APLICACIÓN PARA LA GESTIÓN DE AFOROS EN LAS UNIVERSIDADES: RESUNIVERSITAS

APPLICATION FOR MANAGING CAPACITY IN UNIVERSITIES: RESUNIVERSITAS

JOSÉ MIGUEL CABRERA LLAMAS
VÍCTOR CHOZA MERINO
ÁLVARO PENALVA ALBERCA
PEDRO SÁNCHEZ ESCRIBANO

GRADO EN INGENIERÍA DE SOFTWARE E INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO DE FIN DE GRADO
CURSO 2021-2022

DIRECTOR
ANTONIO SARASA CABEZUELO

Agradecimientos

Este trabajo es el resultado de varios meses de esfuerzo y aprendizaje. En especial, agradecer a nuestras familias y amigos, que nos han brindado su apoyo durante este camino universitario.

Resumen

En este proyecto se recoge el diseño, la especificación y la implementación tanto de una solución Android como de una solución web de valor añadido.

En el presente trabajo se desarrolla un software que facilita la gestión de aforos en universidades ofreciendo funcionalidades como reservar un asiento en una determinada aula o comprobar la disponibilidad de los espacios de las facultades que conforman las universidades. También, a nivel de administrador, se permite modificar cualquier aspecto de las salas y las facultades, gestionar las plazas de las instalaciones de la universidad y crear eventos y noticias para los usuarios de estas.

La extracción de una parte de los datos usados en la aplicación proviene de las páginas oficiales de las entidades universitarias. Entre estos datos se encuentra información como la dirección, correo, descripción e imágenes. Los datos restantes han sido generados de manera intuitiva.

PALABRAS CLAVE

Universidad, facultad, ocupación, reservar, Android, Angular, Firebase

Abstract

This project includes the design, specification and implementation of both, an Android solution, and a web solution, both of which have added value.

In the present work a software is developed that facilitates the management of capacity in universities offering functionalities such as reserve a seat in a certain classroom or check the availability of the spaces of the faculties. Also, at the administrator level, it is allowed to modify any aspect of the rooms and the faculties, manage the sites of the university facilities, and create events and news for the users of these.

The extraction of a part of the data used in the application comes from the official websites of the university entities. Among this data is information such as address, mail, description, and images. The remaining data has been generated intuitively.

KEYWORDS
University, faculty, occupation, reserve, Android, Angular, Firebase

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este capítulo se realizará un breve resumen de lo que es ResUniversitas, se comentará cuáles han sido las motivaciones que han llevado a su desarrollo, y se explicarán los objetivos a alcanzar en el desarrollo del trabajo.

En definitiva, esté capítulo pretende poner en contexto al lector y situarlo dentro del tema que será tratado en esta memoria.

1.1. Motivación

Las consecuencias de la pandemia COVID-19 han sido nefastas a nivel mundial. En los tres últimos años la población se ha visto obligada a cambiar la forma de relacionarse, y empresas, comercios y universidades entre otras instituciones han tenido que adaptarse a las nuevas normativas establecidas para reducir el contagio del virus.

Debido a la crisis epidemiológica que ha provocado el virus COVID-19, y en consonancia con sus numerosas variantes, se ha producido una revolución en la forma de las reuniones presenciales, sobre todo, en lugares públicos como es el Campus Universitario. En concreto, la medida sanitaria del aforo de personas ha dado lugar a la creación de una necesidad concreta como es el control y la limitación de la capacidad del aforo.

Por otro lado, dejando atrás la situación actual de pandemia, y en consonancia con la experiencia adquirida durante el periodo universitario, se observó que los servicios de gestión de espacios en la universidad no disponen de información al alcance de todos referente a la ocupación, así como no cuentan con un servicio de reserva en dichos espacios. Además, la información de los distintos eventos y conferencias que tienen lugar en las facultades no se muestran de manera sencilla, unificada e intuitiva.

En consonancia con esta situación y para solventar estas limitaciones, en este trabajo se ha propuesto el desarrollo de una aplicación móvil y web en la que, de forma ordenada y planificada, y respetando las medidas que en cada momento sean instauradas, se pueda hacer uso de las instalaciones que la universidad pone a disposición de todos los usuarios.

En esta aplicación se permite visualizar la información referente a la ocupación, así como, la reserva de un determinado asiento mediante vista de plano, dando una solución al problema anteriormente planteado.

También resulta útil la reserva de un asiento para una determinada clase o evento, de modo que, si el estudiante sabe que va a llegar con retraso a su clase, pueda reservar un asiento de antemano, o consultar las estadísticas de ocupación para el periodo de tiempo que ocupa su asignatura.

1.2. Objetivos

El objetivo principal de este trabajo es la implementación de una aplicación Android que permite a sus usuarios reservar un asiento en una determinada aula o comprobar la disponibilidad de los espacios cerrados (a partir de ahora se denominan salas para generalizar, ya que pueden incluir aulas, laboratorios, cafeterías, etc.), así como gestionar las plazas de las instalaciones de la universidad y mostrar eventos y noticias a los usuarios.

Aparte de la aplicación Android, ResUniversitas también está formado por una aplicación web que recogerá únicamente la gestión de las instalaciones, la cual al igual que en la aplicación Android permite a un usuario acreditado como administrador crear y modificar facultades, salas y noticias o eventos...

A continuación, se detallan los objetivos específicos:

- Desarrollo de un módulo que disponga de una visualización de la información correspondiente al aforo de las salas de las universidades.
- Implantación de un módulo específico para usuarios donde se les permite gestionar reservas de asientos y consultar el aforo de las salas.
- Puesta en marcha de un módulo dedicado a la gestión de las instalaciones de las universidades, al cual sólo tendrán acceso los usuarios administradores.
- Diseño e implementación de una página web de apoyo a la aplicación Android que permita realizar las gestiones necesarias a los usuarios administradores de los edificios o zonas.
- Tanto la aplicación de Android como la página web compartirán la misma base de datos y repositorio de almacenamiento.

1.3. Estructura de la memoria

En este apartado, se describen brevemente los capítulos que forman parte de la estructura de la memoria:

Capítulo 1: En este capítulo se describe la motivación del trabajo, los objetivos a conseguir, la estructura de la memoria y el plan de trabajo.

Capítulo 2: En este capítulo se detallan soluciones similares a la que se ha realizado en el trabajo.

Capítulo 3: En este capítulo se describe la tecnología utilizada en el desarrollo del proyecto.

Capítulo 4: En este capítulo se detallan los actores y se especifican los casos de uso (CU) junto a su descripción.

Capítulo 5: En este capítulo se detalla la arquitectura y los patrones utilizados en la aplicación Android, así como en la aplicación web. Además, se describen los modelos utilizados en la base de datos y en el repositorio.

Capítulo 6: En este capítulo se describen los aspectos principales de la funcionalidad y el diseño en la aplicación Android, así como en la aplicación web.

Capítulo 7: En este capítulo se detalla la encuesta de usabilidad y se analizan los resultados obtenidos en dicha encuesta con usuarios reales.

Capítulo 8: En este capítulo se realiza una conclusión y se describen brevemente los futuros casos de uso.

Capítulo 9: En este capítulo se desarrollan las tareas que han llevado a cabo los miembros del equipo.

Bibliografía

Anexo I: Guía de Usuario

Anexo II: Preguntas de la evaluación

1.4. Plan de trabajo

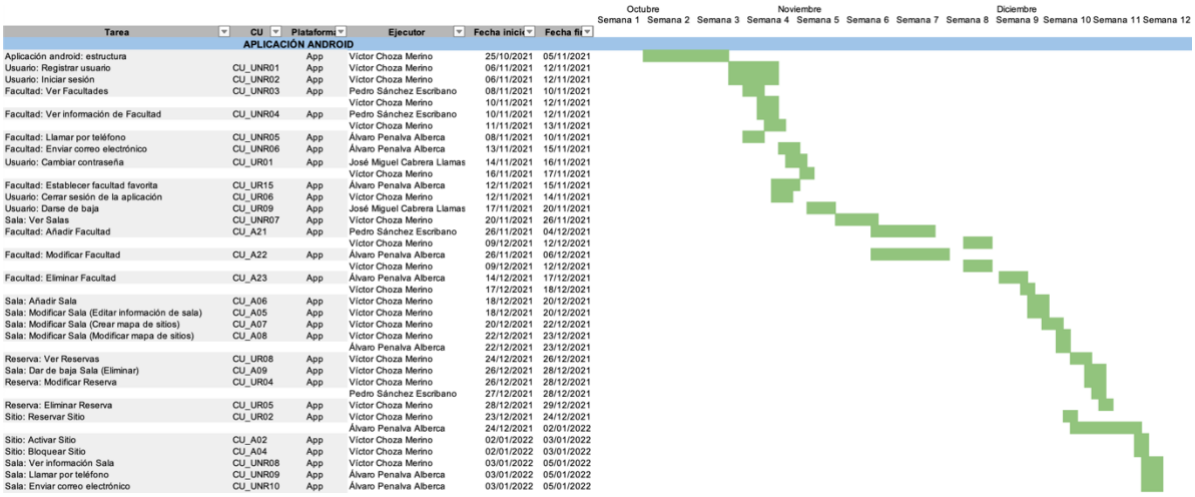
En este apartado se va a realizar un resumen con ayuda de un diagrama de Gantt representado en las Figuras 1, 2, 3, 4, 5, 6 y 7, en él se expondrá el desarrollo del proyecto a lo largo del tiempo, dividiéndose la carga de trabajo en tres bloques principales: aplicación Android, aplicación web y documentación. Es importante señalar que, al estar formado el proyecto por cuatro miembros, se ha trabajado paralelamente en la aplicación Android, la aplicación web y en la documentación.

Inicialmente, se realizó una reunión para determinar y aclarar el objeto de este proyecto como se muestra en la Figura 1. En esta misma figura también se refleja la siguiente reunión en la que se tomaron los requisitos de usuario dando lugar a los casos de uso para comenzar posteriormente el desarrollo de los objetivos.

Figura 1 - Diagrama de Gantt: planificación genérica

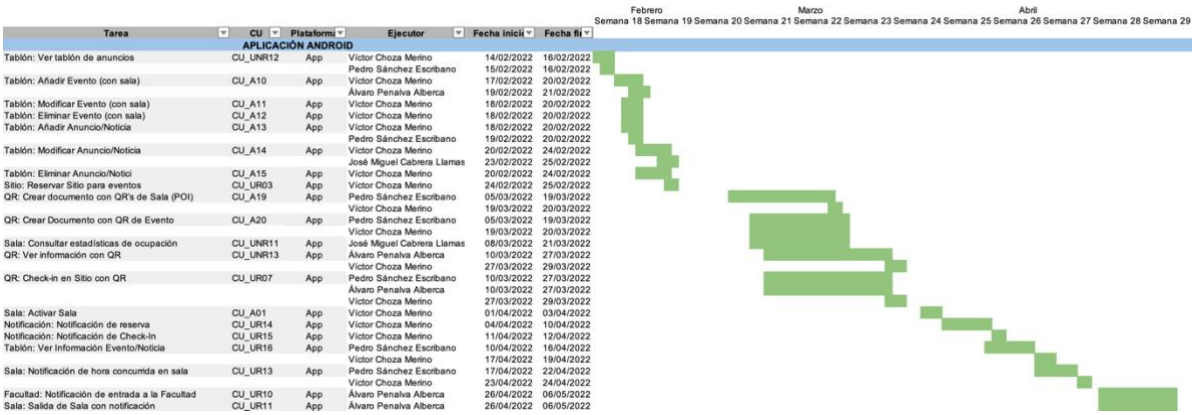
En las Figuras 2 y 3 se reflejan todas las tareas que se han acometido en el desarrollo de la aplicación Android. Se puede ver como este desarrollo es el que más tiempo ha ocupado,

yendo desde el mes de octubre de 2021 al mes de abril de 2022 prácticamente sin periodos vacíos.



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN ANDROID					
Aplicación android: estructura		App	Víctor Choza Merino	25/10/2021	05/11/2021
Usuario: Registrar usuario	CU_UNR01	App	Víctor Choza Merino	06/11/2021	12/11/2021
Usuario: Iniciar sesión	CU_UNR02	App	Víctor Choza Merino	06/11/2021	12/11/2021
Facultad: Ver Facultades	CU_UNR03	App	Pedro Sánchez Escribano	08/11/2021	10/11/2021
			Víctor Choza Merino	10/11/2021	12/11/2021
Facultad: Ver información de Facultad	CU_UNR04	App	Pedro Sánchez Escribano	10/11/2021	12/11/2021
			Víctor Choza Merino	11/11/2021	13/11/2021
Facultad: Llamar por teléfono	CU_UNR05	App	Álvaro Penalva Alberca	08/11/2021	10/11/2021
Facultad: Enviar correo electrónico	CU_UNR06	App	Álvaro Penalva Alberca	13/11/2021	15/11/2021
Usuario: Cambiar contraseña	CU_UR01	App	José Miguel Cabrera Llamas	14/11/2021	16/11/2021
			Víctor Choza Merino	16/11/2021	17/11/2021
Facultad: Establecer facultad favorita	CU_UR15	App	Álvaro Penalva Alberca	12/11/2021	15/11/2021
Usuario: Cerrar sesión de la aplicación	CU_UR06	App	Víctor Choza Merino	12/11/2021	14/11/2021
Usuario: Darse de baja	CU_UR09	App	José Miguel Cabrera Llamas	17/11/2021	20/11/2021
Sala: Ver Salas	CU_UNR07	App	Víctor Choza Merino	20/11/2021	26/11/2021
Facultad: Añadir Facultad	CU_A21	App	Pedro Sánchez Escribano	26/11/2021	04/12/2021
			Víctor Choza Merino	09/12/2021	12/12/2021
Facultad: Modificar Facultad	CU_A22	App	Álvaro Penalva Alberca	26/11/2021	06/12/2021
			Víctor Choza Merino	09/12/2021	12/12/2021
Facultad: Eliminar Facultad	CU_A23	App	Álvaro Penalva Alberca	14/12/2021	17/12/2021
			Víctor Choza Merino	17/12/2021	18/12/2021
Sala: Añadir Sala	CU_A06	App	Víctor Choza Merino	18/12/2021	20/12/2021
Sala: Modificar Sala (Editar información de sala)	CU_A05	App	Víctor Choza Merino	18/12/2021	20/12/2021
Sala: Modificar Sala (Crear mapa de sitios)	CU_A07	App	Víctor Choza Merino	20/12/2021	22/12/2021
Sala: Modificar Sala (Modificar mapa de sitios)	CU_A08	App	Víctor Choza Merino	22/12/2021	23/12/2021
			Álvaro Penalva Alberca	22/12/2021	23/12/2021
Reserva: Ver Reservas	CU_UR08	App	Víctor Choza Merino	24/12/2021	26/12/2021
Sala: Dar de baja Sala (Eliminar)	CU_A09	App	Víctor Choza Merino	26/12/2021	28/12/2021
Reserva: Modificar Reserva	CU_UR04	App	Víctor Choza Merino	26/12/2021	28/12/2021
			Pedro Sánchez Escribano	27/12/2021	28/12/2021
Reserva: Eliminar Reserva	CU_UR05	App	Víctor Choza Merino	28/12/2021	29/12/2021
Sitio: Reservar Sitio	CU_UR02	App	Víctor Choza Merino	23/12/2021	24/12/2021
			Álvaro Penalva Alberca	24/12/2021	02/01/2022
Sitio: Activar Sitio	CU_A02	App	Víctor Choza Merino	02/01/2022	03/01/2022
Sitio: Bloquear Sitio	CU_A04	App	Víctor Choza Merino	02/01/2022	03/01/2022
Sala: Ver información Sala	CU_UNR08	App	Víctor Choza Merino	03/01/2022	05/01/2022
Sala: Llamar por teléfono	CU_UNR09	App	Álvaro Penalva Alberca	03/01/2022	05/01/2022
Sala: Enviar correo electrónico	CU_UNR10	App	Álvaro Penalva Alberca	03/01/2022	05/01/2022

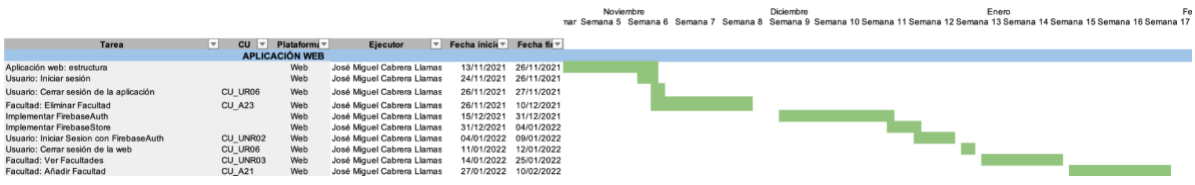
Figura 2 - Diagrama de Gantt: Aplicación Android 1



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN ANDROID					
Tablón: Ver tablón de anuncios	CU_UNR12	App	Víctor Choza Merino	14/02/2022	16/02/2022
			Pedro Sánchez Escribano	15/02/2022	16/02/2022
Tablón: Añadir Evento (con sala)	CU_A10	App	Víctor Choza Merino	17/02/2022	20/02/2022
			Álvaro Penalva Alberca	19/02/2022	21/02/2022
Tablón: Modificar Evento (con sala)	CU_A11	App	Víctor Choza Merino	18/02/2022	20/02/2022
Tablón: Eliminar Evento (con sala)	CU_A12	App	Víctor Choza Merino	18/02/2022	20/02/2022
Tablón: Añadir Anuncio/Noticia	CU_A13	App	Víctor Choza Merino	18/02/2022	20/02/2022
			Pedro Sánchez Escribano	19/02/2022	20/02/2022
Tablón: Modificar Anuncio/Noticia	CU_A14	App	Víctor Choza Merino	20/02/2022	24/02/2022
			José Miguel Cabrera Llamas	23/02/2022	25/02/2022
Tablón: Eliminar Anuncio/Notici	CU_A15	App	Víctor Choza Merino	20/02/2022	24/02/2022
Sitio: Reservar Sitio para eventos	CU_UR03	App	Víctor Choza Merino	24/02/2022	25/02/2022
QR: Crear documento con QR's de Sala (POI)	CU_A19	App	Pedro Sánchez Escribano	05/03/2022	19/03/2022
			Víctor Choza Merino	19/03/2022	20/03/2022
QR: Crear Documento con QR de Evento	CU_A20	App	Pedro Sánchez Escribano	05/03/2022	19/03/2022
			Víctor Choza Merino	19/03/2022	20/03/2022
Sala: Consultar estadísticas de ocupación	CU_UNR11	App	José Miguel Cabrera Llamas	08/03/2022	21/03/2022
QR: Ver información con QR	CU_UNR13	App	Álvaro Penalva Alberca	10/03/2022	27/03/2022
			Víctor Choza Merino	27/03/2022	29/03/2022
QR: Check-in en Sitio con QR	CU_UR07	App	Pedro Sánchez Escribano	10/03/2022	27/03/2022
			Álvaro Penalva Alberca	10/03/2022	27/03/2022
			Víctor Choza Merino	27/03/2022	29/03/2022
Sala: Activar Sala	CU_A01	App	Víctor Choza Merino	01/04/2022	03/04/2022
Notificación: Notificación de reserva	CU_UR14	App	Víctor Choza Merino	04/04/2022	10/04/2022
Notificación: Notificación de Check-In	CU_UR15	App	Víctor Choza Merino	11/04/2022	12/04/2022
Tablón: Ver Información Evento/Noticia	CU_UR16	App	Pedro Sánchez Escribano	10/04/2022	16/04/2022
			Víctor Choza Merino	17/04/2022	19/04/2022
Sala: Notificación de hora concurrida en sala	CU_UR13	App	Pedro Sánchez Escribano	17/04/2022	22/04/2022
			Víctor Choza Merino	23/04/2022	24/04/2022
Facultad: Notificación de entrada a la Facultad	CU_UR10	App	Álvaro Penalva Alberca	26/04/2022	06/05/2022
Sala: Salida de Sala con notificación	CU_UR11	App	Álvaro Penalva Alberca	26/04/2022	06/05/2022

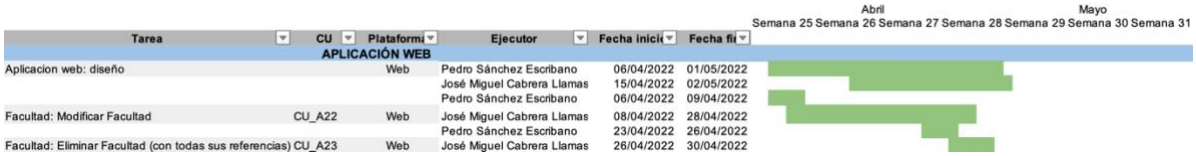
Figura 3 - Diagrama de Gantt: Aplicación Android 2

A continuación, en las Figuras 4 y 5 se muestran las tareas realizadas en el desarrollo de la aplicación web. En este caso los desarrollos más importantes se dieron a lo largo de noviembre, diciembre y enero, para luego culminar los trabajos en el mes de abril.



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN WEB					
Aplicación web: estructura		Web	José Miguel Cabrera Llamas	13/11/2021	26/11/2021
Usuario: Iniciar sesión		Web	José Miguel Cabrera Llamas	24/11/2021	26/11/2021
Usuario: Cerrar sesión de la aplicación	CU_UR06	Web	José Miguel Cabrera Llamas	26/11/2021	27/11/2021
Facultad: Eliminar Facultad	CU_A23	Web	José Miguel Cabrera Llamas	26/11/2021	10/12/2021
Implementar FirebaseAuth		Web	José Miguel Cabrera Llamas	15/12/2021	31/12/2021
Implementar FirebaseStore		Web	José Miguel Cabrera Llamas	31/12/2021	04/01/2022
Usuario: Iniciar Sesion con FirebaseAuth	CU_UNR02	Web	José Miguel Cabrera Llamas	04/01/2022	09/01/2022
Usuario: Cerrar sesión de la web	CU_UR06	Web	José Miguel Cabrera Llamas	11/01/2022	12/01/2022
Facultad: Ver Facultades	CU_UNR03	Web	José Miguel Cabrera Llamas	14/01/2022	25/01/2022
Facultad: Añadir Facultad	CU_A21	Web	José Miguel Cabrera Llamas	27/01/2022	10/02/2022

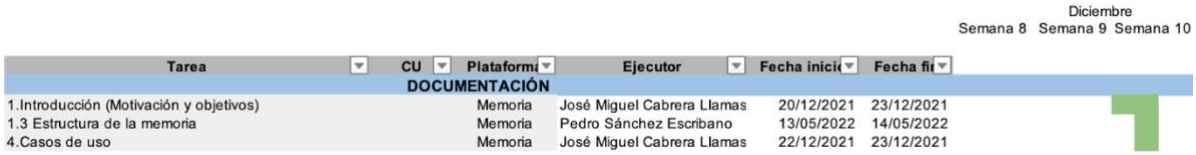
Figura 4 - Diagrama de Gantt: Aplicación web 1



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN WEB					
Aplicacion web: diseño		Web	Pedro Sánchez Escribano	06/04/2022	01/05/2022
			José Miguel Cabrera Llamas	15/04/2022	02/05/2022
			Pedro Sánchez Escribano	06/04/2022	09/04/2022
Facultad: Modificar Facultad	CU_A22	Web	José Miguel Cabrera Llamas	08/04/2022	28/04/2022
			Pedro Sánchez Escribano	23/04/2022	26/04/2022
Facultad: Eliminar Facultad (con todas sus referencias)	CU_A23	Web	José Miguel Cabrera Llamas	26/04/2022	30/04/2022

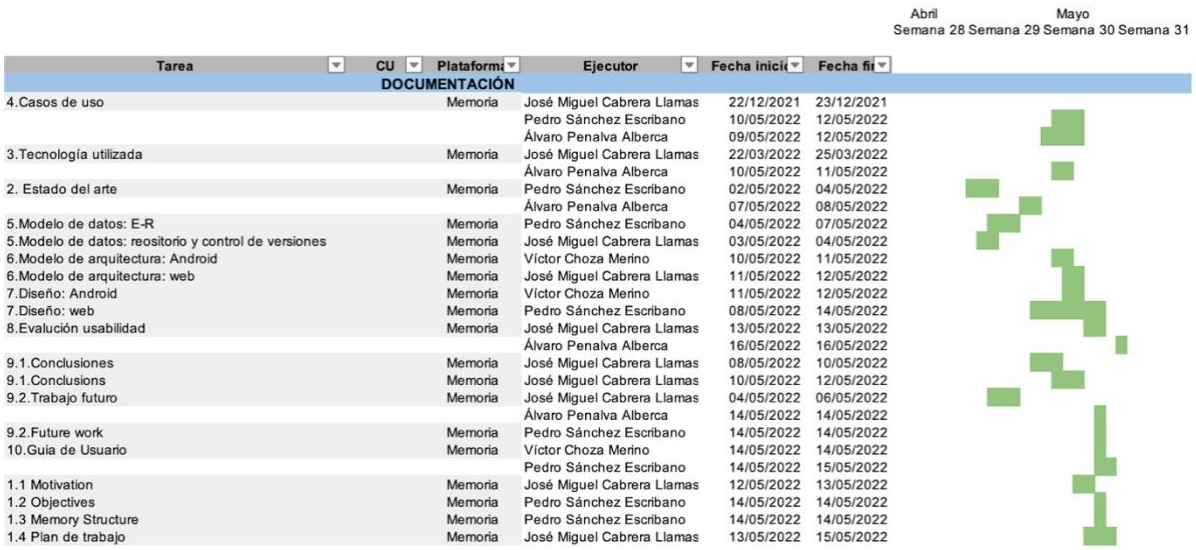
Figura 5 - Diagrama de Gantt: Aplicación web 2

Finalmente, en las Figuras 6 y 7, se recogen las tareas que constituyen la fase de documentación del proyecto y hacen referencia a la elaboración del contenido de esta memoria. Como se puede ver, lo primero que se llevó a cabo fue la elaboración de una introducción y los casos de uso necesarios para el desarrollo de las aplicaciones. Después se retomaron el resto de las partes de la memoria una vez las aplicaciones estaban prácticamente completadas.



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
DOCUMENTACIÓN					
1.Introducción (Motivación y objetivos)		Memoria	José Miguel Cabrera Llamas	20/12/2021	23/12/2021
1.3 Estructura de la memoria		Memoria	Pedro Sánchez Escribano	13/05/2022	14/05/2022
4.Casos de uso		Memoria	José Miguel Cabrera Llamas	22/12/2021	23/12/2021

Figura 6 - Diagrama de Gantt: Documentación 1



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
DOCUMENTACIÓN					
4.Casos de uso		Memoria	José Miguel Cabrera Llamas	22/12/2021	23/12/2021
			Pedro Sánchez Escribano	10/05/2022	12/05/2022
			Álvaro Penalva Alberca	09/05/2022	12/05/2022
3.Tecnología utilizada		Memoria	José Miguel Cabrera Llamas	22/03/2022	25/03/2022
			Álvaro Penalva Alberca	10/05/2022	11/05/2022
2. Estado del arte		Memoria	Pedro Sánchez Escribano	02/05/2022	04/05/2022
			Álvaro Penalva Alberca	07/05/2022	08/05/2022
5.Modelo de datos: E-R		Memoria	Pedro Sánchez Escribano	04/05/2022	07/05/2022
5.Modelo de datos: reositorio y control de versiones		Memoria	José Miguel Cabrera Llamas	03/05/2022	04/05/2022
6.Modelo de arquitectura: Android		Memoria	Víctor Choza Merino	10/05/2022	11/05/2022
6.Modelo de arquitectura: web		Memoria	José Miguel Cabrera Llamas	11/05/2022	12/05/2022
7.Diseño: Android		Memoria	Víctor Choza Merino	11/05/2022	12/05/2022
7.Diseño: web		Memoria	Pedro Sánchez Escribano	08/05/2022	14/05/2022
8.Evalución usabilidad		Memoria	José Miguel Cabrera Llamas	13/05/2022	13/05/2022
			Álvaro Penalva Alberca	16/05/2022	16/05/2022
9.1.Conclusiones		Memoria	José Miguel Cabrera Llamas	08/05/2022	10/05/2022
9.1.Conclusions		Memoria	José Miguel Cabrera Llamas	10/05/2022	12/05/2022
9.2.Trabajo futuro		Memoria	José Miguel Cabrera Llamas	04/05/2022	06/05/2022
			Álvaro Penalva Alberca	14/05/2022	14/05/2022
9.2.Future work		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
10.Guia de Usuario		Memoria	Víctor Choza Merino	14/05/2022	14/05/2022
			Pedro Sánchez Escribano	14/05/2022	15/05/2022
1.1 Motivation		Memoria	José Miguel Cabrera Llamas	12/05/2022	13/05/2022
1.2 Objectives		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
1.3 Memory Structure		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
1.4 Plan de trabajo		Memoria	José Miguel Cabrera Llamas	13/05/2022	15/05/2022

Figura 7 - Diagrama de Gantt: Documentación 2

Chapter 1. Introduction

In this chapter, a brief summary of what ResUniversitas is will be made, the motivations that have led to its development will be discussed, and the objectives to be achieved in the development of the work will be explained.

In short, this section is intended to put the reader in context and place them within the topic that will be covered in this part of the project.

1.1. Motivation

The consequences of the COVID-19 pandemic have been disastrous worldwide. In the last three years, the population has been forced to change the way they interact, and companies, businesses, and universities, among other institutions, have had to adapt to the new regulations established to reduce the spread of the virus.

Due to the epidemiological crisis caused by the COVID-19 virus, and in line with its numerous variants, there has been a revolution in the form of face-to-face meetings, especially in public places such as the University Campus. Specifically, the sanitary measure of the capacity of people has given rise to the creation of a specific need such as the control and limitation of the capacity of the capacity.

On the other hand, and leaving behind the current pandemic situation, and in line with the experience acquired during the university period, it was observed that space management services at the university do not have information available to everyone regarding occupancy, just as they do not have a reservation service in those spaces. In addition, the information of the different events and conferences that take place in the faculties is not shown in a simple, unified, and intuitive way.

On the other hand, leaving behind the current pandemic situation, and in line with the experience acquired during the university period, it was seen that space management services at the university do not have information available to everyone regarding occupation, as well as they do not have a reservation service in these spaces. In addition, the information of the different events and conferences that take place in the faculties are not shown in a simple, unified, and intuitive way.

Following this situation and to solve these limitations, in this project the development of a mobile and web application has been proposed in which, in an orderly and planned manner, and respecting the measures that are established at any time, users can use the facilities that the university makes available.

In this application it is possible to visualize the information referring to the occupation, as well as the reservation of a certain seat through plan view, giving a solution to the problem previously raised.

It is also useful to reserve a seat for a certain class or event, so that if the student knows that they are going to be late for their class, they can reserve a seat in advance and check the occupancy statistics for the duration of the class.

1.2. Objectives

The main objective of this project is the implementation of an Android app that allows its users to reserve a seat in a certain classroom or check the availability of enclosed spaces (from now on they will be called rooms, since there might be classrooms, labs, canteens, etc.), as well as manage the places of the facilities of the university and show events and news to users.

It is also useful to reserve a seat for a certain class or event, so that if the student knows that they will be late for their class, they can reserve a seat in advance, or check the occupancy statistics for the time of the subject.

The specific goals are detailed below:

- Development of a module that has a visualization of the information corresponding to the capacity of the rooms of the universities.
- Establishment of a specific module for users where they are allowed to manage seat reservations and consult the capacity of the rooms.
- The launch of a module dedicated to the management of university facilities, to which only administrative users will have access.
- Design and implementation of a web page to support the android application that allows the necessary procedures to be conducted by the administrator users of buildings or areas.
- The android app and the web app will share the same database and storage repository.

1.3. Memory structure

The chapters that are part of the structure of the document are now briefly described:

Chapter 1: This chapter describes the motivation of the work, the objectives to be achieved, the structure of the document and work planning.

Chapter 2: This chapter details solutions like the one that has been made in the project.

Chapter 3: This chapter describes the technology used in the development of the project.

Chapter 4: This chapter details the actors and specifies the use cases along with their description.

Chapter 5: This chapter details the architecture and patterns used in the Android application as well as in the web application. In addition, describes the models used in the database and repository.

Chapter 6: This chapter describes the main aspects of functionality and design in the Android application as well as in the web application.

Chapter 7: This chapter details the usability survey and analyzes the results obtained in this survey with real users.

Chapter 8: This chapter makes a conclusion and briefly describes future use cases.

Chapter 9: This chapter develops the tasks that the team members have conducted.

Bibliography

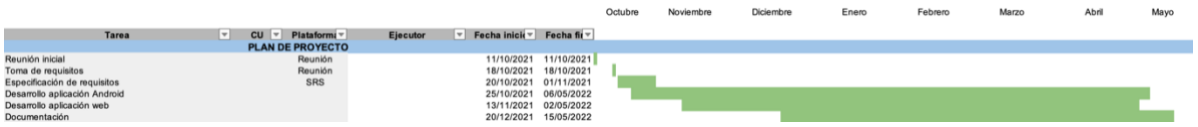
Annex I: User guide

Annex II: Evaluation questions

1.4. Work planning

In this section a summary will be made with the help of a Gantt diagram represented in Figures 1, 2, 3, 4, 5, 6 and 7, in it the development of the project will be exposed over time, dividing the workload in three main blocks: Android application, web application and documentation. It is important to point out that, since the project is made up of four members, work has been done in parallel on the Android application, the web application, and the documentation.

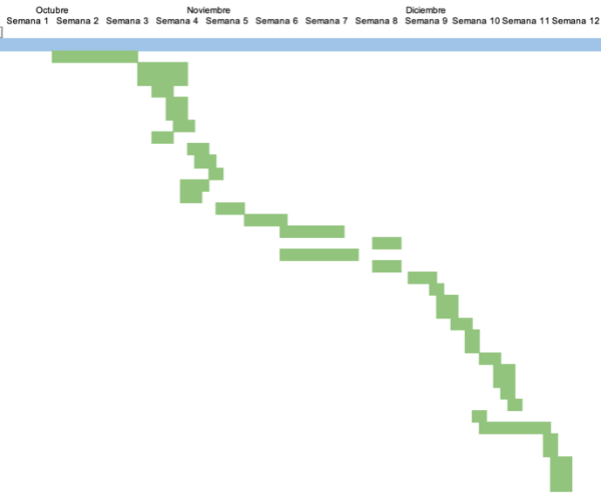
Initially, a meeting was held to determine and clarify the purpose of this project as shown in Figure 1. This same figure also reflects the following meeting in which the user requirements were taken, giving rise to the use cases to subsequently begin the development of the objectives.



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
PLAN DE PROYECTO					
Reunión inicial		Reunión		11/10/2021	11/10/2021
Toma de requisitos		Reunión		18/10/2021	18/10/2021
Especificación de requisitos		SRS		20/10/2021	01/11/2021
Desarrollo aplicación Android				25/10/2021	06/05/2022
Desarrollo aplicación web				13/11/2021	02/05/2022
Documentación				20/12/2021	15/05/2022

Figura 1 - Diagrama de Gantt: planificación genérica

Figures 2 and 3 show all the tasks that have been undertaken in the development of the Android application.



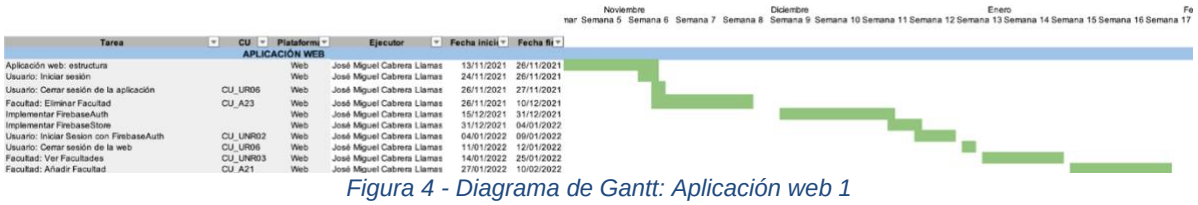
Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN ANDROID					
Aplicación android: estructura		App	Víctor Choza Merino	25/10/2021	05/11/2021
Usuario: Registrar usuario	CU_UNR01	App	Víctor Choza Merino	06/11/2021	12/11/2021
Usuario: Iniciar sesión	CU_UNR02	App	Víctor Choza Merino	06/11/2021	12/11/2021
Facultad: Ver Facultades	CU_UNR03	App	Pedro Sánchez Escribano	08/11/2021	10/11/2021
			Víctor Choza Merino	10/11/2021	12/11/2021
Facultad: Ver información de Facultad	CU_UNR04	App	Pedro Sánchez Escribano	10/11/2021	12/11/2021
			Víctor Choza Merino	11/11/2021	13/11/2021
Facultad: Llamar por teléfono	CU_UNR05	App	Álvaro Penalva Alberca	08/11/2021	10/11/2021
Facultad: Enviar correo electrónico	CU_UNR06	App	Álvaro Penalva Alberca	13/11/2021	15/11/2021
Usuario: Cambiar contraseña	CU_UR01	App	José Miguel Cabrera Llamas	14/11/2021	16/11/2021
			Víctor Choza Merino	16/11/2021	17/11/2021
Facultad: Establecer facultad favorita	CU_UR15	App	Álvaro Penalva Alberca	12/11/2021	15/11/2021
Usuario: Cerrar sesión de la aplicación	CU_UR06	App	Víctor Choza Merino	12/11/2021	14/11/2021
Usuario: Darse de baja	CU_UR09	App	José Miguel Cabrera Llamas	17/11/2021	20/11/2021
Sala: Ver Salas	CU_UNR07	App	Víctor Choza Merino	20/11/2021	26/11/2021
Facultad: Añadir Facultad	CU_A21	App	Pedro Sánchez Escribano	26/11/2021	04/12/2021
			Víctor Choza Merino	09/12/2021	12/12/2021
Facultad: Modificar Facultad	CU_A22	App	Álvaro Penalva Alberca	26/11/2021	06/12/2021
			Víctor Choza Merino	09/12/2021	12/12/2021
Facultad: Eliminar Facultad	CU_A23	App	Álvaro Penalva Alberca	14/12/2021	17/12/2021
			Víctor Choza Merino	17/12/2021	18/12/2021
Sala: Añadir Sala	CU_A06	App	Víctor Choza Merino	18/12/2021	20/12/2021
Sala: Modificar Sala (Editar información de sala)	CU_A05	App	Víctor Choza Merino	18/12/2021	20/12/2021
Sala: Modificar Sala (Crear mapa de sitios)	CU_A07	App	Víctor Choza Merino	20/12/2021	22/12/2021
Sala: Modificar Sala (Modificar mapa de sitios)	CU_A08	App	Víctor Choza Merino	22/12/2021	23/12/2021
			Álvaro Penalva Alberca	22/12/2021	23/12/2021
Reserva: Ver Reservas	CU_UR08	App	Víctor Choza Merino	24/12/2021	26/12/2021
Sala: Dar de baja Sala (Eliminar)	CU_A09	App	Víctor Choza Merino	26/12/2021	28/12/2021
Reserva: Modificar Reserva	CU_UR04	App	Víctor Choza Merino	26/12/2021	28/12/2021
			Pedro Sánchez Escribano	27/12/2021	28/12/2021
Reserva: Eliminar Reserva	CU_UR05	App	Víctor Choza Merino	28/12/2021	29/12/2021
Sitio: Reservar Sitio	CU_UR02	App	Víctor Choza Merino	23/12/2021	24/12/2021
			Álvaro Penalva Alberca	24/12/2021	02/01/2022
Sitio: Activar Sitio	CU_A02	App	Víctor Choza Merino	02/01/2022	03/01/2022
Sitio: Bloquear Sitio	CU_A04	App	Víctor Choza Merino	02/01/2022	03/01/2022
Sala: Ver información Sala	CU_UNR08	App	Víctor Choza Merino	03/01/2022	05/01/2022
Sala: Llamar por teléfono	CU_UNR09	App	Álvaro Penalva Alberca	03/01/2022	05/01/2022
Sala: Enviar correo electrónico	CU_UNR10	App	Álvaro Penalva Alberca	03/01/2022	05/01/2022

Figura 2 - Diagrama de Gantt: Aplicación Android 1

Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN ANDROID					
Tablón: Ver tablón de anuncios	CU_UNR12	App	Víctor Choza Merino	14/02/2022	16/02/2022
			Pedro Sánchez Escribano	15/02/2022	16/02/2022
Tablón: Añadir Evento (con sala)	CU_A10	App	Víctor Choza Merino	17/02/2022	20/02/2022
			Álvaro Penalva Alberca	19/02/2022	21/02/2022
Tablón: Modificar Evento (con sala)	CU_A11	App	Víctor Choza Merino	18/02/2022	20/02/2022
Tablón: Eliminar Evento (con sala)	CU_A12	App	Víctor Choza Merino	18/02/2022	20/02/2022
Tablón: Añadir Anuncio/Noticia	CU_A13	App	Víctor Choza Merino	18/02/2022	20/02/2022
			Pedro Sánchez Escribano	19/02/2022	20/02/2022
Tablón: Modificar Anuncio/Noticia	CU_A14	App	Víctor Choza Merino	20/02/2022	24/02/2022
			José Miguel Cabrera Llamas	23/02/2022	25/02/2022
Tablón: Eliminar Anuncio/Notici	CU_A15	App	Víctor Choza Merino	20/02/2022	24/02/2022
Sitio: Reservar Sitio para eventos	CU_UR03	App	Víctor Choza Merino	24/02/2022	25/02/2022
QR: Crear documento con QR's de Sala (POI)	CU_A19	App	Pedro Sánchez Escribano	05/03/2022	19/03/2022
			Víctor Choza Merino	19/03/2022	20/03/2022
QR: Crear Documento con QR de Evento	CU_A20	App	Pedro Sánchez Escribano	05/03/2022	19/03/2022
			Víctor Choza Merino	19/03/2022	20/03/2022
Sala: Consultar estadísticas de ocupación	CU_UNR11	App	José Miguel Cabrera Llamas	08/03/2022	21/03/2022
QR: Ver información con QR	CU_UNR13	App	Álvaro Penalva Alberca	10/03/2022	27/03/2022
			Víctor Choza Merino	27/03/2022	29/03/2022
QR: Check-in en Sitio con QR	CU_UR07	App	Pedro Sánchez Escribano	10/03/2022	27/03/2022
			Álvaro Penalva Alberca	10/03/2022	27/03/2022
			Víctor Choza Merino	27/03/2022	29/03/2022
Sala: Activar Sala	CU_A01	App	Víctor Choza Merino	01/04/2022	03/04/2022
Notificación: Notificación de reserva	CU_UR14	App	Víctor Choza Merino	04/04/2022	10/04/2022
Notificación: Notificación de Check-In	CU_UR15	App	Víctor Choza Merino	11/04/2022	12/04/2022
Tablón: Ver Información Evento/Noticia	CU_UR16	App	Pedro Sánchez Escribano	10/04/2022	16/04/2022
			Víctor Choza Merino	17/04/2022	19/04/2022
Sala: Notificación de hora concurrida en sala	CU_UR13	App	Pedro Sánchez Escribano	17/04/2022	22/04/2022
			Víctor Choza Merino	23/04/2022	24/04/2022
Facultad: Notificación de entrada a la Facultad	CU_UR10	App	Álvaro Penalva Alberca	26/04/2022	06/05/2022
Sala: Salida de Sala con notificación	CU_UR11	App	Álvaro Penalva Alberca	26/04/2022	06/05/2022

Figura 3 - Diagrama de Gantt: Aplicación Android 2

Next, Figures 4 and 5 show the tasks performed in the development of the web application.



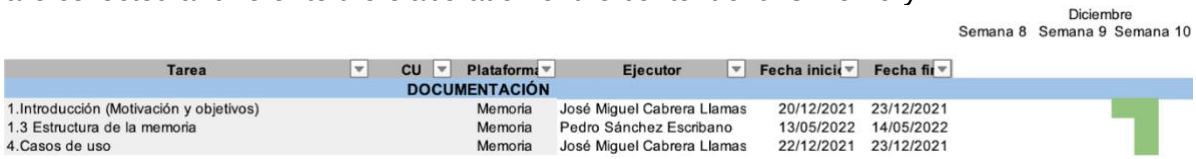
Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN WEB					
Aplicación web: estructura		Web	José Miguel Cabrera Llamas	13/11/2021	26/11/2021
Usuario: Iniciar sesión		Web	José Miguel Cabrera Llamas	24/11/2021	26/11/2021
Usuario: Cerrar sesión de la aplicación	CU_UR06	Web	José Miguel Cabrera Llamas	26/11/2021	27/11/2021
Facultad: Eliminar Facultad	CU_A23	Web	José Miguel Cabrera Llamas	26/11/2021	10/12/2021
Implementar FirebaseAuth		Web	José Miguel Cabrera Llamas	15/12/2021	31/12/2021
Implementar FirebaseStore		Web	José Miguel Cabrera Llamas	31/12/2021	04/01/2022
Usuario: Iniciar Sesion con FirebaseAuth	CU_UNR02	Web	José Miguel Cabrera Llamas	04/01/2022	09/01/2022
Usuario: Cerrar sesión de la web	CU_UR06	Web	José Miguel Cabrera Llamas	11/01/2022	12/01/2022
Facultad: Ver Facultades	CU_UNR03	Web	José Miguel Cabrera Llamas	14/01/2022	25/01/2022
Facultad: Añadir Facultad	CU_A21	Web	José Miguel Cabrera Llamas	27/01/2022	10/02/2022

Figura 4 - Diagrama de Gantt: Aplicación web 1

Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
APLICACIÓN WEB					
Aplicacion web: diseño		Web	Pedro Sánchez Escribano	06/04/2022	01/05/2022
			José Miguel Cabrera Llamas	15/04/2022	02/05/2022
			Pedro Sánchez Escribano	06/04/2022	09/04/2022
Facultad: Modificar Facultad	CU_A22	Web	José Miguel Cabrera Llamas	08/04/2022	28/04/2022
			Pedro Sánchez Escribano	23/04/2022	26/04/2022
Facultad: Eliminar Facultad (con todas sus referencias)	CU_A23	Web	José Miguel Cabrera Llamas	26/04/2022	30/04/2022

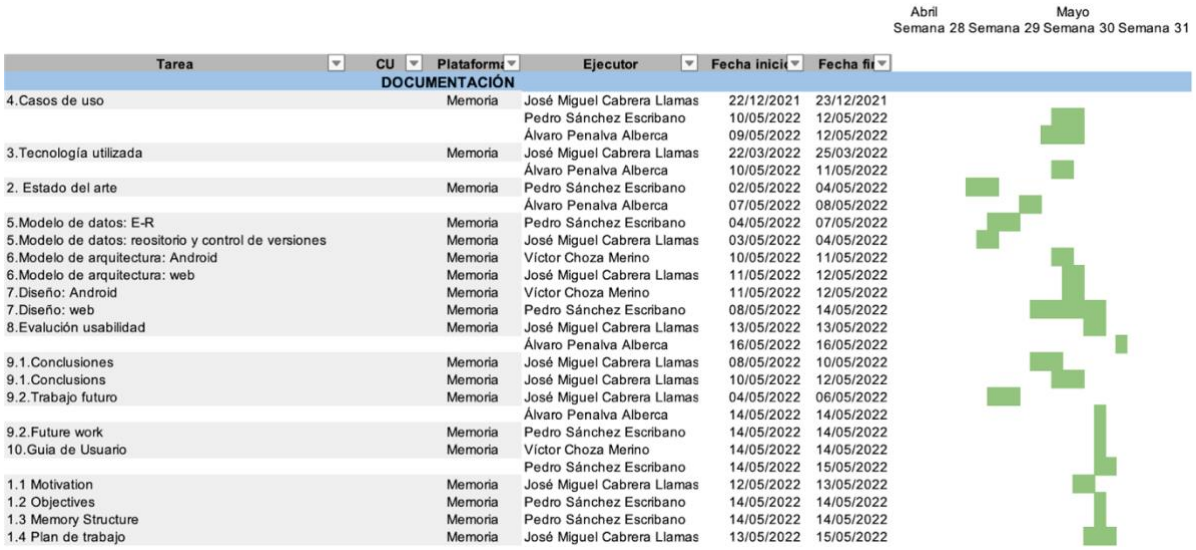
Figura 5 - Diagrama de Gantt: Aplicación web 2

Finally, in Figures 6 and 7, the tasks that constitute the documentation phase of the project are collected and refer to the elaboration of the content of this memory.



