BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS
DE LA COMPUTACIÓN

Diseño y modelado de un software administrador para gestión empresarial.

TESIS PARA OBTENER EL GRADO
EN LICENCIATURA EN INGENIERÍA
EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
Claudia Berenice Hernández Anota

ASESOR:
Dr. Mariano Larios Gómez

Puebla, Puebla
Noviembre 2024

Dedicatoria

A mi mamá, por siempre apoyarme en todo momento, por su paciencia y aliento en mis momentos más dificiles.

A mi hermana, que siempre me brindo sus porras, sus palabras de ánimo y sobre todo su amor incondicional.

A mis amigos, quienes con su compañía y motivación hicieron de esta experiencia un trayecto más enriquecedor y memorable.

A mis abuelos, mi abuelita Lupita que me ha brindado siempre su amor y apoyo, y una dedicatoria hasta el cielo para mi abuelito José, quien tuvo la oportunidad de acompañarme en mi graduación sabiendo que desde el cielo me acompaña y me guía en este proceso.

A mis suegros y cuñados, por recibirme con los brazos abiertos y brindarme su apoyo en todo momento.

Y por supuesto a ti, mi amor y compañero de vida Eric que siempre me ha apoyado y me ha sostenido aún cuando yo no puedo hacerlo. Gracias por ser mi ancla en los momentos difíciles, mi impulso en cada paso, y mi motivación para alcanzar nuevas metas.

Claudia Berenice Hernández Anota

Agradecimientos:

Quiero agradecer a todas las personas que me acompañaron y brindaron su apoyo en este proceso.

A mis profesores, por su guía, conocimiento y paciencia en cada etapa de este proyecto. Con especial mención a mi asesor y profesor el Dr. Mariano Larios Gómez quien me guio y brindo su apoyo en esta etapa como estudiante, de igual manera al profesor Carlos Zamora Lima, que sin el apoyo de ambos no hubiera sido posible este logro.

A todos ustedes, mi más profundo agradecimiento.

ÍNDICE

Capítulo 1.

1.1 Resumen

La necesidad de optimizar los procesos de entrada y salida de usuarios, garantizar el control de acceso y simplificar la administración de laboratorios a través de esta aplicación web es evidente, es por ello que en esta tesis se propone el diseño y desarrollo de una aplicación hecha con la tecnología de Angular, orientada a gestionar de manera eficiente y fácil los recursos y actividades de un laboratorio dentro de la facultad.

La elección de Angular como framework de desarrollo se sustenta en su capacidad para crear aplicaciones web robustas y escalables, ofreciendo una interfaz de usuario intuitiva y una experiencia de usuario optimizada. La implementación de esta solución tecnológica permitirá agilizar los procesos administrativos, mejorar la eficiencia operativa y reducir el margen de error en las tareas de gestión.

Asimismo, la seguridad de los datos y el control de acceso son aspectos fundamentales en cualquier entorno laboral, y los laboratorios no son la excepción. La aplicación web propuesta garantizará la confidencialidad de la información a través de mecanismos de seguridad robustos, como la autenticación de usuarios en múltiples niveles y el cifrado de datos sensibles. Además, facilitará la colaboración entre los diferentes actores involucrados en la gestión del laboratorio, mejorando la comunicación y la toma de decisiones.

El desarrollo de esta aplicación web representa un avance significativo en la gestión de laboratorios, al ofrecer una herramienta versátil, segura y eficiente que permitirá optimizar los procesos, mejorar la productividad y garantizar la calidad de los servicios ofrecidos.

1.2 Introducción

En la actualidad, la eficiente gestión de laboratorios se ha convertido en un requisito fundamental en diversos ámbitos, tales como en la investigación científica, la industria y la educación. La implementación de una aplicación web para el registro de usuarios basada en tecnologías web y móviles, como Angular, se presenta como una solución altamente efectiva para optimizar los procesos de entrada y salida de usuarios, garantizar la seguridad y control de acceso, y simplificar la administración de los laboratorios.

Esta aplicación web permitirá a los usuarios administradores registrar a las personas que acceden al laboratorio, guardando información importante como lo son el: nombre, fecha, hora de entrada y salida, actividades dentro del laboratorio y notas extras para una mayor información. Al mismo tiempo la aplicación web permitirá a los usuarios registrados gestionar de manera centralizada usuarios, préstamos de equipo y el inventario del este mismo.

Reconociendo las áreas de oportunidad, se torna como propuesta esta problemática desarrollar la aplicación web en un entorno que sea capaz de ayudar a la reducción de dichos problemas presentes, considerando contexto, la lógica de negocio y la meta final, por lo tanto, se planifican las siguientes etapas desde el frontend:

Primero se tendrá en cuenta la elección del lenguaje y herramientas tecnologicas a utilizar, para este trabajo se eligio el framework de Angular, ya que tiene un nivel de complejidad de medio a elevado y ofrece soluciones robustas, escalables y optimizadas para lograr un estilo de codificación homogéneo y de gran

modularidad. Su desarrollo se realiza por medio de TypeScript o JavaScript, en este último se ofrecen diversas herramientas adicionales al lenguaje como tipado estático o decoradores. Su nombre «Angular» proviene del concepto de paréntesis angulares (< >) que se utilizan en HTML.

En cuanto al diseño y modelado de la aplicación web se baso en un diseño sencillo y fácil de usar, para esto se utilizo la herramienta Balsamiq, que es una herramienta de prototipado rápido ampliamente reconocida en el mundo del diseño de experiencia de usuario (UX) por su enfoque en la simplicidad y eficiencia. Facilita a los diseñadores la creación de bocetos o "wireframes" para interfaces de usuario, permitiendo una rápida iteración y colaboración, ya que simula la experiencia de bosquejar en un cuaderno o en una pizarra blanca, pero en un entorno digital. Su interfaz de arrastrar y soltar permite a los usuarios crear diseños funcionales y estéticamente agradables sin preocuparse por detalles gráficos complejos.

1.3 Justificación

En un contexto donde la organización y la comunicación son factores clave para el buen funcionamiento de una empresa o de una institución, la implementación de una página web para gestionar el flujo de personal y el control de materiales existentes se vuelve escencial.En este caso los laboratorios de software y la gestion requieren un sistema que no solo permita controlar inventarios y acceso a recursos, sino que también facilite la organización de las actividades diarias.

Una plataforma web para una gestión especificica como la propuesta de este proyecto, aporta multiples beneficios, por ejemplo, reduce la pérdida de información, optimiza el tiempo en la búsqueda de esta y facilita la toma de desciciones basadas en los datos actualizados en tiempo real. Esto asegura en un porcentaje alto el mal uso de la información.

Por otro lado, el uso de un framework com Angular y una base de datos como Firebase en el desarrollo de la pagina web no solo garantiza una interfaz entendible y una experiencia de usuario buena, sino que también ofrece seguridad,

escalabilidad y un buen manejo de datos en tiempo real, en este caso el método propuesto es el de SCRUM, ya que permite un mejor manejo de los tiempos de desarrollo y con el feedback constante de los usuarios se puede obtener un mejor control.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar una aplicación web integral de control y registro de laboratorios, respaldada por tecnologías como Angular y una base de datos NoSQL, con el fin de proporcionar una solución segura y eficiente para la gestión de laboratorios en diversos sectores.

1.3.2 Objetivos especificos

1. Proponer un modelo de un software administrador para la gestión de laboratorios e implementarlo a nivel empresarial.

2. Mantener y controlar el flujo de visitas en un laboratorio.

3. Tener un mejor manejo y cuidado de los equipos que entran y salen del laboratorio.

Capítulo 2. Estado del arte

Macías et al., (2020). Con el aparecimiento de la web y del internet cada vez se exige mayor agilidad, eficiencia y dinamismo en los diferentes procesos por ejemplo: en atención al usuario, en el cumplimiento del accionar laboral, asignar responsabilidades, entre otras; pero deben tomarse decisiones de innovación y actualización de los procedimientos para poder acceder a información oportuna y actualizada, de lo contrario se queda en la incompetencia dejando una imagen de retroceso y obsoleta (Voces, 2010). Sin embargo, esta apreciación debe ser eliminada proporcionando alternativas de progreso y desarrollo corporativo.

Asimismo, en Traverso et al., (2017) indican que la incorporación de aplicaciones en instituciones educativas fortalece la academia y brinda una mejor administración a nivel general, facilitando soluciones a problemas y automatizando procesos que se realizan manual y repetitivamente.

Por otra parte Jimenez Jaramillo, E. E. (2020), menciona que en la actualidad las organizaciones, entidades y empresas comerciales están incorporando tecnología a sus procesos, como es la gestión de inventario siendo importante en toda empresa ya que tienen como finalidad obtener buenos resultados beneficiosos en su producción y generar utilidad, por esta razón en la actualidad se implementan software que son de gran ayuda en la optimización de procesos en cuanto al inventario y su manejo, una vez que los métodos son automatizados en dicha empresa esta comienza a generar incrementos en sus ingresos económicos dependiendo de su actividad comercial.

En cuanto a la base de datos utilizada, Firebase, Medina Gutiérrez, C. M. (2022). *Decia que* las bases de datos necesitan ser administradas, de allí nace el sistema de gestión de base de datos, que permite al usuario interactuar con la base de datos en tiempo real. Por lo que se cuenta con varias herramientas, tales como manejo de copias de seguridad, manipulación de datos permitiendo actualizar, crear, eliminar u obtener los mismos, construcción, definición y recuperación de datos. Al trabajar con un sistema de gestión de base de datos se cuenta con el control de datos repetidos, se restringe el acceso a usuarios no deseados, se representan los datos vinculados entre sí, existe una recuperación ante fallos, se crean copias de seguridad, en sí, permite controlar el manejo de los datos.

Rodríguez Rodríguez, M. C. (2014). En esta era de tecnología e información, las empresas han pasado de utilizar sistemas rudimentarios para el manejo de información de la organización, de sus procesos y en general de todo lo que implique almacenamiento y actualización de datos, al empleo de sistemas tecnológicos que permiten al usuario no solamente manipular y almacenar información, sino organizarla, tener acceso a ella de manera rápida, insertarla más fácilmente, analizarla y gestionarla más eficientemente. Cabe resaltar que aún hay pequeñas empresas que siguen utilizando las carpetas y los fólderes en físico o las miles de hojas de cálculo con información redundante, ya que la percepción que tienen sobre el cambio de un sistema rudimentario hacia un sistema tecnológico de gestión es bastante negativa, sobretodo porque no ven el valor agregado para una empresa de un tamaño reducido y por el contrario perciben este cambio como una inversión innecesaria que no solamente implica capacitación del personal sino un gasto de dinero y tiempo en la configuración inicial del sistema.

En el trabajo de López Maradiaga, A. R. (2021) define que Cloud Firestore es la base de datos más reciente de Firebase para el desarrollo de apps para dispositivos móviles. Aprovecha lo mejor de Realtime Database con un modelo de datos nuevo

y más intuitivo. Con Cloud Firestore también se pueden realizar consultas más ricas y rápidas, y el escalamiento se ajusta a un nivel más alto que Realtime Database.

Computing, C. (2012) define a la computación en la nube como la entrega de servicios de computación a través de la Internet, o “la nube”. En lugar de utilizar servidores locales o computadoras personales para almacenar, gestionar y procesar datos, la computación en la nube permite acceder a estos recursos desde centros de datos remotos gestionados por proveedores de servicios en la nube.

Makoveev Routskaia, E. (2024). **Angular** proporciona herramientas ya integradas, para la gestión de los estados de la aplicación, como del enrutamiento de la página web, así como de la validación de formularios y de la comunicación con servidores a través de solicitudes HTTP. Gracias a estas características, permiten construir aplicaciones completas con todas las funcionalidades necesarias directamente desde el framework, sin necesidad de depender de bibliotecas externas.

Para la metodologia utilizada, SCRUM, Ramírez et al., (2019), menciona que la metodología ágil SCRUM, integra buenas prácticas y el trabajo colaborativo de equipo y obtener mejores resultados, mediante la colaboración de un equipo altamente competitivo. En Scrum se presentan entregas parciales del proyecto, las cuales son priorizadas de acuerdo a las aportaciones que realizan al proyecto y son valoradas por los usuarios finales del proyecto. Scrum se recomienda en proyectos con entornos complejos, con requerimientos que se modifican y que además es necesario obtener resultados rápidos, en donde la innovación, la flexibilidad y la productividad son básicas.

Capítulo 3. Marco teórico

3.1 Angular

Angular es un framework de JavaScript de código abierto mantenido por Google que se utiliza para crear páginas web de tipo SPA (Single Page Application). Esto significa que la página se carga solo al inicio y, luego, las sucesivas actualizaciones se producen sin necesidad de recargarla en forma completa.

Para desarrollar con esta herramienta es aconsejable tener conocimientos sobre HTML, CSS, JavaScript y programación orientada a objetos. El lenguaje de programación principal de Angular es TypeScript. La heramienta Angular CLI se utiliza para la creación de un proyecto en Angular, la cual, se explicara en capitulos posteriores.

Las partes importantes a considerar cuando se trabaja con Angular son:

1. Módulos: permiten organizar el código de tal manera que ayuda a gestionar el desarrollo de aplicaciones complejas y diseñarlas para poder ser reutilizadas

2. Componentes: representan una porción de la aplicación y este esta contenido dentro de un modulo. Cada componente define una clase que contienen datos y lógica de la aplicación que se asocia con un HTML y CSS que define una vista. Un componente esta orientado a la experiencia de usuario.

3. Servicios: es un componente lógico de código reutilizable que cumple con una función especifica, su objetivo principal es proveer una funcionalidad a un componente.

Dentro de las principales definiciones de una interfaz de usuario son

1. HTML: no es un lenguaje de programación; es un lenguaje de marcado que define la estructura del contenido. HTML consiste en una serie de elementos que se usan para encerrar diferentes partes del contenido y hacer que se vean o comporten de una determinada manera. Las etiquetas de encierre pueden hacer de una palabra o una imagen un hipervínculo a otro sitio, se pueden cambiar palabras a cursiva, etc.

2. CSS: es un lenguaje de hojas de estilo que permite aplicar estilos de manera selectiva a elementos en documentos HTML.

3. Bootstrap: incluye plantillas de diseño basadas en HTML y CSS para tipografía, formularios, botones, tablas, navegación, modales, carruseles de imágenes y muchos otros, así como complementos de JavaScript opcionales.
   Bootstrap también te da la posibilidad de crear fácilmente diseños responsivos.

3.2 Firebase

Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo.

Su función esencial es hacer más sencilla la creación aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.

Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma. Las finalidades de las mismas se pueden dividir en cuatro grupos: desarrollo, crecimiento, monetización y análisis.

Firebase cuenta con un sistema de almacenamiento que permite guardar ficheros de aplicaciones y sincronizarlos. Este almacenamiento es de gran ayuda para tratar archivos de los usuarios (por ejemplo, fotografías que hayan subido), que se pueden servir de forma más rápida y fácil. También hace la descarga de referencias a ficheros más segura.

En firebase existe el acrónimo CRUD (Create, Read, Update, Delete), esta definición es muy importante ya que en la mayoría de las secciones de la página web se utiliza la creación, lectura, modificación y eliminación de elementos.

Capítulo 4. Diseño del proyecto de software

El diseño de software es una etapa fundamental en el desarrollo de cualquier aplicación, ya que define cómo se organizará y estructurará el sistema antes de que comience la codificación. En el caso de la aplicación móvil desarrollada en Angular para la gestión de un laboratorio de desarrollo tecnológico, el diseño se ha enfocado en asegurar que el sistema sea eficiente, fácil de usar y adaptable a las necesidades específicas de este tipo de entorno.

El laboratorio de desarrollo tecnológico requiere una plataforma que permita gestionar recursos, equipos de trabajo y el seguimiento del desarrollo de productos o soluciones tecnológicas. Para cumplir con estos objetivos, el diseño se ha centrado en dos componentes principales: la interfaz de usuario y la arquitectura de la aplicación.

4.1 Interfaz de Usuario (UI)

La interfaz de usuario ha sido diseñada para que los profesionales del laboratorio puedan gestionar proyectos tecnológicos de manera simple y eficiente. Para lograr esto, se estructuró la aplicación con un menú principal que permite acceder a funciones clave como:

4.1.1 Gestión de usuarios

Aquí se registran a los usuarios para posteriormente hacer un seguimiento de cuantos alumnos/personas ajenas son las que ingresan diariamente al laboratorio.

Figura 1. Vista general de visitas

Como se puede ver en la Fig. 1 se la vista general de la lista de visitas con botones para agregar una nueva visita y un botón Editar en cada fila agregada.

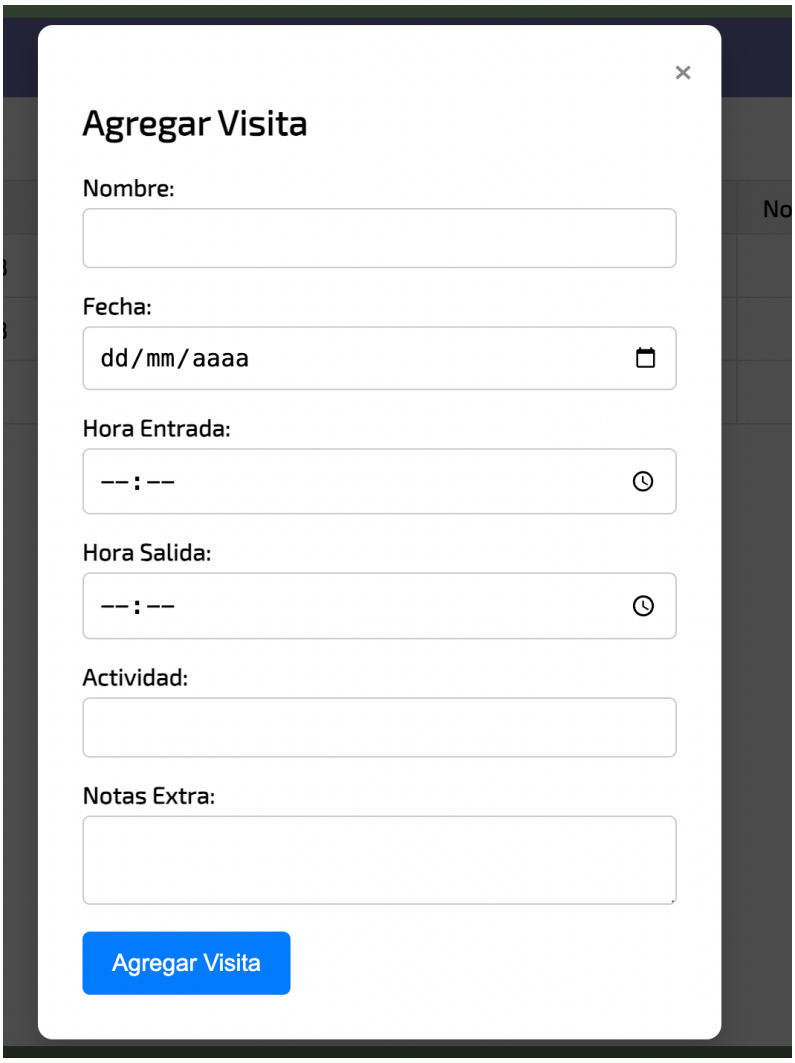

Figura 2. Vista agregar visita

En la Fig. 2 se puede ver el formulario para agregar una nueva visita, en esta vista los campos obligatorios son: nombre, fecha, hora de entrada y actividad.
Para los campos hora de salida y notas extras se pueden modificar con el botón editar que se muestra en la vista general.

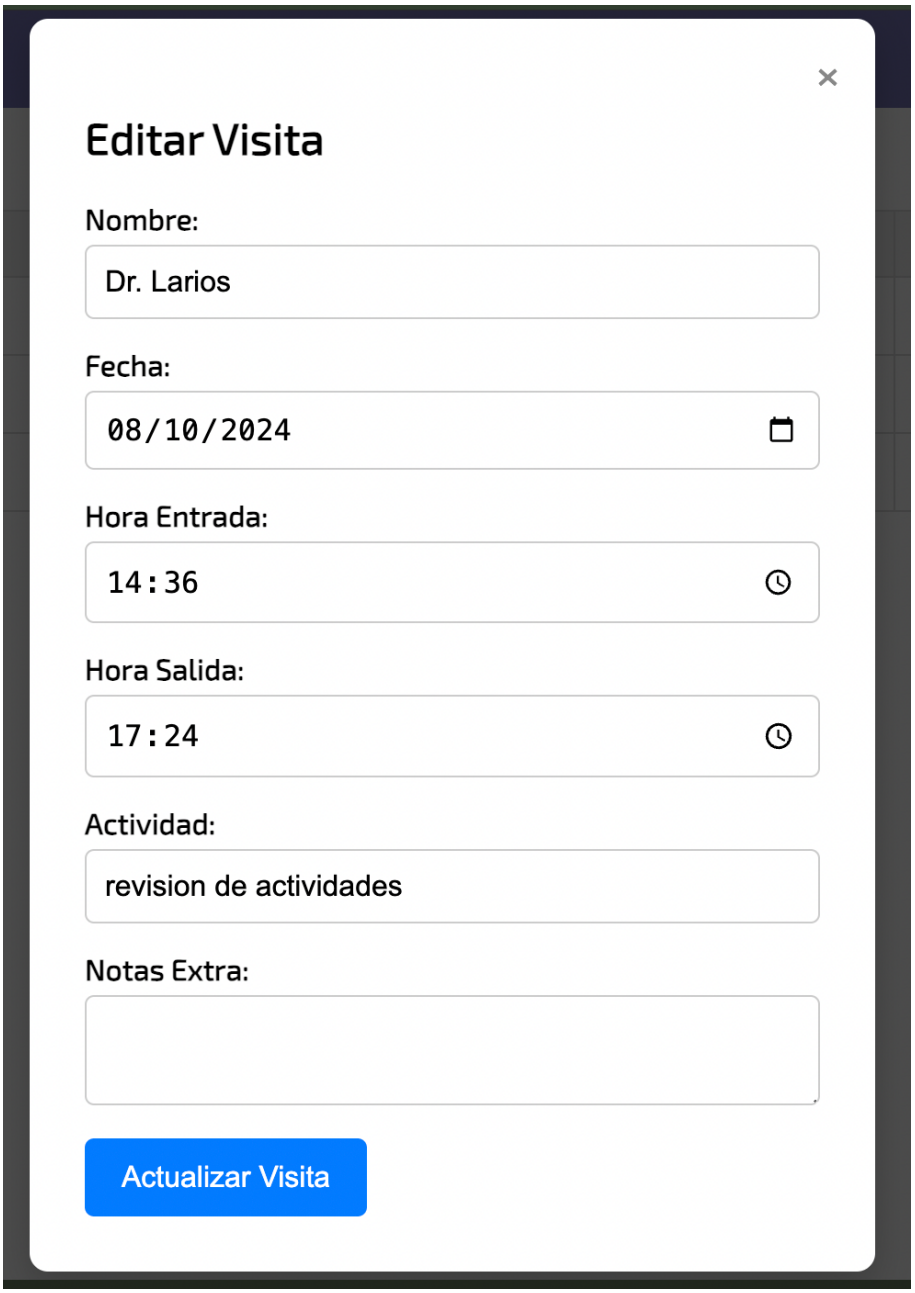

Figura 3. Vista editar visita

En la Fig. 3 se puede ver el mismo formulario que en la vista de agregar visita (ver Fig. 2) con la diferencia de la recuperación de datos del item seleccionado para poder tener la referencia y saber que elemento se esta modificando.

Para esta función de gestión de usuarios no se utilizó el CRUD completo, ya que al permitir la opción de Delete se estaría perdiendo información importante para en un futuro generar reportes.

4.1.2 Control de recursos y materiales

Esta sección permite a los usuarios gestionar el inventario del laboratorio, como componentes electrónicos, equipos de prueba, y prototipos. Se pueden registrar entradas y salidas de materiales, así como solicitar nuevos recursos.

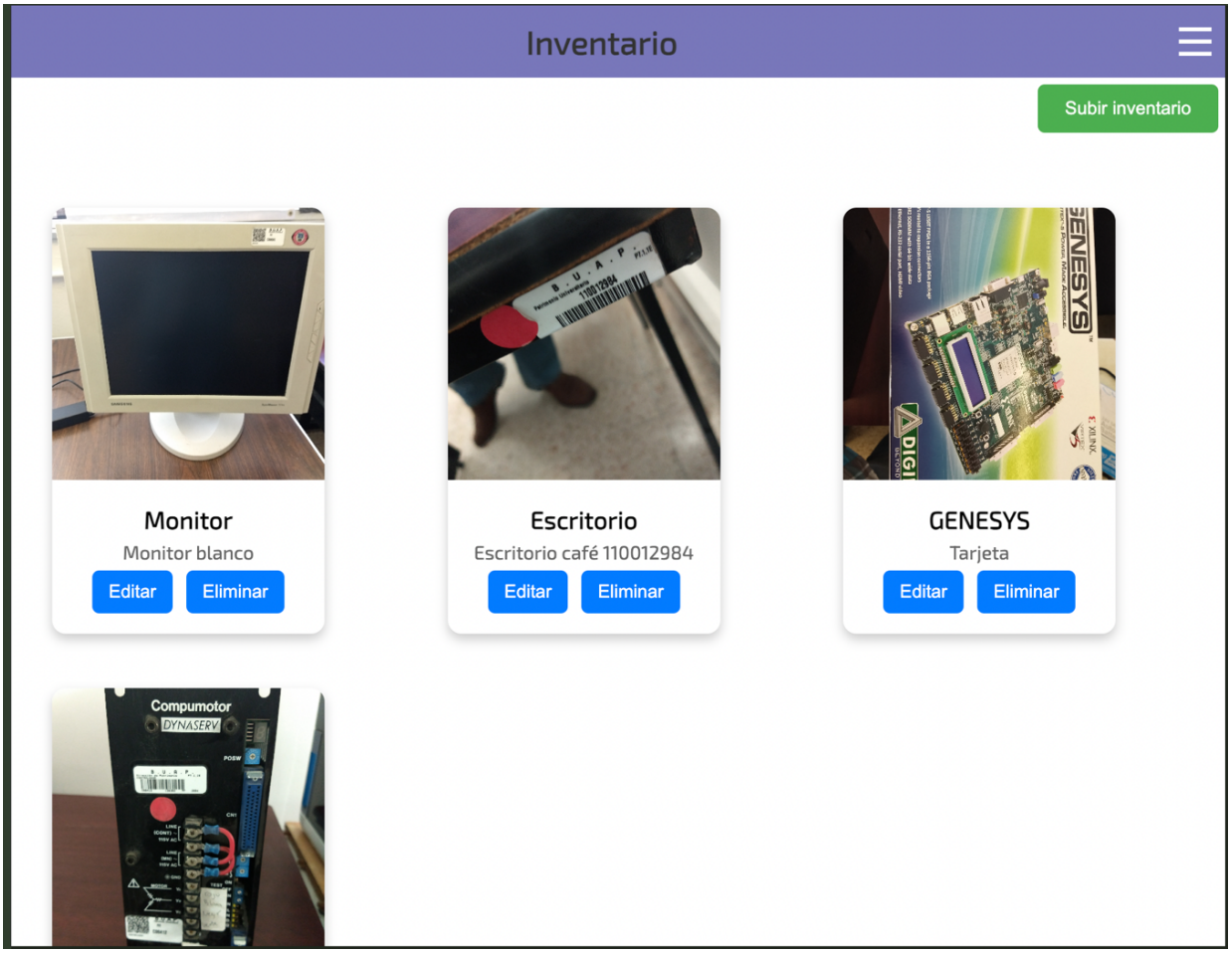

Figura 4. Vista general de inventario AUTORIA PROPIA

En la Fig. 4 se muestra la vista general de los equipos existentes en el laboratorio. Para esta función si se usa el CRUD completo, se crea un nuevo elemento en el botón ‘Subir inventario’ (ver Fig. 5), se lee la información y se muestra en los cuadros de imagen, se puede hacer update que es el botón editar (ver Fig. 6) y por ultimo el delete en el botón eliminar (ver Fig. 7)

