**Título**

**Smart Room Project (SRP)**

**Integrantes:**

Qian, Zhijie

Ibañez, Miguel

Palacio, Rodolfo

Tabla de contenido

[Introducción 4](#_Toc180948877)

[Planificación y alcance 5](#_Toc180948878)

[Objetivo 5](#_Toc180948879)

[Objetivos específicos: 5](#_Toc180948880)

[Fases del proyecto 5](#_Toc180948881)

[Requisitos del proyecto 5](#_Toc180948882)

[Requisitos Funcionales 5](#_Toc180948883)

[Requisitos no Funcionales 6](#_Toc180948884)

[Diseño del sistema 6](#_Toc180948885)

[Arquitectura general 6](#_Toc180948886)

[Base de datos 6](#_Toc180948887)

[Swagger 6](#_Toc180948888)

[Docker 7](#_Toc180948889)

[Spring Boot 7](#_Toc180948890)

[Angular 7](#_Toc180948891)

[Diagrama UML de Clases 8](#_Toc180948892)

[Modelo de bases de datos 8](#_Toc180948893)

[Prototipo de la interfaz 9](#_Toc180948894)

[Estructura del prototipo 9](#_Toc180948895)

[Diseño del prototipo 10](#_Toc180948896)

[Implementación del back-end y front-end 11](#_Toc180948897)

[Back-end: Spring Boot 11](#_Toc180948898)

[Estructura de la arquitectura 11](#_Toc180948899)

[Estructura del código 11](#_Toc180948900)

[Front-end: Angular 12](#_Toc180948901)

[Estructura del código 12](#_Toc180948902)

[Servicios de comunicación con el Back-end 12](#_Toc180948903)

[Interfaz y visualización de datos 12](#_Toc180948904)

[Presentación de la aplicación Web 13](#_Toc180948905)

[Conclusiones 18](#_Toc180948906)

[Referencias 19](#_Toc180948907)

Índice de tablas

[Tabla 2: Requisitos funcionales del sistema 6](#_Toc180947739)

[Tabla 3: Requisitos no funcionales del sistema 6](#_Toc180947740)

Índice de figuras

[Figura 1: Diagrama UML de Clases 8](#_Toc180947722)

[Figura 2: Prototipo de pantalla principal 10](#_Toc180947723)

[Figura 3: Prototipo representación visual de una habitación 10](#_Toc180947724)

[Figura 4: Pantalla principal 13](#_Toc180947725)

[Figura 5: Crear una habitación 14](#_Toc180947726)

[Figura 6: Elementos de una habitación 15](#_Toc180947727)

[Figura 7: Gráfico de número de personas en la habitación 16](#_Toc180947728)

[Figura 8: Campos disponibles para editar una habitación 17](#_Toc180947729)

# Introducción

El SRP (Smart Room Project) es una solución de software diseñado para administrar y supervisar salas inteligentes, permitiendo un control detallado de elementos como puertas, ventanas, luces, ventiladores y condiciones de temperatura, niveles de CO2 y número de personas por salas.

Para el diseño del prototipo de este sistema se utilizará figma para la creación de las pantallas y componentes de las salas y los botones para las acciones correspondientes. En este proyecto se combina tecnologías modernas como angular y Spring Boot complementadas con la base de datos de MySQL.

Además, el uso de Docker facilita el despliegue y asegura la portabilidad del sistema en distintos entornos. La interfaz web desarrollada permite a los usuarios visualizar y gestionar cada sala de forma intuitiva, utilizando gráficos y controles que proporcionan información en tiempo real.

# Planificación y alcance

## Objetivo

Diseñar el prototipo de un sistema de control para la gestión de salas inteligentes en tiempo real, control de dispositivos y funcionalidades de seguridad empleando las tecnologías angular y MySQL.

### Objetivos específicos:

* Gestionar salas permitiendo la creación, actualización y eliminación.
* Controlar remotamente los dispositivos como luces, puertas, ventiladores y ventanas.
* Garantizar la seguridad configurando la automatización de las luces y alarmas.
* Visualizar los datos en tiempo real como la temperatura, el nivel de dióxido de carbono e indicar la disponibilidad de las habitaciones.