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
DOCUMENTACIÓN					
1.Introducción (Motivación y objetivos)		Memoria	José Miguel Cabrera Llamas	20/12/2021	23/12/2021
1.3 Estructura de la memoria		Memoria	Pedro Sánchez Escribano	13/05/2022	14/05/2022
4.Casos de uso		Memoria	José Miguel Cabrera Llamas	22/12/2021	23/12/2021

Figura 6 - Diagrama de Gantt: Documentación 1



Tarea	CU	Plataforma	Ejecutor	Fecha inicio	Fecha fin
DOCUMENTACIÓN					
4.Casos de uso		Memoria	José Miguel Cabrera Llamas	22/12/2021	23/12/2021
			Pedro Sánchez Escribano	10/05/2022	12/05/2022
			Álvaro Penalva Alberca	09/05/2022	12/05/2022
3.Tecnología utilizada		Memoria	José Miguel Cabrera Llamas	22/03/2022	25/03/2022
			Álvaro Penalva Alberca	10/05/2022	11/05/2022
2. Estado del arte		Memoria	Pedro Sánchez Escribano	02/05/2022	04/05/2022
			Álvaro Penalva Alberca	07/05/2022	08/05/2022
5.Modelo de datos: E-R		Memoria	Pedro Sánchez Escribano	04/05/2022	07/05/2022
5.Modelo de datos: reositorio y control de versiones		Memoria	José Miguel Cabrera Llamas	03/05/2022	04/05/2022
6.Modelo de arquitectura: Android		Memoria	Víctor Choza Merino	10/05/2022	11/05/2022
6.Modelo de arquitectura: web		Memoria	José Miguel Cabrera Llamas	11/05/2022	12/05/2022
7.Diseño: Android		Memoria	Víctor Choza Merino	11/05/2022	12/05/2022
7.Diseño: web		Memoria	Pedro Sánchez Escribano	08/05/2022	14/05/2022
8.Evalución usabilidad		Memoria	José Miguel Cabrera Llamas	13/05/2022	13/05/2022
			Álvaro Penalva Alberca	16/05/2022	16/05/2022
9.1.Conclusiones		Memoria	José Miguel Cabrera Llamas	08/05/2022	10/05/2022
9.1.Conclusions		Memoria	José Miguel Cabrera Llamas	10/05/2022	12/05/2022
9.2.Trabajo futuro		Memoria	José Miguel Cabrera Llamas	04/05/2022	06/05/2022
			Álvaro Penalva Alberca	14/05/2022	14/05/2022
9.2.Future work		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
10.Guia de Usuario		Memoria	Víctor Choza Merino	14/05/2022	14/05/2022
			Pedro Sánchez Escribano	14/05/2022	15/05/2022
1.1 Motivation		Memoria	José Miguel Cabrera Llamas	12/05/2022	13/05/2022
1.2 Objectives		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
1.3 Memory Structure		Memoria	Pedro Sánchez Escribano	14/05/2022	14/05/2022
1.4 Plan de trabajo		Memoria	José Miguel Cabrera Llamas	13/05/2022	15/05/2022

Figura 7 - Diagrama de Gantt: Documentación 2

Capítulo 2. Estado del arte

En este capítulo se expone un análisis de algunas de las aplicaciones más similares a la desarrollada en este proyecto.

La aplicación ResUniversitas tiene como funcionalidad controlar el aforo de los espacios cerrados de las universidades, ofreciendo la posibilidad de realizar reservas de sitios en los mismos. Dichos espacios cerrados son todas las estancias en las que un usuario puede tomar asiento, incluyendo la información necesaria de cada sala, en concreto: el nombre, una descripción, una imagen, el horario de apertura, los días festivos o la ocupación prevista y actual.

La información de cada espacio será proporcionada y actualizada por el administrador de la universidad correspondiente.

A continuación, se muestran las aplicaciones existentes más similares a ResUniversitas:

2.1. Aplicaciones similares

2.1.1. Kalena

Kalena ([1]) es una solución software que se centra en la reserva de espacios y recursos corporativos de la empresa, como salas de reuniones, puestos de trabajo y plazas de aparcamiento. Entre sus características principales se encuentran las siguientes:

- Dispone de dos vistas diferentes entre calendario y plano para elegir una reserva como se muestra en la Figura 8.
- Tiene la opción de hacer check-in y check-out manual o mediante código QR para la gestión inmediata de la ocupación de los espacios.
- Posibilidad de consulta de informes y estadísticas de ocupación en tiempo real.
- Se permite mostrar todos los espacios de trabajo a la vez.
- Ofrece la posibilidad de gestionar permisos de usuario sobre los espacios a reservar, de forma que controles quién puede crear, modificar o eliminar reservas de otros usuarios.
- Se permite a los usuarios la gestión de los espacios de trabajo para poder crearlos, modificarlos, o eliminarlos.

Seguidamente, se visualiza la Figura 8, donde se puede ver una captura de pantalla del módulo de reserva mediante vista de plano de la aplicación Kalena.

Figura 8 - Aplicación Kalena

2.1.2. Affluences

Affluences ([2]) es una solución software que se especializa en la medición, previsión, comunicación y gestión de la asistencia en tiempo real: tiempos de espera y tasas de ocupación de espacios.

Actualmente se encuentra disponible en bibliotecas universitarias y públicas, museos, centros deportivos y redes de metro, autobús, tranvía y estaciones.
Como características principales destacan las siguientes:

- Permite la reserva de espacios, consultar ocupación en tiempo real como se muestra en la Figura 9 y revisión de los horarios.
- Dispone de un algoritmo de análisis predictivo que proporciona un pronóstico de afluencia para permitir que los lugares públicos gestionen mejor sus flujos.

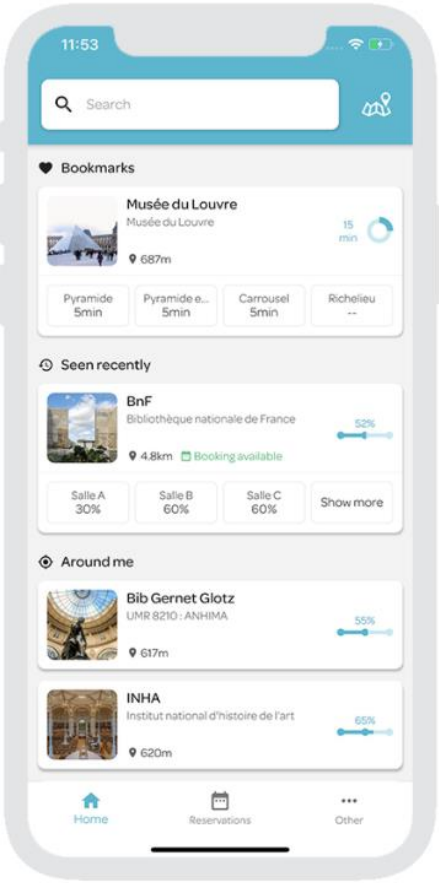


Figura 9 - Aplicación Affluences

2.1.3. Tenea-Talent

Tenea-Talent ([3]) es una solución software que permite la reserva de salas de reuniones, parkings, puestos de trabajo y comedor. Cuenta con un conjunto de características similares a las descritas anteriormente:

- Dispone de vista de calendario o de plano para realizar las reservas.
- Se permite el check-in y el check-out mediante código QR ayudando a tener una visión realista de la afluencia de personas en la oficina.
- Dispone de informes avanzados sobre todas las reservas realizadas por los usuarios, el listado de check-in y check-out.
- Posibilidad de visualizar informes y estadísticas del uso y la ocupación de las oficinas.

2.1.3. Hybo

Hybo ([4]) es un software integral que se centra en la gestión y reserva de espacios de trabajo híbridos como se refleja en la Figura 10. Permite realizar la reserva de escritorios, la

reserva de salas de reuniones o espacios comunes, la reserva de plazas de aparcamiento o la reserva de turnos de comedor de forma centralizada.

Sus características principales son:

- Vista de calendario para realizar las reservas.
- Se permite el check-in/out mediante código QR ayudando a tener una visión realista de la afluencia de personas en la oficina.
- Se permite saber quién estará en la oficina según el día en que se quiere realizar la reserva, es decir posibilita la visión de quién ha reservado un espacio en ese día.
- Posibilidad de crear los mapas de los espacios.
- Referente a la gestión de la seguridad en espacios, como administrador de Hybo, se pueden aplicar normas generales y personalizadas para los trabajadores, asignando roles, creando grupos y ámbitos según su entidad, y se pueden bloquear zonas o priorizar zonas de reserva.

Figura 10 - Página web de Hybo

Capítulo 3. Tecnología utilizada

En este capítulo se van a especificar todas las herramientas tecnológicas utilizadas para la implementación del proyecto.

3.1. Herramientas de desarrollo

3.1.1. Android Studio

Android Studio ([5]): Es el entorno de desarrollo (IDE) oficial elegido por Google para la plataforma Android. JetBrains ([6]) ha publicado el software gratuitamente bajo la licencia Apache 2.0 basándose en el software IntelliJ IDEA ([7]).

El IDE admite varios lenguajes de desarrollo, pero desde hace un par de años Kotlin es el lenguaje recomendado por Google.

Sus principales características son: su editor de diseño de interfaces que permite la creación con Drag & Drop, su emulador virtual de dispositivos móviles que permite realizar pruebas de las aplicaciones desarrolladas y la ayuda de autocompletado de código.

3.1.2. Visual Studio Code

Visual Studio Code ([8]): Es un IDE gratuito y de código abierto que ha desarrollado Microsoft sobre el framework Electron ([9]) para varias plataformas. Incluye una integración con control de versiones y soporte para la depuración. También, dispone de un repositorio central que incluye herramientas para la interpretación de otros lenguajes de programación.

3.2. Lenguajes de programación

3.2.1. Kotlin

Kotlin ([10]): Es un lenguaje de programación desarrollado por JetBrains de tipado estático que puede ser compilado a código fuente de JavaScript ([11]), pero principalmente se ejecuta sobre la máquina virtual de Java.

Actualmente es el lenguaje de programación más utilizado para desarrollar aplicaciones Android.

Una función principal que ayuda a la mejora del software es su interoperabilidad con el lenguaje Java.

3.2.2. TypeScript

TypeScript ([12]): Se caracteriza por ser un lenguaje de programación libre y de código abierto desarrollado por Microsoft. Su núcleo es JavaScript, pero añade los objetos de la programación orientada a objetos (POO) y tipos estáticos.

Al tener el núcleo de JavaScript, permite ejecutarse en el lado del servidor o del cliente indistintamente.

Actualmente, es el lenguaje de programación de primer nivel más utilizado en Microsoft Visual Studio.

3.2.3. HTML5

HTML5 ([13]): Es la última versión del lenguaje de World Wide Web ([14]). Incluye dos variantes de sintaxis HTML clásico y XHTML ([15]) en formato XML que permite poder validarse contra un schema.

Principalmente, en esta versión se incluyen nuevos elementos como en los formularios. También aparecen nuevas etiquetas de audio y video que permiten mostrar contenido multimedia.

3.2.4. CSS

CSS ([16]): Es un lenguaje de Hojas de estilo en cascada para generar e implementar la presentación de documentos web escritos en lenguajes de marcado.

Principalmente, está diseñado para separar el formato del contenido permitiendo un renderizado distinto dependiendo del tamaño de las pantallas. Esta característica posibilita tener una mayor flexibilidad y un mayor control en el diseño de los documentos.

La especificación de este lenguaje es mantenida por el World Wide Web Consortium (W3C) ([17]) el cual proporciona una herramienta de validación de código.

3.3. Frameworks

3.3.1. Angular

Angular ([18]): Es un framework de JavaScript de código abierto escrito en TypeScript y se mantiene por Google para la creación de páginas web. Ha aparecido como una evolución de AngularJS.

Entre sus características se puede destacar que trabaja con el modelo SPA (Single Page Application) generando una aplicación web donde todas las vistas se muestran en la misma página, sin recargar el navegador y la página. Así pues, una división de código en componentes dispone de un renderizado instantáneo con Node.js, siendo un intérprete de comandos que facilita la creación de elementos mediante plantillas.

3.4. Base de datos, repositorio y hosting

3.4.1. Firebase

Firebase ([19]): Es una plataforma almacenada en la nube que mantiene Google en la que se incluyen distintas herramientas para el desarrollo de aplicaciones móviles y web en una SDK o conjunto de herramientas. En este SDK se incluyen distintas librerías para facilitar la implementación en diferentes plataformas. Además, Firebase proporciona al usuario la documentación suficiente para comprender todas las funcionalidades que ofrece.

En este proyecto, los servicios que se han utilizado son los siguientes:

- Firebase Auth ([20]): Es un servicio que provee una gestión de usuarios y un proceso de autenticación a través de un sistema con distintas formas de validación e implementación, como puede ser el uso de proveedores de inicio de sesión externos.
- Cloud Firestore ([21]): Es una base de datos NoSql a la que se facilita su conexión y actualización del contenido en tiempo real entre distintos sistemas. Dispone de un procedimiento en el que se utiliza la caché para realizar actualizaciones en caso de producirse un error de conexión.
- Cloud Storage ([22]): Es un servicio que proporciona almacenamiento de archivos mediante procesos de carga y descarga seguros incluyendo la seguridad de Google junto con la autenticación de la aplicación en su implementación.
- Firebase Hosting ([23]): Es un servicio que provee alojamiento de sitios web con conexiones seguras (SSL), sirviendo contenido dinámico y estático. Actualmente, también permite almacenar microservicios.

En este SDK se incluyen distintas librerías y versiones del conjunto de herramientas para facilitar la implementación y configuración en distintas plataformas. Además, Firebase proporciona al usuario la documentación suficiente para comprender todas las funcionalidades que ofrece.

3.5. Control de versiones

3.5.1. Git

Git ([24]): Git es un sistema de control de versiones distribuido. Esto significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios

locales plenamente funcionales permiten trabajar sin conexión o de forma remota fácilmente. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor.

3.5.2. GitHub

GitHub ([25]): Es un repositorio de desarrollo colaborativo de código mediante un control de versiones que es propiedad de Microsoft. La mayoría de los proyectos almacenados están de manera pública.

3.6. Varios

3.6.1. Código QR

Código QR ([26]): Es una matriz de puntos o un código de barras bidimensional en la que se encuentra información. Su predecesor es el código de barras. Al igual que estos últimos, tiene que escanearse, en este caso desde un lector de códigos QR instalado en un dispositivo móvil o Smartphone.

3.6.2. GPS

GPS ([27]): Con sus siglas procedentes de sistema de posicionamiento global, este consiste en el establecimiento de conexión de un receptor con como mínimo cuatro satélites de este sistema para recibir una señal de ellos y comprobará la hora de emisión de la señal con su hora del sistema para así conocer la distancia aproximada y poder estimar la ubicación en la que se encuentra mediante el método de trilateración inversa ([28]).

El método nombrado anteriormente consiste en el uso de la distancia al satélite para saber que radio alrededor del satélite se debe tener en cuenta para conocer la posición del receptor, al combinarse la señal de varios satélites, el receptor debe encontrarse en la intersección de todos esos radios.

3.6.3. Google Forms

Google Forms ([29]): Es una aplicación web desarrollada por Google que permite la creación y edición de encuestas. Además, se puede consultar el resultado de las encuestas y descargar un informe de estos.

Capítulo 4. Especificación de requisitos

El sistema consta de una aplicación web utilizada para la gestión administrativa y una aplicación Android para el usuario.

A continuación, se describen tanto las características de los usuarios futuros del sistema como los servicios definidos para el desarrollo de la aplicación Android, la aplicación web y los actores del sistema.

4.1. Actores del sistema

Se distinguen tres tipos de actores que interactúan con el sistema para realizar los diferentes casos de uso:

4.1.1. Usuario no registrado

Acceden a la aplicación Android sin identificarse y tienen acceso a las funciones básicas:

- Información sobre salas y facultades: Ocupación en tiempo real, estadísticas de ocupación, eventos y noticias, información de contacto de los espacios.

No tienen acceso a las funciones de la aplicación relacionadas con la reserva de sitios ni la posibilidad de realizar check-in y check-out en los sitios de las salas.

Los usuarios no registrados tienen la opción de registrarse o identificarse como usuario registrado en el caso de disponer de una cuenta (iniciar sesión).

4.1.2. Usuario registrado

Acceden a la aplicación Android identificándose mediante un correo y contraseña.

Tienen acceso tanto a las funciones básicas previamente descritas como funciones de la aplicación relacionadas con la reserva de sitios en sala y la posibilidad de hacer check-in/out en los sitios de sala.

4.1.3. Administrador

Acceden a la aplicación web iniciando sesión mediante un correo y contraseña entregados por el vendedor de la aplicación.

Además de poder acceder a las funciones comentadas para los usuarios anteriores, tienen acceso a funciones relacionadas con la gestión de salas y sitios, así como la creación de mapas:

- Crear, editar y eliminar salas junto a sus mapas de sitios (con información de contacto y otros).
- Crear, editar y eliminar facultades (con información de contacto y otros).
- Crear, editar y eliminar noticias y eventos en sala.

4.2. Diagrama de casos de uso

En este apartado se incluye un diagrama de casos de uso por cada perfil de usuario que accede a la aplicación. Los diagramas están reflejados en las Figuras 11, 12 y 13 para los usuarios no registrados, usuarios registrados y usuarios administradores respectivamente.

4.2.1. Usuario no registrado

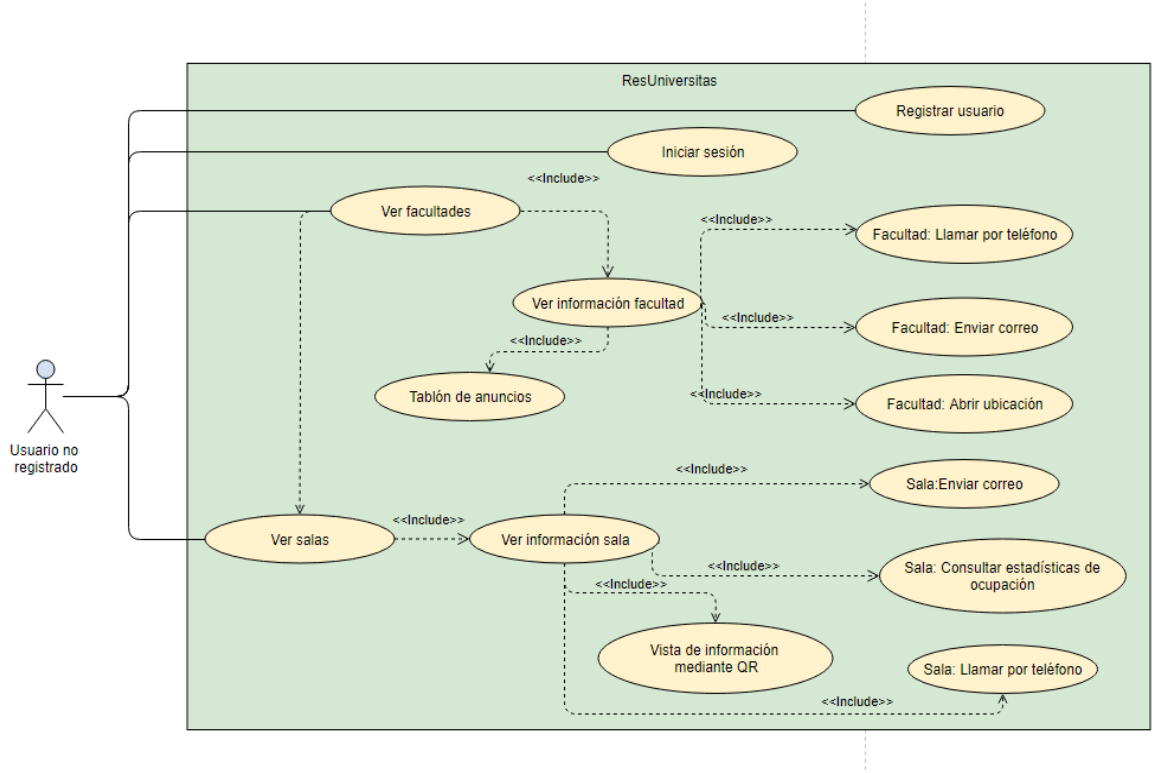


Figura 11 - Diagrama de CU de usuario no registrado

4.2.2. Usuario registrado

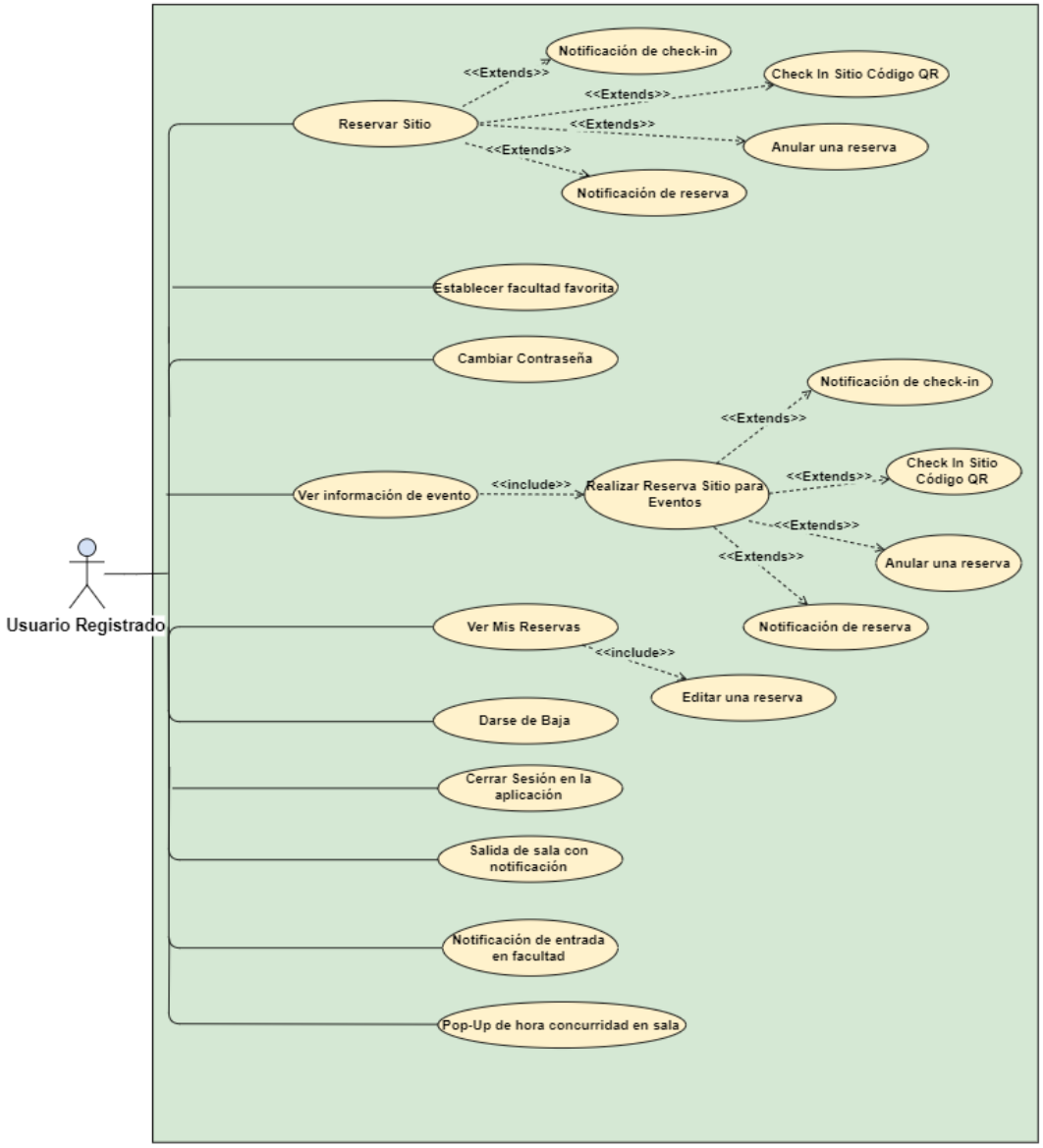


Figura 12 - Diagrama de CU de usuario registrado

4.2.3. Administrador

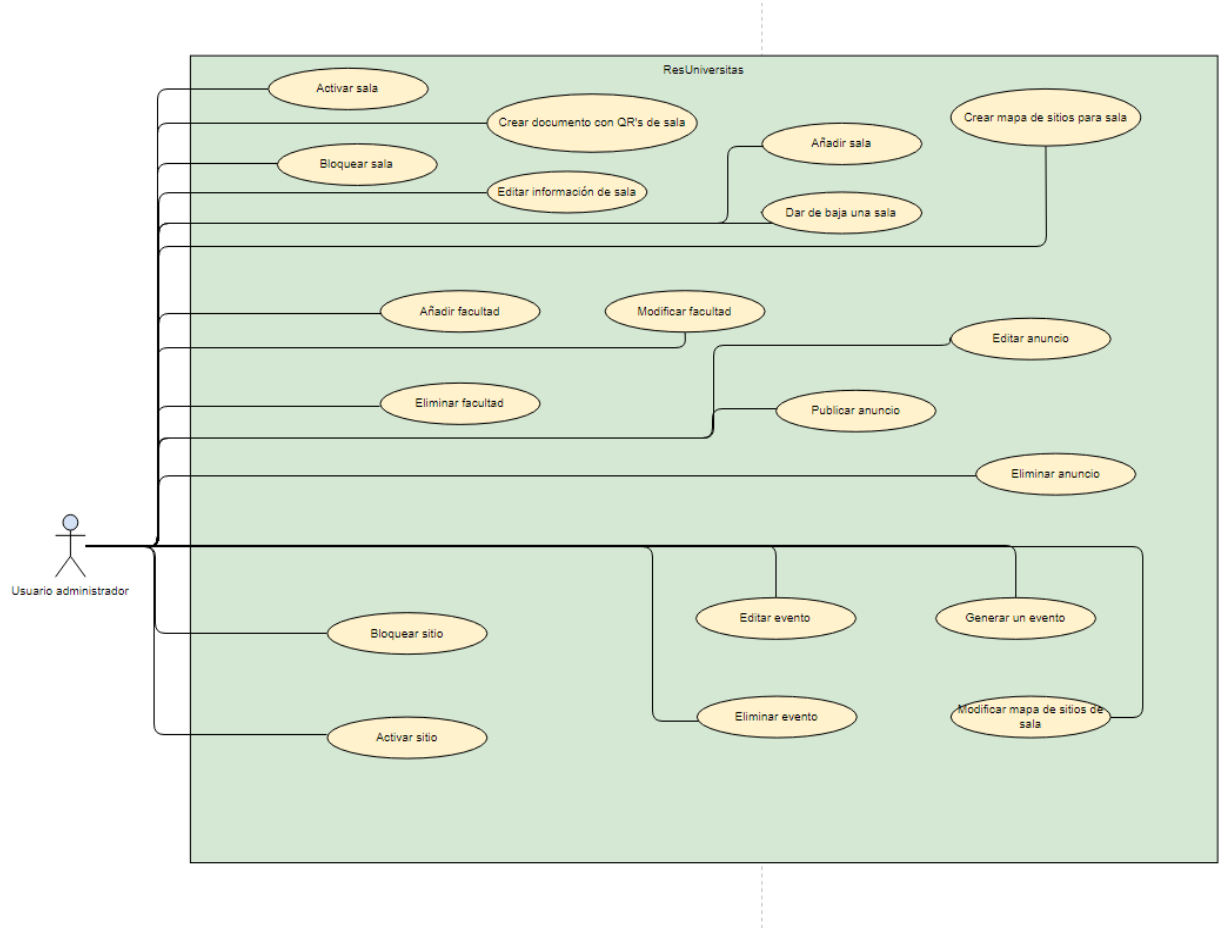


Figura 13 - Diagrama de CU de usuario administrador

4.3. Casos de uso

En este apartado se especifican los casos de uso que se han llevado a cabo en el desarrollo de las aplicaciones, haciendo referencia a los usuarios no registrados desde la Tabla 1 hasta la Tabla 13, a los usuarios registrados desde la Tabla 14 hasta la Tabla 20 y, por último, a los usuarios administradores desde la Tabla 30 hasta la Tabla 49.

4.3.1. Casos de uso de usuario no registrado

Detalles	Descripción
Identificador	CU_UNR01
Nombre	Registrar usuario
Prioridad	Alta
Estabilidad	Alta
Descripción	Añadir un nuevo usuario a la base de datos compuesto por un correo electrónico, un nombre y una contraseña
Entrada	Correo electrónico, nombre, contraseña, fecha del sistema
Salida	Se abre la aplicación con la sesión del usuario y su información
Origen	GUI
Destino	GUI
Necesita	Correo electrónico, nombre y contraseña
Acción	Se añade a la base de datos el nuevo usuario y se inicia sesión con las credenciales introducidas
Precondición	El correo electrónico introducido no existe en la base de datos y es válido, y la contraseña es de al menos 6 caracteres
Postcondición	En la aplicación se inicia la sesión del usuario y la base de datos contiene el nuevo usuario
Efectos laterales	Se muestra un mensaje de error en caso de que el correo electrónico ya exista en la base de datos o no sea válido, y en caso de que la contraseña tenga menos de 6 caracteres. Se muestra un mensaje de error en caso de producirse alguno

Tabla 1 - CU usuario no registrado: Registrar usuario

Detalles	Descripción
Identificador	CU_UNR02
Nombre	Iniciar sesión
Prioridad	Alta
Estabilidad	Alta
Descripción	Se inicia sesión en la aplicación con las credenciales introducidas
Entrada	Correo electrónico y contraseña
Salida	Se inicia sesión en la aplicación
Origen	GUI
Destino	GUI
Necesita	Correo electrónico y contraseña
Acción	Se inicia sesión con las credenciales introducidas mostrando la pantalla de inicio
Precondición	El correo electrónico introducido existe en la base de datos y la contraseña coincide con la que está asociada al correo electrónico en la base de datos
Postcondición	Se inicia la sesión del usuario en la aplicación
Efectos laterales	Muestra un mensaje de error en caso de que el correo electrónico no exista en la base de datos y en caso de que la contraseña introducida no coincida con la contraseña almacenada asociada al correo electrónico. Se muestra un mensaje de error en caso de producirse alguno

Tabla 2 - CU usuario no registrado: Iniciar sesión

Detalles	Descripción
Identificador	CU_UNR03
Nombre	Ver facultades
Prioridad	Baja
Estabilidad	Media
Descripción	Ver un listado de las facultades existentes. Al hacer clic en una facultad se muestra el listado de las salas de dicha facultad. Se muestra un botón para ver la información de la facultad
Entrada	N/A
Salida	Se muestra por pantalla el listado de las facultades
Origen	GUI
Destino	GUI
Necesita	N/A
Acción	Se extraen de la base de datos las facultades existentes y se muestran por pantalla
Precondición	Existe al menos una facultad en la base de datos
Postcondición	El listado de facultades se muestra en la pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 3 - CU usuario no registrado: Ver facultades

Detalles	Descripción
Identificador	CU_UNR04
Nombre	Ver información de facultad
Prioridad	Baja
Estabilidad	Media
Descripción	Ver la información de una facultad, como su teléfono de contacto y correo electrónico en caso de existir (se muestran con un botón para llamar y/o enviar correo electrónico)
Entrada	idFacultad
Salida	Se muestra por pantalla la información de la facultad
Origen	GUI
Destino	GUI
Necesita	Facultad
Acción	Se extrae de la base de datos la información de la facultad consultada y se muestra en la pantalla
Precondición	La facultad existe en la base de datos
Postcondición	La información se muestra en la pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 4 - CU usuario no registrado: Ver información de facultad

Detalles	Descripción
Identificador	CU_UNR05
Nombre	Facultad: Llamar por teléfono
Prioridad	Baja
Estabilidad	Baja
Descripción	Abrir la aplicación de Teléfono si hay un número de teléfono disponible en la facultad consultada
Entrada	idFacultad
Salida	Se abre la aplicación de Teléfono con el número de la facultad consultada marcado
Origen	GUI
Destino	Aplicación Teléfono
Necesita	Facultad
Acción	Se abre la aplicación de Teléfono pasándole como parámetro el número de teléfono extraído de la base de datos correspondiente a la facultad consultada
Precondición	La facultad existe en la base de datos y tiene un número de teléfono asignado
Postcondición	Se pasan los parámetros a la aplicación Teléfono
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 5 - CU usuario no registrado: Facultad, llamar por teléfono

Detalles	Descripción
Identificador	CU_UNR06
Nombre	Facultad: Enviar correo electrónico
Prioridad	Baja
Estabilidad	Baja
Descripción	Abrir la aplicación de Correo electrónico si hay una dirección de correo electrónico disponible en la facultad consultada
Entrada	idFacultad
Salida	Se abre la aplicación de Correo electrónico con la dirección de correo electrónico de la facultad consultada
Origen	GUI
Destino	Aplicación Correo electrónico
Necesita	Facultad
Acción	Se abre la aplicación de Correo electrónico pasándole como parámetro la dirección de correo electrónico extraída de la base de datos correspondiente a la facultad consultada
Precondición	La sala existe en la base de datos y tiene una dirección de correo electrónico asignada
Postcondición	Se pasan los parámetros a la aplicación de Correo electrónico
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 6 - CU usuario no registrado: Facultad, enviar correo electrónico

Detalles	Descripción
Identificador	CU_UNR07
Nombre	Ver salas
Prioridad	Alta
Estabilidad	Alta
Descripción	Ver un listado de las salas correspondientes a la facultad consultada. En el listado se puede ver la ocupación actual de cada sala y acciones (botones): Uno para más información, otro para hacer el check-in (leyendo un código QR) y el último para reservar (utilizando el mapa de asientos de una sala). Estos dos últimos botones sólo están disponibles para un usuario registrado
Entrada	idFacultad, idSala, y fecha del sistema idUsuario (Opcional)
Salida	Se muestra por pantalla el listado de las salas con la ocupación y sus opciones disponibles
Origen	GUI
Destino	GUI
Necesita	Facultad, sala, y fecha del sistema Usuario (Opcional)
Acción	Se extrae de la base de datos las salas de la facultad consultada y su información
Precondición	La facultad existe en la base de datos
Postcondición	El listado de salas se muestra en la pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 7 - CU usuario no registrado: Ver salas

Detalles	Descripción
Identificador	CU_UNR08
Nombre	Ver información de sala
Prioridad	Media
Estabilidad	Alta
Descripción	Ver la información de una sala, como su capacidad, características, aforo, ocupación actual, teléfono de contacto y correo electrónico en caso de tenerlos (Botón para llamar y/o enviar correo electrónico). También se muestran 2 botones: uno para hacer el check-in (Con código QR) y otro para reservar (Mapa de asientos de sala). Estos dos últimos botones sólo están disponibles para un usuario registrado
Entrada	idSala, y fecha del sistema idUsuario (Opcional)
Salida	Se muestra por pantalla la información de la sala
Origen	GUI
Destino	GUI
Necesita	Sala, y fecha del sistema Usuario (Opcional)
Acción	Se extrae de la base de datos la información de la sala consultada y se muestra por pantalla
Precondición	La sala existe en la base de datos
Postcondición	La información se muestra en la pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 8 - CU usuario no registrado: Ver información de sala

Detalles	Descripción
Identificador	CU_UNR09
Nombre	Sala: Llamar por teléfono
Prioridad	Baja
Estabilidad	Baja
Descripción	Abrir la aplicación de Teléfono si hay un número de teléfono disponible en la sala consultada
Entrada	idSala
Salida	Se abre la aplicación de Teléfono con el número de la sala consultada
Origen	GUI
Destino	Aplicación Teléfono
Necesita	Sala
Acción	Se abre la aplicación de Teléfono pasándole como parámetro el número de teléfono extraído de la base de datos correspondiente a la sala consultada
Precondición	La sala existe en la base de datos y tiene un número de teléfono asignado
Postcondición	Se pasan los parámetros a la aplicación Teléfono
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 9 - CU usuario no registrado: Sala, llamar por teléfono

Detalles	Descripción
Identificador	CU_UNR10
Nombre	Sala: Enviar correo electrónico
Prioridad	Baja
Estabilidad	Baja
Descripción	Abrir la aplicación de Correo electrónico si hay una dirección de correo electrónico disponible en la sala consultada
Entrada	idSala
Salida	Se abre la aplicación de Correo electrónico con la dirección de correo electrónico de la sala consultada
Origen	GUI
Destino	Aplicación Correo electrónico
Necesita	Sala
Acción	Se abre la aplicación de Correo electrónico pasándole como parámetro la dirección de correo electrónico extraída de la base de datos correspondiente a la sala consultada
Precondición	La sala existe en la base de datos y tiene una dirección de correo electrónico asignada
Postcondición	Se pasan los parámetros a la aplicación de Correo electrónico
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 10 - CU usuario no registrado: Sala, enviar correo electrónico

Detalles	Descripción
Identificador	CU_UNR11
Nombre	Sala: Consultar estadísticas de ocupación
Prioridad	Media
Estabilidad	Alta
Descripción	Se muestra un histórico de la ocupación en la sala en fechas pasadas y una predicción de ocupación junto a la cantidad de reservas confirmadas para fechas futuras. En la pantalla se mostrará la ocupación por horas del día elegido
Entrada	idSala y fecha seleccionada
Salida	Se muestra por pantalla la información de ocupación del día elegido
Origen	GUI
Destino	GUI
Necesita	Sala y fecha
Acción	Se extrae de la base de datos la ocupación de una fecha pasada o futura y en caso de ser futura se calcula con datos almacenados
Precondición	La sala existe en la base de datos y tiene información de ocupación de fechas pasadas
Postcondición	La información se muestra en pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 11 - CU usuario no registrado: Sala, consultar estadísticas de ocupación

Detalles	Descripción
Identificador	CU_UNR12
Nombre	Consultar tablón de anuncios
Prioridad	Baja
Estabilidad	Alta
Descripción	Ver una lista de anuncios de una facultad publicados por un administrador, o notificaciones de cierre de sala, eventos próximos u otros
Entrada	idFacultad
Salida	Se muestra por pantalla la lista de anuncios y notificaciones de la facultad
Origen	GUI
Destino	GUI
Necesita	Facultad
Acción	Se extrae de la base de datos la lista de anuncios y notificaciones de la facultad
Precondición	La facultad existe en la base de datos
Postcondición	La información se muestra en la pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 12 - CU usuario no registrado: Consultar tablón de anuncios

Detalles	Descripción
Identificador	CU_UNR13
Nombre	Vista de información mediante QR
Prioridad	Baja
Estabilidad	Alta
Descripción	El usuario puede leer un código QR de distintos tipos, ya sea para acceder a la información de una facultad de una sala para observar su ocupación, de un evento, el lector de códigos QR le llevará a su vista de información correspondiente
Entrada	String identificativo del QR entre: idSala, idEvento, idFacultad.
Salida	Vista de información del elemento.
Origen	Externo
Destino	GUI
Necesita	QR, Permiso para acceder a cámara.
Acción	Se muestra la información sobre lo que corresponda al QR ya sea evento, facultad o sala.
Precondición	La sesión está activa y el QR pertenece a la aplicación
Postcondición	N/A
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 13 - CU usuario no registrado: Vista de información mediante QR

4.3.2. Casos de uso de usuario registrado

Detalles	Descripción
Identificador	CU_UR01
Nombre	Cambiar contraseña
Prioridad	Alta
Estabilidad	Alta
Descripción	Una vez introducida la contraseña anterior del usuario para confirmar su identidad, se guarda la nueva contraseña introducida en la base de datos encriptada
Entrada	idUsuario, contraseña anterior y contraseña nueva
Salida	Se muestra un mensaje confirmando el cambio de contraseña
Origen	GUI
Destino	GUI
Necesita	Usuario, contraseña anterior y contraseña nueva
Acción	Se compara la contraseña anterior introducida por el usuario con la almacenada en la base de datos. Si esta es correcta, se valida la nueva contraseña y se almacena encriptada en la base de datos si cumple los criterios
Precondición	El usuario tiene la sesión iniciada y ha introducido la contraseña anterior correctamente. La nueva contraseña no debe ser igual a la anterior y cumplir los criterios de contraseña
Postcondición	Se guarda la nueva contraseña encriptada en la base de datos y se muestra un mensaje al usuario de que el cambio de contraseña se ha realizado correctamente

Efectos laterales	Muestra un mensaje de error en caso de que el usuario no haya introducido la contraseña anterior correctamente o si la nueva contraseña es igual que la anterior. Se muestra un mensaje de error en caso de producirse alguno

Tabla 14 - CU usuario registrado: Cambiar contraseña

Detalles	Descripción
Identificador	CU_UR02
Nombre	Reservar sitio
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario realiza una reserva en el sitio de la sala y en la fecha seleccionados durante la/s hora/s que haya elegido el sitio. La disponibilidad de la sala disminuye en dicha fecha y hora/s.
Entrada	idUsuario, idSala, idSitio, fecha y hora/s
Salida	Se muestra un mensaje de confirmación al usuario
Origen	GUI
Destino	GUI
Necesita	Usuario, sala, sitio, fecha y hora
Acción	Se añade a la base de datos la reserva del usuario en el sitio de la sala durante la fecha y hora/s elegidas. La disponibilidad disminuye en una unidad para estos mismos parámetros
Precondición	La sala existe, está activa y el sitio está libre para la fecha y hora/s seleccionadas
Postcondición	La reserva se almacena en la base de datos y se muestra un mensaje al usuario de que la reserva se ha realizado correctamente
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 15 - CU usuario registrado: Revisar sitio

Detalles	Descripción
Identificador	CU_UR03
Nombre	Realizar reserva de sitio para eventos
Prioridad	Media
Estabilidad	Alta
Descripción	Se realiza la reserva de un sitio disponible en una sala
Entrada	idUsuario, idEvento, idSitio
Salida	Notificación con la confirmación de la reserva de la sala
Origen	GUI
Destino	GUI
Necesita	Usuario, evento, sitio
Acción	Se crea un nuevo registro de reserva en el sistema asignándole un ID único
Precondición	El sitio que se quiere reservar está libre
Postcondición	La reserva del sitio en el evento queda registrada
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 16 - CU usuario registrado: Realizar reserva de sitio para eventos

Detalles	Descripción
Identificador	CU_UR04
Nombre	Editar una reserva
Prioridad	Alta
Estabilidad	Alta
Descripción	Se realiza la modificación de una reserva
Entrada	idUsuario, idReserva, idSitio, día, hora
Salida	Notificación con la confirmación de la modificación de la reserva
Origen	GUI
Destino	GUI
Necesita	Usuario, reserva, sitio, fecha
Acción	Se actualiza el registro de la reserva en el sistema
Precondición	La sala no tiene el aforo completo en el evento/horario nuevo
Postcondición	Los datos de la reserva quedan actualizados
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 17 - CU usuario registrado: Editar una reserva

Detalles	Descripción
Identificador	CU_UR05
Nombre	Anular una reserva
Prioridad	Alta
Estabilidad	Alta
Descripción	Se realiza la anulación de una reserva
Entrada	idReserva
Salida	Notificación con la confirmación de la anulación de la reserva
Origen	GUI
Destino	GUI
Necesita	Reserva
Acción	Se da de baja el registro de la reserva en el sistema
Precondición	La reserva está activa
Postcondición	No existe la reserva en la sala y evento/horario por cuenta de correo. El aforo de la sala en el evento/horario seleccionado no está completo
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 18 - CU usuario registrado: Anular una reserva

Detalles	Descripción
Identificador	CU_UR06
Nombre	Cerrar sesión en la aplicación
Prioridad	Alta
Estabilidad	Alta
Descripción	Se realiza un cierre de sesión en la aplicación
Entrada	idUsuario
Salida	Mensaje con la confirmación del cierre de sesión mostrado en la pantalla principal de la aplicación
Origen	GUI
Destino	GUI
Necesita	Usuario
Acción	Se cierra la sesión en la aplicación
Precondición	La sesión está activa
Postcondición	No una sesión activa del usuario
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 19 - CU usuario registrado: Cerrar sesión en la aplicación

Detalles	Descripción
Identificador	CU_UR07
Nombre	Check-In Sitio con código QR
Prioridad	Alta
Estabilidad	Alta
Descripción	Se registra el acceso al sitio de una sala una mediante la lectura de un código QR que identifica dicho sitio
Entrada	idUsuario, idSala, idSitio
Salida	Mensaje de aviso de entrada correcta.
Origen	Externo
Destino	GUI
Necesita	Usuario, sala, sitio
Acción	Se realiza un registro al sitio de una sala mediante la lectura código QR
Precondición	La sesión está activa, el código QR corresponde a un sitio libre o reservado por el propio usuario
Postcondición	Disminuye en uno la disponibilidad de la sala y el usuario queda registrado en el sitio de la sala
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 20 - CU usuario registrado: Check-In Sitio con código QR

Detalles	Descripción
Identificador	CU_UR08
Nombre	Ver mis reservas
Prioridad	Alta
Estabilidad	Alta
Descripción	Se muestran todas las reservas realizadas
Entrada	idUsuario, fecha del sistema
Salida	Información de las próximas reservas y las ya disfrutadas
Origen	GUI
Destino	GUI
Necesita	Usuario
Acción	Se muestran las reservas pendientes y las disfrutadas ofreciendo la opción de filtrar
Precondición	La sesión está activa y el usuario ha efectuado alguna reserva
Postcondición	N/A
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno. Si el sitio está reservado, se muestra un aviso de que no puede ocuparse

Tabla 21 - CU usuario registrado: Ver mis reservas

Detalles	Descripción
Identificador	CU_UR09
Nombre	Darse de Baja
Prioridad	Media
Estabilidad	Alta
Descripción	Se elimina el usuario.
Entrada	idUsuario
Salida	Mensaje de confirmación del proceso de baja
Origen	GUI
Destino	GUI
Necesita	Usuario
Acción	Se da de baja el usuario en la aplicación
Precondición	La sesión está activa
Postcondición	Usuario dado de baja del sistema
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 22 - CU usuario registrado: Darse de baja

Detalles	Descripción
Identificador	CU_UR10
Nombre	Notificación de entrada en facultad
Prioridad	Media
Estabilidad	Alta
Descripción	El sistema notifica al usuario cuando acceda al perímetro de la facultad favorita.
Entrada	Externo
Salida	Notificación de entrada en facultad
Origen	Sistema
Destino	GUI
Necesita	Facultad, ubicación
Acción	El sistema muestra una notificación al usuario de que accede a una facultad.
Precondición	La ubicación está activa y la ubicación del usuario está dentro del perímetro de una facultad
Postcondición	Se manda una notificación al sistema de notificaciones.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 23 - CU usuario registrado: Notificación de entrada en facultad

Detalles	Descripción
Identificador	CU_UR11
Nombre	Salida de sala (Con notificación)
Prioridad	Alta
Estabilidad	Alta
Descripción	El sistema comprueba si el usuario ha abandonado el radio de ubicación de una sala y le manda una notificación para que realice el check-out de la sala manualmente.
Entrada	Ubicación
Salida	Mensaje de notificación de salida de sala
Origen	Externo
Destino	GUI
Necesita	Usuario, sala, sitio, ubicación
Acción	El sistema notifica al usuario en caso de que salga del radio de la sala.
Precondición	La ubicación está activa, el usuario ha hecho Check-In en el sitio de una sala y su ubicación está fuera del perímetro de dicha sala
Postcondición	Se manda una notificación al sistema de notificaciones.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 24 - CU usuario registrado: Salida de sala (Con notificación)

Detalles	Descripción
Identificador	CU_UR12
Nombre	Pop-up de hora concurrida en sala
Prioridad	Media
Estabilidad	Alta
Descripción	El sistema notifica al usuario cuando el horario de mayor concurrencia de la sala en la que se encuentra se acerca
Entrada	idUsuario, idSala
Salida	Notificación de aviso de hora concurrida
Origen	Externo
Destino	GUI
Necesita	Usuario, Sala
Acción	El sistema muestra una notificación al usuario informando de la llegada de una hora concurrida
Precondición	La sesión está activa
Postcondición	N/A
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 25 - CU usuario registrado: Pop-up de hora concurrida en sala

Detalles	Descripción
Identificador	CU_UR13
Nombre	Establecer facultad favorita
Prioridad	Media
Estabilidad	Alta
Descripción	El usuario puede establecer una facultad como favorita. De este modo al entrar en la app lo primero que le aparecerá es la lista de salas de esa facultad.
Entrada	idUsuario, idFacultad
Salida	Mensaje de confirmación
Origen	GUI
Destino	GUI
Necesita	Usuario, facultad
Acción	Se establece la facultad favorita del usuario
Precondición	La sesión está activa y la facultad existe
Postcondición	Queda registrado en la base de datos la facultad favorita del usuario
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 26 - CU usuario registrado: Establecer facultad favorita

Detalles	Descripción
Identificador	CU_UR14
Nombre	Notificación de reserva
Prioridad	Media
Estabilidad	Alta
Descripción	El usuario recibe una notificación cuando falten 10 minutos para la hora a la que ha reservado.
Entrada	idReserva
Salida	Notificación de reserva cercana
Origen	Externo
Destino	GUI
Necesita	idReserva
Acción	Se genera una notificación hacia el usuario creada en el momento de realizar la reserva programada previamente.
Precondición	La reserva existe.
Postcondición	Se manda una notificación al sistema de notificaciones
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 27 - CU usuario registrado: Notificación de reserva

Detalles	Descripción
Identificador	CU_UR15
Nombre	Notificación de check-in
Prioridad	Media
Estabilidad	Alta
Descripción	El usuario a la hora de realizar check-in en algún sitio recibirá una notificación por parte del sistema de que el check-in se ha realizado correctamente.
Entrada	N/A
Salida	Notificación de confirmación de check-in
Origen	Sistema
Destino	GUI
Necesita	IdReserva
Acción	Se envía una notificación al sistema de notificaciones.
Precondición	El check-in es correcto.
Postcondición	La notificación se muestra al usuario
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 28 - CU usuario registrado: Notificación de check-in

Detalles	Descripción
Identificador	CU_UR16
Nombre	Ver información evento
Prioridad	Media
Estabilidad	Alta
Descripción	Ver la información completa de un evento, cuando ocurre en qué sala sucede, y el nombre del evento.
Entrada	idEvento
Salida	Se muestra por pantalla la información del evento
Origen	GUI
Destino	GUI
Necesita	idEvento
Acción	Se extrae de la BBDD toda la información del evento.
Precondición	El evento existe en la BBDD.
Postcondición	La información se muestra en pantalla
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 29 - CU usuario registrado: Ver información evento

4.3.3. Casos de uso de usuario administrador

Detalles	Descripción
Identificador	CU_A01
Nombre	Activar Sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se activa una sala que estaba bloqueada
Entrada	idSala
Salida	Mensaje de confirmación de activación
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador del lugar
Acción	Se actualiza la sala en la base de datos como activo
Precondición	La sala debe estar inactiva
Postcondición	La sala se encuentra activa
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno.