Figura 5. Vista subir nuevo inventario

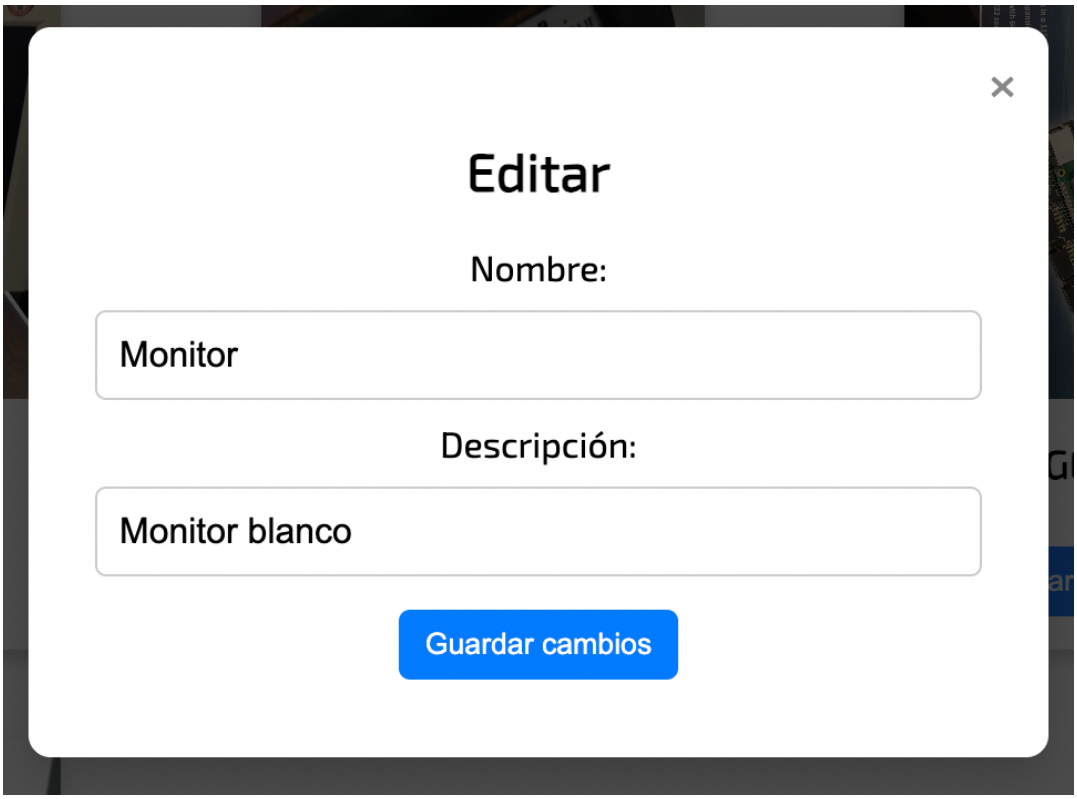

Figura 6. Vista editar inventario existente

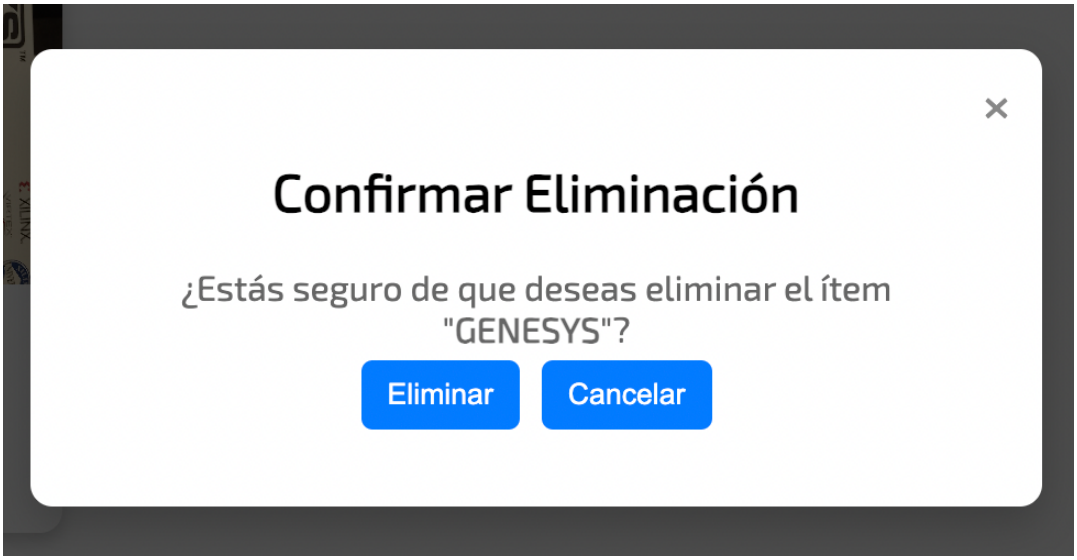

Figura 7. Vista eliminar inventario

Para garantizar una experiencia de usuario óptima, la interfaz está diseñada de forma intuitiva, con menús desplegables, botones grandes y accesibles, y pantallas que presentan la información de manera clara. Además, la aplicación es

responsiva, lo que significa que se adapta a cualquier tamaño de pantalla, permitiendo su uso tanto en smartphones como en tablets.

4.2 Arquitectura del Sistema

El diseño arquitectónico de la aplicación se basó en una estructura modular utilizando el patrón Model-View-Controller (MVC), lo que facilita la organización y el mantenimiento del sistema a lo largo del tiempo. Este patrón separa claramente los datos, la lógica de negocio y la presentación visual, lo que asegura que cada componente cumpla con una función específica sin interferir en los demás.

4.2.1 Modelo (Model)

Se encarga de gestionar los datos del laboratorio, como los proyectos, el personal, los materiales y el estado de los equipos. Estos datos se almacenan en una base de datos que puede ser actualizada de manera automática cada vez que un usuario realiza una acción en la aplicación. (ver Fig. 8)

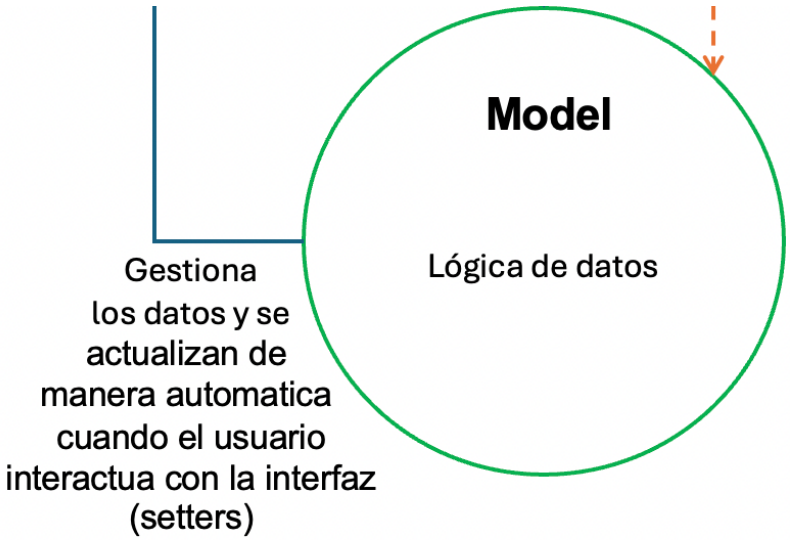

Figura 8. Patron Model

4.2.2 Vista (View)

La vista es la interfaz de usuario que permite a los profesionales del laboratorio interactuar con el sistema. Angular facilita la creación de vistas dinámicas, por lo que cualquier cambio en los datos se refleja automáticamente en la interfaz, sin necesidad de recargar la página. (ver Fig. 9)

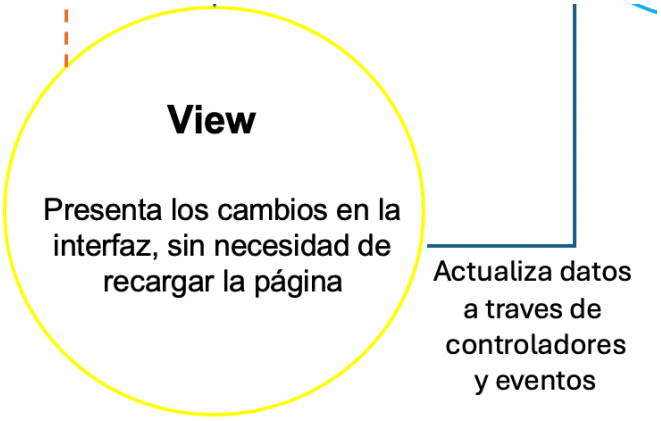

Figura 9. Patron View

4.2.3 Controlador (Controller)

Es el intermediario que procesa las acciones del usuario y actualiza el modelo y la vista. Por ejemplo, cuando un usuario registra un nuevo proyecto o solicita materiales, el controlador toma esa solicitud, actualiza la base de datos y refleja los cambios en la interfaz. (ver Fig. 10)

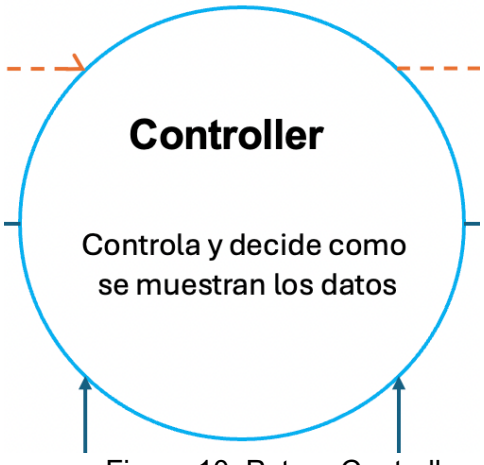

Figura 10. Patron Controller

La modularidad de la arquitectura hace que el sistema sea flexible y escalable, lo que permite agregar nuevas funciones o mejorar las existentes sin afectar el rendimiento general de la aplicación. Además, como se muestra en la Fig. 11 el diseño considera la posibilidad de integrarse con otras plataformas tecnológicas que el laboratorio ya esté utilizando, como sistemas de gestión de proyectos o herramientas de análisis de datos.

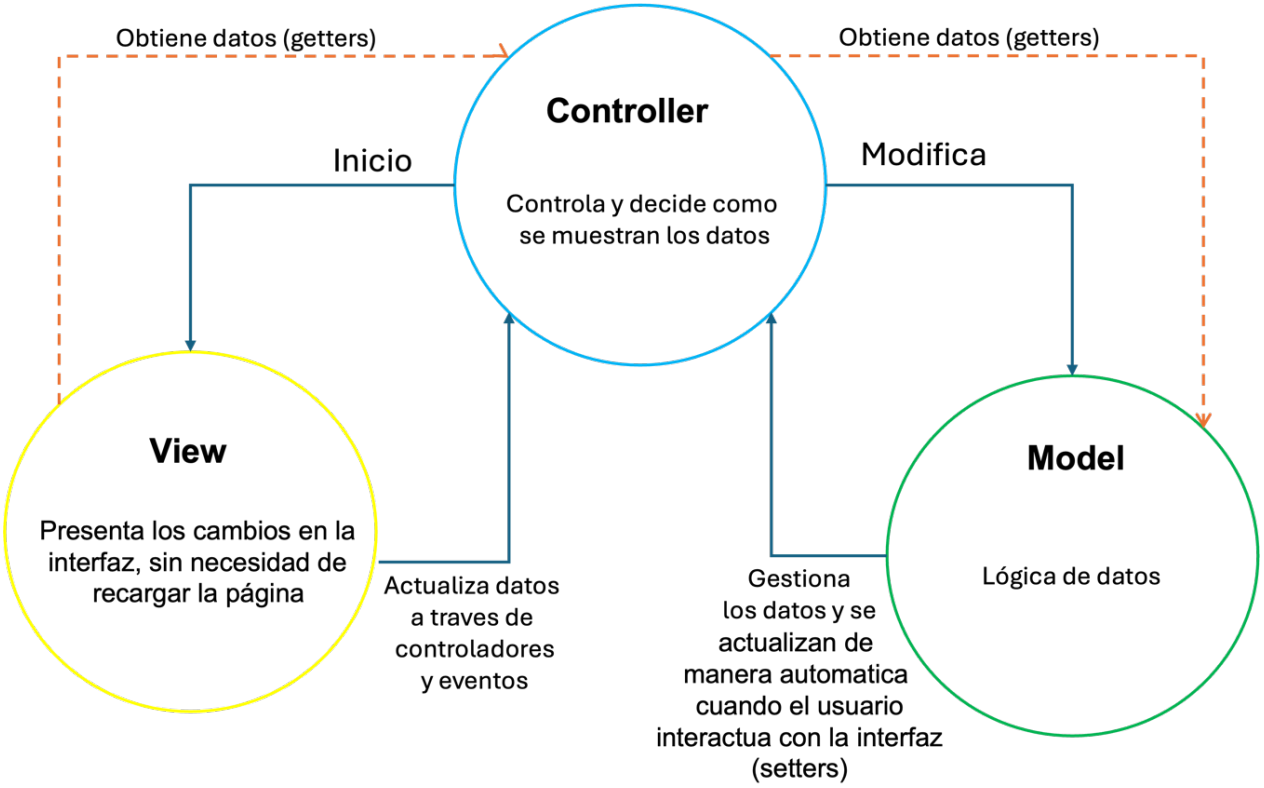

Figura 11. Modelo MVC

4.3 Implementación

4.3.1 Desarrollo de Módulos y Funcionalidades

La aplicación web para la gestión del laboratorio de desarrollo tecnológico se ha dividido en varios módulos clave, cada uno diseñado para cubrir una función específica y optimizar el flujo de trabajo dentro del laboratorio. A continuación, se detallan los módulos principales implementados y las funcionalidades asociadas a cada uno.

- **Módulo de Control de Inventario**

  Este módulo está destinado a gestionar el inventario de equipos y materiales utilizados en los proyectos del laboratorio. Incluye la siguiente funcionalidad:

a) **Registro de Materiales y Equipos:** Los usuarios pueden registrar nuevos ítems en el inventario, incluyendo detalles como nombre y una pequeña descripción donde se podra capturar la ubicación en el laboratorio, y estado de uso, lo que permite también un mejor control sobre su ciclo de vida y mantenimiento

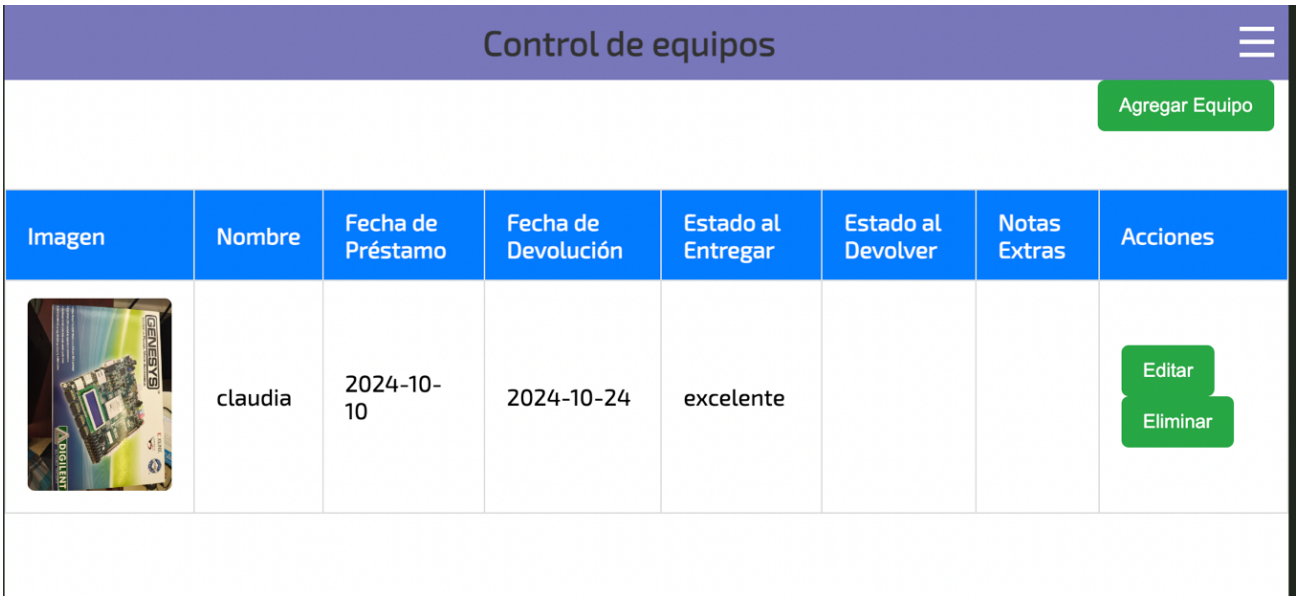

Figura 12. Vista general control de equipos

Como se puede ver en la Fig. 12, se muestra la lista de los equipos prestados, cada elemento se puede editar, eliminar y agregar un nuevo equipo.

