PlayPadel

Sistema de gestión de reservas de pistas de pádel

Booking management system for paddle courts

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Memoria

Julio de 2021

Alumna: Naiara Sánchez García

Tutor: Pablo Chamoso Santos

CERTIFICADO DEL TUTOR

D. Pablo Chamoso Santos, Profesor Ayudante Doctor del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR:

Que el presente trabajo titulado “Sistema de gestión de reservas de pistas de pádel”, ha sido realizado por Naiara Sánchez García, con DNI 70899235X y este documento constituye la memoria del trabajo realizado para la asignatura Trabajo de Fin de Grado de la Titulación de Grado en Ingeniería Informática de la Universidad de Salamanca.

Y para que así conste a todos los efectos oportunos.

En Salamanca a 02 de julio de 2021

CHAMOSO SANTOS PABLO - 70888952P
2021.07.04 07:02:23 -05'00'

Pablo Chamoso Santos

RESUMEN

En este documento se detalla el proceso de desarrollo del Trabajo de Fin de Grado del proyecto "**Sistema de gestión de reservas de pistas de pádel**".

El proyecto es un sistema que usa la **arquitectura Cliente - Servidor Firebase** para su funcionamiento, en la que el dispositivo móvil (Android) obtiene el rol de cliente. Una vez la aplicación móvil sea instalada en estos dispositivos móviles, su principal función será la de **gestionar y permitir a los usuarios** realizar una reserva de la pista, el día y hora que él mismo desee, realizar el cobro de dicha reserva y proporcionar un código QR para poder acceder a dicha pista el día de tu reserva. Por tanto, el sistema debe contar con una correcta y eficiente gestión de usuarios y reservas, para que puedan reservar sin problema.

Toda la información relacionada con las reservas y usuarios estará gestionada por la **plataforma Firebase**, que se usará como servidor del sistema. Se ha optado por elegir esta plataforma por su rapidez y eficacia, y por su sencilla implantación en el sistema.

Para la verificación correcta del código QR proporcionado por la aplicación, se hace uso de **Arduino**, en el cual con la ayuda de un lector de barras se crea una simulación de la apertura de la puerta si dicho QR es válido para esa pista, ese día y a esa hora.

PALABRAS CLAVE

Aplicación, Android, gestión de reservas, pádel, Firebase.

ABSTRACT

This document has the details of the development process of the Final Degree Project " **Booking management system for paddle courts**".

The project is a system that uses the Firebase Client-Server architecture for its operation in which the mobile device (Android) obtains the client role. Once the mobile application is installed on these mobile devices, its main function will manage and allow users to make a booking for the paddle court, the day and hour that they wish, collect the booking, and provide a code QR to access the court the day of your reservation. Therefore, the system must have correct and efficient management of users and bookings, so that they can book without problem.

All information related to reservations and users will be managed by the Firebase platform which will be used as the system's server. We have chosen this platform because of its speed, efficiency, and its simple implementation in the system.

For the correct verification of the QR code provided by the application, use is made of Arduino, in which with the help of a bar reader a simulation of the opening of the door is created if the QR is valid for that paddle court, that day and at that time.

KEYWORDS

Application, Android, booking, paddel, Firebase.

Tabla de contenido

1. INTRODUCCIÓN 13
2. OBJETIVOS DEL PROYECTO 17
2.1 Objetivos marcados por los requisitos 17
2.2 Objetivos personales 18
3. CONCEPTOS TEÓRICOS 19
3.1 API 19
3.2 Kit de desarrollo de software 19
3.3 Android SDK 20
3.4 Arquitectura cliente-servidor 21
3.4.1 Ventajas del modelo cliente-servidor 22
3.4.2 Desventajas del modelo cliente-servidor 22
3.5 Base de datos NoSQL 23
4. TÉCNICAS Y HERRAMIENTAS UTILIZADAS 25
4.1 Microsoft Project 25
4.2 UML 25
4.2.1 Visual Paradigm 26
4.2.2 REM 27
4.3 Desarrollo del servidor 27
4.3.1 Firebase Authentication 29
4.3.2 Firebase Cloud Firestore 30
4.3.3 Crashlytics 31
4.3.4 Test Lab 32
4.3.5 Google Analytics 32
4.4 Desarrollo de la aplicación 35
4.4.1 Android Studio 35
4.4.2 Stripe 38

4.4.3 Arduino 38
4.4.4 Git (SourceTree) 40
5. ASPECTOS RELEVANTES DEL PROYECTO 43
5.1 Proceso unificado 43
5.1.1 Características del Proceso Unificado 43
5.1.2 Fases del Proceso Unificado 44
5.2 Planificación temporal 45
5.2.1 Fase de investigación 45
5.2.2 Fase de inicio 45
5.2.3 Fase de elaboración 46
5.2.4 Fase de implementación 46
5.2.5 Fase de transición 48
5.2.6 Elaboración de la documentación 49
5.3 Catálogo de requisitos 49
5.3.1 Objetivos 49
5.3.2 Actores 50
5.3.3 Requisitos de información 51
5.3.4 Requisitos de restricción 51
5.3.5 Requisitos no funcionales 52
5.3.6 Requisitos funcionales 52
5.4 Modelo de análisis 54
5.4.1 Modelo de dominio 55
5.4.2 Propuesta de arquitectura 55
5.5 Modelo de diseño 55
5.5.1 Subsistemas de diseño y servicio 56
5.5.2 Clases de diseño 58
5.5.3 Diagramas de despliegue 59

5.6 Implementación del sistema 60
6. CONCLUSIONES 65
7. LÍNEAS DE TRABAJO FUTURO 67
8. BIBLIOGRAFÍA 69

TABLA DE ILUSTRACIONES

Ilustración 1 – Logotipo de la aplicación 13
Ilustración 2 – Logotipo secundario de la aplicación 13
Ilustración 3 - Comparación SQL / NoSQL [77] 23
Ilustración 4 - Microsoft Project 25
Ilustración 5 - Visual Paradigm 26
Ilustración 6 - REM 27
Ilustración 7 - Firebase 28
Ilustración 8 - Firebase Authentication 29
Ilustración 9 - Firebase Cloud Firestore 30
Ilustración 10 - Estadística de fallas 31
Ilustración 11 - Problemas abiertos 31
Ilustración 12 - Test Lab 32
Ilustración 13 - Usuarios activos 33
Ilustración 14 - Interacción diaria de los usuarios 33
Ilustración 15 - Cohortes de retención 34
Ilustración 16 - Audiencia por ubicación 34
Ilustración 17 - Actividad de los usuarios 35
Ilustración 18 - Android Studio 36
Ilustración 19 - Aplicación en Android Studio 37
Ilustración 20 – Stripe 38
Ilustración 21 - Arduino IDE 39
Ilustración 22 - Arduino UNO [11] 40
Ilustración 23 - Arduino MEGA[11] 40
Ilustración 24 - SourceTree 41
Ilustración 25 - Fase de investigación 45
Ilustración 26 - Fase de inicio 46
Ilustración 27 - Fase de elaboración 46
Ilustración 28 - Fase de implementación iteración 1 47
Ilustración 29 - Fase de implementación iteración 2 47
Ilustración 30 - Fase de implementación iteración 3 48
Ilustración 31 - Fase de transición 48
Ilustración 32 - Elaboración de la documentación 49

Ilustración 33 – Actores 50
Ilustración 34 - Registro de usuarios 53
Ilustración 35 - Gestión de reservas 53
Ilustración 36 - Información 54
Ilustración 37 - Ayuda 54
Ilustración 38 - Modelo de dominio 55
Ilustración 39 - Propuesta de arquitectura 55
Ilustración 40 - Subsistema de diseño 56
Ilustración 41 - Gestión de usuarios 57
Ilustración 42 - Gestión de reservas 57
Ilustración 43 - Clase gestión de usuarios 58
Ilustración 44 - Clase gestión de reservas 59
Ilustración 45 - Diagrama de despliegue 59
Ilustración 46 - Uso del lenguaje NoSQL 60
Ilustración 47 – AlertDialog 61
Ilustración 48 - Presione dos veces para salir 61
Ilustración 49 - Sistema de pagos 62
Ilustración 50 – Vista de pagos realizados en la plataforma Stripe 62
Ilustración 51 - Envío correo ayuda 63
Ilustración 52 - Correo de ayuda 63
Ilustración 53 - Google Maps 64
Ilustración 54 - Generar código QR 64

1. INTRODUCCIÓN

En este documento se recoge la memoria del Trabajo de Fin de Grado (TFG) de Naiara Sánchez García, alumna del Grado en Ingeniería Informática de la Universidad de Salamanca. Dicho TFG consiste en el desarrollo desde cero de una aplicación para Android la cual se llamará PlayPadel.

Ilustración 1 – Logotipo de la aplicación

Ilustración 2 – Logotipo secundario de la aplicación

Muchos usuarios hoy en día se encuentran con el problema de ir a jugar al pádel a una pista que han reservado y tener que ir a por las llaves de la pista a un lugar lejano de ella o tener que esperar a que le abran la puerta (lo cual también es cansado para el administrador de las pistas). Además, actualmente en los tiempos que corren debido a la COVID-19 hay algunos clientes que no quieren andar con llaves que han sido utilizadas por otras personas o incluso los propios propietarios de las pistas no quieren tener contacto con los clientes por precaución propia. Por ello, se propone un proyecto capaz de abrir dichas puertas para aquellos clientes que han reservado sin necesidad de utilizar llaves ni comunicarse con nadie.

El objetivo de este proyecto es poder llegar a crear una aplicación Android que gestione las reservas de un conjunto de pistas de pádel donde los usuarios puedan consultar, seleccionar y pagar de la manera más intuitiva posible y sin tener ninguna complicación, así como la posterior entrada a la pista ya que con un simple código QR proporcionado por la propia aplicación se podrá entrar sin problema.

Como servidor de la aplicación se propone el uso de la plataforma Firebase [1] que se usará para manejar la lógica de datos del sistema, la gestión de los usuarios, detección de errores y varios temas más que se detallarán más adelante.

Por último, se pretende poner en práctica los conocimientos adquiridos durante la titulación para conseguir que el sistema cumpla con todos estos objetivos propuestos, así como adquirir nuevos conocimientos que serán de utilidad para el futuro.

En esta memoria se van a recoger los aspectos más relevantes del desarrollo del proyecto. Aparte de esta memoria, se adjuntarán unos anexos con la documentación más técnica.

Los apartados de esta memoria son los siguientes:

- **Introducción:** Se presenta brevemente el tema que se aborda en el proyecto, a la vez que se hace un resumen del contenido de esta memoria y de los anexos que la acompañan.
- **Objetivos del proyecto:** se especifican de forma precisa los objetivos tanto técnicos como los marcados por los requisitos del proyecto.
- **Conceptos teóricos:** se detallan algunos aspectos del proyecto que necesitan una profunda explicación teórica para su comprensión.
- **Técnicas y herramientas:** se explican las técnicas y herramientas usadas en el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** se recogen los apartados más importantes del desarrollo del sistema software del proyecto.
- **Conclusiones y líneas de trabajo futuras:** se detallan las conclusiones más relevantes fruto del desarrollo del proyecto, además de un informe detallando las partes donde se puede ampliar o mejorar el proyecto.
- **Referencias:** se recogen las fuentes desde donde se ha obtenido información para el desarrollo del documento.

Aparte de este documento, se adjunta la documentación técnica, dividida en los siguientes anexos:

- **Anexo I – Planificación temporal:** contiene la planificación temporal y la estimación de esfuerzo del proyecto.
- **Anexo II – Catálogo de requisitos:** recoge los requisitos y los objetivos principales del sistema, y lo plasma con las descripciones de los casos de uso de este.
- **Anexo III – Modelo de análisis:** Contiene los elementos de la fase de análisis del sistema, así como la realización de los casos de uso.
- **Anexo IV – Modelo de diseño:** Contiene las fases del diseño del sistema, el modelo de despliegue y el diseño de la interfaz.

- **Anexo V – Documentación técnica de programación:** Contiene la información sobre la organización y funcionamiento del sistema, para que sea entendible por otros desarrolladores.
- **Anexo VI – Manual de usuario:** Recoge los aspectos más relevantes para el buen uso del sistema por parte de los usuarios.

También se han incluido todos los ficheros fuentes en la entrega del proyecto, así como los ficheros generados con Visual Paradigm y REM. Se han adjuntado los archivos de las bases de datos listas para importar en un sistema.

2. OBJETIVOS DEL PROYECTO

En esta sección se detallan los objetivos marcados por los requisitos del proyecto y los objetivos técnicos los cuales guiarán el desarrollo de este proyecto.

2.1 Objetivos marcados por los requisitos

El objetivo principal del desarrollo de este trabajo es implementar una aplicación para dispositivos Android capaz de gestionar reservas de pistas de pádel realizando un cobro a los clientes por cada una de estas reservas y tras ello proporcionar un código QR para la posterior apertura de la puerta sin hacer uso de unas llaves. De esta forma, estos usuarios podrán consultar, reservar, pagar y visualizar las pistas existentes en el establecimiento. Así mismo, el propietario de las pistas también podrá reservar, aunque también tendrá permisos para bloquear una pista, visualizar todas las reservas realizadas por los clientes o incluso eliminar una reserva.

A continuación, se definirán los objetivos marcados por los requisitos del sistema:

- **Gestionar reservas:** El sistema deberá permitir a los usuarios gestionar reservas de las pistas que se encuentren libres en el establecimiento realizando el pago de esta con la tarjeta de crédito o débito.
- **Gestionar usuarios:** El sistema deberá contar con una gestión de usuarios. Se tiene que habilitar una pantalla de registro mediante la cual los usuarios no registrados procederán a hacerlo; así como una pantalla de login para poder acceder a su cuenta o pedir una contraseña nueva en caso de olvido. También habrá una manera de iniciar sesión con Google y Facebook.
- **Integrar la aplicación con la plataforma Firebase:** El sistema deberá integrarse con la plataforma Firebase, la cual actuará como servidor de la aplicación. Esta plataforma se utilizará principalmente para controlar la gestión de usuarios y para almacenar los datos existentes del sistema.
- **Generar, escanear y validar códigos QR:** Una vez realizada la reserva, se le proporcionará al cliente un código QR que, con ayuda de un lector y de una simulación realizada con Arduino se comprobará el día y hora que se encuentre en ese QR y si es la fecha y hora actual en la pista correcta entonces se encenderá una luz LED verde simulando la apertura de la puerta de la pista. En caso de que no lo sea se encendería una luz LED roja simulando que la puerta no se abriría.

2.2 Objetivos personales

Los objetivos personales para la alumna al finalizar este proyecto son los siguientes:

- Demostrar distintas habilidades y capacidades, mediante el diseño y construcción de una aplicación Android, la cual será creada desde cero en todos los ámbitos sin el uso de plantillas ni reutilización de código.
- Aprender a usar y familiarizarse con las aplicaciones con sistema operativo Android, y su entorno de desarrollo, Android Studio, así como el lenguaje usado en el código fuente (Java, XML).
- Aprender a manejar la plataforma Firebase, especialmente las herramientas Firebase Authentication y Firestore Database.
- Aprender el manejo de APIs (*Application Programming Interfaces)* y servicios proporcionados por los servicios de Android, Google y Firebase.
- Utilizar algoritmos, técnicas y patrones de programación aprendidos durante la titulación, así como los conocimientos de las asignaturas de Ingeniería del Software I, Ingeniería del Software II y Gestión de Proyectos para planificar el desarrollo del proyecto.

3. CONCEPTOS TEÓRICOS

En este apartado se detallarán los aspectos teóricos más importantes para la comprensión del desarrollo del proyecto.

3.1 API

La interfaz de programación de aplicaciones, conocida también por la sigla API [2], es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Una API representa la capacidad de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas del API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.

3.2 Kit de desarrollo de software

Un kit de desarrollo de software (*Software Development Kit* o SDK) [3] es generalmente un conjunto de herramientas de desarrollo de software que permite a un desarrollador de software crear una aplicación informática para un sistema concreto, por ejemplo, ciertos paquetes de software, entornos de trabajo, plataformas de hardware, computadoras, sistemas operáticos, etc.

Es algo tan sencillo como una interfaz de programación de aplicaciones o API creada para permitir el uso de cierto lenguaje de programación o, puede también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas de desarrollo de software más comunes incluyen soporte para la detección de errores de

programación como un entorno de desarrollo integrado (IDE) y otras utilidades. Los SDK frecuentemente también incluyen códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario.

Utilizaremos este término para incorporar los mapas de Google (para situar la localización del establecimiento de las pistas de pádel) y proporcionarles una API para interactuar con ellos. La API maneja automáticamente el acceso a los servidores de Google Maps, la descarga de datos, la visualización del contenido del mapa y la respuesta a los gestos que hace el usuario con este. También permite agregar marcadores, polígonos y superposiciones a un mapa básico.

Además, también se utilizará para crear un usuario mediante Google y Facebook sin necesidad de realizar un registro haciendo uso de la SDK para conectarnos a su API correspondiente.

3.3 Android SDK

El SDK de Android [4] incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, bibliotecas y emuladores, entre otros. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS, y Windows. También puede utilizarse el propio sistema Android para desarrollos utilizando las aplicaciones AIDE - Android IDE - Java, C++ y el editor de Java. La plataforma integral de desarrollo (Integrated Development Environment) soportada oficialmente es Android Studio junto con el complemento ADT (*Android Development Tools plugin*). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML, y utilizar comandos en un terminal (se necesitan los paquetes JDK y Apache Ant) para crear y depurar aplicaciones, así como controlar dispositivos Android que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.)

