



UNIVERSIDAD
COMPLUTENSE
MADRID

Grado en Ingeniería Informática

Trabajo Fin de Grado

**Desarrollo de una aplicación para la gestión de espacios en
la Biblioteca de la Facultad de Informática**

Realizado por:

Carlos Gavidia Ortiz
David Gorricho San Juan
Iván Monterrubio Cerezo

Director:

Antonio Sarasa Cabezuelo

Facultad de Informática
Universidad Complutense de Madrid
Madrid, Junio 2018

I. Prólogo

“Las buenas noticias solo llegan a los que se embarcan dispuestos a naufragar”

Allá por el mes de febrero de 2017, tres amigos estudiantes de Ingeniería Informática de la Universidad Complutense de Madrid comenzaron a hablar sobre su Trabajo de Fin de Grado del curso siguiente. Tenían claro que querían realizar un proyecto que perdurara en el tiempo, que quedara para la posteridad y que sirviera de ayuda para gran cantidad de gente. Hoy, 16 meses después, estos tres alumnos por fin podemos decir que hemos finalizado un proyecto del que nos sentimos realmente orgullosos.

Tras tres años en la facultad tocaba hacer uso de todo el conocimiento adquirido y de demostrar a profesores y a nosotros mismos que éramos capaces de desarrollar un verdadero proyecto informático con todos sus apartados completamente desde cero. Además, nuestros años de universidad y los períodos interminables de estudio en la biblioteca, junto a algún que otro intento frustrado de estudiar por no haber puestos libres, nos habían hecho darnos cuenta de un grave problema existente en las bibliotecas.

Así pues, ya teníamos idea sobre la que realizar el trabajo de fin de grado: una aplicación móvil que permitiese ocupar el puesto y evitase que los descansos fuesen demasiado largos, todo ello haciendo uso de la geolocalización. Añadimos algunas funcionalidades extras a nuestra idea inicial, la pulimos un poco y nos embarcamos en la aventura del desarrollo del proyecto.

Casi 500 días después de la primera quedada del proyecto, tenemos una aplicación completamente funcional disponible en Play Store con su respectiva página web para la gestión por parte de los bibliotecarios. Confiamos en que se haga uso de ella en la biblioteca y que, ojalá, pueda expandirse a otras muchas más bibliotecas para ayudar a cuantos mayor número de estudiantes mejor.

II. Agradecimientos

“No es verdad que las personas paran de perseguir sueños porque se hacen viejos, se hacen viejos porque paran de perseguir sus sueños.”

No podemos comenzar esta memoria sin antes dar las gracias a nuestro director Antonio Sarasa Cabezuelo. Muchas gracias por confiar en el proyecto desde el primer momento, por todos los consejos que nos ha ido aportando y por todas las ideas que nos ha ido sugiriendo. Probablemente, este proyecto no podría haber sido desarrollado sin su ayuda.

Gracias a nuestras familias y novias por aguantarnos en los días más difíciles del desarrollo del proyecto, en los que siempre han estado ahí apoyando, y sobre todo por estos 4 largos años en los que hemos tenido momentos buenos y momentos malos, siendo en estos últimos donde más se ha notado su gran apoyo, inquebrantable confianza e imprescindible ayuda. Muchas gracias. Ellos, junto a nuestros amigos a los que también queremos agradecer todo su apoyo, ánimos, sus sugerencias sobre el proyecto y esos días tan necesarios que nos han ayudado a respirar, han sido los que nos han aportado las energías necesarias para poder seguir con ilusión en los momentos en los que las cosas se torcían un poco.

También, debemos agradecer al resto de profesores de la facultad que nos han dado clase durante estos años y que por tanto, de un modo u otro han aportado su granito de arena a este proyecto. A las bibliotecarias de la Facultad de Informática, a los alumnos de la misma y a todos los usuarios de la aplicación en general, muchas gracias por usarla y proporcionar el Feedback necesario con el que mejorar la aplicación.

Por último, no queríamos despedirnos sin hacer un par de menciones especiales a dos amigos que nos han ayudado mucho durante el proyecto. A ti, Mario, por estar ahí, ayudarnos con algunas dudas sobre Android que nos han surgido durante la realización del proyecto, gracias. Y a ti, Elena, por haber sido la betatester número uno, por tus recomendaciones en cuanto al diseño y funcionalidades, y estar siempre pendiente de nuestro progreso, muchas gracias también.

III. Resumen

Biblioteko es una aplicación ofrecida a estudiantes, profesores, visitantes y a todos los usuarios en general de la biblioteca de la Facultad de Informática que permite conocer qué puestos están ocupados en tiempo real. Pero no solo eso, Biblioteko permite enviar sugerencias a los bibliotecarios/as, consultar el horario de la biblioteca, conocer estadísticas de el tiempo de estudio y descanso y compararlo con el tiempo medio de todos los estudiantes...

Sin embargo, el principal atractivo de Biblioteko es intentar solucionar la problemática a la que cientos de estudiantes se ven sometidos durante los meses de enero y mayo. Durante estos meses las bibliotecas suelen estar más llenas y existen problemas con el número de puestos disponibles. Algunos estudiantes espabilados acuden a ellas y se van a descansar durante tiempos superiores a las dos horas, dejando su puesto ocupado con un boli e impidiendo que otro estudiante lo ocupe.

En este escenario se puede apreciar el gran potencial de Biblioteko, puesto que permite al usuario descansar hasta un máximo de 30 minutos, dependiendo de la cantidad de tiempo que haya estado estudiando. Si transcurrido el tiempo de descanso disponible, no ha vuelto a la biblioteca, su puesto será liberado automáticamente.

Además, Biblioteko cuenta con una página web diseñada exclusivamente para bibliotecarios/as y con acceso exclusivo para ellos. Desde esta página, los bibliotecarios/as tienen acceso de una manera muy rápida y sencilla a funcionalidades específicas para ellos tales como responder los mensajes enviados por los estudiantes, establecer el horario o consultar estadísticas de estudio referidas a cualquier rango de fecha de su interés.

Palabras clave

- Biblioteca
- Aplicación
- Estudio
- Puesto
- Descanso
- Biblioteko
- Informática

IV. Abstract

Biblioteko is an app available for students, teachers, visitants and all users of the library of the Computer Science Faculty that offers real time information about which study spaces are busy. Furthermore, with Biblioteko users can send suggestions to the librarians, check the library opening hours, consult statistics about their study time and their study breaks, and compare them with the average of all the students...

However, the main attraction of Biblioteko is a functionality that tries to solve the problem of a lot of students during the months of January and May. In these months, libraries are really busy and there are problems sometimes with the number of study spaces available. Some students go to the library and have never-ending breaks (even for more than 2 hours), leaving a pen in their study spaces and not letting other students use that study space.

We can appreciate the big potential of Biblioteko here, as it allows the user to have breaks for a period no longer than 30 minutes, according to how many time he/she has been studying. If after the break time that was available for the user, he/she has not returned to the library, his/her study space will be released automatically.

In addition, Biblioteko has a web page designed only and exclusively for the librarians and with exclusive access for them. From this web page, librarians have immediately and easy access to some specific functionalities for them such as answer the messages sent by the students, set the opening hours and view statistics of the desired range by them.

KeyWords

- Library
- Applicacion
- Study
- Study space
- Study break
- Biblioteko
- Computer Science

Índice

I. Prólogo	I
II. Agradecimientos	II
III. Resumen	III
IV. Abstract	IV
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estado del arte	3
1.4. Estructura de la memoria	5
2. Especificación de los requisitos de la aplicación	8
2.1. Funcionalidad aplicación web	8
2.2. Funcionalidad aplicación móvil	15
2.3. Fiabilidad	33
3. Tecnologías empleadas	34
3.1. Tecnologías en la aplicación web	34
3.2. Tecnologías en la aplicación móvil	35
3.3. Bases de datos	37
4. Arquitectura de la aplicación	38
4.1. Aplicación web	39
4.2. Aplicación móvil	39
4.3. Bases de datos	40
4.4. Servidor externo	40
4.5. APIs	40
5. Modelo de datos	41
5.1. Base de datos MySQL	41
5.2. Base de datos SQLite	47
5.3. Comunicación entre los modelos de datos	48
6. Diseño de la aplicación	50
6.1. Parte web	51
6.2. Parte móvil	51
7. Implementación de la aplicación	53
7.1. Aplicación web	53
7.1.1. Inicio de sesión	54
7.1.2. Buzón de sugerencias	57
7.1.3. Estadísticas	59
7.1.4. Horario	61
7.1.5. Cerrar sesión	64
7.1.6. Quiénes somos	64
7.1.7. Header	64
7.1.8. Footer	65

7.2.	Aplicación móvil	67
7.2.1.	Starter	70
7.2.2.	Iniciar sesión	72
7.2.3.	Registrar	75
7.2.4.	Bibliotecas	77
7.2.5.	Plantas	78
7.2.6.	Mesa	83
7.2.7.	Ocupar puesto	86
7.2.8.	Vaciar puesto	88
7.2.9.	Hacer descanso/Finalizar descanso	92
7.2.10.	Estadísticas	99
7.2.11.	Contacto	101
7.2.12.	Horario	103
7.2.13.	Modificar contraseña	105
7.2.14.	Cerrar sesión	109
7.2.15.	Ayúdanos a mejorar	110
7.2.16.	Service Ubicación	111
7.2.17.	Alarma	115
7.2.18.	Alarma Estoy Vivo	117
7.3.	Otros componentes del sistema	118
8.	Evaluación de usuarios y pruebas	124
8.1.	Inicio del proyecto	124
8.2.	Durante el proyecto	124
8.3.	Tras el proyecto	124
9.	Conclusiones y trabajo futuro	132
9.1.	Conclusiones	132
9.2.	Conclusions	133
9.3.	Trabajo futuro	134
9.4.	Trabajo individual	135
9.4.1.	Carlos Gavidia Ortiz	135
9.4.2.	David Gorricho San Juan	137
9.4.3.	Iván Monterrubio Cerezo	140
10.	Bibliografía	142
11.	Apéndices	144
11.1.	Apéndice A: Manual de instalación	144
11.1.1.	Página web	144
11.1.2.	Android Studio	144
11.1.3.	Sublime Text	144
11.1.4.	MySQL	144
11.1.5.	Aplicación Android	145
11.2.	Apéndice B: Manual de usuario	145
11.2.1.	Aplicación web	145
11.2.1.1.	Iniciar sesión	145
11.2.1.2.	Buzón de sugerencias	146
11.2.1.3.	Estadísticas	147
11.2.1.4.	Establecer Horario	148
11.2.1.5.	Descargar app	148
11.2.1.6.	Quiénes somos	149

11.2.1.7. Contáctanos/Redes Sociales	149
11.2.1.8. Cerrar sesión	149
11.2.2. Aplicación móvil	149
11.2.2.1. Iniciar sesión	149
11.2.2.2. Registrarse	150
11.2.2.3. Bibliotecas	151
11.2.2.4. Plantas	152
11.2.2.5. Hacer descanso	153
11.2.2.6. Liberar puesto	154
11.2.2.7. Estadísticas	154
11.2.2.8. Contacto	155
11.2.2.9. Horario	156
11.2.2.10. Modificar contraseña	157
11.2.2.11. Cerrar sesión	157
11.2.2.12. Ayúdanos a mejorar	158

12. Anexo

159

1. Introducción

En este primer capítulo de la memoria se va a desarrollar la motivación del proyecto desarrollado, cuáles han sido sus objetivos y la estructura general de esta memoria. Además, es necesario destacar que se ha realizado la publicación de la aplicación ‘Biblioteko’ en Google Play y registrado el dominio <http://biblioteko.es> en el que se encuentra alojada la aplicación web.

In this first chapter of the memory, the motivation, objectives of the project and the general structure of the project are going to be explained. In addition, it is important to mention that the application Biblioteko has been released in Google Play and the webpage is located in the domain <http://biblioteko.es>

1.1. Motivación

Durante los meses de Enero y Mayo, la ocupación en las bibliotecas llega a su punto máximo, repletas de estudiantes que tratan de ocupar hasta el último puesto disponible.

Para ayudar a que todos los estudiantes puedan estudiar, la mayoría de las bibliotecas retrasan los horarios de cierre, de forma que los estudiantes puedan aprovechar las instalaciones durante más tiempo. Durante el tiempo de estudio, es común en los estudiantes realizar descansos con el fin de despejar la mente periódicamente.

Cuando se realizan los descansos es cuando se producen ciertos problemas con respecto a la ocupación de los sitios de estudio de la biblioteca. Cuando una persona abandona su sitio y regresa del descanso puede verse sorprendido y que su sitio haya sido ocupado por otra persona. Por esa razón, muchos estudiantes se ven obligados a dejar objetos encima de su puesto antes de tomar ese descanso para despreocuparse de este problema. Por ello, es habitual ver varios sitios con objetos y sin ocupación aparentemente.

Sin embargo, el hecho de reservar puestos hace que surja otra problema puesto que, a veces, algunos estudiantes reservan puestos por tiempos excesivamente largos, privando a otros usuarios de poder ocupar un puesto.

El sistema que se propone en este trabajo tiene como objetivo ayudar a aquellos estudiantes que quieran descansar (siempre que respeten y no sobrepasen el tiempo de descanso) a que ya no tengan que dejar objetos encima de la mesa, con el riesgo que supone, debido a que pueden sustraerlo. Así un estudiante que entrase a la biblioteca vería que este puesto está ocupado y no puede ocuparlo. Un mapa de la ocupación de la biblioteca será donde el estudiante podrá visualizar los sitios disponibles.

Además, el sistema sería el encargado de liberar automáticamente un puesto en el caso de que el usuario haya superado el tiempo de descanso que tenía disponible. También ofrece la posibilidad de consultar estadísticas, el horario disponible y permite enviar mensajes a los/as bibliotecarios/as.

La aplicación desarrollada también dispondría de una aplicación web, de acceso exclusivo para los bibliotecarios en la cual podrían contestar los mensajes de los estudiantes, consultar estadísticas generales de la biblioteca y especificar su horario.

During the months of January and May, the occupation level of the libraries is really high, full of students that occupy every study space.

To help the students, lot of libraries open until late, in order to ease the students to approach the libraries more. During the study time, it is usual to have study breaks to rest few minutes during the long study journeys.

When the students have their study breaks some problems occur related with the study spaces of the library. When someone leave his/her study space for having a break and return from it, can happen that in his/her study space another student can be seated. For this reason, some students leave some belongings in their student space to avoid this problem. For this reason, it is common to see ‘free’ study spaces but with belongings.

However, this generate another problem, as some students leave their study spaces for really long breaks not letting other students occupy that space.

The system proposed in this project has the objective of helping the students that want to have a break (if they do not exceed their study break time), avoiding to leave belongings in their spaces with the risk that they could be stolen. When a student access to the library, he/she can see the spaces that are free and the ones who are empty in a map, having the possibility of occupy the free ones.

In addition, the system would release the spaces automatically in the case that a student would exceed his/her study break time. Also, it offers the possibility of see statistics, consult the opening hours and send messages to the librarians.

The application developed would also have a web application, exclusive for the librarians in which they could answer messages from the students, see general statistics of the library and set its opening hours.

1.2. Objetivos

El objetivo principal en este proyecto ha sido desarrollar un sistema informático para gestionar la ocupación de los sitios de estudio de una biblioteca y ayudar a los bibliotecarios y estudiantes en este problema. Este objetivo principal se concreta en los siguientes subobjetivos:

- Diseñar y desarrollar una aplicación web sencilla e intuitiva orientada a los/as bibliotecarios/as o responsables de cada biblioteca que les ayude a realizar sus funciones más eficazmente.
- Permitir a los bibliotecarios desde la web responder mensajes, ver estadísticas y cambiar el horario de cierre y de apertura de la biblioteca
- Desarrollar una app sencilla e intuitiva orientada hacia los usuarios que les permita gestionar el uso de los puestos de estudio de las bibliotecas.
- Almacenar la información de la actividad de los usuarios de las bibliotecas con el fin de facilitar la gestión de los puestos de estudio en las bibliotecas.
- Conocer mediante la app si cada estudiante se encuentra en la biblioteca o no para así ofrecerle unas funcionalidades acorde a su posición.

- Desarrollar una aplicación móvil que permita a todos los estudiantes hacer uso de su tiempo de descanso disponible, sin excederse, ni correr el riesgo de que al realizar su descanso, su puesto hubiera sido ocupado o sus pertenencias sustraídas.
 - Desarrollar un método de comunicación directa entre los usuarios de la app y los responsables de la biblioteca.
 - Desarrollar herramientas que permitieran a los estudiantes en todo momento desde la app conocer sus estadísticas y el horario de la biblioteca.
-

The main objective of this project has been develop a system to manage the study spaces of a library and help the students and librarians in this problem. This main objective can be divided into these subobjectives:

- Design and develop a simple and intuitive web page for librarians that help them with their functions.
- Let the librarians from the webpage answer messages, see statistics and set the opening hours of the library.
- Develop a simple and intuitive app for students that allows them to manage the student spaces of the libraries.
- Save the information of the activity of the users in order to ease the management of the study spaces in libraries.
- Know in the app the location of the user to offer him/her specific functionalities according to its location.
- Develop an application that allows to every student use his/her study break time (not exceeding it) and avoiding the risks of losing his/her study space or his/her belongings.
- Develop a communication channel between users and librarians.
- Develop functionalities that allow the users to consult their statistics and the opening hours in every moment.

1.3. Estado del arte

Tras la búsqueda de aplicaciones similares, no se han encontrado muchas coincidencias. Sin embargo, sí que hay algunos competidores con respecto al proyecto.

El más similar que existe son las pantallas de los puestos disponibles de los laboratorios situados en la planta 2 de la facultad de informática de la Universidad Complutense de Madrid. En ellos aparecen indicados de un modo gráfico qué puestos se encuentran ocupados y cuáles están disponibles para su uso. No ofrece al usuario ningún tipo de funcionalidad extra como podría ser el tiempo de estudio.

Además, existe un servicio web [1] que permite consultar de forma textual los puestos y laboratorios que están abiertos y disponibles. Se trata de una interfaz no muy amigable, bastante discreta y de un servicio muy escondido en la página web de la Facultad de Informática.

Otro referente en cuanto al ámbito académico y bibliotecario son la reserva de salas y de espacios de trabajo en grupo de la facultad de Educación de la UCM. Accediendo a su aplicación web [2] se pueden consultar qué salas se encuentran disponibles y realizar su reserva.

Se trata de una aplicación sencilla, con una interfaz fácil de usar. Una vez se haya producido el logueo del usuario con sus datos personales, se procede a introducir la fecha para la cual se quiere realizar la reserva. Tras esto, se muestran al usuario todas las salas de trabajo en grupo disponibles para esa fecha y se permite la reserva de cualquiera de las salas disponibles.

Sin embargo, aunque los dos referentes previos sí que están relacionados con el ámbito educativo, no se trata de aplicaciones en las que se pueda reservar específicamente un puesto en concreto. Es en el terreno de las aplicaciones móviles pero en otros ámbitos en el que sí se han encontrado aplicaciones que permitan la ocupación de un puesto determinado.

En primer lugar, una de las aplicaciones que se tuvieron como referencia a la hora de realizar el proyecto es la aplicación móvil de Cinesa [3]. A pesar de ser una aplicación que cuenta con múltiples funcionalidades, la que está relacionada con este proyecto es la de la compra de entradas para una determinada película.

Una vez seleccionada la película y la sesión en cuestión sobre la que se quiere efectuar se procede a la selección de la butaca deseada. En este proyecto la similitud de la película y la sesión serían la selección de la biblioteca y la planta para así seleccionar el puesto deseado.

Tras ello, se muestra al usuario una ventana en la cual aparece una representación fidedigna de los diferentes puestos ocupados y libres de la sala. Se trata de una interfaz sencilla, sin excesos que permite al usuario seleccionar fácilmente la butaca deseada.

Otra de las aplicaciones móviles que presenta un sistema de ocupación de puestos es la de la compra de billetes de autobús de PLM.[4]

Se trata de una aplicación con una interfaz sencilla que permite, tras haber seleccionado las fechas en las que se quiere viajar, el número de billetes deseados y el origen y destino del viaje, seleccionar cuáles son los asientos que se desean ocupar.

Para la representación del mapa de puestos se ha optado por un mapa sencillo, en el que cada butaca representa una butaca del autobús y con elementos como el volante y las escaleras que permiten al usuario situarse espacialmente en el autobús.

After the search of similar applications, not a lot of coincidences have been found. However, there are some competitors in relation with the project.

The most similar one are the screens located in the 2nd floor of the Computer Science Faculty in the UCM that offer information about the available spaces. In them, study spaces are displayed graphically indicating which of them are free and which are not. It does not offer any other information to the users like could be the study time.

Also, there is a web service [1] that brings the possibility of consult in a text format the

spaces and labs that are open and available. It has not a friendly interface and is a service a bit hide in the faculty webpage.

Another referent in the academic and librarian ambit is the web page of the faculty of education of UCM that let the users book study group rooms. In its website [2] , study rooms can be checked and booked if available.

It is a simple application with an easy to use interface. Once the user has been logged in with his/her personal data, the booking desired date is introduced. After this, all the rooms available are shown to the user and he/she can proceed with the booking of it.

However, although the two previous referents are related with education, they are not application in which users can book specifically a space. It is in mobile applications and in other ambits where this kind of application have been found.

First of all, one of the applications that was taken into account when developing the project is the Cinesa [3] app. Although it is an application with a lot of functionalities, the one related with this project is the solution that appears when buying tickets for a movie.

After choosing the movie and the session a window that offers the seats available is displayed. In that window, a representation of the seats is displayed and the desired seats can be selected. The similarity with the project would be the relation between choose the movie and session with choose the library and floor.

It has a simple interface with no superfluous options that let the user choose easily the desired seat.

Another application that has an occupation system is the one of PLM [4] for buy bus tickets.

It is an app with a simple interface that brings the user the ability to select the desired seats after having chosen date, number of tickets and origin and destination.

To represent the seats map, a simple map has been used in which every seat represents a seat from the bus and with elements such as the steering wheel and the stairs that help the user with his/her location in the bus.

1.4. Estructura de la memoria

La estructura de nuestra es la siguiente:

- **Capítulo 1: Introducción**

En este capítulo se describe la motivación del proyecto, los objetivos planteados y un estado del arte sobre otras aplicaciones similares a la desarrollada.

- **Capítulo 2: Especificación de requisitos de la aplicación**

En este capítulo se describe la especificación de las funciones del sistema desarrollado.

- **Capítulo 3: Tecnologías empleadas**

En este capítulo se describen las herramientas tecnológicas usadas en el trabajo.

- **Capítulo 4: Arquitectura de la aplicación**

En este capítulo se presenta la arquitectura utilizada para desarrollar el sistema.

- **Capítulo 5: Modelo de datos**
En este capítulo se muestra el modelo de datos utilizados para realizar la persistencia de la información del sistema desarrollado.
- **Capítulo 6: Diseño de la aplicación**
En este capítulo se describe el diseño de la aplicación web y de la aplicación móvil.
- **Capítulo 7: Implementación de la aplicación**
En este capítulo se describe la implementación de todo el proyecto implementada.
- **Capítulo 8: Evaluación de usuarios y pruebas**
En este capítulo se describe cómo se ha realizado la evaluación del sistema y se presentan los resultados obtenidos.
- **Capítulo 9: Conclusiones y trabajo futuro**
En este capítulo se presentan las principales conclusiones de este trabajo, la aportación personal que ha realizado cada miembro del grupo, y se plantean algunas posibles líneas de trabajo futuro. También daremos algunas mejoras al proyecto que podrían potenciar la aplicación.
- **Capítulo 10: Bibliografía**
En este capítulo se enumeran las referencias utilizadas durante el proyecto.
- **Anexo I: Guía de uso**
En este anexo se encuentra una guía de cómo usar de una forma correcta la aplicación y la página web.
- **Anexo II: Guía de instalación de la aplicación**
En este anexo se explica paso a paso cómo instalar y configurar la aplicación y la página web.

The main structure of the memory is as follows:

- **Chapter 1: Introduction**
In this chapter, motivation of the project, objectives and state of art are described.
- **Chapter 2: System requirements specification**
In this chapter system requirements specification are described.
- **Chapter 3: Technologies used**
In this chapter are described the technologies used in the project.
- **Chapter 4: Application architecture**
In this chapter, the architecture used for the project is presented.
- **Chapter 5: Data model**
In this chapter, the data model used for guarantee the persistence of the information is shown.
- **Chapter 6: Design of the application**
In this chapter is described the design of the web application and the Android application.
- **Chapter 7: Implementation of the application**
In this chapter is described the implementation of all the project.
- **Chapter 8: Evaluation of the users and testing**
In this chapter is described how the evaluation has been done and the obtained results.

- **Chapter 9: Conclusions and future work**

In this chapter, conclusions are presented in addition to the individual work of every member of the team. Also some future solutions are provided.

- **Chapter 10: Bibliography**

In this chapter, references used are included.

- **Annex I: Use guide**

In this annex a guide explaining the correct use of the web application and the Android app can be found.

- **Annex II: Installation guide**

In this annex is explained step by step how to use the web application and the Android app.

2. Especificación de los requisitos de la aplicación

En esta sección se van a mostrar la especificación de requisitos del sistema. Para ello se ha dividido en varias partes. En la primera parte, se presentan los requisitos referidos a la parte web del proyecto. En la segunda parte aparecen los referidos a la parte Android. Finalmente en la tercera es un apartado reservado para la fiabilidad del proyecto.

2.1. Funcionalidad aplicación web

La página web será una herramienta con la que contarán y podrán consultar todos los días los/as bibliotecarios/as y podrán realizar algunas de sus funciones diarias más rápidamente.

Por esto el desarrollo de la página web tiene como objetivo que el bibliotecario pueda contestar y leer los correos del buzón de sugerencias que los alumnos hayan enviado, actualizar los horarios de apertura y cierre de la biblioteca o consultar distintas estadísticas de ocupación y estudio.

A continuación en la Figura 1 se presenta el diagrama de casos de uso, que resume todas las funcionalidades que tendrá el usuario de la página web.

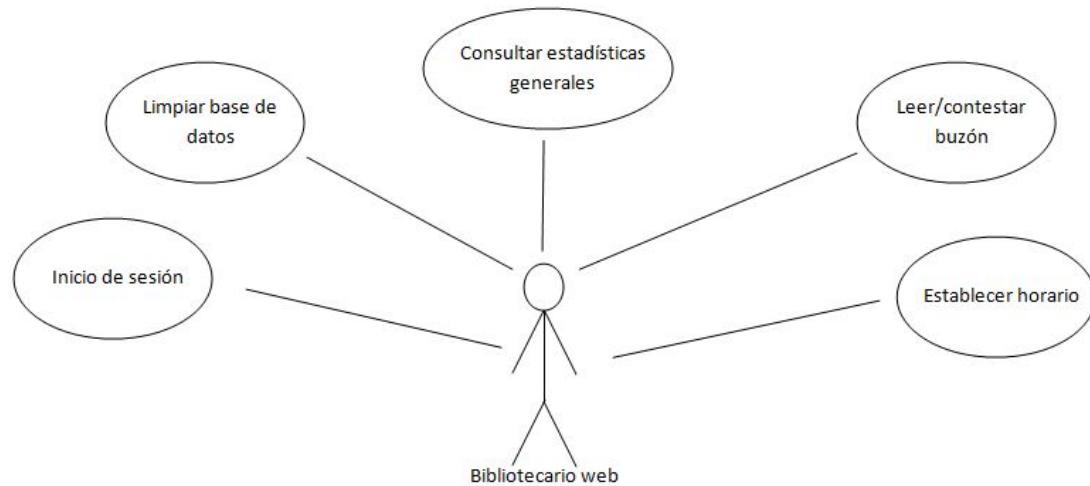


Figura 1: Casos de uso aplicación web

Funcionalidad 1: Inicio de sesión

Será la pantalla principal de la aplicación web y en ella el bibliotecario podrá iniciar su sesión introduciendo una cuenta y contraseña(ambas suministradas por la facultad, por lo que todos los bibliotecarios iniciarán sesión con la misma cuenta) en dos áreas de texto designadas para ello. Si los datos suministrados no coinciden con los de la base de datos de la aplicación, el usuario será notificado.



Figura 2: Iniciar sesión web

Funcionalidad 2: Limpiar base de datos

Esta función se realizará automáticamente, sin intervención del usuario, y consistirá en una limpieza de la base de datos, es decir las estadísticas de los usuarios pasado un tiempo, para evitar el colapso de la base de datos.



Figura 3: Limpieza de base de datos

Funcionalidad 3: Establecer horario

Como la biblioteca no tendrá unos horarios fijos durante todo el año(por ejemplo en épocas de exámenes estará abierta más tiempo),el bibliotecario se encargará de escribir en la página el horario de cada día de la semana; el horario de apertura y de cierre gracias a la interfaz sencilla, informando además de periodos no lectivos o días especiales.

The screenshot shows a web interface for setting a schedule. At the top, there's a header bar with the BiblioMasters logo, a search bar containing the URL <https://www.bibliomasters.com>, and several navigation links: Baja usuario, Limpiar base de datos, Sancionar usuario, Estadísticas, Buzón de sugerencias, Establecer horario (which is highlighted in blue), and Cerrar sesión.

The main content area has a light blue background. On the left, there's a section titled "Días:" with a list of days of the week, each with a checkbox. The checked days are Lunes, Martes, Miércoles, Jueves, and Viernes. To the right of this list is a large rectangular input field divided into three horizontal sections: "Planta 1", "Planta 2", and "Planta3".

Below these sections are two time inputs: "Apertura:" set to 9:00 and "Cierre:" set to 20:30. To the right of the input field, there's a "Localización:" field containing the address "Profesor Gómez Santamaría, 9".

On the far right, there's a dashed-line box containing optional fields: "Añadir información adicional" (checkbox), "Título:" (text input field with placeholder "Insertar título"), and "Contenido:" (text input field with placeholder "Introducir el contenido").

Figura 4: Establecer horario

Funcionalidad 4: Consultar estadísticas generales

El bibliotecario podrá ver las mismas estadísticas que las recogidas en la app.



Figura 5: Consultar estadísticas generales

Funcionalidad 5: Leer/contestar buzón de sugerencias

Los correos que lleguen al bibliotecario de los usuarios de la app a partir del buzón de sugerencias podrán ser leídos y contestados a la persona que los envió. Además, el bibliotecario no sabrá quienes los han mandado (serán anónimos).

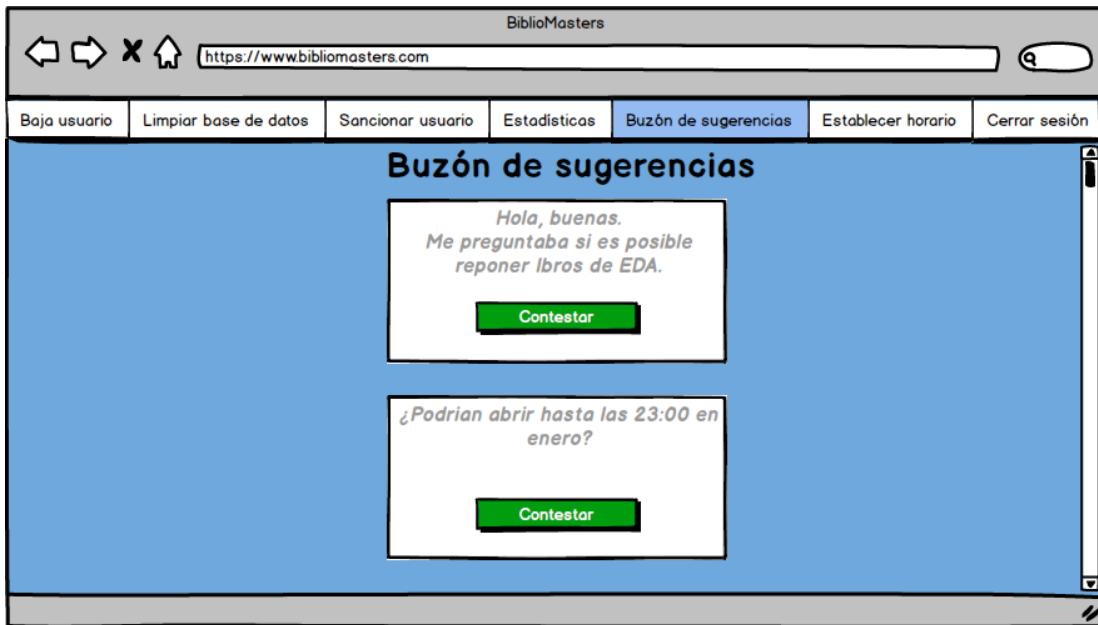


Figura 6: Buzón de sugerencias - Mensajes recibidos

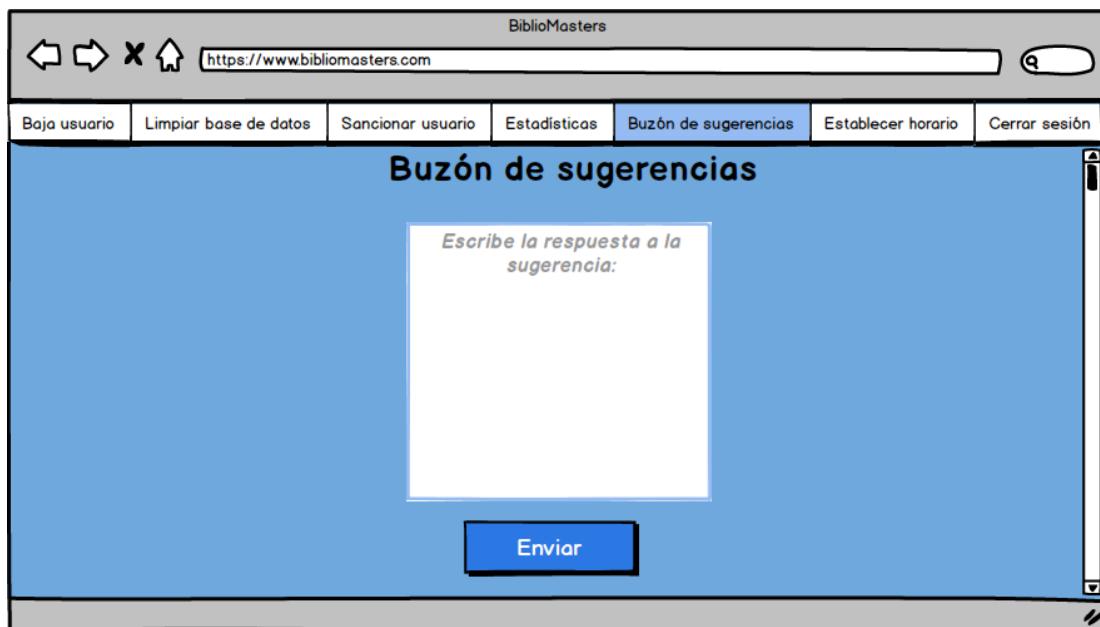


Figura 7: Buzón de sugerencias - Contestar mensaje

2.2. Funcionalidad aplicación móvil

Las funciones que podrán realizar los usuarios serán ocupar/desocupar un sitio dentro de la biblioteca, visualizar el mapa donde podrán observar los sitios libres y ocupados, además de ver información de la biblioteca como el horario de apertura y cierre, enviar mensajes a los bibliotecarios a partir de un buzón de sugerencias, ver sus estadísticas personales y globales de tiempo de estudio y ocupación y por último revisar su información personal (email,cambiar su contraseña,etc.).

A continuación, en la Figura 8 se incluye el diagrama de casos de uso en el que se pueden observar todas las funcionalidades que tendrá el usuario de la aplicación móvil Android.

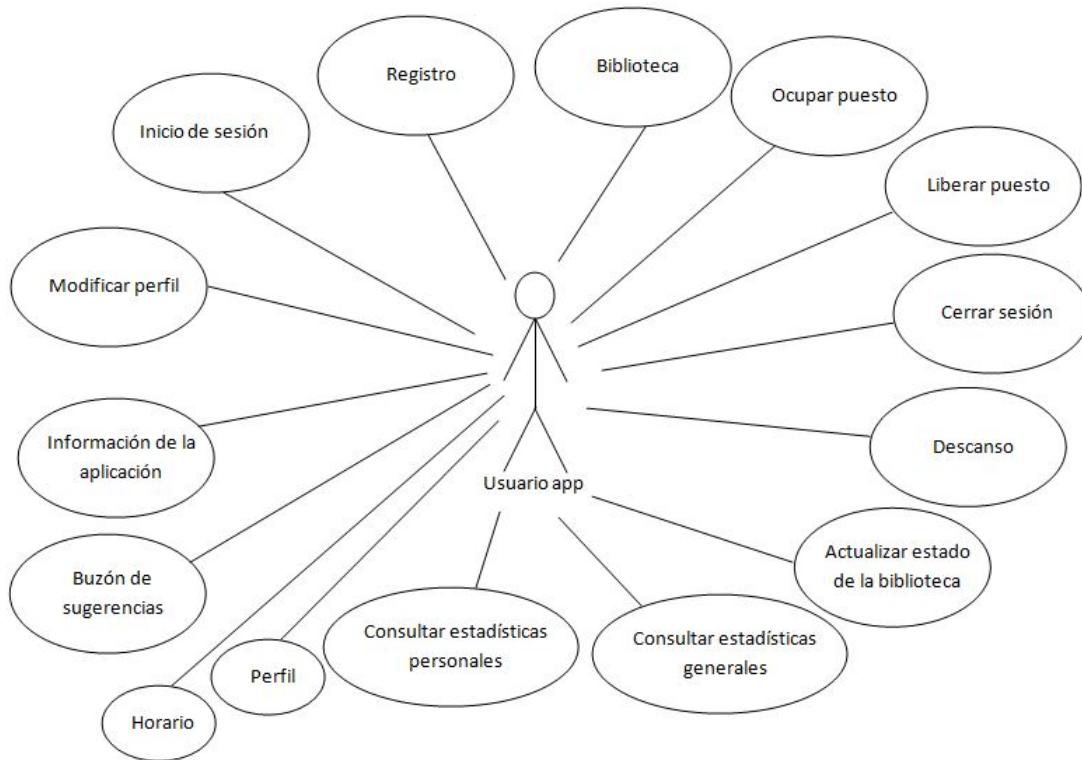


Figura 8: Casos de uso aplicación móvil

Funcionalidad 1: Inicio de sesión

Esta será la primera ventana que aparecerá al usuario al abrir la aplicación, en caso de que sea la primera vez que entre en la app. En ella se podrá conectar al sistema, si ya está registrado, con su email y su contraseña. Si los datos suministrados no coinciden con los almacenados en la base de datos, el usuario será notificado. En caso de no estar registrado, podrá pulsar en un enlace que le llevará a la ventana de registro.



Figura 9: Iniciar sesión aplicación móvil

Funcionalidad 2: Registro

En el caso de que no se encuentre registrado en la aplicación, llegará a esta pantalla a través del acceso que aparece en la pantalla de inicio de sesión. En la misma, el usuario proporcionará su correo electrónico y la contraseña que quiera para registrarse. Si las contraseñas no coinciden o el email ya está registrado en la base de datos, el registro no se completará y se informará al usuario del error acontecido. Una vez se ha finalizado el registro, un email será enviado a la cuenta de correo electrónico para confirmar el alta y que el usuario tiene acceso a ese correo.



Figura 10: Registrarse aplicación móvil

Funcionalidad 3: Biblioteca

Esta será la pantalla principal que aparecerá al usuario cada vez que abra la aplicación. Con esta funcionalidad, el usuario puede ver gráficamente cuáles son los puestos que están ocupados y libres en cada mesa de cada planta en tiempo real. La aplicación se conectará a la base de datos e irá recorriendo todos los asientos de la biblioteca para así ir pintando si el puesto se encuentra libre o no. Si el puesto se encuentra libre, y el usuario no ha ocupado ningún puesto previamente, podrá pulsar sobre él y será dirigido a la función de ocupar puesto.

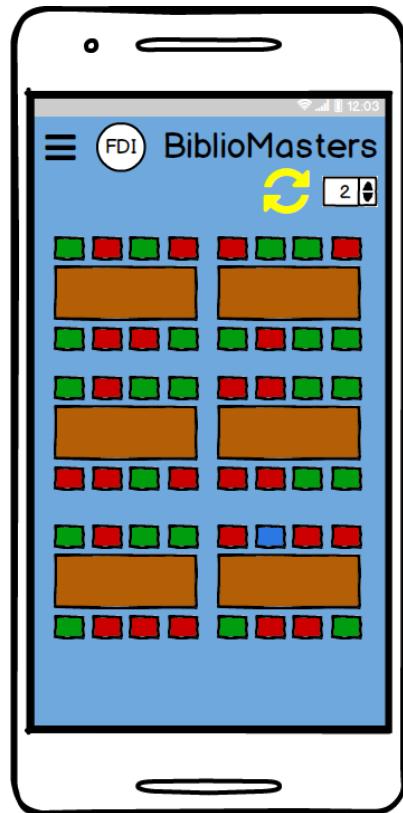


Figura 11: Biblioteca

Funcionalidad 4: Ocupar puesto

A esta funcionalidad se llegará tras haber marcado un puesto libre en el plano de la biblioteca. Una vez en ella, aparecerá un mensaje para confirmar realmente que el usuario quiere ese puesto. Tras aceptarlo, la aplicación confirmará con la base de datos que las coordenadas del usuario están dentro del perímetro de la biblioteca y procederá a ocupar el sitio. Dicho sitio se ocupará en la base de datos y se mostrará al usuario final de un color diferente al del resto de puestos ocupados. También, el temporizador con el que se controlará el tiempo que puede descansar empezará a sumar minutos hasta un máximo de 30 y el de ocupar puesto se restablecerá.



Figura 12: Ocupar puesto

Funcionalidad 5: Liberar puesto

Esta será una de las opciones que aparecerá en el menú, siempre que un puesto haya sido ocupado previamente. Tras pulsar sobre ella, se le preguntará al usuario si realmente lo quiere liberar. Si decide liberar su puesto, su estado pasará a disponible en la base de datos y se cambiará su color en la app. Además, los dos temporizadores se resetearán.



Figura 13: Liberar puesto

Funcionalidad 6: Descanso

Otra de las opciones que estarán disponible si un puesto ha sido ocupado previamente será descanso. Tras ser pulsada por el usuario, se le preguntará si realmente quiere iniciar el descanso. Si confirma, el temporizador comenzará a restar minutos del tiempo que tenga disponible. Una vez vuelva a ocupar el puesto, el temporizador comenzará a sumar minutos desde donde lo dejó.

Además, esta misma opción pasará a ser “fin descanso” cuando el usuario haya iniciado previamente su descanso. Cuando el usuario de por finalizado su descanso, solamente tendrá que pulsar sobre dicha opción y la aplicación comprobará que ya se encuentra en la biblioteca.



Figura 14: Descanso

Funcionalidad 7: Consultar estadísticas generales

La aplicación recogerá datos de todos los usuarios que la utilicen y los guardará en la base de datos para generar a partir de ellos unas estadísticas que todos podrán consultar. Estas estadísticas mostrarán la media de los datos recogido sobre información relevante como el tiempo que los estudiantes pasan en la biblioteca, la frecuencia de descansos, cuál es la duración de estos, cuántos días por semana acuden a estudiar, etcétera.

Estos datos serán accesibles por los estudiantes a través de la app, mediante un botón ubicado en el menú, y por los bibliotecarios a través de la aplicación web.

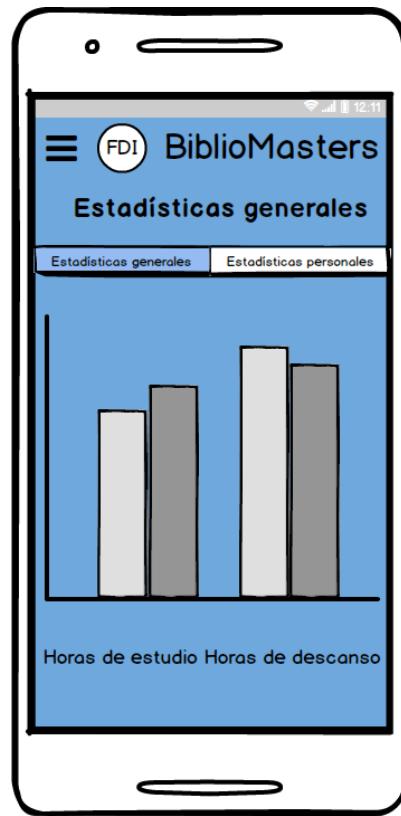


Figura 15: Estadísticas generales

Funcionalidad 8: Consultar estadísticas personales

A partir de los mismos datos que almacenamos para mostrar las estadísticas generales, se generarán estadísticas particulares de cada usuario, que serán visibles únicamente por él a través de la app mediante el mismo botón que muestra las generales, ofreciéndole la posibilidad de decidir cuál desea ver. Así, podrá comparar sus estadísticas personales con las generales.

