Alejandro Ezequiel Mendez

Plataforma de laboratorios remotos : Optimización de gestión y acceso a laboratorios remotos

2023

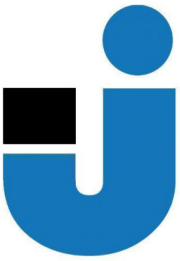
Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática



PRÁCTICA PROFESIONAL SUPERVISADA
Informe final

Plataforma de laboratorios remotos

Mendez Alejandro Ezequiel

PRÁCTICA PROFESIONAL SUPERVISADA (PPS)
Plataforma de laboratorios remotos
Informe final

DATOS DEL ESTUDIANTE

Apellido y Nombres: Mendez Alejandro Ezequiel

DNI: 34452525

N º de Legajo: 1967

Correo electrónico: mendezalejandro.e@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 43

PPS enmarcada en el artículo (4 ó 7) de la Resolución (CS) 103/16. (en caso de ser artículo 7 aclarar en cuál de las dos alternativas posibles se encuadra)

DOCENTE SUPERVISOR

Apellido y Nombres: Ing. Florencia Ayala

Correo electrónico: florencia.ayala@unaj.edu.ar

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: UNAJ

Dirección: Av. Calchaquí 6200 – Florencio Varela – Bs As - Argentina

Teléfono: +54 11 4275-6100

Sector: Departamento de desarrollos y nuevas tecnologías

TUTOR DE LA ORGANIZACIONAL

Apellido y Nombres: Dr. Ing. Martín Morales

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

FIRMA DEL COORDINADOR DE LA CARRERA

Firma Estudiante:	Firma Docente Supervisor:	Firma Docente tutor TAPTA:	Firma Tutor organizacional:

Resumen

El presente informe es un trabajo de desarrollo en el marco de la materia Práctica Profesional Supervisada (PPS) enfocado en la creación de una plataforma para optimizar la gestión de laboratorios remotos, los cuales se consideran entidades clave en diversas disciplinas académicas como herramientas de aprendizaje. El enfoque principal de esta PPS es proporcionar una solución integral para unificar criterios de seguridad, acceso y tiempo de uso en estos laboratorios y proporcionar a los estudiantes la capacidad de autogestión, solicitud de turnos y acceso a laboratorios, al mismo tiempo, ofrecer al personal administrativo una herramienta eficaz para gestionar usuarios y laboratorios.

Abstract

The present report represents a development effort within the framework of the Supervised Professional Practice (SPP) course, with a focus on creating a platform to optimize the management of remote laboratories. These laboratories are considered pivotal entities in various academic disciplines, serving as essential learning tools. The primary goal of the SPP is to provide a comprehensive solution for unifying criteria related to security, access, and usage time in these laboratories. Its specific objective is to empower students with the ability to self-manage, request appointments, and access laboratories, while concurrently offering administrative staff an effective tool for managing users and laboratories.

Agradecimientos

A los profesores de la universidad por su vocación y dedicación.

A los compañeros y amigos que me apoyaron siempre.

A mi hijo Santino por ser mi horizonte en todo el camino.

A mi familia especialmente por el apoyo incondicional desde que tengo memoria.

¡A todos, muchas gracias!

Índice

1. Introducción.....5

1.1. Resumen.....5

1.2. Objetivos.....6

1.3. Tareas a ejecutar.....6

2. Desarrollo.....7

2.1. Contexto.....7

2.2. Relevamiento.....7

2.3. Requisitos y Alcance.....8

2.4. Tareas.....10

2.4.1. Análisis y diseño de arquitectura.....10

2.4.2. Metodología de desarrollo.....11

2.4.3. Definición de tecnologías y herramientas.....12

2.4.4. Prototipado.....14

2.4.5. Desarrollo de interfaces.....15

2.4.5.1. Marco de trabajo.....15

2.4.5.2. Instalación del entorno de trabajo.....16

2.4.5.3. Configuración y estructura inicial de la aplicación.....17

2.4.5.4. Definición de temas de usuarios.....19

2.4.5.5. Componentes reutilizables..... 20

2.4.5.6. Interfaces de usuarios..... 25

2.4.6. Adaptación de laboratorio remoto de Arduino..... 40

2.4.7. Implementación..... 43

2.4.8. Pruebas..... 44

6. Conclusiones..... 48

7. Reflexión sobre la Práctica Profesional Supervisada como espacio de formación..... 49

8. Índice de figuras..... 50

9. Bibliografías..... 50

1. Introducción

1.1. Resumen

La Universidad Nacional Arturo Jauretche (UNAJ) cuenta con laboratorios remotos orientados a distintas disciplinas tales como Física, Química, Tecnologías de la Información y otros. Un laboratorio remoto consiste en un acceso remoto y naturaleza real que representa la conexión entre un usuario y un laboratorio físico real, y permite la interacción entre ambos a través de una conexión de red.

Los laboratorios remotos de la universidad presentan una diversidad de criterios en términos de tiempo de utilización, control de acceso y medidas de seguridad. Algunos ofrecen acceso público desde fuera de la universidad mediante un registro previo, mientras que otros carecen de cualquier forma de control de acceso, permitiendo el acceso privado o físico. Además, es importante destacar que el mecanismo de uso es manual e informal, ya que los estudiantes que necesitan utilizar estos laboratorios pactan de manera informal con los profesores el tiempo de uso. Asimismo, hay laboratorios que no proporcionan acceso remoto ni implementan medidas de control, limitando su accesibilidad únicamente de manera presencial.

La presente Práctica Profesional Supervisada (PPS) consiste en un trabajo de desarrollo de una aplicación web multiplataforma y multilenguaje que unifique los criterios de seguridad, control de acceso, tiempos de uso y accesibilidad a dichos laboratorios.
La aplicación será responsiva para llegar a la mayor cantidad de dispositivos de manera que se pueda utilizar desde un móvil, tableta o PC. Será multiplataforma para que se pueda

acceder desde cualquier navegador, y multilenguaje para proveer el acceso de estudiantes de otras universidades en el futuro. De esta manera, la universidad podrá contar con una herramienta que digitalice la información de sus laboratorios, permita a los estudiantes autogestionar sus turnos y a los administradores de la universidad, delegar el control de acceso y seguridad al sistema, automatizar y gestionar la solicitud de turnos, administración de usuarios y laboratorios.

El alcance de este trabajo será el desarrollo del *frontend* de la aplicación utilizando el *framework* Angular, en conjunto con el equipo de *backend* de otro alumno y los responsables del desarrollo y mantenimiento de los laboratorios remotos.

1.2. Objetivos

El objetivo principal de la PPS es desarrollar una plataforma web para la gestión de turnos de laboratorios remotos que sea multiplataforma, multilenguaje, permita a los estudiantes autogestionar sus turnos y al personal de la universidad administrarlos.

Los objetivos específicos son:

- Diseñar la estructura para una aplicación sólida y con componentes reutilizables que sea fácil de mantener en el futuro.
- Centralizar el control de accesos y de turnos.
- Unificar los criterios de gestión de turnos.
- Permitir que el alumnado se autogestione.
- Delegar el control de turnos al sistema.

- Digitalizar y delegar el control de laboratorios y usuarios a los administradores.

1.3. Tareas a ejecutar

Las tareas a ejecutar para cumplir con el proyecto son:

- Análisis de la arquitectura actual
    - Reunión de entendimiento acerca de los conceptos básicos del proceso, la problemática, y la arquitectura actual.
    - Relevamiento de información.
- Definición de requisitos y alcance.
    - Reunión con el equipo interno de la universidad para definir los requisitos del nuevo sistema y el alcance del mismo.
- Diseño y arquitectura del sistema.
    - Reunión con el equipo de desarrollo para definir y diseñar la arquitectura del nuevo sistema.
- Análisis de tecnologías
    - Investigar y definir la tecnología a utilizar acorde a este proyecto con el equipo de desarrollo y el equipo interno de la universidad.
    - Reunión para establecer la arquitectura y tecnologías a utilizar en el proyecto.
- Diseño y prototipado de interfaces
    - Prototipado de las interfaces para la aplicación.
- Desarrollo de la aplicación

- Desarrollo de las interfaces de la aplicación con las funcionalidades requeridas.

- Implementación y pruebas
  - Implementación de la aplicación en la infraestructura de la universidad.
  - Optimización y pruebas de rendimiento.
  - Pruebas funcionales con el equipo interno de la universidad.
- Entrega final
  - Entrega de código fuente, contenedores y documentación del proyecto en los repositorios de la universidad.

2. Desarrollo

2.1. Contexto

El presente trabajo se desarrolla en la Universidad Nacional Arturo Jauretche como parte de la materia Práctica Profesional Supervisada con el objetivo de consolidar los contenidos aprendidos durante la carrera en un proyecto de utilidad tanto para el alumno como para la universidad. En base a esta premisa, se decidió continuar el proyecto de Laboratorios Remotos iniciado por diferentes estudiantes de la universidad el cual busca dotar a los estudiante con más herramientas educativas que los acerquen a los entornos reales de trabajo de distintas disciplinas como Física, Informática, Robótica, Electrónica, y otras.

En este contexto, y para unificar los distintos proyectos de Laboratorios Remotos se definió el proyecto "Plataforma de Laboratorios Remotos" cuyos objetivos son:

- Proveer a la universidad de una herramienta que permita digitalizar los laboratorios remotos
- Permitir el acceso directo de los estudiantes a los laboratorios a través de dicha plataforma.

2.2. Relevamiento

A partir de la premisa del proyecto, se inició un proceso de relevamiento durante las primeras semanas con el equipo de la universidad para relevar el funcionamiento de los laboratorios remotos. En dicho proceso se especificó que existen actualmente 2 laboratorios remotos configurados y funcionando en distintas aulas de la universidad, estos son:

- Sistemas embebidos - Laboratorio de Arduino:
    - Permite a los estudiantes manipular una placa Arduino a través de una interfaz web que se encuentra instalada en una de las PCs de los laboratorios de la universidad.
    - El control de usuarios y turnos se encuentra limitado a la utilización en la propia PC.
    - No dispone de un acceso hacia el exterior, quiere decir que únicamente es accesible físicamente en el laboratorio y no a través de internet.
- Física - Laboratorios de Ondas:

- Permite a los estudiantes manipular un dispositivo mecánico que mediante un control permite la oscilación de una cuerda.
- No tiene un control de usuarios ni de turnos y se encuentra limitado a la utilización en la propia PC.
- No dispone completamente de una salida al exterior.

En el proceso también se relevó que, para acceder a estos se coordinan fechas y horarios entre profesores y estudiantes para la utilización de los mismos. De aquí surge la necesidad de poder contar con un control de usuarios y turnos centralizado que elimine la informalidad, controle y organice los accesos y digitalice la información. Además, se planteó la necesidad de que puedan ser accedidos en el futuro por estudiantes de otras universidades.

En base a la información relevada se plantearon las siguientes necesidades:

- Centralización del control de accesos a los laboratorios remotos y de la organización de turnos.
- Autogestión de los estudiantes y administradores.
- Digitalización de los laboratorios remotos y de los usuarios.
- Mayor accesibilidad posible.

2.3. Requisitos y Alcance

A partir del análisis del relevamiento realizado, en las semanas posteriores se fueron definiendo los requisitos técnicos, funcionales y no funcionales del proyecto con el equipo de la universidad.

Funcionales:

- Registración de nuevos usuarios con validación de email
- Acceso con usuario/email y password
- Recuperación de cuenta
- Operaciones para administradores
    - Creación, modificación, eliminación y búsqueda de Laboratorios.
    - Restricción de accesos por laboratorios.
    - Creación, modificación, eliminación y búsqueda de Usuarios.
    - Búsqueda y cancelación de turnos.
- Operaciones para estudiantes
    - Programación de turnos por laboratorios.
    - Restricción de turnos por laboratorio.
    - Envío de emails con confirmación de turno.
    - Cancelación de turnos.
    - Acceso a laboratorios remotos.

No funcionales:

- Disponibilidad en dispositivos reducidos (Web, Tablet y Mobile).
- Disponibilidad de distintos idiomas.
- Accesibilidad a través de internet.
- Interoperabilidad entre laboratorios remotos y la plataforma.

Técnicos:

- Utilización de la infraestructura de la universidad.
- Utilización de herramientas de código libre.

En cuanto al alcance del proyecto, se estableció que esta PPS se enfocaría en el desarrollo de la aplicación web, con especial atención a las interfaces de usuario, así como en la modificación de algunas funcionalidades en el laboratorio de Arduino. Esto incluye además de los requisitos ya mencionados para la aplicación web, las siguientes modificaciones en el laboratorio de Arduino:

- Quitar el control de usuarios y turnos.
- Implementar la validación de accesos.

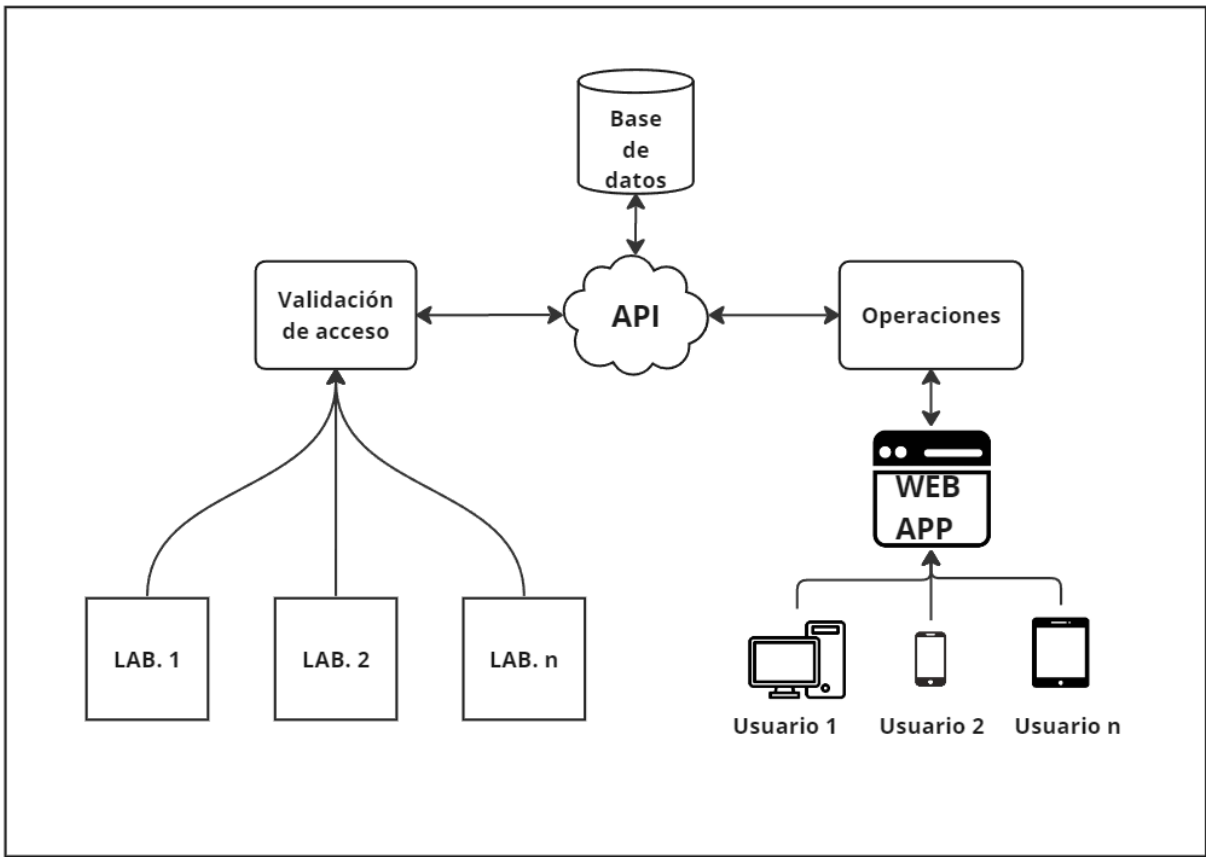
2.4. Tareas

2.4.1. Análisis y diseño de arquitectura

A partir del relevamiento inicial y el análisis de los requisitos, se comenzó con el proceso de análisis y diseño de la arquitectura del nuevo sistema "Plataforma de laboratorios remotos", de ahora en más llamado "Plataforma", "Proyecto" o "Aplicación".