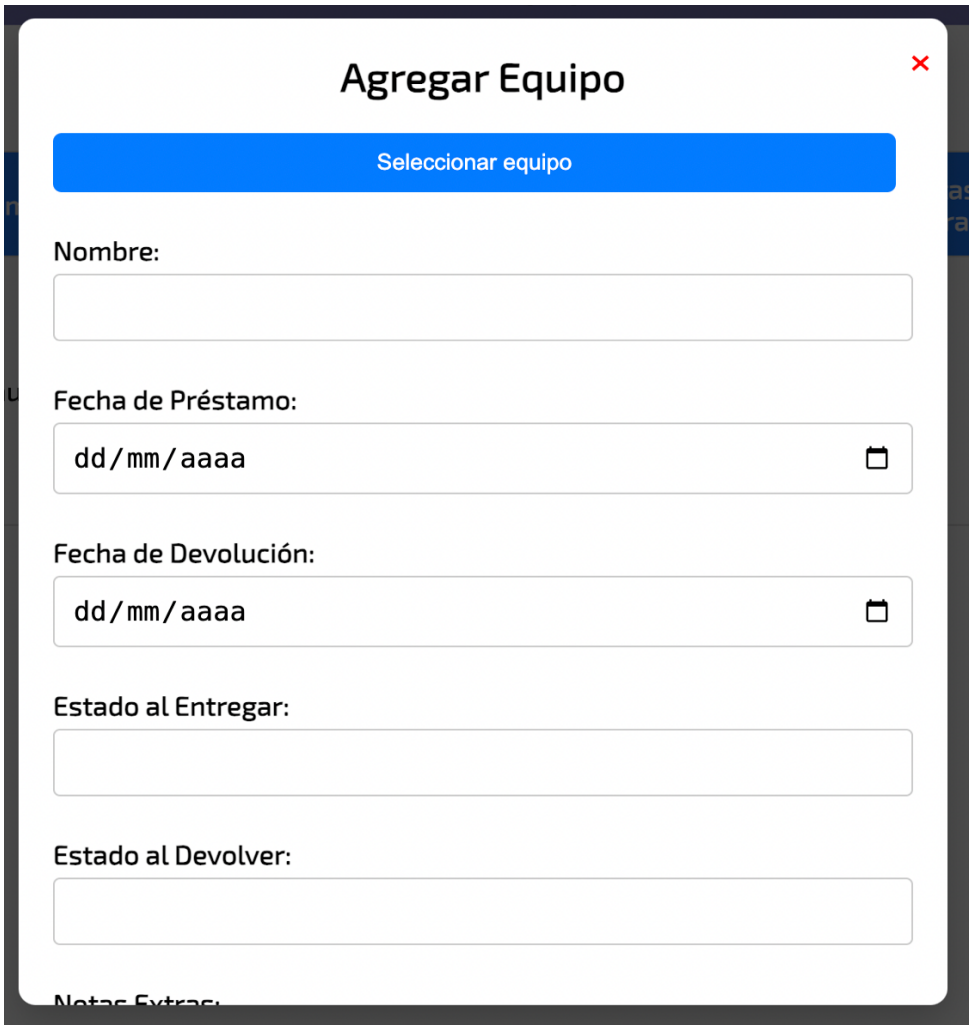

Figura 13. Vista agregar equipo

En la Fig.13 se muestra la interfaz para agregar un nuevo elemento, en esta sección el botón “Seleccionar equipo” abre un nuevo modal en donde se muestran las imágenes dadas de alta desde el inventario, como se muestra en la Fig. 14.

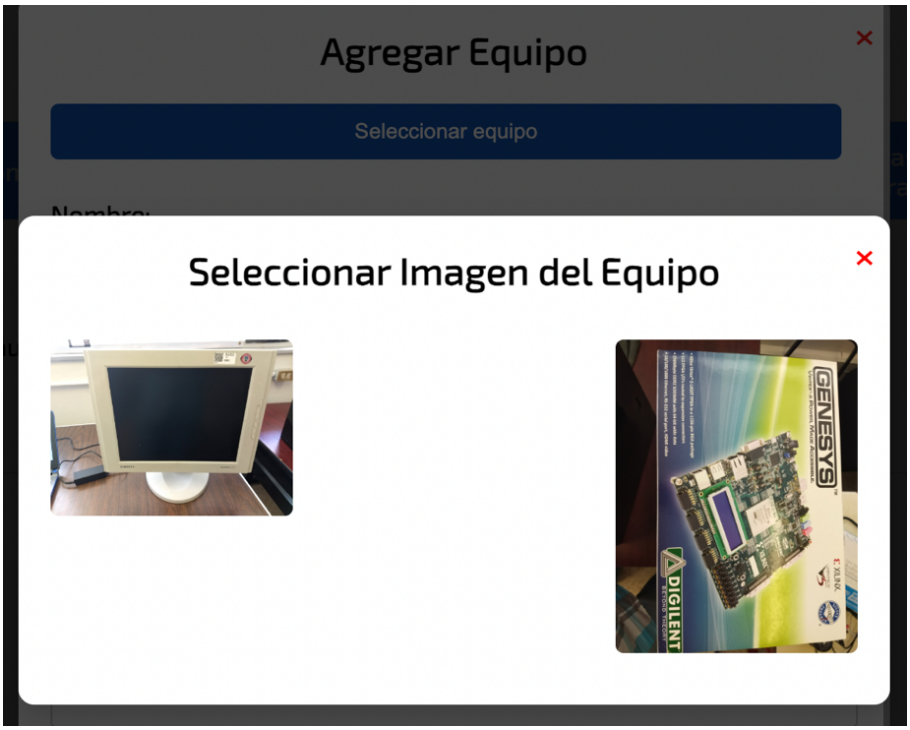

Figura 14. Vista seleccionar imagen del equipo

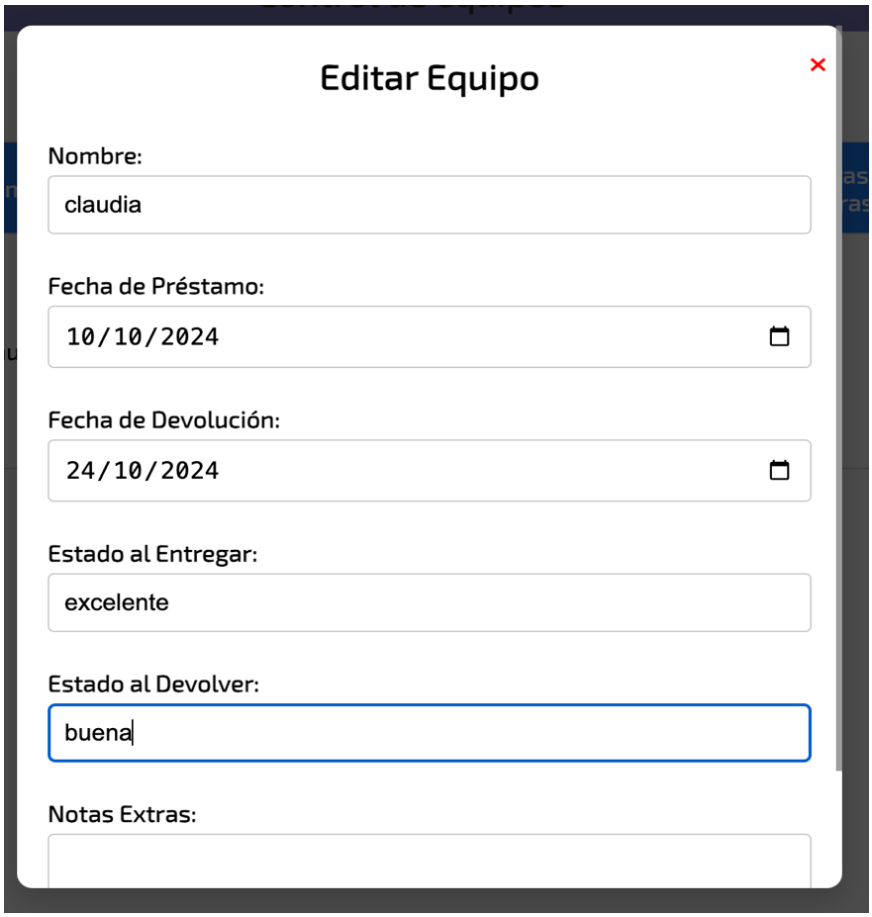

Figura 15. Vista editar equipo

En la Fig. 15 se observa la interfaz para editar cualquier atributo del elemento a excepción de la imagen, esto para tener un mejor control del inventario que entra y sale del laboratorio.

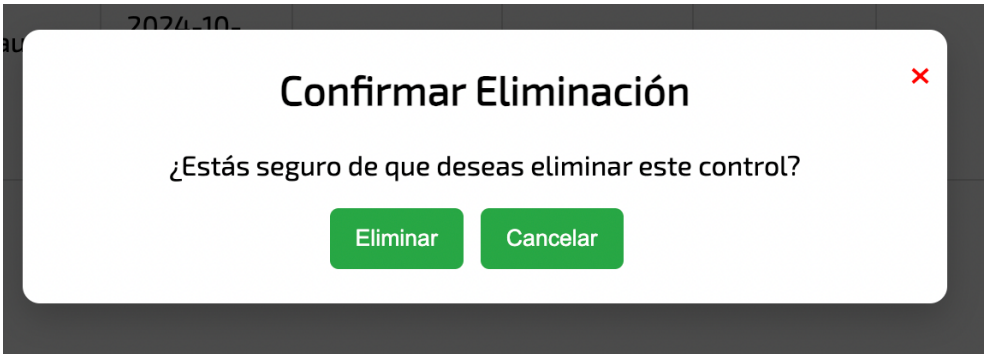

Figura 16. Vista eliminar elemento

Por último en la Fig. 16 se muestra una ventana para confirmar la eliminación de los elementos.

- **Módulo de Gestión de Usuarios**

El módulo de gestión de usuarios está diseñado para controlar la asignación de roles y tareas dentro del laboratorio. Las funcionalidad desarrollada es:

a) **Registro de Usuarios:** En la Fig. 17 se muestra la vista general de la sección de usuarios, mostrando los campos de User y Password si la persona que inicio sesión tiene el rol de ‘admin’ al igual que los botones en la columna de acciones y el boton Agregar usuario. Si el usuario que inicia sesión tiene el rol de user se les mostrara la interfaz que aparece en la Fig. 18, esto con la finalidad de tener un mejor control de la información.

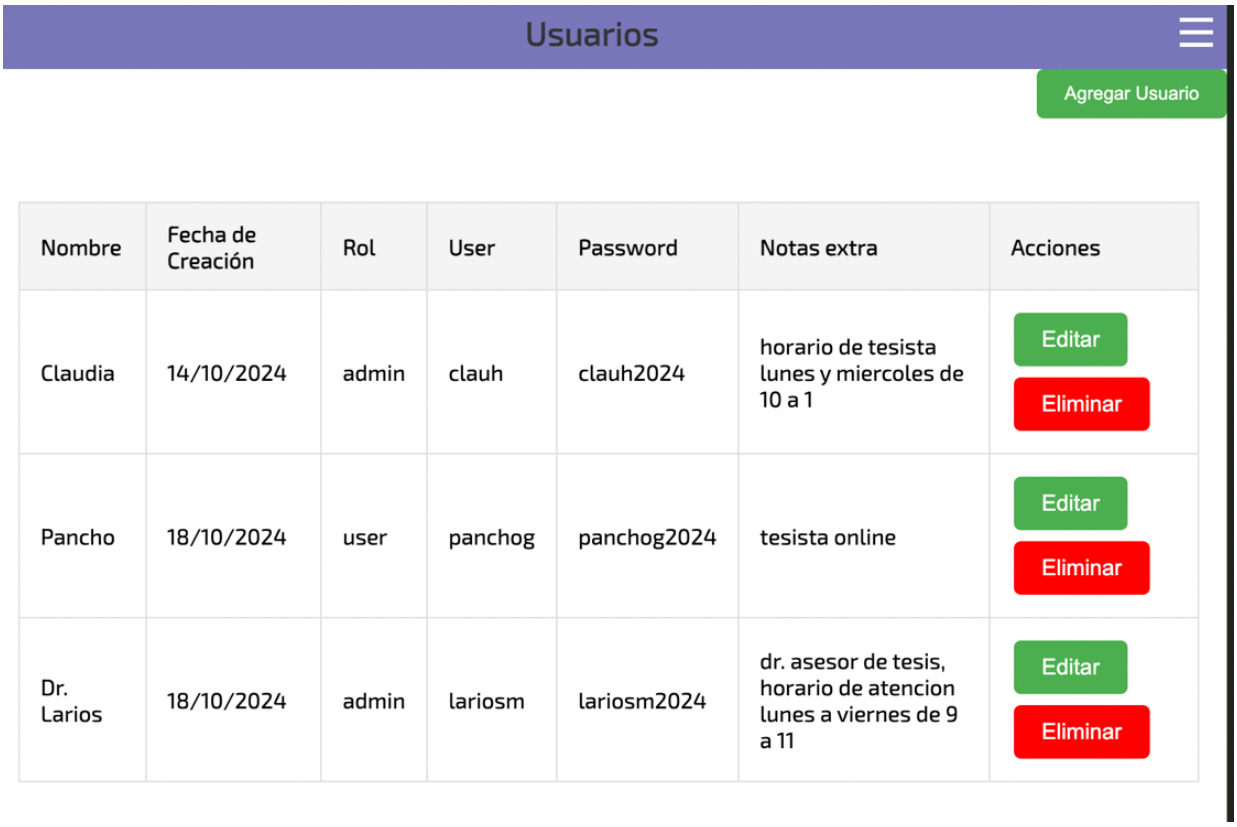

Figura 17. Vista general usuarios (rol admin)