Las actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario, root, por razones de seguridad). Un paquete APK incluye ficheros con extensión .dex (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

3.4 Arquitectura cliente-servidor

Una de las arquitecturas utilizadas para la realización del proyecto es la arquitectura cliente-servidor. Esta arquitectura [5] es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Algunos ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son el correo electrónico, un servidor de impresión o la World Wide Web.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc.

Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

En el caso del proyecto, los clientes serán los usuarios que tengan la aplicación Android instalada y mediante esta harán peticiones al servidor (para acceder a una base de datos, por ejemplo), que residirá en la plataforma Firebase de la que se hablará más adelante.

3.4.1 Ventajas del modelo cliente-servidor

- **Centralización del control:** Los accesos, los recursos y la integridad de los datos del sistema son controlados por el servidor. Esta centralización también facilita la tarea de poner al día datos u otros recursos.
- **Escalabilidad:** Se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- **Fácil mantenimiento:** Al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de cliente-servidor que **aseguran la seguridad** en las transacciones, la amigabilidad de la interfaz, y la facilidad de empleo.

3.4.2 Desventajas del modelo cliente-servidor

- **La congestión del tráfico:** cuando una gran cantidad de clientes envían peticiones simultáneas al mismo servidor, puede ser que cause muchos problemas para este (a mayor número de clientes, más problemas para el servidor).
- El paradigma de cliente-servidor clásico puede tener **fallos de robustez**. Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas.
- El software y el hardware de un servidor son generalmente muy determinantes, siendo, de esta manera, más **costoso**.

- En las redes cliente-servidor la única forma de obtener la información es a través de la que proporciona el servidor el cual **los clientes no pueden compartir información entre ellos**.

3.5 Base de datos NoSQL

NoSQL [6] es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no sólo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

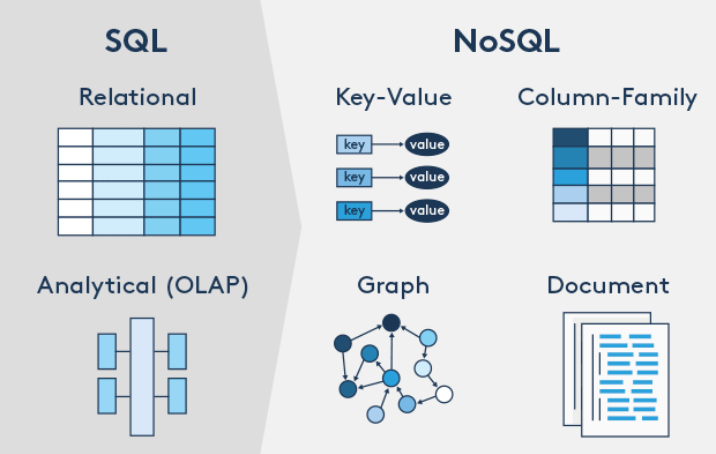


Ilustración 3 - Comparación SQL / NoSQL [77]

A menudo, las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad de almacenar los registros, por ejemplo, almacenamiento clave-valor. La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

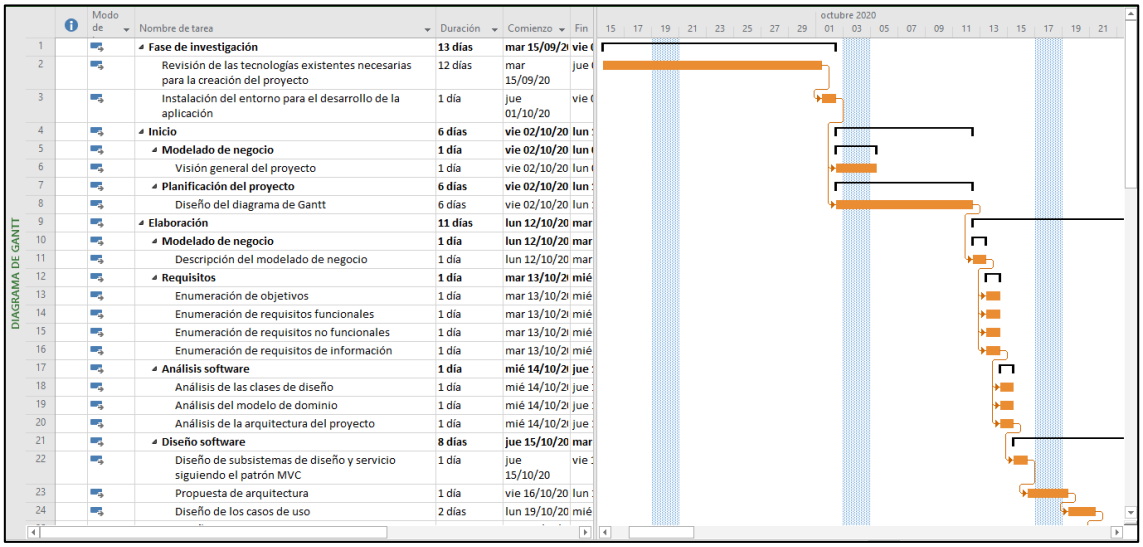
En nuestro proyecto, utilizaremos la base de datos NoSQL de Firebase llamada Cloud Firestore.

4. TÉCNICAS Y HERRAMIENTAS UTILIZADAS

En este apartado se presentarán las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Se incluyen las herramientas usadas para el desarrollo de la aplicación móvil, para la configuración del servidor y las utilizadas para la documentación del proyecto.

4.1 Microsoft Project

Microsoft Project es un software de administración de proyectos y programas de proyectos desarrollado y comercializado por Microsoft para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.



	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		**Fase de investigación**	**13 días**	**mar 15/09/2**	**vie**
2		Revisión de las tecnologías existentes necesarias para la creación del proyecto	12 días	mar 15/09/20	jue
3		Instalación del entorno para el desarrollo de la aplicación	1 día	jue 01/10/20	vie
4		**Inicio**	**6 días**	**vie 02/10/20**	**lun**
5		**Modelado de negocio**	**1 día**	**vie 02/10/20**	**lun**
6		Visión general del proyecto	1 día	vie 02/10/20	lun
7		**Planificación del proyecto**	**6 días**	**vie 02/10/20**	**lun**
8		Diseño del diagrama de Gantt	6 días	vie 02/10/20	lun
9		**Elaboración**	**11 días**	**lun 12/10/20**	**mar**
10		**Modelado de negocio**	**1 día**	**lun 12/10/20**	**mar**
11		Descripción del modelado de negocio	1 día	lun 12/10/20	mar
12		**Requisitos**	**1 día**	**mar 13/10/2**	**mié**
13		Enumeración de objetivos	1 día	mar 13/10/2	mié
14		Enumeración de requisitos funcionales	1 día	mar 13/10/2	mié
15		Enumeración de requisitos no funcionales	1 día	mar 13/10/2	mié
16		Enumeración de requisitos de información	1 día	mar 13/10/2	mié
17		**Análisis software**	**1 día**	**mié 14/10/2**	**jue**
18		Análisis de las clases de diseño	1 día	mié 14/10/2	jue
19		Análisis del modelo de dominio	1 día	mié 14/10/2	jue
20		Análisis de la arquitectura del proyecto	1 día	mié 14/10/2	jue
21		**Diseño software**	**8 días**	**jue 15/10/20**	**mar**
22		Diseño de subsistemas de diseño y servicio siguiendo el patrón MVC	1 día	jue 15/10/20	vie
23		Propuesta de arquitectura	1 día	vie 16/10/20	lun
24		Diseño de los casos de uso	2 días	lun 19/10/20	mié

Ilustración 4 - Microsoft Project

Gracias a esta herramienta ha sido posible realizar un diagrama de Gantt. Se trata de una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado, las dependencias y los recursos utilizados.

4.2 UML

Para el modelaje y documentación del sistema, se ha hecho uso del lenguaje unificado de modelado o UML, que es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Las herramientas utilizadas para el desarrollo de los diagramas y la gestión de requisitos han sido las siguientes.

Existen dos tipos de diagramas que podemos realizar con UML:

- **Diagramas estructurales:** muestran la estructura estática de los elementos del sistema. Se incluyen los diagramas de clases, de componentes, de despliegue, de objetos, de paquetes, de perfiles y de estructura compuesta.
- **Diagramas de comportamiento:** muestran la estructura dinámica del sistema. Se incluyen los diagramas de actividades, de casos de uso, de máquina de estado, y los diagramas de interacción.

4.2.1 Visual Paradigm

Visual Paradigm es una herramienta CASE que da soporte a los diagramas UML del sistema. Además de poder realizar el soporte de modelado, proporciona capacidades de generación de informes e ingeniería de código, incluida la generación de código.

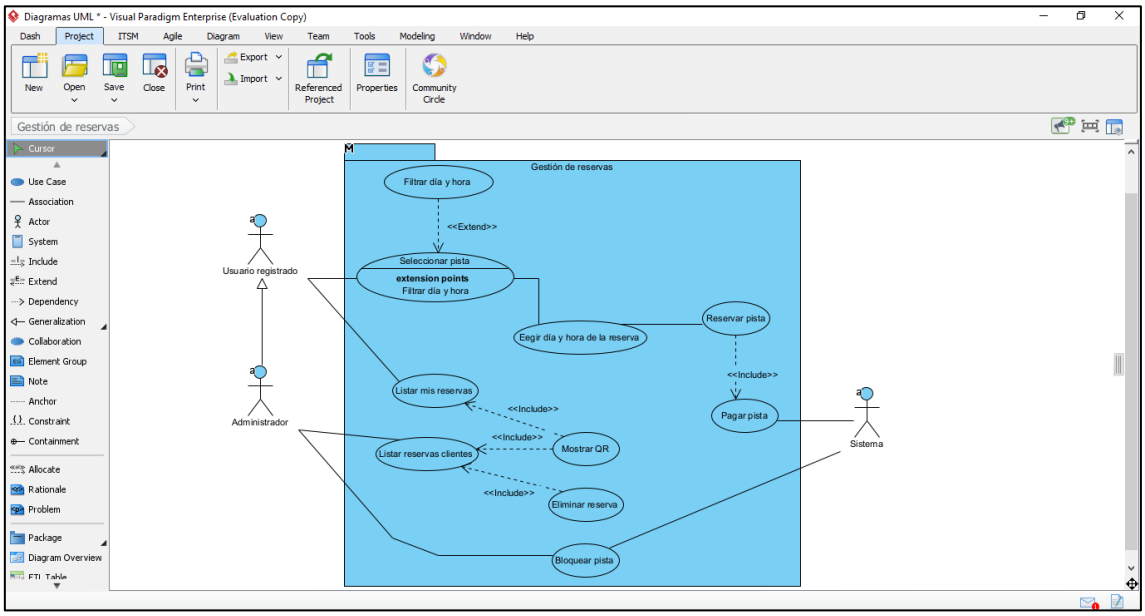


Ilustración 5 - Visual Paradigm

Se dispone de una licencia gratuita para alumnos de la Universidad de Salamanca, que se ha empleado para la realización del proyecto.

Con esta herramienta ha sido posible realizar todos los diagramas necesarios para documentar y modelar la aplicación desarrollada. Entre ellos los diagramas de casos de uso, de clases, de secuencia, de comunicación, de actividad, de despliegue, de componentes, el modelo de análisis, las máquinas de estado, etc.

4.2.2 REM

REM (*REquirements Management*) es una herramienta experimental gratuita de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software.

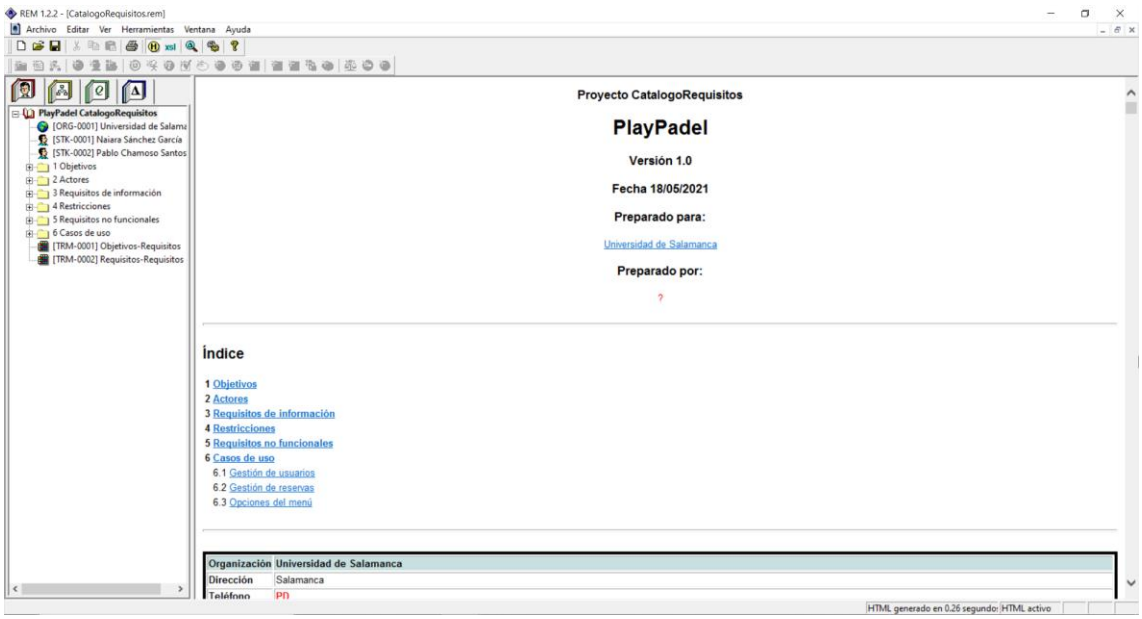


Ilustración 6 - REM

Con esta herramienta ha sido posible describir y especificar los actores, objetivos, requisitos de información, funcionales y no funcionales obtenidos de los diagramas UML realizados anteriormente con Visual Paradigm.

4.3 Desarrollo del servidor

Para la realización de la gestión de usuarios y el manejo de los datos del proyecto, no se ha integrado un servidor común, sino que se ha generado un proyecto adscrito a la plataforma Firebase.

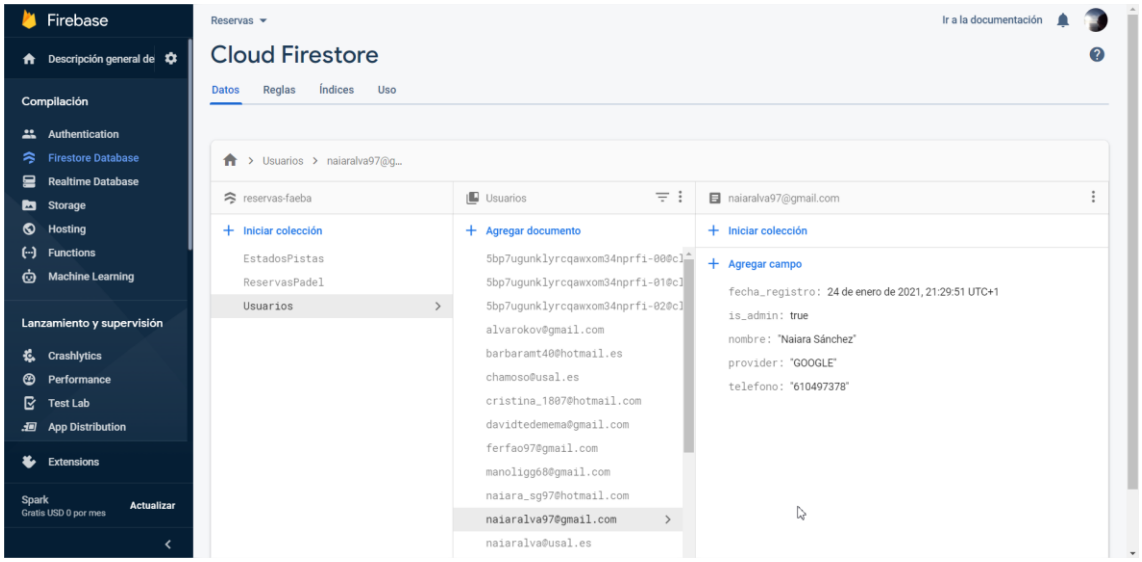


Ilustración 7 - Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles comprada por Google en 2014.

Esta plataforma se encuentra ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Algunas ventajas del uso de Firebase son las siguientes:

- **Usa un conjunto de herramientas multiplataforma**: se integra fácilmente tanto para plataformas web como para aplicaciones móviles. Es compatible con grandes plataformas: IOS, Android, aplicaciones web, Unity, C++, etc.
- **Usa la infraestructura de Google** y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.
- **Crea proyectos sin necesidad de un servidor:** Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.
- **Fácil integración con el sistema operativo Android:** como se ha visto en los ejemplos de código, con unos métodos simples podemos tener implementada la gestión de usuarios y la obtención e inserción de datos del sistema.
- Es un sistema eficaz, fiable y tiene el factor de la inmediatez como punto fundamental de la plataforma.

- Se dispone de una **gran documentación**, tanto en la página web de la plataforma como en otros foros de consulta, como StakOverflow y GitHub.

Este último punto es importante de analizar, ya que será fundamental para el diseño del sistema.

