Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA (SG)

Grado en Ingeniería Informática de Servicios y Aplicaciones

eMuseum: Sistema de gestión del inventario y documental del museo informático de la Escuela de Ingeniería Informática de Segovia

Alumno: Mario Borrego Rodríguez

Tutor: Fernando Díaz Gómez

A pesar de todo una sonrisa me acompaña

AGRADECIMIENTOS

Durante los años que han transcurrido para cursar este grado, he recibido el apoyo de gente querida a la que tengo especial interés en mencionar. Quiero dar un agradecimiento especial a mi familia; mi padre Manuel, mi madre Susana y mi hermana Silvia, a mis tías Mamen y Mónica porque a pesar de la situación vivida en estos años tan difíciles, me han acompañado en todo momento, demostrándome su cariño y apoyo.

Cómo no dar las gracias, por tanta paciencia, a los amigos que he hecho durante el transcurso de mis estudios universitarios y mis dos años de carrera profesional. De entre ellos quiero hacer mención especial a tres personas que han estado en los momentos mas oscuros: María, Anabel y Sergio.

Por último, necesito mostrar mi agradecimiento a todos mis profesores por el tiempo y disposición que me han dedicado, los conocimientos transmitidos y su empeño en formarnos como Ingenieros Informáticos. Mencionar especialmente a mi tutor de TFG, Fernando Díaz Gómez, por su guía y sabios consejos.

Muchas gracias.

Mi yo futuro vino a verme y me dio un consejo

Dijo: "trabaja duro, luego déjalos perplejos"

RESUMEN

El objetivo del proyecto objeto del trabajo fin de carrera consiste en el diseño e implementación de una solución software que permita a la Escuela de Ingeniería Informática de Segovia tener un control sobre todos los recursos que posee, así como la localización de estos. Para ello, se ha implementado un sistema de clasificación basado en cases, un sistema de localización basado en laboratorios y armarios y un sistema de filtrado basado en clases.

De esta manera se pretende que la Escuela de Ingeniería Informática de Segovia tenga una correcta trazabilidad de los recursos que posee y ninguno caiga en el olvido.

Palabras Clave:

Desarrollo Web, Angular, Firebase, SPA

INDICE DE CONTENIDO

1 INTRODUCCIÓN 12

1.1 INTRODUCCIÓN 12
1.2 MOTIVACIÓN 12
1.3 OBJETIVOS Y ALCANCE 12
1.3.1 Objetivos 12
1.3.2 Alcance 13
1.4 ENTORNO TECNOLÓGICO 13
1.5 ORGANIZACIÓN DEL DOCUMENTO 14
1.6 CONTENIDO DEL CD-ROM 15

2 CONTEXTO DEL PROYECTO 16

2.1 ESTADO DEL ARTE 16
2.1.1 Museo Virtual de la Informática 16
2.1.2 Museo 3D Informática 17
2.2 EVALUCIÓN DEL ESTADO DEL ARTE 18
2.3 METODOLOGÍA 18

3 PLANIFICACIÓN, ESTIMACIÓN Y PRESUPUESTO 20

3.1 PLANIFICACIÓN 20
3.2 ESTIMACIÓN 21
3.2.1 Listado de entradas 22
3.2.2 Listado de Salidas 22
3.2.3 Listado de Consultas 23
3.2.4 Listado de Ficheros Lógicos internos 24
3.2.5 Listado de Ficheros Lógicos externos 24
3.2.6 Puntos de Función sin AJUSTAR 24
3.3 ESTIMACIÓN 26
3.4 PRESUPUESTOS 27
3.4.1 Recursos humanos 27
3.4.2 Hardware 27
3.4.3 Software 28
3.4.4 Imputacion al proyecto 29

4 ANÁLISIS 30

4.1 ACTORES 30
4.2 REGLAS DE NEGOCIO 30
4.3 OBJETIVOS DEL SISTEMA 30
4.4 REQUISITOS DE USUARIO 31
4.5 DIAGRAMAS DE CASO DE USO 32
4.6 REQUISITOS FUNCIONALES 42
4.6.1 Gestión de usuarios 42
4.6.2 Administrador 42
4.7 REQUISITOS NO FUNCIONALES 43
4.7.1 Disponibilidad 43
4.7.2 Accesibilidad 43
4.7.3 Escalabilidad 43
4.7.4 Usabilidad 43

4.7.5 Seguridad *43*
4.7.6 Restricciones *44*
4.7.7 Atributos de calidad *44*
4.8 REQUISITOS DE INFORMACIÓN 44
4.8 MODELO ENTIDAD – RELACIÓN 44
5 DISEÑO 46
5.1 ARQUITECTURA LÓGICA 46
5.2 ARQUITECTURA FÍSICA 47
5.3 DISEÑO BASE DE DATOS 48
5.4 DICCIONARIO DE DATOS 49
5.4.1 Clase *49*
5.4.2 Localizacion *49*
5.4.3 Recurso *49*
5.4.4 Usuario *50*
5.5 ACUERDOS DE DISEÑO DE LA INTERFAZ 50
5.6 DIAGRAMA DE CLASES 53
5.6.1 Componentes *55*
5.6.2 Guardianes *56*
5.6.3 Servicios *56*
5.7 DIAGRAMAS DE SECUENCIA 57
5.7.1 Secuencia login *57*
5.7.2 Secuencia listado de Recursos *58*
5.7.3 Secuencia Agregar Recursos *58*
5.7.4 Secuencia Editar Recurso *59*
5.7.5 Secuencia borrar Recurso *59*
6 IMPLEMENTACIÓN 60
6.1 SOFTWARE UTILIZADO 60
6.1.1 Primera aproximacion *60*
6.1.2 Segunda iteración *60*
6.2 ESQUEMA DEL PROYECTO 61
6.2.1 Esquema del proyecto JAVA *61*
6.2.2 Esquema del proyecto Angular *63*
6.2 DETALLES SIGNIFICATIVOS DE IMPLEMENTACIÓN 65
6.2.3 Detalles significativos en Java *65*
6.2.3.1 Detalles significativos de la base de datos 65
6.2.3.2 Detalles significativos en Java 66
6.2.3.3 Client 69
6.2.4 Detalles significativos en Angular *73*
6.2.4.1 Google Firebase 73
6.2.4.1.1 RealTime Database 73
6.2.4.1.2 Firebase Authentication 74
6.2.4.1.3 Cloud Storage 76
6.2.4.1.4 Firebase Hosting 76
6.3.2.1 Plataforma Angular 78
6.3.2.2 Aplicaciones Móviles 82
6.3 PROBLEMAS 83
6.3.1 Problemas encontrados en Java *83*
6.3.1.1 Netbeans 11.X 83
6.3.1.2 Netbeans 8.2 84
6.3.1.3 glassfish5.0 – Mail 85
6.3.1.4 Inclusión de Código JavaScript 85
6.3.1.5 Inclusión de la librería Bootstrap 85

6.3.2 ProblEmas encontrados en Angular *85*

7 PRUEBAS **86**

7.1 PRUEBAS DE CAJA BLANCA 86
7.2 PRUEBAS DE CAJA NEGRA 86

8 CONCLUSIONES **88**

9 REFERENCIAS. **89**

9.1 ENLACES A WEBS CONULTADAS 89
9.1.1 Mail *89*
9.2 DEFINICIONES 89
9.3 MIGRACIÓN 90
9.4 GUIA DE REFERENCIA 90

INDICIE DE ILUSTRACIONES

Figura 1: Museo Virtual de la informática 16
Figura 2: Museo 3D de la informática 17
Figura 3: Desarrollo de software basado en la metodología en cascada 19
Figura 4: Calendario laboral Madrid 20
Figura 5: Tabla de Complejidades 22
Figura 6: Planificación del proyecto 26
Figura 7: Planificación temporal del proyecto 26
Figura 8: Esquema de usuarios 32
Figura 9: Caso de uso de usuario general 33
Figura 10: Caso de uso de usuario administrador 35
Figura 11: Diagrama entidad relación. 45
Figura 12: Arquitectura lógica 46
Figura 13: Arquitectura Física 48
Figura 14: Diseño de base de datos 48
Figura 15: Diagrama de clases 54
Figura 15: Diagrama de clases de componentes 55
Figura 15: Diagrama de clases de Guardianes 56
Figura 15: Diagrama de clases de Servicios 56
Figura 15: Diagrama de secuencia de login 57
Figura 15: Diagrama de secuencia de listado de recursos 58
Figura 15: Diagrama de secuencia de agregar recursos 58
Figura 15: Diagrama de secuencia de editar recursos 59
Figura 15: Diagrama de secuencia de editar recursos 59
Figura 16: Estructura cliente 62
Figura 17: Estructura cliente 62
Figura 18: Estructura Servidor 63
Figura 19: Código de creación de la base de datos 66
Figura 20: Código de clases entidad 67

Figura 21: Código de clases entidad 68
Figura 22: Código de servicios REST 68
Figura 23: Código de servicios REST 69
Figura 24: Código de Reader 70
Figura 25: Código de Writer 70
Figura 26: Código de BackingBean 71
Figura 27: Código de BackingBean 72
Figura 28: Código de ClientBean 72
Figura 29: Pagina de control Firebase 73
Figura 30: Pagina de control Firebase- Firestore Database 74
Figura 31: Pagina de control Firebase – Authentication 75
Figura 32: Pagina de control Firebase – Storage 76
Figura 33: Pagina de control Firebase – Hosting 77
Figura 34: Configuración del app-routing.module 78
Figura 35: Configuración del entorno de conexión a Firebase. 78
Figura 36: Configuración de funcionamiento de la aplicación. 79
Figura 37: Configuración de funcionamiento de la aplicación. 79
Figura 38: Configuración de Angular.json. 80
Figura 39: Modelo de datos. 80
Figura 40: Servicio 81
Figura 41: Componente - TS 81
Figura 42: Componente – HMTL 82
Figura 43: Guardián 82
Figura 44: Instalación de cordova 83
Figura 45: Instalación de cordova 83
Figura 15: Error Netbeans 8.2 84
Figura 15: Solución Netbeans 8.2 84
Figura 15: Glassfish 5.0 85

INDICE DE TABLAS

Tabla 1: Listado de entradas 22
Tabla 2: Listado de Salidas 23
Tabla 3: Tabla de consultas 23
Tabla 4: Tabla de ficheros lógicos internos 24
Tabla 5: Tabla de complejidades 25
Tabla 6: Tabla de calculo los FC 26
Tabla 7: Recursos humanos 27
Tabla 8: Imputación de Hardware 28
Tabla 9: Imputación de Software 29
Tabla 10: Coste de Proyecto 29
Tabla 11: Actores del sistema 30
Tabla 12: Reglas de negocio 30
Tabla 13: Objetivos del sistema – gestión de recursos 31
Tabla 14: Objetivos del sistema – préstamo de recursos 31
Tabla 15: Requisitos de usuario de usuario no registrado 31
Tabla 16: Requisitos de usuario de usuario administrador 31
Tabla 17: Caso de uso de CU-01 33
Tabla 18: Caso de uso de CU-02 34
Tabla 19: Caso de uso de CU-03 34
Tabla 20: CU-04: Iniciar Sesión. 35
Tabla 21: CU-05: Administrar clases. 36
Tabla 22: CU-06: Crear clase. 36
Tabla 23: CU-07: Modificar clase. 37
Tabla 24: CU-08: Eliminar clase. 37
Tabla 25: CU-09: Administrar Recursos. 38
Tabla 26: CU-10: Crear recurso. 38
Tabla 27: CU-11: Modificar recurso. 39
Tabla 28: CU-12: Eliminar Recurso. 39

Tabla 29: CU-13: Administrar Recursos. 40
Tabla 30: CU-14: Crear localizaciones. 40
Tabla 31: CU-15: Modificar localización. 41
Tabla 32: CU-16: Eliminar Recurso. 41
Tabla 33: Diccionario de Clase. 49
Tabla 34: Diccionario de localización. 49
Tabla 35: Diccionario de localización. 49
Tabla 36: Diccionario de usuario. 50
Tabla 37: Diseño de interfaz. 50
Tabla 38: Diseño de interfaz. 51
Tabla 39: Diseño de interfaz. 52
Tabla 40: Diseño de interfaz. 52
Tabla 41: PCN-01 86
Tabla 42: PCN-02 87
Tabla 43: PCN-03 87
Tabla 44: PCN-04 87
Tabla 45: PCN-05 87
Tabla 46: PCN-06 87

1 INTRODUCCIÓN

1.1 INTRODUCCIÓN

La Escuela de Informática de Segovia tiene un largo recorrido en el mundo de la enseñanza. La escuela empezó en el año 1992, como centro asociado al Colegio Universitario Domingo de Soto de la Universidad Complutense de Madrid, la cual tenía su sede en el edificio de "Santa Eulalia". Luego en el 2001 se integró en la Universidad de Valladolid manteniendo su ubicación.

Durante estos años, la escuela ha ido realizando adquisiciones de recursos de diferente índole, escáneres, ordenadores, placas de programación, material de laboratorio para física, etc. Que según han ido pasando los años, estos han quedado anticuados o se han ido rompiendo, quedando los pocos funcionales apartados a un armario en algún lugar.

Además, el personal docente que compone la escuela también ha ido cambiando con el paso de los años, ya sea por traslados, jubilaciones u otras causas, el personal docente y administrativo también ha ido cambiando. Esto ha supuesto que la información del material que funcionaba y se ha guardado no había sido registrado en ningún sitio quedándose esta en el olvido.

Por otra parte, las nuevas tecnologías o tecnologías de la información han ido evolucionando. Cuando la escuela empezó, las aplicaciones web estaban naciendo en aquel momento. Esto suponía que el desarrollo y despliegue de estas era caro y complicado ya que no existían muchas herramientas.

1.2 MOTIVACIÓN

El cambio de sede de la Escuela de Ingeniería Informática de Segovia ha supuesto una mudanza de un edificio situado en Santa Eulalia a la segunda fase del campus María Zambrano. Durante este proceso se descubrió una serie de recursos tanto documentales como de aparatos electrónicos que nadie sabía de su existencia, o por lo menos nadie se acordó de ellos.

Por este motivo se decidió el proponer un TFG que tuviera como objetivo crear un registro en el que se recogiera toda esta serie de materiales y que sirviera para saber exactamente lo que se tenía y dónde se tenía. Esto permite a la Escuela no solo conocer todos los recursos que posee, sino que además puede prestarlos si así lo considera oportuno y poder saber qué alumno o profesor, tiene cierto recurso. Por lo que en ningún momento se perderá ningún objeto.

1.3 OBJETIVOS Y ALCANCE

1.3.1 OBJETIVOS

El principal objetivo de este proyecto es el obtener una aplicación web que permita inventariar los recursos de la Escuela, permitiendo categorizar los recursos, lo que da la posibilidad de filtrar por categorías los productos y poder encontrar con una mayor facilidad el recurso que se desea.

Además del objetivo principal que he expuesto arriba, el proyecto también persigue los siguientes objetivos secundarios:

- Obtener un inventariado de los recursos de los que dispondrá la Escuela.
- Poder obtener información de un recurso.
- Obtener un sistema que clasifique los recursos

Por otra parte, con la finalidad que el usuario tenga la mejor experiencia de uso con la aplicación, se han fijado una serie de objetivos de diseño que se deben cumplir en el desarrollo de la aplicación:

- Utilización de colores con altos contrastes: Usar colores que tengan un alto contraste entre sí, nos permite facilitar la lectura por parte de los usuarios
- Concisión en la transmisión de la información: Todos los diálogos que se incluyan en la aplicación deben de ser lo más concisos y exactos posibles con el fin de transmitir de la forma más breve posible la información al usuario.
- Los mensajes de error en la medida que sea posible deben de aparecer en los recuadros donde no se han incluido la información o un mensaje que el usuario pueda identificar de manera inequívoca el error producido

1.3.2 ALCANCE

La herramienta objeto de este TFG está enfocada tanto al personal docente de la Escuela como a los propios alumnos. La aplicación es accesible para cualquier alumno de la Escuela y tendrá las capacidades completas que se le proporciona.

1.4 ENTORNO TECNOLÓGICO

eMusseum se ha concebido y desarrollado como una aplicación en la nube, por lo que se ha desarrollado utilizando el framework de Google Angular, en este caso se ha utilizado la versión 11. Además, se ha utilizado la plataforma Firebase como hosting y base de datos. Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles, la cual posee una base de datos NoSQL y un hosting gratuito para el despliegue de aplicaciones

Los motivos por los que se han utilizado este entorno tecnológico son los siguientes:

- Arquitectura basada en componentes.
- Posibilidad de carga dinámica de componentes y la eliminación de rutas.
- Utilización de un *hosting* sin necesidad de configuración en entorno de producción, base de datos y seguridad.
- Posibilidad de utilización de varios proveedores de entidad, sin necesidad de realizar una programación y configuración complejas.

- Desarrollo de aplicaciones web sin necesidad de desarrollar un *BackEnd*.

- Compilación de la aplicación para los sistemas operativos móviles Android e iOS.

Además del entorno tecnológico que se ha utilizado para el desarrollo del presente proyecto, se ha utilizado un entorno específico para el desarrollo de la aplicación. En concreto que se ha utilizado las siguientes herramientas:

- Webstorm e Intlij: Se han utilizado los IDEs de JetBrains debido a la gran cantidad de *plugins* que tienen y a las facilidades que ofrecen.

- macOS: Se ha utilizado el sistema operativo de Mac debido a la facilidad de uso para la instalación de las herramientas necesarias. En concreto se han utilizado las versiones de macOS 10.14: Mojave, macOS 10.15 Catalina y macOS 11 Big Sur.

- Node.js y npm: Para la utilización del framework de Angular ha sido necesaria la instalación de estas herramientas.

- MS Office: Se ha utilizado la suite ofimática de Microsoft para la creación de la documentación y el almacenamiento de datos en su nube Office 365.

- GitHub: Se ha utilizado el presente repositorio de datos para el control de versiones del proyecto y como *backup* del mismo.

1.5 ORGANIZACIÓN DEL DOCUMENTO

En este punto se expone la organización del documento, dividiéndolo en secciones. Para una mayor compresión de los temas a tratar en cada sección, se acompañará de una pequeña explicación sobre los mismos:

- **Sección 1– Introducción:** La sección de introducción se ha concebido como un pequeño resumen, el cual tiene como principal objetivo proporcionar al lector una pequeña visión sobre la motivación, el alcance del trabajo, el entorno tecnológico sobre el cual se sustenta la aplicación, así como una breve descripción del contenido que se entrega al presentar el TFG.

- **Sección 2 – Contexto del proyecto:** La sección de contexto del proyecto se ha concebido como un apartado en el que se explica en que condiciones se ha decidido ejecutar el proyecto, así como la metodología utilizada para la consecución de este.

- **Sección 3 – Planificación, Estimación y Presupuesto:** A lo largo de este punto, se expondrá los pasos seguidos para obtener una estimación y planificación de las tareas necesarias para el desarrollo del proyecto. Además, se calcularán los gastos asociados al mismo, es decir, el importe de llevar a cabo el proyecto.

- **Sección 4 – Análisis:** Durante este punto, se expondrá la información más vital para el desarrollo del sistema, ya que se analiza el funcionamiento del sistema. El mencionado análisis se realiza desde tres puntos de vista, desde el del usuario, desde el sistema y desde el de los datos.

- **Sección 5 – Diseño:** La sección de diseño, se expondrá la información necesaria para explicar cómo se desarrollará el sistema y cómo será el comportamiento de este. Para ello, se irá explicando la arquitectura lógica y física que tendrá el sistema, y los diferentes diagramas a partir de los cuales se implementará posteriormente el sistema.

- **Sección 6 – Implementación:** A lo largo del presente capitulo se describirá la estructura del proyecto que se ha realizado para la implementación de este, así como las herramientas que se han utilizado para cada apartado del desarrollo.

- **Sección 7 – Pruebas:** Se expondrán las pruebas de caja blanca y caja negra para la verificación de las funcionalidades desarrolladas.

- **Sección 8 – Conclusiones:** Se expondrá las conclusiones extraídas de la realización del presente proyecto.

- **Sección 9 – Referencias:** Se visualizará un listado de sitios, libros, documentos consultados a la hora desarrollar el presente proyecto.

1.6 CONTENIDO DEL CD-ROM

Acompañado de este proyecto, se entrega al tutor del presente TFG una cuenta de Google con la aplicación configurada y desplegada en Google Cloud

2 CONTEXTO DEL PROYECTO

A lo largo de esta sección se irá proporcionando la información necesaria al lector con el fin de que este pueda obtener una visión general de cómo se ha diseñado la aplicación.

2.1 ESTADO DEL ARTE

A la hora de realizar este proyecto se han tenido en cuenta otros proyectos que existen en la actualidad. Por ello se han revisado proyectos de museos, tanto de especialidad informática, como los de especialidad general, ya que lo que nos interesa es cómo han tratado el tema y la forma de organizar los recursos.

2.1.1 MUSEO VIRTUAL DE LA INFORMÁTICA

Este museo pertenece a la Escuela Superior de Informática de la Universidad de Castilla la Mancha. Este museo se caracteriza por ser no solo un museo virtual sino también físico. Este se encuentra instalado en el vestíbulo de su edificio principal (Edificio Fermín Caballero). Dicho museo permite seguir la historia de la informática al visitante, a través de la evolución de los numerosos equipos que dispone.

Figura 1: Museo Virtual de la informática

Una vez analizada la web se pueden extraer las siguientes características positivas y negativas:

- Positivas:
  - Es una web muy compacta y con la información muy bien expuesta.
  - Posee modelos 3d de sus elementos que se pueden descargar.
- Negativa:
  - Es una página antigua que no se ha retocado en bastante tiempo.
  - No permite realizar filtros.
  - Los recursos de los que dispone son muy escasos.

2.1.2 MUSEO 3D INFORMÁTICA

El museo de la Informática de la Universidad de Valladolid se encuentra en la Escuela Técnica Superior de Ingeniería Informática, conteniendo en sus aulas una excelente colección de material informático. Este museo tal y como dice la web surge con la idea de realizar una página Web en la que se pueda ver los objetos destacados en el mundo de la informática de una forma diferente, en 3 dimensiones, en la que se pueden ver e interactuar con los objetos, rotarles y verlos por todos sus lados. Una manera diferente de ver la informática.

Dicha web fue creada por la iniciativa, colaboración y participación de:

- Alberto Sanz Prieto (Alumno de I.T. de Diseño Industrial).
  - E-mail: albertoataquines@hotmail.com
- David Escudero Mancebo (Profesor de la EUP)
- Javier Bastida Ibáñez (Profesor de la Escuela Técnica Superior de Ingeniería Informática)

Figura 2: Museo 3D de la informática

Encontrada la web se procedió al análisis de esta. En un primer momento en la barra lateral derecha te dicen que necesitas instalar el complemento de Adobe Shockwave Player para poder visualizar los equipos, por ello se procedió a instalarlo. El problema es que no se puede instalar ya que desde 2017 está sin comercializarse para macOS, y para

Windows desde el 9 de abril del 2019. Por lo que no se ha podido ver la información publicada. Aun así, se ha procedido a hacer un análisis de la web con puntos positivos y negativos y son los siguientes:

- Positivos:
  - Varias opciones.
  - Una variedad mayor que la vista en la anterior.
  - Posibilidad de ver fotos, videos y modelos 3d.
  - Incluye filtros.
- Negativos:
  - Tecnología antigua.
  - Sin actualizar.
  - Imposibilidad de usar la página debido a la discontinuidad de los módulos usados.
  - Enlaces a paginas inexistentes o que se han movido.
  - Fuerza al usuario a usar complementos con importantes fallos de seguridad.

2.2 EVALUCIÓN DEL ESTADO DEL ARTE

Tras la realización de un proceso de investigación y análisis de las soluciones existentes en el mercado, se llegó a la conclusión que ninguna solución existente cumple con todos los objetivos empresariales que se quieren cumplir con el presente proyecto.

2.3 METODOLOGÍA

Esta aplicación ha nacido de la necesidad real que tenía la Escuela de Ingeniería Informática de Segovia de tener inventariado los recursos que posee. Este TFG fue propuesto por D. Fernando Díaz con el fin de dar solución a este problema. Esta situación se podría resumir como la de un cliente que busca que una empresa o autónomo realice el proyecto. Para la mayoría de este tipo de desarrollo de aplicaciones web, van apareciendo o descubriendo nuevos requisitos según se van desarrollando, por lo que se ha decidido usar un desarrollo en cascada.

La metodología de desarrollo software en cascada se es un enfoque metodológico riguroso basado en las etapas del proceso para el desarrollo de software. Esta forma de trabajar requiere que, para poder empezar una etapa, la anterior debe estar totalmente finalizada. El motivo por el cual se ha escogido esta metodología se basa en que desde un primer momento los requisitos están muy claros, por los que no eran susceptibles a cambios.

Figura 3: Desarrollo de software basado en la metodología en cascada

Las etapas que conforma está metodología son las siguientes:

- **Análisis:** Durante esta primera etapa, el cliente especifica todos los requisitos con los que la aplicación cubrirá todas las necesidades del cliente.
- **Diseño:** A lo largo de esta etapa, se definirá la estructura y diseño que tendrá la aplicación.
- **Implementación:** En esta etapa, se lleva a cabo el desarrollo de la aplicación siguiendo las pautas y diseños realizados en la etapa anterior.
- **Verificación:** Se comprueba que las funcionalidades desarrolladas cumplen con lo esperado.
- **Pruebas:** Se verifica que las funcionalidades cumplen con lo especificado en la fase de análisis, además de que las salidas cumplieran con lo esperado.

3 PLANIFICACIÓN, ESTIMACIÓN Y PRESUPUESTO

3.1 PLANIFICACIÓN

Una vez elegido y puesto en contexto la metodología que se va a usar, se procedió con la planificación del proyecto.

La planificación inicial del proyecto se realizó fijando el día 9 de marzo de 2020 con un horario de trabajo de lunes a viernes de 18:00 a 20:00. El motivo de este horario fue por motivos laborales. Cabe destacar que el calendario de trabajo que se ha utilizado fue el laboral de la empresa de 2021.

