Nicolás Mansotti

Plataforma web para gestión de turnos y usuarios en laboratorios remotos

2023

Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía
Ingeniería en Informática

Práctica Profesional Supervisada
Informe Final

Plataforma web para gestión de turnos y usuarios en laboratorios remotos

Estudiante:
Nicolás Mansotti

Florencio Varela, Diciembre 2023

DATOS DEL ESTUDIANTE

Apellido y Nombres: Mansotti Nicolás

Nº de Legajo: 7490

Correo electrónico: mansottinicolas@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 44 materias con finales incluidos PPS enmarcada en artículo 7 inciso A de la Resolución (CS) 103/16

Periodo en que se realizó la PPS: junio del 2023 hasta diciembre del 2023

DOCENTE SUPERVISOR

Apellido y Nombre: Ing. Ayala, María Florencia.

Correo electrónico: florenciaayala.unaj@gmail.com

DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS DE LA UNAJ

Apellido y Nombres: Mg. Leone, Nelson

Correo electrónico: nelsonleone2012@gmail.com

DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS

Nombre o Razón Social: Universidad Nacional Arturo Jauretche

Dirección: Av. Calchaquí 6200

Teléfono: +54 11 4275-6100

Sector: Laboratorio de Informática

TUTOR ORGANIZACIONAL

Apellido y Nombres: Dr. Ing. Morales, Martín

Correo electrónico: martin.morales@unaj.edu.ar

FIRMA DEL COORDINADOR DE LA CARRERA

Resumen

Este proyecto de Práctica Profesional Supervisada (PPS) tiene por objetivo investigar y desarrollar un sistema para gestionar turnos y usuarios en laboratorios remotos, integrados en una aplicación web de la UNAJ. Los estudiantes pueden reservar turnos sin superposiciones y acceder a los laboratorios de forma remota. La aplicación permite a los usuarios visualizar sus turnos, la disponibilidad de los laboratorios y les proporciona URLs para acceder a los turnos reservados. Además, un administrador tiene acceso para configurar laboratorios, duración, horarios, monitorear los turnos y filtrar las actividades de los alumnos. La PPS se llevó a cabo en colaboración con mi compañero Alejandro Méndez, teniendo cada uno tareas específicas: el desarrollo del *backend* y *frontend*. Este trabajo abordó la centralización de la gestión de turnos en un entorno virtual de enseñanza y aprendizaje, proporcionando una solución integral para la reserva y acceso a los laboratorios remotos de la universidad.

Palabras clave: Gestión de usuarios, Laboratorios remotos, Aplicación web, Reserva de turnos

Abstract

This Supervised Professional Practice (SPP) project focuses on researching and developing a system to manage appointments and users in remote laboratories, integrated into a UNAJ web application. Students can book appointments without overlaps and access the labs remotely. The application enables users to view their appointments, lab availability, and provides URLs to access reserved slots. Additionally, an administrator has access to configure labs, duration, schedules, monitor appointments, and filter student activities. The

SPP was carried out collaboratively with Alejandro Méndez, each assigned specific tasks: backend and frontend development. This project addressed centralizing appointment management in a virtual teaching and learning environment, offering a comprehensive solution for reserving and accessing university remote labs.

Keywords: User management, Remote labs, Web application, Appointment scheduling

Dedicatorias y agradecimientos

Agradecer a mis tutores en este trabajo por acompañarme estos meses.

A Florencia Ayala, por su guía y apoyo en esta Práctica Profesional Supervisada.

A Nelson Leone, por las contribuciones hechas y la orientación en la confección de este informe.

A Martín Morales, por la organización de la PPS.

A la Universidad Nacional Arturo Jauretche por las herramientas que aportaron a lo largo de este camino. Gracias a los profesores, compañeros y amigos con los que compartí cursadas.

Agradecer y dedicar este trabajo y la culminación de esta etapa a mi familia, especialmente a mi mamá y mi papá, por impulsarme a ser mejor, y por el apoyo y la ayuda que me brindan en cada paso que doy.

Índice

1. Introducción

La presente Práctica Profesional Supervisada (PPS) consiste en un trabajo de investigación e implementación de un sistema de gestión de turnos y usuarios para el acceso y el uso de laboratorios remoto en el Entorno Virtual de Enseñanza y Aprendizaje, con el objetivo de centralizar y alojar en una aplicación web todos los laboratorios remotos que tenga la UNAJ, en donde los alumnos puedan crearse un usuario, reservar turnos y poder acceder a los laboratorios disponibles. El sistema de turnos contempla que no haya superposiciones entre los alumnos en la utilización de los Laboratorios Remotos.

Se implementará un sistema de registración de usuarios, donde luego los alumnos podrán visualizar los turnos que tienen, los laboratorios disponibles, sus horarios y días en los que podrán sacar un turno. El sistema diseñado, habilita la URL(Localizador de recursos uniforme) para acceder al laboratorio remoto en el día y horario que el alumno haya reservado el turno.

Además, la aplicación web cuenta con la posibilidad de que un administrador cargue y configure los laboratorios disponibles, junto con una serie de configuraciones, como por ejemplo la duración particular de cada laboratorio y los días y horarios en los que los laboratorios estarán disponibles. Además, el administrador tiene permisos y funcionalidades especiales donde podrá monitorear con más información los turnos activos y disponibles y podrá filtrar por usuarios para ver la actividad de cada uno de los alumnos.

La PPS se realiza en forma conjunta con mi compañero de carrera Alejandro Mendez, en donde cada uno tiene un rol y tareas asignadas en el proyecto. La aplicación está compuesta por el *backend* y el *frontend*. El *backend* es la parte de una aplicación que se encarga del procesamiento y gestión de datos, la lógica de negocio y la interacción con la

base de datos. El *frontend*, en cambio, corresponde a la interfaz visible para el usuario final, encargada de presentar la información y facilitar la interacción con el sistema.

En mi caso, trabajé en la parte del *backend* de la aplicación y Alejandro trabajó en la parte del *frontend*. Fue un trabajo en conjunto, ya que a lo largo de lo que duró el proyecto, fuimos teniendo reuniones para definir funcionalidades de la aplicación web.

2. Temas de estudio, análisis e investigación

A continuación, se detallan los tópicos de estudio abordados para la realización del laboratorio remoto y su comprensión como herramienta de aprendizaje en el contexto académico.

2.1 Sistema de gestión de turnos y usuarios

El objetivo de crear un sistema de gestión de turnos y usuarios es poder unificar y alojar en un mismo lugar a todos los laboratorios remotos de la UNAJ, para poder tener una mejor organización, tener más control sobre el acceso a los laboratorios y un acceso unificado y simplificado, y que los usuarios al acceder a la aplicación puedan visualizar todos los laboratorios disponibles. Cada alumno tiene la posibilidad de crearse un usuario y así poder reservar turnos para realizar las pruebas que deseen necesarias en los laboratorios.

El sistema de gestión de turnos y usuarios también busca optimizar la asignación de recursos en los laboratorios. Permite a los administradores supervisar la disponibilidad de cada laboratorio en términos de duración, horarios y días disponibles. Esto asegura una planificación más efectiva y un uso óptimo de los espacios y equipos.

Para los alumnos, ofrece una interfaz intuitiva y amigable. No solo les permite reservar turnos, sino que al mismo tiempo les proporciona información detallada sobre la disponibilidad de los laboratorios y les permite revisar el estado de sus reservas de manera fácil y rápida.

Además, el sistema implementará un mecanismo de notificaciones mediante correo electrónico para recordarle a los usuarios sus turnos reservados. Esto asegura una comunicación efectiva y contribuye a una experiencia más satisfactoria para quienes utilizan el sistema.

2.2 Laboratorios remotos

Los laboratorios remotos se refieren a recursos de acceso remoto, donde representan el acceso a entornos reales de experimentación, pero sin la necesidad de estar físicamente presentes en el lugar. Este acceso se lleva a cabo mediante internet y permite a los estudiantes controlar procesos o dispositivos a distancia.

Un laboratorio remoto refleja las acciones de un estudiante que interactúa con equipos y dispositivos reales a través de una red. En contraste con programas de simulación, estos laboratorios permiten el uso y control directo de recursos en entornos reales mediante sensores e instrumentación.

La mayoría de los laboratorios remotos se enfocan en áreas de la ingeniería como la física y las matemáticas. Sus características principales incluyen:

- Acceso en tiempo real que permite a los usuarios estudiar, observar y recopilar datos de elementos en observación.

- Utilización de sensores para transmitir información en tiempo real desde el lugar físico de experimentación a las terminales conectadas, facilitando la recolección de datos para verificar hipótesis preestablecidas.
- Posibilidad de analizar los datos recopilados mediante modelos matemáticos para establecer conclusiones y compararlas con las hipótesis iniciales.
- La planificación previa de objetivos, competencias, metodologías y actividades es fundamental para garantizar el éxito de los laboratorios remotos.

Los elementos clave que interactúan en un laboratorio remoto son:

- Alumnos o usuarios: individuos que realizan experimentos remotamente a través de una computadora conectada a internet.
- Servidor de enlace: encargado de administrar las conexiones de los usuarios interesados en el Laboratorio Remoto.
- Servidor multimedia: proporciona imágenes y audio del ambiente experimental en tiempo real al servidor de enlace.
- Servidor de laboratorio: ofrece soporte a aplicaciones específicas y gestiona la base de datos conectada al servidor de enlace.
- Objeto experimental: controlado por el servidor de laboratorio, permite adquirir mediciones o realizar acciones que se pueden observar o escuchar gracias al servidor multimedia.

3. Metodología de trabajo

Al ser un trabajo entre 2 personas, al inicio del proyecto se acordaron y definieron las actividades y las tareas que cada uno iba a llevar a cabo y a desarrollar. Por mi parte, en el

trabajo me encargué de la parte del *backend* de la aplicación. Por el lado de Alejandro, se encargó de la parte del *frontend* de la aplicación web.

Dentro de mis tareas, se incluyen:

1. Capacitaciones y estudios.
2. Relevar requerimientos.
3. Elegir, diseñar y crear la base de datos.
4. Elegir el lenguaje de programación adecuado para el desarrollo de la aplicación.
5. Diseñar, modelar y crear los servicios para cada funcionalidad de la aplicación web.
6. Realizar modificaciones en la base de datos y servicios a medida que iban surgiendo nuevas necesidades y requerimientos en la aplicación.
7. Realizar pruebas de servicios y de los flujos completos de la aplicación web.

La comunicación con Alejandro era semanal o diaria, mediante mensajes o reuniones virtuales. El procedimiento de trabajo era ir haciendo cada módulo y mediante reuniones definir y coordinar las distintas características que iban a tener los servicios de *backend* si surgía una modificación, ya sea los métodos que se iban a usar, los campos del *request* (las solicitudes con datos específicos que el *frontend* envía al *backend*), la respuesta de cada servicio, etc. El armado de los servicios de *backend* se definió en base a las funcionalidades y requerimientos de la aplicación. Cualquier modificación que se requería, se agregaba para suplir las necesidades de cada funcionalidad.

A su vez, a medida que se iban completando módulos de los servicios de *backend*, Alejandro tenía que integrarlos con el *frontend*. Para realizar esto, Alejandro los integraba y luego teníamos una reunión para ver si lo establecido había quedado funcional y de acuerdo

a lo pactado y esperado, o si habría que hacer alguna modificación en alguna funcionalidad o método del *backend*.

La comunicación con Florencia Ayala era por mensajes y cada 2 o 3 semanas teníamos reuniones virtuales, en donde se revisaba cómo iban los avances del trabajo y que cosas se podían agregar, modificar o quitar. En estas reuniones le realizaba consultas y dudas que iban surgiendo a medida que se avanzaba con el desarrollo de la API(Interfaz de programación de aplicaciones), en las que ella me respondía y a partir de eso, definimos nuevas funcionalidades y/o realizamos cambios en un servicio.

4. Tecnologías utilizadas

Java

Es un lenguaje de programación de alto nivel, orientado a objetos y multiplataforma. Se utiliza en el desarrollo de aplicaciones empresariales y de *software*. Java destaca por su portabilidad y su capacidad para ejecutarse en distintos entornos sin modificar el código fuente. Ofrece una amplia gama de librerías y herramientas que facilitan el desarrollo de aplicaciones robustas y escalables.

Spring Boot

Es el framework elegido para la creación de aplicaciones Java basadas en Spring. Ofrece una configuración sencilla y una estructura convencional para simplificar el desarrollo, eliminando la necesidad de una configuración exhaustiva. Proporciona características como inyección de dependencias y automatización de tareas, lo que acelera la creación de aplicaciones web y servicios.

Spring Tool Suite 4 (STS)

Es un entorno de desarrollo integrado (IDE) basado en Eclipse, diseñado específicamente para trabajar con tecnologías de Spring. Ofrece herramientas y complementos que facilitan el desarrollo y la depuración de aplicaciones basadas en Spring, proporcionando características como autocompletado, depuración y soporte para pruebas unitarias.

Docker

Es una plataforma de código abierto que permite empaquetar, distribuir y ejecutar aplicaciones en contenedores. Estos contenedores son entornos ligeros y portables que encapsulan la aplicación y sus dependencias, garantizando la consistencia en distintos entornos, desde el desarrollo hasta la producción.

Docker se compone de varios elementos que conforman un ecosistema en el que se potencia el desarrollo, el despliegue y la gestión de aplicaciones en contenedores, ofreciendo portabilidad, escalabilidad y aislamiento para las aplicaciones.

Los elementos son:

- Imagen de Docker: una imagen es un paquete que contiene todo lo necesario para ejecutar una aplicación: el código, las bibliotecas, las herramientas, las configuraciones y cualquier otro elemento necesario. Se crean a partir de un archivo llamado Dockerfile, que contiene instrucciones para construir la imagen capa por capa. Las imágenes son inmutables y se utilizan como plantillas para crear contenedores. Pueden ser almacenadas localmente o en un registro remoto.
- Contenedor Docker: un contenedor es una instancia en ejecución de una imagen. Es una especie de espacio aislado en el sistema operativo que contiene todo lo necesario para ejecutar una aplicación. Los contenedores son ligeros, portátiles y se ejecutan de manera independiente, aislada del entorno del sistema operativo

subyacente. Permiten la ejecución de aplicaciones de manera consistente en diferentes entornos.

- Docker Hub: es un servicio en la nube que actúa como un repositorio centralizado de imágenes de contenedores. Permite compartir, buscar, descargar y almacenar imágenes. Proporciona imágenes públicas gratuitas de cualquier tecnología, para usar directamente en el proyecto y también permite crear repositorios privados para almacenar imágenes de manera segura.
- Docker Registry: es un repositorio donde se almacenan las imágenes de contenedores. Puede ser público o privado y actúa como un almacén centralizado para guardar y distribuir imágenes.
- Dockerfile: es un archivo de texto plano que contiene instrucciones detalladas sobre cómo construir una imagen de Docker capa por capa. En este archivo se define la configuración del entorno, las dependencias y los comandos necesarios para crear una imagen. Los Dockerfiles son la base para reproducir y compartir ambientes de desarrollo y producción de manera consistente.
- Docker Compose: es una herramienta que permite definir y gestionar aplicaciones multicontenedor. Utiliza un archivo YAML para configurar los servicios de una aplicación, lo que facilita el manejo de múltiples contenedores y su interacción. Docker Compose simplifica la orquestación de aplicaciones complejas con múltiples servicios y contenedores.
- Docker Engine: es el responsable de la creación y ejecución de contenedores. Proporciona una interfaz para la interacción entre los contenedores y el sistema

operativo subyacente. Gestiona los recursos del sistema, como el acceso a la red, la administración de almacenamiento y la ejecución de los contenedores.

