**APLICACIÓN PEÑAPP**

**Alumno: José Rodríguez Baena**

**Tutor: Andrés Parra**

**PEÑAPP**

1. Definición del sistema

Se quiere realizar una aplicación para dispositivos móviles destinado a hacer un uso y gestión de las peñas para que puedan realizar dichas tareas de una forma más actualizada y moderna, facilitando así las principales tareas y llevando un seguimiento de sus acciones, interacciones de los usuarios con la app, etc. La idea es que cualquier peña pueda usar la aplicación, tan sólo un usuario ha de darse de alta como administrador y compartir un código autogenerado que funcionará a modo de invitación para el resto de individuos de la peña.

Actualmente no existen aplicaciones de este tipo, las más cercanas proponen la gestión de peñas deportivas de forma muy rudimentaria aunque suficiente. El alcance que puede tener Peñapp no es baladí. A pesar de que las peñas son más bien una realidad rural, pocos son los que, aún siendo de ciudad, no disponen de un pueblo y de una peña de la cuál disfrutar en fiestas patronales, vacaciones, quintos, etc. Eso significa multitud de peñas gestionando los pedidos, eventos, cuotas… de forma manual, o bien, acudiendo a diversas apps no especializadas para tal fin.

El login y registro permite crear peña y usuarios asociados a tal peña. Un usuario no puede pertenecer a más de una peña.

La app ha de ser útil para los siguientes campos: Pedidos, Eventos y Finanzas. Con ello permitir a un usuario hacer un pedido, crear un evento e informar de su pago de cuota.

Se usarán las siguientes tecnologías para su desarrollo:

* Lenguaje de programación: Kotlin.
* Entorno de desarrollo: Android Studio
* Git y github para el seguimiento del avance/versiones
* BBDD: Room

La app ha de evitar mucha complejidad, precisamente por la variedad de usuarios que pueda llegar a tener. Así pues se identificó con el cliente (la persona que gestiona mi propia peña desde hace años) la implantación de varias secciones en donde llevar a cabo esta gestión: Pedidos, Eventos y Finanzas. La app ha de partir, por supuesto, de un login que permita el registro de una nueva peña y que mediante código deje ingresas al resto de usuarios, los cuales solo pueden pertenecer a dicha peña. Si se logean, ya lo harán automáticamente dentro de su peña.

FASES DE DESARROLLO:

* Análisis del sistema y requisitos:

Se produjo la discusión con el cliente sobre las necesidades a cubrir que debiera tener la app. Debía ser un sistema sencillo y manejable, pero directo. Aquí se identificó la necesidad de usar Room y el patrón ORM para mayor rapidez de transferencia de datos sin que la app sea muy pesada. Se identificó la necesidad de una activity base sobre la cual parten las demás. Las secciones tendrán varias opciones como fragments. Los usuarios pueden realizar pedidos y eventos, y hacer un seguimiento de los mismos. Toda la peña podrá ver los próximos eventos disponibles.

* Diseño del software:

Se comienza desarrollando la capa ORM de la aplicación. El punto de partida fue diseñar la base de datos, identificando las entidades y sus relaciones. A partir de ahí pude implementar el modelo en código, preparándolo para Room y para el uso en la App, diseñando las entities, los Daos,los mapper y las clases modelo. La idea es seguir el patrón MVVM. También se diseña el flujo de datos de la app, desde el login hasta las distintas secciones, haciendo un mapa general del funcionamiento de la aplicación.

* Programación e implementación:

A partir del modelado del ORM se comenzaron a desarrollar las activities y el flujo de datos. Partiendo de la main activity derivaría en una activity login. Para hacerla más rápida y gestionable, el apartado de registro, tanto de peña como de usuario, serían implementados en fragments, matando la activity en cuando se hubiesen registrado o logeado. Devueltos a la main activity se enseña la home, que muestra información y permite navegar hacia cada una se las secciones, las cuales tendrán sus propias activities centrales y, a partir de ellas, los fragments correspondientes hasta que el usuario termine la gestión y se mate la activity, regresando a home.