Tabla 30 - CU usuario administrador: Activar Sala

Detalles	Descripción
Identificador	CU_A02
Nombre	Activar Sitio
Prioridad	Alta
Estabilidad	Alta
Descripción	Se activa un sitio no disponible en una sala
Entrada	idSitio, idSala
Salida	Mensaje de confirmación de activación
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador del lugar
Acción	Se actualiza el sitio en la base de datos como activo
Precondición	El sitio debe estar inactivo
Postcondición	El sitio se encuentra activo y puede ser reservado mientras la sala esté activa
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno.

Tabla 31 - CU usuario administrador: Activar sitio

Detalles	Descripción
Identificador	CU_A03
Nombre	Bloquear Sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se bloquea toda interacción sobre una sala para reservas y check-in
Entrada	IdSala
Salida	Mensaje de confirmación de bloqueo
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador del lugar
Acción	Se actualiza la sala en la base de datos como bloqueada
Precondición	La sala debe estar desbloqueada
Postcondición	El sitio se encuentra activo y puede ser reservado mientras la sala esté activa.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 32 - CU usuario administrador: Bloquear sala

Detalles	Descripción
Identificador	CU_A04
Nombre	Bloquear Sitio
Prioridad	Alta
Estabilidad	Alta
Descripción	Se bloquea un sitio en una sala para que este no pueda ser reservado ni se permite el check-in en el mismo
Entrada	idSitio, idSala
Salida	Mensaje de confirmación de bloqueo
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador del lugar
Acción	Se actualiza el sitio en la base de datos como inactivo
Precondición	El sitio debe estar activo
Postcondición	El sitio se bloquea y no puede ser reservado
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 33 - CU usuario administrador: Bloquear sitio

Detalles	Descripción
Identificador	CU_A05
Nombre	Editar información de sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se modifica la información de una sala
Entrada	Teléfono, correo electrónico, aforo
Salida	Se muestra la nueva información
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador del lugar
Acción	Se modifican los datos de una sala ya sea teléfono o sus características en la base de datos
Precondición	La sala debe estar activa
Postcondición	Los nuevos datos mostrados están actualizados
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno o no se introduzca el tipo de datos adecuado

Tabla 34 - CU usuario administrador: Editar información de sala

Detalles	Descripción
Identificador	CU_A06
Nombre	Añadir sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se genera una nueva sala con su información como el teléfono, correo electrónico, nombre, ubicación y perímetro
Entrada	Nombre de sala, ubicación y perímetro Teléfono y correo electrónico (Opcional)
Salida	Mensaje de confirmación de creación de sala
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador
Acción	Se da de alta una sala con sus características en la base de datos
Precondición	La sala no debe existir previamente
Postcondición	Todos los datos de la sala quedan registrados
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno o no se introduzca el tipo de datos adecuado

Tabla 35 - CU usuario administrador: Añadir sala

Detalles	Descripción
Identificador	CU_A07
Nombre	Crear mapa de sitios para sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se crea un mapa de sitios con las características y disposición definidas por el usuario
Entrada	Parámetros y disposición de los asientos, idSala
Salida	Mensaje de confirmación de creación de mapa de sitios
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador y sala
Acción	Se crea un mapa de sitios para una sala
Precondición	La sala debe existir previamente y no debe tener un mapa de sitios asociado
Postcondición	El mapa de sitios queda registrado y asociado a la sala
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 36 - CU usuario administrador: Crear mapa de sitios para sala

Detalles	Descripción
Identificador	CU_A08
Nombre	Modificar mapa de sitios de sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se modifica un mapa de sitios con las características y disposición definidas por el usuario
Entrada	Parámetros y disposición de los asientos, idSala
Salida	Mensaje de confirmación de modificación de mapa de sitios
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador y sala
Acción	Se modifica el mapa de sitios de una sala
Precondición	La sala debe existir previamente y debe tener un mapa de sitios asociado
Postcondición	El mapa de sitios asociado a la sala queda actualizado
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 37 - CU usuario administrador: Modificar mapa de sitios de sala

Detalles	Descripción
Identificador	CU_A09
Nombre	Dar de baja una sala
Prioridad	Alta
Estabilidad	Alta
Descripción	Se da de baja una sala
Entrada	idSala
Salida	Mensaje de confirmación de sala dada de baja
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador
Acción	Se elimina una sala de la base de datos
Precondición	La sala debe existir previamente
Postcondición	La sala se da de baja y no se podrá interactuar más con ella
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 38 - CU usuario administrador: Dar de baja una sala

Detalles	Descripción
Identificador	CU_A10
Nombre	Generar evento
Prioridad	Alta
Estabilidad	Alta
Descripción	Se genera un evento reservando por completo una sala para el evento
Entrada	Nombre de evento, fecha del evento, teléfono de información, correo electrónico e idSala
Salida	Mensaje de confirmación de generación de evento
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador
Acción	Se activa un evento sobre una sala completa y se reservan los sitios para este evento
Precondición	La sala debe estar activa
Postcondición	El evento queda registrado y asociado a una sala, fecha y hora
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno o en caso de que la sala esté bloqueada.

Tabla 39 - CU usuario administrador: Generar evento

Detalles	Descripción
Identificador	CU_A11
Nombre	Editar evento
Prioridad	Alta
Estabilidad	Alta
Descripción	Se modifica un evento ya existente en una sala
Entrada	idEvento, idSala
Salida	Mensaje de confirmación de modificación de evento
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, evento, sala
Acción	Se modifica un evento existente de una sala
Precondición	La sala debe estar activa y el evento debe existir
Postcondición	El nuevo evento queda registrado
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 40 - CU usuario administrador: Editar evento

Detalles	Descripción
Identificador	CU_A12
Nombre	Eliminar evento
Prioridad	Alta
Estabilidad	Alta
Descripción	Se elimina un evento ya existente en una sala
Entrada	idEvento
Salida	Mensaje de confirmación de eliminación de evento
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, evento
Acción	Se elimina un evento existente de una sala
Precondición	La sala debe estar activa y el evento debe existir
Postcondición	El nuevo evento queda eliminado
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 41 - CU usuario administrador: Eliminar evento

Detalles	Descripción
Identificador	CU_A13
Nombre	Publicar anuncio
Prioridad	Media
Estabilidad	Alta
Descripción	Se crea un anuncio que aparecerá en el tablón de anuncios de la facultad
Entrada	idFacultad, texto de anuncio, fecha de publicación, fecha fin de publicación
Salida	Mensaje de confirmación de creación de anuncio en tablón de anuncios
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, facultad, texto de anuncio, fecha de publicación, fecha fin de publicación
Acción	Se crea un anuncio para el tablón de anuncios de una facultad
Precondición	La facultad debe existir
Postcondición	Queda registrado el anuncio asociado a la facultad
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 42 - CU usuario administrador: Publicar anuncio

Detalles	Descripción
Identificador	CU_A14
Nombre	Editar anuncio
Prioridad	Media
Estabilidad	Alta
Descripción	Se modifica un anuncio existente en el tablón de anuncios de la facultad
Entrada	idFacultad, idAnuncio, texto de anuncio, fecha de publicación, fecha fin de publicación
Salida	Mensaje de confirmación de modificación de anuncio en tablón de anuncios
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, facultad, anuncio, texto de anuncio, fecha de publicación, fecha fin de publicación
Acción	Se modifica un anuncio del tablón de anuncios de una facultad
Precondición	La facultad y el anuncio deben existir
Postcondición	Queda registrado el nuevo anuncio asociado a la facultad
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 43 - CU usuario administrador: Editar anuncio

Detalles	Descripción
Identificador	CU_A15
Nombre	Eliminar anuncio
Prioridad	Media
Estabilidad	Alta
Descripción	Se elimina un anuncio existente
Entrada	idFacultad, idAnuncio
Salida	Mensaje de confirmación de eliminación de anuncio
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, facultad, anuncio
Acción	Se elimina un anuncio de una facultad
Precondición	La facultad y el anuncio deben existir
Postcondición	Queda eliminado el anuncio asociado a la facultad
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 44 - CU usuario administrador: Eliminar anuncio

Detalles	Descripción
Identificador	CU_A16
Nombre	Añadir Facultad
Prioridad	Alta
Estabilidad	Alta
Descripción	Se da de alta una facultad en el sistema, tanto desde aplicación Android como web
Entrada	Nombre, acrónimo, dirección, descripción, ciudad, email, vacaciones, horarios, imagen, región y teléfono
Salida	Mensaje de confirmación de alta de facultad
Origen	GUI (Android, Web)
Destino	GUI (Android, Web)
Necesita	Cuenta de administrador, facultad
Acción	Se da de alta en el sistema una facultad con los datos proporcionados por el administrador.
Precondición	La facultad no debe existir previamente
Postcondición	La facultad se da de alta en el sistema.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 45 - CU usuario administrador: Añadir Facultad

Detalles	Descripción
Identificador	CU_A17
Nombre	Modificar Facultad
Prioridad	Alta
Estabilidad	Alta
Descripción	Permite modificar los atributos de una facultad ya existente, ya sea desde Android o la aplicación web.
Entrada	idFacultad, nombre, acrónimo, dirección, descripción, ciudad, email, vacaciones, horarios, imagen, región y teléfono
Salida	Mensaje de confirmación de modificación de facultad
Origen	GUI (Android, Web)
Destino	GUI (Android, Web)
Necesita	Cuenta de administrador, facultad
Acción	Se actualizan los datos modificados por el administrador
Precondición	La facultad debe existir previamente
Postcondición	La facultad se actualiza con los datos modificados por el administrador
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 46 - CU usuario administrador: Modificar facultad

Detalles	Descripción
Identificador	CU_A18
Nombre	Eliminar Facultad
Prioridad	Alta
Estabilidad	Alta
Descripción	Se elimina la facultad del sistema desde Android o desde el sitio web.
Entrada	idFacultad
Salida	Mensaje de confirmación de eliminación de una facultad
Origen	GUI (Android, Web)
Destino	GUI (Android, Web)
Necesita	Cuenta de administrador, idFacultad
Acción	Se elimina una facultad existente
Precondición	La facultad debe existir previamente
Postcondición	La facultad se elimina del sistema
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 47 - CU usuario administrador: Eliminar facultad

Detalles	Descripción
Identificador	CU_A19
Nombre	Crear documento con códigos QR de sala
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario administrador tiene la opción de generar un PDF con los códigos QR correspondientes a la sala en sí y a los sitios que hay en la sala.
Entrada	idSala
Salida	Mensaje de confirmación de documento generado.
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, idSala
Acción	Se almacena en la memoria interna un PDF con los códigos QR de la sala
Precondición	La sala debe existir.
Postcondición	Se genera un PDF en memoria interna.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 48 - CU usuario administrador: Crear documento con códigos QR de sala

Detalles	Descripción
Identificador	CU_A20
Nombre	Crear documento con códigos QR de evento
Prioridad	Alta
Estabilidad	Alta
Descripción	El usuario administrador tiene la opción de generar un PDF con los códigos QR correspondientes al evento en sí y a los sitios que hay en el evento.
Entrada	idEvento
Salida	Mensaje de confirmación de documento generado.
Origen	GUI
Destino	GUI
Necesita	Cuenta de administrador, idEvento
Acción	Se almacena en la memoria interna un PDF con los códigos QR del evento
Precondición	El evento debe existir.
Postcondición	Se genera un PDF en memoria interna.
Efectos laterales	Se muestra un mensaje de error en caso de producirse alguno

Tabla 49 - CU usuario administrador: Crear documento con códigos QR de evento

Capítulo 5. Arquitectura

En este capítulo se describe y explica la arquitectura del sistema desarrollado además de los patrones de diseño empleados tanto en el desarrollo de la aplicación web como de la aplicación Android.

Además, se explican los modelos utilizados en el almacenamiento de datos en los servicios de Firebase, Firestore Database y Storage.

5.1. Arquitectura del sistema

Al disponer de dos aplicaciones software completamente distintas, cada una de ellas está desarrollada en una arquitectura distinta.

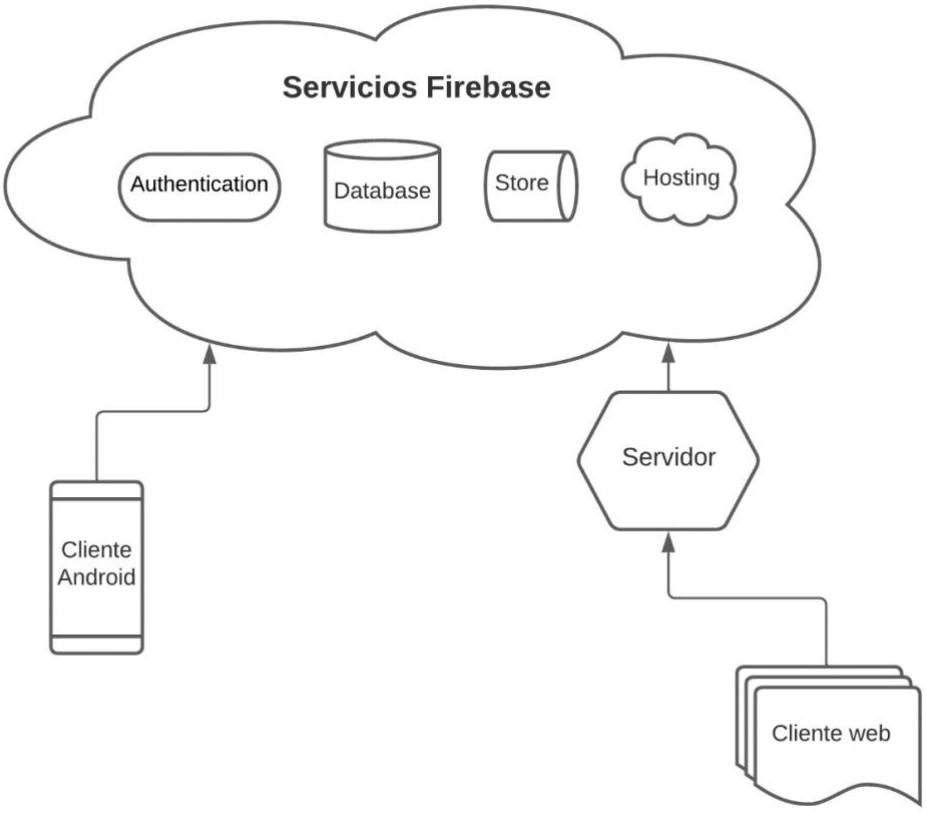


Figura 14 - Diagrama de arquitectura

En la Figura 14 se muestra tanto la arquitectura de la aplicación Android como, la arquitectura de la aplicación web siendo en este caso una arquitectura cliente(s)-servidor ([30]) compartiendo servicios externos de Firebase.

En el caso de la aplicación Android, se hace uso de los servicios Authentication, Firestore Database y Storage de Firebase a través de una API ([31]) de Firebase para desacoplar a la aplicación de esas tareas y mantener los datos actualizados desde cualquier dispositivo móvil.

Por otro lado, en la aplicación web, se hace uso además de los servicios Authentication, Firestore Database y Storage, del servicio de Hosting para dar almacenamiento web y poder conectarse a través de cualquier navegador a la URL ([32]) pública. Al igual que en la aplicación Android, los servicios de Firebase se han utilizado a través de una API.

5.2. Patrones de diseño y arquitectónicos Android

5.2.1. MVVM (Mode-View-ViewModel)

Se ha utilizado el patrón MVVM ([33]) para separar la parte de vistas (Activity y Fragment), de la lógica de negocio. Las vistas se encargan de recoger los eventos generados por el usuario mediante la pantalla, para realizar alguna función en los ViewModel, la cual puede emplear acceso a las bases de datos mediante los repositorios. Finalmente, es el ViewModel el encargado de notificar a la vista cualquier cambio que esta deba realizar en la pantalla.

5.2.2. Singleton

El patrón de diseño Singleton ([34]) se ha utilizado para obtener una instancia de los diferentes servicios que ofrece Firebase. En concreto, se ha utilizado para 3 servicios: Firebase Firestore, Firebase Authentication y Firebase Storage.

```
private val db = FirebaseFirestore.getInstance()
private val storage = FirebaseStorage.getInstance()
```

Figura 15 - Patrón Singleton

5.2.3. Kotin DataClass

Para poder usar los documentos de Firestore obtenidos de Firebase se han de convertir en objetos que se puedan interpretar en el código. Por esto, se usan las DataClasses ([35]) de Kotlin, en las que se pueden convertir los documentos obtenidos de Firestore fácilmente y así poder manejarlos.

```
@Parcelize
data class University (
    @DocumentId val id: String = "",
    var name: String = "",
    var acronym: String = ""
) : Parcelable
```

Figura 16 - Patrón Kotlin DataClass

5.2.4. Kotin StateFlow

Se han utilizado StateFlows ([36]) para almacenar variables de las que se quiere saber su valor más actualizado tanto en las vistas, como en porciones del código. En estas variables de estado se almacenan por ejemplo el listado de facultades y salas obtenidas de la base de datos, la facultad o sala seleccionada, o el usuario que ha iniciado sesión.

```
private val _user = MutableStateFlow( value: savedStateHandle.get(::user.name) ?: User())
val user: StateFlow<User> = _user.asStateFlow()

private val _currentUniversityIt = MutableStateFlow( value: savedStateHandle.get(::currentUnive
val currentUniversityIt: StateFlow<Int> = _currentUniversityIt.asStateFlow()
```

Figura 17 - Patrón Kotlin StateFlow

Además, los StateFlows ofrecen la persistencia de las variables con cambios de estado del dispositivo, lo que puede ocurrir al girar la pantalla o al activar y desactivar el modo oscuro.

5.2.5. Kotlin SharedFlow

Para la comunicación de eventos desde los fragmentos a las actividades se han empleado SharedFlows ([37]) con observadores en la actividad pertinente. De este modo, la actividad puede ser la encargada de ejecutar acciones de sus fragmentos, como realizar una navegación, mostrar un mensaje flotante, o lanzar una nueva actividad.

```
private val _shortToastRes = MutableSharedFlow<Int>()
val shortToastRes: SharedFlow<Int> = _shortToastRes.asSharedFlow()

private val _longSnackbarRes = MutableSharedFlow<Pair<Int,Array<String>>>()
val longSnackbarRes: SharedFlow<Pair<Int,Array<String>>> = _longSnackbarRes.asSharedFlow()
```

Figura 18 - Patrón Kotlin SharedFlow

5.2.6. Observer

Se ha implementado el patrón observador ([38]) para obtener actualizaciones de los anteriormente mencionados StateFlows y SharedFlows. Se usan los llamados colectores para recibir el estado actualizado de las variables.

```
sharedViewModel.shortToastRes.collectFlow( owner: this) { stringRes ->
    Toast.makeText( context: this, getString(stringRes), Toast.LENGTH_SHORT).show()
}

sharedViewModel.createGeofence.collectFlow( owner: this) { room ->
    generateRoomGeofence(room)
}
```

Figura 19 - Patrón Observer

5.2.7. Data Binding

Se ha usado la librería Data Binding ([39]) de Android para optimizar el código, ya que este permite un formato declarativo en vez de programático, el cual ofrece un código más limpio y fácil de mantener y, un control de errores en tiempo de compilación en vez de ejecución. Además, es más eficiente al realizar la carga en tiempo de compilación.

Para el uso de ciertos aspectos del Data Binding en las vistas se han empleado Binding Adapters, los cuáles permiten la creación y carga de RecyclerViews en las propias vistas.

5.2.8. Corrutinas de Kotlin

Para ciertos aspectos del sistema como las llamadas a bases de datos se han utilizado corrutinas ([40]), las cuáles existen de forma nativa en Kotlin. Estas permiten ejecutar código de forma asíncrona a la ejecución de la aplicación, con lo que la aplicación sigue siendo completamente funcional, aunque se esté esperando una respuesta de la base de datos, por ejemplo. También permiten realizar diferentes funciones de forma simultánea, con lo que se consigue una ejecución más rápida.

Además, para optimizar las operaciones de llamadas a la base de datos se ha especificado a las corrutinas correspondientes que hagan uso de un despachador de entrada/salida (IO), el cual ofrece mayor eficiencia en dichas operaciones.

5.2.9. SavedStateHandle

Se ha incorporado el módulo de Saved State ([41]), el cuál junto a los StateFlows y ViewModels permite que el estado de las variables permanezca incluso al iniciar la aplicación después de que esta haya pasado a segundo plano.

```
private val _currentFaculty = MutableStateFlow( value: savedStateHandle.get(::currentFaculty.name) ?: Faculty())
val currentFaculty: StateFlow<Faculty> = _currentFaculty.asStateFlow()
```

Figura 20 - Patrón SavedStateHandle 1

```
_currentFaculty.value = something
savedStateHandle.set(::currentFaculty.name, currentFaculty.value)
```

Figura 21 - Patrón SavedStateHandle 2

5.3. Patrones de diseño y arquitectónicos web

5.3.1. Model-View-ViewModel (MVVM)

Al igual que en la aplicación Android, también, se ha utilizado el patrón MVVM en la página web, permitiendo que se puedan realizar actualizaciones en las vistas a través del modelo observador que usa ViewModel, con un enlace bidireccional entre View y ViewModel. Este modelo también contempla la reutilización de vistas como componentes.

5.3.2. Observer

El patrón observador se ha utilizado para recibir en el modelo una notificación de actualización de los datos en el momento en el que se modifiquen en Firestore o para detectar un cambio en las vistas. De esta forma, en el modelo de la página web se pueden mantener actualizados los datos en todo momento.

```
private getFaculties() :void {
  this._firebaseService.getAllFaculties().subscribe(
    (respuesta: IFaculty[]) => {
      //console.table(respuesta);
      this.faculties = respuesta
    },
    (error) => alert(`Error al obtener las facultades de Firebase: ${error}`),
    () => console.log('Facultades obtenidas con éxito')
  )
}
```

Figura 22 - Patrón Observer

5.3.2. Facade

Se ha utilizado el patrón fachada ([42]) adaptando y encapsulando las llamadas de distintos servicios de Firebase. Por ejemplo, a la hora de dar de alta una facultad, se llama el servicio “FirebaseService” en el que se encuentra implementada la llamada inicial a la carga de ficheros para almacenar la imagen en Firebase Storage, y a continuación, se realiza la llamada al servicio de Firebase Database para almacenar la información de la facultad.

5.3.3. Proxy

También llamado patrón interceptor ([43]), se ha implementado en el control de acceso de navegación a los recursos de la aplicación.

```
const routes: Routes = [
  { path: '', redirectTo: 'index' , pathMatch: 'full'},
  //{ path: 'index', canActivate: [LoggedInGuard]}, //lista de guard para ver si está logueado y es admin
  { path: 'index', component: IndexViewComponent},
  { path: 'about', component: AboutViewComponent},
  { path: 'faculties/list', component: FacultyListViewComponent, canActivate: [LoggedInGuard]},
  { path: 'faculties/new', component: FacultyNewViewComponent, canActivate: [LoggedInGuard]},
  { path: 'faculties/:id', component: FacultyEditViewComponent, canActivate: [LoggedInGuard]},//ruta como parametro
  { path: 'users/list', component: UserListViewComponent, canActivate: [LoggedInGuard] },
  { path: 'login', component: LoginViewComponent}
];
```

Figura 23 - Patrón Proxy 1

Para realizar comprobaciones de acceso, se ha utilizado la interfaz canActivate implementada en el interceptor (Guard) “LoggedInGuard” verificando si el usuario actual tiene acceso a la ruta solicitada:

```
canActivate(
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot):
  Observable<boolean | UrlTree>
  | Promise<boolean | UrlTree>
  | boolean
  | UrlTree {

    if(this._authService.isLoggedIn() !== true) {
      this._router.navigate(['login']);
    }
    return true;
}
```

Figura 24 - Patrón Proxy 2

```
//Se verifica que el usuario haya iniciado sesión en la aplicación
isLoggedIn(): boolean {
  return (localStorage.getItem('user')
  && localStorage.getItem('user') !== null
  && localStorage.getItem('user') !== '{}') ? true : false;
}
```

Figura 25 - Patrón Proxy 3

5.3.4. Data Binding

En Angular se ha utilizado la sintaxis Data Binding ([44]) para mantener los datos del modelo actualizados entre la vista y el componente mediante enlaces. Estos enlaces que se pueden definir de distinto tipo, al igual que en la aplicación Android, facilitan la programación y la mantenibilidad del código.

5.4. Modelo de repositorio de documentos

Como almacenamiento de documentos del sistema, se ha utilizado la herramienta Storage que proporciona el servicio de Firebase. Partiendo desde la raíz del servicio, la estructura que se ha utilizado para el almacenamiento es la siguiente:

/images/
    faculties/

rooms/

Hasta el momento, en este proyecto solo se almacenan imágenes de facultades y salas, por lo que, se ha creado únicamente un directorio llamado “images”.

Dentro de este directorio, se encuentran por separado dos directorios más, que engloban lo siguiente:

- faculties: almacena todas las imágenes de las facultades que se almacenan en el sistema.
- rooms: contiene todas las imágenes de las salas que se almacenan en el sistema.

Para denominar a las imágenes se ha utilizado un patrón con el nombre de la facultad o sala correspondiente seguido de la fecha del sistema en formato de año, mes, día, hora, minutos y segundos, todo ello sin espacios. Por ejemplo, si se subiera la imagen de una facultad llamada “Facultad de Informática” el día 5 de junio de 2022 a las 12:30:15 horas, esta imagen se subiría al directorio “/images/faculties/” con el nombre “FacultaddeInformatica20220605123015”.

5.5. Modelo de base de datos NoSQL

Como modelo de datos se ha usado una base de datos no relacional creada por Google llamada Cloud Firestore orientada a documentos. Los datos se almacenan en documentos que se organizan en colecciones como se puede ver en la imagen de la parte inferior.

En cada documento se definen un conjunto de pares clave-valor. La clave-valor se puede usar para referenciar a otro documento y establecer relaciones.

El sistema contiene universidades. De cada ***universidad*** se guarda su nombre y acrónimo. Cada universidad puede tener varias facultades.

Cada ***facultad*** tendrá un nombre, acrónimo, dirección, ciudad, provincia, correo electrónico, nombre, descripción, imagen, horarios por día y fechas de festivos. Cada facultad puede tener varias salas.

Una ***sala*** contiene información sobre su nombre, capacidad, mapa de sitios, teléfono, correo electrónico, imagen y horarios por día. También tienen atributos para especificar si la sala está bloqueada, hasta cuando, y el motivo del bloqueo.

En la facultad existen **eventos y noticias**. Ambos contienen un nombre y una descripción, y opcionalmente una URL y fecha de inicio y de fin. En caso de ser un evento en sala se guardará obligatoriamente una fecha de inicio y de fin, y la sala en la que tendrá lugar el evento.

Los usuarios registrados que accedan a la aplicación también se guardan en la base de datos. Un ***usuario*** contiene información de nombre, apellidos, email y tipo de usuario.
Un usuario también puede tener una y solo una universidad favorita. Se guardan también atributos para saber si el usuario ha hecho check-in y hasta cuándo dura este.

Las ***reservas*** realizadas por los usuarios se guardan incluyendo el usuario que ha realizado la reserva, la sala y el asiento de la reserva, la facultad de la sala, y la fecha y hora de inicio y fin de la reserva.

Finalmente se guardan las ***ocupaciones***, las cuales llevan un registro de los usuarios que han hecho check-in en las distintas salas, guardando únicamente la sala del check-in y la fecha y hora de check-in y check-out.

En la Figura 26 que aparece a continuación se muestra la vista de la base de datos donde se pueden diferenciar los siguientes elementos:

1. Representa el nombre de la base de datos.
2. Representa las distintas colecciones de la base de datos.
3. Representa los distintos documentos que se agrupan dentro de una colección en concreto. El nombre de cada documento equivale a su identificador.
4. Representa los distintos campos de cada documento junto a su valor mediante una estructura clave-valor.

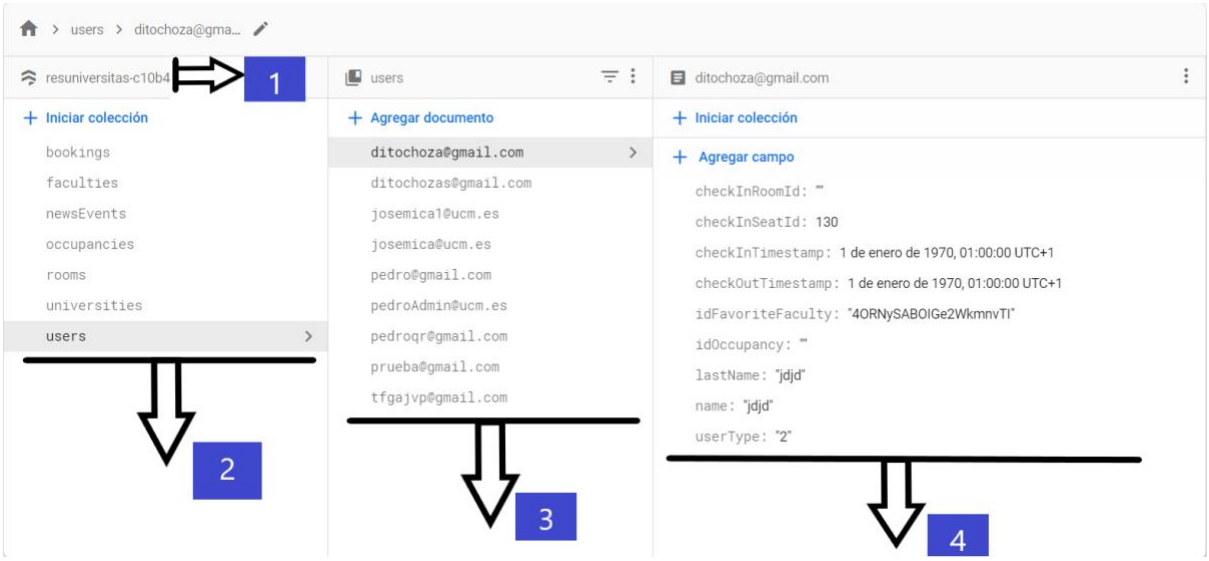


Figura 26 - Firestore Database: Colección de usuarios

En este apartado se describe la estructura de cada una de las tablas de la base de datos organizadas por el tipo de entidad de la que se guarda información.

5.5.1. Tabla sobre universidades

Contiene información sobre una entidad universitaria:

- acronym: Representa el acrónimo de la universidad a la que pertenece la facultad y es de tipo string.
- name: Representa el nombre de la universidad y es de tipo string.

5.5.2. Tabla sobre facultades

Contiene información sobre una facultad:

- idUniversity: Representa el identificador de la universidad a la que pertenece la facultad y es de tipo string.
- acronym: Representa el acrónimo de la facultad y es de tipo string.
- active: Representa si la facultad se encuentra activa o inactiva y es de tipo boolean.
- address: Representa la dirección de la facultad y es de tipo string.
- city: Representa la ciudad de la facultad y es de tipo string.
- coordinates: Representa las coordenadas de la facultad y es de tipo array de números.
- description: Representa la descripción de la facultad y es de tipo string.
- email: Representa el email de la facultad y es de tipo string.
- holidays: Representa la fecha de los días festivos o de cierre de la facultad y es de tipo array de timestamps.
- hoursEnd: Representa la hora de cierre de la facultad para cada día de la semana y es de tipo array de strings.
- hoursStart: Representa la hora de apertura de la facultad para cada día de la semana y es de tipo array de strings.
- imageUrl: Representa el URL de la imagen de la facultad y es de tipo string.
- name: Representa el nombre de la facultad y es de tipo string.
- region: Representa la provincia de la facultad y es de tipo string.
- tlfNumber: Representa el número de teléfono de la facultad y es de tipo string.

5.5.3. Tabla sobre salas

Contiene información sobre una sala:

- idFaculty: Representa el identificador de la facultad a la que pertenece la sala y es de tipo string.
- active: Representa si la sala se encuentra activa o inactiva y es de tipo boolean.

- coordinates: Representa las coordenadas de la sala, junto al radio correspondiente al área circular y es de tipo array.
- email: Representa el email de la sala y es de tipo string.
- hoursEnd: Representa la hora de cierre de la sala para cada día de la semana y es de tipo array de strings.
- hoursStart: Representa la hora de apertura de la sala para cada día de la semana y es de tipo array de strings.
- imageUrl: Representa el URL de la imagen de la sala y es de tipo string.
- messageLocked: Representa el mensaje de sala bloqueada y es de tipo string.
- name: Representa el nombre de la sala y es de tipo string.
- seats: Representa el mapa de sitios de una sala mediante una cadena y es de tipo string.
- timestampEndLockdown Representa la fecha y hora de finalización del bloqueo de sala y es de tipo timestamp.
- tlfNumber: Representa el número de teléfono de la sala y es de tipo string.

5.5.4. Tabla sobre eventos y noticias

Contiene información sobre un evento o noticia:

- idFaculty: Representa el identificador de la facultad a la que pertenece el evento o noticia y es de tipo string.
- idRoom: Representa el identificador de la sala en caso de que se trate de un evento en sala y es de tipo string.
- description: Representa la descripción del evento o noticia y es de tipo string.
- link: Representa la URL del evento o noticia y es de tipo string.
- showTimestampEnd: Representa si se muestra la fecha de finalización del evento o noticia y es de tipo boolean.
- showTimestampStart: Representa si se muestra la fecha de comienzo del evento o noticia y es de tipo boolean.
- timestampEnd: Representa la fecha y hora de finalización del evento o noticia y es de tipo timestamp.
- timestampStart: Representa la fecha y hora de comienzo del evento o noticia y es de tipo timestamp.
- title: Representa el título del evento o noticia y es de tipo string.

5.5.5. Tabla sobre usuarios

Contiene información sobre un usuario:

- checkInRoomId: Representa el identificador de sala donde el usuario ha realizado el check-in y es de tipo string.
- checkInSeatId: Representa el identificador del sitio donde el usuario ha realizado el check-in y es de tipo entero.

- checkInTimestamp: Representa la fecha y hora del check-in y es de tipo timestamp.
- checkOutTimestamp: Representa la fecha y hora del check-out y es de tipo timestamp.
- idFavoriteFaculty: Representa el identificador de la facultad favorita del usuario y es de tipo string.
- idOcuppancy: Representa el identificador de ocupación y es de tipo string.
- lastName: Representa los apellidos del usuario y es de tipo string.
- name: Representa el nombre del usuario y es de tipo string.
- userType: Representa el tipo de usuario y es de tipo string.

5.5.6. Tabla sobre reservas

Contiene información sobre una reserva:

- idFaculty: Representa el identificador de la facultad a la que pertenece la reserva y es de tipo string.
- idRoom: Representa el identificador de la sala a la que pertenece la reserva y es de tipo string.
- idSeat: Representa el identificador del sitio y es de tipo string.
- idUser: Representa el identificador del usuario y es de tipo string.
- timestampEnd: Representa la fecha y hora de fin de la reserva y es de tipo timestamp.
- timestampStart: Representa muestra la fecha y hora de comienzo de la reserva y es de tipo timestamp.

5.5.7. Tabla sobre ocupación

Contiene información sobre la ocupación en salas:

- idRoom: Representa el identificador de la sala en la que se ha hecho check-in y es de tipo string.
- timestampEnd: Representa la fecha y hora del check-out y es de tipo timestamp.
- timestampStart: Representa muestra la fecha y hora del check-in y es de tipo timestamp.

Capítulo 6. Diseño e implementación

En este capítulo se van a desarrollar los aspectos referentes al diseño y a la implementación de la aplicación Android y la web.

6.1. Diseño Android

En términos generales, se han seguido las guías de diseño marcadas por Google en su sistema de diseño más reciente llamado Material Design 3 ([45]).

6.1.1. Colores

Para la elección de colores se ha usado Material Theme Builder ([46]), con el cual, introduciendo los colores primario, secundario y terciario, se han generado el resto de los colores que se han usado en la aplicación. Se ha decidido usar colores fríos, con tonos azules para el color primario y secundario, y un tono morado para el color terciario. También se ha generado como añadido el color marrón, el cual se usa para representar las mesas en el mapa de sitios de las salas.

A continuación, tal como se visualiza en la Figura 27 se puede observar el tema generado con Material Theme Builder donde se puede ver la gama de colores que le componen junto a su prioridad.

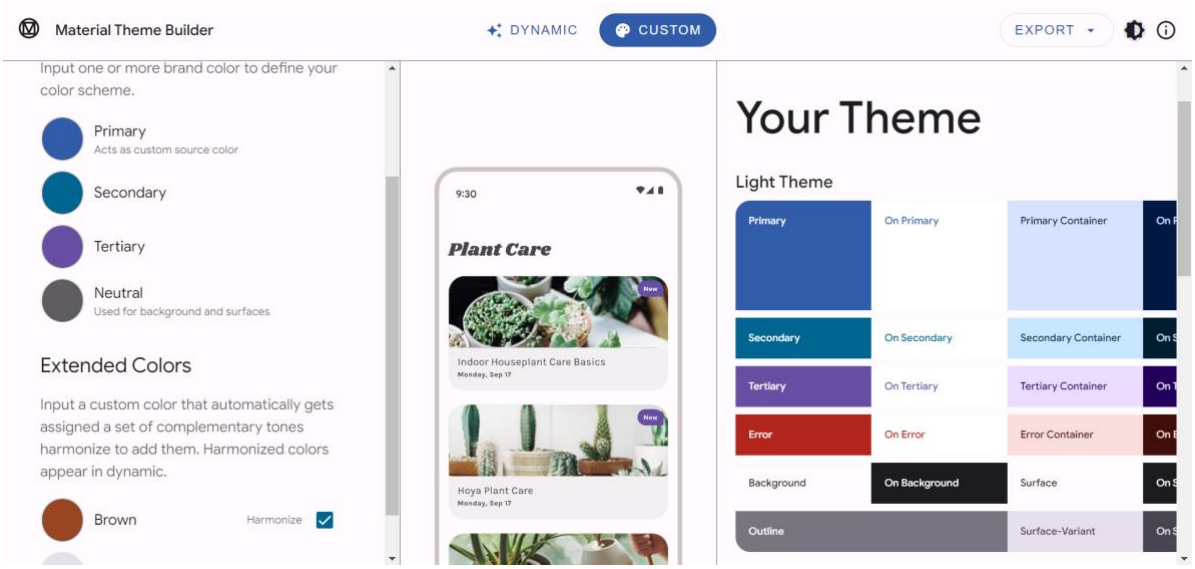


Figura 27 - Diseño Android: Material Theme Builder

En los colores exportados del Material Theme Builder también se encuentran colores adaptados al modo oscuro de Android, por lo que la app se adaptará al tema oscuro cuando el usuario lo active, mejorando así la visualización de esta.

Se ha decidido utilizar también los colores dinámicos (Dynamic Colors) ([47]), con los cuales los colores de la aplicación, si esta se está ejecutando en un dispositivo con Android 12 o superior, se adaptarán al tema elegido por el usuario en su dispositivo. Se puede ver cómo se adaptan los colores de la aplicación a los diferentes temas en la Figura 28.

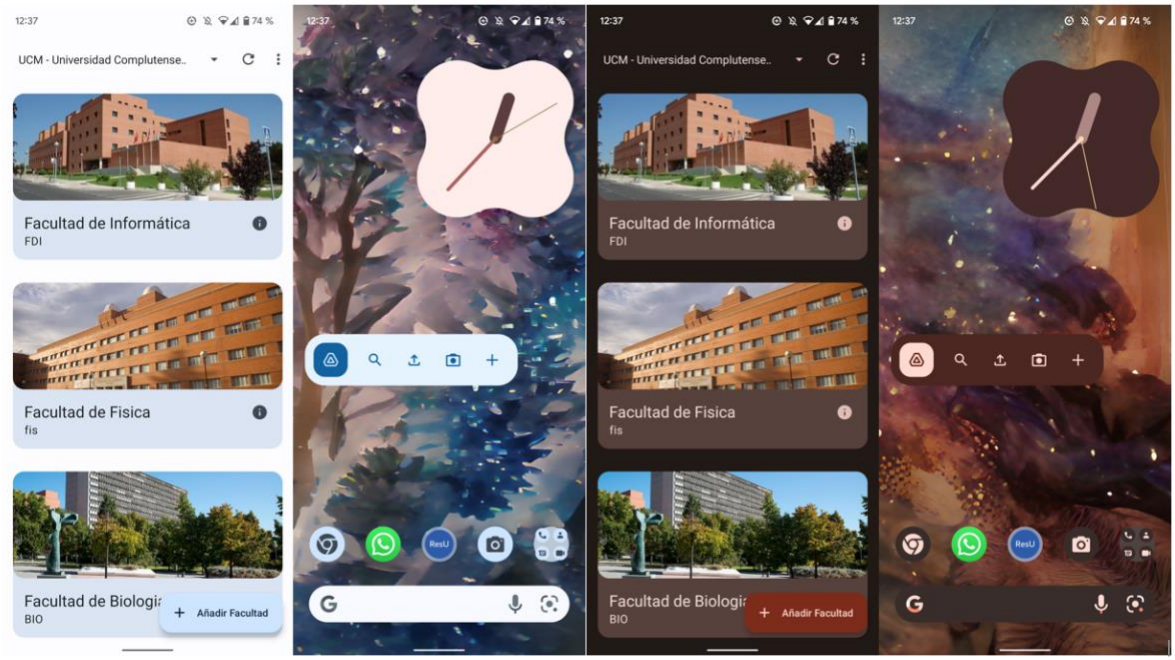


Figura 28 - Diseño Android: Dynamic Colors

6.1.2. Temas

Para conseguir que la aplicación se visualice correctamente en las distintas versiones de Android partiendo de la versión Android 5.0 (La mínima soportada por la aplicación), se han creado varios recursos de temas. De este modo se consigue que las últimas versiones aprovechen las características de diseño más recientes, mientras que las versiones más antiguas no se ven afectadas por estas nuevas características no soportadas.

Seguidamente, tal como se visualiza en las Figuras 29 y 30, se puede observar los diferentes recursos de temas anteriormente mencionados presentes en la aplicación y un ejemplo a nivel de código de uno de estos recursos.

