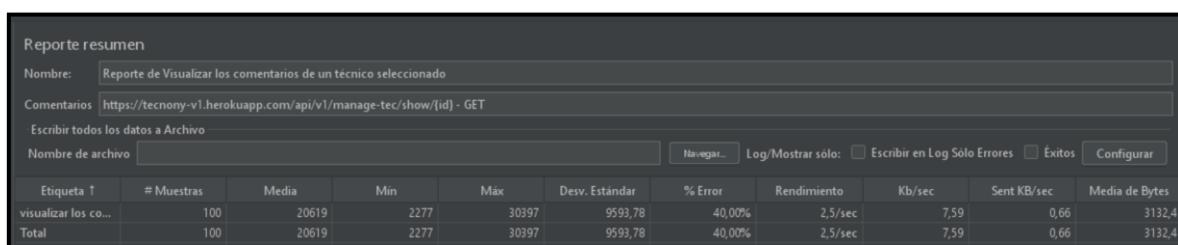


Fig. 102: Prueba de carga a visualizar los comentarios de un técnico seleccionado.

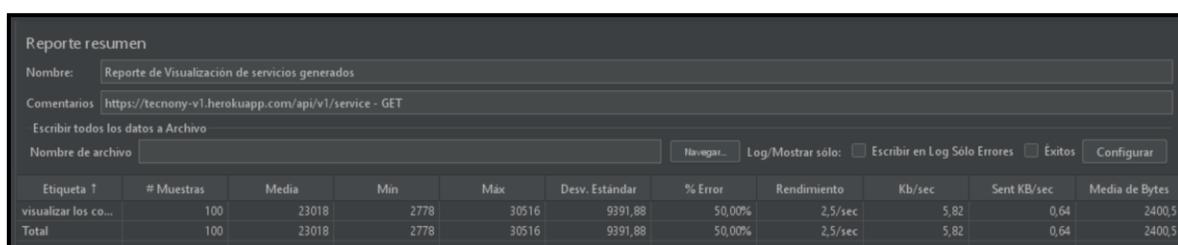


Fig. 103: Prueba de carga a visualización de servicios generados.

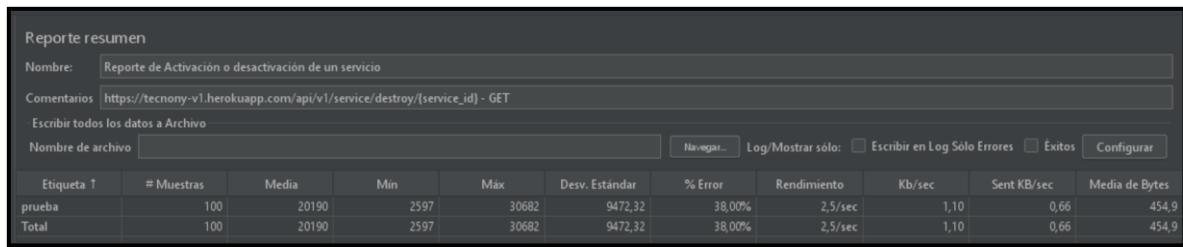


Fig. 104: Prueba de carga a activación o desactivación de un servicio.



Fig. 105: Prueba de carga a visualización de solicitudes de contratación.

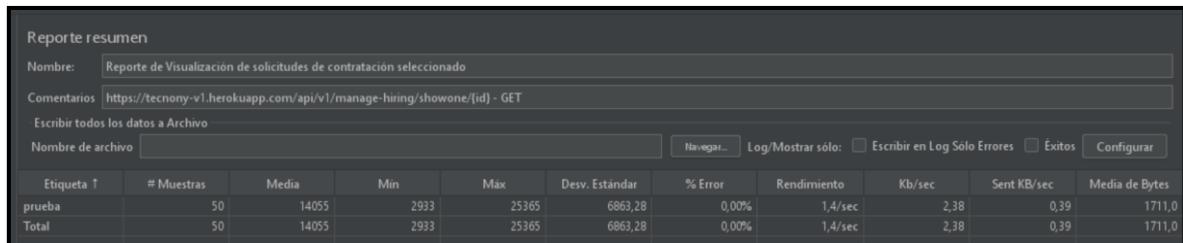


Fig. 106: Prueba de carga a visualización de solicitudes de contratación seleccionado.

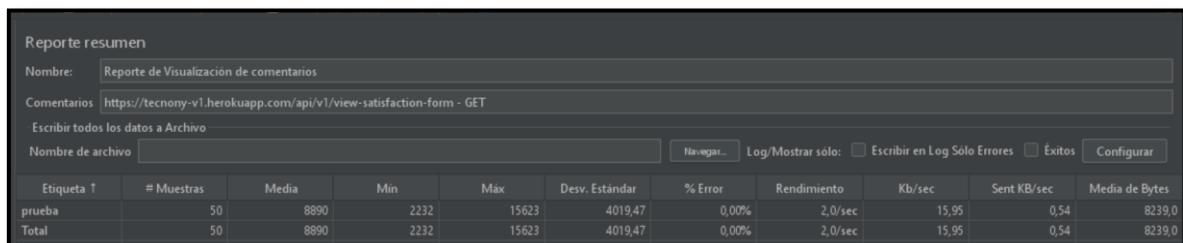


Fig. 107: Prueba de carga a visualización de comentarios.