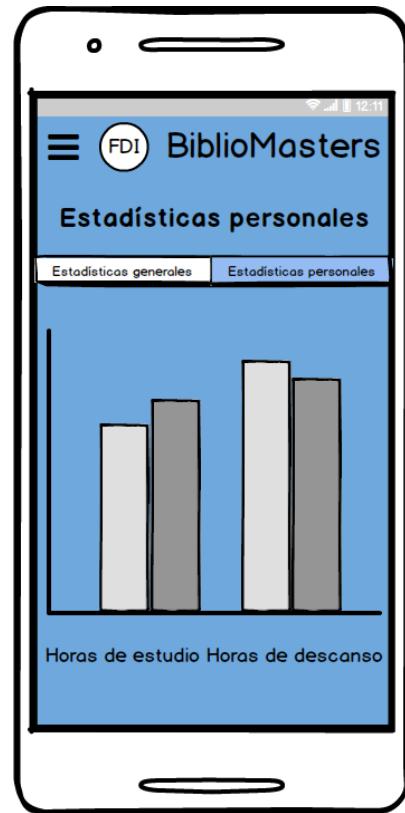


Figura 16: Estadísticas personales

Funcionalidad 9: Modificar contraseña

Desde la misma, el usuario podrá modificar la contraseña tras introducir su contraseña antigua e introducir la nueva contraseña dos veces para garantizar que no se ha equivocado. Una vez hecho, pulsará el botón de aceptar y la app se comunicará con la base de datos para modificar la contraseña del usuario almacenada.



Figura 17: Modificar contraseña

Funcionalidad 10: Horario

Los usuarios podrán consultar el horario de la biblioteca, así como información relevante, a través de una opción del menú. Este horario será asignado por el bibliotecario a través de la aplicación web.



Figura 18: Horario

Funcionalidad 11: Buzón de sugerencias

Esta opción tendrá una doble funcionalidad. Al ser pulsada aparecerán dos opciones. La primera de ellas será escribir sugerencia. Se tratará de sugerencias que el bibliotecario leerá de forma anónima y podrá responder también sin conocer quién fue el que la escribió.

Es aquí cuando entra en escena la otra funcionalidad, llamada leer respuestas. En el caso de que un bibliotecario conteste a una de las sugerencias enviadas por un usuario determinado, éste podrá leer su respuesta desde esta opción.

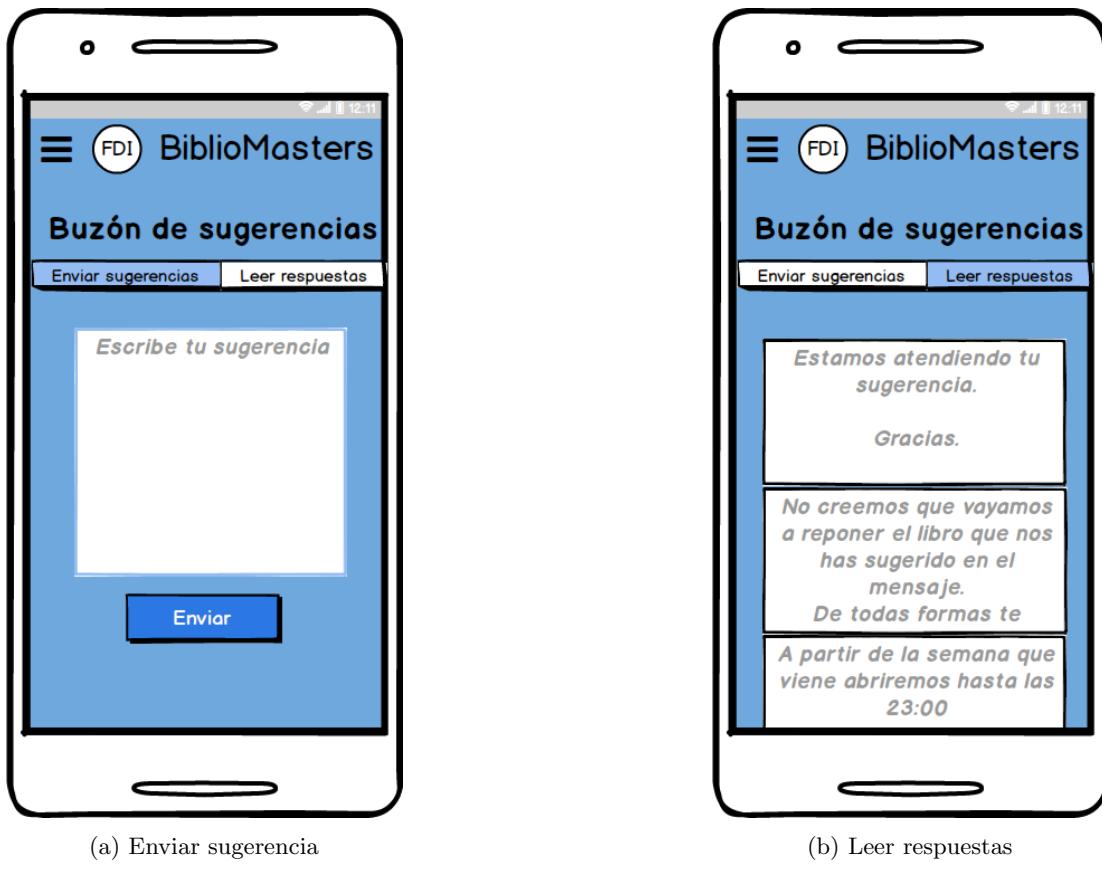


Figura 19: Buzón de sugerencias

Funcionalidad 12: Información aplicación

Estará disponible desde el menú, información sobre la aplicación, como su versión, datos sobre sus desarrolladores o una opción para ponerse en contacto con ellos.



Figura 20: Información de la aplicación

Funcionalidad 13: Comprobar disponibilidad puesto

Una vez el usuario haya seleccionado el puesto que desea ocupar, la app se comunicará con la base de datos para garantizar que el puesto que quiere ocupar está disponible y pasará a ser de otro color diferente al resto de puestos ocupados. En caso contrario, se notificará al usuario que ha habido un error y podrá volver a seleccionar puesto.



Figura 21: Comprobar disponibilidad de puesto

Funcionalidad 14: Temporizador ocupar puesto

La aplicación debe encargarse de comprobar que un usuario no se encuentre en la biblioteca ocupando un puesto de estudio sin haberlo notificado. Para ello, cuando detecte que el usuario ha entrado en la biblioteca, comenzará a incrementar un temporizador.

En el caso de que este llegue a 20 minutos y el estudiante aún no haya ocupado un puesto, se le mostrará un mensaje en el que se le sugerirá que por favor, ocupe un puesto de estudio si va a permanecer en la biblioteca durante más tiempo.

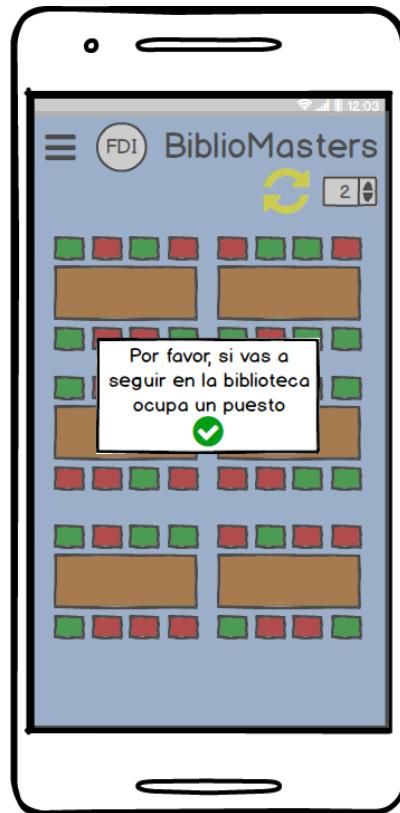


Figura 22: Mensaje: 'Por favor ocupa puesto'

Funcionalidad 15: Preguntar por descanso

Si el usuario decide salir de la biblioteca sin haber anunciado que iba a realizar un descanso, automáticamente se pondrá en modo descanso siempre que le quedan minutos de descanso disponibles.

Además, se mostrará un mensaje al usuario que le preguntará si ha abandonado la biblioteca o si por el contrario se está tomando un descanso. De no contestar el usuario en un plazo de 5 minutos, se liberará su puesto y quedará disponible para el resto de estudiantes.



Figura 23: Mensaje: 'Preguntar por descanso'

Funcionalidad 16: Temporizador descanso

Una vez que el usuario ha decidido tomarse un descanso, el temporizador de tiempo disponible (tiempo que el usuario había ido acumulando mientras estudiaba) comienza a decrementar. Cuando nos encontramos a cinco minutos de que llegue a 0, un mensaje es mostrado al usuario avisándole de que sólo quedan 5 minutos para que le dé tiempo a regresar a la biblioteca para estudiar.



Figura 24: Mensaje: 'Aviso de fin de descanso'

Funcionalidad 17: Envío de localización

La aplicación necesitará saber la ubicación del usuario en ciertas situaciones para comprobar que este no está tratando de introducir datos que no son reales. Cada cierto intervalo de tiempo, que será establecido por los desarrolladores, la aplicación enviará al servidor la ubicación exacta del usuario, utilizando para ello el sistema de geolocalización integrado en todos los smartphones.

La aplicación hará uso de estos datos para varias funcionalidades, como saber si el usuario se encuentra dentro o fuera de la biblioteca, incrementar o decrementar el temporizador de tiempo de estudio, avisar al estudiante si ha terminado su descanso y debe volver a entrar en la biblioteca, etcétera.

Funcionalidad 18: Temporizador estudio

Tras haber iniciado el estudio el usuario, se pondrá en marcha un temporizador de estudio que se incrementará en un minuto por cada minuto que el usuario pase estudiando hasta un máximo de 30 minutos. Este tiempo será del que podrá disponer el usuario para tomarse descansos. Una vez vuelva de descansar, el temporizador volverá a incrementarse desde los minutos que ha dejado sin usar el usuario, hasta un máximo total de 30.

Funcionalidad 19: Comprobar situación dentro/fuera biblioteca

La base de datos tendrá almacenada la ubicación exacta de la biblioteca en la que se encuentra el estudiante, de modo que cuando se envíe la localización, la app podrá saber si se encuentra dentro de la misma o ha salido fuera del perímetro establecido. Esto permitirá comprobar que la posición del estudiante coincide con el estado actual que ha establecido en la app, los cuales pueden ser estudiando, descansando o fuera de la biblioteca.

De esta forma, se evita que el usuario falsifique su posición. Por ejemplo, si en la app aparece como estudiando pero se detecta que se encuentra fuera de la biblioteca, se le preguntará si está realizando un descanso, como se explica en el apartado ‘Preguntar por descanso’.

Funcionalidad 21: Aviso fin de descanso

Se mostrarán diferentes mensajes al usuario que le avisaran de que le quedan tan solo 5 minutos para que finalice su descanso, así como otro que avisará de que su descanso ha finalizado.

Funcionalidad 22: Cerrar sesión

Si el usuario desea cerrar sesión en la aplicación móvil, podrá hacerlo. Esto conllevará eliminar la sesión activa de la base de datos interna del móvil y parar de comprobar la ubicación del usuario.

2.3. Fiabilidad

La aplicación Android está desarrollada en la versión 7 de Android, permitiendo que pueda funcionar para versiones anteriores, siendo esta versión donde se han realizados las pruebas durante el proyecto.

La gestión de errores de la aplicación está pensada para informar al usuario en todo momento de lo que ocurre en la aplicación, con mensajes claros, especificando los fallos que se han producido. Algunos de ellos son referidos al usuario y otros a la comunicación con el servidor, temporizadores o de conexión.

Uno de los puntos fundamentales para el limpiado de datos en la base de datos, es la función automática que se realiza en la web, la cual agrupa las estadísticas de los usuarios de tiempo de estudio, usuarios inactivos de hace más de un año o mensajes del buzón de sugerencias, para así poder reciclar espacio en la base de datos. De esta manera se consigue reducir el número de problemas que pueden surgir al tratar con tablas de gran tamaño.

3. Tecnologías empleadas

En este capítulo se van a explicar las herramientas, servicios, bases de datos que se han usado y sus características. Además, de igual modo que en la sección anterior, se va a realizar una división en tres secciones. En la primera de ellas se explicarán las tecnologías utilizadas en la página web, en la segunda las usadas en la base de datos y en la tercera las usadas en la parte móvil.

3.1. Tecnologías en la aplicación web

La página web ha sido desarrollada en HTML5, CSS, PHP, JavaScript... Además, se ha hecho uso de lenguajes como MySQL y se han planificado y lanzado algunos trabajos CRON. Además de la utilización de algunas librerías y Frameworks, entre ellas Bootstrap, que permitió desarrollar un diseño responsive, muy útil para ajustar la visualización de la web a distintos dispositivos.

A continuación haremos un pequeño resumen de cada tecnología, y como la hemos utilizado en nuestro proyecto.

HTML5

HTML5 es el lenguaje por excelencia para la creación de páginas webs. Además, se trata de un lenguaje que ofrece prácticamente infinitas posibilidades y que da una gran libertad a la hora de diseñar los diferentes elementos de la web. También, el hecho de que cuente con diferentes APIs como la API de Geolocalización u otras muchas y la prácticamente perfecta integración con otros lenguajes como PHP, CSS o JavaScript fueron puntos positivos a la hora de decidir realizar el proyecto con este lenguaje.

Pero sin duda, uno de los puntos fuertes de este lenguaje es su gran compatibilidad con casi todos los navegadores y dispositivos del mercado, haciendo que a la hora de desarrollar sobre él, los problemas se reduzcan muchísimo y se llegue a una cantidad de público muy elevada.

CSS

Se trata de un lenguaje que permite personalizar prácticamente cualquier elemento de una página web, desde la fuente hasta los botones pasando por los encabezados, diseño de las imágenes, sus bordes y otra gran cantidad de opciones que resultaría imposible enumerar aquí todas.

Además, la realización del proyecto se apoyó en el proyecto de código abierto BootStrap que ofrece a los usuarios herramientas para el diseño de sitios web responsive de forma sencilla y con un acabado muy bueno. Es uno de los proyectos más destacados en GitHub y es usado en una gran cantidad de páginas webs.

PHP

PHP es el lenguaje utilizado para la comunicación con el servidor, con la base de datos y la recogida de la información aportada por los bibliotecarios. Con este lenguaje se obtienen los datos introducidos en los diversos formularios (sugerencias u horario por ejemplo), se procesan, se garantiza que la información introducida es correcta y, por último, se inserta en la base de datos.

Es el lenguaje con el que se ha implementado toda la lógica de la página: la página inicial en la que se produce el logueo de los bibliotecarios, las diferentes pestañas que muestran todas las funcionalidades...

Gracias a este lenguaje se ha realizado la conexión con el servidor externo, además de poder realizar consultas.

JavaScript

Otro de los lenguajes que se han utilizado es JavaScript. Este lenguaje ha permitido incorporar un estilo mucho más visual a la página web, permitiendo además añadir diversas funcionalidades extra.

Una de estas funcionalidades es el hecho de mostrar un mapa con la ubicación de la biblioteca. Mediante la integración de la API de Google Maps, basada en JavaScript se ha conseguido dicho mapa.

Otra de las funcionalidades de la página web implementadas con Javascript es el hecho de que el gráfico que muestra las estadísticas sea completamente personalizable en el rango en el que el usuario quiera verlo. El mismo está implementado mediante JavaScript y algunas librerías desarrolladas en este lenguaje, lo que proporciona un estilo muy intuitivo y amigable.

jQuery

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM y manejar eventos entre otras cosas.

Ha sido utilizada para modificar la estructura del árbol DOM en aquellas situaciones en las que realizarlo mediante Javascript suponía una elevada dificultad. Ha simplificado en gran medida el tratamiento los eventos que realizaban grandes cambios en el código HTML5, como por ejemplo el hecho de añadir nuevas plantas a la biblioteca en el apartado Horario.

CRON Jobs

Mediante los trabajos de CRON se ha conseguido mantener la base de datos en un tamaño razonable de forma automática, reduciendo la carga de trabajo y liberando de preocupaciones al bibliotecario. Los CRON Jobs son una herramienta de trabajo muy potente que permiten al usuario automatizar prácticamente cualquier tipo de tarea.

3.2. Tecnologías en la aplicación móvil

Para la implementación de esta parte se ha hecho uso del que es el IDE más usado para el desarrollo de aplicaciones Android, Android Studio. Mediante Android Studio y principalmente Java y XML se ha logrado realizar gran parte de la interfaz gráfica de la aplicación y una gran parte de la lógica también, sin olvidar el uso de Material Design (como se puede apreciar en las carpetas de 'res') para la interfaz de usuario, dando además algún otro beneficio como poder ajustar las imágenes a diferentes tipos de resolución de las pantallas de los diferentes móviles. Para las partes de la lógica que no se quedaban a nivel interno en la aplicación, si no que necesitaban comunicarse con el servidor se ha hecho uso nuevamente de PHP. Además, se ha usado JSON para transferir los resultados de los diversos PHP a la aplicación móvil. Para las consultas con la base de datos, se ha recurrido a MySQL y, también aquí, se han hecho uso de los CRON Jobs.

Android Studio

Android Studio ofrece una gran cantidad de posibilidades a la hora de desarrollar cualquier tipo de aplicación para este sistema operativo móvil y, lo que es casi mejor, cuenta con una gran cantidad de usuarios detrás dispuestos a ayudar y ofrecer soporte ante cualquier problema que pueda surgir programando. Permite implementar prácticamente cualquier funcionalidad y conseguir una aplicación perfectamente optimizada para la amplia gama de dispositivos Android existentes.

Java

Java se trata de un lenguaje de programación orientada a objetos. Es uno de los lenguajes más utilizados para el desarrollo de aplicaciones nativas en Android y es el lenguaje recomendado para programar en este sistema operativo.

Las bondades de Java son múltiples y conocidas (gran cantidad de librerías, soporte de muchas funcionalidades de forma nativa) pero las más relevantes para este proyecto han sido el hecho de su perfecta integración con otros lenguajes como XML o PHP para poder realizar todas las consultas con el servidor.

XML

XML es un lenguaje basado en etiquetas que permite organizar la información en un formato ajustado a las necesidades de cada usuario. Para este proyecto fue de vital importancia ya que ofreció muchas opciones para diseñar la interfaz gráfica de la aplicación y colocar prácticamente cada elemento al milímetro donde fuera necesario. [5]

PHP

Se trata de una tecnología ya explicada en la subsección anterior, por lo que no se aporta información repetitiva sobre ella.

JSON

JSON se trata de un formato ligero para el intercambio de datos. Entre sus ventajas destacan que permite intercambiar datos de forma muy rápida, con un formato simple y que los archivos que produce son de poco tamaño. Sin embargo, puede resultar un poco difícil de interpretar a simple vista. En este proyecto ha resultado de gran utilidad por su gran capacidad para transferir datos rápidamente entre PHP y las clases Java.[5]

MySQL

Se trata del lenguaje por excelencia para tratar con bases de datos. Es el lenguaje utilizado para tratar con la base de datos y es por tanto el lenguaje a utilizar cuando se produce la preparación de las queries, dentro de los PHP, que se van a realizar sobre la base de datos. Ofrece una gran capacidad de respuesta, un alto rendimiento y ejecutar prácticamente cualquier tipo de consulta. Además, cuenta con algunas funciones ya implementadas que facilitan el filtrado de datos en las queries.[6]

CRON Jobs

Se trata de una tecnología ya explicada en la subsección anterior, por lo que no se aporta información repetitiva sobre ella.

Servidor externo

Se trata de un servidor externo en el que se alojan los datos de usuarios, mensajes, puestos de la biblioteca... El servidor elegido ofrece diversas ventajas como son la posibilidad de crear trabajos de Cron, el almacenamiento en disco SSD que aumenta la velocidad y el rendimiento del sistema. Además incluye la posibilidad de reescalado de los servicios contratados a medida que puedan aumentar las necesidades del proyecto.

3.3. Bases de datos

A la hora de estructurar la información, organizarla y almacenarla se ha optado por una base de datos SQL, alojada en un servidor en la nube y una base de datos SQLite interna en cada móvil del usuario.

En la base de datos en MySQL se encuentra almacenada toda la información general de la aplicación. Esto engloba el histórico de todas las conexiones, los usuarios registrados, las diferentes bibliotecas implementadas, los mensajes de todos los usuarios... El acceso a la misma se realiza a través de PHPMyAdmin.

Por otra parte, en la base de datos SQLite se almacenan datos que se obtienen del servidor una vez se inicia la aplicación. De esta manera se evita tener que hacer múltiples llamadas al servidor, logrando así un gran ahorro, tanto de batería como de datos móviles, además de una importante mejora en la rapidez de la aplicación, reduciendo la latencia al mínimo nivel. Aunque posteriormente se explicarán todos los campos que se almacenan en la base de datos SQLite algunos de ellos son nombres de las plantas, nombre de usuario o contraseña entre otros.

Además, una de las principales ventajas que ofrece el hecho de hacer esta base de datos SQLite es que, una vez logueado el usuario por primera vez en la aplicación, dicho usuario y contraseña quedan almacenados. De esta forma, la próxima vez que se inicia la aplicación no es necesario que el estudiante introduzca de nuevo el usuario y la contraseña si no que carga automáticamente la pantalla de selección de biblioteca.

4. Arquitectura de la aplicación

Para implementar este proyecto se ha utilizado un patrón MVC, (Modelo-Vista-Controlador). Se trata de un patrón utilizado en el desarrollo de software que separa la interacción del usuario en la interfaz, de la lógica de los datos, y la utilización de un controlador para gestionar las peticiones de los eventos.

- **Vista:** está representada en la aplicación web, mediante una interfaz de usuario. El usuario interactúa con dicha interfaz desarrollada (completando cuadros de texto, pulsando botones...). Para desarrollar la aplicación web se usará HTML5 y CSS3. Lo mismo pasará para la aplicación móvil, pero la interfaz se ha desarrollado mediante recursos de XML.
- **Controlador:** una vez el usuario haya interactuado con la interfaz, gestionará el evento que llega. Por ejemplo, tanto en la aplicación web o móvil al pulsar un botón, el controlador gestionará mediante un gestor dicho evento. Para desarrollar el controlador en web, se ha hecho mediante PHP y Javascript, en móvil mediante java.
- **Modelo:** después de que el controlador delegue el evento, se actualizan los datos en el modelo. Al igual que el controlador, se ha realizado mediante PHP y Javascript, para web y java y PHP para móvil. Es en este módulo donde se realiza la lógica como la actualización de la base de datos o las diferentes acciones de la geolocalización.

A continuación, en la Figura 25 se observa un esquema de cómo ha sido implementada la arquitectura del proyecto, y de cómo se relacionan los diferentes módulos y tecnologías empleadas:

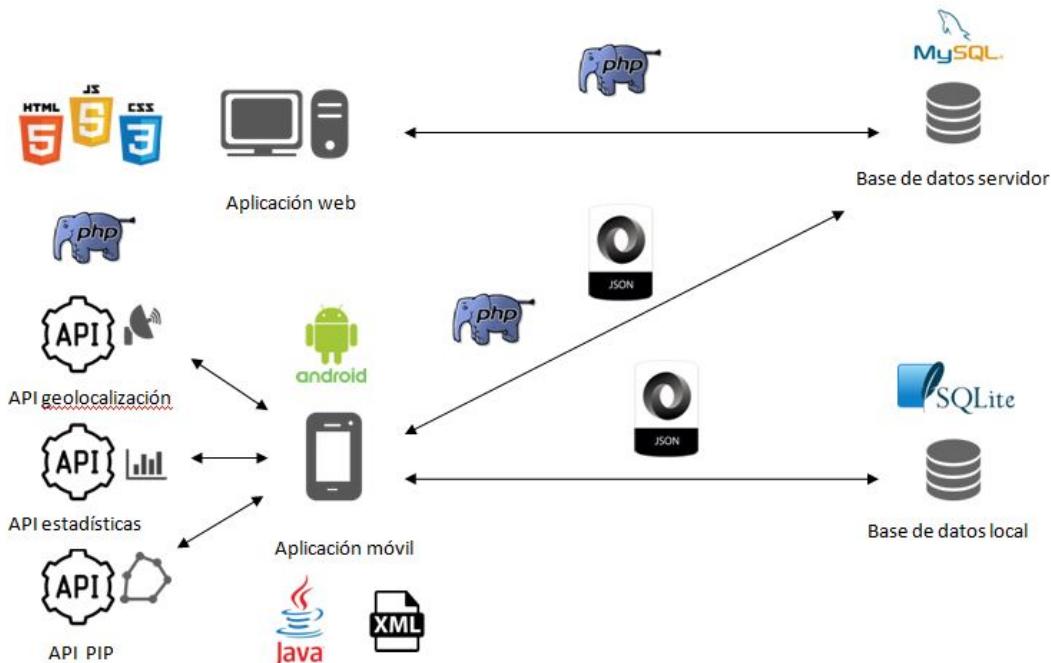


Figura 25: Visión general de la arquitectura del proyecto

En cuanto a la estructura del proyecto se divide en dos bloques, en los cuáles se han empleado diversas tecnologías.

Por un lado, un servicio web, utilizado para los bibliotecarios/as, y por otro una aplicación Android, que será utilizada para los estudiantes que acudan a la biblioteca a reservar un puesto de estudio.

También se ha utilizado una base de datos relacional, para almacenar toda la información, y una base de datos interna, para guardar la información referida al usuario en concreto en el móvil.

Para guardar los datos de la base de datos relacional, se ha utilizado un servidor externo.

Además se ha contado con la ayuda de un Service, una aplicación en segundo plano que se inicia para conseguir la ubicación del móvil cada cierto periodo de tiempo. Para la ubicación también fue necesaria una API de geolocalización que sirve para conseguir el mapeado geoespacial a tiempo real.

También han utilizado algunas otras APIs, para poder conseguir estadísticas y otra en especial, llamada API PIP, utilizada para poder determinar si la ubicación del usuario está dentro del perímetro de la biblioteca. Esta última API es muy útil, puesto que nos permite determinar si el usuario se encuentra dentro o fuera de la biblioteca y actuar en consecuencia.

4.1. Aplicación web

Uno de los bloques del proyecto, se basa en la ayuda a los bibliotecarios/as en la realización de sus tareas diarias en la facultad. Realizamos entrevistas con ellos para poder determinar que, a pesar de estar en una facultad informatizada, algunas de las funciones que hacían, las realizaban sin el uso de ninguna página especializada. Como ejemplo claro tenemos los horarios de apertura y cierre de la biblioteca. Éstos se escriben a mano por los bibliotecarios y se ponen en la entrada, pero solo los técnicos de la página web pueden configurar los horarios.

Es por ello que una de las funciones en la página web es la de configurar ellos mismos el horario de apertura y cierre de la biblioteca de una manera fácil y sencilla.

Tampoco hay una línea de contacto directa con los bibliotecarios electrónicamente. Por ello se ha creado un buzón para que los bibliotecarios puedan responder directamente las sugerencias que los alumnos envían desde la aplicación.

Otra función muy importante que está incorporado en la web, son las estadísticas (en base a las estadísticas de estudio de los usuarios) muy útil para poder planificar el horario de apertura, cierre, número de sitios...

4.2. Aplicación móvil

Es la parte del proyecto utilizada por toda persona que quiera ocupar un sitio en la biblioteca, permitiendo así la gestión de los puestos de estudio de la biblioteca. Permitiendo conocer en tiempo real qué puestos están libres y cuáles se encuentran ocupados, además de poder ocupar un puesto en caso de encontrarnos en la biblioteca. De este modo y una vez ocupado el puesto, podremos realizar un descanso de una duración determinada en función del tiempo que se haya estado estudiando sin necesidad de dejar ningún elemento en la mesa. Sin embargo, no solo tiene esta única funcionalidad.

Además, se pueden consultar las estadísticas personales y globales de tiempo de estudio y descanso, para así poder hacer un balance de estudio personal. También permite enviar y recibir mensajes a los bibliotecarios, pudiendo enviar sugerencias y dudas y permitiendo un hilo de comunicación entre bibliotecarios y estudiantes.

Como añadido, presenta otras funciones como ver horarios, siempre que éstos hayan sido configurados por los bibliotecarios, tal y como se describió en el apartado anterior.

4.3. Bases de datos

Las funciones que tienen las bases de datos son la de almacenar la información general (en el caso de la base de datos relacional), y del propio usuario (la del SQLite).

La base de datos relacional, la cual utiliza MySQL, guarda datos de todos los usuarios, mensajes, horarios, puestos y plantas de las bibliotecas dadas de alta en la base de datos, coordenadas de cada una de ellas... Además, almacena un histórico con todas las conexiones de todos los usuarios a las diferentes bibliotecas, para así poder mostrarle a cada uno sus estadísticas de estudio.

Por otro lado, la base de datos local de cada teléfono móvil tan solo se encarga de guardar los datos del usuario, así como el puesto ocupado actualmente, las coordenadas actuales del usuario, estado, tiempo que tiene para descansar...

4.4. Servidor externo

El servidor externo es utilizado para guardar toda la información de la base de datos, además de alojar todos los archivos necesarios para el correcto funcionamiento tanto de la página web como de la aplicación.

Se ha optado por un servidor de pago, con 6GB de almacenamiento SSD que nos garantiza un rápido acceso a todos los archivos, con una latencia extremadamente baja y 1GB de RAM. Considerando que estas características eran suficientes para este determinado proyecto.

4.5. APIs

La API de geolocalización es necesaria para conseguir las coordenadas geoespaciales del usuario móvil, una vez este haya iniciado sesión.

Para poder obtener la ubicación del usuario cada cierto período de tiempo se ha usado una aplicación en segundo plano, un Service, que está consultando con la API anterior, la de geolocalización para así poder conocer la latitud y longitud del usuario. En el capítulo de implementación se desarrollará por qué fue necesario conocer la ubicación de cada usuario cada cierto tiempo.

Para las estadísticas, tanto del cliente web como del cliente móvil se ha utilizado una determinada API, con varias funciones en JavaScript que permiten visualizar los gráficos interactivos que tenemos implementados.

Y en cuanto a la API PIP, punto en polígono, es utilizada para, una vez suministrado el polígono formado por la biblioteca, determinar así si las coordenadas geográficas que obtenemos del usuario están dentro del recinto de esta, o fuera de ella.

5. Modelo de datos

En este capítulo se detalla la base de datos empleada y se describe la funcionalidad y significado de cada tabla y campo. En primer lugar se realiza la explicación de la base de datos ubicada en el servidor para, posteriormente, explicar la que se aloja localmente en el dispositivo móvil de cada usuario. Finalmente, se concluye con una sección en la que se relacionan los dos modelos, los cuales son ambos relacionales.

Se ha optado por la implementación de un modelo de datos relacional para tratar de garantizar la persistencia de los datos y hacer uso de su gran rendimiento y estabilidad, logrando así un ahorro tanto en batería como en datos.

5.1. Base de datos MySQL

La base de datos MySQL, la cual ofrece un gran rendimiento y es estable, es la base de datos en la que concentraremos toda la información de la aplicación. Esto incluye todos los usuarios registrados, el histórico de los mismos o todos los horarios de las bibliotecas implementadas entre otras cosas. A ella se accede en múltiples ocasiones, cuando se quiere comprobar la disponibilidad de puestos, ver las bibliotecas disponibles, los mensajes intercambiados con los bibliotecarios, las estadísticas...

Se trata de la base de datos maestra y es la que prevalece en caso de conflicto con la base de datos local del móvil de cada usuario. Sobre ella se produce la concentración de todos los datos provenientes de los diferentes usuarios registrados en la aplicación. Se trata de una base de datos MySQL.

Consta de 9 tablas relacionadas entre sí mediante claves foráneas y atributos: bibliotecas, envia, historico, horarios, ocupa, puestos, usuarios, usuariosActivos y usuariosTemp. Cada una de ellas guarda información específica acerca de diferentes aspectos necesarios para el funcionamiento de la app.

A continuación, en la Figura 26 se muestra el diagrama entidad-relación, para poder resumir de una manera visual y gráfica como está dispuesta la base de datos y qué relaciones hay entre las diferentes tablas.

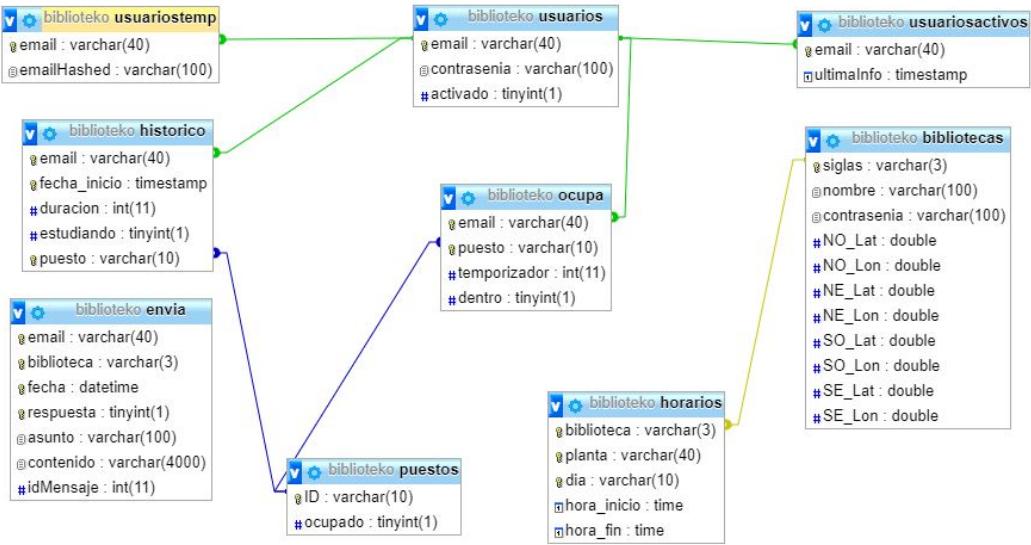


Figura 26: Diagrama entidad-relación de la base de datos MySQL

Bibliotecas

Se trata de una tabla en la cual se almacenan las diferentes bibliotecas que están implementadas. Aunque en la actualidad tan solo existe una fila con la biblioteca de la facultad de informática, el propósito es que la aplicación crezca e ir implementando más bibliotecas, para lo cual es necesario disponer de esta tabla.

Es una tabla a la cual solo se puede acceder mediante PHPMyAdmin, de forma que únicamente pueden acceder a la misma los 3 miembros del equipo de desarrollo. De esta forma, cuando se produzca la implementación de una nueva biblioteca, los desarrolladores serán los encargados de insertar todos los campos de la misma.

Consta de once campos, los cuales son los siguientes:

- **siglas**: Es la clave primaria de la tabla, se trata de una cadena de 3 caracteres mayúsculas que sirven para identificar a la biblioteca.
- **nombre**: Es un campo varchar con una longitud máxima de 100 caracteres. En este campo es donde se almacena el nombre completo de la biblioteca.
- **contrasenia**: De igual manera que en el campo anterior, se trata de un campo varchar con una longitud máxima de 100 caracteres. Aquí se almacena la contraseña hasheada necesaria, junto a las siglas, para acceder a la página de gestión de los bibliotecarios.
- **NO_Lat**: Se trata de un campo double en el cual se almacena la latitud del punto situado más al noroeste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **NO_Lon**: Se trata de un campo double en el cual se almacena la longitud del punto situado más al noroeste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.

- **NE_Lat:** Se trata de un campo double en el cual se almacena la latitud del punto situado más al noreste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **NE_Lon:** Se trata de un campo double en el cual se almacena la longitud del punto situado más al noreste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **SO_Lat:** Se trata de un campo double en el cual se almacena la latitud del punto situado más al suroeste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **SO_Lon:** Se trata de un campo double en el cual se almacena la longitud del punto situado más al suroeste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **SE_Lat:** Se trata de un campo double en el cual se almacena la latitud del punto situado más al sureste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.
- **SE_Lon:** Se trata de un campo double en el cual se almacena la longitud del punto situado más al sureste del recinto de la biblioteca. Sirve para determinar si nos encontramos dentro de la biblioteca o fuera.

Envia

En esta tabla, están recogidos los diferentes mensajes intercambiados entre los usuarios y la biblioteca correspondiente. A ella se accede cuando se produce el envío de un mensaje, tanto por parte del usuario como por parte del bibliotecario. En ese caso, se produce la inserción de una nueva fila en la tabla. Además, también se accede a ella, como bibliotecarios, cuando quieren leer los mensajes que les han mandado los usuarios, y, como usuarios, cuando quieren leer las respuestas que les han dado los bibliotecarios.

A pesar de que los mensajes almacenados en esta tabla constan de un campo email, de forma que sería posible saber quién es el usuario que ha escrito cada mensaje, al bibliotecario se le mostrarán sin dicho campo para garantizar el anonimato de todos los estudiantes.

Consta de siete campos, los cuales son los siguientes:

- **email:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo email de la tabla usuarios. Es un varchar de 40 caracteres que permite saber quién es el usuario que ha escrito el mensaje, para así poder mostrar la respuesta tan solo a dicho usuario, no a todos.
- **biblioteca:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo siglas de la tabla bibliotecas. Es un varchar de 3 caracteres que permite saber hacia qué biblioteca iba dirigido el mensaje, para así mostrar dicho mensaje a tan solo esa biblioteca.
- **fecha:** Se trata de un datetime que permite almacenar la fecha en la que fue escrito el mensaje.
- **respuesta:** Forma parte de la clave primaria, se trata de un tinyint, también conocido como booleano que permite saber si el mensaje se trata de un mensaje enviado por el usuario o de una respuesta del bibliotecario. En el caso de que se trate de un mensaje

enviado por el usuario estará a 0, en el caso de que sea respuesta del bibliotecario, estará a 1.

- **asunto:** Se trata de un campo varchar de longitud 100 caracteres en el cual se almacena el asunto del mensaje.
- **contenido:** Se trata de un campo varchar de longitud 4000 caracteres en el cual se almacena el contenido del mensaje, es decir, el mensaje propiamente.
- **idMensaje:** Forma parte de la clave primaria, se trata de un entero que se va autoincrementando a medida que van llegando nuevos correos de los estudiantes. Cuando un bibliotecario responda a uno de estos mensajes, se introducirá en la tabla envia con el mismo idMensaje pero con el campo respuesta a 1.

Historico

Esta tabla es la que se usa para almacenar y mantener un histórico de todas las conexiones de los diferentes usuarios a todas las bibliotecas implementadas en la aplicación. Es muy útil puesto que sin ella luego no sería posible mostrar las estadísticas, ni al usuario ni a los bibliotecarios, del tiempo que han estado estudiando o descansando los estudiantes.

Puesto que se produce una nueva inserción cada vez que un usuario pasa de estado estudiando a descansando y, por tanto, el número de inserciones por usuario y por día puede ser muy elevado, se ha realizado un CRON Job que, cada día de madrugada, agrupe las conexiones anteriores a un año en años, quedándose solo con las conexiones específicas del último año y agrupando el resto por años. Así, se consigue reducir el tamaño de la tabla historico considerablemente.

Consta de cinco campos, los cuales son los siguientes:

- **email:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo email de la tabla usuarios. Es un varchar de 40 caracteres que permite saber quién es el usuario que ha estado estudiando/descansando en ese momento, para así tener luego sus estadísticas personales que poder mostrar.
- **fecha_inicio:** Forma parte de la clave primaria. Se trata de un timestamp que permite almacenar cuándo comenzó el estado de ese usuario. Es decir, el momento en el que comenzó a estudiar o a descansar.
- **duracion:** Se trata de un entero en el cual se almacena la duración en segundos de ese determinado estado. Es decir, si un estudiante ha estado estudiando 10 minutos antes de iniciar su descanso, su entrada respectiva tendrá el campo duracion a 600.
- **estudiando:** Se trata de un tinyint, también conocido como booleano que permite saber si el estudiante estaba estudiando o descansando. En el caso de que estuviera estudiando estará a 1, mientras que si estaba descansando, estará a 0.
- **puesto:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo ID de la tabla puestos. Se trata de un varchar de longitud 10 que permite conocer en qué puesto estaba estudiando el usuario.

Horarios

En esta tabla se almacenan los horarios según cada día de la semana para las diferentes plantas de cada biblioteca implementada en la aplicación. El bibliotecario es el único que puede modificar los horarios de las diferentes plantas de la biblioteca a la que pertenece, según vayan a estar abiertas.

Es una tabla a la cuál se accede para modificarla desde el apartado establecer horario de la página web si se quieren modificar los horarios que hay establecidos o desde la aplicación móvil, como estudiantes, para así poder consultar el horario de la biblioteca en la que se encuentre.

Consta de cinco campos, los cuales son los siguientes:

- **biblioteca:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo siglas de la tabla bibliotecas. Es un varchar de 3 caracteres que permite saber a qué biblioteca hace referencia dicho horario.
- **planta:** Forma parte de la clave primaria, se trata de un varchar de longitud máxima 40 caracteres que permite saber el nombre de la planta a la cual hace referencia dicho horario. Esto está así implementado puesto que hay bibliotecas en las cuales cada planta puede cerrar a una determinada hora. El nombre de la planta es especificado por el bibliotecario cuando establece el horario.
- **dia:** Forma parte de la clave primaria, se trata de un varchar de longitud 10 en el cual se almacena el día de la semana al cual hace referencia dicho horario.
- **hora.inicio:** Se trata de un time en el cual se almacena la hora de apertura de la biblioteca para ese día. Por convenio, si está cerrada ese día se almacenará 11:11:11 y si está abierta 24 horas 23:59:59.
- **hora.fin:** Se trata de un time en el cual se almacena la hora de cierre de la biblioteca para ese día. Por convenio, si está cerrada ese día se almacenará 11:11:11 y si está abierta 24 horas 23:59:59.

Ocupa

Esta tabla contiene la ocupación actual de todas las bibliotecas que estén implementadas en la aplicación. Es muy útil puesto que es la tabla que se comprueba a la hora de realizar las estadísticas en las que se muestran a tiempo real el número de puestos que están ocupados

Además, se usa para garantizar la integridad entre la tabla puestos y la tabla ocupa, de forma que todos los puestos que aparezcan como ocupados en la tabla puestos deben tener su respectiva entrada en la tabla ocupa.

Consta de cuatro campos, los cuales son los siguientes:

- **email:** Forma parte de la clave primaria, se trata de una clave foránea que hace referencia al campo email de la tabla usuarios. Es un varchar de 40 caracteres que permite saber quién es el usuario que está ocupando ese determinado puesto. Es una información que es útil para así mostrar su propio puesto de otro color al usuario en el mapa de puestos de la biblioteca.
- **puesto:** Se trata de una clave foránea que hace referencia al campo ID de la tabla puestos. Se trata de un varchar de longitud 10 que permite conocer en qué puesto está estudiando el usuario. En combinación con el campo anterior permite saber qué usuario es el que está ocupando cada puesto.

- **temporizador:** Se trata de un entero en el cual se almacena el tiempo que lleva el estudiante en dicho puesto. Originalmente iba a servir para ver cuánto era el tiempo de descanso disponible de cada usuario pero finalmente no se hace uso del mismo puesto que la implementación del tiempo disponible se realiza desde la app para evitar conexiones con el servidor que gasten batería.
- **dentro:** Se trata de un tinyint, también conocido como booleano que permite saber si el estudiante si encuentra descansando o estudiando. Si está a 0 está descansando y si está a 1 estudiando.

Puestos

Esta tabla incluye todos los puestos de todas las bibliotecas implementadas en la aplicación. Además, existen puestos comodín del estilo XXXGlobal y TGlobalXXX que permiten agrupar todas las estadísticas de un determinado usuario en años pasados (para la problemática de que la tabla historico fuera demasiado grande).

Es una tabla en la cual ni los bibliotecarios ni los usuarios pueden hacer nuevas inserciones. Son los tres desarrolladores los responsables de incluir nuevos puestos en el caso en el que se incluya una nueva biblioteca en la aplicación. El acceso desde la aplicación a esta tabla solo se producirá en el caso de que un puesto pase a estar ocupado.

Consta de tan solo dos campos, los cuales son los siguientes:

- **ID:** Es la clave primaria, se trata de un varchar de longitud 10 caracteres con la forma SIG.PL_XXX que permite identificar cada puesto de cada biblioteca implementada. SIG son las siglas de la biblioteca a la que se refiere, PL el id de la planta de esa biblioteca y XXX el número de puesto de esa planta en concreto.
- **ocupado:** Se trata de un tinyint, también conocido como booleano que permite saber si el puesto ocupado o no. Si está a 0 está libre y si está a 1 está ocupado. Como se ha indicado antes, para cada puesto ocupado, debe haber una entrada en la tabla ocupa.

Usuarios

En esta tabla aparecen todos los usuarios que se encuentran registrados en la aplicación, con sus respectivas contraseñas. Es una tabla a la cual se accede en el momento en el que se crea un usuario nuevo (insertándolo en la tabla) y cuando un estudiante abre la aplicación para verificar que su contraseña es correcta.

Consta de tres, los cuales son los siguientes:

- **email:** Es la clave primaria, se trata de un varchar de longitud 40 caracteres que permite identificar al usuario.
- **contraseña:** Se trata de un varchar de longitud 100 caracteres en el cual se almacena la contraseña del usuario hasheada. Permite comprobar que la contraseña introducida es correcta.
- **activado:** Se trata de un tinyint, también conocido como booleano que permite saber si el usuario está activado o no. Si está a 0 no está activado y si está a 1 sí lo está. A la hora de acceder a la aplicación se comprueba que este campo está a 1 para así permitir el acceso a la misma.