Uno de los inconvenientes que presentaba la base actual de turnos y usuarios, era que cada laboratorio remoto tenía diferentes criterios en cuanto al control de usuarios y turnos. A partir de este problema se buscó como unificar todos esos criterios, digitalizar y centralizar la información. Para solucionar estos problemas, se decidió crear una base de datos donde guardar la información en lugar de que cada laboratorio lo haga por su cuenta. Una interfaz web desde donde los usuarios pudieran autogestionarse y que permitiera unificar los criterios de acceso, seguridad y tiempo de uso de los laboratorios; además, permitir a los administradores gestionar los laboratorios y turnos en caso de que los estudiantes tuvieran inconvenientes y además, la creación de una API que permita consumir la información de la base de datos, y valide el acceso desde los laboratorios. En resumen, la arquitectura de la plataforma estaría basada en 3 pilares esenciales compuestas por, una interfaz web para los usuarios finales, una base de datos para persistir y centralizar la información y una API (*Application Programming Interface*) que funcione como puente de comunicación entre los laboratorios y la plataforma.

Figura 1. Flujo de interacciones entre partes del sistema.



2.4.2. Metodología de desarrollo

Antes de comenzar a explicar la metodología de desarrollo, es importante señalar que las tareas del proyecto se dividieron entre 4 personas y se asignaron roles específicos para cada uno de ellos, o simplemente se heredaron por el contexto.

- 1 *Product Owner/Stakeholder*: encargado de validar que las entregas de los desarrollos cumplan con los requisitos planteados.
- 1 *Stakeholder*: responsable del laboratorio de Arduino.
- 1 *Frontend developer*: responsable del desarrollo de aplicación web para la plataforma y de esta PPS.

- 1 *Backend developer*: responsable del desarrollo de la API y base de datos para la plataforma.

Dado el equipo mencionado para el proyecto, se utilizó una metodología ágil como enfoque, la cual es un concepto de gestión de proyectos que implica dividir el proyecto en fases y hace hincapié en la colaboración y la mejora continua donde los equipos siguen un ciclo de planificación, ejecución y evaluación (Somervielle, Ingeniería de software). El motivo de esta elección fue para ir presentando avances del proyecto en tiempos dinámicos e ir obteniendo un feedback constante con el *product owner*.

Específicamente se utilizó la metodología Kanban para la gestión del proyecto dado que es un enfoque más orientado a un flujo de trabajo continuo que a la entrega de valor continuo (no había presión de ir presentando avances en ciclos tan cortos), además este enfoque se basa en tareas visuales para gestionar dichos flujos con lo cual es mucho más simple para trabajar y dar seguimiento en equipos chicos y que no requieren de mucha burocracia organizativa.

El procedimiento se ejecutó de la siguiente manera:

1. Desarrollo de la aplicación.
2. Reuniones virtuales para presentar avances o nuevas funcionalidades y definir nuevos requisitos en periodos de entre 1 y 3 semanas.
3. Reuniones presenciales en la universidad para implementar o testear en el laboratorio.

2.4.3. Definición de tecnologías y herramientas

Con respecto a las tecnologías y herramientas utilizadas en este proyecto, se especificarán por temática:

- **Comunicacionales**
  - Google Meet: es la aplicación de videoconferencias de Google, para navegadores web y dispositivos móviles, enfocada al entorno laboral y educativo.
    - Utilizada para realizar las reuniones periódicas de avances con el equipo.
  - WhatsApp:
    - Para coordinar reuniones y un contacto directo entre el equipo.
- **Organizacionales**
  - Trello: es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas.
    - Para organizar las tareas de una manera gráfica y sencilla entre el equipo.
    - La elección de esta herramienta es porque es muy sencilla, extensible, permite integraciones y además colaborar con otras personas.
  - Miró: es una aplicación que permite organizar ideas, tomar decisiones, crear gráficos y mapas mentales.
    - Para crear diagramas de flujos y ayudar al desarrollo de la aplicación.

- Se eligió esta herramienta porque permite crear diagramas de flujo de manera muy sencilla e intuitiva.

- **De gestión de cambios**
  - Github: es una plataforma basada en Git, un sistema de control de versiones que permite a los desarrolladores mantener un registro completo y detallado de los cambios realizados en sus proyectos.
    - Para gestionar y organizar los cambios de código realizados en el desarrollo de la aplicación.
    - Se eligió esta herramienta porque ya se tenía una cuenta en esta plataforma, además es gratis, sencilla y muy popular.
- **De infraestructura**
  - Docker: es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.
    - Para desplegar la aplicación en la infraestructura de la universidad, y realizar pruebas locales como si estuviera en un ambiente productivo, y compartir la aplicación con el equipo sin necesidad de compartir el código.

        - La universidad definió esta herramienta como obligatoria para poder entregar la PPS con tecnologías que puedan ser utilizadas en su infraestructura.
    - Porteiner: es una herramienta web *open-source* que permite gestionar contenedores Docker. Permite administrar contenedores de forma remota o local, la infraestructura de soporte y todos los aspectos de las implementaciones de Kubernetes, Docker standalone y Docker Swarm.
        - La universidad definió esta herramienta como obligatoria para poder entregar la PPS con tecnologías que puedan ser utilizadas en su infraestructura. Se utilizó para alojar la aplicación y realizar pruebas reales en un ambiente de producción.
- **De desarrollo**
    - Visual Studio Code: es un editor de código gratuito, ligero y extensible para la construcción de aplicaciones web, de escritorio y móviles.
        - Para editar el código de la aplicación en todo el desarrollo.
        - Se eligió esta herramienta porque tiene una interfaz muy sencilla, rápida y permite integraciones con otras herramientas como Github Copilot y Docker.
    - GitHub Copilot: es una herramienta de inteligencia artificial basada en la nube desarrollada por GitHub y OpenAI para ayudar y asistir a los usuarios de

Visual Studio Code, Visual Studio, Neovim y los entornos de desarrollo integrado (IDE) de JetBrains mediante el autocompletado de código.

- Se aprovechó una licencia personal existente para utilizarla como ayuda en el autocompletado de código.

- Figma: es una herramienta de prototipado de aplicaciones que permite diseñar vistas interactivas y facilita la creación de interfaces de usuarios.
- Angular: es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.
    - La elección de este framework se explicará en detalle en la siguiente sección.
- Angular Material: es una biblioteca de componentes y directivas de diseño de interfaz de usuario para aplicaciones web Angular.
- Ngx-Bootstrap: es una biblioteca de herramientas de código abierto optimizadas para el diseño de sitios y aplicaciones web responsivas.
- Ngx-Translate: es una biblioteca fundamental para internacionalización (i18n) en el desarrollo con Angular que facilita la creación de una interfaz multilingüe.
- RxJS: es una biblioteca para componer programas asíncronos y basados en eventos, mediante secuencias observables.

2.4.4. Prototipado

El prototipado permite visualizar y probar de manera práctica la funcionalidad y el flujo de la aplicación antes de entrar en la implementación completa. Para el proyecto se utilizó Figma como herramienta con el objetivo de crear vistas de la aplicación y así contar con un panorama general de cómo quedaría la aplicación con respecto a la estructura de las interfaces. El objetivo fué diseñar una aproximación inicial, que si bien fueron cambiando en el tiempo con el *feedback* de los *stakeholders*, permitieron sentar las bases estructurales de las vistas de la aplicación tales como la utilización de grillas para mostrar información, ubicación de contenedores, textos y botones.

A continuación se muestra el prototipo inicial del menú, contenedores principales e interfaz de usuarios.

Figura 2. Prototipo de interfaz Lista de Usuarios.

2.4.5. Desarrollo de interfaces

2.4.5.1. Marco de trabajo

En cuanto al desarrollo de aplicación, como se especificó en la sección 2.4.3, se optó por utilizar Angular como marco de trabajo y la elección de este *framework* se basó en el hecho de que se necesitaba crear una aplicación de interfaz sencilla, escalable, con pocos recursos, que sea multilenguaje y funcional a la mayoría de los navegadores y dispositivos. Sin embargo la elección de esta en particular con respecto a otras, fue por el conocimiento y experiencia previa en el uso de esta tecnología.

Angular es un marco (*framework*) de desarrollo de aplicaciones web creado y mantenido por Google. Se utiliza para construir aplicaciones web de una sola página (SPA) utilizando como base los lenguajes de marcado y programación HTML, Typescript y SCSS. Una SPA “Single Page Application" (Aplicación de Página Única) es un tipo de aplicación web que carga una sola página HTML y actualiza dinámicamente el contenido de esa página mientras el usuario interactúa con la aplicación. En lugar de cargar páginas web completamente nuevas cada vez que se realiza una acción, como hacer clic en un enlace, una SPA utiliza tecnologías JavaScript para cambiar el contenido de la página de manera fluida.
Algunas de las características importantes de Angular y que sirvieron para solucionar cada una de las necesidades son:

- Un marco basado en componentes para crear aplicaciones web escalables.
    - Los componentes son el bloque de construcción fundamental para crear aplicaciones en Angular. Al aprovechar la arquitectura de componentes, Angular proporciona una estructura para organizar el proyecto en partes manejables y bien organizadas con responsabilidades claras para que el código sea mantenible y escalable.
- Una colección de bibliotecas bien integradas que cubren una amplia variedad de funciones, incluido el enrutamiento, la administración de formularios y la comunicación cliente-servidor entre otros.
- Un conjunto de herramientas de desarrollo que ayudan a desarrollar, compilar, probar y actualizar el código de una manera eficiente.

- Fácil integración con librerías de terceros.

El detalle de cada funcionalidad, herramienta o librería utilizada se especificará durante la descripción de las tareas de desarrollo y específicamente en el lugar donde se utilizaron.

2.4.5.2. Instalación del entorno de trabajo

El primer paso para comenzar a desarrollar la aplicación fué instalar todas las herramientas necesarias. Como se describió en la sección anterior, la herramienta elegida fue Angular la cual está basado en Javascript y para poder ejecutar localmente la aplicación se necesita de un servidor, para ello se instaló Node JS el cual es un entorno de ejecución del lado del servidor que permite ejecutarlo y se integra perfectamente con otras funcionalidades que son necesarias para construir la aplicación como el CLI (Command Line Interface), este permite ejecutar comandos de compilación, creación, depuración y pruebas.

NodeJS no solo funciona como un servidor local, sino también como un administrador de dependencias, esto nos habilita a ejecutar comandos de descarga e instalación de librerías de terceros que facilitan el desarrollo.

Con estas herramientas instaladas el entorno de desarrollo quedó configurado y listo para trabajar.

2.4.5.3. Configuración y estructura inicial de la aplicación

Estructura de la aplicación

A partir de la instalación de Angular CLI se procedió a crear la estructura básica de la aplicación. Para ello se ejecutaron los comandos necesarios para iniciar un nuevo proyecto a través de la CLI y se generó la estructura básica de la aplicación donde se incluyó todos los componentes y dependencias necesarias. Un punto importante a destacar es la elección de una estructura de carpetas organizada que permita en el futuro agregar nuevas características con cierta nomenclatura. Las estructuras de carpetas de Angular por defecto son bastantes sencillas, pero cuando se manejan muchas interfaces se va tornando más compleja la organización. En la siguiente captura se observa cómo quedó la estructura final pero que se planteó desde esta etapa.

Figura 3. Estructuras de carpetas en el código fuente del proyecto.

Aquí se destacan las carpetas:

- **Core**: donde se guardan todos los componentes que son parte del núcleo de la aplicación y ejecutan acciones básicas. Por ejemplo, un componente de conexión a la API.
- **Features**: donde se guardan específicamente las interfaces navegables por los usuarios. Por ejemplo, un componente de acceso a la interfaz de Turnos.
- **Shared**: donde se guardan los componentes que son compartidos por las features, pero que no son parte del núcleo de la aplicación. Por ejemplo, contenedores de elementos.

Instalación de dependencias

Luego de instalar y configurar el entorno de desarrollo, se procedió a instalar las bibliotecas necesarias para comenzar a desarrollar las interfaces de la aplicación. A continuación se detalla cuales fueron y el motivo de la elección.

- Angular Material: es una biblioteca de componentes de interfaz de usuario (UI) desarrollada por el equipo de Angular en colaboración con Google. Está diseñada para facilitar la creación de interfaces de usuario atractivas y consistentes en aplicaciones web desarrolladas con Angular.

Esta librería ofrece una lista de componentes que se pueden reutilizar o modificar a gusto del usuario, también ofrece una serie de lineamientos que permiten definir una estructura organizada y a la vez, reducir el tiempo de desarrollo en cuestiones como definición de colores o tipografías.

Utilizar esta biblioteca ahorró mucho tiempo en la creación de la aplicación ya que evitó crear componentes básicos como tarjetas, contenedores, botones, listas, tablas, entre otros.

