Programación en dispositivos móviles (06MASW)

ACTIVIDAD 1: Aplicación Android

SpendTogether

Miembros del equipo:

Natalia Santos Martínez
Adrián Hernández Monterroso
Ana Regalado Arregui

Git:
https://github.com/NataliaSantosMart/DispositosMoviles

ÍNDICE

ÍNDICE FIGURAS

1. Introducción

Se ha desarrollo una aplicación móvil nativa para Android, denominada “SpendTogether”. Esta iniciativa busca aplicar los conocimientos teóricos y prácticos adquiridos durante la primera parte del curso, mientras se emplean herramientas y metodologías que faciliten la gestión de tareas y el seguimiento del progreso del proyecto.

1.1. Motivación

Esta aplicación podría ser especialmente útil en situaciones donde los gastos son compartidos de manera equitativa entre todos los miembros del grupo, independientemente de quién haya realizado la compra inicial. Por ejemplo, en vacaciones, regalos de cumpleaños, cenas grupales o cualquier otra actividad en la que se compartan gastos de manera equitativa entre todos los participantes.

Al proporcionar una solución simple y justa para dividir los gastos, la aplicación podría ayudar a evitar malentendidos y conflictos sobre quién debe pagar qué cantidad. Además, al facilitar la gestión de los gastos compartidos, la aplicación puede ahorrar tiempo y esfuerzo a los usuarios, lo que la hace aún más atractiva.

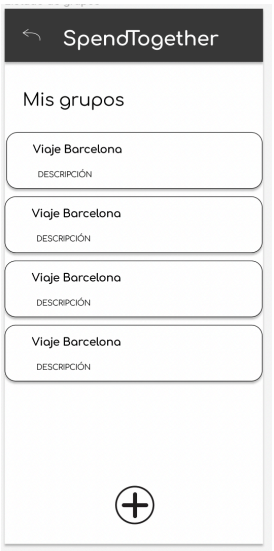
1.2. Objetivos

Los objetivos para la implementación de SpendTogether han sido:

- **Facilitar la gestión de gastos compartidos:** Permitir a los usuarios registrar y compartir fácilmente los gastos realizados en actividades grupales.
- **Promover la equidad financiera**: Calcular automáticamente las contribuciones de cada usuario según los gastos totales compartidos.
- **Mejorar la transparencia en la división de gastos**: Proporcionar un registro claro y detallado de todos los gastos compartidos y las contribuciones de cada miembro del grupo. Permitiendo a los usuarios visualizar fácilmente quién ha contribuido con qué cantidad y para qué gastos.
- **Ofrecer una experiencia intuitiva**: Diseñar una interfaz de usuario amigable y fácil de usar para garantizar una experiencia fluida para todos los usuarios,

2. Mockups

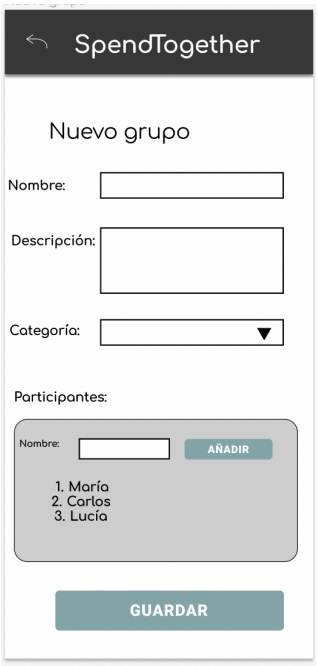
2.1. Listado de grupos



A continuación se puede observar el mockup correspondiente al listado de todos los grupos que han sido añadidos por el usuario. Este listado contendrá el nombre del grupo y una pequeña descripción. Desde esta vista se redireccionará a la ventana de añadir un nuevo grupo pulsando sobre el botón “+”.

1. Listado de grupos

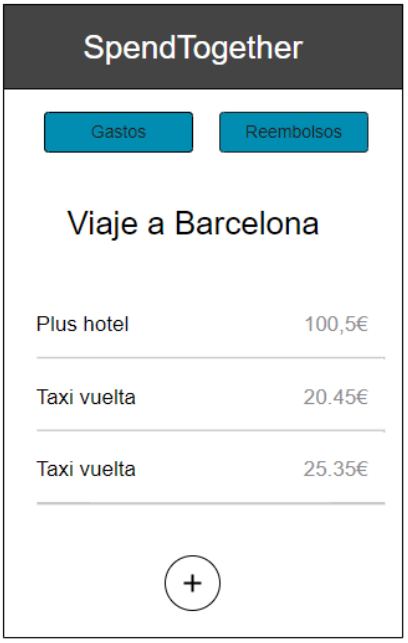
2.2. Nuevo grupo



La vista para la creación de un nuevo grupo debe incluir el nombre, la descripción, la categoría a la que pertenece y los participantes que serán miembros del grupo.