UsuariosActivos

En esta tabla se almacena el email del usuario junto a su último envío de señal activo. Con ello, se puede determinar si ese móvil sigue estando encendido o no.

Es una tabla en la cual se inserta una fila cuando un usuario ocupa un puesto y dicha fila se va actualizando periódicamente cada 15 minutos, siempre que no se libere dicho puesto o se apague el móvil. Además, a esta tabla accede periódicamente un CRON Job para verificar qué móviles siguen activos.

Consta de tan solo dos campos, los cuales son los siguientes:

- **email:** Es la clave primaria, se trata de una clave foránea que hace referencia al campo email de la tabla usuarios. Es un varchar de 40 caracteres que permite saber quién es el usuario el cuál su móvil sigue encendido. Es una información importante, puesto que, de no aparecer aquí su email, su puesto se liberaba en la siguiente ejecución del CRON Job.
- **ultimaInfo:** Se trata de un timestamp que permite conocer cuándo fue mandada por última vez información desde el móvil de su usuario. Es una información muy importante ya que, dependiendo de ese valor, el CRON Job interpretará que el móvil ha sido apagado o no y liberará su puesto asociado o lo mantendrá.

Usuariostemp

Se trata de una tabla temporal, usada para verificar los correos de los diferentes usuarios que se registran en el sistema. Cuando un usuario se registra en el sistema, se almacena su email y contraseña hasheada en la tabla usuarios. Pero además, también se almacena aquí su email y su email hasheado y se le manda un email con un enlace para que certifique que el email que ha registrado le pertenece.

Consta de tan solo dos campos, los cuales son los siguientes:

- **email:** Es la clave primaria, se trata de una clave foránea que hace referencia al campo email de la tabla usuarios. Es un varchar de 40 caracteres que permite saber quién es el usuario que acaba de ser verificado al pulsar sobre el enlace.
- **emailHashed:** Se trata de un varchar de longitud 100 en el cual almacenamos el email del usuario hasheado. Permite verificar a qué usuario pertenece la cadena de caracteres a la cuál se acaba de acceder mediante el PHP, para así activar dicho usuario.

5.2. Base de datos SQLite

Esta base de datos es utilizada únicamente de manera local en la aplicación móvil. Gracias a ella se realiza una sesión correspondiente al usuario al iniciar sesión. La primera vez que el usuario inicia sesión en la aplicación se procede a la generación de esta base de datos permanente en la memoria del dispositivo.

De esta forma, cada vez que el usuario abra la app, se inicia la sesión directamente obteniendo sus credenciales de la base de datos, logrando así una mejor experiencia de usuario y evitando el repetitivo proceso de iniciar sesión cada vez que se inicia la app. Además, cuenta con otros campos en los que se almacenan otros datos que permiten reducir el número de consultas a la base de datos MySQL.

Esta sesión consta de los siguientes campos:

- **email:** Corresponde al email del usuario que esté registrado en la aplicación en ese móvil en ese momento.
- **contrasenia:** Se trata de la contraseña hasheada para evitar poner en riesgo la cuenta del usuario.
- **estado:** Se trata del estado actual del usuario. Estos estados pueden ser 'Descansando', 'Ocupando_puesto', 'Dentro_biblioteca' o 'Fuera_biblioteca'. Estos estados varían según la acción que esté desarrollando el usuario y permiten que se mantengan activas unas u otras funciones en la app.
- **latitud:** Corresponde a la latitud del usuario actualmente, medida mediante redes gps.
- **longitud:** Corresponde a la longitud del usuario actualmente, medida mediante redes gps.
- **puesto:** En esta variable se almacena el puesto ocupado por el usuario actualmente, en el siguiente formato: nombre de la biblioteca + planta de la biblioteca + sitio. Un ejemplo podría ser: FDI_01_020.
- **inicioEstado:** En este campo se almacena la fecha en la que el usuario inició su estado actual. Es un campo utilizado para, por ejemplo, determinar el tiempo de descanso disponible por un usuario determinado, como se desarrollará más adelante.
- **duraciónAcumulada:** Aquí se almacena la duración acumulada, en segundos, que tiene el usuario disponible para poder descansar.
- **planta1:** Aquí se almacena el nombre de la primera planta de la Biblioteca de la Facultad de Informática para poder mostrarlo en la app.
- **planta2:** Aquí se almacena el nombre de la segunda planta de la Biblioteca de la Facultad de Informática para poder mostrarlo en la app.
- **planta3:** Aquí se almacena el nombre de la tercera planta de la Biblioteca de la Facultad de Informática para poder mostrarlo en la app.

5.3. Comunicación entre los modelos de datos

Existe una conexión entre las dos bases de datos que implementan el proyecto, mediante la cual se intercambian datos para mantener la consistencia y actualizar el sistema a medida que vayan produciéndose acontecimientos.

En la Figura 27 se puede observar de manera gráfica las relaciones entre los algunos de los campos de la base de datos local con los campos de varias tablas de la base de datos del servidor.

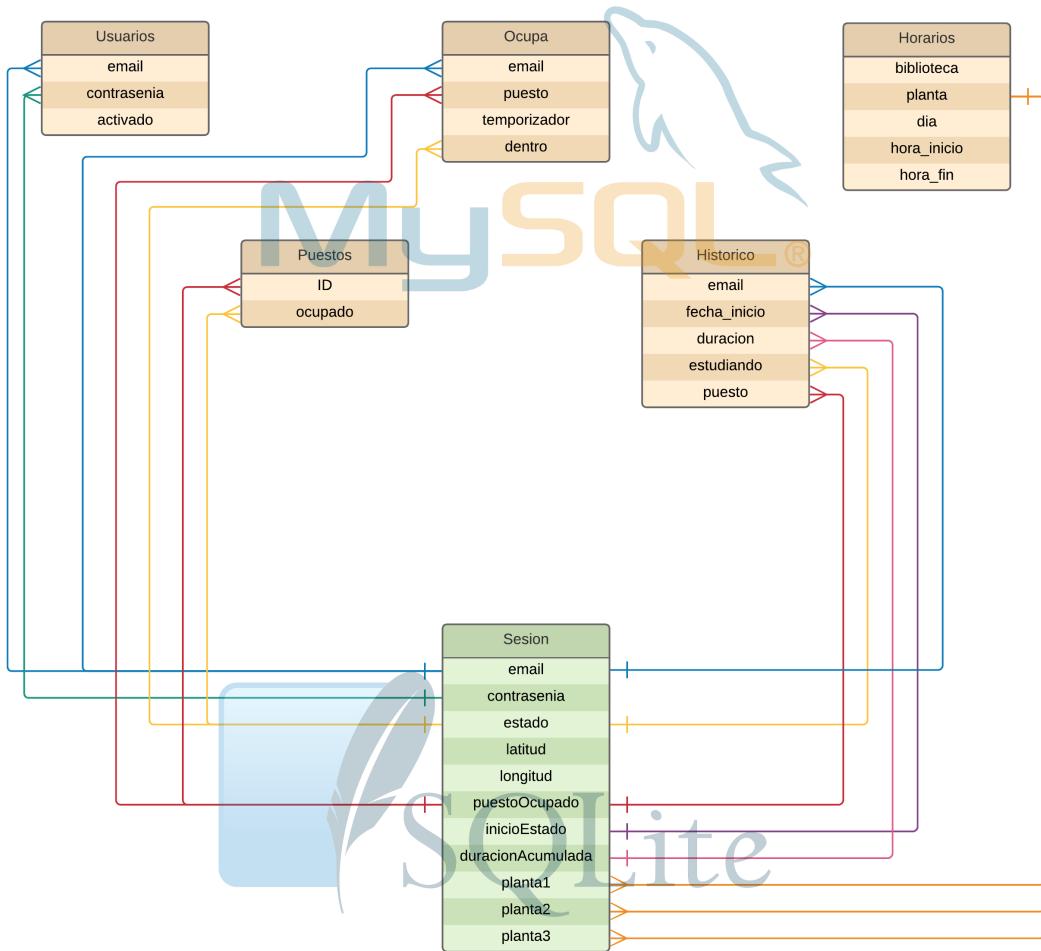


Figura 27: Diagrama de las relaciones entre las dos bases de datos

En todas las conexiones, a excepción de una, la base de datos SQLite envía información relevante al usuario que quiere interactuar con el sistema, ya sea iniciando sesión, registrándose, ocupando un puesto, desocupándolo o haciendo descansos. En todos estos casos, el servidor recibe el email del usuario junto al resto de datos necesarios para efectuar la operación e inserta o actualiza en las tablas correspondientes el nuevo acontecimiento.

Esto resulta imprescindible para mantener la unicidad y consistencia de los datos.

La única excepción es la relativa a la tabla horarios de la base de datos MySQL. Cada aplicación móvil lee de ahí el nombre de las plantas de la biblioteca para poder identificarlas dentro de la app. Es el único caso en el que la base de datos SQLite recibe datos para almacenarlos en lugar de enviar los que ya incorporan sus tablas.

6. Diseño de la aplicación

En este capítulo se va a hablar sobre el diseño de la aplicación. Versará sobre los diferentes colores elegidos, el tipo de iconos escogidos, la distribución de las diferentes funcionalidades de las aplicaciones...

Es importante destacar que se ha tratado de no romper con la relación establecida ya existente en la mayoría de usuarios del mundo digital. Diversas relaciones como puede ser el hecho de que al pulsar en otra pestaña se mostrará información similar sobre dicho apartado, o que al pulsar sobre un botón flotante se desplegarán nuevas características, son ejemplos de estas relaciones ya establecidas. Asimismo, se ha tratado de seguir en la medida de los posible los principios del diseño establecidos por Don Norman [7].

Se trata de una aplicación que ha sido desarrollada siguiendo la paleta de la gama cromática de la Universidad Complutense de Madrid. [8]. Al tratarse de un proyecto desarrollado en la UCM, se ha optado por utilizar los colores corporativos de ella, además de otros colores no presentes en la gama cromática de la UCM y que han sido necesarios incluir para la correcta implementación del proyecto. En la Figura 28 se pueden apreciar todos los colores utilizados, con su nombre y el código hexadecimal.

colorPrimario	colorPrimarioDark	colorPrimarioLight	colorComplu
#9f0000	#780000	#f05545	#b71234
colorFacultad	colorNegro	colorBlanco	colorPlata
#0098c3	#1e1e1e	#ffffff	#979492
colorOro	colorRojo	colorPlataTransp	colorVerde
#9b8942	#e53935	#73979492	#8bc34a
colorFondo	colorEstudiando	colorDescansando	colorPulsado
#fff2f4	#00bcd4	#ffc107	#84a0aaac

Figura 28: Paleta de colores utilizados para el desarrollo

Se va a realizar la explicación del diseño en dos partes. En una primera se va a hacer un repaso por los elementos claves del diseño de la aplicación web para, posteriormente pasar a explicar los de la aplicación móvil.

6.1. Parte web

Para el diseño de la parte web se ha tratado de conseguir una página que fuese intuitiva, con un diseño simple pero elegante en el que todas las opciones apareciesen a la vista del bibliotecario. De esta manera, en el momento en el que se inicia sesión, aparecen desplegadas en el encabezado todas las opciones que el/la bibliotecario/a tiene disponibles. Así se garantiza la accesibilidad y el fácil acceso a todas las funciones que están presentes en la página web.

Además, estas opciones presentan un gran tamaño sobre el que pulsar, dejando de lado pequeños botones que resultan difíciles de pulsar e interactuar con ellos. Además de ello, los botones utilizados son claramente autoexplicativos, tanto a nivel de ícono como a nivel de texto, haciendo que no requieran ninguna explicación extra sobre la funcionalidad que realizan.

Se ha optado por seguir un diseño amigable, con botones redondeados que ayuden a conseguir este objetivo. Además, existe un gran contraste entre los diferentes colores que componen la web para facilitar la navegación de los usuarios. También, un aspecto a remarcar es que el usuario sabe en todo momento sobre qué opción es la que va a pulsar debido al resaltado que se produce cuando se pasa con el ratón sobre cada una de las opciones. Es aquí donde se aprecia uno de los aspectos principales de los principios de diseño de Don Norman, el hecho de dar siempre un Feedback al usuario constantemente.

Se trata de una web responsive, de forma que al acceder a ella con cualquier dispositivo se va a mostrar a los usuarios correctamente. Para lograrlo, se ha hecho uso de BootStrap que facilita el desarrollo de webs de este tipo. Además, para la implementación de los iconos se ha usado los iconos de Font Awesome[9] y de Glyphicons[10] que garantizan el diseño responsive en todas las páginas web.

6.2. Parte móvil

Para el diseño de la aplicación, se realizó un estudio del tipo de usuario al que iba a dirigida. La aplicación está pensada para los usuarios que suelen frecuentar la biblioteca para estudiar sin importar edad ni estudios. Sin embargo, es cierto que el estudiante típico de la aplicación estudiará Informática, en la UCM, y tendrá un rango de edad de entre 18 a 30 años. Por este motivo, los usuarios deberán estar acostumbrados a utilizar aplicaciones móviles.

Para el desarrollo de la aplicación se eligieron una serie de principios[11]:

- **Simplicidad:** no sobrecargar la interfaz del usuario con mucha información, colores, o demasiada información que, lejos de resultar útil distraigan el foco de atención al usuario descartando el resto.
- **Consistencia:** favorecer que la aplicación sea intuitiva, aplicando ciertos estándares al diseño, para que el usuario pueda encontrar toda la información y funciones que busca rápidamente.
- **Navegación:** lo más importante de la aplicación, es que el usuario pueda recorrer el contenido, siempre de una manera fácil. Eso se consigue cuando la aplicación es intuitiva. Para ello se ha aplicado diferentes recursos:
 - Pestañas/tabs, para cambiar de pantalla a un contenido que sea relativo a la pantalla anterior, pudiendo así filtrar contenidos.
 - Listas, para ordenar verticalmente la información, en cajones. Esta información suele tener algo en común.

- Menú, desplazando desde la derecha y utilizado para agrupar en cajones diferentes funcionalidades. En numerosas aplicaciones es utilizado de una forma similar por lo que a los usuarios les será fácil e intuitivo navegar en la aplicación. Gracias a este patrón, se ha ahorrado en espacio y se ha conseguido que el usuario focalice su atención en la información.
- Botón flotante, utilizado para superponer un botón a la pantalla y anclarlo a la vista actual. Gracias a este recurso se ahorra espacio en la interfaz.

Para poder llevar a cabo el desarrollo de un buen diseño, *responsive*. se han seguido las pautas de Material Design de Google[12], aplicando diferentes capas, iconos y demás recursos, que puedan adaptarse a cualquier tipo de móvil con diferentes resoluciones y tamaños. La tipografía que se ha aplicado es *Roboto*[13], debido a que es el predefinido en el desarrollo de las aplicaciones móviles Android y tal y como aconseja la guía de Google. Además, se ha mantenido la homogeneidad de los colores a lo largo del proyecto.

El desarrollo del diseño pasó por varias fases. Primero se desarrollaron bocetos sobre todas las pantallas que habría en la aplicación a bajo nivel, exponiendo solo lo principal y sin centrarse en colores. Una vez se había tenido en cuenta la idea de lo que iba a aparecer en cada pantalla, se comenzó a desarrollar la interfaz con todos los detalles, realizando numerosas iteraciones y modificando algunos aspectos del diseño que podrían suponer grandes mejoras para el usuario.

Una vez finalizó el desarrollo de la aplicación, se realizó una prueba de experiencia a usuario a familiares y amigos para así tener una opinión de nivel de usuarios no expertos y expertos de alguna modificación de estilo-diseño. Tras los resultados obtenidos, se hicieron unas pequeñas modificaciones sobre el proyecto para finalizarlo. Los resultados del diseño definitivos de la aplicación se pueden observar en la siguiente sección.

Para finalizar, también se decidió crear un nombre para la aplicación y un logo (Figura 29) mediante el programa Dotgrid.



Figura 29: Logo de la aplicación

7. Implementación de la aplicación

En esta sección se van a describir las funcionalidades implementadas y los recursos que han sido necesarios para ello. Se dedicará una sección para cada uno de los componentes principales del sistema: aplicación web, aplicación móvil y otros componentes del sistema. Además, se tratará el tema del diseño, argumentando en cada caso el porqué de la elección de los colores, la disposición y qué recursos se han utilizado.

7.1. Aplicación web

Para desarrollar la aplicación web se han utilizado las siguientes tecnologías: HTML5, CSS3, PHP, Javascript y la framework de diseño Bootstrap. El mismo aporta un diseño responsive accesible para todos los dispositivos.

HTML5 y CSS3 son los lenguajes utilizados para generar el diseño de la página web, hacer que sea responsive, generar los encabezados, etc. Estos lenguajes, junto con Javascript, son la base del diseño de este apartado.

Con HTML5 se generan todos los formularios, los encabezados, los botones y prácticamente cualquier elemento presente en la web. CSS3 es el encargado de dar formato y estilo a todo lo que ha sido creado previamente por HTML5.

Por último, Javascript es usado para algunas funciones de la web, como pueden ser el mapa interactivo de la ubicación de la biblioteca o el gráfico donde se muestran las estadísticas entre otras que son indispensables desarrollar mediante dicho lenguaje.

Por otra parte, PHP es utilizado para la conexión y consultas con el servidor de la base de datos, cuyo proceso de conexión se explicará a continuación.

En cuanto a la estructura de la web, está distribuida en subcarpetas (véase Figura 30), cada una de ellas de acuerdo a la funcionalidad que implementan. Es decir, existe una carpeta para el login, otra para el iniciar sesión, otra para estadísticas, etc. Algunas carpetas tan solo contienen un PHP que realiza el procesamiento de la información mientras que otras disponen también de un PHP que implementa la interfaz gráfica de la funcionalidad. Además, existe una carpeta llamada resources (véase Figura 31) que contiene, a su vez, una carpeta diferenciada por cada recurso que utiliza la web, quedando agrupados en una única ubicación todos los ficheros CSS, los ficheros Javascript, las librerías de Bootstrap y jQuery empleadas, las imágenes, y otros archivos los cuales poseen funciones que son utilizadas repetidas veces durante la implementación de la web. Con ellas se consigue reutilizar gran parte de código, simplificarlo y optimizar la web, a la vez que se acerca al lenguaje natural.

Name	Size	Last Modified	Type	Permissions
buzon	70 bytes	11 may. 2018 10:21	httpd/unix-directory	0755
cgi-bin	6 bytes	9 may. 2018 12:31	httpd/unix-directory	0755
estadisticas	64 bytes	11 may. 2018 10:10	httpd/unix-directory	0755
horario	94 bytes	12 may. 2018 5:17	httpd/unix-directory	0755
limpiarbd	44 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
login	40 bytes	11 may. 2018 10:25	httpd/unix-directory	0755
logout	23 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
mapa	39 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
quienesSomos	40 bytes	11 may. 2018 10:38	httpd/unix-directory	0755
resources	121 bytes	11 may. 2018 10:28	httpd/unix-directory	0755
android.zip	2,14 MB	9 may. 2018 18:09	package/x-generic	0644
android9518.zip	2,14 MB	9 may. 2018 18:47	package/x-generic	0644
androidMigrar.zip	2,14 MB	9 may. 2018 13:53	package/x-generic	0644

Figura 30: Estructura general de las carpetas

Name	Size	Last Modified	Type	Permissions
css	141 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
fonts	4 KB	9 may. 2018 18:47	httpd/unix-directory	0755
img	198 bytes	11 may. 2018 10:39	httpd/unix-directory	0755
js	138 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
library	134 bytes	9 may. 2018 18:47	httpd/unix-directory	0755
template	42 bytes	11 may. 2018 10:30	httpd/unix-directory	0755
config.php	1,01 KB	9 may. 2018 18:08	application/x-httpd-php	0644
funciones.php	3,13 KB	11 may. 2018 10:28	application/x-httpd-php	0644

Figura 31: Estructura de la carpeta resources

A continuación se describirá cada funcionalidad de la aplicación web.

7.1.1. Inicio de sesión

Lo primero que se muestra al acceder a la web es el formulario de inicio de sesión. Como se puede observar en la ruta del navegador, es el index de la aplicación y se encuentra en la raíz.

Destaca el hecho de no disponer de una función de registro, que es debido a que, a diferencia de la aplicación móvil donde cualquiera se puede registrar, aquí no es posible. Son los desarrolladores los encargados de dar de alta a la biblioteca en cuestión una vez se haya llevado a cabo su implementación y es por eso por lo que no existe dicha funcionalidad.

Se ha utilizado una API de Google para la comprobación de la autenticidad del usuario y así poder asegurarse de que no es un robot. Dicho recaptcha (veáse Figura 32) aparecerá tras haber

ingresado las credenciales de manera incorrecta más de una vez. Esta es una de las barreras que se han implementado en la web, a continuación se detallarán otras.

```
<!-- Google ReCaptcha-->
<script src='https://www.google.com/recaptcha/api.js'></script>
```

Figura 32: Fragmento de código correspondiente al recaptcha

Se puede observar que el archivo que muestra dicha página se llama index.php, el cuál tiene gran parte de código en HTML y una porción en PHP, de ahí que su extensión sea .php.

En dicho código se puede ver cómo se utilizan los diferentes componentes de Bootstrap. También se puede observar cómo está incluido un header y un footer que son insertados desde dos ficheros PHP externos.

La conexión con el PHP correspondiente al login, el cual se encarga de comunicarse con la base de datos y comprobar si el usuario existe, se hace mediante un formulario que ofrece dos campos: el usuario, que son las siglas de la biblioteca, y la contraseña. Una vez llenados se envía esta información mediante un botón ubicado en la parte inferior.

En caso de que haya algún campo incorrecto, se hace saber al usuario visualmente mediante un mensaje que genera una función Javascript, aparte de poder ver el error en la URL.

Una vez esté completo, se realiza la comprobación de que el usuario no haya insertado caracteres extraños ni espaciados o código en HTML con etiquetas. Es decir, se hace una limpieza del código antes de poder tratarlo.

Después, se realiza una conexión con la base de datos del servidor para poder comprobar la existencia de la biblioteca y su contraseña correspondiente. Dicha contraseña está hasheada para evitar que alguien que entre en la base de datos o intercepte el mensaje la averigüe.

Si la consulta en la base de datos es favorable, se inicia la sesión para que el usuario pueda navegar por la página identificado y es redirigido a la página de buzón de sugerencias.

En cuanto al diseño empleado, se puede apreciar que el encabezado es diferente al resto de las demás páginas, en el cual solo aparece el logo de la Universidad Complutense de Madrid (UCM) tras un fondo rojo en relación a los colores definidos de la universidad.

En cuanto al footer, es prácticamente igual que en el resto de las páginas, proporcionando homogeneidad con el objetivo de que el usuario sienta pleno control sobre la aplicación web.

Todo lo que compete al encabezado y pie de página será desarrollado más detenidamente en una sección posterior.

A continuación se va a describir el diseño de la página de iniciar sesión.

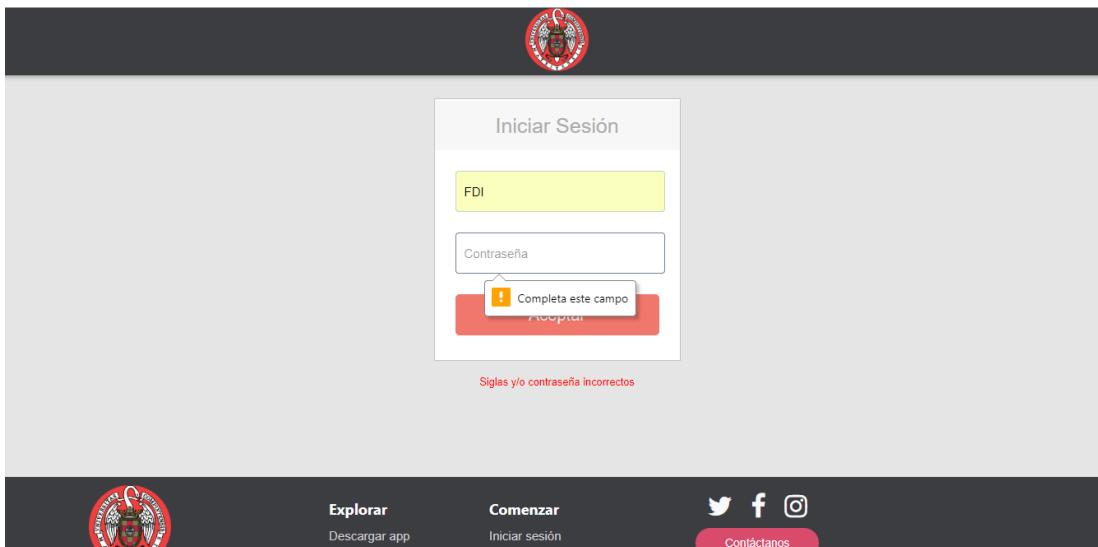


Figura 33: Errores al iniciar sesión

En la Figura 33 se observan los dos posibles errores que pueden llegar a aparecer, los cuales son:

- **Siglas y/o contraseña incorrectos:** aparece en caso de que el usuario o la contraseña no aparezca en la base de datos. Se ha elegido el color rojo por convenio para que el usuario sepa reconocer que se trata de un error. Este mensaje de error se muestra o se oculta gracias a una función en Javascript.
- **Completa este campo:** sale durante unos segundos en el campo que falte por llenar. Únicamente se comprueba antes de enviar la información del usuario y la contraseña al PHP.

En cuanto a los colores, el negro ha sido elegido para el header y el footer, para los botones, una gama de colores rojos (debido al ser el color de la UCM), y para el resto el blanco, que combina visualmente bien con los dos colores anteriores.

Por último, destacar que el diseño de esta página ha sido consensuado para que sea lo más simple y sencillo posible, con el objetivo de que el usuario identifique fácilmente las potencialidades de la página y de darle el menor número de funciones posibles para no sobrecargarla. En este caso solo tendría que llenar dos campos.

7.1.2. Buzón de sugerencias

Esta funcionalidad proporciona un canal de comunicación entre estudiantes y bibliotecarios.

En un vistazo general a la página web se puede ver que el header y el footer se repiten a lo largo de las diferentes páginas, con la misma estructura. En este apartado se pueden ver todos los mensajes que la biblioteca ha recibido (en este caso la Biblioteca de la Facultad de Informática), donde cada uno está dividido en tres columnas: ‘Respondido’, que indica si se ha respondido o no el mensaje, ‘Fecha’ y ‘Asunto’. A la derecha se encuentra el botón ‘Mostrar’, sobre el que se puede clicar para leer el diálogo que se ha mantenido con el usuario, es decir, los mensajes que se han intercambiado.

Es importante saber que no se muestra al bibliotecario la identidad del usuario que mandó el mail para poder respetar su privacidad. Se ha utilizado el icono de una caja vacía para el caso en el que el mensaje no se ha respondido, y una caja con un check si el mensaje ya ha sido respondido.

Se ha definido a 10 el número máximo de mensajes a mostrar por página.

The screenshot shows a user interface for a 'Sugestions' application. At the top, there is a dark navigation bar with five items: 'BUZÓN DE SUGERENCIAS' (with an envelope icon), 'ESTADÍSTICAS' (with a bar chart icon), a logo of the University of Valencia, 'ESTABLECER HORARIO' (with a clock icon), and 'CERRAR SESIÓN' (with a cross icon). Below the navigation bar is a table listing ten messages. The columns are 'Respondido' (checkbox), 'Fecha' (date), 'Asunto' (subject), and 'Mostrar' (button). The 'Mostrar' button is a light blue rectangle with white text. The 'Respondido' column contains checkboxes, some of which are checked (indicated by a blue square with a white checkmark). The 'Mostrar' button for each row is aligned with its respective checkbox. The table has a light gray background and thin black borders between the rows and columns.

Respondido	Fecha	Asunto	Mostrar
<input type="checkbox"/>	2018-05-24 11:18:26	Libros MMI	Mostrar
<input type="checkbox"/>	2018-05-22 12:05:27	libros mdl	Mostrar
<input type="checkbox"/>	2018-05-21 14:46:43	jssjjs	Mostrar
<input type="checkbox"/>	2018-05-21 14:46:11	La Biblioteca me ha amonestado	Mostrar
<input checked="" type="checkbox"/>	2018-05-21 12:44:34	hola	Mostrar
<input checked="" type="checkbox"/>	2018-05-07 08:17:33	Prueba de ECO	Mostrar
<input checked="" type="checkbox"/>	2018-04-27 12:09:00	Prueba Gorricho	Mostrar
<input type="checkbox"/>	2018-04-25 10:43:37	f	Mostrar
<input type="checkbox"/>	2018-04-25 10:42:12	ewq	Mostrar

Figura 34: Buzón de sugerencias - Aplicación web

Al pulsar el botón ‘Mostrar’ de un mensaje se abrirá para poder interactuar con él. En el caso de que ya se haya respondido, aparecerán tanto el mensaje recibido como la respuesta del bibliotecario. En caso contrario, el bibliotecario tendrá la opción de responder con un campo en el que se podrá introducir texto. El recurso utilizado para mostrar el mensaje es un pop-up para poder superponer el mensaje a la lista de mensajes sin tener que redirigir a otra página distinta, disminuyendo la sobrecarga de pestañas al usuario.

En las Figuras 35 y 36 se pueden ver dichos ejemplos respectivamente:

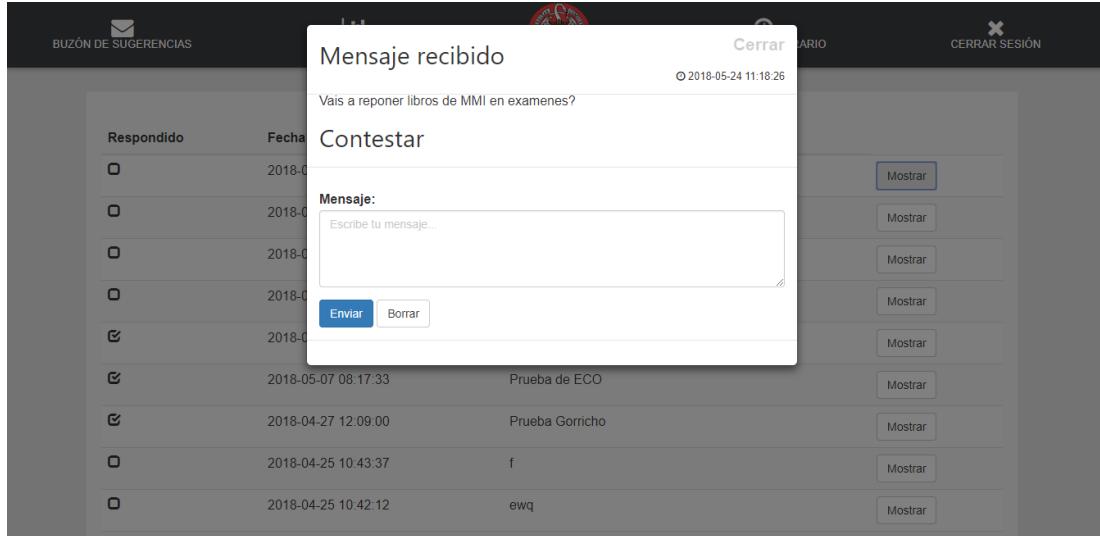


Figura 35: Buzón de sugerencias - Contestar mensaje

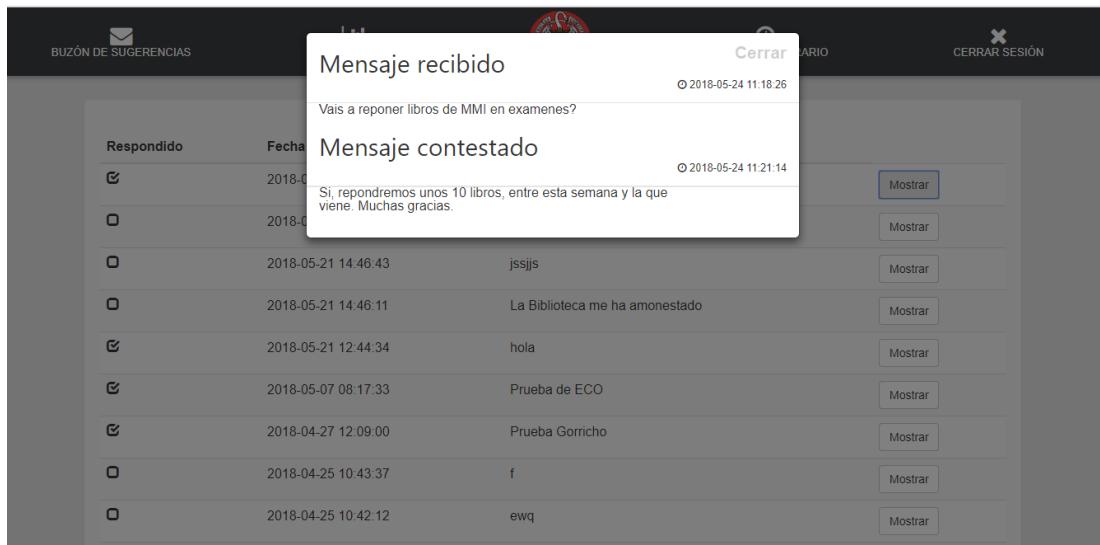


Figura 36: Buzón de sugerencias - Mensaje ya contestado

En lo que respecta a la implementación, en primer lugar se comprueba que el bibliotecario estélogueado. En caso de que el usuario quiera entrar en esta página sin haber iniciado sesión antes, se le redirigirá directamente a la página de login para que inicie sesión. Este es un aspecto que sucede en todas las funcionalidades en las cuales es necesario registrarse.

Los mensajes se muestran en la lista en orden de fecha descendente y se establece el límite de mensajes por página a 10.

Lo primero que se muestra de los mensajes es si ha sido respondido (si el id de dicho mensaje aparece en dos ocasiones en la base de datos) o está pendiente de ello (eso querrá decir que solo se ha encontrado una coincidencia con el id).

Una vez mostrada la fila de cada mensaje se habilita el botón ‘Mostrar’, para que se pueda conocer el contenido del mensaje. Al pulsar en dicho botón, primero se muestra el mensaje que corresponde a dicho id y luego existen dos posibles caminos: mostrar el mensaje contestado, en caso de que aparezca dos veces el mismo id en la base de datos de la tabla envía, o mostrar un cuadro de texto para poder insertar una nueva fila en dicha tabla si tan solo aparece una.

En cuanto al diseño, se emplean las cajas de respuesta del mensaje con los iconos de *glyphicon*, que aportan un diseño responsive. Al ser el header y el footer negros, se ha utilizado una gama de colores blanco para no sobrecargar al usuario con colores muy llamativos.

7.1.3. Estadísticas

Se trata de una funcionalidad para la cual es necesario hacer uso de Highcharts [16], una biblioteca a la que se accede mediante Javascript y con la que se puede especificar el intervalo de tiempo que se quiere consultar. En la Figura 37 se aprecia cómo se establece la conexión con dicha librería y se configura alguna opción del gráfico como es el selector.

```
<script src="https://code.jquery.com/jquery.js"></script>
<!-- Importo el archivo Javascript de Highcharts directamente desde su servidor -->
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
<script>

Highcharts.setOptions({
    global: {
        useUTC: false
    }
});

chartCPU = new Highcharts.StockChart({
    chart: {
        renderTo: 'contenedor'
        //defaultSeriesType: 'spline'

    },
    rangeSelector : {
        enabled: true
    }
});
```

Figura 37: Fragmento de código correspondiente a la generación de la gráfica

Para la obtención de los datos desde la base de datos es necesario hacer una consulta a la misma. La lógica de la funcionalidad está implementada en estadísticas.php de la carpeta que lleva su mismo nombre. En ese fichero se encuentra, entre otras cosas, la query con la que se selecciona del histórico la suma total de la duración de los tiempos de estudio y se agrupan por días: “*SELECT ((SUM(duracion) / COUNT(DISTINCT(email))) / 60) as minutosPorEstudiante, DATE_FORMAT(fecha_inicio, 'asd %e- %m- %Y') as dia FROM ‘historico’ WHERE estudiando = 1 AND SUBSTRING(puesto, 1, 3) = ‘\$usuario’ GROUP by DATE_FORMAT(fecha_inicio, ‘%e- %m- %Y’) ORDER BY DATE_FORMAT(fecha_inicio, ‘%Y- %m- %d’)*”

Uno de los principales problemas encontrados en el desarrollo de esta funcionalidad fue el hecho del desarrollo de la query. Como se puede ver, se trata de una consulta bastante compleja, por lo que llevó un poco de tiempo al equipo de desarrollo dar con la query que satisfaciera todas las condiciones planteadas.

Además, como problema añadido a la hora de migrar a la versión final, el gráfico mostraba una línea hacia atrás desde, por ejemplo, el día 19 de cada mes hasta el día 2. Este pequeño problema era debido a que a la hora de rescatar los datos de la BBDD, no se obtenía el 0 delantero de los números del 1 al 9, por lo que la ordenación no era la adecuada.

Por último, otro gran problema que surgió al pasar al servidor de pago, con unas configuraciones horarias diferentes, fue que el gráfico de Highcharts no funcionaba, no mostraba nada en pantalla. Finalmente y tras probar varias soluciones, se dió con la solución correcta, que consistió en establecer la zona horaria cuando se generaba el gráfico a la zona horaria de Madrid mediante la siguiente línea de código:

```
date_default_timezone_set('Europe/Madrid')
```

En lo que concierne al diseño, al igual que en el resto de funcionalidades implementadas en la web, se muestra en la parte superior un encabezado en el cual aparecen disponibles todas las funcionalidades implementadas en la página web. Además, aparece el mismo footer que está desarrollado para todas las páginas una vez se ha iniciado sesión. En cuanto al diseño propiamente de la parte de las estadísticas, se trata de un gráfico en el cual mediante un selector se puede establecer el rango concreto sobre el que queremos hacer la consulta.



Figura 38: Gráfica estadísticas en la web

Como se puede apreciar en la Figura 38, en la parte inferior del gráfico existe un selector corredero que permite ajustar las fechas a consultar. A medida que se establece el rango, el gráfico se va adaptando dinámicamente, modificando su escala a los datos proporcionados para ese intervalo.

7.1.4. Horario

Otro de los atractivos que tiene la aplicación web es que el bibliotecario podrá establecer el horario de su biblioteca y modificarlo las veces que sean necesarias. Incluso existe la posibilidad de asignar horarios diferentes a cada una de las plantas o salas que conforman la biblioteca en el caso de que no lo compartan entre ellas, como ocurre en la biblioteca de la Facultad de Informática.

Esta funcionalidad cubre una doble necesidad. Por un lado, en determinadas circunstancias la biblioteca se ve obligada a alterar su horario normal de apertura y/o cierre y necesita que los usuarios que frecuentan las instalaciones conozcan la nueva situación. Puede encontrarse aquí una herramienta idónea donde actualizar tal información, que quedará reflejada instantáneamente en la aplicación móvil. Gracias a que el horario se establece de manera semanal, resultará muy sencillo cambiar únicamente el día o días que se deseen sin tener que modificar el resto de la semana. Por otro lado, para quien acude a la biblioteca siempre puede existir la duda de qué días y a qué hora abría o cerraba cierta biblioteca. Seleccionar en la app la biblioteca que se quiera consultar y acceder al apartado Horario resolverá la incógnita de manera muy sencilla y rápida.

En lo relativo al diseño y estructura, en la Figura 39 se puede el aspecto de este apartado de la web.

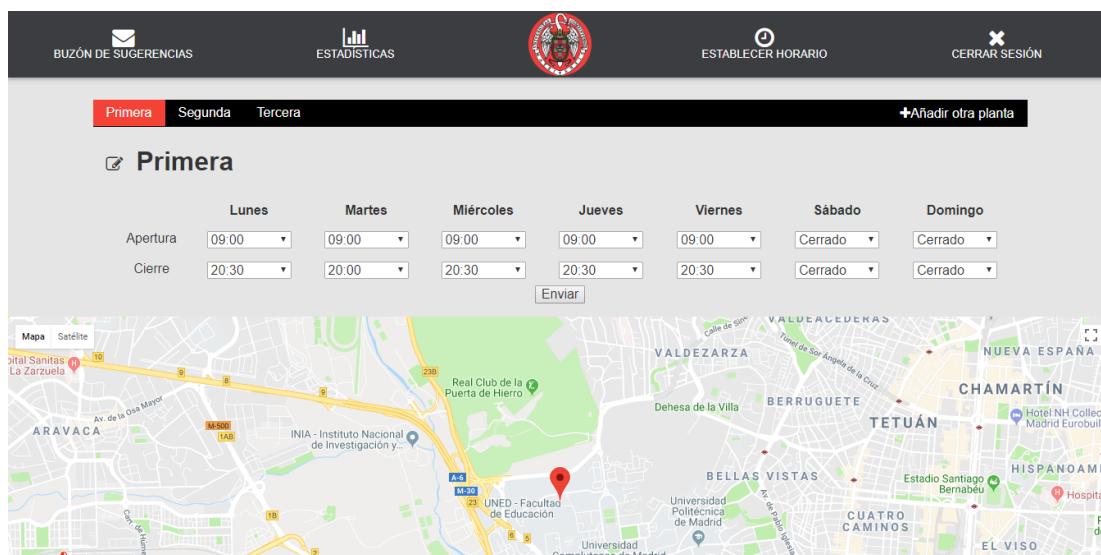


Figura 39: Establecer horario en la web

A esta funcionalidad de la aplicación web se puede acceder a través de la pestaña del header en la que se puede leer “Establecer Horario”. Al hacer clic, será redirigido al fichero index.php ubicado en la carpeta horario del árbol de directorios. La página está dividida en pestañas, correspondiéndose cada una de ellas con las diferentes plantas o salas de las que dispone la biblioteca, mostrándose en esta imagen la pestaña correspondiente a la primera planta. El color rojo de la barra de pestañas indica cuál está seleccionada y se muestra actualmente, pudiendo cambiar entre plantas para asignar los distintos horarios. Como indica el símbolo a la izquierda del nombre de la planta situado bajo la barra, ese nombre puede cambiarse y ser sustituido por el que el bibliotecario considere oportuno. En este caso, al tratarse de la biblioteca de la

Facultad de Informática, las plantas están identificadas como Primera, Segunda y Tercera, pero si se tratase de la Biblioteca María Zambrano, por ejemplo, las salas podrían llamarse Sala de Filología o Sala de Derecho para que los usuarios de la aplicación sepan identificarlas.

A la derecha de la barra de tareas se encuentra un botón en el que se lee “Añadir otra planta”. Al pulsarlo, se generará automáticamente otra pestaña que tendrá por nombre la palabra “Planta” seguido del número de pestaña que le corresponda. Es decir, en la imagen anterior, al pulsar en ese botón se añadiría otra pestaña con el nombre de “Planta 4”, y quedaría seleccionada y abierta para poder realizar directamente cambios en ella. Si se cambia el nombre de alguna de las plantas, será necesario guardar y refrescar la página para que el cambio quede reflejado en la barra de pestañas.

El horario está representado por una tabla horizontal que dispone de 8 columnas, 7 de ellas se designan a cada uno de los días de la semana, y 3 filas, siendo 2 de ellas para indicar la hora de apertura y cierre de la biblioteca. Obviando la primera fila y la primera columna, que son empleadas por los títulos, el resto de las celdas se corresponden con las horas de apertura y cierre de los diferentes días de la semana. Cada una de estas celdas la compone un botón desplegable en el que aparecen como opción, por cada una de las 24 horas del día, la hora en punto y la media hora. Es decir, al desplegar uno de los botones se puede elegir una hora entre las 00:00, 00:30, 01:00, 01:30, hasta las 23:30, además de dos opciones especiales que tienen por nombre “Cerrado” y “Todo el día”. Al seleccionar alguna de estas dos opciones como hora de apertura, se selecciona automáticamente la misma opción como hora de cierre en ese día y además se deshabilita ese botón para que no pueda ser modificado y las horas de apertura y cierre sean coincidentes.

En la parte inferior de la tabla se dispone de un botón con la palabra “Enviar” con el que se guarda el nuevo horario en la base de datos, creándolo si no existe o modificándolo en otro caso. Cabe destacar que al pulsar este botón se guardarán en la base de datos todas las plantas que existan en la página con sus horas establecidas, no sería necesario enviar cada planta individualmente, si no que todo funciona como un mismo formulario.

Por último, cabe destacar la existencia en la parte inferior de la página web de un mapa donde aparece señalada la ubicación de la biblioteca que tiene la sesión iniciada en la web, con objeto de comprobar que su localización en la base de datos es correcta.

Tras conocer el diseño, se explicará ahora cómo ha sido su implementación.

Como ya se ha dicho, este apartado de la web se corresponde con el fichero index.php de la carpeta horario. La primera misión de este php es conectarse con la base de datos y comprobar, en la tabla horarios, si ya existe un horario guardado para esta biblioteca. Si es así, se recogen los datos y para posteriormente ir recorriéndolo y pintándolo. En el caso contrario, se mostrará una plantilla para que el encargado de asignarlo lo complete con los datos correctos.

Se expondrá en primer lugar el caso en el que no se encuentren datos sobre esta biblioteca. Lo primero que genera este fichero php es la barra de pestañas en las que se pueden ver las diferentes plantas. Se mostrará solamente una, llamada “Planta 1”, que directamente aparecerá seleccionada en color rojo, y el botón “Añadir otra planta”.