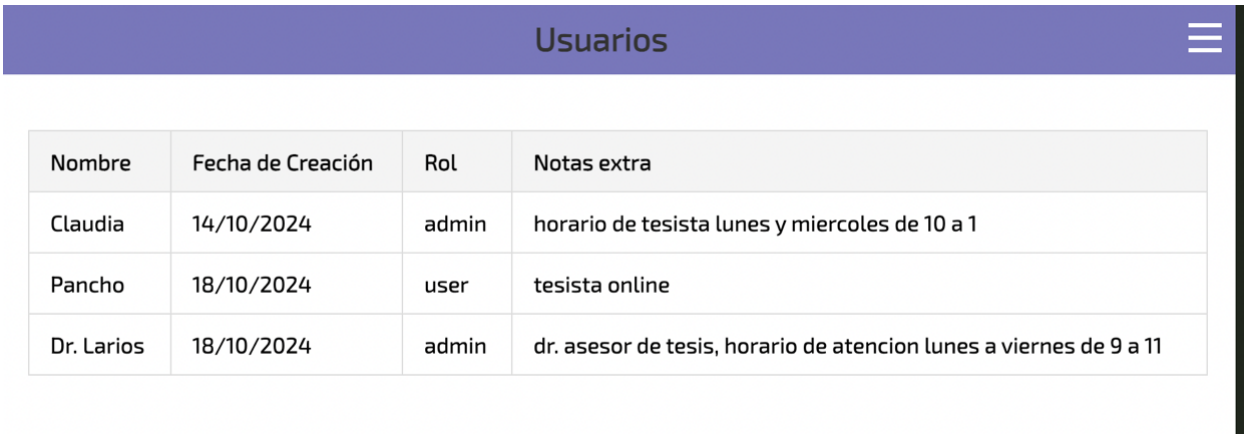

Figura 18. Vista general usuarios (rol user)

Funciones del rol admin:

Agregar usuario: El usuario podrá agregar nuevos usuarios y decidir el rol que tendrán (ver Fig. 19), recordando que si el rol es user solo mostrará información y si es admin podrá acceder a todas las funciones (ver Fig. 17)

Figura 19. Vista agregar usuario (rol admin)

Editar usuario: La interfaz que se muestra en la Fig. 20, recupera los datos del usuario a editar y automaticamente actualizará la información en la vista general.

Figura 20. Vista editar usuario (rol admin)

Eliminar usuario: En la Fig. 21 se muestra el modal para confirmar que se desea eliminar el usuario seleccionado.

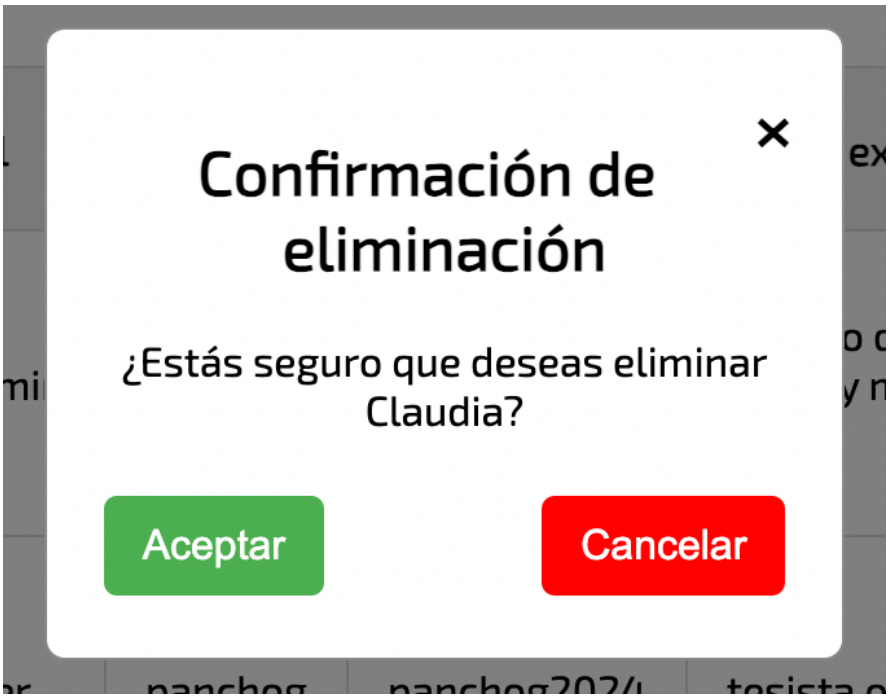

Figura 21. Vista eliminar usuario (rol admin)

- **Módulo de Seguridad y Autenticación**

Este módulo garantiza que solo usuarios autorizados puedan acceder a la aplicación y a las funciones críticas. La funcionalidad incluida es:

a) **Autenticación de Usuarios:** Se desarrolló un sistema de login con autenticación basada en roles (admin, user), que controla el nivel de acceso a las diferentes funcionalidades.

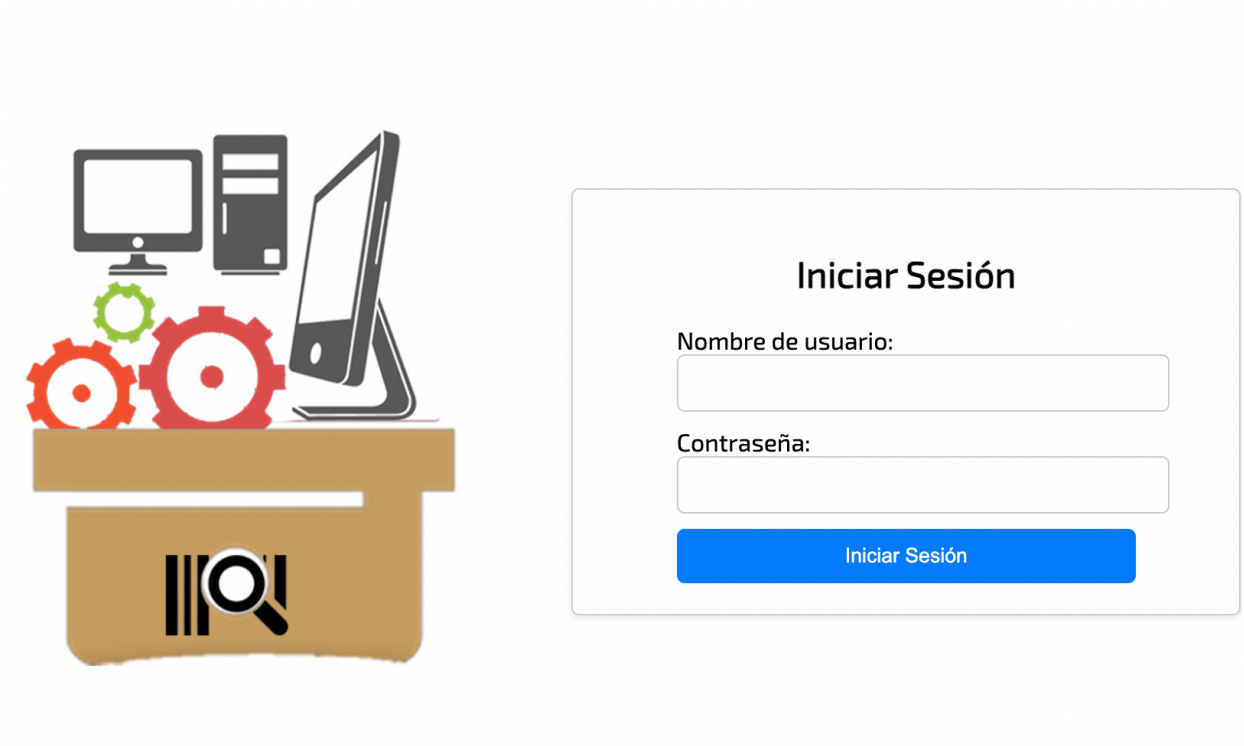

Figura 22. Vista login

Como se muestra en la Fig. 22, la interfaz de login verifica el rol del usuario que inicia sesión para poder asignarle las funcionalidades a las que tiene acceso, en este caso la unica sección que afecta el rol es la de usuarios.

4.3.2 Lenguajes y herramientas utilizadas

El desarrollo de la aplicación web para la gestión de laboratorios se llevó a cabo utilizando una combinación de tecnologías y herramientas que garantizan tanto la robustez como la escalabilidad del sistema. Los lenguajes de programación y las herramientas empleadas en el proceso se detallan de la siguiente manera:

Lenguajes de Programación

1. **TypeScript**
   Angular se basa en TypeScript, un lenguaje de programación tipado que extiende JavaScript. TypeScript ofrece varias ventajas sobre JavaScript

estándar, como la detección temprana de errores a través del tipado estático y un mejor soporte para el desarrollo de aplicaciones a gran escala.
La tipificación estricta de TypeScript permitió estructurar el código de manera clara y evitar errores comunes, lo que facilitó el mantenimiento y la evolución de la aplicación a largo plazo.

Figura 23. Tipado del objeto “Inventario”

En la Fig. 23 se muestra un ejemplo de las varias tipificaciones que se utilizaron en los diferentes componentes, en este tipado se puede observar los atributos del componente Inventario y lo que permite que se pueda acceder de una manera sencilla y eficaz a las propiedades, para posteriormente utlizarlo en su CRUD.

2. **HTML5**

   HTML5 fue utilizado para estructurar las vistas y elementos visuales de la aplicación. Esta versión de HTML incluye diversas mejoras que permitieron crear interfaces más dinámicas e interactivas. Los elementos semánticos de HTML5 contribuyeron a una mejor accesibilidad y organización del contenido visual. (ver Fig. 24)

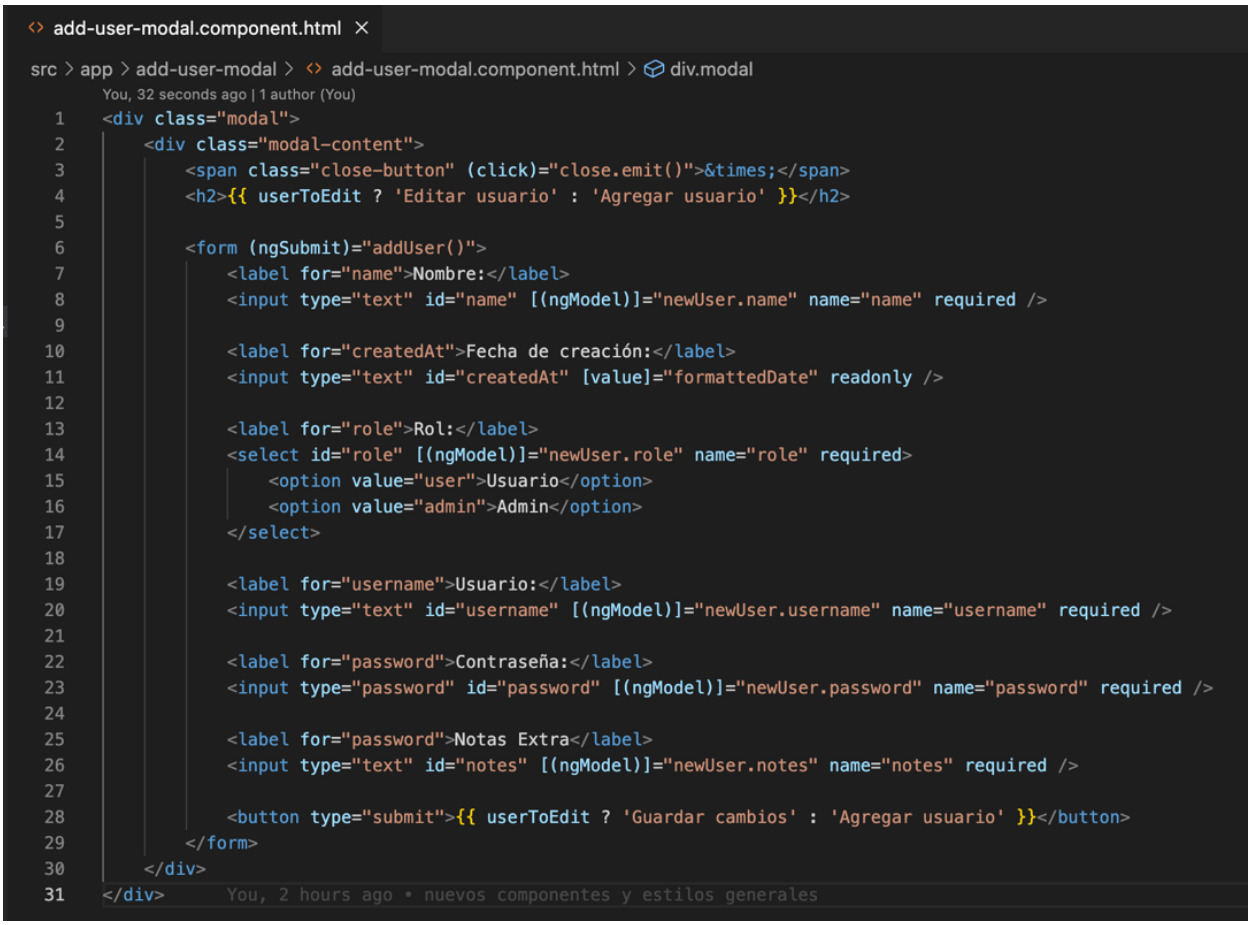

Figura 24. Ejemplo de HTML (Componente modal - agregar usuario)

3. **CSS3**

   CSS3 fue utilizado para diseñar y estilizar la interfaz de usuario (UI), brindando una experiencia de usuario más atractiva y responsiva. Se emplearon técnicas modernas como el uso de **Flexbox** y **Grid** para optimizar el diseño de la interfaz, garantizando que se adapte adecuadamente a dispositivos móviles y tablets, asegurando así un diseño responsive.

