Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

DESARROLLO DE UNA PLATAFORMA DE LABORATORIO REMOTO Y VIRTUAL MEDIANTE APLICACIÓN WEB

Dña. Barril Aller, Sofía
TUTOR: D. Navarro Rodríguez, Ángel

JULIO 2024

Resumen

El trabajo realizado a lo largo de este proyecto consiste en el desarrollo de una plataforma de laboratorio remoto y virtual mediante una aplicación web para un proyecto de innovación docente, financiado por el Instituto de Investigación en Innovación Educativa (INIE) de la Universidad de Oviedo.

La pandemia del COVID19 puso en manifiesto la necesidad de buscar alternativas a los laboratorios presenciales. Este proyecto se centra en la creación de una aplicación web para albergar dos de las alternativas de prácticas no presenciales más atractivas, el uso de laboratorios virtuales basados en simulación y el acceso a laboratorios remotos. Aparte de permitir su uso desde cualquier lugar y momento, también permitirá mejorar el aprendizaje de los estudiantes de Ingeniería, aunque se podrá extender a otras áreas.

En concreto, se les permite realizar de forma *online* distintas simulaciones que podrán reiniciar y parar en cualquier momento para tomar apuntes. También pueden interactuar con distintos experimentos mediante una conexión con equipos reales. Esta plataforma no solo aporta un aprendizaje de una forma práctica, sino que también hay contenidos teóricos relacionados con la práctica, de esta forma se garantiza un aprendizaje integro. Los profesores por su parte podrán incluir recursos en la aplicación de una forma flexible. El objetivo de este trabajo es implementar la aplicación web que permita estas funcionalidades mediante el uso de tecnologías modernas y eficaces.

Contenidos

1. Introducción

En titulaciones de ingeniería, tanto las simulaciones como las prácticas con equipos de laboratorio y la interacción del alumnado con procesos físicos y elementos reales son imprescindibles para asegurar una formación de calidad. No solo les permite asentar los conocimientos teóricos, sino que además permite mantener el enfoque práctico de la ingeniería, y mejorar la motivación. Esta necesidad se cubre en las titulaciones mediante prácticas de laboratorio y prácticas de aula, sin embargo, en muchas ocasiones la limitación de tiempo de estas sesiones limita el contenido que puede impartir el docente y no se ajusta a las necesidades de todos los estudiantes. Además, en muchas ocasiones la formación práctica se ve lastrada por la baja relación entre el número de equipamientos o licencias disponibles y el de estudiantes. Como respuesta innovadora a estas necesidades, en los últimos años han surgido una serie de proyectos de laboratorio accesibles de manera remota y laboratorios virtuales con simulaciones *online* que buscan la mejora en la calidad de enseñanza en titulaciones a distancia y presenciales [3-7].

Un laboratorio virtual es un espacio interactivo en línea donde se integran todos los aspectos tecnológicos, pedagógicos y humanos para facilitar el aprendizaje práctico de los estudiantes [1]. Estas plataformas permiten a los estudiantes aprender de diferentes temas mediante simulaciones, que pueden pausar y reiniciar para tomar apuntes, también hay distintos contenidos a los que pueden acceder en cualquier momento.

Un laboratorio remoto es un conjunto de *hardware* y *software* que permite al estudiante llevar a cabo un experimento como si estuvieran en un laboratorio presencial. Para ello el estudiante controla un recurso mediante una red interactuando así con un equipo real [2].

1.1.- Comparativa de laboratorios virtuales y remotos actuales

Los laboratorios en línea son una herramienta eficaz para ayudar a los estudiantes de ingeniería, a continuación, se enumeran distintos laboratorios virtuales, remotos o ambos que han ayudado a la educación.

- **Netlab** es un laboratorio remoto en la disciplina de circuitos electrónicos por lo que admite hacer mediciones y cableado. Está desarrollado en la tecnología Java. Este laboratorio virtual está siendo implementado por la Universidad de Australia. Permite llevar a cabo distintos experimentos mediante interfaces gráficas de usuario como si estuvieran interactuando con un equipo real mediante Internet [3].
- **Unilab** es un laboratorio remoto y virtual 3D en la disciplina de óptica y control. Las tecnologías usadas son: EjsS, Blockly, Chart.js, Moodle y LabVIEW. Está implementado por la UNED y está diseñado con una gran variedad de distintos experimentos sobre diferentes campos y temáticas de la Ingeniería de control [4].
- **WebLab-Deusto** es un laboratorio remoto y virtual 2D en la disciplina de electrónica y control. Esta desarrollado en tecnologías como Unity y WebGL. Está implementado por la Universidad de Deusto y algunas de sus funciones son acceder a laboratorios remotos, crear nuevos laboratorios e incluso adquirir laboratorios diseñados a medida [5].
- **Grid of *Online* Laboratory Devices Ilmenau (GOLDi)** es un laboratorio remoto y virtual 2D en la disciplina de control desarrollado en HTML5 y Java. Está implementado por la Universidad tecnológica de Illmenau y permite probar algoritmos de control propios en distintos sistemas virtuales [6].

Aunque cada uno de estos laboratorios están diseñados para distintas disciplinas y funcionalidades, aún hay algunos problemas que hay que considerar [7]:

- En primer lugar, se quiere una plataforma que tenga disponible estos experimentos que pueda cubrir todas las áreas, es decir, un marco unificado y experimentación híbrida. Esto permitiría que sean accesibles para más personas y permitir que todos los experimentos estén en un solo lugar por lo que facilita la experiencia del estudiante.

- Además, también se quiere que estas plataformas sean escalables, ya que de esta forma es más fácil añadir nuevos experimentos o actualizar la aplicación sin mucha dificultad.
- Otra mejora que considerar es la interfaz gráfica del usuario y la interactividad, ya que algunos de los laboratorios tienen experimentos que no son fáciles de comprender y de usar, por lo que esto permitiría que los estudiantes entendieran y manejaran mejor estos experimentos.
- Por último, habría que incorporar también contenidos teóricos no sólo prácticos de esta forma se ayudaría a los estudiantes a entender mejor los conceptos y fórmulas teóricas aplicándolo en la práctica.

2. Objetivos y alcance del proyecto

Este trabajo nace como un proyecto de innovación docente de la Universidad de Oviedo, llamado *VirtyRemLab*[1], cuyo objetivo es mejorar la formación híbrida en titulaciones de ingeniería de la Universidad de Oviedo, proponiendo la implementación de un laboratorio flexible virtual y remoto accesible de manera *online* mediante acceso web, enfocado en primera instancia a ingeniería eléctrica, electrónica industrial, sistemas de control y automática, que podrá ser extendido a otras áreas. La propuesta no sustituye a la metodología actual en estudios de ingeniería, sino que tiene una función complementaria para facilitar la formación de los estudiantes.

El objetivo específico de este trabajo es el desarrollo de una plataforma *online* para gestionar el acceso remoto a contenidos didácticos, simulaciones guiadas y prácticas experimentales de manera asíncrona y autónoma.

La arquitectura de esta aplicación web constará de un *Backend*, de un *Frontend* y de una base de datos. El *Backend* es el término utilizado en informática para referirse a la parte de programación del servidor, es decir, se encarga de recibir, procesar y devolver datos en un sitio web [8]. Mientras que el *Frontend* es la parte visible de una web, con la que el usuario interactúa. La base de datos es un conjunto de datos que se almacenan de forma electrónica en un sistema [9].

Algunas de las funcionalidades que tiene que cumplir esta arquitectura son la posibilidad de listar todos los contenidos, experimentos y simulaciones, poder visualizar uno en concreto e interactuar con las simulaciones. Aunque estas funcionalidades no son las únicas que ofrecerá la arquitectura.

[1] Convocatoria pública de ayudas en concurrencia competitiva para el desarrollo de proyectos de innovación docente 2024-2025, número de expediente: 177789.

Además de ayudar al estudiantado, se quieren mejorar los problemas de los laboratorios actuales descritos en el apartado anterior, cumpliendo así distintas características que se explican a continuación.

En cuanto al problema del marco unificado y experimental híbrido se requiere que sea una plataforma integral por lo que todos los experimentos, simulaciones y contenidos estarán disponibles en el mismo lugar. De esta forma la experiencia de los usuarios será más completa al no tener que cambiar de sitio web para encontrar los distintos recursos.

La aplicación web será escalable de esta forma se podrá añadir nueva materia de una forma sencilla. También cumplirá la modularidad para que cuando se añadan o se borren recursos no afecte al resto del sistema.

La interfaz gráfica del usuario será intuitiva y fácil de usar permitiendo a los usuarios que entiendan como utilizarla rápidamente. Además, esta plataforma contendrá tanto contenido educativo teórico como práctico, asegurando así un aprendizaje integro.

Los usuarios principales de este proyecto son el estudiante que utilizará los recursos que forman parte de la plataforma y los administradores que mantendrán actualizada esta web añadiendo nuevos materiales educativos.

Este proyecto se dividirá en tres componentes principales, que se explican a continuación.

El primer componente será la vista de contenidos, donde el usuario podrá ver listados los distintos materiales teóricos y el usuario administrador podrá subir nuevos contenidos PDF, imágenes o contenido eXeLearning, además de eliminarlos. Este último tipo de contenido se hace con una herramienta que facilita la creación de nuevos contenidos educativos de una forma sencilla sin ser experto en programación. Además de contenido teórico permite añadir también contenido interactivo. Esta herramienta fue implementada por distintas universidades entre ellas están la Universidad de Auckland y la Politécnica de Tairāwhiti [10].

El segundo componente será el apartado de simulaciones, donde el usuario podrá acceder a distintas simulaciones que estarán ordenadas por temáticas. Los administradores subirán estas simulaciones que estarán hechas mediante EasyJava.

Por último, está el componente del laboratorio remoto que permitirá a los estudiantes realizar experimentos reales de manera remota obteniendo datos en tiempo real a través de una conexión entre el servidor web y una *Raspberry,* que funcionará de servidor, mediante el protocolo MQTT.

Seguidamente se muestra un diagrama de flujo del funcionamiento de la aplicación web.

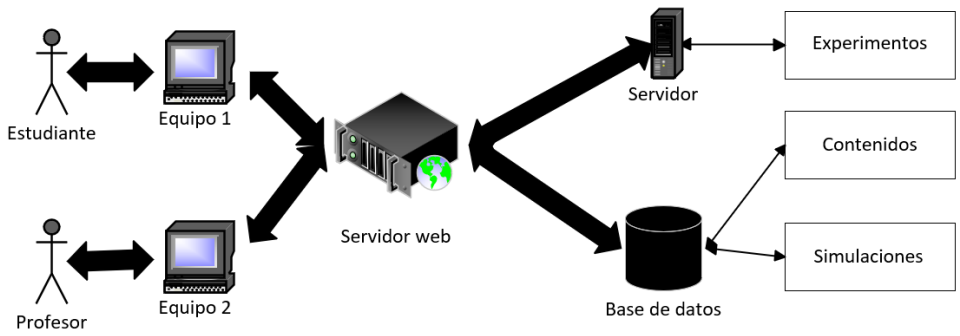

Figura 2.1.- Diagrama de funcionamiento de *VirtyRemLab*.

3. Análisis del proyecto

En este apartado se enumerarán todos los requisitos que debe de tener la aplicación web *VirtyRemLab.*

3.1.- Requisitos funcionales

Los requisitos funcionales definen como debe de comportarse un sistema para cumplir con los objetivos del proyecto, estos requisitos detallan las funcionalidades que el sistema debe de hacer. Este apartado se divide en los requisitos funcionales que tiene cada una de las capas de *VirtyRemLab.*

3.1.1.- Requisitos funcionales del *Frontend*

RF.F.1. El sistema permitirá el registro de nuevos usuarios.

RF.F.2. El sistema permitirá iniciar sesión a usuarios registrados.

RF.F.3. El sistema permitirá cerrar sesión a los usuarios registrados.

RF.F.4. El sistema mostrará el apartado de contenidos según el rol del usuario:

- RF.F.4.1. Estudiante
  - RF.F.4.1.1. El sistema mostrará los contenidos existentes de la plataforma ordenados por temática.
- RF.F.4.2. Administrador
  - RF.F.4.2.1. El sistema mostrará los contenidos existentes de la plataforma ordenados por temática junto a un botón de eliminar, para borrar el contenido.

RF.F.4.2.2. El sistema mostrará una vista para subir nuevos contenidos.

RF.F.5. El sistema mostrará el apartado de simulaciones según el rol del usuario:

RF.F.5.1. Estudiante

RF.F.5.1.1. El sistema mostrará las simulaciones existentes de la plataforma ordenadas por temática y nombre.

RF.F.5.2. Administrador

RF.F.5.2.1. El sistema mostrará las simulaciones existentes de la plataforma ordenadas por temática y nombre junto a un botón de eliminar, para borrar la simulación.

RF.F.5.2.2. El sistema mostrará una vista para subir nuevas simulaciones.

RF.F.6. El sistema mostrará el apartado de laboratorio remoto dependiendo del rol del usuario

RF.F.6.1. Estudiante

RF.F.6.1.1. El sistema mostrará un listado con los experimentos disponibles.

RF.F.6.2. Administrador

RF.F.6.2.1. El sistema mostrará los experimentos disponibles junto a un botón de eliminar para borrar el hipervínculo al experimento de la plataforma.

RF.F.6.2.2. El sistema mostrará una vista para subir nuevos hipervínculos a los experimentos.

RF.F.7. El sistema permitirá ver el perfil del usuario.

RF.F.7.1. El sistema permitirá editar el nombre y los apellidos del usuario.

RF.F.7.2. El sistema permitirá editar la contraseña.

RF.F.8. El sistema mostrará las reservas del usuario.

RF.F.8.1. El sistema mostrará cada una de ellas junto a un botón para eliminar la reserva.

RF.F.8.2. El sistema mostrará un botón para hacer una nueva reserva.

RF.F.8.3. El sistema permitirá ver un calendario con las reservas de los usuarios y los huecos disponibles para hacer una nueva reserva.

RF.F.9. El sistema mostrará notificaciones según el rol:

RF.F.9.1. Estudiante

RF.F.9.1.1. Cuando el estudiante edite su perfil.

RF.F.9.1.2. Cuando el estudiante edite su contraseña.

RF.F.9.1.3. Cuando haya algún error de autenticación al iniciar sesión.

RF.F.9.1.4. Cuando al registrarse haya algún error.

RF.F.9.1.5. Cuando haya algún error en el servidor.

RF.F.9.1.6. Cuando el estudiante haga una nueva reserva.

RF.F.9.2. Administrador

RF.F.9.2.1. Cuando el administrador edite su perfil.

RF.F.9.2.2. Cuando el administrador edite su contraseña.

RF.F.9.2.3. Cuando el administrador suba un contenido.

RF.F.9.2.4. Cuando el administrador suba una simulación.

RF.F.9.2.5. Cuando haya algún error de autenticación al iniciar sesión.

RF.F.9.2.6. Cuando al registrarse haya algún error.

RF.F.9.2.7. Cuando haya algún error en el servidor.

RF.F.9.2.8. Cuando el administrador haga una nueva reserva.

RF.F.10. El sistema contará con una barra de navegación que permitirá el acceso fácil a todas las secciones de la plataforma:

RF.F.10.1. La barra de navegación estará visible en todas las páginas de la aplicación.

RF.F.10.2. La barra de navegación incluirá enlaces a las siguientes secciones, según el rol del usuario:

RF.F.10.2.1. Estudiante

RF.F.10.2.1.1. Inicio

RF.F.10.2.1.2. Contenidos

RF.F.10.2.1.3. Simulaciones

RF.F.10.2.1.4. Laboratorio Remoto

RF.F.10.2.1.4.1. Laboratorio remoto

RF.F.10.2.1.4.2. Reservar

RF.F.10.2.1.5. Perfil

RF.F.10.2.1.5.1. Ver perfil

RF.F.10.2.1.5.2. Mis Reservas

RF.F.10.2.1.5.3. Cerrar sesión

RF.F.10.2.2. Administrador

RF.F.10.2.2.1. Inicio

RF.F.10.2.2.2. Contenidos

RF.F.10.2.2.2.1. Contenidos

RF.F.10.2.2.2.2. Subir contenido

RF.F.10.2.2.3. Simulaciones

RF.F.10.2.2.3.1. Simulaciones

RF.F.10.2.2.3.2. Subir simulación

RF.F.10.2.2.4. Laboratorio Remoto

RF.F.10.2.2.4.1. Laboratorio remoto

RF.F.10.2.2.4.2. Subir experimento

RF.F.10.2.2.4.3. Reservar

RF.F.10.2.2.5. Perfil

RF.F.10.2.2.5.1. Ver perfil

RF.F.10.2.2.5.2. Mis Reservas

RF.F.10.2.2.5.3. Cerrar sesión

RF.F.10.3. La barra de navegación incluirá un indicador de la página actual en la que se encuentra el usuario.