A continuación se creará un nuevo elemento <div> cuyo identificador será el nombre de la planta, en este caso “plantal”, y que servirá para identificar la tabla correspondiente a esta planta. En él, se abrirá un formulario y se comenzará a generar, mediante un bucle que recorre todos los días de la semana, la tabla del horario con cada una de sus opciones, quedando selec-

cionadas por defecto las 09:00 como hora de apertura y las 21:00 como hora de cierre. Requieren especial atención los atributos ‘id’ y ‘name’ de cada una de estas celdas, ya que asignarles un correcto identificador solucionó un gran problema que surgió y que se explicará más adelante. Para concluir el formulario, se crea el botón “Enviar” que transferirá los datos del formulario a otro fichero ubicado en la misma carpeta llamado “guardarHorario.php”, que insertará este horario en la base de datos. Este archivo se encargará de recoger los datos que se mandan, contabilizar el número de plantas que se van a ingresar en la base de datos y mediante un bucle que dará tantas vueltas como platas haya, recogerá los datos de cada planta y los insertará o reemplazará en la tabla ‘horarios’ de la base de datos.

Si ya existen datos almacenados para esa biblioteca, antes de generar ningún elemento HTML se hace una nueva consulta para determinar el número de plantas almacenadas en la base de datos, con el objetivo producir tantas pestañas y tantos bloques como sean necesarios. Se crea la barra y se le asigna a las pestañas el nombre que se lee de la base de datos. Ahora se crean los bloques que se corresponden con las distintas plantas, cada uno con su nombre correspondiente. De la misma manera que en la plantilla, se generan las celdas de la tabla, pero esta vez comprobando qué opción de los desplegables debe quedar seleccionada. Para ello, se realiza una nueva consulta que devuelve los datos correspondiente a una planta, ya que se consiguen los nombres de las mismas en la consulta anterior. Por cada una, se leen los valores de las horas de apertura y cierre y es el que queda seleccionado al generar las celdas de la tabla, repitiendo este proceso en todas las plantas existentes. Finalmente, tras haber sido creados todos los bloques con su correspondiente tabla, se crea el botón “Enviar” y se cierra el formulario, que englobará a todas las plantas, quedando cada una de sus variables perfectamente diferenciadas por sus atributos ‘id’ y ‘name’.

Al desarrollar todo este proceso surgieron varios problemas, siendo dos los más importantes: uno fue cómo hacer para que un mismo botón sirviera para enviar los datos de distintos formularios y otro diferenciar las horas de apertura y cierre de las distintas plantas. La solución del primero fue simplemente agrupar todas las plantas en un único formulario, cada una contenida en un elemento `<div>` y con las variables diferenciadas. Esta solución, que parece tan sencilla, no se pensó desde el primer momento debido a que las diferentes plantas estaban siendo interpretadas como elementos completamente independientes, pero cuando se comprendió bien que aunque los bloques estén ocultos siguen siendo parte de la web, incluidas sus variables, se colocaron todos dentro de un único formulario, quedando así englobadas por un único botón.

Esto lleva al segundo problema, saber cómo diferenciar las diferentes variables de cada una de las plantas, es decir, cómo diferenciar el valor correspondiente a, por ejemplo, la hora que abría el lunes la planta 1 y la planta 2. Dado que la forma de identificar cada una de las variables de un formulario pasa por consultar su atributo ‘name’, la solución era darle un identificador único y que fuese lo suficientemente claro, el cual fue “numeroDePlanta_diaDeLaSemana_abre” para la hora de apertura y “numeroDePlanta_diaDeLaSemana_cierra” para la hora de cierre. De manera similar se hizo para el ‘id’. Por ejemplo, para la hora de apertura del lunes de la planta 1, el ‘name’ de su variable era “1_Lunes_abre”.

Por último, es de mención cómo funciona la barra de pestañas de las distintas plantas. Su correcto funcionamiento es posible gracias a dos funciones, una en Javascript llamada “abrirPlanta” para mostrar las diferentes plantas y otra en jQuery encargada de añadir una nueva al pulsar en el botón correspondiente. La primera buscará el bloque `<div>` que tenga como identificador el nombre de la planta que aparece en la pestaña que pulsada, lo hará visible, ocultará el bloque de la pestaña que se estaba mostrando y pondrá la pestaña seleccionada de color rojo. La segunda de ellas es más compleja, ya que tiene que generar una nueva pestaña, un nuevo bloque y nombrar cada uno de sus elementos correctamente para identificarlos individualmente

de manera inequívoca. El procedimiento pasa por clonar el <div> de la primera planta, para posteriormente cambiar todos sus atributos y valores. El nombre y el id se conseguirá averiguando cuál es el número de la nueva planta, contabilizando las ya creadas. Se le asigna ese nombre y se coloca justo encima del botón “Enviar”. Lo siguiente es renombrar el ‘id’ y ‘name’ de todas las variables del formulario de la manera explicada anteriormente, y dejar seleccionadas las horas que se definieron en la plantilla. Por último, se crea la nueva pestaña en la barra, con el nombre adecuado, y se hace clic mediante una función Javascript para que se abra automáticamente tras haber pulsado en “Añadir otra planta”.

7.1.5. Cerrar sesión

Como se comentó previamente en este capítulo, algunas de las carpetas en las que se divide la web no presentan ninguna interfaz gráfica, tan solo ejecutan una funcionalidad. Cerrar sesión es una de ellas. Se trata de una funcionalidad presente siempre en el lateral derecho del encabezado que permite cerrar la sesión iniciada en la web. Se le ha asignado ese emplazamiento siguiendo el patrón Sign-In tools, el cual dice que es una forma de situar al usuario en la página web y que si quiere realizar la función de cerrar sesión la encontrará rápidamente, debido a que está acostumbrado a tener esa opción en el mismo sitio en la mayoría de portales web.

Para cerrar sesión, una vez se pulsa sobre dicha opción, se desenlazan todos las variables de la sesión con session_unset() y se destruye la sesión con session_destroy(). Una vez realizado esto, se redirige a la página principal, en la que existe la posibilidad de volver a loguearse.

7.1.6. Quiénes somos

Esta página, presente en el pie de página, es la huella, marca o presentación de los integrantes y desarrolladores de la web. Al acceder a ella, se observa una página sencilla pero elegante en la que aparecen el nombre, los apellidos y la foto de cada uno de los tres estudiantes desarrolladores del presente Trabajo de Fin de Grado.

Es una página a la cual se puede acceder indistintamente de haber iniciado sesión, no requiere de ninguna contraseña ni usuario para ser accesible como sí que ocurre con la mayoría de páginas restantes.

7.1.7. Header

El encabezado es un elemento que está siempre presente en la web. A la hora de realizarlo, se pensó cómo hacer que fuera intuitivo, rápido y, sobre todo fácil, de usar. Tras barajar varias opciones, se determinó que, puesto que las funciones que iban a estar disponibles en la página web eran, a grandes rasgos, cuatro, lo ideal sería que estuvieran todas a la vista del usuario, sin tener que navegar por complejos menús para encontrarlas.

Así pues, se estableció un header que contase con las cuatro funcionalidades principales: buzón de sugerencias, estadísticas, establecer horario y cerrar sesión. En el centro de las cuatro opciones, puesto que de momento la única biblioteca implementada es la de la Facultad de Informática de la UCM, se decidió colocar el escudo de la Universidad Complutense de Madrid. Se trata de un header responsive, que se adapta perfectamente a las diferentes dimensiones de pantalla y a dispositivos móviles.

En cuanto al diseño, se optó por el color negro, con los iconos de Material Design de color blanco para darle un estilo sobrio y elegante. Al pasar el mouse sobre cada uno de los elementos del encabezado, se remarca en un color gris claro, poniendo el ícono de color negro para así ofrecer al usuario información clara de qué opción es en la que va a pulsar.

Preside el encabezado (véase Figura 40) el escudo de la UCM sobre una circunferencia de color rojo.



Figura 40: Encabezado con la sesión iniciada

No tendría sentido que las cuatro opciones del encabezado se mostraran cuando la sesión no está iniciada. En ese caso (como ocurre por ejemplo en la pestaña de iniciar sesión), las opciones desaparecen quedando tan solo el escudo de la UCM, como se puede apreciar en la Figura 41 siguiente.



Figura 41: Encabezado sin haber iniciado sesión

7.1.8. Footer

De igual modo que en el caso del header, se trata de un elemento que se encuentra siempre presente en la página web. A la hora de diseñarlo, se han incluido las diferentes opciones que deben estar disponibles a cualquier usuario que accediese a la web, independientemente de que estuviera logueado o no.

Además, es necesario que sea muy visual, intuitivo y amigable. Es por ello por lo que se apostó por el diseño actual y se decidió incluir todas esas opciones. Se ha creído necesario que cualquier usuario que acceda a la web pueda informarse sobre quiénes son los desarrolladores del proyecto y pudiese descargar la app móvil, que es la parte del proyecto que está destinada al gran público. Por eso ha sido creado un grupo de opciones llamado Explorar en el que se encuentran estas dos opciones: Quiénes somos y Descargar app. Descargar app se trata de un enlace a Play Store donde se puede descargar la app móvil Biblioteko.

Tras este grupo de opciones aparece otro, denominado Comenzar, en el cual estña la opción Iniciar sesión, que redirige a la página de login. Esta es la única parte del footer que difiere según se haya iniciado sesión o no, ya que no tiene sentido que haya un enlace a iniciar sesión si ya está iniciada. Por último, a la derecha, está la parte en la que los usuarios pueden comunicarse con los desarrolladores. Es ahí donde aparecen los tres iconos de las redes sociales en las cuales existen perfiles creados y activos (Twitter, Facebook e Instagram). Además, existe un botón Contáctanos mediante el cual, al pulsarlo, se le abre al usuario su programa de correo para que pueda enviar un correo electrónico al equipo desarrollador.

En lo que respecta al diseño, se ha optado por el mismo color negro que en el encabezado, para así dar homogeneidad a la página y un aspecto serio. A la izquierda del pie de página se encuentra el escudo de la Universidad Complutense de Madrid, al igual que en el encabezado sobre un fondo rojo. Hacia la derecha podemos apreciar los dos grupos de opciones comentados anteriormente. Al pasar por encima de alguna de las opciones, ésta se resaltará en un blanco más intenso. Por último, a la derecha del todo, se ubican los enlaces de las redes sociales y un botón en color rojo, que que destaca del resto, para que los usuarios puedan contactar con

los desarrolladores. Los botones de las redes sociales se resaltan al pasar por encima de ellos, para asegurar al usuario que está pulsando sobre ellos, cada uno con su color característico. Por último, decir que el logo de la Universidad Complutense de Madrid, además de darle un color rojo (predeterminado color de la universidad), está presente en todas las páginas con el objetivo de dejar constancia en el bibliotecario de qué funciones está haciendo y sobre qué universidad. En la Figura 42 se puede observar el pie de página.

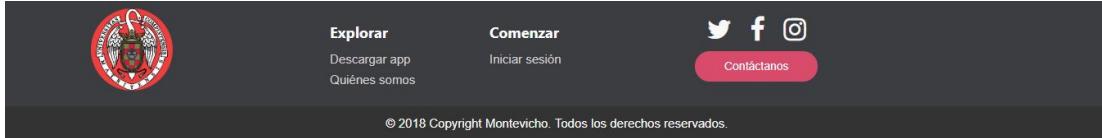


Figura 42: Pie de página

7.2. Aplicación móvil

La aplicación móvil está desarrollada en Java para la parte de la lógica, XML para las vistas de cada pantalla, PHP para la conexión con el servidor y JSON para el intercambio de datos entre la aplicación y el servidor.

Se trata de una aplicación móvil nativa desarrollada específicamente para dispositivos Android, optimizando su uso y rendimiento en este sistema operativo pero con el inconveniente de ser la única plataforma que la soporta. Por ello, ha sido desarrollada con el entorno de desarrollo que Android[17] proporciona: Android Studio. La estructura del proyecto se encuentra dividida en carpetas de Android Studio, tal y como se ve en la Figura 43.

	java	21/05/2018 13:52	Carpeta de archivos
	res	21/05/2018 13:52	Carpeta de archivos
	AndroidManifest	22/05/2018 11:08	Archivo XML

Figura 43: Directorio raíz del proyecto

Un archivo a destacar es Android Manifest, donde están definidas todas las clases, actividades, services y permisos necesarios que emplea la aplicación, entre otras cosas. Asimismo, todas las bibliotecas utilizadas también están incluidas aquí, además de indicarse la API mínima que puede ejecutar la aplicación.

Todos los archivos Java están organizados en la carpeta ‘java’, dentro de la cual hay dos subcarpetas, que serán paquetes en la organización del proyecto Java, ‘SQLite’ (que define la base de datos SQLite empleada para guardar la información del usuario una vez inicie sesión) y ‘PIP’ (que contiene clases utilizadas para determinar la ubicación de la biblioteca a partir de unos algoritmos basados en polígonos). A continuación se muestran dichas subcarpetas y algunos archivos Java.

	BibliotecaFDIplanta3	21/05/2018 19:03	Archivo JAVA
	BuzonSugerencias	21/05/2018 19:03	Archivo JAVA
	MainActivity	21/05/2018 19:03	Archivo JAVA
	OcuparPuesto	21/05/2018 19:03	Archivo JAVA
	Registro	21/05/2018 19:03	Archivo JAVA
	Bibliotecas	21/05/2018 19:03	Archivo JAVA
	AdaptadorMensajes	21/05/2018 19:02	Archivo JAVA
	MensajesRecibidos	21/05/2018 19:02	Archivo JAVA
	VaciarPuesto	21/05/2018 19:02	Archivo JAVA
	PIP	22/05/2018 11:07	Carpeta de archivos
	SQLite	22/05/2018 11:07	Carpeta de archivos

Figura 44: Directorio Java del proyecto

Las vistas se dividen en varias carpetas dentro de la carpeta madre ‘res’, como son: ‘layouts’, donde están las vistas por cada Activity, ‘menu’, donde se observan los diferentes menús implementados, y las diferentes carpetas de imágenes de distintos tamaños correspondientes a Material Design, entre muchas otras.

 drawable	24/05/2018 15:49	Carpeta de archivos
 drawable-hdpi	24/05/2018 15:49	Carpeta de archivos
 drawable-mdpi	24/05/2018 15:49	Carpeta de archivos
 drawable-v24	24/05/2018 15:49	Carpeta de archivos
 drawable-xhdpi	24/05/2018 15:49	Carpeta de archivos
 drawable-xxhdpi	24/05/2018 15:49	Carpeta de archivos
 drawable-xxxhdpi	24/05/2018 15:49	Carpeta de archivos
 layout	24/05/2018 15:49	Carpeta de archivos
 menu	24/05/2018 15:49	Carpeta de archivos
 mipmap-anydpi-v26	24/05/2018 15:49	Carpeta de archivos
 mipmap-hdpi	24/05/2018 15:49	Carpeta de archivos
 mipmap-mdpi	24/05/2018 15:49	Carpeta de archivos
 mipmap-xhdpi	24/05/2018 15:49	Carpeta de archivos
 mipmap-xxhdpi	24/05/2018 15:49	Carpeta de archivos
 mipmap-xxxhdpi	24/05/2018 15:49	Carpeta de archivos
 values	24/05/2018 15:49	Carpeta de archivos
 values-420dpi	24/05/2018 15:49	Carpeta de archivos
 values-hdpi	24/05/2018 15:49	Carpeta de archivos
 values-v21	24/05/2018 15:49	Carpeta de archivos
 values-xhdpi	24/05/2018 15:49	Carpeta de archivos
 values-xxhdpi	24/05/2018 15:49	Carpeta de archivos
 values-xxxhdpi	24/05/2018 6:46	Carpeta de archivos

Figura 45: Directorio Res del proyecto

En la Figura 46 se observa el contenido de la carpeta ‘layouts’, siendo cada archivo de los que contiene cada una de las vistas de las diferentes actividades:

5	activity_ayudanos	21/05/2018 19:10 Archivo XML
5	activity_biblioteca_fdiplanta1	21/05/2018 19:10 Archivo XML
5	activity_biblioteca_fdiplanta2	21/05/2018 19:10 Archivo XML
5	activity_biblioteca_fdiplanta3	21/05/2018 19:10 Archivo XML
5	activity_bibliotecas	21/05/2018 19:10 Archivo XML
5	activity_buzon	21/05/2018 19:10 Archivo XML
5	activity_cerrar_sesion	21/05/2018 19:10 Archivo XML
5	activity_estadisticas_ejemplo	21/05/2018 19:10 Archivo XML
5	activity_estadisticas_ocupacion	21/05/2018 19:10 Archivo XML
5	activity_horario	21/05/2018 19:10 Archivo XML
5	activity_main	21/05/2018 19:10 Archivo XML
5	activity_mensaje	21/05/2018 19:10 Archivo XML
5	activity_mesa4_1	21/05/2018 19:10 Archivo XML

Figura 46: Directorio Layouts del proyecto

7.2.1. Starter

Esta es la actividad inicial, la primera que se lanza cada vez que se abre la aplicación, lo que explica su nombre.

Su función es comprobar si existe una sesión en la base de datos SQLite del dispositivo, realizando una consulta sobre la misma.

En el caso de que se encuentre, se envían las credenciales recogidas al servidor para comprobar que son correctas, del mismo modo que lo hace la actividad Iniciar sesión y gestionando los mismos errores, lo cual se explica en el siguiente apartado. Si el resultado es satisfactorio, se pasa directamente a la pantalla de selección de las bibliotecas, saltándose el paso por la actividad Iniciar sesión.

Si no existe ninguna sesión en la base de datos se pasa a la pantalla de Iniciar sesión para que el usuario pueda loguearse.

Esto permite que el usuario solo tenga que iniciar sesión la primera vez que abre la app.

Esta actividad incorpora un vista muy sencilla con la que no se puede interactuar. El único elemento que se mostrará será el logo de la aplicación (diseñado por los desarrolladores del proyecto para usarlo exclusivamente en el mismo) en el centro de la pantalla. El objeto de esta pantalla es ofrecer al usuario una interfaz mientras se produce el proceso de comprobación que se acaba de explicar, y no desaparecerá hasta que se haya completado. El aspecto de la misma es el que se puede ver en la Figura 47.



Figura 47: Actividad Starter

En el caso de que no se haya podido establecer conexión con el servidor, bien porque no está disponible o porque el dispositivo no dispone de conexión a internet, se notificará por pantalla, como se observa en la Figura 48.



Figura 48: Error en la actividad Starter

7.2.2. Iniciar sesión

Será la primera pantalla al iniciar la aplicación en el móvil. Tendrá la apariencia que se observa en la Figura 49.

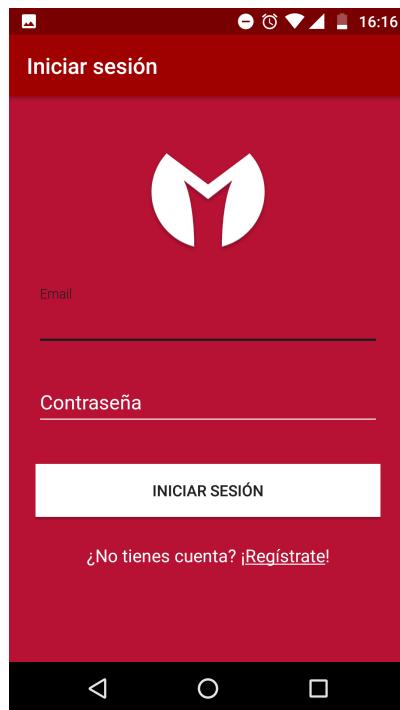


Figura 49: Inicio de sesión en Android

El primer requisito indispensable para poder iniciar sesión es estar registrado, además de haber activado el usuario. Si aún no se ha hecho, podrá hacerse pulsando en el enlace inferior.

El formulario para iniciar sesión dispone tan solo de dos campos, el email y la contraseña. En caso de no completar alguno de ellos se notifica al usuario el error con un mensaje que dice “Completa todos los campos”. Del mismo modo ocurre en el caso de que no exista el usuario en la base de datos o la contraseña no sea la correcta. Si el proceso resulta satisfactorio, se realizará un intent (lanzamiento asíncrono de una nueva actividad) a la página de bibliotecas.

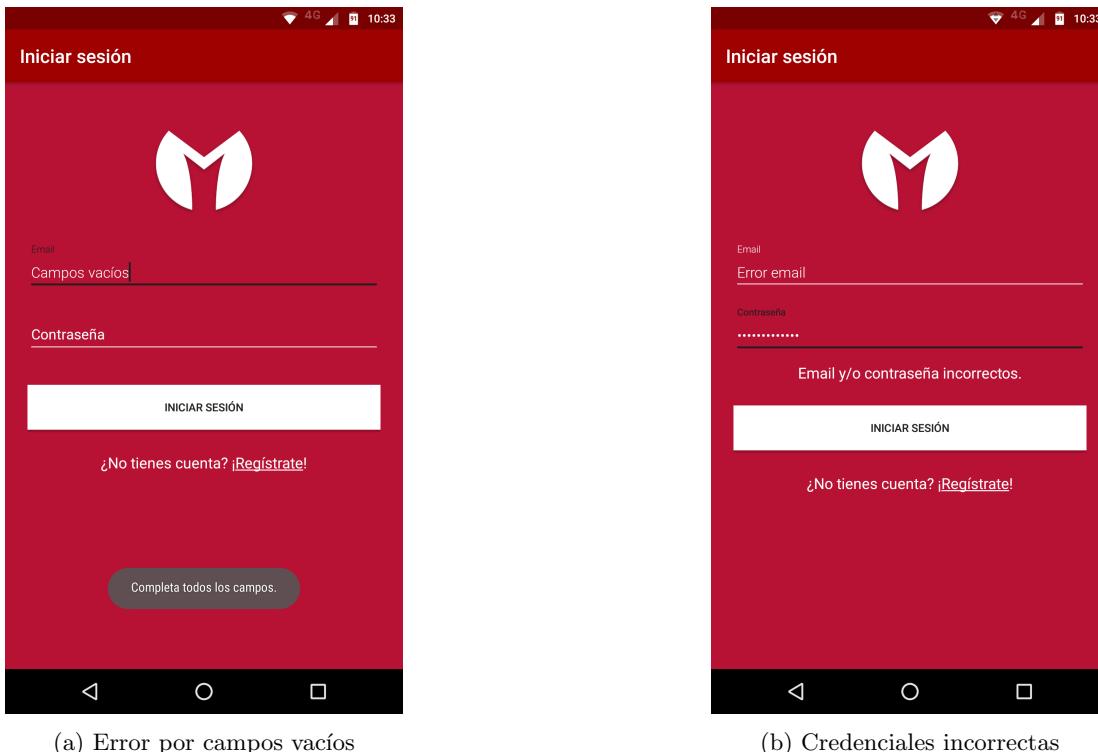


Figura 50: Error al iniciar sesión en la app

En cuanto a la disposición de los elementos se ha optado por colocar el logo de la aplicación en el centro de la página y los dos campos para el inicio de la sesión. Como se puede observar en las diferentes figura de este apartado, una vez clicas en un campo la letra del rotulo disminuye de tamaño, se sitúa en la parte superior y cambia la raya del campo de color blanco a negro, dando feedback instantáneo al usuario de que el campo está seleccionado. Todos los elementos están anclados al centro de la imagen para poder hacer un buen reajustado de ellos en las diferentes dimensiones que tienen las pantallas de los dispositivos Android.

En el encabezado está indicado el título de la actividad actual para guiar al usuario cuando se desplace por las diferentes pantallas de la aplicación (en este caso Iniciar sesión). El enlace a la pantalla de registro está subrayado para aumentar su visibilidad y darle aspecto de enlace.

En cuanto a la implementación, esta tarea se trata de un Activity, debido a que tiene una vista XML asociada, como se puede ver en la carpeta ‘layouts’. En este layout se han utilizado recursos como un editText personalizado para dar la función de minimizar la letra y poder introducir texto y un button para el botón, además de un TextView para indicar el enlace a Registro y un ImageView para insertar el logo.

El archivo Java correspondiente a esta actividad tiene el nombre de MainActivity.

Una vez se haya cargado la vista se esperará a que el usuario haya completado los campos y pulse el botón, entonces se recogerán las credenciales introducidas y se ejecutará un método asíncrono para poder contactar con la base de datos. Dicha clase heredará de AsyncTask, que ofrece varios métodos. En la Figura 51 se muestra un fragmento del código de dicha clase, con

uno de los métodos implementados, y la llamada al php que realiza la comprobación del usuario.

```
private class AttemptLogin extends AsyncTask<String, String, JSONObject> {

    @Override

    protected void onPreExecute() {

        super.onPreExecute();

    }

    @Override

    protected JSONObject doInBackground(String... args) {

        String email = args[0];
        String password = args[1];

        ArrayList<NameValuePair> params = new ArrayList<~>();
        params.add(new BasicNameValuePair( name: "email", email));
        params.add(new BasicNameValuePair( name: "password", password));

        JSONObject json = jsonParser.makeHttpRequest(URL, method: "POST", params);

        return json;

    }

}
```

Figura 51: Fragmento de código de Iniciar Sesión

En el PHP al que se accede mediante el AsyncTask conseguiremos el usuario y la contraseña (hasheada, para no poner en peligro al usuario), y después se comprobará en la base de datos del servidor la existencia de ese usuario, que la contraseña introducida es correcta y sea un usuario activado.

El PHP devolverá a la clase java un objeto JSON en el que se podrá saber si la operación se ha efectuado con éxito, y se notificará al usuario, con un TextView, en el caso de que el usuario no haya sido activado o haya errores en las credenciales introducidas. En caso de que no haya dichos errores se pasará a la siguiente pantalla de la app y quedará guardada la sesión en la base de datos SQLite, para que cuando se abra la aplicación de nuevo, el usuario no tenga que volver a iniciar sesión, ofreciendo de esta manera ventajas considerables de tiempo y usabilidad.

7.2.3. Registrar

Esta funcionalidad permite crear una cuenta a los usuarios que aún no lo hayan hecho. Tan solo se han de llenar 3 campos, el correo electrónico y la contraseña, que será introducida dos veces para que si se comete alguna equivocación al escribirla, el usuario no se registre con una contraseña errónea.

A continuación podemos ver como sería la pantalla de registrar, en la figura 52.

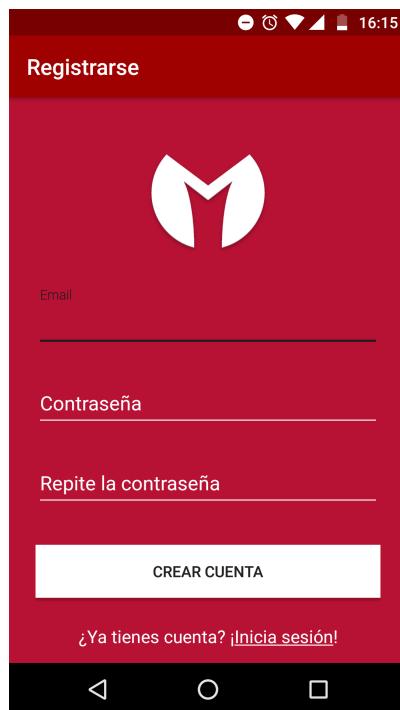


Figura 52: Registro en Android

La única restricción que habrá es que la contraseña debe tener como mínimo de 8 caracteres incluyendo letras y números, haciendo así que la contraseña sea segura. Ambas contraseñas deberán coincidir. También se asegura que el usuario ha elegido un usuario que no existe en la base de datos. A continuación, en la Figura 53 se ven los posibles errores que podrían darse.

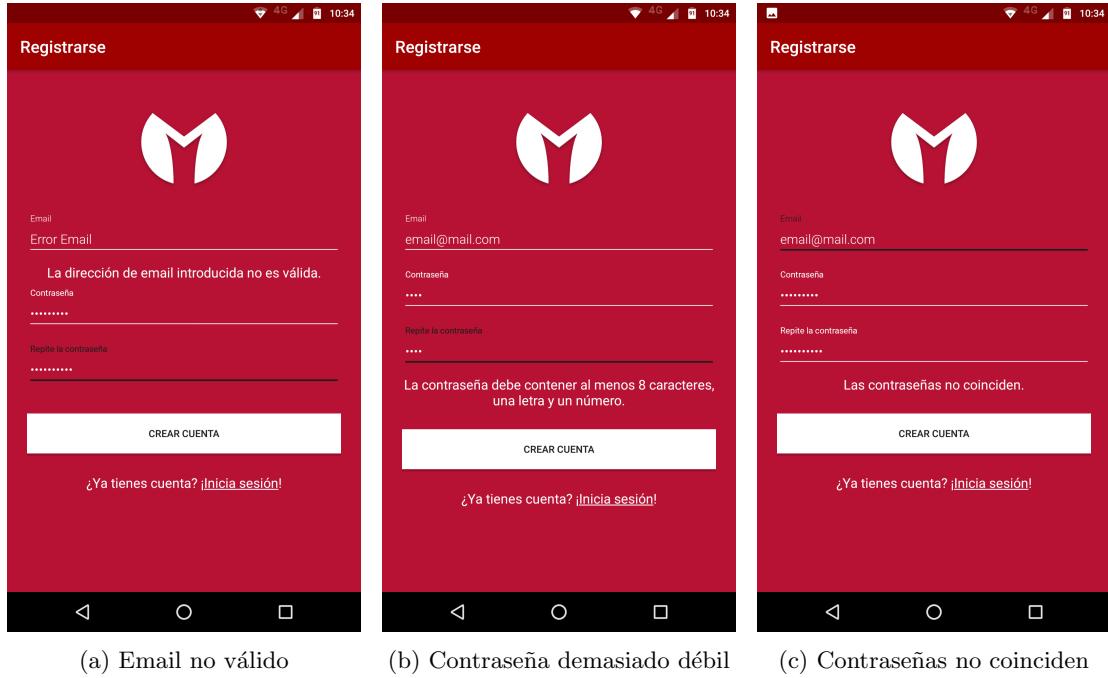


Figura 53: Posibles errores en el registro

En cuanto al diseño empleado, se ha elegido los mismos colores y la misma disposición que en iniciar sesión, promoviendo así una familiaridad con el usuario respecto a las diferentes pantallas. Se podrá volver a la pantalla de inicio sesión mediante el link que aparece en la parte inferior, para que no se produzca el denominado “callejón sin salida en las pantallas”.

Para la implementación de dicha actividad, se ha realizado un archivo XML definido en la carpeta ‘layouts’ definido para esta vista, basado en editText, button y textView.

La clase Java donde se ha implementado esta actividad es Registro. Como en la anterior sección, también se comienza por cargar la vista y esperar a que se rellenen los campos. Una vez llenados y pulsado el botón de registrar, se lanza una clase asíncrona que conectará con la base de datos. Se recuperará el email y la contraseña y se llamará al PHP de registrar, el cuál esperará dichos campos.

Una vez recibidos, comprobará que no el email no existía anteriormente en la base de datos y lo insertará junto con la contraseña en la tabla usuarios. Será justo aquí donde se envíe por correo al usuario un mensaje de activación para que pueda activar su cuenta. El correo que recibirá será como el que aparece en la Figura 54.

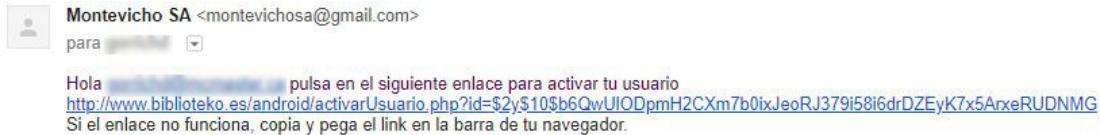


Figura 54: Mensaje para activar el usuario

Al clicar en dicho link, aparecerá la confirmación de que se ha activado la cuenta del usuario, se dará de alta en la tabla de usuarios, y se podrá iniciar sesión con total normalidad.

7.2.4. Bibliotecas

En esta pantalla se muestran las bibliotecas implementadas en la aplicación y se podrá elegir cualquiera de ellas (como ya se sabe, por ahora la única implementada es la de la Facultad de Informática). Tendrá la función de home o pantalla principal ya que, tras iniciar sesión por primera vez, no volvrá a aparecer la pantalla de login si no que se redirigirá directamente a esta pantalla, la pantalla de bibliotecas. Además, es la pantalla a la que se volverá si, estando en cualquiera de las opciones disponibles dentro de cada una de las bibliotecas, se pulsa el botón atrás.

Como se aprecia en la siguiente captura, se trata de una pantalla bastante sencilla en la que se ofrecen todas las bibliotecas disponibles en la app con su dirección y su logo. El resto se encuentran en este ListView a modo de ejemplo para mostrar cómo se vería cuando estuvieran más bibliotecas implementadas.

A pesar de tan solo contar con una biblioteca, el proyecto se ha desarrollado para permitir la implementación de otras bibliotecas. Además, está realizado de un modo que su implementación resulte sencilla, siendo lo más costoso el diseño y esquema de las diferentes salas de la nueva biblioteca a implementar. Tal como se puede apreciar en la Figura 55, este es el diseño que tiene dicha pantalla.



Figura 55: Pantalla de selección de biblioteca en Android

Una vez pulsada la biblioteca deseada, en este caso la de la Facultad de Informática, se redirigirá a la pantalla de la planta principal de la biblioteca. En el caso de la biblioteca de

la FDI, se ha determinado que la planta principal es la planta 1, puesto que es ahí donde se encuentra el mostrador de préstamo. Por tanto, tras pulsar ahí se nos mostrará la primera planta de la biblioteca con sus puestos libres y ocupados.

7.2.5. Plantas

Una vez se seleccione una biblioteca, se accederá al mapa de la biblioteca en cuestión. En este proyecto se ha implementado la de la Facultad de Ingeniería Informática, por lo que aparecerá una representación real de la disposición de los sitios y las mesas de la primera planta.

A la izquierda de la pantalla, oculto y accesible a través de un botón en el encabezado o deslizando la pantalla hacia la derecha, se encuentra el menú principal citado anteriormente, a través del cual se podrá acceder a las diversas funciones implementadas. Esta funcionalidad está implementada en él, ya que si el usuario se encuentra en otra página, como por ejemplo estadísticas, y quiere volver a plantas, puede pulsar en esa función para volver a ver el mapa de la biblioteca. En cuanto al diseño de este apartado en el menú, se ha tratado de utilizar un ícono representativo, el cual son tres láminas superpuestas que aparentan ser las plantas de un edificio y que se ha obtenido de Material Design.

En la esquina inferior derecha de la pantalla se aprecia un botón flotante que permite seleccionar la planta que se quiere consultar (habiéndose 3 plantas en total). Este botón contiene el mismo ícono de las tres láminas utilizado en el menú. Al pulsar en él se despliegan tres botones que identifican a cada una de las plantas, pudiendo pulsar en cualquiera de ellos para elegir una de las tres, como se puede apreciar en la Figura 56.

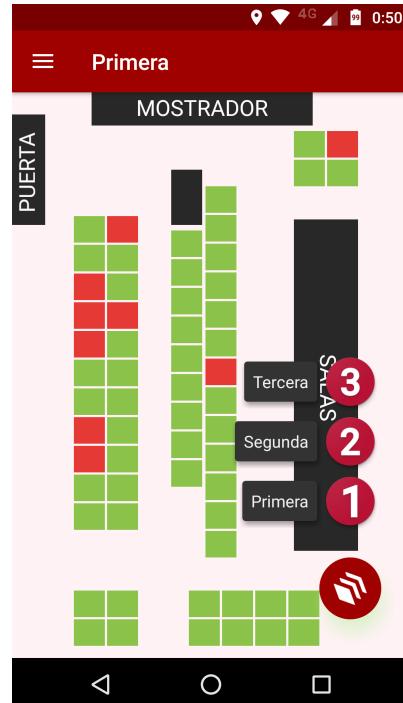


Figura 56: Botón flotante desplegado

Los puestos son representados en la aplicación mediante cuadrados, agrupándose de la manera que mejor convenga para representar las mesas de cada planta. El estado de cada uno de ellos se indica mediante diferentes colores, como se explica a continuación.

Aunque de manera minimalista, cada planta se ha intentado representar con la mayor fielidad posible respecto a la distribución real que presentan en la biblioteca de la facultad. La interfaz posee elementos como PUERTA o MOSTRADOR para poder orientar y situar al usuario de manera que fácilmente pueda identificar y relacionar los puestos de la aplicación con el que le corresponde a cada uno en la planta real.

Los colores elegidos para indicar el estado de los puestos han sido el verde para un puesto que se encuentra libre, el color rojo para señalar que el sitio está ocupado por otro estudiante, el color azul que indica que ese es el puesto que está ocupando el usuario y el naranja para el puesto que está ocupando el usuario mientras realiza un descanso.

A continuación, en la Figura 57 vamos a observar las diferentes plantas, con puestos ocupados y libres, pero sin que el usuario actual haya ocupado ninguno sitio.

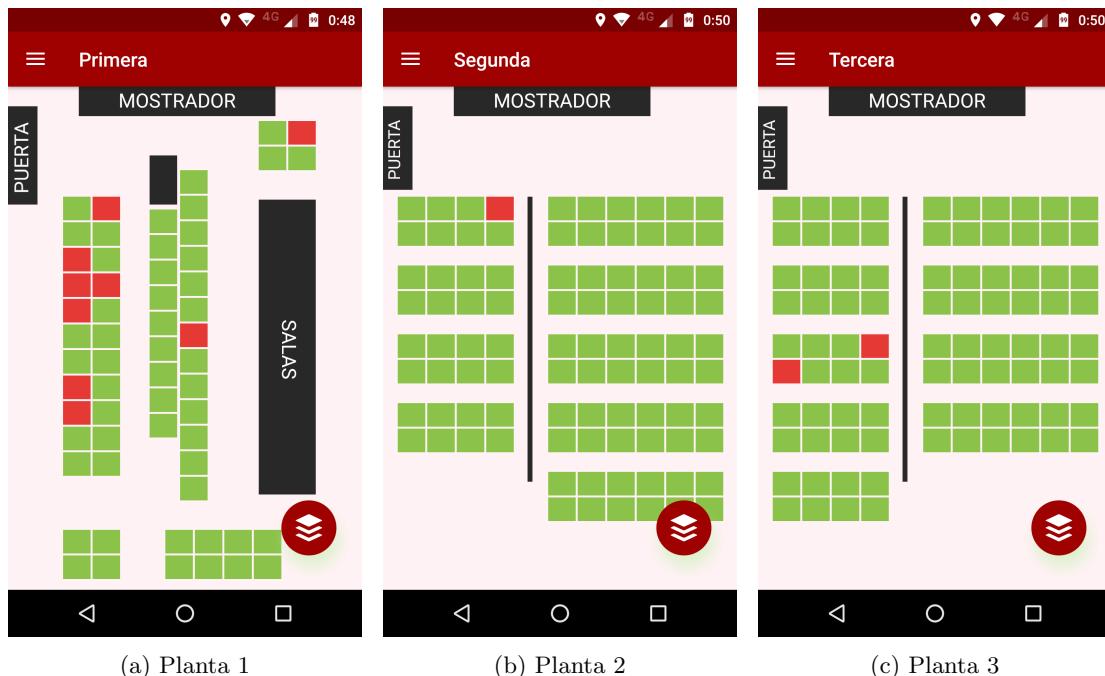


Figura 57: Vista de las plantas de la Biblioteca FDI

En la Figura 58 se observa cómo aparece un puesto ocupado por un usuario y el mensaje en el que se le notifica que se ha ocupado correctamente.

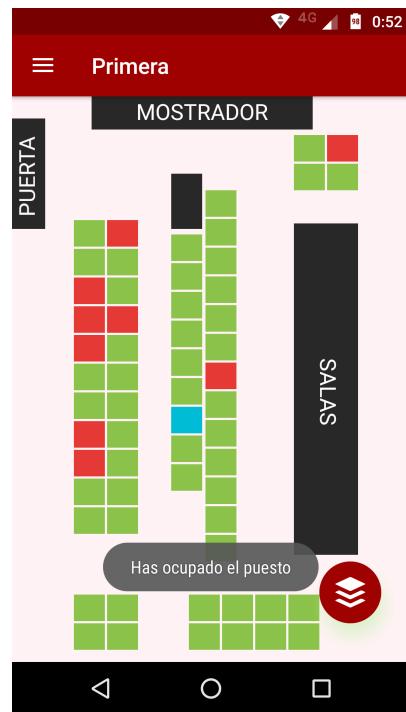


Figura 58: Usuario estudiando

Si el usuario, realiza un descanso mientras tiene un puesto ocupado, el color de su puesto cambiará de azul a naranja (véase Figura 59).

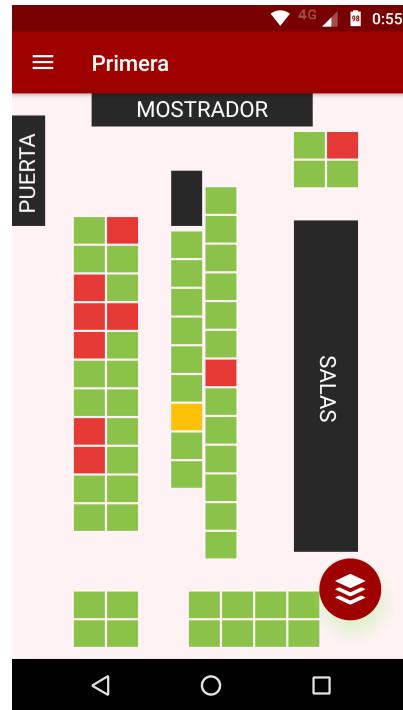


Figura 59: Usuario descansando

La elección de dichos colores tan distintos tiene el propósito de que los usuarios diferencien con facilidad los diferentes estados, a la vez de ayudar en la medida de lo posible a aquellos usuarios que puedan sufrir daltonismo.

El color del fondo es un blanco ligeramente rojizo que combina con la paleta de colores utilizada. El propósito de que sea blanco es no distraer la atención del usuario y que toda su atención vaya dirigida a las mesas y puestos de las plantas.

Es destacable que existen distintos archivos XML para cada una de las plantas, estando dirigido cada uno de ellos a las diferentes dimensiones de las pantallas de los dispositivos móviles. Para cada tamaño y resolución se muestra la vista correspondiente, consiguiendo así que no se descuadren los sitios en distintos dispositivos.

La implementación es similar en las 3 plantas. En cada una, los puestos son imágenes cuadradas que se agrupan para formar mesas y son envueltos por un botón transparente que cubre todos los sitios de una determinada mesa. La intención de este botón es evitar la dificultad que tendría pulsar sobre un puesto de pequeño tamaño y que está rodeado por otros botones, pues la imprecisión en el toque sería alta. Para pulsar en un puesto se pulsará sobre toda la mesa, que se expandirá (iniciando la actividad Mesa) y permitirá seleccionar de manera mucho más sencilla un puesto en concreto.

Para el botón flotante se usa la clase FloatingButton para superponer dicho botón a la página y se añade un menú en la carpeta ‘menu’ con todas las opciones a las que puede clicar el usuario.

En cuanto a la lógica de estas plantas, lo que primero es cargar todo el contenido de la página y a continuación se determinan las mesas que hay en cada planta para dibujarlas (véase Figura 60).

```
String mesa, aux;
for(int i = 0; i < 100; i++)
{
    if (i < 10)
        aux = "00" + i;
    else if (i < 100)
        aux = "0" + i;
    else
        aux = "" + i;

    mesa = "MESA_01_" + aux;
    // si coincide la mesa que acabamos de buscar con alguna que haya dibujada:
    int mesaId = getResources().getIdentifier(mesa, defType: "id", getPackageName());
    if (mesaId != 0)
    {
        //cogemos el puesto inicial de la mesa(ej. 009)
        final String puestoInicial;
        if ( mesa.substring(8).equals("024"))
            puestoInicial="023";
        else
            puestoInicial = mesa.substring(8);

        final Button btnMesa=findViewById(mesaId );
    }
}
```

Figura 60: Fragmento de código en el que se ve el bucle para recorrer las mesas

Por cada mesa, se obtiene el número de plazas, la planta y la biblioteca, y se envía mediante un intent a la clase Mesa para que pueda dibujar la mesa con sus sitios.

Como se puede apreciar en el código, también se hace uso de la clase AsyncTask como anteriormente para poder obtener la información de la biblioteca seleccionada y hacer la consulta correspondiente en la base de datos.

También se transmite al PHP el email del usuario, que obtendremos de la sesión, para poder consultar si tiene algún puesto ocupado en la tabla ocupa y, de ser así, pintar el sitio ocupado. En caso de que sí que tenga un puesto ocupado, aparecerán en el menú las opciones de hacer descanso y liberar puesto, que se desarrollarán en profundidad en las siguientes secciones.