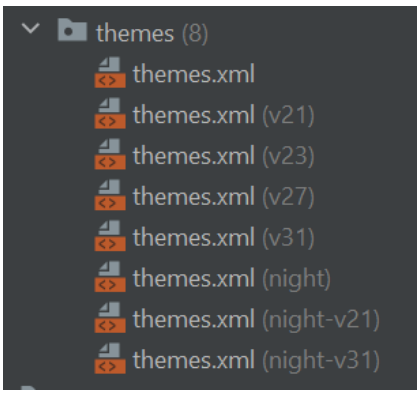


Figura 29 - Diseño Android: Temas con diferentes versiones

themes.xml

```
<style name="BaseAppTheme.ResUniversitas" parent="Theme.Material3.Light.NoActionBar">
    <item name="colorPrimary">@color/md_theme_light_primary</item>
    <item name="colorOnPrimary">@color/md_theme_light_onPrimary</item>
    <item name="colorPrimaryContainer">@color/md_theme_light_primaryContainer</item>
    <item name="colorOnPrimaryContainer">@color/md_theme_light_onPrimaryContainer</item>
    <item name="colorSecondary">@color/md_theme_light_secondary</item>
    <item name="colorOnSecondary">@color/md_theme_light_onSecondary</item>
    <item name="colorSecondaryContainer">@color/md_theme_light_secondaryContainer</item>
    <item name="colorOnSecondaryContainer">@color/md_theme_light_onSecondaryContainer</item>
    <item name="colorTertiary">@color/md_theme_light_tertiary</item>
    <item name="colorOnTertiary">@color/md_theme_light_onTertiary</item>
    <item name="colorTertiaryContainer">@color/md_theme_light_tertiaryContainer</item>
    <item name="colorOnTertiaryContainer">@color/md_theme_light_onTertiaryContainer</item>
    <item name="colorError">@color/md_theme_light_error</item>
    <item name="colorErrorContainer">@color/md_theme_light_errorContainer</item>
    <item name="colorOnError">@color/md_theme_light_onError</item>
    <item name="colorOnErrorContainer">@color/md_theme_light_onErrorContainer</item>
    <item name="android:colorBackground">@color/md_theme_light_background</item>
    <item name="colorOnBackground">@color/md_theme_light_onBackground</item>
    <item name="colorSurface">@color/md_theme_light_surface</item>
    <item name="colorOnSurface">@color/md_theme_light_onSurface</item>
    <item name="colorSurfaceVariant">@color/md_theme_light_surfaceVariant</item>
    <item name="colorOnSurfaceVariant">@color/md_theme_light_onSurfaceVariant</item>
    <item name="colorOutline">@color/md_theme_light_outline</item>
```

Figura 30 - Diseño Android: Temas versión por defecto

6.1.3. Splash Screen

Otro añadido en cuestión de diseño ha sido la implementación de una Splash Screen ([48]), la cual, es una pantalla que se muestra al iniciar la aplicación cuando esta no estaba cargada en segundo plano. En esta pantalla se ha decidido utilizar el texto “ResU” como icono, que sería el diminutivo de “ResUniversitas”. Además, empezando en Android 12, este icono se ha animado ligeramente para aportar atractivo visual. En las capturas de la derecha de la Figura 31 se puede observar el aspecto visual de la Splash Screen.

Cabe destacar que se ha programado la Splash Screen para que no muestre la interfaz de la propia aplicación hasta que no estén cargados todos los elementos visuales. De este modo, cuando se muestre la interfaz al usuario, este podrá ver todo sin necesidad de esperar a que ningún elemento termine de cargar.

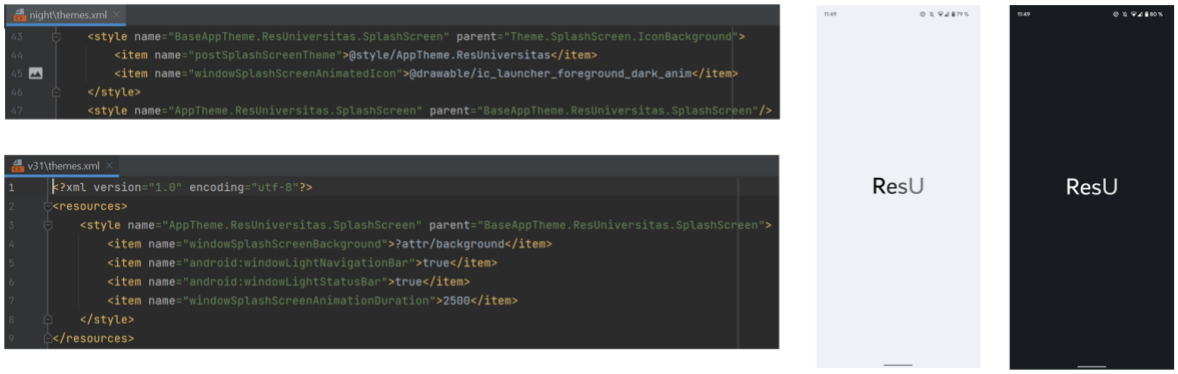


Figura 31 - Diseño Android: Splash Screen

6.2. Diseño Web

A continuación, se describen los aspectos fundamentales del diseño de la aplicación web donde se ha implementado un diseño responsive mediante los frameworks Bootstrap ([49]) y Angular.

6.2.1. Diseño

El diseño se ha implementado usando Bootstrap, un framework de Javascript para el desarrollo de aplicaciones de front-end que tiene compatibilidad con la mayoría de los navegadores y Angular, un framework de Javascript escrito en Typescript.

Por otra parte, se han implementado archivos CSS3 para dar diseño a los documentos HTML y complementar las clases Bootstrap anteriormente mencionadas.

Seguidamente, como se visualiza en las Figura 32 y 33 se muestra un ejemplo de un fragmento del archivo *faculty-list.component.css* y su repercusión en la vista *faculty-list.component.html.*

```css
.description {
    max-width: 200px;
    max-height: 160px;
    text-overflow: ellipsis;
    overflow:hidden;
    white-space: nowrap;
}

.botones {
    margin-bottom: 1em;
}

.buttonsAction{
 width: 100%;
}

.imageFaculty{
    height: 150px;
    width: 150px;
    border-radius: 50%;
    border: 5px solid rgb(133, 190, 228);
}

#tdCenterColumn{
    margin-top: 25%;
    margin-bottom: 25%;
}
```

Figura 32 - Diseño web: Diseño, CSS

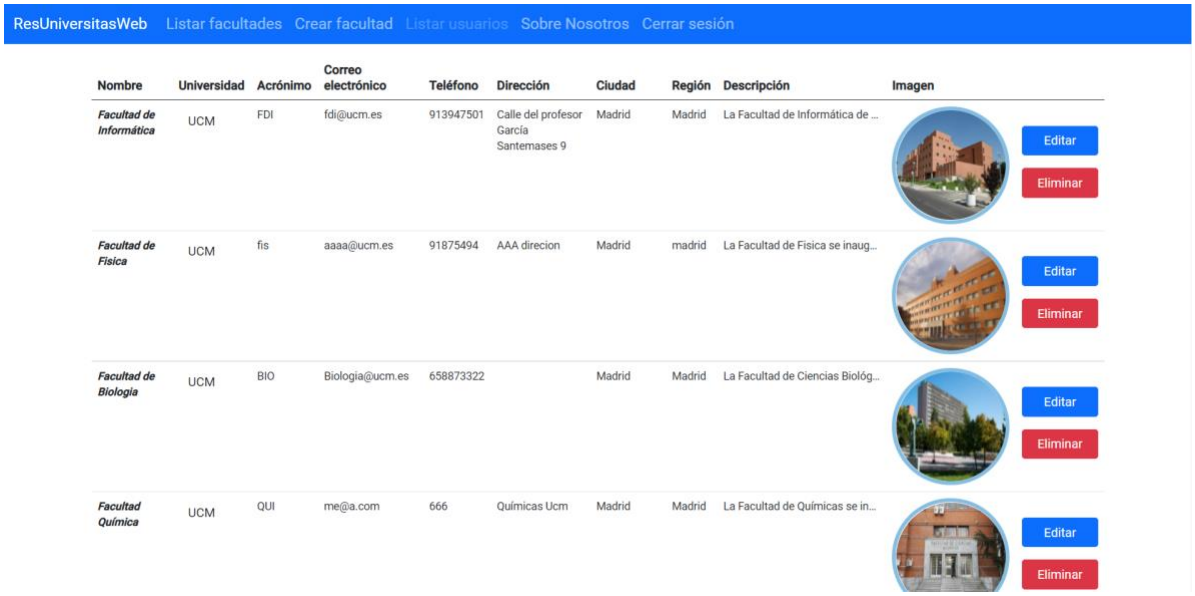


Figura 33 - Diseño web: Diseño pantalla listado de facultades

6.2.2. Bootstrap

Es un framework de Javascript enfocado al desarrollo de aplicaciones de front-end que proporciona una librería de estilos CSS, aplicando un diseño responsive ([50]) adaptable a cualquier dispositivo.

Actualmente la compatibilidad de Bootstrap se extiende a los siguientes navegadores:

- Navegadores en dispositivos móviles

Chrome	**Firefox**	**Safari**	**Android Browser & WebView**	**Microsoft Edge**	
Android	Supported	Supported	Supported	Android v5.0+ supported	Supported
IOS	Supported	Supported	Supported	Supported	Supported

Tabla 50 - Diseño web: Bootstrap, navegadores en dispositivos móviles

- Navegadores en ordenadores

Chrome	**Firefox**	**Internet Explorer**	**Microsoft Edge**	**Opera**	**Safari**	
Mac	Supported	Supported	N/A	N/A	Supported	Supported
Windows	Supported	Supported	Supported , IE10+	Supported	Supported	Supported

Tabla 51 - Diseño web: Bootstrap, navegadores en ordenadores

Tal como se visualiza en la Figura 34 se muestra la inclusión del framework en la aplicación web mediante el uso de los CDN ([51]) proporcionados en la página oficial de Bootstrap. De esta manera, se evita cargar el framework en el servidor de la aplicación

```
9   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
10  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" i
```

Figura 34 - Diseño web: Bootstrap, utilización

A continuación, como se visualiza en la Figura 35 se muestra un ejemplo a nivel de código del archivo HTML *navbar.component.html* donde se usan clases de Bootstrap para el diseño de la barra de navegación de la aplicación.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary" id="margin">
  <div class="container-fluid">
    <a class="navbar-brand" [routerLink]="['index']">ResUniversitasWeb</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
      aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse fs-5" id="navbarNav">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link" [routerLink]="['faculties/list']">Listar facultades</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" [routerLink]="['faculties/new']">Crear facultad</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" tabindex="-1" aria-disabled="true" [routerLink]="['/users/list']">Listar
            usuarios</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" [routerLink]="['about']">Sobre Nosotros</a>
        </li>
        <li class="nav-item" *ngIf='!authService.isLoggedIn()'>
          <a class="nav-link" [routerLink]="['login']">Iniciar sesión</a>
        </li>
        <li class="nav-item" *ngIf='authService.isLoggedIn()'>
          <div class="nav-link" (click)="logout()">Cerrar sesión</div>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Figura 35 - Diseño web: Bootstrap, utilización en elementos HTML

En la Figura 36 que se muestra a continuación se puede ver a nivel de vista la barra de navegación, cuyos estilos únicamente se han definido mediante Bootstrap.

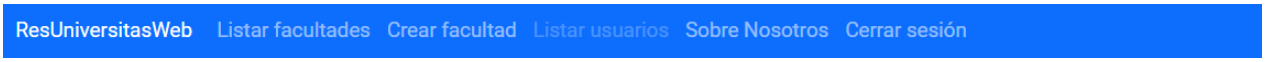


Figura 36 - Diseño web: Bootstrap, menú de navegación

6.2.3. Angular 13

Otro añadido en cuestión de diseño ha sido el uso de Angular 13, el cual, es un framework de Javascript escrito en Typescript ([52]), basado en componentes. En cada componente se define una clase que contiene datos y lógica la cual está vinculada a una página HTML y un archivo CSS3.

Cabe destacar que gracias a estos componentes el diseño de la aplicación web se ha implementado de manera escalable donde los distintos componentes son reutilizables, ver Figura 37.

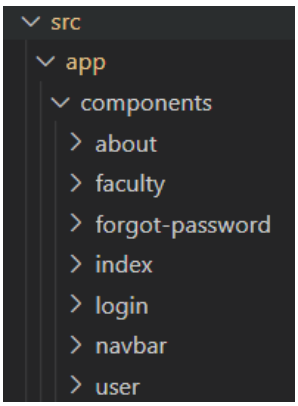


Figura 37 - Diseño web: Angular 13, estructura de proyecto

Es importante añadir que existe un archivo CSS3 general de la aplicación web el cual se encuentra vinculado a todas las páginas HTML llamado *styles.css*.

Por otra parte, cabe señalar el uso de las directivas que proporciona Angular que proveen una lógica de programación y extienden la funcionalidad del lenguaje HTML. Se han usado directivas estructurales que modifican la estructura del archivo HTML ya sea añadiendo o eliminando elementos HTML.

Las directivas usadas se especifican a continuación:

- *ngIf permite evaluar de forma condicional una condición y dependiendo del resultado normalmente se suele utilizar con la finalidad de mostrar u ocultar un elemento HTML.
- *ngFor permite iterar un array.

A continuación, tal como se visualiza en la Figura 38 se puede observar un ejemplo a nivel de código del archivo HTML *faculty-forn.component.html* donde se usan las directivas **ngIf* y **ngFor.*

```
  <option
    [value]="university.id"
    *ngFor="let university of universities"
  >
    {{ university.name }}
  </option>
</select>

<div *ngIf="idUniversity.invalid">
  <small
    class="form-text"
    style="color: red"
    *ngIf="idUniversity.errors?.['required']"
  >
    El campo es requerido</small
  >
</div>
```

*Figura 38 - Diseño web: Angular 13, directiva *ngFor*

Es importante destacar el uso de data binding proporcionado por Angular donde se produce una sincronización de datos entre el modelo y la vista.

Se han implementado las diferentes variantes de data binding que angular proporciona, que son las siguientes:

- Property Binding da la posibilidad de vincular valores a propiedades de elementos HTML o a directivas.
- Event Binding permite escuchar los eventos del documento HTML.
- Two Ways Binding permite enviar información desde el documento HTML hacia las propiedades de los componentes y viceversa.

Seguidamente, en la Figura 39, se puede ver un ejemplo a nivel de código del documento HTML *faculty-list.component.html* donde se puede ver el uso de data binding.

```
  <select name="idUniversity" id="idUniversity" class="form-control" [(ngModel)]="faculty.idUniversity"
    #idUniversity="ngModel" disabled style="border:0; background:none;">
    <option [value]="university.id" *ngFor="let university of universities">{{university.acronym}}</option>
  </select>
</td>
<td>{{faculty.acronym}}</td>
<td>{{faculty.email}}</td>
<td>{{faculty.tlfNumber}}</td>
<td>{{faculty.address}}</td>
<td>{{faculty.city}}</td>
<td>{{faculty.region}}</td>
<td>
  <div class="description">{{faculty.description}}</div>
</td>
<td><img class="imageFaculty" src="{{faculty.imageUrl}}" /></td>

<td>
  <div id="tdCenterColumn">
    <a [routerLink]="['/faculties/' + faculty.id]">
      <button class="btn btn-primary botones buttonsAction">Editar</button>
    </a>

    <button class="btn btn-danger buttonsAction" (click)="clickMethodDelete(faculty)">Eliminar</button>
  </div>

</td>
```

Figura 39 - Diseño web: Angular 13, variantes data binding

6.2.4. Fuente

Para la elección de la fuente en la aplicación web se han usado diferentes fuentes importadas de Google Fonts ([53]), de esta manera, se evita almacenar dichas fuentes en el servidor.

A continuación, tal como se visualiza en la Figura 40, se puede observar un ejemplo a nivel de código de la importación de la fuente en el archivo CSS3 general mencionado anteriormente.

```
# styles.css > ...
/* You can add global styles to this file, and also import other style files */
@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@100;400&display=swap');

html, body { height: 100%; }
body { margin: 0; font-family: 'Roboto', sans-serif; }
```

Figura 40 - Diseño web: Fuente, CSS

6.3. Funcionalidad de la aplicación Android

En este apartado se desarrollan algunas de las funcionalidades más importantes que se han implementado en la aplicación Android, acompañadas de sus respectivas vistas.

6.3.1. Inicio de sesión y registro de usuario

Al abrir la aplicación por primera vez se lanzará la pantalla de inicio de sesión, desde la que se puede iniciar sesión o navegar a la pantalla de registro, ambas representadas en la Figura 41. Tanto la función de iniciar sesión como la de registro de usuario, que se encuentran en el ViewModel compartido por estas dos pantallas, se llaman desde sus respectivos botones de la vista.

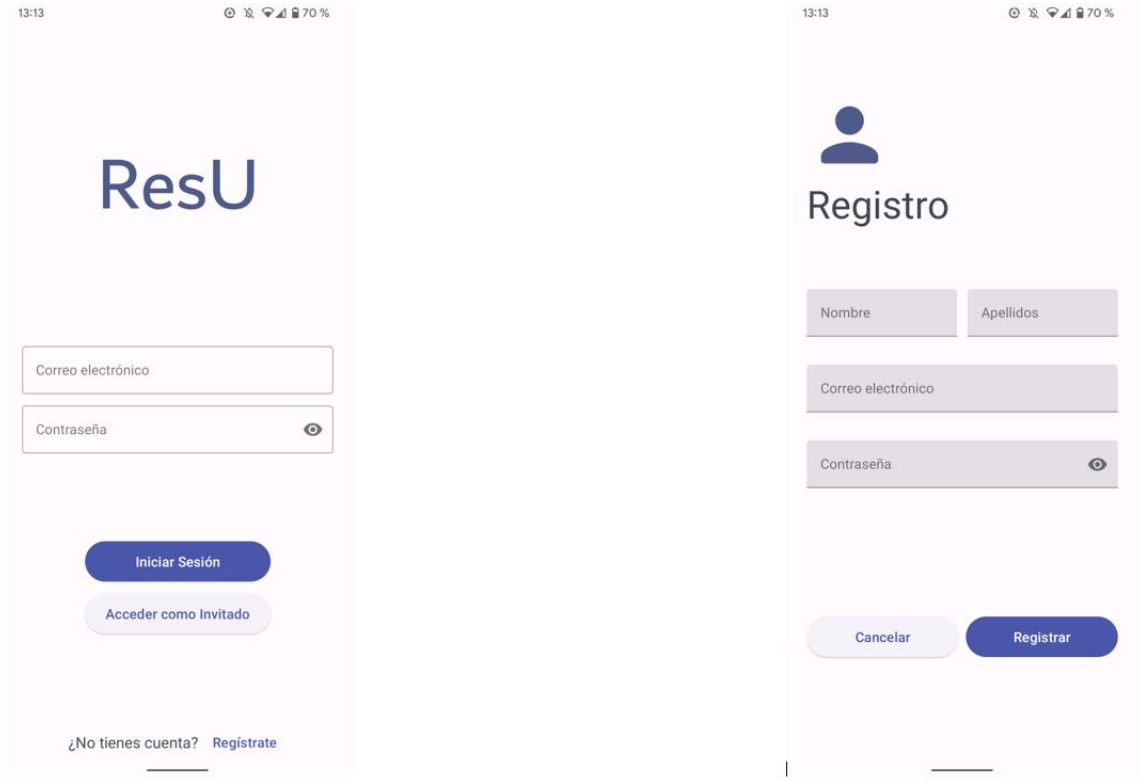


Figura 41 - Funcionalidades Android: Vista de inicio de sesión o registro

En la Figura 42 se puede ver como ejemplo la función de iniciar sesión, la cual primero comprueba si hay algún error en los datos introducidos, en cuyo caso muestra dichos errores en sus respectivos cuadros de texto, y después llama a la función del repositorio que realiza el inicio de sesión usando Firebase Authentication (ver Figura 43).

```
fun onClickValidateLogin(email: String, password: String, tilEmail: TextInputLayout, tilPassword
    var error = false

    //Comprueba errores en el correo elctrónico
    if (tilEmail.tilError(email.isEmpty(), "Field cannot be empty")) error = true
    else if (tilEmail.tilError(!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches(), "E

    //Comprueba errores en la contraseña
    if (tilPassword.tilError(password.isEmpty(), "Field cannot be empty")) error = true
    else if (tilPassword.tilError( conditionError: password.length < 8, "Password must be at least 8 c

    if (!error) {
        //Inicia el login
        viewModelScope.launch {  this: CoroutineScope
            _fUserState.value = DatabaseResult.Loading()
            val result = userRepository.login(email, password)
            result.let { _fUserState.value = it }
        }
    }
}
```

Figura 42 - Funcionalidades Android: Función onClickValidateLogin de StartViewModel

```
suspend fun login(email: String, password: String): DatabaseResult<FirebaseUser?> =
    try { withContext(Dispatchers.IO) {  this: CoroutineScope
        auth.signOut()
        auth.signInWithEmailAndPassword(email, password).await()
        DatabaseResult.success(auth.currentUser)  ^withContext
    } } catch (e: FirebaseTooManyRequestsException) { //Cuenta bloqueada por demasiados intentos
        DatabaseResult.failed("Too many failed attempts. Try again later")
    } catch (e: FirebaseAuthInvalidUserException) { //No existe una cuenta con ese correo electr
        DatabaseResult.failed("There is no account associated with this email")
    } catch (e: FirebaseAuthInvalidCredentialsException) { //La contraseña introducida no es cor
        DatabaseResult.failed("Password is not correct")
    } catch (e: Exception) { DatabaseResult.failed("Failed to login") }
```

Figura 43 - Funcionalidades Android: Función login de UserRepository

Por último, existe un colector del StateFlow de usuario ubicado en la actividad de los fragmentos de inicio de sesión y registro de usuario. Este colector que se muestra en la Figura 44 se encarga de lanzar la actividad principal de la aplicación en caso de que todo haya ido bien, o mostrar un Toast con el mensaje de error en caso de que haya habido algún problema al iniciar sesión con Firebase Authentication.

```
startViewModel.fUserState.collectLatestFlow( owner: this) { fUserState ->
    startViewModel.setLoading(fUserState is DatabaseResult.Loading)
    when (fUserState) {
        is DatabaseResult.Success -> fUserState.data?.let { user -> launchMainActivity(user)
        is DatabaseResult.Failed -> Toast.makeText(baseContext, fUserState.resMessage, Toast
        else -> Unit
    }
}
```

Figura 44 - Funcionalidades Android: Colector de fUserStateFlow de StartActivity

6.3.2. Listados de facultades, salas y reservas

Una vez el usuario ha iniciado sesión se muestra la pantalla principal de la aplicación, consistente en una lista de las facultades correspondientes a la universidad seleccionada. Al pulsar en una facultad se navegará a otra pantalla con otro listado, esta vez de las salas de dicha facultad. La tercera pantalla que contiene un listado es la de reservas, ya sean todas las del usuario, las de una facultad, o las de una sala. Estas tres pantallas pueden observarse en la Figura 45.

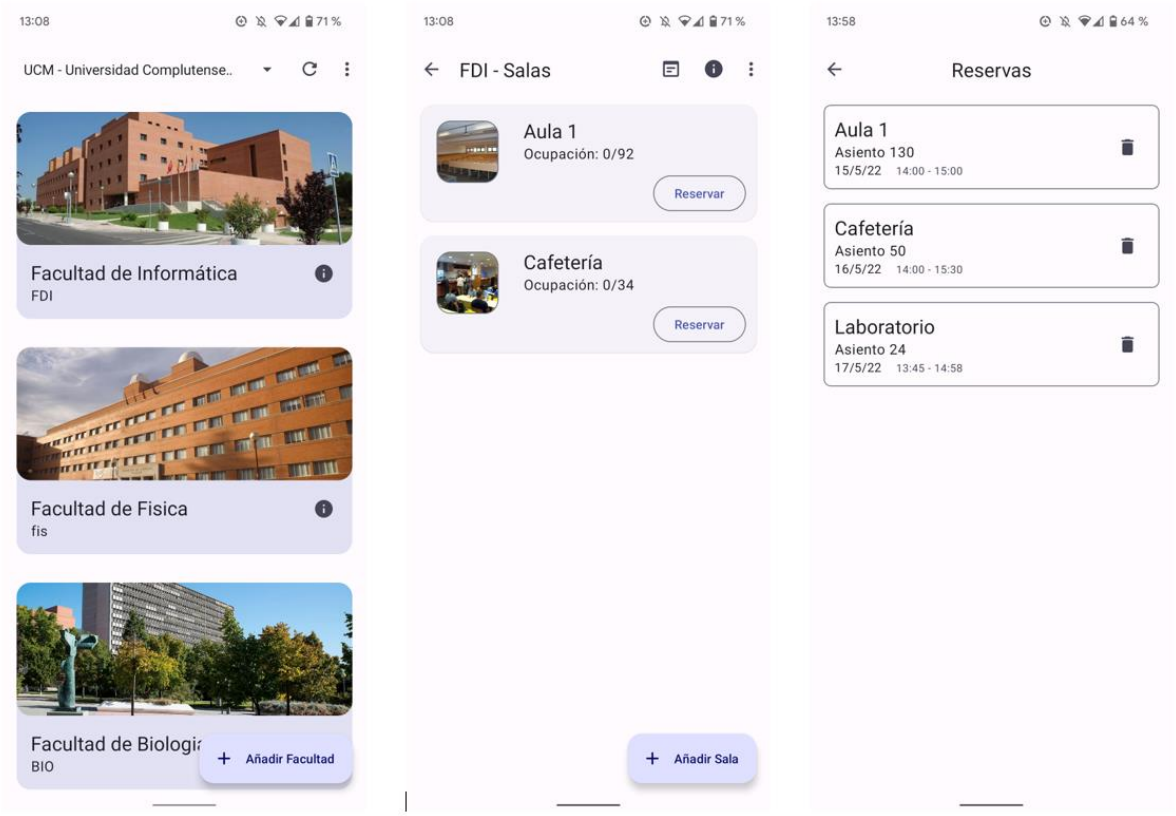


Figura 45 - Funcionalidades Android: Pantallas de listas de la aplicación

Lo que tienen en común estos listados es que están generados con un RecyclerView, un componente gráfico de Android que consiste en una lista optimizada en el que los elementos se reciclan para ahorrar recursos. Estos RecyclerViews dependen de un adaptador, el cual se encarga de insertar cada elemento de la lista en la interfaz gráfica. Se puede ver un ejemplo de un adaptador para el listado de facultades en las Figuras 46 y 47.

```
class FacultiesRecyclerViewAdapter(
    private var facultiesArrayList: ArrayList<Faculty>,
    private val sharedViewModel: SharedViewModel,
    private val facultiesFragment: FacultiesFragment
) : RecyclerView.Adapter<FacultiesRecyclerViewAdapter.ViewHolder>()  {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val binding = FacultyItemBinding.inflate(LayoutInflater.from(parent.context), parent,
        return ViewHolder(binding)
    }

    //Por cada facultad de la lista rellena los datos de cada item con la información de la facultad
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(facultiesArrayList[position])
        holder.binding.executePendingBindings()
    }

    override fun getItemCount() = facultiesArrayList.size
```

Figura 46 - Funcionalidades Android: Clase FacultiesRecyclerViewAdapter (Adaptador de lista de facultades) 1

```
    inner class ViewHolder(val binding: FacultyItemBinding) : RecyclerView.ViewHolder(binding.root)
        fun bind(faculty: Faculty) {
            binding.faculty = faculty
            binding.sModel = sharedViewModel
            binding.fragment = facultiesFragment
        }
    }
}
```

Figura 47 - Funcionalidades Android: Clase FacultiesRecyclerViewAdapter (Adaptador de lista de facultades) 2

A este adaptador le llegan los datos correspondientes a través de un BindingAdapter (ver Figura 48), que se encarga de obtener los datos del elemento RecyclerView de la vista XML (ver Figura 49) y construir el adaptador.

```
@BindingAdapter( ...value: "faculties", "universities", "itUniversity", "sModel", "fragment", requireAll
fun RecyclerView.adapterFaculties(faculties: ArrayList<Faculty>, universities: ArrayList<University>,
                                  itUniversity: Int?, sModel: SharedViewModel, fragment: FacultiesFra
    if (universities.isNotEmpty()) {
        val universityId =
            if (itUniversity == null) universities[0].id
            else universities[itUniversity].id
        val currentFacultiesArrayList = ArrayList(faculties.filter { it.idUniversity == universityId

        if (this.adapter == null) {
            this.layoutManager = LinearLayoutManager(context)
            this.adapter = FacultiesRecyclerViewAdapter(currentFacultiesArrayList, sModel, fragment)
        } else {
            (adapter as FacultiesRecyclerViewAdapter).updateList(currentFacultiesArrayList)
        }
    }
}
```

Figura 48 - Funcionalidades Android: Función BindingAdapter.adapterFaculties de BindingAdapters

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rv_faculties"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="70dp"
    android:clipToPadding="false"
    app:faculties="@{sModel.faculties}"
    app:universities="@{sModel.universities}"
    app:itUniversity="@{sModel.currentUniversityIt}"
    app:sModel="@{sModel}"
    app:fragment="@{fragment}"/>
```

Figura 49 - Funcionalidades Android: Elemento RecyclerView de facultades faculties_fragment.xml

Finalmente, en la vista XML de cada ítem de la lista, gracias al Data Binding, se recogen los datos obtenidos del adaptador y se insertan en los respectivos textos, botones, y demás elementos visuales. En las Figuras 50, 51 y 52 se puede visualizar un ejemplo de ítem de la lista de facultades con sus distintos elementos.

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <data>
        <variable name="fragment" type="com.ajpv.resuniversitas.view.FacultiesFragment" />
        <variable name="sModel" type="com.ajpv.resuniversitas.viewModel.SharedViewModel" />
        <variable name="faculty" type="com.ajpv.resuniversitas.model.Faculty" />
    </data>
```

Figura 50 - Funcionalidades Android: Layout de facultad faculty_item.xml 1

```xml
                <com.google.android.material.textview.MaterialTextView
                    android:id="@+id/tv_name"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="16dp"
                    android:layout_marginTop="16dp"
                    android:maxLines="1"
                    android:ellipsize="end"
                    android:text="@{faculty.name}"
                    android:textAppearance="?attr/textAppearanceTitleLarge"/>

                <com.google.android.material.textview.MaterialTextView
                    android:id="@+id/tv_acronym"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="16dp"
                    android:layout_marginBottom="16dp"
                    android:text="@{faculty.acronym}"
                    android:textAppearance="?attr/textAppearanceBodyLarge" />

            </LinearLayout>
```

Figura 51 - Funcionalidades Android: Layout de facultad faculty_item.xml 2

```xml
            <androidx.appcompat.widget.AppCompatImageButton
                android:id="@+id/btn_info"
                android:layout_width="40dp"
                android:layout_height="40dp"
                android:layout_margin="12dp"
                android:contentDescription="Information"
                android:background="?attr/selectableItemBackgroundBorderless"
                android:backgroundTint="@android:color/transparent"
                android:src="@drawable/ic_round_info_24"
                android:onClick="@{() -> sModel.navigateAndSet(@id/action_facultiesFragment_to_f

        </LinearLayout>
    </LinearLayout>
</com.google.android.material.card.MaterialCardView>
</layout>
```

Figura 52 - Funcionalidades Android: Layout de facultad faculty_item.xml 3

Para obtener estas listas, guardadas en un StateFlow en el ViewModel compartido (ver Figura 53), se realiza una llamada asíncrona a la BBDD desde una clase repositorio con la que se obtienen los documentos solicitados (ver Figura 54). Tal como se muestra en la Figura 54, desde el SharedViewModel se llama a la función del repositorio mencionada anteriormente y el resultado, en este caso la lista de facultades se introduce en el StateFlow correspondiente.

```
private val _faculties = MutableStateFlow<ArrayList<Faculty>>( value: savedStateHandle.get(::faculty
val faculties: StateFlow<ArrayList<Faculty>> = _faculties.asStateFlow()
```

Figura 53 - Funcionalidades Android: Variable StateFlow faculties de SharedViewModel

```
suspend fun fetchFaculties(): DatabaseResult<ArrayList<Faculty>> =
    try { withContext(Dispatchers.IO) {  this: CoroutineScope
        DatabaseResult.success(dbFaculties.get().await().toObjectsArrayList(Faculty::class.java))
    } } catch (e: Exception) { DatabaseResult.failed("Failed to fetch faculties") }
```

Figura 54 - Funcionalidades Android: Función fetchFaculties de UniversityRespository

```
private suspend fun fetchFaculties() {
    val resultFaculties =  universityRepository.fetchFaculties()
    if (resultFaculties is DatabaseResult.Success) {
        _faculties.value = resultFaculties.data
        _currentFaculty.value = faculties.value.firstOrNull { it.id == currentFaculty.value.id } ?: Fac
        savedStateHandle.set(::faculties.name, faculties.value)
        savedStateHandle.set(::currentFaculty.name, currentFaculty.value)
    }
    else if (resultFaculties is DatabaseResult.Failed) _shortToastRes.emit(resultFaculties.resMessage)
}
```

Figura 55 - Funcionalidades Android: Función fetchFaculties de SharedViewModel

6.3.3. Reserva de sitios

Para comenzar con esta función el usuario debe pulsar el botón “Reservar” de una sala, el cuál mostrará una ventana emergente para elegir fecha y horas de la reserva, como se puede ver en la Figura 56.

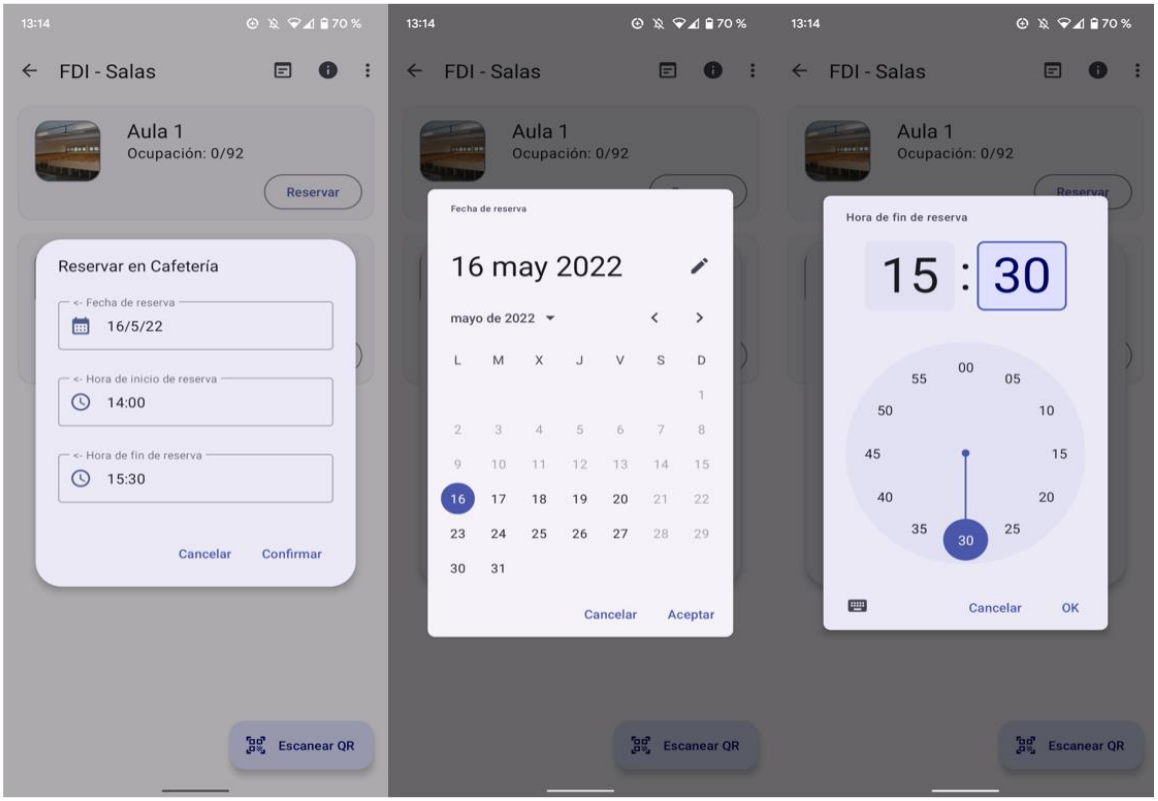


Figura 56 - Funcionalidades Android: Pantallas de reservar en sala

Tal como aparece en la función de las Figuras 57 y 58, una vez comprobado que el usuario no sea de tipo invitado, se configura la funcionalidad de los selectores de fecha y hora para los cuadros de texto de la ventana emergente.

```
val binding = DialogRoomsDateTimeBookingBinding.inflate(activity.layoutInflater)
val alertDialogReserveDateTime = this.createAlertDialog("Reserve in {room.name}",
    ok = "Confirm" to {},
    cancel = "Cancel" to {}
).setView(binding.root).create()

//Fecha de reserva
supportFragmentManager.createEditTextDatePicker(binding.etDate, binding.tilDate, date,
    "Booking date", CustomDateValidator(room, sharedViewModel.currentFaculty.value))
{ dateTimePicked: LocalDateTime ->
    dateTimeStart = dateTimeStart.withDate(dateTimePicked)
    dateTimeEnd = dateTimeEnd.withDate(dateTimePicked)
}

//Hora de inicio de reserva
createEditTextTimePicker(binding.etTimeStart, binding.tilTimeStart,
    time, "Booking start time") { dateTimeStart = dateTimeStart.withTime(it) }

//Hora de fin de reserva
createEditTextTimePicker(binding.etTimeEnd, binding.tilTimeEnd,
    time: "", "Booking end time") { dateTimeEnd = dateTimeEnd.withTime(it) }
```

Figura 57 - Funcionalidades Android: Función FragmentActivity.handleReserveSeatRoom de RoomUtils 1

```
if(room.bookingsByCalendar.size != room.seats.length) {
    //Sobreescribimos el Listener del botón de aceptar para que no cierre el Dialog
    alertDialogReserveDateTime.setOnShowListener {  it: DialogInterface!
        alertDialogReserveDateTime.getButton(AlertDialog.BUTTON_POSITIVE).setOnClickList
            viewModel.validateAndNavigateToReserveSeatRoom(binding.etDate.string, bindir
                binding.etTimeStart.string, binding.tilTimeStart, binding.etTimeEnd.stri
                binding.tilTimeEnd, room, alertDialogReserveDateTime, navActionRes, shar
        }
    }

    alertDialogReserveDateTime.show()  ^let
}
else  {
    sharedViewModel.emitToast("Full Room . Reservation is not possible")  ^let
}
```

Figura 58 - Funcionalidades Android: Función FragmentActivity.handleReserveSeatRoom de RoomUtils 2

Una vez el usuario haya terminado de completar los campos pulsará el botón confirmación que llamará a la función de RoomViewModel encargada de validar los datos y navegar a la siguiente pantalla. Esta función se puede visualizar en la Figura 59.

```
fun validateAndNavigateToReserveSeatRoom(date: String, tilDate: TextInputLayout, timeStart: Stri
                                         tilTimeStart: TextInputLayout, timeEnd: String, tilTime
                                         room: Room, alertDialog: AlertDialog, navActionRes: Int
    val result = ValidationRoomUtil.validateToReserveSeat(
        date = date to tilDate,
        hourStart = timeStart to tilTimeStart,
        hourEnd = timeEnd to tilTimeEnd,
        room = room,
        newsEvents = sModel.newsEvents.value
    )

    if (result is ValidationResult.Success) {
        val newBooking = Booking(
            idRoom = room.id,
            idFaculty = room.idFaculty,
            idUser = sModel.user.value.email,
            timestampStart = dateAndTimeStringToDateTime(date, timeStart).toTimestamp(),
            timestampEnd = dateAndTimeStringToDateTime(date, timeEnd).toTimestamp(),
        )
        sModel.navigateToBookingWithAlertDialog(navActionRes, newBooking, alertDialog)
    } else if (result is ValidationResult.FailedSnackbar) {
        sModel.emitSnackbar(result.snackbar.first, result.snackbar.second)
    }
}
```

Figura 59 - Funcionalidades Android: Función validateAndNavigateToReserveSeatRoom de RoomViewModel

La validación de datos se lleva a cabo en otra clase, con la función que se puede ver en las Figuras 60 y 61. En esta función también se define el error que se debe mostrar en caso de que haya alguno y cómo mostrarlo, si en un mensaje flotante o en alguno de los cuadros de texto.

```
fun validateToReserveSeat(
    date: Pair<String, TextInputLayout>,
    hourStart: Pair<String, TextInputLayout>,
    hourEnd: Pair<String, TextInputLayout>,
    room: Room,
    newsEvents: List<NewsEvent>
) : ValidationResult {
    var result: ValidationResult = ValidationResult.success()

    if (date.second.tilError( /** Error si fecha está vacía */
            date.first.isEmpty(),
            resError = "Field cannot be empty")
    ) result = ValidationResult.failed()

    if (hourStart.second.tilError( /** Error si hora inicio está vacía */
            hourStart.first.isEmpty(),
            resError = "Field cannot be empty")
    ) result = ValidationResult.failed()
    else if (hourStart.second.tilError( /** Error si hora inicio es anterior a hora actual */
            (dateAndTimeStringToDateTime(date.first, hourStart.first) < LocalDateTime.now()),
            resError = "This time must be later than the current time")
    ) result = ValidationResult.failed()
```

Figura 60 - Funcionalidades Android: Función validateToReserveSeat de ValidationRoomUtil 1

```
        /** Error si ya existe una reserva del usuario con esta fecha y hora */
        for (booking in room.bookingsByUser)
            if (isDateTimePickedInsideTimestamps(booking.timestampStart, booking.timestampEnd, d
                return ValidationResult.failedSnackbar("Existing booking (%1$s - %2$s)" to
                    arrayOf(booking.timestampStart.timeString(), booking.timestampEnd.timeString

        /** Error si existe un evento con esta fecha y hora */
        for (newsEvent in newsEvents.filter { it.idRoom == room.id })
            if (isDateTimePickedInsideTimestamps(newsEvent.timestampStart, newsEvent.timestampEn
                && newsEvent.idRoom.isNotEmpty())
                    return ValidationResult.failedSnackbar("There is an event at this time (%1$s
                        arrayOf(newsEvent.timestampStart.timeString(), newsEvent.timestampEnd.ti
    }

    return result
}
```

Figura 61 - Funcionalidades Android: Función validateToReserveSeat de ValidationRoomUtil 2

Si se ha completado la validación de datos sin ningún error se pasa a la función representada en la Figura 62, ubicada en el ViewModel compartido, la cual obtiene las reservas de la BBDD para esa sala y las filtra por el horario escogido para luego introducirlas en el StateFlow de sala actual. Esta variable es la que se va a usar para dibujar el mapa de sitios con las reservas existentes en la siguiente pantalla.

```
if (booking.timestampEnd >= Timestamp.now()) {
    val resultBookings =
        universityRepository.fetchBookingsByStringFieldValue(DbConstants.BOOKING_ID_ROOM, booking.idRo

    if (resultBookings is DatabaseResult.Success) {
        alertDialog?.dismiss()

        if (booking.id.isNotEmpty() && resultBookings.data.firstOrNull { it.id == booking.id } == null
            _shortToastRes.emit("This booking does not exist anymore")
        } else {
            //Filtramos las reservas que se cruzan con las horas que hemos elegido
            val listBookings = resultBookings.data.filter { it: Booking
                isTimestampPickedInsideTimestamps(it.timestampStart, it.timestampEnd, booking.timestam
                        && it.idUser != user.value.email
            }
            if (booking.idRoom.isNotEmpty()) {
                val room = rooms.value.firstOrNull { it.id == booking.idRoom } ?: Room()
                _currentRoom.value = room.copy(bookingsByCalendar = ArrayList(listBookings))
                savedStateHandle.set(::currentRoom.name, currentRoom.value)
            }
            navigateAndSet(navActionRes, booking, edit: false)
        }
    } else if (resultBookings is DatabaseResult.Failed) _shortToastRes.emit(resultBookings.resMessage)
} else _shortToastRes.emit("This booking has already ended")
```

Figura 62 - Funcionalidades Android: Función navigateToBookingWithAlertDialog de SharedViewModel

En la Figura 63 se puede observar cómo queda el mapa de sitios para la reserva en la sala elegida. Este mapa está formado por un ScrollView horizontal y otro vertical anidado, los cuáles se rellenan con vistas que representan cada sitio, mesa o espacio vacío. La función de seleccionar que se puede ver en la Figura 64 es la encargada de representar el sitio seleccionado y repintar el sitio anterior al pulsar en uno nuevo.

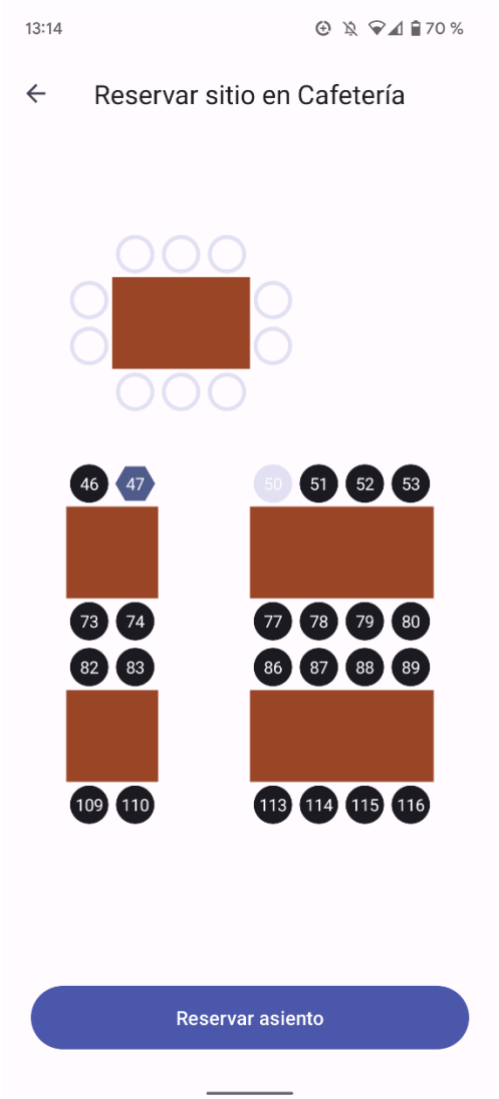


Figura 63 - Funcionalidades Android: Pantalla de realizar reserva en mapa

```
val seatStatus = when (newView.tag) {
    SeatStatusEnum.AVAILABLE -> {
        SeatStatusEnum.SELECTED
    }
    SeatStatusEnum.SELECTED -> {
        SeatStatusEnum.AVAILABLE
    }
    else -> {
        null
    }
}

if (seatStatus != null) {
    if (previousSeatSelectedView != null && seatStatus == SeatStatusEnum.SELECTED) {
        seatsStringBuilder.setCharAt(previousSeatSelectedView.id, SeatStatusEnum.AVAILABLE.char)
        previousSeatSelectedView.setBackgroundResource(SeatStatusEnum.AVAILABLE.iconRes)
        previousSeatSelectedView.tag = SeatStatusEnum.AVAILABLE
    }

    seatsStringBuilder.setCharAt(newView.id, seatStatus.char)
    newView.setBackgroundResource(seatStatus.iconRes)
    newView.tag = seatStatus
}

return seatsStringBuilder.toString()
```

Figura 64 - Funcionalidades Android: Función onClickReserveSeat de RoomViewModel

Para terminar, cuando el usuario pulsa en el botón para confirmar la reserva, una función del ViewModel compartido realiza una llamada a la función representada en la Figura 65 que se encuentra en una clase repositorio. Esta función se encarga de subir a la base de datos la nueva reserva del usuario con el id del sitio seleccionado.

```
suspend fun uploadBooking(newBooking: Booking): DatabaseResult<Booking> =
    try { withContext(Dispatchers.IO) {  this: CoroutineScope
        var newIdBooking = newBooking.id
        if (newBooking.id.isNotEmpty()) //Actualizamos reserva
            dbBookings.document(newBooking.id).set(newBooking).await()
        else //Creamos reserva
            newIdBooking = dbBookings.add(newBooking).await().id

        DatabaseResult.success(newBooking.copy(id = newIdBooking))  ^withContext
    } } catch (e: Exception) { DatabaseResult.failed("Failed to upload booking") }
```

Figura 65 - Funcionalidades Android: Función uploadBooking de UniversityRepository¡

6.3.4. Creación y actualización de facultades, salas y eventos o noticias

Para acceder a la creación o actualización de un objeto de la aplicación se debe o bien mantener pulsado dicho ítem de la lista y hacer clic en “Editar”, o pulsar en el botón flotante “Añadir X” en la pantalla correspondiente. Una vez hecho esto aparecerá una pantalla con un formulario para completar, como ejemplo se puede ver la Figura 66 de modificar facultad.