RF.F.10.4. La barra de navegación incluirá un botón de cierre de sesión accesible en todo momento.

RF.F.11. El sistema permitirá cambiar el idioma a inglés o español.

3.1.2.- Requisitos funcionales del *Backend*

RF.B.1. Gestión de usuarios

RF.B.1.1. El sistema permitirá el registro de nuevos usuarios.

RF.B.1.2. El sistema permitirá la autenticación de usuarios ya registrados.

RF.B.1.3. El sistema permitirá la edición de la información personal del usuario.

RF.B.1.4. El sistema permitirá la edición de la contraseña del usuario.

RF.B.1.5. El sistema gestionará el rol y los permisos del usuario.

RF.B.2. Gestión de contenidos

RF.B.2.1. El sistema permitirá listar todos los contenidos existentes ordenados por temática.

RF.B.2.2. El sistema permitirá la subida de nuevos contenidos.

RF.B.2.3. El sistema permitirá la eliminación de contenidos.

RF.B.3. Gestión de simulaciones

RF.B.3.1. El sistema permitirá listar todas las simulaciones existentes ordenadas por temática y nombre.

RF.B.3.2. El sistema permitirá la subida de nuevas simulaciones.

RF.B.3.3. El sistema permitirá la eliminación de simulaciones.

RF.B.4. Gestión de reservas

RF.B.4.1. El sistema permitirá listar todas las reservas existentes.

RF.B.4.2. El sistema permitirá crear nuevas reservas.

RF.B.4.3. El sistema permitirá la eliminación de reservas.

RF.B.5. Gestión de experimentos.

RF.B.5.1. El sistema permitirá listar todos los experimentos disponibles.

RF.B.5.2. El sistema permitirá subir nuevos hipervínculos a experimentos.

RF.B.5.3. El sistema permitirá la eliminación de los hipervínculos.

RF.B.6. Seguridad

RF.B.6.1. El sistema encriptará las contraseñas de los usuarios

RF.B.6.2. El sistema gestionará los permisos y roles de cada usuario mediante los tokens.

RF.B.7. Gestión de datos

RF.B.7.1. El sistema se comunicará con la base de datos para almacenar y recuperar datos

RF.B.7.1.1. Datos de usuarios.

RF.B.7.1.2. Datos de contenidos.

RF.B.7.1.3. Datos de simulaciones.

RF.B.7.1.4. Datos de reservas.

RF.B.8. API de comunicación

RF.B.8.1. El sistema se comunicará con el *Frontend* mediante una API.

RF.B.8.2. El sistema permitirá el acceso a diferentes funcionalidades de gestión

RF.B.8.2.1. Gestión de usuarios.

RF.B.8.2.1. Gestión de contenidos.

RF.B.8.2.1. Gestión de simulaciones.

RF.B.8.2.1. Gestión de experimentos.

RF.B.8.2.1. Gestión de reservas.

3.1.3.- Requisitos funcionales de la base de datos

RF.BD.1. El sistema permitirá almacenar, modificar y borrar información sobre los usuarios:

RF.BD.1.1. Identificador

RF.BD.1.2. Nombre

RF.BD.1.3. Apellidos

RF.BD.1.4. Correo electrónico

RF.BD.1.5. Contraseña encriptada

RF.BD.1.6. Rol

RF.BD.2. El sistema permitirá almacenar, modificar y borrar información sobre los contenidos:

- RF.BD.2.1. Identificador
- RF.BD.2.2. Identificador del usuario que ha subido el contenido
- RF.BD.2.3. Nombre
- RF.BD.2.4. Temática
- RF.BD.2.5. URL
- RF.BD.2.6. Tamaño
- RF.BD.2.7. Si es un contenido eXeLearning
    - RF.BD.2.7.1. Fichero principal

RF.BD.3. El sistema permitirá almacenar, modificar y borrar información sobre las simulaciones:

- RF.BD.3.1. Identificador
- RF.BD.3.2. Identificador del usuario que ha subido la simulación
- RF.BD.3.3. Nombre
- RF.BD.3.4. Temática
- RF.BD.3.5. Fichero principal
- RF.BD.3.6. URL
- RF.BD.3.7. URL de la imagen
- RF.BD.3.8. Tamaño

RF.BD.4. El sistema permitirá almacenar, modificar y borrar información sobre las reservas:

- RF.BD.4.1. Identificador
- RF.BD.4.2. Identificador del usuario que ha hecho la reserva
- RF.BD.4.3. Fecha de inicio
- RF.BD.4.4. Fecha de finalización
- RF.BD.4.5. Fecha de creación

3.2.- Requisitos no funcionales

Los requisitos no funcionales detallan como se debe de comportar un sistema internamente como la seguridad, el rendimiento, la usabilidad, es decir, estos requisitos se centran más en el comportamiento que debe tener un sistema al realizar una funcionalidad. Este apartado se divide en los requisitos no funcionales que debe de tener cada una de las distintas capas de *VirtyRemLab.*

3.2.1.- Requisitos no funcionales del *Frontend*

RNF.F.1. Usabilidad

RNF.F.1.1. El sistema debe de ser intuitivo y fácil de usar.

RNF.F.1.2. El sistema debe de aportar comentarios al realizar funcionalidades que necesiten confirmación de cambios (Subir un nuevo contenido, editar el perfil, …).

RNF.F.2. El sistema debe de ser accesible para todo tipo de usuarios.

RNF.F.3. El sistema debe de ser compatible con todo tipo de navegadores y dispositivos.

RNF.F.4. El sistema debe de manejar los errores y dar información clara sobre estos para poder solucionarlos.

RNF.F.5. El sistema debe de mostrar las distintas páginas de forma rápida.

RNF.F.6. Internacionalización

RNF.F.6.1. El sistema debe soportar múltiples idiomas.

RNF.F.6.2. El sistema debe de permitir agregar nuevos idiomas de forma sencilla.

RNF.F.7. Escalabilidad

RNF.F.7.1. El sistema debe de permitir agregar nuevas funcionalidades de forma sencilla y sin afectar al resto de la aplicación.

RNF.F.8. Seguridad

RNF.F.8.1. El sistema debe de transmitir los datos sensibles de forma segura a través de conexiones cifradas HTTPS.

RNF.F.8.2. El sistema no debe de permitir a los usuarios el acceso a vistas de administrador sin los permisos adecuados.

RNF.F.8.3. El sistema no debe de permitir a los usuarios acceder a las vistas sin haber iniciado sesión.

3.2.2.- Requisitos no funcionales del *Backend*

RNF.B.1. Rendimiento

RNF.B.1.1. El sistema no debe de degradar su rendimiento, aunque aumente el número de usuarios conectados.

RNF.B.1.2. El sistema debe de procesar y devolver los datos de forma rápida.

RNF.B.2. Escalabilidad

RNF.B.2.1. El sistema debe de ser escalable para poder aumentar su carga de trabajo.

RNF.B.2.2. El sistema debe permitir añadir nuevas funcionalidades de forma sencilla.

RNF.B.3. Seguridad

RNF.B.3.1. El sistema debe de proteger los datos sensibles mediante encriptación.

RNF.B.3.2. El sistema debe de proteger las funcionalidades del administrador mediante roles.

RNF.B.4. El sistema debe de estar disponible la mayor parte del tiempo.

RNF.B.5. Compatibilidad

RNF.B.5.1. El sistema debe de ser compatible con las tecnologías usadas en el *Frontend*.

RNF.B.5.2. El sistema debe de ser compatible con la base de datos elegida.

RNF.B.6. El sistema debe de registrar las actividades importantes como el inicio y cierre de sesión.

3.2.3.- Requisitos no funcionales de la base de datos

RNF.BD.1. Rendimiento

RNF.BD.1.1. El sistema debe de almacenar los datos y hacer consultas de forma rápida.

RNF.BD.1.2. El sistema debe de hacer estas transacciones de forma eficiente, aunque aumente el número de estas.

RNF.BD.2. Escalabilidad

RNF.BD.2.1 El sistema debe de ser escalable para aumentar la cantidad de datos sin afectar al sistema.

RNF.BD.3. Seguridad

RNF.BD.3.1. El sistema debe de proteger los datos sensibles mediante encriptación.

RNF.BD.3.2. El sistema debe de controlar el acceso a los datos y permitírselo solo a personas autorizadas.

RNF.BD.4. El sistema debe de estar disponible la mayor parte del tiempo.

RNF.BD.5. El sistema debe de ser compatible con las tecnologías usadas en el *Backend*.

3.3.- Identificación de actores del sistema

- **Estudiante:** Es el usuario que utiliza la aplicación web para acceder a los contenidos, las simulaciones y los experimentos en remoto.

- **Administrador:** Es el usuario que utiliza el sistema para mantener actualizada la página web, es decir, subir nuevos contenidos, nuevas simulaciones y gestionar la plataforma. Este rol se verá adoptado por los diferentes profesores o colaboradores docentes que participen en el proyecto de VirtyRemLab.

3.4.- Especificación de casos de uso

En este apartado se describen los diferentes casos de uso. Estos representan las interacciones que los actores (estudiante y administrador) tienen con *VirtyRemLab*.

3.4.1.- Estudiante

CU.E.1. Registro de usuario

- **Descripción:** El estudiante se registra como usuario en *VirtyRemLab*.
- **Condiciones:** Necesita tener el correo corporativo de la Universidad de Oviedo.
- **Flujo:**
    - El estudiante entra en la plataforma y accede a la sección de registro.
    - El estudiante introduce los datos personales (nombre, apellidos, correo electrónico, contraseña).
    - El sistema valida la información.
        - Si algún dato es incorrecto devuelve un error.
        - Si todo es correcto almacena la información en la base de datos.
    - El sistema informa de que el usuario se ha registrado correctamente y le lleva a la sección de iniciar sesión.

CU.E.2. Inicio de sesión

- **Descripción:** El estudiante inicia sesión en *VirtyRemLab*.
- **Condiciones:** Necesita haberse registrado como usuario.
- **Flujo:**
    - El estudiante entra en la plataforma y accede a la sección de inicio de sesión.
    - El estudiante introduce los datos personales (correo electrónico y contraseña).
    - El sistema valida la información.
        - Si algún dato es incorrecto devuelve un error.
        - Si todo es correcto genera un token con su información y lo añade a los tokens válidos.
    - El sistema redirige al usuario a la página principal de la aplicación.

CU.E.3. Visualización de contenidos

- **Descripción:** El estudiante visualiza los contenidos de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
    - El estudiante accede a la sección de contenidos.
    - El sistema muestra las temáticas de los contenidos disponibles.
    - El estudiante escoge una o varias temáticas.
    - El sistema muestra el listado de contenidos disponibles.
    - El estudiante selecciona el contenido elegido.
    - El sistema le redirige al contenido escogido.

CU.E.4. Visualización de simulaciones

- **Descripción:** El estudiante visualiza las simulaciones de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
    - El estudiante accede a la sección de simulaciones.
    - El sistema muestra las temáticas de las simulaciones disponibles.

- El estudiante escoge una o varias temáticas.
- El sistema muestra el listado de simulaciones disponibles.
- El estudiante selecciona la simulación elegida.
- El sistema le redirige a la simulación escogida.

CU.E.5. Visualización de experimentos

- **Descripción:** El estudiante visualiza los experimentos de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El estudiante accede a la sección de laboratorio remoto.
  - El sistema muestra los experimentos disponibles.
  - El estudiante escoge el experimento.
  - El sistema muestra el experimento escogido.

CU.E.6. Visualización de reservas

- **Descripción:** El estudiante visualiza las reservas de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El estudiante accede a la sección de reservas.
  - El sistema muestra un calendario con las reservas y los huecos disponibles.
  - El estudiante escoge un hueco.
  - El sistema valida la información.
    - Si la información es inválida devuelve un error.
    - Si todo es correcto almacena los datos en la base de datos.
  - El sistema informa que la reserva se ha hecho correctamente.

CU.E.7. Visualización de perfil

- **Descripción:** El estudiante visualiza su perfil de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El estudiante accede a la sección de ver perfil.

- El sistema muestra la información (nombre, apellidos y correo electrónico) del usuario.

CU.E.8. Edición del perfil

- **Descripción:** El estudiante edita su perfil de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y estar en la sección de ver perfil.
- **Flujo:**
  - El estudiante pulsa en el botón de editar.
  - El sistema muestra la información (nombre, apellidos y correo electrónico) del usuario.
  - El estudiante cambia el nombre, los apellidos o ambos.
  - El estudiante pulsa en editar.
  - El sistema valida la información.
    - Si algún dato es incorrecto devuelve error.
    - Si todo es correcto almacena la información en la base de datos.
  - El sistema informa al usuario de que su perfil se ha editado correctamente y le redirige al apartado de ver perfil.

CU.E.9. Edición de contraseña

- **Descripción:** El estudiante edita su contraseña de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y estar en la sección de editar perfil.
- **Flujo:**
  - El estudiante pulsa en el botón de editar contraseña.
  - El sistema muestra los campos de contraseña actual, nueva contraseña y repetir contraseña.
  - El estudiante rellena los campos.
  - El estudiante pulsa en editar.
  - El sistema valida la información.
    - Si algún dato es incorrecto devuelve error.

        - Si todo es correcto almacena la información en la base de datos.
    - El sistema informa al usuario de que su contraseña se ha editado correctamente y le redirige al apartado de ver perfil.

CU.E.10. Visualización y gestión de reservas propias

- **Descripción:** El estudiante visualiza sus reservas de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
    - El estudiante accede al apartado de mis reservas.
    - El sistema muestra en una tabla las reservas del usuario junto a un botón para eliminar, además de un botón para hacer una nueva reserva.
    - El estudiante pulsa en el botón de eliminar.
    - El sistema elimina de la base de datos la reserva elegida y actualiza la página con las reservas restantes.
    - El estudiante pulsa en el botón de nueva reserva.
    - El sistema le redirige al apartado de reservar.

CU.E.11. Cambiar de idioma

- **Descripción:** El estudiante cambia el idioma de *VirtyRemLab*.
- **Condiciones:** Necesita haber entrado en la plataforma.
- **Flujo:**
    - El estudiante pulsa en la bandera del idioma elegido.
    - El sistema muestra la aplicación en el idioma escogido.

CU.E.12. Cerrar sesión

- **Descripción:** El estudiante cierra sesión en *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión.
- **Flujo:**
    - El estudiante pulsa el botón de cerrar sesión.
    - El sistema borra el token de los token válidos y le redirige a la página principal de la plataforma.

3.4.2.- Administrador

CU.P.1. Registro de usuario

- **Descripción:** El profesor se registra como usuario en *VirtyRemLab*.
- **Condiciones:** Necesita tener el correo corporativo de la Universidad de Oviedo.
- **Flujo:**
    - El profesor entra en la plataforma y accede a la sección de registro.
    - El profesor introduce los datos personales (nombre, apellidos, correo electrónico, contraseña).
    - El sistema valida la información.
        - Si algún dato es incorrecto devuelve un error.
        - Si todo es correcto almacena la información en la base de datos.
    - El sistema informa de que el usuario se ha registrado correctamente y le lleva a la sección de iniciar sesión.

CU.P.2. Inicio de sesión

- **Descripción:** El profesor inicia sesión en *VirtyRemLab*.
- **Condiciones:** Necesita haberse registrado como usuario.
- **Flujo:**
    - El profesor entra en la plataforma y accede a la sección de inicio de sesión.
    - El profesor introduce los datos personales (correo electrónico y contraseña).
    - El sistema valida la información.
        - Si algún dato es incorrecto devuelve un error.
        - Si todo es correcto genera un token con su información y lo añade a los tokens válidos.
    - El sistema redirige al usuario a la página principal de la aplicación.

CU.P.3. Visualización y gestión de contenidos

- **Descripción:** El profesor visualiza y gestiona los contenidos de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**

- El profesor accede a la sección de contenidos.
- El sistema muestra las temáticas de los contenidos disponibles.
- El profesor escoge una o varias temáticas.
- El sistema muestra el listado de contenidos disponibles.
- El profesor selecciona el contenido elegido.
- El sistema le redirige al contenido escogido.
- El profesor pulsa el botón de eliminar del contenido elegido.
- El sistema borra el contenido de la base de datos y del servidor, actualizando la página para volver a mostrar los contenidos restantes.