- Ngx-Bootstrap: es una biblioteca de componentes que se construye sobre el popular *framework* de diseño *front-end* Bootstrap. Esta biblioteca proporciona componentes reutilizables y servicios que están diseñados para integrarse fácilmente con este tipo de proyectos. En lugar de utilizar la versión de jQuery de Bootstrap, ngx-bootstrap se ha reconstruido utilizando Angular.

  Se decidió utilizar esta biblioteca porque provee clases de CSS que ayudan a construir interfaces responsivas.

- Ngx-Translate: es una biblioteca especializada para la gestión de internacionalización (i18n) y localización (i10n). Desarrollada para simplificar la implementación de múltiples idiomas, ngx-translate ofrece herramientas y servicios que permiten la fácil integración de contenido traducido en la interfaz de usuario. Se decidió utilizar esta biblioteca porque a diferencia de otras soluciones, no solo se limita a traducciones estáticas, sino que también facilita la actualización dinámica de contenido basada en

el idioma seleccionado por el usuario.

2.4.5.4. Definición de temas de usuarios

Una de las características útiles para el desarrollador que tiene Angular Material son el manejo de *Themes* (Tema, conjunto de colores y tipografías) ya que, con poca configuración define y aplica colores y tipografías a todos los componentes de la aplicación como botones, contenedores, tarjetas, etc. Esto evita duplicar código de estilos gráficos en todas las interfaces, además ofrece un mecanismo de personalización que permite aplicar manualmente el tema a los componentes personalizados que se hayan desarrollado sin salirse del estándar. Utilizando esta caracteristica se definió una paleta de colores acorde a los utilizados en las paginas de la universidad que incluyen una gama de Negro, Celeste y Blanco. En cuanto a tipografía se definió utilizar una fuente muy popular llamada “Poppins” por la alta aceptación de usuarios que tiene por su estilo moderno, también tiene una excelente legibilidad en interfaces móviles en adelante la cual fué un requisito no funcional para el proyecto.

2.4.5.5. Componentes reutilizables

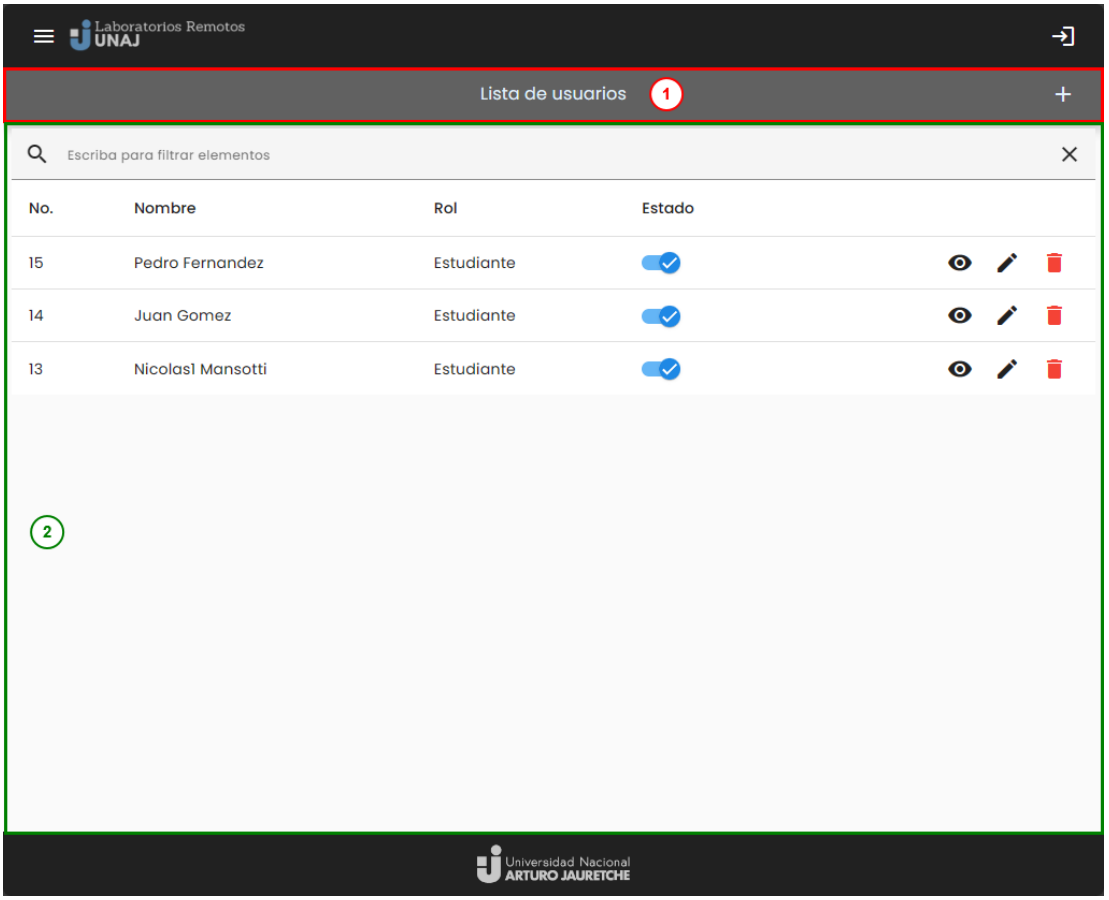
<u>Componentes</u>

Los componentes son bloques fundamentales de construcción de una aplicación. Representan unidades autónomas y reutilizables de la interfaz de usuario y la lógica de presentación. Cada componente está asociado a una vista y controla una parte específica de la interfaz de usuario. Los *Components* (Componentes) son parte integral de la arquitectura de

Angular y siguen el patrón de diseño Modelo-Vista-Controlador (MVC). Si bien todas las interfaces desarrolladas son componentes en sí. Se desarrollaron 4 que son reutilizables en toda la aplicación. Estas son:

- Main Container Component: funciona como contenedor de interfaz principal y es utilizado para darle una estructura estándar a todas las interfaces. La función de este componente es aplicar la misma estructura HTML en todas las interfaces implementando una barra con el título de la interfaz, y un espacio para el contenido, además, aplica un margen entre secciones, y calcula automáticamente la altura para adaptarse al tamaño del monitor o de la ventana del usuario.

Figura 4. Interfaz Lista de Usuarios con resaltado de secciones.



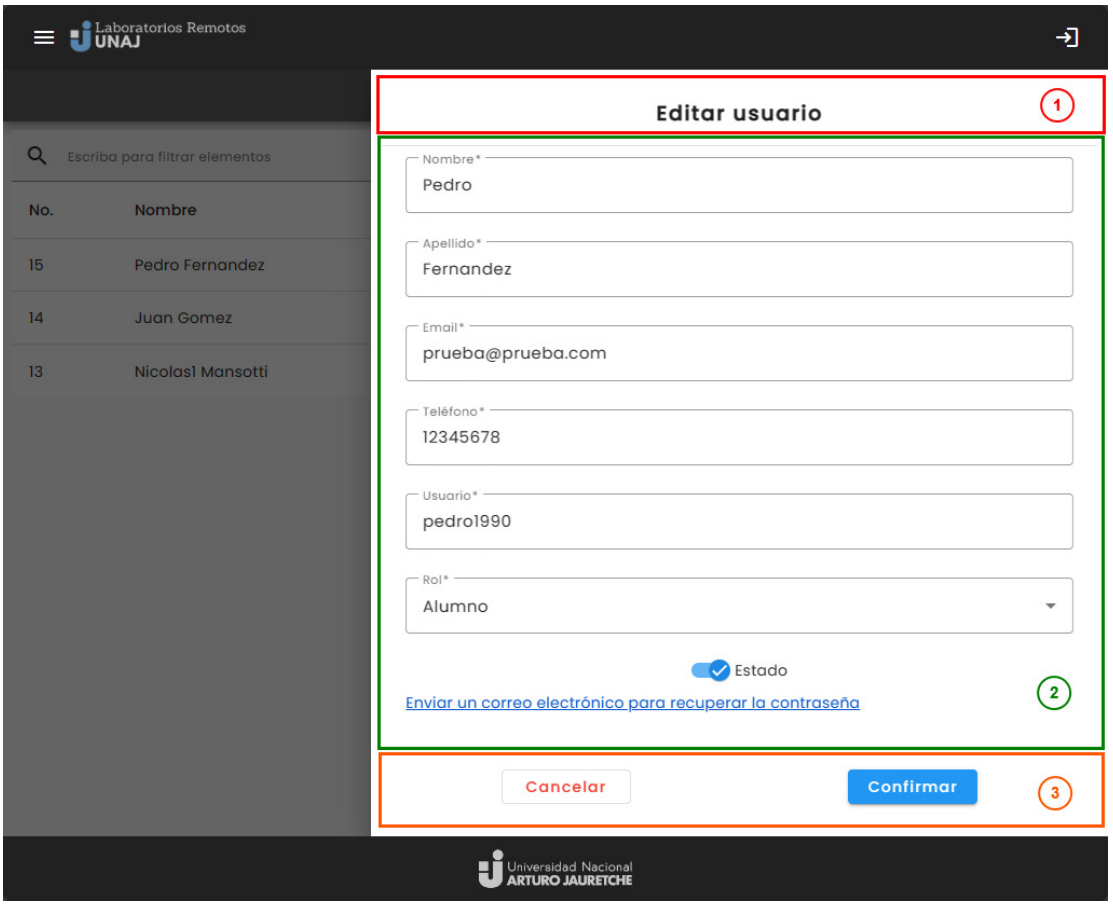
Nota. La sección 1 demarca el contenedor para el título, y la sección 2 el contenido.

- Crud Container Component: funciona como contenedor de operaciones CRUD "Create, Read, Update, Delete" (Crear, Leer, Actualizar y Borrar). Es utilizado para darle una estructura estándar a interfaces de este tipo e incluye el título de interfaz, un contenedor para el contenido y un espacio para el pie de la página. Con esto se asegura que las implementaciones respeten el mismo formato.

  Algunas características que tiene este componente son:

- Utiliza la funcionalidad de *Inputs* que permite recibir información de la interfaz que lo implementa.
- Re calcula dinámicamente la altura del contenedor. Esto permite que al cambiar la resolución o el tamaño de la pantalla se adapte correctamente.
- Utiliza la funcionalidad *NgContent* y que permite inyectar contenido dinámico. Esto es necesario debido a que, todas las implementaciones tienen el contenido diferente. El objetivo de este componente es brindar la estructura del contenedor, pero el contenido es manejado por la interfaz que lo implementa.

Figura 5. Interfaz de edición de usuario con resaltado de secciones.

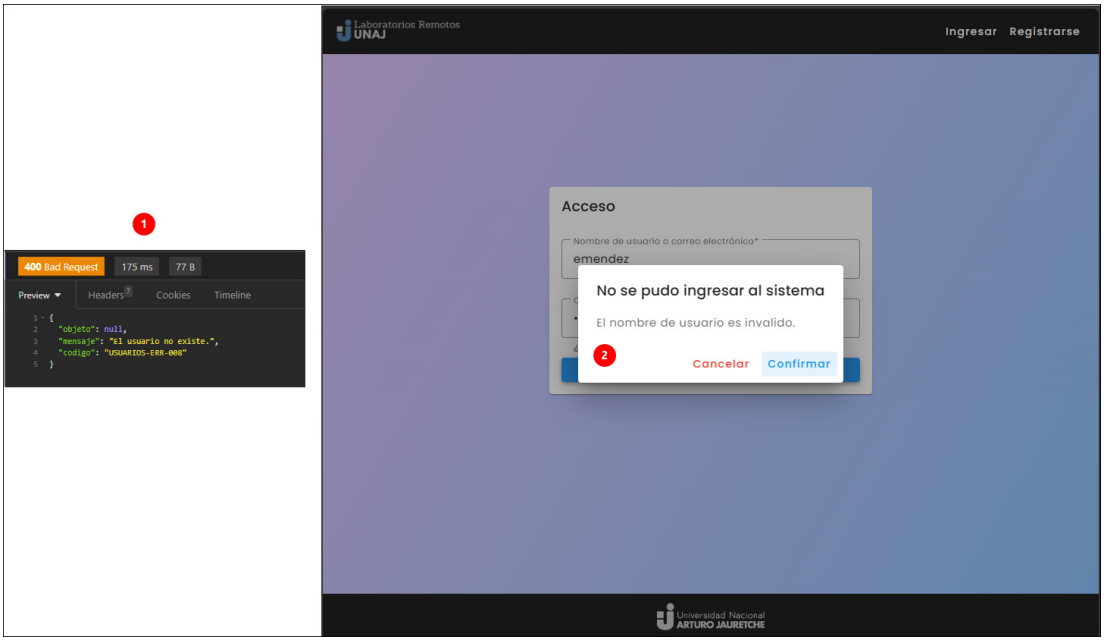


Nota: La sección 1 demarca el contenedor para el título, la sección 2 el contenido y 3 el pié de página.

- Error Component: este componente desempeña un papel importante en la usabilidad de la aplicación y experiencia de usuario. Su función principal es traducir los mensajes de errores informados por la capa de servidor y presentarlos al usuario de una manera amigable y legible. Funciona como un punto centralizado de errores mediante la herramienta *Interceptor* de Angular, la cual actúa como *middleware* entre

el cliente y servidor interceptando las respuestas HTTP y traduciendo códigos de error a mensajes con una sintaxis que sea entendible para el usuario en diferentes idiomas y que permita informar sobre cualquier problema surgido durante la interacción con la aplicación.

Figura 6. Ejemplo de traducción de errores con el componente Error Component.

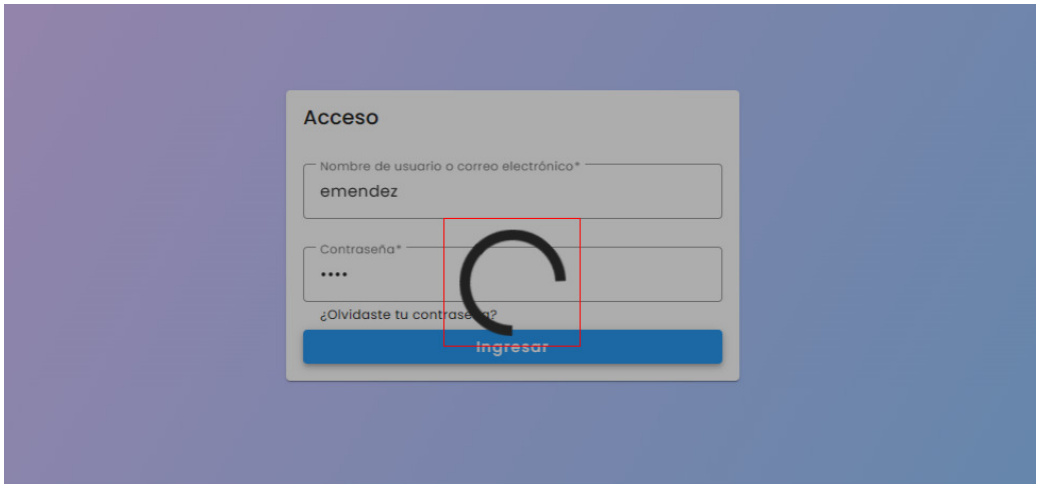


Nota. Un error con código "USUARIOS-ERR-008" es capturado, procesado y traducido.