MySQL

Es un sistema de gestión de bases de datos relacional, ampliamente utilizado por su rendimiento, confiabilidad y facilidad de uso. Permite el almacenamiento y la gestión eficiente de grandes volúmenes de datos, ofreciendo capacidades robustas de consulta y escalabilidad.

MySQL Workbench 8.0 CE

Es una herramienta de diseño visual y desarrollo para MySQL. Proporciona una interfaz gráfica para crear, modelar y administrar bases de datos MySQL, permitiendo realizar consultas, modelar datos y administrar esquemas de bases de datos de manera intuitiva.

JWT

JWT(JSON Web Token) es un estándar definido en el documento RFC 7519 que ofrece un mecanismo seguro para compartir la identidad de un usuario entre dos partes. Este intercambio se realiza a través de una cadena de texto que tiene tres partes codificadas en Base64, separadas por puntos.

Un token JWT tiene un encabezado que especifica el tipo de token y el algoritmo utilizado, un cuerpo que contiene datos del usuario y privilegios codificados en formato JSON(notación de objeto de JavaScript), y una firma que se genera utilizando un algoritmo de cifrado basado en HMACSHA256.

La firma se utiliza para verificar la autenticidad del remitente y garantizar que el mensaje no se haya modificado en tránsito. Se genera aplicando una función *hash* a la codificación en

Base64 del encabezado, la codificación en Base64 del cuerpo y un secreto compartido establecido por la aplicación.

Al haber una autenticación, el ciclo de vida de un token JWT implica que un cliente realiza una petición de inicio de sesión, se verifica la información de inicio de sesión, se genera un token JWT y se devuelve al cliente. Posteriormente, el cliente utiliza este token en sus solicitudes al servidor, el cual verifica la firma para asegurar la autenticidad del usuario y decide si conceder o denegar el acceso a los recursos solicitados.

Figura 1. Ciclo de vida de un token JWT.

Fuente: Qué es Json Web Token y cómo funciona

Disponible en https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

Postman

Es una plataforma de colaboración para el desarrollo de API que permite probar, desarrollar y documentar APIs de manera sencilla. Ofrece una interfaz gráfica para enviar solicitudes HTTP a un servidor y analizar las respuestas, facilitando la creación y el análisis de API RESTful.

Trello

Es una herramienta de gestión de proyectos basada en tableros, que permite organizar tareas, asignar responsabilidades y realizar un seguimiento del progreso. Con una interfaz intuitiva tipo Kanban, Trello ayuda a visualizar y priorizar actividades, facilitando la colaboración y el trabajo en equipo.

Git

Es un sistema de control de versiones distribuido que permite el seguimiento de cambios en el código fuente de manera eficiente y colaborativa. Permite gestionar el historial de cambios, trabajar en ramas de desarrollo, fusionar código y coordinar el trabajo en equipo, ofreciendo un entorno robusto para el desarrollo de *software*.

Dentro de las tecnologías elegidas a utilizar, las más determinantes eran el lenguaje de programación con el que se iba a hacer la API, y que base de datos se iba a usar.

En este contexto, se eligieron Java y MySQL. A continuación se explicarán los beneficios y el porqué de la elección de ambas decisiones.

Java:

- Portabilidad y versatilidad: Java es un lenguaje de programación altamente portátil, es decir, que el código Java puede ejecutarse en múltiples plataformas sin cambios significativos y funciona en distintos sistemas operativos sin problemas.
- Amplia comunidad y soporte: Java tiene una de las comunidades más grandes y activas en el mundo del desarrollo de *software*. La comunidad proporciona una amplia gama de librerías, frameworks y herramientas que facilitan el desarrollo y mantienen a Java actualizado con las últimas mejoras.
- Seguridad y estabilidad: es un lenguaje de programación robusto y seguro. Las actualizaciones regulares y el enfoque en la seguridad han posicionado a Java como una opción confiable para aplicaciones críticas y sensibles.

Beneficios:

- Multiplataforma: su capacidad para ejecutarse en diferentes sistemas operativos facilita el desarrollo de aplicaciones que no están limitadas a una plataforma específica.
- Orientado a objetos: la programación orientada a objetos de Java promueve la reutilización de código, facilitando el mantenimiento y la escalabilidad de las aplicaciones.
- Escalabilidad y rendimiento: Java está diseñado para manejar aplicaciones de gran tamaño y alta demanda, ofreciendo un rendimiento sólido incluso en entornos complejos.

Además, influyó en la elección, el haber trabajado con Java en mi carrera profesional y tener experiencia en este lenguaje de programación.

MySQL:

- Facilidad de uso y administración: MySQL es conocido por su facilidad de instalación, configuración y mantenimiento.
- Confiabilidad y rendimiento: es una base de datos robusta y estable que ofrece un rendimiento rápido, incluso con grandes conjuntos de datos. Esto la hace adecuada para aplicaciones que necesitan manejar una gran cantidad de información.
- Amplia adopción y soporte: es una de las bases de datos relacionales más populares y cuenta con una comunidad activa de desarrolladores y una amplia documentación.

Beneficios:

- Software Libre: MySQL es una opción de código abierto, lo que significa que es gratuita para la mayoría de los casos de uso, lo que reduce los costos de licencia.
- Eficiencia y velocidad: su capacidad para manejar consultas complejas y grandes volúmenes de datos con eficiencia lo convierte en una elección confiable para aplicaciones que requieren tiempos de respuesta rápidos.
- Compatibilidad: Ofrece una amplia compatibilidad con muchas aplicaciones y frameworks, lo que facilita su integración en diferentes entornos de desarrollo.

En la elección de la base de datos, estaba en duda si usar Oracle o MySQL. Me decidí por MySQL por ser de código abierto, ya que Oracle es propiedad de Oracle Corporation y su licencia suele ser paga.

Además, Oracle es conocido por ser robusto, complejo y escalable, adecuado para grandes proyectos y empresas con requisitos complejos. En cambio MySQL es más ligero y fácil de usar, adecuado para aplicaciones de tamaño mediano o pequeño.

Además, al utilizar Docker, como MySQL es de código abierto y tiene una comunidad activa, es más fácil encontrar imágenes y soluciones en Docker. Igualmente en este punto ambos tienen soluciones en Docker, pero me incliné por MySQL por los otros beneficios y ventajas.

5. Desarrollo de las tareas

5.1 Etapas del trabajo

Inicialmente, el enfoque fue armar la estructura inicial de la base de datos con las tablas necesarias y definir los servicios principales, entendiendo cuáles serían los flujos de la aplicación web y las problemáticas y la interacción con los laboratorios y los usuarios. Luego una vez que estuvo creada la base de datos y los servicios principales funcionando, integramos las partes de *backend* y *frontend*. Estas integraciones se hicieron subiendo los servicios de *backend* a docker para que el *frontend* pueda consumirlos y realizar pruebas de la aplicación. Luego surgieron mejoras en los servicios de *backend*, validaciones de datos, seguridad de la aplicación y envío de mails. Por último se probó el flujo completo de la aplicación web integrado con los laboratorios.

5.2 Capacitaciones y estudios que tuve que hacer

En la primera reunión que tuvimos junto con Florencia Ayala, Alejandro Mendez y yo, definimos el alcance del proyecto, las tecnologías a usar, la división de tareas y otras

cuestiones referidas al trabajo por hacer. Una de las tecnologías que se planteó usar fue Docker. Esta decisión ya fue impuesta por la UNAJ, ya que la universidad trabaja con sus proyectos desplegados en el ambiente de Docker. Esto fue un desafío, ya que no contaba con los conocimientos, experiencias ni formación sobre cómo usar Docker y por ende, tuve que hacer un trabajo de investigación de la herramienta, haciendo cursos, mirando videos y ejemplos sobre cómo utilizarla.

Otra herramienta a aprender fue Portainer. La curva de aprendizaje fue menor que la de Docker. Ya que lo que más importaba era familiarizarse con el entorno y cómo desplegar la aplicación en producción.

Además, dentro del proceso de desarrollo, me encontré investigando temas más específicos y técnicos. Uno de los desafíos fue el de agregarle e implementar seguridad a la aplicación, donde exploré distintas estrategias para mejorar la robustez de la API y la gestión de datos, abordando aspectos como la autenticación de usuarios y la gestión de permisos.

5.3 Relevamiento de requerimientos

En el descubrimiento inicial del proyecto y las primeras reuniones llevadas a cabo con Florencia Ayala y Alejandro Mendez, se realizó un proceso de relevamiento de requerimientos. Este proceso, conocido como *discovery*, tiene como objetivo comprender, identificar y documentar exhaustivamente las necesidades, expectativas y funcionalidades esenciales para el desarrollo de la aplicación web.

A medida que el proyecto avanzó, surgieron nuevos requerimientos y se presentaron dudas sobre cómo implementar ciertas funcionalidades. Este proceso de nuevas funcionalidades

era esperable y es parte integral del ciclo de desarrollo de *software*. Algunos de estos nuevos requerimientos son:

- Token de acceso al laboratorio: surgió para fortalecer la seguridad del acceso a los laboratorios remotos. Esto ayuda a validar y asegurar la identidad de los usuarios, evitando accesos no autorizados o fraudulentos. Además, estos tokens permiten un control más estricto sobre quién accede a qué recursos, lo que mejora significativamente la protección de datos sensibles.
- Permisos del administrador: los permisos del rol de Administrador surgieron para garantizar un control más preciso sobre las acciones que pueden realizar los usuarios con privilegios elevados al momento de administrar la aplicación. Estos permisos le otorgan al Administrador de la aplicación, más funciones y decisiones de control, monitoreo y registro de los laboratorios, turnos y usuarios.
- Mejoras en los servicios de búsqueda: la evolución en los servicios de búsqueda se originó para optimizar la experiencia del usuario administrador. Se agregaron filtros en el *request* de los servicios de búsqueda, como por ejemplo la posibilidad de buscar por fechas, laboratorios, disponibilidad y usuarios. La mejora en la respuesta y la precisión de los resultados busca proporcionar una navegación más efectiva y rápida dentro de la plataforma.
- Mejoras en la respuesta de servicios: estas mejoras se enfocaron en la optimización del rendimiento de la aplicación y en la búsqueda de una estandarización de la estructura de las respuestas de los servicios de la API.
- Mejoras y cambios en la lógica de funcionalidad: surgieron debido a la evolución de los requisitos del usuario o a la identificación de procesos que podrían ser más

eficientes. Estos cambios tienen como objetivo principal mejorar la usabilidad y la lógica interna de la aplicación, adaptándola a las necesidades cambiantes y optimizando su desempeño.

- Mejoras en el manejo de errores e incorporación de mensajes: estas mejoras apuntan a proporcionar más información sobre los errores en la aplicación y asimismo se busca delimitar una estructura y manejo de mensajes que sea conocido por el *frontend*.
- Diseño de la API para el control de token de sesión: este requerimiento surgió debido a que la API podía usarla cualquier persona, esté logueado o no en la aplicación. Esto iba a provocar una vulnerabilidad muy grande en la seguridad, y que no se tenía contemplado en un principio el control de que los que usen y consuman los servicios sean los usuarios de la aplicación, y también es para controlar que los alumnos no puedan consumir servicios que son específicos para el Administrador.

  Este control le aportó una mejora significativa en materia de seguridad de la aplicación.

5.4 Arquitectura de la aplicación

La arquitectura de la aplicación se fundamenta en una estructura cliente-servidor, en donde diferentes componentes interactúan entre sí.

La base de datos MySQL es la fuente principal de persistencia de datos. Esta base de datos es accesible a través de un *backend* desarrollado en Java, que actúa como el servidor encargado de procesar las solicitudes provenientes del *frontend*.

El *backend* expone servicios y una API RESTful que proporcionan puntos de acceso para la interacción y recuperación de datos. Estos servicios son el puente entre la lógica de negocio y la base de datos, permitiendo al *frontend* consumirlos de manera eficiente. Los usuarios, a través de la interfaz del *frontend*, acceden a estos servicios para interactuar con la aplicación, solicitar información, enviar datos y recibir respuestas a sus peticiones.

En esta arquitectura, el *frontend* actúa como la interfaz de usuario, presentando la información de manera amigable y permitiendo a los usuarios interactuar con la aplicación. Los servicios del *backend* facilitan la comunicación entre el *frontend* y la base de datos, asegurando que las operaciones solicitadas por los usuarios se realicen de manera eficaz y segura.

5.5 Base de datos

En una aplicación web, la base de datos es esencial para almacenar, acceder y gestionar la información de manera eficiente y segura, lo que contribuye significativamente a la funcionalidad y efectividad general de la aplicación.

En este apartado, se realizará una explicación detallada de la base de datos utilizada, sus tablas, esquemas y relaciones.