CU.P.4. Subir nuevo contenido

- **Descripción:** El profesor sube un nuevo contenido a *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y ser administrador.
- **Flujo:**
  - El profesor accede al apartado de subir contenido.
  - El sistema muestra un formulario con campos a rellenar (nombre, temática, archivo, si es un contenido eXelearning y si es así el nombre del fichero principal).
  - El profesor rellena los campos.
  - El sistema valida la información.
    - Si hay algo incorrecto devuelve un error.
    - Si todo es correcto añade el contenido a la base de datos y al servidor.
  - El sistema informa al usuario de que el archivo se ha subido correctamente.

CU.P.5. Visualización y gestión de simulaciones

- **Descripción:** El profesor visualiza y gestiona las simulaciones de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El profesor accede a la sección de simulaciones.

- El sistema muestra las temáticas de las simulaciones disponibles.
- El profesor escoge una o varias temáticas.
- El sistema muestra el listado de simulaciones disponibles.
- El profesor selecciona la simulación elegida.
- El sistema le redirige a la simulación escogida.
- El profesor pulsa el botón de eliminar de la simulación elegida.
- El sistema borra la simulación de la base de datos y del servidor, actualizando la página para volver a mostrar las simulaciones restantes.

CU.P.6. Subir nueva simulación

- **Descripción:** El profesor sube una nueva simulación a *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y ser administrador.
- **Flujo:**
  - El profesor accede al apartado de subir simulación.
  - El sistema muestra un formulario con campos a rellenar (nombre, temática, archivo, archivo principal e imagen).
  - El profesor rellena los campos.
  - El sistema valida la información.
    - Si hay algo incorrecto devuelve un error.
    - Si todo es correcto añade la simulación a la base de datos y al servidor.
  - El sistema informa al usuario de que el archivo se ha subido correctamente.

CU.P.7. Visualización y gestión de experimentos

- **Descripción:** El profesor visualiza los experimentos de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El profesor accede a la sección de laboratorio remoto.
  - El sistema muestra los experimentos disponibles.
  - El profesor escoge el experimento.

- El sistema muestra el experimento escogido.
- El profesor borra el hipervínculo del experimento.
- El sistema borra el hipervínculo de la plataforma actualizando la página para volver a mostrar los experimentos restantes.

CU.P.8. Subir nuevos hipervínculos a experimentos

- **Descripción:** El profesor sube un nuevo experimento a *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y ser administrador.
- **Flujo:**
  - El profesor accede a la sección de subir experimento.
  - El sistema muestra un formulario con distintos campos a rellenar (nombre, temática y URL).
  - El profesor rellena el formulario y pulsa el botón de subir experimento.
  - El sistema valida la información.
    - Si hay algo incorrecto devuelve un error.
    - Si todo es correcto añade el experimento a la base de datos.
  - El sistema informa al usuario de que el archivo se ha subido correctamente.

CU.P.9. Visualización de reservas

- **Descripción:** El profesor visualiza las reservas de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El profesor accede a la sección de reservas.
  - El sistema muestra un calendario con las reservas y los huecos disponibles.
  - El profesor escoge un hueco.
  - El sistema valida la información.
    - Si la información es inválida devuelve un error.
    - Si todo es correcto almacena los datos en la base de datos.
  - El sistema informa que la reserva se ha hecho correctamente.

CU.P.10. Visualización de perfil

- **Descripción:** El profesor visualiza su perfil de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
    - El profesor accede a la sección de ver perfil.
    - El sistema muestra la información (nombre, apellidos y correo electrónico) del usuario.

CU.P.11. Edición del perfil

- **Descripción:** El profesor edita su perfil de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y estar en la sección de ver perfil.
- **Flujo:**
    - El profesor pulsa en el botón de editar.
    - El sistema muestra la información (nombre, apellidos y correo electrónico) del usuario.
    - El profesor cambia el nombre, los apellidos o ambos.
    - El profesor pulsa en editar.
    - El sistema valida la información.
        - Si algún dato es incorrecto devuelve error.
        - Si todo es correcto almacena la información en la base de datos.
    - El sistema informa al usuario de que su perfil se ha editado correctamente y le redirige al apartado de ver perfil.

CU.P.12. Edición de contraseña

- **Descripción:** El profesor edita su contraseña de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma y estar en la sección de editar perfil.
- **Flujo:**
    - El profesor pulsa en el botón de editar contraseña.
    - El sistema muestra los campos de contraseña actual, nueva contraseña y repetir contraseña.

- El profesor rellena los campos.
- El profesor pulsa en editar.
- El sistema valida la información.
  - Si algún dato es incorrecto devuelve error.
  - Si todo es correcto almacena la información en la base de datos.
- El sistema informa al usuario de que su contraseña se ha editado correctamente y le redirige al apartado de ver perfil.

CU.P.13. Visualización y gestión de reservas propias

- **Descripción:** El profesor visualiza sus reservas de *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión en la plataforma.
- **Flujo:**
  - El profesor accede al apartado de mis reservas.
  - El sistema muestra en una tabla las reservas del usuario junto a un botón para eliminar, además de un botón para hacer una nueva reserva.
  - El profesor pulsa en el botón de eliminar.
  - El sistema elimina de la base de datos la reserva elegida y actualiza la página con las reservas restantes.
  - El profesor pulsa en el botón de nueva reserva.
  - El sistema le redirige al apartado de reservar.

CU.P.14. Cambiar de idioma

- **Descripción:** El profesor cambia el idioma de *VirtyRemLab*.
- **Condiciones:** Necesita haber entrado en la plataforma.
- **Flujo:**
  - El profesor pulsa en la bandera del idioma elegido.
  - El sistema muestra la aplicación en el idioma escogido.

CU.P.15. Cerrar sesión

- **Descripción:** El profesor cierra sesión en *VirtyRemLab*.
- **Condiciones:** Necesita haber iniciado sesión.

- **Flujo:**
    - El profesor pulsa el botón de cerrar sesión.
    - El sistema borra el token de los token válidos y le redirige a la página principal de la plataforma.

4. Desarrollo del proyecto

En este apartado se describe detalladamente el proceso de desarrollo de *VirtyRemLab*, incluyendo el diseño del sistema, las tecnologías y herramientas usadas, así como la estructura del proyecto.

4.1.- Diseño del sistema

En el ciclo de vida de un producto *software* una de las etapas más importantes es decidir que arquitectura se va a usar, esta define como los componentes del sistema interactúan entre sí y como se organizan para cumplir los requisitos funcionales y no funcionales del sistema.

Proporciona una visión completa del proyecto, de esta forma permite a los desarrolladores comprender su diseño y tomar decisiones durante la implementación y mantenimiento [11].

Otros beneficios de utilizar una arquitectura *software* es que la aplicación será escalable, reduce costes ya que evita la duplicación de código, permite que la actualización de la aplicación sea más sencilla y también que se corrijan errores más fácilmente al tener una visión global de la plataforma [12].

Para *VirtyRemLab* se tuvieron en cuenta varios tipos de arquitectura, a continuación, se hace una comparativa de dos enfoques típicos, que son cliente-servidor y N capas. Justificando así la elección del modelo N capas.

4.1.1.- Arquitectura Cliente-Servidor

La arquitectura Cliente-Servidor es una aplicación de red donde se divide el trabajo entre el cliente y el servidor que residen en el mismo sistema o que están vinculados por una red.

El cliente utiliza un servicio, según términos informáticos, se refieren a un ordenador o dispositivo, que utiliza el servicio y acepta la información. Mientras que el servidor es aquel ordenador remoto que brinda el acceso a los servicios y datos a los clientes [13].

El funcionamiento de este modelo es el siguiente [14]:

- Cliente y servidor se conectan entre sí mediante una red, normalmente utilizando los protocolos de control de transmisión y el protocolo de Internet, es decir, TCP/IP.
- El cliente solicita un servicio al servidor.
- El servidor responde al cliente.

Esta arquitectura tiene distintos tipos [15]:

- **1 nivel**: Todas las capas (*Frontend*, *Backend*, base de datos) están en un mismo grupo, de esta forma todo el procesamiento y almacenamiento de datos se realiza en un solo lugar.

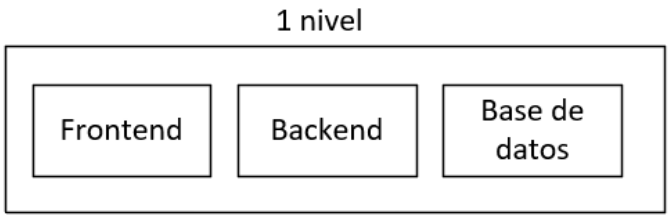

Figura 4.1.- Arquitectura de un nivel.

- **2 niveles**: En este caso hay dos capas la aplicación cliente (*Frontend*) y el servidor de datos (*Backend*, Base de datos).

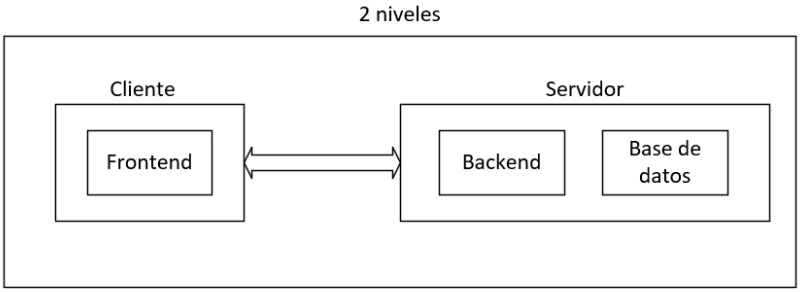

Figura 4.2.- Arquitectura de dos niveles.

Algunos de los beneficios de usar este modelo son [16]:

- Garantiza la coherencia y la integridad de los datos.
- Control centralizado para una mejor implementación de las medidas de seguridad.
- Gestión de datos sencillo y controlado.

A su vez tiene distintos desafíos [16]:

- Escalabilidad limitada, puede ser difícil a medida que la aplicación crece.
- Si hay algún punto de falla este puede invalidar el acceso a los datos lo cual afecta a los clientes conectados.
- Puede haber problemas de compatibilidad entre los distintos clientes.

Aunque el modelo Cliente-Servidor ofrece ventajas significativas para las aplicaciones como una implementación más rápida, presenta limitaciones en términos de escalabilidad y como *VirtyRemLab* va a estar en constante desarrollo se ha decidido no elegir esta arquitectura para el proyecto.

4.1.2.- Arquitectura N capas

En esta arquitectura el desarrollo del *software* se divide en distintos niveles o capas. El número de capas viene dado por la "N" y este depende de los requisitos y complejidad de la aplicación.

El funcionamiento de este modelo es el siguiente:

- El cliente se conecta al nivel de aplicación enviando solicitudes mediante una API.
- El servidor procesa estas solicitudes y si es necesario consulta a la base de datos.
- La base de datos gestiona la información mandándole una respuesta al servidor, que a su vez se la envía al cliente.

Dentro de esta arquitectura el modelo más común es el modelo de 3 capas, que es el que implementa *VirtyRemLab*, consta de estos niveles [17]:

- **Capa de presentación:** Es la capa de la interfaz de usuario, donde los usuarios interactúan con la aplicación (*Frontend*).
- **Nivel de aplicación:** Es el núcleo de la aplicación contiene la lógica de negocio, se procesan las solicitudes del *Frontend*, es decir, es el *Backend*.
- **Capa de datos:** Esta capa es la responsable de la administración de los datos, maneja la comunicación entre la base de datos y el *Backend*.

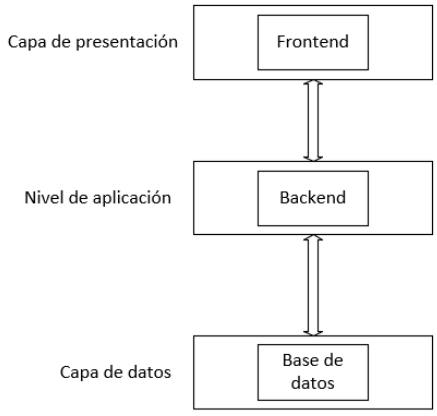

Figura 4.3.- Arquitectura de N capas.

Las ventajas de usar este modelo son las siguientes [18]:

- **Mantenibilidad:** La separación entre distintas capas hace que sea más sencillo detectar errores y corregirlos.
- **Escalabilidad:** La aplicación se puede escalar de forma eficiente agregando nuevos componentes donde sea necesario sin afectar al resto del sistema.
- **Reutilización:** Los distintos componentes se pueden usar en otras partes del proyecto e incluso en otros proyectos.

Aun así, este modelo requiere un esfuerzo inicial, ya que diseñar y configurar este tipo de arquitectura puede ser más complejo que en Cliente-Servidor.

Como se ha aclarado en la explicación del modelo Cliente-Servidor, *VirtyRemLab* necesita una arquitectura que permita su posterior evolución y mantenimiento, admitiendo nuevos componentes en cualquier capa sin afectar al resto del sistema, por esto y porque es un modelo que al estar estructurado en distintas capas permite una mejor comprensión del sistema se ha elegido el modelo N capas como modelo de arquitectura para llevar a cabo este proyecto.

4.2.- Herramientas y tecnologías

En esta sección se van a explicar las distintas herramientas y tecnologías aplicadas en *VirtyRemLab* para su implementación basada en el modelo de N capas.

4.2.1.- *Frameworks* y librerías

Un *framework* es una estructura sobre la que se puede construir *software*, esto permite escribir código en un menor tiempo y reduciendo el riesgo de errores, además de que es código más seguro ya que han sido probados, evitando también el código duplicado.

Las librerías son un conjunto de código escrito previamente que se puede adaptar a tu código para solucionar funcionalidades concretas. Los *frameworks* pueden incluir librerías, aunque hay algunos que no las utilizan [19].

4.2.1.1.- *Frontend*

Para el desarrollo del *Frontend* se han estudiado dos alternativas muy usadas hoy en día, estas son AngularJS y ReactJS. A continuación, se explican los beneficios y problemas de cada una de ellas, justificando así la elección de ReactJS como herramienta principal del desarrollo del *Frontend* de *VirtyRemLab*.

AngularJS es un *framework* basado en JavaSript desarrollado en 2009 por el equipo de Google que se utiliza para la creación de aplicaciones web dinámicas. Este UI está basado en la arquitectura Modelo-Vista-Controlador (MVC) lo que permite un proyecto bien organizado desde el primer momento. Utilizando como lenguaje TypeScript, tiene algunas ventajas como la inyección de dependencias o el enlace de datos bidireccional, pero esto supone un gran inconveniente y es que es más difícil de comprender y de utilizar, al tener que comprender conceptos como módulos o sobre la inyección de dependencias [20, 21].

ReactJS es una biblioteca de JavaScript que permite la creación de la interfaz de usuario combinando distintas vistas individuales llamadas componentes. React fue creado en 2013 por Facebook, es una biblioteca de código abierto, es decir, es accesible de manera pública, además se está actualizando constantemente. La principal ventaja de React es que es una librería fácil de entender gracias a su sencilla sintaxis y por lo tanto fácil de usar. Como se basa en distintos componentes, en vez de enviar una petición al servidor cada vez que se quiere cambiar de vista, el contenido de esta se carga directamente de los componentes, esto permite una representación más rápida. Aun así ReactJS no tiene una documentación adecuada debido a sus actualizaciones continuas por lo que no hay un patrón de desarrollo común y hay que tomar bastantes decisiones [22, 23].

Tras estudiar las distintas opciones se ha escogido la librería de React para la implementación de la capa de presentación del sistema, esto es debido a que al ser una biblioteca da la opción de elegir otras librerías y herramientas que se adapten a las necesidades específicas de *VirtyRemLab* y al estar en constante desarrollo es sencillo encontrar recursos o soluciones a problemas comunes, además de que este proyecto va a seguir en desarrollo y la curva de aprendizaje es más suave, ya que el lenguaje utilizado es JavaScript con HTML.

El *framework* que se ha escogido para utilizar React es Ant Design, que es el segundo *framework* más utilizado del mundo, permite el uso de componentes prácticos de React que permiten personalización y flexibilidad y además están listos para usar [24].

Esta UI fue implementada por Ant Financial y su departamento de Tecnología de Experiencia teniendo como objetivo unificar las especificaciones de la interfaz de usuarios para proyectos, además de reducir costes innecesarios de las diferencias de diseño e implementación de un *Frontend*. Este *framework* contiene algunas bibliotecas como Bootstrap [25], la cual tiene una gran variedad de componentes desarrollados.