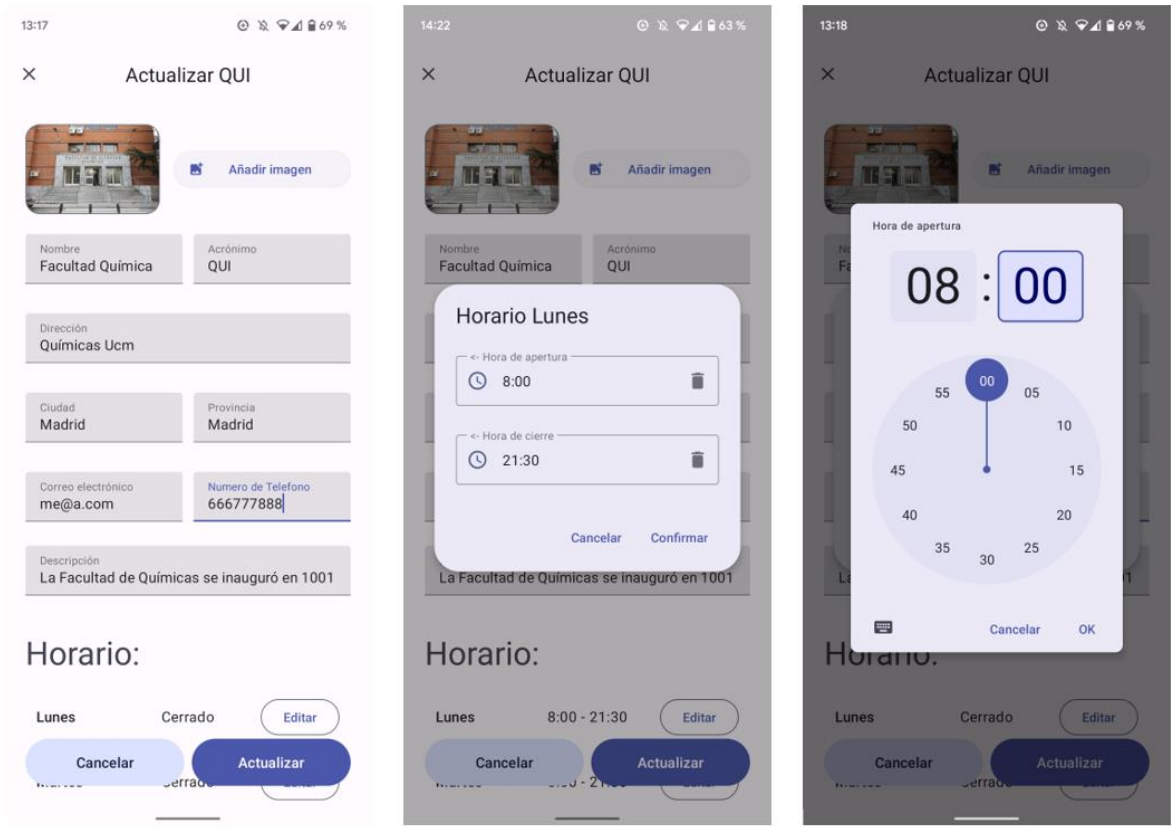


Figura 66 - Funcionalidades Android: Pantallas de modificar facultad

Siguiendo con el ejemplo de modificar facultad, para que el usuario pueda seleccionar una imagen del almacenamiento de su dispositivo se llama a la función de la Figura 67 que se encuentra en la actividad principal de la aplicación.

```
//Se lanza el intent para elegir una imagen del almacenamiento interno
fun selectImageFromGallery() = selectImageFromGalleryResultFlow.launch( input: "image/*")

//Al volver del intent se recupera el uri de la imagen elegida
private val selectImageFromGalleryResultFlow =
    registerForActivityResult(ActivityResultContracts.GetContent()) { uri: Uri? ->
    uri?.let { sharedViewModel.setTempImage(uri) }
}
```

Figura 67 - Funcionalidades Android: Función selectImageFromGallery de MainActivity

Una vez se pulse en el botón de completar la creación se llamará a una función del ViewModel de facultad que se puede ver en la Figura 68. Este se encargará de llamar a la función de validación de las Figuras 69 y 70 ubicada en otra clase.

```
fun validateAndUploadFaculty(tilName: TextInputLayout, tilAcronym: TextInputLayout, tilAddress: TextInputLayout,
                             tilCity: TextInputLayout, tilRegion: TextInputLayout, tilEmail: TextInputLayout,
                             tilDescription : TextInputLayout, ivFaculty: ImageView, sModel: SharedViewModel) {
    val validationResult = ValidationFacultyUtil.validateUpdateFaculty(
        name = (tempFaculty.value.name to tilName),
        acronym = (tempFaculty.value.acronym to tilAcronym),
        address = (tempFaculty.value.address to tilAddress),
        city = (tempFaculty.value.city to tilCity),
        region = (tempFaculty.value.region to tilRegion),
        email = (tempFaculty.value.email to tilEmail),
        description = (tempFaculty.value.description to tilDescription),
        image = ivFaculty
    )

    if (validationResult is ValidationResult.Success) {
        val newFaculty = tempFaculty.value.copy(
            idUniversity = sModel.universities.value[sModel.currentUniversityIt.value].id)
        sModel.uploadFaculty(ivFaculty, newFaculty)
    } else if (validationResult is ValidationResult.FailedToast) {
        sModel.emitToast(validationResult.resMessage)
    }
}
```

Figura 68 - Funcionalidades Android: Función validateAndUploadFaculty de FacultyViewModel

```
fun validateUpdateFaculty(
    name: Pair<String, TextInputLayout>,
    acronym: Pair<String, TextInputLayout>,
    address: Pair<String, TextInputLayout>,
    city: Pair<String, TextInputLayout>,
    region: Pair<String, TextInputLayout>,
    email: Pair<String, TextInputLayout>,
    description: Pair<String, TextInputLayout>,
    image: ImageView
): ValidationResult {
    var result: ValidationResult = ValidationResult.success()

    if (name.second.tilError( /** Error si el nombre está vacío */
            name.first.isEmpty(),
            resError = "Field cannot be empty")
    ) result = ValidationResult.failed()

    if (acronym.second.tilError( /** Error si el acrónimo está vacío */
            acronym.first.isEmpty(),
            resError = "Field cannot be empty")
    ) result = ValidationResult.failed()
```

Figura 69 - Funcionalidades Android: Función validateUpdateFaculty de ValidationFacultyUtil 1

```
    if (email.second.tilError( /** Error si el formato de email es incorrecto cuando no esta vac
            (email.first.isNotEmpty() && !android.util.Patterns.EMAIL_ADDRESS.matcher(email.firs
            resError = "Email address is not valid")
    ) result = ValidationResult.failed()

    if (description.second.tilError( /** Error si la descripción está vacía */
            description.first.isEmpty(),
            resError = "Field cannot be empty")
    ) result = ValidationResult.failed()

    if (image.drawable == null) /** Error si no hay imagen */
        return ValidationResult.failedToast("Image not selected")

    return result
}
```

Figura 70 - Funcionalidades Android: Función validateUpdateFaculty de ValidationFacultyUtil 2

Cuando se haya realizado la validación correctamente, en cuyo caso contrario se mostrará el error correspondiente, se llamará a la función de la Figura 71 en el ViewModel compartido. Esta hará una llamada a una función de una clase repositorio que se puede ver en las Figuras 72 y 73 que se encarga de subir o actualizar el objeto facultad que ha obtenido junto a su imagen mediante llamadas asíncronas a la base de datos.

```
fun uploadFaculty(ivFaculty: ImageView, newFaculty: Faculty) {
    viewModelScope.launch {  this: CoroutineScope
        _loading.emit( value: true)
        val resultFaculties = universityRepository.uploadFacultyAndImage(newFaculty, ivFaculty,  isNewImageUplo
        if (resultFaculties is DatabaseResult.Success)
            listOf(
                async { fetchRooms() },
                async { fetchUserByEmail(user.value.email) }
            ).awaitAll()
        _loading.emit( value: false)

        if (resultFaculties is DatabaseResult.Success) {
            setTempImage(Uri.EMPTY)
            _faculties.value = resultFaculties.data
            savedStateHandle.set(::faculties.name, faculties.value)
            _shortToastRes.emit("Faculty updated")
            navigate(R.id.action_createFacultyFragment_to_facultiesFragment)
        } else if (resultFaculties is DatabaseResult.Failed) _shortToastRes.emit(resultFaculties.resMessage)
    }
}
```

Figura 71 - Funcionalidades Android: Función uploadFaculty de SharedViewModel

```
suspend fun uploadFacultyAndImage(newFaculty: Faculty, ivFaculty: ImageView, isNewImageUploaded: Boolean): DatabaseResult<Arra
    try { withContext(Dispatchers.IO) {  this: CoroutineScope
        val deferreds = ArrayList<Deferred<Any>>()
        if (isNewImageUploaded) { //Si hay nueva imagen se actualiza y se borra la anterior
            val byteArray = ivFaculty.getByteArray( quality: 75)
            val imageRef = storage.getImageRef(DbConstants.COLLECTION_FACULTIES, newFaculty.name)

            if (newFaculty.imageUrl.isNotEmpty()) //Si se está actualizando la imagen se elimina la anterior
                deferreds.add(async { storage.getReferenceFromUrl(newFaculty.imageUrl).delete().await() })

            deferreds.add(
                async { //Subimos la nueva imagen
                    imageRef.putBytes(byteArray).await()
                    val newImageUrl = imageRef.downloadUrl.await().toString()
                    //Una vez subida la imagen se sube la faultad con la URL obtenida
                    newFaculty.imageUrl = newImageUrl
                }
            )
        }
        deferreds.awaitAll() //Esperamos a que se complete la subida de imagen para subir la facultad
        deferreds.removeAll(deferreds)
```

Figura 72 - Funcionalidades Android: Función uploadFacultyAndImage de UniversityRepository 1

```
    if (newFaculty.id.isNotEmpty()) { //Actualizamos facultad
        deferreds.add(async { dbFaculties.document(newFaculty.id).set(newFaculty).await() })
        deferreds.add(
            async { //Eliminamos las reservas que estén en días festivos
                val batch = db.batch()
                dbBookings.whereEqualTo(BOOKING_ID_FACULTY, newFaculty.id)
                    .get().await().forEach { bookingDocument ->
                        if (isTimestampDateInTimestampsDates(newFaculty.holidays,
                                bookingDocument.toObject(Booking::class.java).timestampStart)
                        )
                            batch.delete(bookingDocument.reference)
                    }
                batch.commit().await()  ^async
            }
        )
    } else //Creamos facultad
        deferreds.add(async { dbFaculties.add(newFaculty).await() })

    deferreds.awaitAll()

    fetchFaculties()  ^withContext
} } catch (e: Exception) { DatabaseResult.failed("Failed to upload faculty") }
```

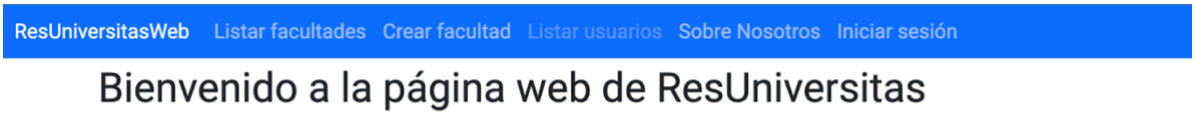
Figura 73 - Funcionalidades Android: Función uploadFacultyAndImage de UniversityRepository 2

6.4. Funcionalidad de la aplicación web

En este apartado se desarrollan algunas de las funcionalidades más importantes que se han implementado en la aplicación web, acompañadas de sus respectivas vistas.

6.4.1. Pantalla de Bienvenida

Al acceder a la aplicación por primera vez se abrirá una pantalla de bienvenida como se muestra en la Figura 74 con una barra de navegación.



Bienvenido a la página web de ResUniversitas

Figura 74 - Funcionalidades web: Pantalla de bienvenida

En el modelo de programación utilizado en el desarrollo de la aplicación web, se han formado las pantallas de la aplicación modulables y reutilizables. Esto quiere decir, que se han utilizado ficheros denominados views para englobar los distintos componentes reutilizables.

Como se muestra en la Figura 75, el fichero index-view.component.html realiza una inclusión del componente del fichero index.component.html denominado en la Figura 76 con el nombre app-index como ocurre en la especificación del resto de componentes.

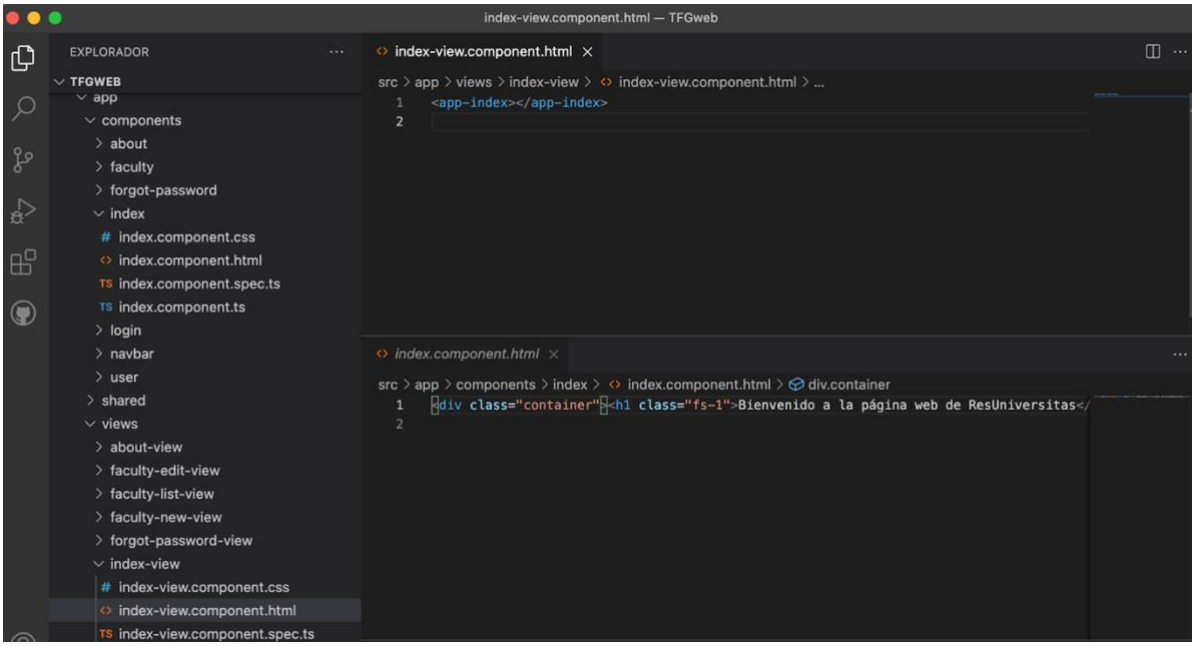


Figura 75 - Funcionalidades web: Componentes reutilizables

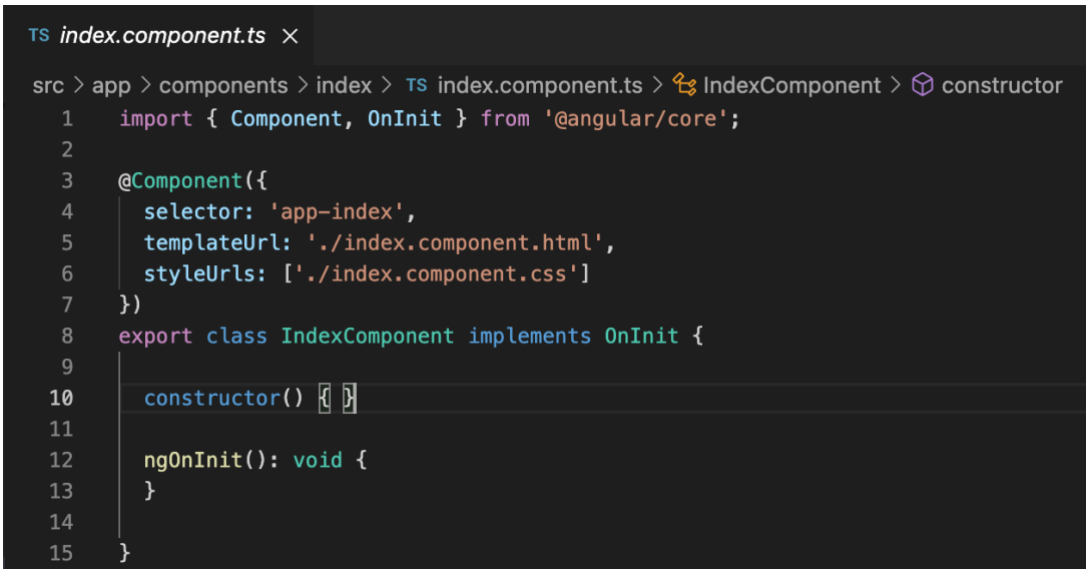


Figura 76 - Funcionalidades web: Fichero Typescript que especifica el componente app-index

6.4.2. Inicio de sesión

Para acceder a las funcionalidades de la aplicación es necesario que se inicie sesión con un usuario con perfil administrador en la pantalla de inicio de sesión que se muestra en la Figura 77.

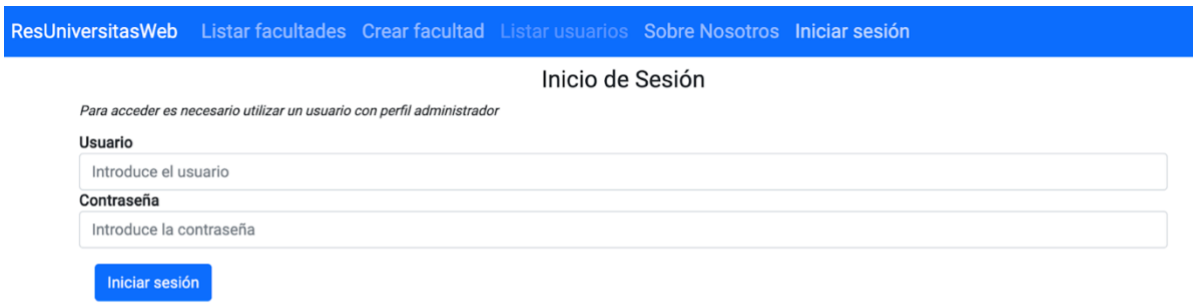


Figura 77 - Funcionalidades web: Pantalla de inicio de sesión

La comprobación del usuario se realiza como se muestra en la Figura 78 haciendo uso de un servicio en el que implementan las funciones de Firestore Authentication como se muestra en las Figuras 79 y 80.

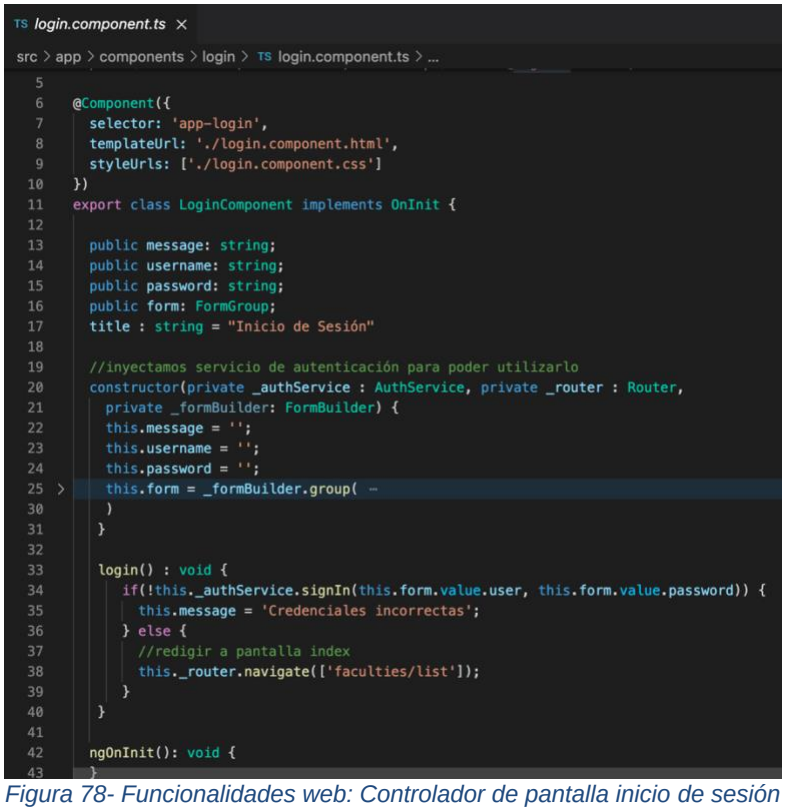


Figura 78- Funcionalidades web: Controlador de pantalla inicio de sesión

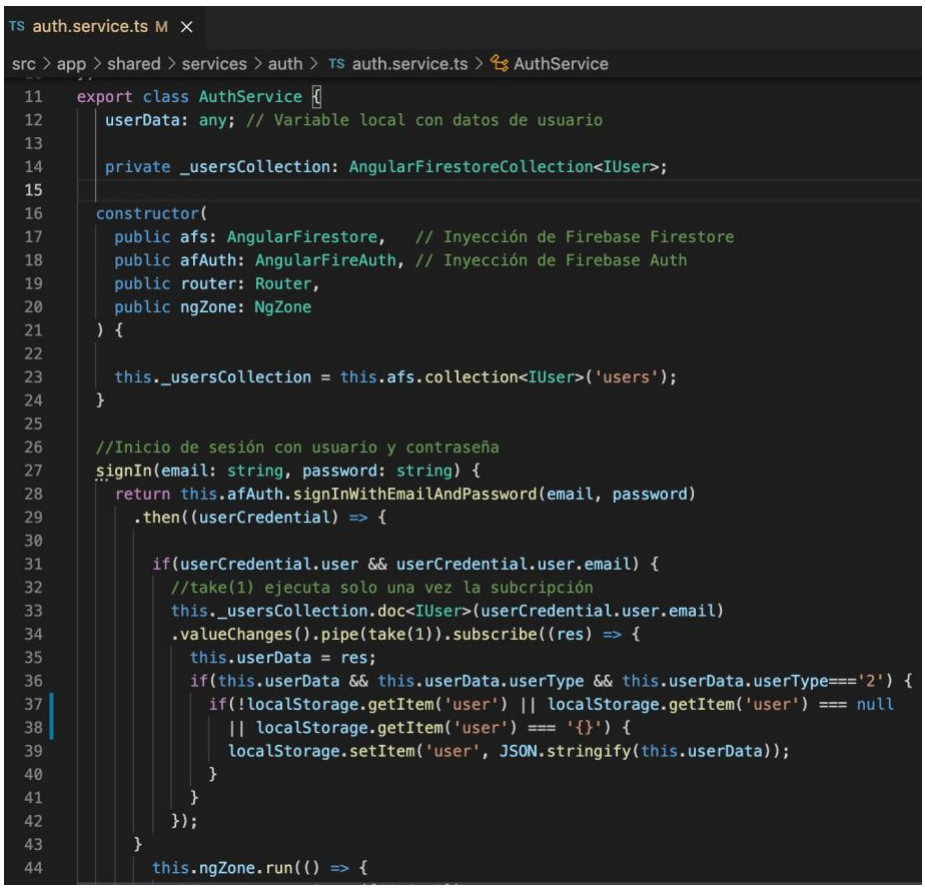


Figura 79 - Funcionalidades web: Servicio de implementación de funciones de Authentication 1

Figura 80 - Funcionalidades web: Servicio de implementación de funciones de Authentication 2

6.4.3. Listado de facultades

La pantalla de facultades se visualiza como se observa en la Figura 81 a través de un listado utilizando la directiva de Angular *ngFor como se muestra en la Figura 82.

Figura 81 - Funcionalidades web: Pantalla de facultades

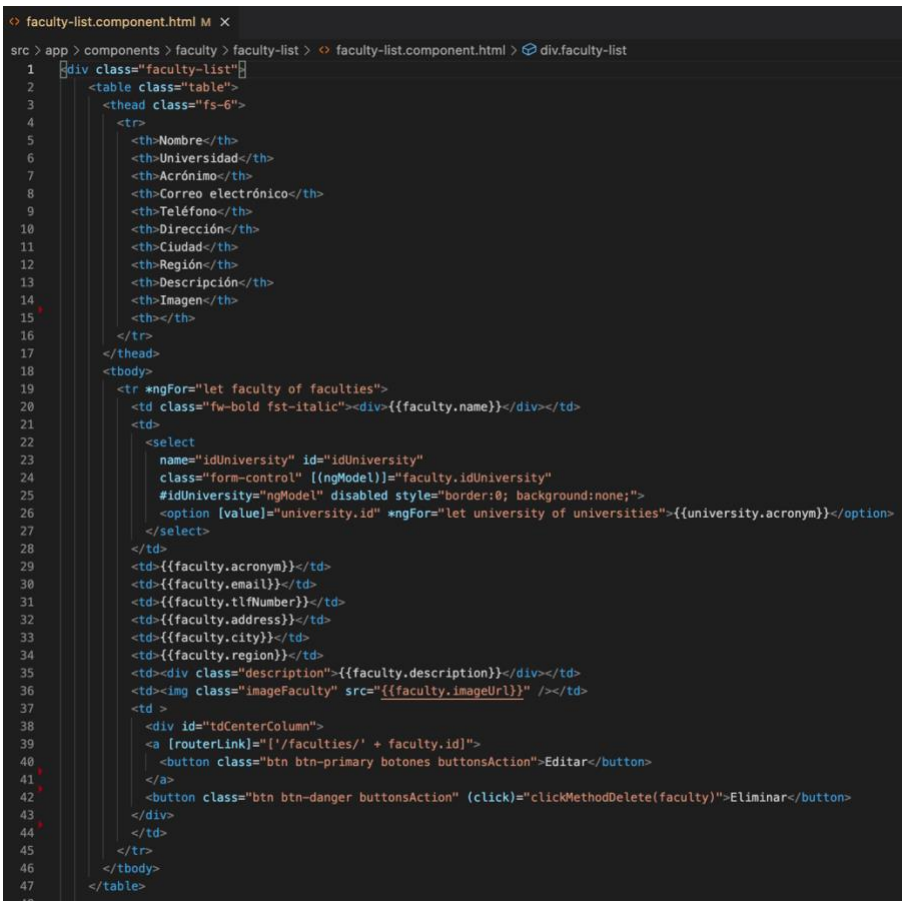


Figura 82 - Funcionalidades web: Fichero HTML de listar facultades

En la Figura 83 se especifica cómo se cargan todas las facultades a través de un servicio que inicializa una lista de observables para cada tipo de tabla consultada de Firestore Database como se muestra en la Figura 84.
En la Figura 83 también se muestra el método utilizado para eliminar facultades solicitando una confirmación del usuario a través de un mensaje del navegador como se muestra en la Figura 85.

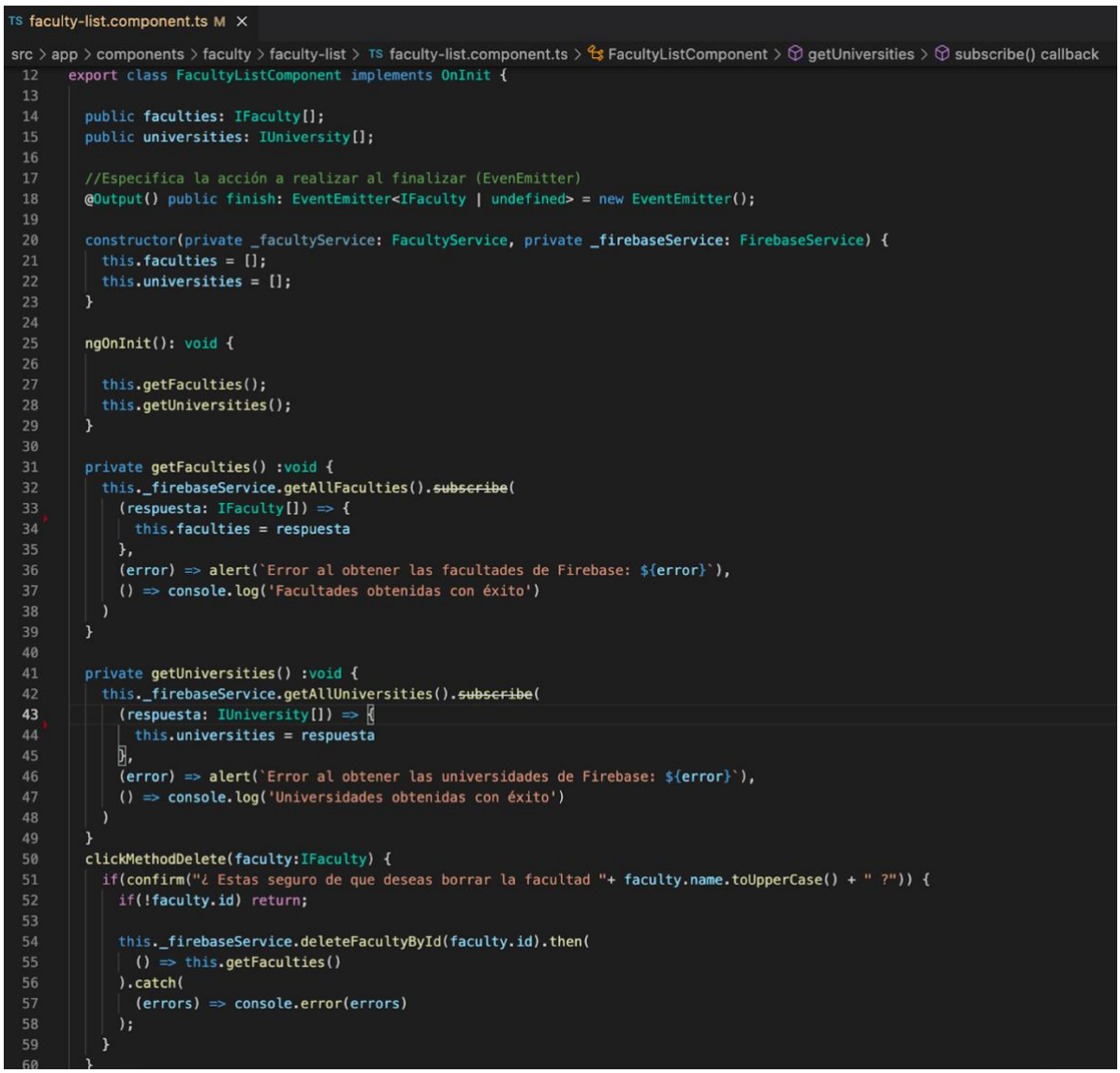


Figura 83 - Funcionalidades web: Typescript que controla la pantalla de listar facultades

```
export class FirebaseService {

  private _facultiesCollection: AngularFirestoreCollection<IFaculty>;
  private _faculties$: Observable<IFaculty[]>;
  ...

  constructor(private _db: AngularFirestore, private _uploadService: FileUploadService) {
    this._facultiesCollection = this._db.collection<IFaculty>('faculties');

  //Binding automatizado con los cambios en la colección de contactos de firestore
    this._faculties$ = this._facultiesCollection.snapshotChanges().pipe(
      map(actions => {
        return actions.map(action => {
          const data = action.payload.doc.data(); // los datos de documento, sin el ID
          const id = action.payload.doc.id; // ID del documento
          return { id, ...data }
        })
      })
    );
  ...
  }

  getAllFaculties(): Observable<IFaculty[]>{
    return this._faculties$;
  }

  getFacultyById(id: string): Observable<IFaculty | undefined>{
    return this._facultiesCollection.doc<IFaculty>(id).valueChanges();
  }

  createFaculty(faculty: IFaculty): Promise<DocumentReference<IFaculty>>{
    return this._facultiesCollection.add(faculty);
  }

  updateFacultyById(id: string | undefined, faculty: IFaculty): Promise<void>{
    return this._facultiesCollection.doc<IFaculty>(id).update(faculty);
  }
```

Figura 84 - Funcionalidades web: Operaciones de obtención, actualización y creación de datos

Figura 85 - Funcionalidades web: Mensaje de confirmación de eliminación de facultad

En las Figuras 86 y 87 se muestra la función que elimina las facultades de Firestore Database realizando una eliminación de todos los datos dados de alta y que hagan referencia a la facultad.

```
async deleteFacultyById(id: string){

  //Contains refereces of rooms
  await this._roomsCollection.ref.where('idFaculty', '==', id).get().then(async querySnapshotRooms => {
      if (querySnapshotRooms.empty) {
          console.log('Rooms of faculty no data found');
      } else {
        await querySnapshotRooms.forEach(async querySnapshotRoom => {
          //OCCUPANCIES ROOM
          await this._occupanciesCollection.ref.where('idRoom', '==', querySnapshotRoom.id).get().then(async querySnapshotOccupancies => {
            if (querySnapshotOccupancies.empty) {
                console.log('Occupancies of room no data found');
            } else {
              await querySnapshotOccupancies.forEach(async documentSnapshotOccupancies => {
                await this._occupanciesCollection.doc(documentSnapshotOccupancies.id).delete();
              });
            }
          });

          //NEWSEVENTS ROOM
          await this._eventsCollection.ref.where('idRoom', '==', querySnapshotRoom.id).get().then(async querySnapshotEvents => {
            if (querySnapshotEvents.empty) {
                console.log('Events of room no data found');
            } else {
              await querySnapshotEvents.forEach(async querySnapshotEvents => {
                await this._eventsCollection.doc(querySnapshotEvents.id).delete();
              });
            }
          });

          //BOOKINGS ROOM
          await this._bookingsCollection.ref.where('idRoom', '==', querySnapshotRoom.id).get().then(async querySnapshotBookings => {
            if (querySnapshotBookings.empty) {
                console.log('Events of room no data found');
            } else {
              await querySnapshotBookings.forEach(async querySnapshotBookings => {
                await this._bookingsCollection.doc(querySnapshotBookings.id).delete();
              });
            }
          });

          //comprobar si se elimina bien o cuando cambia el contenido
          await this._roomsCollection.doc<IRoom>(querySnapshotRoom.id).valueChanges().subscribe(async roomSnapshot => {
            //IMAGE ROOM
            if(roomSnapshot?.imageUrl) {
              await this._uploadService.deleteFileURL(roomSnapshot?.imageUrl);
            }
          });
```

Figura 86 - Funcionalidades web: Operación de eliminación de facultades 1

```
            //ROOM
            await this._roomsCollection.doc(querySnapshotRoom.id).delete();
          });
        }
      });

  //NEWSEVENTS FACULTY
  await this._eventsCollection.ref.where('idFaculty', '==', id).get().then(async querySnapshotEvents => {
    if (querySnapshotEvents.empty) {
        console.log('Events of faculty no data found');
    } else {
      await querySnapshotEvents.forEach(async querySnapshotEvents => {
        await this._eventsCollection.doc(querySnapshotEvents.id).delete();
      });
    }
  });

  //BOOKINGS FACULTY
  await this._bookingsCollection.ref.where('idFaculty', '==', id).get().then(async querySnapshotBookings => {
    if (querySnapshotBookings.empty) {
        console.log('Events of faculty no data found');
    } else {
      await querySnapshotBookings.forEach(async querySnapshotBookings => {
        await this._bookingsCollection.doc(querySnapshotBookings.id).delete();
      });
    }
  });

  //comprobar si se elimina bien o cuando cambia el contenido
  await this._facultiesCollection.doc<IFaculty>(id).valueChanges().subscribe(async facultySnapshot => {
    //IMAGE FACULTY
    if(facultySnapshot?.imageUrl) {
      await this._uploadService.deleteFileURL(facultySnapshot?.imageUrl);
    }
  });

  //FACULTY
  return await this._facultiesCollection.doc<IFaculty>(id).delete();
}
```

Figura 87 - Funcionalidades web: Operación de eliminación de facultades 2

6.4.4. Crear facultad

La pantalla de alta de facultades se muestra en las Figuras 88 y 89 con las validaciones de los campos activas de manera que se puede observar a simple vista los campos obligatorios. Las validaciones utilizadas se especifican en cada elemento a través de la funcionalidad que ofrece Angular como se muestra en la Figura 90.

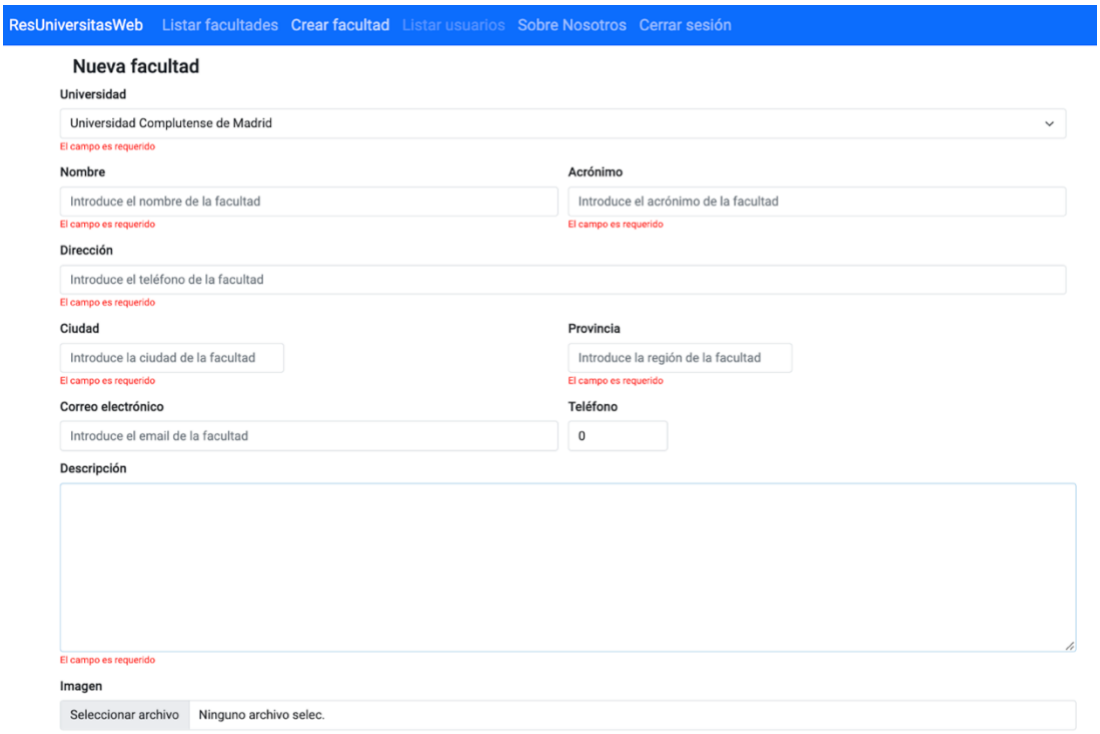


Figura 88 - Funcionalidades web: Pantalla de alta de facultades 1

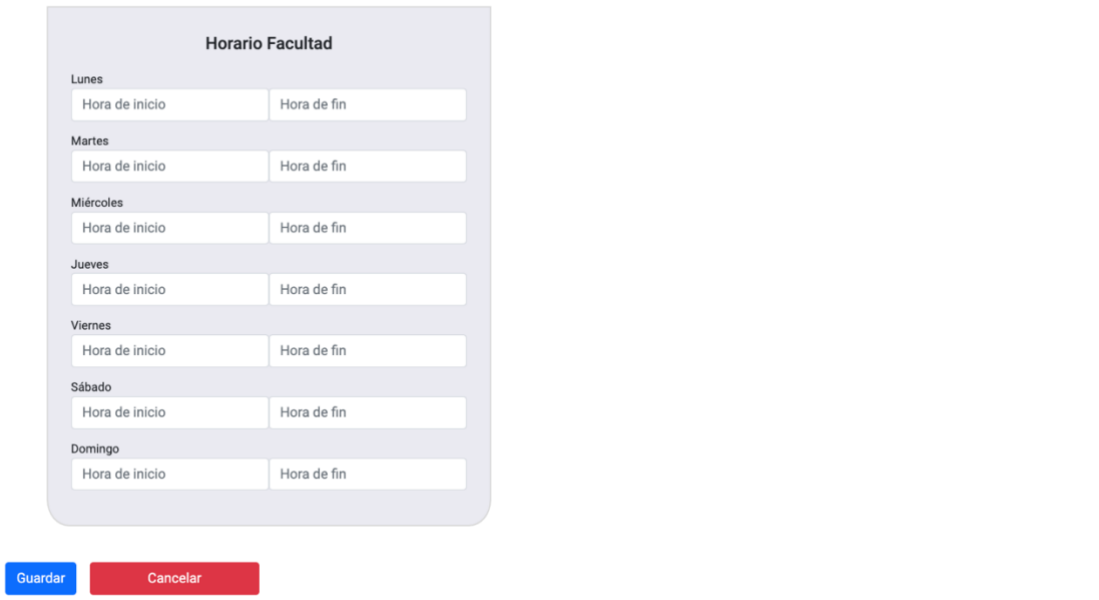


Figura 89 - Funcionalidades web: Pantalla de alta de facultades 2

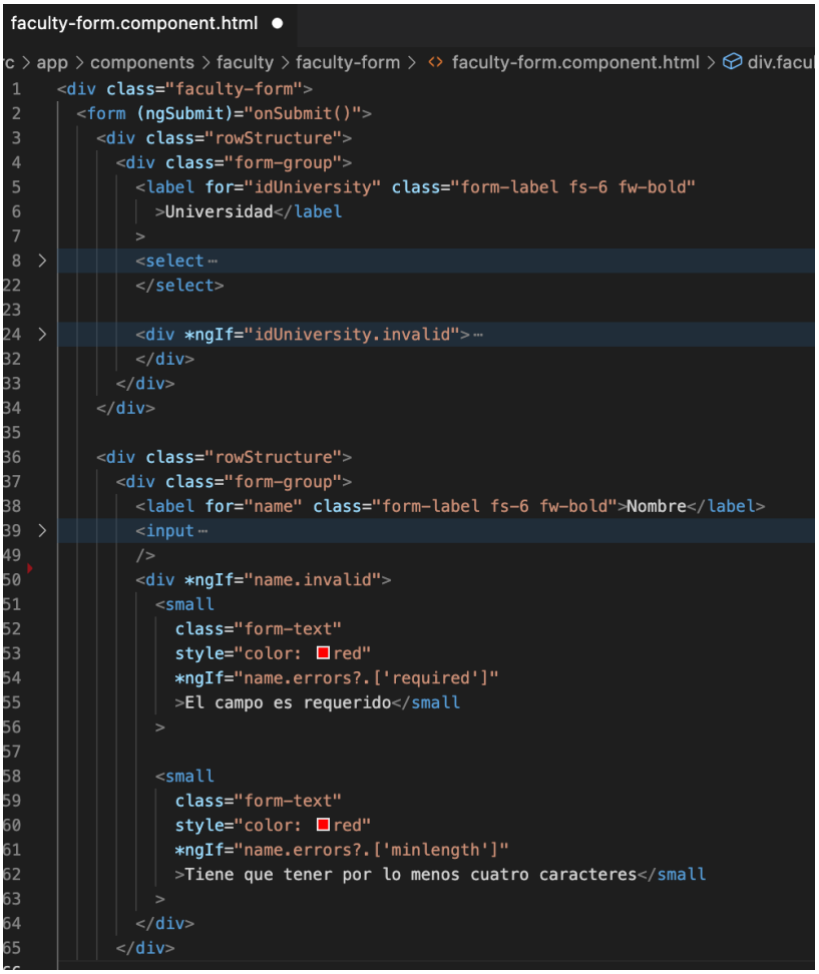


Figura 90 - Funcionalidades web: Fichero HTML de formulario de facultades

Esta pantalla se carga desde el componente faculty-new-view.component.html (ver Figura 91) que carga a su vez el componente app-faculty-form (formulario de facultades utilizado en las pantallas de alta y modificación) con el título de “Nueva facultad” y un método de regreso al listado al finalizar la acción de volver o guardar.

Figura 91 - Funcionalidades web: Fichero de vista que carga el componente de formulario

En el formulario se muestra una previsualización de la imagen seleccionada (ver Figura 92) que debe de tener un tamaño inferior a 1 MB o de la imagen almacenada previamente en Storage de Firebase se está modificando una facultad.