Una vez se haya hecho la consulta con la base de datos para poder saber si el usuario que ha iniciado la sesión tiene un puesto ocupado, se resolverá de la siguiente manera (véase Figura 61).

```

if (!puestoOcupado.equals("NINGUNO") && puestoOcupado.substring(0,6).equals("FDI_01"))
{
    int puestoOcupadoID = getResources().getIdentifier(puestoOcupado, defType: "id",getPackageName());
    if (puestoOcupadoID!=0) {
        final ImageView btnPuestoOcu = findViewById(puestoOcupadoID);
        if(Control_Sesion.getInstancia ( BibliotecaFDIplantal.this ).selectSesion ().getEstado ().equals ( "OCUPANDO_PUESTO" ))
            btnPuestoOcu.setBackgroundDrawable ( ContextCompat.getColor ( context: BibliotecaFDIplantal.this, R.color.colorEstudiando));
        else
            btnPuestoOcu.setBackgroundDrawable ( ContextCompat.getColor ( context: BibliotecaFDIplantal.this, R.color.colorDescansando));
    }
}

```

Figura 61: Fragmento de código en el que se determina si el puesto es del usuario o no

Como se observa, el usuario tiene un puesto ocupado, y debido a que el estado que aparece en la base de datos SQLite del usuario es “OCUPANDO_PUESTO” y no “DESCANSANDO” se pintará en azul. En caso contrario sería naranja. Gracias a los estados implementados en el proyecto se ofrecen a los usuarios diferentes funciones en cada momento, dependiendo de si este es “FUERA_BIBLIOTECA”, “DENTRO”, “OCUPANDO_PUESTO” o “DESCANSANDO”.

7.2.6. Mesa

Tras pulsar en una de las mesas de las plantas de la biblioteca, se llega a esta pantalla. En ella se puede ver más detalladamente los puestos que hay, cuáles están disponibles y se puede proceder a ocupar un puesto (si no se tenía ya uno ocupado) o a liberar el que se está ocupando.

En cada mesa se ve un rectángulo negro, que representa la misma, y los puestos alrededor de ella. Hay múltiples tipos de mesas, de diferentes tamaños. Para cada una de ellas, se ha realizado un XML diferente en el que se han situado los puestos manualmente, dependiendo de cómo fuera la distribución de la mesa (véase Figura 62).

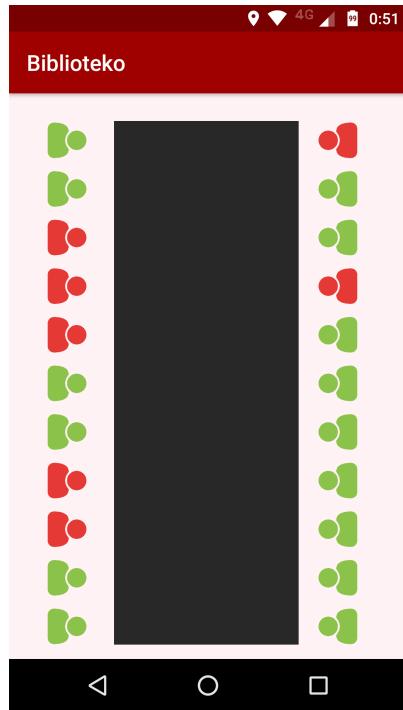


Figura 62: Pantalla mesa en Android

Se trata de una de las pantallas que más problemas planteó durante su desarrollo desde el punto de vista lógico. No era lógico tener que realizar una clase Java para cada una de las mesas disponibles en la biblioteca. De modo que se buscó un método que permitiera, con una única clase Java, desarrollar todas las mesas. Finalmente, y combinado con los valores que se obtienen al llamar a Mesa desde Plantas, se logró que cada puesto tuviera su ID correspondiente.

En primer lugar, cada vez que se produce la creación de una nueva mesa, se obtienen los valores extras que han sido establecidos en la llamada desde Plantas. En este caso, los primeros valores que se recogen son número de plantas y tipo de mesa. Con estos valores ya es posible llamar a una de las vistas XML, la que corresponda a la mesa en cuestión. Una vez dibujada la mesa, se obtienen el resto de datos, como son la biblioteca (sus siglas), la planta (su ID) y el puestoInicial. Este último dato es de vital importancia puesto que es el dato con el que, en combinación con planta y biblioteca, se determina el puesto en concreto del cual se quiere conocer su estado.

Tras esto, y puesto que también se conoce el número de puestos que hay en esa mesa, se genera una consulta (véase Figura 63) que nos devuelva el estado de esos puestos, si están ocupados o no. Una vez hecho esto y según estén libres u ocupados, se establece un listener (véase Figura 64) determinado para cada uno de ellos, en el que se introducen como datos la biblioteca, la planta y el ID del puesto entre otros.

Llegados a este punto, hay varias formas de interactuar con los puestos. Se puede pulsar sobre uno que está libre para ocuparlo (si se encuentra dentro de la biblioteca) o pulsar sobre el que se está ocupando actualmente para desocuparlo y que vuelva a quedar libre. Cabe indicar que, al igual que ocurre en la vista de las plantas, y como se puede observar en la imagen, el estado de los puestos se identifica por los mismos colores que anteriormente, siendo el rojo para un puesto ocupado, verde si se encuentra libre, azul el que se está ocupando en ese momento y naranja si se está realizando un descanso.

Al pulsar un puesto libre para ocuparlo, se realiza un intent a la actividad OcuparPuesto que se encargará de realizar los cambios necesarios a nivel local y a nivel de servidor para que quede registrado que ese puesto pasará a ser ocupado, como se explicará en detalle en la siguiente sección. Del mismo modo ocurrirá al pulsar el puesto que se está ocupando para poder liberarlo, llamando a la actividad VaciarPuesto, además de mostrar un Dialog para confirmar la acción. En ambos casos se enviarán dentro del intent los datos correspondientes al puesto pulsado, que serán la biblioteca a la que pertenece, la planta en la que se encuentra, el id del puesto, y en el caso de querer ocuparlo, los valores de la latitud y la longitud donde se encuentra el usuario, obteniéndolos desde la base de datos SQLite del dispositivo, para que la actividad correspondiente pueda saber el puesto sobre el que se quiere efectuar la acción.

```

String puestoInicial = args[0];
String plazas = args[1];
String biblioteca = args[2];
String planta = args[3];
String latitud = args[4];
String longitud = args[5];
int puesto = Integer.parseInt(puestoInicial);
int plazasNum = Integer.parseInt(plazas);
int i;
String puestoString;
String tag = biblioteca + " " + planta + " ";
String miSql = "SELECT ID FROM `puestos` WHERE ocupado = 0 and (";
for (i = 0; i < plazasNum; i++)
{
    if (puesto < 10)
        puestoString = "00" + puesto;
    else if (puesto < 100)
        puestoString = "0" + puesto;
    else
        puestoString = "" + puesto;
    if(i < plazasNum - 1)
        miSql = miSql + "ID = '" + tag + puestoString + "' or ";
    else
        miSql = miSql + "ID = '" + tag + puestoString + "'";
    puesto = puesto + 1;
}
ArrayList<NameValuePair> params = new ArrayList<~>();
params.add(new BasicNameValuePair( name: "biblioteca", biblioteca));
params.add(new BasicNameValuePair( name: "planta", planta));
params.add(new BasicNameValuePair( name: "puestoInicial", puestoInicial));
params.add(new BasicNameValuePair( name: "plazas", plazas));

```

Figura 63: Query consulta de puestos de la mesa

```
for(int i=0; i < puestos.length; i++){
    final String latitud = result.getString("latitud");
    final String longitud = result.getString("longitud");
    final String puestoActual = puestos[i];

    int puestoLibre = Integer.parseInt(puestos[i].substring(7));
    String idPuestoLibreMesa = biblioteca + "_" + plazas + "_" + (puestoLibre - puestoInicial);

    int resId = getResources().getIdentifier(idPuestoLibreMesa, defStyle: "id", getPackageName());
    final Button btnpuesto=findViewById(resId);
    btnpuesto.setBackgroundTintList (ColorStateList.valueOf(ContextCompat.getColor ( context: Mesa.this, R.color.colorVerde)));
}

//Si no hay ningún puesto ocupado por el usuario actual, establecemos los listener para los libres
if(puestoOcupado.equals ("NINGUNO"))
{
    btnpuesto.setOnClickListener((view) → {
        Intent intentOcupar= new Intent ( packageContext: Mesa.this,OcuparPuesto.class);
        myVib.vibrate( milliseconds: 50);
        intentOcupar.putExtra( name: "biblioteca", biblioteca);
        intentOcupar.putExtra( name: "planta", planta);
        intentOcupar.putExtra( name: "puestoAOcupar", puestoActual);
        intentOcupar.putExtra( name: "latitud", String.valueOf((Control_Sesion.getInstancia(Mesa.this).selectSesion().ge
        intentOcupar.putExtra( name: "longitud", String.valueOf((Control_Sesion.getInstancia(Mesa.this).selectSesion().ge
        Mesa.this.startActivity(intentOcupar);
        finish();
    });
}
//Si no, establecemos los listener pero informamos que no lo puedes ocupar porque ya tienes uno ocupado
else
{
    btnpuesto.setOnClickListener((view) → {
        Toast.makeText(getApplicationContext(), text: "Ya tienes un puesto ocupado", Toast.LENGTH_SHORT).show();
    });
}
```

Figura 64: Listener sobre los puestos

Es así como se consigue que con una sola clase Java se pueda acceder correctamente a todos los puestos y consultar su estado, independientemente de a qué planta y biblioteca pertenezcan.

7.2.7. Ocupar puesto

Esta actividad será la encargada de realizar la funcionalidad de ocupar un puesto. Se llegará hasta a ella a través de un intent lanzado en la clase Mesa, y de él se extraerán los datos del puesto que se quiere ocupar, los cuales son la biblioteca a la que pertenece, la planta en la que se encuentra, el id del puesto y los valores de latitud y longitud actuales del dispositivo.

Tras almacenar estos datos en variables, se procede a ejecutar el método execute de la clase interna Ocupar definida en esta actividad. Esta clase hereda de AsyncTask y gracias a ella se puede establecer una conexión con el servidor en segundo plano. De esta manera, la aplicación espera a recibir los resultados del servidor y en función de los mismos realiza la acción conveniente.

Esta clase consta principalmente de dos métodos: `doInBackground` y `onPostExecute`. El primero de ellos se encarga de preparar la conexión con el servidor. Se inicializará un `ArrayList` en el que se introducen parejas clave-valor con los atributos del puesto obtenido anteriormente del intent, además del email del usuario que quiere ocupar el puesto, el cual se consigue de la base de datos SQLite del dispositivo. A continuación, se conecta con un archivo PHP que ocupará el puesto si es posible.

El PHP, llamado ocuparPuesto.php, realiza una serie de comprobaciones para verificar que se cumplen todas las condiciones necesarias para que se pueda ocupar un puesto. Esto se consigue mediante una secuencia de if-else en los que cada if son un requisito a superar para poder realizar la operación con éxito.

La primera de estas comprobaciones es consultar la tabla ocupa de la base de datos SQL del servidor para cerciorarse de que no existe ningún puesto ocupado por ese usuario. Si esto se cumple, se pasa a comprobar en la tabla puestos que el sitio que se quiere ocupar no lo está ya por ningún otro usuario. El tercer y último requisito es que el usuario se encuentre dentro de la biblioteca, y fue el que más costó resolver. Para ello, se obtienen las coordenadas de las biblioteca en cuestión, obteniéndolas de la base de datos, y mediante un algoritmo de código abierto y licencia libre llamado PIP[18] (punto en polígono) se comprueba que las coordenadas que llegan de la conexión con la aplicación móvil se encuentran dentro del recinto establecido por la biblioteca. Si esto se cumple, se alcanza el cuarto y último paso, en el que se actualiza la base de datos y se asigna al puesto el estado de ocupado. Para ello, hay que dejar constancia en dos tablas: en la tabla puestos, donde se asigna al campo ocupado del puesto en cuestión el valor ‘true’, que se designa con el entero ‘1’; y en la tabla ocupa, donde se inserta una fila cuyos valores serán el email del usuario, el id del puesto que está ocupando, y el campo booleano dentro a ‘1’, que denota que se encuentra dentro de la biblioteca (puede ser ‘0’ cuando el puesto está reservado para el usuario pero este se encuentra fuera de la biblioteca realizando un descanso).

Finalmente, el PHP (véase Figura 65) devolverá un objeto JSON a la aplicación móvil con un entero que indicará si la operación se ha efectuado con éxito o a habido algún error, indicando cuál ha sido ese error mediante distintos enteros que se gestionarán en la clase Java al recibir este dato.

```

if(!empty($biblioteca) && !empty($usuario) && !empty($puestoAOcupar) && $latitud != null && $longitud != null)
{
    $query = "SELECT * FROM `ocupa` WHERE email = '$usuario' ";
    $consulta=mysqli_query($db,$query);
    if(mysqli_num_rows($consulta) == 0)
    {
        $query2 = "SELECT * FROM `puestos` WHERE ID = '$puestoAOcupar' and ocupado = 1";
        $consulta2=mysqli_query($db,$query2);
        if(mysqli_num_rows($consulta2) == 0)
        {
            $query3 = "SELECT * FROM `bibliotecas` WHERE siglas = '$biblioteca' ";
            $consulta3=mysqli_query($db,$query3);
            if(mysqli_num_rows($consulta3) > 0)
            {
                $coordBib=mysqli_fetch_object($consulta3);
                $pointLocation = new pointLocation();
                $A = $coordBib->NO_Lon . " " . $coordBib->NO_Lat;
                $B = $coordBib->NE_Lon . " " . $coordBib->NE_Lat;
                $C = $coordBib->SE_Lon . " " . $coordBib->SE_Lat;
                $D = $coordBib->SO_Lon . " " . $coordBib->SO_Lat;
                $poligono = array($A, $B, $C, $D, $A);
                $punto = $longitud . "," . $latitud;
                $dentroOfuera = $pointLocation->pointInPolygon($punto, $poligono);
                if ($dentroOfuera != "outside")
                {
                    $query4 = "UPDATE `puestos` SET `ocupado` = '1' WHERE `puestos`.`ID` = '$puestoAOcupar' ";
                    $consulta4=mysqli_query($db,$query4);
                    $query5 = "INSERT INTO `ocupa`(`email`, `puesto`, `temporizador`, `dentro`) VALUES ('$usuario', '$puestoAOcupar', '0', '1') ";
                    $consulta5=mysqli_query($db,$query5);

                    $json_array['success'] = 1;
                    echo json_encode($json_array);
                }
            }
        }
    }
}
else
{
    $json_array['success'] = 2;
}

```

Figura 65: PHP de ocupar puesto

El resultado que devuelve el archivo PHP es capturado en el segundo de los métodos principales que contenía la clase Ocupar: onPostExecute. En él, tras comprobar que el objeto JSON no se ha devuelto vacío, se mira el valor del entero que contiene. Si es el ‘1’ significa que la tarea se ha resuelto satisfactoriamente, y se procederá a actualizar la base de datos SQLite que es la que controla el ciclo de estados del usuario. Se actualiza su estado a “OCUPANDO_PUESTO”, al campo puestoOcupado se le asigna el id del puesto y se guarda el momento en que ha sido ocupado el puesto mediante un objeto Timestamp. Este es un paso muy importante ya que a partir de este dato se podrá saber cual ha sido el tiempo que el usuario a estado estudiantan-

do (en modo “OCUPANDO_PUESTO”), y esto servirá más adelante para poder establecer el tiempo de descanso y para insertar una nueva fila en la tabla histórico de la base de datos del servidor. Por último, se observa un toast que dice que se ha podido ocupar el puesto correctamente.

Si el resultado obtenido en el JSON no ha sido favorable, se muestra por pantalla cuál ha sido el causante del error, que se identifica por el valor entero que contiene.

Tras finalizar todo este proceso, se volverá directamente a la planta en la que se encuentra el puesto que se ha tratado de ocupar. Si ha sido posible se mostrará en azul, gracias al método onResume implementado en cada una de las plantas. Si no ha sido así, el puesto permanecerá en el estado en el que se encontraba.

7.2.8. Vaciar puesto

Esta funcionalidad aparecerá en el menú lateral una vez el usuario haya ocupado un puesto. Para hacerlo ha debido de pasar las restricciones como se ha visto en el capítulo anterior.

En caso de no haber ocupado un puesto, no se mostrará al usuario esta funcionalidad, ya que sería un error, porque no puede liberar un puesto que no posee. A continuación, en la Figura 66, se puede ver cómo se muestra el menú, sin haber ocupado un puesto.

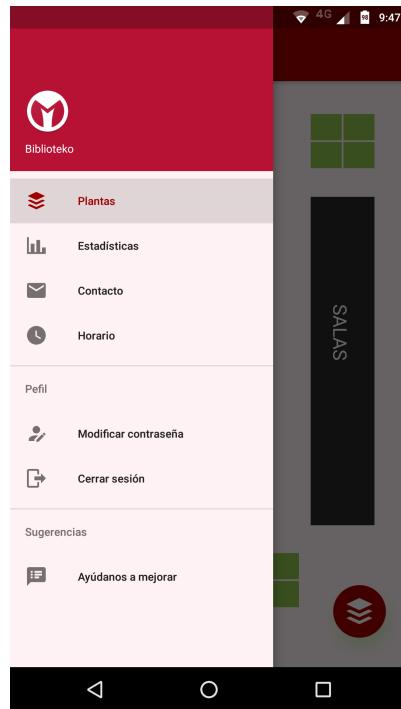


Figura 66: Menú de Android sin haber ocupado un puesto

Al ocupar un puesto, aparecerán automáticamente en el menú dos nuevas opciones (véase Figura 67), *hacer descanso* y *liberar puesto*. Esta sección se centra en *liberar puesto* y la desarrollará en profundidad, la otra opción se verá en la sección siguiente.

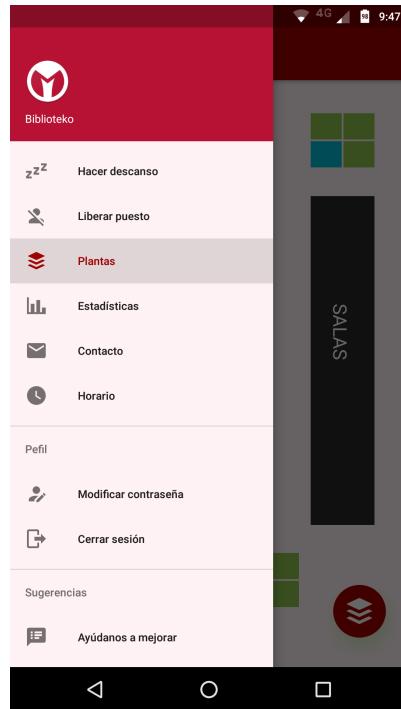


Figura 67: Menú de Android tras ocupar un puesto

Para liberar un puesto, no solo se puede hacer desde el menú, si no que también es posible ir a la mesa en la que estamos ocupando el puesto y clicar en el sitio ocupado. En ese caso saldrá la misma alerta que si se hubiese dado a liberar puesto en el menú. Se preguntará si se desea confirmar dicha operación, y se liberará el puesto correctamente. Este cuadro de confirmación, que se puede ver a continuación, está implementado para confirmar con el usuario que es esa operación la que quiere realizar, ya que si desocupa el puesto perderá su tiempo acumulado para descansar. Es por tanto necesario asegurarse (véase Figura 68) de que el usuario le ha dado a propósito y que quiere terminar su estudio.



Figura 68: Mensaje de confirmación de liberar puesto

El ícono del menú pertenece a Material Design. Se dispone de la imagen en diferentes tamaños, colores y resoluciones, para que se pueda ajustar a los distintos móviles. Al igual que en el resto de aspectos de la interfaz, todos los iconos de las funciones tienen la misma tonalidad de color, para así poder mantener una misma gama de color y homogeneidad. Se ha pretendido que el ícono sea representativo y se asemeja a la opción de liberar, para representar que el usuario quiere abandonar el puesto, haciendo que sea intuitivo con tan solo con ver el dibujo.

En cuanto a la implementación de dicha funcionalidad, se acopló una fila más en el menú del XML (se puede ver en el archivo *activity_biblioteca_fdiplanta1_drawer.xml*).

Por otro lado, en cada Java en los que aparece el *listener* de cada botón del menú, se añadió la funcionalidad de liberar puesto (véase Figura 69).

```
else if (id==R.id.liberar){
    android.support.v7.app.AlertDialog.Builder builder = new android.support.v7.app.AlertDialog.Builder( context: BibliotecaFDIplantal.this)

    builder.setMessage("¿Estás seguro de que quieres liberar tu puesto?")
    .setTitle("¿Liberar puesto?")
    .setIcon(R.drawable.ic_alert_black_48dp)
    .setPositiveButton( text: "Sí", (dialog, which) -> {
        Intent intentLiberar = new Intent( packageContext: BibliotecaFDIplantal.this, VaciarPuesto.class);
        BibliotecaFDIplantal.this.startActivity(intentLiberar);
    })
    .setNegativeButton( text: "No", (dialog, which) -> {
        itemLiberar.setChecked(false);
        itemPlantas.setChecked(true);
    })
    .show();
}
```

Figura 69: Fragmento de código - Listener liberar puesto

Y para poder ocultar dicha función se ha utilizado la función `itemLiberar.setChecked(false)`, donde itemLiberar se refiere al botón de liberar puesto, y con esto se oculta dicho apartado del menú. Si el usuario de la sesión tiene ocupado un puesto se habilita haciendo el opuesto de dicha función (es decir, poniéndolo a true). A continuación, en la Figura 70, vemos el código que realiza dicha función.

```
if(Control_Sesion.getInstancia ( BibliotecaFDIplantal.this ).selectSession ().getEstado ().equals ( "OCUPANDO_PUESTO" )) {  
    itemDescansar.setVisible ( true );  
    itemDescansar.setTitle ( "Hacer descanso" );  
    itemLiberar.setVisible ( true );  
    itemLiberar.setTitle ( "Liberar puesto" );  
}
```

Figura 70: Fragmento de código - Habilitar opciones

Una vez hayamos clicado en desocupar el puesto haremos un intent al Java de VaciarPuesto. En él, a partir de la sesión del usuario, conseguiremos el puesto que está ocupando y el usuario en cuestión, la fecha de inicio del estado y el tiempo que ha estado estudiando. Con todo eso se conecta con el PHP para, en primer lugar, eliminar dicha fila de la tabla ocupa y así que se muestre desocupado el sitio en el mapa, y para, posteriormente, actualizar el histórico y así añadir nueva información del tiempo de estudio de dicho usuario a sus estadísticas personales y globales.

Una vez hecho esto se reinicia el estado del usuario a “FUERA_BIBLIOTECA”, hasta que consiga de nuevo la ubicación del usuario y comience de nuevo el ciclo de estados del usuario.

7.2.9. Hacer descanso/Finalizar descanso

Ambas funciones aparecerán (una u otra), una vez el usuario haya ocupado un sitio, como se ha visto en el capítulo anterior. Está diseñada e implementada de la misma manera que antes, es decir, se ha añadido a los XML una nueva fila en la que aparece “Hacer descanso” o “Finalizar descanso”, cuando la sesión del usuario marque un puesto ocupado. En caso de no tener puesto no aparecerá dicha opción.

En la Figura 66 de la sección anterior se puede observar como no aparece ninguna de estas opciones si no se está ocupando un puesto, mientras que en la Figura 67 se ve que tras haberlo ocupado, se ha añadido la opción “Hacer descanso”. En cambio, aparecerá la opción “Finalizar descanso” si está realizando un descanso, como se aprecia en la Figura 71.

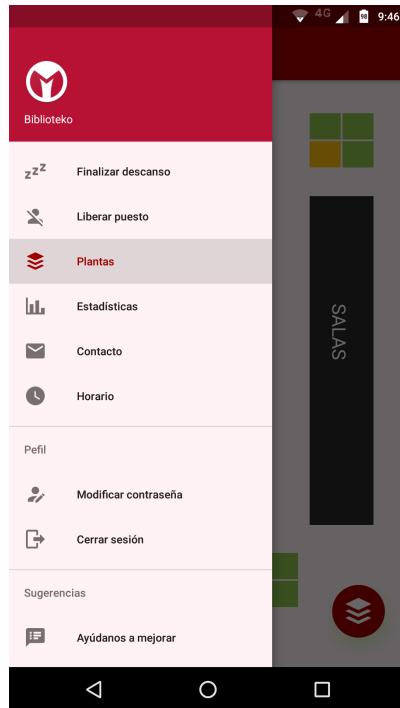


Figura 71: Menú con usuario descansando

En el caso de que se ocupase un puesto y se esté estudiando, si se llevaran unos 22 minutos estudiando y se quisiera realizar un descanso (para tomar un café, por ejemplo), se le puede dar a esa opción del menú. En ese momento (véase Figura 72) saldrá el tiempo acumulado estudiando (en este caso 22 minutos que se puede desocupar el puesto sin perderlo).

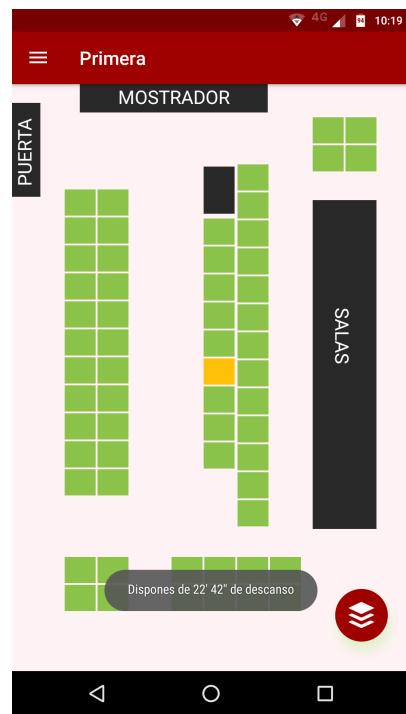


Figura 72: Mensaje que avisa sobre tiempo de descanso disponible

Automáticamente, el puesto cambiará de azul a naranja, para indicar que se está descansando.

Si se sobrepasa este tiempo de descanso, se liberará el puesto automáticamente si el usuario se encuentra fuera de la biblioteca, o volverá a mostrarse ocupado si se encuentra dentro. Pero para que el usuario sepa cuánto tiempo le queda antes de que finalice su descanso, saldrá una notificación (véase Figura 73) 3 minutos antes de que acabe su tiempo de descanso disponible advirtiéndole de ello.

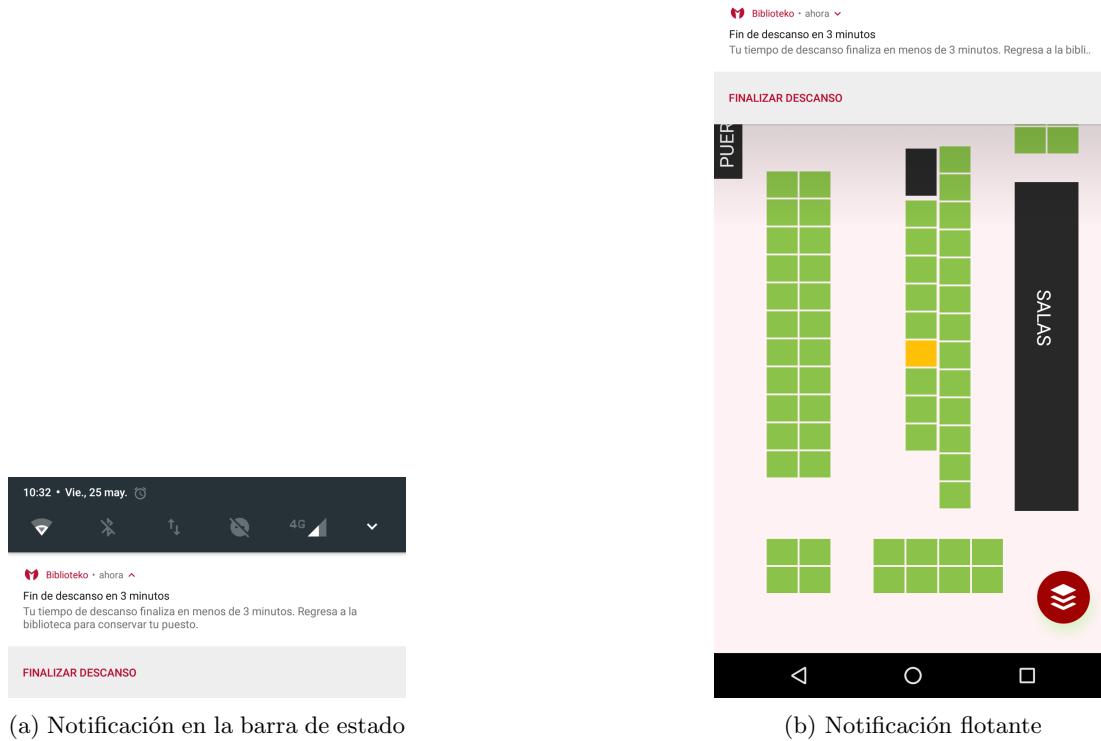


Figura 73: Mensaje de 3 minutos restantes para fin de descanso

Si se pulsa en “Finalizar descanso” antes de que este termine, se acumulará el tiempo sobrante para el siguiente descanso y se le informará de cuál es ese tiempo acumulado mediante un mensaje en la pantalla (véase Figura 74).

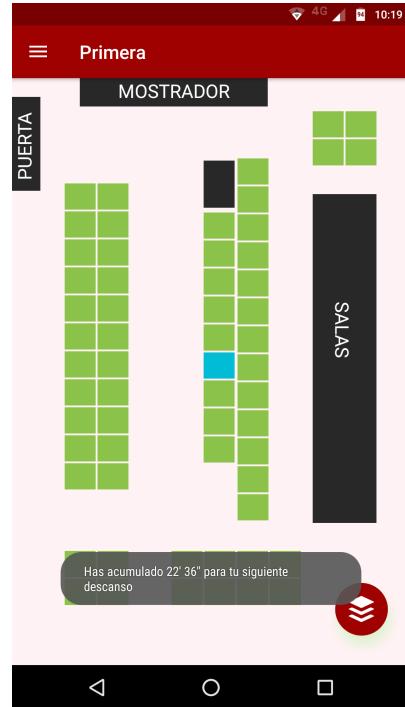


Figura 74: Mensaje que avisa sobre tiempo acumulado para el siguiente descanso

Se puede dar la situación en la que el usuario se encuentre dentro de la biblioteca ocupando un puesto y sin hacer un descanso ni liberar el puesto salga de ella. Ante esto, se pasa directamente su estado a “DESCANSANDO” para que no pierda su sitio, comenzando la cuenta atrás de su tiempo de descanso disponible. Se le notificará esta situación y se le da la opción de liberar su puesto, para evitar tener un puesto ocupado por nadie si su intención es abandonar la biblioteca definitivamente (véase Figura 75).

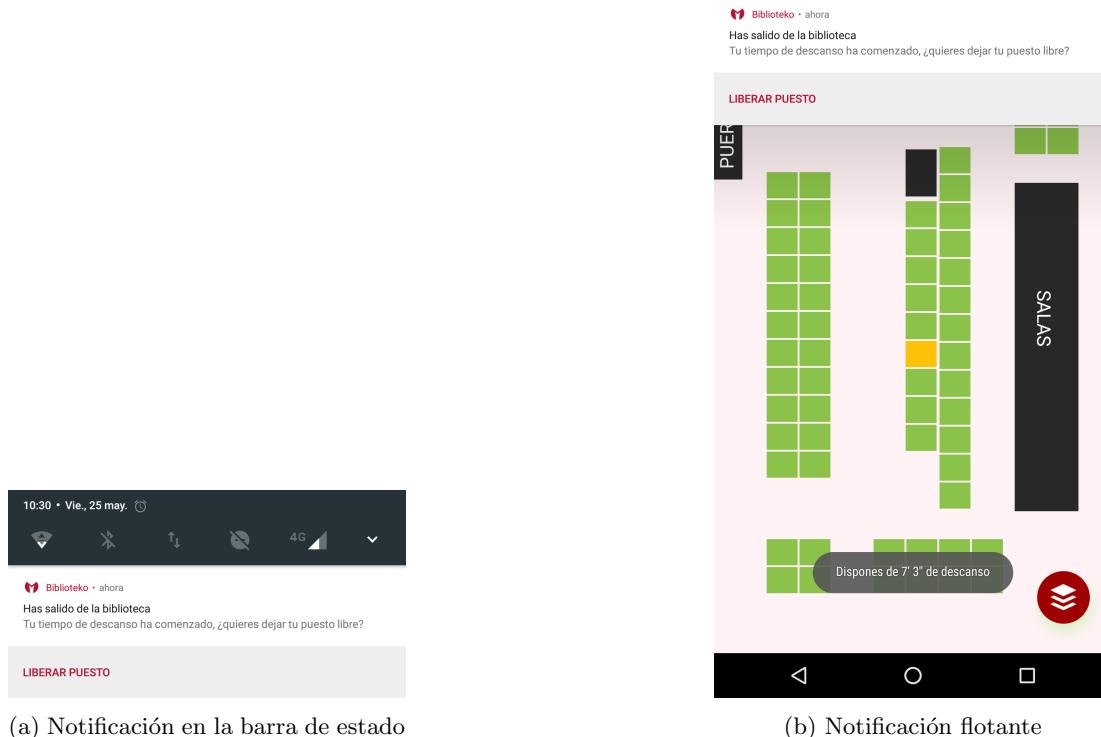


Figura 75: Notificación ‘Has salido’

En el caso de que el usuario se encuentre descansando fuera de la biblioteca y se detecte que vuelve a entrar a ella también se lo notificará para que, si quiere volver al estudio, finalice el descanso y acumule de esta manera el tiempo de descanso que le ha sobrado (véase Figura 76).

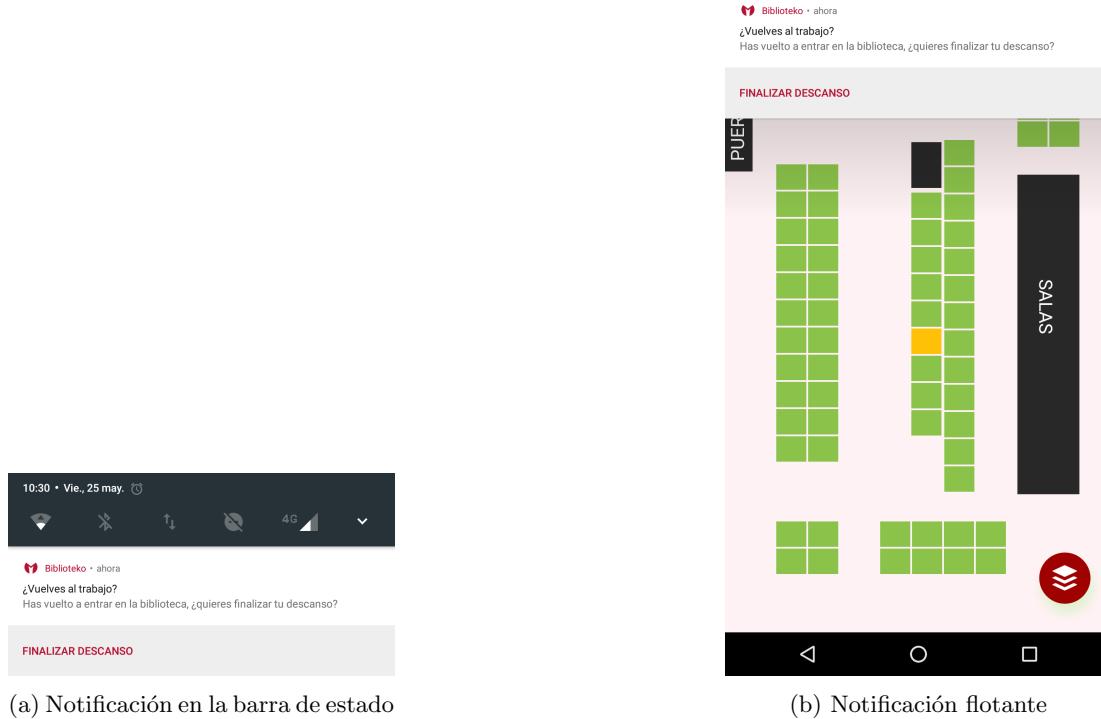


Figura 76: Notificación ‘Has vuelto’

En cuanto al diseño de estas dos nuevas filas del menú, han sido implementadas de igual forma que en la sección anterior, por lo que no se explicará de nuevo.

En cuanto al Java, para poder desarrollar todas las funciones, se ha implementado la clase **Descansar**, la cuál será llamada una vez el usuario haya clicado en la función de “Hacer descanso” o “Finalizar descanso”. Se obtendrá el email del usuario de la sesión junto al tiempo que ha estado estudiando/descansando y comprobaremos su estado para así determinar si antes estaba estudiando y ahora quiere realizar el descanso o por lo contrario, si estaba descansando y ahora va a seguir estudiando.

Se verá el primer lugar el caso en el que el usuario está estudiando y quiere realizar un descanso. Su estado cambiará a “DESCANSANDO”, se actualizará su tiempo de descanso acumulado, que será la diferencia del momento actual con el momento en el que inició el estudio, y se iniciará una alarma que saltará 3 minutos antes de que termine el descanso y lo volverá a hacer cuando finalice totalmente. Este hecho se puede apreciar en la Figura 77.

```

if(Control_Sesion.getInstancia(Descansar.this).selectSesion().getEstado ().equals ( "OCUPANDO_PUESTO" )){
    estudiando = "1";
    int tiempoDisponible = Integer.parseInt ( duracion ) + usuario.getDuracionAcumulada ();

    if(tiempoDisponible > 1800)
        tiempoDisponible = 1800;

    usuario.setEstado ( "DESCANSANDO" );
    usuario.setInicioEstado ( new Timestamp ( System.currentTimeMillis () ) );
    usuario.setDuracionAcumulada ( tiempoDisponible );
    Control_Sesion.getInstancia(Descansar.this).updateSesion ( usuario );
    alarmMgr = (AlarmManager)Descansar.this.getSystemService(Context.ALARM_SERVICE);
    Intent intent = new Intent( packageContext: Descansar.this, Alarma.class);
    alarmIntent = PendingIntent.getBroadcast( context: Descansar.this, requestCode: 0, intent, flags: 0 );
    alarmIntent2 = PendingIntent.getBroadcast( context: Descansar.this, requestCode: 1, intent, flags: 0 );
    |
    Calendar calendar = Calendar.getInstance();
    Calendar calendar2 = Calendar.getInstance();
    Timestamp antes= new Timestamp (calendar.getTimeInMillis());

    calendar.setTimeInMillis(System.currentTimeMillis())+((tiempoDisponible*1000)-(180*1000));
    calendar2.setTimeInMillis(System.currentTimeMillis())+(tiempoDisponible*1000);
    Timestamp despues= new Timestamp (calendar.getTimeInMillis());
    alarmMgr.setRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), intervalMillis: 0, alarmIntent);
    alarmMgr.setRepeating(AlarmManager.RTC_WAKEUP, calendar2.getTimeInMillis(), intervalMillis: 0, alarmIntent2);
}

```

Figura 77: Fragmento de código - Inicio de descanso

Si por el contrario el usuario estaba descansando y quiere volver a estudiar, se comprobará el tiempo que le ha sobrado del descanso y se establecerá como nuevo tiempo acumulado. Su estado cambiará a “OCUPANDO_PUESTO” (véase Figura 78) y actualizará la hora del cambio del estado.

```

else if(Control_Sesion.getInstancia(Descansar.this).selectSesion().getEstado ().equals ( "DESCANSANDO" )){
    estudiando = "0";
    int tiempoSobrante = usuario.getDuracionAcumulada() - (int) ((System.currentTimeMillis () - usuario.getInicioEstado ().getTime ()) / 1000);

    if(tiempoSobrante < 0)
        tiempoSobrante = 0;

    usuario.setEstado ( "OCUPANDO_PUESTO" );
    usuario.setInicioEstado ( new Timestamp ( System.currentTimeMillis () ) );
    usuario.setDuracionAcumulada ( tiempoSobrante );
    Control_Sesion.getInstancia(Descansar.this).updateSesion ( usuario );
}

```

Figura 78: Fragmento de código - Fin de descanso

A continuación, se llamará al PHP descansar.php, para poder actualizar el la tabla histórico del servidor añadiendo una nueva fila con el tiempo estudiando o descansando. Una vez vuelva del PHP, se notificará al usuario del tiempo que dispone para descansar o el tiempo que le ha sobrado del descanso.

7.2.10. Estadísticas

Otra de las funciones de las que dispondrá el usuario de la aplicación es la de poder ver sus propias estadísticas, tanto de tiempo de estudio que ha realizado personalmente, como por todos los alumnos que hayan utilizado la aplicación, además de poder ver las estadísticas de puestos ocupados a tiempo real de la biblioteca.

Al pinchar en dicha función se mostrarán las estadísticas de la ocupación real de la biblioteca. Existen las pestañas que representan las plantas de la biblioteca, para poder observar la ocupación de cada planta. A continuación, en la Figura 79 se puede ver un ejemplo del gráfico que se muestra.

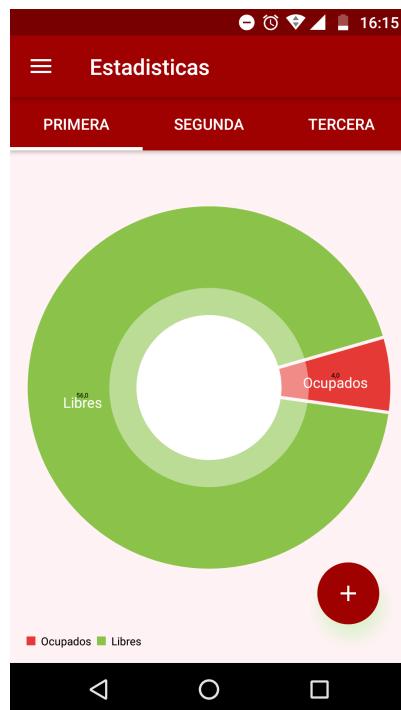


Figura 79: Estadísticas de ocupación de puestos

Al igual que en las plantas, se dispone de un botón flotante pero este contará diferentes funcionalidades al de las plantas, debido a que ahora las opciones que saldrán será para poder ver las estadísticas personales o las globales.

El usuario podrá observar sus estadísticas personales (véase Figura 80) sobre su tiempo de estudio y descanso en diferentes meses. En la parte superior también habrá dos pestañas, una para ver la gráfica que contendrá los valores del tiempo de estudio, y la otra del tiempo de descanso.

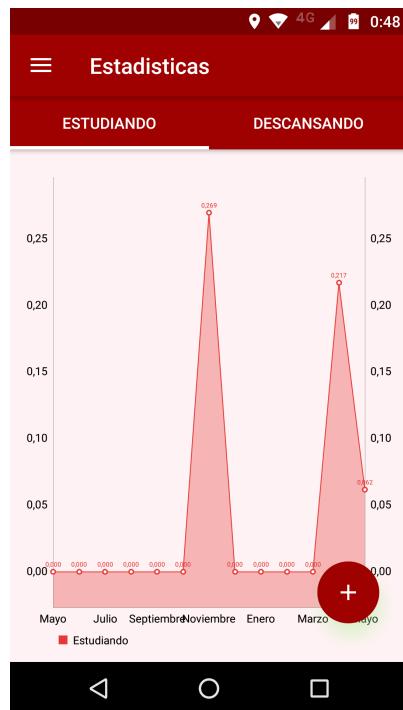


Figura 80: Estadísticas individuales

Para las gráficas de la ocupación real, se han utilizado gráficos circulares para representar los sitios, debido a que son más representativos que en una gráfica de puntos o lineal. Sin embargo, para representar el tiempo de estudio/descanso, se usa una gráfica de dos dimensiones, al tener dos variables: tiempo de estudio/descanso y la variable tiempo general.

Respecto a la implementación de las estadísticas, para añadirlas al menú lateral se han seguido los mismos pasos que en las secciones anteriores, por lo que, de nuevo, no se explicará en esta sección.

Las clases Java utilizadas son:

- **Conexión estadísticas:** es una clase asíncrona cuyo propósito es llamar a un archivo PHP para poder consultar tanto el estado real de la biblioteca elegida por el usuario como las estadísticas del mismo y las globales de esa biblioteca.
- **Estadísticas ocupación:** creará y pintará, con la ayuda de la clase *FragmentAdapter*, específica para el desarrollo de gráficos, la proporción de los puestos libres y ocupados de la biblioteca en un gráfico circular.
- **Estadísticas individuales:** carga los datos del usuario de la tabla histórico del servidor, y con la misma clase que antes, *FragmentAdapter*, dibujará el gráfico con las estadísticas de tiempo de estudio y tiempo de descanso del estudiante.

- **Estadísticas globales:** será similar al punto anterior, se cargarán los datos de todos los estudiantes de esa biblioteca, y con la misma clase mencionada anteriormente se mostrara el gráfico también en dos dimensiones.

7.2.11. Contacto

Otra de las funciones que podrá hacer el estudiante será la de enviar y recibir mensajes hacia el/la bibliotecario/a de cualquier biblioteca dada de alta en la base de datos (de momento solo está dada de alta la de la Facultad de Informática).

Para poder dividir los mensajes recibidos de los enviados, se emplea el recurso de los tabs o pestañas, al igual que en estadísticas y en horario, que se verá en la siguiente sección.

En la Figura 81 se ve cómo podríamos escribir un mensaje a una biblioteca.