- Busy Component: desempeña un papel fundamental en la experiencia de uso al gestionar visualmente el estado de las operaciones del usuario cuando realiza una acción que puede demorar, tales como consultas de datos grandes o acciones cuando

la conexión es lenta. Este componente muestra un *spinner* (componente de la librería de Angular Material) cuando una solicitud HTTP se encuentra en curso y lo hace automáticamente utilizando un interceptor personalizado que captura todas las solicitudes a la API solamente, bloqueando al usuario para que no pueda realizar otra acción hasta que no se haya completado. Además de darle la percepción al usuario de que su tarea se está ejecutando, también mejora la apreciación de la velocidad y respuesta de la aplicación.

Figura 7. Ejemplo de Busy Component en la carga en interfaz de acceso.



Módulos

Un *Module* (Módulo) en Angular es un mecanismo de organización que agrupa componentes, directivas, servicios y otros artefactos relacionados para formar una unidad

funcional de la aplicación para dividirla en partes más pequeñas y manejables, lo que facilita la comprensión, el mantenimiento y la reutilización del código.

Las interfaces pueden importar módulos para incluir todo el contenido y funcionalidad que estos agrupan. En base a esto, se desarrollaron 4 módulos como parte del núcleo de la aplicación, estos son:

- Auth Module: este módulo agrupa las funcionalidades necesarias para restringir a los usuarios a las interfaces que les correspondan según el rol que tenga y funciona como un mecanismo de seguridad de acceso. Los roles que tienen los usuarios se verán en la sección 2.4.5.6.1.
- Material Module: este módulo agrupa todos los componentes de Angular Material comúnmente más utilizados en la aplicación. Se utiliza para evitar importar en cada interfaz los componentes más comunes y dejar solo aquellos que son específicos.
- UI Library Module: este módulo agrupa los componentes que son utilizados principalmente en las interfaces de usuarios y se importan solamente en aquellas donde se utilizan. Al importar este módulo se incorporan los componentes detallados en la sección 2.4.4.5.1:
    - Main Container Component
    - Crud Container Component
- Base Module: este módulo agrupa los componentes básicos que son utilizados en la aplicación y se importa a nivel general para incorporar los siguientes componentes detallados en la sección 2.4.4.5.1:

- Busy Component
- Error Component

Enrutamiento

En Angular, *routing* (enrutamiento) se refiere a la capacidad de la aplicación para navegar entre diferentes vistas o componentes de manera organizada y estructurada. El enrutamiento permite que una aplicación de una sola página (SPA) tenga diferentes estados o secciones que se pueden cambiar sin necesidad de recargar la página completa. Esto mejora la experiencia del usuario al proporcionar una navegación fluida y una estructura clara en la aplicación.

Se agregaron todos los accesos a las interfaces desarrolladas al enrutamiento de Angular y adicionalmente, se agregó a cada una de ellas una capa de seguridad que evita el acceso no deseado a interfaces que no corresponden al rol que tiene el usuario. Esto quiere decir que si el usuario intenta escribir la URL manual que accede a una interfaz sin el permiso necesario, el enrutamiento lo interceptará y redirigirá a la página principal.

2.4.5.6. Interfaces de usuarios

Esta sección aborda en detalle la creación y diseño de las UI “User Interfaces” (Interfaces de Usuario). La interfaz de usuario es un elemento esencial que conecta directamente a los usuarios con la funcionalidad de la aplicación, por lo que su desarrollo y presentación fueron aspectos críticos para lograr una experiencia positiva y eficiente. Para no redundar sobre la estructura implementada en cada interfaz, todas utilizaron componentes de

la librería de Angular Material, en conjunto con una estructura HTML personalizada, clases de Bootstrap para lograr que sean adaptables a móviles y de una hoja de estilo propia. Luego en el detalle de cada interfaz se comentará aquellas que fueron más específicas en cada implementación.

Es importante destacar que existen 2 tipos de usuarios que consumen la aplicación y por este motivo, los accesos a las interfaces se restringieron acorde a las necesidades de cada uno. A fines prácticos se utilizará la denominación “usuario” para referencia a cualquier de estos.

- Estudiantes: pueden registrarse en la aplicación y utilizar las funcionalidades de solicitud, cancelación de turnos, y el acceso a los laboratorios remotos.
- Administradores: no pueden registrarse en la aplicación y solo tienen acceso mediante la carga manual de usuarios. Tienen permisos para administrar laboratorios, usuarios y cancelar turnos.

Registración

La interfaz “Registración” tiene un objetivo principal que es permitir a los estudiantes registrarse en la aplicación sin la intervención de ningún tercero. Para lograr tal efecto fue importante diseñar un procedimiento sencillo e intuitivo.

Figura 8. Interfaz de registración de usuario.

Las características más relevantes implementadas fueron:

- Directivas personalizadas: una directiva es una función con una entrada y una salida que se aplica a un input de HTML y permite dotar de funcionalidad adicional al mismo. Para esta interfaz y en general para ser compartido con otras que veremos luego, se desarrollaron las siguientes:
    - AlphaNumeric Directive: agrega un control adicional al input HTML que no permite escribir caracteres de teclado que no sean números ni letras.
    - Alphabetic Directive: agrega un control adicional al input HTML que solo permite escribir caracteres del teclado que sean letras.
- Validaciones de:

- Pasos incompletos que inhabilitan al estudiante a continuar hasta no haber cargado los datos requeridos.
- Cantidad de caracteres permitidos por campo.
- Formato de email para evitar una registración incompleta (que al estudiante nunca le llegue el email).
- Cantidad, tipo de caracteres y doble entrada para la contraseña.
- Implementación del componente *Stepper* de Angular Material que permite dividir un procedimiento en pasos. Esto permite que la primera vista que vea el estudiante sean los datos más relevantes y luego el proceso lo va guiando hasta el final, esto evitó la sobrecarga de información inicial.
- Mensaje final para indicar al estudiante que su registración se encuentra pendiente de confirmación, y que debe validarlo a través de su email.

- Responsividad: para lograr la responsividad en esta interfaz se utilizó la librería *RxJS*. Esta librería contiene herramientas que permiten escuchar eventos en tiempo real y cambiar variables. Para este caso, el componente *Stepper* de Angular Material ya disponía de una configuración para verse en forma vertical que se adapta mejor a móvil sin embargo la complejidad estaba dada en decidir cuando lo es en tiempo real, por ejemplo, al modificar el tamaño de la ventana del navegador.

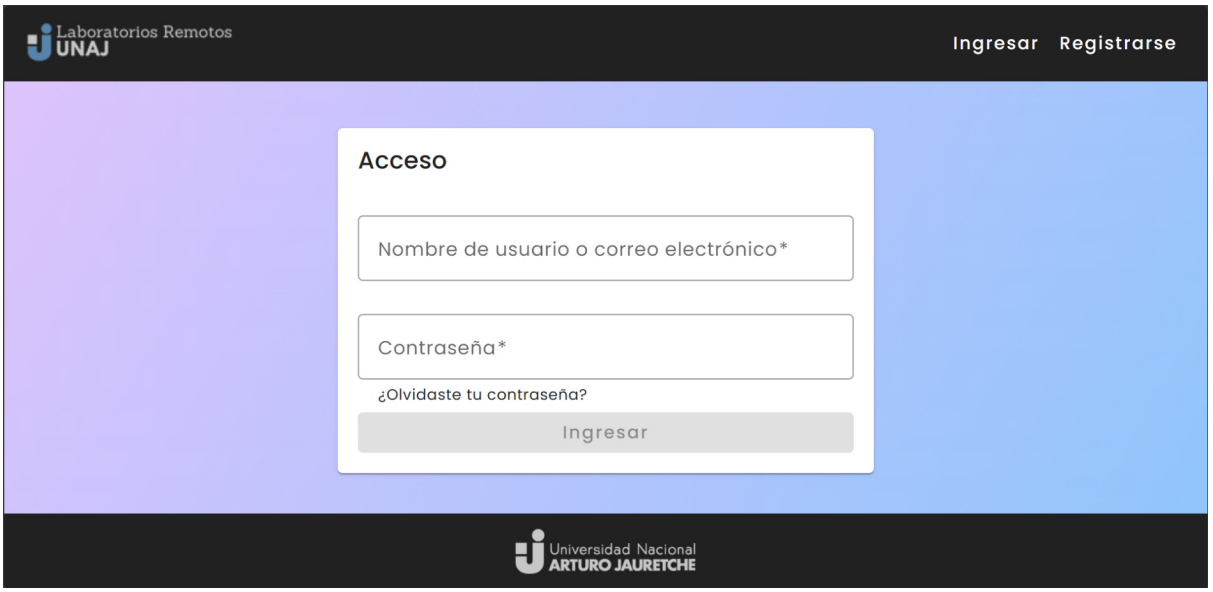
  Para solucionar esto se utilizó un *pipe* (tubería de RxJS) que escucha eventos que cumplen cierta condición y permiten implementar lógica adicional. La condición fue, cuando la resolución de la pantalla llegará hasta cierto punto, entonces cambiar la

configuración del *Stepper* a un formato vertical. De esta manera, la interfaz de registración de estudiantes quedó totalmente responsiva.

Acceso

La interfaz "Acceso" consiste en un formulario interactivo que solicitará al usuario su información de inicio de sesión, como nombre de usuario o email y contraseña.

Figura 9. Interfaz de acceso.



Las características más relevantes implementadas fueron:

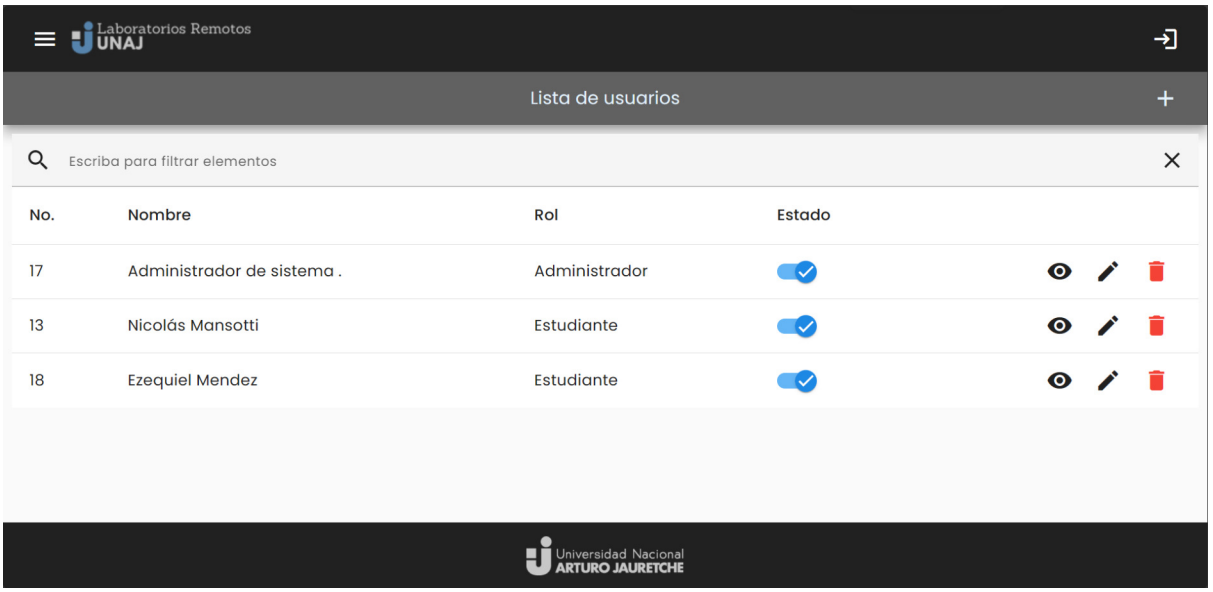
- Campo de doble entrada: permite al usuario ingresar por nombre de usuario o email. Esta característica mejora la usabilidad de la aplicación al agregar una opción más al usuario de acceso en caso de que olvide alguna de ellas.

- Deshabilitación de acceso dinámico: mejora la performance de la aplicación al evitar realizar peticiones al servidor de manera innecesaria cuando el usuario aún no cargó los datos requeridos.
- Persistencia de acceso: se implementó el guardado de la sesión del usuario post validación de acceso en la caché del navegador por el tiempo de 1hr. Para lograr esto se utilizaron las funciones nativas del navegador "localStorage" que permiten persistir información en una memoria interna del navegador que se mantiene aun cuando se cierra el navegador y solo se elimina al limpiar la caché o cumplirse el tiempo de guardado. Esta característica mejora la navegabilidad de la aplicación al permitir mantener la sesión del usuario abierta y adiciona una capa de seguridad extra al implementar al vencimiento.
- Recuperación de contraseña: se implementó una función de recuperación de contraseña que se activa al seleccionar la opción correspondiente en la interfaz de acceso. Al utilizar esta función, se solicita al usuario que proporcione un email para enviar un nuevo enlace de acceso que permitirá cambiar la contraseña. Si el email corresponde a un usuario registrado, la API enviará un enlace seguro al email del usuario, proporcionando así una nueva interfaz para la creación de una nueva contraseña. Esta característica permite a los usuarios gestionar de forma autónoma sus credenciales de acceso sin necesidad de intervención externa.

Lista de Usuarios

La interfaz de “Lista de Usuarios” es una sección accesible únicamente por los administradores cuyo objetivo es brindarle funcionalidades para ver, editar, borrar o agregar usuarios. Esta sección se diseñó con el objetivo de darle al personal de la universidad, el control y mantenimiento eficiente de los usuarios registrados en el sistema. Los usuarios administradores tienen la capacidad de deshabilitar a otro usuario en caso de que no tenga permitido utilizar el sistema por cualquier motivo, editar los datos personales en caso de que algún estudiante haya cargado mal sus datos personales, recuperar la contraseña en caso de que el estudiante no supiera cómo hacerlo o tuviera algún tipo de inconveniente, agregar nuevos usuarios administradores y estudiantes. Además cuenta con un filtro rápido para búsquedas y la opción de deshabilitar directamente a través de la grilla.