Figura 4: Calendario laboral Madrid

En el apartado Metodología, se explicó que la metodología aplicada al presente proyecto es en cascada, por lo que la planificación del proyecto se realizó teniendo en cuenta esta. Para ello, se definieron las siguientes etapas que iba a tener el proyecto:

- **Análisis:** En esta primea etapa, se definieron los requisitos de la aplicación.
- **Diseño:** En esta segunda etapa, se realizó el diseño de la aplicación en base a los requisitos identificados y elicitados en la fase anterior.
- **Implementación:** En esta tercera etapa se realiza la implementación de las funcionalidades definidas en la fase anterior.
- **Verificación:** Se comprueba que las funcionalidades desarrolladas cumplen con lo esperado.
- **Pruebas:** Se verifica que las funcionalidades cumplen con lo especificado en la fase de análisis, además de que las salidas cumplieran con lo esperado.

3.2 ESTIMACIÓN

El proceso de estimar se considera el cálculo del esfuerzo necesario para la consecución de un fin. En informática este proceso se usa para saber el esfuerzo, recursos, etc., que son necesarios para que una empresa o persona desarrolle un producto. Citando a Paco González Cabrera en su asignatura de Gestión de Proyectos Tic del primer cuatrimestre del 4º curso del "Grado de Ingeniería Informática de Servicios y Aplicaciones de la Escuela de Informática de Segovia" dijo: "La estimación son unas cuentas rápidas que se realizan para tener una idea de lo que va a costar un proyecto".

Siguiendo un poco esta filosofía, he escogido para estimar este proceso el método de estimación mediante puntos de función, ya que está en el punto intermedio entre precisión y rapidez. Para realizar este método el primer paso es calcular los puntos de función sin ajustar. Para ello se tiene que listar los siguientes elementos:

- Entradas de usuario.
- Salidas de usuario.
- Consultas de usuario.
- Ficheros Lógicos Internos.
- Ficheros de Interfaces Externos.

Además de listar los elementos antes mencionados, hay que asignarles una complejidad por cada elemento, ya que así se calculan los PFSA (Puntos de Función Sin Ajustar). La tabla utilizada es la mostrada a continuación:

Ficheros lógicos externos e internos				Salidas y consultas				Entradas			
Registros elementales	Datos elementales			Tipos de ficheros	Datos elementales			Tipos de ficheros	Datos elementales		
	1-19	20-50	>51		1-5	6-19	>20		1-4	5-15	>16
1	Baja	Baja	Media	0-1	Baja	Baja	Media	0-1	Baja	Baja	Media
2-5	Baja	Media	Alta	2-3	Baja	Media	Alta	2-3	Baja	Media	Alta
>6	Media	Alta	Alta	>4	Media	Alta	Alta	>3	Media	Alta	Alta

Figura 5: Tabla de Complejidades

3.2.1 LISTADO DE ENTRADAS

A continuación, se detallan la entrada de datos que se realizaran a la aplicación

Entradas	Ficheros elementales	Datos elementales
Crear Recurso	Recursos	Id_recurso, nombre, descripción, localización, imagen
	Propiedades específicas	Id_propieda, descripción, valor
	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
	Manual	Id_manual, url
Crear Clase	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
Madificar Recurso	Recursos	Id_recurso, nombre, descripción, localización, imagen
	Propiedades específicas	Id_propieda, descripción, valor
	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
	Manual	Id_manual, url
Modificar Clase	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre

Tabla 1: Listado de entradas

3.2.2 LISTADO DE SALIDAS

A continuación, se detallan las salidas de datos de la aplicación

Salidas	Ficheros elementales	Datos elementales
Pantalla principal 1	Recursos	Id_recurso, nombre, descripción, localización, imagen
Pantalla principal 2	Recursos	Id_recurso, nombre, descripción, localización, imagen
Pantalla modificación recursos	Recursos	Id_recurso, nombre, descripción, localización, imagen

	Propiedades específicas	Id_propieda, descripción, valor
	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
	Manual	Id_manual, url
Pantalla Modificar Clase	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
Listado de clases	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
Mostrar Manual	Manual	Id_manual, url
Filtrado de recurso	Clase	Id_clase, nombre, descripción
Ver recurso	Recursos	Id_recurso, nombre, descripción, localización, imagen
	Propiedades específicas	Id_propieda, descripción, valor
	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
	Manual	Id_manual, url
Recurso eliminado		Mensaje
Manual Eliminado		Mensaje
Modificar Manual	Manual	Id_manual, url

Tabla 2: Listado de Salidas

3.2.3 LISTADO DE CONSULTAS

A continuación, se detallan las consultas que realizará la aplicación

Consultas	Ficheros elementales	Datos elementales
Consultar recursos	Recursos	Id_recurso, nombre, descripción, localización
Consultar clases	Clase	Id_clase, nombre, descripción
	Clase	Id_clase, nombre, descripción
Consulta de localizaciones	Localización	Id, Localización, Superlozalización

Tabla 3: Tabla de consultas

3.2.4 LISTADO DE FICHEROS LÓGICOS INTERNOS

A continuación, se detallan los ficheros lógicos internos de la aplicación.

Ficheros lógicos internos	Fichero	Datos elementales
Base de datos	Recursos	Id_recurso, nombre, descripción, localización
	Propiedades específicas	Id_propieda, descripción, valor
	Clase	Id_clase, nombre, descripción
	Propiedades	Id_propiedad, nombre
	Manual	Id_manual, url

Tabla 4: Tabla de ficheros lógicos internos

3.2.5 LISTADO DE FICHEROS LÓGICOS EXTERNOS

Este programa no realiza conexiones a ningún dato externo por lo que no se dispone de ninguna conexión externa.

3.2.6 PUNTOS DE FUNCIÓN SIN AJUSTAR

Una vez que se han identificado los elementos de entradas, salidas, consultas, FLI y FLE se procede a calcular los puntos de función sin ajustar. En este caso se ha procedido a utilizar el método de Albrecht. La tabla de complejidades ha quedado de la siguiente forma:

Tipo	Nombre	Complejidad
Entrada	Crear Recurso	Alta
	Crear Clase	Media
	Madificar Recurso	Alta
	Modificar Clase	Media
Salida	Pantalla principal 1	Baja
	Pantalla principal 2	Baja
	Pantalla modificación recursos	Alta
	Pantalla Modificar Clase	Baja
	Listado de clases	Baja
	Mostrar Manual	Baja

	Filtrado de recurso	Baja
	Ver recurso	Alta
	Recurso eliminado	Baja
	Manual Eliminado	Baja
	Modificar Manual	Baja
Consultas	Consultar recursos	Baja
	Consultar clases	Media
Ficheros lógicos internos	BBDD	Baja

Tabla 5: Tabla de complejidades

Una vez que se sabe cuál es la complejidad de cada uno se procede a calcular los puntos de función sin ajustar, este cálculo se hace de la siguiente manera

$$PFNA = Entradas * Complejidad + Salidas * Complejidad + Consultas * Complejidad + FLI * Complejidad + FLE * Complejidad$$

En este caso de estudio seria de la siguiente forma:

$$PFNA = (2 * 6 + 2 * 4)_{Entradas} + (2 * 7 + 9 * 5)_{Salidas} + (1 * 4 + 1 * 3)_{Consultas} + 1 * 7_{FLI} + 0_{FLE}$$

$$PFNA = 93$$

Los puntos de función sin ajustar son 93. Para ajustar estos puntos de función se va a usar la tabla de factores de complejidad que nos dará los FC necesarios para calcular el factor de ajuste. La fórmula usada para el factor de ajuste es la siguiente:

$$FA = \left(0{,}01 * \sum FC\right) + 0{,}65$$

Lo primero de todo es calcular los FC para ello se ha utilizado la siguiente tabla:

Factor	Valor
Respaldo y recuperación	0
Comunicaciones de datos	2
Procesamiento distribuido	0
Desempeño critico	4
Entorno operativo existente	5
Entrada de datos en línea	5
Transacción de entrada sobre pantallas múltiples	5
Actualización en línea	5
Complejo de valores de dominio de información	2
Complejo de procesamiento interno	2
Código diseñado para reutilización	5
Conversión/instalación en diseño	5
Instalaciones múltiples	0

Aplicación diseñada para cambio	5
Suma	40

Tabla 6: Tabla de calculo los FC

Con los valores de la tabla el resultado de FC ha sido de 40. Con este valor y la fórmula de antes se obtiene que el FA en este caso es de 1,05.

$$FA = (0{,}01 * 40) + 0{,}65 = 1{,}05$$

Una vez obtenido este valor, se ajustan los PFNA para determinar los PFA que serán los utilizados para realizar la estimación. Este cálculo se realiza de la siguiente manera:

$$PFA = PFNA * FA = 93 * 1{,}05 = 97{,}65$$

Una vez que se ha realizado el cálculo los Puntos de Función Ajustados (PFA), se continua con el cálculo de la estimación de la duración del proyecto. Para ello, se considera que cada punto de función equivale a 4,5h de trabajo. Por lo que el cálculo es el siguiente.

$$\text{Esfuerzo en horas} = \text{PFA} * 4 = 97{,}65 * 4{,}5 = 440$$

3.3 ESTIMACIÓN

Una vez calculadas las horas estimadas del proyecto, se procedió con la planificación de este. Para ello, se utilizó en modelo iterativo, en este caso se ha escogido 4 iteraciones.

	Emusseum	441 horas	mar 10/03/20	mar 12/01/21		Mario
	Análisis	50 horas	mar 10/03/20	lun 13/04/20		Mario
	Diseño	95 horas	mar 14/04/20	jue 18/06/20	2	Mario
	Implementación	250 horas	jue 18/06/20	jue 10/12/20	3	Mario
	Validación	0 horas	jue 10/12/20	jue 10/12/20	4	Mario
	Pruebas	46 horas	jue 10/12/20	mar 12/01/21	5	Mario
	Fin de proyecto	0 días	mar 12/01/21	mar 12/01/21	6	Mario

Figura 6: Planificación del proyecto

Una vez planificado los días dedicados a cada fase de la iteración, se procedió con la construcción del diagrama de Gantt, siendo este el siguiente:

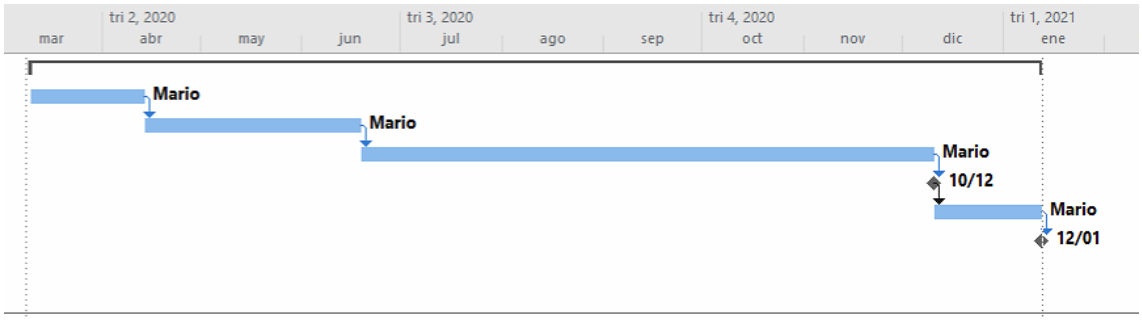


Figura 7: Planificación temporal del proyecto

3.4 PRESUPUESTOS

3.4.1 RECURSOS HUMANOS

Para el presente desarrollo, se debe tener en cuenta que el equipo de desarrolló será únicamente formado por un perfil (sin el título de grado en ingeniería informática se servicios y aplicaciones), el cual realizará los 4 diferentes roles que se necesita en el proyecto (analista, desarrollador, tester y documentador).

Los roles mencionados anteriormente, realizarán las siguientes acciones dentro del proyecto:

- Analista: Este rol es el encargado de realizar todo el proceso de elicitación, análisis y diseño de la aplicación, la cual es la pieza sobre la que se sustenta todo el proyecto.
- Desarrollador: en este caso es un desarrollador con conocimientos en Angular y Firebase, el cual se encarga del desarrollo de todas las funcionalidades
- Tester: En el presente proyecto, es el encargado de realizar las pruebas necesarias para comprobar que todos

El número de horas que se le ha asignado a cada rol son las estimadas en la planificación inicial del proyecto. El coste hora se ha obtenido en función del salario bruto del mercado en Madrid.

Recursos Humanos	Tiempo	Coste hora	Coste total
Analista / Programador	441 horas	17.95 €	7.915,95 €
Total			7.915,95 €

Tabla 7: Recursos humanos

3.4.2 HARDWARE

Para el cálculo de los costes asociados a la parte hardware que se han usado a lo largo del proyecto, se han tenido en cuenta el uso que se dará en el proyecto. A continuación, se mostrará un listado del hardware que ha sido necesario para la ejecución del presente proyecto, así como el porcentaje de imputación respecto a su vida útil, teniendo en cuenta lo estipulado según el fabricante.

- Ordenador Custom montado a piezas: Intel Core i5-6400, 2ssd (1 nvme y 1 sata), placa base Gigabyte B150-HD3P, RX580 4g Sapphire. HHD 4TB y memoria RAM 16GB. Vida Útil 8 años.

  11 meses / vida útil 8 años = 96 meses

  Imputación es de 11,45%

- Ordenador MacBook Pro-Retina 15 Pulgadas

  11 meses / vida útil 8 años = 96 meses

Imputación es de 11,45%

- Impresora Pantum M6700DW Laser

11 meses /vida útil 6 años = 72 meses

Imputación es de 15,27%

- Periféricos: 2 pantalla, ratón, teclado, alfombrilla y cascos. (Vida útil: 6 años).

11 meses /vida útil 6 años = 72 meses

Imputación es de 15,27%

- Conexión a internet. Fibra óptica de movistar de 1GB. Se cobrará por los meses de uso-

- Material de oficina

Hardware	Precio	Uso	Coste Total
Ordenador Custom montado	1.547,55 €	11,45%	177,19 €
MacBook Pro-Retina 15”	1.750,00 €	11,45%	200,38 €
Impresora pantum M6700DW	169,90 €	15,27%	25,94 €
Periféricos	369,90 €	15,27%	56,48 €
Conexión a internet	52 €	11 meses	572,00 €
Material de oficina	53 €	100 %	53,00 €
Total			1.085,00 €

Tabla 8: Imputación de Hardware

3.4.3 SOFTWARE

Para el cálculo de los costes asociados a la parte software que se han usado a lo largo del proyecto, se han tenido en cuenta el uso que se dará en el proyecto. Cabe destacar que una parte del software utilizado es gratuito por lo que no supondrá coste, pero otro porcentaje sí que lo es y habrá que calcular la parte imputable:

- Gratuito
  - Google Chrome
  - StarUML
  - Diagrams.net
  - MacOS
  - ninjamock
- De pago
  - Suite JetBrains

- Windows 10
- Office 365

Software	Precio	Uso	Coste total
Office 365	16,90 €/Mes	11 meses	185,90 €
Suite JetBrains	649,00 €/Año	11 meses	594,92€
Windows 10 Pro	259,00 €	11,45%	29,65 €
		TOTAL	810,47 €

Tabla 9: Imputación de Software

3.4.4 IMPUTACION AL PROYECTO

Una vez realizados los costes (RRHH, Hardware, Software) el coste del proyecto es el siguiente:

Imputaciones	Costes
RRHH	7.915,95 €
Hardware	1.085,00 €
Software	810,47 €
TOTAL	**9.811,42 €**

Tabla 10: Coste de Proyecto

Los 9.811,42 € es el coste de la empresa para desarrollar el proyecto. A este precio hay que sumarle el beneficio de la empresa que suele ser el 12% del coste del recurso, siendo en este caso de 1.177,37 €. Por tanto, el precio final del proyecto para un cliente será de 10.988,79 €

4 ANÁLISIS

A lo largo de la presente sección de la documentación, se expondrá el análisis realizado para su posterior implementación en la aplicación.

4.1 ACTORES

En este punto se expondrán los diferentes usuarios que harán uso de la aplicación, los cuales son los siguientes:

Actores		
ACT-1	Usuario no registrado	Será aquella persona que no está registrada en la plataforma, estará habilitada para poder realizar las operaciones más básicas del sistema.
ACT-4	Administrador	Será aquella persona que sí está identificada en el sistema puede realizar las labores de administración de la plataforma. Como puede ser aceptar o rechazar las reservas, añadir nuevos recursos…

Tabla 11: Actores del sistema

4.2 REGLAS DE NEGOCIO

Las reglas de negocio comprenden las diferentes políticas corporativas, regulaciones gubernamentales o estándares de la industria que debe respetar el producto software.

Reglas de Negocio	
RN-01	Solo el personal de la Escuela Ingeniería Informática de Segovia, encargado del mantenimiento de la información de la plataforma, puede iniciar sesión.
RN-02	Solo puede existir un usuario con el mismo email.
RN-03	Solo el usuario que tenga acceso a la cuenta puede hacer labores de mantenimiento de la aplicación.
RN-04	En la aplicación solo pueden estar dado de alta recursos que existan en la realidad.

Tabla 12: Reglas de negocio

4.3 OBJETIVOS DEL SISTEMA

Los objetivos del sistema expresan el fin último del sistema, los cuales permiten establecer una línea de finalización del proyecto. En el prese proyecto los objetivos del sistema son los siguientes:

OBJ-01	Gestión de Recursos
Descripción	El sistema deberá de permitir administrar todos los recursos de los que dispone la Escuela, es decir, darlos de alta y baja, mostrar y guardar su información y la localización de dónde se almacena.
Importancia	Alta

Tabla 13: Objetivos del sistema – gestión de recursos

OBJ-02	Clasificación de Recursos
Descripción	El sistema deberá permitir que un usuario con los permisos adecuados pueda realizar las acciones necesarias para realizar una clasificación correcta de los recursos.
Importancia	Alta

Tabla 14: Objetivos del sistema – préstamo de recursos

4.4 REQUISITOS DE USUARIO

A lo largo de este punto se expondrán todos los requisitos de usuario que se han descubierto durante la fase de análisis.

Usuario no registrado	
RU-01	El usuario no registrado podrá observar un listado de recursos que posee la Escuela, además podrá realizar búsquedas de recursos en base a clases.
RU-02	Cualquier usuario no administrador no puede realizar un inicio de sesión.

Tabla 15: Requisitos de usuario de usuario no registrado

Administrador	
RU-06	El administrador podrá agregar una clase primaria.
RU-07	El administrador podrá listar las clases primarias dadas de alta en la plataforma.
RU-08	El administrador podrá agregar una clase secundaria.
RU-09	El administrador podrá listar las clases secundarias dadas de alta en la plataforma.
RU-10	El administrador podrá agregar un recurso en la plataforma.
RU-11	El administrador podrá modificar un recurso de la plataforma.

Tabla 16: Requisitos de usuario de usuario administrador

Para un mejor entendimiento de la terminología empleada, a continuación, se expondrá una definición de la terminología utilizada en el dominio de la aplicación:

- Clase Primera: Se denomina clase primera a la agrupación de diversos elementos de categorización que resultan ser padres de elementos de categorización más específicos. Las clases primarias son generalistas y no disponen de una superclase, es decir de una clase mayor.
- Clase secundaria: Se denomina clase secundaria a la agrupación de diversos elementos de categorización que resultan ser hijos de elementos de categorización más generales.
- Recurso: Elemento físico del que es propietario la Escuela de informática
- Localización: Laboratorio y/o armario en el que se ha depositado un determinado recurso.

4.5 DIAGRAMAS DE CASO DE USO

A lo largo de este punto se expondrán los casos de uso. Los casos de uso son la descripción del comportamiento que tiene el sistema en relación con las acciones del usuario. En la Figura 8: Esquema de usuarios se muestra la jerarquía de los actores del sistema.

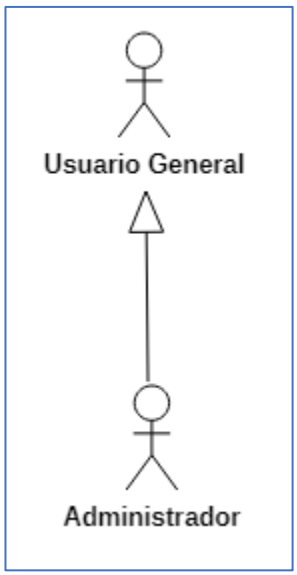


Figura 8: Esquema de usuarios

Una vez definidos los actores del sistema, se prosiguió a definición de los casos de uso. El primero en definir fue el diagrama de casos de uso del usuario general.

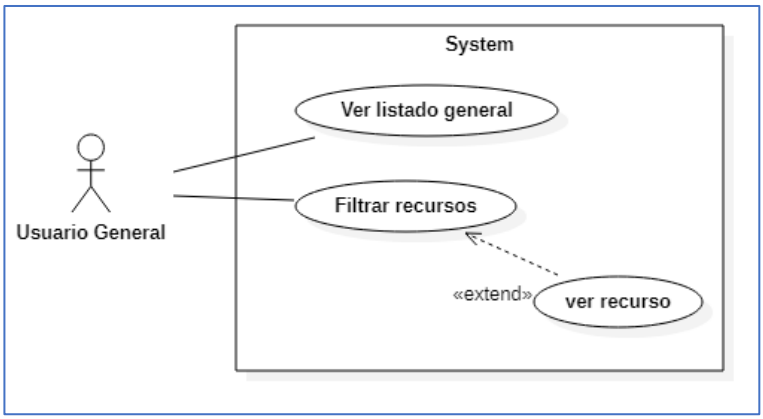


Figura 9: Caso de uso de usuario general

ID y Nombre:	CU-01: Ver listado general
Versión:	1.0
Actor principal	Usuario General
Descripción	Cualquier usuario que entre a la plataforma visualizará un listado de todos lo recursos que se han dado de alta en la plataforma
Precondiciones	El navegador desde que accede debe ser relativamente moderno
Postcondiciones	-
Flujo Normal	1. El usuario accede al sistema 2. El sistema le muestra una interfaz en el que se muestra un listado con todos los recursos dados de alta.
Excepciones	2. Si el sistema no es capaz de conectarse la base de datos, mostrará una lista de elementos vacía.
Trazabilidad	-
Prioridad	Alta

Tabla 17: Caso de uso de CU-01

ID y Nombre:	CU-02: Filtrar recursos
Versión:	1.0
Actor principal	Usuario General
Descripción	El usuario podrá realizar un filtrado de los recursos basándose en las clases que se utilizan para categorizarse.
Precondiciones	-
Postcondiciones	-
Flujo Normal	1. El usuario accede al sistema. 2. El sistema le muestra la interfaz de inicio. 3. El usuario selecciona la opción de filtrado de recursos.

	4. El sistema carga la interfaz correspondiente. 5. El sistema le mostrará una interfaz en la que se irán habilitando los elementos necesarios para el filtrado de recursos.
Excepciones	2. Si el sistema no es capaz de conectarse la base de datos, mostrará una lista de elementos vacía.
Trazabilidad	-
Prioridad	Alta

Tabla 18: Caso de uso de CU-02

ID y Nombre:	**CU-03: Ver Recurso**
Versión:	1.0
Actor principal	Usuario General
Descripción	El usuario podrá ver todos los detalles de un recurso en particular
Precondiciones	El usuario debe de haber seleccionado un recurso que este dado de alta en la plataforma.
Postcondiciones	-
Flujo Normal	1. El usuario selecciona un recurso. 2. El sistema realiza una petición a la BBDD con los datos del recurso. 3. El sistema una vez que recupera la información la muestra en el visualizador de recursos.
Excepciones	2. Si el sistema no es capaz de conectarse la base de datos, mostrará una lista de elementos vacía.
Trazabilidad	CU-02
Prioridad	Alta

Tabla 19: Caso de uso de CU-03

Finalizado la definición de los casos de uso del usuario general, se prosiguió con la definición de los casos de uso del administrador.

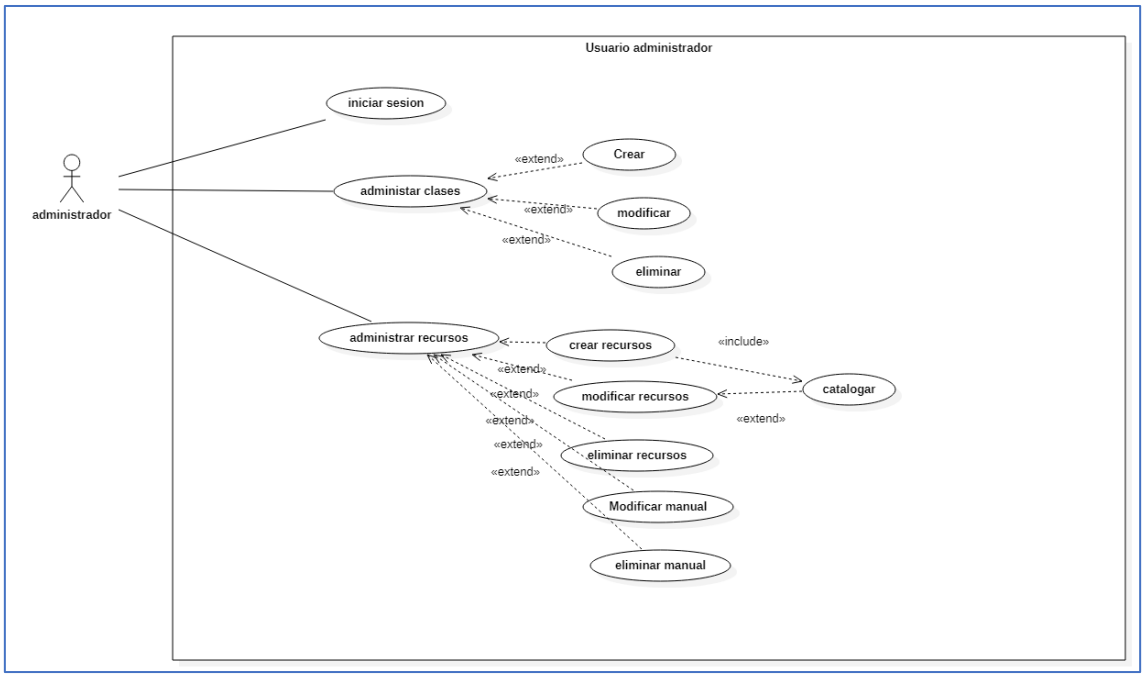


Figura 10: Caso de uso de usuario administrador.