2. Nuevo grupo

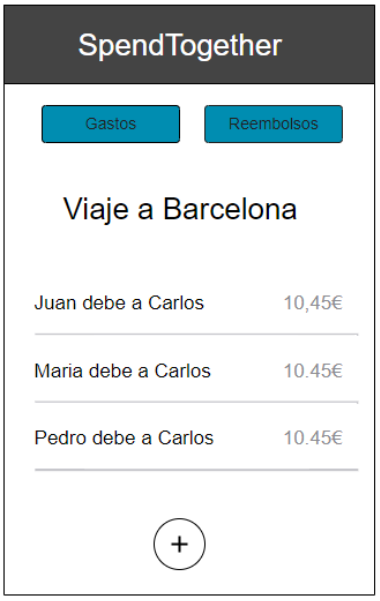
2.3 Vista de gastos



Esta es la vista donde se puede ver la lista de todos los usuarios que están relacionados con el grupo seleccionado. A partir de esta pestaña puedes navegar a la pestaña de reembolsos para poder añadir un gasto

3. *Vista de gastos*

2.4 Vista de Reembolsos



En esta vista se puede ver cual es la cuentia que debe cada usuario y a quien se lo deben de pagar. A partir de esta puedes navegar a ver los gastos o realizar uno nuevo

4. *Vista de reembolsos*

3. Funcionamiento aplicación

Para describir el funcionamiento de la aplicación se va a detallar que es lo que ocurre en cada una de las 4 vistas principales:

3.1. Listado de grupos (vista inicio)

La vista representada en el archivo XML (activity_main.xml) y su lógica en Kotlin definen la pantalla principal de una aplicación Android que muestra una lista de grupos.

El código Kotlin asociado define la funcionalidad de la actividad MainActivity. En el método onCreate(), se configura la vista y se inicializan los elementos, como el RecyclerView y el botón flotante. Además, se realiza una solicitud al servidor para obtener la lista de grupos, y una vez que se completa la solicitud, se actualiza el RecyclerView con los datos obtenidos.

```
//Crear la petición
    val apiGroupsService = RetrofitServiceFactory.getApiService()
    lifecycleScope.launch { this: CoroutineScope
        //Hacemos petición
        val data = apiGroupsService.getGroups(("groups"))
        groupsInit.clear()
        groupsInit.addAll(data)
        //Repintar el RecyclerView
        groupsAdapter.notifyDataSetChanged()
    }
```

5. Funcionamiento listado de grupos

3.2. Crear grupo

En esta vista se realiza la acción de crear un nuevo grupo, esto lo hace el método initSaveGroup que recoge todos los campos del formulario, comprueba de que no esten vacios (si alguno lo esta se muestra un error) y posteriormente se realiza una acción en el servidor que guarda dicho grupo

6. Funcionamiento crear grupo

3.3. Listado de gastos

La actividad UsersChargesActivity en la aplicación Android está diseñada para mostrar y gestionar los cargos de usuarios dentro de un grupo específico. La vista asociada a esta actividad, definida en el archivo XML activity_users_charges.xml, ofrece una interfaz donde los usuarios pueden visualizar los cargos de cada usuario.

Después de obtener el nombre del grupo, se hace otra solicitud a la API para recuperar todos los gastos (expenses). Estos gastos se filtran para obtener solo los asociados al grupo actual, lo que garantiza que solo se muestran los cargos relevantes para el usuario en la lista. A partir de esta pantalla podemos navegar a la vista de reembolsos donde se pueden añadir gastos al grupo.

7. Funcionamiento crear grupo

3.4. Crear gasto

En esta vista se realiza la acción de crear un nuevo gasto, esto lo hace el método initSaveExpense que recoge todos los campos del formulario, comprueba de que no esten vacios (si alguno lo esta se muestra un error) y posteriormente se realiza una acción en el servidor que guarda dicho gasto

8. Funcionamiento crear gasto

Servicios web

Para poder realizar las peticiones al servidor se han creado una serie de archivos para gestionar los grupos y gastos.

APIExpense :

En este archivo, encontramos una interfaz llamada ApiExpense, que define métodos para interactuar con los recursos relacionados con los gastos en la API. Por ejemplo, tiene métodos para obtener todos los gastos, obtener gastos por grupo y agregar un nuevo gasto.

Además, el objeto RetrofitExpenseServiceFactory proporciona un método estático getApiService() para inicializar y devolver una instancia de la interfaz ApiExpense utilizando Retrofit. Este objeto configura Retrofit con la URL base de la API y un convertidor Gson para serializar y deserializar objetos JSON.

APIGroups :
Este archivo es similar a ApiExpense, pero se enfoca en las operaciones relacionadas con los grupos en la API. Define métodos para obtener todos los grupos, obtener un grupo por su ID y agregar un nuevo grupo.

Json-server

En el contexto de una aplicación Android que se comunica con este servidor, las solicitudes HTTP se realizan a las URL proporcionadas por json-server para obtener y manipular los datos de grupos y gastos. Por ejemplo, al cargar la lista de grupos en la interfaz de usuario de la aplicación, se realizaría una solicitud GET a la URL correspondiente para recuperar los datos del archivo JSON. De manera similar, al agregar un nuevo gasto a un grupo, la aplicación enviaría una solicitud POST al servidor para registrar el nuevo gasto en el archivo JSON.

4. Conclusiones

La conclusión de este proyecto destaca el cumplimiento de los objetivos establecidos, evidenciando la creación exitosa de una aplicación móvil funcional para la gestión de gastos compartidos. A través del uso de tecnologías modernas como Android Studio, Retrofit y JSON Server, se logró desarrollar una plataforma intuitiva y eficiente que facilita la administración financiera entre grupos de usuarios. Este proceso no solo fortaleció el conocimiento técnico del equipo, sino que también resaltó la importancia de la colaboración y la planificación efectiva en el logro de metas comunes. Además, las lecciones aprendidas durante el proyecto sientan las bases para futuras mejoras y desarrollos, promoviendo una cultura de mejora continua y adaptación a los desafíos en el ámbito del desarrollo de software.