# Fases del proyecto

## Requisitos del proyecto

### Requisitos Funcionales

|  |  |  |
| --- | --- | --- |
| **Nº** | **Descripción del requisito** | **Obligatorio/Opcional** |
| F1 | El sistema debe permitir importar datos desde un fichero | Obligatorio |
| F2 | El sistema debe permitir exportar estructura de saldas a un archivo csv |  |
| F3 | El sistema debe permitir la gestión de salas | Obligatorio |
| F4 | El sistema debe permitir controlar los dispositivos | Obligatorio |
| F5 | El sistema debe mostrar datos de las temperaturas | Obligatorio |
| F6 | El sistema debe indicar y mostrar el nivel de dióxido de carbono | Obligatorio |
| F7 | El sistema debe mostrar la disponibilidad en salas | Obligatorio |
| F8 | El sistema debe permitir alarmas para realizar acciones automáticas dependiendo de la temperatura | Obligatorio |
| F9 | El sistema debe permitir alarmas para realizar acciones automáticas dependiendo del nivel del dióxido de carbono | Obligatorio |
| F10 | El sistema debe permitir cambiar el color a verde-amarillo-rojo dependiendo del nivel de CO2 | Obligatorio |
| F11 | El sistema debe permitir acciones mediante comando de voz. | Opcional |

Tabla 2: Requisitos funcionales del sistema

### Requisitos no Funcionales

|  |  |  |
| --- | --- | --- |
| **Nº** | **Descripción del requisito** | **Obligatorio/Opcional** |
| RF1 | El sistema debe tener un diseño responsive | Obligatorio |
| RF2 | El sistema debe ser intuitivo y fácil de usar | Obligatorio |
| RF3 | El sistema debe ser escalable para soportar más salas | Obligatorio |
| RF4 | El sistema debe ser escalable para soportar más dispositivos. | Obligatorio |
| RF5 | El sistema debe reconocer comando de voz | Opcional (RF11) |

Tabla 3: Requisitos no funcionales del sistema

# Diseño del sistema

## Arquitectura general

El SRP está diseñado utilizando una arquitectura cliente-servidor, tanto el back-end y front-end se comunican a través de una API RESful.

Componentes principales en esta arquitectura del proyecto

### Base de datos

Se utiliza para el desarrollo del laboratorio MySQL como base de datos relacional para almacenar toda la información de las salas, dispositivos y datos de monitoreo. Las entidades del backend como “Room”, “Door”, “Fan” y “Light”,están mapeadas en la base de datos. Esto permite el almacenamiento y recuperación de la información de cada sala y dispositivo.

### Swagger

En este proyecto, swagger se utiliza para generar una interfaz web donde se pueden ver todos los endpoints del back-end, junto con sus parámetros, métodos HTTP y respuestas esperadas.

### Docker

Se utiliza Docker para facilitar el despliegue y la portabilidad del SRP.

### Spring Boot

El backend está implementado en spring boot, donde se gestiona las operaciones sobre las salas y dispositivos. Y maneja la integración con la base de datos y facilita los endpoints necesarios para que el front-end pueda realizar solicitudes y recibir actualizaciones.

### Angular

Se ha desarrollado la parte front-end en Angular, este framework basado en TypeScript. Con la interfaz permite al usuario interactuar con las funcionalidades del sistema, como el control de dispositivos y el monitoreo de temperatura, CO2 y ocupación en las salas.

### Diagrama UML de Clases

Diagrama

Descripción generada automáticamente

Figura 1: Diagrama UML de Clases

En la Figura 1: Diagrama UML de Clases se ha representado la estructura de datos y las relaciones entre las principales entidades del sistema. Se muestra la organización en el back-end, incluyendo las clases principales, sus atributos y métodos, así como las asociaciones entre ellas.

### Modelo de bases de datos

La base de datos del sistema está desarrollada en MySQL. Se ha almacena la información detallada sobre las saldas, dispositivos y datos de monitoreo. Este modelo de base de datos se diseñó en base al Diagrama UML de Clases del sistema.