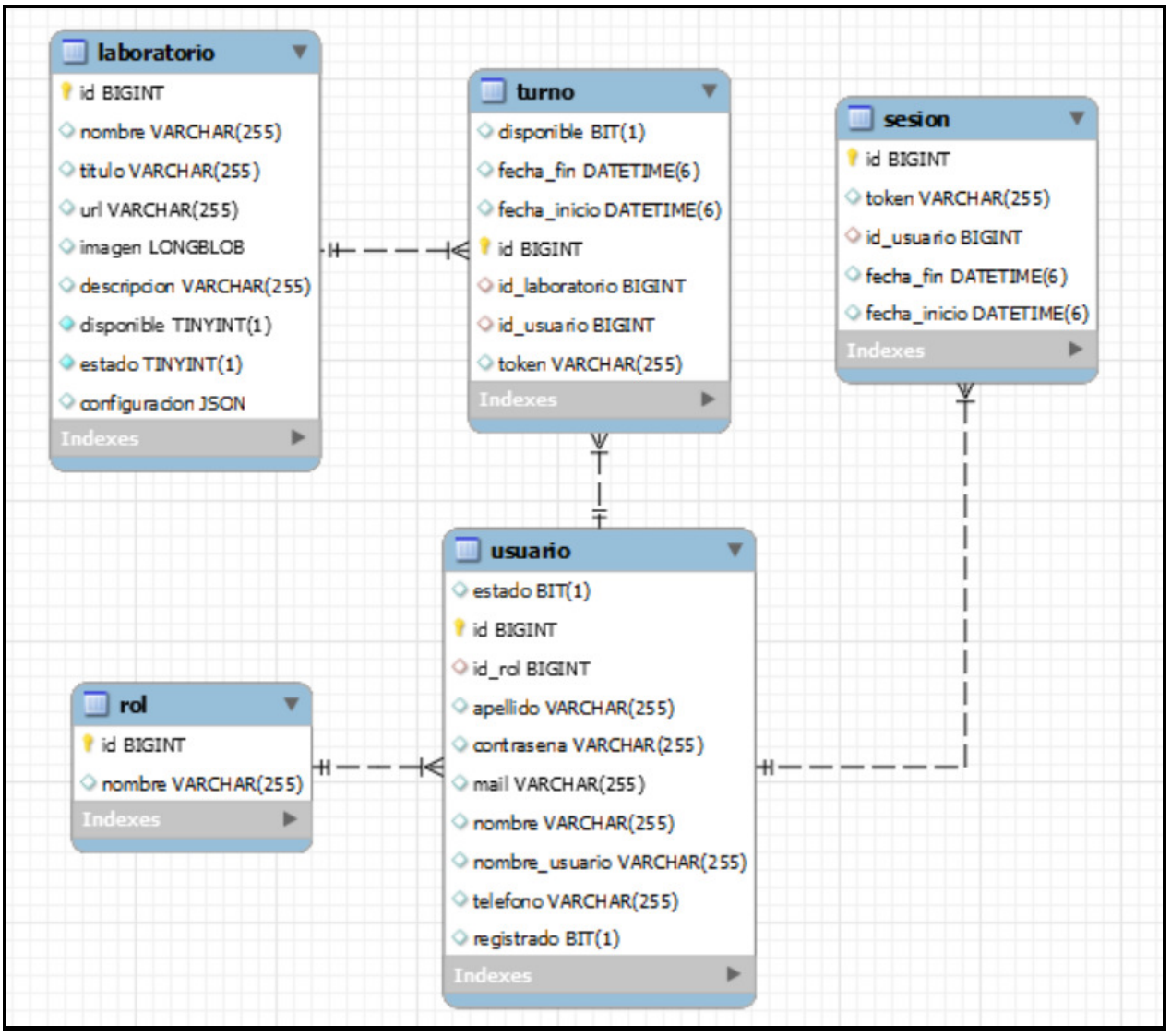
5.5.1 Diagrama de Entidad Relación

En el proceso de diseño de bases de datos, el Diagrama de Entidad-Relación (ER) es una herramienta gráfica que permite modelar la estructura lógica y las interrelaciones de los datos.

El Diagrama de Entidad-Relación se centra en identificar y describir las entidades significativas dentro de un sistema, así como las conexiones y vínculos que existen entre ellas. Las entidades, se representan mediante rectángulos y se caracterizan por sus atributos distintivos.

Estos diagramas proporcionan una vista esquemática de la estructura lógica de la base de datos, revelando cómo los datos se relacionan y cómo fluye la información entre las distintas entidades. A través de símbolos y líneas conectivas, se ilustran las relaciones, que pueden ser de diferentes tipos, como uno a uno, uno a muchos o muchos a muchos.

Figura 2. DER base de datos



Fuente: Elaboración propia.

5.5.2 Tablas de la base de datos

Laboratorio

La tabla laboratorio es donde se almacenan los laboratorios creados. Esta tabla cuenta con un campo id, que es su clave primaria, también tiene los campos nombre, título, descripción, disponible y estado. El campo “url” es donde se va a almacenar la URL para acceder al laboratorio. El campo imagen es para almacenar una imagen en formato jpg o jpeg que va a llevar el laboratorio. Por último, el campo configuración es de tipo JSON donde tiene la configuración del laboratorio respecto a la duración de cada turno y los días y los horarios disponibles para sacar turnos en cada laboratorio.

Usuario

La tabla Usuario contiene todos los usuarios registrados en la aplicación. Esta tabla contiene los datos personales del usuario, su mail, su rol, el nombre de usuario y la contraseña. Además, cuenta con el campo registrado, que lo tiene que activar el usuario al ingresar al link enviado por mail para completar el registro. También cuenta con un campo id como clave primaria de esta tabla.

Turno

La tabla Turno contiene los turnos junto con su detalle, como por ejemplo el laboratorio, el usuario, la fecha, hora de inicio y fin del turno, el token del turno y un id que reconoce al turno como clave primaria.

Rol

La tabla Rol tiene los roles establecidos en la aplicación. Los campos que tiene son id y nombre.

Sesion

La tabla Sesion fue la última en crearse. El motivo de esta tabla es para almacenar el token de sesión en las conexiones realizadas por los usuarios al loguearse en la aplicación web. Los campos que tiene esta tabla son: id, token, usuario, fecha/hora inicial y final.

5.5.3 JPA

Para manejar la interacción entre la lógica de la aplicación y la capa de persistencia de datos, se implementó la tecnología Java Persistence API (JPA). JPA es una especificación estándar de Java para el mapeo objeto-relacional (ORM). Su principal función es abstraer la complejidad de la interacción entre objetos de Java y una base de datos relacional. Al utilizar JPA, se logra simplificar las consultas a la base de datos, el mapeo de entidades y la gestión de la persistencia.

5.6 Módulos de la aplicación web

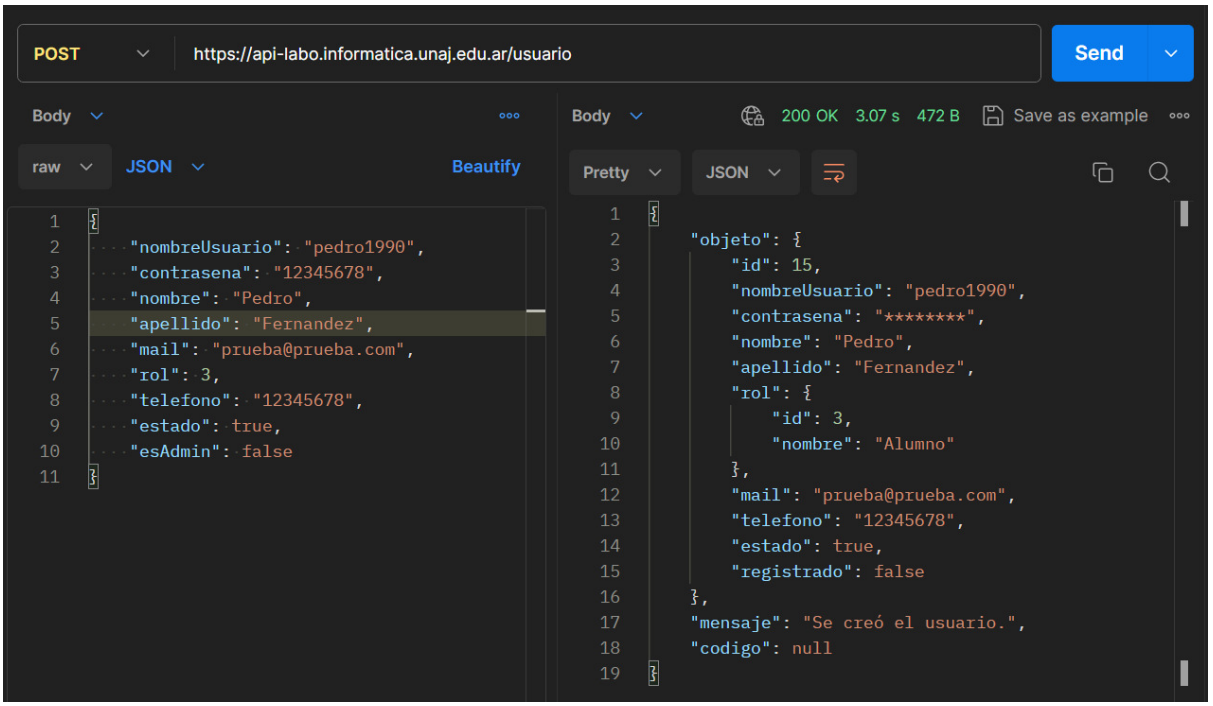
En este apartado, se describirán las funcionalidades, cómo están constituidos cada módulo de la aplicación, junto a una descripción de sus servicios y cómo interactúan entre sí.

5.6.1 Módulo Usuarios

En el módulo de usuarios se encuentran los servicios relacionados a los usuarios.

El servicio para registrarse está diseñado para que un alumno ingrese sus datos personales, datos de usuario de la aplicación y luego se le envía un mail al correo indicado, para terminar su registración. Hasta que no confirme su identidad vía mail, el usuario no va a poder acceder a la plataforma web con sus credenciales.

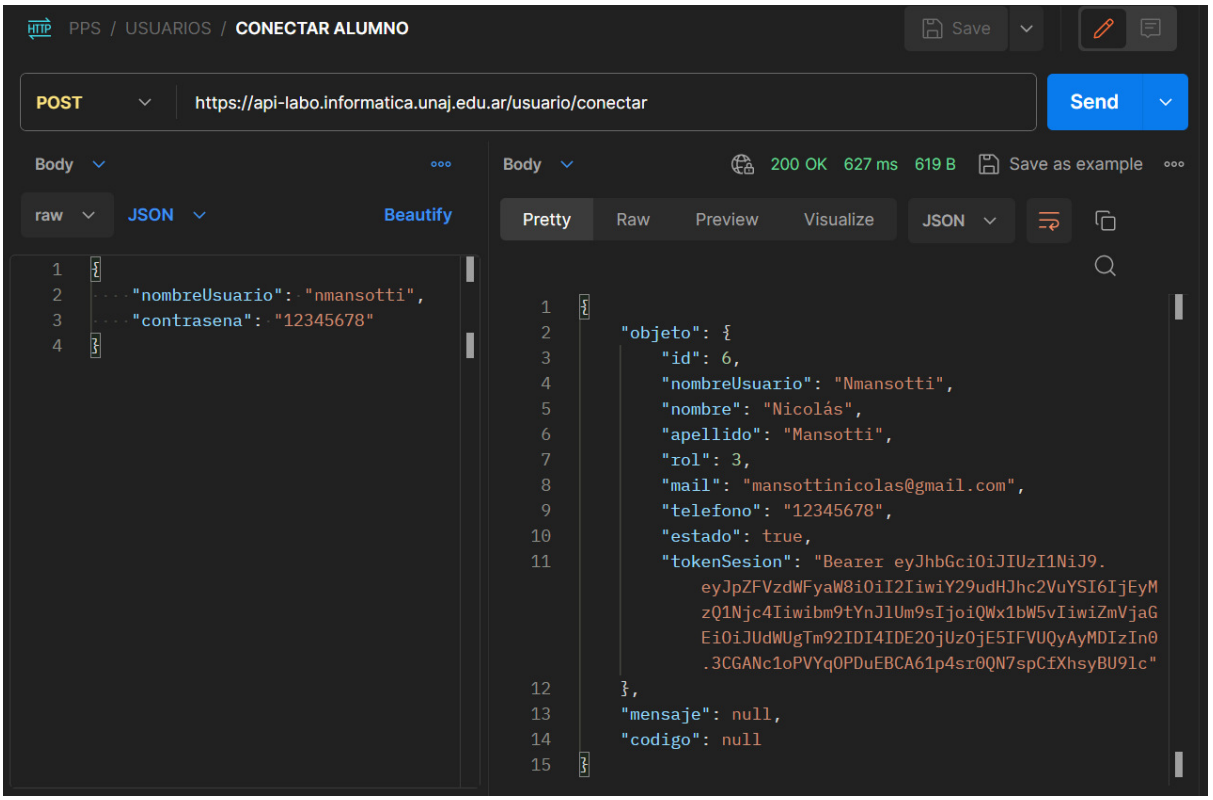
Figura 3. Registro de usuario.



Además, existe el servicio para restablecer contraseña, donde el usuario informará su mail al que quiere que se le envíe un correo con un link, y posteriormente cambiar su contraseña.

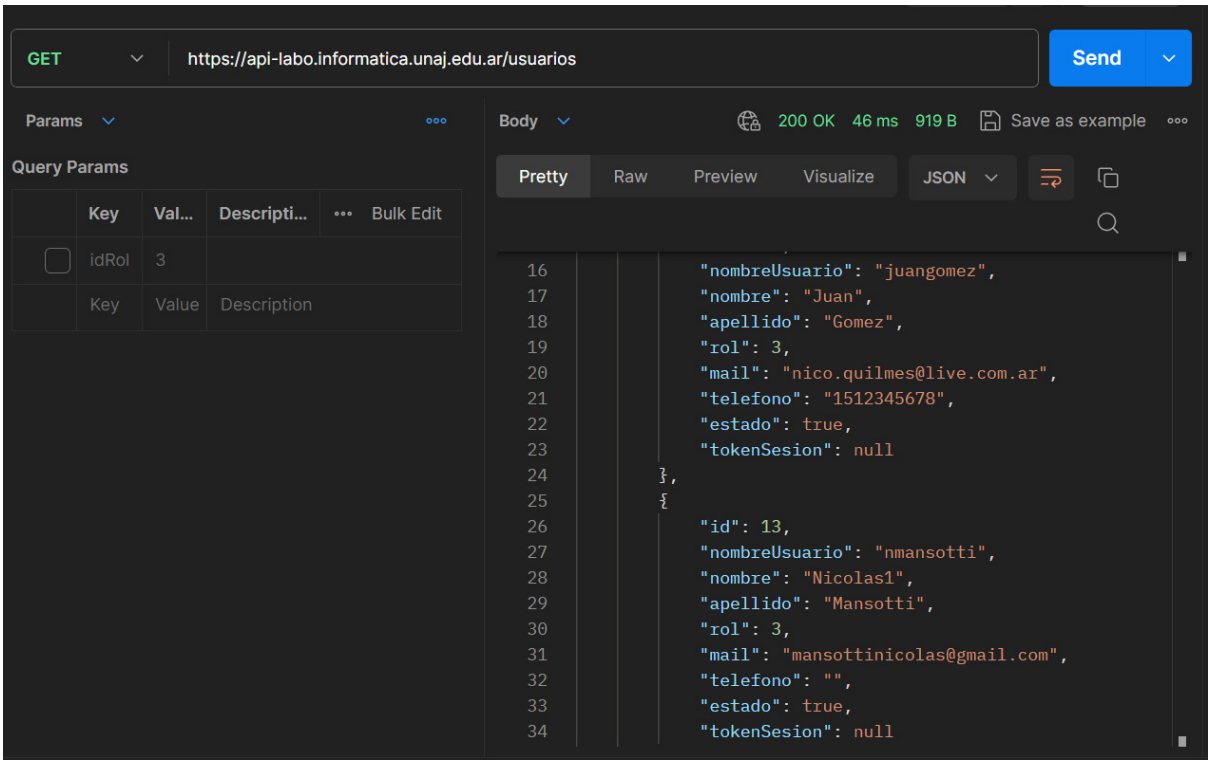
El servicio “conectar” se utiliza para loguearse a la aplicación tiene la posibilidad de loguearse ya sea con el nombre de usuario o mail de registro. Además, deberá ingresar la contraseña. Este servicio, devuelve el token de sesión que luego se usará para consumir los demás servicios de la aplicación web.

Figura 4. Conectar usuario.



Además, hay un servicio para obtener los usuarios registrados, y otro servicio para activar a un usuario que se esté registrando en la aplicación web, mediante el enlace que se le envía al mail.

Figura 5. Obtener usuarios.