Aunque React y Ant Design se llevan la mayor parte de la implementación del *Frontend*, se han utilizado algunas otras librerías y *frameworks* que se describen a continuación:

- **React-*Router*-Dom:** Permite la navegación fluida en una sola página. El usuario cambia de URL y al mismo tiempo se actualiza la vista sin necesidad de recargar la página [26].
- **React-i18next:** Permite la internacionalización de la página, facilitando la integración de distintos idiomas de una forma sencilla [27].
- **FullCalendar:** Permite añadir a la página el componente de un calendario más complejo con distintas funcionalidades de las que carece el componente calendario de Ant Design [28].

4.2.1.2.- *Backend*

En la actualidad hay muchas alternativas para desarrollar el nivel de aplicación de nuestro proyecto, seguidamente se va a explicar el estudio que se ha llevado a cabo para acabar eligiendo NodeJS como lenguaje para desarrollar la parte del *Backend* de *VirtyRemLab*.

Spring Boot es un *framework* de código abierto basado en Java que facilita la creación de aplicaciones independientes con Spring. Este permite una configuración de la aplicación automática por lo que los desarrolladores tienen que invertir menos tiempo, además estas configuraciones son flexibles. Tiene el servidor integrado, llamado TomCat, aunque puedes elegir otro, lo cual es una ventaja ya que no se tiene que tomar decisiones sobre que servidor elegir. Aun así, aunque esta herramienta cree una aplicación de forma sencilla y rápida, la configuración automática crea dependencias innecesarias que da lugar a un *Backend* de un gran tamaño, además de una curva de aprendizaje compleja al tener que comprender conceptos y clases nuevas, lo cual puede llevar a errores por no entender bien lo que se está haciendo [29, 30].

Flask es un *framework* basado en Python, desarrollado en 2010 por Armin Ronacher, que crea aplicaciones WSGI, es decir, que facilitan la conexión entre el servidor web y la aplicación web. Flask es fácil y rápido de aprender, además de que ser flexible al poder instalar una gran cantidad de bibliotecas diferentes para cumplir con las especificaciones del sistema y aunque eso es una ventaja también es un inconveniente ya que requiere más configuración y esto afecta al mantenimiento de la aplicación [31].

Ambas opciones han sido descartadas por la misma razón, *VirtyRemLab* va a seguir en crecimiento y por ello se requiere de un lenguaje o un *framework* que sea sencillo de utilizar y mantenible. Por esta razón se optó por NodeJS.

NodeJS es un lenguaje, de código abierto, basado en JavaScript. Está diseñado para hacer aplicaciones de red escalables, además tiene un gran rendimiento, ya que como tiene sólo un hilo para ejecutar la aplicación, se pueden manejar múltiples conexiones simultáneas sin bloquear el hilo principal, aunque esto puede suponer un problema en cuanto al rendimiento si no se manejan bien las operaciones de entrada y salida. Al igual

que en ReactJS el lenguaje es JavaScript, lo cual favorece al mantenimiento y al desarrollo siendo estos más coherentes y eficientes. Sin embargo, puede ser complicado el acceso a la base de datos, ya que al ser programación asíncrona requiere un mayor esfuerzo a la hora de manejar errores [32,33].

El *framework* que se ha elegido para implementar NodeJS es Express debido a su fácil implementación y flexibilidad, también permite el uso de *middlewares* para añadir más funcionalidades a la aplicación [34].

Asimismo, se han usado las siguientes bibliotecas en la implementación del *Backend*:

- En términos de seguridad se ha utilizado la biblioteca **JSONWebToken**, para la autenticación de usuarios y la autorización a recursos de los Administradores [35]. JSON (JavaScript Object Notation), es un formato para guardar e intercambiar datos conteniendo solo texto [47].
- Para la compresión y descompresión de archivos se ha utilizado la librería **Admzip** [36].
- Con la finalidad de subir contenidos y simulaciones se ha usado la librería **Multer** [37].
- Para tratar con los archivos y URLs se han utilizado las bibliotecas de NodeJS, **path** y **fs**.
- En cuanto a la conexión con la base de datos, se ha empleado la librería de NodeJS, **mysql**.
- Para encriptar las contraseñas de los usuarios, se ha utilizado la librería **Bcrypt** [48]**.**
- Por último, la librería **util** ha sido de gran ayuda para la implementación de programación asíncrona a la hora de gestionar los datos.

4.2.1.3.- Base de datos

La primera decisión que tomar a la hora de decidir la base de datos a usar es elegir si se quiere una base de datos relacional o no relacional.

Una base de datos relacional es aquella que almacena la información en estructuras definidas, es decir, tablas, donde en cada se fila tiene una identificación única y datos, cada columna es un atributo de los datos a guardar, aunque esta forma de guardar los datos puede acabar siendo una limitación al ser un esquema fijo. El lenguaje que utilizan es SQL y permiten hacer aplicaciones con integridad de datos, además de permitir consultas más complejas. También cumplen con el estándar ACID [38], es decir, cumplen con las propiedades cruciales de la transacción de datos, estas son:

- **Atomicidad:** Define todos los elementos que conforman una transacción completa.
- **Uniformidad:** Define reglas para mantener todos los datos en un estado correcto tras la transacción.
- **Aislamiento:** Evita que las transacciones de datos sean visibles hasta que no acaben.
- **Durabilidad:** Garantiza que los datos se vuelvan permanentes tras la transacción.

Las bases de datos no relacionales se caracterizan por no utilizar obligatoriamente el lenguaje SQL para sus consultas, pero esto tiene la limitación de que las consultas serán más simples al no soportar complejas. Además, los datos no se almacenan en tablas sino en documentos, grafos, … lo que permite que estas bases de datos ofrezcan más flexibilidad. Este modelo no cumple con el estándar ACID y se utilizan para aplicaciones que manejan un gran volumen de datos [39].

Tras un análisis profundo, se ha elegido la opción de la base de datos relacional para *VirtyRemLab*, esta elección es debida a varias razones, una de ellas es que la integridad de datos y consistencia en la aplicación es importante para guardar la información de los usuarios, de las reservas de laboratorio remoto y de los recursos. También se necesita hacer consultas más complejas para gestionar las relaciones entre datos. Además, SQL es

un lenguaje sencillo de aprender y de implementar lo que facilita el desarrollo de la aplicación.

Dentro de una base de datos relacional se puede elegir entre varios sistemas de gestión de bases de datos, algunos de ellos son:

PostgreSQL, es una base de datos que fue desarrollada por la Universidad de California en 1986, es compatible con todos los sistemas operativos, combina SQL con funciones que permite hacer bases de datos más complicadas, también tiene una gran variación de tipos de datos. Cumple con el estándar ACID por lo que es una base de datos robusta y confiable, además de ser escalable. El problema es que, en una base de datos pequeña, las transacciones como insertar o almacenar datos se hacen de una forma lenta, ya que este sistema está diseñado para manejar grandes volúmenes de datos aparte de que al ampliar SQL hay funciones que no son fáciles de entender [40, 41].

MySQL es una base de datos utilizada por grandes aplicaciones como Netflix o Facebook, este sistema de gestión de bases de datos es fácil de usar y de gestionar, además es escalable y tiene una alta disponibilidad y recuperación de datos ante desastres. También es flexible, permitiendo guardar una gran variedad de tipos de datos entre los que destacan documentos JSON. Además, es compatible con muchas plataformas y lenguajes de programación, entre ellos NodeJS, que cuenta con una biblioteca para facilitar la integración con el servidor [42, 43]. Por lo que MySQL es una buena opción para implementar la capa de datos de *VirtyRemLab*.

4.2.2.- Lenguajes de programación

A continuación, se enumeran los distintos lenguajes de programación que se utilizan para desarrollar las distintas capas de la aplicación web *VirtyRemLab*.

Los lenguajes de programación utilizados para implementar el *Frontend* son:

- **JSX:** Este lenguaje combina HTML y CSS con JavaScript, por lo que estos contenidos ya no tienen que estar en archivos diferentes. Es el lenguaje que se utiliza para desarrollar los distintos componentes en React. Al juntar la lógica de JavaScript con HTML se garantiza que, aunque se edite el componente estos van a seguir sincronizados. La mayor dificultad de JSX es que todos los elementos tienen que estar dentro de una única etiqueta raíz [22].
- **CSS3:** Es un lenguaje de estilos que se utiliza para el diseño y la presentación de los elementos HTML en una página web.
- **HTML5:** Es el lenguaje que sirve para definir la estructura, el diseño y el contenido de una página web.

Para el *Backend* se utiliza JavaScript que es un lenguaje de programación que está orientado a objetos y se utiliza para programar cómo se comportan los elementos dentro de una página web y su sintaxis es similar a la de Java.

En cuanto a la base de datos el lenguaje de programación utilizado es SQL, que permite administrar y recuperar datos de un sistema de gestión de base de datos, en *VirtyRemLab*, es MySQL.

4.2.3.- Herramientas de desarrollo

Para desarrollar el sistema se han utilizado las siguientes herramientas de desarrollo:

- **Visual Studio Code:** Es una aplicación de escritorio que permite la implementación de múltiples lenguajes como JavaScript, Python, PHP, …. Además, es flexible y permite adaptarse a las necesidades del desarrollador pudiendo instalar extensiones que faciliten el trabajo. También cuenta con un código de depuración integrado para poder encontrar y corregir errores fácilmente. Cuenta con Git integrado lo que permite controlar las versiones de la aplicación [44]. Por lo que esta herramienta es la indicada para el desarrollo de *VirtyRemLab*.

- **Github:** Esta herramienta permite alojar proyectos para el control de versiones con Git. Permite tener varias ramas para mantener el orden de la aplicación con distintas versiones, además de permitir a los desarrolladores tener su proyecto como público y que las personas puedan usarlo o desarrollar más funcionalidades o privado para que nadie salvo los colaboradores que se escojan puedan interactuar con el proyecto [45]. En este trabajo se utiliza este programa para controlar las distintas versiones de *VirtyRemLab*.
- **HeidiSQL:** Esta herramienta es utilizada en el trabajo para la gestión de la base de datos, ya que es una herramienta fácil de usar que permite de una forma sencilla la implementación de la base de datos [46].

4.3.- Implementación del *Frontend*

En este apartado se explicará cómo ha sido desarrollada la capa de presentación de *VirtyRemLab*, explicando la estructura, los componentes y la integración con el nivel de aplicación.

4.3.1.- Estructura

El cliente está compuesto por distintas carpetas, estas son:

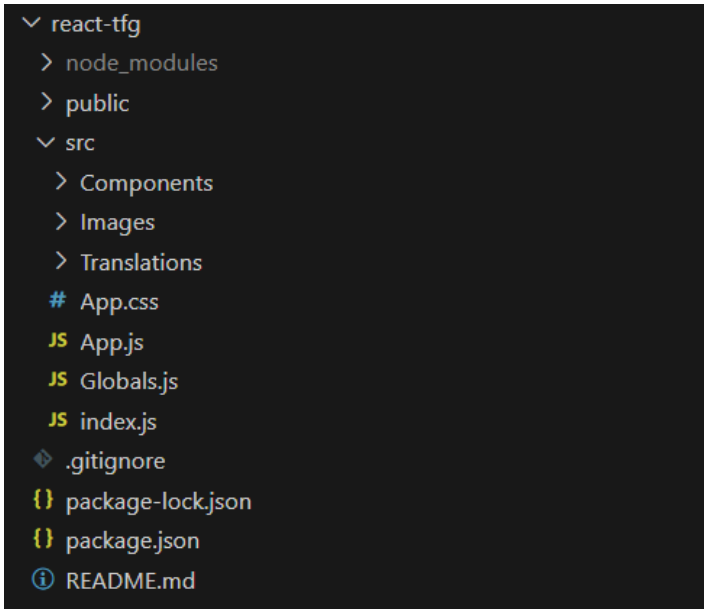

Figura 4.4.- Estructura *Frontend*.

- **Node_modules, package-lock.json, package.json:** Contienen toda la información de las distintas librerías, versiones, dependencias que componen el programa.
- **Public:** Es el directorio que contiene los archivos estáticos, es decir, los archivos que se almacenan en el servidor, como por ejemplo el logo de la Universidad de Oviedo.
- **Src:** Contiene el código fuente de la aplicación.
    - **Components:** Este directorio engloba todas las vistas de la aplicación
    - **Images:** Contiene las imágenes que usa el *Frontend*, como por ejemplo las banderas de los idiomas.
    - **Translations:** Esta carpeta incluye las traducciones a los distintos idiomas que soporta la plataforma.
    - **App.css:** Es la hoja de estilos de la aplicación, en ella están diferentes diseños para los elementos que la conforman.
    - **App.js:** Es el componente principal de la aplicación, dentro se definen las rutas a las otras vistas, el menú de navegación y el pie de página.
    - **Globals.js:** Dentro esta la URL del servidor web.
    - **Index.js:** Es el punto de entrada de la aplicación React.

- **.gitignore, README.md:** Ambas pertenecen a la herramienta de Github, .gitignore sirve para no subir al control de versiones algunas carpetas, en este caso node_modules, y README.md para dar información sobre el proyecto en la herramienta.

4.3.2.- Componentes

VirtyRemLab consta de diferentes vistas, hay unas comunes al administrador y al estudiante y otras que son exclusivas del administrador o del estudiante.

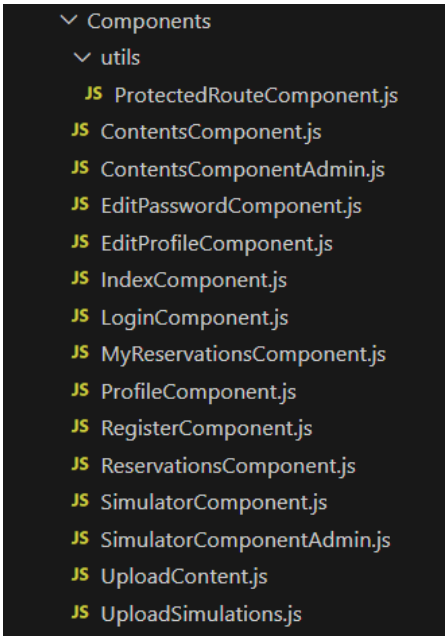

Figura 4.5.- Componentes *Frontend*.

Las vistas comunes son:

- **IndexComponent.js:** Es la vista principal de la aplicación consta de un carrusel que va cambiando de texto, mostrando las distintas funcionalidades que ofrece *VirtyRemLab*.
- **RegisterComponent.js:** Es la vista que el usuario visualiza cuando entra en *VirtyRemLab* para registrarse, está formada por un formulario que tiene campos necesarios para el posterior registro, estos son: nombre, apellidos, correo

electrónico, contraseña y repetir contraseña. También tiene un botón para registrarse.

- **LoginComponent.js:** Es la vista para iniciar sesión, está formado por un formulario con el campo para introducir el correo electrónico y otro para la contraseña y un botón para el inicio de sesión.
- **ProfileComponent.js:** Este componente es lo que el usuario visualiza cuando se quiere ver su perfil, está formado por un formulario con campos deshabilitados que muestran el nombre, apellidos y correo electrónico del usuario, junto a un botón que le redirige a la página para editar su perfil.
- **EditProfileComponent.js:** En esta vista se le permite al usuario editar su perfil, está formada por un formulario con dos campos para cambiar su nombre y apellidos y otro campo deshabilitado mostrando su correo electrónico. Además, cuenta con tres botones, uno de editar donde puede confirmar los cambios que ha hecho, otro para cambiar la contraseña, que le redirige a la vista correspondiente y un último para volver atrás a la vista de su perfil.
- **EditPasswordComponent.js:** Esta vista permite editar la contraseña del usuario, está formada por tres campos, uno para poner la contraseña actual, otro la nueva contraseña y otro para repetir la nueva contraseña. Tiene dos botones uno de editar para confirmar los cambios y otro de atrás para volver a la vista de su perfil.
- **MyReservationsComponent.js:** Este componente muestra una tabla con las distintas reservas que haya hecho el usuario para el laboratorio remoto, muestra el número de reserva, la fecha de comienzo y de finalización, el estado de la reserva junto a un botón para eliminarla. Debajo de la tabla hay un botón que permite hacer una nueva reserva, redirigiendo al usuario a la vista de reservar.
- **ReservationsComponent.js:** Esta vista muestra un calendario semanal con todas las reservas que hayan hecho los usuarios de la página y permite interactuar con el calendario para hacer una nueva reserva.

Las siguientes vistas son exclusivas de los estudiantes:

- **ContentsComponent.js:** Esta vista de estudiante muestra los contenidos existentes mediante unas temáticas que son desplegables, cuando pulsas sobre

ellas aparece un listado de enlaces con todos los contenidos relacionados a dicha temática.