## Prototipo de la interfaz

Para el diseño del prototipo se utilizó la herramienta figma enfocado en la parte visual e interactivo de la interfaz de usuario, donde se define la disposición de todos los elementos, menús, navegación, botones, representación gráfica y funcionalidades.

### Estructura del prototipo

* Pantalla de inicio (panel informativo)
  + Número de salas
  + Estados de los dispositivos
  + Gráficos de temperatura
  + Gráficos de nivel de CO2
* Gestión de salas
  + Formulario con datos básicos como: nombre, tamaño, número de puertas, número de ventanas y sensores.
* Panel de control
  + Botón de encender/apagar luces
  + Botón de encender/apagar dispositivos
  + Visualizar el estado actual de luces
  + Visualizar el estado actual de puertas
* Proyección de datos en tiempo real
  + Gráficos de temperatura
  + Gráficos de nivel de CO2
  + Gráfico de ocupación en cada sala
* Seguridad
  + Si la temperatura es superior a 70º activar alarma
  + Si la temperatura es superior a 70º desbloquear todas las puertas
  + Si los valores de CO2 son > 1000 partes por millón (ppm), activar ventiladores y abrir las ventanas.

### Diseño del prototipo

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Figura 2: Prototipo de pantalla principal

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 3: Prototipo representación visual de una habitación

## Implementación del back-end y front-end

La conexión entre front-end y back-end se gestiona a través del API, y para poder trabajar de manera paralela, se ha introducido otras tecnologías y conceptos que aumentará nuestra productividad. El primer concepto que se introduce se basa en una respuesta común que devuelve cada API, esta respuesta está compuesta por tres componentes, el código del estado que será SUCCESS o FAILURE, el mensaje de aviso o de error y el propio dato. El segundo concepto sería Global exception handler, donde cuando existe un fallo en el sistema, automáticamente devuelve una respuesta con código FAILURE y un mensaje de error. Finalmente, una nueva tecnología que se ha utilizado es el Swagger, una herramienta muy potente para las especificaciones de cada API, de esta manera, el backend publicará las APIs poco a poco y el frondend podrá consumirar esas APIs según los parámetros de entrada y salida que se ha definido.

## Back-end: Spring Boot

El back-end del SRP está desarrollado en Spring Boot. Este back-end proporciona la lógica de negocio y expone una API REST que permite la comunicación con el front-end.

### Estructura de la arquitectura

* Se utiliza Maven para la gestión de dependencias, permitiendo una administración sencilla de las bibliotecas y configuraciones.
* Se configuró para ser desplegado en un contenedor Docker para orquestar el back junto con la base de datos MySQL, que facilita un despliegue consistente y portátil.

### Estructura del código

* Los controladores gestionan los endpoints relacionados con la gestión de salas.
* Los servicios están organizados por dispositivos y funcionalidades, con métodos para su control.
* Cada entidad incluye atributos y métodos de acceso definidos en el diagrama de clases.
* Los repositorios manejan las operaciones CRUD y consultas personalizadas para cada entidad.

## Front-end: Angular

El front-end de este SRP está desarrollado en Angular. El front-end se comunica con el back-end a través de solicitudes HTTP para visualizar y controlar en tiempo real el estado de cada sala y sus dispositivos.

### Estructura del código

* En los Componentes se representan las vistas y secciones de la aplicación, donde se muestra y controla el estado de las salas y dispositivos.
* En los Modelos se define las interfaces y clases necesarias para representar los datos que se manejan en la aplicación.
* En los servicios están los servicios que facilitan la comunicación.

### Servicios de comunicación con el Back-end

* RoomService que gestiona las solicitudes relacionadas con las salas como la creación, actualización y obtención de datos.
* DeviceService permite controlar los dispositivos.

### Interfaz y visualización de datos

* Se tiene los componentes de visualización para los niveles de CO2, temperatura y si están o no ocupadas y contar las cantidades.
* Actualización de los datos con los datos más recientes, reflejando cambios en el estado de los dispositivos.

# Presentación de la aplicación Web

La aplicación web del SRP está diseñada para gestionar las salas. Esta plataforma permite a los usuarios visualizar y controlar diferentes parámetros y dispositivos.