Firebase posee varios servicios o herramientas (muchas de uso gratuito hasta cierto límite) donde a lo largo del proyecto se usan varias de ellas.

4.3.1 Firebase Authentication

Es un servicio que puede autenticar los usuarios utilizando únicamente código del lado del cliente. Incluye la autenticación mediante proveedores de inicio de sesión como Facebook, Twitter y Google, entre otros; así como los métodos clásicos de inicio de sesión mediante correo electrónico y contraseña.

La herramienta se encargará de dar registro a los nuevos usuarios, y de controlar el acceso a los usuarios existentes.

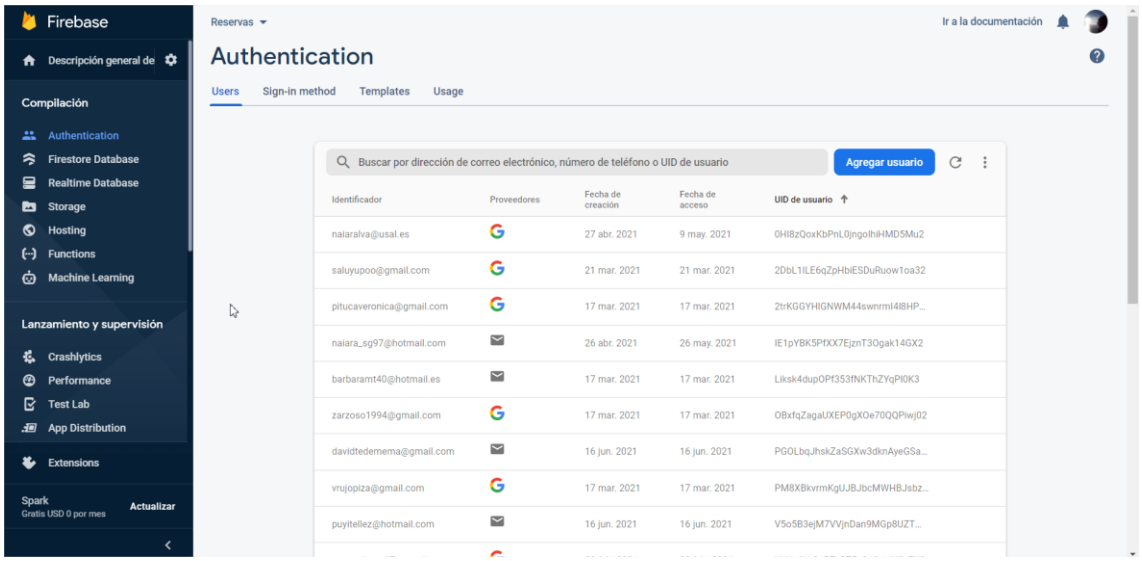


Ilustración 8 - Firebase Authentication

Para que la aplicación Android se comunique con la herramienta de Firebase Authentication se deben implementar unas funciones provenientes de la API de esta, en concreto, funciones para registrar usuarios y para controlar el acceso de usuarios ya registrados a las principales funcionalidades de la aplicación.

4.3.2 Firebase Cloud Firestore

Es un servicio de almacenamiento de datos derivado de Google Cloud Platform, adaptado a la plataforma de Firebase. Al igual que Realtime Database, es una base de datos NoSQL, aunque presenta diversas diferencias. Cloud Firestore tiene varias características:

- **Funciona en tiempo real:** usa la sincronización de datos para actualizar los datos de cualquier dispositivo conectado. Es una gran opción para desarrollar proyectos donde esta característica sea necesaria, aunque también para aplicaciones más tradicionales.
- Se organiza en forma de **documentos agrupados en colecciones**, y en ellos se pueden incluir tanto campos de diversos tipos (cadenas de texto, números, arrays, booleanos…) como otras subcolecciones.
- Puedes usar consultas para **recuperar documentos individuales específicos** o para recuperar todos los documentos de una colección que coinciden con los parámetros de la consulta.
- **Almacena en caché datos que usa tu app de forma activa**, por lo que la app puede escribir, leer, escuchar y consultar datos, aunque el dispositivo se encuentre sin conexión. Cuando el dispositivo vuelve a estar en línea, Cloud Firestore sincroniza todos los cambios locales de vuelta.

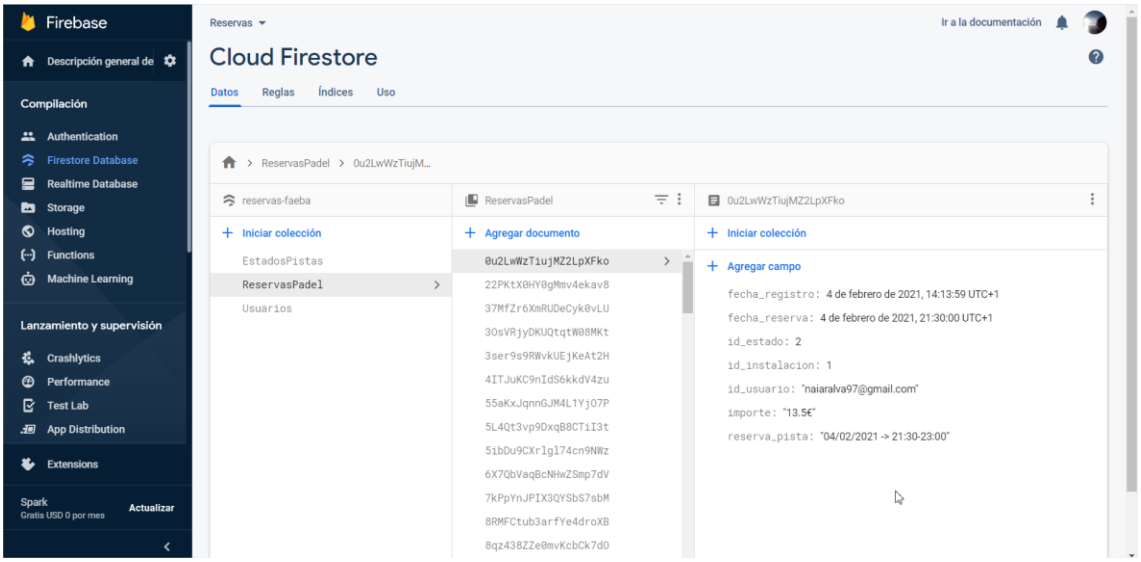


Ilustración 9 - Firebase Cloud Firestore

Entre sus limitaciones más destacadas encontramos la de no soportar las búsquedas de texto tipo "LIKE" (buscar por subcadenas del texto almacenado) y la de no poder filtrar

las búsquedas con condiciones que impliquen más de un campo, si no es por búsquedas por el texto exacto.

4.3.3 Crashlytics

Firebase Crashlytics es una herramienta liviana para provisión de informes de fallas en tiempo real que permite hacer un seguimiento de los problemas de estabilidad que afectan la calidad de tu app, priorizarlos y corregirlos. Crashlytics agrupa las fallas de forma inteligente y destaca las circunstancias en las que se produjeron, lo que te permite ahorrar tiempo en la solución de problemas.

Descubre si una falla específica afecta a muchos usuarios. Recibe alertas cuando la gravedad de un problema aumenta repentinamente. Determina qué líneas de código están provocando fallas.

Ilustración 10 - Estadística de fallas

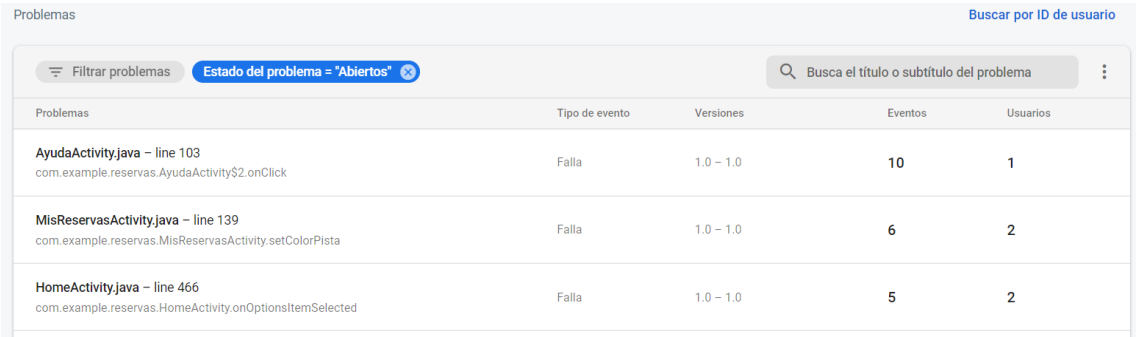


Ilustración 11 - Problemas abiertos

En las ilustraciones anteriores podemos observar el número de usuarios afectados y las fallas que ha habido en los últimos 90 días. Podemos comprobar que en el momento del desarrollo de la aplicación ha dado varios problemas en diversas pantallas. La semana de pruebas, que fue sobre el 20 de marzo, vemos que no dio ningún fallo (los que andan por esa fecha son de días anteriores) y que todo funcionó correctamente. Todos los problemas actualmente tienen el caso en modo “Cerrado” ya que han sido todos solucionados.

4.3.4 Test Lab

Es una infraestructura de prueba de aplicaciones basada en la nube que te permite probar tus aplicaciones en una gran variedad de dispositivos y configuraciones a fin de que tengas una idea más clara de cómo será la experiencia para los usuarios activos.

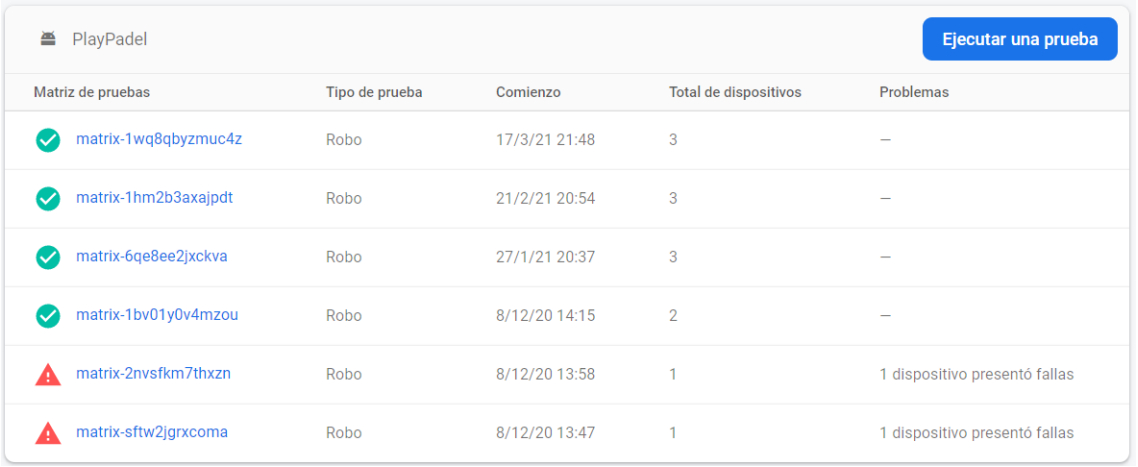


Ilustración 12 - Test Lab

En el caso de esta aplicación se realizaron un total de seis pruebas las cuales las dos primeras dieron algún fallo y el resto se puede comprobar que fue todo correctamente con un total de tres dispositivos diferentes entre ellos con orientación horizontal uno de ellos y una duración total de unos 30 minutos cada prueba.

4.3.5 Google Analytics

Google Analytics es una solución gratuita de medición de apps que proporciona estadísticas sobre el uso de las apps y la participación de los usuarios.

Google Analytics es una solución de análisis ilimitada que ocupa un lugar central en Firebase. Analytics se integra a distintas funciones de Firebase y te proporciona una capacidad ilimitada de generar informes sobre un total de hasta 500 eventos distintos que puedes definir con el SDK de Firebase. Los informes de Analytics te permiten entender claramente cómo se comportan tus usuarios para que puedas tomar decisiones fundamentadas en relación con el marketing de las apps y las optimizaciones del rendimiento.

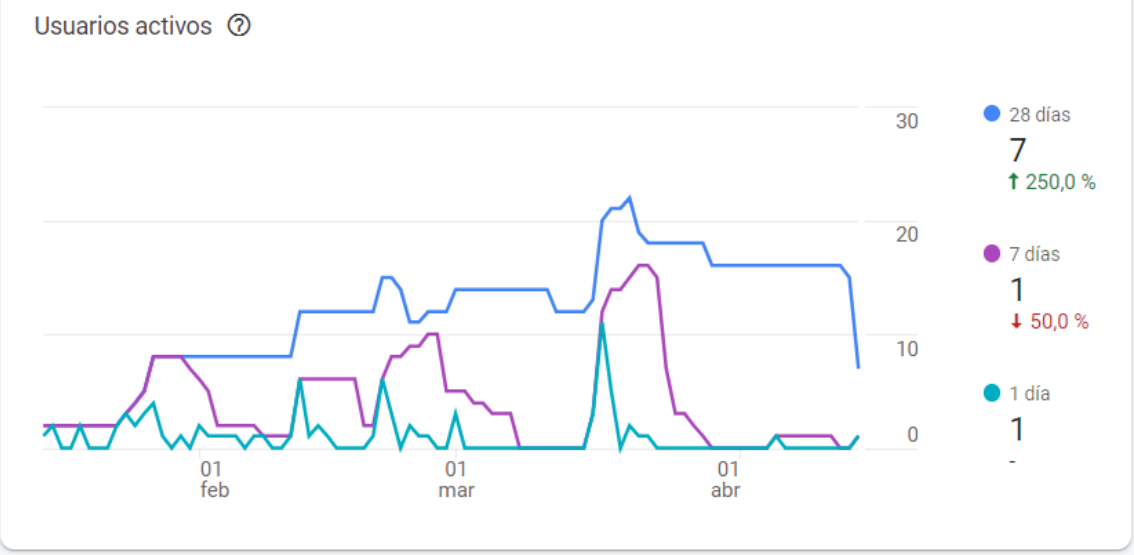


Ilustración 13 - Usuarios activos

En la Ilustración 13 podemos comprobar los usuarios activos que ha habido en los últimos 90 días por mes, semana y día. Se puede ver cómo, sobre el 20 de marzo, se produce un pico más alto ya que sobre esa fecha se realizó el periodo de pruebas de la aplicación y se registraron más usuarios de lo normal llegando a 22 como se puede observar.

Ilustración 14 - Interacción diaria de los usuarios

En la Ilustración 14 se muestra la participación diaria promedio por usuario para el período seleccionado, incluida la fluctuación porcentual respecto del período anterior. En nuestro caso, se observa que la interacción media diaria es de doce minutos aproximadamente.

El apartado *Pantallas/páginas principales* son las tres pantallas principales en función de la participación, incluido el porcentaje de participación general y el tiempo promedio de cada pantalla.

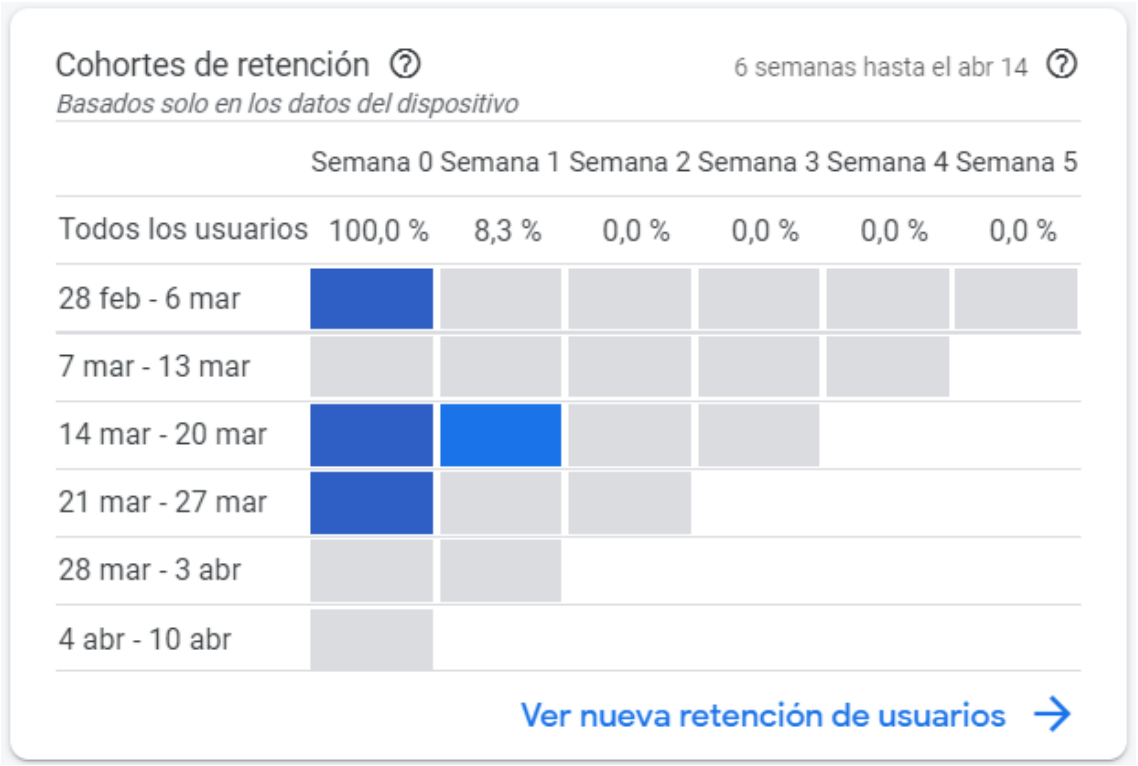


Ilustración 15 - Cohortes de retención

En la Ilustración 15 se puede observar el informe de la cohorte de retención. Una cohorte es un conjunto de usuarios que comenzaron a utilizar su aplicación al mismo tiempo (por ejemplo, el mismo día o durante la misma semana). Este informe muestra el nivel de retención de los usuarios en la aplicación.