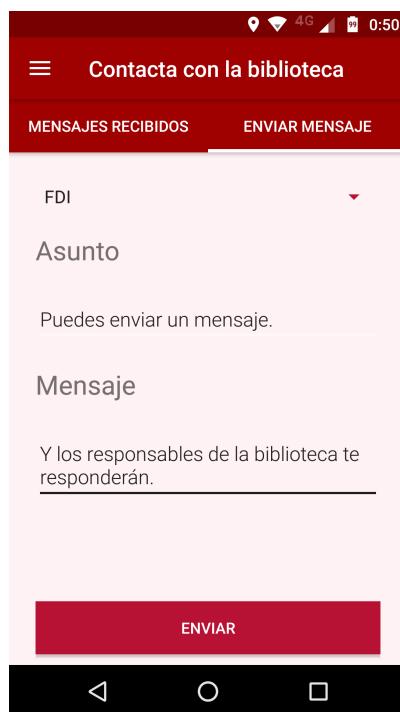


Figura 81: Envío de mensajes a los/as bibliotecarios/as

Para el diseño de enviar un mensaje se utiliza un selector, mediante el recurso de un Spinner, para poder seleccionar una opción de una lista. Para el asunto y mensaje se trata de un editText de varias lineas para poder escribir el texto del mensaje.

Para la bandeja de correos recibidos, se usa el recurso de listView para añadir filas dinámicamente en un fragmento de pantalla. Se mostrará la biblioteca, la fecha y el asunto del mensaje. Cada fila aparecerá solo en caso de haber sido respondido por la biblioteca, es decir si nosotros enviamos un mensaje hasta que no nos conteste la biblioteca no aparecerá dicha fila. En cambio, aparecerá una fila con el mensaje “No tienes mensajes recibidos”.

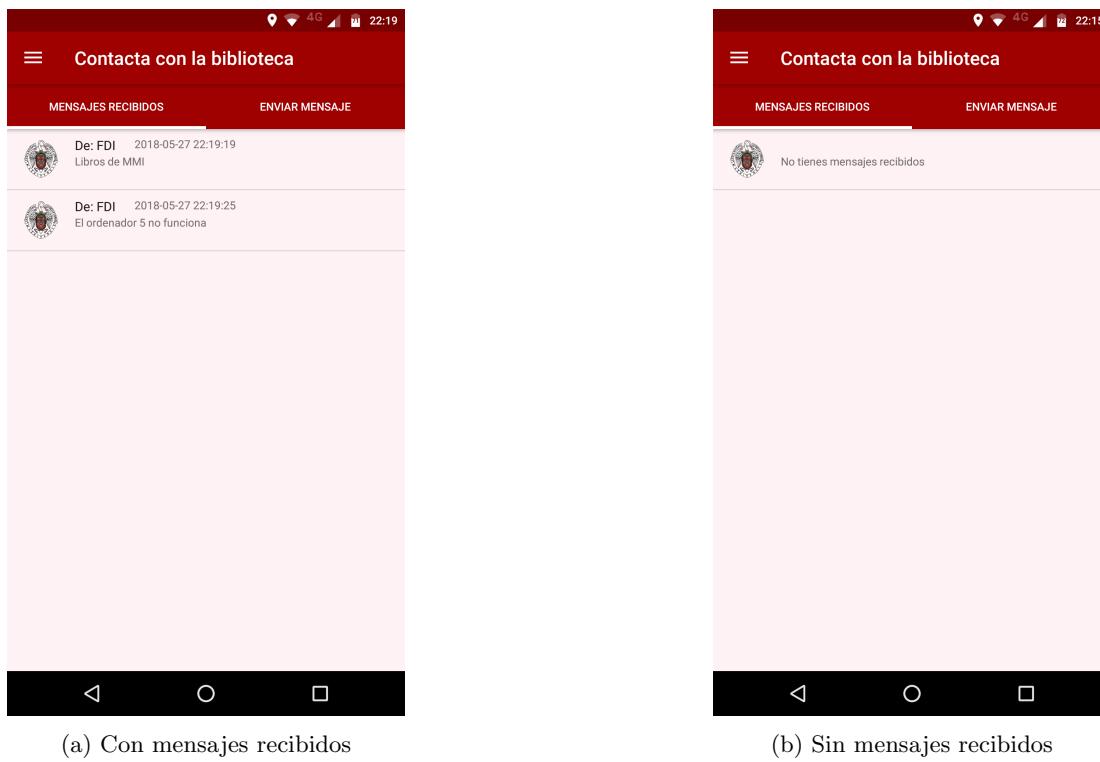


Figura 82: Bandeja de entrada mensajes

El usuario podrá pulsar en cualquier fila para poder ver la conversación que ha tenido: tanto su pregunta como la respuesta de la biblioteca.

Para implementar dicha funcionalidad, se ha añadido un apartado nuevo con esta función al menú lateral de la misma manera que en las anteriores secciones.

Respecto a la lógica del buzón de entrada, primero se cargan todos los mensajes respondidos del usuario. Si no hay ninguno, se lo hacemos saber al usuario. En cambio, si sí que los tiene, se almacenan en una lista de la clase Mensaje, la cual tiene los atributos asunto, emisor y fecha. En la Figura 83 se muestra el método por el cuál se rellena cada fila.

```

private ArrayList<Fila> GetArrayItems(){
    ArrayList<Fila> lista= new ArrayList<>();
    int cont=0;
    if (asuntos.length>0) {
        for (int i = 0; i < asuntos.length; i = i + 3) {
            lista.add(new Fila( emisor: "De: FDI", asunto: asuntos[i], R.drawable.ucm_logo, asuntos[i + 1]));
            did[cont]=asuntos[i+2];
            cont++;
        }
    } else
        lista.add(new Fila( emisor: "", asunto: "No tienes mensajes recibidos", R.drawable.ucm_logo, fecha: ""));
    return lista;
}

```

Figura 83: Fragmento de código - Rellenar filas de mensaje

Por cada fila se puede ver el hilo de la conversación, es decir, los mensajes que se han intercambiado. Así, al seleccionar una fila, se pasará mediante un intent la información y se mostrará mediante editText.

Para enviar un mensaje, aparecen las bibliotecas dadas de alta en la base de datos y los campo asunto y mensaje. Se obtiene su información con el *listener* del botón enviar.

Una vez conseguida la información, se conecta con el PHP correspondiente de envío de mensajes se inserta una nueva fila en la tabla envia de la base de datos con un id nuevo.

7.2.12. Horario

Otra de las funciones que podrá realizar el usuario será la de observar el horario de apertura y cierre de las diferentes plantas de las bibliotecas.

Esta función proporciona al usuario una manera rápida y sencilla de conocer las horas en las que las bibliotecas permanecen abiertas. El horario que se muestra es el que haya configurado el bibliotecario desde la página web.

Se ha optado por un diseño minimalista (véase Figura 84), sin necesidad de poner filas o columnas definidas por una linea, sobre fondo blanco para focalizar al usuario en los horarios.



Figura 84: Horario

En cuanto a la implementación es muy similar a estadísticas, ya que se establece una conexión con la base de datos para obtener el horario de la biblioteca.

Esta información se envía a la clase Horario para que se muestre. Los cambios que se produzcan en la biblioteca por parte del bibliotecario, tanto de cambio en el nombre de las plantas como el horarios de apertura o cierre, se verán reflejados también aquí.

Al igual que en las secciones anteriores, se hace uso de tabs o pestañas para poder separar los horarios de las diferentes plantas.

7.2.13. Modificar contraseña

Si el usuario desea modificar su contraseña podrá hacerlo con esta funcionalidad. Esta función, junto con Cerrar sesión, pertenece a un apartado del menú lateral llamado perfil. Se realiza el mismo proceso descrito en las anteriores secciones para incorporar esta función la menú.

A continuación, en la Figura 85 se muestra lo que vería el usuario al clicar en dicho apartado.

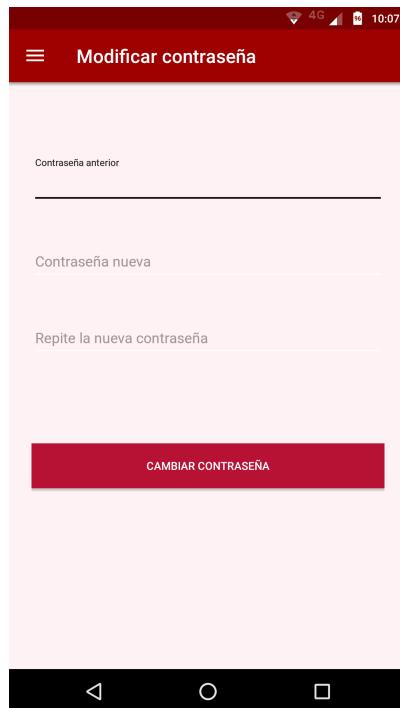


Figura 85: Modificar contraseña

Como se puede observar, habrá tres campos referidos todos ellos a las contraseñas. Primero se debe escribir la contraseña actual del usuario para asegurar que no ha entrado alguien en la aplicación que no sea el propio usuario y que quiera cambiar su contraseña, para quedarse él con la cuenta. En caso de no ser correcta se hará saber al usuario (véase Figura 86).

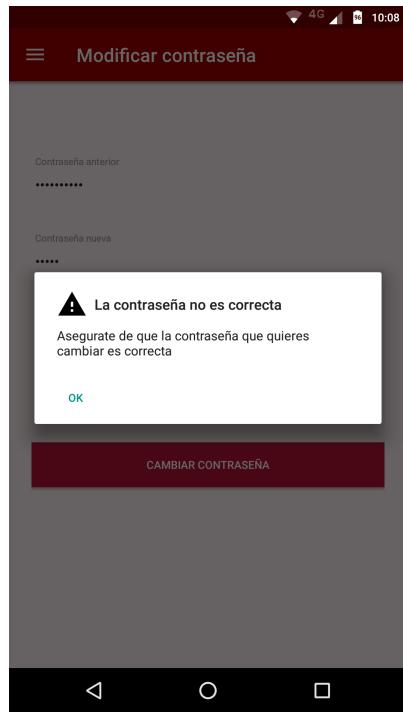


Figura 86: Modificar contraseña - Contraseña incorrecta

A continuación se le pedirá que ingrese la nueva contraseña, la cuál deberá tener una longitud de 8 caracteres incluyendo letras y números (la misma condición que cuando un usuario nuevo se registra). En caso de no cumplir con dicha restricción, se le hará saber al usuario con la siguiente alerta (véase Figura 87).

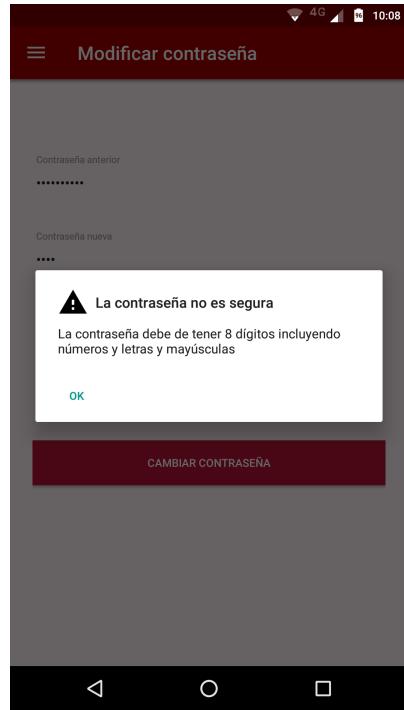


Figura 87: Modificar contraseña - Contraseña no segura

Si se cumplen estos requisitos además de coincidir los dos campos en los que hay que introducir la nueva contraseña, se modificará la contraseña. En otro caso, le aparecerá el mensaje de la Figura 88.



Figura 88: Modificar contraseña - Las contraseñas no coinciden

Como se observa, se han comprobado todos los posibles casos de fraude hacia la cuenta de un usuario, comprobando que el usuario es realmente él antes de cambiar su contraseña. Además, con la doble verificación de la nueva contraseña es perfectamente consciente de cuál es la nueva contraseña a utilizar.

En cuanto a la lógica, se puede observar en 'modificarContrasenia.java'. Después de conseguir las contraseñas, se comprueba que las dos contraseñas nuevas sean iguales y que respetan la restricción de los 8 caracteres con números y letras. A continuación, se hashean para mantener la privacidad del usuario y evitar vulnerabilidades. Se envían al PHP donde se comprueba en primer lugar que la contraseña a cambiar coincide con la de la base de datos de ese usuario para, posteriormente, actualizar la fila de ese usuario cambiando su contraseña por la nueva.

Después de realizar esta función, al volver a iniciar sesión, como es lógico, no será válida la contraseña que teníamos anteriormente, únicamente la que se ha introducido como nueva.

7.2.14. Cerrar sesión

Esta función se encargará de cerrar la sesión actual. Además, también parará la alarma implementada encargada de mantener la conexión con la base de datos y el usuario y se detendrá el servicio de ubicación que determina la ubicación del dispositivo. Una vez cerrada la sesión, se redirigirá a la página de Iniciar sesión.

En este caso, no se mostrará ninguna pantalla adicional para no sobrecargar al usuario, tan solo se mostrará un dialogo de alerta (véase Figura 89) para que el usuario esté convencido de que quiere cerrar su sesión.



Figura 89: Diálogo de cerrar sesión

La implementación de esta función es simple, ya que al llamar a esta función se elimina la sesión activa de la base de datos SQLite y se detienen las alarmas actuales. Además, si el usuario tenía un puesto ocupado, se liberará el puesto llamando a VaciarPuesto. En la Figura 90 se puede ver un fragmento de código donde se realiza la destrucción de la sesión y la detención de las alarmas.

```

if (requestCode == 20) {
    if(resultCode == Activity.RESULT_OK){
        Control_Sesion.getInstancia ( CerrarSesion.this ).borrarSesion ( usuario );
        stopService ( new Intent ( packageContext: CerrarSesion.this, ALARM_SERVICE.getClass () ) );
        stopService ( new Intent ( packageContext: CerrarSesion.this, ServiceUbicacion.class ) );
        Intent intentBiblio = new Intent ( packageContext: CerrarSesion.this, MainActivity.class );
        startActivity ( intentBiblio );
        finish ();
    }
}

```

Figura 90: Fragmento de código - Cerrar sesión

7.2.15. Ayúdanos a mejorar

Como función extra para la aplicación, se ha realizado un formulario de Google con el que se pretende conseguir un feedback directo de los destinatarios finales de la aplicación. En él se pueden encontrar preguntas como “¿Entiendes bien qué es Biblioteko y para qué sirve?”, “¿Qué te parece que haya un tiempo de descanso máximo de 30 minutos?” o “¿Utilizarías esta aplicación?” entre otras.

Esta función se encuentra en el menú lateral de la aplicación, dentro de una sección llamada Sugerencias. Al clicar en dicho apartado aparece un pequeño texto en el que se explica quién ha desarrollado la aplicación y se pide a los usuarios que rellenen el formulario al que se puede acceder pulsando en la imagen que aparece bajo el texto (véase Figura 91).

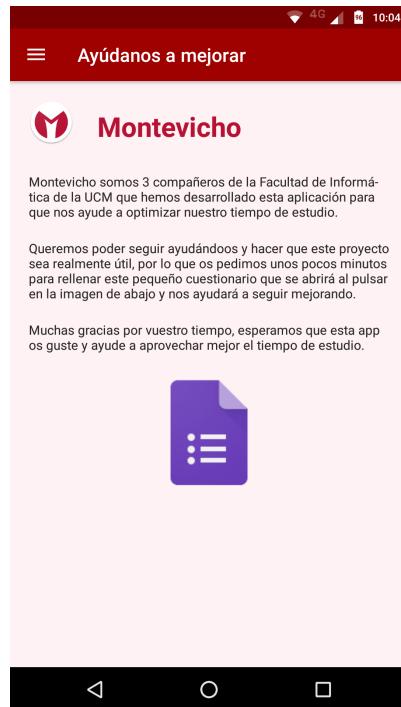


Figura 91: Formulario ayúdanos

Las respuestas recibidas por los usuarios serán analizadas en el siguiente capítulo y en el apartado de conclusiones. Se estudiarán las posibles correcciones propuestas por los mismos y se propondrán las correspondientes soluciones a partir de las necesidades de los usuarios.

7.2.16. Service Ubicación

Uno de los platos fuertes de la aplicación es el uso de la ubicación GPS del dispositivo para ubicar al usuario en el interior o exterior del recinto de las bibliotecas. Como ya se explicó, la intención de este proyecto es gestionar la ocupación de los puestos de estudio de la biblioteca, de manera que un puesto solamente aparezca ocupado si realmente el usuario se encuentra en él, a excepción del tiempo en el que se le permite estar fuera descansando. El objetivo de esta condición es que no haya puestos desocupados, todo puesto en el que no haya alguien sentado debe aparecer como libre en la aplicación, optimizando así la ocupación de las bibliotecas y su aprovechamiento por parte de los estudiantes.

Esto supone un gran reto, ya que es necesario saber dónde se encuentra el usuario de la app en cada momento. Y para esto se hace uso de la señal GPS que prácticamente todos los dispositivos Android incorporan y que tantas posibilidades de explotación tiene.

La intención inicial era que la aplicación pudiera detectar únicamente con la señal GPS en qué puesto de la biblioteca se encontraba el usuario. Rápidamente se tuvo que desechar esta idea al comprobar que, a pesar de lo mucho que ha avanzado esta tecnología en los últimos años, la imprecisión de este método era, para este propósito, tremadamente elevada.

Se barajaron otras posibilidades y tras varias propuestas, la opción que parecía idónea para solucionar el problema fue la que definitivamente ha quedado implementada. La aplicación detecta cuándo ha habido un cambio en la ubicación del dispositivo y comprueba, cada vez que esto ocurre, si se encuentra en el interior o en el exterior de alguna de las bibliotecas implementadas. La solución para saber qué puesto está ocupando o quiere ocupar el usuario no es tan automatizada como se pretendía, pero es altamente efectiva. Cuando un estudiante quiera ocupar un puesto, debe abrir la aplicación e indicar manualmente cuál es el puesto exacto en el que se sentará, y se procederá a realizar la acción tal y como se explica en las secciones Mesa y Ocupar Puesto.

Este es, en detalle, el proceso para obtener la ubicación del dispositivo.

Antes de nada, y acorde con el sistema de seguridad que implementa Android en todos sus dispositivos, es imprescindible que el usuario acepte los permisos de ubicación de la aplicación. Esto autorizará a la app a acceder a la ubicación del dispositivo. Sin este permiso, la aplicación no podrá funcionar correctamente.

La ubicación se conoce gracias a la ejecución ininterrumpida de un service. En Android, un service es un componente sin interfaz gráfica que se ejecuta en segundo plano y puede hacer todo tipo de operaciones. La app lo utiliza como método de tratamiento de los cambios en la ubicación del dispositivo. Se lanza este service en el momento en el que el usuario inicia sesión y selecciona una biblioteca. Desde entonces, se ejecuta en segundo plano y se controla que nunca se detenga, o que si lo hace, sea automáticamente lanzado de nuevo sin tener que realizar ninguna acción, independientemente de que la aplicación esté abierta o cerrada. Incluso si el dispositivo se apaga, al volverse a encender se iniciaría de nuevo por sí solo. De este modo, aunque el usuario reinicie su dispositivo y no vuelva a abrir la app se podrá seguir comprobando si se encuentra dentro o fuera de la biblioteca. Cabe destacar que la ubicación tan solo queda almacenada en la base de datos SQLite incorporada en el dispositivo, en ningún caso se almacena en el servi-

dor ni existe forma de saber cuál es la ubicación de un usuario, todo queda gestionado localmente.

Este service es el que se encarga de gestionar el ciclo de estados del usuario, además de las clases OcuparPuesto, VaciarPuesto, Descansar, CerrarSesion, Alarma y AlarmaEstoyVivo. Los posibles estados que puede adoptar un usuario son los siguientes: “FUERA_BIBLIOTECA”, que indica que no se encuentra en ninguna biblioteca, “DENTRO”, el usuario está dentro de una biblioteca pero sin ocupar ningún puesto (únicamente se comprueba la Biblioteca de la Facultad de Informática por ser, de momento, la única implementada), “OCUPANDO_PUESTO”, el usuario tiene un puesto ocupado, y “DESCANSANDO”, tiene un puesto ocupado pero está en su tiempo de descanso.

Al crear el service, se inicializa un objeto de la clase Polygon (una clase Java que junto a las clases Line y Point, todas de código abierto y licencia libre, conforman el algoritmo PIP[19] que permite saber si un punto está contenido en un polígono) con las coordenadas de la biblioteca de la facultad, que será el que permita saber si el usuario se encuentra dentro o fuera de la misma.

También en su creación se establece un listener de la ubicación del dispositivo que el sistema Android proporciona, y en el método onLocationChanged, el principal de esta clase, es donde se gestiona el estado en función de la nueva ubicación. Tras varias comprobaciones, se ha detectado que la precisión con la que el dispositivo capta nuevas ubicaciones puede ser extremadamente variada, del orden de 15 hasta 700, y tan solo las que tenían una precisión inferior a 20 localizaban el dispositivo con la exactitud necesaria, de modo que son solo estas las que se aceptan como válidas.

Una vez se percibe una ubicación lo suficientemente precisa, se actualiza la base de datos local con la ubicación recibida más recientemente y se comprueba si la misma se encuentra contenida en el polígono que se creó anteriormente.

Si lo está, se realizan una serie de comprobaciones para actualizar, si fuese necesario, el estado del usuario, además de otras acciones complementarias. En el caso de que el estado del usuario fuese “FUERA_BIBLIOTECA”, se actualiza el estado a “DENTRO” y se lanza una notificación (véase Figura 92) en la que se le sugiere que si va a ocupar un puesto lo indique en la app. El otro caso a tener en cuenta en esta situación es cuando estaba descansando y su anterior ubicación era fuera de la biblioteca (lo que se consigue saber comprobando si la última ubicación que conocíamos está o no dentro del polígono). Si esto ocurre, estaba descansando fuera y entra en la biblioteca, se le muestra otra notificación (véase Figura 93) en la que se le invita a finalizar su descanso para que acumule el tiempo que le sobra, si es que realmente desea finalizarlo, aunque esto no se hace automáticamente.

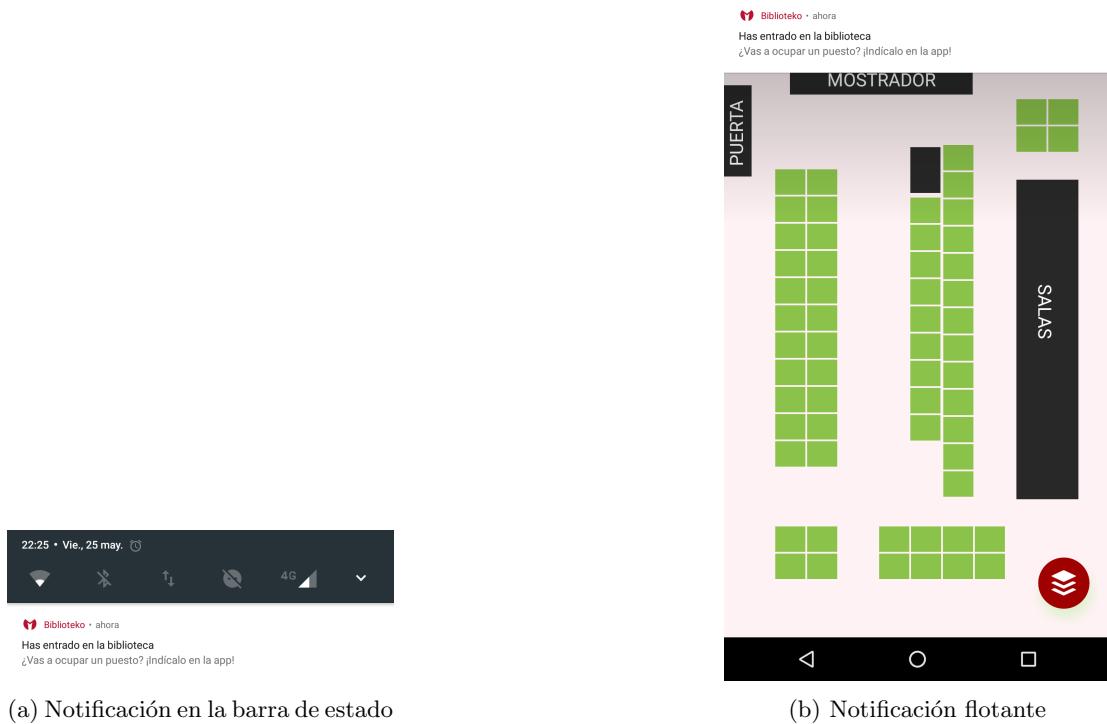


Figura 92: Notificación al entrar en la biblioteca

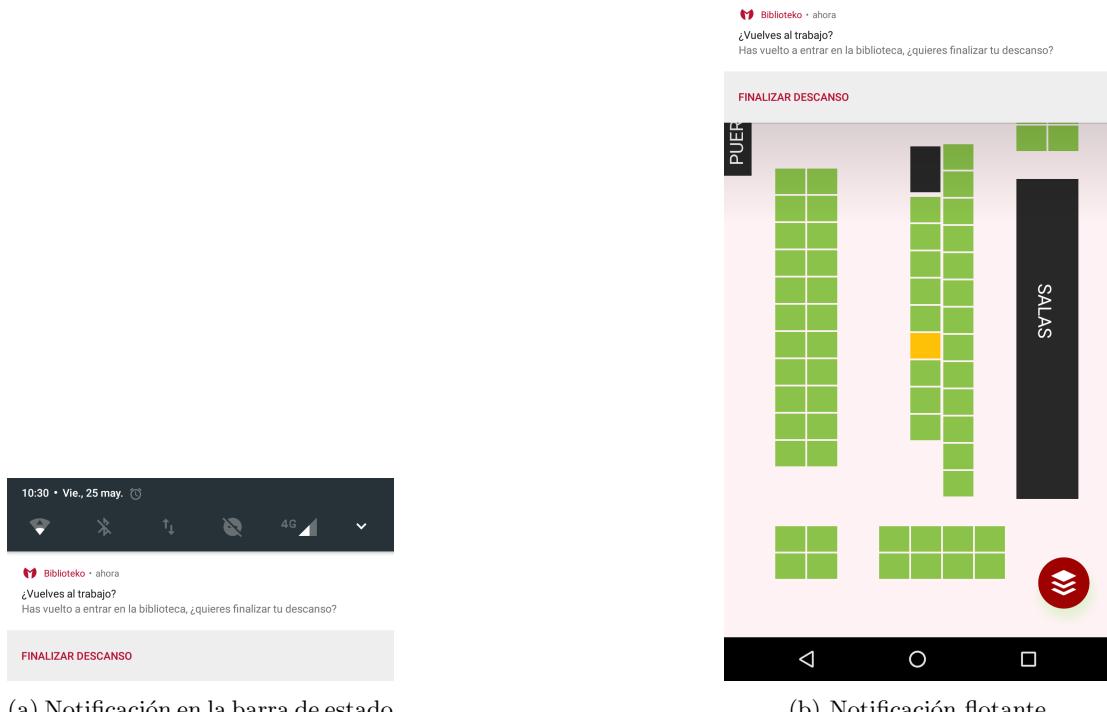


Figura 93: Notificación al volver del descanso

Si, por el contrario, la nueva ubicación recibida es fuera de la biblioteca, se comprueba si su anterior situación era “DENTRO”, para pasar únicamente a modo “FUERA_BIBLIOTECA”. En cambio, si su estado era “OCUPANDO_PUESTO”, se realiza un intent a la clase Descansar, que actualiza su estado, inicia la alarma con el tiempo de descanso disponible y le muestra por pantalla cuál es ese tiempo. A su vez, se lanza una notificación (véase Figura 94) en el que se le pide que si va a abandonar la biblioteca definitivamente, por favor libere el puesto, para que así no haya un puesto ocupado por nadie durante el tiempo que durase su descanso, aumentando así aún más el aprovechamiento de los puestos.

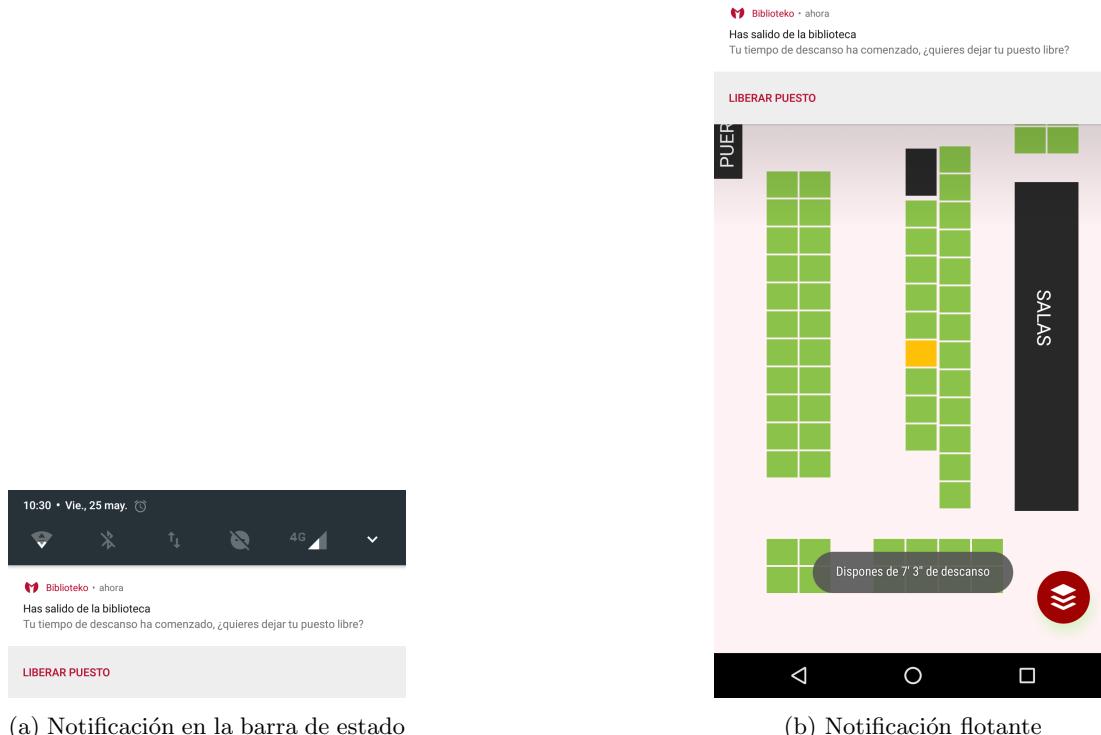


Figura 94: Notificación al salir de la biblioteca

Este traspaso de estados ha quedado definido de esta forma tras un largo estudio para optimizar el uso de los puestos. En la siguiente Figura 95 se puede observar el método onLocationChanged del service Ubicación.

```
public void onLocationChanged(Location location)
{
    //Solo aceptamos ubicación con precisión menor de 25
    if(location.getAccuracy () < 20) {
        usuario = Control_Sesion.getInstancia ( ServiceUbicacion.this ).selectSesion ();
        if (usuario != null ) {
            //Comprobamos si la ubicación anterior era dentro o fuera de la biblioteca
            anteriormente_dentro = polygon.contains ( new Point ( usuario.getLatitude (), usuario.getLongitude () ) );
            //Actualizamos la ubicación
            usuario.setLatitud ( location.getLatitude () );
            usuario.setLongitud ( location.getLongitude () );

            point = new Point ( usuario.getLatitude (), usuario.getLongitude () );
            dentro = polygon.contains ( point );

            if (dentro) {
                //SI ACABA DE ENTRAR EN LA BIBLIOTECA:
                if (usuario.getEstado ().equals ( "FUERA_BIBLIOTECA" )) {
                    //cambiamos a dentro
                    usuario.setEstado ( "DENTRO" );
                    //NOTIFICAR DE QUE ESTUDIE
                    //Acción al pulsar la notificación
                    Intent intent = new Intent ( packageContext: ServiceUbicacion.this, BibliotecaFDIplantal.class );
                }
            }
        }
    }
}
```

Figura 95: Fragmento de código - On Location Changed

7.2.17. Alarma

Es el método utilizado para determinar cuándo debemos liberar el puesto y cuándo debemos mostrar al usuario la notificación de que le quedan tan solo 3 minutos para que su tiempo de descanso se agote. En Android, todos estos eventos programados para ejecutarse en un determinado tiempo, se programan por eventos.

Aunque ya se mencionó previamente la utilización de la alarma, es aquí donde se tratará más detalladamente y se explicará cómo funciona. En el momento en el que se pulsa sobre el botón de realizar descanso se programan dos alarmas: una es la encargada de avisar al usuario que su tiempo está próximo a agotarse, y otra es la encargada de liberar el puesto si se ha agotado el tiempo disponible para el descanso y el usuario no se encuentra en la biblioteca. Ambas alarmas están implementadas en la clase 'Alarma.java'.

Al comenzar el descanso, se programa esta alarma para que salte en un momento que será igual a la hora actual más el tiempo de descanso disponible menos 3 minutos, es decir, 3 minutos antes de que finalice el tiempo de descanso. La otra alarma, encargada de liberar el puesto o regresar al estado anterior, saltará en el tiempo actual más el tiempo que tengamos disponible para descansar.

```

//Aquí solo va a entrar en el caso de la llamada de la primera alarma, puesto que salta cuando duracionRestante es 180 segundos aprox.
if (Control_Sesion.getInstancia ( context ).selectSesion () .getEstado () .equals ( "DESCANSANDO" ) && duracionRestante > 150) {
    //Acción al pulsar la notificación
    Intent intentF = new Intent ( context, BibliotecaFDIplantal.class );
    intent.setFlags ( Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK );
    PendingIntent pendingIntent = PendingIntent.getActivity ( context, requestCode, intentF, flags );
}

//Botón 'Finalizar descanso'
Intent finalizarIntent = new Intent ( context, Descansar.class );
//finalizarIntent.setAction("");
//finalizarIntent.putExtra(EXTRA_NOTIFICATION_ID, 0);
PendingIntent finalizarPendingIntent =
    PendingIntent.getActivity ( context, requestCode, finalizarIntent, PendingIntent.FLAG_CANCEL_CURRENT );

//Set the notification content
android.support.v4.app.NotificationCompat.Builder mBuilder = new android.support.v4.app.NotificationCompat.Builder ( context/*, CHANNEL_ID*/ )
    .setSmallIcon ( R.mipmap.montevicho_color_sinfondo )
    .setContentTitle ( "Fin de descanso en 3 minutos" )
    .setContentText ( "Tu tiempo de descanso finaliza en menos de 3 minutos. Regresa a la biblioteca para conservar tu puesto." )
    ..setStyle ( new android.support.v4.app.NotificationCompat.BigTextStyle ()
        .bigText ( "Tu tiempo de descanso finaliza en menos de 3 minutos. Regresa a la biblioteca para conservar tu puesto." ) )
    .setPriority ( android.support.v4.app.NotificationCompat.PRIORITY_HIGH )
    .setContentIntent ( pendingIntent )
    .addAction ( icon, 0, title: "FINALIZAR DESCANSO", finalizarPendingIntent )
    .setAutoCancel ( true )
    .setVibrate ( new long[]{100, 250, 100, 500} )
    .setColor ( ContextCompat.getColor ( context, R.color.colorComplu ) );

```

Figura 96: Fragmento de código - Alarma a tres minutos de finalizar descanso

Como se aprecia en la Figura 96, en el momento en que resten 3 minutos de descanso se entrará en la clase 'Alarma.java'. Sin embargo, esto solo ocurrirá si el estado es "DESCANSANDO", ya que no tiene sentido en otro caso. Además, para determinar qué alarma es la que se debe realizar, la duración restante calculada deberá ser mayor de 150 segundos. El hecho de poner 150 y no 180 (3 minutos), que es cuando teóricamente avisa la alarma, es que, tras un periodo de pruebas se observó que el planificador de alarmas no siempre es tan preciso como se esperaría y consta de un desfase de unos segundos. Si se cumplen estas condiciones establecidas, se procederá a mostrar el aviso al usuario.

La otra alarma mencionada anteriormente es la encargada de liberar el puesto o establecer nuevamente el estado del usuario a "OCUPANDO_PUESTO" cuando el tiempo de descanso llegue a su fin. De forma análoga a la anterior, se comprueba que el usuario se encuentra en estado "DESCANSANDO" y que además la duración restante en este caso es menor que 5, por la misma razón que en el caso anterior. Si se cumple la condición pueden ocurrir dos cosas, como se observa en la Figura 97.

```

boolean dentro = polygon.contains ( new Point ( Control_Sesion.getInstancia ( context ).selectSesion () .getLatitud (), Control_Sesion.getInstancia ( context ).selectSesion () .get
if (dentro) {
    Intent intentDescanso = new Intent ( context, Descansar.class );
    intentDescanso.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity ( intentDescanso );
} else {
    Intent intentVaciar = new Intent ( context, VaciarPuesto.class );
    intentVaciar.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity ( intentVaciar );
}

```

Figura 97: Fragmento de código - Alarma descanso finalizado

Se determina mediante un booleano y haciendo uso de la clase Polygon si el usuario se encuentra dentro del recinto de la biblioteca o no. En caso afirmativo, se realiza un intent a VaciarPuesto para que su estado pase a "FUERA_BIBLIOTECA" y su puesto se libere. En el otro caso, el intent llama a Descansar para así finalizar su descanso de manera correcta y conservar su puesto pasando a estado "OCUPANDO_PUESTO".

7.2.18. Alarma Estoy Vivo

Como se ha explicado en el apartado anterior, las alarmas es la forma que tiene Android para planificar eventos. Así pues, aunque también se ha mencionado previamente cómo se gestionan las conexiones vivas con los usuarios, es aquí donde se procederá a explicar cómo realmente está programada la alarma encargada de mandar periódicamente la señal de estoy vivo.

Una vez se ocupa un puesto, una señal de estoy vivo es mandada al servidor para informar de que ese usuario está activo. Además, se produce la programación de una alarma periódica que va a enviar cada 15 minutos una nueva señal de estoy vivo al servidor para evitar que el trabajo CRON elimine su conexión y proceda a la liberación de su puesto.

Sin embargo, esta señal no se enviará en todos los casos. Para empezar, es importante que el estado actual del usuario sea ocupando puesto o descansando. De otro modo, no será necesario enviar al servidor ningún dato de que el usuario está vivo, ya que al no tener ningún puesto ocupado no habrá constancia de ello en ninguna tabla de la base de datos.

Además, el usuario debe tener la ubicación activada ya que en caso contrario no es posible determinar si se encuentra dentro o no de la biblioteca y podría estar ocupando un puesto sin encontrarse realmente en la biblioteca. Lo mismo sucedería si un usuario ocupa un puesto y apaga su dispositivo. Podría disponer de un puesto ocupado por su cuenta indefinidamente sin saber dónde se encuentra realmente, pero al no estar ejecutándose esta alarma, el puesto se desocuparía por efecto del trabajo CRON. En la siguiente Figura 98 se puede ver parte del código mencionado previamente.

```
Log.e ( tag: "ESTOY VIVO", msg: "Ha entrado en OnReceive" );
LocationManager lm = (LocationManager) context.getSystemService ( Context.LOCATION_SERVICE );
boolean gps_enabled = false;
boolean network_enabled = false;

try {
    gps_enabled = lm.isProviderEnabled ( LocationManager.GPS_PROVIDER );
} catch (Exception ex) {
}

try {
    network_enabled = lm.isProviderEnabled ( LocationManager.NETWORK_PROVIDER );
} catch (Exception ex) {
}

if (!gps_enabled && !network_enabled && (Control_Sesion.getInstancia ( context ).selectSesion ().getEstado () .equals ( "OCCUPANDO_PUESTO" ) || Control_Sesion.getInstancia ( context ).selectSesion ().getEstado () .equals ( "DESCANSANDO" )) {
    Intent intentLiberar = new Intent ( context, VaciarPuesto.class );
    intentLiberar.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    context.startActivity ( intentLiberar );
    Toast.makeText ( context, text: "No ha sido posible acceder a la ubicación del dispositivo", Toast.LENGTH_LONG ).show ();
}
else if (Control_Sesion.getInstancia ( context ).selectSesion ().getEstado () .equals ( "OCCUPANDO_PUESTO" ) || Control_Sesion.getInstancia ( context ).selectSesion ().getEstado () .equals ( "DESCANSANDO" )) {
    AlarmaEstoyVivo.attemptEstoyVivo attemptLogin = new AlarmaEstoyVivo.attemptEstoyVivo ();
    attemptLogin.execute ( Control_Sesion.getInstancia ( context ).selectSesion ().getEmail () );
    //Toast.makeText ( context, "Estoy Vivo", Toast.LENGTH_SHORT ).show ();
    Log.e ( tag: "ESTOY VIVO", msg: "Estado OCCUPANDO o DESCANSANDO" );
}
```

Figura 98: Fragmento de código - Alarma estoy vivo

7.3. Otros componentes del sistema

Para poder guardar toda la información tanto para la web como la del usuario móvil, fue necesario tener una base de datos. Al principio todas las tablas y la información de cada una de ellas se guardaban de manera local, es decir en localhost, utilizando MySQL.

Después de realizar todas las pruebas, se alojó en un servidor gratuito, x10hosting, y ahí, dando de alta un dominio, se creó la base de datos y se insertaron las tablas que se fueron necesitando, insertando también filas en cada una de ellas.

En un primer momento, este servidor ofrecía todo lo necesario, tanto en el tiempo de respuesta de las consultas como una buena estabilidad de servicio.

Pero a medida que se fue desarrollando el proyecto, hubo que migrar a otro servidor, ya que fue necesaria una funcionalidad que no nos aportaba dicho servidor gratuito, por lo que migramos la base de datos entera al servidor SERED, servidor de pago. En la sección siguiente se desarrolla el porqué tomamos la decisión de realojar el proyecto a dicho servidor y qué funcionalidad ofrecía este de la cual carecía el anterior.

La funcionalidad que compartían dichos servidores era la administración de su base de datos, la cuál se hacia mediante phpMyAdmin. A continuación, en la Figura 99 se puede ver como están desarrolladas las tablas.

bibliotecas	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	1	InnoDB	utf8_spanish_ci	16 KB
envia	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	39	InnoDB	latin1_swedish_ci	16 KB
historico	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	853	InnoDB	latin1_swedish_ci	128 KB
horarios	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	21	InnoDB	utf8_spanish_ci	16 KB
ocupa	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	utf8_spanish_ci	32 KB
puestos	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	249	InnoDB	utf8_spanish_ci	16 KB
usuarios	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	71	InnoDB	utf8_spanish_ci	16 KB
usuariosActivos	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	0	InnoDB	latin1_swedish_ci	16 KB
usuariostemp	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar	19	InnoDB	utf8_spanish_ci	16 KB
9 tablas	Número de filas						1,253	InnoDB	latin1_swedish_ci	272 KB

Figura 99: Tablas existentes en la base de datos

Una de las funciones principales que no son visibles son los trabajos CRON, los cuales son funciones que se activan en un determinado momento en la base de datos. En la base de datos existen dos funciones CRON. En la Figura 100 aparecen todos los CRON programados para su ejecución en el servidor.

Trabajos de cron actuales						Acciones
Minuto	Hora	Día	Mes	Día de la semana	Comando	
*/5	*	*	*	*	/usr/local/bin/php /home/bibliot7/public_html/android/gestionarConexionesVivas.php	Editar Borrar
15	3	*	*	*	/usr/local/bin/php /home/bibliot7/public_html/android/limpiarBBDD.php	Editar Borrar

Figura 100: Trabajos CRON lanzados en el servidor

- **Limpiar Base de Datos:** a las 3:15 de la mañana, todos los días se ejecuta la función por la que se limpiará la base de datos, dicha función se halla en “limpiarBBDD.php”.
- **Gestión de conexiones vivas:** cada 15 minutos se produce la llamada a un PHP para la gestión de las conexiones de la app y determinar si sigue habiendo conexión entre el móvil y la base de datos.

Es conveniente y necesario reducir cada cierto tiempo el número de registros en la tabla historico, evitando que se llegase a un número demasiado elevado de filas en la tabla que hiciera peligrar y ralentizara la aplicación.

En un primer momento se pensó en la idea de implementarlo como una funcionalidad que iba a estar disponible en la página web, para que el bibliotecario cuando creyera conveniente pulsara sobre el botón y se realizase la limpieza de la base de datos. Sin embargo, tras pensarla detenidamente, se concretó que esto suponía una carga añadida para los bibliotecarios y que, tal vez, no supieran cuando era necesario realizar dicha limpieza de la base de datos.

Así pues, se empezó a investigar sobre cómo automatizar tareas en MySQL y primero se encontró el planificador de eventos de PHPMyAdmin. Parecía una solución sencilla y eficaz, era justo lo se buscaba. Tras realizar varias pruebas con diferentes consultas y ver cómo se comportaba el planificador se decidió darlo por bueno y activar los eventos que se planificaron.

Sin embargo, al pasar de realizar las pruebas en local al servidor gratuito, se encontró un problema. A pesar de que sí se podían crear eventos, no era posible activar el planificador por lo que no fue posible llevar a cabo las tareas programadas. Se consultó con los responsables del hosting y su respuesta fue que era una característica que no tenían disponible por cuestión de privilegios, así que se puso en marcha la migración a otro servidor, esta vez de pago, en el que alojar la versión definitiva.