- **SimulatorComponent.js:** Este componente de estudiante muestra las simulaciones existentes mediante unas temáticas que son desplegables, cuando pulsas sobre ellas aparece una tarjeta con todos los contenidos relacionados a dicha temática junto a una imagen que las describe.

Las siguientes vistas son las exclusivas de los administradores.

- **ContentsComponentAdmin.js:** Esta vista exclusiva del administrador es parecida al del estudiante, cuenta con unas temáticas desplegables con todos los contenidos existentes relacionados a dicha temática, pero junto a ellos está un botón para eliminar el contenido.
- **UploadContent.js:** Este componente permite a los administradores subir nuevo contenido, es un formulario con unos campos a rellenar y un botón para proceder a subir el recurso al servidor, estos campos son: nombre del contenido, temática, archivo, si es un contenido eXeLearning y si es así cual es el fichero principal de la carpeta, poniendo por defecto index.html.
- **SimulatorComponentAdmin.js:** Este componente permite visualizar las simulaciones existentes de *VirtyRemLab*. Son unas temáticas desplegables y en ellas aparecen todas las simulaciones relacionadas a la temática junto a un botón para eliminar las simulaciones.
- **UploadSimulations.js:** Permite al administrador subir nuevas simulaciones a la plataforma, consta de un formulario con unos campos a rellenar y un botón para proceder a subir la simulación al servidor. Estos campos son: nombre de la simulación, temática, archivo, fichero principal de este e imagen que describe la simulación.

El componente de **ProtectedRouteComponent.js** no es una vista, si no un componente de seguridad para proteger las URLs exclusivas de los administradores, de forma que los usuarios con rol de estudiante no puedan modificar la URL para acceder a vistas de administrador sin tener dicho rol, redirigiéndoles a la página de inicio de la aplicación.

4.3.3.- Autenticación

En *VirtyRemLab*, además del componente **ProtectedRouteComponent.js** se utilizan dos métodos para proteger las vistas exclusivas de los distintos usuarios, estos dos métodos aseguran que sólo los usuarios con los permisos puedan visualizar las distintas vistas.

Uno de los métodos es *checkLogin()*, este comprueba si el usuario ha iniciado sesión, si el resultado de la función es verdadero entonces permite visualizar los distintos componentes que forman parte de *VirtyRemLab*, sino le redirige a la página de inicio. Esta función comprueba también que el token del usuario siga siendo válido.

El siguiente método es *checkAdmin*(), que comprueba que el usuario tenga el rol de administrador, si la función devuelve verdadero entonces el menú de navegación muestra las vistas exclusivas de los administradores y permite a los usuarios acceder a ellas. Además, también comprueba que el token del usuario siga siendo válido.

Con estos métodos se restringe el acceso solo a los usuarios deseados, estas medidas de seguridad son importantes para proteger la plataforma, Aun así, por si uno de estos métodos falla, el *Backend* cuenta con métodos de validación antes de procesar las solicitudes del *Frontend*.

4.3.4.- Integración con el *Backend*

La capa de presentación y el nivel de aplicación tienen que comunicarse entre sí para poder procesar las distintas solicitudes. En esta sección se explicará como cada uno de los componentes se comunican con el *Backend*.

Para lograr esta comunicación en todos los componentes se utiliza la API Fetch de ReactJS, es decir, es una interfaz que permite obtener información de una URL. Para guardar los datos que devuelven las solicitudes se utilizan diferentes tipos de funciones junto a variables:

- **UseState:** Esta función permite al usuario agregar una variable de estado, es decir, una variable que actualice su información cada vez que esta cambie. Para ello se declara una variable con el nombre y un método para actualizar su estado, este método será al que se invoque cuando se quiera cambiar la información de la variable, y se llamará a la variable cuando se quiera mostrar su información.
- **UseEffect:** Esta función permite sincronizar una variable con un servidor web. Se utiliza para llamar a una función nada más entrar en la vista o cada vez que una variable de estado cambie.
- **UseNavigate:** Esta función permite redirigir al usuario a otras vistas de la aplicación.

A continuación, se explica como cada componente se conecta con el *Backend* para solicitar datos y mostrarlos.

RegisterComponent.js	
Método	ClickRegister
Descripción	Se envía una solicitud POST al *endpoint* "/users" del *Backend* con los datos del nuevo usuario
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se muestra una notificación indicando que el usuario se ha registrado y se redirige a este a la vista de iniciar sesión.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.1.- Fetch de RegisterComponent.js

LoginComponent.js	
Método	ClickLogin
Descripción	Se envía una solicitud POST al *endpoint* "/users/login" del *Backend* con los datos del usuario
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se redirige al usuario a la página principal de la aplicación y se

	almacena su token en localStorage, actualizando la variable setLogin a true.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.2.- Fetch de LoginComponent.js

ProfileComponent.js	
Método	GetUser
Descripción	Se envía una solicitud GET al *endpoint* "/users" del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se muestra la información correspondiente del usuario.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.3.- Fetch de ProfileComponent.js

EditProfileComponent.js	
Método	GetUser
Descripción	Se envía una solicitud GET al *endpoint* "/users" del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se muestra la información correspondiente del usuario.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.4.- Fetch EditProfileComponent.js 1

EditProfileComponent.js	
Método	EditClick
Descripción	Se envía una solicitud PUT al *endpoint* "/users" del *Backend* junto a su token que contiene la identificación del usuario y los datos cambiados del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se muestra una notificación de que la edición ha sido exitosa y se

	redirige al usuario a la vista de su perfil.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.5.- Fetch EditProfileComponent.js 2

EditPasswordComponent.js	
Método	EditPassword
Descripción	Se envía una solicitud PUT al *endpoint* "/users/password" del *Backend* junto a su token que contiene la identificación del usuario, contraseña actual, contraseña nueva y su repetición.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se muestra una notificación de que la edición ha sido exitosa y se redirige al usuario a la vista de su perfil.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.6.- Fetch EditPasswordComponent.js

MyReservationsComponent.js	
Método	GetReservations
Descripción	Se envía una solicitud GET al *endpoint* "/reservations/myReservations" del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se guarda en una variable de estado las reservas del usuario.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.7.- Fetch MyReservationsComponent.js 1

MyReservationsComponent.js	
Método	DeleteReservation
Descripción	Se envía una solicitud DELETE al *endpoint* "/reservations/myReservations" del *Backend* junto a la identificación de la reserva y el token que contiene la identificación

	del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en una variable de estado las reservas actualizadas del usuario.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.8.- Fetch MyReservationsComponent.js 2

ReservationsComponent.js	
Método	GetReservations
Descripción	Se envía una solicitud GET al *endpoint* “/reservations” del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en una variable de estado las reservas de los usuarios.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.9.- Fetch ReservationsComponent.js 1

ReservationsComponent.js	
Método	ClickOK
Descripción	Se envía una solicitud POST al *endpoint* “/reservations” del *Backend* junto a su token que contiene la identificación del usuario y la fecha de comienzo de la reserva.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en la base de datos la nueva reserva y se muestra en el calendario.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 9.10.- Fetch ReservationsComponent.js 2

ContentsComponent.js	
Método	GetFiles
Descripción	Se envía una solicitud GET al *endpoint* “/contents” del *Backend*

	junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en una variable de estado los contenidos.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.11.- Fetch ContentsComponent.js

SimulatorComponent.js	
Método	GetFiles
Descripción	Se envía una solicitud GET al *endpoint* “/simulations” del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en una variable de estado las simulaciones.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.12.- Fetch SimulatorComponent.js

ContentsComponentAdmin.js	
Método	GetFiles
Descripción	Se envía una solicitud GET al *endpoint* “/contents” del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”
Respuesta exitosa	Se guarda en una variable de estado los contenidos.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.13.- Fetch ContentsComponentAdmin.js 1

ContentsComponentAdmin.js	
Método	DeleteFile
Descripción	Se envía una solicitud DELETE al *endpoint* “/contents” del *Backend* junto a la identificación del contenido y el token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con “application/json”

Respuesta exitosa	Se guarda en una variable de estado los contenidos actualizados.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.14.- Fetch ContentsComponentAdmin.js 2

SimulatorComponentAdmin.js	
Método	GetFiles
Descripción	Se envía una solicitud GET al *endpoint* "/simulations" del *Backend* junto a su token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se guarda en una variable de estado las simulaciones.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.15.- Fetch SimulatorComponentAdmin.js 1

SimulatorComponentAdmin.js	
Método	DeleteFile
Descripción	Se envía una solicitud DELETE al *endpoint* "/simulations" del *Backend* junto a la identificación de la simulación y el token que contiene la identificación del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se guarda en una variable de estado las simulaciones actualizadas.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.16.- Fetch SimulatorComponentAdmin.js 1

UploadContent.js	
Método	UploadContent
Descripción	Se envía una solicitud POST al *endpoint* "/contents" del *Backend* junto al token que contiene la identificación del usuario y los datos del nuevo contenido.
Cabeceras	Se indica que los datos son un objeto "FormData"
Respuesta exitosa	Se notifica al usuario de que el contenido se ha subido

	exitosamente.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.17.- Fetch UploadContent.js

UploadSimulator.js	
Método	UploadSimulation
Descripción	Se envía una solicitud POST al *endpoint* "/simulations" del *Backend* junto al token que contiene la identificación del usuario y los datos de la nueva simulación.
Cabeceras	Se indica que los datos son un objeto "FormData"
Respuesta exitosa	Se notifica al usuario de que la simulación se ha subido exitosamente.
Respuesta fallida	Se obtiene el error del *Backend* y se muestra al usuario.

Tabla 4.18.- Fetch UploadSimulator.js

App.js	
Método	*CheckLogin*
Descripción	Se envía una solicitud GET al *endpoint* "/users/*checkLogin"* del *Backend* junto al token del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se actualiza la variable de estado setLogin a true.
Respuesta fallida	Se actualiza la variable de estado setLogin a false.

Tabla 4.19.- Fetch App.js 1

App.js	
Método	*CheckAdmin*
Descripción	Se envía una solicitud GET al *endpoint* "/users/*checkAdmin"* del *Backend* junto al token del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se actualiza la variable de estado setAdmin a true.

Respuesta fallida	Se actualiza la variable de estado setAdmin a false.

Tabla 4.20.- Fetch App.js 2

App.js	
Método	Disconnect
Descripción	Se envía una solicitud GET al *endpoint* "/users/disconnect" del *Backend* junto al token del usuario.
Cabeceras	Se indica que los datos son un JSON con "application/json"
Respuesta exitosa	Se borra el token de localStorage y se redirige al usuario a la página de inicio.

Tabla 4.21.- Fetch App.js 3

4.4.- Implementación del *Backend*

En este apartado se explicará cómo ha sido desarrollada el nivel de aplicación de *VirtyRemLab*, explicando la estructura, los *routers* y la conexión con la base de datos.

4.4.1.- Estructura

La estructura que sigue el *Backend* de *VirtyRemLab* es la siguiente:

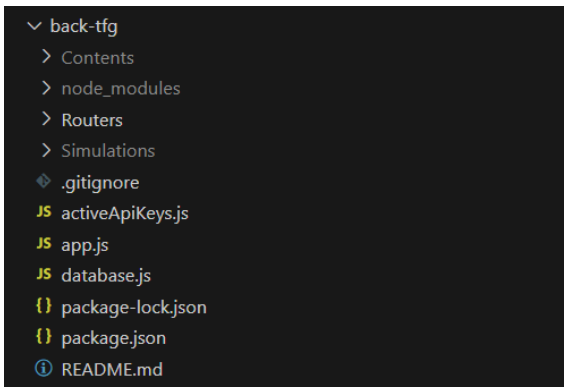

Figura 4.6.- Estructura *Backend*

Cada una de estas carpetas o archivos contiene:

- **Contents, Simulations:** En estas carpetas se guarda cada uno de los contenidos y simulaciones existentes en la base de datos.
- **Node_modules, package-lock.json, package.json:** Estos archivos guardan las librerías, versiones y dependencias del *Backend*.
- ***Routers*:** Contiene los distintos *routers* que conforman el servidor, es decir las distintas APIs con sus respectivos *endpoint*s.
- **ActiveApiKeys.js:** Este archivo contiene los tokens válidos, cada vez que un usuario inicia sesión se guarda su token aquí, y cuando cierra sesión o expira se borra.
- **App.js:** Es el fichero principal del *Backend*, contiene el puerto, ruta de los *Routers*, *middleware* de autenticación y el servidor.
- **Database.js:** Es el fichero que conecta con la base de datos de *VirtyRemLab*.
- **.gitignore,README.md:** Estos archivos son propios de GitHub, .gitignore contiene las carpetas que no se quieren subir al control de versiones, en este caso Contents y Simulations. README.md contiene la información del proyecto para GitHub.

4.4.2.- *Routers*

Un *router* es un elemento propio del *framework* Express para manejar las distintas rutas de la aplicación. En estas rutas están los distintos *endpoint*s, es decir, los distintos métodos HTTP para procesar las solicitudes del *Frontend* y gestionar la información con la base de datos.

VirtyRemLab consta de los siguientes *Routers*:

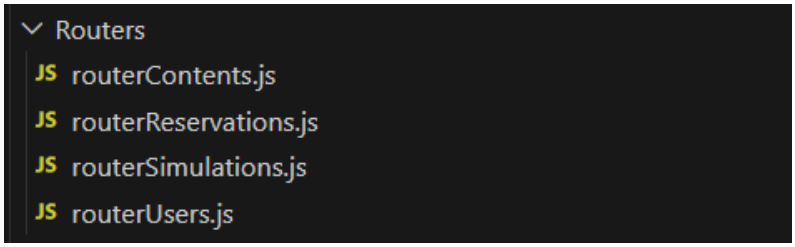

Figura 4.7.- *Routers Backend*

A continuación, se explica cada uno de los *routers*:

***Router*Contents.js**		
Este *router* procesa las solicitudes propias a los contenidos.		
Metodo	***Endpoint***	**Descripción**
GET	/contents	Permite recuperar de la base de datos la información de todos los contenidos.
POST	/contents	Permite insertar un nuevo contenido a la base de datos. Necesita los campos de nombre, temática, archivo, si es un fichero eXeLearning y si es así el nombre del archivo principal.
DELETE	/contents/:id	Permite borrar un contenido de la base de datos, necesita la identificación del contenido.
Métodos auxiliares		
DescomprimirExeLearning		Este método descomprime el archivo eXeLearning para almacenarlo en el servidor
VerifyApiKey		Verifica si el usuario tiene el rol de administrador
HandleUpload		Sube el archivo al servidor mediante la librería Multer.

Tabla 4.22.- *Router*Contents.js

***Router*Reservations.js**		
Este *router* procesa las solicitudes propias a las reservas.		
Metodo	***Endpoint***	**Descripción**
GET	/reservations	Permite recuperar de la base de datos la información de todas las reservas.
POST	/reservations	Permite insertar en la base de datos una nueva reserva.
GET	/reservations/myReservations	Permite recuperar de la base de datos la información de las reservas de un usuario.
DELETE	/reservations/myReservations/:id	Permite borrar una reserva de la base de

		datos, necesita la identificación de la reserva.

Tabla 4.23.- *Router*Reservations.js

***Routers*simulations.js**		
Este *router* procesa las solicitudes propias a las simulaciones.		
Metodo	***Endpoint***	**Descripción**
GET	/simulations	Permite recuperar de la base de datos la información de todas las reservas.
POST	/simulations	Permite insertar una nueva simulación a la base de datos. Necesita los campos de nombre, temática, archivo, fichero principal y una imagen.
DELETE	/simulations/:id	Permite borrar una simulación de la base de datos, necesita la identificación de la simulación.
Métodos auxiliares		
DescomprimirSimulacion		Este método descomprime el archivo para almacenarlo en el servidor
VerifyApiKey		Verifica si el usuario tiene el rol de administrador
HandleUpload		Sube el archivo al servidor mediante la librería Multer.

Tabla 4.24.- *Routers*imulations.js

***Router*Users.js**		
Este *router* procesa las solicitudes propias a los usuarios.		
Metodo	***Endpoint***	**Descripción**
GET	/users	Permite recuperar de la base de datos la información de un usuario.
GET	/users/disconnect	Cierra la sesión de un usuario borrando el token de activeApiKeys.js.
GET	/users/*checkLogin*	Comprueba si el token del usuario está en activeApiKeys.js.