* Pruebas y revisión:

Se realizan las pruebas de código a medida que se va programando e implementando si el código lleva nuevas funcionalidades o complejidades. Además se realizan pruebas una vez terminada un fragment o activity, asegurándose de su correcto funcionamiento. En caso de detectar fallos o incorreciones (cosa frecuente), se vuelve a la fase de diseño del software y se repite el ciclo. Al terminar una sección se enseña al cliente y se discute sobre posibles mejoras, lo que provoca de nuevo repetir el ciclo de desarrollo

1. Análisis del sistema

Requisitos:

* Login/Registro:

El login ha de hacerse con el correo electrónico de la persona en cuestión. La persona inicial que registre la peña será el administrador, pudiendo acceder al código autogenerado de invite para el resto de miembros.

* Home:

La home ha de ser sencilla y descriptiva, mostrar los eventos próximos y el saldo, además de permitir la navegación hacia las secciones.

* Pedidos:

La sección de pedidos ha de permitir realizar pedidos según una categoría predefinida (se han seleccionado las más usuales que se pueden producir en una peña, amén de poder aumentar la información en la descripción)

El usuario puede ver sus pedidos y el administrador todos los pedidos de la peña.

* Eventos:

La sección de eventos permite crear eventos, ver los eventos personales y los eventos pasados. Los eventos más cercanos al presente pueden verse en la home también, para un mejor seguimiento de éstos. La fecha puede ser seleccionada cómodamente en un calendario, sin necesidad de escribirla.

* Finanzas:

La sección finanzas permite llevar un seguimiento del saldo de la peña, los usuarios pueden pagar cuotas, validadas por el administrador. El administrados puede añadir facturas.

1. Diseño del sistema

* Diseño de interfaces:

Se ha de implementar interfaces sencillas de pocos colores, se priorizarán los colores fríos, sin elementos demasiado llamativos. Es lo que he observado buscando entre apps que generen pedidos y/o eventos.

* Diseño del modelo de datos:

La relación entre tablas de la base de datos y la aplicación en ORM se hace a través de entities y daos. Las entidades son objetos que equivalen a las tablas de la aplicación. Los dao son Data Access Object, con ellos se pueden realizar consultas e insertar y borrar datos. Para desarrollar el patrón MVVM y poder relacionar los datos a la capa modelo se mapean las entities transformándolas en objetos de la capa modelo para ser más manejables. Además de eso he necesitado desarrollar funciones convertidoras de datos para poder insertar correctamente ciertos tipos de datos como fechas. Se ha tratado de aprovechar al máximo el modelaje de clases, delegando responsabilidades de las activities hacia los fragments para un mejor flujo de datos sin tener que ir y volver de unas activities a otras de forma repetitiva.

1. Pruebas de Accesibilidad y Usabilidad

Se ha tratado de realizar una aplicación sencilla que con pocos “clicks” o acciones del usuario se pueda abarcar la totalidad de la aplicación sin por ello perder la funcionalidad necesaria para la gestión y orden de una peña y la facilidad para el administrador de tener los datos necesarios para tal fin.

5. Problemas encontrados

Considero que el proyecto era demasiado complejo para las capacidades adquiridas. Traté de usar patrones de diseño actuales y eficientes para evitar el problema de realizar una aplicación desde cero sin ningún tipo de orden. Cosa la cuál también hice con el modelaje de datos y el ORM, a parte de que Kotlin era un lenguaje relativamente nuevo para mí. El proyecto ha requerido un enorme tiempo de estudio para comprender y entender el cómo y el qué del desarrollo de una buena aplicación. Ha faltado bastante competencia técnica y lógica, lo que significó sobreestimar mis capacidades y no poder cumplir con los plazos a tiempo.

1. Conclusiones:

Considero haber hecho un buen trajo en el desarrollo de las partes logradas, pero a costa de no haber podido realizar el proyecto complejo. He dejado inconclusas varias partes del anteproyecto como es la sección de finanzas, el modo administrador de la aplicación para tener listado de pedidos y eventos, y algunas cosas más.

Mejoras:

* Uso de alguna metodología ágil:

No usé ninguna, y eso me perjudicó a la hora de llevar un seguimiento de la evolución del proyecto, el punto real en el que me encontraba, o repartir mejor las tareas a realizar.

* Mejor documentación del proyecto y de las tecnologías necesarias:

Sin una verdadera información de saber cómo manejar las tecnologías que he utilizado, he sufrido graves retrasos que se podrían haber solucionado habiendo estudiado un poco las tecnologías a usar antes de decidirme por cual tirar.

* No sobreestimar el tiempo, repartirlo de forma adecuada:

Respetar los días y horas de trabajo, así como los de descanso.