Este servidor, desgraciadamente, tampoco tenía disponible el planificador de eventos, una vez más por temas de privilegios, pero lo que sí que estaba disponible era la posibilidad de crear CRON Jobs. Tras un poco de investigación en la red acerca de como crear este tipo de trabajos automatizados, se creó el primer CRON Job y lo se puso en funcionamiento para que se ejecutase cada día a las 3:15h de la mañana.

El CRON Job en cuestión lo que hace es lanzar un PHP (véase Figura 101) a las 3:15 de la mañana en el cual se han incluido las queries que se encargaban de realizar la limpieza de la base de datos en el planificador de eventos de MySQL.

```
<?php
$query3 = "INSERT into `historico` (email, fecha_inicio, duracion, estudiando, puesto) SELECT email, DATE_ADD(MAKEDATE(YEAR(fecha_inicio), 2), INTERVAL
estudiando SECOND) , SUM(duracion) as duracion, estudiando, CONCAT('Tglobal',SUBSTRING(puesto, 1, 3)) FROM `historico` WHERE `fecha_inicio` < DATE_SUB(NOW()
, INTERVAL 1 YEAR) GROUP BY YEAR(fecha_inicio), estudiando, email, SUBSTRING(puesto, 1, 3)";
$consulta3=mysqli_query($db,$query3);

$query4 = "DELETE FROM `historico` WHERE SUBSTRING(puesto, 4, 9) = 'global'";
$consulta4=mysqli_query($db,$query4);

$query5 = "DELETE FROM `historico` WHERE `fecha_inicio` < DATE_SUB(NOW(), INTERVAL 1 YEAR) and SUBSTRING(puesto, 1, 7) <> 'Tglobal'";
$consulta5=mysqli_query($db,$query5);

$query6 = "INSERT into `historico` (email, fecha_inicio, duracion, estudiando, puesto) SELECT email, DATE_ADD(MAKEDATE(YEAR(fecha_inicio), 1), INTERVAL
estudiando SECOND) , duracion, estudiando, CONCAT(SUBSTRING(puesto, 8, 10),'global') FROM `historico` WHERE `fecha_inicio` < DATE_SUB(NOW(), INTERVAL 1
YEAR) and SUBSTRING(puesto, 1, 7) = 'Tglobal' GROUP BY YEAR(fecha_inicio), estudiando, email, SUBSTRING(puesto, 8, 10)";
$consulta6=mysqli_query($db,$query6);

$query7 = "DELETE FROM `historico` WHERE SUBSTRING(puesto, 1, 7) = 'Tglobal'";
$consulta7=mysqli_query($db,$query7);
?>
```

Figura 101: PHP lanzado por el CRON Job que limpia la base de datos

Como se puede apreciar, lo primero que hace es insertar en un puesto temporal llamado TGlobalXXX donde XXX son las siglas de la biblioteca la suma total del tiempo que ha estado estudiando o descansando ese usuario agrupado por años. Es importante tener en cuenta que solo se van a agrupar datos anteriores al año, pues se considera importante conocer al detalle los momentos de estudio del último año. Una vez hecho esto, se borran los datos de los puestos XXXglobal, puesto que ya están incluidos esos datos en el puesto temporal TGlobalXXX. Tras esto, se eliminan todos los registros anteriores al año que sean diferentes de TGlobalXXX para así realizar la limpieza propiamente dicha de los puestos ocupados. Una vez hecho esto, se insertan los datos de los puestos TGlobalXXX en XXXglobal, poniendo como fecha el día 1 de enero de cada año, a las 00:00:00 si se corresponde con descansando y a las 00:00:01 si se corresponde con estudiando. Por último, borramos los puestos TGlobal puesto que, como ya se ha mencionado, se trata de un puesto temporal.

En cuanto al otro tipo de CRON implementado, tiene la misión de gestionar la conexión entre la aplicación y la base de datos. Con esto se asegura que no se ha perdido la conexión. Por ejemplo, si el usuario de la aplicación móvil ha reservado un puesto de estudio, y a continuación apaga el móvil, se deja de tener contacto con él y no se podría liberar su puesto.

Este fue uno de los problemas que surgieron sobre cómo gestionar las conexiones con el usuario y ver si estaba con el móvil encendido, y que se consiguió resolver finalmente gracias a dicho método. En un principio no fue la opción que se planteó para resolverlo. Antes de ella, se estuvo pensando en cómo hacerlo y, tras investigar, vimos que tal vez se podía hacer con Firebase. Sin embargo, tras probar e investigar un poco con todas las opciones que había disponibles, se observó que no era exactamente lo que se buscaba.

Tras seguir durante un par de días dándole vueltas a cómo resolverlo, un día surgió la feliz idea de que fuera el servidor el encargado de comprobar y gestionar periódicamente las conexiones con los diferentes dispositivos activos. Para ello, una vez el usuario inicie sesión, enviará un mensaje al servidor que almacenará a ese usuario como 'vivo' en una tabla junto a la última vez que le ha llegado una prueba de dispositivo vivo. Una vez ocupado el puesto y cada 15 minutos, siempre que la ubicación permanezca activada, el dispositivo volverá a mandar una nueva prueba de vida. Paralelamente, cada 5 minutos se produce la ejecución de un CRON Job que verifica si los dispositivos presentes en la tabla usuariosActivos cumplen la condición de estar vivos (esto es que su última información recibida haya sido recibida hace como mucho 20 minutos). De no estar vivos, el PHP actúa en consecuencia liberando el puesto.

Así se asegura que si al usuario se le ha acabado la batería del móvil y tenía un sitio reservado pero se va a ir de la biblioteca, si no se vuelve a tener respuesta de dicho estudiante, se desocupa el sitio en este periodo de tiempo. Con esto se consigue que los sitios que están ocupados en la biblioteca sean reales, y, en segundo lugar, que si algún estudiante quiere hacer trampas, ocupando indefinidamente un sitio en la biblioteca, se elimine esta posibilidad.

En cuanto a la implementación, se basa en un código PHP, que se puede observar en la Figura 102.

```
<?php  
$db = mysqli_connect('localhost','bibliot7_montevichosa','TFG\masters_18','bibliot7_biblioteko');  
$query = "UPDATE puestos set ocupado = 0 where ID IN (select puesto from ocupa where email IN (select email from usuariosActivos where ultimaInfo is not null  
and ultimaInfo < DATE_SUB(NOW(), INTERVAL 20 MINUTE)));";  
$consulta=mysqli_query($db,$query);  
$query2 = "DELETE from ocupa where email in (select email from usuariosActivos where ultimaInfo is not null and ultimaInfo < DATE_SUB(NOW(), INTERVAL 20  
MINUTE));";  
$consulta2=mysqli_query($db,$query2);  
$query3 = "DELETE from usuariosActivos where ultimaInfo is not null and ultimaInfo < DATE_SUB(NOW(), INTERVAL 20 MINUTE)";  
$consulta3=mysqli_query($db,$query3);  
?>
```

Figura 102: PHP lanzado por el CRON Job que gestiona las conexiones vivas

Y que se puede traducir a nivel usuario, como: en primer lugar seleccionar el email de todos los usuariosActivos cuya última información es de hace más de 20 minutos y establecer aquellos sitios de la tabla puestos que estén ocupados por esos usuarios a 0 (sitios desocupados). Además, se borran todos los usuarios de los que no hemos recibido información en los últimos 20 minutos de la tabla ocupa y por último se eliminan también de la tabla usuariosActivos.

Gracias a este servidor se ha podido implementar esta funcionalidad, la cuál en el anterior gratuito no se disponía de tal posibilidad y que ha aportado al proyecto un control sobre la aplicación.

Otra gran funcionalidad que ha propiciado el hecho de migrar a un servidor de pago ha sido la posibilidad de poder mandar correos automáticamente. Prácticamente, desde el primer momento se vio necesario que los usuarios verificaran sus correos para evitar que cualquiera se pudiera registrar en la app incluso con un correo inexistente.

Se estuvo pensando cómo conseguir esto y buscando ideas para lograrlo. Un buen día, tras finalizar una de las clases surgió la solución que hemos implementado finalmente. En primer lugar, se debía añadir un campo a la tabla usuarios, un campo booleano activado que nos permitiese saber si ese usuario había sido activado o no para permitirle o negarle el acceso a la aplicación. Además, se creó otra tabla, usuariostemp, en la cual se almacena el email de cada usuario que se registra y su email hasheado.

Una vez hecho esto, se genera un email que se manda al usuario que se acaba de registrar en el cual se encuentra un enlace con el emailHasheado para que active su usuario. Si al acceder al enlace, el emailHasheado coincide con uno de los emailHasheados de la tabla usuariostemp, se produce la activación que del email que corresponde a dicho emailHasheado en la tabla usuarios y se elimina la entrada de usuariosTemp. Todo esto se realiza en el PHP que se puede observar en la Figura 103.

```

<?php

$path = '../';
require_once($path."resources/config.php");
$db = conectarBBDD();

$idUsuario = $_GET["id"];
$sql="SELECT email FROM usuariostemp WHERE emailHashed = '$idUsuario' Limit 1";
$consulta=realizarConsulta($db, $sql);
if(mysqli_num_rows($consulta) == 1)
{
    $email=mysqli_fetch_object($consulta);
    $sql="DELETE FROM usuariostemp WHERE emailHashed = '$idUsuario'";
    $consulta=realizarConsulta($db, $sql);
    $sql="UPDATE `usuarios` SET `activado` = '1' WHERE `usuarios`.`email` = '$email->email'";
    $consulta=realizarConsulta($db, $sql);
}
else
    echo "User not found";
?>

```

Figura 103: PHP que activa un usuario determinado

Tras esto, se consiguió solucionar el problema de cómo activar los usuarios. Ahora bien, quedaba el hecho de que el correo se enviase automáticamente. Para ello, se encontró finalmente una clase PHP, PHPMailer, en internet que permitía generar los correos en el mismo momento en el que el usuario se registraba. Se descargó la clase, se configuró y se probó, y funcionaba a la perfección.

El problema llegó al hacer la migración al servidor gratuito. Se encontró el problema de que los puertos para el envío de correos estaban capados y, a pesar de realizarse la inserción del usuario correctamente en las dos tablas, el correo no se enviaba. Se consultó de nuevo con el equipo de soporte e indicaron que al tratarse de un servidor gratuito no estaba permitido el envío masivo y automático de correos. Así pues, tras barajar varias opciones se decidió pasar al actual servidor, con el cual se volvió a recuperar la funcionalidad de enviar correos.

La otra base de datos utilizada ha sido una base de datos interna SQLite que guardará los datos relativos al propietario de la aplicación.

Se tuvo que investigar sobre como crear una base de datos interna y de como mantener la sesión de un usuario a lo largo del uso de la aplicación.

Se descubrió que hay muchas variaciones para guardar la información, como tarjetas SD, ficheros, etc. Se ha preferido guardar la información en una base de datos, para poder preservar la privacidad del usuario, entre otros aspectos.

Como se puede ver en la implementación, para poder trabajar con SQLite en Android Studio se ha creado una carpeta llamada SSqlite, con dos clases Java:

- **SQLiteHelper:** clase utilizada para gestionar la base de datos, en la que se crea la base de datos interna (sesion).

Usada en su apertura y cierre, así como para facilitar su gestión en el ciclo de vida de las actividades.

Utilizará los siguientes métodos: ***onCreate*** y ***onUpgrade***, utilizados para la creación y la destrucción de la sesión activa en ese momento.

- **Control_Sesion**: utilizada para crear la sesión. Esta clase se llamará en la página principal, una vez que el usuario haya iniciado la sesión.

Tendrá métodos como ***crearUsuario*** que creará una sesión a partir del nombre, estado, la ubicación... Y métodos para seleccionar, añadir, actualizar o insertar un usuario ***selecionSesion, insertarSesion, updateSesion, borrarSesion***. A continuación, en la Figura 104 se puede ver un fragmento de código de ejemplo de como se inserta una sesión:

```
//Inserta en la tabla 'sesion' la sesión pasada por parámetro
public boolean insertarSesion(Usuario usuario){
    boolean ok=false;
    long i=db.insert ( table: "sesion", nullColumnHack: null, usuario.toInsert ());
    if(i>0){
        ok=true;
    }
    return ok;
}
```

Figura 104: Fragmento de código que inserta una sesión en SQLite

Para finalizar esta sección, se va a hablar de la implicación en las redes sociales sobre la aplicación.

Se ha creado un usuario en Instagram, Twitter, y Facebook, con la misión de dar a conocer el proyecto a nivel de la facultad y de la Universidad Complutense de Madrid, para incentivar a la gente a que pruebe la aplicación, y que así sirva de ayuda para mejorarla y corregirla de fallos o funciones/diseño que puedan no gustar. Cabe recordar que la misión principal al iniciar este proyecto era que la gente lo pudiera usar en la biblioteca, por lo que había que asegurarse de su correcto funcionamiento.

8. Evaluación de usuarios y pruebas

Cuando se comenzó la realización de este proyecto, el objetivo era llegar a tener finalizada la app para el 11 de mayo para que así estuviera disponible para los exámenes y poder conseguir el feedback de los estudiantes. Sin embargo, el feedback de los usuarios no solo se ha obtenido tras finalizar la aplicación, si no que ha estado presente en varias fases, que son las siguientes.

8.1. Inicio del proyecto

En un inicio, se contó con la opinión de una de las bibliotecarias de la facultad para la versión web, puesto que es más relevante su opinión en este caso que la de estudiantes que ni siquiera van a tener acceso a la página web. Así pues, los principales aspectos que fueron mencionados por ella son que debía tratarse de una web sencilla y muy intuitiva para que todas las bibliotecarias pudieran adaptarse rápidamente a la utilización de la misma. Además, destacó que sería muy útil que estuvieran presentes funciones como poder establecer el horario, para evitar tener que poner un cartel a mano en el que se reflejase el horario.

Para la versión móvil, se optó por enseñar un prototipo de la aplicación y ver cómo era el comportamiento de los estudiantes con ella. Tras ver sus reacciones y escuchar sus opiniones, se procedió a modificar algunos aspectos del prototipo inicial. Una de los aspectos modificados tras esta prueba inicial fue el hecho de implementar la posibilidad de liberar el puesto tanto desde el mismo puesto como desde una nueva opción incluida en el menú desplegable si existía un puesto ya seleccionado. Otro aspecto criticado por los usuarios en este prototipo y cuyo cambio se ha visto reflejado en la versión final ha sido el color del fondo. Gran número de los encuestados resaltaron que el color azul era demasiado intenso. Es por ello que en la versión final se ha establecido un color blanco ligeramente rojizo, para no desentonar con la paleta de colores de la UCM.

8.2. Durante el proyecto

Tras la realización de la primera iteración de la aplicación móvil, se procedió a la realización de la evaluación de los usuarios. En dicha evaluación no se produjo ninguna crítica reseñable por lo que se procedió a continuar con la segunda iteración.

En la tercera evaluación realizada a los usuarios tras la segunda iteración sí que se obtuvieron resultados relevantes. De igual modo que en la primera evaluación, los colores utilizados para representar los puestos libres y ocupados no fueron del agrado de los estudiantes, quienes criticaron el hecho de que fueran demasiado llamativos para el ojo humano. Así pues, finalmente se decidió implementar con unos colores de los propuestos por Material Design, presentes en multitud de aplicaciones desarrolladas en Android.

8.3. Tras el proyecto

Uno de los principales errores más reportados por los usuarios tras haber publicado la aplicación en Google Play fue el hecho de que, en las primeras versiones, la app se bloqueaba y se cerraba continuamente. Una vez leídos los informes de Google Play Console, se consiguió establecer un patrón común en todos los móviles afectados por este fallo. Se trataba de un error que afectaba a móviles que no disponían de Android puro, con capas de personalización como las de Xiaomi, Samsung o LG.

Este error fue solucionado tras un período de investigación sobre posibles soluciones y se publicó la nueva versión actualizada en Google Play que solventaba la problemática. La solución implementada consiste en añadir `intentProblematico.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)` antes de lanzar determinados 'intent'. En la Figura 105 se puede observar, como se ha ido reduciendo el número de bloqueos de la aplicación a medida que han ido pasando los días y nuevas versiones que solventaban bugs iban siendo actualizadas en Google Play. Es por tanto que la aplicación móvil ha sido testeada en casi todo tipo de móviles, con diferentes versiones (desde 5.0 a 8.1), y diferentes tamaños y resoluciones de pantalla.



Figura 105: Gráfica que muestra el número de bloqueos diarios de la app

Otro de los aspectos que se ha podido perfeccionar en la app tras lanzarla al mercado y ponerla al servicio de los estudiantes ha sido el hecho de determinar correctamente si un usuario está dentro de la biblioteca o no. Tras observar que la tabla ocupaba información de un puesto ocupado por un usuario a unas horas en las que era imposible que lo estuviera se determinó que existía un problema en el caso de desactivar la ubicación una vez el puesto había sido ocupado. En ese caso, el puesto quedaba ocupado sin importar si el usuario abandonaba la biblioteca o no hasta que se activase la ubicación.

Se trataba de un fallo muy grave que permitía a los estudiantes descansar ilimitadamente. Es por ello que fue solventado rápidamente, añadiendo una condición nueva a la alarmaEstoyVivo. La nueva condición implementada consiste en que el usuario debe tener la ubicación activada para que se produzca el envío de dicha señal. En caso contrario, el puesto del usuario es liberado.

Existen disponibles varias vías de comunicación para que los usuarios se pongan en contacto con los desarrolladores del proyecto. Tales vías son cualquiera de los perfiles del proyecto en redes sociales (Twitter, Instagram y Facebook) las cuales son actualizadas diariamente, correo electrónico y, por supuesto, el boca a boca. Además, existe un cuestionario [20] del que ya se ha hablado previamente disponible en la app para que los usuarios dejen su opinión. Hasta la

fecha de hoy se han registrado 31 respuestas, las cuáles se anexarán a continuación.

¿Entiendes bien qué es Biblioteko y para qué sirve?

31 respuestas

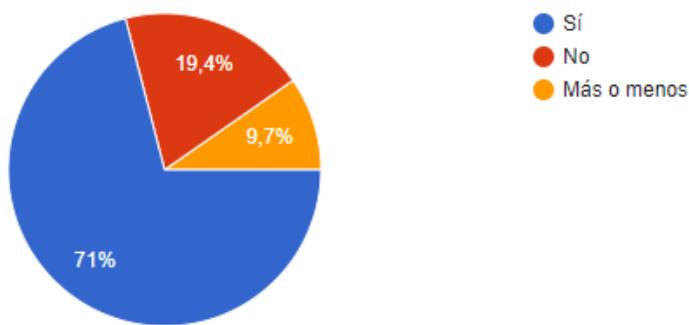


Figura 106: Pregunta : ¿Entiendes bien qué es Biblioteko y para qué sirve?

Como se puede apreciar en la Figura 106, más del 80 % de los encuestados entienden qué es Biblioteko y para qué sirve, frente al 19'4 % que dice no entenderlo. Se trata de una pregunta clave para el inicio y el número de descargas y desinstalaciones de la aplicación, puesto que de ser este número muy elevado, la aplicación no permanecerá en los dispositivos de los usuarios, ya que desconocen su funcionamiento.

¿Recuerdas algún momento en el que te hubiese sido útil conocer la ocupación actual de la biblioteca?

31 respuestas

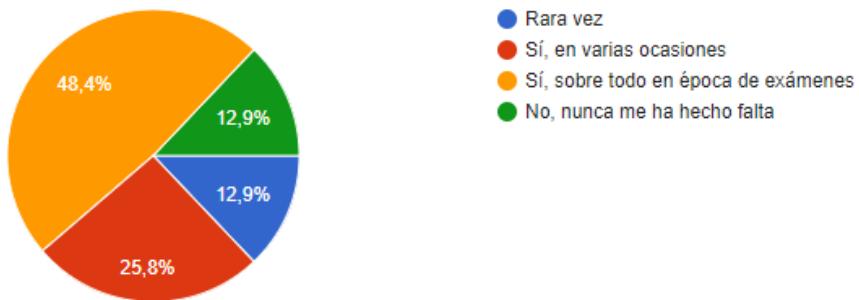


Figura 107: Pregunta : ¿Recuerdas algún momento en el que te hubiese sido útil conocer la ocupación actual de la biblioteca?

En la Figura 107 se puede observar que, tan solo algo menos del 13% de los estudiantes afirman no haber necesitado nunca conocer la ocupación en un determinado instante. Es por ello que se puede determinar que Biblioteko viene a ocupar un gran hueco existente en lo que a gestión de puestos de bibliotecas se refiere, ya que se trata de algo que hubiera sido útil para la mayoría de los estudiantes en algún momento de sus vidas.

¿Cuándo crees que sería útil saber qué puestos están libres?

31 respuestas

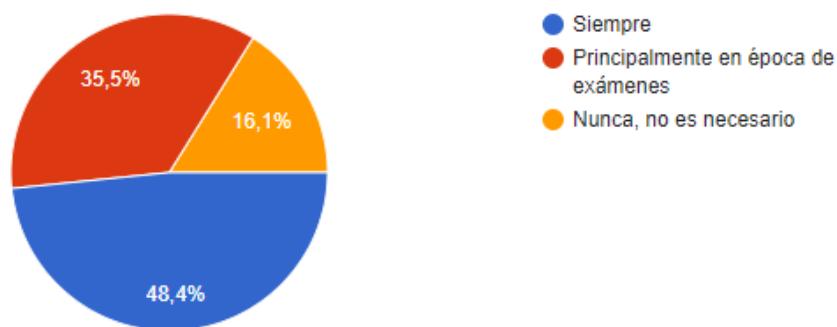


Figura 108: Pregunta : ¿Cuándo crees que sería útil saber qué puestos están libres?

En la Figura 108 existe prácticamente un 85 % de los encuestados que considera que sí que sería útil conocer la disponibilidad de puestos de la biblioteca, frente a poco más de un 15 % que no lo considera necesario. Del 85 % que sí lo consideran, casi un 58 % considera que sería útil siempre mientras que el 42 % lo vería útil principalmente en época de exámenes, cuando las bibliotecas presentan mayor afluencia.

¿Qué te parece que haya un tiempo de descanso máximo de 30 minutos?

31 respuestas

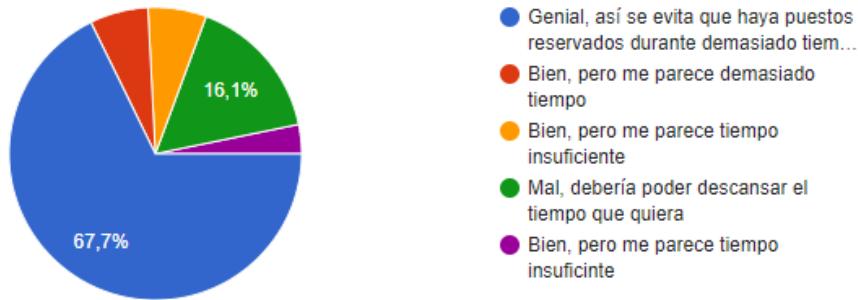


Figura 109: Pregunta : ¿Qué te parece que haya un tiempo de descanso máximo de 30 minutos?

Nota aclaratoria con respecto a la Figura 109: Tras la publicación del cuestionario, se advirtió de la existencia de una errata, por lo que se subsanó y hay que mencionar que la opción morada y la naranja son la misma. Ante esta pregunta, poco mas del 15 % de los encuestados consideran que el tiempo de descanso debería ser libre, frente al 67,7 % que piensan que 30 minutos está genial. Además, hay un 9,7 % que lo considera insuficiente y un 6,5 % al que le parece demasiado.

¿Tienes claro cuáles son las funciones de Biblioteko?

31 respuestas



Figura 110: Pregunta : ¿Tienes claro cuáles son las funciones de Biblioteko?

En la Figura 110 se puede observar como una abrumadora mayoría, superior al 80 % dice tener claras las funciones de la aplicación. Algo más del 9 % dice no tener claro qué es lo que puede hacer ni cómo hacerlo mientras que es la misma cantidad de encuestados la que es más tajante y afirma no saber lo qué puede hacer y lo qué no.

¿Te resulta fácil utilizar la app?

31 respuestas

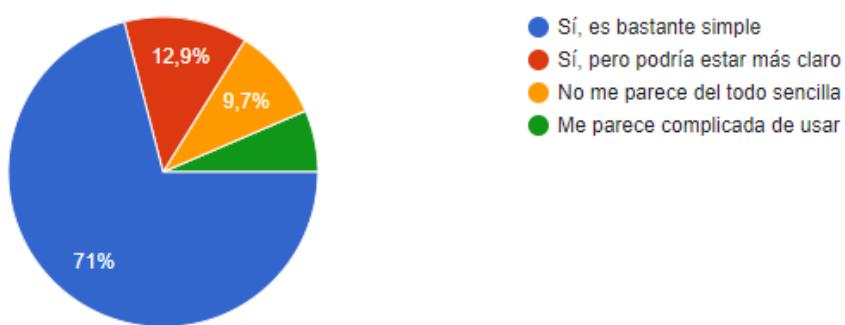


Figura 111: Pregunta : ¿Te resulta fácil utilizar la app?

En la Figura 111 se observa que a casi el 85 % de los encuestados le resulta fácil utilizar la app. De todos ellos, prácticamente un 85 % la encuentran simple de usar mientras que el 15 % restante considera que podría ser más clara. Por otra parte, existen un 9,7 % de los encuestados a los que no les parece del todo sencilla y tan solo un 6,4 % a los que les parece complicada.

¿Has encontrado algún fallo en la aplicación?

1 respuesta

Creo que debería existir la posibilidad de cerrar sesión ya que no tiene mucho sentido cambiar la contraseña si no puedes volver a logarte.

Figura 112: Pregunta : ¿Has encontrado algún fallo en la aplicación?

En la Figura 112 se aprecia que tan solo un usuario ha reportado un error, en este caso una sugerencia. Inicialmente no se desarrolló la funcionalidad Cerrar sesión para evitar que a través de un mismo dispositivo se pudiera iniciar sesión en varias cuentas y ocupar más de un puesto, pero este problema de seguridad se subsanó con la implantación de la Alarma Estoy Vivo y su correspondiente CRON de conexiones vivas explicados anteriormente, por lo que esto ya no suponía ningún riesgo para el uso de la aplicación. Por lo tanto, se consideró muy útil esta respuesta y se incorporó la citada funcionalidad gracias a la aportación de uno de los usuarios.

¿Consideras útil esta aplicación?

31 respuestas

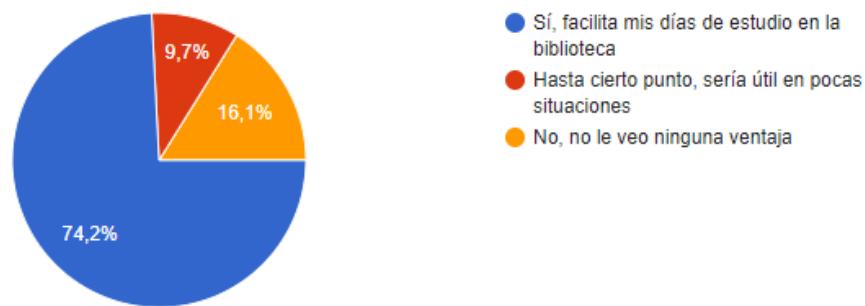


Figura 113: Pregunta : ¿Consideras útil esta aplicación?

La Figura 113 supone un gran impulso para la aplicación ya que en ella se puede apreciar como casi el 85 % de los encuestados consideran útil esta aplicación. De ellos, más del 88 % consideran que la aplicación facilita sus días de estudio mientras que el resto consideran que es útil en algunas ocasiones. Son tan solo poco más del 15 % de los encuestados los que no ven ninguna ventaja en la aplicación.

¿Utilizarías esta aplicación?

31 respuestas

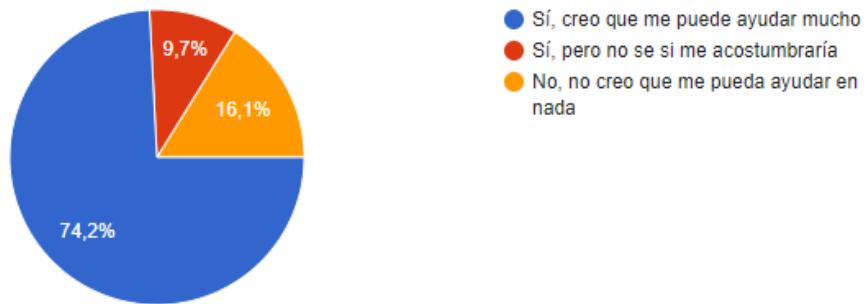


Figura 114: Pregunta : ¿Utilizarías esta aplicación?

Como pregunta final, se puede observar en la Figura 114 como casi el 85 % de los encuestados utilizarían esta app. Se puede apreciar una correlación con los resultados de la pregunta anterior en la que existe el mismo porcentaje de encuestados que consideran que les puede ayudar mucho con el que afirman que les facilitan mucho los días de estudio. De igual modo, los que afirman que la usarían pero no saben si se acostumbrarían coinciden en número con los que afirman que les sería útil en algunas ocasiones. Finalmente, los que no creen que les puede ayudar en nada son el mismo porcentaje que no le ve ninguna ventaja.

Por otro lado, a través del correo electrónico se han producido algunas sugerencias que han sido solventadas o implementadas en la versión final como es el hecho del problema que tuvieron algunos usuarios en la recepción del mail de validación o dotar de mayor visibilidad al apartado en el que registrarse.

La evaluación de usuarios es un aspecto clave a la hora de desarrollar un proyecto y es por ello que es preciso contar con la opinión de los usuarios finales a los que va destinada la app. Son los usuarios finales, los cuales se encuentran desde 0 con la app sin ninguna idea de cómo funciona, los que deben considerar si es fácil de usar o no y si se trata de una aplicación que usarían o no.

Hasta el momento de la redacción de esta memoria, han sido 8 las versiones que se han publicado en la tienda, cada una corrigiendo, mejorando o implementando ciertas funcionalidades que han ido aconsejando los diferentes usuarios por la app. La versión más reciente es la 1.3.5 y actualmente no presenta ningún tipo de fallo ANR ni bloqueo reportado en Play Console. A todos los usuarios, dar las gracias nuevamente puesto que sin ellos no hubiera sido posible la corrección de ciertas características de la aplicación.

9. Conclusiones y trabajo futuro

9.1. Conclusiones

Tras la finalización del proyecto, una buena forma de ver todo lo que se ha conseguido, es echar la vista atrás y comparar la idea y lo que se quería desarrollar al comienzo del proyecto con lo que se ha terminado haciendo.

La idea principal era la realización de un proyecto que no quedara en el olvido tras su finalización, una herramienta que tuviera su uso en la actualidad y que sirviera de ayuda. Es por esto que se pensó en una aplicación que sirviera a los estudiantes en la reserva de puestos de estudio en la biblioteca, haciendo uso de la geolocalización.

Aunque se tenía claro cómo sería la aplicación, qué funcionalidades tendría y qué aportaría al usuario, ninguno de los integrantes del grupo había desarrollado una aplicación móvil de esta dimensión ni se tenía conocimiento de las numerosas APIs que se han tenido que utilizar.

Es por ello que durante todo el desarrollo del proyecto se investigó y aprendió sobre todo tipo de conocimientos que se desconocían y que no se habían enseñado en la carrera. Y aunque no siempre se aprendieran cosas que luego se han aplicado al proyecto y otras muchas veces no se encontrasen ideas o recursos para resolver un cierto problema, siempre se acabaron solventando y se acabó desarrollando la aplicación que ha sido publicada ahora, tal y como pretendíamos desde el inicio.

Por tanto, se concluye, que a pesar de que el proyecto ha sido un camino lleno de aprendizaje de multitud de recursos y de nuevas tecnologías, se ha tenido la capacidad de no rendirse y siempre tener claro de que se podía conseguir.

9.2. Conclusions

After the final release of the project, a good manner to have an overall view of all that we have done, is to take a look back and compare the idea and the original project of we wanted to do at the beginning of the project and we have finally done.

The original idea was to develop a project that left a mark in time, an application that would be in use for years among the students and would help them. For this reason, the idea was to develop an application that would help them with the management of the study spaces, using geolocation. Although it was defined how the application would be, what functionalities it would have and how it would help to the users, no one of the members of the group had worked before in developing an application of this kind and had no idea about the large number of APIs that have been finally used.

Because of this, during the development of the project, it was necessary to investigate and learn about some knowledges that have not been learnt during the degree. Most of the things that were learnt have been used in the project but some not, because they were not good enough for the project or because they were not the exact solution that the developers were looking for. However, after every problem that occurred during the project a solution that solved it was finally found and the application have been released as originally, or even better, was planned.

Because of all of these reasons, it can be concluded that although the project has been a long journey in which a lot of new technologies and resources have been learned, there was the ability of not giving up and keep pushing because the idea of it could be done it was always present.

9.3. Trabajo futuro

Aunque se han conseguido implementar las funciones que se pensaron desde el inicio casi en su totalidad, hay varios aspectos que no han llegado a desarrollarse, bien por falta de tiempo o bien porque no se vio necesario para un primer desarrollo de la aplicación.

Alguno de estos aspectos han sido:

- **Adición de nuevas bibliotecas:** la limitación de tiempo tan solo ha permitido que se implemente la Biblioteca de la Facultad de Informática, pero el primer y principal objetivo es aumentar la lista de bibliotecas de la Universidad Complutense de Madrid disponibles en la aplicación, como ya se ha visto en la Figura 55 del apartado 7.2.4 Bibliotecas.
- **Exposición del proyecto a los responsables de las bibliotecas:** al tratarse de un proyecto en desarrollo, han sido los propios desarrolladores los encargados de manejar la aplicación web para la biblioteca implementada, pero para que el proyecto tuviese el éxito esperado es necesario presentarlo a los responsables de cada biblioteca ya que son los destinatarios de la aplicación web, con intención de exponerles las ventajas que les ofrece y que pasen a utilizarla en una versión definitiva.
- **Pegatinas NFC:** la intención inicial fue que la aplicación detectase, por la ubicación del dispositivo, en qué puesto exacto se encuentra el usuario. Esto no ha sido posible debido a que la geolocalización de los dispositivos móviles no es todo lo precisa que se necesita, por lo que durante el desarrollo de la aplicación surgió la idea de desarrollar pegatinas con conexión NFC que, colocadas en cada uno de los puestos de la biblioteca, indicase que un estudiante está ocupando ese puesto tan solo con pasar el móvil por encima. Esto pareció buena idea pero no se llevó a cabo porque se coincidió en que excedía del propósito de este proyecto, además de por otros problemas que surgían, como por ejemplo los descansos, aunque siempre se ha tenido en cuenta la opción de desarrollar esta idea para mejoras futuras.
- **Nuevas estadísticas:** por ahora, se puede comprobar el tiempo medio de estudio y de descanso por estudiante y por biblioteca, pero se pretende añadir nuevas estadísticas que fuesen de utilidad tanto para los bibliotecarios como para los usuarios de la app, como biblioteca más visitada, horas de más afluencia de cada biblioteca, o plantas más solicitadas.
- **Horario especial:** una funcionalidad añadida a la de Horario ya implementada y que sería de gran utilidad es introducir en la aplicación web la posibilidad de añadir un horario especial, en el que detallar días o períodos en los que el horario normal de la biblioteca se ve modificado, como ocurre en muchas de ellas en época de exámenes. En consecuencia, se desarrollaría el mismo apartado en la aplicación móvil para su consulta o incluso para hacer una búsqueda de determinadas fechas y ver el horario en el tiempo especificado.
- **Inclusión del proyecto en la AEPD:** un aspecto importante es el tratamiento de los datos que se obtienen de los usuarios al registrarse. Hasta ahora no ha sido un punto de conflicto puesto que se trata de un prototipo, pero es necesario que si el proyecto continúa hacia delante, se ingrese el proyecto en la Agencia Española de Protección de Datos (AEPD) con el fin de cumplir con la legislación actual, además de generar un informe de términos y condiciones de uso y política de protección de datos.
- **Mejoras y correcciones:** son muchas las cosas que quedan por mejorar, como introducir el apartado “He olvidado mi contraseña” que se dejó para el final por ser poco relevante en el proyecto y finalmente no se desarrolló por falta de tiempo. Se espera seguir recibiendo el feedback de los usuarios y continuar desarrollando este proyecto con la misma ilusión para tener una herramienta útil y eficaz que llegue al máximo número de personas posible y pueda ayudarles en su estudio diario.

9.4. Trabajo individual

9.4.1. Carlos Gavidia Ortiz

Para empezar a comentar mi aportación a este proyecto, tenemos que irnos al inicio, cuando a partir de una idea original que se tenía para hacer el proyecto. Mi aporte al igual que el resto de mis dos compañeros, fue una lluvia de ideas que resultasen interesantes y que fuesen viables para el desarrollo de la aplicación y que tuvieran futuro en cuanto a la viabilidad y el uso de proyecto, en caso de que saliesen bien. Me estoy refiriendo por ejemplo al caso de funcionalidades que utilizaríamos en la aplicación y en la web y que tuviéramos una idea de como resolverlas.

Tras estas primeras reuniones de lluvia de ideas los tres, junto con nuestro director, empezamos con la realización del prototipado a nivel bajo de la aplicación para móvil, y para web y de como se relacionarían estas dos con las bases de datos implementadas. También hicimos una reunión los miembros del grupo, para decidir a nivel colectivo, los datos que iban a guardar las bases de datos y el porqué deberían guardar dichos datos.

Gracias a las opiniones de todos los integrantes del grupo, realizamos los bocetos, al principio en una hoja dibujándolos, y luego con programas específicos para la realización de estos mismos. Después de la realización de los mismos, nos dispusimos a realizar una prueba a personas, como eran nuestros familiares y amigos, para que nos dieran su opinión de que les parecía la aplicación y la web en cuanto a funcionalidades y diseño. Para asegurarnos así de que sería fácil de comprender y utilizar, para personas de diferentes edades con cierta experiencia o no en aplicaciones, y que suelen ir o no a estudiar en las bibliotecas.

Después de tener claro como queríamos que fuese la aplicación web y móvil, nos dispusimos a realizar en primer lugar la página web. Como teníamos conocimientos previos sobre la realización de las mismas, debido a que habíamos hecho numerosos proyectos a lo largo de la carrera, nos pudimos dividir las tareas de la página web.

En mi caso realicé el buzón de sugerencias que aparece en la página web, primero desarrolle el diseño de como sería el buzón de entrada, y la pantalla de envío de un mensaje. Una vez esta el diseño acabado, puse en una reunión con los miembros del equipo, mi propuesta de buzón de sugerencias. Tras algunas mejoras en diseño que observamos, recompuse el buzón hasta que definitivamente el diseño estaba completamente acabado.

Una vez terminado el diseño comencé con la lógica del mismo, tanto la conexión con la base de datos, como las demás funciones de la misma al recibir, enviar o mostrar un mensaje del mismo. También en numerosas ocasiones tuve que realizar una búsqueda de información, ya que había aspectos que no sabía como implementarlos o llevarlos a cabo.

Siguiendo el mismo proceso, realizamos una reunión con los miembros y una puesta en marcha para probar las funciones y encontrar fallos. Una vez corregí estos fallos, integré esta función en el proyecto, juntándolo con las funciones de la página web.

Más tarde, cuando terminamos la aplicación web, nos dispusimos a la realización de la aplicación Android. En la plataforma de Android Studio, la cual nunca había desarrollado ningún proyecto en dicho framework. Como había un miembro en el grupo, que tenía conocimientos de Android, ya que se matriculó en una asignatura orientada a este tipo de desarrollo en móviles, hicimos una reunión para que nos enseñase o fundamental. Una vez nos enseñó lo básico para empezar, durante varias semanas estudié por mi cuenta varias guías, y tutoriales [21] de desarrollo con Android.

Después de tener experiencia en la programación en estos dispositivos móviles, comenzamos los tres miembros del grupo a la realización de iniciar sesión y registrarse. Trabajábamos en paralelo los tres, investigando y desarrollando dicha pagina. Gracias a los aportes de todos los miembros conseguimos realizar estas páginas y tras esto reunimos mucha experiencia.

A continuación, teníamos que elegir entre realizar las páginas de estadísticas, buzón de sugerencias,u horarios que ofrecían una menor dificultad, o comenzar con la página de plantas ya que era la página más importante del proyecto.

Elegimos comenzar con la página de plantas, que a pesar de su dificultad, al no tener demasiada experiencia, nos importante comenzar con esta parte por si acaso no hubiéramos tenido tiempo suficiente de realizar las demás páginas.

Para comenzar a desarrollar dicha pantalla, primero realizamos una reunión los tres miembros junto con el director, para poder pensar como podríamos implementar el diseño de las mesas y los puestos de la biblioteca. Aporte al igual que el resto de mis compañeros ideas de como realizarlo. Hasta que se nos ocurrió realizarlo con botones, cada puesto de estudio.

Fueron muchas reuniones las que hicimos los tres miembros para poder realizar dicha pantalla, hasta que conseguimos realizarla trabajando tanto los tres juntos presenciales, como cada uno en paralelo.

A continuación tuvimos que pensar como poder conocer la ubicación de un usuario cada cierto periodo de tiempo, para determinar si se haya o no en la biblioteca. Para ello realizamos varias reuniones con el director y tras varios aportes tanto mio, como del resto de los miembros tuvimos la idea de utilizar un *Service*, para obtenerla cada cierto tiempo. Esta fue el periodo más costoso y en la que nos encontrábamos mas perdidos a la hora de saber como solucionarlo, pero gracias a esa idea pudimos solventarla.

Un vez terminamos la página de las pantallas, nos dividimos las funciones que quedaban, y que para seguir el orden normal que habíamos llevado en la aplicación web, me asigné el buzón de sugerencias de la aplicación móvil.

El proceso fue el mismo, primero me centré en el diseño tanto del buzón de entrada como de enviar mensajes, una vez termine de implementar dicho diseño, hicimos una reunión para hablar posibles modificaciones en cuanto a colores, estilos... Tras cambiar dichos aspectos, implementé la lógica del buzón de sugerencias como fueron el desarrollo de los eventos, inserción actualización en la base de datos, entre otras funciones.

Tras acabar esta función juntamos las funciones que nos habíamos asignado y realicé junto con el resto de mis compañeros, las pequeñas funciones que quedaban como por ejemplo, modificar contraseña o cerrar sesión.

Cuando acabamos todas las funciones lanzamos a Play Store para que la aplicación fuese probada y corregimos los pequeños errores que había.

Por último he colaborad como el resto del grupo, en la realización de la memoria, asignándonos diferentes capítulos, y luego leyendo y modificando las redacciones de quien lo había hecho desde un primer momento. Por lo que cada capítulo que aparece ha sido hecho y leído por los tres integrantes del grupo.

9.4.2. David Gorracho San Juan

Al igual que el resto de mis compañeros, he colaborado en muchas partes del proyecto, investigando en multitud de tecnologías, y aportando, al igual que el resto de los integrantes del grupo, el máximo rendimiento posible.

Cuando comenzamos el proyecto, por el mes de octubre, lo primero que hicimos fue plantear cómo desarrollar el tema que desde el primer momento propusimos. Nos sentamos los tres junto a nuestro director y, partiendo desde la idea inicial que teníamos, pensamos en qué cosas se podían implementar fácilmente y que sabíamos cómo desarrollarlas, y cuáles eran las que veíamos que podíamos desarrollar pero que para ello deberíamos aprender nuevas tecnologías y recurso.

Una vez tuvimos la idea de cómo queríamos desarrollar el proyecto, qué funciones extra iban a aparecer y cómo iba a ser mostrado al usuario, nos dispusimos a realizar los bocetos correspondientes de las diferentes pantallas y funcionalidades que habría en las aplicaciones.

Tras tener realizados tanto yo como mis compañeros del grupo los bocetos repartidos en bloques, los pasamos a formato digital mediante un programa de generación de bocetos. A medida que una nueva pantalla era realizada, acordábamos una nueva reunión para estar todos de acuerdo en cómo había quedado finalmente desarrollada dicha pantalla y qué funciones y cómo habían sido implementadas.

Una vez finalizado el trabajo anterior me dispuse a obtener, al igual que el resto de mis compañeros, información y feedback de los posibles usuarios de la aplicación. En un primer momento, obtuve opinión del círculo más cercano (novia, padres y familia) para después acabar recibiendo Feedback también del grupo de amigos y estudiantes de la facultad.

Tras obtener este Feedback, tuvo lugar una reunión en la que compartimos impresiones sobre las opiniones que habíamos recolectado previamente. Consideramos cuáles de ellas eran más relevantes y nos repartimos los diferentes aspectos a modificar por recomendación de los usuarios.

Además, una vez estos cambios fueron realizados, nos centramos todos juntos en la creación de la página web. Tras un día entero en la facultad que sirvió para sentar las bases, diseño y distribución final de la página web, nos repartimos las funcionalidades a desarrollar de la misma, para así poder ir avanzando en paralelo. En mi caso, decidí encargarme de la parte de estadísticas.

En lo que concierne a esta parte, he de destacar que la implementación no fue fácil. Quería que se tratase de una funcionalidad que aportase la mayor cantidad de información a los bibliotecarios y para ello precisaba de una implementación que les permitiese modificar dinámicamente el rango de fechas a consultar. Finalmente, tras una profunda investigación por la red di con el recurso que necesitaba, Highcharts.