Otro servicio del módulo de usuarios es el de modificar datos personales. En este servicio, el administrador tiene la posibilidad de cambiar algunos de los datos personales de un alumno, como el nombre, el apellido, el teléfono y el estado del usuario. El campo “estado” es un booleano. Siempre que el estado de un usuario sea *true*, el alumno podrá utilizar el servicio “conectar” sin problemas. Cuando el estado del alumno sea *false*, el alumno no podrá loguearse en la aplicación.

5.6.2 Módulo Turnos

El módulo de turnos es el que más complejidad y lógica tiene. Esto se debe a que es el *core*(el elemento central y fundamental) del sistema de gestión de turnos y usuarios. Desde

la aplicación web, un alumno puede loguearse, ver los laboratorios y turnos disponibles y sacar un solo turno por laboratorio, mientras no tenga ningún turno activo. Luego en cada laboratorio hay que hacer una validación para chequear que el usuario que está intentando usarlo sea quien reservó el turno.

Los servicios de este módulo son:

- obtenerTurnosDisponibles
- obtenerTurnos
- obtenerTurnoLaboratorio
- obtenerTurnoToken
- crearTurno
- eliminarTurno
- obtenerHorariosDisponibles

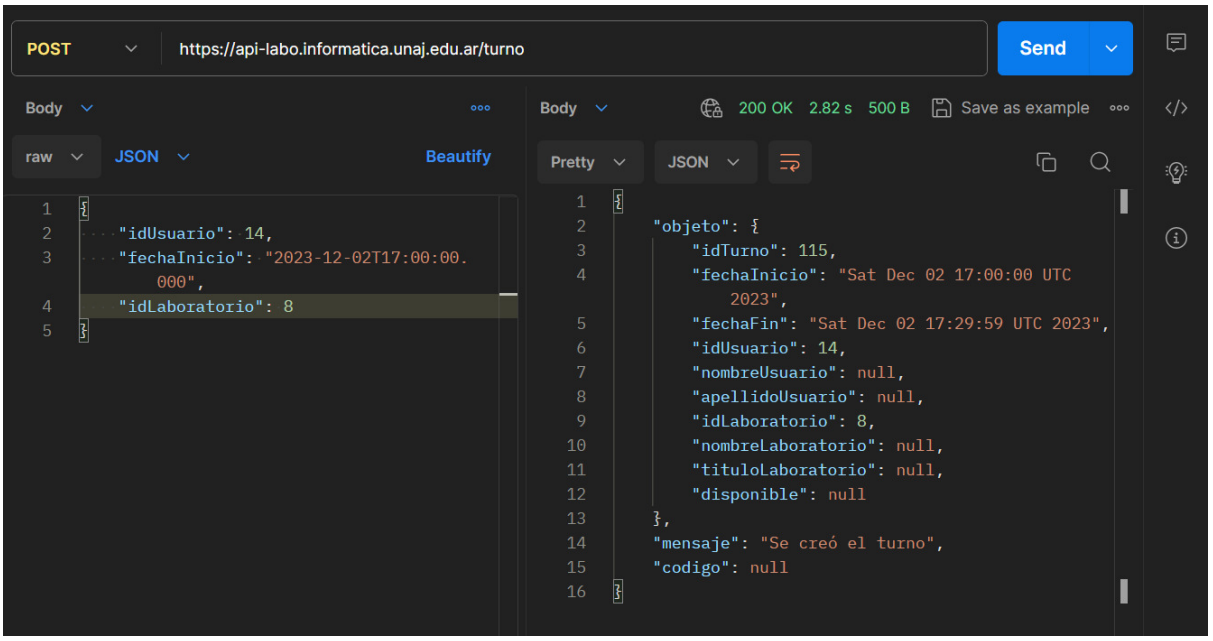
El servicio crearTurno es el utilizado para que un alumno pueda reservar un turno en un laboratorio. En este servicio, se hacen una serie de validaciones, en donde se van a tener que cumplir todas para poder sacar un turno. Estas validaciones son:

- Elegir la fecha y hora posterior a la fecha y hora actual. Es decir, no se puede sacar un turno para una fecha y/o hora que ya pasó, porque no tendría sentido.
- El laboratorio elegido para sacar turno tiene que existir y estar disponible.
- El idUsuario que saque el turno, tiene que existir.
- El horario elegido para reservar el turno tiene que estar disponible.
- No tener un turno reservado activo.

Si todas esas condiciones se cumplen, el servicio crearTurno guardará en la base de datos toda la información referida al turno, reservará un turno para el alumno y se enviará un

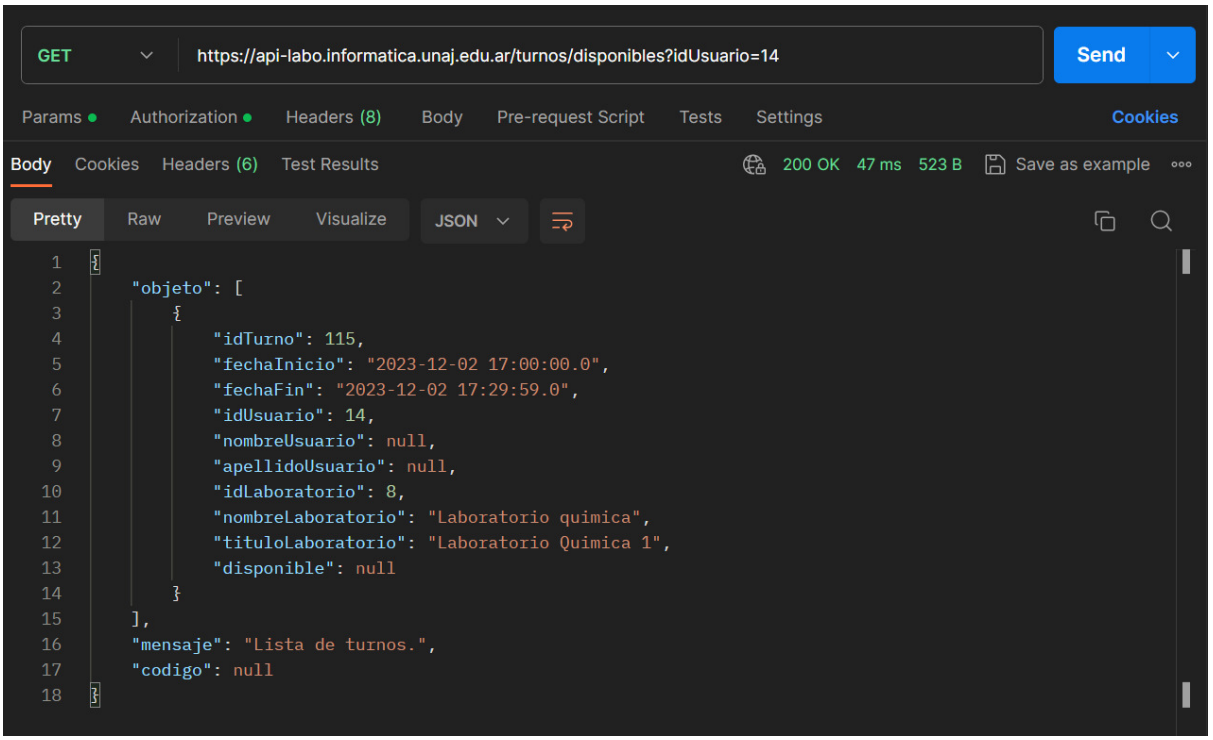
email al correo electrónico con el que el usuario se registró, para dar un aviso recordatorio del turno.

Figura 6. Crear turno.



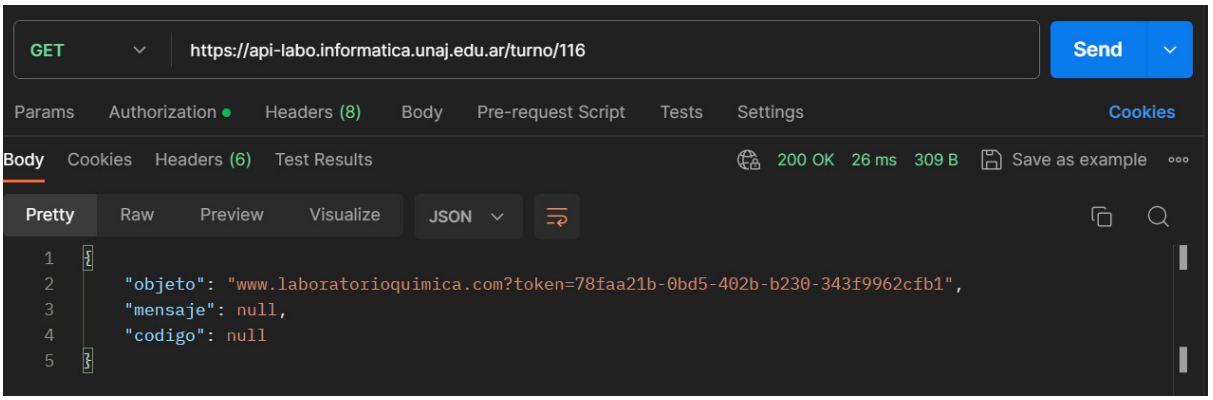
El servicio obtenerTurnosDisponibles retorna los turnos activos que un usuario reservó. En cada laboratorio que el usuario haya reservado un turno y esté activo, lo va a devolver en una lista. Este servicio es para que el alumno pueda ver las reservas que ha hecho y la información de cada reserva, como el laboratorio y la fecha y hora elegida.

Figura 7. Obtener turnos disponibles



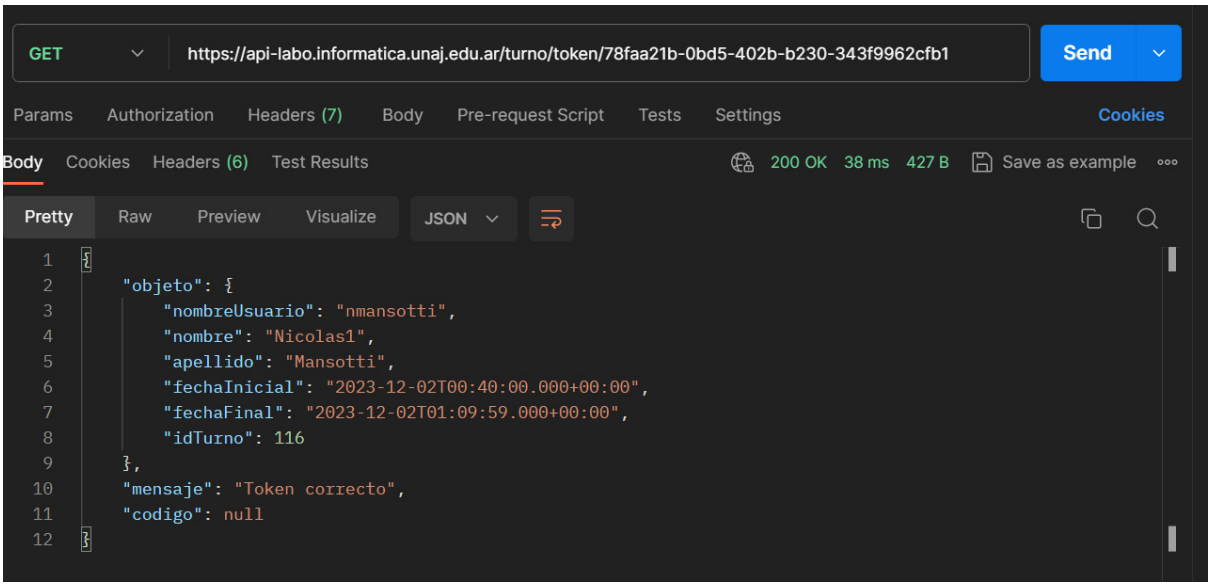
El servicio obtenerTurnoLaboratorio se utiliza al momento de la fecha y hora en la que el alumno reservó el laboratorio. La respuesta de este servicio es la URL del laboratorio, junto con el token que luego se verifica en la página del laboratorio remoto. Esta URL del laboratorio remoto es el acceso que tiene el alumno para utilizar el laboratorio y solo está disponible dentro de la fecha y hora que se realizó la reserva.

Figura 8. Obtener turno laboratorio



Junto a la información y los campos guardados en la tabla Turno, también se genera automáticamente un token, que luego el usuario al momento de ingresar al laboratorio que reservó el turno, se le pide este token, para validar que realmente sea él quien hizo la reserva. Este chequeo de token se hace mediante el servicio obtenerTurnoToken en cada laboratorio. Los laboratorios tendrán que consumir la API y consumir el servicio obtenerTurnoToken, pasándole este token de esta manera:

Figura 9. Obtener turno token.



El servicio obtenerTurnoToken va a responder un *status code* 200 si el token es correcto, y va a retornar un *status code* 400, junto con un mensaje de error si el token es incorrecto. Luego cada laboratorio tiene que manejar internamente como interpreta la respuesta de este servicio, otorgando acceso o no al alumno para utilizar el laboratorio. Dentro de los errores, este token puede ser correcto pero pudo haber expirado el tiempo para utilizar el laboratorio, o también el token al no ser correcto, el servicio va a devolver un mensaje de error que el token es incorrecto.

El servicio obtenerHorariosDisponibles retorna los horarios disponibles de un laboratorio en una determinada fecha. Al servicio hay que pasarle por parámetro el id del laboratorio y la fecha en la que se quiere consultar los horarios disponibles para sacar un turno. Si un horario ya está tomado por otro usuario, se muestra el campo “disponible” en *false*.

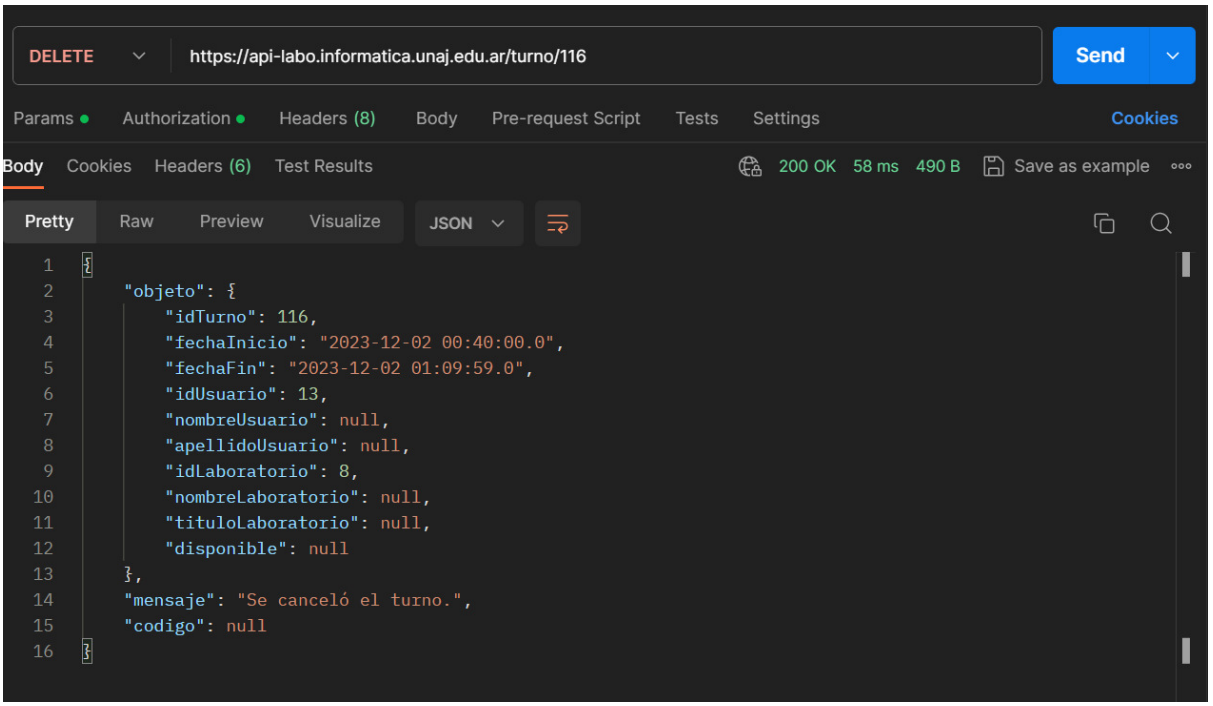
El servicio obtenerTurnos es para que lo utilice el administrador. Este servicio retorna los turnos reservados por todos los usuarios. Este servicio le da un monitoreo y un control de las reservas al administrador. Tiene filtros para poder obtener un turno según el criterio que se desee. Estos filtros son parámetros del *request* del servicio, que pueden determinar la fecha y hora de inicio y de fin de las reservas, el laboratorio, el usuario y si un turno está disponible o no.