   Si bien la aplicación fue diseñada para computadoras y/o laptos, también se podrá abrir la página en un dispositivo móvil, garantizando que la interfaz se vea de una manera estética y entendible.

Figura 25. Vista desde una laptop

Como se puede notar en la Fig. 25, la técnica de Flexbox y Grid permite ubicar los elementos de acuerdo al tamaño de la pantalla.

En la Fig. 26, se muestra la simulación de la interfaz de como se veria si se abre la página web en un navegador desde un dispositivo movil.

Figura 26. Simulación de vista desde una dispositivo movil

Hay dos maneras de usar los estilos:

a) Hoja de estilos del componente

Esta hoja de estilos solo le afectará al componente en el que se esta trabajando, para este ejemplo, los estilos que se declaren en **inventory.component.css** solo afectará a la hoja **inventory.component.html.**

b) Hoja de estilos global

Esta hoja por lo contrario, afectará a cualquier componente en donde se mande a llamar un estilo especifico, esta hoja de estilos ayuda a que el codigo sea mantenible y más fácil de encontrar ya que evita la redundancia. En este componente “inventario” se uso los estilos globales para los botones de editar y eliminar.

Frameworks y Bibliotecas

1. **Angular**

   Angular fue el framework principal utilizado para el desarrollo de la aplicación. Este framework de desarrollo frontend, basado en componentes, permitió la creación de una aplicación escalable, modular y fácil de mantener, como la creación de **componentes reutilizables**, lo que ahorra tiempo y esfuerzo en el desarrollo de aplicaciones. Los componentes encapsulan la lógica y la presentación, lo que facilita su implementación en diferentes partes de la aplicación.

2. **Bootstrap**

   Bootstrap es una biblioteca de componentes CSS y JavaScript que facilita la creación de interfaces responsivas. Esta herramienta proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y muchos otros elementos.

3. **Node.js y npm**

   Node.js se utilizó para la gestión del entorno de desarrollo, ejecutando tareas como la compilación del código TypeScript a JavaScript y la automatización de pruebas. Además, npm (Node Package Manager) fue fundamental para la gestión de dependencias y bibliotecas adicionales necesarias para el desarrollo, como Angular CLI y otros paquetes. (ver Fig. 27).

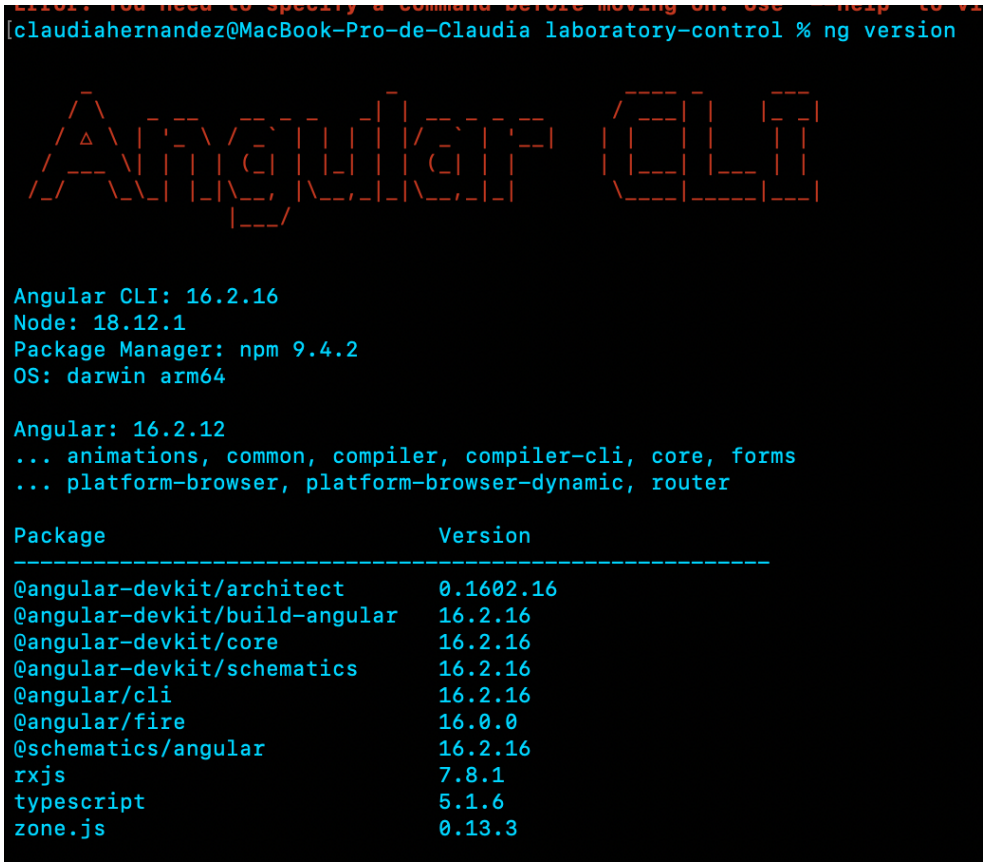

Figura 27. Versión de Angular y paquetes instalados

Base de Dato Firebase

La arquitectura de Firebase está formada por varios componentes, que funcionan en conjunto para proporcionar un marco y una infraestructura para desarrollar y alojar aplicaciones web y móviles.

Cloud Storage: también llamado Firebase Storage, es un servicio de almacenamiento de objetos que se ofrece en Google Cloud Platform. Cuando Google Cloud Storage se incorpora a las aplicaciones de Firebase, obtiene acceso a las medidas de seguridad de Google y la capacidad de proteger cualquier carga o descarga en su aplicación. A través del SDK, también puede administrar sus medios y acceder a ellos directamente desde su cuenta de almacenamiento.

Cloud Firestore: un servicio de base de datos flexible y escalable para el desarrollo de servidores, web y dispositivos móviles. Sirve como base de datos de documentos

NoSQL. Puede usarlo para almacenar, consultar y sincronizar los datos de su aplicación.

La base de datos **Firestore** de **Firebase** fue seleccionada como la base de datos NoSQL en tiempo real para almacenar toda la información del laboratorio, como inventario, control de visitas, usuarios para acceder a la pagina web, y control de prestamos de equipos existentes dentro del laboratorio.

Herramientas de Desarrollo

1. **Visual Studio Code**

   El editor de código utilizado para el desarrollo fue **Visual Studio Code**. Esta herramienta ofrece un entorno ligero pero potente, con múltiples extensiones para soportar el desarrollo en Angular y TypeScript, facilitando la depuración, el autocompletado de código y la integración con Git.

2. **Git y GitHub**

   El control de versiones fue gestionado mediante Git, lo que permitió llevar un registro detallado de los cambios en el código y colaborar eficientemente con otros desarrolladores. El repositorio del proyecto se alojó en GitHub, lo que facilitó la gestión de versiones, la resolución de problemas mediante issues y el manejo de ramas para las diferentes etapas de desarrollo.

3. **Karma y Jasmine**

   Karma y Jasmine fueron las herramientas empleadas para las pruebas unitarias. Estas herramientas permitieron verificar el correcto funcionamiento de los componentes de Angular y garantizar que la aplicación cumpla con los requisitos técnicos mediante la ejecución de pruebas automatizadas.

Metodología de Desarrollo

SCRUM

Para gestionar el desarrollo de la aplicación se utilizó la metodología ágil SCRUM (Fig. 28). El desarrollo se organizó en 4 sprints con duración de 3 semanas cada uno, lo que permitió entregar funcionalidades incrementales y recibir retroalimentación constante por parte de los usuarios del laboratorio. Esta metodología facilitó la adaptación a cambios y la mejora continua del sistema.

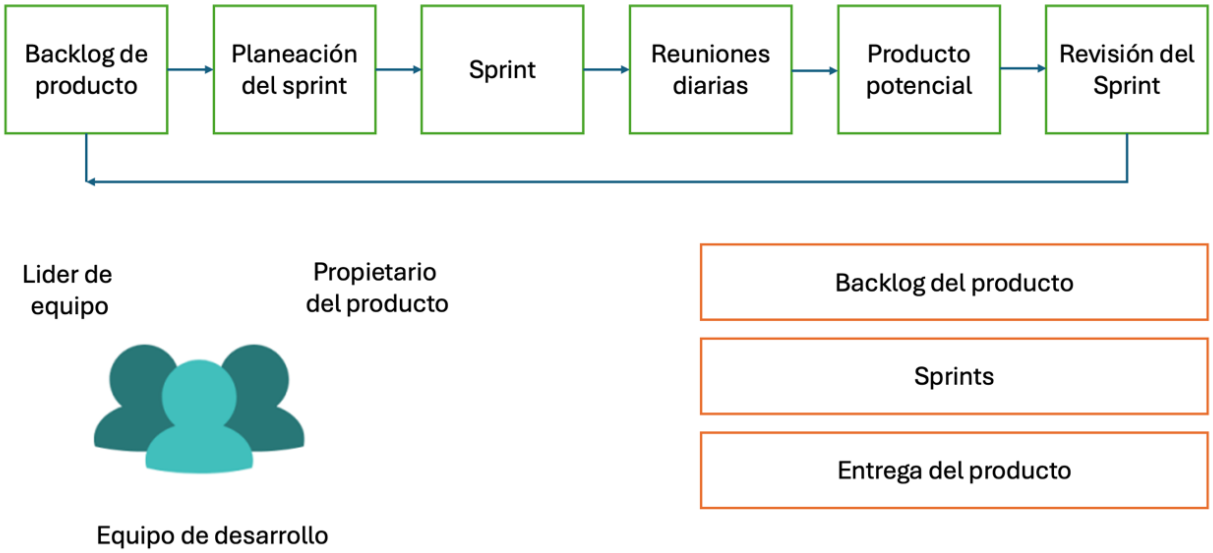

Figura 28. Esquema método SCRUM

4.3.3 Arquitectura y estructura del código

La arquitectura y estructura del código de la aplicación móvil, se diseñaron siguiendo las mejores prácticas del desarrollo basado en componentes que Angular ofrece. Se optó por una arquitectura modular y escalable, con una clara separación de responsabilidades para facilitar el mantenimiento y evolución de la aplicación.

Lo siguiente son los elementos principales de la arquitectura y la organización del código.

Estructura de la Aplicación por Módulos

Angular utiliza un enfoque modular que permite dividir la aplicación en varias secciones independientes, conocidas como módulos. Esta modularidad facilita el mantenimiento, la escalabilidad y el reuso de componentes. La estructura del proyecto se organizó en varios módulos principales, cada uno con una función específica en la gestión del laboratorio:

- **AppModule.ts**

  Este archivo actúa como el módulo principal de la aplicación. En AppModule.ts, se centraliza la configuración inicial, la declaración de los componentes principales, y la importación de otros módulos necesarios para el correcto funcionamiento de la aplicación. Aquí se encuentran registradas todas las dependencias globales que serán utilizadas a lo largo del proyecto.

  - Importa los módulos necesarios como BrowserModule, FormsModule, y otros específicos.
  - Declara los componentes de la aplicación que serán gestionados por Angular, como **AppComponent**.
  - Configura los servicios globales y cualquier dependencia externa utilizada en la aplicación.

- **AppComponent.ts**

  El archivo AppComponent.ts contiene la lógica principal de la aplicación. Este componente actúa como el controlador global que gestiona el estado general de la aplicación y sirve como punto de interacción entre el usuario y la aplicación.

  - Define las propiedades y métodos que gestionan los datos y la lógica de la vista.
  - Implementa las interacciones del usuario, como la captura de eventos (clics, formularios).

- Se asegura de que la aplicación responda de manera adecuada a las diferentes interacciones y gestiona la actualización de la vista en consecuencia.

- **AppComponent.html**

El archivo AppComponent.html define la estructura de la interfaz de usuario que verá el usuario al interactuar con la aplicación. Este archivo está vinculado directamente al AppComponent.ts para representar dinámicamente los datos gestionados por este último.

- Contiene el HTML que organiza la interfaz de la aplicación.
- Se incluyen directivas de Angular como *ngFor o *ngIf para manipular la presentación de datos en función del estado actual de la aplicación.
- Actúa como contenedor principal del contenido y los componentes hijos.

- **AppRoutingModule.ts**

AppRoutingModule.ts es responsable de definir las rutas de navegación dentro de la aplicación. Aquí se gestionan las diferentes vistas o secciones de la aplicación, permitiendo al usuario moverse entre ellas sin recargar toda la página.

- Configura las rutas principales de la aplicación.
- Define qué componentes se deben cargar en función de la URL visitada.
- Implementa un sistema de **guards** para proteger rutas restringidas que requieren autenticación.

- **SRC**

El directorio src es el corazón del proyecto, donde se encuentran todos los archivos y recursos necesarios para el desarrollo y despliegue de la aplicación. Dentro de este directorio, se almacenan todos los componentes, servicios,

módulos y otros recursos, organizados en subcarpetas para facilitar la estructura del código.

- **app/**: Contiene todos los componentes y módulos de la aplicación.
- **assets/**: Directorio donde se almacenan los recursos estáticos como imágenes, íconos, y archivos de configuración.
- **environments/**: Aquí se definen los archivos de configuración del entorno (por ejemplo, producción o desarrollo).

Componentes y Servicios

En Angular, los componentes y servicios son piezas clave de la estructura del código. Cada componente tiene una función específica y gestiona una parte de la interfaz de usuario, mientras que los servicios manejan la lógica de negocio y las interacciones con la base de datos. Esta separación ayuda a mantener el código limpio y fácil de probar.

Existen los componentes que cada uno tres partes principales:

1. Template HTML: Define la estructura visual del componente.

2. Archivo CSS/SCSS: Contiene el estilo visual que se aplica al componente.

3. Clase TypeScript: Gestiona la lógica del componente, incluyendo el manejo de eventos y la manipulación de datos.

Los servicios son clases que contienen la lógica de negocio y gestionan la interacción con las APIs y la base de datos. Los servicios de Angular usan la inyección de dependencias para proporcionar datos a los componentes. Por ejemplo, el servicio UserService gestiona todas las operaciones relacionadas con los usuarios, como la creación, lectura, actualización y eliminación de usuarios en Firestore.