GET	/users/*checkAdmin*	Comprueba si el usuario tiene rol de administrador.
POST	/users	Permite insertar a un nuevo usuario en la base de datos.
POST	/users/login	Permite iniciar sesión a un usuario.
PUT	/users	Permite editar los campos de un usuario.
PUT	/users/password	Permite editar la contraseña de un usuario.
Métodos auxiliares		
Encrypt	Encripta la contraseña antes de añadirla a la base de datos.	
Compare	Compara si la contraseña introducida es igual a la contraseña encriptada.	

Tabla 4.25.- *Router*Users.js

4.4.3.- Conexión con la base de datos

El servidor web se tiene que conectar a la base de datos para recuperar, almacenar o eliminar información de esta, para ello en el *Backend* se dispone de un archivo **database.js**, donde están los métodos para configurar, acceder y salir de la base de datos.

Para configurar la base de datos, se necesitan los siguientes parámetros:

- ***Host:*** Que es la dirección de la máquina en la que está la base de datos, en este caso localhost.
- ***User:*** Es el usuario que tiene acceso a la base de datos.
- ***Password:*** Es la contraseña del usuario que tiene acceso a la base de datos.
- ***Database:*** Es la base de datos con la que se quiere conectar, en este caso laboratory.
- ***MultipleStatements:*** Por si se quiere permitir las multiconsultas, de esta forma si se envían todas las consultas a la vez, se ahorra tiempo, aunque requiere un manejo especial, en este caso está en verdadero.

El fichero también dispone de los siguientes métodos para tratar con la base de datos.

Database.js	
Método	**Descripción**
Connect	Permite conectarse a la base de datos, creando una conexión con esta.
Disconnect	Se desconecta de la base de datos, cerrando la conexión con esta.

Tabla 4.26.- database.js

A continuación, se muestra como son las consultas para acceder a la base de datos y recuperar o almacenar información en cada uno de los *routers*.

***Router*Contents.js**	
Método	GET
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT * FROM contents ORDER BY thematic	Recupera los contenidos ordenados por temática.
Disconnect	Se desconecta de la base de datos.
Método	POST
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
INSERT INTO contents (userId, name, thematic, path, size, exeLearning,mainFile) VALUES (?, ?, ?, ?, ?, ?)	Inserta la ruta al nuevo contenido eXeLearning detallando los campos y los valores.
INSERT INTO contents (userId, name, thematic, path, size, exeLearning) VALUES (?, ?, ?, ?, ?, ?)	Inserta la ruta al contenido (imagen, PDF) detallando los campos y los valores
Disconnect	Se desconecta de la base de datos.
Método	DELETE
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT path FROM contents WHERE id=?	Comprueba si el contenido está en la

	base de datos
DELETE FROM contents WHERE id=?	Si está entonces lo borra.
Disconnect	Se desconecta de la base de datos.

Tabla 4.27.- Conexión base de datos – *router*Contents.js

***Router*Reservations.js**	
Método	GET
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT * FROM reservations	Recupera todas las reservas.
Disconnect	Se desconecta de la base de datos.
Método	GET
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT * FROM reservations WHERE idUser=?	Recupera todas las reservas de un usuario específico.
Disconnect	Se desconecta de la base de datos.
Método	DELETE
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT idUser FROM reservations WHERE id=?	Comprueba si el usuario tiene una reserva con esa identificación.
DELETE FROM reservations WHERE id=?	Si está entonces la borra.
Disconnect	Se desconecta de la base de datos.
Método	POST
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
INSERT INTO simulations (userId, startDate, finishDate, creationDate) VALUES (?, ?, ?, ?)	Inserta una nueva reserva en la base de datos.
Disconnect	Se desconecta de la base de datos.

Tabla 4.28.- Conexión base de datos – *router*Reservations.js

***Routers*simulations.js**	
Método	GET
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT * FROM simulations ORDER BY thematic,name	Recupera las simulaciones ordenadas por temática y nombre.
Disconnect	Se desconecta de la base de datos.
Método	POST
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
INSERT INTO simulations (userId, name, thematic,mainFile, path, imagePath, size) VALUES (?, ?, ?, ?, ?, ?, ?)	Inserta la ruta a la nueva simulación indicando todos los campos y su respectivo valor.
Disconnect	Se desconecta de la base de datos.
Método	DELETE
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT path, imagePath FROM simulations WHERE id=?	Comprueba si la ruta de la simulación y de su imagen está en la base de datos
DELETE FROM simulations WHERE id=?	Si está entonces lo borra.
Disconnect	Se desconecta de la base de datos.

Tabla 4.29.- Conexión base de datos – routerSimulations.js

***Router*Users.js**	
Método	GET
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT * FROM users WHERE id=?	Recupera la información de un usuario

	concreto.
Disconnect	Se desconecta de la base de datos.
Método	POST
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT email FROM users WHERE email = ?	Comprueba que no haya un usuario con el mismo email.
INSERT INTO users (name,surname,email,password) VALUES (?, ?, ?, ?)	Si no lo hay entonces inserta el nuevo usuario en la base de datos.
Disconnect	Se desconecta de la base de datos.
Método	POST
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT id, email,password, rol FROM users WHERE email = ?	Se selecciona el usuario con el email introducido para comprobar la contraseña con la introducida.
Disconnect	Se desconecta de la base de datos.
Método	PUT
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
UPDATE users SET name = ?, surname = ?, email = ? WHERE id = ?	Actualiza los campos de nombre, apellidos en el usuario concreto.
Disconnect	Se desconecta de la base de datos.
Método	PUT
Procedimiento	**Descripción**
Connect	Se conecta a la base de datos.
SELECT password FROM users WHERE id=?	Recupera la contraseña del usuario para después compararla con la introducida.
UPDATE users SET password = ? WHERE id=?	Si la contraseña es correcta entonces

	actualiza la nueva contraseña en la base de datos.
Disconnect	Se desconecta de la base de datos.

Tabla 4.30.- Conexión base de datos – *router*Users.js

4.4.4.- Validación y autenticación

Antes de procesar solicitudes del *Frontend* hay que validar los datos que han llegado junto a la solicitud. Manejar errores es importante para evitar que el servidor falle por un dato erróneo o una respuesta fallida. Por ello todos los *routers* antes de conectar con la base de datos tienen que confirmar que la información sea correcta. Seguidamente se enumeran las validaciones que hace cada *router*, para impedir que el servidor falle.

- ***Router*Contents**:
    - Comprobar que el usuario que inserta y elimina contenido sea administrador.
    - Que el archivo indicado tenga todos los campos rellenados.
    - Que se hayan podido recuperar de la base de datos todos los contenidos sin dar error.
    - Que el contenido se haya insertado o borrado sin problemas tanto en la base de datos como en el servidor.

- ***Router*Reservations:**
    - Que se hayan podido recuperar de la base de datos las reservas sin problemas.
    - A la hora de borrar una reserva hay que comprobar que sea el usuario que ha hecho la reserva el que la borre.
    - Que la reserva se haya borrado correctamente.
    - Que cuando se vaya a hacer una nueva reserva, la fecha de comienzo no esté ya ocupada y que no sea anterior a la fecha actual.

- ***Routers*simulations:**
    - Comprobar que el usuario que inserta y elimina simulaciones sea administrador.
    - Que el archivo indicado tenga todos los campos rellenados.
    - Que se hayan podido recuperar de la base de datos todas las simulaciones sin dar error.
    - Que la simulación se haya insertado o borrado sin problemas tanto en la base de datos como en el servidor.

- ***Router*Users:**
    - A la hora de mostrar la información de un usuario hay que comprobar que este exista y que se han recuperado la información de la base de datos.
    - Cuando se registra un nuevo usuario hay que comprobar que ha rellenado todos los campos, que el correo electrónico sea el corporativo de la Universidad de Oviedo, que la contraseña sea mayor de 5 caracteres, que coincida con la confirmación de contraseña y que el correo electrónico no exista ya en la base de datos. También hay que comprobar que el usuario se ha insertado correctamente en la base de datos.
    - Cuando un usuario inicia sesión hay que comprobar que ha rellenado todos los campos y que la contraseña sea correcta, además de que se haya podido recuperar la información de la base de datos.
    - Que cuando se edita el perfil o la contraseña los datos introducidos sean correctos y estén todos rellenados.

También es importante proteger las rutas con los tokens, para esto están los tokens de *JSON Web Token*, esta es la librería que se utiliza para generar estos tokens. Cuando un usuario inicia sesión se crea automáticamente este token que guarda los siguientes datos: identificación del usuario, email y el rol de este.

Cada vez que un usuario solicita un servicio al *Backend*, este comprueba que el token que le llega con la solicitud sea válido, comprobando que esté en la lista de activeApiKeys.js,

de esta forma se evita que un usuario que no esté registrado o no haya iniciado sesión pueda entrar a las funcionalidades de la plataforma.

En el caso de las funcionalidades exclusivas de los administradores, si un usuario estudiante consigue evitar la comprobación del *Frontend* y accede al componente, el *Backend* cuando le llega la solicitud del estudiante y comprueba que su rol no es de administrador no le permite llevar a cabo la funcionalidad, mostrándole que su rol no es de administrador.

De esta forma se protegen las rutas de los usuarios no deseados asegurándose de que solo los usuarios con permisos puedan acceder a la aplicación *VirtyRemLab*.

4.5.- Implementación de la base de datos

En esta sección se va a explicar cómo ha sido llevada a cabo la implementación de la capa de datos de *VirtyRemLab*, enseñando la estructura y las tablas que forman la base de datos de la plataforma.

4.5.1.- Estructura y tablas

La base de datos de esta aplicación web está formada por cuatro tablas, en el siguiente esquema se muestra la estructura y las relaciones que hay entre las tablas.

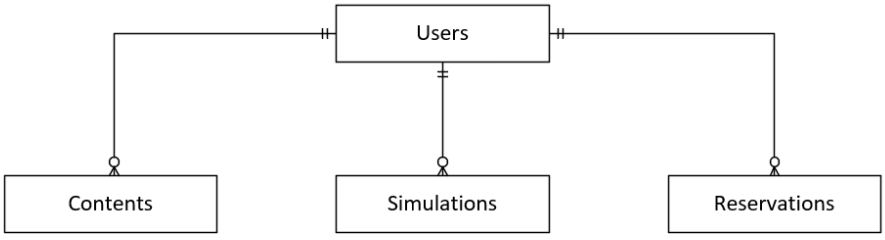

Figura 4.8.- Estructura de la base de datos.

A continuación, se explica cada tabla, junto a sus campos y las llaves.

Users			
Almacena los datos de los usuarios registrados en *VirtyRemLab*.			
Campo	**Tipo**	**Llave**	**Descripción**
ID	Entero	Primaria	Identificación del usuario.
Name	Varchar	-	Nombre del usuario.
Surname	Varchar	-	Apellidos del usuario.
Email	Varchar	-	Correo electrónico del usuario.
Password	Texto	-	Contraseña del usuario.
Rol	Varchar	-	Rol del usuario, por defecto *Student*.

Tabla 4.31.- Tabla users

Contents			
Almacena los datos de los contenidos existentes en *VirtyRemLab*.			
Campo	**Tipo**	**Llave**	**Descripción**
ID	Entero	Primaria	Identificación del contenido.
UserID	Entero	Foránea	Usuario que ha subido el contenido.
Name	Varchar	-	Nombre del contenido.
Thematic	Varchar	-	Temática del contenido.
Path	Texto	-	Donde está alojado el contenido.
Size	Float	-	Tamaño del contenido.
ExeLearning	Char	-	Si el contenido es eXeLearning o no.
MainFile	Text	-	En caso de que sea eXeLearning fichero principal, por defecto index.html

Tabla 4.32.- Tabla contents

Simulations			
Almacena los datos de las simulaciones existentes en *VirtyRemLab*.			
Campo	**Tipo**	**Llave**	**Descripción**

ID	Entero	Primaria	Identificación de la simulación.
UserID	Entero	Foránea	Usuario que ha subido la simulación.
Name	Varchar	-	Nombre de la simulación.
Thematic	Varchar	-	Temática de la simulación.
MainFile	Text	-	Fichero principal de la simulación.
Path	Texto	-	Donde está alojado la simulación.
imagePath	Texto	-	Donde está alojada la imagen que describe la simulación.
Size	Float	-	Tamaño de la simulación.

Tabla 4.33.- Tabla simulations

Reservations			
Almacena los datos de las reservas en *VirtyRemLab*.			
Campo	**Tipo**	**Llave**	**Descripción**
ID	Entero	Primaria	Identificación de la reserva.
UserID	Entero	Foránea	Usuario que ha realizado la reserva.
StartDate	TimeStamp	-	Fecha de comienzo de la reserva.
FinishDate	TimeStamp	-	Fecha de finalización de la reserva.
CreationDate	TimeStamp	-	Fecha de creación de la reserva.

4.34.- Tabla reservations

Las tablas que se han explicado anteriormente están relacionadas entre sí de la siguiente forma:

Contents-Users	
Tipo de relación	N a 1
Descripción	Un contenido puede estar subido por solo un usuario, pero varios contenidos pueden estar subidos por el mismo usuario.

Tabla 4.35.- Relación contents-users

Simulations-Users	
Tipo de relación	N a 1
Descripción	Una simulación puede estar subida por solo un usuario, pero varias simulaciones pueden estar subidas por el mismo usuario.

Tabla 4.36.- Relación simulations-users

Reservations-Users	
Tipo de relación	N a 1
Descripción	Una reserva puede estar realizada por solo un usuario, pero varias reservas pueden estar realizadas por el mismo usuario.

Tabla 4.37.- Relación reservations-users

5. Pruebas

El apartado de pruebas es una fase muy importante al desarrollar un producto *software*, garantiza que el sistema funcione correctamente y que cumpla con los requisitos establecidos. Durante la fase de desarrollo de *VirtyRemLab* se han realizado pruebas unitarias para verificar que cada componente del *Backend* y del *Frontend* funcione correctamente de forma aislada, estas pruebas han sido cruciales para identificar y corregir distintos errores que han surgido durante la fase de implementación. Algunas de estas pruebas son:

- Solo se puede registrar un usuario con el correo corporativo de la Universidad de Oviedo y que no esté ya en la base de datos.
- Solo pueden iniciar sesión los usuarios registrados.
- Se listan correctamente los contenidos y las simulaciones existentes.
- Se suben correctamente los nuevos contenidos y las simulaciones.
- Se puede editar el nombre y el apellido y la contraseña correctamente.
- Se puede cerrar sesión correctamente.

Esta sección se centrará en las pruebas de integración y funcionalidad. Las pruebas de integración verificarán que las distintas capas de la plataforma se comunican correctamente y las de funcionalidad que la aplicación cumple con todos los requisitos funcionales especificados. La herramienta utilizada para todas las pruebas ha sido Visual Studio Code y su depurador.

5.1.- Plan de pruebas y ejecución

- P1.- Registrar un nuevo usuario
    - P1.1.- Con el correo no corporativo de la Universidad de Oviedo.
    - P1.2.- Con un correo que ya esté en la base de datos.
    - P1.3.- Sin rellenar todos los campos.

- P1.4.- Con todos los campos correctos.

Prueba	Respuesta esperada	Respuesta obtenida
P1.1	Error, se debe de utilizar el correo corporativo de la Universidad de Oviedo.	Error, se debe de utilizar el correo corporativo de la Universidad de Oviedo.
P1.2	Error, no se ha podido registrar al usuario.	Error, no se ha podido registrar al usuario.
P1.3	Error, los campos son obligatorios	Error, los campos son obligatorios
P1.4	Se registra el usuario correctamente	Se registra el usuario correctamente

Tabla 5.1.- Pruebas de registro de usuario

- P2.- Inicio de sesión
  - Usuario no registrado
  - Usuario registrado

Prueba	Respuesta esperada	Respuesta obtenida
P2.1	Error, no se ha podido iniciar sesión.	Error, no se ha podido iniciar sesión.
P2.2	Se inicia sesión correctamente.	Se inicia sesión correctamente.

Tabla 5.2.- Pruebas de inicio de sesión

- P3.- Contenidos
  - P3.1.- Listar contenidos siendo administrador o estudiante.
  - P3.2.- Acceder a contenidos siendo administrador o estudiante.
  - P3.3.- Eliminar contenido siendo un estudiante.
  - P3.4.- Eliminar contenido siendo un administrador.
  - P3.5.- Subir contenido siendo un estudiante.
  - P3.6.- Subir contenido siendo un administrador.