ID y Nombre:	CU-04: Iniciar Sesión
Versión:	1.0
Actor principal	Administrador
Descripción	El usuario iniciará sesión para obtener un token oauth y se le habilitará para realizar las operaciones de administración.
Precondiciones	El administrador debe de haber accedido a la consola de Firebase y metido su usuario y contraseña de acceso al sistema.
Postcondiciones	El sistema habilitará las operaciones de administración de los recursos en la plataforma.
Flujo Normal	1. El usuario solicitará al sistema identificarse en el sistema. 2. El sistema solicitará al usuario el email y la contraseña. 3. El usuario facilitará la información y le indicará al sistema que verifique. 4. El sistema comprobará la información y le habilitará las operaciones que puede ejecutar.
Excepciones	3.1 Si el sistema no puede conectarse a la Base de datos, el sistema no realizará la autentificación mostrando un mensaje de error. 3.2 Si el sistema no autentifica al usuario, le mostrará un mensaje de error.
Trazabilidad	-
Prioridad	Alta

Tabla 20: CU-04: Iniciar Sesión.

ID y Nombre:	CU-05: Administrar clases
Versión:	1.0
Actor principal	Administrador
Descripción	Se le mostrará una interfaz al usuario en el que se listará las clases y se habilitará las diferentes acciones.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	Se debe de habilitar las diferentes opciones de administración.
Flujo Normal	1. El usuario debe de seleccionar la opción de administrar clases 2. El sistema cargara la interfaz de administración de clases
Excepciones	2. Si el sistema no es capaz de conectarse la base de datos, mostrará una lista de elementos vacía.
Trazabilidad	CU-04
Prioridad	Alta

Tabla 21: CU-05: Administrar clases.

ID y Nombre:	CU-06: Crear Clase
Versión:	1.0
Actor principal	Administrador
Descripción	El usuario rellenará un formulario mediante el cual podrá dar de alta una nueva clase. Tanto primaria como secundaria.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema debe de haber introducido en la base de datos la nueva clase.
Flujo Normal	1. El usuario solicita al sistema dar de alta una nueva clase. 2. El sistema muestra un formulario al usuario solicitando la información. 3. El usuario facilita la información solicitada al sistema. 4. El sistema valida la información facilitada y la introduce en la base de datos.
Excepciones	3.1 Si la información no es válida, le advierte al usuario del error.
Trazabilidad	CU-05
Prioridad	Alta

Tabla 22: CU-06: Crear clase.

ID y Nombre:	CU-07: Modificar Clase
Versión:	1.0
Actor principal	Administración
Descripción	El usuario podrá modificar cierta información de las clases dadas de alta en el sistema, siempre y cuando no haya nada asociado a ella.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá modificado la información en de la clase.
Flujo Normal	1. El usuario solicita al sistema modificar las clases dadas de alta. 2. El sistema habilita al usuario modificar aquellas clases que no tengan recursos dados de alta. 3. El usuario selecciona la clase que quiere modificar. 4. El sistema le solicita la información nueva que puede modificar. 5. El usuario aporta la información que desea modificar. 6. El sistema valida la información y la almacena en el sistema.
Excepciones	6.1 El sistema si detecta algún error en la información, muestra una advertencia
Trazabilidad	CU-05
Prioridad	Alta

Tabla 23: CU-07: Modificar clase.

ID y Nombre:	CU-08: Eliminar Clase
Versión:	1.0
Actor principal	Administración
Descripción	El usuario eliminar una clase dada de alta en el sistema
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá eliminado la información de la clase en la base de datos.
Flujo Normal	1. El usuario solicita al sistema eliminar las clases dadas de alta. 2. El sistema habilita al usuario a eliminar clases. 3. El usuario indica la clase que quiere eliminar. 4. El sistema verifica que no tiene ningún recurso asociado. 5. El sistema elimina la información.
Excepciones	5. El sistema si detecta algún error en la información, muestra una advertencia
Trazabilidad	CU-05
Prioridad	Alta

Tabla 24: CU-08: Eliminar clase.

ID y Nombre:	CU-09: Administrar Recursos
Versión:	1.0
Actor principal	Administrador
Descripción	Se le mostrará una interfaz al usuario en el que se listará las clases y se habilitará las diferentes acciones.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	Se debe de habilitar las diferentes opciones de administración
Flujo Normal	1. El usuario debe de seleccionar la opción de administrar recursos. 2. El sistema cargara la interfaz de administración de recursos.
Excepciones	2. Si el sistema no es capaz de conectarse a la base de datos, mostrará el visualizador vacío.
Trazabilidad	CU-04
Prioridad	Alta

Tabla 25: CU-09: Administrar Recursos.

ID y Nombre:	CU-10: Crear recurso
Versión:	1.0
Actor principal	Administrador
Descripción	El usuario facilitará la sistema la información necesaria para dar de alta un nuevo recurso.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema debe de haber dado de alta el nuevo recurso.
Flujo Normal	3. El usuario solicita al sistema dar de alta un nuevo recurso. 4. El sistema muestra un formulario al usuario solicitando la información. 5. El usuario facilita la información solicitada al sistema. 6. El sistema valida la información facilitada y dará de alta el recurso en el sistema.
Excepciones	3.1 Si la información no es válida, le advierte al usuario del error.
Trazabilidad	CU-09
Prioridad	Alta

Tabla 26: CU-10: Crear recurso.

ID y Nombre:	CU-11: Modificar Recurso
Versión:	1.0
Actor principal	Administración
Descripción	El usuario podrá modificar cierta información de los recursos dados de alta en el sistema.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá modificado la información del recurso.
Flujo Normal	1. El usuario solicita al sistema modificar los recursos dados de alta. 2. El sistema habilita al usuario modificar los recursos. 3. El usuario indicará el recurso que quiere modificar. 4. El sistema muestra la información del recurso deseado que puede modificar. 5. El usuario la nueva información del recurso. 6. El sistema valida la información y la almacena en el sistema.
Excepciones	6.1 El sistema si detecta algún error en la información, muestra una advertencia.
Trazabilidad	CU-09
Prioridad	Alta

Tabla 27: CU-11: Modificar recurso.

ID y Nombre:	CU-12: Eliminar Recurso
Versión:	1.0
Actor principal	Administración
Descripción	El usuario eliminar un recurso dado de alta en el sistema.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá eliminado la información de un recurso.
Flujo Normal	1. El usuario solicita al sistema eliminar un determinado recurso. 2. El sistema habilita al usuario a eliminar recursos. 3. El usuario indica el recurso que quiere eliminar. 4. El sistema verifica la solicitud. 5. El sistema elimina la información.
Excepciones	4. El sistema si detecta algún error en la información, muestra una advertencia.
Trazabilidad	CU-05
Prioridad	Alta

Tabla 28: CU-12: Eliminar Recurso.

ID y Nombre:	CU-13: Administrar Localizaciones
Versión:	1.0
Actor principal	Administrador
Descripción	Se le mostrará una interfaz al usuario en el que se listará las localizaciones y se habilitará las diferentes acciones.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	Se debe de habilitar las diferentes opciones de administración
Flujo Normal	1. El usuario debe de seleccionar la opción de administrar localizaciones. 2. El sistema le mostrara las opciones de administración de las localizaciones.
Excepciones	2. Si el sistema no es capaz de conectarse a la base de datos, no mostrará información
Trazabilidad	CU-04
Prioridad	Alta

Tabla 29: CU-13: Administrar Recursos.

ID y Nombre:	CU-14: Crear localizaciones
Versión:	1.0
Actor principal	Administrador
Descripción	El usuario facilitará la información al sistema de la localización que quiere dar de alta.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema debe de haber introducido en la base de datos la nueva localización.
Flujo Normal	1. El usuario solicita al sistema dar de alta una nueva localización. 2. El sistema solicita al usuario la información de la nueva localización. 3. El usuario facilita la información solicitada al sistema. 4. El sistema valida la información facilitada y la introduce en la base de datos.
Excepciones	3.1 Si la información no es válida, le advierte al usuario del error.
Trazabilidad	CU-13
Prioridad	Alta

Tabla 30: CU-14: Crear localizaciones.

ID y Nombre:	CU-15: Modificar localizaciones
Versión:	1.0
Actor principal	Administración
Descripción	El usuario podrá modificar cierta información de las localizaciones dadas de alta en el sistema, siempre y cuando no haya nada asociado a ella.
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá modificado la información de la localización.
Flujo Normal	1. El usuario solicita al sistema modificar las localizaciones dadas de alta. 2. El sistema habilita al usuario modificar los recursos. 3. El usuario indica la localización que quiere modificar. 4. El sistema solicita la nueva información de la localización que se quiere modificar. 5. El usuario modifica la información que desea modificar. 6. El sistema valida la información y la almacena en el sistema.
Excepciones	6.1 El sistema si detecta algún error en la información, muestra una advertencia.
Trazabilidad	CU-13
Prioridad	Alta

Tabla 31: CU-15: Modificar localización.

ID y Nombre:	CU-16: Eliminar localización
Versión:	1.0
Actor principal	Administración
Descripción	El usuario eliminar una localización dada de alta
Precondiciones	El usuario debe de estar identificado y autorizado en el sistema.
Postcondiciones	El sistema habrá eliminado la información de una localización
Flujo Normal	1. El usuario solicita al sistema eliminar una determinada localización. 2. El sistema habilita al usuario a eliminar localizaciones. 3. El usuario indica la localización que quiere eliminar. 4. El sistema verifica la solicitud. 5. El sistema elimina la información.
Excepciones	4. El sistema si detecta algún error en la información, muestra una advertencia.
Trazabilidad	CU-13
Prioridad	Alta

Tabla 32: CU-16: Eliminar Recurso.

4.6 REQUISITOS FUNCIONALES

A continuación, se mostrarán los requisitos funcionales que han sido identificados a partir de los requisitos de usuario. Estos requisitos identifican las acciones que el sistema debe realizar ante las interacciones del usuario.

4.6.1 GESTIÓN DE USUARIOS

RF-01: El sistema mostrará un listado de todos los recursos dados de alta en el sistema.

RF-02: El sistema mostrará un menú con las funcionalidades que puede realizar el usuario.

RF-03: El sistema mostrará una opción de login en el sistema.

RF-04: El sistema comprobará que está registrado en la aplicación.

RF-05: El sistema permitirá autentificarse al usuario.

RF-06: El sistema iniciará una sesión en el sistema una vez que se verifique la información introducida por el usuario.

RF-07: El sistema mostrará la información que el sistema tiene registrada sobre el usuario.

RF-08: El sistema actualizará la información registrada que el sistema tiene dada de alta.

RF-09: El sistema mostrará la información actualizada del usuario

4.6.2 ADMINISTRADOR

RF-10: El sistema visualizará todas las clases primarias dadas de alta en el sistema.

RF-11: El sistema permitirá dar de alta nuevas clases primarias.

RF-12: El sistema permitirá administrar todas las clases primarias existentes.

RF-13: El sistema visualizará todas las clases secundarias dadas de alta en el sistema.

RF-14: El sistema permitirá administrar todas clase secundaria en el sistema.

RF-15: El sistema permitirá dar de alta un nuevo recurso en el sistema.

RF-16: El sistema permitirá actualizar los datos de un determinado recurso.

RF-17: El sistema permitirá eliminar los datos de un determinado recurso

RF-18: El sistema permitirá visualizar todos los recursos dados de alta en el sistema.

RF-19: El sistema permitirá visualizar todas las localizaciones existentes en el sistema

RF-20: El sistema permitirá administrar todas las localizaciones dadas de alta en el sistema y registrar nuevas.

4.7 REQUISITOS NO FUNCIONALES

Una vez definidos los requisitos funcionales, proseguí con la definición de los requisitos no funciones. Estos requisitos especifican criterios para evaluar la operación de un servicio de tecnología de información, en contraste con los requerimientos funcionales que especifican los comportamientos específicos.

4.7.1 DISPONIBILIDAD

RNF-01: La plataforma deberá estar disponible 24/7 los 365 días del año.

4.7.2 ACCESIBILIDAD

RNF-02: La plataforma web será accesible desde cualquier dispositivo, ya sea ordenadores, tabletas o Smartphones.

4.7.3 ESCALABILIDAD

RNF-03: La plataforma será escalable en cualquier momento, sin que ello repercuta en las funcionalidades disponibles.

4.7.4 USABILIDAD

RNF-04: La plataforma será lo más sencilla posible, permitiendo que el periodo de aprendizaje de los usuarios sea lo mínimo posible.

RNF-05: La interfaz gráfica de la plataforma permitirá adaptarse a las dimensiones de los dispositivos.

RNF-06: La plataforma validara los datos introducidos por el usuario. En caso de que algún dato no sea no valido, el sistema le mostrará un mensaje de error al usuario.

4.7.5 SEGURIDAD

RNF-07: Los usuarios serán exclusivamente usuarios de la “Escuela de Ingeniería Informática de Segovia” dados de alta por el administrador de la plataforma.

RNF-08: Las contraseñas tendrán una longitud mínima de 8 caracteres y una longitud máxima de 16.

RNF-09: Las contraseñas nunca se almacenarán en plano, se almacenarán utilizando el algoritmo de cifrado “SHA-256” en la BBDD.

RNF-10: La plataforma tendrá diferentes partes con acceso restringido, dependiendo de los permisos de los que disponga el usuario, otorgado por los guardianes establecidos.

RNF-11: Los usuarios no autentificados en el sistema únicamente podrán acceder a la parte pública de la plataforma.

RNF-12: El sistema cumplirá la Ley de Protección de Datos.

RNF-13: El sistema creará una copia de seguridad de la base de datos todos los días de madrugada, coincidiendo con la franja de menos uso de esta.

4.7.6 RESTRICCIONES

RNF-14: Los usuarios serán únicamente dados de alta por el administrador de la plataforma.

RNF-15: Únicamente se podrá introducir una URL de la imagen del recurso.

4.7.7 ATRIBUTOS DE CALIDAD

RNF-16: El sistema dispondrá del idioma español.

RNF-17: Los usuarios utilizarán UTF-8 como formato de caracteres.

4.8 REQUISITOS DE INFORMACIÓN

A lo largo de este punto, se expondrán los requisitos de información que necesita la plataforma para funcionar. Para ello, se identificará la información que se debe guardar de cada uno de los elementos almacenados en el sistema.

- **RI-01:** El sistema deberá almacenar toda la información relacionada con los datos de un determinado recurso.
- **RI-02:** El sistema deberá almacenar toda la información relacionada con los datos de una determinada clase.
- **RI-03:** El sistema deberá almacenar toda la información relacionada con los datos de una determinada localización.
- **RI-04:** El sistema deberá almacenar todas las imágenes de los recursos.
- **RI-05:** El sistema deberá almacenar toda la información relacionada con los datos de acceso del administrador.
    - Email de acceso al sistema.
    - Contraseña de acceso al sistema.

4.8 MODELO ENTIDAD – RELACIÓN

Finalizado el proceso de análisis de la información que se debe de almacenar en la base de datos, se prosiguió con la realización del diagrama entidad relación. Este diagrama muestra las entidades y las relaciones que tendrá la información que se almacena en ella. El diagrama resultado de este proceso puede observarse en la siguiente ilustración:

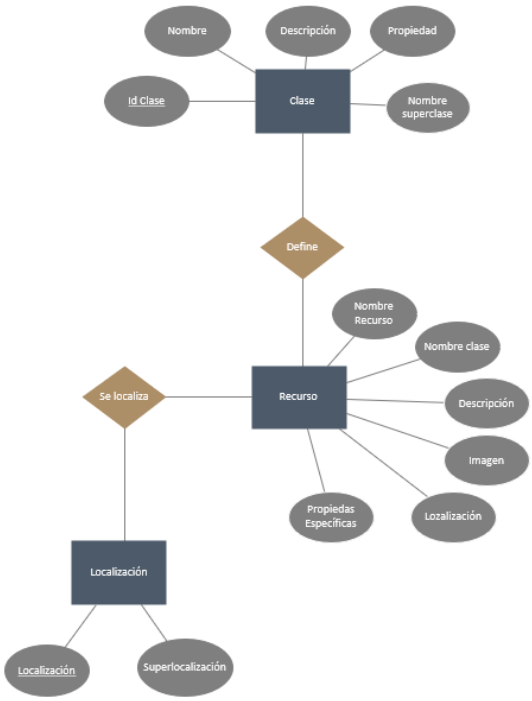


Figura 11: Diagrama entidad relación.

5 DISEÑO

Una vez concluida la fase de análisis, se continuó con la fase de diseño de la solución. En el presente apartado se expondrán todos los procesos que se siguieron para llegar al diseño final. Cabe destacar que en este apartado solo se expondrá el diseño que se hizo para la solución de "Angular y Firebase" y no para la de "Java EE y MySQL" que fue la otra alternativa que se consideró

5.1 ARQUITECTURA LÓGICA

En primer lugar, se realizó el diseño de la arquitectura lógica de la solución. Esta establece los patrones y las abstracciones necesarias que proporcionan un marco definido y claro de interactuación de los diferentes elementos software.

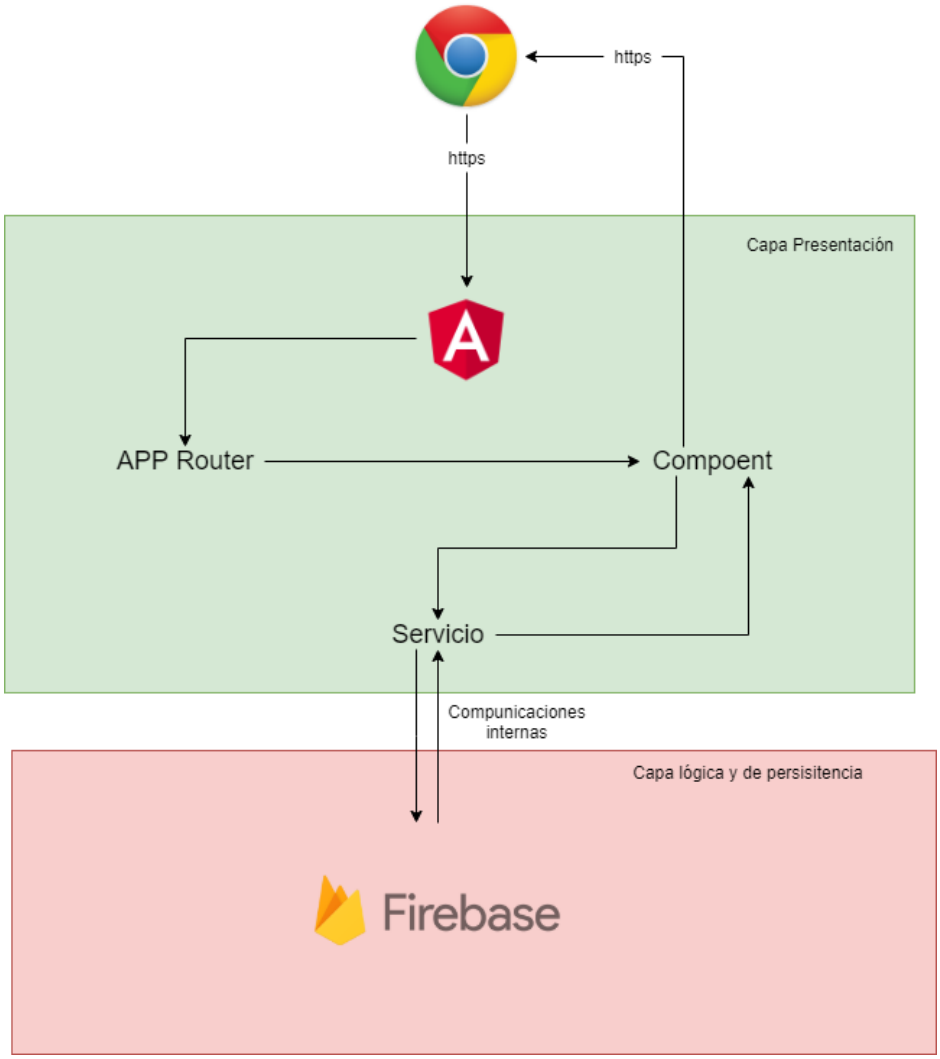


Figura 12: Arquitectura lógica

Tal y como se ha comentado anteriormente, el proyecto se ha basado en la utilización de Angular y Firebase. Antes de nada, hay que explicar que es Google Firebase.

Google Firebase es una plataforma *SAAS (Software as a Service)* para el desarrollo tanto de aplicaciones web como para dispositivos móviles. Esta plataforma está concebida como un *SaaS*, es decir, un software como servicio. En este caso se ha decidido utilizar esta plataforma debido a la poca complejidad que tenía el *BackEnd*. Firebase ha proporcionado todas las herramientas necesarias para el desarrollo de la aplicación, excluyendo el diseño y desarrollo de la parte *back*.

En este proyecto se ha decidido utilizar un modelo por capas, en concreto se tienen las siguientes dos capas:

- Capa de presentación: Se encarga de procesar todas las peticiones del usuario. El servidor de Angular recibe las peticiones, las pasa al app router, que se encarga de determinar si tiene los privilegios adecuados para acceder a ellas. Si los tiene entonces acceden a ellos y realizan las operaciones, en caso contrario entra el guardián que protege la ruta e impedirá que el usuario acceda a esa parte de la aplicación. Una vez que el usuario accede a la parte correspondiente, se ejecuta la acción y se pasa la información a Firebase.

- Capa lógica y capa de persistencia: Al utilizar Firebase, parte de la lógica se implementa en la capa de presentación gracias al uso de directivas específicas. De Firebase se han usado dos apartados:

    - El primero es el de autorización: Éste se encarga de autentificar y autorizar a un usuario para poder acceder a la parte de administración de la aplicación, así como de darle los privilegios necesarios para poder modificar la información.

    - La segunda es de almacenamiento: Gracias al uso de Firebase y a sus reglas de seguridad, se ha atraído el nivel de seguridad de la aplicación. Ya que con las reglas establecidas no es posible el acceso a la modificación de la información.

5.2 ARQUITECTURA FÍSICA

Una vez completado el diseño de la arquitectura lógica de la aplicación, se continuó con la arquitectura física, la cual establecerá los elementos físicos necesarios para la ejecución de la aplicación. Cabe destacar que este diseño satisface de forma completa todos los requisitos de seguridad, disponibilidad y escalabilidad mencionados en el apartado: “Requisitos No Funcionales”.

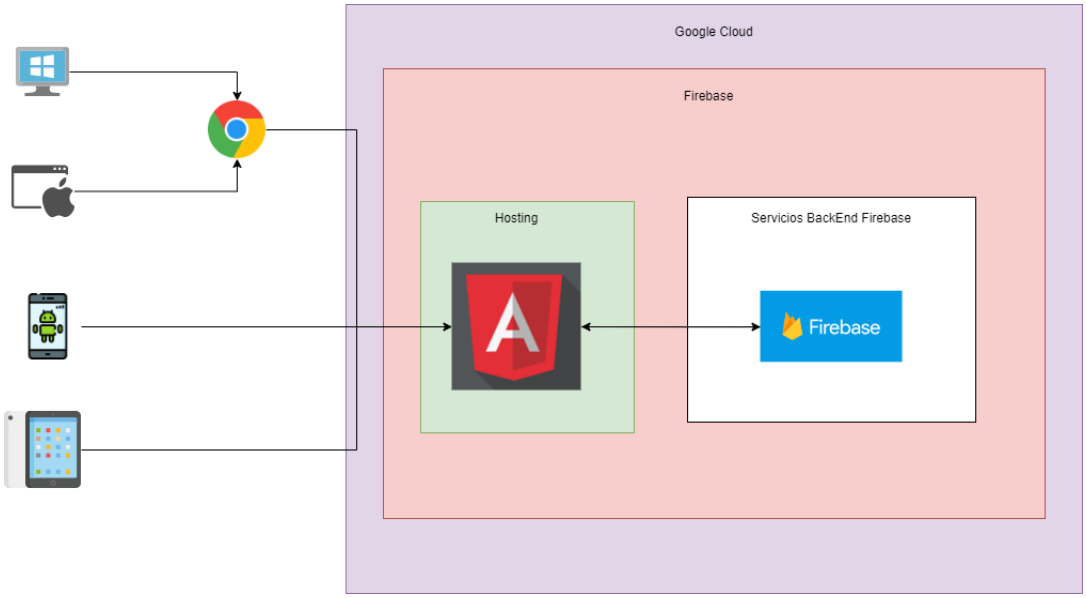


Figura 13: Arquitectura Física

El diseño de la arquitectura física se ha llevado a cabo teniendo en cuanta las 2 capas anteriormente comentadas en la arquitectura lógica del sistema.

En concreto, para el diseño físico, solo se debe de tener en cuenta la capa de presentación, ya que la capa lógica y de persistencia se delega al servicio de Firebase.

En la capa de presentación se dispone de 3 puntos de acceso, el navegador web y las aplicaciones móviles para los sistemas Android e iOS. Todas las peticiones son filtradas por el *firewall*, a continuación, éstas son enviadas y distribuidas a los balanceadores de carga. De ahí pasan a los servidores que se encargan de procesar y responder todas las peticiones.

La seguridad y la escalabilidad del sistema están garantizada. Ya que para la seguridad se dispone de diferentes *firewalls* y la seguridad del *BackEnd* y la persistencia están aseguradas por la plataforma Firebase. La escalabilidad se asegura por los diferentes balanceadores y servidores que se utilizan.