Gestión de Rutas

La navegación dentro de la aplicación se maneja mediante el sistema de enrutamiento de Angular, que permite cambiar de vista sin recargar toda la página. La aplicación utiliza un archivo app-routing.module.ts donde se definen las rutas que vinculan las vistas (templates) con los componentes correspondientes. Cada ruta corresponde a una sección de la aplicación, como se muestra en la Fig. 29

```
const routes: Routes = [
  { path: '', component: LoginComponent }, // Ruta de login
  { path: 'home', component: HomeComponent },
  { path: 'visitas', component: VisitsComponent, data: { title: 'Visitas' }  },
  { path: 'inventario', component: InventoryComponent, data: { title: 'Inventario' } },
  { path: 'controlequipos', component: EquipmentControlComponent, data: { title: 'Control de equipos' } },
  { path: 'staffroles', component: UserListComponent, data: { title: 'Usuarios' } },
  { path: '', redirectTo: '/login', pathMatch: 'full' }, // Ruta por defecto
];
```

Figura 29. Gestión de rutas

Además, se implementó la funcionalidad de lazy loading para cargar los módulos sólo cuando son necesarios, optimizando así el rendimiento de la aplicación.

Comunicación entre Componentes

La comunicación entre componentes de Angular se gestiona de diferentes maneras, según la jerarquía de los componentes. En la aplicación, se utilizó:

- @Input() y @Output(): Para la comunicación entre componentes padre e hijo. Los componentes hijo reciben datos desde el componente padre a través de @Input(), y el componente padre recibe eventos del hijo mediante @Output().

- Servicios Compartidos: Para la comunicación entre componentes no relacionados, se utilizaron servicios compartidos que almacenan y gestionan el estado común entre ellos. Por ejemplo, el servicio StorageService gestiona el estado del inventario, permitiendo que diferentes componentes accedan y actualicen los datos de manera coherente.

Seguridad y Autenticación

Se implementaron guardas de rutas (Route Guards) para restringir el acceso a ciertas secciones de la aplicación según el rol del usuario (administrador o usuario).

- Guards de Autenticación: Estos guards verifican si el usuario está autenticado antes de permitirle acceder a rutas específicas.

- Roles y Permisos: Se implementó un sistema de roles en Firebase para definir permisos de acceso, asegurando que solo los usuarios autorizados puedan acceder a la función de administar los usuarios.

4.4 Integración de la base de datos

4.4.1 Diseño del Modelo de Datos

El diseño de modelo de datos en Firebase se basa en colecciones y subcolecciones, con esta base el modelo de datos se diseño de la siguiente manera

Control de usuarios

Nombre: Nombre del usuario a dar de alta
Fecha de creación: Fecha en la que se esta creando el usuario
Usuario: Nombre que va a usar para poder inicar sesión
Contraseña: Contraseña para iniciar sesión
Rol: Admin o User para las diferentes acciones dentro de la página
Notas: Notas extras como por ejemplo, el horario en el que se puede encontrar a la persona con fines informativos.

Control de inventario

Nombre: Nombre del equipo que se esta registrando
Descripción: Una breve descripción del equipo para poder identificarlo
Imagen: Archivo alojado en el Storage, que una vez en la nube se mostrará en la lista del inventario

Control de visitas

Nombre: Nombre de la persona que entra al laboratorio

Fecha: Fecha en la que igresa

Hora de entrada: Hora en la que llega

Hora de salida: Hora en la que sale

Actividad: Actividad que va a realizar dentro del laboratorio

Notas: Notas que funcionan para una mejor comunicación

Control del préstamo de equipos

Nombre: Nombre de la persona que recibe el equipo

Fecha de prestamo: Fecha en la que sale del laboratorio

Fecha de devolucion: Fecha en la que se recibe el equipo

Estado del equipo en la entrega: Condiciones en la que se entrega el equipo

Estado del equipo en la devolución: Condiciones en las que se devuelve

Imagen del equipo: Imagen de referencia del equipo a prestar

Notas extra: Notas que pueden ayudar con el mantenimiento del equipo

4.4.2 Conexión entre Frontend y Backend

Para poder conectar la base de datos con el proyecto se necesita hacer la instalación de angular firebase, esta instalación se realiza con el comando:

```
npm install @angular/fire firebase
```

Posteriormente se abre el archivo **environment.ts**, que se encuentra en la carpeta src/environments/environment.ts y se agregan las llaves de Firebase que se generan al momento de crear una base de datos dentro del mismo. Dentro de esta configuuración se puede modificar el entorno en el que se esta trabajando (producción o desarrollo)

Por último se debe hacer la importación de la librería instalada, esta importación se hace en el archivo app.module.ts que se encuentra en src/app/app.module.ts (ver Fig. 30)

```
import { environment } from '../environments/environment';
import { AngularFireModule } from '@angular/fire/compat';
import { AngularFireAuthModule } from '@angular/fire/compat/auth';
import { AngularFirestoreModule } from '@angular/fire/compat/firestore';
      You, 12 months ago • initial commit …
You, 3 days ago | 1 author (You)
@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    LoginComponent,
    MenuComponent,
    VisitsComponent,
    InventoryComponent,
    EquipmentControlComponent,
    UserListComponent,
    AddUserModalComponent,
    ModalConfirmationComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    AngularFireModule.initializeApp(environment.firebaseConfig),
    AngularFireAuthModule,
    AngularFirestoreModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 30. Importación de librerias Firebase en componente app.modules.ts

La ventaja de poder tener en un solo archivo la configuración del entorno en que se va a trabajar es: poder cambiar las configuraciones sin necesitad de modificar el código fuente.

Una vez instalada y declarada la librería en el modulo principal ya se puede hacer uso de ella en cualquier componente que se solicite, de la siguiente manera:

En la Fig. 31, se resalta en amarillo la manera de importar una de las liberias de Firestore para posteriormente poder usar sus funciones

```
import { Component, EventEmitter, Input, Output } from '@angular/core';
import { AngularFirestore } from '@angular/fire/compat/firestore';
import { Timestamp } from 'firebase/firestore';

You, 1 second ago | 1 author (You)
@Component({
  selector: 'app-add-user-modal',
  templateUrl: './add-user-modal.component.html',
  styleUrls: ['./add-user-modal.component.css']
})
export class AddUserModalComponent {
  newUser = {
    name: '',
    createdAt: Timestamp.now(),
    role: '',
    username: '',
    password: '',
    notes: ''
  };

  formattedDate: string = '';

  @Output() close = new EventEmitter<void>();
  @Output() userAdded = new EventEmitter<void>();
  @Input() userToEdit: any = null;

  constructor(private firestore: AngularFirestore) {
    this.formattedDate = this.getFormattedDate(new Date());
  }      You, 5 days ago • nuevos componentes y estilos generales
```

Figura 31. Importación de librerias Firebase en un componente especifico

Una vez importada y declarada la variable en el constructor, podemos hacer uso de la base de datos, por ejemplo, en la Fig. 32 se muestra la función addUser, que se

encarga de agregar o editar el usuario elejido, esta elección se hará dependiendo de la variable userToEdit.

Si userToEdit es verdadero entrara a la función update la cual se compone de:

- this.firestore: se llama a la variable declarada en el constructor
- collection('users'): hace referencia a la colección users en la base de datos Firestore.
- doc(this.userToEdit.id): se obtiene el ID de ese usuario de la base de datos para tener referencia de que user se modificará

- update(objeto): Update se encarga de editar ese usuario con las nuevas variables

De esta manera se hace referencia a la colecciones en la base de datos.