Figura 92 - Funcionalidades web: Imagen seleccionada

Las validaciones de la imagen se muestran en las Figuras 93 y 94.

```
<div class="form-group" id="divUploadImage">
  <label class="form-label fs-6 fw-bold" for="imageUrl">Imagen</label>
  <input
    type="file"
    class="form-control text-nowrap text-truncate"
    id="imageUrl"
    placeholder="Selecciona la imagen la facultad"
    (change)="selectFile($event)"
    name="imageUrl"
    accept=".jpg, .jpeg, .png"
    required
    maxlength="1024"
  />

  <div *ngIf="imageUrl.invalid">
    <small
      class="form-text"
      style="color: red"
      *ngIf="imageUrl.errors?.['required']"
      >El campo es requerido</small
    >

    <small
      class="form-text"
      style="color: red"
      *ngIf="imageUrl.errors?.['maxlength']"
      >Tiene que tener un tamaño inferior a 1MB</small
    >
  </div>

  <div [hidden]="viewImageHidden" id="centrarImagenFacultad">
    <input
      type="hidden"
      class="form-control"
      id="imageUrl"
      [(ngModel)]="faculty.imageUrl"
      name="{faculty.name}}"
      #imageUrl="ngModel"
    />

    <img id="imagenFacultad" src="{{ faculty.imageUrl }}" />
  </div>
</div>
```

Figura 93 - Funcionalidades web: Fichero HTML de formulario de facultades (elemento imagen)

```
selectFile(event: any): void {
  if (event.target.files[0].size < 1048576) {
    this.faculty.imageUrl = event.target.files[0].name;
    let fileToUpload = event.target.files[0];

    //Show image preview
    let reader = new FileReader();
    reader.onload = (event: any) => {
      this.faculty.imageUrl = event.target.result;
      this.viewImageHidden = false;
    };
    reader.readAsDataURL(fileToUpload);

    this.selectedFiles = event.target.files;
  } else {
    alert(`El tamaño tiene que ser inferior a 1 MB`);
  }
}
```

Figura 94 - Funcionalidades web: Validación de tamaño al seleccionar la imagen de la facultad

Al guardar el formulario se realiza primero un almacenamiento de la imagen seleccionada en Storage y posteriormente se almacena la URL pública de la misma en la facultad como se muestra en la Figura 95. Estas últimas operaciones se realizan en un servicio que implementa las funciones de Firebase Storage como se muestra en la Figura 96.

```
async onSubmit(): Promise<void> {
  registerLocaleData(es);
  if (!this.facultyId) {
    if (this.selectedFiles) {
      const file: File | null = this.selectedFiles.item(0);
      this.selectedFiles = undefined;
      if (file) {
        this.currentFileUpload = new FileUpload(file);
        this.currentFileUpload.name = formatDate(
          new Date(),
          "yyyyMMddHHmmss",
          "es"
        );

        await (
          await this._uploadService.pushFileToStorage(this.currentFileUpload)
        ).subscribe((downloadURL) => {
          if (downloadURL) {
            this.faculty.imageUrl = downloadURL;
            this._firebaseService
              .createFaculty(this.faculty)
              .then(() => this.finish.emit())
              .catch((errors) => {
                console.error(errors);
                //eliminar foto
                this.deleteFile(this.currentFileUpload);
              });
          }
        });
      }
    }
  } else {
```

Figura 95 - Funcionalidades web: Método de guardar nueva facultad

```
//Subir y descargar archivos con Firebase Storage
async pushFileToStorage(fileUpload: FileUpload): Promise<Observable<any>> {
  const filePath = `${this._basePath}/${fileUpload.name}`;
  const storageRef = await this._storage.ref(filePath);
  const uploadTask = await this._storage.upload(filePath, fileUpload.file);

  return storageRef.getDownloadURL();
}
```

Figura 96 - Funcionalidades web: Implementación de subida de una imagen en Storage

6.4.5. Modificar facultad

La pantalla de alta de facultades (ver Figuras 97 y 98) se carga desde el componente de vista de editar facultar como se muestra en la Figura 99.

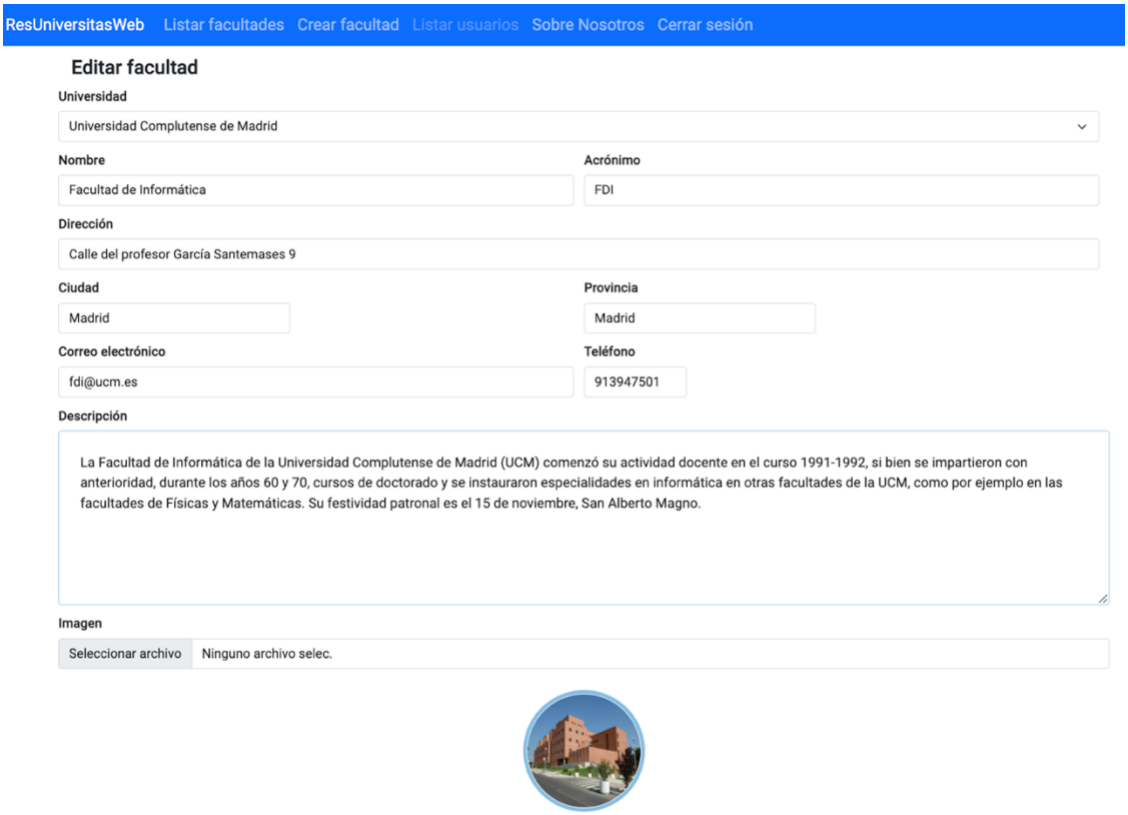


Figura 97 - Funcionalidades web: Pantalla de modificación de facultad 1

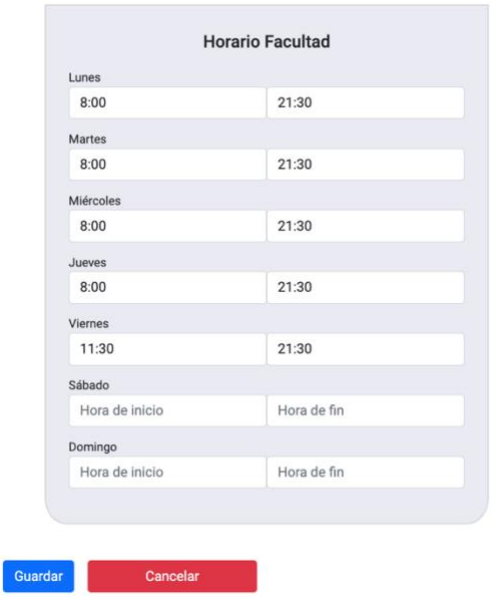


Figura 98 - Funcionalidades web: Pantalla de modificación de facultad 2

Figura 99 - Funcionalidades web: Fichero de vista que carga el componente de formulario

A la hora de actualizar los datos de una facultad, se utiliza un método compartido con el alta de las facultades como se muestra en la Figura 100 desde donde se elimina inicialmente la imagen de Storage a través de la URL pública utilizando el método de la Figura 101 que el nombre de la imagen a eliminar. Posteriormente, se sube la nueva imagen a Storage.

faculty-form.component.ts

src > app > components > faculty > faculty-form > faculty-form.component.ts > FacultyFormComponent

```
104     registerLocaleData(es);
105     if (!this.facultyId) {…
134     } else {
135       if (this.urlImage) {
136         await this._uploadService.deleteFileURL(this.urlImage);
137       }
138
139       if (this.selectedFiles) {
140         const file: File | null = this.selectedFiles.item(0);
141         this.selectedFiles = undefined;
142         if (file) {
143           this.currentFileUpload = new FileUpload(file);
144           this.currentFileUpload.name = formatDate(
145             new Date(),
146             "yyyyMMddHHmmss",
147             "es"
148           );
149
150           await (
151             await this._uploadService.pushFileToStorage(this.currentFileUpload)
152           ).subscribe((downloadURL) => {
153             if (downloadURL) {
154               this.faculty.imageUrl = downloadURL;
155               this._firebaseService
156                 .updateFacultyById(this.facultyId, this.faculty)
157                 .then(() => this.finish.emit())
158                 .catch((errors) => {
159                   console.error(errors);
160                   //eliminar foto
161                   this.deleteFile(this.currentFileUpload);
162                 });
163             }
164           });
165         }
166       }
167
168       //this._facultyServide.updateFaculty(this.faculty).subscribe((res) => this.finish.emit(res));
169       this._firebaseService
170         .updateFacultyById(this.facultyId, this.faculty)
171         .then(() => this.finish.emit())
172         .catch((error) =>
173           alert(`Error al actualizar la facultad de Firebase: ${error}`)
174         ); //.finally()
175     }
176   }
```

Figura 100 - Funcionalidades web: Método de actualizar facultad que actualiza la imagen en Storage

```
//Eliminar archivo del almacenamiento de Firebase
async deleteFileURL(filePathURL: string): Promise<void> {
  var name = filePathURL.substring(filePathURL.search('images%2Ffaculties%2F')+('images%2Ffaculties%2F').length, filePathURL.search('alt=media')-1).toString();

  const filePath = `${this._basePath}/${name}`;
  const storageRef = (await this._storage.ref(filePath));
  await storageRef.delete();
}
```

Figura 101 - Funcionalidades web: Implementación de eliminación de una imagen en Storage

Capítulo 7. Evaluación de usabilidad

En este capítulo se han realizado unas encuestas para captar la valoración de algunos usuarios sobre la usabilidad que presenta esta aplicación.

7.1. Metodología

Para llevar a cabo esta valoración se ha realizado una encuesta mediante la herramienta de Google Forms. Se han elaborado unas preguntas utilizando la escala de Likert. En estas preguntas, se plantea a los usuarios una afirmación para que ellos puedan mediante una valoración numérica del uno al cinco, establecer el grado de conformidad que tienen con la afirmación presentada. Con este método se ha desarrollado un total de veintisiete preguntas centradas en valorar si la aplicación es intuitiva en cuanto al acceso a las distintas utilidades que esta presenta. Tales son como ejemplo si ha sido sencillo acceder a vistas de información de una facultad, utilizar los QR 's para distintos casos, etc.

7.2. Resultados

Comprobando los resultados que hemos obtenido como conclusión se obtiene que la aplicación por lo general es bastante intuitiva para los usuarios. Los usuarios encuestados han sido en su mayoría alumnos de la facultad de informática de la Complutense, y en menor medida algunas personas de mayor edad.

Como resultados generales se puede observar en la Figura 102 que la valoración es más que positiva, puesto que supone en su mayoría una media de valoración por encima de los 4 puntos. Después de este análisis general se pasa a analizar algunos rasgos más específicos de la aplicación.

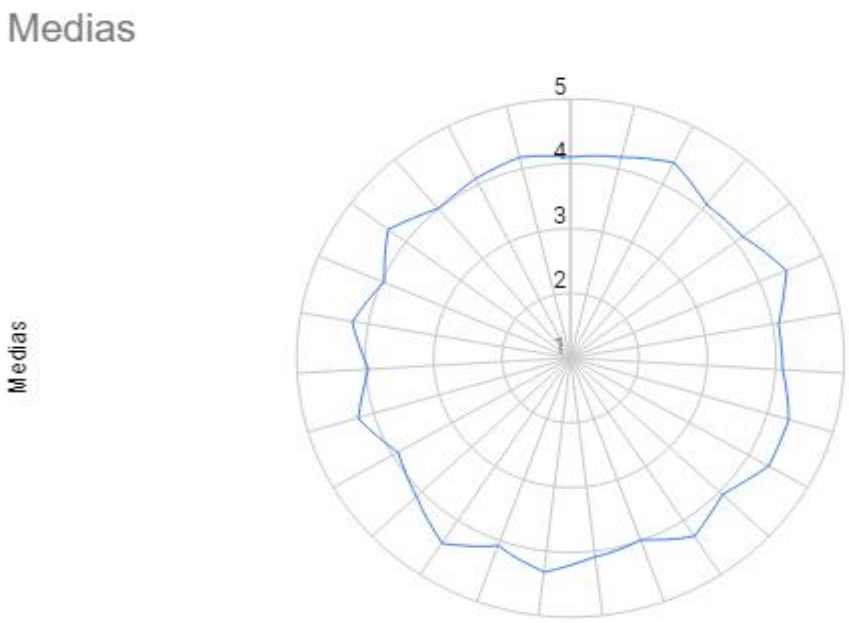


Figura 102 - Media de respuestas del formulario

7.2.1. Registro en la aplicación

Sobre este punto, como se puede ver en la Figura 103 los usuarios han valorado como perfectamente intuitiva la funcionalidad de registro en la aplicación. Un 94.4% de los usuarios ha opinado que el registro en sí es perfecto.

Figura 103 - Registro en la aplicación

7.2.2. Escáner QR

Las valoraciones obtenidas sobre las funcionalidades relacionadas con los códigos QR son las que han obtenido una valoración más baja, siendo aun así elevada. Este resultado puede deberse a la presencia de usuarios de mayor edad que probaron la aplicación. Se puede comprobar en las Figura 104, 105 y 106.En los tres casos las valoraciones situadas en un 3 se encuentran por debajo del 50% siendo su rango máximo un 44.4 % de los usuarios en la cuestión de acceder a la información de una facultad y el 33.3% en el caso de hacer check-in, por supuesto en las tres figuras no se aprecia, pero los usuarios que valoran de un 4 a un 5 se encuentran en todas por encima del 50%.

Figura 104 - Escáner QR 1

Figura 105 - Escáner QR 2

Figura 106 - Escáner QR 3

7.2.3. Creación de mapas

Otra de las funcionalidades que ha resultado algo más compleja de utilizar para algunos usuarios ha sido la generación de mapas de salas. Como se puede observar en las Figuras 107 y 108, a los usuarios les ha resultado más difícil el primer contacto con la funcionalidad de mapas de sala, ya que luego los resultados mejoran para la opción de modificar el mapa de sitios. En el caso de la creación de mapas los valores se situaron en 3 para un 44.4% de

los usuarios, mientras que en la modificación ese valor se reduce a cero y se reparte entre un 4 y un 5, siendo el 4 la respuesta del 55,6% de los usuarios.

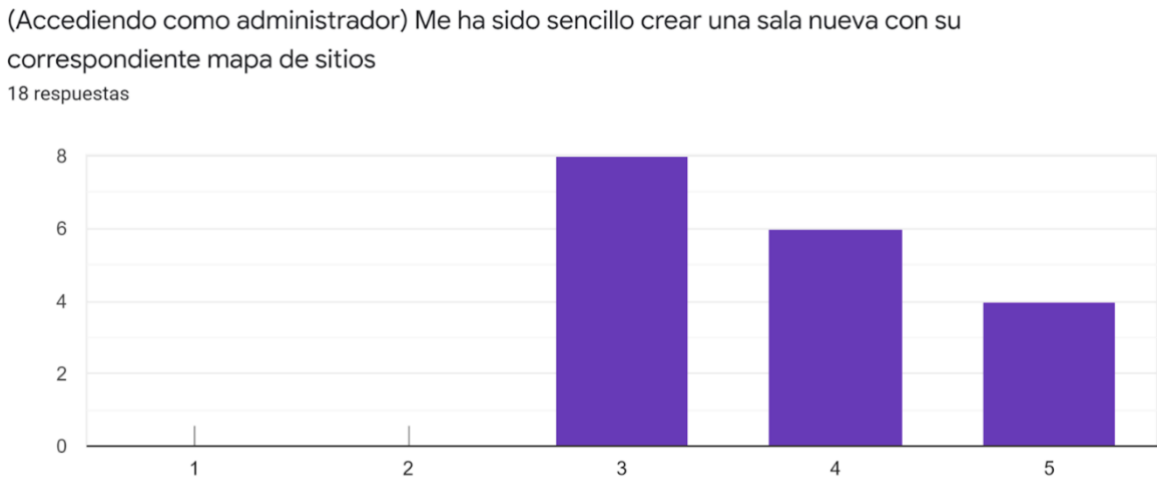


Figura 107 - Creación de mapas 1

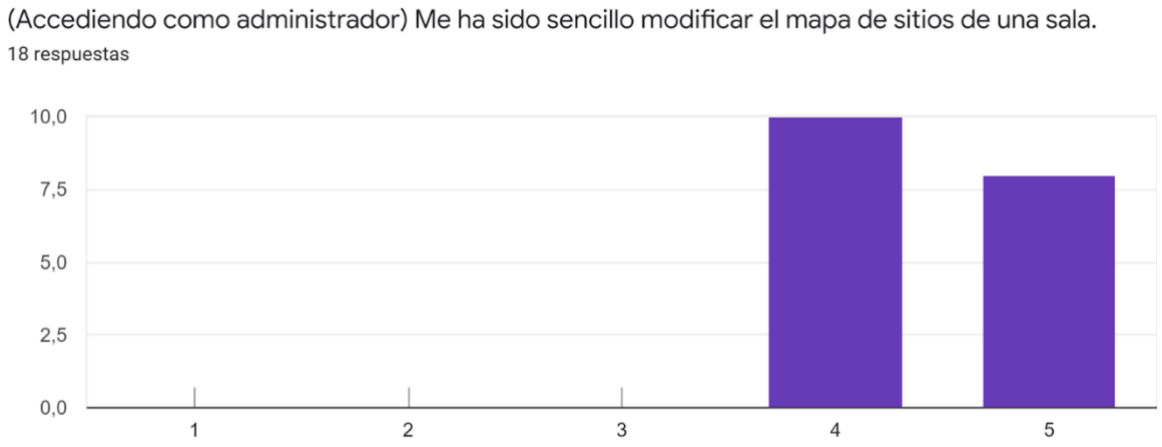


Figura 108 - Creación de mapas 2

7.2.4. Uso de botones con texto y/o icono

Sobre las funcionalidades que incluyen el uso de botones para dar acceso a la utilidad correspondiente se han obtenido valoraciones también altas en cuanto a si han sido intuitivas o no. Esto se interpreta como un buen uso de los iconos y textos para transmitir al usuario qué es lo que puede realizar con cada botón. Las Figuras 109, 110 y 111 representan los resultados comentados. En estas figuras se puede apreciar que únicamente en uno de los casos se ha valorado con un 3 el uso de estos botones de manera intuitiva, que ha sido en el caso del tablón de anuncios. Esta valoración corresponde al 5.6% del total sobre esa pregunta. Con respecto a los datos correspondientes a un 4 de valoración se sitúan entre un 77% y un 50%, siendo esta la valoración que mejor han considerado los usuarios para la intuitividad de los botones.

Figura 109 - Uso de botones con texto y/o icono 1

Figura 110 - Uso de botones con texto y/o icono 2

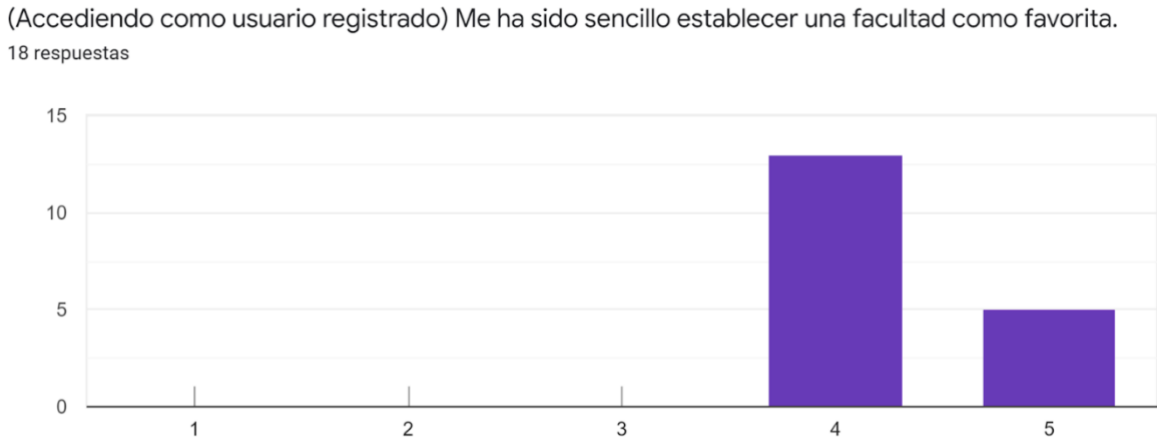


Figura 111 - Uso de botones con texto y/o icono 3

Capítulo 8. Conclusiones y trabajo futuro

En este capítulo se reflejan las conclusiones del proyecto extraídas a lo largo de la realización de este, así como los desarrollos y ampliación de las funcionalidades que se podrían realizar en un futuro.

8.1. Conclusiones

Al no tratarse de un trabajo de investigación propiamente dicho, sino más bien un proyecto que pretende desarrollar una aplicación para Smartphone con el sistema operativo Android, las conclusiones que se plantean serán un breve resumen de los resultados finalmente obtenidos y de las reflexiones a las que se ha llegado con su elaboración.

Al analizar su funcionalidad, se puede observar que el presente proyecto ha cumplido con los objetivos inicialmente propuestos. Se ha creado un servicio que permite a los usuarios disponer de información de las facultades, visualizar eventos y noticias, comprobar el estado de ocupación de las salas, y reservar con antelación asientos en las diferentes salas.

Es importante señalar que la aplicación no sólo está dirigida a los usuarios más comunes de la universidad, los estudiantes, puesto que abarca a cualquier usuario de la universidad que estando registrado necesite hacer uso de las instalaciones y, en el caso de no estar registrado, igualmente podrá acceder a toda la información de las distintas facultades.

Con este servicio el usuario también podrá realizar un registro de entrada a través de la lectura de un código QR, y dispondrá de un aviso automático a la salida de la sala para que realice el registro de salida manualmente. De esta manera se asegura la precisión del espacio reservado en salas.

Además, la aplicación también cuenta con un sistema de notificaciones que permite avisar al usuario de la existencia de una reserva con una antelación de 10 minutos. Pulsando esta notificación el usuario accede a una vista en la que se mostrará el mapa de sitios de la sala con su asiento seleccionado, en la cual podrá modificar su reserva o realizar el registro de entrada.

Por otro lado, se permite el acceso de usuarios administradores, solicitados bajo petición, que tendrán la posibilidad añadir, modificar y eliminar facultades y salas, incluyendo sus respectivos mapas de sitios, y podrán gestionar las noticias y eventos del tablón de anuncios de cada facultad.

Como un plus al proyecto se ha desarrollado otro servicio de apoyo a la gestión para los usuarios administradores, que permite la gestión de las facultades sin la necesidad de disponer de un dispositivo con sistema operativo Android. Se trata de un servicio web que consta de un acceso público con una autentificación de usuario, de manera que sólo podrán acceder al servicio los usuarios con perfil administrador. Desde este servicio podrán consultar, dar de alta y modificar facultades con la información que se muestra en la aplicación Android y, asimismo, tendrán la capacidad de eliminarlas junto con todas sus relaciones de las distintas funcionalidades.

En conclusión, se ha desarrollado una aplicación que ayuda a tener una organización en las aulas, con la certeza de que, si un usuario reserva un asiento, va a poder utilizarlo. Al mismo tiempo, las universidades podrán comprobar la ocupación y utilización de las distintas salas con el objeto de prestar un mejor servicio a los usuarios de estas instalaciones. En este mismo enfoque, en los tiempos de pandemia en los que aún se encuentra la población, la planificación y gestión de los espacios es esencial, y ResUniversitas es capaz de satisfacer estas necesidades.

Además, es importante señalar que para el desarrollo de este proyecto se han utilizado dos repositorios en GitHub, con una visibilidad privada para separar los desarrollos de la aplicación Android de la página web y mantener una seguridad en el acceso a los mismos. En estos repositorios se ha modificado la visibilidad para hacer pública la versión final de los desarrollos realizados.

A continuación, se indican los enlaces tanto de la aplicación Android como de la aplicación web, de esta manera se puede acceder al código desarrollado:

- Aplicación Android: https://github.com/JoseMiguelC/TFG
- Aplicación web: https://github.com/JoseMiguelC/TFGweb

Por otra parte, se ha generado un fichero APK [(54)] para permitir la instalación del software en un Smartphone.

En referencia a la aplicación web, se ha publicado en el siguiente enlace: https://resuniversitas-c10b4.web.app/.

Por último, para la publicación de los documentos que forman este proyecto se han subido los mismos a un repositorio en Google Drive (https://drive.google.com/drive/folders/1EbkW03SGrS8HrbV4H9C9mKywGlu0HCVH) proporcionado por Formularios de la Facultad de Informática. Dichos documentos son los siguientes:

- TFG-master.zip: Fichero que contiene el código actualizado del repositorio de la aplicación Android.

- TFGweb-master.zip: Fichero que contiene el código actualizado del repositorio de la aplicación web.
- ResUniversitas.apk: APK Instalable generado con la última versión del código del repositorio.
- Memoria ResUniversitas.pdf: El presente documento de la memoria del proyecto.
- Diagrama Gantt.xlsx: Diagrama de Gantt utilizado en la planificación del proyecto.

Para poder acceder a la aplicación Android y a la aplicación web con el perfil de usuario administrador, se ha creado un usuario de prueba con el correo electrónico resuniversitas@ucm.es y la contraseña “ResU1234”.

8.2. Trabajo a futuro

Algunas mejoras de funcionalidad que se podrían desarrollar en los servicios son la siguientes:

- **Control de pertenencia a una universidad o facultad:** Actualmente solo existe un perfil administrador que puede gestionar todas las facultades de la aplicación indistintamente de la universidad a la que pertenezcan. Para poder realizar una utilización de varias universidades independientemente se tendría que ampliar el control de acceso al nivel de la universidad para los usuarios administradores. Por tanto, solo tendrán acceso a las facultades que pertenezcan a la universidad del usuario.

- **Desarrollo en iOS:** Para poder alcanzar al mayor número de usuarios posibles y que todos los mismos pudieran realizar usos de los servicios que proporciona la aplicación Android, sería necesario realizar el desarrollo de una nueva aplicación para el sistema operativo iOS.

- **Ampliación de la gestión web:** Hasta el momento, solo se permite una gestión web simple. Para poder englobar todas las funcionalidades que dispone la aplicación Android mediante los usuarios administradores, sería necesario realizar una ampliación de estas en la página web. Así se dispondría de una mayor accesibilidad para los usuarios administradores.

- **Formulario de errores y mejoras:** De cara a una mejora constante de los servicios prestados, se desarrollaría un formulario que permita informar errores producidos en la aplicación Android y en la página web, además de, recoger las necesidades de funcionalidades que no contienen los servicios.

- **Ampliar patrones y arquitecturas:** En cuanto a la aplicación Android, con el objetivo de facilitar el posterior desarrollo y mantenibilidad de la aplicación, sería adecuado cambiar ciertos aspectos de la arquitectura. Se podrían aplicar los principios de Clean Architecture ([55]), organizando el código en casos de uso entre

otras cosas. La implementación de inyección de dependencias permitiría ampliar el uso del patrón Singleton e incorporar el patrón Factory ([56]), por ejemplo, para los ViewModels, además de optimizar el código.

- **Personalización de las notificaciones:** Con el objetivo de que la aplicación se pueda adaptar a las necesidades de cada usuario, se daría la posibilidad de personalizar ciertos aspectos de las notificaciones, como la antelación con la que se desea ser notificado de una reserva o la posibilidad de deshabilitar dichas notificaciones desde la pantalla de ajustes de la aplicación.

- **Mejoras en las reservas de sitios:** Para asegurar el correcto funcionamiento del sistema, se podrían añadir comprobaciones para impedir problemas de consistencia de datos en caso de que dos usuarios realicen una reserva a la vez en el mismo sitio para el mismo periodo de tiempo.

- **Pruebas unitarias y de integración:** Con el objetivo de facilitar las tareas de testeo de la aplicación al añadir nuevas funciones o cambiar porciones del código, sería conveniente la implementación de pruebas automatizadas tanto unitarias como de integración, para así testear funciones, repositorios y and ViewModels entre otros.

- **Gestión de acceso a universidades web:** Buscando reducir los permisos de administrador a sus correspondientes facultades y universidades a futuro se espera añadir herramientas guard correspondientes al framework de angular para evitar la carga de componentes a los que no deba tener acceso un administrador, como se ha mencionado en el primer punto de este apartado.

- **Cambio de estilos web:** Con intención de establecer una similitud entre aplicación Android y aplicación web se incorporará la librería Angular Material para incorporar nuevos estilos a la aplicación web que serán tomados en base a los estilos actuales de la aplicación Android.

- **Transformación de la web a PWA:** Con esta idea de transformación de la web a PWA (Progressive Web Application) se facilitará en un futuro el acceso a las funcionalidades de la web para los usuarios sin necesidad de que tengan que instalar la aplicación Android, en caso de que quieran un funcionamiento más fluido en sus móviles.

- **Transformación de la web a multilenguaje:** Para poder extender la aplicación a otros países y tener un mayor alcance de usuarios se pretende una vez terminada la aplicación web convertir esta en multilenguaje, así una vez acabada se podrá visualizar con los idiomas más utilizados actualmente.

- **Añadir tests a la aplicación web:** Para poder mantener constantemente el correcto funcionamiento de la app se añadirán un conjunto de tests que permitan la observación de errores de todo tipo para una respuesta más rápida ante cualquier problema al añadir un elemento nuevo a la aplicación o durante la época de mantenimiento de esta.

Chapter 8. Conclusions and future work

This chapter reflects the final conclusions of the project drawn throughout the realization of the same as well as the developments and expansion of the functionalities that could be done in the future.

8.1. Conclusions

Assuming that this degree project is not a research project, otherwise it is a development project that pursues developing a smartphone application with Android as the operating system. Therefore, the conclusions that it sets out will be a brief summary of the results achieved and the reflections that have been reached with the project elaboration.

When analyzing its functionality, this project has met the initially proposed objectives. A service has been created that allows users to have information on the faculties, view events and news, check the occupancy status of the rooms, and reserve seats in advance in the different rooms.

It is important to point out that the application is not only aimed at the most common users of the university, the students, since it covers any user of the university who, being registered, needs to use the facilities and, in the case of not being registered, you can also access all the information of the different colleges.

With this service, the user will also be able to check-in by reading a QR code, and will have an automatic check-out by using their geolocation with respect to the coordinates of the room that they are in. In this way, once the user leaves said area, the application will release the site he was using so that another user can reserve it.

In addition, the application also has a notification system that allows the user to be notified of the existence of a reservation 10 minutes in advance. By pressing this notification, the user accesses a view in which the site map of the room with their selected seat will be shown, in which they can modify their reservation or check-in.

On the other hand, access is allowed for administrator users, requested upon request, who will have the possibility to add, modify and delete faculties and rooms, including their respective site maps, and will be able to manage the news and events on the bulletin board of each faculty.

As a plus to this project, another management support service has been developed for administrator users, which allows the management of faculties without the need for a device with an Android operating system. It is a web service that consists of public access with user authentication, so that only users with an administrator profile can access the service. From this service they will be able to consult, register and modify faculties with the information that is shown in the Android application and, likewise, they will have the capacity to eliminate them together with all their relations of the different functionalities.

In conclusion, an application has been developed that helps to have an organization in the classrooms, with the certainty that if a user reserves a seat, he will be able to use it. At the same time, the colleagues will be able to check the occupation and use of the different rooms to provide a better service for the users of these facilities. In this same approach, in times of pandemic in which the population is still found, the planning and management of spaces is essential, and ResUniversitas can meet these needs.

Also, it is important to note that for the development of this project two repositories have been used on GitHub, with a private visibility to separate the developments of the android application, of the web application and maintain a security in the access to them. The visibility has been modified in these repositories so as to make public the final version of the developments carried out.

Below are the links of both the Android application and the web application, in this way you can access the code:

- Android Application: https://github.com/JoseMiguelC/TFG
- Web Application: https://github.com/JoseMiguelC/TFGweb

On the other side, an APK [(54)] file has been generated to allow the installation of the software in a Smartphone.

In reference to the Web Application, it has been published in the following link: https://resuniversitas-c10b4.web.app/.

Finally, for the publication of the documents that make up this project, they have been uploaded to a Google Drive Repository (https://drive.google.com/drive/folders/1EbkW03SGrS8HrbV4H9C9mKywGlu0HCVH) provided by Formularios de la Facultad de Informática. These documents are the following:

- TFG-master.zip: File containing the updated code of the Android application repository.
- TFGweb-master.zip: File containing the updated code of the web application repository.
- ResUniversitas.apk: Installable APK generated with the latest version of the repository code.

- Memoria ResUniversitas.pdf: This project report document.
- Diagrama Gantt.xlsx: Gantt chart used in project planning.

In order to access the Android application and the web application with the administrator user profile, a test user has been created with the email address resuniversitas@ucm.es and the password “ResU1234”.

8.2. Future work

Some functionality improvements that could be developed in the services are the following:

- **Control of membership of a university or faculty:** Currently there is only one administrator profile that can manage all the faculties of the application regardless of the university to which they belong. To be able to make use of several universities independently, access control would have to be extended to the university level for administrator users. This being the case, they will only have access to the faculties that belong to the user's university.

- **iOS development:** To be able to reach as many users as possible and that all of them could make use of the services provided by the Android application, it would be necessary to develop a new application for the iOS operating system.

- **Expansion of web management:** At this time, only simple web management is allowed... To be able to encompass all the functionalities available in the Android application through the administrator users, it would be necessary to make an extension of these on the website. In this way, greater accessibility would be available for administrator users.

- **Bug and Improvement Form:** To constantly improve the services provided, a form would be developed that allows you to report errors in the Android application and on the website, in addition to collecting the needs of functionalities that the services do not contain.

- **Extend patterns and architectures:** As for the Android application, with the aim of facilitating the further development and maintainability of the application, it would be appropriate to change certain aspects of the architecture. Clean Architecture ([55]) principles could be applied, organizing the code into use cases. The implementation of dependency injection would allow to expand the use of the Singleton pattern and incorporate the Factory ([56]) pattern for example for ViewModels, in addition to optimizing the code.

- **Notification Customization:** With the objective that the application can be adapted to the needs of each user, the possibility of customizing certain aspects of

notifications, such as the advance notice you want to be notified of a reservation or the ability to disable such notifications from the application's settings screen.

- **Site booking improvements:** To ensure proper system operation, checks could be added to prevent data consistency issues in case two users make a reservation at the same time on the same site for the same period.

- **Unit and integration testing:** With the objective of facilitating the tasks of testing the application by adding new functions or changing portions of the code, it would be convenient to implement automated tests both unitary and integration, in order to test functions, repositories and ViewModels, among others.

- **Management access to web universities:** Seeking to reduce administrator permissions to their corresponding colleges and universities in the future, it is expected to add guard tools corresponding to the angular framework to prevent loading of components that an administrator should not have access to, as mentioned in the first point of this section.

- **Change web styles:** With the intention of establishing a similarity between the android application and the web application, the Angular Material library will be incorporated to add new styles to the web application that will be taken based on the current styles of the android application.

- **Transforming the web to PWA:** With this idea of transforming the web to PWA (Progressive Web Application) will facilitate in the future access to the functionalities of the web for users without having to install the android application in case they want a smoother operation in their mobiles.

- **Transformation of the web to multilanguage:** To be able to extend the application to other countries and have a greater reach of users it is intended once the web application is finished to convert it into multilanguage, so once finished it can be viewed with the most used languages currently.

- **Add tests to application:** To be able to constantly maintain the correct functioning of the app, a set of tests will be added that allow the observation of errors of all kinds for a faster response to any problem when adding a new element to the application or during the maintenance period of the application.

Capítulo 9. Aportaciones Individuales

9.1. José Miguel Cabrera Llamas

- Aplicación Android:
  Ha realizado tareas de desarrollo para el perfil de usuario registrado, por ejemplo, el cambio de contraseña y la baja del usuario. También, se ha encargado de implementar las gráficas de ocupación de las salas. Además, ha participado en el desarrollo de las altas y modificaciones de las notificaciones y eventos.

- Aplicación web:
  Ha trabajado en la creación y desarrollo de la aplicación web. También se ha encargado de realizar la coordinación de las tareas realizadas en la misma.

- Documentación:
  Ha coordinado el trabajo realizado en la documentación y ha trabajado en la redacción de varios apartados de la memoria, así como en la especificación de requisitos (SRS).

9.2. Víctor Choza Merino

- Aplicación Android:
  Ha coordinado las tareas realizadas en la aplicación Android. Se ha encargado del desarrollo de la arquitectura de la aplicación, de la migración a nuevas tecnologías y de la creación de la gran mayoría de funciones tanto de usuarios administradores como de usuarios registrados e invitados. También ha implementado el diseño general utilizado a lo largo de la aplicación. Finalmente ha dado soporte al resto de integrantes y ha pulido las implementaciones realizadas por otros miembros del equipo.

- Documentación:
  Ha trabajado en la especificación de requisitos y se ha encargado de redactar en esta memoria los apartados referidos a la aplicación Android, como pueden ser el diseño, la arquitectura, las funcionalidades y la guía de usuario. También ha ofrecido soporte al resto de integrantes y ha realizado tareas de revisión de la memoria, sobre todo en cuanto a temas de orden y redacción.

9.3. Álvaro Penalva Alberca

- Aplicación Android:
  Ha realizado tareas de los distintos perfiles de usuario como la implementación de funciones referentes a la geolocalización y ha dado soporte a los distintos integrantes del equipo.

- Documentación:
  Ha trabajado en el SRS y en la redacción de varios apartados de la memoria como los casos de uso y la evaluación de usabilidad entre otros.

9.4. Pedro Sánchez Escribano

- Aplicación Android:
  Ha realizado tareas correspondientes a los distintos perfiles de usuario como la creación de los códigos QR y el documento PDF y ha dado soporte en otras tareas.

- Aplicación web:
  Se ha encargado de aplicar estilo y también, ha trabajado en el alta y en la modificación de facultades implementando el horario y los días festivos.

- Documentación:
  Ha trabajado en el SRS y en la redacción de varios apartados de la memoria como la guía de usuario y el modelo de la base de datos entre otros.

Bibliografía

En este apartado se va a especificar todas las herramientas tecnológicas utilizadas para la imple

[1]. Kalena. https://kalena.es/. Recuperado el 20 de mayo de 2022.

[2]. Affluences. https://affluences.com/?lang=e. Recuperado el 20 de mayo de 2022.

[3]. Tenea-Talent. https://www.teneatalent.com/gestor-reserva-espacios-recursos. Recuperado el 20 de mayo de 2022.

[4]. Hybo. https://hybo.raona.com/. Recuperado el 20 de mayo de 2022.

[5]. Android Studio. https://es.wikipedia.org/wiki/Android_Studio. Recuperado el 20 de mayo de 2022.

[6]. JetBrains. https://es.wikipedia.org/wiki/JetBrains. Recuperado el 20 de mayo de 2022.

[7]. IntelliJ IDEA. https://es.wikipedia.org/wiki/IntelliJ_IDEA. Recuperado el 20 de mayo de 2022.

[8]. Visual Studio Code. https://es.wikipedia.org/wiki/Visual_Studio_Code. Recuperado el 20 de mayo de 2022.

[9]. Electron. https://es.wikipedia.org/wiki/Electron_(software). Recuperado el 20 de mayo de 2022.

[10]. Kotlin. https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n). Recuperado el 20 de mayo de 2022.

[11]. JavaScript. https://es.wikipedia.org/wiki/JavaScript. Recuperado el 20 de mayo de 2022.

[12]. TypeScript. https://es.wikipedia.org/wiki/TypeScript. Recuperado el 20 de mayo de 2022.

[13]. HTML5. https://es.wikipedia.org/wiki/HTML5. Recuperado el 20 de mayo de 2022.

[14]. World Wide Web. https://es.wikipedia.org/wiki/World_Wide_Web. Recuperado el 20 de mayo de 2022.

[15]. XHTML. https://es.wikipedia.org/wiki/XHTML. Recuperado el 20 de mayo de 2022.

[16]. CSS. https://es.wikipedia.org/wiki/CSS. Recuperado el 20 de mayo de 2022.

[17]. W3C. https://es.wikipedia.org/wiki/World_Wide_Web_Consortium. Recuperado el 20 de mayo de 2022.

[18]. Angular. https://es.wikipedia.org/wiki/Angular_(framework). Recuperado el 20 de mayo de 2022.

[19]. Firebase. https://es.wikipedia.org/wiki/Firebase. Recuperado el 20 de mayo de 2022.

[20]. Firebase Authentication. https://firebase.google.com/docs/auth. Recuperado el 20 de mayo de 2022.

[21]. Cloud Firestore. https://firebase.google.com/docs/firestore. Recuperado el 20 de mayo de 2022.

[22]. Cloud Storage. https://firebase.google.com/docs/storage. Recuperado el 20 de mayo de 2022.

[23]. Firebase Hosting. https://firebase.google.com/docs/hosting. Recuperado el 20 de mayo de 2022.

[24]. Git. https://es.wikipedia.org/wiki/Git. Recuperado el 20 de mayo de 2022.

[25]. GitHub. https://es.wikipedia.org/wiki/GitHub. Recuperado el 20 de mayo de 2022.

[26]. Código QR. https://es.wikipedia.org/wiki/C%C3%B3digo_QR. Recuperado el 20 de mayo de 2022.

[27]. GPS. https://es.wikipedia.org/wiki/GPS. Recuperado el 20 de mayo de 2022.

[28]. Trilateración inversa. https://acolita.com/como-funcionan-los-dispositivos-gps-trilateracion-vs-triangulacion/. Recuperado el 20 de mayo de 2022.

[29]. Google Forms. https://es.wikipedia.org/wiki/Formularios_de_Google. Recuperado el 20 de mayo de 2022.

[30]. Cliente(s)-Servidor. https://es.wikipedia.org/wiki/Cliente-servidor. Recuperado el 20 de mayo de 2022.

[31]. API. https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones. Recuperado el 20 de mayo de 2022.

[32]. URL. https://en.wikipedia.org/wiki/URL. Recuperado el 20 de mayo de 2022.

[33]. MVVM. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. Recuperado el 20 de mayo de 2022.

[34]. Singleton. https://es.wikipedia.org/wiki/Singleton. Recuperado el 20 de mayo de 2022.

[35]. DataClasses. https://kotlinlang.org/docs/data-classes.html. Recuperado el 20 de mayo de 2022.

[36]. StateFlows. https://developer.android.com/kotlin/flow/stateflow-and-sharedflow. Recuperado el 20 de mayo de 2022.

[37]. SharedFlows. https://developer.android.com/kotlin/flow/stateflow-and-sharedflow. Recuperado el 20 de mayo de 2022.

[38]. Patrón Observador. https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o). Recuperado el 20 de mayo de 2022.

[39]. DataBinding. https://developer.android.com/topic/libraries/data-binding. Recuperado el 20 de mayo de 2022.

[40]. Corrutinas. https://developer.android.com/kotlin/coroutines. Recuperado el 20 de mayo de 2022.

[41]. SavedState. https://developer.android.com/topic/libraries/architecture/viewmodel-savedstate. Recuperado el 20 de mayo de 2022.

[42]. Façade. https://refactoring.guru/es/design-patterns/facade. Recuperado el 20 de mayo de 2022.

[43]. Proxy. https://refactoring.guru/es/design-patterns/proxy. Recuperado el 20 de mayo de 2022.

[44]. AngularBinding. https://angular.io/guide/binding-syntax. Recuperado el 20 de mayo de 2022.

[45]. Material Design 3. https://m3.material.io/. Recuperado el 20 de mayo de 2022.

[46]. Material Theme Builder. https://material-foundation.github.io/material-theme-builder. Recuperado el 20 de mayo de 2022.

[47]. Dynamic Colors. https://m3.material.io/styles/color/dynamic-color/overview. Recuperado el 20 de mayo de 2022.

[48]. Splash Screen. https://developer.android.com/guide/topics/ui/splash-screen. Recuperado el 20 de mayo de 2022.

[49]. Bootstrap. https://es.wikipedia.org/wiki/Bootstrap_(framework). Recuperado el 20 de mayo de 2022.

[50]. Diseño Responsive. https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable. Recuperado el 20 de mayo de 2022.

[51]. CDN. https://es.wikipedia.org/wiki/Red_de_distribuci%C3%B3n_de_contenidos. Recuperado el 20 de mayo de 2022.

[52]. Typescript. https://es.wikipedia.org/wiki/TypeScript. Recuperado el 20 de mayo de 2022.

[53]. Google Fonts. https://es.wikipedia.org/wiki/Google_Fonts. Recuperado el 20 de mayo de 2022.

[54]. APK. https://es.wikipedia.org/wiki/APK_(formato). Recuperado el 20 de mayo de 2022.

[55]. Clean Architecture. https://devexperto.com/clean-architecture-android/. Recuperado el 20 de mayo de 2022.

[56]. Patrón Factoría. https://es.wikipedia.org/wiki/Factory_Method_(patr%C3%B3n_de_dise%C3%B1o). Recuperado el 20 de mayo de 2022.

ANEXO I. Guía de Usuario

En este apartado se va a explicar el funcionamiento de la aplicación Android y de la página web con la intención de facilitar el uso y aprendizaje de estas.

I.I. Aplicación Android

I.I.I. Vista de inicio de sesión o registro

Al abrir la aplicación, se muestra la vista de inicio de sesión (Primera captura de la Figura 40) donde se puede iniciar sesión como usuario registrado o invitado, y también permite navegar a una pantalla de registro (Segunda captura de la Figura 112) para crear una nueva cuenta de usuario mediante un formulario.