Prueba	Respuesta esperada	Respuesta obtenida

P3.1	Se listan los contenidos correctamente.	Se listan los contenidos correctamente.
P3.2	Se abren los contenidos correctamente.	Se abren los contenidos correctamente.
P3.3	Error, usted no tiene rol de administrador.	Error, usted no tiene rol de administrador.
P3.4	Se elimina el contenido correctamente.	Se elimina el contenido correctamente.
P3.5	Error, usted no tiene rol de administrador.	Error, usted no tiene rol de administrador.
P3.6	Se sube el contenido correctamente.	Se sube el contenido correctamente.

Tabla 5.3.- Pruebas de contenidos

- P4.- Simulaciones
    - P4.1.- Listar simulaciones siendo administrador o estudiante.
    - P4.2.- Acceder a simulaciones siendo administrador o estudiante.
    - P4.3.- Eliminar simulación siendo un estudiante.
    - P4.4.- Eliminar simulación siendo un administrador.
    - P4.5.- Subir simulación siendo un estudiante.
    - P4.6.- Subir simulación siendo un administrador.

Prueba	Respuesta esperada	Respuesta obtenida
P4.1	Se listan las simulaciones correctamente.	Se listan las simulaciones correctamente.
P4.2	Se accede a la simulación correctamente.	Se accede a la simulación correctamente.
P4.3	Error, usted no tiene rol de administrador.	Error, usted no tiene rol de administrador.
P4.4	Se elimina la simulación correctamente.	Se elimina la simulación correctamente.

P4.5	Error, usted no tiene rol de administrador.	Error, usted no tiene rol de administrador.
P4.6	Se sube la simulación correctamente.	Se sube la simulación correctamente.

Tabla 5.4.- Pruebas de simulaciones

- P5.- Reservas
    - P5.1.- Listar todas las reservas.
    - P5.2.- Eliminar reserva.
    - P5.3.- Hacer nueva reserva siendo la fecha de comienzo anterior a la actual.
    - P5.4.- Hacer una reserva en una fecha en la que ya había una reserva hecha.
    - P5.5.- Hacer una nueva reserva correctamente.

Prueba	Respuesta esperada	Respuesta obtenida
P5.1	Se listan las reservas correctamente.	Se listan las reservas correctamente.
P5.2	Se elimina la reserva correctamente.	Se elimina la reserva correctamente.
P5.3	Error, la fecha de comienzo es anterior a la actual.	Error, la fecha de comienzo es anterior a la actual.
P5.4	Error, ya existe una reserva en esa fecha	Error, ya existe una reserva en esa fecha
P5.5	Se hace la reserva correctamente.	Se hace la reserva correctamente.

Tabla 5.5.- Pruebas de reservas

- P6.- Edición de perfil
    - P6.1.- Editar nombre sólo.
    - P6.2.- Editar apellidos sólo.
    - P6.3.- Editar nombre y apellidos.
    - P6.4.- Editar correo.

Prueba	Respuesta esperada	Respuesta obtenida
P6.1	Se edita correctamente.	Se edita correctamente.
P6.2	Se edita correctamente.	Se edita correctamente.
P6.3	Se edita correctamente.	Se edita correctamente.
P6.4	No se puede.	No se puede.

Tabla 5.6.- Pruebas de edición de perfil

- P7.- Edición de contraseña
    - P7.1.- Rellenar la contraseña actual mal.
    - P7.2.- Que la contraseña y su repetición no sean iguales.
    - P7.3.- Rellenar campos correctamente.

Prueba	Respuesta esperada	Respuesta obtenida
P7.1	Error, la contraseña actual no coincide.	Error, la contraseña actual no coincide.
P7.2	Error, las contraseñas no son iguales.	Error, las contraseñas no son iguales.
P7.3	Se edita la contraseña correctamente.	Se edita la contraseña correctamente.

Pruebas de edición de contraseña

- P8.- Cierre de sesión
    - P8.1.- Cerrar sesión.

Prueba	Respuesta esperada	Respuesta obtenida
P8.1	Se cierra sesión correctamente.	Se cierra sesión correctamente.

Tabla 5.8.- Pruebas de cierre de sesión

- P9.- Cambio de idioma
    - P9.1.- Cambiar el idioma a inglés.
    - P9.2.- Cambiar el idioma a español.

Prueba	Respuesta esperada	Respuesta obtenida
P9.1	Se cambia el idioma a inglés.	Se cambia el idioma a inglés.
P9.2	Se cambia el idioma a español.	Se cambia el idioma a español.

Tabla 5.9.- Pruebas de cambio de idioma

6. Planificación

A continuación, se muestra el diagrama de Gantt de las tareas principales del proyecto.

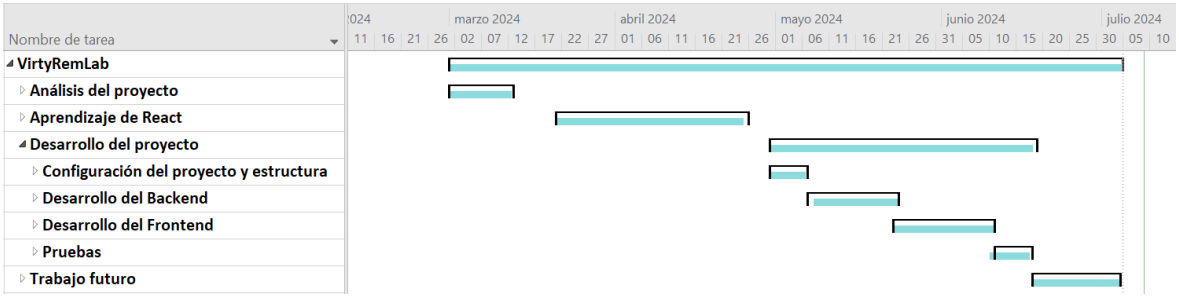

Figura 7.1.- Diagrama de Gantt

Se agrega también una tabla con la planificación detallada del desarrollo del proyecto *VirtyRemLab,* este proceso comienza el 1 de marzo del 2024 y termina el 4 de julio de 2024, sumando un total de 720 horas.

Nombre de tarea	Duración	Comienzo	Fin
VirtyRemLab	**720 horas**	**vie 01/03/24**	**jue 04/07/24**
Análisis del proyecto	**64 horas**	**vie 01/03/24**	**mar 12/03/24**
Objetivos del proyecto	12 horas	vie 01/03/24	lun 04/03/24
Obtención de requisitos	18 horas	lun 04/03/24	mié 06/03/24
Identificación de actores y casos de uso	14 horas	mié 06/03/24	jue 07/03/24
Estudio de alternativas	20 horas	vie 08/03/24	mar 12/03/24
Aprendizaje de React	**208 horas**	**jue 21/03/24**	**jue 25/04/24**
Tutoriales de React	160 horas	jue 21/03/24	mié 17/04/24
Desarrollo de proyectos pequeños	48 horas	mié 17/04/24	mié 24/04/24
Desarrollo del proyecto	**288 horas**	**mar 30/04/24**	**mar 18/06/24**
Configuración del proyecto y estructura	**40 horas**	**mar 30/04/24**	**lun 06/05/24**
Configuración del entorno	20 horas	mar 30/04/24	jue 02/05/24
Definición de la estructura del proyecto	20 horas	jue 02/05/24	lun 06/05/24
Desarrollo del Backend	**104 horas**	**mar 07/05/24**	**jue 23/05/24**
Implementación de la base de datos	48 horas	mié 08/05/24	mié 15/05/24
Desarrollo de los routers	56 horas	mié 15/05/24	jue 23/05/24
Desarrollo del Frontend	**104 horas**	**jue 23/05/24**	**lun 10/06/24**
Desarrollo de componentes	72 horas	jue 23/05/24	mar 04/06/24
Integración con el backend	32 horas	mié 05/06/24	lun 10/06/24

Pruebas	**40 horas**	**lun 10/06/24**	**lun 17/06/24**
Pruebas unitarias	10 horas	lun 10/06/24	mar 11/06/24
Pruebas de integración	20 horas	mié 12/06/24	vie 14/06/24
Pruebas de uso	10 horas	vie 14/06/24	lun 17/06/24
Trabajo futuro	**100 horas**	**mar 18/06/24**	**jue 04/07/24**
Desarrollo Backend	50 horas	mar 18/06/24	mié 26/06/24
Desarrollo Frontend	**50 horas**	**mié 26/06/24**	**jue 04/07/24**
Creación de vista de laboratorio remoto	40 horas	mié 26/06/24	mar 02/07/24
Adaptar componente de reservas	10 horas	mié 03/07/24	jue 04/07/24

Tabla 7.1.- Planificación *VirtyRemLab*

7. Presupuesto

A continuación, se mostrará el presupuesto de cada una de las partes que forman la planificación, junto con sus costes y horas totales. El proyecto ha sido llevado a cabo en su completitud por una sola persona.

Recurso	Salario bruto / hora (€)	Coste / hora (€)
Programadora junior	20	35

Tabla 8.1.- Coste persona

7.1.- Parte 1: Análisis del proyecto

Tarea	Duración (horas)	Total (€)
Objetivos del proyecto	12	525
Obtención de requisitos	18	630
Identificación de actores y casos de uso	14	490
Estudio de alternativas	20	700
Coste Total Análisis del proyecto	**64**	**2240**

Tabla 8.2.- Presupuesto análisis del proyecto

7.2.- Parte 2: Costes de formación específica para el desarrollo del proyecto

Tarea	Duración (horas)	Total (€)
Costes de formación específica para el	**208**	**7280**

desarrollo del proyecto		
Turoriales de React	160	5600
Desarrollo de proyectos	48	1680

Tabla 8.3.- Presupuesto de costes de formación

7.3.- Parte 3: Desarrollo del proyecto

Tarea	**Duración (horas)**	**Total (€)**
Desarrollo del proyecto	**288**	**10080**
Configuración del proyecto y estructura	**40**	**1400**
Configuración del entorno	20	700
Definición de la estructura del proyecto	20	700
Desarrollo del *Backend*	**104**	**3640**
Implementación de la base de datos	48	1680
Desarrollo de los *routers*	56	1960
Desarrollo del *Frontend*	**104**	**3640**
Desarrollo de componentes	72	2520
Integración con el *Backend*	32	1120
Pruebas	**40**	**1400**
Pruebas unitarias	10	350
Pruebas de integración	20	700
Pruebas de uso	10	350

Tabla 8.4.- Presupuesto desarrollo del proyecto

7.4.- Parte 4: Trabajo futuro

Tarea	Duración (horas)	Total (€)
Desarrollo del proyecto	**100**	**3500**
Desarrollo del *Backend*	**50**	**1750**
Desarrollo del *Frontend*	**50**	**1750**
Creación componente laboratorio remoto	40	1400
Adaptación reservas	10	350

Tabla 8.5.- Presupuesto trabajo futuro

7.5.- Resumen y total del Presupuesto

Parte	Nombre	Total (€)
1	Análisis del proyecto	2240
2	Aprendizaje de React	7280
3	Desarrollo del proyecto	10080
4	Trabajo futuro	3500
Total de las partes		**23100**
IVA (21%)		**4841**
Total del presupuesto		**27951**

Tabla 8.6.- Presupuesto total

8. Conclusiones

Con la realización de este proyecto se ha conseguido desarrollar una aplicación web para el proyecto de innovación docente de la Universidad de Oviedo, llamado *VirtyRemLab.*

Este desarrollo supone un avance significativo para la formación híbrida en Ingeniería, y facilita enormemente el aprendizaje *online* de los estudiantes, permitiéndoles seguir estudiando desde cualquier sitio con recursos tanto teóricos como prácticos.

El desarrollo de este proyecto también ha supuesto varios desafíos para la estudiante, como el aprendizaje de la herramienta de React, o la implementación de seguridad mediante los JSON Web Tokens para asegurar la autenticación y autorización de los usuarios en la plataforma, además de la gestión de la base de datos e integración de las distintas capas de la aplicación. Todos estos desafíos han supuesto valiosos aprendizajes para la autora que ha tenido la oportunidad de desarrollar la aplicación web *VirtyRemLab.*

Además, con el desarrollo de esta plataforma se han mejorado los conocimientos relacionados con la implementación de aplicaciones web como el uso de tecnologías web, aumentando los conocimientos de JavaScript o SQL y adquiriendo nuevos como ReactJS o NodeJS.

Aunque este proyecto todavía sigue en ejecución, la plataforma web desarrollada supone un avance significativo del proyecto *VirtyRemLab*, completando dos de las tareas principales definidas en el proyecto. La implementación de distintas tecnologías, pruebas exhaustivas y un enfoque centrado en el estudiante ha permitido desarrollar una plataforma robusta y eficiente, pasando por todas las fases del desarrollo de *software* e incluyendo distintas disciplinas como las tecnologías web, desarrollo de APIs y base de datos relacionales.

9. Trabajo futuro

Como trabajo futuro queda implementar la funcionalidad del laboratorio remoto, para que los estudiantes de Ingeniería puedan interactuar con los distintos experimentos disponibles. Este trabajo futuro se podría dividir en dos partes:

La primera parte sería implementar la conexión de los laboratorios remotos al servidor web. Para ello se propone el uso de plataformas *hardware Raspberry* Pi, que comunicarán los prototipos de laboratorio remotos con la aplicación web mediante el protocolo MQTT. Para ello habría que implementar un *broker* MQTT, como Mosquitto, en el *Backend* para gestionar la comunicación.

La segunda parte sería desarrollar la vista del laboratorio remoto, donde el usuario podrá encontrar un hipervínculo a los distintos experimentos disponibles, pero, para poder acceder a este componente el usuario deberá tener una reserva hecha y que la fecha de comienzo coincida con la fecha actual, por lo que habrá que conectar también el sistema de reservas al componente del laboratorio remoto.

En resumen, el trabajo futuro de *VirtyRemLab* se centrará en implementar la funcionalidad del laboratorio remoto, conectándose a los experimentos mediante el protocolo MQTT y gestionando el acceso a estos experimentos con reservas. Esta ampliación del proyecto permitirá a los usuarios tener una experiencia de aprendizaje más interactiva y *VirtyRemLab* será una plataforma completa cumpliendo así todos los objetivos del proyecto de innovación docente de la Universidad de Oviedo.

10. Referencias

[1] *Laboratorio virtual para la enseñanza de la química*. Recuperado el 10 de junio de 2024, de https://labvirtualquimica.weebly.com/historia.html

[2] *WebLab Deusto.* Recuperado el 10 de junio de 2024, de https://weblab.deusto.es/olarex/cd/UD/Incubator_EN_final/remote_laboratory_definition.html

[3] *NetLab.* Recuperado el 10 de junio de 2024, de https://netlab.unisa.edu.au/

[4] *UniLab*. Recuperado el 10 de junio de 2024, de https://unilabs.dia.uned.es/

[5] *WebLab-Deusto*. Recuperado el 10 de junio de 2024, de https://weblab.deusto.es/website/

[6] *GOLDi*. Recuperado el 10 de junio de 2024, de https://www.goldi-labs.net/index.php?Site=1

[7] Lei, Z., Zhou, H.,Wenshan, H., Liu, G. P. (2022). *Flipping Laboratories Toward Future Experimentation Systems.*

[8] *Gluo.* Recuperado el 10 de junio de 2024, de https://www.gluo.mx/blog/*backend*-que-es-y-para-que-sirve

[9] *Oracle*. Recuperado el 10 de junio de 2024, de https://www.oracle.com/es/database/what-is-database/

[10] *ExeLearning.* Recuperado el 10 de junio de 2024, de https://exelearning.net/html_manual/exe_es/qu_es_exelearning.html

[11] *Gluo.* Recuperado el 18 de junio de 2024, de https://www.gluo.mx/blog/arquitectura-de-*software*-que-es-y-que-tipos-hay

[12] *ApiumHub.* Recuperado el 18 de junio de 2024, de https://apiumhub.com/tech-blog-barcelona/benefits-of-*software*-architecture/

[13] Terra J. (2023). *What is Client-Server Architecture? Everything You Should Know.*

[14] Sharanagowda K. (2022). *A Study on the Client Server Architecture and Its Usability.*

[15] Spencer Nguyen. (2023). *Common Types of Software Architecture.*

[16] *Hidden Brains.* Recuperado el 18 de junio de 2024, de https://www.hiddenbrains.com/blog/importance-of-client-server-architecture-in-application-development.html

[17] Abdelrahman Hassan Hamdy. (2024). *System Design N-tier architecture.*

[18] Jose Francisco Ojeda Montoya. (2023). *Aplicaciones N-Capas: Estructurando el Desarrollo de Software de Forma Eficiente.*

[19] *CodeAcademy*. Recuperado el 18 de junio de 2024, de https://www.codecademy.com/resources/blog/what-is-a-*framework*/

[20] AngularJS. Recuperado el 18 de junio de 2024, de https://angularjs.org/

[21] Nihar Raval. (2024). React vs Angular: Which JS *Framework* to choose for Front-end Development?