```
async addUser() {
  if (this.userToEdit) {
    try {
      await this.firestore.collection('users').doc(this.userToEdit.id).update({
        name: this.newUser.name,
        createDate: this.newUser.createdAt,
        rol: this.newUser.role,
        user: this.newUser.username,
        password: this.newUser.password,
        notes: this.newUser.notes
      });
      //console.log('Usuario actualizado con éxito');
      this.userAdded.emit();
      this.close.emit();
    } catch (error) {
      //console.error('Error al actualizar usuario:', error);
    }
  } else {
    try {
      await this.firestore.collection('users').add({
        name: this.newUser.name,
        createDate: Timestamp.now(),
        rol: this.newUser.role,
        user: this.newUser.username,
        password: this.newUser.password,
        notes: this.newUser.notes
      });
      //console.log('Usuario agregado con éxito');
      this.userAdded.emit();
      this.close.emit();
    } catch (error) {
      //console.error('Error al agregar usuario:', error);
    }
  }
}      You, 5 days ago • nuevos componentes y estilos generales …
```

Figura 32. Uso de Firebase en el proyecto

4.5 Pruebas

Las pruebas realizadas para esta aplicación web se hicieron con personal del laboratorio, en la que ellos hicieron uso de esta y posteriormente dar su feedback.

4.5.1 Tipos de Pruebas Realizadas

Pruebas Funcionales

Las pruebas funcionales se realizaron para asegurar el correcto funcionamiento de la aplicación web y para validar que el comportamiento haya sido el esperado.

Para estas pruebas el personal hizo uso de las diferentes funciones que aparecen en las secciones, por ejemplo el CRUD de cada una de ellas, es decir, se comprobo la Creación, Lectura, Actualización y Eliminación correcta en la base de datos de Firestore.

Pruebas de Interfaz de Usuario (UI)

Se evaluó la interfaz de usuario para garantizar que fuera intuitiva y fácil de usar.
Los usuarios realizaron un feedback sobre la posición de los elementos, la coherencia del diseño y la maner de navegar en la aplicación, a partir de estos comentarios se realizaron cambios en el diseño para poder mejorar su experiencia como usuarios.

Pruebas de Usabilidad

Este tipo de pruebas se centraron en evaluar la facilidad con la que los usuarios podian interactuar con la página, abriendo la página en diferentes dispositivos (ordenadores y móviles) para verificar que esta fuera accesible y fácil de usuar sin necesidad de alguna capacitación.

Pruebas de Seguridad

Se llevaron a cabo pruebas de seguridad para garantizar que los datos sensibles de los usuarios, como credenciales y roles, estuvieran protegidos. Se verificó la correcta implementación de los roles para los diferentes accesos a las funcionalidades en la sección de usuarios, ya que un usuario que tiene el rol como (user) solo podrá usar los datos como informativos.

Pruebas de Rendimiento

Las pruebas de rendimiento nos ayudaron a evaluar el comportamiento de la página con distintas cargas de trabajo, simulando el uso con varios usuarios a la vez y así comprobar que el tiempo de respuesta fuera el adecuado.

Pruebas de Compatibilidad

Se probó la aplicación en diferentes plataformas, navegadores y resoluciones de pantalla para garantizar su correcta visualización y funcionamiento en diversos entornos. Esto incluyó pruebas en navegadores como Chrome, Firefox y Edge, así como en sistemas operativos como Windows y Android.

4.5.2 Resultados de las Pruebas

Durante las pruebas iniciales en cuanto a funcionalidad, se detectaron errores menores relacionados con el CRUD en tiempo real, el principal error en la mayoria de las secciones fue el Delete, ya que no se hacia la referencia correcta al elemento a eliminar por no obtener el ID correctamente, estos errores se corrigieron durante la marcha y al final fue validada satisfactoriamente.

En la interfaz de usuario se mencionó lo fácil que era usar y navegar en la página, sin embargo se recomendó mejorar el contraste de colores en cuanto al ecabezado de las tablas.

Como se puede ver en la Fig. 33, en la sección de Visitas, el color azul daba un contraste con el color morado del título principal de la página, esto gneraba en los usuarios un poco de ruido visual.

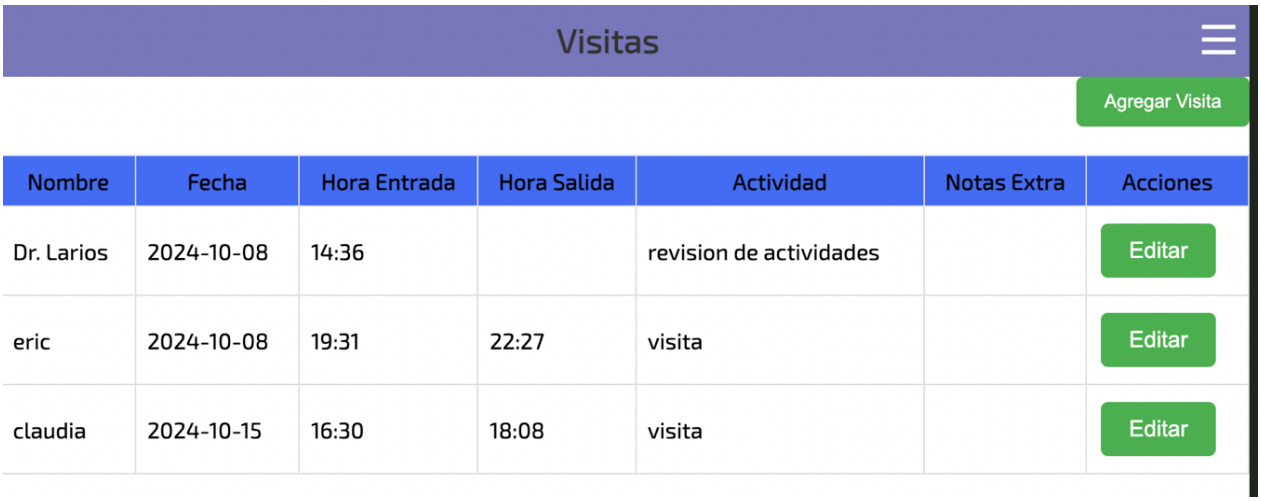

Figura 33. Interfaz de usuario en la sección de visitas (antes)

Por lo que se optó por cambiar el color a uno neutro como se muestra en la Fig. 34

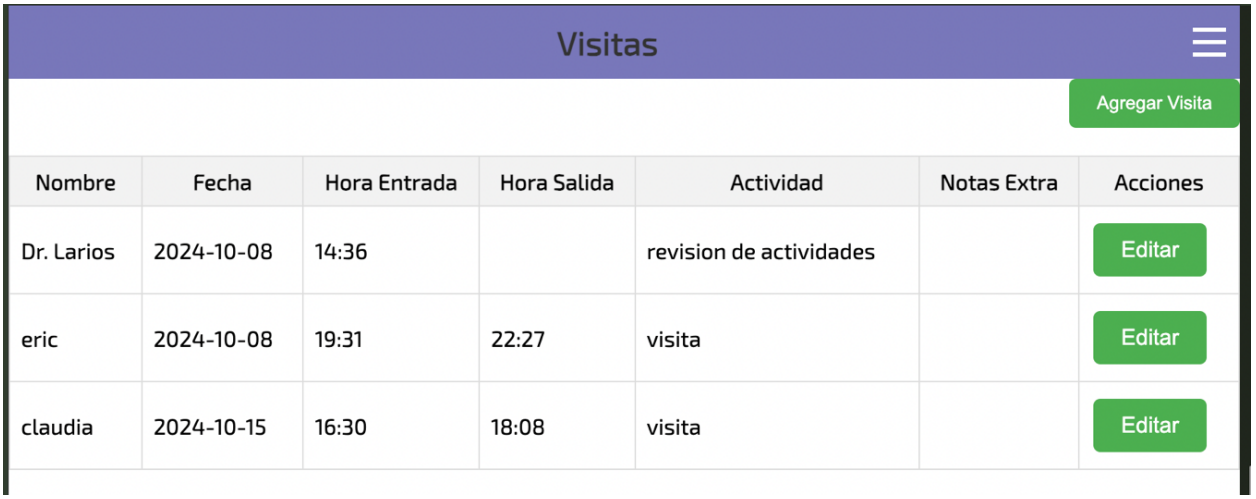

Figura 34. Interfaz de usuario en la sección de visitas (después)

Con este cambio el usuario pudo identificar mejor los componentes de esta sección.

Para las pruebas de usabilidad se recibio una buena respuesta por parte de los usuarios ya que mencionaron que pudieron completar las tareas clave como el registro de equipos en el inventario, la eliminación de ellos y la opción de modificar los elementos, de igual manera en las diferentes secciones de la página.

Hubo algunas sugerencias en cuanto a la mejoria del modo responsivo cuando se mostraban las secciones de tablas cuando la página se abria en un dispositivo movil.

Y en general el 90% de los usuarios encontraron la aplicación fácil de entender tras las mejoras.

En el tema de las pruebas de seguridad no hubo mayor inconveniente ya que como se ha mencionado antes, la única sección afectada para mostrar diferente información dependiendo del rol del usuario que inicia sesión es la de “Usuarios”.
Esta funcionalidad se corrigio en el momento de estar programando ya que se detecto que este rol no se guardaba cuando se recargaba la página, los usuarios lo dieron por valido ya que a ellos se les entrego la versión con esta parte corregida.

El rendimiento de la página fue un tiempo promedio de respuesta de 3 segundos cuando los usuarios conectados simultaneamente eran de 5 a 8 usuarios. Cuando los usuarios conectados eran más de 10 y haciendo operaciones en la página los tiempos de respuesta se vieron ligeramente elevados, para resolver esta parte se optimizó el codigo y las consultas a Firestore para reducir la carga de datos en cada solicitud.

La aplicación funcionó correctamente en todos los navegadores y dispositivos probados, incluidos Chrome, Firefox y Edge, tanto en entornos de escritorio como móviles. Se hicieron pequeños ajustes para mejorar la visualización en tablets, pero en general, la compatibilidad fue satisfactoria. La aplicación ahora se adapta de manera óptima a diferentes resoluciones de pantalla sin comprometer la experiencia del usuario.

4.6 Validación y evaluación

4.6.1 Evaluación con Usuarios Finales

Se llevó a cabo una evaluación de usabilidad con 5 usuarios finales, personal del laboratorio, para determinar el grado en que la aplicación desarrollada cumplía con sus necesidades y expectativas.
Estos usuarios realizaron las tareas ya antes mecionadas, en general el CRUD de las secciones que tiene la página (ver Fig. 35)

Figura 35. Menú de opciones

Los resultados indicaron una alta aprobación en general con la interfaz de usuario, fue calificada como amigable, intuitiva y fácil de entender.
Sin embargo, hubieron algunas sugerencias

1. Un apartado en el que se viera el usuario que en ese momento esta logueado,
2. Un botón que permita la extracción de información en un excel
3. Una sección en la que se pueda ver los trabajos que han hecho los usuarios dentro del laboratorio.

Estas sugerencias serán consideradas en futuras iteraciones del desarrollo.

En conclusión, la evaluación de usabilidad revela que la aplicación cumple con los requisitos funcionales y de usabilidad establecidos, proporcionando una buena experiencia de usuario y demostrando ser una herramienta útil para la gestión del inventario y control de accesos en el laboratorio.

Capítulo 5. Análisis de Resultados y Discusión

5.1 Interpretación de los Resultados

La implementación de esta página web para la gestión de un laboratorio tuvo una gran aceptación por parte de los usuarios finales, obteniendo así los siguientes resultados:

Uno de los objetivos principales del sistema era mantener y controlar el flujo en un laboratorio, los resultados obtenidos mostraron que tras la implementación de esta página web, el tiempo de administración y seguimiento de visitas al día tuvo un impacto positivo en comparación a llevar un control manual, esto indica que se cumplió con la función de facilitar la carga administrativa obteniendo un mejor flujo de registros.

De igual manera el desarrollo y modelo propuesto para tener un mejor manejo y cuidado de los equipos que entran y salen de un laboratorio, redujo en un 90% las complicaciones que se tenian al tener la información en papel y que en ocasiones se llegará a traspapelar. Este objetivo en especifico ayudo a tener un conocimiento más a detalle de los equipos más utilizados en el laboratorio para en un futuro poder tener más stock, y tener en cuenta el mantenimiento de cada uno de ellos.

La retroalimentación de los usuarios finales obtenida a traves de las criticas constructivas y sugerencias en las diferentes etapas del desarrollo ayudó mucho para obtener una versión funcional de este modelo propuesto, los usuarios mencionaron lo fácil de usuar la página y la intuitividad de ella, esto indica que la interfaz y la arquitectura propuesta y diseñada son los adecuados para las necesidades del personal del laboratorio.

La base de datos elegida, Firebase, fue una parte fundamental del sistema para asegurar que los datos estuviesen protegidos, ya que, esta herramienta proporciona medidas de seguridad robustas como la autenticación segura para los datos. El uso de estas dos tecnologias en conjunto (Angular y Firebase) permite que el sistema sea escalable y adaptable a las futuras necesidades de un laboratorio o una empresa, durante las pruebas de carga se obtuvo el resultado de que el sistema era capaz de manejar multiples solicitudes simultaneas sin afectar en gran parte el rendimiento, lo que indica que se puede soportar un cierto aumento de usuarios sin afectar la funcionalidad.

En términos generales, los usuarios aprobaron con una alta satisfacción en el sistema, por los factores ya mencionados y con ello la posibilidad de aumentar funciones y/o ajustes en un futuro.

5.2 Comparación con Trabajos Previos

En comparación con sistemas previos, el uso de un framework como Angular permitió la creación de una interfaz amigable y fácil de usar, teniendo asi una mejor navegación sin necesidad de alguna capacitación. Algunos sistemas anteriores eran complejor y poco accesibles, lo que causaba tener un mayor tiempo de adaptación

En el trabajo de Chiluisa Pallo et al., (2014), menciona lo siguiente:

- El sistema no permite registrar más de un usuario administrador

- Su registro de un estado de algún equipo del laboratorio no permite registrar otro estado que no sea: funcional, reparado, dañado
- Para el estado de uso no permite registrar otro más que: en uso, dado de baja y devuelto

La página web propuesta en este trabajo, si permite la creación de más de un usuario como administrador, esto con la finalidad de tener un personal más amplio para que el trabajo del laboratorio sea más ameno, en cuanto al registro de un equipo, se puede tener el estado deseado, este es un campo sin opciones ya que en ocasiones se necesita más información sobre el equipo, adicional se cuenta con un apartado de notas extras para un mejor detalle.

5.3 Limitaciones del Proyecto

Una parte fundamental que los usuarios finales hicieron notar, es el uso de los reportes, si bien el sistema permite la eliminación de elementos en cuanto al inventario y préstamos de equipos, es importante contar con un registro para tener una información más precisa, por ejemplo, obtener la fecha en la que estuvo un elemento por ultima vez dado de alta en el sistema, o llevar con control sobre el prestamo de equipos.

A pesar de los avances y mejoras propuestas en el desarrollo, es importante considerar algunas limitaciones que puede causar un bajo desempeño y en un porcentaje bajo la escalabilidad.

1. Dependencia de la conectividad a Internet: La página esta fuertemente ligada a servicios en la nube, lo que implica una conexión buen y estable a Internet, esto puede ser una limitante en entornos donde la conectividad es inestable o inexistente.
2. Costos en Firebase: Aunque Firebase ofrece una uso gratuito en el nivel básico, a medida que los usuarios crecen también aumenta el costo del uso de esta herramienta. Para un laboratorio con un número de usuarios

limitados, el costo es manejable; sin embargo, si el sistema se plantea en un entorno con más flujo es necesario tener el cuenta el costo para usar Firebase como base de datos.

3. Adaptabilidad a otros entornos: El sistem fue propuesto y diseñado para usarse en un laboratorio de software y en un futuro en una empresa, por lo que es posible adaptarlo a entornos más grandes, teniendo en cuenta la realización de ajustes en la configuración y/o estructura de datos.

4. Experiencias de usuario: La interfaz fue aprobada por los usuarios de una manera satisfactoria, eso no garantiza que algunas interacciones o animaciones avanzadas se puedan implementar en el proyecto con Angular, esto puede causar una limitante en cuanto a la experiencia en comparación a una aplicación móvil o de escritorio más avanzada.

Capítulo 6. Conclusiones

6.1 Conclusiones

La automatización de tareas relacionadas con el registro de equipos en el inventario ha demostrado ser un recurso importante y favorable para el personal del laboratorio, esto ha ayudado y reducido el tiempo dedicado a obtener la información manualmente, mejorando la operatividad dentro del laboratorio.

En conclusión, la implementación del sistema ha tenido un buen impacto en cuanto a las mejoras del seguimiento de los equipos que se prestan, ya que es más fácil identificar cuanto tiempo se usa, el estado en el que entra y sale del laboratorio y las personas que usan esos dispositivos. A pesar de las limitaciones mencionadas el proyecto muestra un gran potencial de soluciones basadas en la nube para optimizar el trabaho y la gestion de recursos en un ambito educativo y empresarial. sariales.

6.3 Aplicaciones Futuras

El sistema propuesto para la gestión de un laboratorio tiene el potencial de implementarse en diversos ambitos, como por ejemplo:

1. **Modulo de Reportes**
   Un módulo de reportes avanzados permitiría ver y obtener un análisis detallado sobre el uso de recursos, ver áreas de oportunidad y mantener en

buen estado los elementos existentes, esto ayudaria a tomar desiciones más estrategicas permitiendo reducir costos de mantenimiento.

2. **Notificaciones y recordatorios automatizados**

   El sistema de notificaciones seria crucial para los recordatorios de entrega de algún equipo, esta implementación ayudaría a las personas para que puedan tener un mejor control sobre sus tiempos de uso, incluso este sistema puede usarse para otras aplicaciones como por ejemplo, la toma de algún medicamento importante, días de visitas con el doctor o algún amigo, recordar fechas importantes, todo esto de la mano con el objetivo que se quiere alcanzar.

3. **Escalabilidad para otros departamentos o instituciones**

   Con algunos ajustes y objetivos más especificos, este sistema se podría adaptar para el uso en bibliotecas, ya que permitiria gestionar los recursos en diferentes contextos y ámbitos.

4. **Integración con herramientas de empresas**

   Una futura integración con las empresas podría de igual manera facilitar el control y uso de la información de los recursos existentes y claro en diferentes departamentos, para esto se tendria que tomar en cuenta los alcances y objetivos para llegar al resultado esperado, y basado en el sistema propuesto en este proyecto se podrá tener una mejora para el uso de empresas más grandes.

5. **Control de acceso mediante código QR**

   Una mejora importante para el manejo de inventarios y visitas seria trabajar en conjunto con una aplicación móvil, este proceso seria, generar un QR por medio de la página web y este posteriormente ser enviado via correo al usuario que este solicitando algún prestamo, esto con la finalidad de optimizar aún más el flujo, ya que cuando este usuario devuelva el equipo

prestado solo mostraria su QR al encargado y asi poder obtener los datos de una manera más eficiente.

Estas aplicaciones y mejoras futuras ofrecen una visión diferente e importante de como poder adaptar el sistema propuesto en este proyecto, para asi podr aprovechar al máximo las nuevas tecnologias.

Bibliografías y Referencias

¿Qué es Angular? Características y ventajas. (2023, 20 enero). *Hubspot*. Internet https://blog.hubspot.es/website/que-es-angular

Chiluisa Pallo, A. P., & Loarte Cajamarca, B. G. (2014). *Desarrollo e implantación del sistema de control de inventarios y gestión de laboratorios para la de la facultad de Ciencias* (Bachelor's thesis, Quito: EPN, 2014.).

Computing, C. (2012). Que es "Computación en la Nube". *Recuperado el*, *6*.

Conceptos básicos de HTML - Aprende desarrollo web | MDN. (2024, 28 julio). MDN Web Docs. Internet https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

Jimenez Jaramillo, E. E. (2020). *DISEÑO DE UNA APLICACIÓN DE INVENTARIO PARA "LABORATORIOS HERBANASE"* (Bachelor's thesis, Instituto Superior Universitario Bolivariano de Tecnología).

López Maradiaga, A. R. (2021). *Desarrollar una aplicación móvil para SO Android con API 23 en adelante, con el fin de controlar la asistencia del personal de la Universidad Nacional Autónoma de Nicaragua, León, en el periodo de marzo a octubre del 2021* (Doctoral dissertation).

Macías, H. J. R., Lara, R. A. M., & Cabrera, C. S. P. (2020). Aplicación Web para Control del Inventario de Laboratorios de Computación. *Dominio de las Ciencias*, *6*(3), 1422-1443.

Makoveev Routskaia, E. (2024). *Clout: una red social para llevar un seguimiento de las prendas de ropa que una persona utiliza* (Doctoral dissertation, Universitat Politècnica de València).

Medina Gutiérrez, C. M. (2022). *Desarrollo de aplicaciones de supervisión industrial de código abierto para el laboratorio de Redes Industriales: diseño e implementación de un interfaz de operador industrial web usando software de código abierto* (Bachelor's thesis, Quito: EPN, 2022.).

Puciarelli, L. (2020). *Angular: TypeScript–Arquitectura–Instalación–Directivas y Bindings–Forms–Ruteo y más*. RedUsers.

Ramírez, M. R., Soto, M. D. C. S., Moreno, H. B. R., Rojas, E. M., Millán, N. D. C. O., & Cisneros, R. F. R. (2019). Metodología SCRUM y desarrollo de Repositorio Digital. *Revista Ibérica De Sistemas e Tecnologias De Informação*, (E17), 1062-1072.

Rodríguez Rodríguez, M. C. (2014). Configuración de software basada en metamodelos y modelos-configuración de software contable y gestión de recursos humanos a traves de construcción de metamodelos y modelos.

Traverso, M., Emilio, H., Prato, I., Beatriz, L., Villoria, M., Noemí, L., Rodriguez, G., Alfredo, G., Cristina, L. P., Caivano, I., Marcela, R., & Nacional, P. (2017). Herramientas de la Web 2 . 0 aplicadas a la educación Resumen Introducción.