5.3 DISEÑO BASE DE DATOS

Presentado el diagrama Entidad Relación del apartado Modelo Entidad – Relación, se mostrará la estructura de la base de datos y como se almacena la información. Teniendo en cuenta que se utiliza un gestor de datos NoSQL

Figura 14: Diseño de base de datos

5.4 DICCIONARIO DE DATOS

5.4.1 CLASE

Columna	Tipo	Nulo	Comentarios
Id	String	No	Identificador único de la clase.
Nombre	String	No	Nombre que se le da a una determinada clase.
Descripción	String	No	Descripción que se le da a una clase.
Propiedades	String	No	Propiedades de la clase.
Nombre Superclase	String	No	Nombre de la clase a la que pertenece.

Tabla 33: Diccionario de Clase.

5.4.2 LOCALIZACION

Columna	Tipo	Nulo	Comentarios
Id	String	No	Identificador único de la clase.
Localización	String	No	Nombre de la localización que se da de alta.
Superlozalización	String	No	Nombre de la localización de la que depende.

Tabla 34: Diccionario de localización.

5.4.3 RECURSO

Columna	Tipo	Nulo	Comentarios
Id	String	No	Identificador único de la clase.
NombreClase	String	No	Nombre de la clase de la que depende.
NombreRecurso	String	No	Nombre del recurso.
PropiedadesExpecíficas	String	No	Propiedades del recurso.
Localización	String	No	Localización en la que se almacena el recurso.
Imagen	String	No	URL de almacenamiento de la imagen.

Tabla 35: Diccionario de localización.

5.4.4 USUARIO

Columna	Tipo	Nulo	Comentarios
Email	String	No	Email del usuario.
Password	String	No	Hash de la contraseña del usuario.

Tabla 36: Diccionario de usuario.

5.5 ACUERDOS DE DISEÑO DE LA INTERFAZ

A lo largo del presente punto, se mostrarán los diferentes bocetos que se utilizaron para la creación de la interfaz de usuario que se utiliza en la aplicación. Se ha utilizado un diseño modular, en el que cada sección que se muestre al usuario depende de los privilegios que tengan los diferentes actores que utilizan la aplicación.

Cabe destacar que el diseño propuesto a lo largo de los siguientes bocetos persigue crear una interfaz de usuario lo más amigable e intuitiva posible. Además, esta interfaz será reutilizada para las versiones de las aplicaciones móviles.

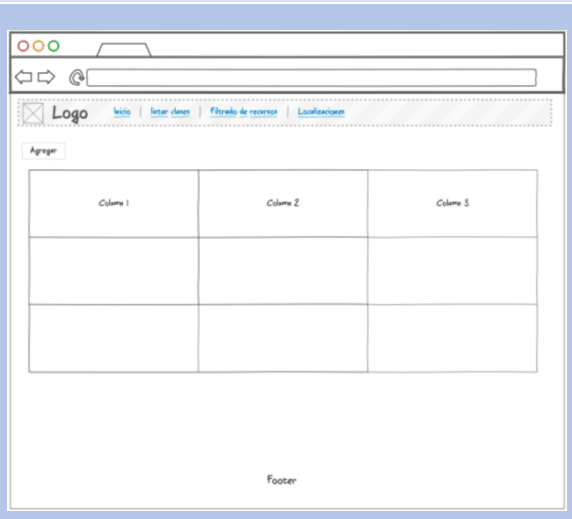
ID	**DI-01**
Nombre	Página Principal
Descripción	Página principal de la aplicación donde cualquier persona podrá visualizarla
Activación	Entrar en la aplicación
Eventos	Agregar nuevo recurso Visualizar todos los recursos Panel de navegación a diferentes funcionalidades
Boceto	

Tabla 37: Diseño de interfaz.

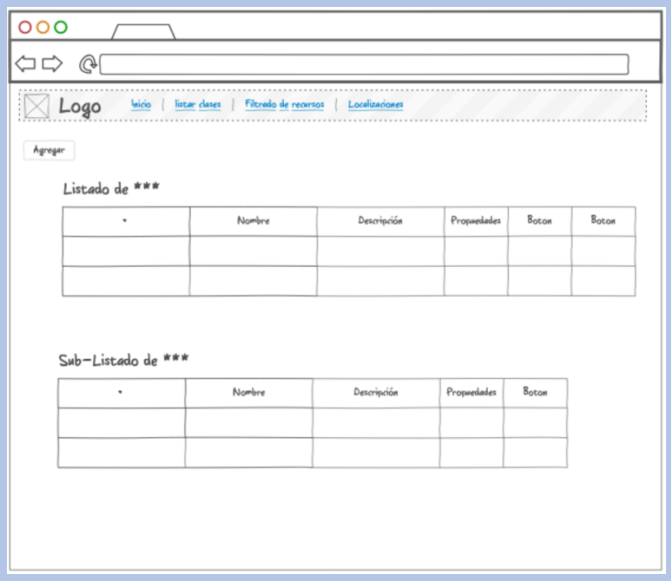
ID	DI-02
Nombre	Listado de clases, recursos y localizaciones
Descripción	Se muestra un listado de las clases, recursos y localizaciones. Además de poder agregar nuevas clases
Activación	Seleccionar la opción de listado de clases, recursos y localizaciones
Eventos	Seleccionar recurso, clase o localización. Agregar recurso, clase o localización Seleccionar elemento
Boceto	

Tabla 38: Diseño de interfaz.

ID	DI-03
Nombre	Filtrado de recursos
Descripción	Un usuario podrá realizar una búsqueda de recursos en base al árbol de clases y la categorización de estos
Activación	Seleccionar la funcionalidad
Eventos	Seleccionar clase primaria y secundaria

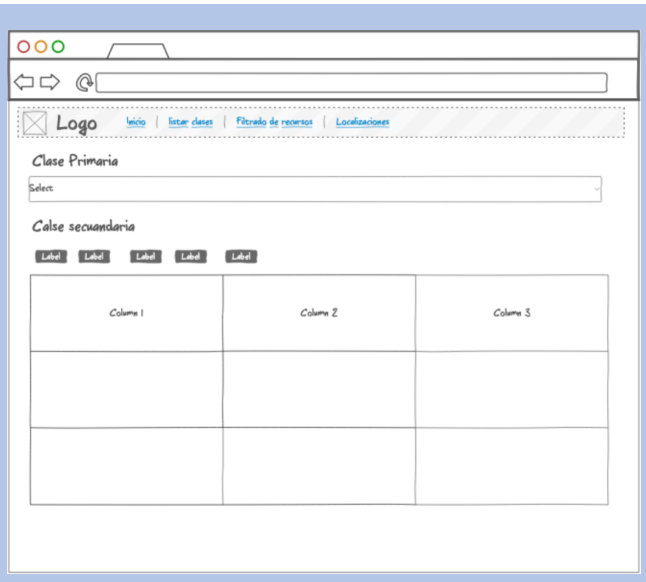
Boceto	

Tabla 39: Diseño de interfaz.

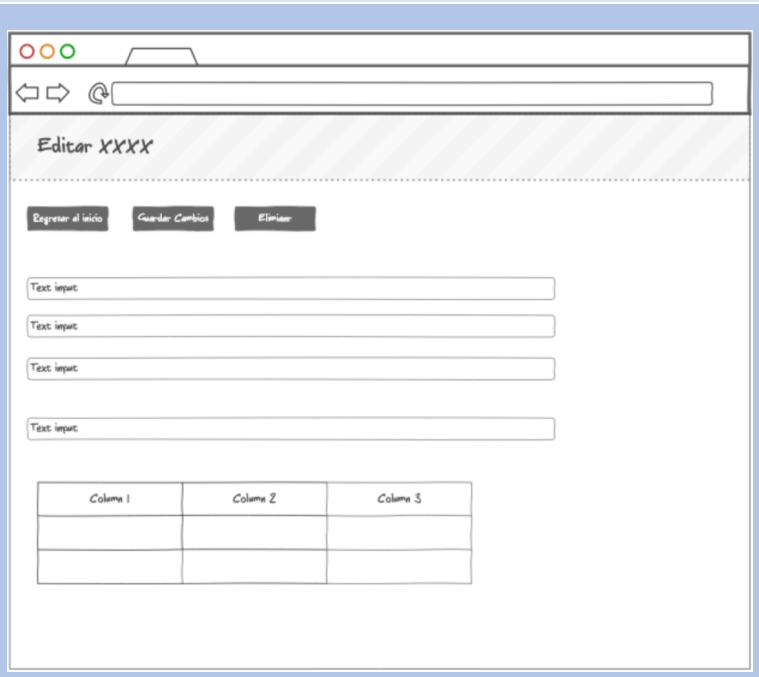
ID	**DI-04**
Nombre	Editar Clase, Recurso y localización
Descripción	Un usuario con los privilegios adecuados podrá editar la Clase, Recurso y localización
Activación	Seleccionar la funcionalidad
Eventos	Editar los campos de los registros
Boceto	

Tabla 40: Diseño de interfaz.

5.6 DIAGRAMA DE CLASES

Una aplicación Angular se compone de múltiples de componentes/clases. Es por ello, que para una mayor compresión de cómo funciona la aplicación y como se relacionan las clases entre sí.

Teniendo en cuenta todas las clases que componen el proyecto, este se puede dividir en los siguientes paquetes:

- Componentes: Contienen todas las clases (Clases HTML, CSS y TypeScript) que dotan de funcionalidades a la aplicación.
- Guardianes: Contiene todas las clases TypeScript encargadas de proteger las rutas de acceso a las diferentes funcionalidades
- Modelo: Contiene todas las clases TypeScript que representan entidades que conforman la base de datos Firebase.
- Servicios: Contiene todas las clases TypeScript que implementan los métodos necesarios para poder leer y escribir registros en la base de datos.

A continuación, se muestra dicho diagrama de clases, el cual fue generado utilizando una herramienta propia de WebStorm.

Figura 15: Diagrama de clases

Tal y como se puede observar la complejidad de este es elevada debido al numero de clases intervinientes. Es por ello, que se ha decidido separarla en las secciones mostradas a continuación.

5.6.1 COMPONENTES

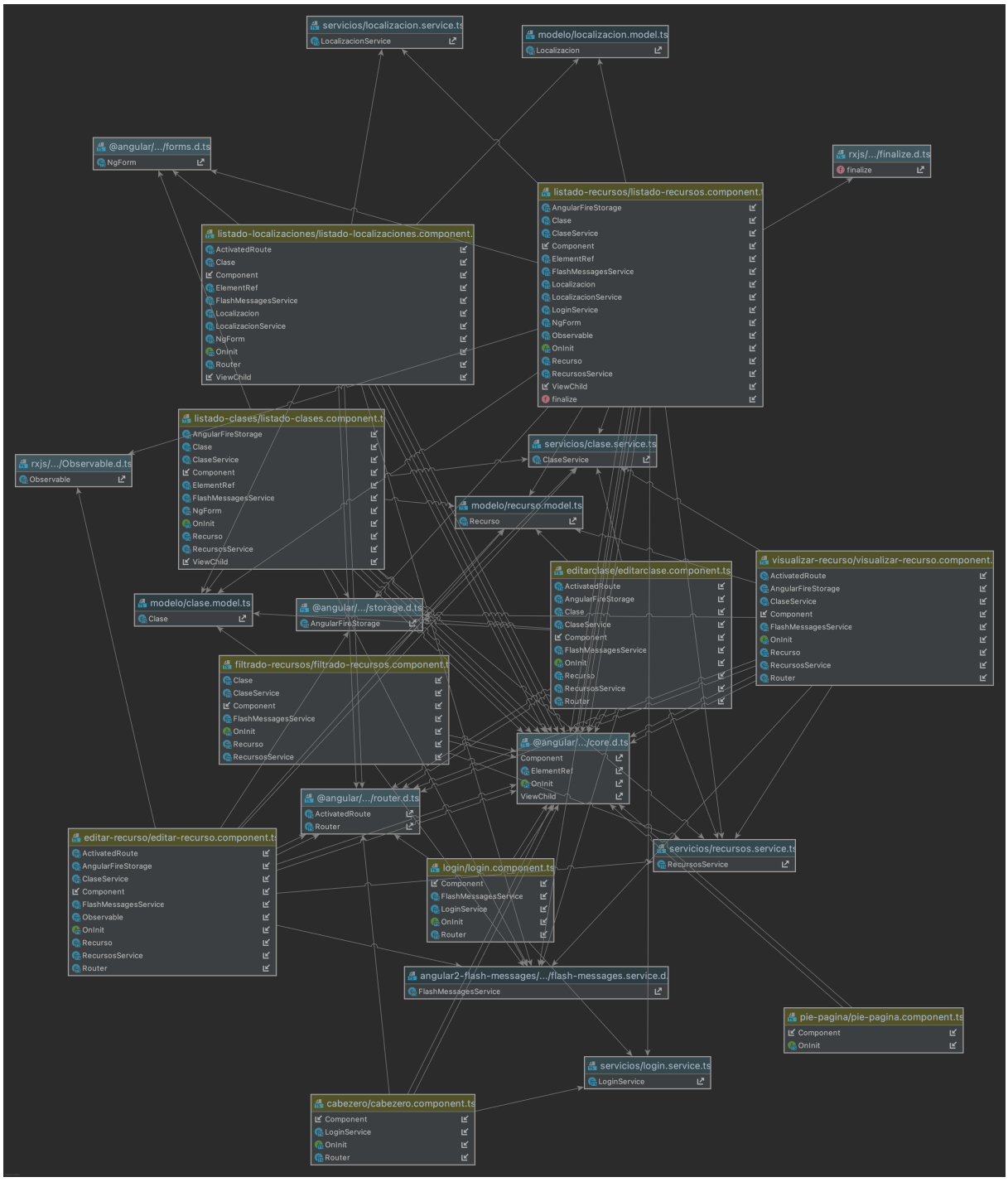


Figura 16: Diagrama de clases de componentes

El patrón utilizado para el desarrollo de la solución es el siguiente:

- Un usuario hace una llamada a una url determinada, la cual es cogida por el router de angular, la cual comprueba a que hace referencia y el guardián que tiene asociado. Si el guardián aprueba el acceso, se carga el componente en lugar del que se tenia.

- Una vez que el componente es cargado, se construye con todas las inyecciones de clase y servicios con los que se ha programado. A continuación, se inicia en ese momento se utiliza los servicios para acceder a *Firebase* para recuperar la información. Una vez que se han ejecutado todos los métodos, el componente construye la parte visual de la pagina web gracias al *HTML* y *CSS* que vienen definidos y establece la inyección del JS generado para el navegador.
- Finalmente se establecen los lisntener y se carga la información el sistema.
- Cabe destacar que los servicios utilizan los modelos para establecer el dominio de datos que se van a recuperar de los servicios externos, en este caso *Firebase.*

5.6.2 GUARDIANES

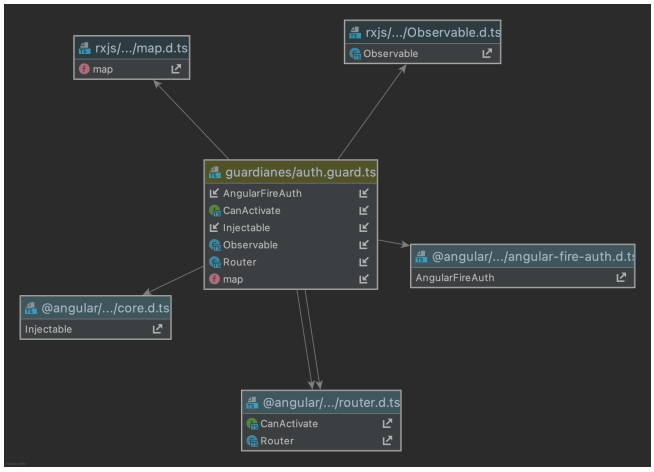


Figura 17: Diagrama de clases de Guardianes

5.6.3 SERVICIOS

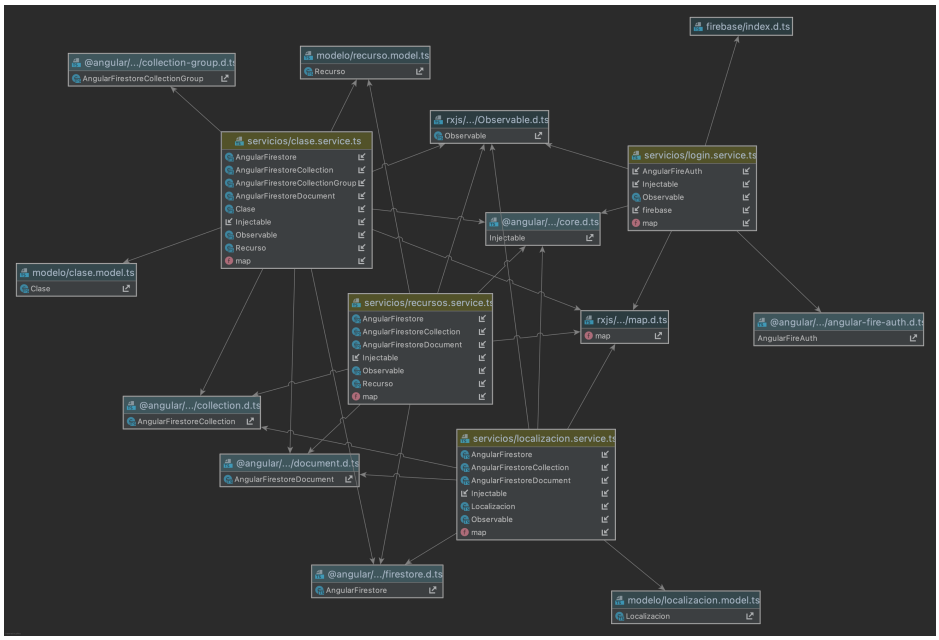


Figura 18: Diagrama de clases de Servicios

5.7 DIAGRAMAS DE SECUENCIA

A lo largo de este punto se expondrán los diagramas de secuencia que se han realizado para el desarrollo de la aplicación. Un diagrama de secuencia es un dibujo que describe las interacciones que hace el usuario con los diferentes componentes que componen el sistema.

Para no sobrecargar la documentación, se ha decido mostrar únicamente aquellos diagramas de secuencia que sean relevantes.