La disponibilidad de un turno es un campo que decide si el turno está activo o no.

Para cancelar un turno existe el servicio cancelarTurno, en donde se le pasa el id del turno y si este turno está activo, se cancela, modificando el campo “disponible” y poniéndolo en *false*.

El servicio de eliminarTurno es para cuando el alumno quiere cancelar un turno que sacó por cualquier motivo. Al realizar esta acción, se enviará un email al correo electrónico con el que el usuario se registró, para dar un aviso que se canceló el turno.

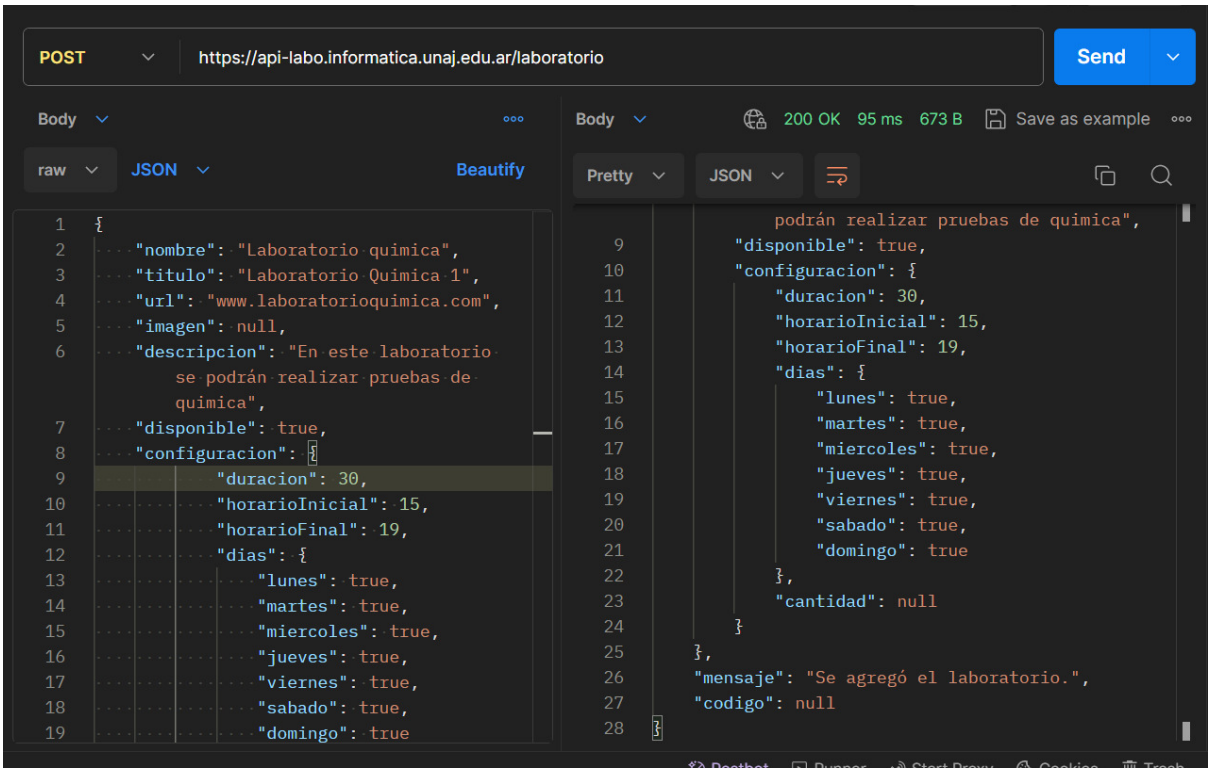
Figura 10. Eliminar turno.



5.6.3 Módulo Laboratorios

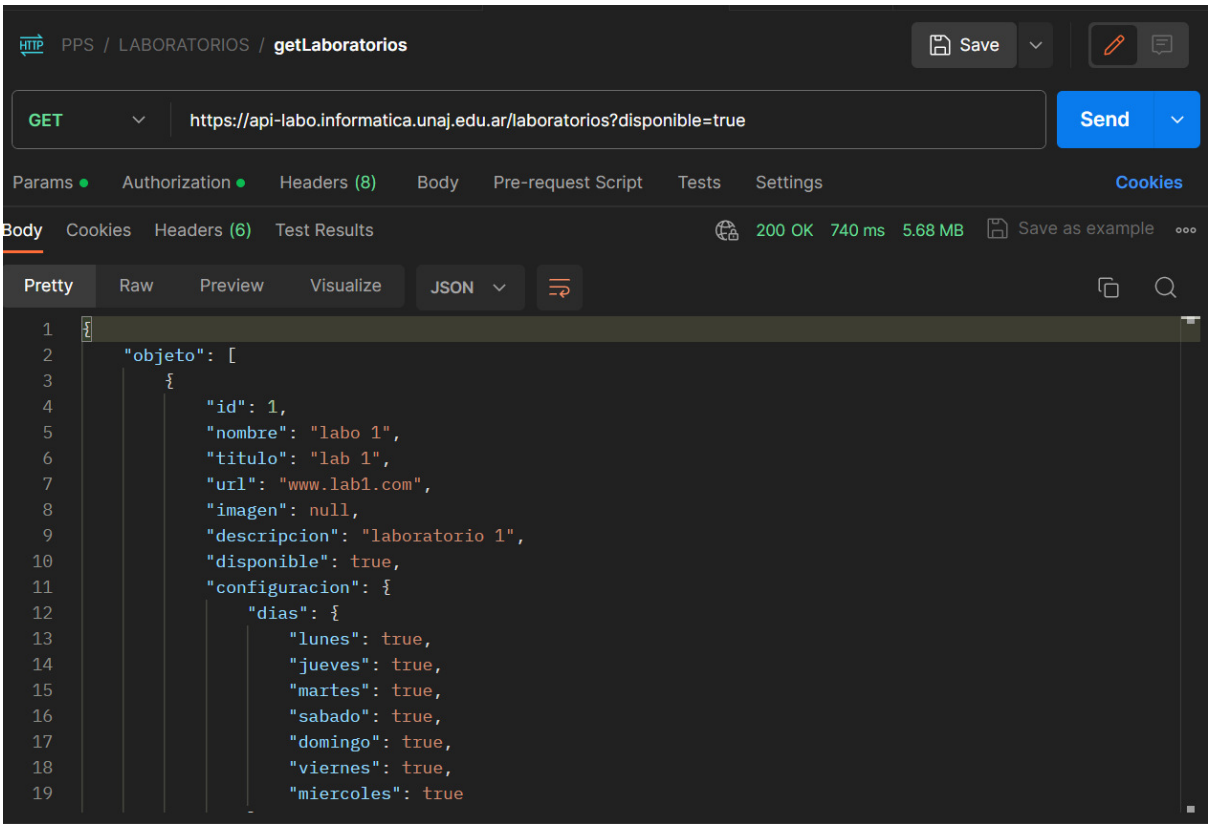
El módulo de laboratorios tiene cuatro servicios: crearLaboratorio, actualizarLaboratorio, eliminarLaboratorio y obtenerLaboratorios. Todos estos servicios son solo para que los pueda utilizar el administrador, a excepción de obtenerLaboratorios, que además lo puede usar el rol de alumno.

Figura 11. Crear laboratorio



El servicio obtenerLaboratorios devuelve todos los laboratorios que existen en la base de datos. También se le puede pasar un parámetro en el *request*, que determina si el laboratorio está disponible o no. El administrador tiene la posibilidad de deshabilitar laboratorios por cuestiones de mantenimiento o para que no se muestren más en la lista de laboratorios a reservar turnos. Por ende, el parámetro “disponible ”para cuando lo use un alumno, siempre va a ser *true*. Cuando el que utilice el servicio de obtenerLaboratorios sea un administrador, el parámetro “disponible” siempre va a ser *false*, ya que puede ver todos los laboratorios, estén o no disponibles.

Figura 12. Obtener laboratorios.



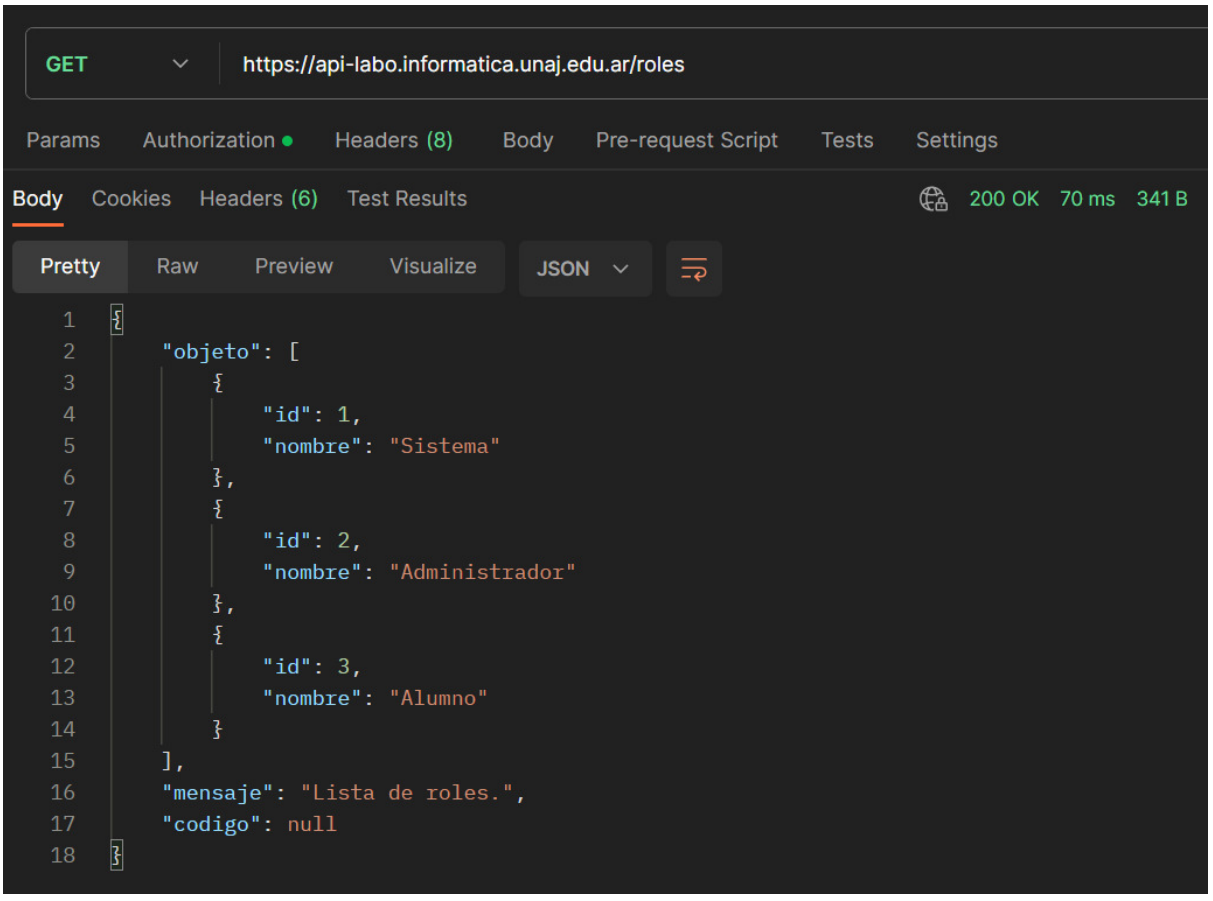
Los otros servicios, como su nombre lo indican, son para crear, para actualizar y para eliminar un laboratorio, pasándole por parámetro todos los campos necesarios para guardarlos en la base de datos y realizar las acciones necesarias.

5.6.4 Módulo Roles

En el módulo de roles hay tres servicios: crearRol, eliminarRol y obtenerRoles. Estos tres servicios son para gestionar los roles de la aplicación. Para crear un nuevo rol, para eliminarlo, y para ver los roles que existen. Mediante un rol, un usuario de la aplicación tiene determinados permisos. En la aplicación web, se dejó preparado un rol de Administrador,

que lo van a usar los profesores y/o personas que van a gestionar y administrar la página web. El otro rol que existe en la aplicación web es el de alumnos. Este rol se va a otorgar a cualquier estudiante que cree un usuario y los permisos y acciones son más limitados que los usuarios que tienen rol de Administrador.

Figura 13. Obtener roles.



5.7 Otras funcionalidades de la aplicación

En este apartado, se detallan otros aspectos y componentes de la aplicación, que en conjunto con los servicios y módulos, conforman el resultado final de la API.

5.7.1 Envío de mail

Se decidió que el *backend* resuelva el envío de mails a los usuarios.

Estos emails se envían desde el servidor al cliente para confirmar la identidad de un usuario al registrarse, para restablecer contraseña, y también al momento de reservar un turno, para recordarle al usuario en qué fecha y hora sacó un turno para determinado laboratorio.

Para enviar los mails se utilizó la librería de java “javax.mail” en donde ya cuenta con funciones para realizar envíos de mails. Se configuró el puerto desde donde se va a enviar el mail, la dirección de correo emisora y su contraseña, el método de autenticación, y el asunto y cuerpo del mail que se va a enviar.

El asunto y cuerpo del mail a enviar se configura en cada servicio que envía mail, de forma particular, según las necesidades del servicio.

Figura 14. Email confirmar cuenta.