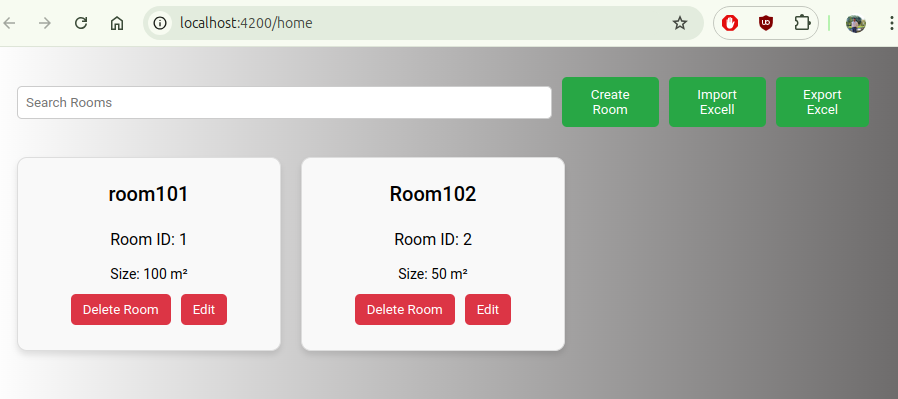


Figura 4: Pantalla principal

La interfaz de usuario del SRP permite la gestión de salas, visualizaciones, crear, editar y eliminar salas, así como importar y exportar datos en formatos Excel.

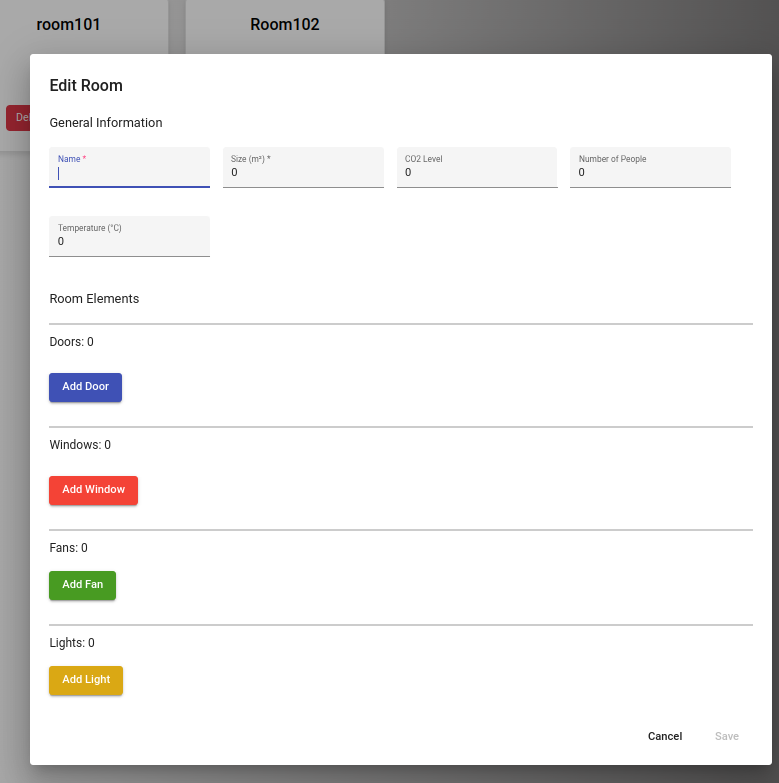


Figura 5: Crear una habitación

Esta interfaz de edición de salas permite a los usuarios gestionar los detalles específicos de cada sala y sus dispositivos asociados. Esta vista es accesible desde la pantalla principal de gestión de salas.

En esta interfaz se pueden ajustar los parámetros de temperatura, niveles de CO2 , número de dispositivos en cada sala.