Ilustración 16 - Audiencia por ubicación

En la Ilustración 16 se muestra el número y porcentaje de sesiones en cada uno de los países, en este caso, de España y Estados Unidos. Los usuarios de Estados Unidos se deben a las pruebas realizadas a través del AVD Manager (creador de dispositivos virtuales) de Android Studio.

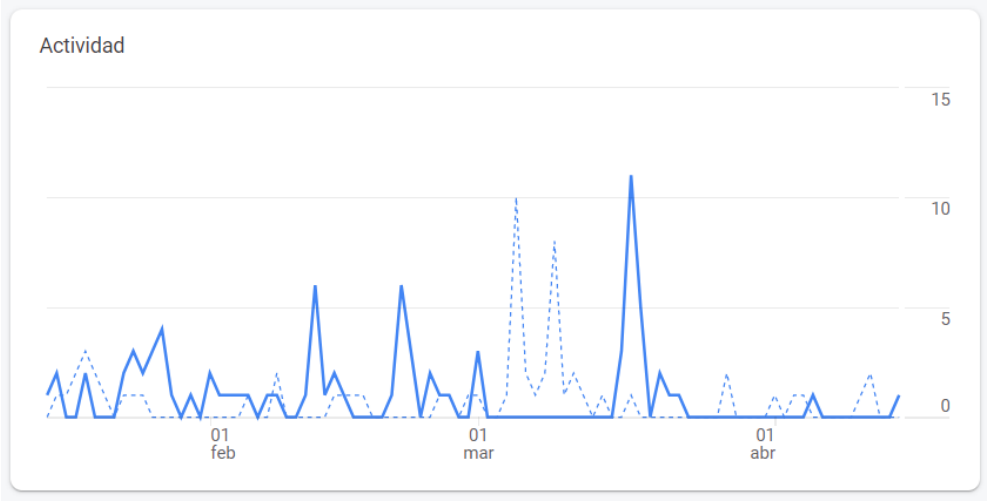


Ilustración 17 - Actividad de los usuarios

En la Ilustración 17 se muestra el número de usuarios que se han conectado a la aplicación cada día viendo que el pico más alto se produce en el periodo de pruebas de esta aplicación llegando a un total de once usuarios en el mismo día.

4.4 Desarrollo de la aplicación

En este apartado se detallarán los ámbitos más importantes para el desarrollo de la aplicación Android, así como los entornos, lenguajes utilizados y GIT.

4.4.1 Android Studio

El entorno principal para el desarrollo de la aplicación es Android Studio [8]. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.

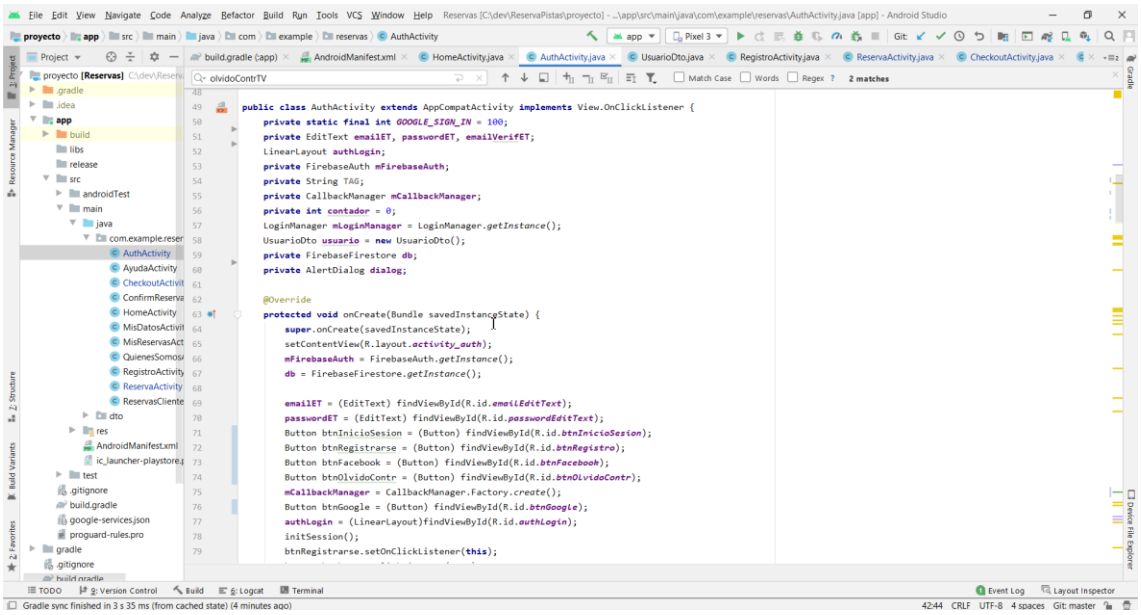


Ilustración 18 - Android Studio

Estas son las características más relevantes de este entorno de desarrollo:

- Soporte para construcción basada en Gradle.
- Refactorización específica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Integración de ProGuard y funciones de firma de aplicaciones.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario.
- Soporte para programar aplicaciones para Android Wear.
- Soporte integrado para Google Cloud Platform, que permite la integración con Firebase Cloud Messaging y Google App Engine.
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.
- Renderizado en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.

Para el desarrollo del código fuente de la aplicación se ha hecho uso de dos lenguajes de programación: Java para las clases del modelo de diseño, control de las actividades y para realizar las peticiones a la plataforma Firebase; y XML para el diseño de las actividades del sistema (interfaz).

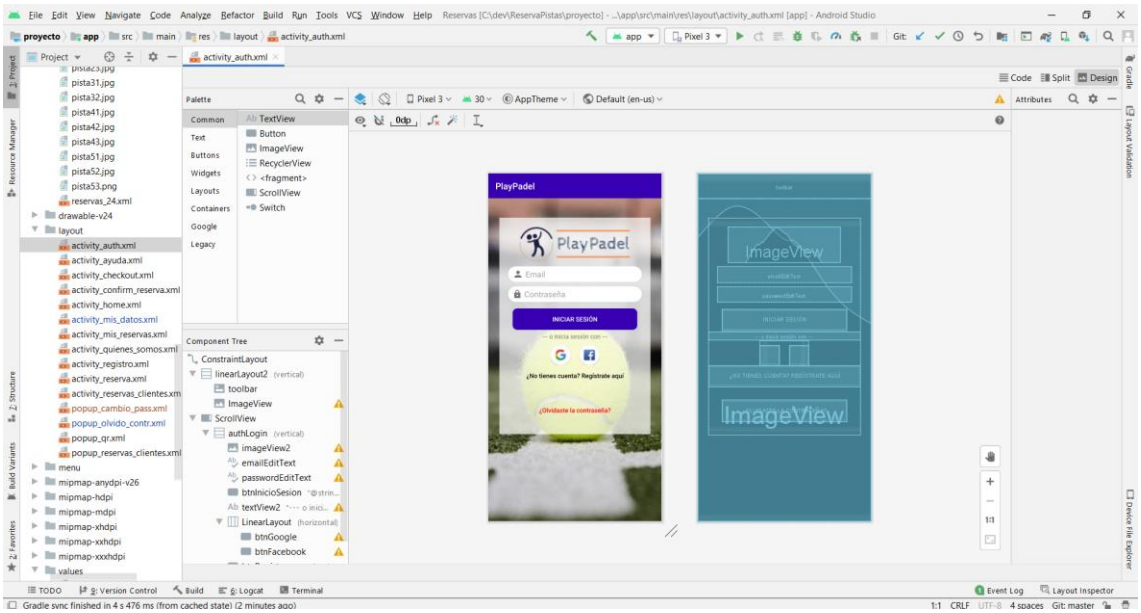


Ilustración 19 - Aplicación en Android Studio

A continuación, detallaremos un poco más los lenguajes utilizados:

- **Java**: es un lenguaje de programación de propósito general, generalmente orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.

  Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una única vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

- **XML (*eXtensible Markup Language*):** podemos decir que es un Lenguaje de Marcas Extensibles, aunque realmente es un metalenguaje. Se componen de etiquetas que nos aportan datos e información que queremos procesar. Estas etiquetas pueden estar de forma individual o anidadas. Habitualmente un fichero XML incluye mucha información y debe de ser procesada correctamente por el usuario, en este caso el desarrollador.

  Dentro de estos archivos XML tendremos una o varias etiquetas, que a su vez tendrán otras etiquetas, que podrán estar o no anidadas.

4.4.2 Stripe

Cabe destacar el sistema de pagos utilizado para la aplicación llamado Stripe [9]. Stripe es una plataforma de pagos del estilo de PayPal mediante la cual puedes conectar tu aplicación con una cuenta bancaria o una cuenta de Stripe y empezar a cobrar por tus productos o servicios.

Posee toda la documentación necesaria para su implementación en cualquier aplicación o página web y, además, puedes utilizarlo en tu entorno de pruebas especializado para desarrolladores con cuentas bancarias virtuales como en el caso de PlayPadel.

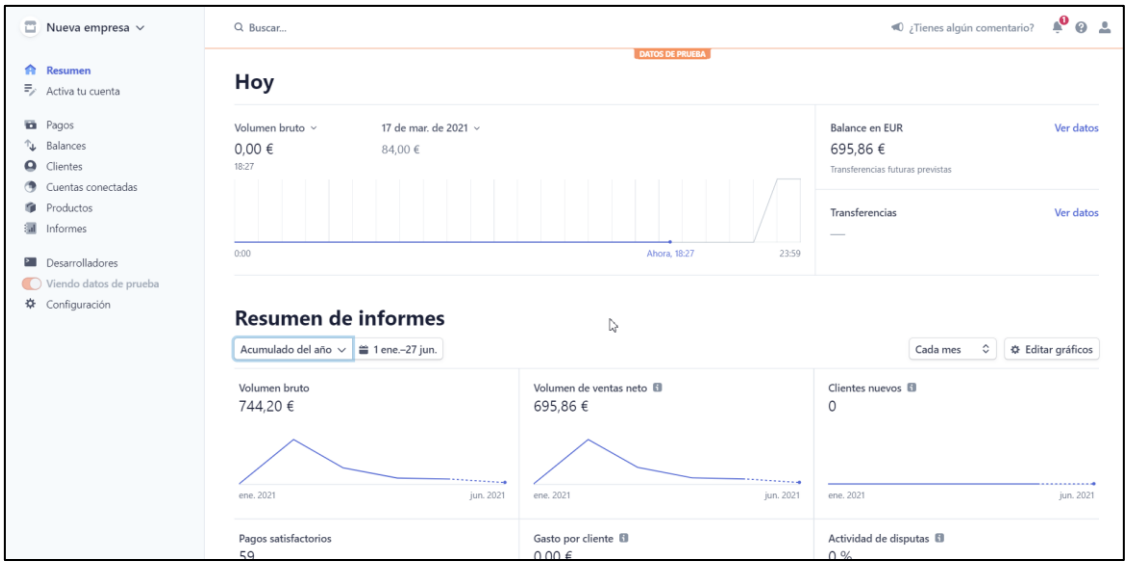


Ilustración 20 – Stripe

4.4.3 Arduino

Es una plataforma de creación de electrónica de código abierto [10], la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

El proyecto nació en 2003 y se trata de una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, y que puede ser programada tanto en Windows como macOS y GNU/Linux. Un proyecto que promueve la filosofía *'learning by doing'*, que viene a querer decir que la mejor manera de aprender es cacharreando.

El Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar en el entorno Arduino IDE. Estas instrucciones permiten crear programas que interactúan con los circuitos de la placa.

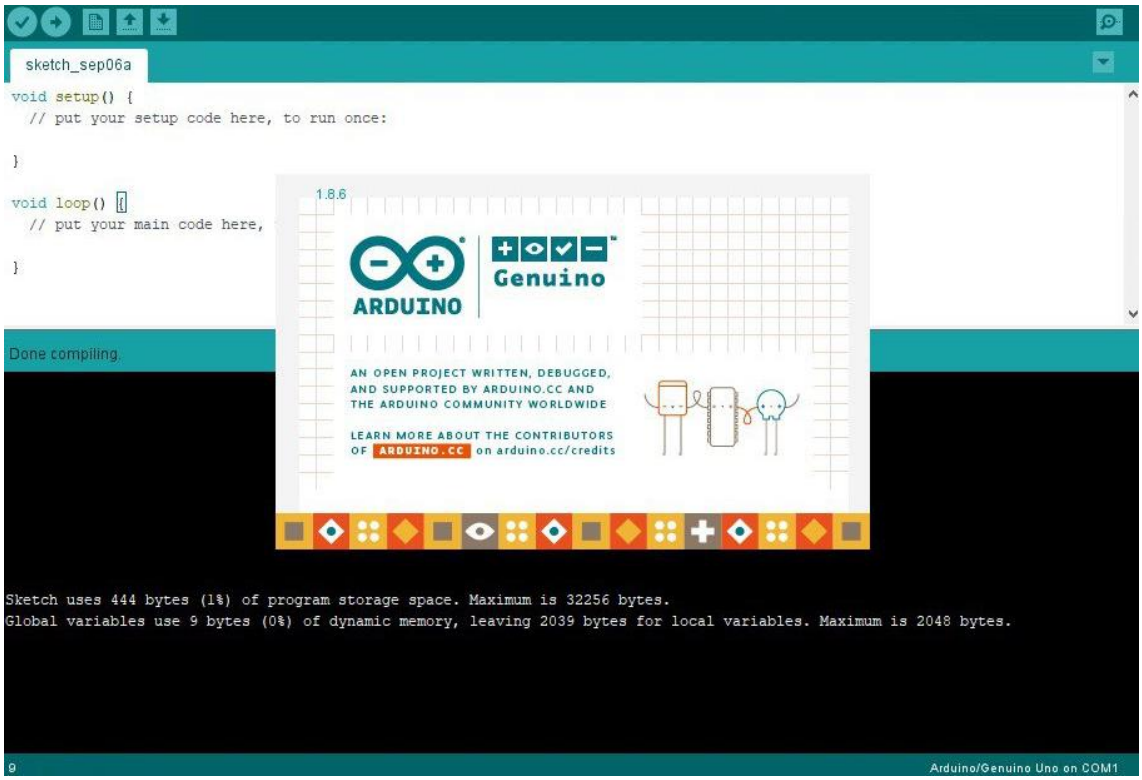


Ilustración 21 - Arduino IDE

Los tipos más importantes de placas son:

- **Arduino UNO:** La placa tiene 14 pines digitales, 6 pines analógicos programables con el Arduino IDE a través de un cable USB. Puede ser alimentado por el cable USB o por una batería externa de 9 voltios, aunque acepta voltajes entre 7 y 20 voltios. Es el buque insignia de Arduino ya que es la placa más popular, la que todo el mundo utiliza para iniciarse y la más sencilla de utilizar. Es el punto de partida de muchos entusiastas de la programación de electrónica y es la utilizada para esta aplicación.

Ilustración 22 - Arduino UNO [11]

- **Arduino MEGA:** La placa con el microcontrolador más potente de la familia Arduino. Con 54 pines digitales que funcionan como entrada y salida; 16 entradas analógicas, un cristal oscilador de 16 MHz, una conexión USB, un botón de reinicio y una entrada para la alimentación de la placa. Arduino MEGA es la placa que se utiliza cuando Arduino UNO no llega a cubrir las necesidades de un proyecto. Se ha utilizado ampliamente, por ejemplo, como centro de control y computación en impresoras 3D.

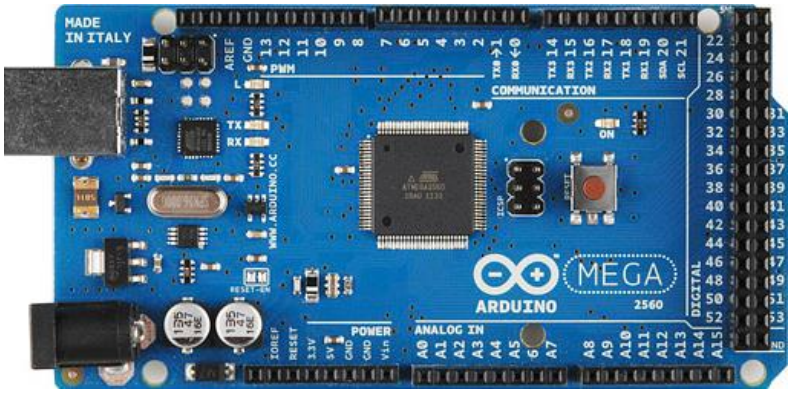


Ilustración 23 - Arduino MEGA[11]

4.4.4 Git (SourceTree)

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Source Tree es un potente GUI (*Graphical User Interface* – Interfaz Gráfica de Usuario) para gestionar todos tus repositorios ya sean Git o Mercurial. Con Source Tree podemos crear, clonar, hacer *commit*, *push*, *pull*, *merge* y algunas cosas más de una forma bastante fácil.