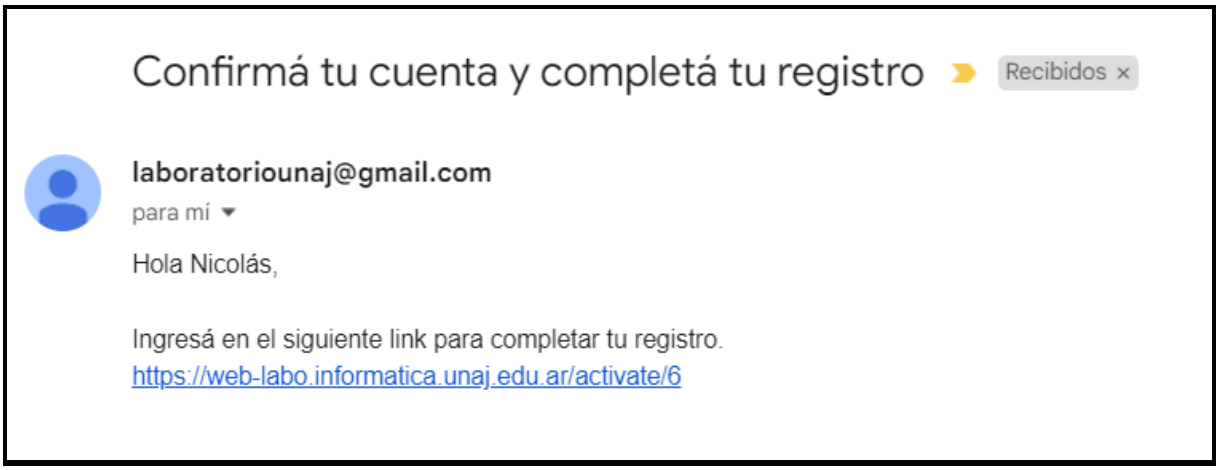


Figura 15. Email recordatorio turno.

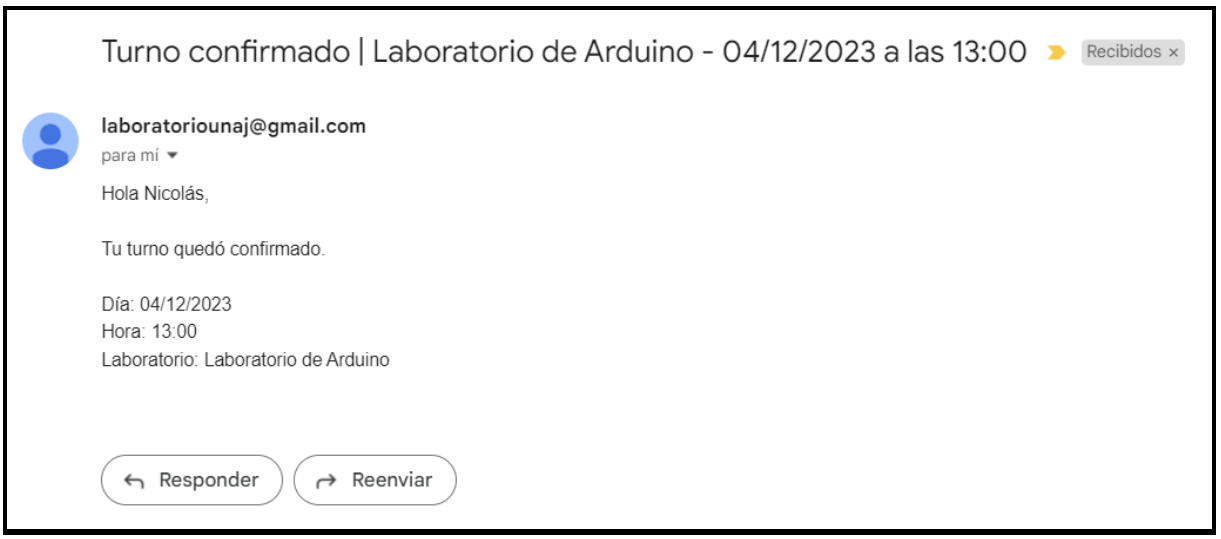


Figura 16. Email cancelar turno.

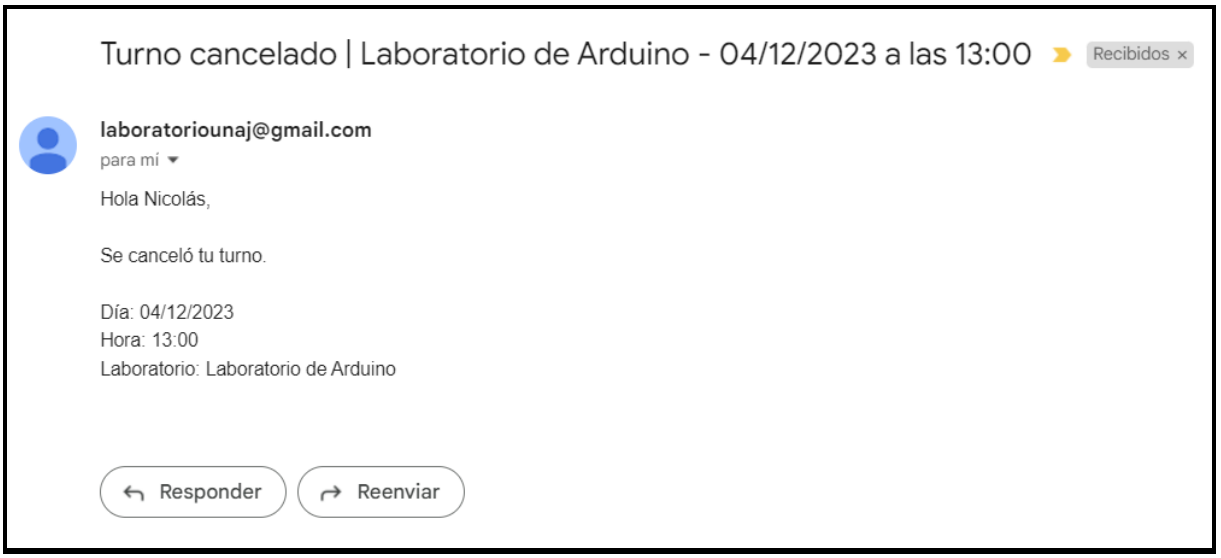


Figura 17. Email restablecer contraseña.

5.7.2 Roles y permisos de rol

Los roles definidos para la aplicación son de Alumno y Administrador, también se agregó un rol de “Sistema” como soporte.

Estos roles tienen permisos distintos. El rol de alumno se limita a ver los laboratorios disponibles, ver los días y horarios disponibles de cada laboratorio para sacar un turno, poder reservar un turno, ver los turnos que tiene reservados, y cancelar un turno.

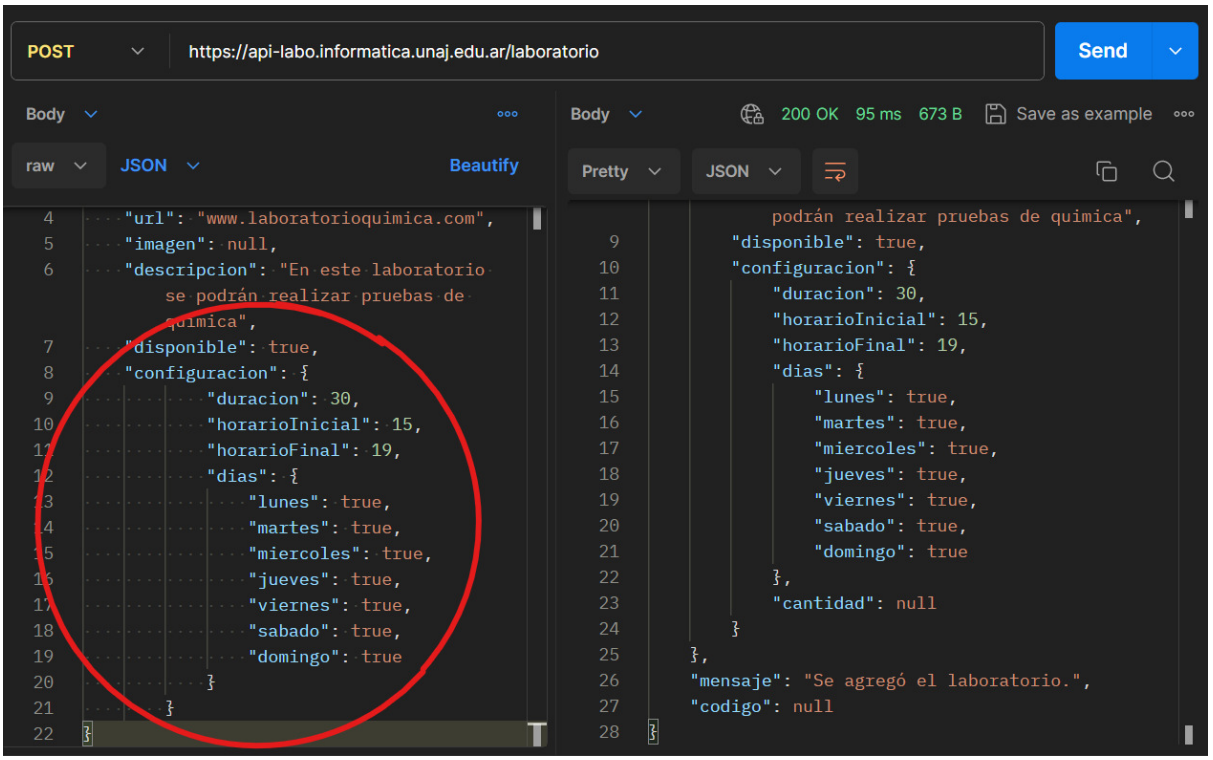
El rol de Administrador tiene muchos más permisos. Puede crear y editar laboratorios, puede ver los usuarios del sistema, y los turnos reservados por los alumnos. El usuario que tenga el rol Administrador es el encargado de administrar la aplicación web.

5.7.3 Configuración de laboratorios

La configuración de laboratorios no fue algo que se planteó desde el principio del proyecto. Si bien se mencionó que un usuario administrador pueda configurarlos, no se planteó ese manejo desde el inicio. Casi al final del desarrollo se agregó la parte de la configuración de laboratorios. Esta configuración es única de cada laboratorio, y va a poder definir qué duración van a tener los turnos de cada laboratorio, entre que horarios se pueden sacar turnos y también los días en los que un alumno pueda reservar un turno.

Al mismo tiempo, se acordó que un usuario alumno solamente pueda tener un turno activo por laboratorio. Esto no es configurable y es una variable fija. Esta configuración se guarda en la tabla de laboratorios, en la columna “configuración”, de tipo JSON, y allí está la información en formato JSON, de los campos ya mencionados.

Figura 18. Configuración de laboratorios.



5.7.4 Manejo de errores

Para el manejo de errores de los servicios se acordó que cada módulo tenga sus códigos de errores y mensajes particulares de cada error definidos.

También según si la respuesta es exitosa o no, se devuelve un *status code*, usando un 200 OK cuando la respuesta es exitosa y un 400 cuando la respuesta a la solicitud no es exitosa y se muestra un código de error en pantalla.

La estructura de la respuesta de los servicios es la siguiente:

{

"objeto": null,

"mensaje": "No se encontró el usuario.",

"codigo": "TURNOS-ERR-005"

}

Estos son los códigos de errores manejados:

USUARIOS-ERR-00X

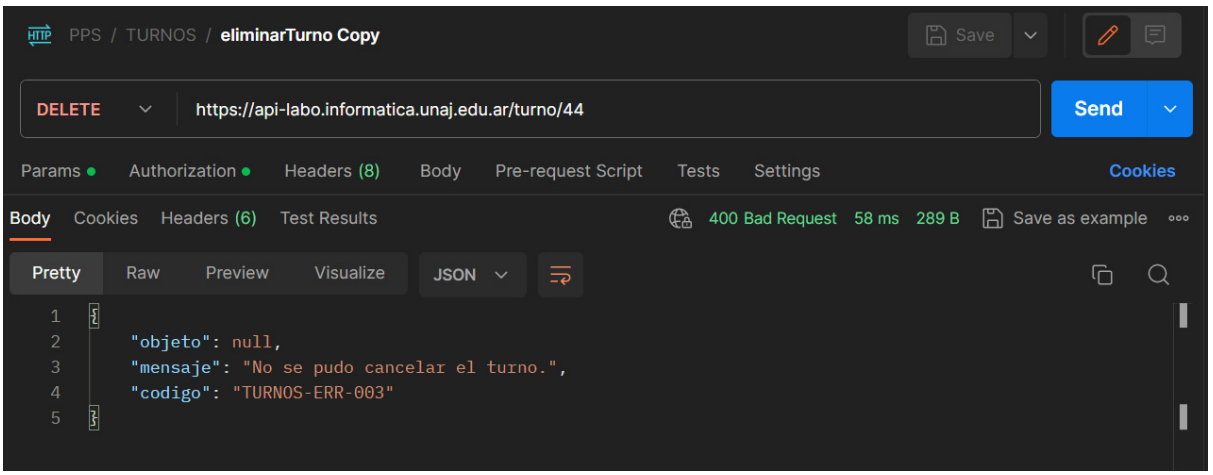
TURNOS-ERR-00X

ROLES-ERR-00X

USUARIOS-ERR-00X

SESION-ERR-00X

Figura 19. Respuesta con error



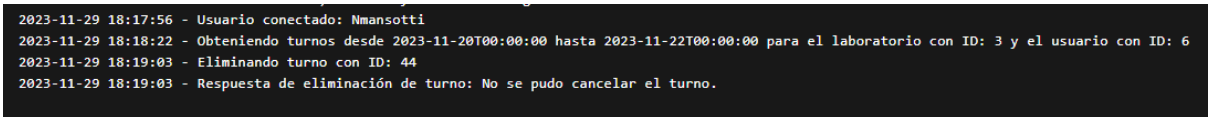
Las validaciones y errores de la aplicación web son manejados en los servicios de *backend* que el *frontend* consume como API.

5.7.5 Agregado de logs

Finalizando los servicios, se agregaron logs a los mismos. Esto se hizo para tener una monitorización de los movimientos de la aplicación y un seguimiento y diagnóstico de un posible problema. Además, aporta auditoría, seguridad y garantiza un historial de los servicios consumidos.

Estos logs brindan información de los servicios utilizados, como también que usuario lo estaba utilizando, el resultado del servicio, y la fecha y hora de cuando se consumió el servicio.

Figura 20. Log de la API.



5.7.6 Seguridad de la API

La seguridad de la API desempeña un papel fundamental al resguardar tanto la aplicación web, como la información que fluye a través de ella. Al implementar medidas de seguridad robustas, se establecen protecciones en torno a los datos sensibles, previniendo posibles riesgos y ataques.

No solo fortalece las defensas contra amenazas externas o pérdida de datos, sino que a su vez influye en la confianza que depositan los usuarios en la plataforma.

En este apartado se detallan las medidas de seguridad que se tomaron para securizar la API.

5.7.7 Token de sesión.

Cuando el desarrollo de la API estaba avanzado, se tomó la decisión de utilizar un token de sesión para otorgarle más seguridad a los servicios. Este token de sesión es un JWT.