5.7.1 SECUENCIA LOGIN

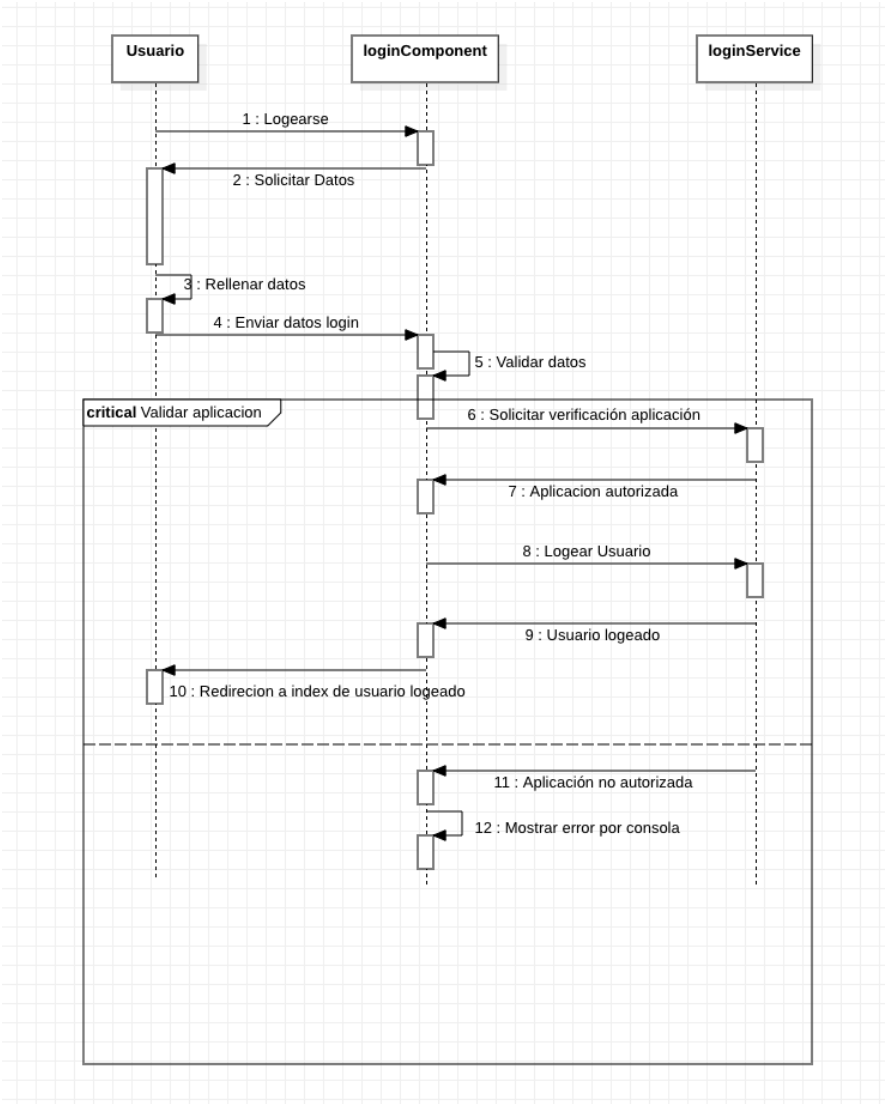


Figura 19: Diagrama de secuencia de login

5.7.2 SECUENCIA LISTADO DE RECURSOS

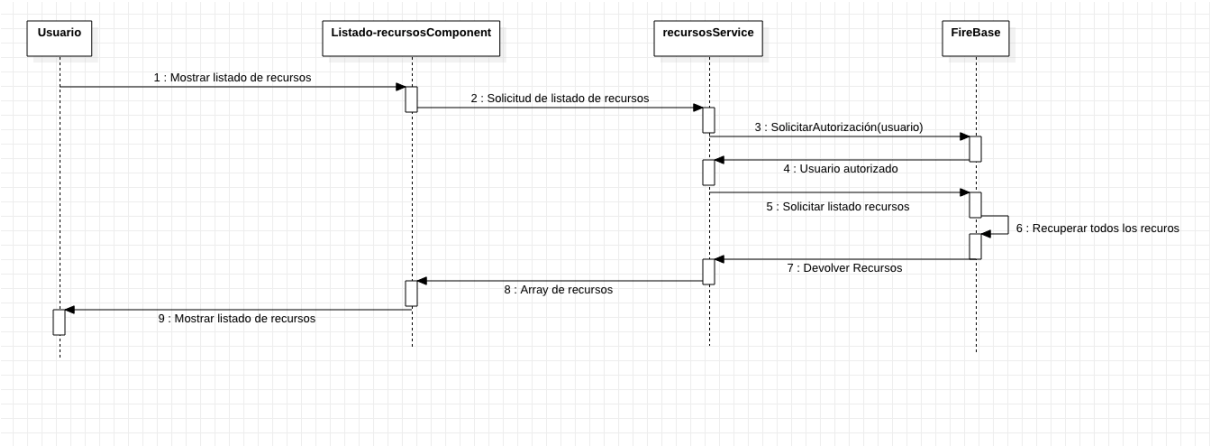


Figura 20: Diagrama de secuencia de listado de recursos

5.7.3 SECUENCIA AGREGAR RECURSOS

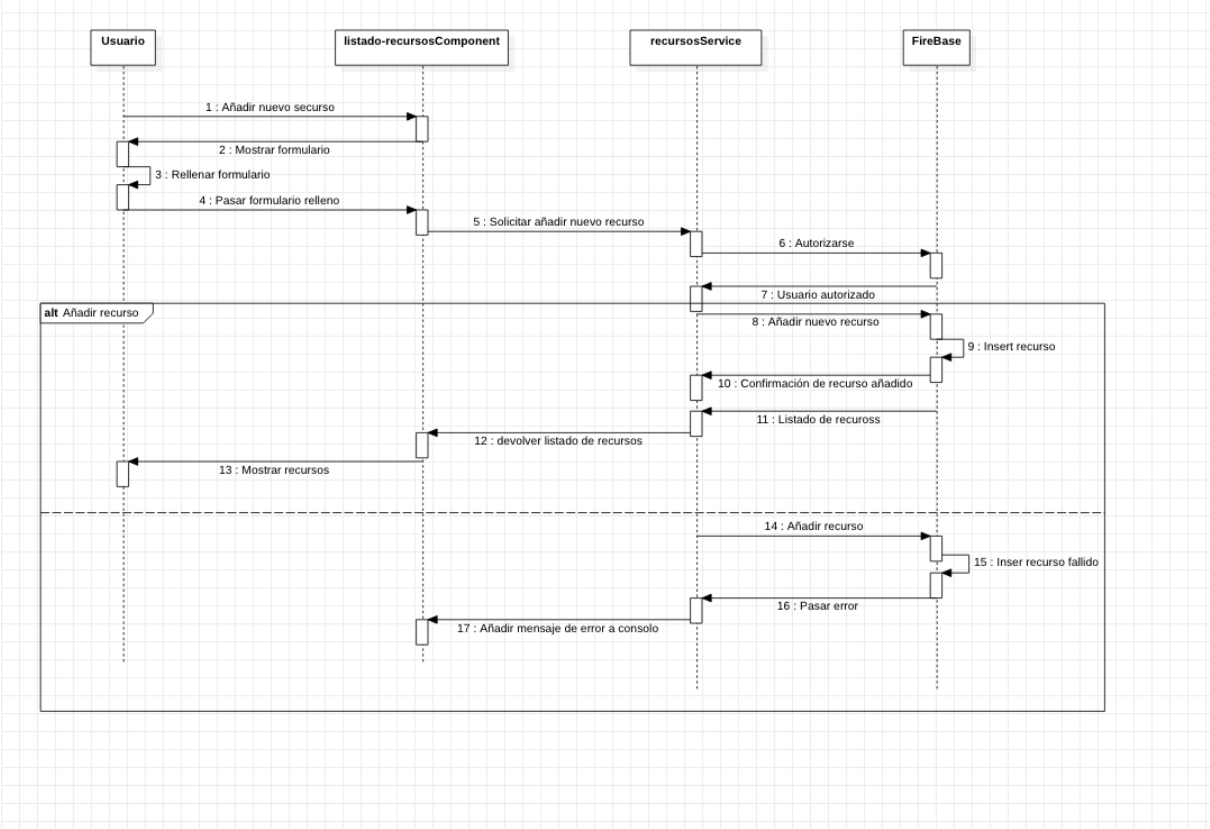


Figura 21: Diagrama de secuencia de agregar recursos

5.7.4 SECUENCIA EDITAR RECURSO

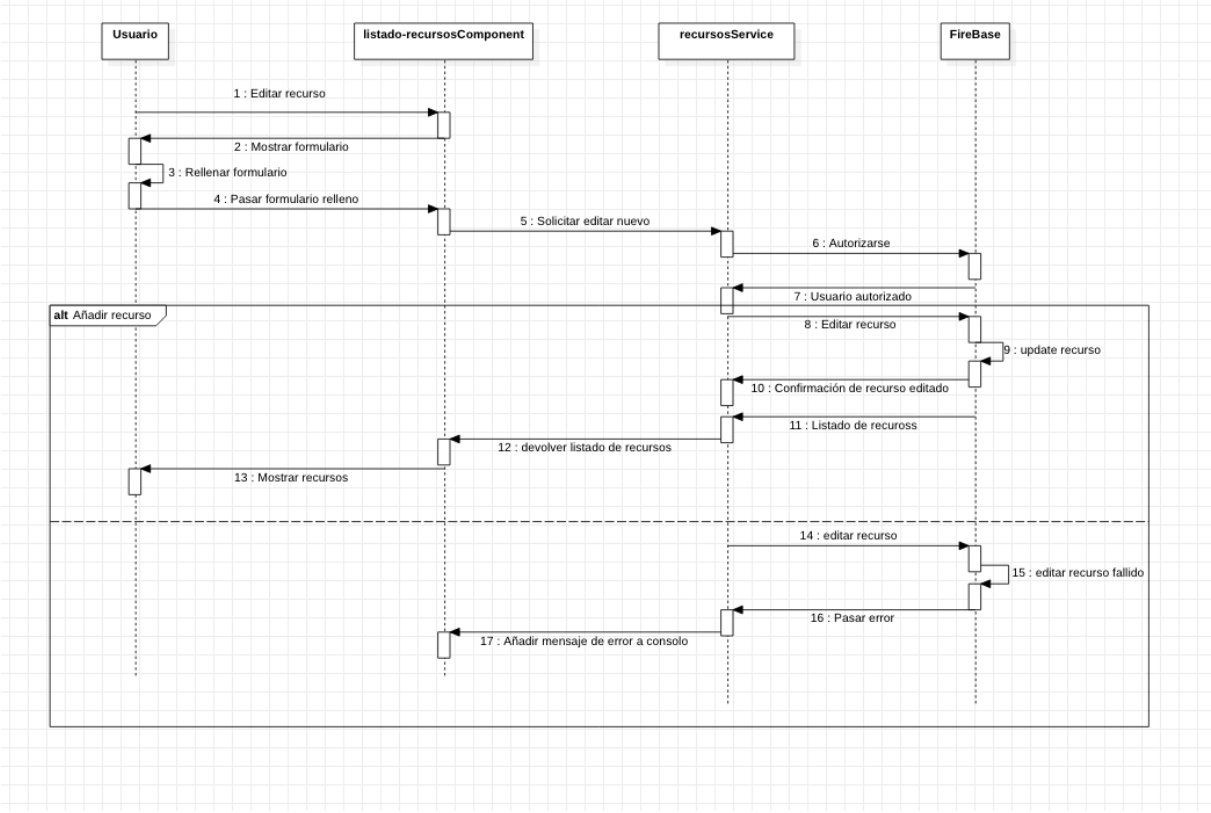


Figura 22: Diagrama de secuencia de editar recursos

5.7.5 SECUENCIA BORRAR RECURSO

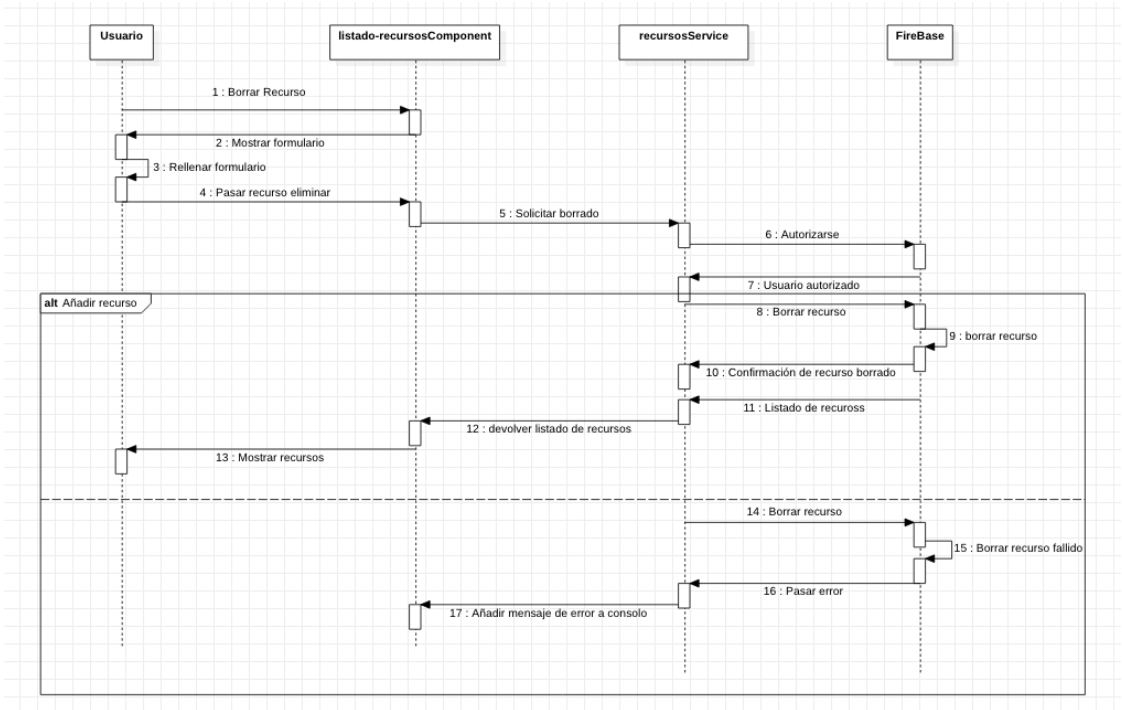


Figura 23: Diagrama de secuencia de editar recursos

6 IMPLEMENTACIÓN

6.1 SOFTWARE UTILIZADO

Tal y como se ha mencionado en los puntos anteriores, este proyecto se ha atacado desde dos aproximaciones diferentes. Es por ello por lo que, en los dos subapartados siguientes, se va a mencionar las herramientas que han sido necesarias para cada una de ellas.

6.1.1 PRIMERA APROXIMACION

En esta primera aproximación, se quiso utilizar una arquitectura empresarial basada en la tecnología JavaEE 8. Durante esta primera aproximación se tuvieron muchos problemas de compatibilidad, sincronización y bugs de los diferentes componentes que se utilizaron para su desarrollo, tanto a nivel de tecnologías como de herramientas de desarrollo. Es por ello, que se utilizaron bastantes herramientas, las cuales son las siguientes:

- Netbeans 8.2, Netbeans 11.0, Netbeans 11.1, Netbeans 11.2 y Netbeans 12.0: Se utilizó la extendida herramienta de desarrollo Netbeans debido a su compatibilidad con Java EE, MySQL y su facilidad para la creación de las clases entidad y los servicios REST.

- MySQL Workbench: Se utilizó la herramienta para sincronizarla con Netbeans y poder utilizar MySQL dentro de Netbeans.

- Draw.io: Software para la creación de diagramas personalizados que dispone de múltiples elementos visuales para la creación de estos.

- GlassFish: es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación.

- GitHub: es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git.

- Microsoft Office 365: Microsoft Office 365 es un conjunto de programas informáticos de la empresa Microsoft de alquiler por un año Microsoft Office (Word, Excel, PowerPoint, Publisher, Access, OneNote, Outlook, Project y SharePoint) para su uso durante un año en vez de pagar el precio completo de la adquisición del producto.

- Suite JetBrains: IDEs para la programación en Java y herramienta para gestión de base de datos.

6.1.2 SEGUNDA ITERACIÓN

En la segunda iteración, se cambió el software utilizado, ya que, debido al cambio de tecnologías para el desarrollo de la solución, se necesitaban herramientas más específicas. En concreto se utilizaron las siguientes:

- Draw.io: Software para la creación de diagramas personalizados, dispone de múltiples elementos visuales para la creación de estos.

- GitHub: es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git.

- Microsoft Office 365: Microsoft Office 365 es un conjunto de programas informáticos de la empresa Microsoft de alquiler por un año Microsoft Office (Word, Excel, PowerPoint, Publisher, Access, OneNote, Outlook, Project, Visio y SharePoint) para su uso durante un año en vez de pagar el precio completo de la adquisición del producto.

- Suite JetBrains: IDEs para la programación de aplicaciones web con el framework de Angular.

- Safari y Google Chrome: Navegadores web que permitieron realizar la configuración necesaria en Firebase, así como la consulta de los resultados de lo programado.

- Tabnine y Github copilot: Se trata de autocompletados de código basado en inteligencia artificial que ha facilitado el desarrollo del proyecto.

6.2 ESQUEMA DEL PROYECTO

Tal y como se ha mencionado en puntos anteriores, se ha realizado dos aproximaciones para el desarrollo del proyecto. Aunque únicamente se ha llegado a completar la solución Angular, creo que es importante mencionar la estructura del proyecto en Java.

6.2.1 ESQUEMA DEL PROYECTO JAVA

Para la aproximación del proyecto utilizando la tecnología de java, se estructuró el proyecto como una aplicación cliente servidor basado en el modelo de 3 capas. Es por ello, que por un lado se tiene todo el lado cliente y por otro el lado servidor.

En el lado del cliente, se agruparon las páginas web por funcionalidad. Por el contrario, el lado servidor, se agrupó el contenido en paquetes, los cuales hacen referencia a cada parte de la estructura de la arquitectura de 3 capas de java.

La estructura del cliente fue la siguiente

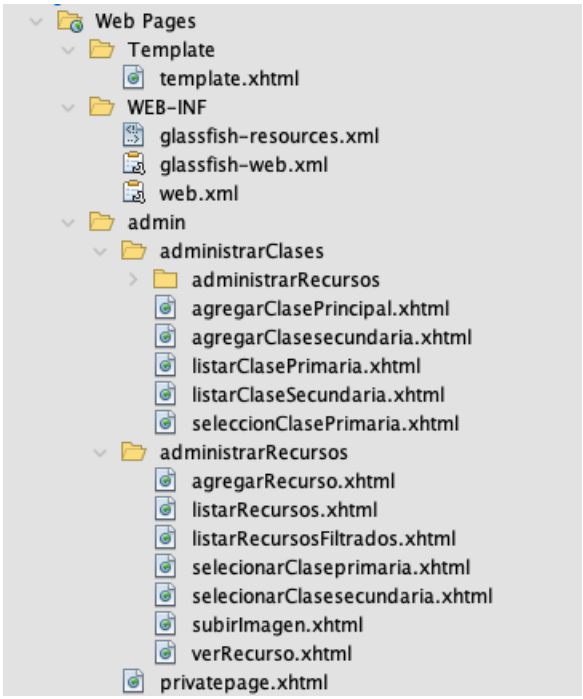


Figura 24: Estructura cliente

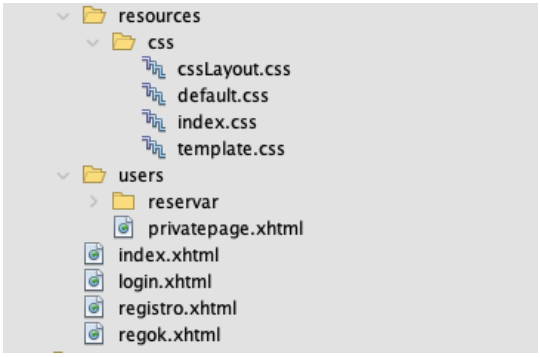


Figura 25: Estructura cliente

Para el lado del cliente se estructuró de la siguiente forma:

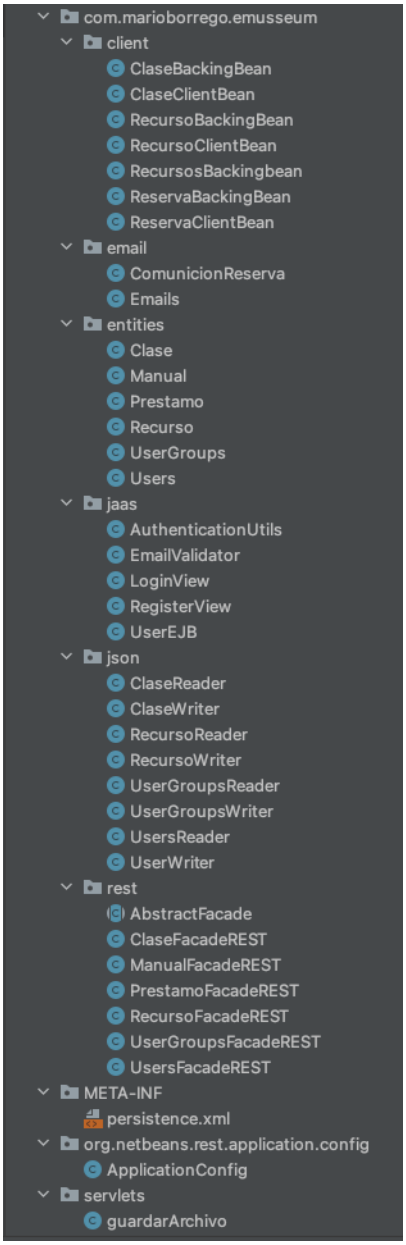


Figura 26: Estructura Servidor

6.2.2 ESQUEMA DEL PROYECTO ANGULAR

Angular propone el desarrollo de aplicaciones basadas en componentes. Es por ello, que se ha estructurado el proyecto en base a este principio. Además, se ha dividido las clases en función a la labor que realiza dentro de la aplicación.

El resultado es el mostrado a continuación:

- `Node_modules:` Contiene todos los módulos que se han utilizado durante el desarrollo

- `Src:` Contiene los archivos que componen la aplicación

- `App:` Contiene los archivos propios de la aplicación
  - `Componentes:` Contiene todos los componentes de la aplicación.
    - `Cabecero`
    - `Editar-recurso`
    - `editarClase`
    - `filtrado-recursos`
    - `listado-clases`
    - `listado-localizaciones`
    - `listado-recursos`
    - `login`
    - `pie-pagina`
    - `visualizar-recurso`
  - `dist:` Contiene los elementos necesarios para el despliegue de la aplicación en un entorno de producción.
  - `Guardianes:` Almacena los archivos de guardianes utilizados para la protección de rutas.
    - `auth.guard.ts:` Guardian de autorización.
  - `Modelo:` Almacena los archivos de los modelos de datos que se utilizan.
    - `clase.model.ts:` Modelo de clase
    - `localizacion.model.ts:` Guarda los datos de localización
    - `recurso.model.ts:` Guarda los datos de recurso
  - `Servicio:` Utiliza los archivos de conexión a Firebase.
    - `clase.service.ts`
    - `localizacion.service.ts`
    - `login.service.ts`
    - `recursos.service.ts`

        - `App.component:` Guarda todos los archivos de inicio de la aplicación. Así como de su composición.
        - `app.module.ts:` Archivo de inicio de la aplicación
        - `app-routing.module.ts:` Archivo para gestionar rutas con los componentes.
    - `Assets:` Contiene los archivos de recurso estáticos.
    - `Enviroment:` Contiene los archivos de los diferentes entornos
        - `environment.prod.ts`
        - `environment.ts`
    - `Index.html:` archivo inicial que renderiza el usuario y en el que se inyectan todos los componentes.

- `Archivos`

6.2 DETALLES SIGNIFICATIVOS DE IMPLEMENTACIÓN

En este apartado se describen los detalles de implementación de la solución desarrollada. Ya que se han realizado dos aproximaciones se ha decidido comenzar los detalles mas significativos de cada aproximación.

6.2.3 DETALLES SIGNIFICATIVOS EN JAVA

La aplicación en Java EE, se concibió como una plataforma web que fuera modular, es decir que en cualquier momento se pudiera añadir nuevas funcionalidades o sustituir las ya existentes. Para ello, se realizó un desacople de cada una de las partes que componen la plataforma.

Para poder dar una visión global de la plataforma, se expondrán los detalles significativos de cada una de las partes de la misma.

6.2.3.1 DETALLES SIGNIFICATIVOS DE LA BASE DE DATOS

Primeramente, se diseñó el modelo de datos que la aplicación iba a utilizar, para luego desarrollar el código de creación de la base de datos

```sql
DROP DATABASE IF EXISTS TFG;
CREATE DATABASE TFG;
USE TFG;

CREATE TABLE RECURSO
(
    NOMBRE VARCHAR(255) NOT NULL UNIQUE,
    DESCRIPCION TEXT NOT NULL,
    LOCALIZACION VARCHAR(255),
    PROPIEDADES_ESPECIFICAS TEXT,
    IMAGEN VARCHAR(255),
    DISPONIBLE ENUM('DISPONIBLE','NO DISPONIBLE'),
    NOMBRE_CLASE VARCHAR(255) NOT NULL,
    PRIMARY KEY (NOMBRE)
);

CREATE TABLE MANUAL
(
    ID_MANUAL INT NOT NULL UNIQUE AUTO_INCREMENT,
    URL VARCHAR(255) NOT NULL UNIQUE,
    NOMBRE_RECURSO VARCHAR(255) NOT NULL,
    PRIMARY KEY (ID_MANUAL)
);

CREATE TABLE CLASE
(
    NOMBRE VARCHAR(255) NOT NULL UNIQUE,
    DESCRIPCION TEXT NOT NULL,
    PROPIEDADES TEXT NOT NULL,
    NOMBRE_SUPERCHASE VARCHAR(255),
    PRIMARY KEY (NOMBRE)
);

CREATE TABLE PRESTAMO
(
    ID_RESERVA INT NOT NULL UNIQUE AUTO_INCREMENT,
    FECHA_PRESTAMO VARCHAR(10),
    DEVUELTO BOOLEAN,
    NOMBRE_RECURSO VARCHAR(255),
    EMAIL_USUARIO VARCHAR(255),
    PRIMARY KEY (ID_RESERVA)

);

CREATE TABLE users
(
    email VARCHAR(255) NOT NULL,
    password VARCHAR(64) NOT NULL,
    name VARCHAR(64) NOT NULL,
    PRIMARY KEY (email)
);
CREATE TABLE user_groups
(
    email VARCHAR(255) NOT NULL,
    groupname VARCHAR(32) NOT NULL,
    PRIMARY KEY (email)
);

ALTER TABLE MANUAL ADD FOREIGN KEY (NOMBRE_RECURSO) REFERENCES RECURSO(NOMBRE);
ALTER TABLE RECURSO ADD FOREIGN KEY (NOMBRE_CLASE) REFERENCES CLASE(NOMBRE);
ALTER TABLE PRESTAMO ADD FOREIGN KEY (EMAIL_USUARIO) REFERENCES users(email);
```

Figura 27: Código de creación de la base de datos

6.2.3.2 DETALLES SIGNIFICATIVOS EN JAVA

Una vez desarrollada la base de datos, se continuó con la creación del servidor Java. Primeramente, se creó las clases entidad, las cuales crean una relación entre las bases de datos y el servidor, mapeando cada uno de los posibles objetos para luego poder hacer las operaciones CRUD.

Cabe destacar que, para facilitar las operaciones de consulta, se desarrollaron una serie de consultas predefinidas con el fin de facilitar el entendimiento del código y ser mas

transparentes en las operaciones. Esto ha permitido que en cualquier momento que se haga cambios en la base de datos solamente haya que modificar un fichero y no múltiples.

```java
package com.marioborrego.emusseum.entities;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

@Entity
@Table(name = "CLASE")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Clase.findAll", query = "SELECT c FROM Clase c"),
    @NamedQuery(name = "Clase.findByNombre", query = "SELECT c FROM Clase c WHERE c.nombre = :nombre"),
    @NamedQuery(name = "Clase.findByNombreSuperchase", query = "SELECT c FROM Clase c WHERE c.nombreSuperchase = :nombreSuperchase"),
    @NamedQuery(name = "Clase.findClasePrincipal", query ="SELECT c FROM Clase c WHERE c.nombreSuperchase=''"),
    @NamedQuery(name = "Clase.findClaseSecundaria", query = "SELECT c FROM Clase c WHERE c.nombreSuperchase != ''")})
public class Clase implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 255)
    @Column(name = "NOMBRE")
    private String nombre;
    @Basic(optional = false)
    @NotNull
    @Lob
    @Size(min = 1, max = 65535)
    @Column(name = "DESCRIPCION")
    private String descripcion;
    @Basic(optional = false)
    @NotNull
    @Lob
    @Size(min = 1, max = 65535)
    @Column(name = "PROPIEDADES")
    private String propiedades;
    @Size(max = 255)
    @Column(name = "NOMBRE_SUPERCHASE")
    private String nombreSuperchase;

    public Clase() {
    }

    public Clase(String nombre) { this.nombre = nombre; }

    public Clase(String nombre, String descripcion, String propiedades) {
        this.nombre = nombre;
        this.descripcion = descripcion;
        this.propiedades = propiedades;
    }

    public String getNombre() { return nombre; }
```

Figura 28: Código de clases entidad

Una vez definidos los datos que componen a cada clase entidad (los cuales tienen su semejante en la base de datos), se implementaron los métodos que iba a tener cada entidad. Es decir las operaciones que se pueden realizar con ellos.

```java
public Clase() {
}

public Clase(String nombre) { this.nombre = nombre; }

public Clase(String nombre, String descripcion, String propiedades) {
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.propiedades = propiedades;
}

public String getNombre() { return nombre; }

public void setNombre(String nombre) { this.nombre = nombre; }

public String getDescripcion() { return descripcion; }

public void setDescripcion(String descripcion) { this.descripcion = descripcion; }

public String getPropiedades() { return propiedades; }

public void setPropiedades(String propiedades) { this.propiedades = propiedades; }

public String getNombreSuperchase() { return nombreSuperchase; }

public void setNombreSuperchase(String nombreSuperchase) { this.nombreSuperchase = nombreSuperchase; }

@Override
public int hashCode() {
    int hash = 0;
    hash += (nombre != null ? nombre.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof Clase)) {
        return false;
    }
    Clase other = (Clase) object;
    if ((this.nombre == null && other.nombre != null) || (this.nombre != null && !this.nombre.equals(other.nombre))) {
        return false;
    }
    return true;
}

@Override
public String toString() { return "com.marioborrego.emusseum.entities.Clase[ nombre=" + nombre + " ]"; }
}
```

Figura 29: Código de clases entidad

Una vez completado el desarrollo de las clases entidad, se continuó con la implementación de los servicios rest full. Estos permiten que se puedan crear una API para poder abstraer la lógica de negocio de la capa de visualización.

```java
package com.marioborrego.emusseum.rest;

import com.marioborrego.emusseum.entities.Clase;

import javax.ejb.Stateless;
import javax.inject.Named;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import java.util.List;

@Named
@Stateless
@Path("com.marioborrego.emusseum.entities.clase")
public class ClaseFacadeREST extends AbstractFacade<Clase> {

    @PersistenceContext(unitName = "com.marioborrego_eMusseum_war_1PU")
    private EntityManager em;

    public ClaseFacadeREST() { super(Clase.class); }

    @POST
    @Override
    @Consumes({MediaType.APPLICATION_JSON})
    public void create(Clase entity) { super.create(entity); }

    @PUT
    @Path("{id}")
    @Consumes({MediaType.APPLICATION_JSON})
    public void edit(@PathParam("id") String id, Clase entity) { super.edit(entity); }

    @DELETE
    @Path("{id}")
    public void remove(@PathParam("id") String id) { super.remove(super.find(id)); }

    @GET
    @Path("{id}")
    @Produces({MediaType.APPLICATION_JSON})
    public Clase find(@PathParam("id") String id) { return super.find(id); }

    @GET
    @Override
    @Produces({MediaType.APPLICATION_JSON})
    public List<Clase> findAll() { return super.findAll(); }

    @GET
    @Path("{from}/{to}")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Clase> findRange(@PathParam("from") Integer from, @PathParam("to") Integer to) {
        return super.findRange(new int[]{from, to});
    }

    @GET
    @Path("ClasePrimaria")
```

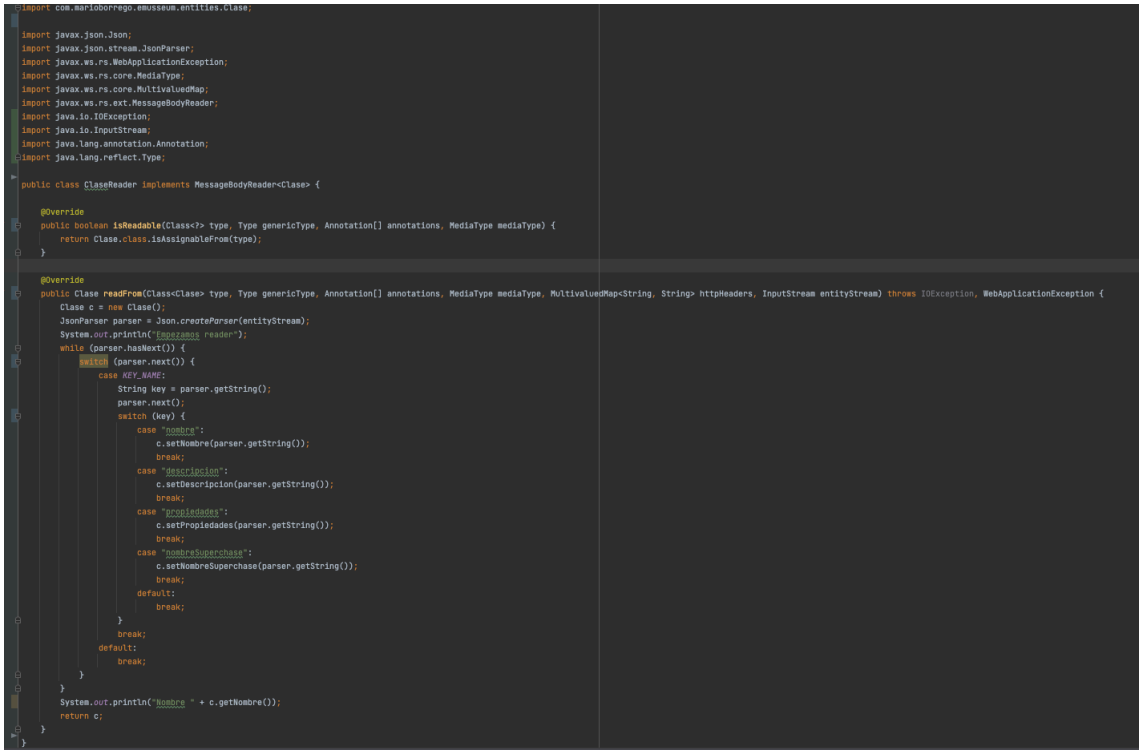
Figura 30: Código de servicios REST

En este caso se utilizó una herramienta que venia en Netbeans para la creación de los principales métodos CRUD del servicio rest, luego se fueron ampliando según fuera necesario.