En el caso de la aplicación 'PlayPadel' no es muy importante el uso de un Git ya que no posee archivos de código fuente en exceso y únicamente está desarrollado por una única persona por lo que el hecho de hacer *merge*, provocar conflictos o el uso de varias ramas tiene menos sentido ya que todo se ha realizado de forma continuada en la misma rama. Aún así he visto una forma muy buena para ir comprobando los cambios que he ido realizando poco a poco en la aplicación y así llevar un mayor control del desarrollo de esta.

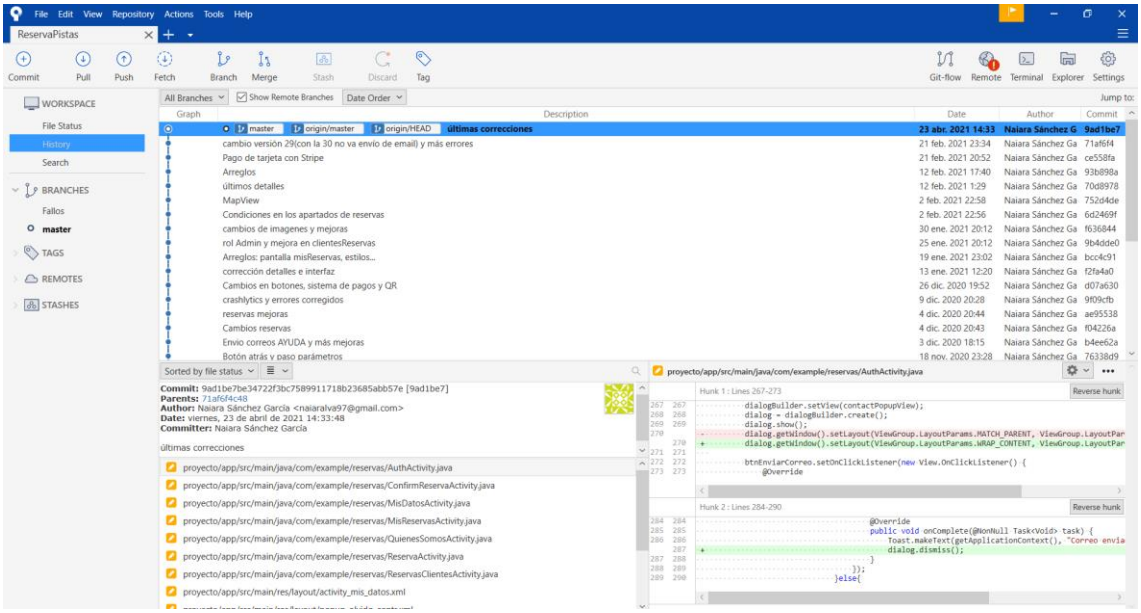


Ilustración 24 - SourceTree

5. ASPECTOS RELEVANTES DEL PROYECTO

En este apartado se mostrarán las partes más relevantes de la planificación, desarrollo de la aplicación y de su documentación, haciendo uso del lenguaje UML de manera resumida.

5.1 Proceso unificado

Para la realización de este proyecto, se ha seguido la metodología del Proceso Unificado [12]. Este proceso es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas y, junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

5.1.1 Características del Proceso Unificado

Las principales características de esta metodología son:

- **Es iterativo e incremental:** es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. En nuestro caso, hemos añadido dos fases más que explicaremos más adelante.

  Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba.
- **Dirigido por los casos de uso:** en esta metodología los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.
- **Centrado en la arquitectura**: el Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La arquitectura se suele representar mediante modelos del sistema, para orientar al desarrollador ver la imagen del sistema completo antes de su implementación.
- **Enfocado en los riesgos:** requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los

resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

5.1.2 Fases del Proceso Unificado

El Proceso Unificado de desarrollo puede ser dividido en cuatro fases para su mejor desarrollo, aunque para este proyecto se ha decidido añadir dos fases más, una al inicio y otra al final. Estas fases, ayudando tanto a la elaboración como a la resolución de problemas, son:

- **Investigación**: Esta fase se ha añadido ya que se ha requerido de cierto tiempo para poder investigar sobre las herramientas que se han utilizado ya que la alumna no requería de algunos conocimientos implementados en el proyecto.
- **Inicio:** En esta fase se define la facilidad de realizar el proyecto, se presenta un modelo, visión, metas, deseos del usuario, plazos, costos y viabilidad. Para este proyecto no se ha necesitado mucho tiempo ya que solamente era tener una idea general de cómo se iba a realizar la aplicación.
- **Elaboración:** En esta fase se obtiene la visión refinada del proyecto a realizar, mostrar aquellos requisitos más importantes y que no pueden faltar en la aplicación como, por ejemplo, la gestión de usuarios, de reservas o el sistema de pagos ya que son imprescindibles.
- **Implementación:** Esta abarca la evolución hasta convertirse en producto listo incluyendo requisitos mínimos. Esta es la parte más larga ya que engloba a toda la construcción de la aplicación. Más adelante vemos la duración de la fase con detalle.
- **Transición:** En esta fase final, el programa debe estar listo para ser probado, instalado y utilizado por el cliente sin ningún problema. Una vez finalizada esta fase, se debe comenzar a pensar en futuras novedades para la misma.
- **Elaboración de la documentación:** esta fase se ha creado únicamente para planificar la duración de la documentación del proyecto. La memoria final se irá realizando en función de lo que se vaya realizando en la fase de implementación.

5.2 Planificación temporal

Para realizar la planificación temporal se ha definido diagrama de Gantt, que permite representar todas las tareas realizadas a lo largo del proyecto, su agrupación en iteraciones y fases, su duración, dependencias y recursos necesarios para su realización.

Como se ha mencionado anteriormente, el Proceso Unificado tiene cuatro fases típicas, que son el inicio, la elaboración, la implementación y la transición. En este caso, se añadió una fase anterior a la fase de inicio, denominada fase de investigación y otra fase posterior indicando la elaboración de la documentación.

Cabe destacar que los plazos propuestos en la planificación realizada inicialmente han sido muy acertados excepto en la fase de implementación ya que se han retrasado varios días aunque se haya conseguido llegar a tiempo sin problema.

Es conveniente remarcar que todas las tareas han sido realizadas por un solo desarrollador, por lo tanto, hay poca simultaneidad en la realización de tareas, sobre todo en la fase de construcción. En la mayoría de los casos ha sido necesario finalizar una tarea para comenzar la siguiente.

Todo esto se encuentra más detallado en el "Anexo I – Planificación temporal".

5.2.1 Fase de investigación

Está fase se decidió añadir porque fue necesario realizar un par de tareas antes de la fase de inicio. En la primera tarea se realizó un estudio de qué tecnologías iban a ser necesarias y en la segunda se instaló el entorno virtual donde se iba a desarrollar la aplicación.

La duración que se indicó inicialmente de esta fase fue de 13 días y se llevó a cabo correctamente.

1			**Fase de investigación**	**13 días**	**mar 15/09/2**	**vie 02/10/20**	
2			Revisión de las tecnologías existentes necesarias para la creación del proyecto	12 días	mar 15/09/20	jue 01/10/20	
3			Instalación del entorno para el desarrollo de la aplicación	1 día	jue 01/10/20	vie 02/10/20	2

Ilustración 25 - Fase de investigación

5.2.2 Fase de inicio

Las tareas realizadas en esta fase básicamente consistieron en realizar una visión general del proyecto, enumerar los actores, requisitos y objetivos necesarios (estas tareas de enumeración

se realizaron todas simultáneamente debido a que son muy similares y presentan poca carga de trabajo), planificar el calendario y diseñar un boceto.

La duración que se indicó inicialmente de esta fase fue de 6 días y se llevó a cabo correctamente.

4			**Inicio**	**6 días**	**vie 02/10/20**	**lun 12/10/20**	
5			**Modelado de negocio**	**1 día**	**vie 02/10/20**	**lun 05/10/20**	
6			Visión general del proyecto	1 día	vie 02/10/20	lun 05/10/20	3
7			**Planificación del proyecto**	**6 días**	**vie 02/10/20**	**lun 12/10/20**	
8			Diseño del diagrama de Gantt	6 días	vie 02/10/20	lun 12/10/20	3

Ilustración 26 - Fase de inicio

5.2.3 Fase de elaboración

Esta fase se enfoca en el análisis y diseño que se ha realizado a la aplicación, listar aquellos requisitos importantes, la estructura del proyecto y las clases que se han utilizado.

La duración que se indicó inicialmente de esta fase fue de 11 días y se llevó a cabo correctamente.

9			**Elaboración**	**11 días**	**lun 12/10/20**	**mar 27/10/20**		
10			**Modelado de negocio**	**1 día**	**lun 12/10/20**	**mar 13/10/20**		
11			Descripción del modelado de negocio	1 día	lun 12/10/20	mar 13/10/20	8	Naiara Sánchez (Desa
12			**Requisitos**	**2 días**	**mar 13/10/2**	**jue 15/10/20**		
13			Enumeración de objetivos	1 día	mar 13/10/2	mié 14/10/20	11	Naiara Sánchez (Desa
14			Enumeración de requisitos de restricción	1 día	mar 13/10/2	mié 14/10/20	11	Naiara Sánchez (Desa
15			Enumeración de requisitos no funcionales	1 día	mar 13/10/2	mié 14/10/20	11	Naiara Sánchez (Desa
16			Enumeración de requisitos de información	1 día	mar 13/10/2	mié 14/10/20	11	Naiara Sánchez (Desa
17			Enumeración de requisitos funcionales	1 día	mié 14/10/2	jue 15/10/20	16	Naiara Sánchez (Desa
18			**Análisis software**	**1 día**	**jue 15/10/20**	**vie 16/10/20**		
19			Análisis de las clases de diseño	1 día	jue 15/10/20	vie 16/10/20	17	Naiara Sánchez (Desa
20			Análisis del modelo de dominio	1 día	jue 15/10/20	vie 16/10/20	17	Naiara Sánchez (Desa
21			Análisis de la arquitectura del proyecto	1 día	jue 15/10/20	vie 16/10/20	17	Naiara Sánchez (Desa
22			**Diseño software**	**7 días**	**vie 16/10/20**	**mar 27/10/20**		
23			Diseño de subsistemas de diseño y servicio siguiendo el patrón MVC	1 día	vie 16/10/20	lun 19/10/20	21	Naiara Sánchez (Desarrolladora)
24			Propuesta de arquitectura	1 día	lun 19/10/20	mar 20/10/20	23	Naiara Sánchez (Desa
25			Diseño de los casos de uso	2 días	mar 20/10/2	jue 22/10/20	24	Naiara Sánchez (Desa
26			Diseño del modelo de dominio	1 día	jue 22/10/20	vie 23/10/20	25	Naiara Sánchez (Desa
27			Diseño de diagramas de secuencia	2 días	vie 23/10/20	mar 27/10/20	26	Naiara Sánchez (Desa

Ilustración 27 - Fase de elaboración

5.2.4 Fase de implementación

Esta fase se divide en varias iteraciones en las cuales se realiza toda la construcción de la aplicación. En la última iteración se lleva a cabo la **fase de transición** la cual se realizan una serie de pruebas finales con varios usuarios y su correspondiente corrección de errores.

La duración que se indicó inicialmente de esta fase fue de 98 días y se llegó a retrasar unos 30 días aproximadamente de lo esperado ya que las pruebas finales con varios usuarios y dispositivos diferentes se empezaron a realizar aproximadamente el 17 de abril.

28			**Implementación**	**98 días**	**mar 27/10/2**	**vie 12/03/21**		
29			**Iteración 1**	**34 días**	**mar 27/10/2**	**lun 14/12/20**		
30			**Requisitos**	**1 día**	**mar 27/10/2**	**mié 28/10/20**		
31			Estudio de la implementación de los requisitos relacionados con la gestión de usuarios	1 día	mar 27/10/20	mié 28/10/20	27	Naiara Sánchez (Desarrolladora)
32			**Análisis software**	**1 día**	**mié 28/10/2**	**jue 29/10/20**		
33			Análisis del costo de implementación de los requisitos para la optimización	1 día	mié 28/10/20	jue 29/10/20	31	Naiara Sánchez (Desarrolladora)
34			**Diseño software**	**1 día**	**jue 29/10/20**	**vie 30/10/20**		
35			Diseño de la interfaz principal	1 día	jue 29/10/20	vie 30/10/20	33	Naiara Sánchez (Desarrolladora)
36			**Construcción**	**28 días**	**vie 30/10/20**	**mié 09/12/20**		
37			Implementación de la interfaz principal	10 días	vie 30/10/20	vie 13/11/20	35	Naiara Sánchez (Desa
38			Implementación del login	8 días	vie 13/11/20	mié 25/11/20	37	Naiara Sánchez (Desa
39			Implementación general de la gestión de usuarios	10 días	mié 25/11/20	mié 09/12/20	38	Naiara Sánchez (Desarrolladora)
40			**Pruebas**	**3 días**	**mié 09/12/2**	**lun 14/12/20**		
41			Pruebas generales de la aplicación	3 días	mié 09/12/2	lun 14/12/20	39	Naiara Sánchez (Desa

Ilustración 28 - Fase de implementación iteración 1

42			**Iteración 2**	**45 días**	**lun 14/12/20**	**lun 15/02/21**		
43			**Requisitos**	**1 día**	**lun 14/12/20**	**mar 15/12/20**		
44			Estudio de la implementación de los requisitos relacionados con la gestón de reservas	1 día	lun 14/12/20	mar 15/12/20	41	Naiara Sánchez (Desarrolladora)
45			**Análisis software**	**1 día**	**mar 15/12/20**	**mié 16/12/20**		
46			Análisis del costo de implementación de los requisitos para la optimización	1 día	mar 15/12/20	mié 16/12/20	44	Naiara Sánchez (Desarrolladora)
47			**Diseño software**	**1 día**	**mié 16/12/2**	**jue 17/12/20**		
48			Diseño de la interfaz principal	1 día	mié 16/12/2	jue 17/12/20	46	Naiara Sánchez (Desa
49			**Construcción**	**40 días**	**jue 17/12/20**	**jue 11/02/21**		
50			Implementación de la interfaz principal	10 días	jue 17/12/20	jue 31/12/20	48	Naiara Sánchez (Desarrolladora)
51			Implementación general de la gestión de reservas	30 días	jue 31/12/20	jue 11/02/21	50	Naiara Sánchez (Desarrolladora)
52			**Pruebas**	**2 días**	**jue 11/02/21**	**lun 15/02/21**		
53			Pruebas generales de la aplicación	2 días	jue 11/02/21	lun 15/02/21	51	Naiara Sánchez (Desarrolladora)

Ilustración 29 - Fase de implementación iteración 2

54			**Iteración 3**	**19 días**	**lun 15/02/21**	**vie 12/03/21**		
55			**Requisitos**	**1 día**	**lun 15/02/21**	**mar 16/02/21**		
56			Estudio de la implementación de los requisitos relacionados con lector QR	1 día	lun 15/02/21	mar 16/02/21	53	Naiara Sánchez (Desarrolladora)
57			**Análisis software**	**1 día**	**mar 16/02/21**	**mié 17/02/21**		
58			Análisis del costo de implementación de los requisitos para la optimización	1 día	mar 16/02/21	mié 17/02/21	56	Naiara Sánchez (Desarrolladora)
59			**Diseño software**	**1 día**	**mar 16/02/21**	**mié 17/02/21**		
60			Diseño de la simulación de arduino	1 día	mar 16/02/21	mié 17/02/21	56	Naiara Sánchez (Desarrolladora)
61			**Construcción**	**15 días**	**mié 17/02/2**	**mié 10/03/21**		
62			Implementación del funcionamiento del lector QR	15 días	mié 17/02/21	mié 10/03/21	60	Naiara Sánchez (Desarrolladora)
63			**Pruebas**	**2 días**	**mié 10/03/2**	**vie 12/03/21**		
64			Pruebas del correcto funcionamiento del lector QR	2 días	mié 10/03/21	vie 12/03/21	62	Naiara Sánchez (Desarrolladora)

Ilustración 30 - Fase de implementación iteración 3

5.2.5 Fase de transición

Esta fase tuvo que ver con la comprobación de los resultados de las pruebas de la aplicación realizadas al final de la fase anterior, la planificación de los cambios para subsanar los fallos encontrados tras las pruebas, el diseño de dichos cambios y su implementación, y finalmente, la realización de las pruebas finales.

La duración que se indicó inicialmente de esta fase fue de 21 días y se llegó a retrasar unos 15 días aproximadamente de lo esperado ya que las pruebas finales con varios usuarios y dispositivos diferentes fueron muy positivas y apenas hubo fallos en la aplicación y, por tanto, no se tardó mucho en subsanar los errores pudiendo recuperar algún día de retraso con respecto a la fase de implementación.