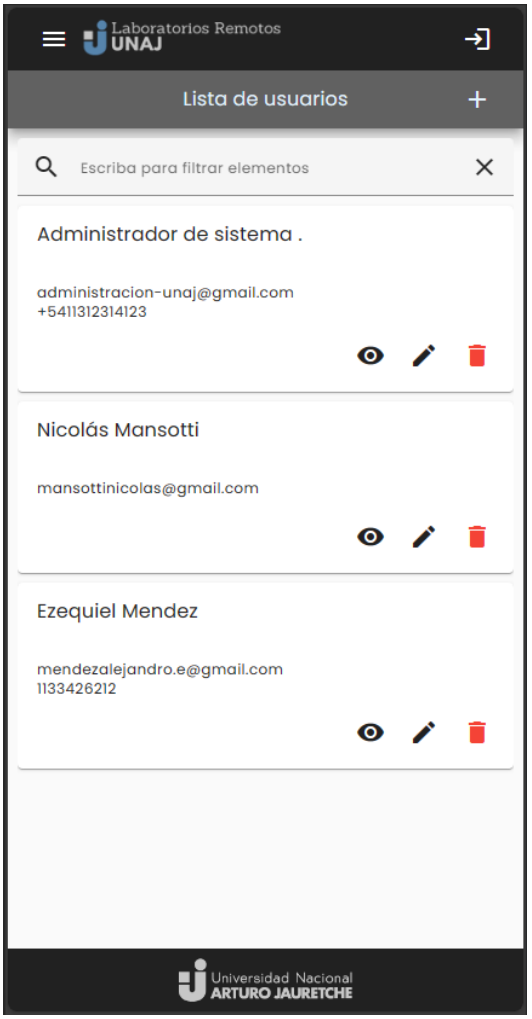
Figura 10. Interfaz Lista de Usuarios.



Algunas de las características destacadas en esta interfaz son:

- Utilización del componente Crud Container Component y Main Container Component que se utiliza para mantener la misma estructura que otras interfaces de usuario.
- Implementación del componente *grid*. Este es un componente incluido en la librería de Angular Material que renderiza una grilla con la información recibida, en este caso, una lista de usuarios.
- Implementación de *RxJS* para cambiar de manera dinámica la presentación de información según la resolución de la pantalla. Esta implementación aprovecha la funcionalidad de RxJS para gestionar la reactividad de los eventos y complementarlos con los componentes de Angular Material para adaptar y cambiar la vista de grilla por tarjetas, optimizando la experiencia del usuario.

Figura 11. Interfaz Lista de Usuarios en versión móvil.



Lista de Laboratorios

La interfaz “Lista de Laboratorios”, también se encuentra restringida únicamente para usuarios administradores y permite dar de alta, modificar, editar o borrar laboratorios. Uno de los objetivos principales de este proyecto era digitalizar la información de los mismos y darle la posibilidad al personal de la universidad para que pudieran gestionarlos.

Figura 12. Interfaz Lista de Laboratorios.

Cada registro representa un laboratorio remoto de la universidad y están identificados por un nombre, título y descripción, sin embargo tienen características que son importantes en el circuito de solicitud de turnos, estos son:

- URL: cada laboratorio remoto se encuentra hospedado en un servidor y debe tener un acceso público mediante una URL "Uniform Resource Locator" (Localizador Uniforme de Recursos). Esto es necesario para que un estudiante pueda acceder y por este motivo, es un dato obligatorio para que el laboratorio pueda estar disponible.
- Disponible: es el estado del laboratorio que puede ser verdadero o falso. Un laboratorio puede existir como un registro en la aplicación pero no disponible por distintos motivos como, no tener un hosting, no tener acceso a internet, estar en reparación y en general cualquier otro que no permita acceder de manera normal.

- Configuración: es una de las características más importantes de esta interfaz que permite configurar la disponibilidad de un laboratorio y sus turnos.

  Esta configuración permite:

  - Modificar la duración de los turnos desde 15 minutos hasta 2 horas.
  - Restringir el acceso entre un rango de horarios.
  - Restringir los días de funcionamiento.

Figura 13. Interfaz de configuración de laboratorio.

Al igual que la interfaz Lista de Usuarios, esta aplica los mismos componentes estructurales mencionados allí, además de la lógica de responsividad y operaciones sobre los registros.

Lista de Turnos

La interfaz "Lista de Turnos" es otra de aquellas a las solo pueden ser accedidas por administradores de sistema y se diseñó con el objetivo de servir como soporte para los administradores en caso de que los estudiantes por motivos específicos no pudieran cancelar turnos. Esta interfaz permite cancelar turnos de cualquier estudiante y cualquier laboratorio. Además dispone de filtros avanzados que permiten realizar una búsqueda más específica por laboratorio, usuario, fechas o disponibilidad.

Figura 14. Interfaz Lista de Turnos.

Algunas características de esta interfaz son:

Firma Estudiante:	Firma Docente Supervisor:	Firma Docente tutor TAPTA:	Firma Tutor organizacional:

- Utilización del componente *MatDialog* de Angular Material que permite visualizar un *popup* (ventana emergente) y renderiza un formulario con los criterios de búsqueda. Este componente es útil para la usabilidad de la aplicación porque permite ver más opciones en pantalla sin salirse del contexto.
- Grilla con disponibilidad del turno en forma de icono que le da al usuario una vista rápida sobre en qué estado se encuentra. También dispone de un botón en forma de icono para cancelar los turnos que solamente se activa cuando el mismo está vigente. Adicionalmente, esta grilla implementa un paginador debido a que la cantidad de turnos para N cantidad de usuarios y laboratorios puede ser excesiva, esto permite mostrar N cantidad de registros en pantalla, desde 15 hasta 100.

Mis turnos

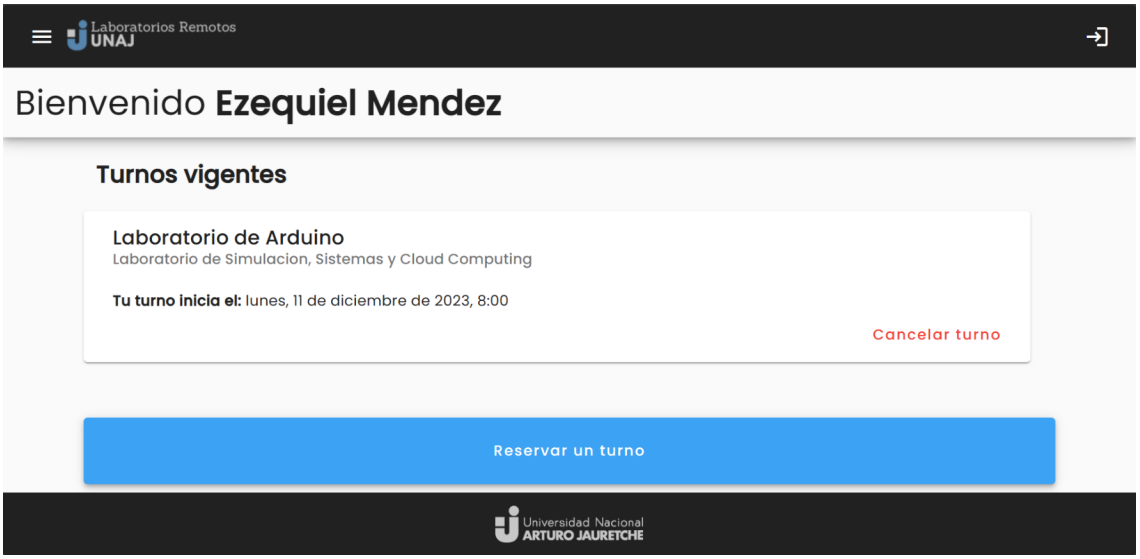
La interfaz "Mis Turnos" es solamente accesible para estudiantes y tiene como objetivo mostrarles sus turnos vigentes, darles acceso a los laboratorios remotos, y permitirles cancelar y solicitar turnos. Para lograr el objetivo se diseñó una pantalla con tipografías, y recursos gráficos más grande que el estándar manejado de manera tal que la información presentada y el flujo desde la solicitud del turno hasta la utilización del mismo sea intuitivo.

- Vista de bienvenida

  Luego de que un estudiante ingrese al sistema, se lo envía automáticamente a esta interfaz que le da una bienvenida y les muestra los turnos vigentes, si los tuviera. Esta

primera vista muestra la información al usuario en una tarjeta con la especificación del laboratorio remoto al cual se solicitó el turno, la fecha y hora del mismo y si es el momento del turno se habilita un botón que envía al usuario a la web del laboratorio remoto.

Figura 15. Vista de bienvenida en la interfaz Mis Turnos.



Para esta vista se destacan las siguientes implementaciones:

- Se implementó el componente *MatCard* (tarjeta) de Angular Material para representar la información de un turno. Este componente destaca el título con el nombre del general del laboratorio, subtítulo con el nombre específico del laboratorio y la fecha/hora del turno. Con esta implementación se aprovecha la responsividad del componente para su adaptación a móvil sin necesidad de código extra, y además por que la cantidad de elementos en pantalla es poca y

en estas condiciones, una tarjeta resalta mejor la información que una grilla. También dispone de un botón para cancelar el turno o "Ir a laboratorio" que envía al estudiante a la web del laboratorio remoto que se habilita únicamente cuando la fecha y hora actual coincide con la de turno, es mayor y no superó la duración con respecto al horario actual.

- Se desarrolló una directiva personalizada de Angular para dotar de funcionalidad a la tarjeta con la información del turno y lograr que el botón "Ir a laboratorio" se muestre en tiempo real. Aquí se aprovechó la funcionalidad reactiva de los componentes de Angular para darle al usuario una mayor usabilidad y que no sea necesario refrescar la pantalla para saber si su turno ya se encuentra listo o no.
- Implementación del componente *MatSnackbar* de Angular Material para mostrar un mensaje efímero e indicar al usuario que la operación se realizó correctamente.

- <u>Solicitud de turno</u>

  En cuanto a la funcionalidad de solicitud de turno, se diseñó un proceso de manera tal que los estudiantes pudieran ir cargando paso a paso la información requerida para solicitar un turno. Una de las características generales de esta interfaz es la implementación de *MatStepper* de Angular Material, el cual contiene una estructura que permite dividir vistas en pasos. Angular permite modificar sus componente y

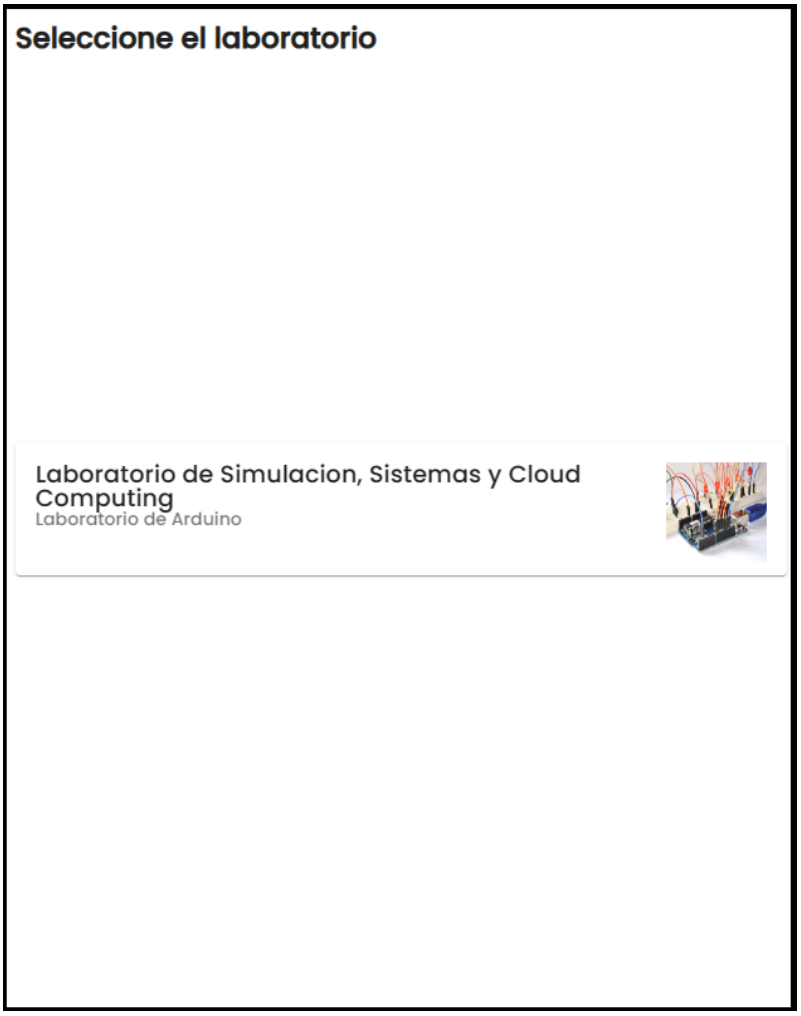
extenderlos para agregar funcionalidad a través de una sección de su librería llamada CDK “Component Dev Kit” (Kit de desarrollo de componentes), este ofrece funciones primitivas para crear componentes personalizados manteniendo las mismas funcionalidades que el original, como las animaciones y los parámetros de entrada. Utilizando CDK se desarrolló el componente TurnoStepper que además de la funcionalidad del original, elimina detalles gráficos como la numeración por pasos y línea de seguimiento para implementar un mensaje claro que indique en qué etapa del proceso se encuentra el usuario. A partir de esta implementación, se definieron 3 pasos para completar la solicitud y se separaron en componentes individuales:

1. Selección de laboratorio
2. Selección de horario
3. Confirmación de turno

1. Vista Selección de laboratorio

Esta interfaz muestra el primer paso para el usuario donde debe elegir, para cual laboratorio desea reservar un turno y mediante un filtro de consulta a la API se obtienen y muestra aquellos que estén disponibles solamente.

Figura 16. Vista de selección de laboratorio.



2. Vista Selección de horario

El segundo paso para el usuario que solicita el turno, luego de elegir el laboratorio, es la selección de un horario para el uso del laboratorio.