Figura 31: Código de servicios REST

6.2.3.3 CLIENT

Para el cliente se utilizó el estandar JSP (*JavaServer Pages*), el cual permite la creación de páginas web utilizando Java y xhtml. El cliente se conecta al servidor mediante los servicios rest que se han creado anteriormente, para ello, es necesario que el cliente pueda parsear los datos. Esta tarea se encarga las clases reader que se han desarrollado.



```java
import com.marioborrego.emusseum.entities.Clase;

import javax.json.Json;
import javax.json.stream.JsonParser;
import javax.ws.rs.WebApplicationException;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.MultivaluedMap;
import javax.ws.rs.ext.MessageBodyReader;
import java.io.IOException;
import java.io.InputStream;
import java.lang.annotation.Annotation;
import java.lang.reflect.Type;

public class ClaseReader implements MessageBodyReader<Clase> {

    @Override
    public boolean isReadable(Class<?> type, Type genericType, Annotation[] annotations, MediaType mediaType) {
        return Clase.class.isAssignableFrom(type);
    }

    @Override
    public Clase readFrom(Class<Clase> type, Type genericType, Annotation[] annotations, MediaType mediaType, MultivaluedMap<String, String> httpHeaders, InputStream entityStream) throws IOException, WebApplicationException {
        Clase c = new Clase();
        JsonParser parser = Json.createParser(entityStream);
        System.out.println("Empezamos reader");
        while (parser.hasNext()) {
            switch (parser.next()) {
                case KEY_NAME:
                    String key = parser.getString();
                    parser.next();
                    switch (key) {
                        case "nombre":
                            c.setNombre(parser.getString());
                            break;
                        case "descripcion":
                            c.setDescripcion(parser.getString());
                            break;
                        case "propiedades":
                            c.setPropiedades(parser.getString());
                            break;
                        case "nombreSuperchase":
                            c.setNombreSuperchase(parser.getString());
                            break;
                        default:
                            break;
                    }
                    break;
                default:
                    break;
            }
        }
        System.out.println("Nombre " + c.getNombre());
        return c;
    }
}
```

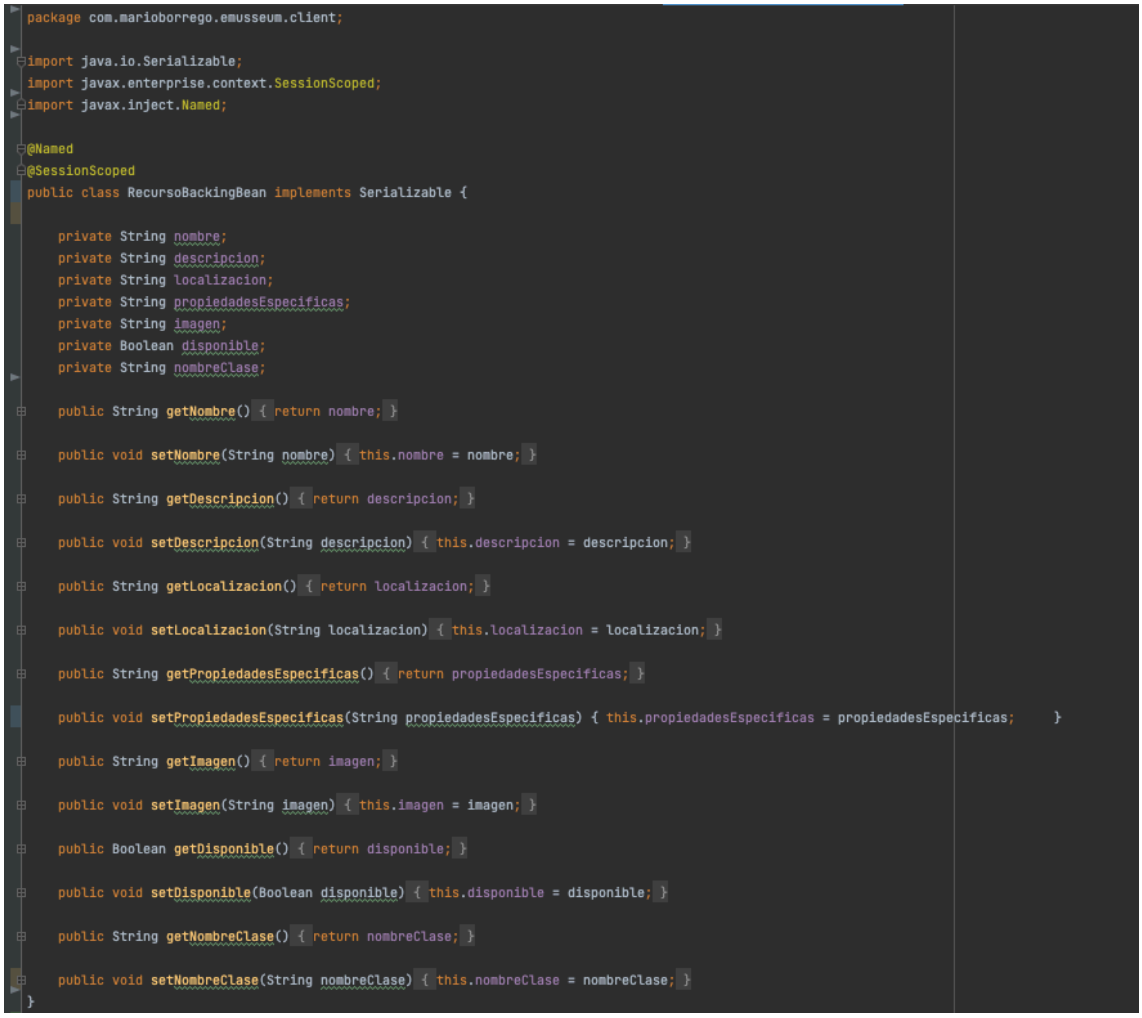
Figura 32: Código de Reader

Para poder crear los JSON que se van a mandar hacia el servidor fue necesario desarrollar un writer, el cual se encarga de parsear los objetos java objetos JSON con cada uno de los valores actualizados.

```java
package com.marioborrego.emusseum.json;

import com.marioborrego.emusseum.entities.Clase;

import javax.json.Json;
import javax.json.stream.JsonGenerator;
import javax.ws.rs.WebApplicationException;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.MultivaluedMap;
import javax.ws.rs.ext.MessageBodyWriter;
import java.io.IOException;
import java.io.OutputStream;
import java.lang.annotation.Annotation;
import java.lang.reflect.Type;

public class ClaseWriter implements MessageBodyWriter<Clase> {

    @Override
    public boolean isWriteable(Class<?> type, Type genericType, Annotation[] annotations, MediaType mediaType) {
        return Clase.class.isAssignableFrom(type);
    }

    @Override
    public long getSize(Clase t, Class<?> type, Type genericType, Annotation[] annotations, MediaType mediaType) {
        return -1;
    }

    @Override
    public void writeTo(Clase t, Class<?> type, Type genericType, Annotation[] annotations, MediaType mediaType, MultivaluedMap<String, Object> httpHeaders, OutputStream entityStream) throws IOException, WebApplicationException {
        System.out.println("Empezamos Writer");

        JsonGenerator gen = Json.createGenerator(entityStream);
        gen.writeStartObject()
                .write( s: "nombre", t.getNombre())
                .write( s: "descripcion", t.getDescripcion())
                .write( s: "propiedades", t.getPropiedades())
                .write( s: "nombreSuperchase", t.getNombreSuperchase())
                .writeEnd();
        gen.flush();

        System.out.println("Terminamos Writer");

    }
}
```

Figura 33: Código de Writer

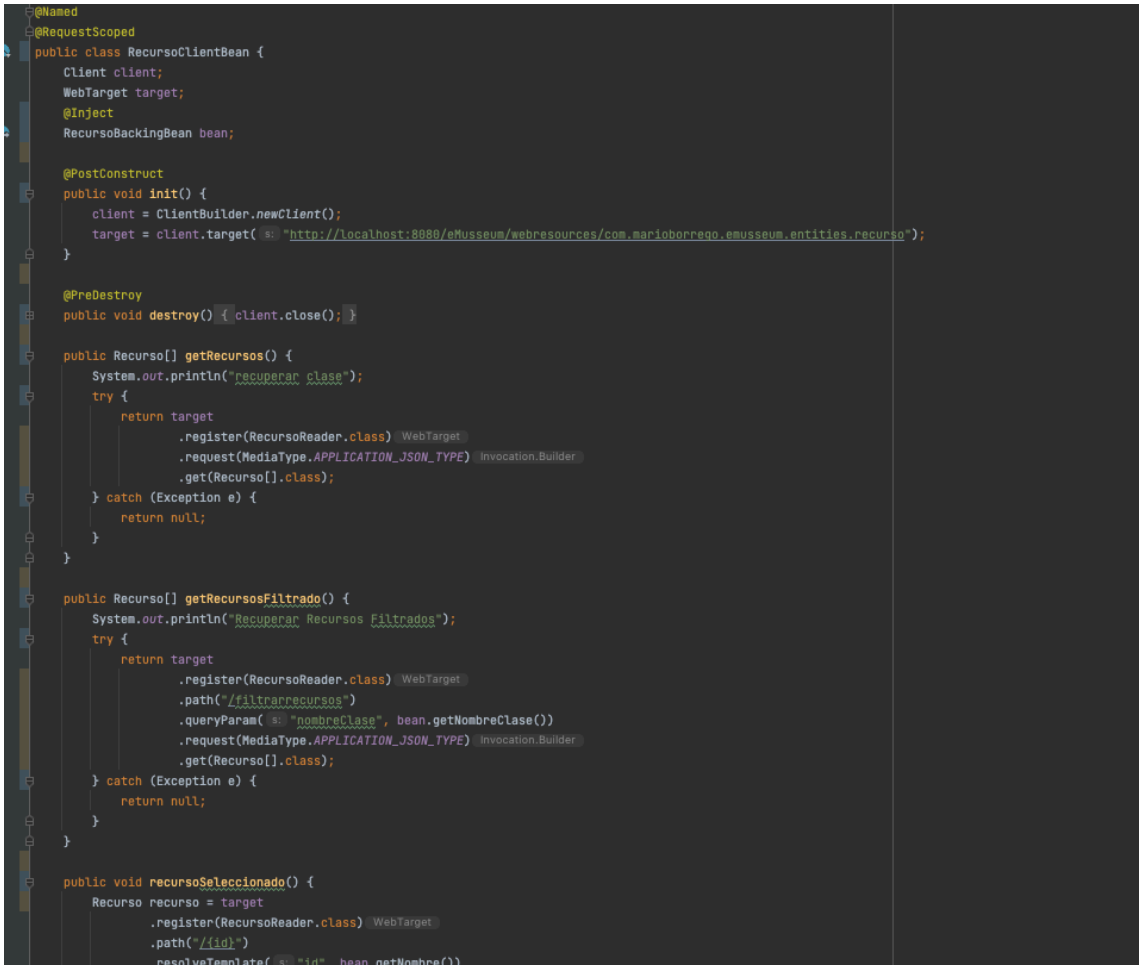
Para poder gestionar cada una de las peticiones y de las operaciones que se ejecutan desde el cliente se crearon los `clientbean` y los `backinbean.`



```java
package com.marioborrego.emusseum.client;

import java.io.Serializable;
import javax.enterprise.context.SessionScoped;
import javax.inject.Named;

@Named
@SessionScoped
public class RecursoBackingBean implements Serializable {

    private String nombre;
    private String descripcion;
    private String localizacion;
    private String propiedadesEspecificas;
    private String imagen;
    private Boolean disponible;
    private String nombreClase;

    public String getNombre() { return nombre; }

    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getDescripcion() { return descripcion; }

    public void setDescripcion(String descripcion) { this.descripcion = descripcion; }

    public String getLocalizacion() { return localizacion; }

    public void setLocalizacion(String localizacion) { this.localizacion = localizacion; }

    public String getPropiedadesEspecificas() { return propiedadesEspecificas; }

    public void setPropiedadesEspecificas(String propiedadesEspecificas) { this.propiedadesEspecificas = propiedadesEspecificas;     }

    public String getImagen() { return imagen; }

    public void setImagen(String imagen) { this.imagen = imagen; }

    public Boolean getDisponible() { return disponible; }

    public void setDisponible(Boolean disponible) { this.disponible = disponible; }

    public String getNombreClase() { return nombreClase; }

    public void setNombreClase(String nombreClase) { this.nombreClase = nombreClase; }
}
```

Figura 34: Código de BackingBean

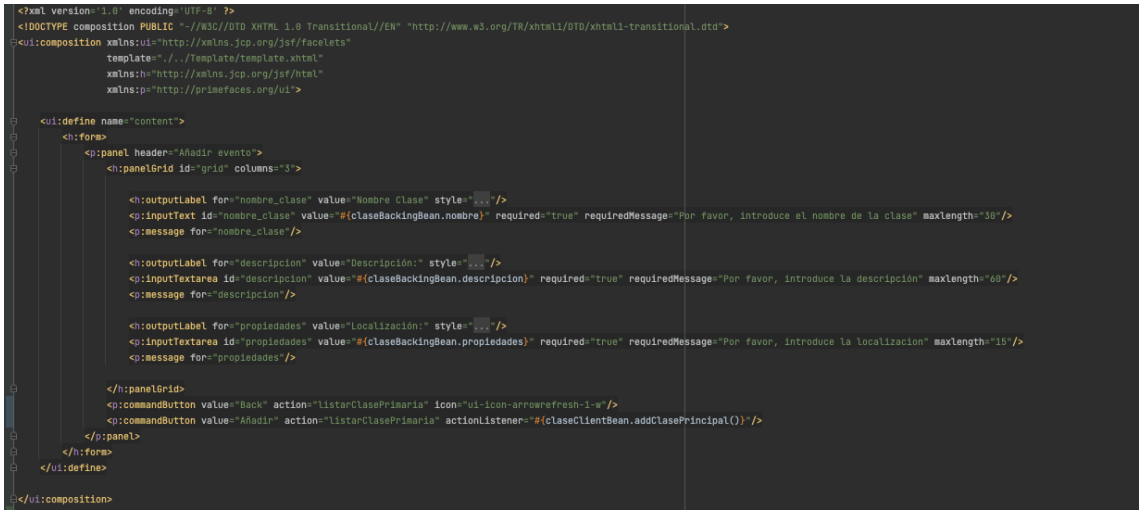
Esto permite establecer un modelo de datos y las operaciones que se ejecutan en el.



```java
@Named
@RequestScoped
public class RecursoClientBean {
    Client client;
    WebTarget target;
    @Inject
    RecursoBackingBean bean;

    @PostConstruct
    public void init() {
        client = ClientBuilder.newClient();
        target = client.target( s: "http://localhost:8080/eMusseum/webresources/com.marioborrego.emusseum.entities.recurso");
    }

    @PreDestroy
    public void destroy() { client.close(); }

    public Recurso[] getRecursos() {
        System.out.println("recuperar clase");
        try {
            return target
                    .register(RecursoReader.class) WebTarget
                    .request(MediaType.APPLICATION_JSON_TYPE) Invocation.Builder
                    .get(Recurso[].class);
        } catch (Exception e) {
            return null;
        }
    }

    public Recurso[] getRecursosFiltrado() {
        System.out.println("Recuperar Recursos Filtrados");
        try {
            return target
                    .register(RecursoReader.class) WebTarget
                    .path("/filtrarrecursos")
                    .queryParam( s: "nombreClase", bean.getNombreClase())
                    .request(MediaType.APPLICATION_JSON_TYPE) Invocation.Builder
                    .get(Recurso[].class);
        } catch (Exception e) {
            return null;
        }
    }

    public void recursoSeleccionado() {
        Recurso recurso = target
                .register(RecursoReader.class) WebTarget
                .path("/{id}")
                .resolveTemplate( s: "id", bean.getNombre())
```

Figura 35: Código de BackingBean

Por último, se crearon los XHTML, el cual establece la GUI que se mostrará al usuario final. Para unir la clase al XHTML, se utilizó el expresion lenguage.



```xml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
                template="./../Template/template.xhtml"
                xmlns:h="http://xmlns.jcp.org/jsf/html"
                xmlns:p="http://primefaces.org/ui">

    <ui:define name="content">
        <h:form>
            <p:panel header="Añadir evento">
                <h:panelGrid id="grid" columns="3">

                    <h:outputLabel for="nombre_clase" value="Nombre Clase" style="..."/>
                    <p:inputText id="nombre_clase" value="#{claseBackingBean.nombre}" required="true" requiredMessage="Por favor, introduce el nombre de la clase" maxlength="30"/>
                    <p:message for="nombre_clase"/>

                    <h:outputLabel for="descripcion" value="Descripción:" style="..."/>
                    <p:inputTextarea id="descripcion" value="#{claseBackingBean.descripcion}" required="true" requiredMessage="Por favor, introduce la descripción" maxlength="60"/>
                    <p:message for="descripcion"/>

                    <h:outputLabel for="propiedades" value="Localización:" style="..."/>
                    <p:inputTextarea id="propiedades" value="#{claseBackingBean.propiedades}" required="true" requiredMessage="Por favor, introduce la localizacion" maxlength="15"/>
                    <p:message for="propiedades"/>

                </h:panelGrid>
                <p:commandButton value="Back" action="listarClasePrimaria" icon="ui-icon-arrowrefresh-1-w"/>
                <p:commandButton value="Añadir" action="listarClasePrimaria" actionListener="#{claseClientBean.addClasePrincipal()}"/>
            </p:panel>
        </h:form>
    </ui:define>

</ui:composition>
```

Figura 36: Código de ClientBean

6.2.4 DETALLES SIGNIFICATIVOS EN ANGULAR

La aplicación Angular se concibió desde el desarrollo de una plataforma que utilizara todas las funcionalidades que nos ofrece la plataforma de Firebase. Para poder entender mejor el desarrollo de la plataforma se irán nombrando los servicios usados que nos ofrece la plataforma de google y después los apartados mas importantes del código.

6.2.4.1 GOOGLE FIREBASE

La plataforma de Google Firebase se ha convertido en la piedra angular del proyecto, ya que toda la plataforma gira en el uso de las herramientas que nos ofrece esta. Es por ello, que a lo largo de las siguientes subsecciones se mencionarán los servicios que se han usado y las características que nos ofrece los mismos.

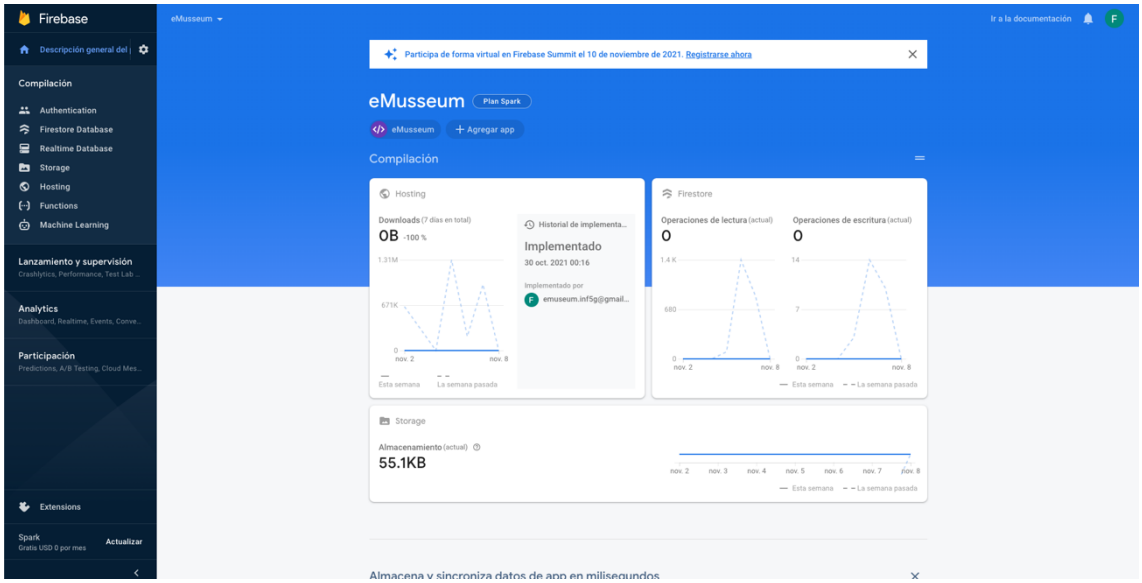


Figura 37: Pagina de control Firebase

6.2.4.1.1 REALTIME DATABASE

Firebase Realtime Database es una base de datos NoSQL alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan con cada cliente conectado en tiempo real. Cuando usa iOS, Android y JavaScript SDK para crear aplicaciones multiplataforma, todos los clientes comparten una instancia de base de datos en tiempo real y reciben automáticamente actualizaciones de los datos más recientes, lo que permite crear aplicaciones que compartan datos entre si.

Entre todas las características que ofrece la base de datos, las mas importantes son las siguientes:

- ***En tiempo real:** Firebase Real-time Database usa sincronización de datos en lugar de una solicitud HTTP típica (cada vez que los datos cambian, el dispositivo conectado recibirá la actualización en milisegundos). Proporciona una experiencia colaborativa e inversiva sin pensar en el código de red.*

- ***Sin conexión:** la aplicación Firebase sigue respondiendo, incluso sin conexión, porque el SDK de Firebase Realtime Database guarda tus datos en el disco.*

Cuando se restablece la conexión, el dispositivo cliente recibe los cambios que faltan y los sincroniza con el estado actual del servidor.

- ***Acceso desde dispositivos cliente:*** *se puede acceder a la base de datos en tiempo real de Firebase directamente desde dispositivos móviles o navegadores web; no se requiere un servidor de aplicaciones. La seguridad y la validación de los datos se pueden lograr a través de reglas basadas en expresiones de Firebase Realtime Database Security Rules que se ejecutan cuando se leen o escriben datos.*

- ***Escala en varias bases de datos:*** *con el plan de precios de Firebase Realtime Database y Blaze, puedes satisfacer las necesidades de datos de tu aplicación a gran escala: puedes dividir la información en varias instancias de base de datos en el mismo proyecto de Firebase.*

- ***Autenticación de Firebase:*** *Optimice el proceso de autenticación en el proyecto. Podrá autenticar usuarios en varias instancias de la base de datos. Controle el acceso a la información en cada base de datos.*

Se decidió utilizar la presente base de datos por la simplicidad de uso que ofrece. Al estar integrada en el SDK de Firebase, la complejidad de las operaciones se redujo considerablemente.[1]

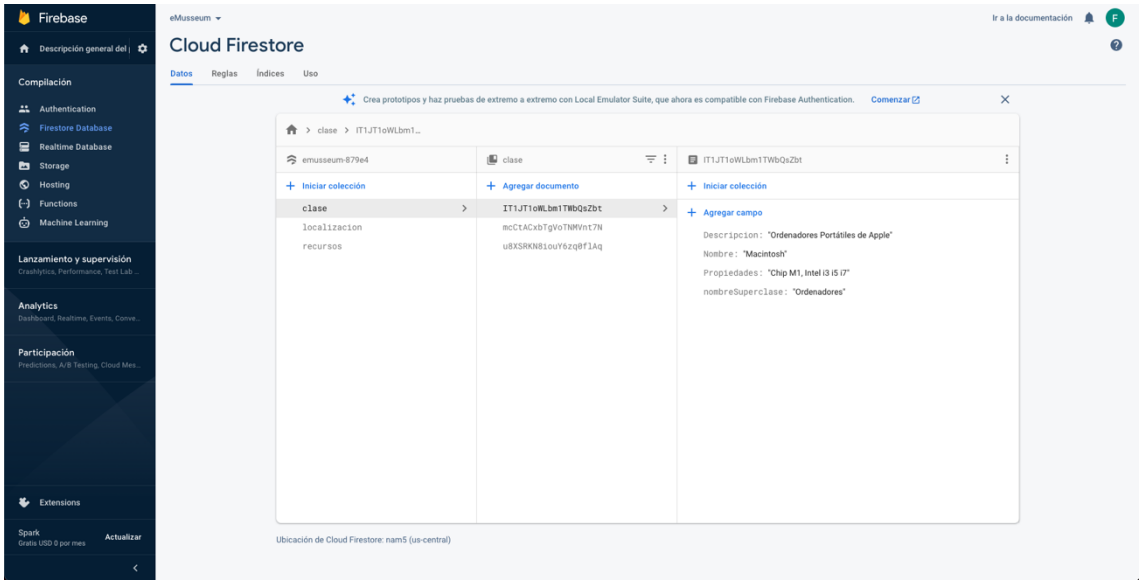


Figura 38: Pagina de control Firebase- Firestore Database

6.2.4.1.2 FIREBASE AUTHENTICATION

Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en la plataforma. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook y Twitter, y mucho más.