65			**Fase de transición**	**21 días**	**vie 12/03/21**	**lun 12/04/21**		
66			**Análisis software**	**1 día**	**vie 12/03/21**	**lun 15/03/21**		
67			Análisis del resultado de las pruebas	1 día	vie 12/03/21	lun 15/03/21	64;53;41	Naiara Sánchez (Desarrolladora)
68			Planificación de los errores a corregir	1 día	vie 12/03/21	lun 15/03/21	64;53;41	Naiara Sánchez (Desarrolladora)
69			**Diseño software**	**1 día**	**vie 12/03/21**	**lun 15/03/21**		
70			Diseño de los cambios a corregir	1 día	vie 12/03/21	lun 15/03/21	53;64;41	Naiara Sánchez (Desarrolladora)
71			**Implementación**	**15 días**	**lun 15/03/21**	**lun 05/04/21**		
72			Implementación de los errores	15 días	lun 15/03/21	lun 05/04/21	70	Naiara Sánchez (Desarrolladora)
73			**Pruebas**	**5 días**	**lun 05/04/21**	**lun 12/04/21**		
74			Pruebas finales	5 días	lun 05/04/21	lun 12/04/21	72	Naiara Sánchez (Desarrolladora)

Ilustración 31 - Fase de transición

5.2.6 Elaboración de la documentación

En esta última fase se realiza toda la documentación del proyecto junto con los anexos donde se explica todo lo que se ha realizado en el proyecto.

La duración de esta fase es de 165 días ya que la memoria del proyecto se lleva haciendo poco a poco desde el principio de este. La fecha final del trabajo de fin de grado se indicó para el día 15 de junio, pero no se terminó hasta el 30 de junio ya que se produjeron algunos retrasos sobre todo en la fase de implementación.

75			**Elaboración de documentación**	**165 días**	**mar 27/10/20**	**mar 15/06/21**		
76			Memoria del proyecto	165 días	mar 27/10/20	mar 15/06/21	27	Naiara Sánchez (Desarrolladora)
77			Anexo I. Planificación temporal	3 días	lun 03/05/21	jue 06/05/21	82	Naiara Sánchez (Desarrolladora)
78			Anexo II. Catálogo de requisitos	12 días	jue 06/05/21	lun 24/05/21	77	Naiara Sánchez (Desarrolladora)
79			Anexo III. Modelo de análisis	8 días	lun 24/05/21	jue 03/06/21	78	Naiara Sánchez (Desarrolladora)
80			Anexo IV. Modelo de diseño	8 días	jue 03/06/21	mar 15/06/21	79	Naiara Sánchez (Desarrolladora)
81			Anexo V. Manual de usuario	5 días	jue 22/04/21	jue 29/04/21	74	Naiara Sánchez (Desarrolladora)
82			Anexo VI. Manual del programador	2 días	jue 29/04/21	lun 03/05/21	81	Naiara Sánchez (Desarrolladora)

Ilustración 32 - Elaboración de la documentación

5.3 Catálogo de requisitos

En este apartado se especificarán los actores, objetivos y requisitos que forman parte de la documentación UML de la aplicación.

Si se precisa obtener una explicación más amplia respecto a este apartado se adjuntará junto a este documento el Anexo II – Catálogo de requisitos.

5.3.1 Objetivos

- **Correcta gestión de usuarios**: el sistema deberá mostrar al usuario una pantalla de registro e inicio de sesión para que estos puedan acceder a su cuenta, cambiar su contraseña, sus datos personales, eliminar su usuario….
- **Integración con Firebase**: el sistema deberá integrarse con el servidor Firebase el cual se encargará de almacenar los datos de las reservas y usuarios, entre otras cosas.
- **Correcta gestión de las reservas**: el sistema deberá realizar una correcta gestión de las reservas de las pistas sin poder reservar una ya reservada anteriormente.

- **Consulta de las reservas**: el sistema deberá poder permitir la consulta de la reserva ya que al ofrecer el código QR es necesario tenerlo siempre a mano para acceder a la pista.
- **Correcta gestión de los pagos**: el sistema deberá ofrecer una gestión de pagos para poder pagar la reserva correctamente.
- **Correcta gestión de los correos**: el sistema deberá tener un correo donde recibir cualquier duda que tenga un usuario.

5.3.2 Actores

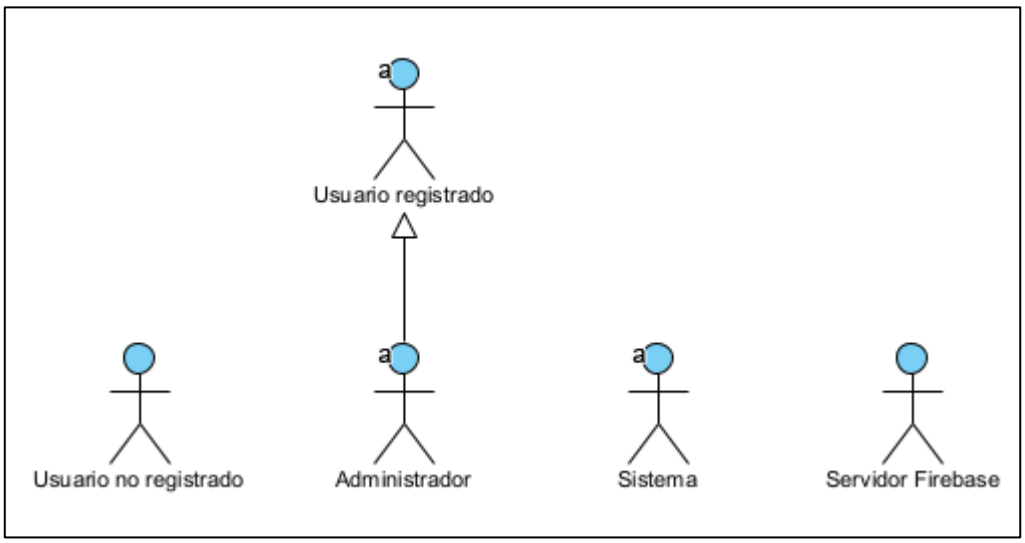


Ilustración 33 – Actores

- **Sistema**: este actor es capaz de realizar acciones por sí solo, representando nuestra plataforma.
- **Usuario no registrado**: este actor representa a un usuario nuevo en la aplicación que o bien cerró su sesión o que aún no se ha registrado. Solo podrá visualizar la pantalla de registro e inicio de sesión.
- **Usuario registrado**: este actor representa a todos aquellos usuarios que ya han iniciado sesión en la aplicación.
- **Administrador**: este actor representa al administrador de las pistas de pádel. Posee ciertos privilegios que el usuario registrado no tiene.
- **Servidor Firebase**: este actor representa al servidor de la aplicación. Es capaz de gestionar usuarios, reservas y devolver datos a la base de datos.

5.3.3 Requisitos de información

- **Usuarios**: el sistema deberá almacenar la información correspondiente a los usuarios registrados en la aplicación.
- **Reservas**: el sistema deberá almacenar la información correspondiente a las reservas realizadas por el usuario.
- **Estado de la reserva**: el sistema deberá almacenar la información en formato texto correspondiente a los estados de cada día y hora de las pistas.

5.3.4 Requisitos de restricción

- **Credenciales del usuario (email)**: la aplicación debe controlar que se hayan completado todos los campos correctamente, es decir, que el email tenga su formato correcto y que sea único.
- **Credenciales del usuario (contraseña)**: la aplicación debe controlar que se hayan completado todos los campos correctamente, es decir, que la contraseña sea de seis caracteres como mínimo y que los campos "Contraseña" y "Repetir contraseña" en el registro sean iguales.
- **Búsqueda de pistas disponibles**: el sistema no permite la búsqueda de fechas anteriores a la actual. Cuando se añade un día y hora en los filtros, se realiza una búsqueda de aquellas pistas que se encuentren libres para ese día y hora. Si no se encentran disponibles entonces estas no aparecerán en la pantalla correspondiente. Al entrar en una pista concreta aparecerán los botones de las horas disponibles en verde, el resto de botones se encontrarán deshabilitados y en color rojo o gris. Nunca se podrá reservar una pista ya reservada anteriormente.
- **Pagos con tarjeta**: el pago se tiene que efectuar con una tarjeta de crédito válida, de lo contrario la reserva nunca podrá realizarse.
- **Roles**: un usuario con rol administrador tendrá ciertos privilegios que el otro usuario no tendrá como, por ejemplo, visualizar las reservas realizadas de cualquier usuario o bloquear las pistas que él mismo desee.
- **Baja de un usuario**: el sistema deberá borrar definitivamente toda la información de un usuario que ha decidido eliminar la cuenta.

5.3.5 Requisitos no funcionales

- **Operatividad con Android**: el sistema deberá estar correctamente construido para utilizarse en dispositivos Android.
- **Integración con Firebase**: el sistema deberá integrarse con Firebase, el cual actuará como servidor de la aplicación realizando una gestión tanto de los usuarios como de las reservas además de informar de las estadísticas y los errores producidos en ella.
- **Lenguaje de desarrollo**: el sistema deberá estar desarrollado en los lenguajes Java y XML exceptuando para la verificación del código QR que se tratará con Arduino.
- **Interfaz gráfica**: el sistema deberá poseer una interfaz gráfica sencilla e intuitiva para el usuario que lo vaya a utilizar.
- **Eficiencia**: el sistema deberá ser eficiente y que la tasa de fallos sea inferior al 2%.
- **Acceso concurrente**: el sistema deberá ser escalable, es decir, que sea capaz de soportar el acceso de un número medianamente alto de usuarios al sistema o un número alto de reservas.
- **Mensajes de error**: el sistema deberá emitir mensajes de error informativos al usuario cuando estos se produzcan.

5.3.6 Requisitos funcionales

Los requisitos funcionales fueron divididos en bloques funcionales o subsistemas según su funcionalidad. A continuación, pondremos un ejemplo de cada bloque.

Bloque funcional gestión de usuarios:

Este bloque funcional lo conforman los casos de uso relacionados con la gestión de usuarios.

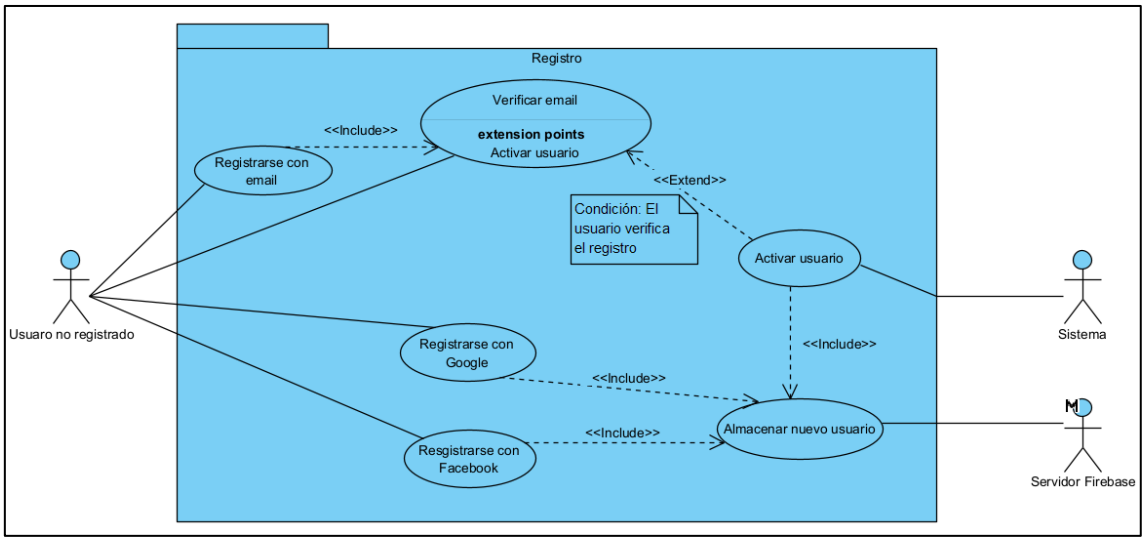


Ilustración 34 - Registro de usuarios

En este caso, añadimos un ejemplo del diagrama de casos de uso del registro de un nuevo usuario no registrado aún en la aplicación.

<u>Bloque funcional gestión de reservas:</u>

Este bloque funcional lo conforman los casos de uso relacionados con la gestión de reservas.

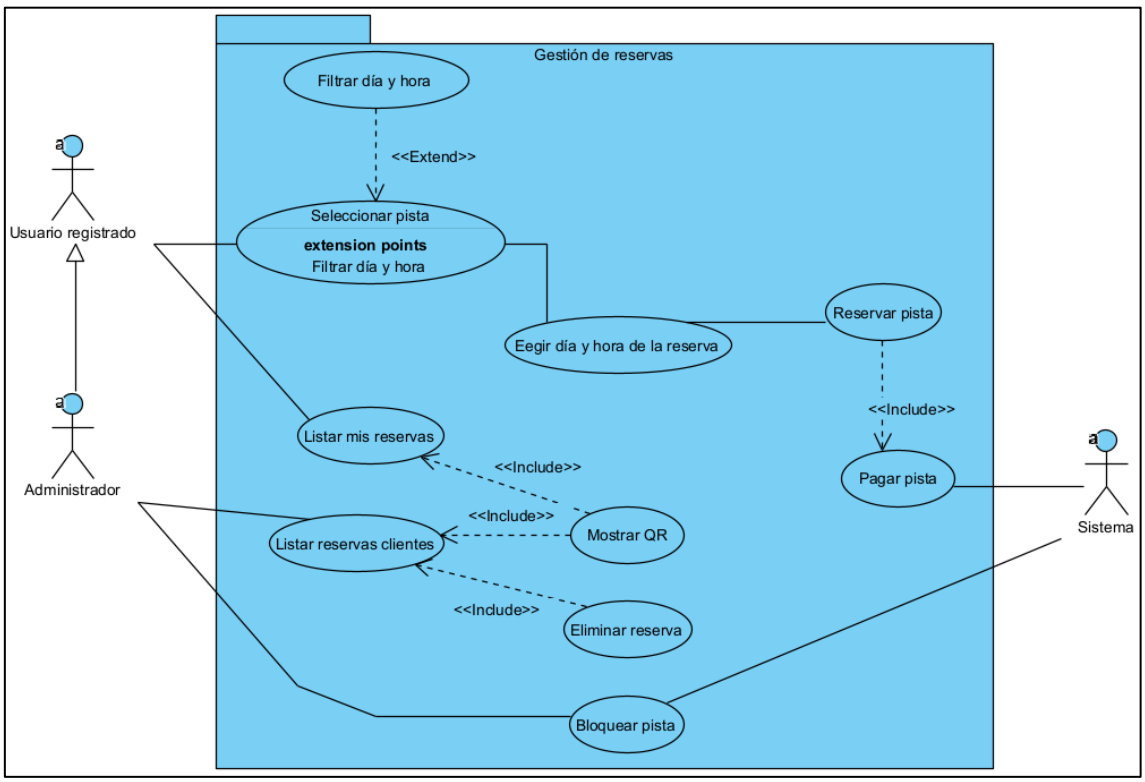


Ilustración 35 - Gestión de reservas

En este caso, añadimos un diagrama de casos de uso que engloba toda la gestión de reservas.

Bloque funcional Información

Este bloque funcional está formado por dos casos de uso. Se utiliza para informar sobre todos los datos de la instalación de pádel incluyendo su localización.

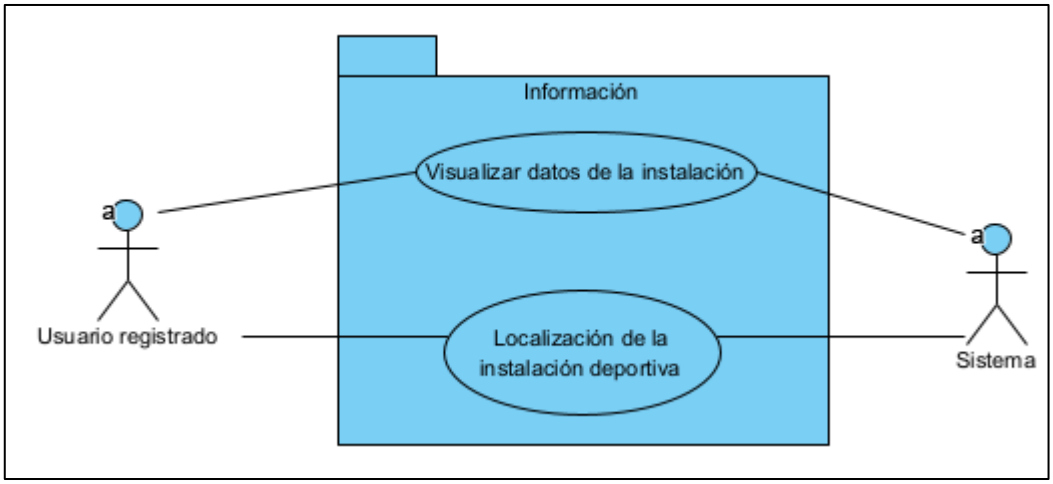


Ilustración 36 - Información

Bloque funcional Ayuda:

Este bloque únicamente tiene un caso de uso y se utiliza para enviar al correo de la instalación (en este caso, *reservas.padel20@gmail.com*) un mensaje de ayuda.

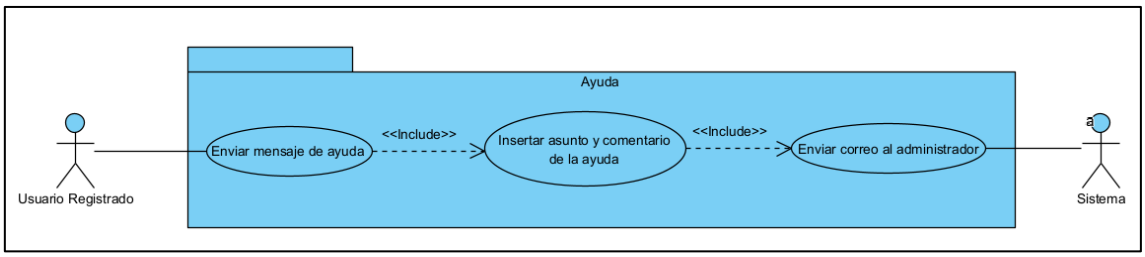


Ilustración 37 - Ayuda

5.4 Modelo de análisis

En este apartado se especificará el modelo de dominio realizado junto propuesta arquitectónica.