[22] *React.js.* Recuperado el 18 de junio de 2024, de https://es.react.dev/

[23] *Solbyte.* Recuperado el 18 de junio de 2024, de https://www.solbyte.com/blog/react-js-ventajas-e-inconvenientes/

[24] *Ant Design.* Recuperado el 18 de junio de 2024, de https://ant.design/

[25] *React Bootstrap.* Recuperado el 18 de junio de 2024, de https://react-bootstrap.github.io/

[26] *React Router.* Recuperado el 18 de junio de 2024, de https://react*router*.com/en/main

[27] *React-i18next.* Recuperado el 18 de junio de 2024, de https://react.i18next.com/

[28] *FullCalendar*. Recuperado el 18 de junio de 2024, de https://fullcalendar.io/docs/react

[29] *Spring Boot.* Recuperado el 19 de junio de 2024, de https://spring.io/

[30] *Studocu.* Recuperado el 19 de junio de 2024, de https://www.studocu.com/ec/document/universidad-de-guayaquil/fundamentos-de-programacion/tarea-003-ventajas-y-desventajas/69796013

[31] *Flask.* Recuperado el 19 de junio de 2024, de https://flask.palletsprojects.com/en/latest/

[32] *NodeJS.* Recuperado el 19 de junio de 2024, de https://nodejs.org/en/

[33] Walther. (2023). *Descubre las Ventajas y Desventajas de Node.js para Potenciar tu Sitio Web.*

[34] *Express*. Recuperado el 19 de junio de 2024, de https://expressjs.com/

[35] *JSONWebToken.* Recuperado el 19 de junio de 2024, de https://jwt.io/

[36] *AdmZip.* Recuperado el 19 de junio de 2024, de https://www.npmjs.com/package/adm-zip

[37] *Multer.* Recuperado el 19 de junio de 2024, de https://www.npmjs.com/package/multer

[38] *Oracle*. Recuperado el 19 de junio de 2024, de https://www.oracle.com/es/database/what-is-a-relational-database/

[39] *Ayudaley*. Recuperado el 19 de junio de 2024, de https://ayudaleyprotecciondatos.es/bases-de-datos/no-relacional/#Que_es_una_base_de_datos_no_relacional_Definicion

[40] *PostgreSQL*. Recuperado el 19 de junio de 2024, de https://www.postgresql.org/

[41] *Todo PostgreSQL.* Recuperado el 19 de junio de 2024, de https://www.todopostgresql.com/ventajas-y-desventajas-de-postgresql/

[42] *MySQL.* Recuperado el 19 de junio de 2024, de https://www.mysql.com/

[43] *Oracle.* Recuperado el 19 de junio de 2024, de https://www.oracle.com/es/mysql/what-is-mysql/#one-choice

[44] *Visual Studio Code.* Recuperado el 19 de junio de 2024, de https://code.visualstudio.com/

[45] *GitHub.* Recuperado el 19 de junio de 2024, de https://github.com/

[46] *HeidiSQL.* Recuperado el 19 de junio de 2024, de https://www.heidisql.com/

[47] Deyimar A. (2023). *¿Qué es JSON?*

[48] *Bcrypt*. Recuperado el 9 de julio de 2024, de https://www.npmjs.com/package/bcryptjs

11. Anexo

11.1.- Manual de usuario

A continuación, a modo de ejemplo se detallarán los pasos que ha de seguir un usuario para llevar a cabo las distintas funcionalidades que ofrece *VirtyRemLab.*

11.1.1.- Registro de un nuevo usuario

Una vez se haya accedido a la página principal de *VirtyRemLab,* el usuario tiene que pulsar sobre el botón “Registrarse” que aparece en la barra de navegación superior.

Cuando aparezca el formulario, el usuario debe de rellenar todos los campos de este, en el correo electrónico se debe de poner el correo corporativo de la Universidad de Oviedo. Después de rellenar los campos el usuario debe de pulsar el botón de “Registrarse”.

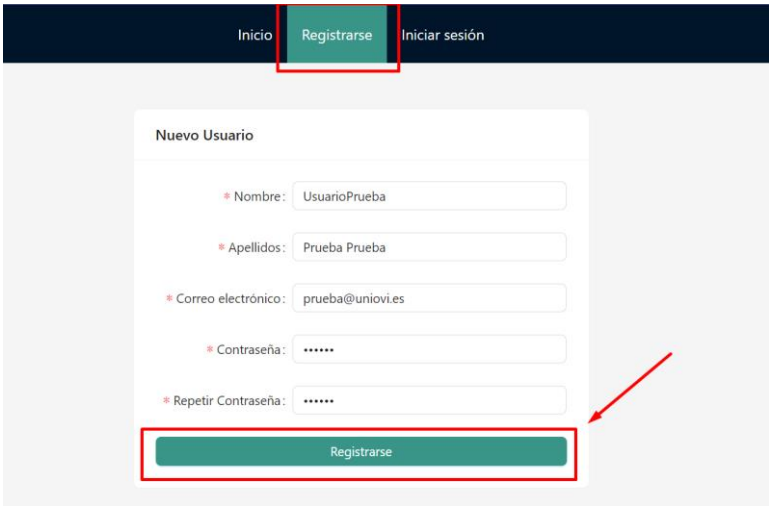

Figura 11.1.- Registro de un nuevo usuario

11.1.2.- Inicio de sesión

Cuando el usuario ya se haya registrado en la aplicación, este deberá de pulsar sobre la sección de “Iniciar sesión” que hay sobre la barra de navegación superior.

Una vez que aparezca el formulario, el usuario debe de rellenar los campos con las credenciales que ha utilizado a la hora de registrarse en la plataforma, después tiene que pulsar sobre el botón de “Iniciar sesión”.

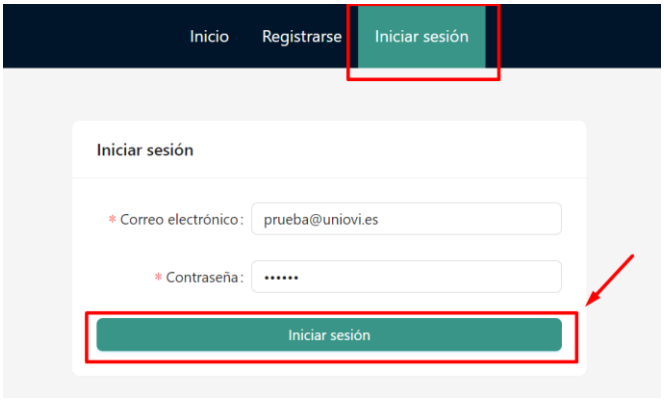

Figura 11.2.- Inicio de sesión de un usuario

Para las siguientes funcionalidades se asumirá que el usuario ha iniciado sesión en la aplicación web.

11.1.3.- Visualización de contenidos

El usuario debe de pulsar sobre el botón de “Contenidos” en la barra superior. Cuando aparezca el listado de los distintos contenidos existentes en la plataforma. Se debe de pinchar sobre la temática que se desee, seguidamente pulsar en el enlace del contenido disponible de esta.

Figura 11.3.- Visualización de contenidos

11.1.4.- Subir nuevo contenido y gestión de contenidos

Esta funcionalidad es exclusiva del usuario administrador y debe de pulsar en el apartado de “Subir contenido” disponible en la sección de “Contenidos” en la barra de navegación superior.

Cuando aparezca el formulario el profesor debe de rellenar los campos de nombre, temática y archivo obligatoriamente. Hay un *checkbox* que sólo se debe de rellenar en el caso de que el contenido sea una carpeta eXeLearning, donde además se debe de indicar el nombre del archivo principal, que por defecto es index.html. Después debe de pulsar el botón de “Subir archivo”.

Figura 11.4.1.- Subir contenido

En el caso de que el usuario administrador quiera borrar un contenido de la plataforma este tendrá que acceder al apartado de “Contenidos” de la barra de navegación.

El profesor accederá a la temática donde esté el contenido que quiera eliminar y pulsará el botón de “Eliminar” que está junto al hipervínculo.

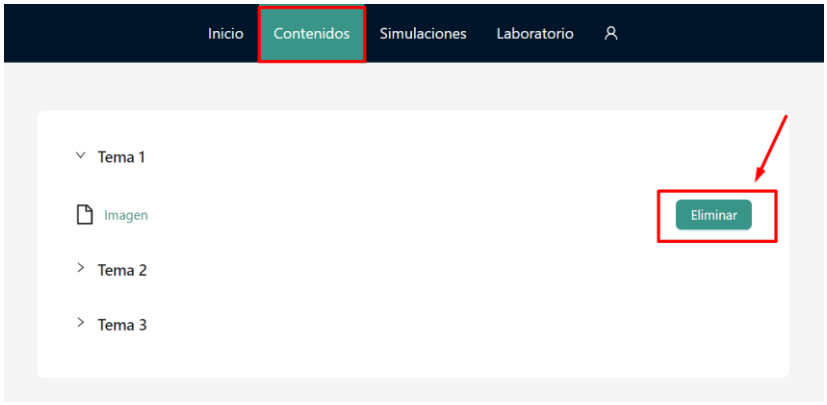

Figura 11.4.2.- Eliminar contenido

11.1.5.- Visualización de simulaciones

El usuario debe de pulsar sobre la sección “Simulaciones” disponible en la barra de navegación superior de la aplicación. Cuando aparezcan las simulaciones disponibles el usuario debe de pulsar sobre la temática que desee y en la simulación que quiera visualizar.

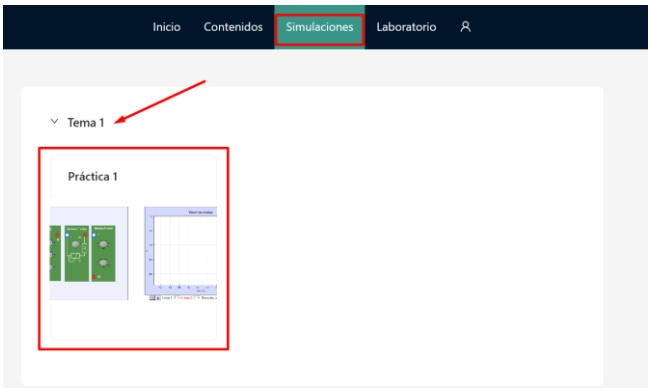

Figura 11.5.- Visualización de simulación.

11.1.6.- Subir nueva simulación y gestión de simulaciones

Esta función es exclusiva de los administradores, para acceder a ella tienen que pulsar sobre el botón de "Subir simulación" que está en el apartado de "Simulaciones" de la barra de navegación superior.

Cuando haya cargado el componente, aparecerá un formulario donde el profesor tiene que rellenar todos los campos y seguidamente pulsar el botón de "Subir simulación".

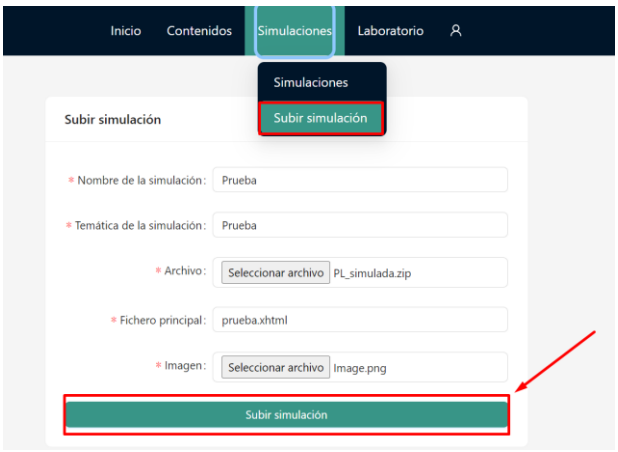

Figura 11.6.1.- Subir simulación

Cuando un profesor quiera borrar una simulación debe de acceder al apartado de "Simulaciones". En este componente el profesor debe de pulsar en la temática que esté relacionada con la simulación que este desee borrar. Aparecerá una tarjeta con las distintas simulaciones y debajo de ellas un botón de "Eliminar", que deben de pulsar.

Figura 11.6.2.- Eliminar simulación

11.1.7.- Visualización de reservas

El usuario debe de pulsar en la subsección de “Reservar” disponible en la sección de “Laboratorio” en la barra de navegación superior. Seguidamente se mostrará un calendario semanal con las reservas programadas. Para hacer una nueva reserva se debe de pulsar en un hueco libre del calendario, rellenar los datos y pulsar en el botón “OK”. En el caso de querer cambiar de semana se pulsará en las flechas de la esquina superior derecha, teniendo en cuenta que no se puede reservar el laboratorio en una fecha anterior a la actual.

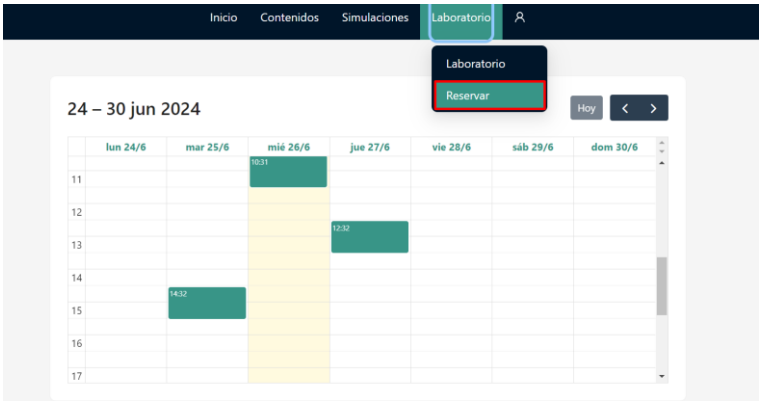

Figura 11.7.1.- Reservas

Cuando el usuario desee de gestionar una reserva propia, deberá de acceder a la sección de “Mis reservas” disponible en el icono del menú de navegación superior. Aparecerá una tabla donde podrá ver todas sus reservas y en el caso de querer eliminar alguna debe de pulsar en el botón de “Eliminar”, en el caso de querer hacer una nueva reserva deberá pulsar en el botón de “Nueva reserva” y el sistema le redirigirá a la figura 11.7.1.

Figura 11.7.2.- Reservas de un usuario

11.1.8.- Visualización de perfil

Para visualizar el perfil de un usuario este tiene que acceder mediante el botón de “Ver Perfil” disponible en el icono del menú de navegación superior.

Figura 11.8.- Ver perfil de usuario

11.1.9.- Edición perfil y de contraseña

Cuando un usuario desee cambiar el nombre o apellidos de su cuenta, este tiene que pulsar en el botón de “Editar perfil” disponible en el componente de “Ver perfil” bajo la información de este. Aparecerá un formulario con el campo de nombre y apellidos para cambiar, también aparecerá el correo electrónico, pero este no se puede cambiar. Cuando se hayan hecho los cambios se pulsará en el botón de “Editar”

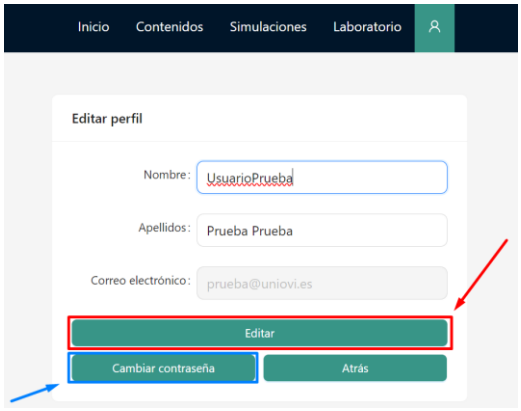

Figura 11.9.1.- Editar perfil de usuario

En el caso de que el usuario quiera cambiar su contraseña este deberá de pulsar en el botón “Cambiar contraseña” marcado en azul en la figura 11.9.1.

El usuario rellenará todos los campos, indicando su contraseña actual, su nueva contraseña y la repetición de esta, estas dos últimas deberán de ser iguales. Seguidamente pulsará en el botón de “Editar”.

Figura 11.9.2.- Editar contraseña de usuario

11.1.10.- Cierre de sesión

Para cerrar la sesión el usuario deberá pulsar en el botón de “Cerrar sesión” disponible en el icono del menú de navegación superior.

Figura 11.10.- Cierre de sesión

11.2.- Ruta al código fuente

A continuación se deja el enlace al código fuente de la aplicación, alojado en el OneDrive.

Código fuente TFG Sofía Barril