Además de manejar los token de sesión, una vez que los usuarios se loguean, también los servicios cuentan con permisos de rol. Es decir, un usuario con rol de Alumno no va a poder utilizar un servicio que solo lo pueda usar un usuario con rol de Administrador.

En los servicios de la API, se tiene que enviar este token en el header, y el mecanismo de envío elegido para enviar el token es el de Bearer Token. Además, el token de sesión tiene un límite de tiempo para ser usado. Si el token generado, ya superó el tiempo permitido para usarse, el usuario tendrá que conectarse nuevamente a la aplicación para generar un nuevo token.

Figura 21. Token de sesión en respuesta de servicio Conectar

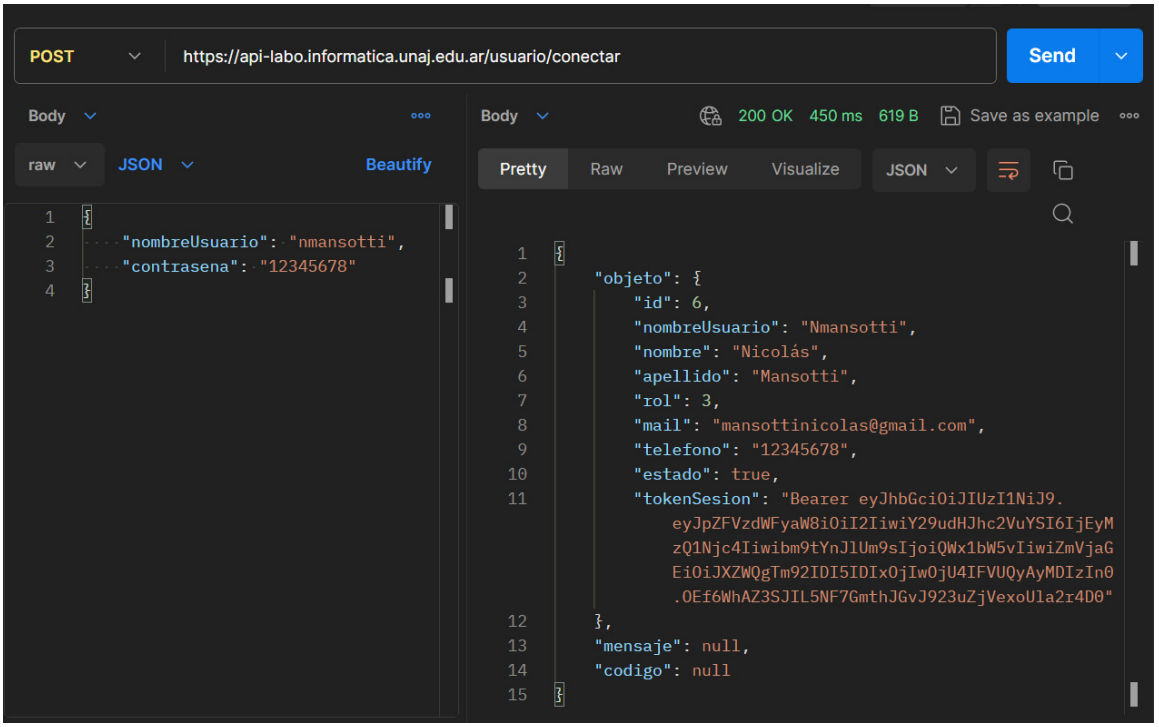
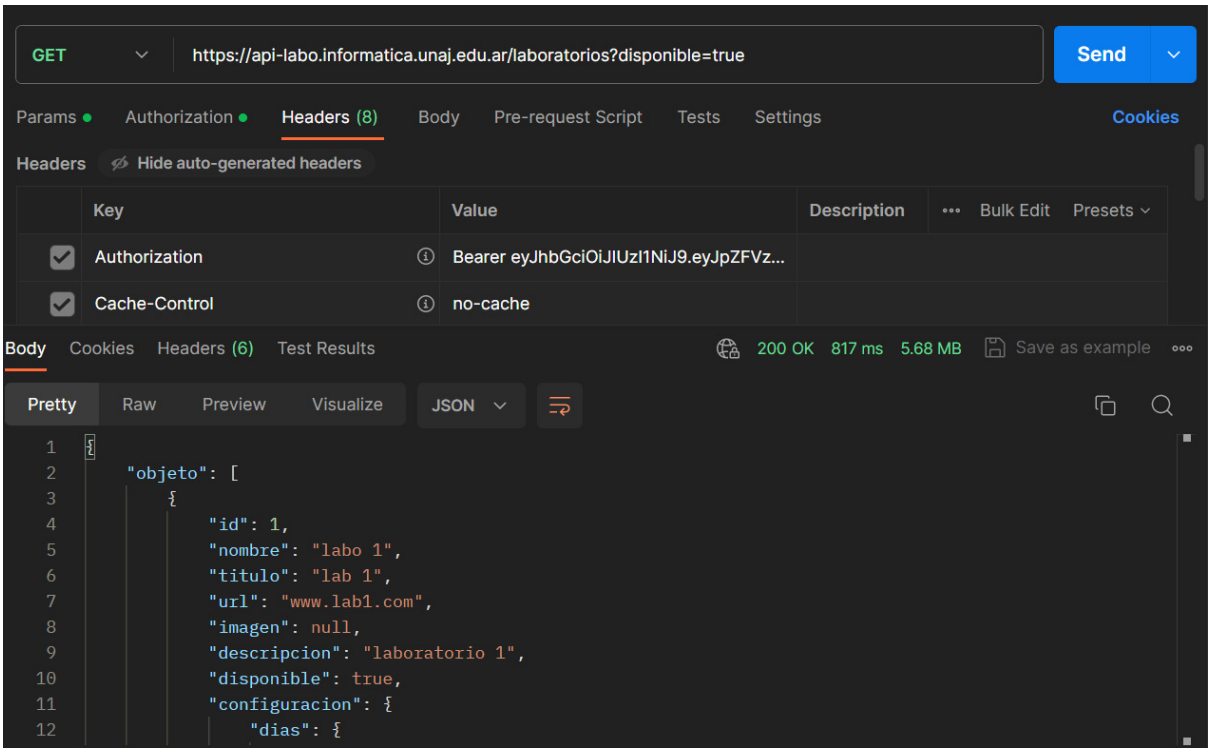


Figura 22. Utilizando el token de sesión en otro servicio



5.7.8 Protocolo HTTPS

HTTPS(Hypertext Transfer Protocol Secure) es un protocolo de comunicación que proporciona una capa adicional de seguridad para la transferencia de datos en internet. Esta capa de seguridad se logra mediante el uso de un cifrado SSL/TLS, lo que permite que la información transmitida entre el navegador del usuario y el servidor web se mantenga encriptada y protegida contra posibles ataques o manipulaciones.

Este protocolo asegura la integridad y confidencialidad de los datos durante su transmisión, evitando que terceros intercepten o manipulen la información sensible que viaja a través de la red. Al implementar HTTPS, se verifica la autenticidad del servidor al que se accede,

garantizando que la conexión sea legítima y protegiendo la privacidad de los usuarios al cifrar los datos intercambiados entre el navegador y el servidor.

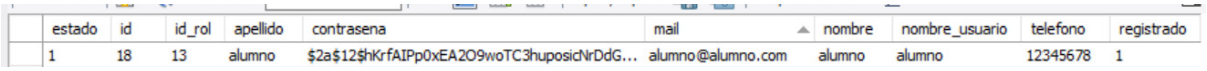
Desde la configuración del proyecto, se implementó el protocolo HTTPS para garantizar la seguridad en la comunicación de la aplicación.

5.7.9 Securización de las contraseñas

La seguridad de las contraseñas se trataron por la librería BCrypt donde se cifran al momento de guardarse en la base de datos y se descifran al realizar una consulta.

BCrypt es un algoritmo de *hashing* diseñado específicamente para el almacenamiento seguro de contraseñas. Está diseñado para ser lento y resistente a ataques de fuerza bruta, haciendo que sea más difícil para los atacantes descifrar contraseñas. De esta manera, se evita que las contraseñas estén en texto plano y puedan ser vulneradas.

Figura 23. Contraseña encriptada en tabla de Usuario



estado	id	id_rol	apellido	contrasena	mail	nombre	nombre_usuario	telefono	registrado
1	18	13	alumno	$2a$12$hKrfAIPp0xEA2O9woTC3huposicNrDdG...	alumno@alumno.com	alumno	alumno	12345678	1

5.7.10 Configuración de variables de la base de datos

En las reuniones con Florencia Ayala, se optó por asegurar la información sensible de la base de datos adoptando un enfoque de seguridad recomendado. Esto implica que los datos confidenciales, como contraseñas o credenciales de acceso, se extraen y almacenan como variables de entorno en el archivo de configuración de Docker Compose. Estas

variables de entorno se vinculan y se utilizan en el archivo application.properties de Spring Boot. Esta práctica es una medida de seguridad estándar, ya que separa los datos confidenciales del código fuente, lo que reduce los riesgos de exposición accidental y garantiza una gestión más segura de la información sensible de la base de datos.

Estas variables se definieron como:

spring.datasource.url=${SPRING_DATASOURCE_URL}

spring.datasource.username=${SPRING_DATASOURCE_USERNAME}

spring.datasource.password=${SPRING_DATASOURCE_PASSWORD}

En donde la información de la URL de la base de datos, el usuario y la contraseña se obtienen del archivo Docker Compose.

Figura 24. Configuración de application.properties.

application.properties

```
server.port=80
spring.datasource.url=${SPRING_DATASOURCE_URL}
spring.datasource.username=${SPRING_DATASOURCE_USERNAME}
spring.datasource.password=${SPRING_DATASOURCE_PASSWORD}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
```

6. Inconvenientes.

6.1 Configuración de CORS

Al empezar a integrar los servicios de *backend* con *frontend*, hubo errores de CORS.

El CORS es el intercambio de recursos de origen cruzado, es decir, es un mecanismo que permite que se puedan solicitar recursos restringidos en una página web desde un dominio diferente del dominio que sirvió el primer recurso.

El error se solucionó agregando clases de configuración para el cors.

El error era el siguiente:

Access to XMLHttpRequest at 'http://localhost:8080/laboratorios' from origin 'http://localhost:4200' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.

Agregando configuración para tratarlo, se le pasó la URL desde la aplicación web que se van a consumir los servicios desarrollados en el *backend*. Además, se le configuró los métodos HTTP con los que se podrá consumir la API.

Figura 25. Configuración de CORS.

Además de la aplicación web, también los laboratorios consumen la API para identificar que el usuario que ingresa al laboratorio sea el que reservó el turno. Internamente, se obtiene de la base de datos las URLs de todos los laboratorios disponibles y que cada laboratorio pueda mapear el dominio de la API.

6.2 Aprendizaje de Docker

La utilización de Docker en el proyecto fue algo que se planteó de entrada en la presentación del proyecto y cómo se iba a llevar a cabo. Esto implicó el estudio de la herramienta, realizar pruebas iniciales, interactuar con todos los elementos de docker, como docker-compose y su configuración para levantar los servicios. La configuración del Dockerfile también es necesaria para dockerizar la API, y los comandos para crear la imagen docker.

A continuación se muestran imágenes de cada módulo de Docker y cómo realizar el paso a paso para utilizarlo.

Figura 26. Archivo Dockerfile.

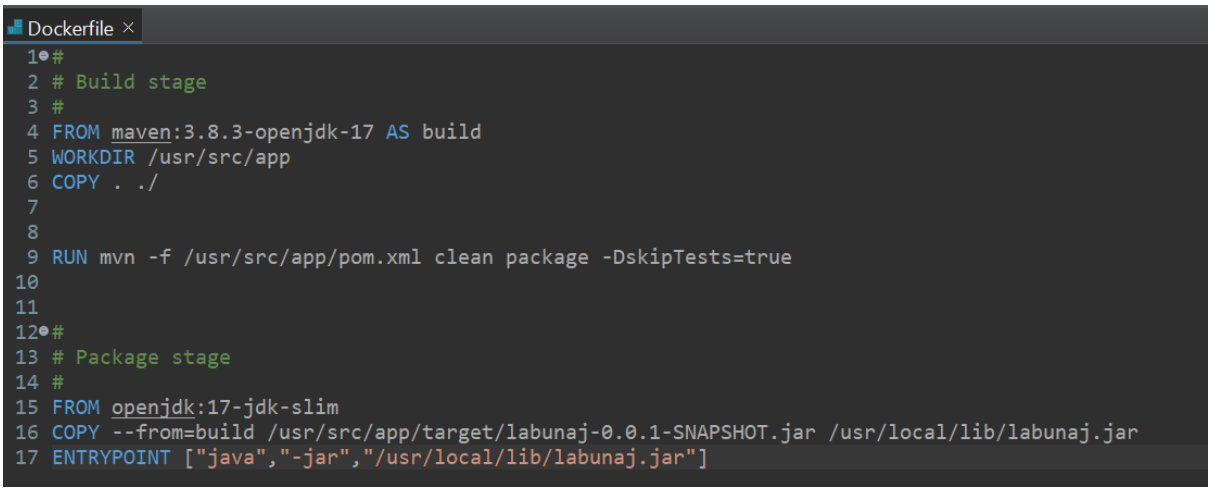
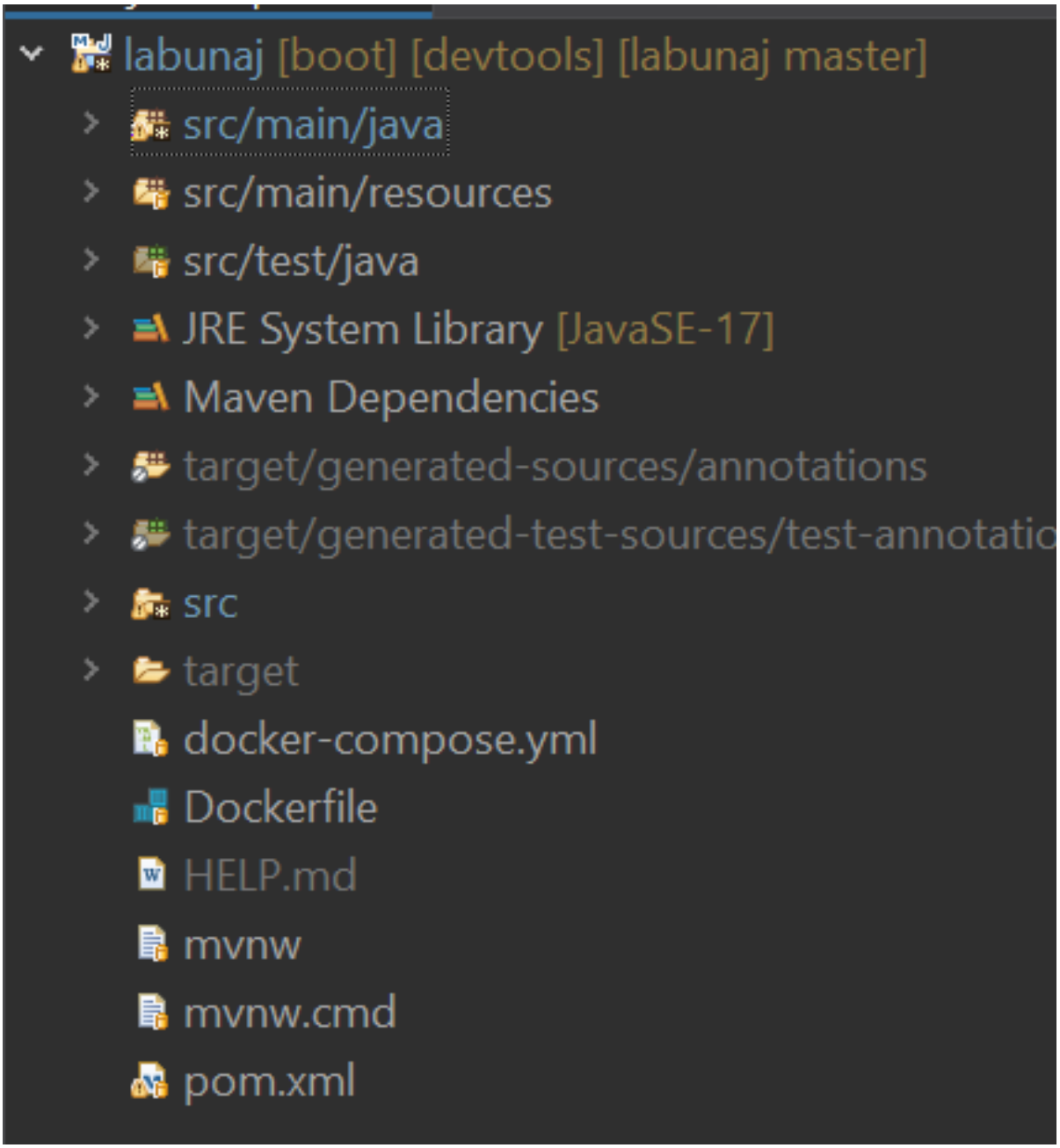
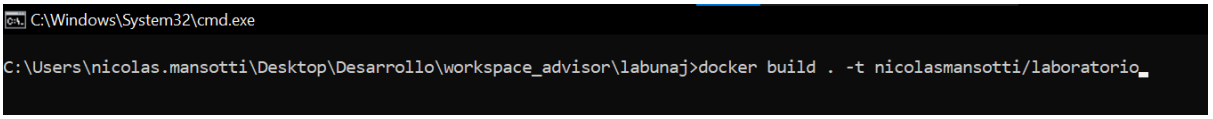