Si se precisa obtener una explicación más amplia respecto a este apartado se adjuntará junto a este documento el Anexo III – Modelo de análisis.

5.4.1 Modelo de dominio

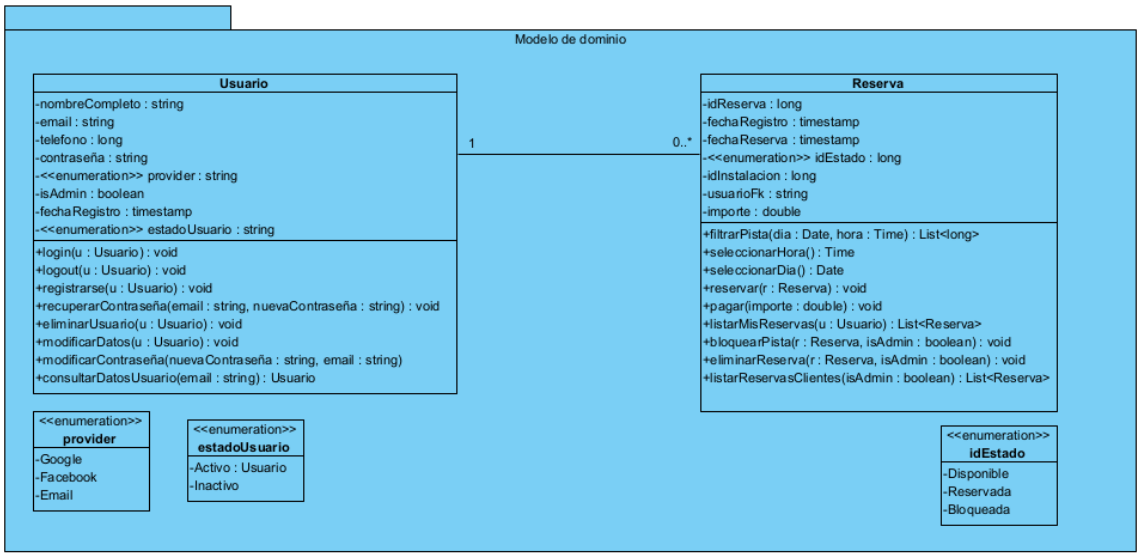


Ilustración 38 - Modelo de dominio

El sistema cuenta con distintos tipos de usuario: usuario no registrado, usuario registrado y administrador siendo el último un subtipo de usuario registrado.

Un usuario puede realizar cero o varias reservas mientras que una reserva solo puede estar realizada por un único usuario.

5.4.2 Propuesta de arquitectura

Una vez realizada la vista de interacción que podemos ver en el anexo correspondiente, el último paso consiste en realizar la propuesta de arquitectura con las entidades utilizadas en dichos diagramas.

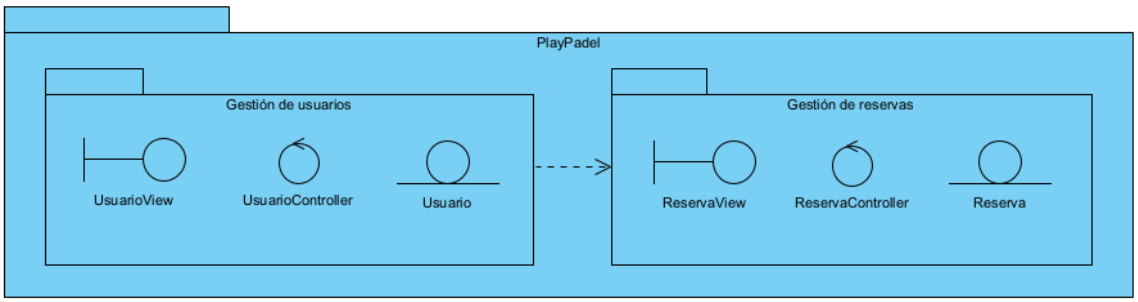


Ilustración 39 - Propuesta de arquitectura

5.5 Modelo de diseño

En este apartado se especificarán los subsistemas de diseño y servicio, las clases de diseño, los diagramas de despliegue y los de componentes.

Si se precisa obtener una explicación más amplia respecto a este apartado se adjuntará junto a este documento el Anexo III – Modelo de diseño.

5.5.1 Subsistemas de diseño y servicio

Los subsistemas de diseño y servicio realizados tienen que ver con la propuesta de arquitectura realizada en el apartado anterior, pero en este caso, siguiendo el patrón MVC.

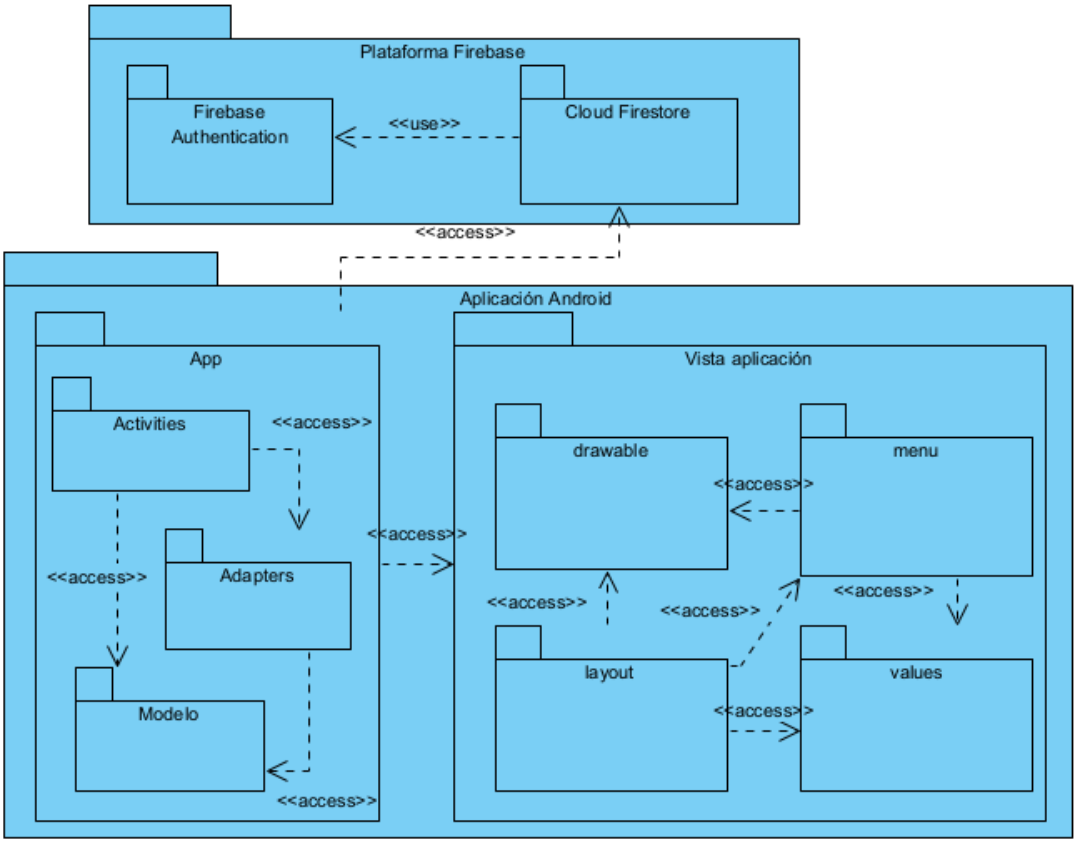


Ilustración 40 - Subsistema de diseño

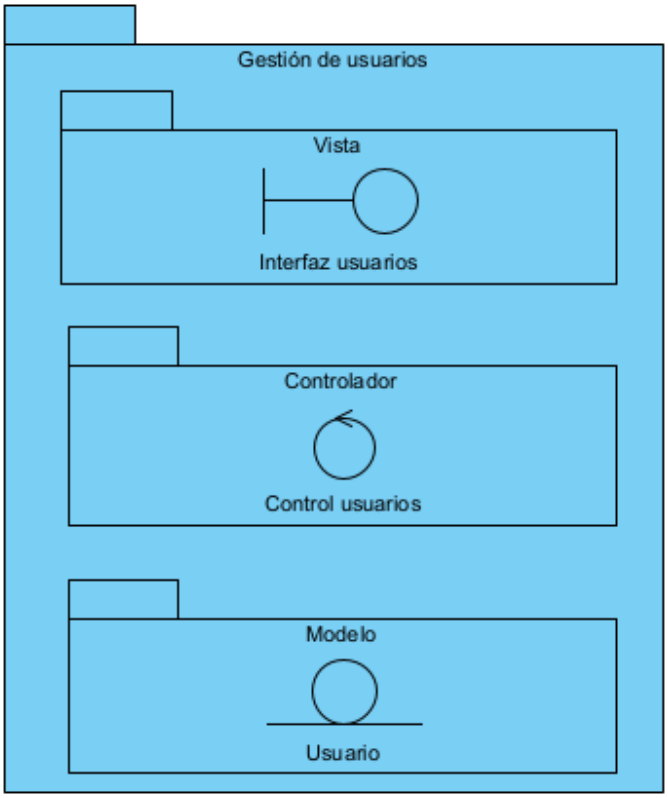


Ilustración 41 - Gestión de usuarios

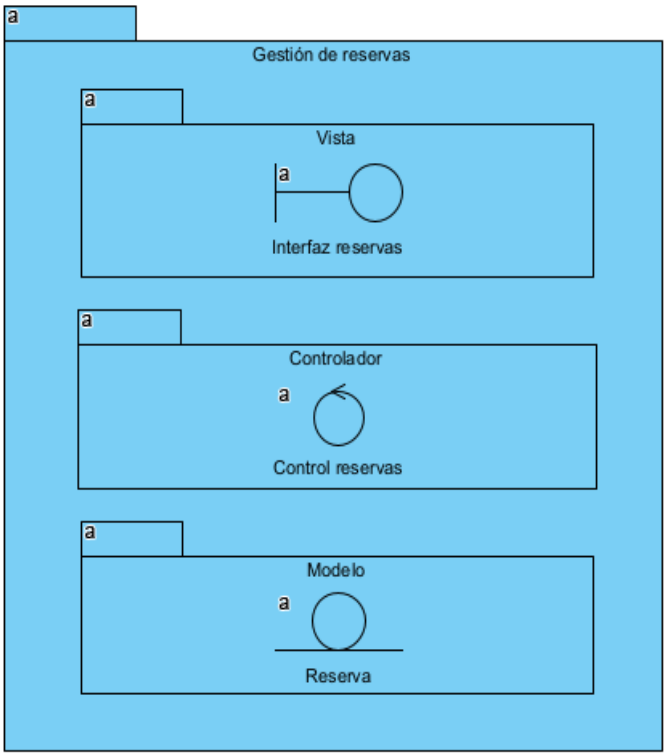


Ilustración 42 - Gestión de reservas

5.5.2 Clases de diseño

Las clases de diseño detallan más en profundidad los subsistemas del apartado anterior. Especifican los atributos y métodos principales que forman cada una de las clases.

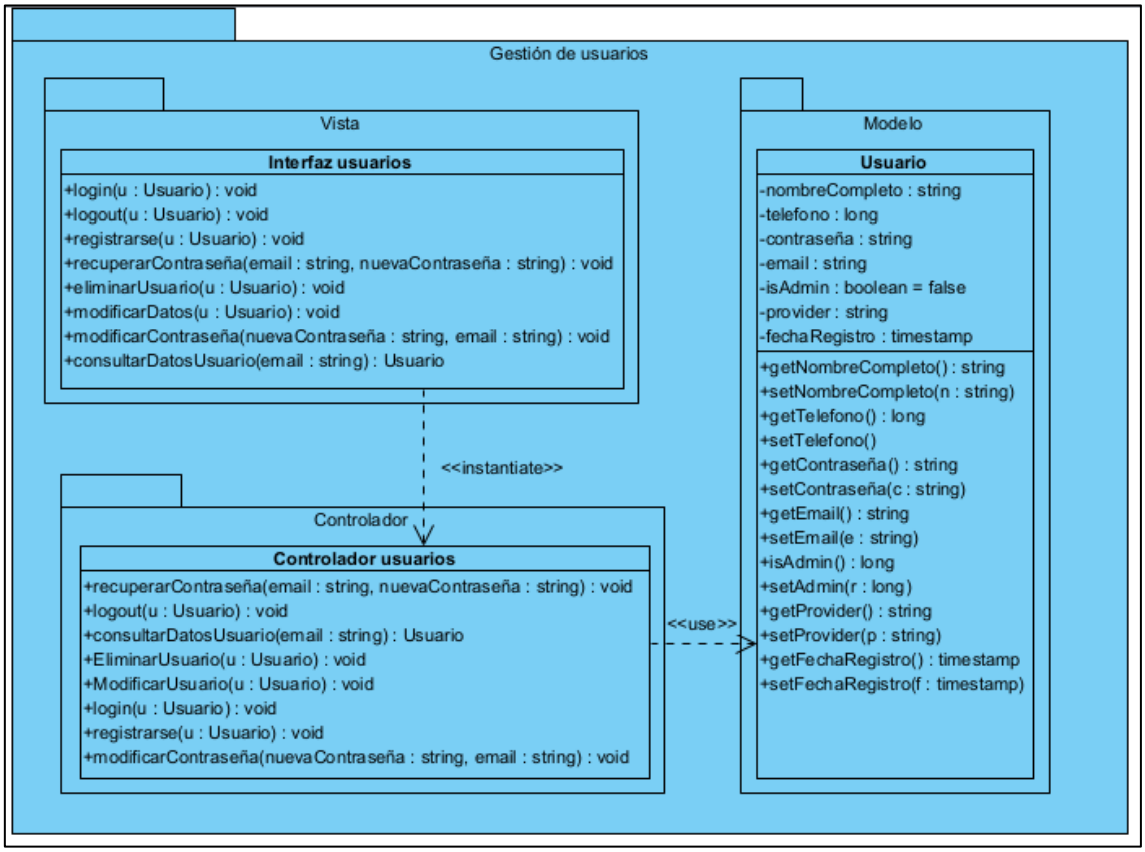


Ilustración 43 - Clase gestión de usuarios

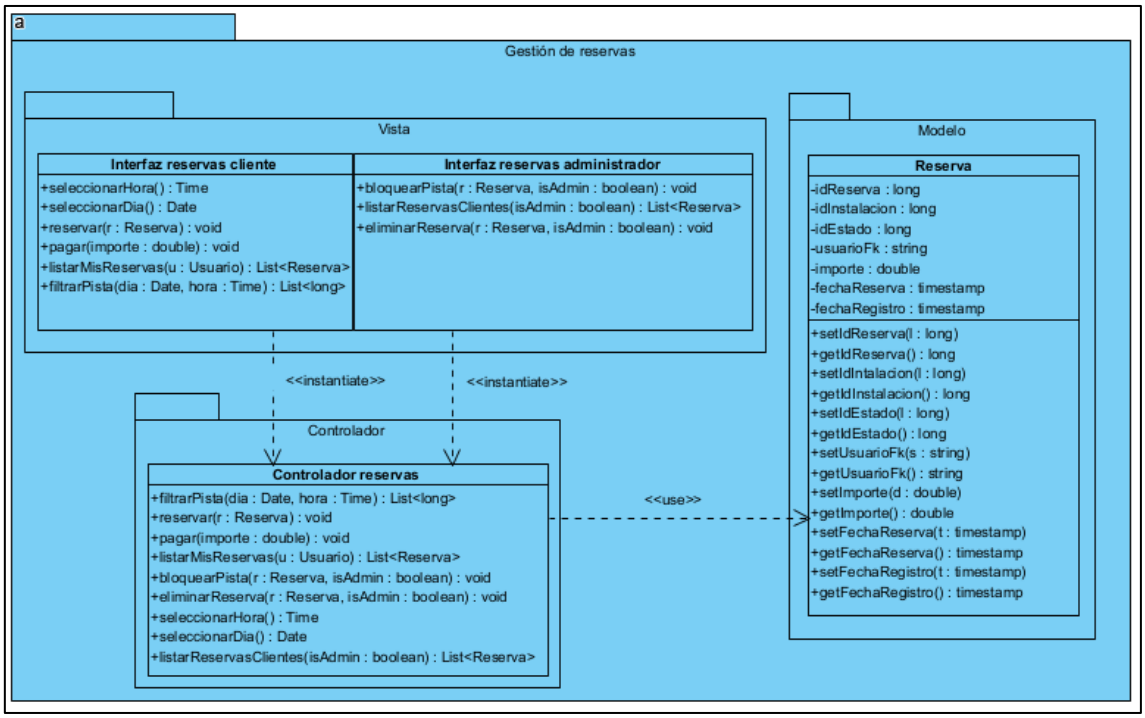


Ilustración 44 - Clase gestión de reservas

5.5.3 Diagramas de despliegue

La Ilustración 45 muestra la vista del modelo de despliegue, útil para modelar la descripción física de los artefactos del sistema, así como su distribución.