[1] Google (2022): *Google Firebase Realtime Database. Consultado en 26/02/2022* https://firebase.google.com/docs/auth/web/firebaseui?hl=es-419

Firebase Authentication se integra estrechamente con otros servicios de Firebase y aprovecha estándares de la industria como Oauth 2.0 y OpenID Connect, por lo que se puede integrar con facilidad en tu backend personalizado.

La mejor característica de este sistema es la de solución de autenticación directa, la cual permite controlar los flujos de IU para los usuarios que acceden con direcciones de correo electrónico y contraseñas, números de teléfono y con proveedores de identidad federada populares, que incluyen el Acceso con Google y el Acceso con Facebook.

El componente de FirebaseUI Auth implementa prácticas recomendadas para la autenticación en sitios web y dispositivos móviles, lo que puede maximizar la conversión de acceso y registro hacia la plataforma. También maneja casos extremos, como recuperación y vinculación de cuentas, que pueden tener repercusiones en la seguridad y ser propensos a generar errores cuando se tratan de manejar correctamente.

FirebaseUI puede personalizarse fácilmente para adaptarse al resto del estilo visual de tu app. Además, es de código abierto, por lo que no tendrás limitaciones para lograr la experiencia de usuario que deseas.

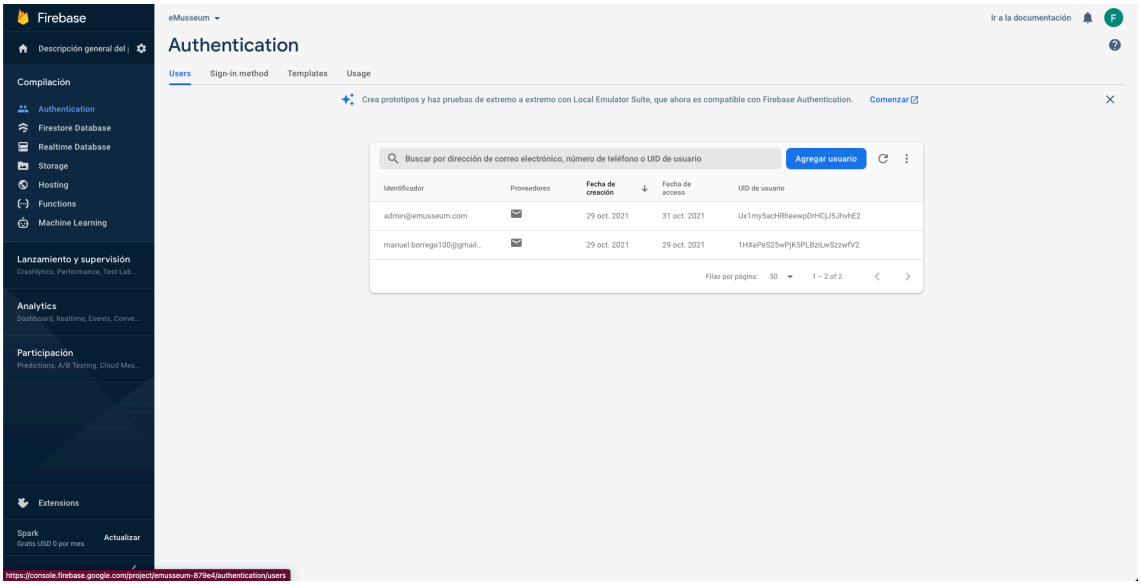


Figura 39: Pagina de control Firebase – Authentication

Además, cabe destacar que Firebase ofrece la autentificación de las siguientes formas:

- *Autenticación basada en correo electrónico y contraseña*
- *Integración con proveedores de identidad federada: Autentica usuarios mediante la integración con proveedores de identidad federada. El SDK de Firebase Authentication proporciona métodos que permiten a los usuarios acceder con sus cuentas de Google, Facebook, Twitter y GitHub.*
- *Autenticación con número de teléfono*
- *Integración de sistemas de autenticación personalizados*

- *Autenticación anónima*[2]

6.2.4.1.3 CLOUD STORAGE

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido para el escalamiento de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos de para tus plataformas de Firebase, sin importar la calidad de la red.

Alguna de las características mas importantes de este servicio son las siguientes:

- ***Operaciones robustas:** Los SDK de Firebase para Cloud Storage realizan las operaciones de carga y descarga sin importar la calidad de la red. Las cargas y descargas son robustas, lo que significa que se reinician en el punto en el que se interrumpieron para así ahorrar tiempo y ancho de banda a los usuarios.*

- ***Seguridad sólida:** Los SDK de Firebase para Cloud Storage se integran con Firebase Authentication a fin de brindar autenticación intuitiva y sencilla para los programadores. Puedes usar nuestro modelo de seguridad declarativa para permitir el acceso según el nombre de archivo, el tamaño, el tipo de contenido y otros metadatos.*

- ***Gran escalabilidad:** Cloud Storage se diseñó con el fin de escalar a exabytes si tu app se vuelve viral. Puedes pasar de la fase de prototipo a la de producción con facilidad mediante la misma infraestructura que respalda a Spotify y Google Fotos.*[3]

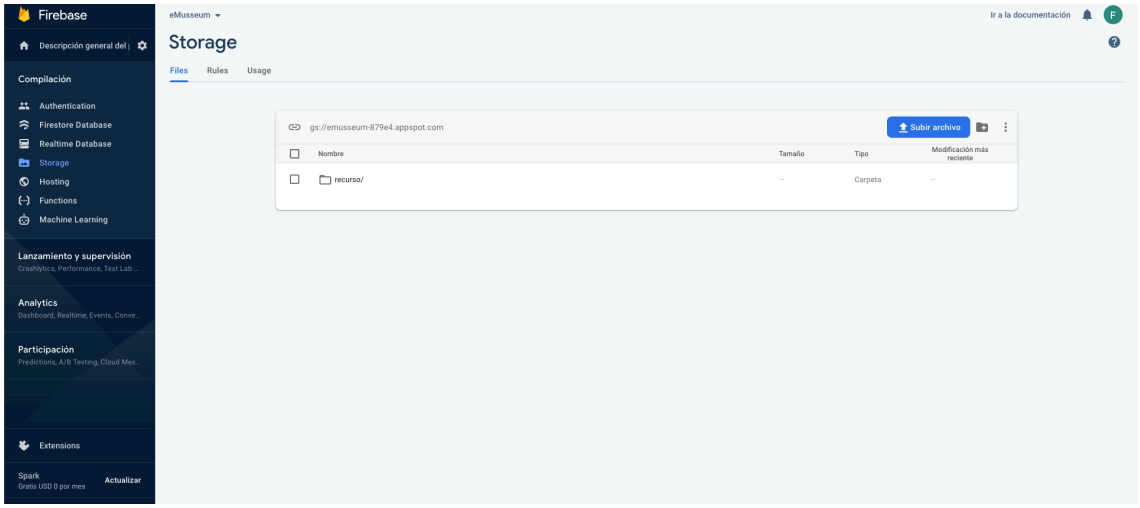


Figura 40: Pagina de control Firebase – Storage

6.2.4.1.4 FIREBASE HOSTING

[2] Google. (2022). *Firebase Authentication. Consultado en 26/02/2022* https://firebase.google.com/products/auth?gclsrc=aw.ds&gclid=EAIaIQobChMIr8uWvZKd9gIV1OJ3Ch04PAzOEAAYASAAEgKMR_D_BwE

[33] Google. (2022). *Cloud Storage para Firebase Consultado en 26/02/2022* https://firebase.google.com/docs/storage

Firebase Hosting es un servicio de hosting de contenido web con nivel de producción orientado a desarrolladores. Con un solo comando, puedes implementar aplicaciones web y entregar contenido dinámico y estático en una CDN (red de distribución de contenidos) global rápidamente. También puedes sincronizar Firebase Hosting con Cloud Functions o Cloud Run para compilar y alojar microservicios en Firebase.

Alguna de las características mas importantes de este servicio son las siguientes:

- ***Entregar contenido mediante una conexión segura:** La Web moderna es segura. Firebase Hosting incluye SSL sin necesidad de configuración para que el contenido se entregue siempre de forma segura.*

- ***Alojar contenido dinámico y estático, además de microservicios:** Firebase Hosting admite todo tipo de contenido para hosting, desde los archivos CSS y HTML hasta las API o los microservicios de Express.js.*

- ***Publicar contenido con rapidez:** Cada archivo que subas se almacena en caché en discos SSD ubicados en el perímetro de la CDN en todo el mundo y se entrega en formato gzip o Brotli.*

- ***Emular y compartir tus cambios antes de publicarlos:** Observa y prueba tus cambios en una URL alojada localmente y después interactúa con un backend emulado.*

- ***Implementar versiones nuevas con un comando**: Con Firebase CLI, puedes poner tu app en funcionamiento en cuestión de segundos. Las herramientas de línea de comandos permiten agregar fácilmente destinos de implementación en el proceso de compilación.[4]*

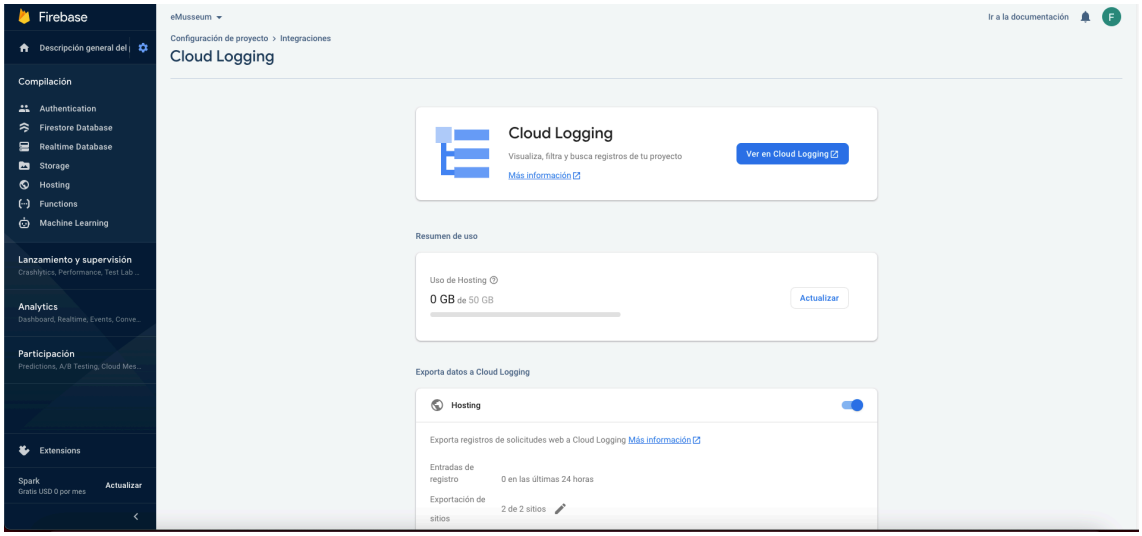


Figura 41: Pagina de control Firebase – Hosting

[4] Google. (2022). *Firebase Hosting. Consultado en 26/02/2022 https://firebase.google.com/products/hosting?gclsrc=aw.ds&gclid=EAIaIQobChMI46ff9pKd9gIVReh3Ch040A-jEAAYASAAEgI4__D_BwE*

6.3.2.1 PLATAFORMA ANGULAR

La plataforma se concibió desde una aplicación SPA (Single Page Application), en la cual se iban cargando dinámicamente los componentes dentro de la misma aplicación. Para ello, se configuró en primer lugar el app-routing.module

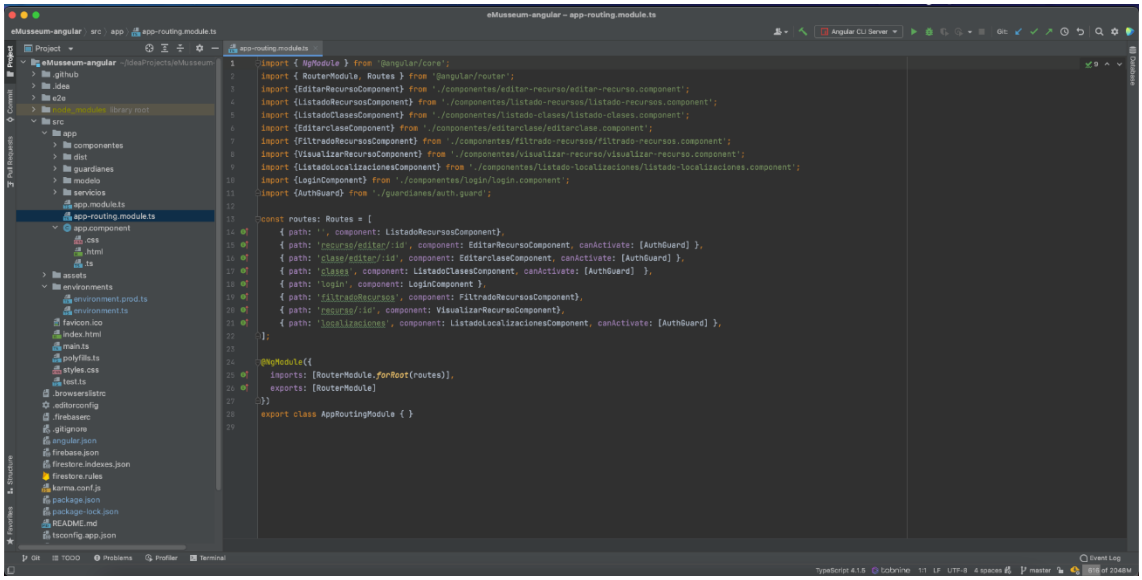


Figura 42: Configuración del app-routing.module

En este fichero, se configuran todas las rutas y los componentes que se cargan en función de la ruta que se acceda.

A continuación, se definió el entorno de desarrollo de la plataforma. Este entono se configura teniendo en cuneta la conexión al proyecto de Firebase eMusseum. Gracias al cual se puede acceder a todos los servicios de Firebase y asociar la aplicación con el proyecto.

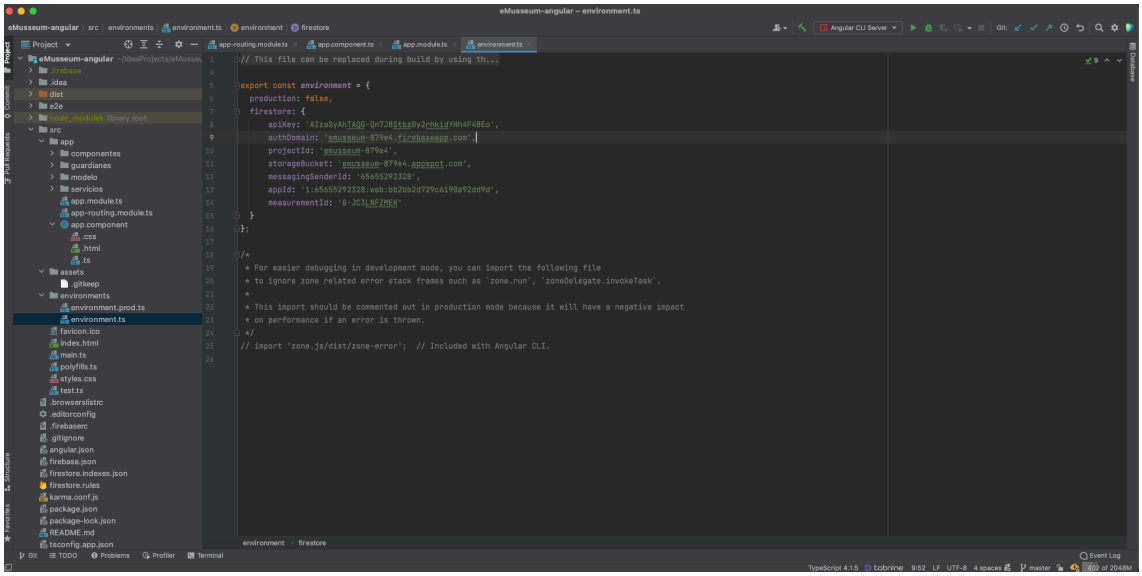


Figura 43: Configuración del entorno de conexión a Firebase.

Una vez establecido el entorno de producción de la aplicación, se prosiguió con la estructura de la SPA. Para ello, se definió en el app.component.html la forma de carga de página, estando siempre cargado los componentes de cabecera y footer y siendo dinámicos el contenido. Para poder cargar dinámicamente el contenido, se utilizó el componente <router-outlet>, el cual ha permitido que dependiendo de la ruta a la que se acceda se cargue un componente u otro.

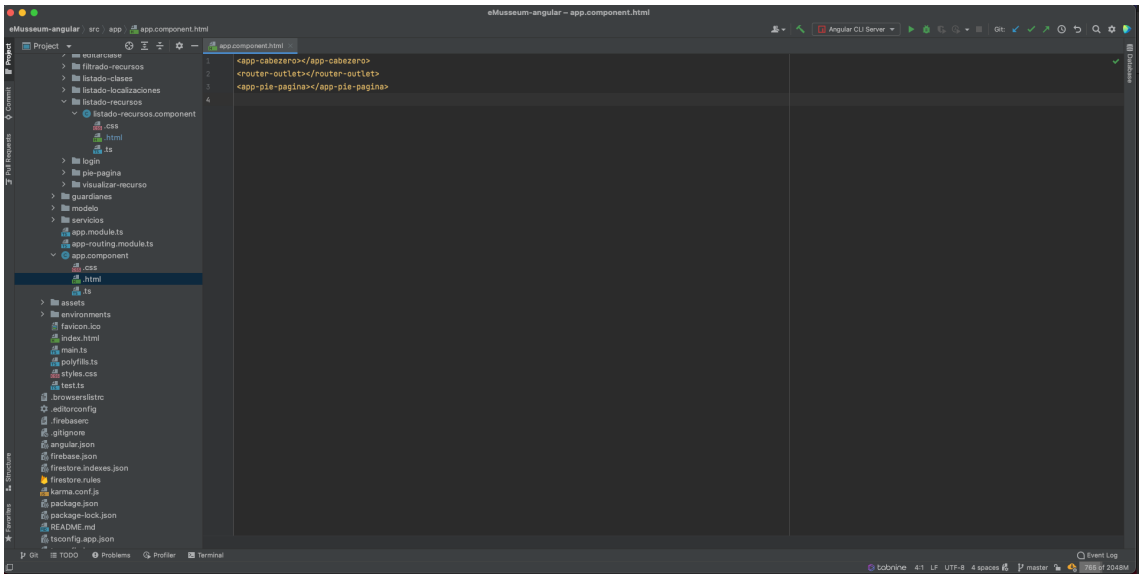


Figura 44: Configuración de funcionamiento de la aplicación.

Una vez definido tanto los componentes como las rutas, se prosiguió con la instalación de las librerías necesarias. Para este proyecto se ha decidido utilizar Bootstrap, FlashMessages, AngularFireModule, FormsModule y ReactiveFormsModule, los cuales se instalaron y se importaron al proyecto.

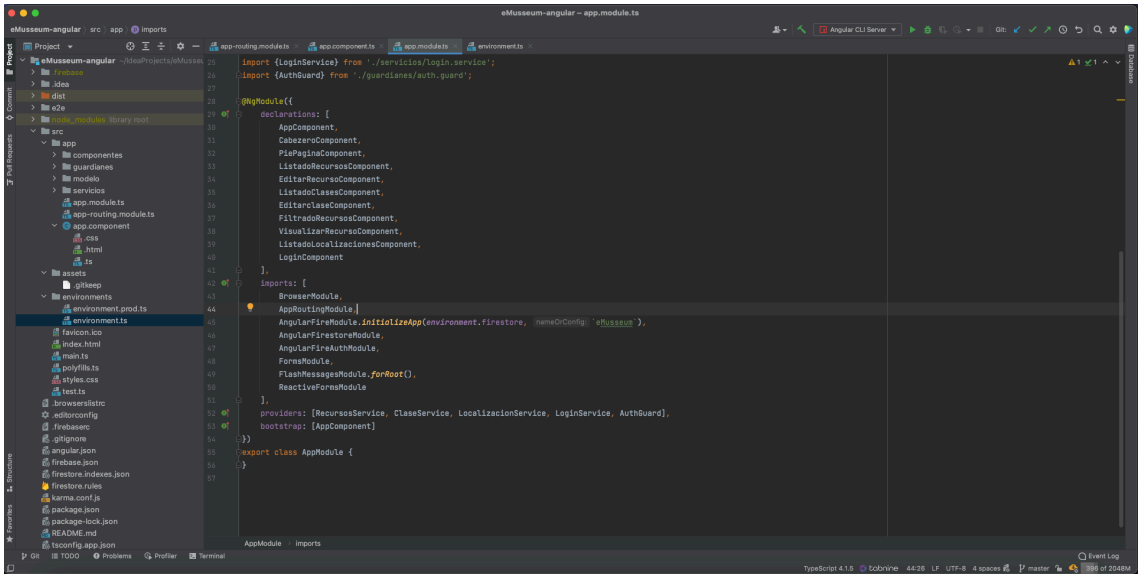


Figura 45: Configuración de funcionamiento de la aplicación.

Además, se modificó el Angular.json indicando que los estilos vienen dados por bootstrap.

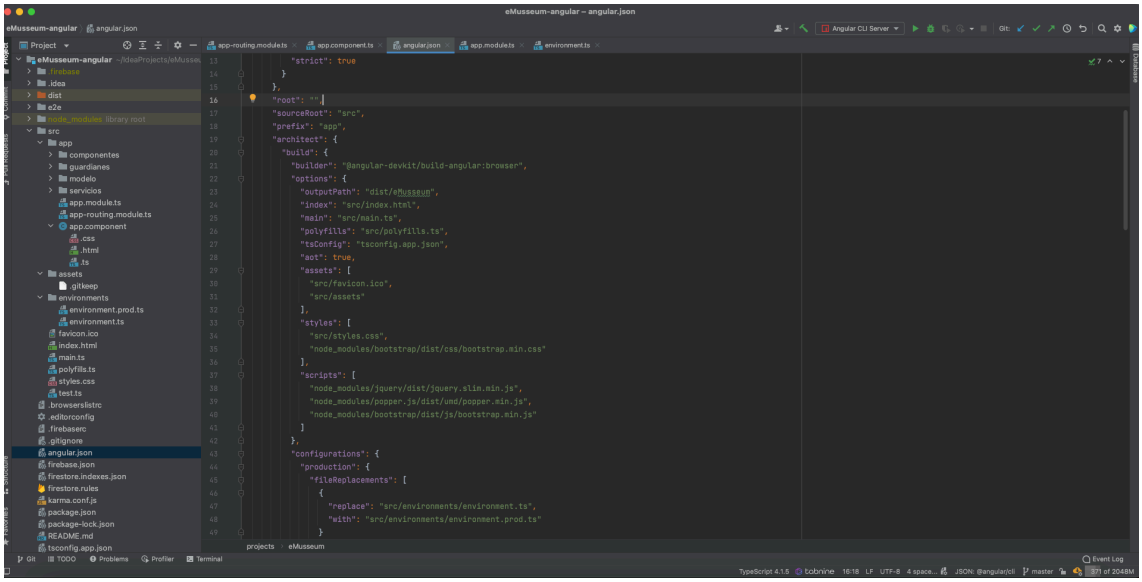


Figura 46: Configuración de Angular.json.

Seguidamente, se fueron desarrollado los componenete. Para ello, primeramente, se crearon los ficheros de modelo de datos. Estos ficheros sirven para definir el domino de datos con los que se van a trabajar y así convertir los JSON que se recibe a array de datos.

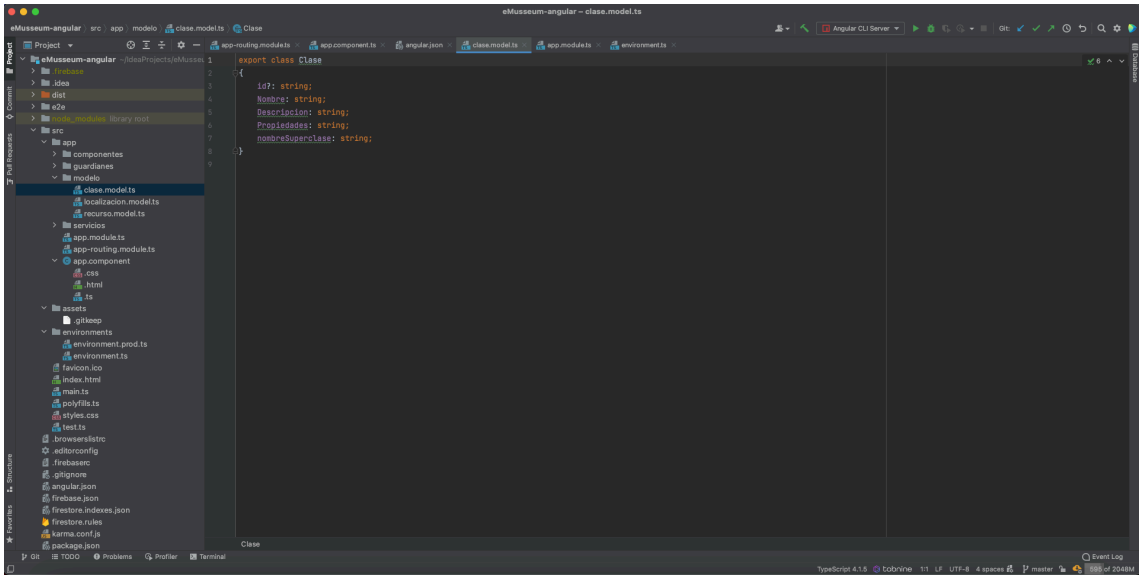


Figura 47: Modelo de datos.

A continuación, se continuó con la creación del servicio. El servicio es el componente que se encarga de conectarse con Firebase y recuperar los datos de la BBDD. Además de establecer las operaciones que se pueden realizar.