Una característica importante de esta vista es que, la disponibilidad de días y horarios se encuentra sujeta a la configuración del laboratorio. Aquí es posible elegir el día de uso del laboratorio, pero las opciones pueden estar restringidas según el laboratorio seleccionado. De la misma forma con los horarios pero adicionalmente, estos también

son filtrados según el momento actual. Esto quiere decir que los horarios disponibles están restringidos por:

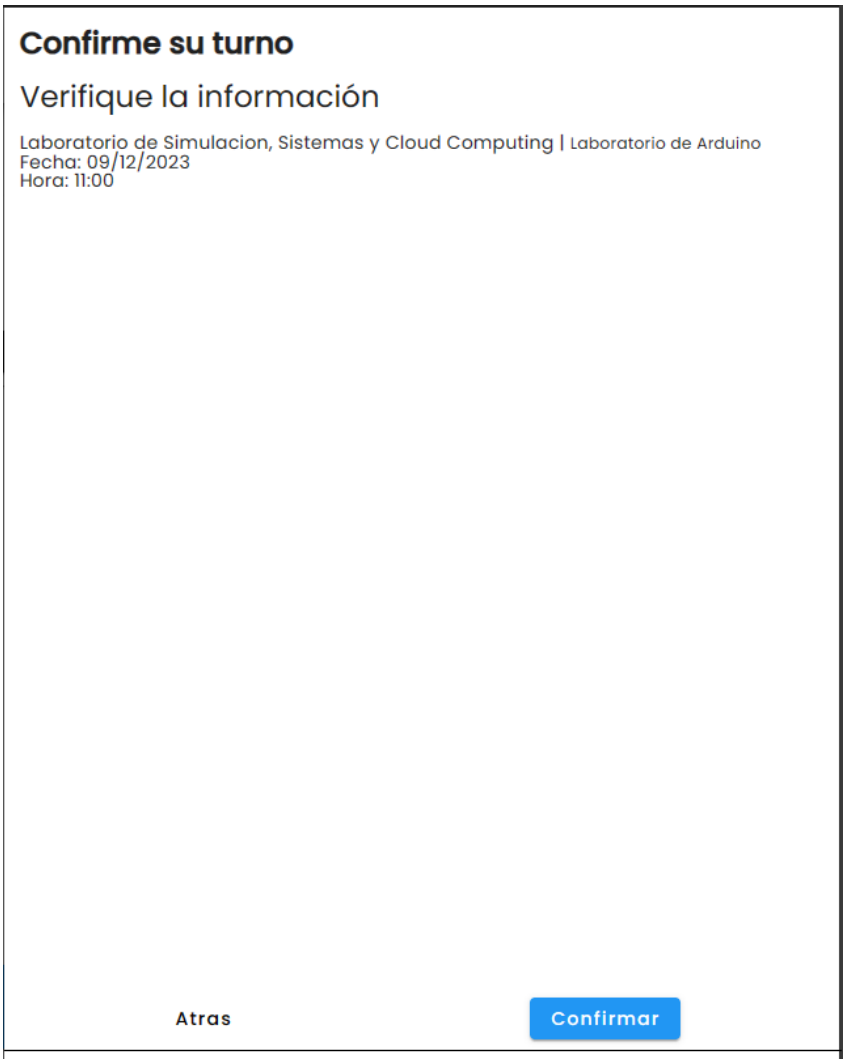
- La hora actual de la PC, permitiendo ver aquellos que son mayores a la actual solamente.
- La configuración del laboratorio indica desde cuándo y hasta qué hora el laboratorio permite reservar turnos.
- Horarios seleccionados por otros usuarios que se muestran bloqueados para que el usuario entienda que no están disponibles por estar ya reservados y no por un error.
- El cálculo realizado por la API en base a la duración del turno configurado para el laboratorio seleccionado. Esto quiere decir que, si un laboratorio tiene configurado turnos de 15 minutos, a partir de un hora puntual se tendrá turnos disponibles a las hh:15, hh:30, hh:45, hh:00 siendo hh la hora actual.

Figura 17. Vista de selección de horario.

3. Vista Confirmación de turno

Esta es una vista para darle al usuario la claridad de que ha completado el proceso y que solo falta la confirmación. Al hacerlo, se envía un email al usuario con la confirmación del turno a través de la API y se muestra en la interfaz de bienvenida el turno solicitado.

Figura 18. Vista de confirmación de turno.



Perfil

La interfaz “Perfil” es para todo tipo de usuarios y el objetivo es darle la posibilidad de personalizar algunas cuestiones de usabilidad para que se adapten a su comodidad como, el cambio de idioma o de tema (colores). Con esta interfaz se ayudó a cumplir el objetivo de

implementar una aplicación que sea accesible por el mayor público posible y pensando en el futuro para usuarios de otras universidades.

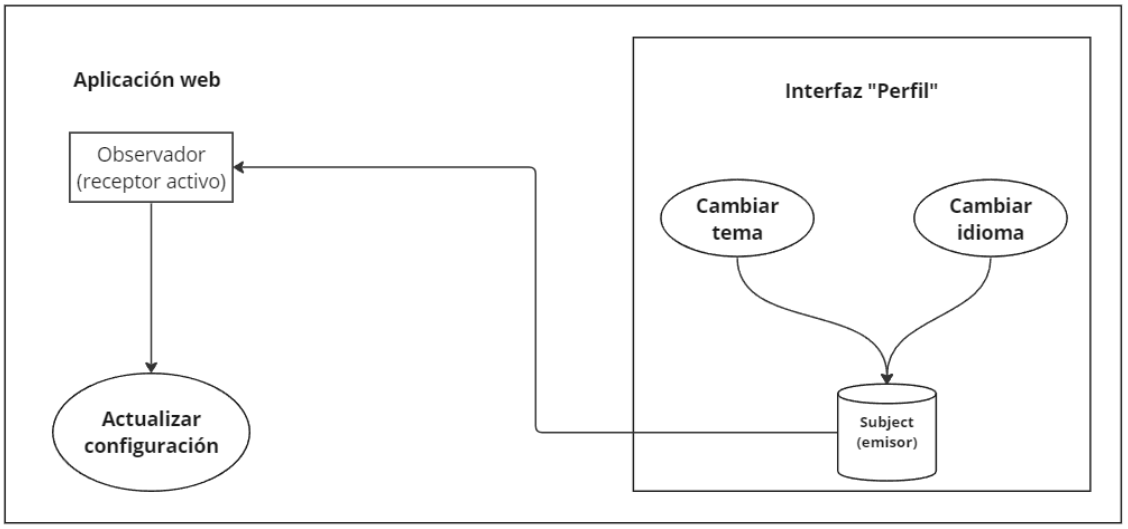
Figura 19. Interfaz Perfil.

Algunas características destacadas en esta interfaz son:

- La implementación de *RxJS* para aplicar los cambios en tiempo real sin necesidad de recargar la pantalla. Esto se logró utilizando la mecánica de Observables que tiene la librería. Un observable es una herramienta que sigue el patrón *Observer* de programación donde una entidad emite información y los consumidores la reciben. Todo esto sucede de forma asincrónica utilizando RxJS y es fácilmente utilizado por la librería con sus objetos *observables*. Esto permitió que al cargarse la aplicación se mantenga un observador escuchando eventos de la configuración de perfil, y en cuanto al emisor que se encuentra en dicha interfaz, emite un valor cuando existe una

modificación y es tomada en tiempo real por el observador que procesa la petición y actualiza por ejemplo, el cambio de colores en el caso del tema, o de archivo de traducción en caso de ser la configuración de idioma.

Figura 20. Flujo de eventos entre la interfaz Perfil y la aplicación utilizando observables.



- La implementación de la librería Ngx-Translate para la internalización de la aplicación. Esta librería de Angular permite hacer una gestión eficiente de los recursos textuales ya que brinda una capa de abstracción que separa el código de los recursos, permitiendo manejar los textos de manera centralizada en un archivo por idioma y además, con herramientas útiles para accederlos de manera directa a través de la lógica de negocio o de presentación reutilizando los mismos sin necesidad de

duplicarlos.

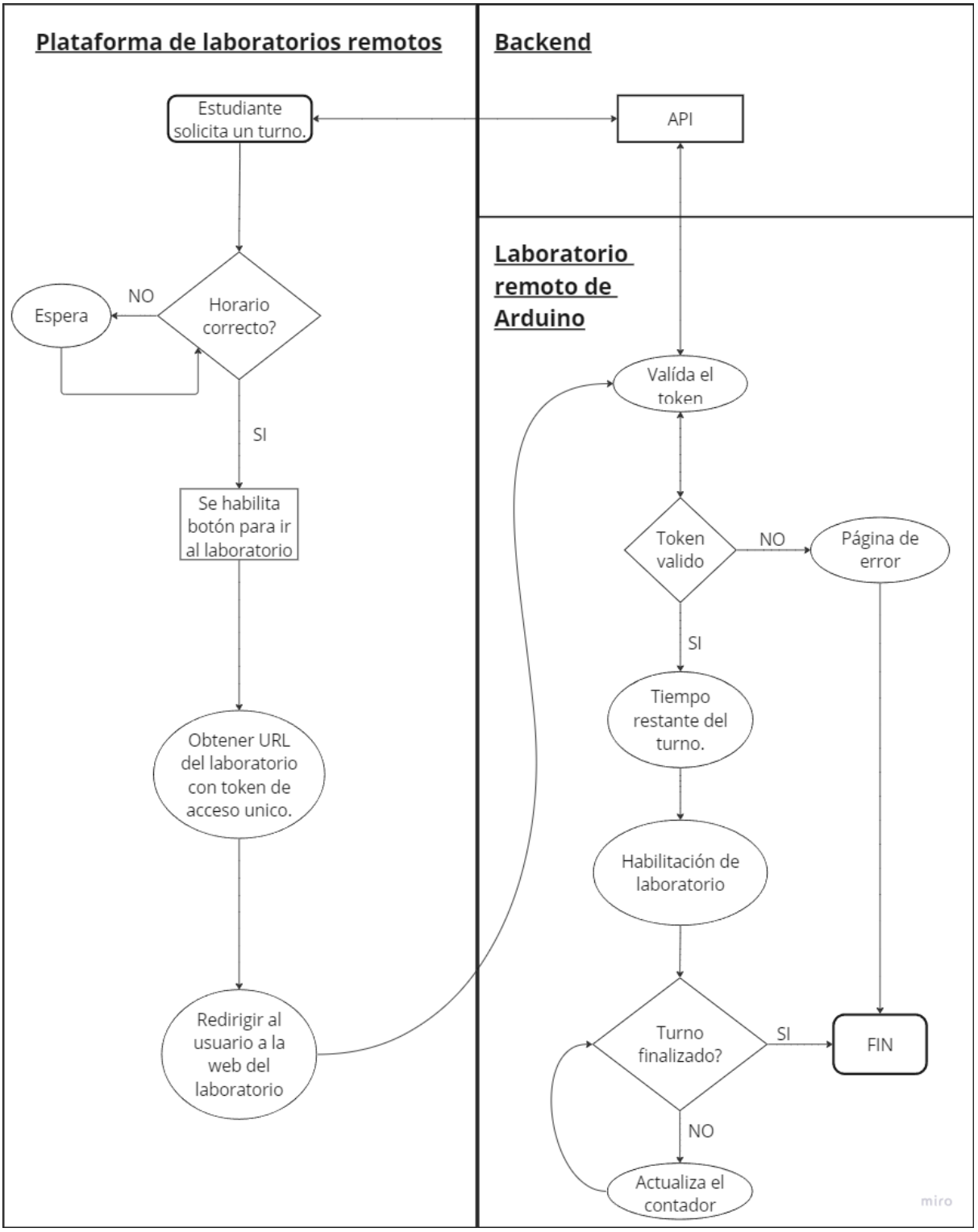
2.4.6. Adaptación de laboratorio remoto de Arduino

Uno de los requisitos de esta PPS fué no solo diseñar y desarrollar la plataforma para poder acceder a los laboratorios remotos, sino también modificar el laboratorio remoto de Arduino (actualmente en funcionamiento) y ajustarlo para que se integre con la nueva plataforma. Esto implicó eliminar partes que ya existían dentro de la aplicación del laboratorio como, el control de usuarios y de turnos. Y adaptar la lógica de la aplicación para que funcione con la nueva API.

El laboratorio remoto de Arduino fue desarrollado por otro estudiante con las tecnologías *PHP*, *HTML*, *CSS* y *Javascript*. Esta aplicación además de las funcionalidades del laboratorio, permitía a los usuarios registrarse y solicitar turnos pero requería hacerlo desde la *PC* físicamente instalada en las aulas de la universidad. A partir del desarrollo de la nueva plataforma, el control de usuarios y turnos nativo del laboratorio quedó depreciado con lo cual se eliminó y se reemplazó por un llamado a la *API* que valide el acceso del usuario.

Para entender el resultado final de esta modificación y la interoperabilidad entre la plataforma y los laboratorios remotos se grafica el siguiente flujo.

Figura 21. Diagrama de flujo de interoperabilidad entre plataforma y laboratorio remoto.



Firma Estudiante:	Firma Docente Supervisor:	Firma Docente tutor TAPTA:	Firma Tutor organizacional:

La validación puede dar distintos resultados y en base a cada uno, se implementaron distintas paginas de *PHP* que muestran al usuario un mensaje personalizado.

Figura 22. Interfaz del laboratorio remoto luego de la validación de acceso correcta.

Figura 23. Interfaz del laboratorio remoto luego de finalizar el turno.