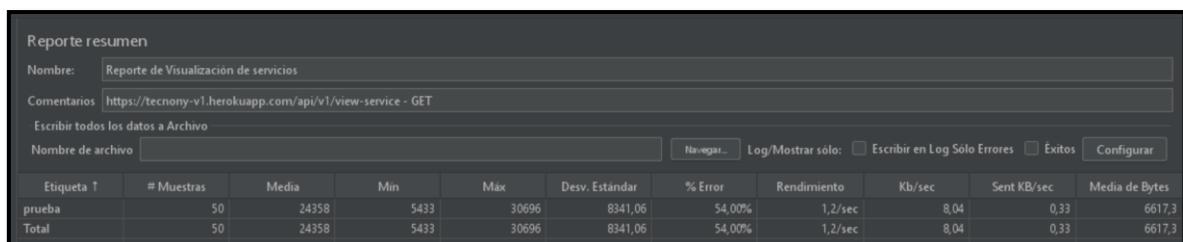


Fig. 108: Prueba de carga a visualización de servicios.

Reporte resumen										
Nombre: Reporte de Visualización de solicitudes de contratación Comentarios https://tecnony-v1.herokuapp.com/api/v1/hiring/show - GET <input type="checkbox"/> Escribir todos los datos a Archivo Nombre de archivo <input type="text"/> <input type="button" value="Navegar..."/> Log/Mostrar solo: <input type="checkbox"/> Escribir en Log Sólo Errores <input type="checkbox"/> Éxitos <input type="button" value="Configurar"/>										
Etiqueta †	# Muestras	Media	Min	Max	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
prueba	50	27497	9586	30672	6209,46	76,00%	1,2/sec	3,43	0,33	2822,0
Total	50	27497	9586	30672	6209,46	76,00%	1,2/sec	3,43	0,33	2822,0

Fig. 109: Prueba de carga a visualización de solicitudes de contratación.

Reporte resumen										
Nombre: Reporte de Visualizar contratación seleccionada Comentarios https://tecnony-v1.herokuapp.com/api/v1/hiring/show/{id} - GET <input type="checkbox"/> Escribir todos los datos a Archivo Nombre de archivo <input type="text"/> <input type="button" value="Navegar..."/> Log/Mostrar solo: <input type="checkbox"/> Escribir en Log Sólo Errores <input type="checkbox"/> Éxitos <input type="button" value="Configurar"/>										
Etiqueta †	# Muestras	Media	Min	Max	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
prueba	50	20999	3902	30395	9233,93	34,00%	1,2/sec	2,09	0,33	1716,5
Total	50	20999	3902	30395	9233,93	34,00%	1,2/sec	2,09	0,33	1716,5

Fig. 110: Prueba de carga a visualizar contratación seleccionada.

Reporte resumen										
Nombre: Reporte de Cancelar una solicitud de contratación Comentarios https://tecnony-v1.herokuapp.com/api/v1/hiring/cancel/{id} - GET <input type="checkbox"/> Escribir todos los datos a Archivo Nombre de archivo <input type="text"/> <input type="button" value="Navegar..."/> Log/Mostrar solo: <input type="checkbox"/> Escribir en Log Sólo Errores <input type="checkbox"/> Éxitos <input type="button" value="Configurar"/>										
Etiqueta †	# Muestras	Media	Min	Max	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
prueba	50	10616	2153	19158	4564,52	0,00%	1,7/sec	0,65	0,46	387,3
Total	50	10616	2153	19158	4564,52	0,00%	1,7/sec	0,65	0,46	387,3

Fig. 111: Prueba de carga a cancelar una solicitud de contratación.

ANEXO III

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

<https://youtu.be/x2TYIOLG98Y>

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

Finalmente, se presenta las credenciales de acceso para el consumo de las API's generadas para cada uno de los *endpoints* elaborados en el trascurso del desarrollo del *backend*, además del vínculo al repositorio de GitHub donde se encuentra alojado el código del proyecto, junto con el README en la cual se describe los pasos para el consumo de los *endpoints*.

Credenciales de acceso

Para acceder al *backend* desplegado, utilizar el siguiente vinculo:

<https://tecnony-v1.herokuapp.com/>

Credenciales para el usuario con perfil administrador:

- **Correo:** admin@example.com
- **Contraseña:** happysad

Credenciales para el usuario con perfil técnico:

- **Correo:** tecnico@example.com
- **Contraseña:** happysad

Credenciales para el usuario con perfil cliente:

- **Correo:** cliente@example.com
- **Contraseña:** happysad

Repositorio de GitHub

Para acceder repositorio, utilizar el siguiente vinculo:

<https://github.com/ManuEly19/tecnony.git>