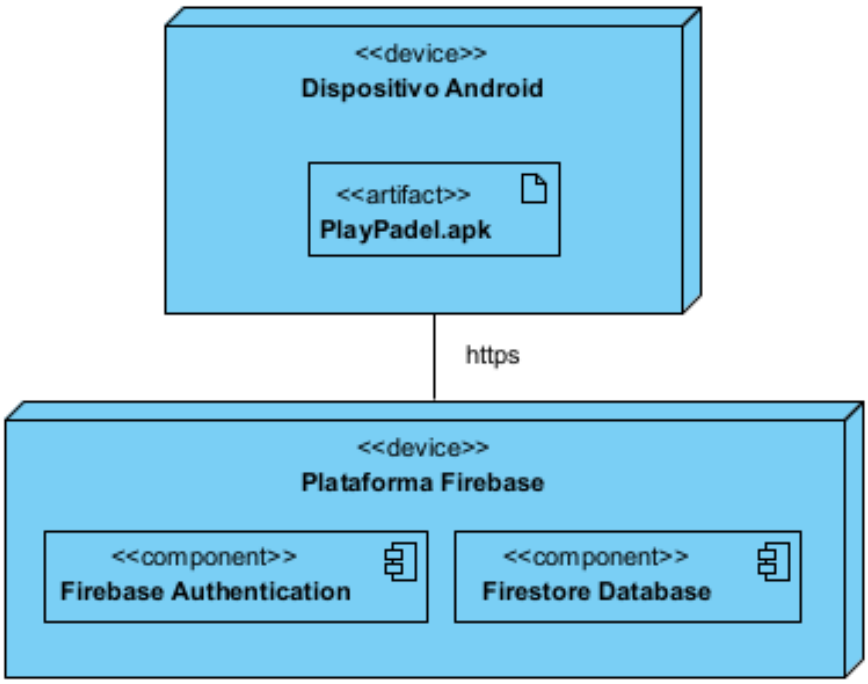


Ilustración 45 - Diagrama de despliegue

5.6 Implementación del sistema

Este sistema ha sido implementado siguiendo los esquemas de las fases anteriores. Como se puede apreciar en la planificación temporal esta ha sido la fase que más tiempo se necesitaba en el proyecto ya que se han utilizado algunas herramientas y técnicas que se desconocían por parte de la alumna y se necesitaba investigar sobre ello.

Para la aplicación Android se ha necesitado Java como lenguaje para el código fuente y XML para la interfaz gráfica. Se eligió Java por el conocimiento previo de este lenguaje de programación. La parte más costosa fue entender el lenguaje NoSQL y su conexión con la plataforma de Firebase.

Se ha llevado a cabo la arquitectura MVC (modelo-vista-controlador) donde:

- **Modelo**: contiene las clases del sistema definidas en el modelo de dominio, pero traducidas a lenguaje Java con sus atributos, *setters* y *getters*.
- **Vista**: contiene todas las clases que forman parte de la interfaz gráfica. Estas están escritas en lenguaje XML.
- **Controlador**: contiene todos los métodos utilizados en la aplicación. Sirve para mandar y recibir peticiones entre el modelo, la vista y la base de datos.

Cabe destacar algunos aspectos importantes del código que iremos destacando en este apartado:

1. En la Ilustración 46 se puede observar las funciones de Firebase utilizadas para la creación de un usuario junto con su correspondiente envío del email de verificación utilizando las siguientes dependencias tanto para esto como para su posterior almacenamiento en la base de datos:

```
implementation 'com.google.firebase:firebase-firestore:22.1.0'
implementation 'com.google.android.gms:play-services-auth:19.0.0'
```

```
FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
    @Override
    public void onSuccess(AuthResult authResult) {
        FirebaseAuth.getInstance().getCurrentUser().sendEmailVerification().addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if(task.isSuccessful()) {
```

Ilustración 46 - Uso del lenguaje NoSQL

2. En la Ilustración 47 podemos observar el uso de la clase AlertDialog para indicar, por ejemplo, que se ha enviado un mensaje de verificación y se ha de verificar la cuenta para poder iniciar la sesión.

```
AlertDialog.Builder alerta = new AlertDialog.Builder( context: RegistroActivity.this);
alerta.setMessage("Se ha enviado un correo de verificación.\nPor favor, verifíca tu email para poder ser miembro de PlayPadel")
        .setCancelable(false)
        .setPositiveButton( text: "Aceptar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent i = new Intent( packageContext: RegistroActivity.this, AuthActivity.class);
                startActivity(i);
            }
        });
AlertDialog titulo = alerta.create();
titulo.setTitle("Verificar cuenta");
titulo.show();
```

Ilustración 47 – AlertDialog

3. En la Ilustración 48 se observa la manera que se ha realizado para salir de la aplicación desde la pantalla de inicio de la aplicación. Se ha de presionar dos veces seguidas el botón de atrás para salir. En el resto de pantallas si le das una vez hacia atrás o al botón superior izquierdo representado por un icono de una flecha se irá a la pantalla anterior.

```
@Override
public void onBackPressed() {
    if (contador == 0){
        Toast.makeText(getApplicationContext(), text: "Presione de nuevo para salir", Toast.LENGTH_SHORT).show();
        contador++;
    }else{
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.addCategory(Intent.CATEGORY_HOME);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    new CountDownTimer( millisInFuture: 3000,  countDownInterval: 1000){

        @Override
        public void onTick(long millisUntilFinished) {

        }

        @Override
        public void onFinish() { contador = 0; }
    }.start();
}
```

Ilustración 48 - Presione dos veces para salir

4. Para el sistema de pagos se ha hecho uso de Stripe (Ilustración 49 e Ilustración 50). Se han utilizado varios métodos para que funcione correctamente en un entorno de pruebas. Si se quiere subir a la Google Play Store y comercializarlo se tendría que modificar. Se han utilizado las siguientes dependencias:

```
implementation 'com.stripe:stripe-android:15.1.0'
implementation 'com.squareup.okhttp3:okhttp:4.4.0'
implementation 'com.google.code.gson:gson:2.8.6'
```

```
private static final class PaymentResultCallback
        implements ApiResultCallback<PaymentIntentResult> {
    @NonNull private final WeakReference<CheckoutActivity> activityRef;
    PaymentResultCallback(@NonNull CheckoutActivity activity) {
        activityRef = new WeakReference<>(activity);
    }
    @Override
    public void onSuccess(@NonNull PaymentIntentResult result) {
        final CheckoutActivity activity = activityRef.get();
        if (activity == null) {
            return;
        }
        PaymentIntent paymentIntent = result.getIntent();
        PaymentIntent.Status status = paymentIntent.getStatus();
        if (status == PaymentIntent.Status.Succeeded) {
            // Payment completed successfully
            Gson gson = new GsonBuilder().setPrettyPrinting().create();
            ReservaActivity reservaActivity = new ReservaActivity();
            reservaActivity.anadirReservaBBDD( estado: 1, reserva, usuario, activity);

        } else if (status == PaymentIntent.Status.RequiresPaymentMethod) {
            // Payment failed – allow retrying using a different payment method
            activity.displayAlert(
                    title: "Payment failed",
                    Objects.requireNonNull(paymentIntent.getLastPaymentError()).getMessage()
            );
        }
    }
```

Ilustración 49 - Sistema de pagos

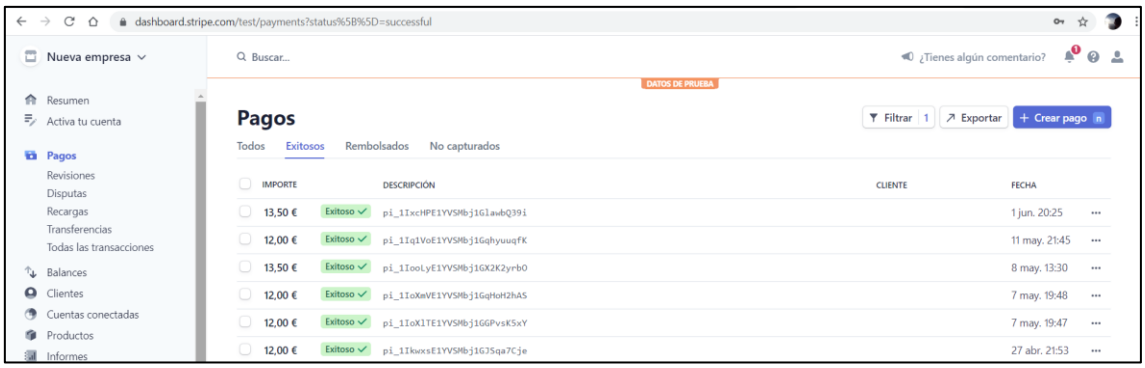


Ilustración 50 – Vista de pagos realizados en la plataforma Stripe

5. Cabe destacar también el envío de correos mediante el protocolo SMTP como se muestra en la Ilustración 51 y en la Ilustración 52. Este se envía directamente al correo *reservas.padel20@gmail.com* para que el administrador pueda comprobar

qué necesita el usuario. El nombre y correo del usuario que envía la ayuda también se mostrará en el correo.

```
public void onClick(View v) {
    btnEnviar.setEnabled(true);
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);
    Properties prop = new Properties();
    prop.put("mail.smtp.host", "smtp.googlemail.com");
    prop.put("mail.smtp.socketFactory.port", "465");
    prop.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
    prop.put("mail.smtp.auth", "true");
    prop.put("mail.smtp.port", "465");

    try{
        session = Session.getDefaultInstance(prop, getPasswordAuthentication() → {
                return new PasswordAuthentication(CORREO, PASSWORD);
        });

        if(session != null){
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(CORREO));
            message.setSubject("AYUDA: "+asuntoET.getText().toString());
            message.setRecipients(Message.RecipientType.TO, InternetAddress.parse( addresslist: "reservas.padel20@gmail.com"));
            String texto="<b>El usuario "+ nombre +" con email " + email + " ha solicitado AYUDA al siguiente problema:</b><br><br>" +
            message.setContent(texto,  type: "text/html; charset=utf-8");

            Transport.send(message);
            Toast.makeText(getApplicationContext(),  text: "Correo enviado", Toast.LENGTH_LONG).show();
            asuntoET.setText("");
            textoET.setText("");
        }
    }catch (Exception e){
        e.printStackTrace();
    }
```

Ilustración 51 - Envío correo ayuda

Ilustración 52 - Correo de ayuda

6. Otra acción a destacar es la implementación de Google Maps en la aplicación (Ilustración 53). Se han utilizado las siguientes dependencias:

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

```
private void initGoogleMap(Bundle savedInstanceState){
    Bundle mapViewBundle = null;
    if(savedInstanceState != null){
        mapViewBundle = savedInstanceState.getBundle(MAPVIEW_BUNDLE_KEY);
    }
    mMapView = (MapView) findViewById(R.id.map);
    mMapView.onCreate(mapViewBundle);

    mMapView.getMapAsync( onMapReadyCallback: this);
}
```

Ilustración 53 - Google Maps

7. Por último, se destacará la generación del código QR que se realizará gracias a la siguiente dependencia y de la manera que se muestra en la Ilustración 54:

```
implementation 'com.github.kenglxn.QRGen:android:2.5.0'
```

```
private void generarQR(){
    int anchura = 800;
    int altura = 800;
    Bitmap bitmap = QRCode.from(reserva).withSize(anchura, altura).bitmap();
    ImageView imagenQR = findViewById(R.id.imageQR);
    imagenQR.setImageBitmap(bitmap);
}
```

Ilustración 54 - Generar código QR

6. CONCLUSIONES

Como conclusión principal, se puede afirmar que se han cumplido los objetivos marcados al inicio del proyecto, destacando además los siguientes aspectos:

- El sistema cuenta con una gestión de sus usuarios, alberga la información de registro de estos y el inicio de sesión mediante Google, Facebook y la manera básica del correo electrónico y contraseña.
- El sistema es capaz de gestionar reservas, es decir, poder consultar, reservar, pagar y obtener el código QR para el acceso de una pista concreta a una fecha y hora concreta.
- Dar opción al administrador de poder consultar cada una de las pistas reservadas por los clientes para llevar un control de todas las reservas que se producen en su instalación deportiva.
- El sistema es capaz de que tanto el usuario como el administrador pueda acceder correctamente al código QR de cada una de las reservas para poder abrir correctamente la puerta de la pista reservada (o, en el caso del administrador, también de las pistas bloqueadas).
- El sistema cuenta con un control de la geolocalización de la instalación y su acceso a Google Maps para ubicar correctamente la instalación deportiva.
- La aplicación está integrada correctamente con la plataforma Firebase, dotando al sistema de una gran eficiencia, robustez e inmediatez de la información.
- La aplicación está desarrollada para dispositivos Android, mediante Java para el código fuente, y XML para la interfaz. Ambos lenguajes se han implementado satisfactoriamente en el sistema gracias a los conocimientos previos del autor.
- La aplicación Android cuenta con una interfaz simple y a la vez intuitiva, lo que hace que los usuarios tengan una buena experiencia de navegación.

Cabe destacar también que, gracias a la realización de este proyecto, la alumna ha logrado adquirir conocimientos en nuevos campos que anteriormente se desconocían, como la plataforma Firebase y la familiarización con el IDE Android Studio.

Por último, también se han reforzado conocimientos aprendidos en la titulación, como el lenguaje de programación Java, Arduino y la familiarización con el Proceso unificado de desarrollo.

7. LÍNEAS DE TRABAJO FUTURO

A partir del sistema presentado y, debido en gran parte a los límites temporales de la asignatura, el sistema podría ser evolucionado de múltiples formas si se decidiera continuar trabajando con él. A continuación, se presentará una lista con las posibles funcionalidades que se podrían implantar en el sistema:

- **Incorporar un sistema de reserva en distintas instalaciones deportivas**: se podría llevar a cabo una aplicación donde diese opción a elegir entre varias instalaciones distintas y no solo las de un centro concreto. De esta manera, se podría conseguir un mayor número de usuarios ya que englobarías más zonas deportivas ya sea de distintas provincias o en la misma.
- **Dar opción de reservas sin equipo**: se podría crear una manera en la cual un usuario reserve una pista, dando la opción de indicar su nivel y cuantos jugadores necesitaría para terminar el equipo. De esta manera, si en el pádel por parejas se necesitan cuatro personas y solo son dos pueden buscar gente mediante la aplicación y poder formar un grupo de cuatro. Esto es muy interesante ya que hay muchos clientes que les gustaría jugar, pero no son suficientes para ello.
- **Integrar pagos con PayPal**: muchos usuarios no suelen querer realizar pagos con tarjeta y ven más seguro hacerlo por una plataforma como lo es PayPal. Esto sería una gran mejora ya que quizá sin ello se podrían perder muchos clientes.
- **Añadir la opción de reservas múltiples**: dar la opción de reservar una pista varios días u horas a la vez ya que muchos clientes quizá les gustaría reservar, por ejemplo, tres días a la semana a una hora concreta. En la aplicación actualmente tendrías que ir pagando una por una en vez de realizar un único pago.
- **Desarrollar el sistema en otras plataformas**: aprovechando que las herramientas de la plataforma Firebase descritas en este documento son multiplataforma, este sistema se podría implementar en otras plataformas, como iOS o un servicio web.

Todas estas líneas de trabajo futuras son bastante realistas y dotarían al sistema en caso de ser hipotéticamente implementadas una mayor distinción y aprobación por parte de los usuarios.

8. BIBLIOGRAFÍA

1. Firebase documentación (consultado por última vez en mayo 2021): https://firebase.google.com/docs/
2. API (consultado por última vez en junio 2021) – Business insider: https://www.businessinsider.es/api-sirve-todo-necesitas-saber-861403?gclid=CjwKCAjwoNuGBhA8EiwAFxomA_iUYk6jYA_Zhcyva9GUYPvAINgitDW9E8ogM49y_qbMa6LZZEA2AhoCvqQQAvD_BwE
3. SDK (consultado por última vez en junio 2021) – Red Hat: https://www.redhat.com/es/topics/cloud-native-apps/what-is-SDK
4. Android SDK (consultado por última vez en junio 2021) - Xataka: https://www.xatakandroid.com/tutoriales/como-instalar-el-android-sdk-y-para-que-nos-sirve
5. Arquitectura cliente-servidor (consultado por última vez en junio 2021) – Infranet working: https://blog.infranetworking.com/modelo-cliente-servidor/
6. Base de datos NoSQL (consultado por última vez en junio 2021) – GMV Blog: https://www.gmv.com/blog_gmv/language/es/el-reto-de-las-bases-de-datos-nosql/
7. Referencia ilustración NoSQL (consultado por última vez en junio 2021) – IThink: https://www.ithinkupc.com/es/blog/sql-nosql-newsql-que-son-historia-y-eleccion/
8. Android Studio (consultado por última vez en junio 2021): https://developer.android.com/studio/releases
9. Stripe (consultado por última vez en junio 2021): https://stripe.com/es
10. Arduino (consultado por última vez en junio 2021): https://www.arduino.cc/en/guide/introduction
11. Ilustraciones de Arduino UNO y MEGA (Consultado por última vez en junio 2021) – Xataka: https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno
12. Proceso unificado (consultado por última vez en junio 2021) – Blog PNF: http://pnfiingenieriadesoftwaregrupocuatro.blogspot.com/2012/07/proceso-unificado-proceso-unificado-de.html