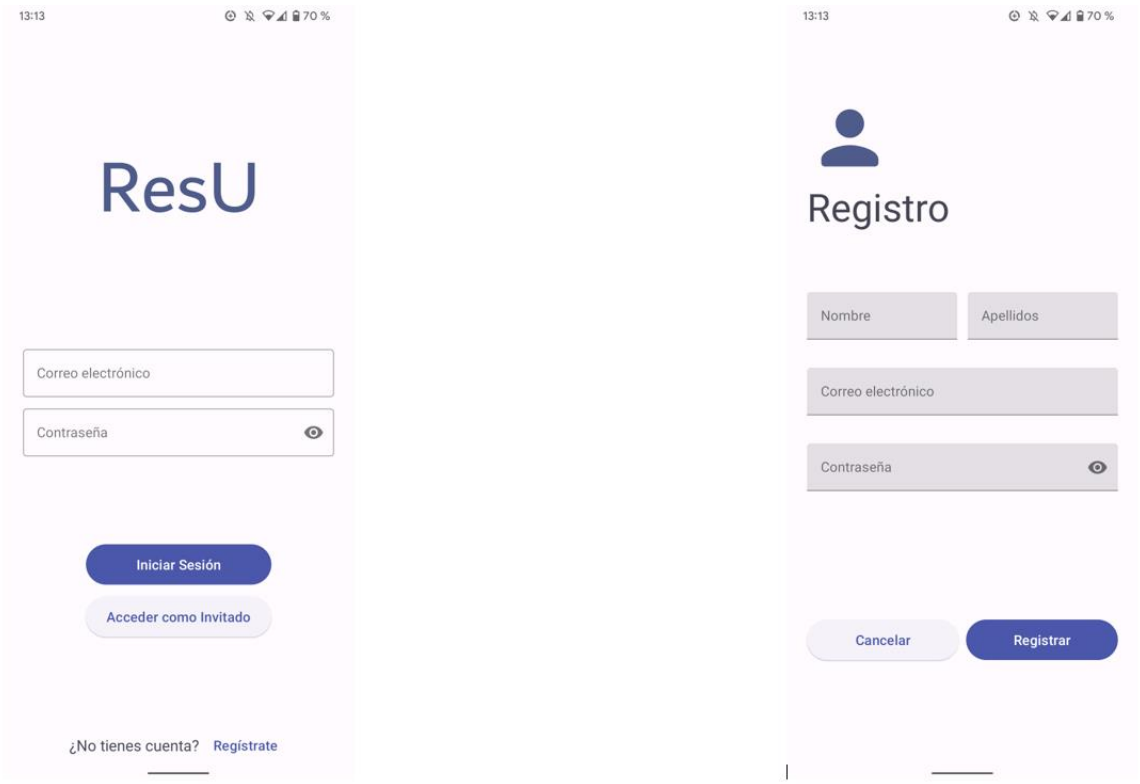


Figura 112 - Guía de usuario, Aplicación Android: Vista de inicio de sesión o registro

I.I.II. Vista de facultades

Una vez se ha iniciado sesión, ya sea como usuario registrado o como usuario invitado, se muestra la vista de facultades, tal como se visualiza en la primera captura de la Figura 113, en la que se puede ver una lista de las distintas facultades pertenecientes a la universidad seleccionada, pudiendo cambiar esta desde el menú desplegable superior.

Al hacer un clic prolongado sobre una facultad se mostrará una ventana emergente en la que se puede elegir la opción de navegar a la vista de edición de la facultad, eliminar dicha facultad o descargar un PDF con el código QR de la facultad. Esta ventana aparecerá únicamente para los usuarios administradores.

Si se quiere navegar a la vista de información de una facultad en concreto, se debe hacer clic sobre el icono de información de la facultad. Mientras que para navegar a la vista de salas de la facultad se debe hacer clic sobre la facultad.

Por último, en caso de haber iniciado sesión con una cuenta de administrador aparecerá un botón en la esquina inferior derecha con el cual se navegará a la pantalla de creación de facultad. Para ello, se debe hacer clic en el botón añadir facultad situado en la parte inferior de la vista. Esta opción se habilita solo para usuarios administradores.

I.I.III. Vista de salas facultad

Al pulsar en una facultad, como se ha comentado en el apartado anterior, se navegará a la pantalla con la lista de salas de la facultad, como se puede ver en la segunda captura de la Figura 113. En cada sala de la lista aparecerá su nombre, su capacidad y ocupación actual, y un botón de “Reservar”. En caso de que el usuario tenga una reserva próxima o de que la sala esté bloqueada, se informará de estos hechos en la tarjeta de la sala correspondiente. Si se quiere navegar a la vista de una sala en concreto se debe hacer clic sobre la sala. Esta opción se habilita para cualquier tipo de usuario.

Para cualquier usuario que no sea invitado se da la opción de navegar a la vista de reservar, previa elección de fecha y horas de la reserva, pulsando en el botón “Reservar” de la sala. Por otra parte, se puede acceder al tablón de anuncios haciendo clic sobre el botón con el icono del documento en la barra de estado de la parte superior de la vista.

Por otra parte, se puede acceder al tablón de anuncios de la facultad haciendo clic sobre el botón situado a la izquierda del icono de información en la parte superior. Al pulsar en dicho botón de información se navegará a la pantalla de información de la facultad.

En caso de haber iniciado sesión con una cuenta de administrador aparecerá un botón en la esquina inferior derecha con el cual se navegará a la pantalla de creación de sala. En caso contrario se mostrará un botón de “Escanear QR” con el que se podrá escanear los códigos QR correspondientes a facultades, salas, eventos, etc.

Finalmente, al hacer un clic prolongado sobre una sala se mostrará una ventana emergente en la que se puede elegir la opción de navegar a la vista de edición de la sala, eliminar dicha sala, descargar un PDF con los códigos QR de la sala y sus sitios, o bloquear y desbloquear la sala. Esta ventana aparecerá únicamente para los usuarios administradores.

I.I.IV. Vista del tablón de anuncios

Una vez en la vista del tablón de anuncios, representada en la tercera captura de la Figura 113, se puede ver una lista de los distintos eventos y las noticias pertenecientes a la facultad. Si se quiere navegar a la vista de información de un evento o una noticia en concreto se debe hacer clic sobre dicho evento o noticia.

Al hacer un clic prolongado sobre un evento o noticia se muestra una ventana emergente en la que se puede elegir la opción de navegar a la vista de edición de evento o noticia, eliminar dicho evento o noticia, o descargar un documento PDF con el código QR de dicho evento o noticia. Esta ventana aparecerá únicamente para los usuarios administradores.

En caso de tratarse de un evento en sala, aparecerá un botón de “Reservar” para navegar a la pantalla de reserva de sitio. Esta opción se habilita para todos los usuarios que no sean de tipo invitado.

Si la noticia o evento tuviera una dirección URL aparecerá un botón con el texto “Abrir enlace” con el que se abrirá el navegador del dispositivo cargando la página web correspondiente a la dirección URL.

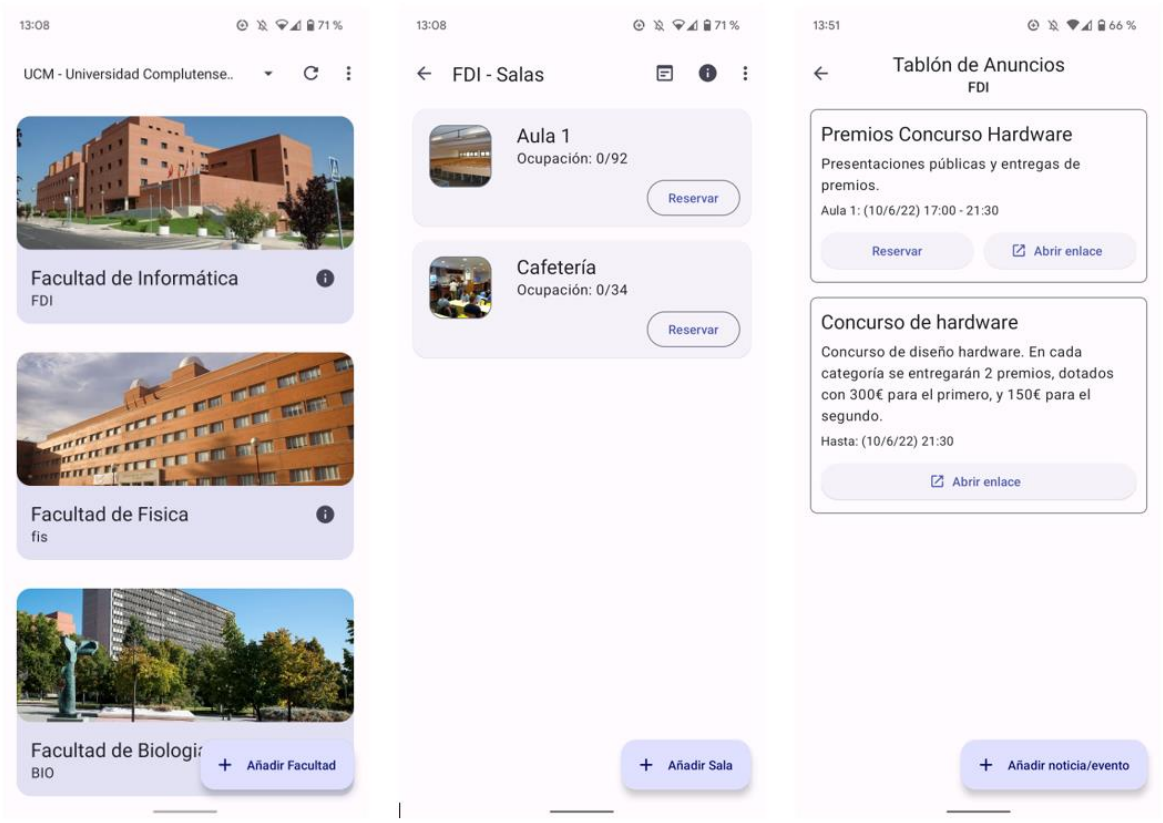


Figura 113 - Guía de usuario, Aplicación Android: Listado de facultades, listado de salas y tablón de anuncios

I.I.V. Vista de información de facultad

En la vista de información de facultad se muestra la información referente a la facultad, como se puede observar en la primera y segunda captura de la Figura 114.

Se da la posibilidad de establecer la facultad como favorita, para ello, se debe hacer clic sobre el botón con el icono de la estrella. Esta opción se habilita para usuarios que no sean de tipo invitado.

Por otra parte, la vista proporciona información sobre la ubicación, teléfono y correo electrónico de la facultad mediante tres botones que tienen la siguiente funcionalidad:

- Al hacer clic sobre el botón navegar se abrirá la aplicación de mapas del dispositivo para mostrar la ubicación de la facultad.
- Al hacer clic sobre el botón del número de teléfono se abrirá la aplicación de teléfono del dispositivo para llamar al número de la facultad.
- Al hacer clic sobre el botón del correo electrónico se abrirá la aplicación de correo del dispositivo para enviar un correo al correo de la facultad.

I.I.VI. Vista de información de sala

Pasando a la vista de sala se puede ver toda la información referente a esa misma sala, lo cual se puede visualizar en la tercera captura de la Figura 114.

Se proporciona información sobre el teléfono y el correo electrónico de la sala mediante dos botones que tienen la siguiente funcionalidad:

- Al hacer clic sobre el botón del número de teléfono se abrirá la aplicación de teléfono del dispositivo para llamar al número de la sala.
- Al hacer clic sobre el botón del correo electrónico se abrirá la aplicación de correo del dispositivo para enviar un correo al correo de la sala.

Por otra parte, se puede navegar a la vista de reserva de sitio haciendo clic sobre el botón reservar. Esta opción se habilita para usuarios que no sean de tipo invitado.

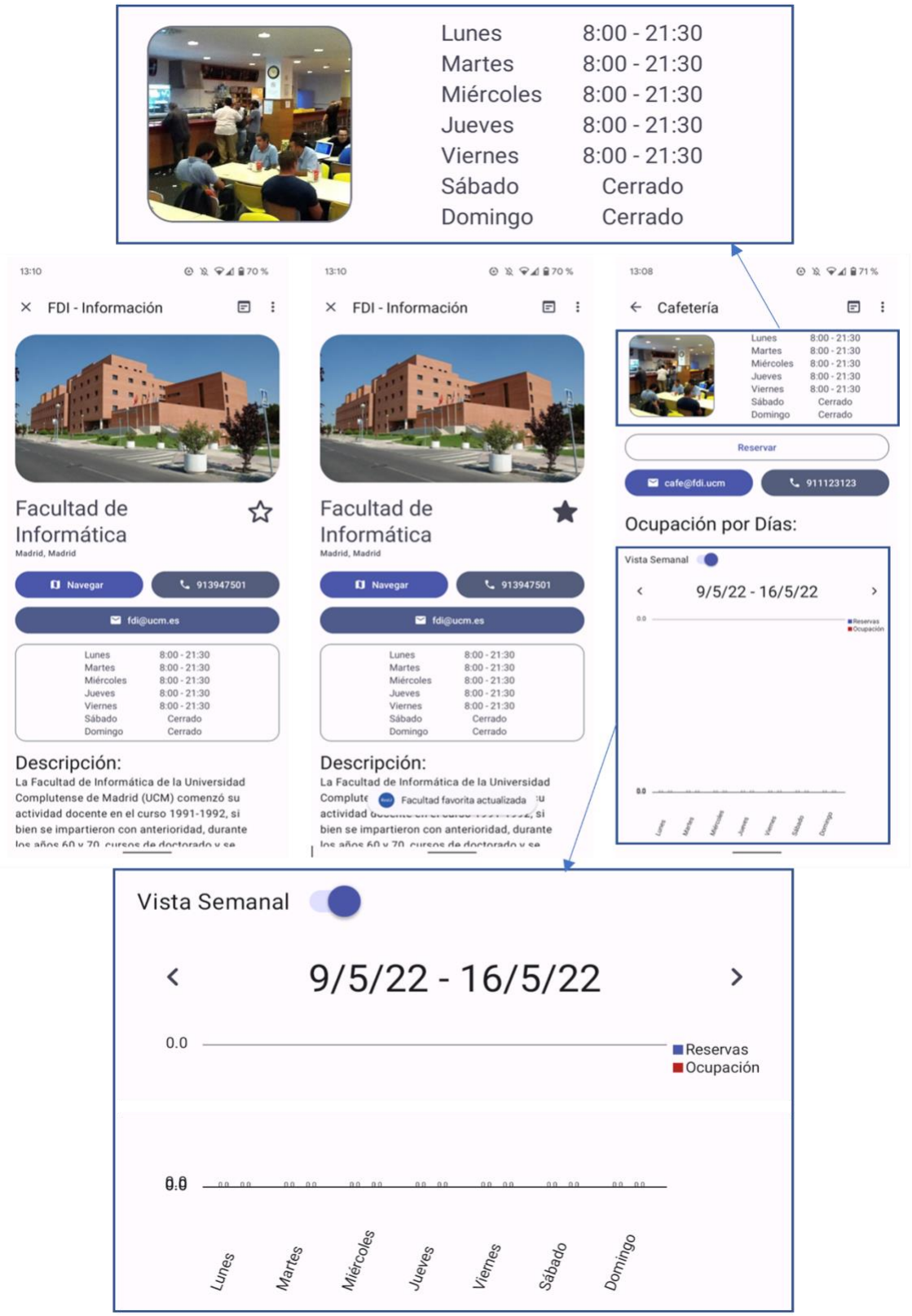


Figura 114 - Guía de usuario, Aplicación Android: Información de facultad y sala con acción de añadir a favorito

I.I.VII. Vista de creación/edición de facultad

En la vista de creación/edición de facultad se muestra un formulario con los campos a cumplimentar, tal como se puede ver en la primera captura de la Figura 115. Para que se complete correctamente la creación o modificación de facultad se deberá elegir una imagen pulsando en el botón “Añadir imagen”.

Más abajo se encuentra un listado con los días de la semana y su horario seleccionado. Para definir o eliminar el horario de un día se debe pulsar el botón “Editar” del día correspondiente. Se abrirá una ventana emergente en el que, con la ayuda de un selector de hora, se podrá elegir una hora de apertura y una de cierre, además de poder eliminar ambas para informar de que la facultad permanecerá cerrada ese día de la semana (Segunda y tercera capturas de la Figura 115).

La tercera captura de la Figura 116 muestra el último apartado con un listado de los días festivos, o en caso de no haberlos se verá únicamente un botón de “Añadir festivo/s”. Con este botón se podrán añadir, a través de una ventana emergente y unos selectores de fecha, los días en los que la facultad permanecerá cerrada (Primera y segunda capturas de la Figura 116). Si ya hay días festivos añadidos se podrán eliminar individualmente con el botón de la papelera, o eligiendo varios con el botón “Eliminar festivo/s” y los selectores de fechas correspondientes (Última captura de la Figura 116).

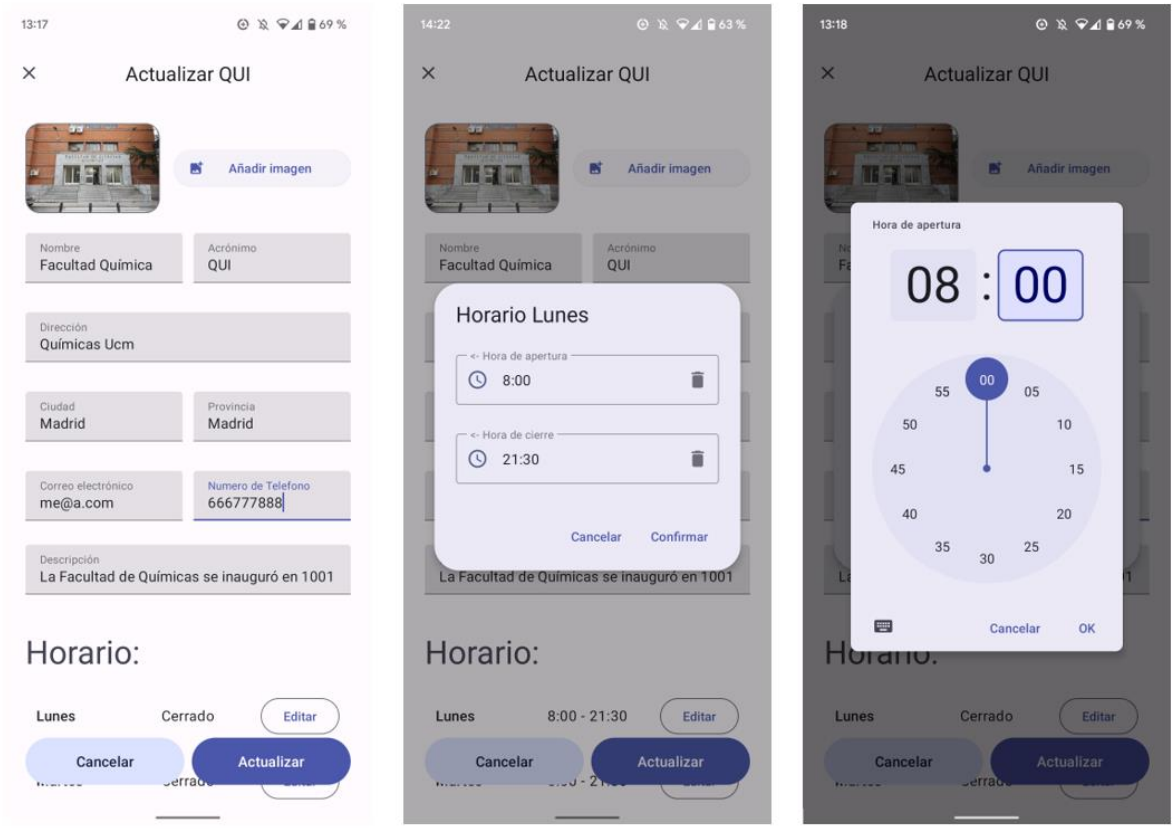


Figura 115 - Guía de usuario, Aplicación Android: Alta y modificación de facultad

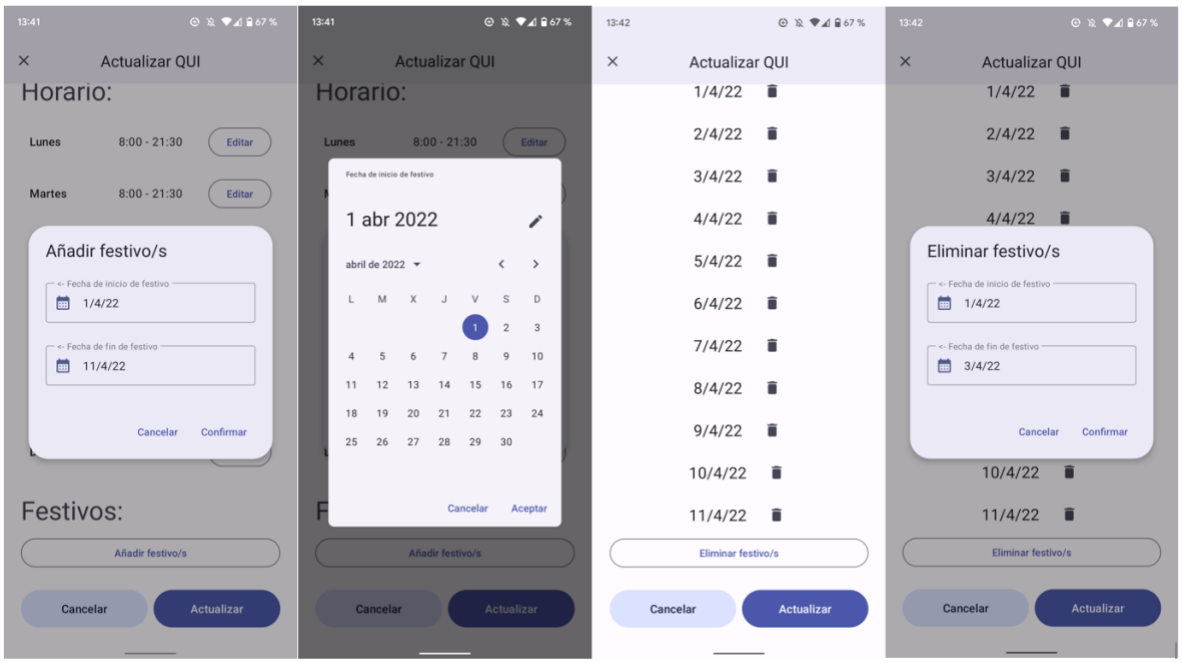


Figura 116 - Guía de usuario, Aplicación Android: Alta y modificación de facultad (días festivos)

I.I.VIII. Vistas de creación/edición de sala

La creación de sala se define como un proceso en el que intervienen varias vistas.

1. En primer lugar, se definen las características del mapa de sitios de sala mediante el formulario que se puede ver en la primera captura de la Figura 117.
2. Posteriormente, se edita el mapa de sitios mediante una vista interactiva, la cual se visualiza en la segunda y tercera captura de la Figura 117. Si se desea cambiar el tipo de sitio únicamente se deben seguir dos pasos:

    - Se debe hacer clic sobre uno de los tipos que aparecen en la parte inferior de la vista.
    - Se debe hacer clic en el mapa de sitios sobre el sitio en el que se desee cambiar el tipo al anteriormente seleccionado.

3. Finalmente, se define la información de sala mediante el formulario visible en la primera captura de la Figura 118. Este, al igual que la creación/edición de facultades, tiene un apartado para definir los horarios de la sala, teniendo estos que estar dentro de los límites de los horarios de la facultad (Segunda y tercera capturas de la Figura 118). En esta pantalla también se deberá elegir una imagen pulsando en “Añadir imagen”.

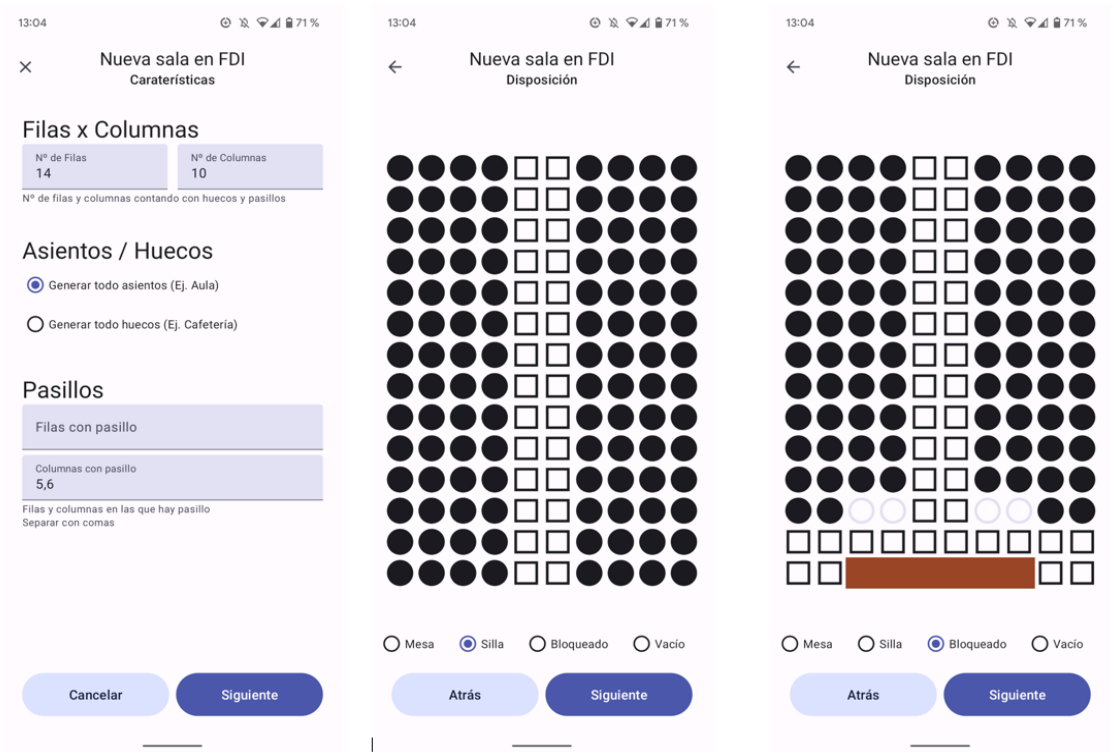


Figura 117 - Guía de usuario, Aplicación Android: Alta y modificación de sala

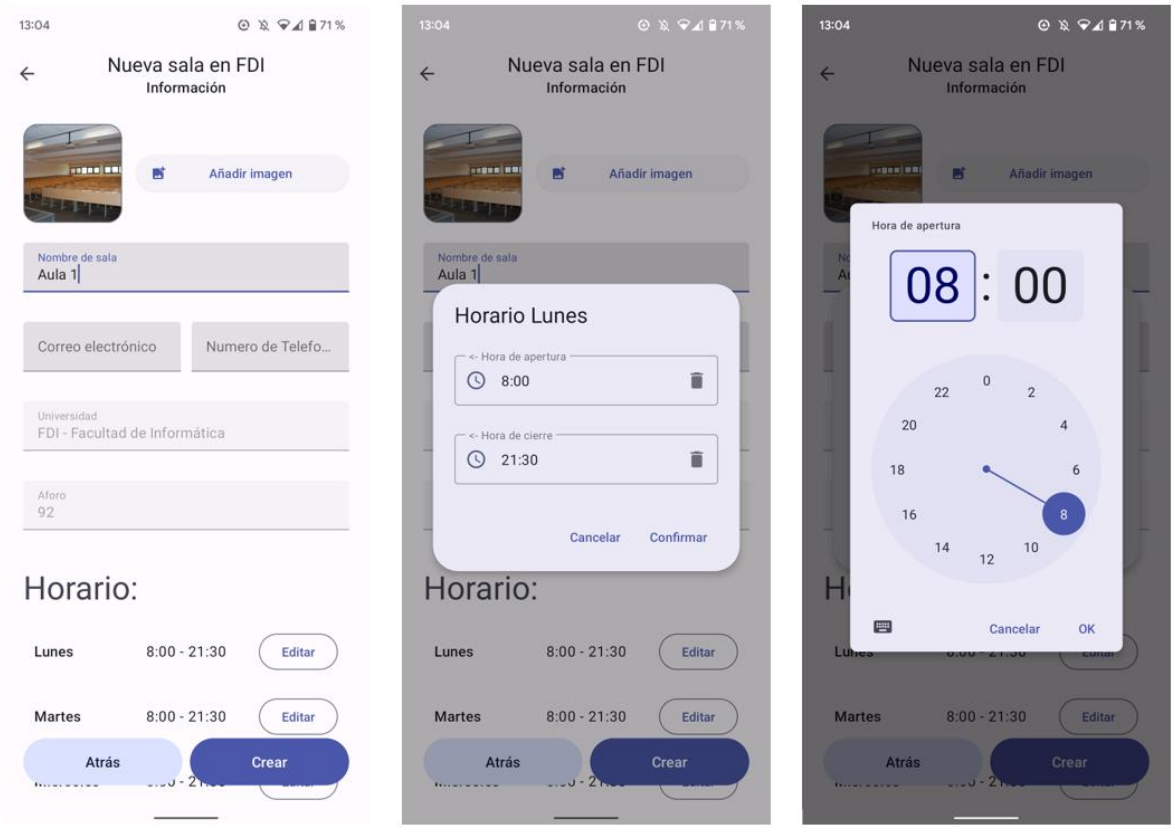


Figura 118 - Guía de usuario, Aplicación Android: Alta y modificación de sala (horario)

I.I.IX. Vista de creación/edición de evento o noticia

Las capturas de la Figura 119 muestran la vista de creación/edición de evento o noticia, la cual contiene un formulario con los campos a cumplimentar.

En esta pantalla se podrá definir si se trata de un evento en sala seleccionando “Es un evento en sala”. En este caso se tendrá que elegir una sala del menú desplegable que se muestra en la tercera y cuarta captura de la Figura 119. Para que se cree correctamente este evento se deberá introducir una fecha y horas de inicio y de fin de evento con la ayuda de los selectores de fecha y hora.

En caso de que se quiera crear una noticia no se deberá seleccionar “Es un evento en sala”. Para estos casos las únicas fechas y horas obligatorias son las de fin de noticia, la cual se tendrá en cuenta para dejar de mostrar la noticia en el tablón de anuncios. También se podrá decidir si se quieren mostrar las fechas y horas de inicio o de fin que se hayan seleccionado.

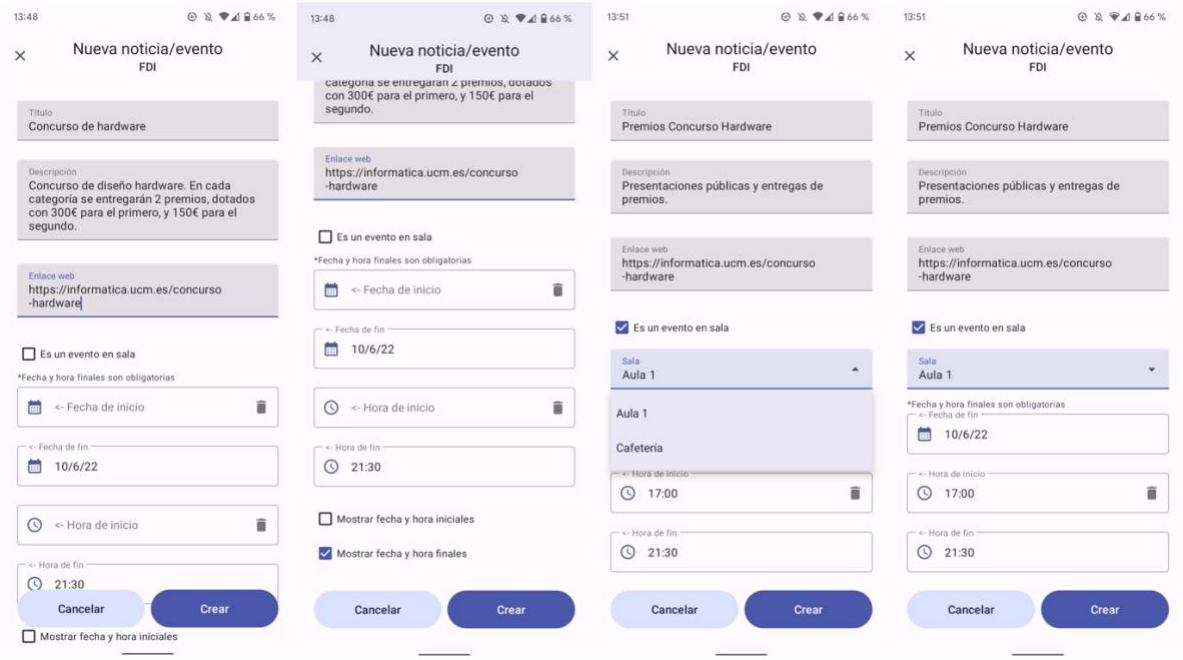


Figura 119 - Guía de usuario, Aplicación Android: Alta y modificación de noticia o evento

I.I.X. Proceso de bloqueo de sala

Como ya se mencionó anteriormente, una vez en la vista de facultad se da la posibilidad de activar o bloquear una sala. Esta opción se habilita para los usuarios administradores. Para acceder a esta función se pulsa prolongadamente sobre la sala correspondiente y se hace clic en “Activar/Bloquear” (Primera captura de la Figura 120).

Seguidamente aparecerá una ventana emergente en la que se podrá cumplimentar opcionalmente la fecha y hora del fin del bloqueo y el mensaje con el motivo del bloqueo (Segunda captura de la Figura 120).

En la tercera y cuarta captura se puede observar la forma en la que se muestra el bloqueo de una sala, en este caso con fecha y hora de fin y con mensaje.

Figura 120 - Guía de usuario, Aplicación Android: Bloqueo o desbloqueo de sala

I.I.XI. Vistas de reserva de sitio

Para reservar un sitio en una sala se debe pulsar en el botón de “Reservar” de la sala correspondiente. Se mostrará una ventana emergente en la que, con ayuda de selectores de fecha y hora, se podrán seleccionar la fecha, la hora de comienzo y la hora de fin de la reserva. Esta acción se puede visualizar en la primera, segunda y tercera captura de la Figura 121.

Si se han introducido correctamente la fecha y las horas de la reserva se navegará a la pantalla de selección de sitio de la reserva, representada en la cuarta captura de la Figura 121. En esta pantalla se mostrarán los sitios disponibles para reservar con forma circular y un número visible, los sitios ocupados se verán con un color más tenue, y los sitios bloqueados aparecerán como un círculo coloreado por los bordes y sin número visible. El asiento seleccionado será de un color destacado y con forma hexagonal.

Para completar la reserva se pulsará en “Reservar asiento”, que tras realizar la reserva con la base de datos volverá a la pantalla anterior.

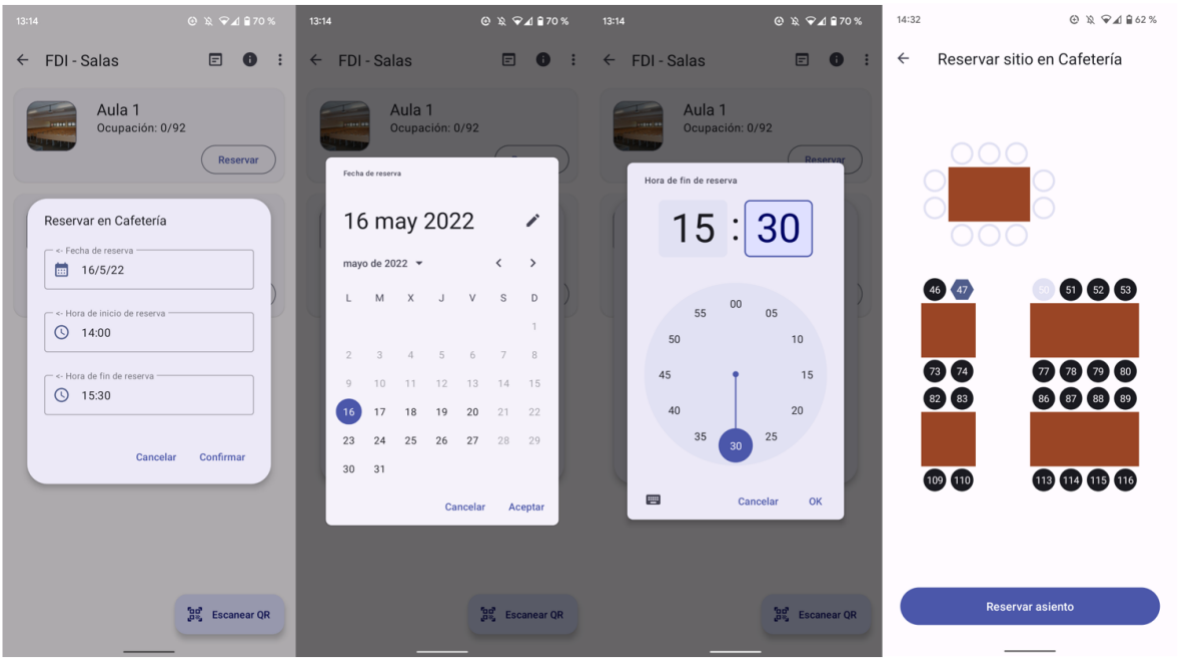


Figura 121 - Guía de usuario, Aplicación Android: Reserva de sitio en sala

I.I.XII. Proceso de check-in y check-out

Si el usuario dispone de una reserva cercana, quedando un periodo inferior a 10 minutos para el comienzo de esta, se le mostrará una notificación de aviso de reserva cercana (Primera captura de la Figura 122), y podrá ver en la sala correspondiente un botón de “Ver Reserva” (Segunda captura de la Figura 122). Tanto el botón como la notificación le llevarán

a la pantalla del sitio de la reserva, en la cual podrá hacer el check-in o cambiar el sitio de la reserva. Esta última pantalla se corresponde con la tercera captura de la Figura 122.

Al pulsar en el botón “Check-in QR Asiento XXX” se abrirá el lector de códigos QR (Primera captura de la Figura 123), el cuál una vez escanee el código QR correcto realizará el proceso de check-in y volverá a la pantalla anterior. Se mostrará una notificación para confirmar el check-in (Segunda captura de la Figura 123), y en la facultad correspondiente aparecerá un aviso del check-in actual (Tercera captura de la Figura 123).

En este mismo aviso se encuentra un botón de “check-out”, el cual al pulsarlo realizará el check-out de la reserva y eliminará el aviso de la pantalla de las salas.

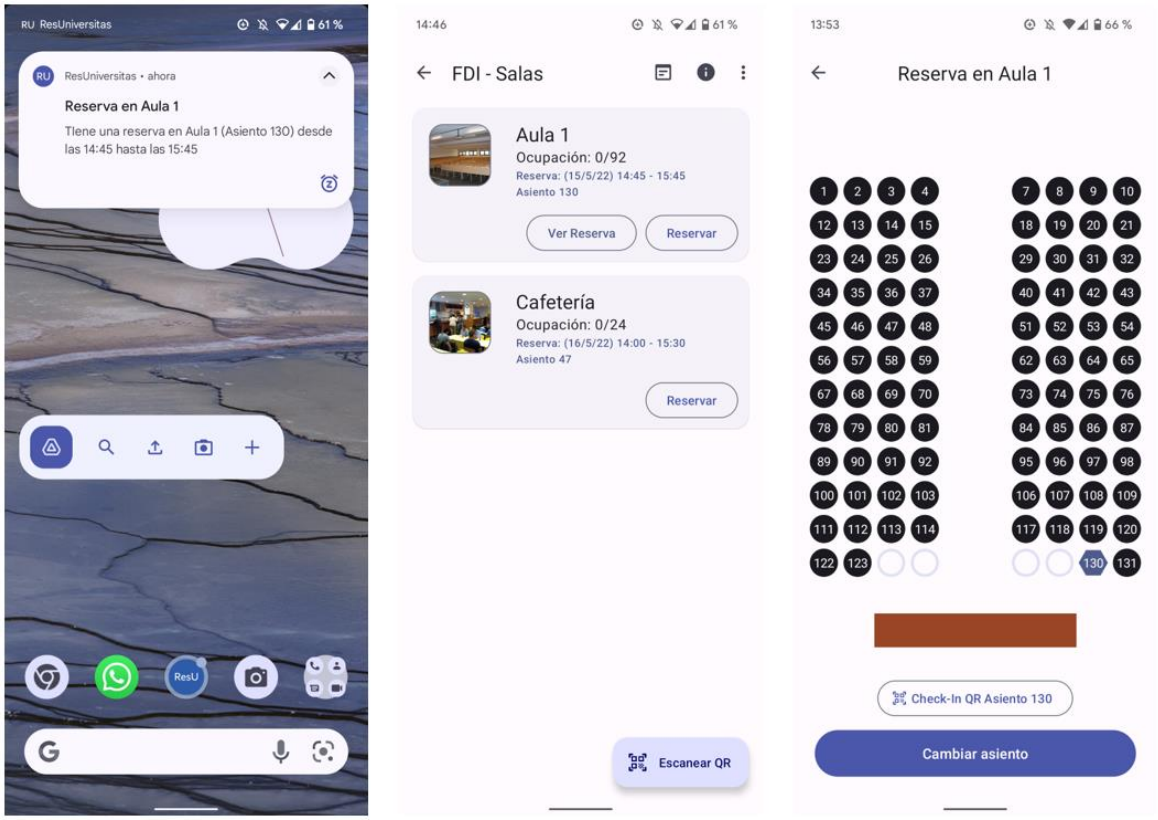


Figura 122 - Guía de usuario, Aplicación Android: Notificación de reserva, vista de sala con reserva cercana y check-in

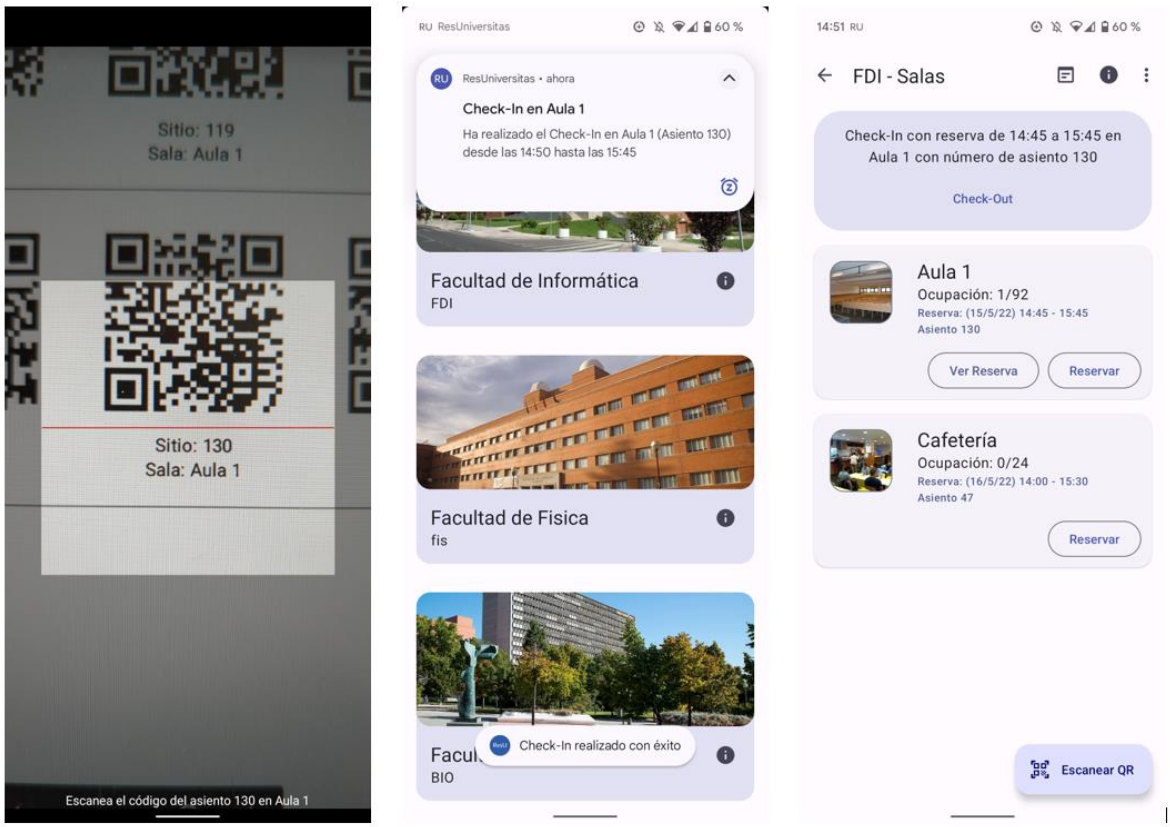


Figura 123 - Guía de usuario, Aplicación Android: Check-in con notificación y vista de salas de facultad con check-in en curso

I.I.XIII. Vista de Ajustes

Para los usuarios que no sean de tipo invitado existe una pantalla de ajustes como la que se muestra en la primera captura de la Figura 124, a la cual se accede desde el menú desplegable en la esquina superior derecha en ciertas pantallas.

Al hacer clic sobre el botón darse de baja se mostrará una ventana emergente donde se debe introducir la contraseña de usuario para verificar la baja (Segunda captura de la Figura 124).

Al hacer clic sobre el botón cambiar contraseña se mostrará una ventana emergente donde se debe introducir la contraseña actual y la nueva para verificar el cambio (Tercera captura de la Figura 124).

Por otra parte, al hacer clic sobre el botón reservas se navegará a la vista de reservas que se muestra en la cuarta captura de la Figura 124, en la que se podrán ver todas las reservas existentes y futuras del usuario, junto a información de cada reserva y un botón de una papelera con el cuál eliminar la reserva correspondiente. Al pulsar sobre una reserva se navegará a la pantalla del mapa de sitios con el sitio seleccionado, en la cual se podrá

cambiar el sitio, y en caso de que la reserva esté próxima en el tiempo se habilitará la opción de hacer check-in.

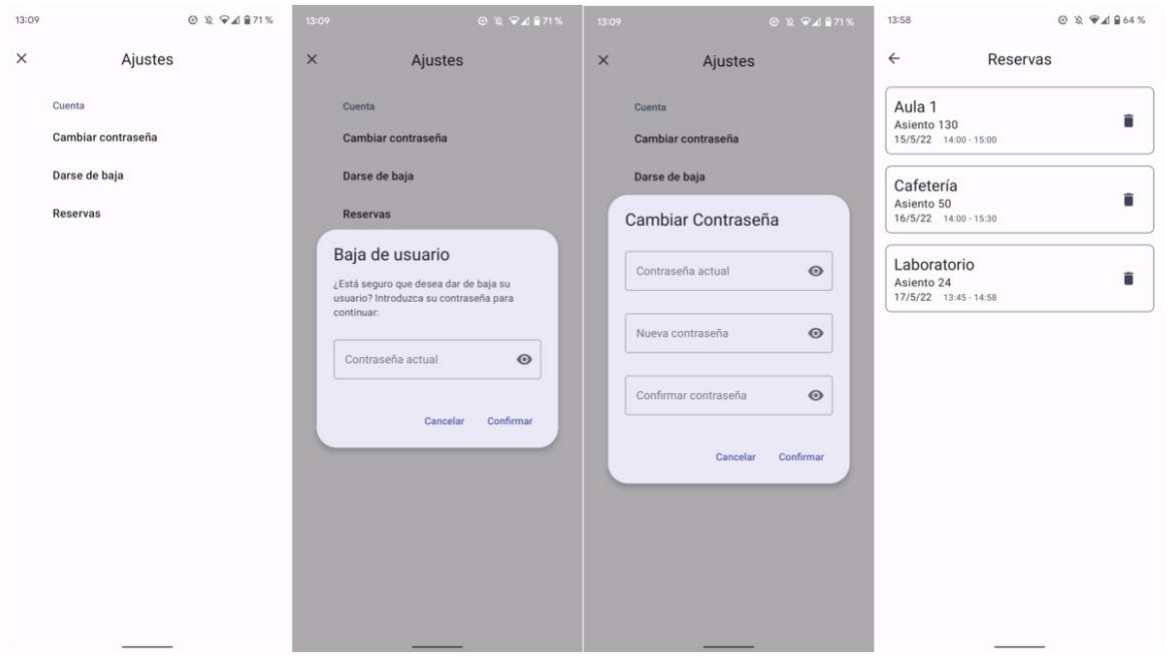


Figura 124 - Guía de usuario, Aplicación Android: Vista de ajustes

I.I.XIV. Vista de reservas

En caso de iniciar sesión con un usuario diferente del invitado se podrá ver un listado de las reservas realizadas que no han concluido. A estas se pueden acceder de tres formas diferentes.