Tal y como se puede observar en la Figura 48: Servicio, no es necesario indicar la ruta de enganche a la base de datos ni los tokens de seguridad. Esto se debe a que se configuro el entorno de Firebase en el app.module.ts, por lo que en el momento de ejecución de la aplicación, esta se conecta a Firebase y se guarda en el navegador del usuario el estado y las credenciales. Además, si se inicia sesión, también se guardan los JWS (Json Web Token), sin necesidad de hacer la programación.

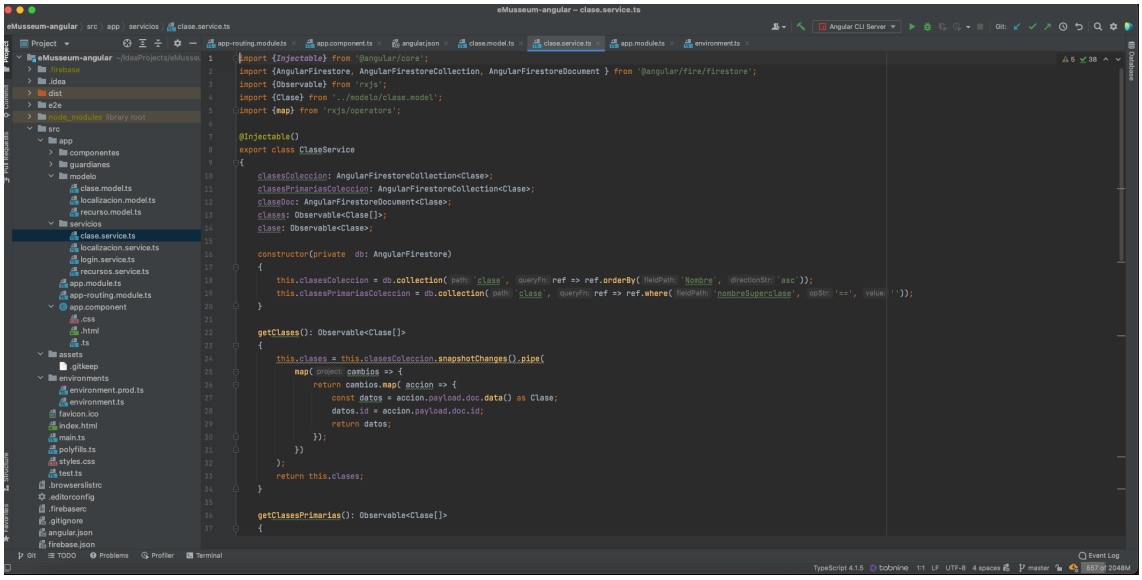


Figura 48: Servicio

En el momento de indicar un componente, se inicia el TS, el cual realiza tanto la carga del HTML como del CSS y los métodos de ngOnInit y de constructor.

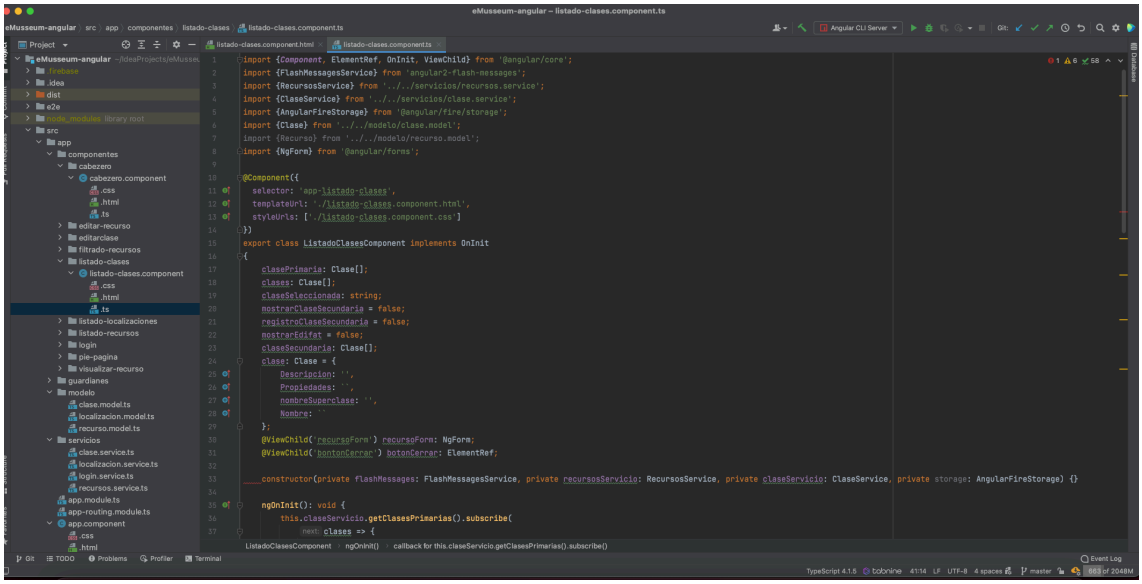


Figura 49: Componente - TS

Desde ese momento se establece las escuchas y las acciones en los componentes.

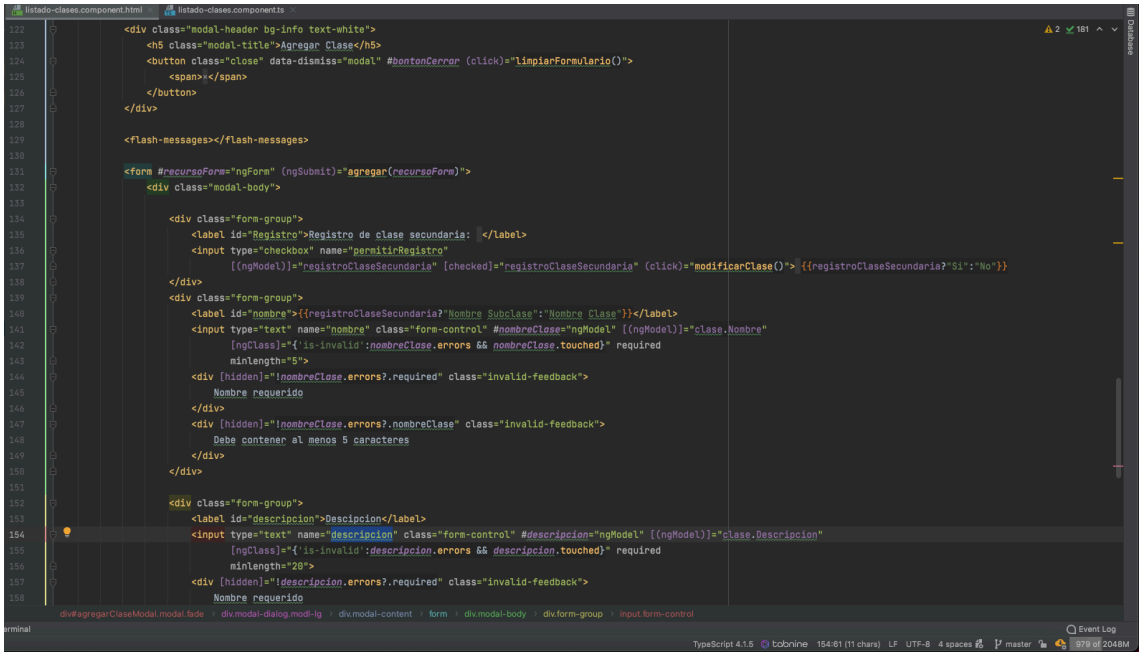


Figura 50: Componente – HMTL

Por último, se implementó el guardián, el cual permite guardar las rutas si no se dan las condiciones adecuadas.

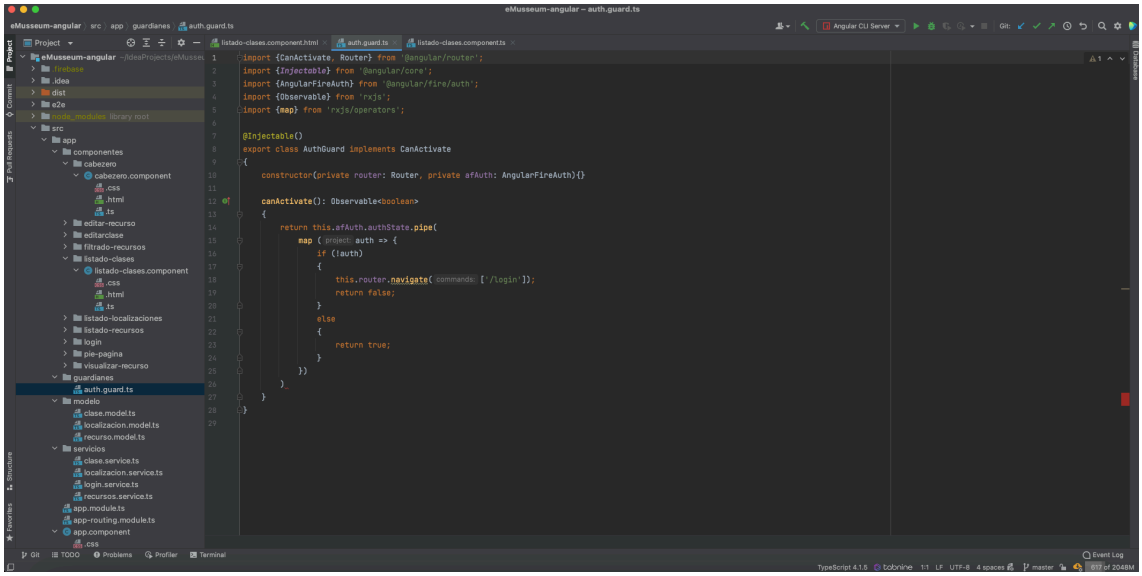


Figura 51: Guardián

6.3.2.2 APLICACIONES MÓVILES

Aunque dentro de los objetivos del presente proyecto no se encontraba desarrollar aplicaciones especificas para las plataformas móviles iOS y Android, se han desarrollado aplicaciones hibridas gracias al uso de cordova.

Cordova ha permitido compilar la aplicación web generada para producción para dichas plataformas móviles. Para ello, primeramente, se instaló globalmente el paquete de cordova.

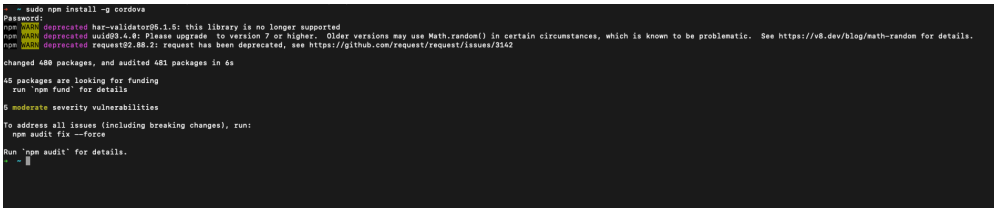


Figura 52: Instalación de cordova

A continuación, se creó una carpeta en específico para albergar dicho proyecto. Cabe destacar que en la carpeta de WWW se sustituyo el contenido existente por la aplicación de angular compilada y con las modificaciones necesarias aplicadas. A continaucion, se añadieron las plataformas de Android e iOS. Una vez añadidas se procedió a la compilación con cordoba build Android –debug y cordoba build ios –debug

Figura 53: Instalación de cordova

Una vez completados los procesos nos genera los ficheros APK para Android y .app para iOS. Cabe mencionar que la aplicación generada funciona perfectamente la parte pública. La parte de administración hay algún pequeño fallo visual pero al no tratarse de un objetivo del proyecto no ha sido aboradado.

6.3 PROBLEMAS

A lo largo de este punto se expondrá tanto los problemas como las dificultades que se encontraron durante la implementación de la solución y las soluciones que se descubrieron.

6.3.1 PROBLEMAS ENCONTRADOS EN JAVA

Durante la implementación de Java, se descubrieron muchísimos problemas, algunos de ellos se pudieron solucionar y otros no. A causa de los problemas que se expondrá a continuación, se decidió cambiar el entorno tecnológico de implementación de la solución.

6.3.1.1 NETBEANS 11.X

Se instaló la versión 11 de Netbeans, esta instalación se realiza a través de la descarga de los binarios y la creación de una carpeta de los mismo en la carpeta de aplicaciones. Luego, se ejecuta a través de la terminal.

El problema está en la configuración por lo que las clases entidad, servicios REST no se generaban correctamente.

6.3.1.2 NETBEANS 8.2

Una vez consultado los problemas que tenía con los anteriores editores, incluido la versión 11 de Netbeans, decidí utilizar la versión 8.2. Para ello, decidí crear el proyecto desde 0 utilizando Maven, en ese momento me salto el siguiente error:

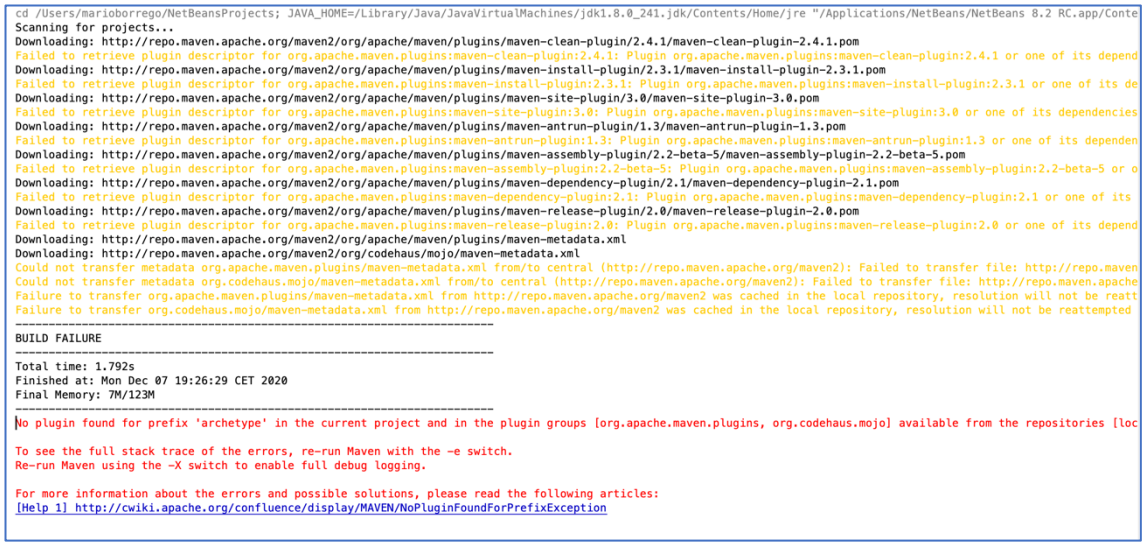


Figura 54: Error Netbeans 8.2

Buscando el error por páginas de desarrolladores encuentro una web[5]. En ella, se detalla que el error es debido a una actualización de la url del plugin, es por ello por lo que tengo que ir al fichero settings.xml de Maven y en <mirror> añadir el siguiente trozo de código:

```
<mirrors>
    <mirror>
        <id>mirror1</id>
        <mirrorOf>central</mirrorOf>
        <name>mirror1</name>
        <url>https://repo.maven.apache.org/maven2/</url>
    </mirror>
</mirrors>
```

Figura 55: Solución Netbeans 8.2

Este fragmento de código actualiza el plugin de Maven y, por consiguiente, resuelve el problema descrito.

[5] Starkoverflow(2022) Error "No plugin found for prefix 'archetype' in the current project and in the plugin groups" getting in creating new java maven project. Fecha consulta: 26/02/2022 https://stackoverflow.com/questions/60124030/error-no-plugin-found-for-prefix-archetype-in-the-current-project-and-in-the

6.3.1.3 GLASSFISH5.0 – MAIL

La versión 5.0 de GlassFish tiene un error en la clase sun/sucrity/ssl/HelloExtension lo cual impide que el código encargado de conectarse con el servidor de Gmail para el envío de mensajes pueda ejecutarse.

```
java.lang.NoClassDefFoundError: sun/security/ssl/HelloExtension
.send()}": java.lang.NoClassDefFoundError: sun/security/ssl/HelloExtension
```

Figura 56: Glassfish 5.0

La solución ha sido utilizar una versión más antigua de GlassFish, en este caso he utilizado un fork de este llamado payara. En concreto, la versión Payara Server 4.1.2.181 la cual no tiene este bug.

6.3.1.4 INCLUSIÓN DE CÓDIGO JAVASCRIPT

A lo largo de la aplicación para intentar dotarla de dinamismo, se desarrollaron diversos códigos JavaScript. El problema nace cuando se intentaron que ejecutaran acciones dentro la aplicación. Estas no se ejecutaban correctamente.

El problema puede deberse a la utilización de un núcleo modificado de Linux, el cual no ejecuta correctamente ciertas instrucciones dentro de la misma.

6.3.1.5 INCLUSIÓN DE LA LIBRERÍA BOOTSTRAP

También se intentó utilizar la librería de CSS Bootstrap dentro del proyecto con el fin de facilitar el desarrollo de una UI amigable para el usuario final. Esta no funcionaba correctamente y puede deberse a los mismos problemas mencionados en el punto anterior.

6.3.2 PROBLEMAS ENCONTRADOS EN ANGULAR

El mayor problema que se encontró en Angular es que la lógica de negocio de la aplicación debía de estar en el lado cliente. Esto supone que un usuario con ciertos conocimientos pudiera realizar acciones a lo que no está autorizado. Pero gracias a la implementación de las reglas de Firebase y al uso de guardianes esta casuística desapareció.

7 PRUEBAS

En este punto se expondrá cada una de las pruebas que se han realizado al sistema, las cuales han permitido verificar el correcto funcionamiento del sistema y en caso de descubrir un comportamiento no deseado, se corrige.

Para ello, se realizaron los siguientes tipos de prueba:

- Pruebas caja blanca
- Pruebas caja negra

7.1 PRUEBAS DE CAJA BLANCA

Las pruebas de caja blanca se encargan de comprobar que la secuencia de información que se mueven dentro de cada una de las funcionalidades que componen la plataforma web se hace de forma correcta. Estas pruebas deben de comprobar que:

- Cada instrucción del programa se ejecuta al menos una vez
- Se garantiza que las decisiones sean verdaderas o falsas.
- Se ejecutan bucles, probando el estado general y el caso extremo
- estructuras internas utilizadas

Las pruebas de caja blanca se realizan en paralelo con el desarrollo de nuevas funciones, para verificar que todas las vías que se pueden crear están funcionando.

7.2 PRUEBAS DE CAJA NEGRA

El tester de caja negra se encarga de comprobar la funcionalidad para la que se desarrolló la aplicación sin tener en cuenta el código.

Se centran en las entradas y salidas del sistema, independientemente de su conocimiento de la estructura interna del programa. Para obtener detalles sobre estas entradas y salidas, las basamos en los requisitos de software y las especificaciones funcionales. A continuación, se muestran algunos detalles de las pruebas realizadas:

ID y Nombre:	PCN-01: Identificar al usuario
Objetivo	Identificar a un usuario en el sistema
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet 3. El usuario debe de haberse dado de alta en el panel de control de la aplicación de google
Datos de entrada	• Usuario: admin@emusseum.com • Password: Prueba1234.
Acción esperada	Al pulsar sobre el botón “login”, el sistema debe de identificar y autorizar en el sistema al usuario. Además de habilitar todas las operaciones que se pueden realizar
Resultado	Correcto

Tabla 41: PCN-01

ID y Nombre:	PCN-02: Listar recursos
Objetivo	Listar todos los recursos que estén dados de alta en el sistema
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet
Datos de entrada	-
Acción esperada	Listar todos los recursos del sistema
Resultado	Correcto

Tabla 42: PCN-02

ID y Nombre:	PCN-03: Filtrado de recursos
Objetivo	Poder listar todos los recursos de una misma clase
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet
Datos de entrada	Las clases dadas de alta en el sistema
Acción esperada	El listado con los recursos filtrados
Resultado	Correcto

Tabla 43: PCN-03

ID y Nombre:	PCN-04: Listar clases
Objetivo	Listar todas las clases que están dadas de alta en el sistema
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet 3. El usuario debe de estar identificado en el sistema
Datos de entrada	-
Acción esperada	Un listado con todas las clases listadas, tanto primarias como secundarias si se selecciona alguna
Resultado	Correcto

Tabla 44: PCN-04

ID y Nombre:	PCN-05: Listar localizaciones
Objetivo	Listar todas las localizaciones dadas de alta en el sistema
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet 3. El usuario debe de estar identificado en el sistema
Datos de entrada	-
Acción esperada	Un listado con todas las localizaciones listadas, tanto primarias como secundarias si se selecciona alguna
Resultado	Correcto

Tabla 45: PCN-05

ID y Nombre:	PCN-06: Protección de rutas
Objetivo	El sistema mostrará una pantalla de login si un usuario no esta identificado en el sistema
Precondición	1. El sistema tiene que estar online 2. El usuario debe de tener acceso a internet
Datos de entrada	-
Acción esperada	Se debe de redirigir al usuario a la pantalla de identificación
Resultado	Correcto

Tabla 46: PCN-06

8 CONCLUSIONES

A lo largo del desarrollo de este proyecto, se ha enfocado el mismo de numerosas formas. Si bien es cierto que el proyecto siempre se ha basado en el desarrollo de una aplicación web para gestionar los recursos que la Escuela tiene disponible, la forma de abordar el problema se ha modificado sustancialmente.

El desarrollo del presente proyecto ha supuesto un reto enorme, ya que he dispuesto de todo el control de este. Esto supone que todas las decisiones que se tenían que tomar recaían sobre mí, además las situaciones vividas por elementos externos no han sido nada fáciles.

Todo ello ha supuesto adquirir un *Know-How* personal de muy importante para mi vida laboral, ya que gracias a las decisiones tomadas (tanto en acertadas, como las equivocadas) me han permitido tomar una consciencia sobre las consecuencias de estas.

Por otra parte, el proyecto me ha permitido investigar y conocer nuevas tecnologías y herramientas de desarrollo, ampliando mis horizontes y sobre todo, permitirme conocerme a mi mismo y salir de zona de confort, siendo capaz de conocer mis propias deficiencias y mis puntos fuertes.

Por todo lo explicado anteriormente el proyecto ha sido un éxito, ya que:

1. A pesar del tiempo empleado, el proyecto se ha finalizado con éxito y se ha puesto en funcionamiento.

2. He sido capaz de salir de la zona de confort y de escoger una mejor solución para el desarrollo de la solución.

3. Se ha adquirido un conocimiento muy importante gracias a la realización de cursos para el desarrollo del proyecto.

9 REFERENCIAS.

9.1 ENLACES A WEBS CONULTADAS

Escuela Superior de Informática. Universidad de Castilla-La Mancha. (2008). *Museo Virtual de la informática. Fecha de consulta 26/02/2022.* http://www.esi.uclm.es/museo/index.html

Universidad de Valladolid. (s.f.).*Museo Virtual 3D de la Informática. . Fecha de consulta 26/02/2022.* https://museo.inf.uva.es/museo3d/contactos.html

Paloma, Steven, Bernal, Alejandro, Rodríguez, Tatiana, (s.f.) *Desarrollo Iterativo e Incremental. Fecha de consulta 26/02/2022* https://danelly1236.files.wordpress.com/2013/12/modelo-iterativo-incremental-presentacion2.pdf

Proyectos agiles org. (s.f.) *Desarrollo Iterativo e Incremental. Fecha de consulta 26/02/2022* https://proyectosagiles.org/desarrollo-iterativo-incremental/

Apache NetBeans (2017-2020) *Video of PrimeFaces Development with NetBeans IDE Fecha de consulta 26/02/2022* https://netbeans.org/kb/docs/javaee/maven-primefaces-screencast.html

PMOinformatica.com (2012-2018) *Requerimientos No Funcionales: Porque son importantes. Fecha de consulta 26/02/2022* http://www.pmoinformatica.com/2013/01/requerimientos-no-funcionales-porque.html

Caso de Uso (2020, 25 octubre) *Wikipedia, la enciclopedia libre*. *Fecha de consulta 26/02/2022* https://es.wikipedia.org/wiki/Caso_de_uso

Arquitectura de Software (2022, 23 febrero) *Wikipedia, la enciclopedia libre*. *Fecha de consulta 26/02/2022* https://es.wikipedia.org/wiki/Arquitectura_de_software

Escobar (2010) Enviar emails desde Java con Yakarta mail. Espai. *Fecha de consulta 26/02/2022* https://www.espai.es/blog/2020/01/enviar-emails-desde-java-con-jakarta-mail/

9.1.1 MAIL

Jlsmorilloblog (11 de octubre 2018). Java Mail. *Fecha de consulta:* 26/02/2022 https://jlsmorilloblog.wordpress.com/2018/10/11/__trashed/

Geovanny0401 (octubre de 2016). Api Javamail con glassfish. *Fecha de consulta:* 26/02/2022 http://geovanny0401.blogspot.com/2016/10/api-javamail-configurada-con-glassfish.html

Gateau27 (enero de 2012). Javamail con glassfish. *Fecha de consulta:* 26/02/2022 https://bateau27.files.wordpress.com/2012/01/javamail-con-glassfish.pdf

Lijianzhao (julio de 2016). Send email with javamail api. *Fecha de consulta:* 26/02/2022 https://lijianzhao.wordpress.com/2016/07/02/send-email-with-javamail-api/

9.2 DEFINICIONES

GlassFish. (2019, 11 de octubre). Wikipedia, La enciclopedia libre. *Fecha de consulta:* febrero 14, 2022 desde https://es.wikipedia.org/w/index.php?title=GlassFish&oldid=120191424.

GlassFish. (2022). Wikipedia, La enciclopedia libre. *Fecha de consulta:* 26/02/2022 https://es.wikipedia.org/wiki/GitHub

Firebase (2022). Google firebase. *Fecha de consulta:* 26/02/2022. https://Firebase.google.com/docs/database?hl=es

9.3 MIGRACIÓN

Angular (2022). Google Angular. *Fecha de consulta:* 26/02/2022

https://update.angular.io/?l=3&v=11.0-12.0

9.4 GUIA DE REFERENCIA

Angular (2022). Google Angular The modern web developer's platform. *Fecha de consulta: 26/02/2022* https://angular.io/