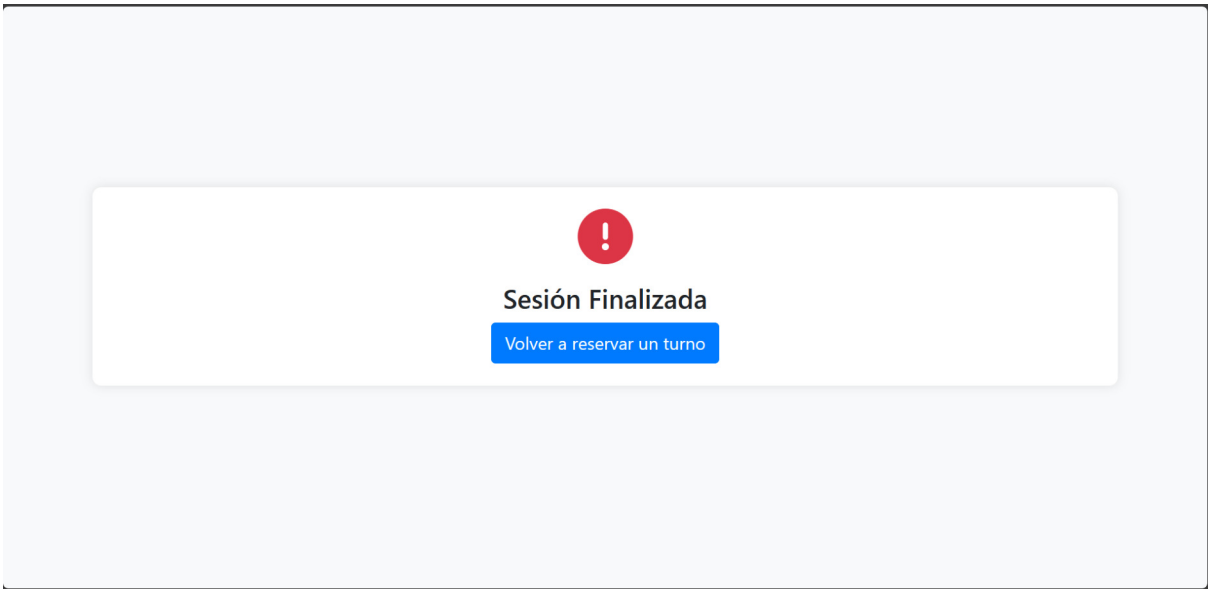
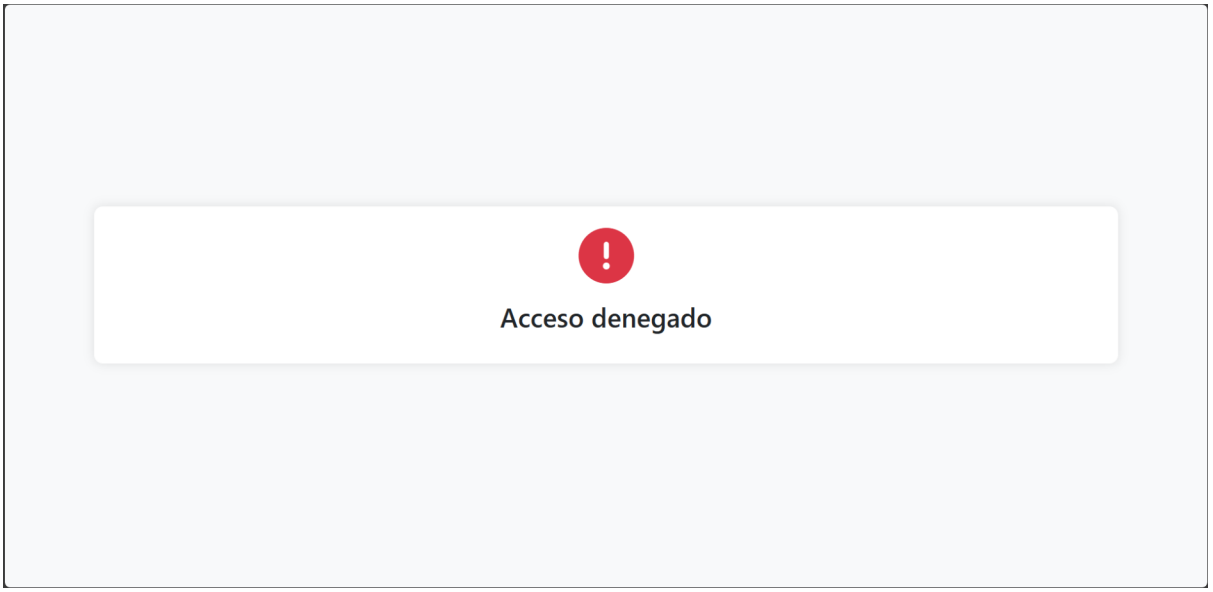


Figura 24. Interfaz del laboratorio remoto con un token invalido.



2.4.7. Implementación

La etapa de implementación implica empaquetar y desplegar la aplicación desarrollada a un ambiente productivo para su posterior utilización luego de haber pasado la etapa de pruebas y de aprobación por parte del *product owner*.

Empaquetado

El empaquetado es el proceso que implica la preparación y organización de los archivos y recursos de la aplicación con el fin de obtener un artefacto listo para su distribución. Para lograr esto se utilizó Docker como herramienta el cual permite desplegar la aplicación en un entorno aislado y portable que facilita la implementación y distribución en diferentes entornos sin preocuparse por las diferencias en las configuraciones de sistemas operativos, a continuación se describen los pasos realizados a fin de entender cuáles fueron las tareas realizadas:

1. Selección de Imagen de Docker: una imagen en Docker es un paquete ligero y portátil que contiene la infraestructura mínima y necesaria para desplegar una aplicación. Se utilizó para este proyecto un imagen de Nginx la cual provee un contenedor que actúa como servidor de web para servir archivos estáticos y dado que la compilación de Angular genera un conjunto de estos archivos, la implementación de esta imagen fue una decisión muy óptima.
2. Configuración de Docker: las imágenes de Docker se configuran a través de un archivo *DockerFile* el cual contiene las instrucciones necesarias para crear una

Imagen personalizada. Para este proyecto se configuró un Dockerfile con las siguientes instrucciones:

a. Utilización de imagen Nginx como base para el contenedor.
b. Copiado de archivo de configuración de Nginx desde el contexto de la aplicación al contenedor para reemplazar la configuración por defecto.
c. Copiado de archivos compilados de Angular desde el contexto de la aplicación al contenedor, específicamente a la carpeta por defecto que utiliza Nginx para servir el contenido.
d. Exponer el puerto 80 para escuchar las peticiones HTTP en ese puerto por defecto.

3. Configuración de Nginx: es un archivo de configuración que contiene instrucciones que van desde la configuración general del servidor, como el puerto y las ubicaciones de archivos, hasta detalles específicos como reglas de redirección, configuración de seguridad, manejo de errores y definición de cómo se sirven y procesan los archivos. Se decidió crear un archivo personalizado para optimizar la carga de recursos al servir la aplicación web, agregando a la configuración estándar, la posibilidad de comprimir los archivos e implementar el guardado en caché de las imágenes. Esto optimizó el tiempo de carga de las interfaces como se verá en la sección 2.4.8.
4. Comandos personalizados: a partir de las configuraciones de Docker y Nginx fué posible generar una imagen personalizada de la aplicación utilizando los comandos de Angular CLI para compilar los archivos de la aplicación, y de Docker para crear la

imagen, sin embargo como estos comandos eran repetitivos se decidió automatizarlos en un script que incluya las tareas de construcción de paquete, imagen y publicación de la misma al repositorio de Docker. Esto permitió simplificar la tarea de desarrollo y pruebas, y preparar para el futuro un esquema de entrega continua.

Despliegue

A partir de la publicación de la imagen personalizada de la aplicación al repositorio de Docker se habilita la opción de crear contenedores. Un contenedor de Docker es un implementación de una imagen que puede ser replicada la cantidad de veces necesarias y funcionan de manera autónoma. A partir de esta definición, existen tecnologías que pueden facilitar a los usuarios la gestión de estos contenedores que son llamados “Orquestadores” y permiten gestionar la automatización, distribución, escalabilidad y balanceo entre otras funcionalidades. Para este proyecto se utilizó como orquestador el servicio *Portainer* el cual fue proveído por la universidad con una configuración inicial que facilitó el despliegue de la aplicación, requiriendo únicamente un repositorio de Docker desde donde consumir la imagen a desplegar.

Con la configuración de la imagen de Docker, y de Portainer se logró tener de manera muy sencilla un entorno productivo con la aplicación lista para ser utilizada a través de una URL accesible para las personas.

2.4.8. Pruebas

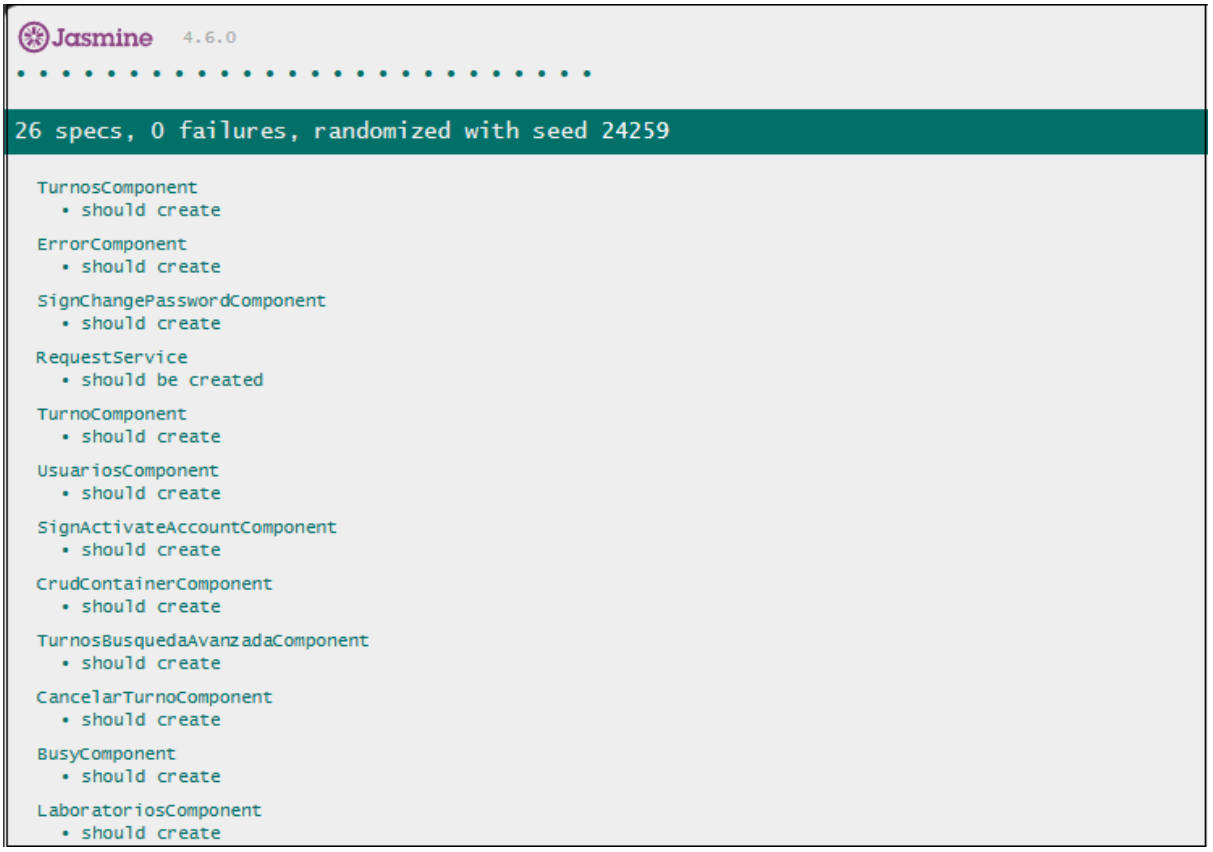
En esta sección se describirán las diferentes etapas de pruebas que se implementaron durante el proceso de desarrollo de este proyecto.

Pruebas unitarias

La primera etapa fueron las pruebas unitarias, las cuales se encargan de validar que los componentes desarrollados funcionen de manera independiente. Para esto se utilizó *Jasmine* que es un marco de pruebas de comportamiento para *JavaScript* que se utiliza para escribir y ejecutar pruebas unitarias, mientras que se utiliza *Karma* como motor que las ejecuta. Estas herramientas ya vienen instaladas por defecto en el proyecto de Angular, y cada componente desarrollado crea por defecto una configuración de prueba unitaria. Este tipo de prueba se encarga de validar en una etapa temprana si existen errores en la aplicación y son comúnmente integrados a un proceso de integración continua que valida constantemente el código que se sube a un repositorio.

Para el alcance de este proyecto, se validaron que los 26 componentes y servicios desarrollados se rendericen correctamente al invocarlos, dejando para una etapa futura la creación de casos más complejos de pruebas.

Figura 25. Ejemplo de ejecución de pruebas unitarias con Karma y Jasmine



Pruebas de rendimiento y usabilidad

La segunda etapa de pruebas, posterior a la finalización del proyecto y despliegue fue, la validación de rendimiento de la aplicación y usabilidad, utilizando la funcionalidad *Lighthouse* de los navegadores basados en *Chromium*. *Lighthouse* es una herramienta de código abierto desarrollada por Google que se utiliza para auditar y mejorar la calidad de las páginas web. Su función principal es evaluar el rendimiento, la accesibilidad, las mejores prácticas, el SEO (Search Engine Optimization) y la progresividad de una aplicación web o sitio. Realiza análisis automáticos y proporciona informes detallados con sugerencias

específicas para mejorar la velocidad de carga, la experiencia del usuario y la visibilidad en los motores de búsqueda.

En esta etapa no se incluyeron pruebas relacionadas con SEO “Search Engine Optimization” (Optimización en motores de búsqueda) debido a que esta característica no era prioridad para el proyecto, ni tampoco PWA “Progressive Web Aplication” (Aplicación Web Progresiva).

Se analizaron las características relacionadas con el rendimiento, accesibilidad y buenas prácticas utilizadas en la aplicación y se utilizaron interfaces con distintos niveles de carga de recursos como escenarios para obtener muestras reales. Los resultados cómo se observarán en las siguientes capturas fueron excelentes.

Figura 26. Resultados de la prueba de rendimiento y accesibilidad en interfaz principal.

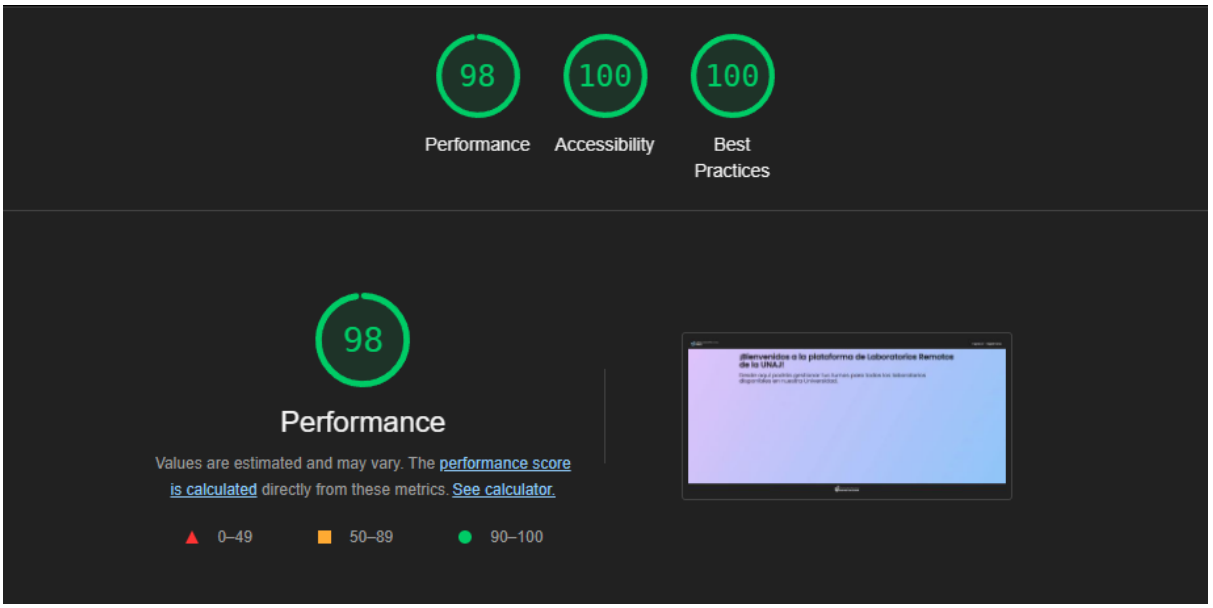


Figura 27. Resultados de la prueba de rendimiento y accesibilidad en interfaz “Mis Turnos”.