Además, era necesario desarrollar una query que permitiese recopilar todos los datos que era necesario mostrar en la gráfica. Tras algún que otro problema, gracias a la experiencia adquirida durante estos años en MySQL logré dar con la query que obtuviese de la base de datos todo ello.

Finalizada ya la funcionalidad de estadísticas, realicé la funcionalidad web de limpiar base de datos, funcionalidad que, como ya se ha mencionado, finalmente no ha sido implementada en la web si no que se ha optado por automatizar. En un principio la automaticé con el planificador de eventos de MySQL pero finalmente la realicé mediante un CRON Job, debido a la restricción

existente en los servidores de la red con el planificador de eventos de MySQL.

Otra de los aspectos en los que más he trabajado, además de en la implementación de la automatización de tareas mediante CRON Jobs, ha sido en la generación, implementación y desarrollo de los PHP.

Después de juntar estas funciones con el resto de mis compañeros, realizamos una reunión para asegurarnos de que no faltaba ninguna funcionalidad y poder empezar a desarrollar la aplicación móvil.

A continuación empecé a ver tutoriales, junto al resto de mis compañeros, y guías de cómo desarrollar aplicaciones Android, debido a que no teníamos ninguna experiencia en este aspecto. Uno de los miembros del grupo, Iván, sí que tenía experiencia en el tema puesto que había estado matriculado en una asignatura de aplicaciones Android y fue el que nos guió en estos compases iniciales.

Tras este proceso de aprendizaje, empezamos los tres miembros del equipo a implementar las primeras pantallas de la aplicación, es decir iniciar sesión y registrarse. Hicimos varias reuniones todos juntos para poder realizarlas, a la vez que buscábamos información y trabajábamos en paralelo para poder conseguir que funcionasen las pantallas. Fue bastante costoso el inicio, puesto que ninguno de nosotros tenía experiencia anterior a este tipo de proyectos.

Una vez conseguimos terminar estas pantallas, decidimos en una reunión realizar la pantalla de plantas, donde se muestra el estado de la biblioteca actual. Acordamos una reunión los miembros del equipo con el director del proyecto e hicimos una lluvia de ideas de cómo poder desarrollar e implementar dicha funcionalidad. Tras dicha reunión, decidimos implementar los puestos mediante botones.

Debido a que era esta pantalla de la aplicación donde se encontraba la principal funcionalidad del proyecto, decidimos realizarla juntos, haciendo reuniones presenciales casi todos los días. Además, los días que no nos encontrábamos personalmente, cada integrante del grupo dedicaba tiempo individual para buscar recursos o tecnologías que ayudasen. Tras el diseño de las diferentes plantas, tuvimos que investigar cómo implementar el tema de la ubicación y conseguir cada cierto período de tiempo la obtención de la misma.

Una vez implementamos la funcionalidad principal del proyecto, nos aseguramos de su funcionamiento y diseño, nos repartimos las funcionalidades sobrantes. En mi caso me encargué de estadísticas, ya que fue esta parte también la que realicé en la página web, de modo que, a priori, me sería más fácil implementarla a mí.

Sin embargo, no muchas veces lo que se espera en teoría es lo que acaba ocurriendo en realidad y esta fue una de ellas. Para empezar, solo una query de la parte web sirvió para la implementación de esta funcionalidad en la parte móvil y. Por ello, me dispuse a realizar el resto de queries que permitiesen obtener la información necesaria de la base de datos. Una vez estuvo obtenida dicha información y verificada que fuera correcta, me centré en cómo mostrarla gráficamente. Hice uso de una de las librerías de Android, investigué con ella acerca de cómo usarla y me dispuse a la creación de los gráficos. Además, para la interfaz gráfica decidí hacer uso de pestañas, funcionalidad que posteriormente ha sido usada en el proyecto.

Una vez acabamos todas las funcionalidades restantes, y realizamos las pruebas, subimos la aplicación a Play Store para que las personas que se la descargasen pudiesen probarla. Después de varios reportes de errores, tanto mis compañeros como yo estuvimos arreglando fallos que se

habían producido, muchos de ellos por problemas de compatibilidad en las versiones no puras del Android.

Para finalizar, también participé junto al resto de mis compañeros en la realización de la memoria. Nos dividimos los capítulos y cada vez que uno de los integrantes del equipo terminaba uno de ellos, los demás lo leíamos, corregíamos y reescribíamos algún punto que nos parecía que quedaba mejor de otro modo, siempre consultándolo con los demás.

Para terminar quería decir que ha habido momentos a lo largo del proyecto, en los que estebamos algo atascados, como en el proceso de determinar la ubicación del usuario, o cuando necesitamos de un *Service* en segundo plano y no daban los resultados que esperábamos. Creo que ha sido algo positivo el hecho de trabajar en grupo puesto que nos apoyábamos unos a los otros en momentos difíciles para tratar de sacarlo adelante, siendo siempre un grupo unido.

9.4.3. Iván Monterrubio Cerezo

Comenzamos el proyecto reflexionando de como sería la aplicación web y móvil, mediante una lluvia de ideas. Pensé tanto yo como el resto de los compañeros del equipo en que funcionalidades habría que tener en cuenta antes de comenzar con el desarrollo del proyecto.

Comenzamos desarrollando bocetos de como iba a ser la aplicación web y móvil, como serían las pantallas, que funciones se le iban a mostrar al usuario a primer vista, que otras funciones podrían estar recogidas en diferentes módulos...

Tras tener un esquema principal de como serían las aplicaciones comenzamos a desarrollar en diferentes programas unos prototipos del aspecto de ambas aplicaciones. Una vez los teníamos desarrollado por completo, enseñamos nuestros bocetos a personas como amigos y familiares, usuarios no expertos en la aplicación para que valorasen como iban a encontrar nuestra aplicación.

Después de varias modificaciones en estos bocetos, comenzamos a dibujar que bases de datos necesitaríamos, y como se iban a relacionar las bases de datos con la aplicación.

A continuación comenzamos desarrollando la aplicación para web debido a que teníamos experiencia en el desarrollo de las mismas, por varias asignaturas que habíamos tenido a lo largo de la carrera sobre este tipo de proyectos. Así que como teníamos experiencia en el desarrollo de este tipo de proyectos, nos dividimos los módulos de funciones, yo me encargue del horario.

Para la realización de la misma, quería que no solo el bibliotecario/a pudiera determinar la hora de apertura y cierre si no también que pudiera crear nuevas plantas y llamarlas de diferente manera, eso fue uno de los aspectos que más me costó. Tuve que investigar bastante para encontrar unos recursos que me ayudaran a conseguir mostrar bien los horarios. Una vez realicé eso y había puesto diseño a los horario, concertamos una reunión los integrantes del equipo para poder determinar los fallos que podría haber y discutir el diseño.

Una vez corregidos los fallos, implementé con ayuda de una API de geolocalización un mapa para poder mostrar la ubicación de la biblioteca.

Después de acabar con el desarrollo de la aplicación web, pasamos al desarrollo de la aplicación móvil. Yo era el único que había tenido experiencia en el desarrollo de aplicaciones móviles, debido a que el año anterior había cursado la asignatura de Programación de Dispositivos Móviles, y había realizado algún proyecto de este tipo. Por lo que les enseñé a Carlos y David, las diferentes formas de realizar una aplicación móvil, como pudiera ser en nativo o con aplicando tecnologías web. Tras hacer varias reuniones donde vimos los pros y los contras entre una y otra opción, terminamos por decidirnos por hacer la aplicación exclusiva para Android. Les enseñé cuál era el framework por el que nos íbamos a ir moviendo, y proyectos básicos para realizar varias pruebas. A continuación tanto ellos como yo investigamos y aprendimos sobre el desarrollo de aplicaciones.

La primera pantalla, la de iniciar sesión y registrarse, la hicimos juntos, para poder ir cogiendo soltura. Una vez depuramos la primera pantalla y nos aseguramos de que funcionaba, y que estábamos de acuerdo en el diseño, decidimos seguir con el desarrollo y empezamos el desarrollo de la página fuente del proyecto, plantas.

Dicha pantalla la realizamos entre los tres componentes del grupo, debido a su complejidad. Primero realizamos una reunión los miembros del grupo, con el director, para poder hacer una lluvia de ideas de como hacer la realización del diseño y como representar los puestos de estudio y las mesas. Tras esta charla salimos con la idea de realizarla mediante botones. Una vez habíamos hecho esto, vino lo más complicado y en lo que invertimos más tiempo, y era saber como obtener la ubicación del usuario en cierto periodo de tiempo. Esto nos llevó muchas reuniones para poder llegar alguna solución, justamente la que tenemos ahora mismo implementada.

Una vez conseguimos terminar el pilar principal del proyecto, nos aseguramos de que no había fallos, empezamos a dividir las funciones restantes entre los miembros del equipo, yo realicé horarios.

Dicha funcionalidad fue mas levadura que la anterior mencionada. Me encargué de leer de la base de datos los horarios y mostrarlos. Una vez había acabado esto, realizamos una reunión para determinar el diseño de los horarios y conforme a esto corregirlos. Una vez corregido añadí esta función al proyecto.

Después de terminar el desarrollo del proyecto, lanzamos la aplicación a la conocida Play Store, para que la gente la pudiese probar. Después de esto, nos enviaron varios reportes los usuarios de fallos y problemas y los acabamos corrigiendo los tres integrantes del grupo.

Una vez finalizado el tema de la aplicación móvil, desarrollé a la par de mis compañeros los capítulos de la memoria, cada vez que un integrante desarrollaba un capítulo los otros dos lo revisaban y corregían.

Por último destacar el trabajo realizado por el grupo, debido al esfuerzo que hemos realizado durante todo el año, en situaciones en la que no estaba muy claro como avanzar con el proyecto.

Además creo que hemos sabido ser un buen grupo de trabajo, apoyándonos, buscando todos información, y siempre con una buena organización. Desde un primer momento siempre quisimos en desarrollar un proyecto que fuese de utilidad y eso fue una de las cosas que nos motivó a seguir el proyecto y a saber disfrutar mientras lo realizábamos.

10. Bibliografía

- [1] Página web en la que aparecen los puestos disponible de los laboratorios de la FDI - <http://informatica.ucm.es/puestos-disponibles-por-laboratorio/>
- [2] Página web en la realizar la reserva de salas de trabajo en grupo de la facultad de educación de la UCM - http://cisne.sim.ucm.es/record=b3601291~S6*spi
- [3] Aplicación móvil de Cinesa - <https://play.google.com/store/apps/details?id=com.codiwans.cinesa>
- [4] Aplicación móvil de PLM - <https://play.google.com/store/apps/details?id=com.ingeniacom.plm>
- [5] Ventajas y desventajas de JSON y de XML - <https://www.oscarblancarteblog.com/2014/07/18/json-vs-xml/>
- [6] Ventajas y desventajas de MySQL - <http://www.foc.es/2013/04/11/988-razones-por-la-que-utilizar-mysql.html>
- [7] Summary of Don Norman's Design Principles - <http://www.csun.edu/science/courses/671/bibliography/preece.html>
- [8] Documento de estilo web de la UCM - <https://www.ucm.es/ssii/introduccion>
- [9] Font Awesome Icons - https://www.w3schools.com/icons/fontawesome_icons_intro.asp
- [10] Glyphicons Icons - https://www.w3schools.com/icons/bootstrap_icons_glyphicons.asp
- [11] Principios aplicación móvil - <http://appdesignbook.com/es/contenidos/patrones-interaccion-moviles/>
- [12] Material Design - <https://material.io/design/>
- [13] Roboto - <https://fonts.google.com/specimen/Roboto>
- [14] Pegatina NFC - <https://computerhoy.com/paso-a-paso/apps/etiquetas-nfc-como-usarlas-que-sirven-4351>
- [15] Descarga de la app Biblioteko desde Google Play - <https://play.google.com/store/apps/details?id=es.tfg.montevichomasters>
- [16] Highcharts, librería escrita en Javascript que permite la creación de gráficas - <https://www.highcharts.com/>
- [17] Android Developers - <https://developer.android.com/>
- [18] Código PHP del algoritmo Punto en Polígono (PIP) - <http://assemblysys.com/es/algoritmo-punto-en-poligono/>
- [19] Código Java del algoritmo Punto en Polígono (PIP) - <https://github.com/sromku/polygon-contains-point>

- [20] Cuestionario realizado para la evaluación de la aplicación - <https://docs.google.com/forms/d/e/1FAIpQLSdOKCU0aHXo3tfffTLBjTn1ksODsAp0xEoZtDOKgPCv2iU609Q/viewanalytics>
- [21] Curso de Programación Android sgoliver - <http://www.sgoliver.net>

11. Apéndices

11.1. Apéndice A: Manual de instalación

Como el pensamiento que se tenía antes de realizar el proyecto era realizar una aplicación web y móvil fácil de usar es por ello que para poder probar la aplicación y para su instalación se trata de un proceso simple. Para probar la aplicación tan solo es necesario descargarse de Google Play la aplicación ‘Biblioteko’ [15] y para acceder a la página web en el link <http://biblioteko.es/>

Se puede acceder al contenido de todo el proyecto a través de una carpeta de Google Drive habilitada para ello. En ella se puede encontrar esta memoria, una carpeta llamada ‘Aplicación móvil’ y otra ‘Aplicación web’. En la primera se encuentra el archivo .apk listo para instalarse en cualquier dispositivo Android y otra carpeta que contiene el proyecto de Android Studio, con todos los archivos, código e imágenes que lo componen. En la segunda de las carpetas se puede ver un archivo .sql con la base de datos de la aplicación web, y otra carpeta a su vez que contiene el árbol de directorios completo de la página web.

A continuación se van a describir los recursos necesarios para poder ver todo el proyecto desarrollado en su totalidad:

11.1.1. Página web

Para visualizar la página web bastará con visitar el siguiente enlace: <http://biblioteko.es/>

Si una biblioteca quiere darse de alta en la plataforma el proceso a seguir es ponerse en contacto con los desarrolladores a través del apartado Contáctanos de la web, desde donde se establecerán los requisitos y se solicitarán todos los datos necesarios para que el equipo desarrollador cree esa nueva biblioteca en ambas aplicaciones. Cuando se haya finalizado, se facilitará a la biblioteca sus datos de acceso y podrá acceder a la web de manera normal.

Si se quiere visualizar en local, primero deberemos de descargar el archivo .sql y la carpeta ‘biblioteko’, que contiene el código para visualizar la página web, ambos se adjuntan en una carpeta de Drive, para la creación de las tablas en la base de datos. A continuación, activar la conexión conxampp y con mysql. Luego se podrá visualizar la página web en el siguiente enlace: <http://localhost:8080/biblioteko>

11.1.2. Android Studio

Utilizado para el desarrollo de la aplicación móvil. Se puede descargar la versión 3.1.2 para Windows en este enlace <https://developer.android.com/studio/?hl=es>

11.1.3. Sublime Text

Aunque no es necesario, se recomienda para poder leer los diferentes archivos HTML, CSS, PHP y Javascript un editor de texto. En este proyecto se ha programado utilizando dicho editor, muy recomendable. A continuación se adjunta un link para la descarga: <https://www.sublimetext.com/3>

11.1.4. MySQL

Para la base de datos, se tendrá que descargar este gestor de base de datos. Se puede hacer la descarga en <https://www.mysql.com/downloads/>

11.1.5. Aplicación Android

Una vez instalado Android Studio, y descargada la carpeta 'montevichomasters', podremos abrir el proyecto seleccionándolo. También podremos ejecutar un emulador dentro de este framework. También se puede hacer una prueba instalando el archivo .apk que se adjunta con el proyecto.

11.2. Apéndice B: Manual de usuario

En este manual se explicará detalladamente las funciones que puede realizar el usuario, para facilitar el uso de la aplicación web y móvil.

11.2.1. Aplicación web

11.2.1.1 Iniciar sesión

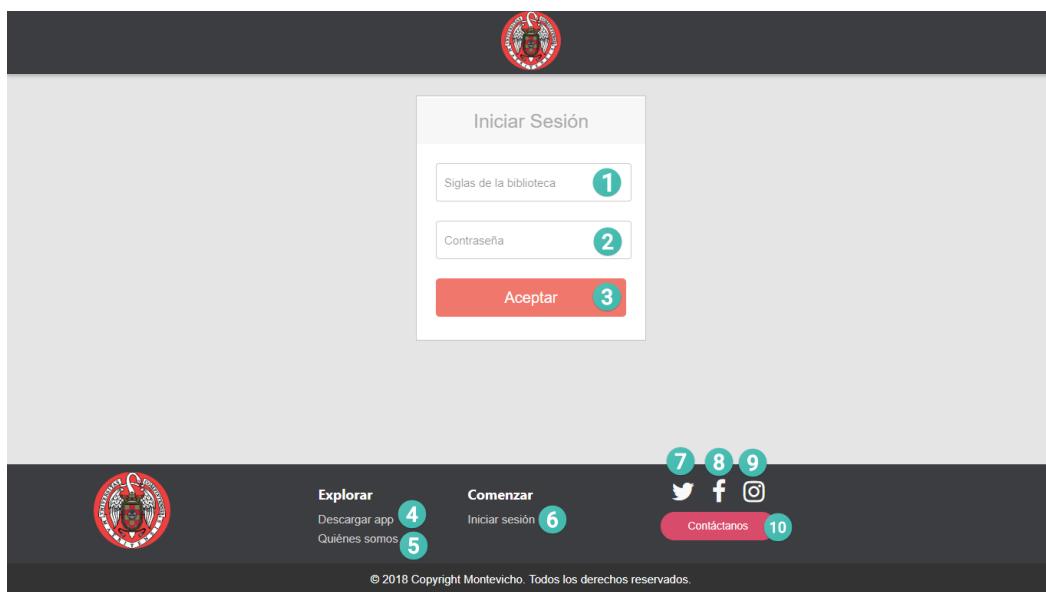


Figura 115: Manual - Iniciar sesión

Al ser esta página exclusiva para los bibliotecarios, no podrán iniciar sesión otra persona que no tengan este rol en la facultad. Para poder iniciar sesión, los administradores (que seremos nosotros) de la aplicación deberán ponerse en contacto con los bibliotecarios para proporcionar un usuario y contraseña. Este usuario y contraseña cambiará dependiendo de la biblioteca, por lo que no será la misma.

Para iniciar sesión correctamente se deberá llenar el usuario (campo 1) y la contraseña (campo 2) y darle a la opción de aceptar(campo 3). En caso de no haber cometido fallos, se pasará a la pantalla principal, el buzón de sugerencias.

En el pie de página podremos volver a esta pantalla (campo 6), teniendo la opción para poder pulsarla en cualquier momento.

11.2.1.2 Buzón de sugerencias

Respondido	Fecha	Asunto	
<input type="checkbox"/>	2018-05-24 11:18:26	Libros MMI	<button>Mostrar</button>
<input type="checkbox"/>	2018-05-22 12:05:27	libros mnl	<button>Mostrar</button>
<input type="checkbox"/>	2018-05-21 14:46:43	jssjjjs	<button>Mostrar</button>
<input type="checkbox"/>	2018-05-21 14:46:11	La Biblioteca me ha amonestado	<button>Mostrar</button>
<input checked="" type="checkbox"/>	2018-05-21 12:44:34	hola	<button>Mostrar</button>
<input checked="" type="checkbox"/>	2018-05-07 08:17:33	Prueba de ECO	<button>Mostrar</button>
<input checked="" type="checkbox"/>	2018-04-27 12:09:00	Prueba Gorricho	<button>Mostrar</button>
<input type="checkbox"/>	2018-04-25 10:43:37	f	<button>Mostrar</button>
<input type="checkbox"/>	2018-04-25 10:42:12	ewq	<button>Mostrar</button>

Figura 116: Manual - Buzón de sugerencias

Esta será la primera página que mostrará al iniciar sesión, se podrá ir a la página de establecer horario (campo 2), a la página de estadísticas (campo 1) o a la página de cerrar sesión (campo 3) pulsando en ellos.

Aquí se podrá ver en una primera instancia los correos recibidos por los usuarios que utilicen la aplicación.

Los correos están ordenados por fecha en orden descendente, por cada mensaje se puede observar en primer lugar el ícono con un tick, si el mensaje ha sido contestado, o un cuadrado vacío en caso contrario, la fecha, el asunto y un botón para mostrar el diálogo de mensajes.

Si pulsamos en el botón de mostrar (campo 4) de un mensaje, podemos ver los siguientes:



Figura 117: Manual - Mensaje

En caso de que el mensaje no este respondido(si esta el icono del cuadrado vacío), el mensaje del usuario y un cuadro de texto (campo 2) para poder responder a dicho usuario. A continuación podrá borrar el texto escrito (campo 4) o enviarlo al usuario (campo 3), también podrá cerrar esta pestaña (campo 1). En cambio si se ha respondido(hay un tick en el cuadrado), se podrá ver el mensaje recibido y el mensaje que se envió al usuario en su momento.

Por ultimo se puede observar que como máximo hay 10 mensajes por pagina, por lo que para ver los mensajes más antiguos se podrá acceder a las páginas posteriores y anteriores.

11.2.1.3 Estadísticas



Figura 118: Manual - Estadísticas

En esta parte de la página web, podremos ver las estadísticas de ocupación y de descanso de los alumnos, en los diferentes meses y años, además si dejamos el cursor encima del a gráfica, se mostrará los datos de ese día en concreto (campo 2).

Se determinar de qué mes a qué mes se quieren observar las estadísticas con el cuadro que aparece arriba al derecha o acortando o alargando la gráfica para ver los meses de interés (campo 3). Si nos fijamos en el lado izquierdo se puede observar que se puede hacer diferentes tipos de zoom en la gráfica (campo 1).

Aparte también se dispone, en un icono de tres lineas horizontales de la posibilidad de descargar la gráfica en numerosos formatos.

11.2.1.4 Establecer Horario

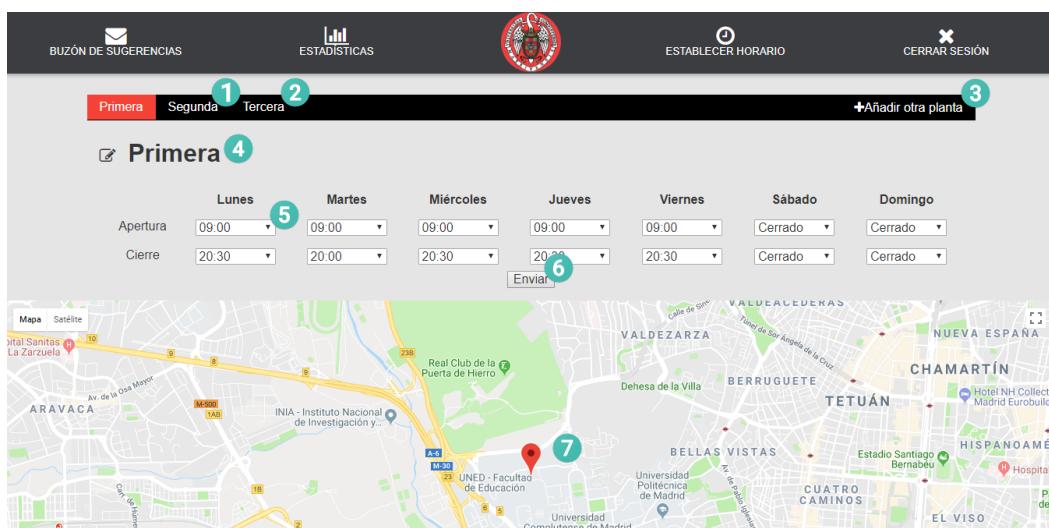


Figura 119: Manual - Establecer horario

Hay varias funcionalidades en esta pantalla, se podrá crear plantas para la biblioteca (campo 3). Cuando se creen estas plantas, aparecerán en las pestañas (campo 2 y 3), además se podrán renombrar (campo 4). Para cada planta se pueden configurar los horarios de apertura y cierre, mediante un desplegable muy fácil de usar (campo 5). Una vez hayamos configurado los horarios podemos guardar los cambios, dándole al botón de enviar (campo 6)

Si observamos abajo también esta representado un mapa, que representa la ubicación de la biblioteca en la que se ha iniciado sesión (campo 7).

11.2.1.5 Descargar app

Podemos ver esta opción en la Figura 115 en el campo 4. Esta página será accesible tanto si ha iniciado sesión, como en caso contrario y llevará a la página de Play Store para la descarga de la aplicación.

11.2.1.6 Quiénes somos

Podemos ver esta opción en la Figura 115 en el campo 5.

Esta página será accesible tanto si ha iniciado sesión, como en caso contrario y se podrá observar los integrantes del grupo.

11.2.1.7 Contáctanos/Redes Sociales

Podemos ver esta opción en la Figura 115 en los campos 7-10.

Esta página será accesible tanto si ha iniciado sesión, como en caso contrario y podremos contactar directamente mediante email con nosotros mediante un email de contacto (campo 10) y se podrá visitar las cuentas de las redes sociales (campo 7-9) donde nos hemos registrado y que están en uso.

11.2.1.8 Cerrar sesión

Podemos ver esta opción en la Figura 116 en el campo 3.

Una vez se haya registrado, si se quiere acabar con la sesión se podrá cerrar. Volveremos a la página de iniciar sesión de nuevo.

11.2.2. Aplicación móvil

11.2.2.1 Iniciar sesión

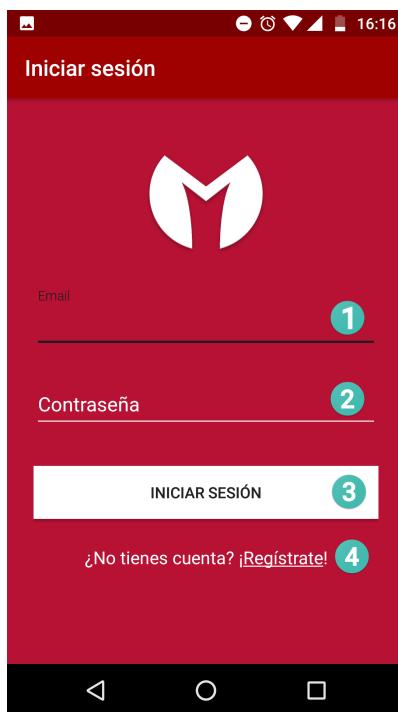


Figura 120: Manual - Iniciar sesión

Será la primera pantalla que se mostrará al usuario, tendrá que llenar los campos de email (campo 1) y contraseña (campo 2), una vez llenados tendrá que pulsar en el botón iniciar sesión

(campo 3) y que el usuario este dado de alta en la base de datos y activado. Si es correcto pasará a la pantalla principal de 'bibliotecas'.

En caso de no tener cuenta, puede pulsar en el enlace (campo 4), que le redirigirá a la página de registrarse.

11.2.2.2 Registrarse

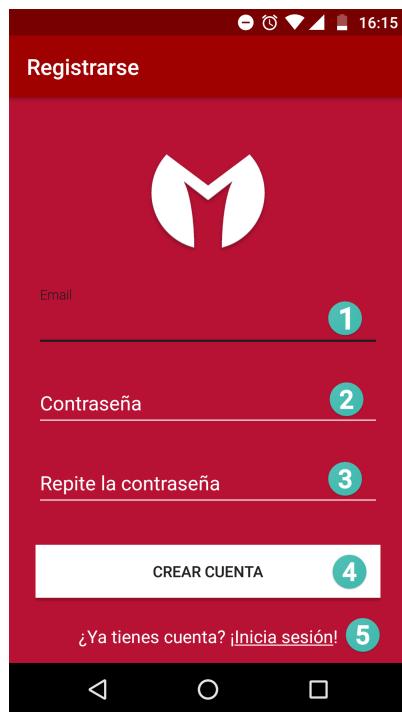


Figura 121: Manual - Registrarse

En caso de no tener una cuenta en la aplicación, se podrá dar de alta. Para ello se tendrá que llenar los tres campos, email (campo 1), contraseña (campo 2) con 8 caracteres y dígitos y repetir de nuevo la contraseña (campo 3) para asegurarse de que se ha escrito bien. A continuación deberá pulsar en el botón de 'crear cuenta' (campo 4), y si ese email no existe en la base de datos, es decir no se ha dado de alta antes, las contraseñas coinciden y la contraseña tiene 8 dígitos con letras y números, entonces se habrá podido registrar.

Una vez registrado se enviará al email un correo de activación, el cuál habrá que seguir el link para terminar este proceso y poder iniciar sesión.

En caso de ya poseer una cuenta, y acceder a la ventana de 'iniciar sesión', se tendrá que pulsar en el enlace de abajo (campo 5).

11.2.2.3 Bibliotecas



Figura 122: Manual - Bibliotecas

Será una lista de todas las bibliotecas que se puede elegir para poder ver su estado actual. Actualmente solo está implementada la primera, Biblioteca de Facultad de Ingeniería Informática.

Para poder acceder a ver el estado de una biblioteca, hay que pulsar sobre la biblioteca en cuestión, al estar implementada solo Biblioteca de Facultad de Ingeniería Informática, será la única donde se podrá pulsar (campo 1).

11.2.2.4 Plantas

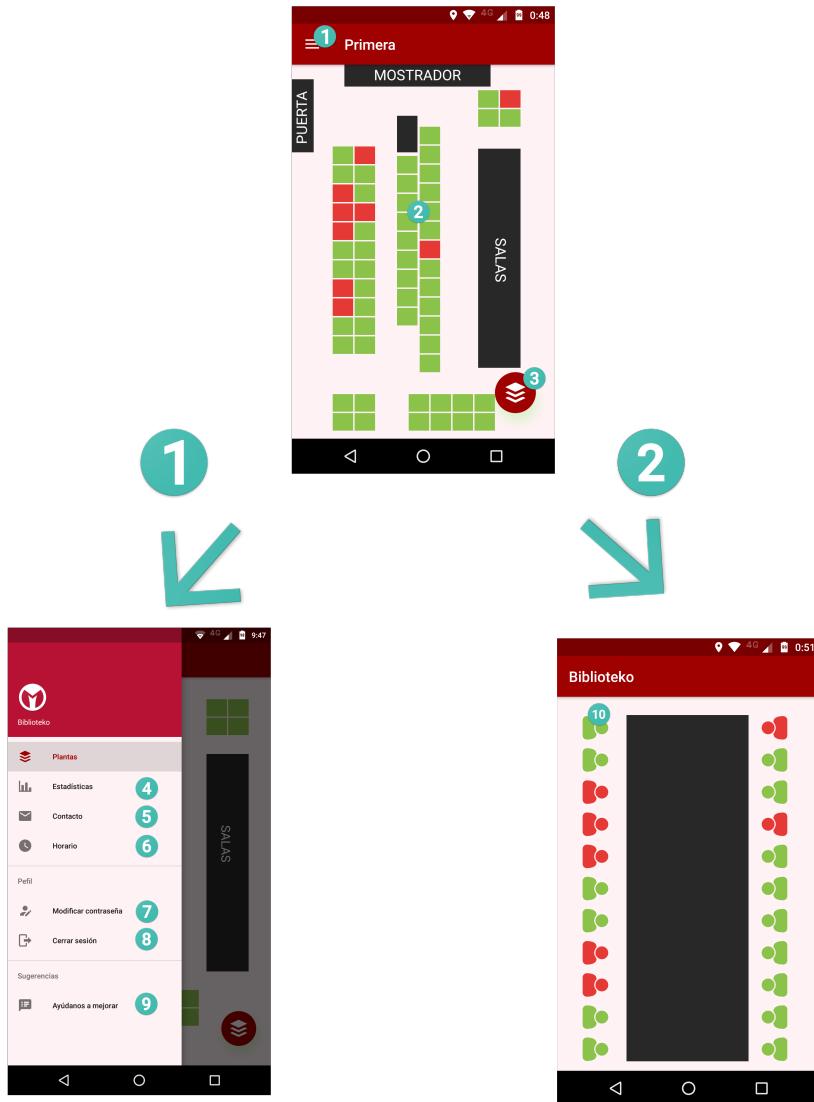


Figura 123: Manual - Plantas

En esta página se podrá ver el estado de las plantas de la biblioteca a tiempo real (pantalla de arriba), se puede cambiar de planta con el botón rojo abajo a la derecha (campo 3). Los colores de cada puesto son: verde: el sitio está libre, rojo: ocupado, azul: tu puesto, naranja: tu puesto en estado descansando.

Para poder ocupar un puesto hay que seleccionar la mesa donde se quiera sentar (por ejemplo la mesa del campo 2) y a continuación en la nueva pantalla el puesto correspondiente(campo 10). Para desocuparlo se puede pinchar en el mismo sitio donde se ha ocupado (campo 10) o en

el menú la opción de liberar puesto (al pulsar en el campo 1, una de las funcionalidades será esa).

Para poder ocupar un puesto se debe estar dentro de la biblioteca, en otro caso no se podrá ocupar.

Si se ha ocupado un sitio, y la ubicación determina que no está dentro de la biblioteca, entonces se tendrá 3 minutos para volver a la misma, en caso de no volver se desocupará el puesto automáticamente.

Como se puede observar arriba a la izquierda, hay un menú (campo 1), el cuál al desplegarse se pueden observar las funciones que se describirán en la siguientes secciones (campos del 4 al 9).

11.2.2.5 Hacer descanso

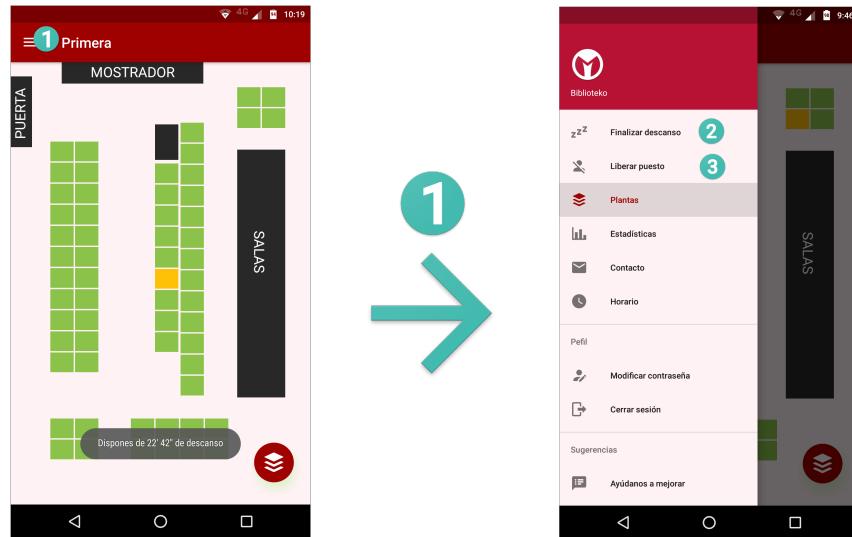


Figura 124: Manual - Hacer descanso

Esta opción está en el menú que se puede deslizar a la derecha, en la Figura 124. Para que aparezca esta opción es necesario que se haya ocupado un puesto, en otro caso esta opción no está visible (se puede observar que ya se ha pulsado esa opción, y ahora aparece, la opción de finalizar descanso, campo 2).

Si el puesto está ocupado y pulse en esta opción, el puesto se pondrá en color naranja, es decir el usuario cambiará a descansando, y que se activará un temporizador de descanso, de modo que el tiempo de descanso será el mismo al que se haya estudiado, hasta un máximo de 30 minutos. Si por ejemplo se ha estudiado 15 minutos, se tendrán 15 minutos de descanso, si se ha estudiado 1 hora, se tendrá media hora de descanso.

En caso de que se agote el tiempo, y el usuario no haya llegado a la biblioteca, se desocupará el sitio automáticamente(se avisará al usuario cuando le queden 3 minutos para concluir el

descanso), si por el contrario el usuario regresa a la biblioteca y no finaliza su descanso, se notificará al usuario para que no se olvide. En caso de que regrese a la biblioteca y haya pulsado en la opción de finalizar descanso (campo 2), cambiará su estado a estudiando, el puesto en azul, y el tiempo sobrante del descanso, se guardará para que pueda acumularlo en el siguiente descanso.

11.2.2.6 Liberar puesto

Esta opción solo aparecerá si se ha ocupado un puesto, como se puede observar en la Figura 124 aparecerá en el mismo menú del que se ha hablado antes, y la opción para liberar puesto se puede ver en el campo 3, y pulsar esta opción supondría eliminar tu puesto de estudio, por tanto el puesto pasaría de azul o naranja(se puede pulsar aunque se esté descansando), a color verde, es decir disponible para que pueda ser ocupado.

11.2.2.7 Estadísticas

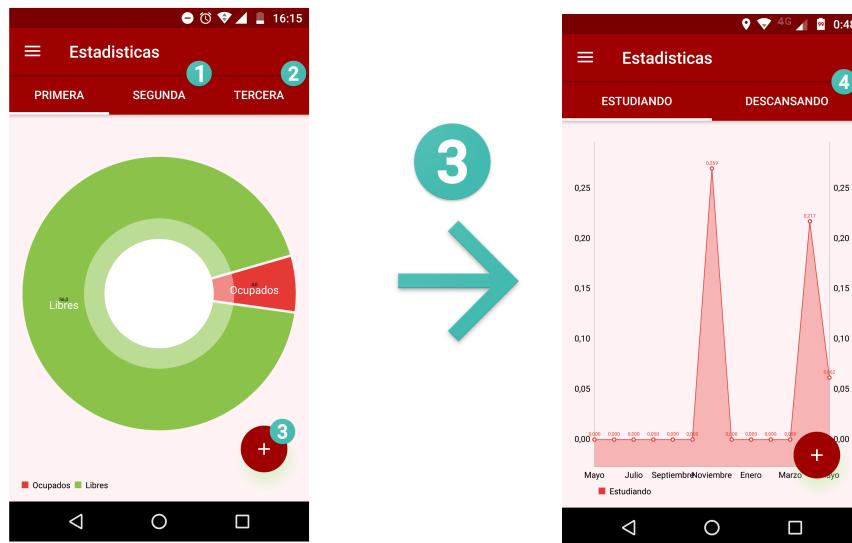


Figura 125: Manual - Estadísticas

Esta opción aparecerá en el menú anterior, Figura 123 (campo 4), y se podrán ver las gráficas de estudio diferentes plantas (campos 1 y 2).

Hay un botón rojo abajo a la derecha(campo 3), con la que poder cambiar de gráfica, a estadísticas individuales, globales, o de ocupación de sitios en la biblioteca(la primera gráfica que sale en estadísticas).

Si se observa la pantalla de estadísticas personales (imagen de la derecha), se pueden ver las estadísticas de estudio y de descanso (campo 4).

También se podrá girar las gráficas pulsando en ellas y hacer zoom en meses concretos.

11.2.2.8 Contacto

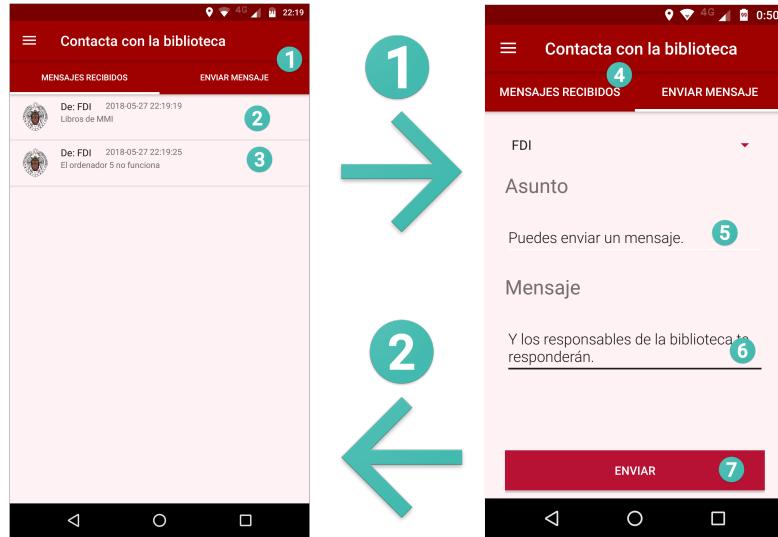


Figura 126: Manual - Contacto

Esta opción aparecerá en el menú anterior, Figura 123 (campo 5) y se podrá leer los correos que han sido respondido por los bibliotecarios, como se puede ver en la imagen de derecha de la Figura 126. Si pulsamos en un mensaje en concreto(campo 2 o 3) podremos ver el mensaje enviado y la contestación.

También se podrá escribir uno yendo a la pestaña de enviar mensaje (campo 1), seleccionando la biblioteca (en este caso solo aparece FDI, debido a que es la única que esta dada de alta). A continuación se escribe el asunto(campo 5), y el mensaje (campo 6) y por último se dará a enviar (campo 7).

Desde esta pestaña podemos volver a la lista de mensajes recibidos(campo 4).

11.2.2.9 Horario



Figura 127: Manual - Horario

Esta opción aparecerá en el menú anterior, Figura 123 (campo 6), se podrá observar los horarios por cada planta, para poder acceder a cada planta bastará con pulsar hacia las diversas plantas que hay (campos 1 y 2) y que haya configurado el bibliotecario de la biblioteca en cuestión desde la aplicación web.

11.2.2.10 Modificar contraseña

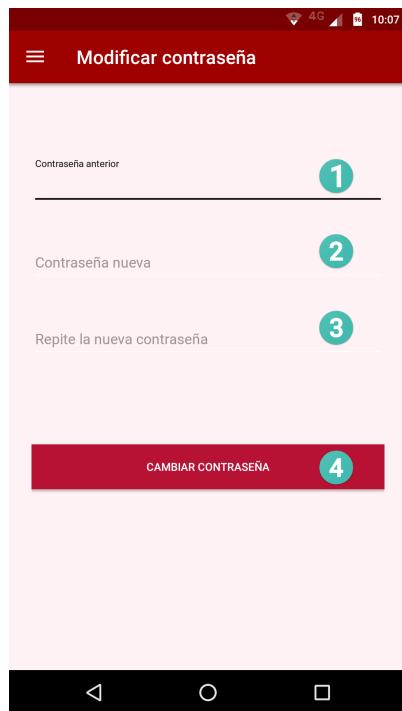


Figura 128: Manual - Modificar contraseña

Esta opción aparecerá en el menú anterior, Figura 123 (campo 7), en la sección de perfil. Dada tu contraseña anterior se podrá modificar por otra nueva, la única restricción es que la nueva contraseña tiene que tener 8 dígitos y caracteres.

Para ello habrá que llenar el campo 1 con la contraseña anterior, el campo 2 y 3 con las nuevas, ambas deben coincidir y tener la restricción mencionada antes. A continuación se pulsará en el modificar(campo 4).

Una vez modificada, en el siguiente inicio de sesión se tendrá que utilizar la nueva contraseña.

11.2.2.11 Cerrar sesión

Esta opción aparecerá en el menú anterior, Figura 123 (campo 8) en la sección de perfil. Cerrará la sesión del usuario y en caso de tener ocupado un puesto se desocupará, continuación volverá a la página de iniciar sesión.

11.2.2.12 Ayúdanos a mejorar

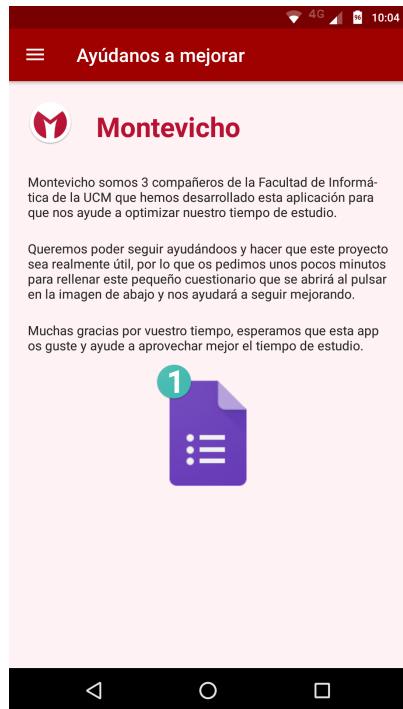


Figura 129: Manual - Ayúdanos a mejorar

Esta opción aparecerá en el menú anterior, Figura 123 (campo 9) en la sección de sugerencias. El cuál tendrá una pequeña información al usuario de quienes somos e información de la aplicación , también se adjunta un link hacia un formulario de Google (campo 1) para valorar la aplicación móvil.

12. Anexo

Se adjunta un impreso que se colocó en los tablones de la Facultad de Informática, con autorización de la Delegación de Estudiantes. Su misión fue dar publicidad a la aplicación móvil para que los usuarios la pudieran probar y contribuyeran con sus comentarios.



¡NUEVA APP PARA LA BIBLIOTECA DE LA FDI!

- ¿Te gusta **estudiar en la biblioteca**?
- ¿Te cuesta **encontrar sitio** en época de exámenes?
- ¿Quieres saber qué puestos están libres y dónde antes de entrar? ¿O desde casa antes de venir?

Podrás consultar **qué puestos están libres**, ocupar uno y **salir a descansar** sin miedo a que te lo quiten, consultar **tus estadísticas** y ver el tiempo que dedicas a estudiar y descansar, ¡y mucho más!

Pruébala ahora que se acercan los exámenes y usa **una app hecha por tus compañeros**. ¡Optimiza tu tiempo de estudio!



Figura 130: Anexo