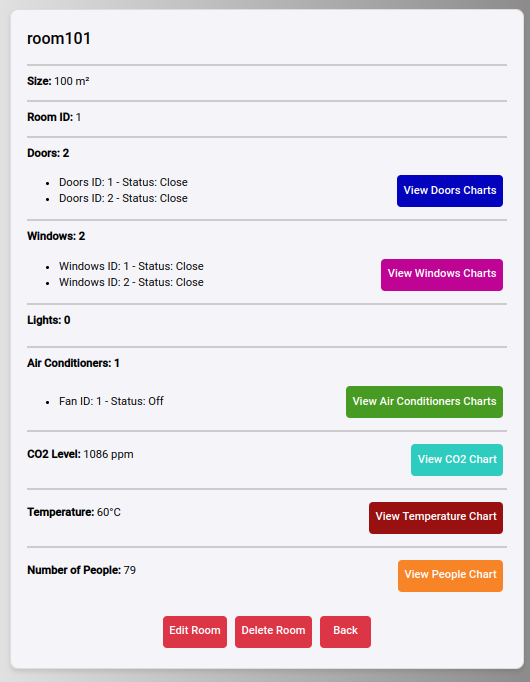


Figura 6: Elementos de una habitación

En esta pantalla se puede ver los detalles de una sala, en ella se puede consultar y monitorear los niveles de CO2, temperatura, número de personas y dispositivos asociados.

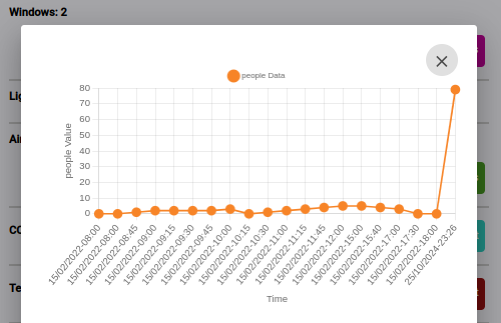


Figura 7: Gráfico de número de personas en la habitación

Para la visualización de los datos se podrá observar la evolución de los valores en la gráfica. En este caso específico la Figura 7: Gráfico de número de personas en la habitación permite observar el número de persona presentes en una sala específica.

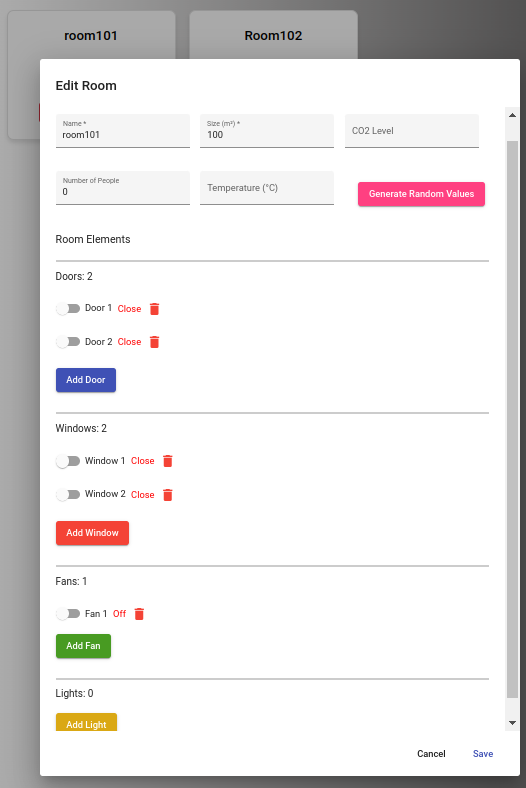


Figura 8: Campos disponibles para editar una habitación

Esta pantalla de edición de sala permite personalizar y gestionar en detalle en los elementos y parámetros de cada sala. Esto es útil para configurar el entorno y funcionalidades del sistema.

# Conclusiones

Se ha construido un sistema empleando las tecnologías **Angular** y **Spring Boot**, junto con la base de datos en **MySQL** y el uso de **Docker** para el despliegue.

Las funcionalidades de monitoreo, generación de gráficos y personalización de salas, proporciona a los usuarios una herramienta para la administración eficiente de espacios, mejorando el control sobre el uso de energía y optimizando la ocupación.

En conclusión, este proyecto representa la integración de tecnologías inteligentes en la gestión de salas. En el futuro, se podrían incorporar nuevas funcionalidades, como la integración con sensores adicionales con el fin de optimizar aún más el control y la automatización del entorno.

# Referencias