Figura 27. Directorio de archivos en IDE.



Para crear la imagen, hay que posicionarse en el directorio en donde se encuentre el archivo Dockerfile y ejecutar el siguiente comando:

Figura 28. Línea de comandos para crear la imagen Docker

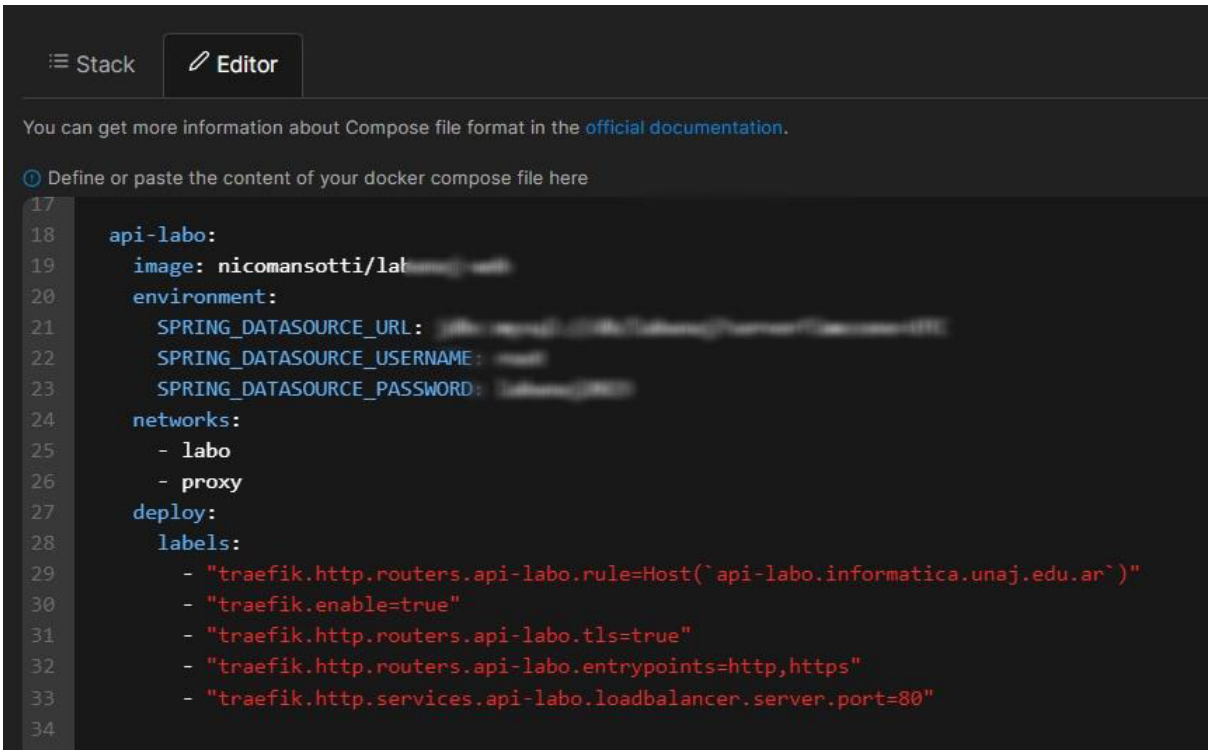


Para subir la imagen a docker-hub ejecutar el siguiente comando, previamente habiéndose logueado en Docker:

Figura 29. Línea de comandos para subir la imagen a Docker-Hub.

Configurar la API en el Docker Compose:

Figura 30. Docker-Compose.



6.3 Zona horaria

Para configurar la zona horaria hubo muchos problemas, ya que automáticamente las funciones de fecha y hora de Java no toman la zona horaria de Argentina. Al principio se pensó en configurar la hora según la franja horaria de donde se realicen solicitudes. Por ejemplo, si las solicitudes se hacen desde otro país, la zona horaria sería otra. Finalmente se decidió acotar la zona horaria a Argentina (GMT-3), ya que el alcance de la aplicación es de carácter nacional, y se va a usar para la UNAJ inicialmente, con posibilidad de alcanzar otras universidades.

6.4 Subir el código a un servidor

El código de la aplicación está separado en lo que es *backend* y *frontend*. El *frontend* tiene que consumir la API.

Para alojar la API a un servidor y que pueda consumirlo el *frontend*, lo ideal era que se subiera a docker-hub, ya que definitivamente el código iba a terminar alojado ahí. Al no contar con los conocimientos necesarios para subirlo a docker-hub se pensaron alternativas para utilizar otro *hosting* y disponibilizarlo para que lo consuma el *frontend*. Se analizó la alternativa de alojar la API a un *hosting* gratuito, ya que esa solución iba a ser temporal, hasta adquirir conocimientos y poder subirlo a docker-hub. Se pudo subir la API a dos *hosting* gratuitos, pero el tiempo de prueba gratuito fueron 3 días y no sirvió para que el *frontend* pueda consumirlo. Por ende, no se encontró la opción y el *hosting* adecuado para alojar el código. Como la solución iba a ser temporal, no se eligió la opción de un *hosting* pago. Lo negativo de esta situación es que el *frontend* no puede consumir la API y hacer pruebas con datos reales.

Luego de un tiempo, se adquirieron los conocimientos y se pudo cargar el código a docker-hub, para luego integrarlo en el *stack* de porteiner, junto a los otros servicios que incluye la aplicación, como la base de datos, el adminer y el *frontend*.

Al levantar el *stack* surgió el problema de que al querer acceder a la API, la respuesta era un 404 not found. Al revisar por qué podía darse este problema, detecté que el puerto en la API estaba apuntando al 8080, y en el *stack* estaba definido el puerto 80. Al modificar el puerto, se pudo utilizar la API sin problemas estando levantada en producción.

6.5 Imagen JPG del laboratorio

Cada laboratorio tiene una imagen que el administrador puede subir para identificar al laboratorio. El tipo de dato usado para guardar esta imagen en la base de datos era un String. Esto provocó que las imágenes con alta calidad no se puedan guardar, ya que la base de datos no soportaba ese tamaño y solamente admitía tamaños pequeños de imágenes jpeg y jpg. La imagen estaba llegando en formato base64. Y se guardaba el texto plano en la base de datos. La gran cantidad de caracteres hacía que no pueda soportar este formato.

Para solucionarlo, se optó por otro tipo de dato. Este nuevo tipo de datos de la imagen a guardar en la base de datos es de tipo *byte[]*, es decir, un *array* de bytes.

La diferencia entre estos dos tipos de datos radica en la representación de los datos y el tipo de información que se almacena.

Al tener un String para guardar la imagen, se convierte la imagen binaria en una cadena de texto utilizando algún método de codificación como Base64.

Es de fácil almacenamiento en la base de datos, pero tiene una sobrecarga en el tamaño, ya que la representación en Base64 incrementa el tamaño del archivo, aproximadamente en un 33%.

Al optar por el tipo de datos de *Array* de bytes (byte[]) para la imagen, se tiene una representación directa de los datos binarios de la imagen.

Es más eficiente en cuanto al tamaño en comparación con la codificación Base64, ya que no hay sobrecarga adicional debido a la codificación.

Hay bases de datos que no soportan el almacenamiento en Base64. Al utilizar MySQL, tenemos la ventaja que si tiene soporte directo para almacenar datos binarios utilizando tipos de datos específicos.

Como se mencionó, MySQL tiene soporte directo para almacenar datos binarios utilizando tipos de datos específicos. Los tipos de datos binarios en MySQL son:

BLOB (Binary Large Object): este tipo de dato se utiliza para almacenar datos binarios de tamaño considerable. Los BLOBs pueden ser de varios tipos, dependiendo del tamaño máximo que admiten:

- TINYBLOB: capacidad máxima de 255 bytes.
- BLOB: capacidad máxima de 65,535 bytes.
- MEDIUMBLOB: capacidad máxima de 16,777,215 bytes.
- LONGBLOB: capacidad máxima de 4,294,967,295 bytes.
- BINARY y VARBINARY: estos tipos de datos están diseñados para almacenar datos binarios de longitud fija o variable, respectivamente.

BINARY: almacena cadenas de bytes de longitud fija.

VARBINARY: almacena cadenas de bytes de longitud variable, con una capacidad máxima definida.

Se eligió por el tipo de dato LONGBLOB, ya que puede almacenar imágenes con grandes tamaños sin problemas.

7. Mejoras a futuro

Integración del login y registro con el SIU

Una de las mejoras a futuro puede ser realizar una integración del login y registro de usuarios con el sistema de datos del SIU Guaraní. Esto facilitaría y simplificaría el proceso de acceso y registro de los alumnos a la aplicación web, ya que los datos de cada alumno se encuentran almacenados en los registros del SIU. Esto mejoraría la experiencia del usuario, reduciendo la necesidad de crear y recordar nuevas credenciales. Asimismo, agregaría más seguridad, ya que la integración con el sistema SIU Guaraní no solo autentica la identidad del usuario como estudiante de la universidad, sino que además, se aprovecharía la autenticación y autorización ya existente en los datos del SIU. Esto no se llegó a aplicar, ya que estaba fuera del alcance inicial del proyecto, pero se podría hacer consumiendo una API de datos del SIU y le daría un salto de calidad a la aplicación.

Aplicación mobile

Otra mejora a futuro podría ser la creación de una aplicación móvil complementaria a la página web. Tener una aplicación en el celular, beneficia la accesibilidad y la usabilidad de los servicios ofrecidos por la plataforma, utilizando la API ya existente.

Una app móvil habilitaría a los usuarios para acceder a los servicios en cualquier momento y desde cualquier ubicación, dándole a los alumnos una experiencia más adaptable y

accesible para utilizar el sistema de turnos. Además, la APP móvil tendría la capacidad de enviar notificaciones instantáneas para alertar sobre turnos y reservas, aprovechando las funcionalidades de los celulares para mejorar la experiencia del usuario.

Agregar calendario de Google

Al momento que un alumno reserva un turno, se envía un mail de recordatorio. Este email puede tener integrado el calendario de Google. La adición del calendario de Google mejorará significativamente la experiencia del usuario al integrar la funcionalidad de recordatorio de turnos en el flujo de reservas. Al permitir la sincronización con el calendario de Google, los alumnos podrán tener una visión más completa de sus turnos, recibiendo alertas automáticas, pudiendo recordar y organizar las reservas programadas, y teniendo también una mayor puntualidad en el cumplimiento de los turnos.

8. Conclusiones

De acuerdo con los requerimientos planteados al principio del proyecto, se logró el objetivo propuesto por la UNAJ, ofreciendo una solución integral para la gestión de usuarios y reserva de turnos de laboratorios remotos a través de una aplicación web. La investigación y desarrollo de este sistema permite centralizar y optimizar la reserva de laboratorios, proporcionando a los estudiantes la capacidad de reservar turnos sin superposiciones y acceder a los recursos de manera remota.

La implementación del sistema de registración de usuarios, junto con la posibilidad de que un administrador configure los laboratorios, sus horarios y demás configuraciones, garantiza una experiencia completa y organizada para toda la comunidad educativa.

La colaboración con mi compañero de proyecto, Alejandro, y el tener roles definidos y tareas asignadas para el trabajo, facilitó un desarrollo equilibrado del *backend* y *frontend* de la aplicación. La coordinación constante a lo largo del proyecto garantizó la definición y aplicación eficaz de las funcionalidades necesarias.

En conclusión, la combinación de investigación, desarrollo y colaboración efectiva fue clave para alcanzar los objetivos planteados y ofrecer una solución robusta para la gestión de turnos y acceso a laboratorios remotos en el ámbito educativo de la UNAJ.

Este desarrollo representa un punto de partida sólido que puede servir como base para futuras innovaciones tecnológicas y mejoras continuas en la gestión de aplicaciones y desarrollos académicos. La implementación exitosa de esta aplicación web no solo beneficia a la comunidad educativa actual, sino que también sienta las bases para un desarrollo continuo y escalable. Las soluciones implementadas en este proyecto pueden servir como referencia para abordar otros desafíos similares en el entorno educativo, dando así la posibilidad de crecimiento y mejora constante en la universidad.

9. Bibliografía

Luis Miguel López Magaña (17/01/2020). Qué es Json Web Token y cómo funciona. https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

Docker Docs. Docker Overview. https://docs.docker.com/get-started/overview/

Intercambio de recursos de origen cruzado. https://developer.mozilla.org/es/docs/Web/HTTP/CORS

Docker Docs. Docker Hub. https://docs.docker.com/docker-hub/

Sommerville,I. (2005). *Ingeniería del Software*.Madrid: Pearson Educación S.A.

Pressman, R. (2010) *Ingeniería del Software. Un enfoque práctico.* México D.F.: McGraw-Hill.