AMOV - TP 01

Trabalho Prático 01
Arquiteturas Móveis

2023/2024

Daniela Correia - a2021143404@isec.pt
João Neves - a2021133564@isec.pt
Tiago Cardoso - a2021138999@isec.pt

Instituto Superior de Engenharia
Politécnico de Coimbra

Índice

Descrição Sumária do Trabalho

Este trabalho consistiu em desenvolver uma aplicação Android, recorrendo ao Jetpack Compose, que disponibiliza informações sobre locais de interesse existentes em diversos locais. A aplicação recorre a geolocalização para guardar as coordenadas dos pontos de interesse, sendo que o utilizador pode utilizar as suas coordenadas atuais para criar um ponto de interesse nessa localização.
Os utilizadores podem, para além de adicionar locais, pontos de interesse e categorias, adicionar comentários/avaliações aos pontos de interesse que tenham já eventualmente visitado.
Qualquer adição/edição que um certo utilizador queira fazer é considerada como uma contribuição, sendo que essas contribuições apenas ficam disponíveis após serem aprovadas por dois utilizadores da aplicação.

Organização do Projeto

O trabalho foi separado nos seguintes packages:

- data (AppData)
- ui (+ MainActivity.kt)
    - composables (composables criados)
    - screens (+ MainScreen.kt e enum de Screens)
        - addview (views associadas)
        - detailview (views associadas)
        - evaluate (views associadas)
        - login_register (views associadas)
        - mapviews (views associadas)
        - searchview (views associadas)
    - theme (cores, tema e fonte)
    - viewmodels (+ AppViewModelFactory.kt)
        - location
- utils
    - file (FileUtils)
    - firebase (classes de FStorage + FAuth)
    - location (classes para gerir a localização)

\+ App.kt

Classes Utilizadas

Descrição das classes

Contribution

Esta open class tem como objetivo representar uma contribuição, associada ao email do autor da contribuição.

Local

Esta open class, que estende *Contribution*, tem como objetivo representar um local. Para além do email do autor (presente na class *Contribution*) possui também um id, nome, imagem associada (uri), coordenadas geográficas, e pode possuir os emails dos users que aprovaram, entre outros atributos. Além disso, possui funções utilizadas na aprovação do local. Esta class vai servir também para ser utilizada na listagem tanto da localização como dos pontos de interesse.

Location

Esta data class, que estende *Local*, representa uma localidade, por exemplo Porto, Lisboa, etc. A class possui os atributos de *Local*, dando override aos mesmos.

PlaceOfInterest

Esta data class, que estende *Local* pretende representar um ponto de interesse (que estará presente numa localidade). Para além de dar override aos atributos presentes na class *Local*, possui também o id da categoria e da localidade às quais pertence.

User

Esta data class representa um utilizador da aplicação, com username, email e eventual fotografia de perfil, guardada como um uri.

Category

Esta data class, que estende *Contribution*, representa uma categoria, à qual podem pertencer pontos de interesse. Para além de dar override ao atributo de *Contribution*, possui um id, nome, descrição e ícone representativo. O ícone será guardado como um uri.

Classification

Esta data class, que estende *Contribution*, tem como objetivo representar uma classificação individual de um ponto de interesse. Possui, além do email do autor, o id do ponto de interesse em questão, uma nota, um comentário e eventual imagem (como uri).

AppData

Esta class vai manter registo das localidades, pontos de interesse, categorias e classificações, para poderem ser utilizadas ao longo de toda a aplicação.

Implementação das funcionalidades

Autenticação/Registo

Para a autenticação/registo, recorremos ao serviço de autenticação disponível através do *Firebase*, de forma a guardarmos os utilizadores registados com passwords encriptadas, id gerado pelo *Firebase* etc. Isto foi feito com auxílio de uma classe *FAuthUtil* que utiliza *Firebase.auth* para criar um utilizador, fazer login e fazer logout.
Quanto à implementação *jetpack compose*, recorremos à criação de um formulário, onde:

- No caso do login, o utilizador pode inserir o seu email e password, que depois serão validados através da função *signInWithEmailAndPassword()* disponível no serviço de autenticação do Firebase
- No caso de registo, o utilizador insere os seus dados, como nome, email, password, que depois servirão para criar um utilizador no *Firebase*, através da função *createUserWithEmailAndPassword()* disponível no serviço de autenticação do *Firebase*

Base de dados

Para implementação da base de dados, recorremos ao *Firestore Database*, onde armazenámos os dados necessários, como as categorias, locais, pontos de interesse e classificações, bem como os atributos de cada uma das classes. Para isso recorremos às classes:

- FStorageAdd - Para adicionar dados
- FStorageEdit - Para criar o uri de uma imagem nova, que substitui uma que exista na base de dados
- FStorageObserver - Para notificar quando há alterações na base de dados
- FStorageOrder - Para fazer ordenar os itens através da base de dados
- FStorageRemove - Apagar dados existentes
- FStorageUpdate - Atualizar a base de dados

Cada uma destas classes tem um propósito diferente mas todas elas têm alguma interação com a base de dados criada.

Jetpack Compose

A aplicação foi desenvolvida inteiramente recorrendo ao *Jetpack Compose*, com a criação de *composables* como formulários, listas personalizadas para os locais de interesse e locais, *floating action buttons*, etc.

Estes *composables* vão sendo utilizados em cada um dos *screens* consoante a necessidade.

Geolocalização

Quanto à geolocalização, utilizámos o serviço de localização do Google Play Services.

Para isso, recorremos à seguinte interface e classe:

- Interface *LocationHandler* - define um conjunto de métodos e propriedades necessários para o controlo da obtenção de localização do dispositivo, incluindo a ativação/desativação das atualizações de localização e a definição de um callback para quando uma nova localização for recebida.
- Classe *FusedLocationHandler* - implementa a interface anterior e usa a API "Fused Location Provider" (FusedLocationProviderClient) do Google Play Services para gerir as atualizações de localização.
    - startLocationUpdates(): Este método é responsável por iniciar as atualizações de localização. Verifica se as atualizações já estão ativadas e, se não estiverem, configura os parâmetros necessários. Além disso, define um callback (locationCallback) que será chamado quando novas atualizações de localização estiverem disponíveis
    - stopLocationUpdates(): Este método para as atualizações de localização e remove o callback registrado (locationCallback) para deixar de receber as atualizações de localização
    - locationEnabled: Indica se as atualizações de localização estão ativadas
    - onLocation: Este é um callback opcional que será acionado quando uma nova localização estiver disponível. Recebe um objeto Location como parâmetro, que tem informações sobre a latitude, longitude e outras características da localização.