Por una parte, se puede acceder a todas las reservas a través del panel de ajustes como se ha comentado en el apartado anterior (Cuarta captura de la Figura 124).

Por otra parte, se puede navegar al listado de reservas de una facultad a través del menú desplegable en la pantalla del listado de salas de una facultad o en la de información de facultad, como se puede ver en la primera y segunda captura de la Figura 125.

Finalmente se pueden ver las reservas de una sala accediendo desde el menú desplegable en la pantalla de información de sala (Tercera y cuarta capturas de la Figura 125).

Como se ha comentado en el apartado anterior, por cada reserva se muestra su información, un botón para eliminarla, y se puede acceder al mapa de sitios de la reserva haciendo clic en ella.

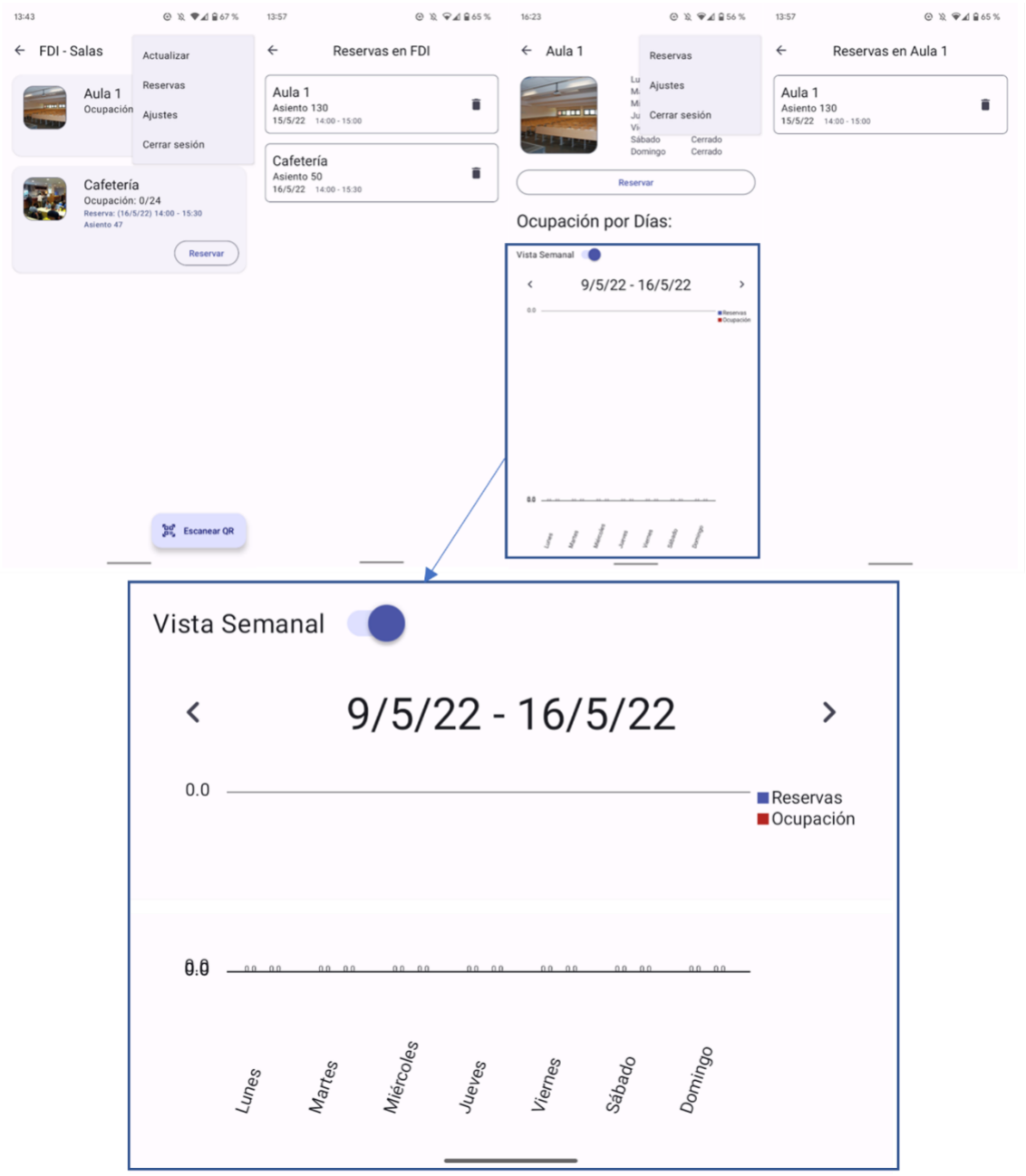


Figura 125 - Guía de usuario, Aplicación Android: Vista de reservas en facultad y sala

I.I.XV. Proceso de descarga del documento PDF de facultad

Como ya se mencionó anteriormente, una vez en la vista de facultades se da la posibilidad de descargar un documento PDF con el código QR de una facultad. Esta opción se habilita para los usuarios administradores.

Al hacer un clic prolongado sobre una facultad de la lista de facultades se muestra una ventana emergente en la que se puede elegir la opción “Descargar PDF-QR, la cual se puede visualizar en la primera captura de la Figura 126.

A continuación, si es la primera vez que se realiza una descarga desde la aplicación, se deben conceder permisos de acceso al almacenamiento del dispositivo (Segunda captura de la Figura 126). Una vez realizado este paso, se notifica al usuario si la descarga se ha producido con éxito o no (Tercera captura de la Figura 126).

Finalmente, para localizar el documento PDF que aparece en la cuarta captura de la Figura 126 se debe navegar a la carpeta de descargas del dispositivo y buscar el documento con el nombre de la facultad sin espacios.

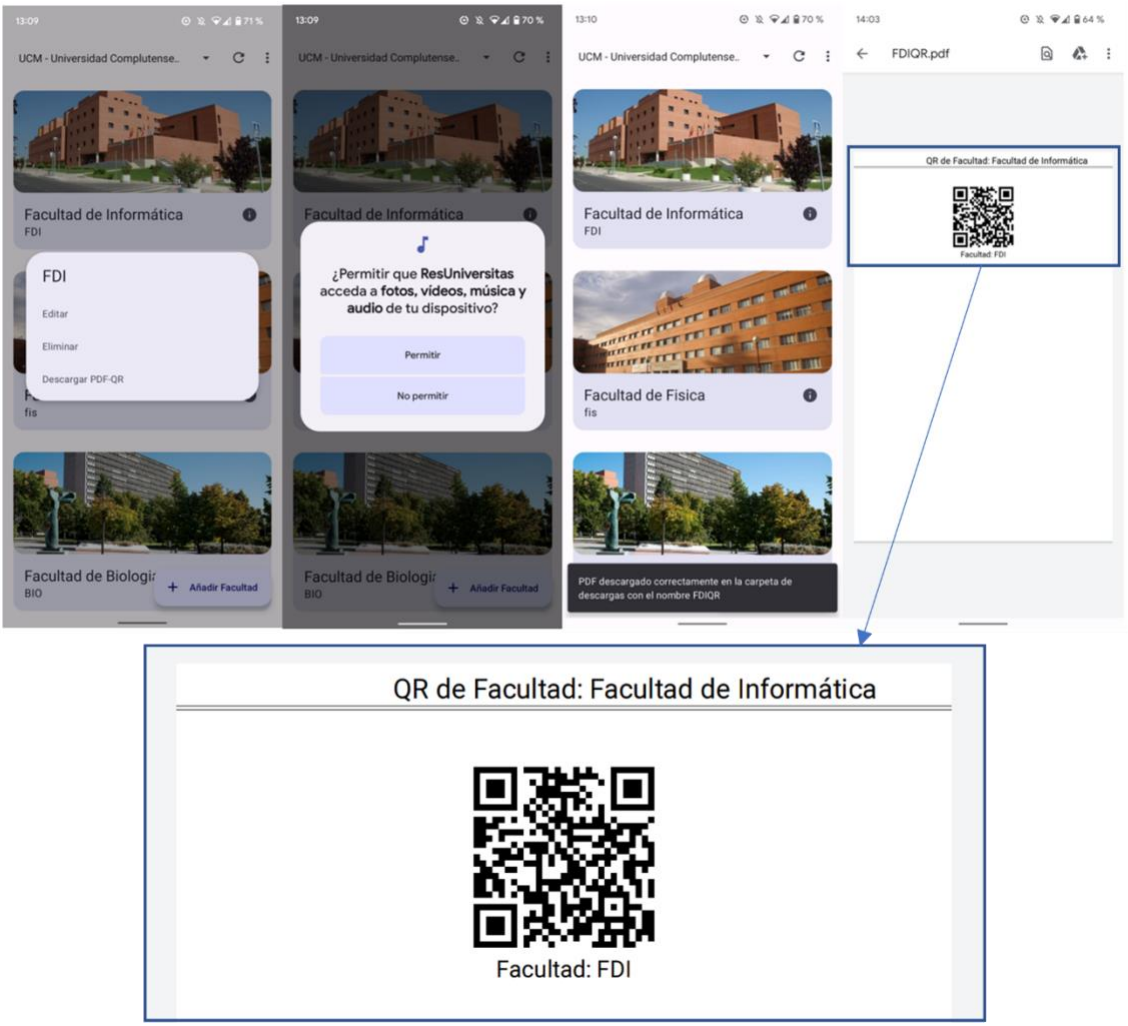


Figura 126 - Guía de usuario, Aplicación Android: descargar PDF de códigos QR de facultad con solicitud de permisos

I.I.XVI. Proceso de descarga del documento PDF de sala

Como ya se mencionó anteriormente, una vez en la vista de salas de una facultad se da la posibilidad de descargar un documento PDF con el código QR de la sala y la lista de códigos QR de los sitios de dicha sala. Esta opción se habilita para los usuarios administradores.

Al hacer un clic prolongado sobre una sala de la lista se muestra una ventana emergente en la que se puede elegir la opción “Descargar PDF-QR” (Primera captura de la Figura 127).

Al igual que al descargar el PDF de códigos QR de la facultad, se pedirán los permisos correspondientes en caso de ser la primera vez que se descarga un PDF de la aplicación. Se notificará al usuario si la descarga se ha producido con éxito o no (Segunda captura de la Figura 127).

Finalmente, para localizar el documento PDF, el cual se muestra en la tercera captura de la Figura 127, se debe navegar a la carpeta de descargas del dispositivo y buscar el documento con el nombre de la sala sin espacios.

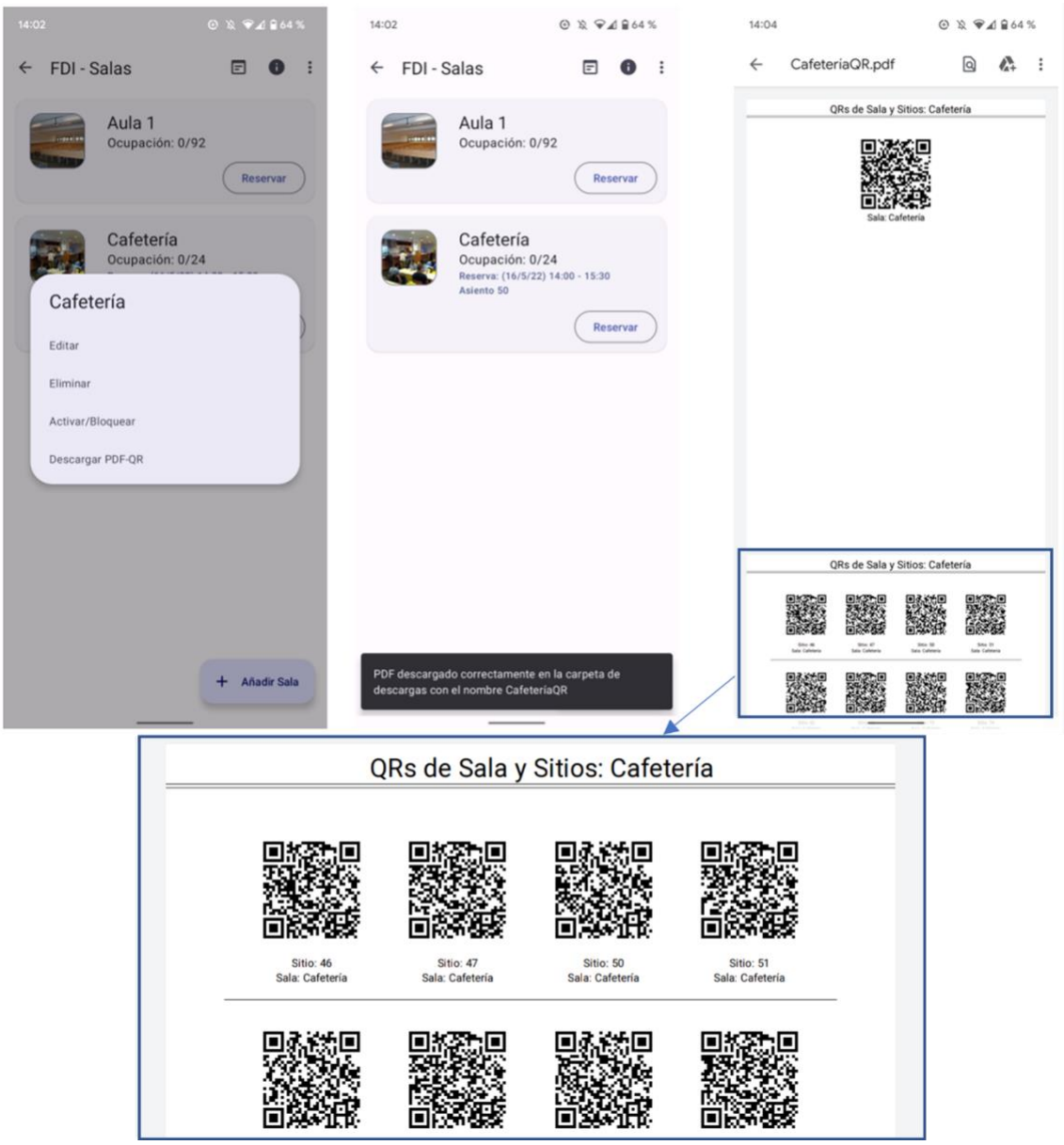


Figura 127 - Guía de usuario, Aplicación Android: Descarga de códigos QR de sala

I.I.XVII. Proceso de descarga del documento PDF de evento o noticia

Como ya se mencionó anteriormente, una vez en la vista del tablón de anuncios se da la posibilidad de descargar un documento PDF con el código QR del evento o noticia. Esta opción se habilita para los usuarios administradores.

Al hacer un clic prolongado sobre un evento o noticia se muestra una ventana emergente en la que se puede elegir la opción “Descargar PDF-QR” (Primera captura de la Figura 128).

Al igual que en los casos anteriores, se pedirán los permisos correspondientes en caso de ser la primera vez que se descarga un PDF de la aplicación. Se notificará al usuario si la descarga se ha producido con éxito o no (Segunda captura de la Figura 128).

Finalmente, para localizar el documento PDF mostrado en la tercera captura de la Figura 128 se debe navegar a la carpeta de descargas del dispositivo y buscar el documento con el nombre del evento o noticia sin espacios.

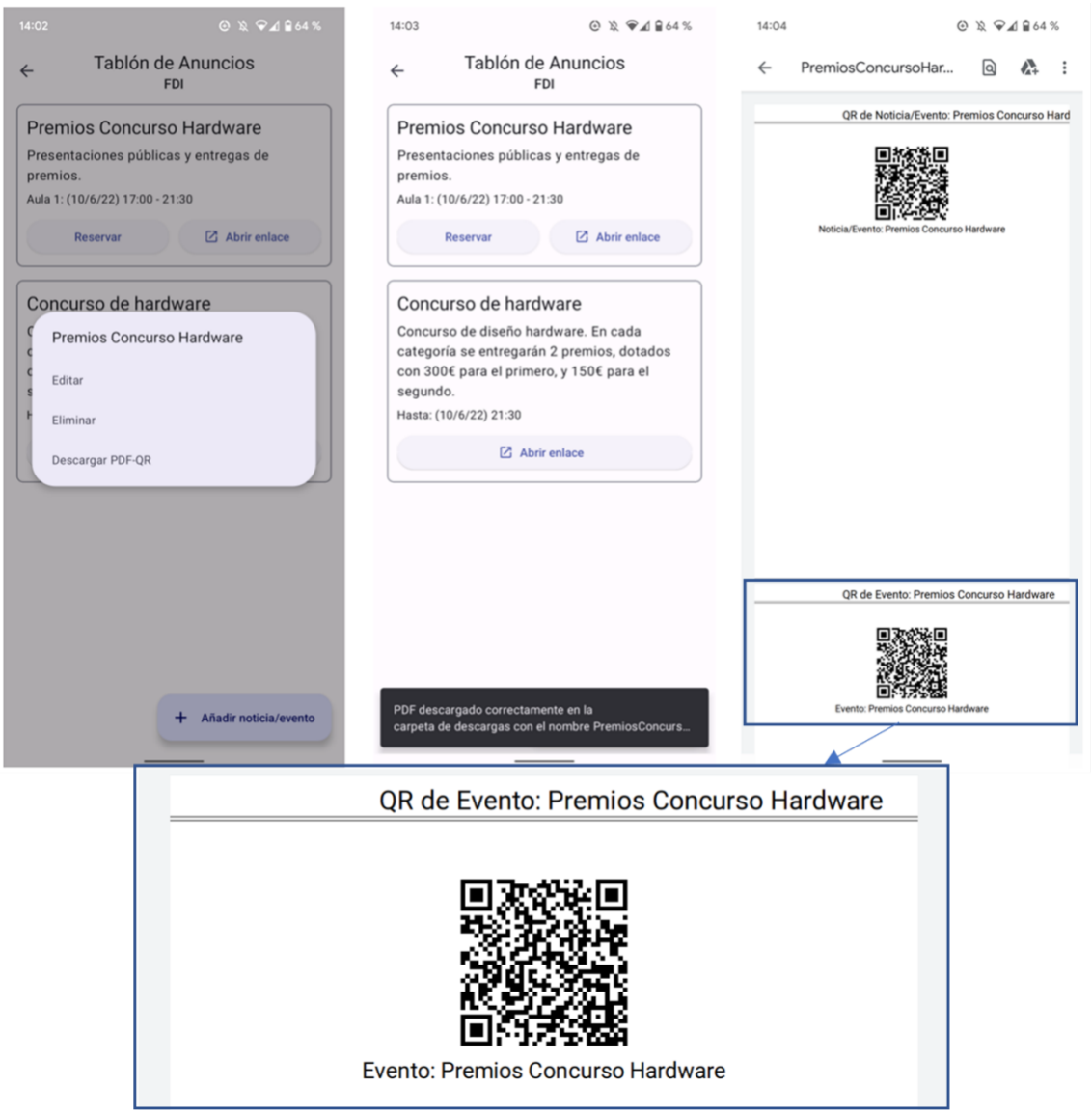


Figura 128 - Guía de usuario, Aplicación Android: Descarga de PDF de códigos QR de evento o noticia

I.II Aplicación web

En primer lugar, se debe señalar que únicamente se podrán utilizar las funcionalidades de la aplicación web por los usuarios administradores, por lo que, en la guía solo se hará alusión a este tipo de usuario.

Al acceder a la aplicación web, se muestra una página de bienvenida con una barra de navegación en la parte superior que da acceso al resto de funciones.

En primer lugar, si el usuario no se encuentra registrado, podrá acceder a la página de inicio donde se encuentra un mensaje de bienvenida, ver Figura 129. También, se da la posibilidad de navegar a la vista de información del proyecto web al hacer clic en el botón de “sobre nosotros” en la barra de navegación. De igual forma, se podrá visualizar el formulario de inicio de sesión haciendo clic en “iniciar sesión”.

Por último, es importante señalar que las opciones de listar facultades y crear facultad de la barra de navegación no están disponibles para usuarios no registrados.

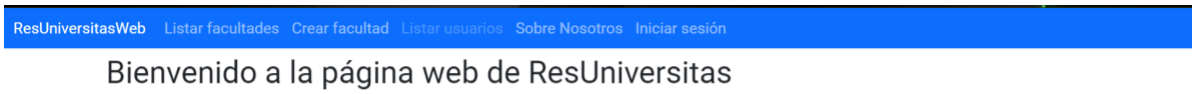


Figura 129 - Guía de usuario, Aplicación web: Pantalla de bienvenida

I.II.I. Vista de inicio de sesión

Por otra parte, al contar con un usuario y contraseña de perfil administrador, se podrá navegar a la vista de “inicio de sesión” y hacer clic en el botón “iniciar sesión” que se encuentra en la barra de navegación de la parte superior. Seguidamente, se mostrará un formulario de acceso para iniciar sesión, donde se introducirá el usuario y contraseña correspondientes como se puede observar en la Figura 130.

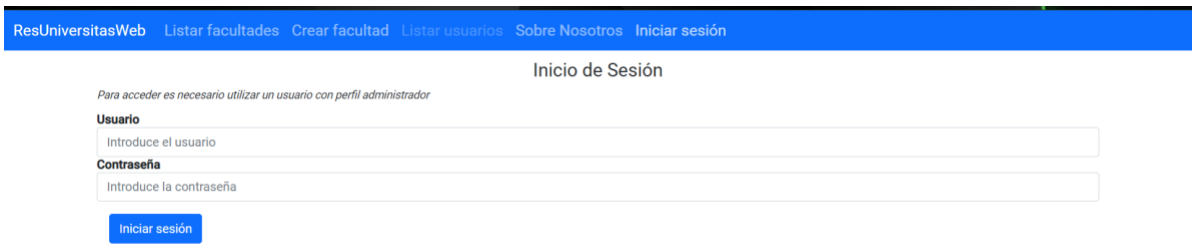


Figura 130 - Guía de usuario, Aplicación web: Pantalla de inicio de sesión

Cabe añadir que una vez se haya iniciado sesión se navegará a la vista de bienvenida donde la barra de navegación habrá cambiado su apariencia intercambiando la opción “iniciar sesión” por “cerrar sesión” y, se habrán habilitado las opciones para acceder a las pestañas de “listar facultades” o “crear facultad”, como se muestra en la Figura 131.

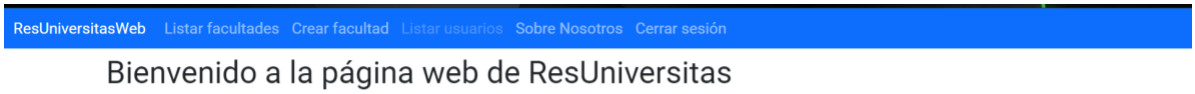


Figura 131 - Guía de usuario, Aplicación web: Pantalla de bienvenida con sesión iniciada

I.II.II. Vista de facultades

Una vez en la vista de inicio de sesión se muestra la lista de facultades, ver Figura 132.

Si se quiere navegar a la vista de edición de una facultad en concreto se debe hacer clic sobre el botón de editar de la facultad.

Por otra parte, se permite eliminar una facultad haciendo clic sobre el botón de eliminar propio de la facultad.

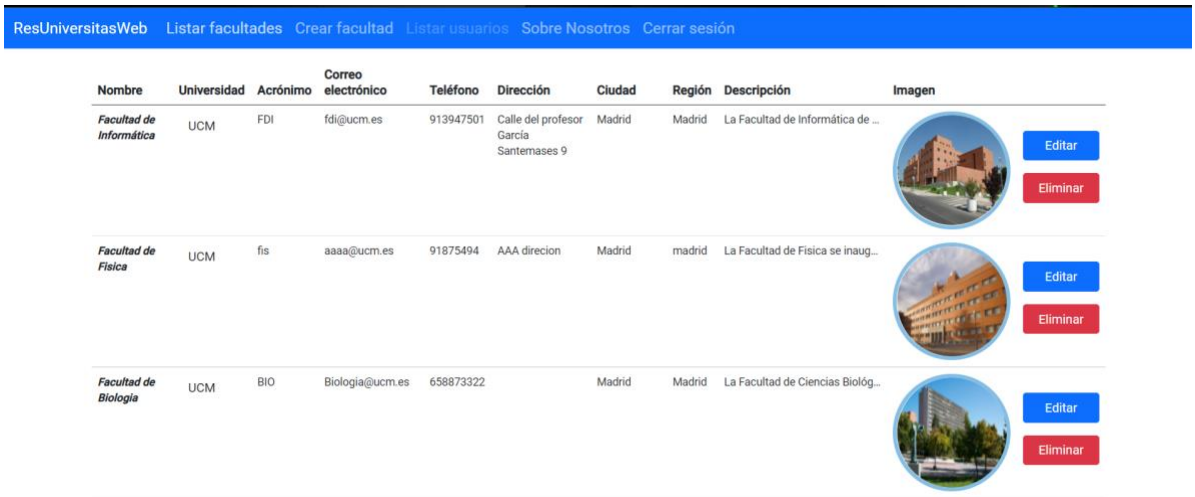


Figura 132 - Guía de usuario, Aplicación web: Pantalla de facultades

I.II.III. Vista de creación/edición de facultad

Una vez en la vista de crear facultad se muestra un formulario para crear una facultad nueva, de la misma forma se tendrá la posibilidad de cancelar el proceso de creación y volver a la vista de facultades, para ello, se debe hacer clic sobre el botón cancelar en la parte inferior de la vista, ver Figuras 133 y 134.

Por último, se debe destacar que el proceso de edición de facultad es similar al de creación de esta y, por tanto, se tienen que cumplir las mismas validaciones.

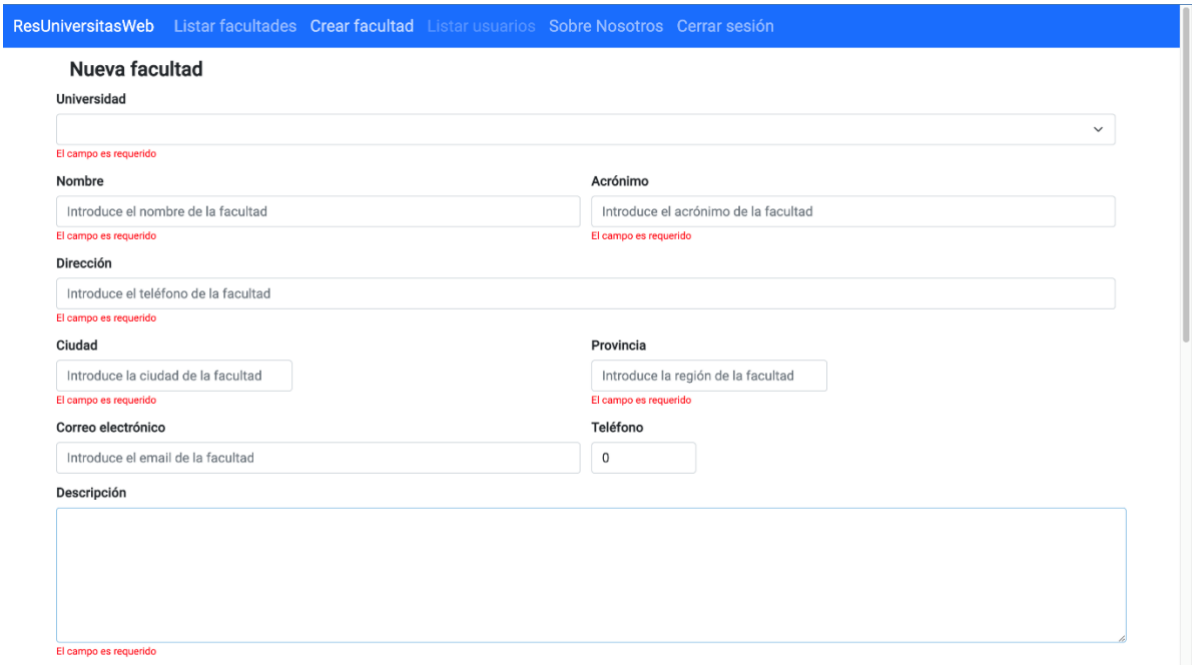


Figura 133 - Guía de usuario, Aplicación web: Pantalla de alta de facultad 1

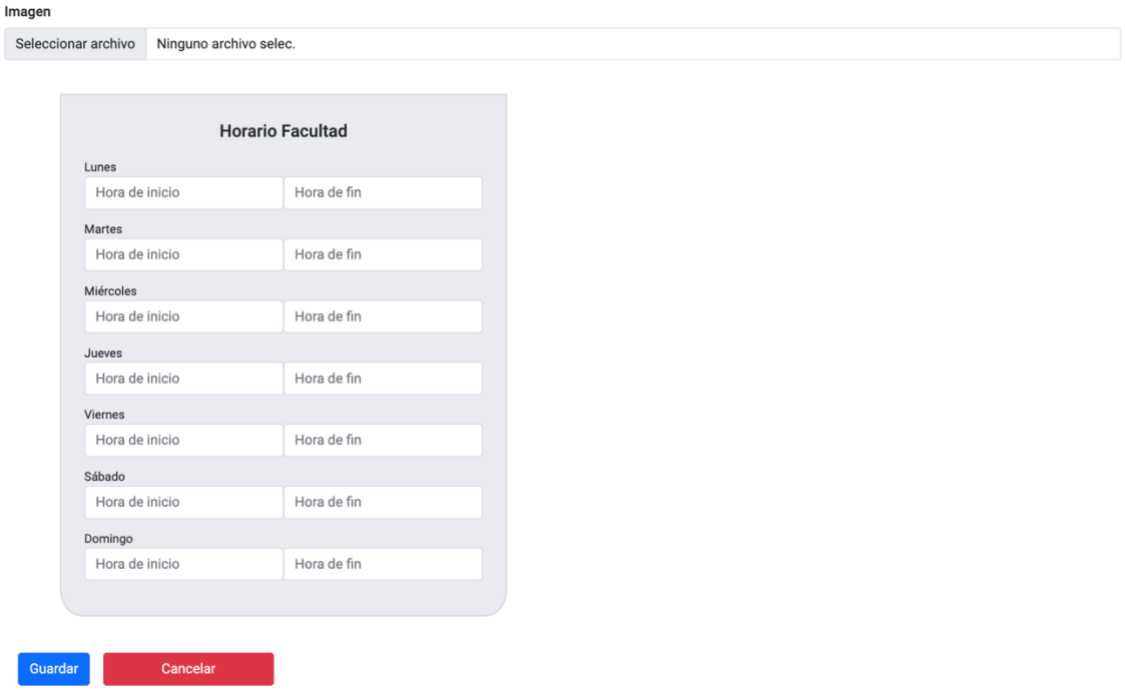


Figura 134 - Guía de usuario, Aplicación web: Pantalla de alta de facultad 2

I.II.IV. Vista de información del proyecto web

En la vista de información del proyecto web se muestra un mensaje informativo sobre el proyecto web, como se observa en la Figura 135.

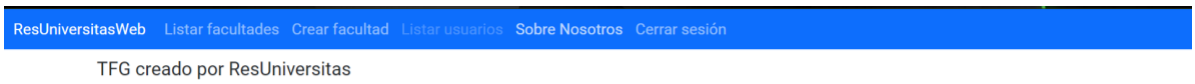


Figura 135 - Guía de usuario, Aplicación web: Pantalla de información del proyecto

ANEXO II. Preguntas de la evaluación

II.I. Preguntas sobre las funcionalidades disponibles para usuarios no registrado

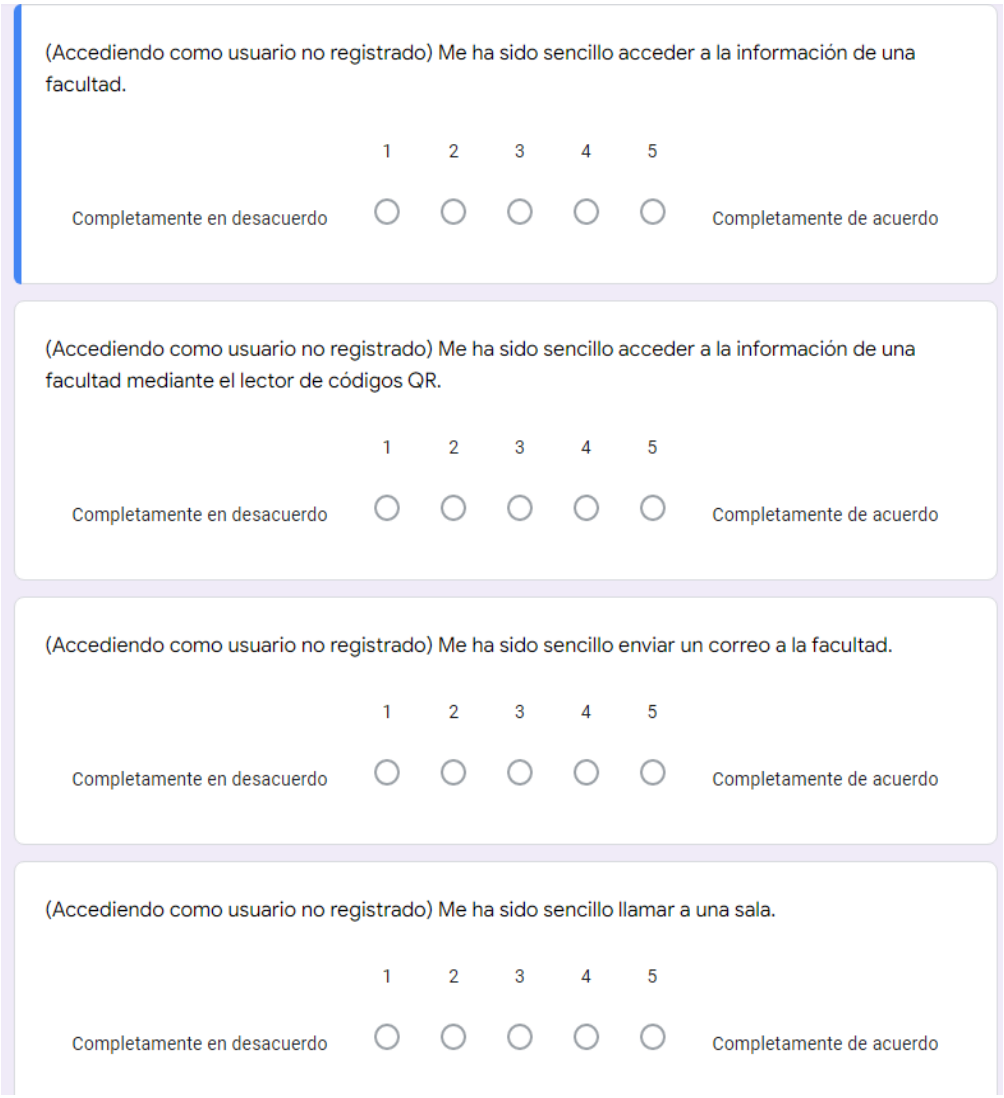


Figura 136 - Formulario: Usuario no registrado 1

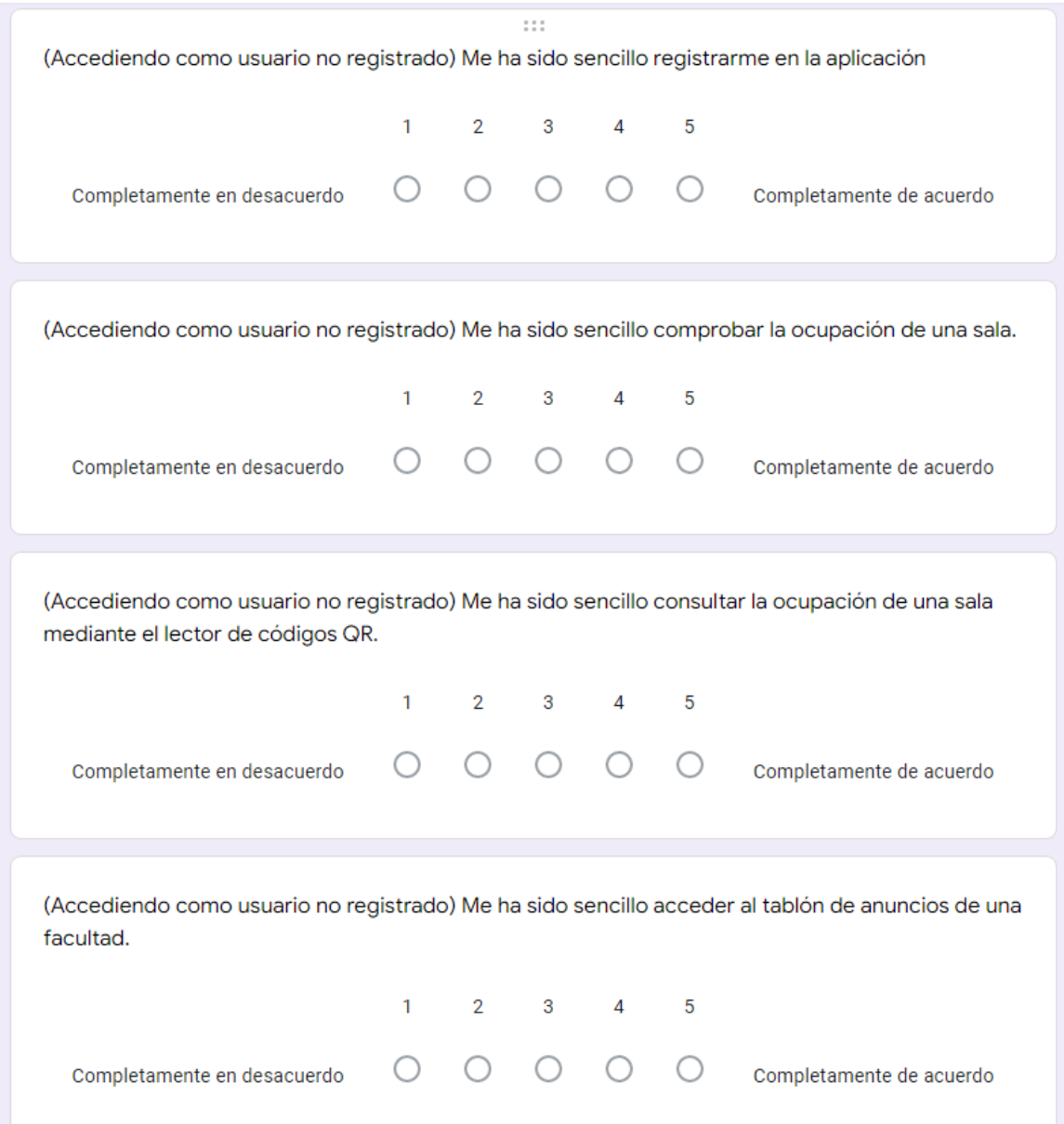


Figura 137 - Formulario: Usuario no registrado 2

II.II. Preguntas sobre las funcionalidades disponibles para usuarios registrados

(Accediendo como usuario registrado) Me ha sido sencillo realizar una reserva

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como usuario registrado) Me ha sido sencillo anular una reserva.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como usuario registrado) Me ha sido sencillo acceder a la lista de mis reservas.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como usuario registrado) Me ha sido sencillo reservar para un evento en la aplicación.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

Figura 138 - Formulario: Usuario registrado 1

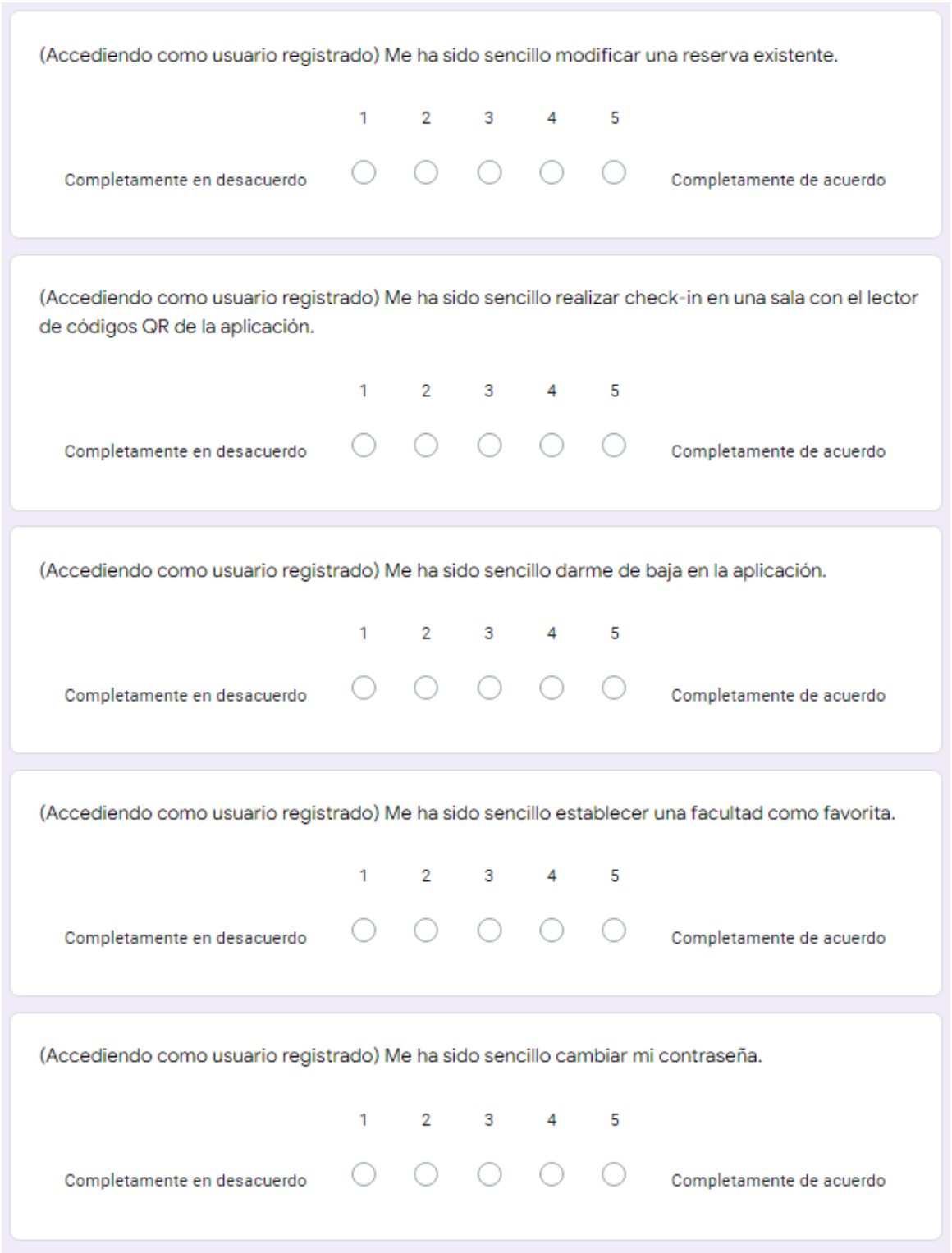


Figura 139 - Formulario: Usuario registrado 2

II.III. Preguntas sobre las funcionalidades disponibles para administradores

(Accediendo como administrador) Me ha sido sencillo añadir una facultad nueva.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como administrador) Me ha sido sencillo modificar una facultad existente.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como administrador) Me ha sido sencillo crear una sala nueva con su correspondiente mapa de sitios

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como administrador) Me ha sido sencillo modificar el mapa de sitios de una sala.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

(Accediendo como administrador) Me ha sido sencillo publicar un anuncio en una sala.

	1	2	3	4	5	
Completamente en desacuerdo	○	○	○	○	○	Completamente de acuerdo

Figura 140 - Formulario: Usuario administrador 1

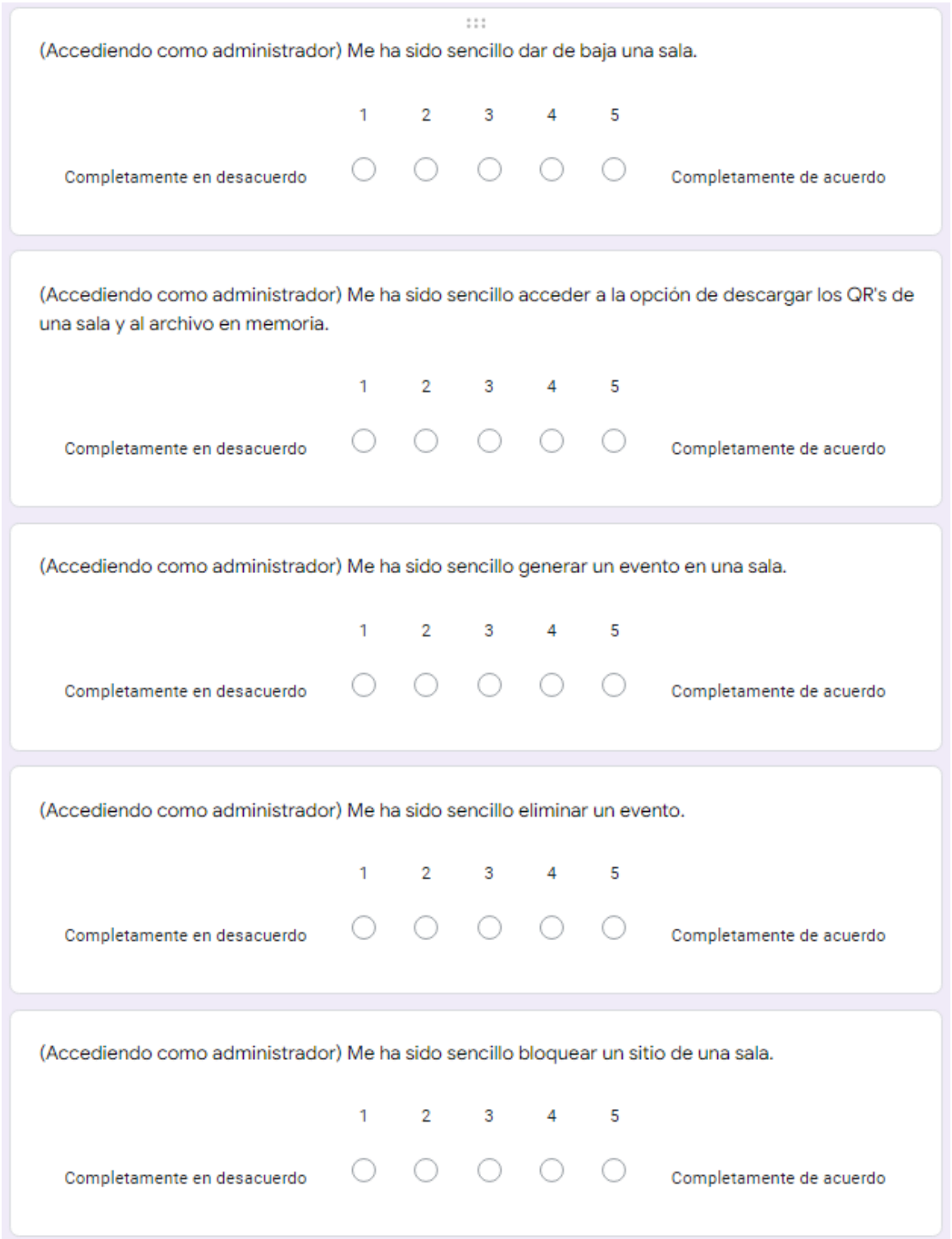


Figura 141 - Formulario: Usuario administrador 2