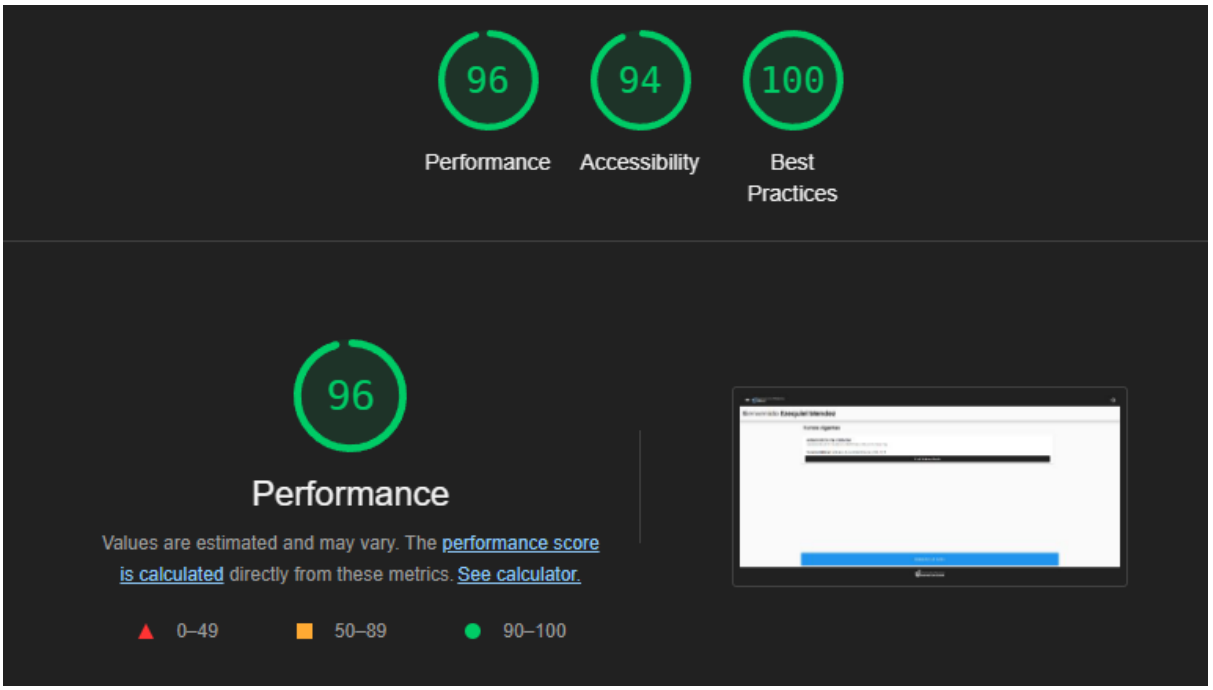
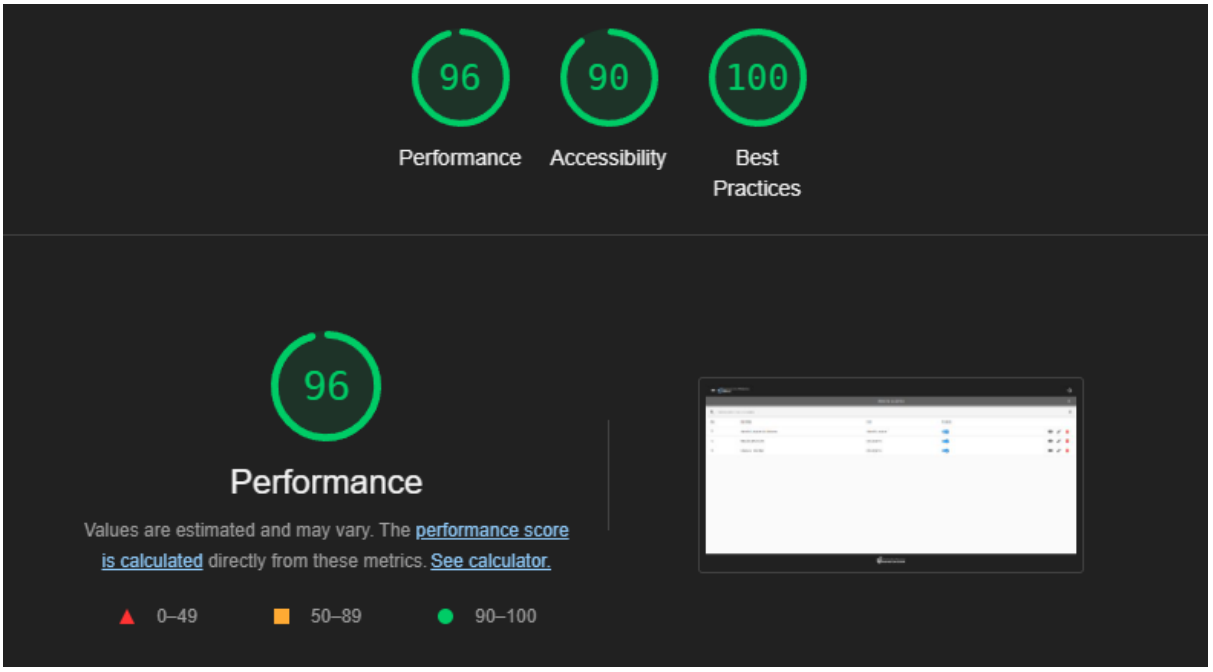


Figura 28. Resultados de la prueba de rendimiento y accesibilidad en interfaz “Lista de Usuarios”.



Pruebas E2E

Una prueba E2E (end to end) implica probar las funcionalidades de la aplicación en su totalidad para validar todos los componentes y cómo interactúan entre sí. Esta tarea se realizó en conjunto con el equipo de *backend* y los procesos validados fueron:

3. Funcionamiento de operaciones CRUD
4. Carga de datos masiva de usuarios y laboratorios
5. Interacciones entre interfaces:
    5.1. Registración, activación y validación de accesos.
    5.2. Carga de usuarios administradores y validación de acceso.
    5.3. Desactivación de usuarios y validación de restricción de acceso.

5.4. Validación de restricciones de disponibilidad de laboratorios por configuración en la solicitud del turno.

5.5. Redireccionamiento desde la plataforma al laboratorio remoto y su posterior validación de acceso.

Se destaca que durante este proceso se encontraron algunos errores que fueron corregidos posteriormente y una lista de mejoras en cada interfaz con respecto a la usabilidad de la aplicación.

6. Conclusiones

Como se mencionó en la introducción de este trabajo, el objetivo de la PPS fue desarrollar una herramienta para la universidad que permitiera dotar a los estudiantes con más recursos educativos que los acerquen a entornos reales. En el proceso de desarrollo del proyecto se destaca el plan de trabajo inicial y los análisis preliminares que permitieron tener una hoja de ruta clara y con pocas desviaciones que permitieron finalizar el proyecto en tiempo y forma. También se destaca la decisión de utilizar la mayor cantidad de herramientas existentes en la comunidad que simplificarán el trabajo, como ejemplo el caso de Angular como *framework* de *frontend* donde los resultados fueron cuantitativos y cualitativamente muy buenos, y los resultados están reflejados en los análisis de rendimiento y usabilidad mostrados en la sección 2.4.8. Así mismo, la utilización de Angular como *framework* permitió que el desarrollo se realice sin complejidades excesivas tales como diseñar un mecanismo de implementación de colores y tipografías, controles de errores, librerías de componentes, y otras funcionalidades descritas en el proceso desarrollo que permitieron

avanzar con la programación de una manera sencilla y ordenada. La implementación de librerías de terceros también fué un acierto que permitió avanzar muy rápido en el desarrollo de algunas características puntuales como, el control de traducciones, la reactividad y responsividad de la aplicación.

El resultado de este proyecto fue una aplicación con una interfaz muy sencilla de utilizar, intuitiva, responsiva, accesible y performante que consiguió solucionar cada una de las necesidades planteadas al inicio de esta PPS las cuales finalmente se pueden resumir en:

- Darle el control de los laboratorios a los administradores de la universidad.
- Darle una herramienta a los estudiantes para conectarse a los laboratorios remotos.

Como espacio de mejora se destaca la posibilidad de utilizar servicios que se encarguen del control de usuarios debido a que, una de las funcionalidades con más complejidad y tiempo de desarrollo fué el Ingreso y Registración de usuarios. Hay que destacar que ya existen servicios y herramientas que se encargan de solventar y mejorar este proceso, tales como *Amazon Cognito* o *Auth0*. La implementación de estas herramientas no solo evita que el desarrollador se encargue de pensar esta lógica tan compleja desde cero, sino también que la mejora permitiendo incluso usar validaciones de acceso biométricos, registración con cuentas de *Google* o *Facebook*, entre otros. Se destaca que no se implementaron debido a que uno de los requisitos no funcionales de este proyecto fue la utilización de herramientas *Open Source*, y estas si bien tienen una capa gratuita generalmente cobran por el servicio. A raíz de esto, surge la posibilidad de una mejora para la

universidad que es contar con su propia versión de esta herramienta, que podría incluso unificar las bases de datos de estudiantes para no duplicar información y unificar los criterios de acceso no sólo con respecto a los laboratorios, sino para cualquier ámbito y aplicación en la universidad.

7. Reflexión sobre la Práctica Profesional Supervisada como espacio de formación

Se destaca como reflexión sobre el desarrollo de la PPS cómo es trabajar profesionalmente en un ámbito académico. Si bien es cierto que, como estudiantes tenemos trabajos integradores que nos acercan al entorno profesional, también se nota la importancia de trabajar en un ámbito académico pensando en el contexto de la sociedad y la comunidad.
Otra reflexión es sobre la utilización del conocimiento adquirido en toda la carrera, donde mientras se van desarrollando las partes del proyecto, se puede ir observando las temáticas que se fueron viendo durante la carrera y nos da la posibilidad de aplicarlas correctamente, algunos temáticas mencionadas durante el desarrollo fueron “Redes y el modelo cliente/servidor”, “Metodología de Programación y los patrones de diseños”, “Ingeniería de Software y el enfoque Agile” entre otras que aportaron también conocimientos para pensar y analizar de manera crítica y detallada cada aspecto del desarrollo de software.

8. Índice de figuras

Figura 1 - Flujo de interacciones entre partes del sistema……..……16

Figura 2 - Prototipo de interfaz “Lista de Usuarios”.....23

Figura 3 - Estructuras de carpetas en el código fuente del proyecto.....26

Figura 4 - Interfaz “Lista de Usuarios” con resaltado de secciones.....31

Figura 5 - Interfaz de “Edición de usuario” con resaltado de secciones….....33

Figura 6 - Ejemplo de traducción de errores con el componente Error Component.....34

Figura 7 - Ejemplo de Busy Component en la carga en interfaz de acceso……....35

Figura 8 - Interfaz de “Registración de usuario”.....39

Figura 9 - Interfaz “Acceso”.....41

Figura 10 - Interfaz “Lista de Usuarios”.....43

Figura 11 - Interfaz “Lista de Usuarios” en versión móvil.....44

Figura 12 - Interfaz “Lista de Laboratorios”.....46

Figura 13 - Interfaz de configuración de laboratorio.....48

Figura 14 - Interfaz “Lista de Turnos”.....49

Figura 15 - Vista de bienvenida en la interfaz “Mis Turnos”.....51

Figura 16 - Vista de selección de laboratorio.....53

Figura 17 - Vista de selección de horario.....55

Figura 18 - Vista de confirmación de turno.....57

Figura 19 - Interfaz “Perfil”.....58

Figura 20 - Flujo de eventos entre la interfaz “Perfil” y la aplicación utilizando observables……...59

Figura 21 - Diagrama de flujo de interoperabilidad entre plataforma y laboratorio remoto…61

Figura 22 - Interfaz del laboratorio remoto luego de la validación de acceso correcta.....62

Figura 23 - Interfaz del laboratorio remoto luego de finalizar el turno.....62

Figura 24 - Interfaz del laboratorio remoto con un token invalido…….....63

Figura 25 - Ejemplo de ejecución de pruebas unitarias con Karma y Jasmine.....67

Figura 26 - Resultados de la prueba de rendimiento y accesibilidad en interfaz principal…..69

Figura 27 - Resultados de la prueba de rendimiento y accesibilidad en interfaz “Mis Turnos”.....70

Figura 28 - Resultados de la prueba de rendimiento y accesibilidad en interfaz “Lista de Usuarios”.....71

9. Bibliografías

Sommerville, I. (2011). *Ingeniería de software*. México: Pearson Educación.

Freeman, E., Robson, E., Bates, B., Sierra, K. (2004). *Head First, Design Patterns*. O'Reilly Media, Inc.

Pressman, R. (2010). *Ingeniería del Software: Un Enfoque Práctico*. México: McGraw Hill Interamericana Editores.

Feature modules. (s.f). Recuperado de https://angular.io/guide/feature-modules.

Component testing scenarios. (s.f). Recuperado de https://angular.io/guide/testing-components -scenarios.

Getting started with NgOptimizedImage. (s.f). Recuperado de https://angular.io/guide/image-directive.

Custom stepper using the CdkStepper. (s.f). Recuperado de https://v16.material.angular.io/guide/creating-a-custom-stepper-using-the-cdk-stepper.

Observer. (s.f). Recuperado de https://rxjs.dev/guide/observer.

Docker Documentation. (s.f). Recuperado de https://docs.docker.com/desktop.

William Bastidas. (2020). *Angular 9: Estructura de carpetas para una aplicación escalable*. Recuperado de https://medium.com/williambastidasblog/angular-9-estructura-de-carpetas-de-para-una-aplicaci%C3%B3n-escalable-a34ab5dd6aaa.

Nginx Documentation. (s.f). Recuperado de https://nginx.org/en/docs/

Metodología Kanban: revoluciona tu manera de trabajar más ágil. (s.f). Recuperado de https://blog.trello.com/es/metodologia-kanban.

Angular Single Page Applications (SPA). (s.f). Recuperado de https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa.