Technical Project Report - Android Module

GreenWalk

Course:	Introdução à Computação Móvel
Date:	Aveiro, <28/11/2020>
Authors:	92926: Pedro Marques 83069: Eleandro Laureano
Project abstract:	<Uma aplicação que promove a mobilidade amiga do ambiente. A aplicação permite registar percursos realizados e partilhá-los com outros users. Fornece também estatísticas sobre a qualidade do ar num percurso bem como no momento>

1 Application Concept

GreenWalk baseia-se numa aplicação já existente designada Ciclogreen. Esta última permite registar percursos realizados pelos seus utilizadores e incentiva-os através da oferta de recompensas, desde descontos em cafés até bilhetes de cinema/teatro ou shows musicais. Como o principal objetivo da app Ciclogreen é diminuir as emissões de CO2, os utilizadores são apenas recompensados se preferirem andar a pé, bicileta ou até de metro ou autocarro em contraste com andar de carro. As pessoas são recompensadas com 'coins' que depois podem trocar pelas recompensas préviamente descritas. A nossa aplicação, GreenWalk, pretende igualmente promover a mobilidade sustentável através da partilha dos percursos realizados. Os utilizadores podem igualmente escolher o método de deslocamento, gravar uma atividade e escolher mantê-la privada ou torná-la pública. Igualmente, os criadores de uma atividade podem adicionar fotos associadas ao percurso e vêm associadas estatísticas sobre a qualidade do ar.

Igualmente, esta aplicação poderia ser usada por qualquer um que estivesse interessado quer em turismo, ajudar ou ambiente ou apenas em obter as recompensas. Assim sendo, abrangimos uma grande audiência. Contribui ainda para a saúde dos utilizadores que têm acesso ao índice da qualidade do ar para a sua localização, atualizada a hora. Estes dados que fornecemos são provenientes duma API.

A nossa aplicação promove então o exercício físico, a saúde dos utilizadores bem como ajuda a diminuir as emissões de dióxido de carbono e outros poluentes provenientes dos veículos. Pode ainda ajudar o turismo local e promover os utilizadores a visitarem certos locais através da dispersão de imagens.

2 Implemented solution

Architectural overview

Firebase Datasets

Os dados desta aplicação são todos guardados na Firebase database. No Firebase, temos duas tabelas difrentes na Realtime Database:

- "Users" – como ID de cada entrada utilizamos o email do utilizador logado. Os atributos de cada entrada são o username e o género do utilizador. O género é utilizado para calcular a distância percorrida a pé e é um dado obrigatório. O username é igual ao da conta Google associada.
- "Paths" – como ID obtemos uma combinação automaticamente gerada de numeros e letras (ObjectID) . Esta tem vários atributos:
    - Activity: lera indicador do tipo de atividade ("E" – eletric Bicycle, "W"-walking, "R" – running, "B" – regular bicycle)
    - airData – objeto JSON; lista de resultados da chamada à API utilizada para obter a qualidade do ar.
    - Avg_speed – velocidade média no percurso em km/h
    - Date –Timestamp/Data e hora de início da atividade
    - Distance – distancia percorrida em km
    - Private – define se a atividade é privada ou pode ser pública
    - Route_points – objeto JSON; lista de objetos LatLng referentes à localização do utilizador
    - Steps – número de passos dados
    - user – email (ID) do utilizador que realizou a atividade.
    - (Opcional) images – lista de ObjectID cujo valor corresponde ao nome de imagens associadas a uma atividade. Estas imagens são guardadas na Storage do firebase.

RoomDB

Igualmente, utilizamos a RoomDatabase como uma espécie de cache que guarda algumas atividades realizadas num dispositivo, estas entidades são designadas de "Path". Embora cada entrada na RoomDB seja uma atividade com quase os mesmos atributos que o dataset "Paths" representado na Firebase Realtime Database, as imagens de uma atividade não são atributos da entidade "Path" pelo que para as obter é sempre necessária uma chamada à firebase database.

A RoomDatabase suporta as operações de insert(), clear() e getAllRoutes(). As duas primeiras operações inserem novos dados no repositório e eliminam todos os dados do mesmo, respetivcamente. Por outro lado, a última operação retorna todas as entradas da entidade "Path" no repositório. As classes utilizadas para implementar o módulo RoomDB estão devidamente identificadas e organizadas no diretório 'database'.

Retrofit

Como já foi explicado antes, utilizámos uma API que nos iria buscar a qualidade do ar para um determinado par de coordenadas. A API WeatherBit.io permite vários modos de utilização, quer seja para ir buscar a meteorologia atual, qualidade do ar ou o histórico de ambas as opções numa dada zona. Assim sendo, só a usamos para obter a qualidade do ar no momento. Como parametros, fornecemos a latitude e longitude, bem como uma key que nos é fornecida ao registarmo-nos para a API.

Assim sendo, através do módulo retrofit conseguimos fazer uma chamada à API, obter os dados em formato JSON e convertê-los num objeto AirCheckr, cuja classe está no diretório 'retrofit2' do projeto.

Services

No diretório 'services' existem 3 classes java que extendem a componente Service. Todos os servicos seguem uma estrutura simples para inicialização e finalização. O intent que faz a chamada ao serviço pode ter as duas funções: finalizar ou inicializar. Os intents têm actions associadas e caso estas possuam o padrão "STOP SERVICE" ou "START SERVICE", as devidas ações ocorrem (finalização e inicialização, respetivamente). Para retornarem a informarção que determinaram, os Serviços utilizam um LocalBroadcastManager para realizar um broadcast de um intent que possui essa informação. Este broadcast é capturado pelos BroadcastReceivers que capturam o intent e utilizam as informações do bundle associado.

1. LocationService
    a. Fundamental para a aplicação, é inicializado logo após o Login. Inicializa um NOTIFICATION_SERVICE que indica ao utilizador quando a aplicação está decorrer e a utilizar a localização. Regista a localização atual utilizando um fusedLocationProvider e captura também a velocidade, através do método Location.getSpeed(). Esta velocidade capturada, sendo menos fiável, só será usada quando a atividade que for gravada não seja do tipo "Walk" ou "Run". Estas têm um tratamento diferente explicado no SensorService.
    b. Broadcasted Intent:
        i. Action: "NewLocation"
        ii. Bundle Fields:
            1. "Lat" – latitude da localização detetada
            2. "Lng" - longitude da localização detetada
            3. "Speed" – velocidade capturada numa localização utilizando o método getSpeed()

2. APIService
    a. Utilizado para fazer as chamadas à API periodicamente (período de 30 segundos). Possui um broadcast receiver que captura intents com action "newRoutePoint". Estes intents são broadcasted pela MainActivity que guarda a informação recebida pelo LocationService, atualiza a variavel mCurrentLocation e envia a latitude e longitude deste objeto (bundle fields "Lat" e "Lng") para o APIService. No entanto,

este receiver só é registado após inicialização do serviço, pelo que a localização inicial é enviada no intent que inicia o APIService. Ste receiver e o intent inicial atualização o objeto LatLng mCurrentLocation. De seguida, um timer executa o método getAirQuality() a cada 30 segundos utilizando o método com o mesmo nome em JsonPlaceholderAPI (no diretório 'retrofit') para fazer a call à API e transformar o objeto JSON recebido em AirCheckr. Este obejto é novamente transformado em JSON para ser enviado como String num broadcast de um intent que também será recebido pela MainActivity

b. Broadcasted Intent:
    i. Action: "newAQIData"
    ii. Bundle Fields:
        1. "AirQuality" – objeto AirCheckr em JSON

3. SensorService
    a. O Sensor Service é inicializado quando se começa a gravar uma atividade do tipo "R" ("Run") ou "W" ("Walk"). Este serviço obtem permissões para ACTIVITY_RECOGNITION mal se abre a página para gravar o percurso. Utilizando o sensor virtual STEP_COUNTER, são capturados os passos dados e calculada uma aproximação da distância percorrida, sabendo que, em média, a passada de uma mulher é de cerca de 70 cm e de um homem de 78cm. O género do utilizador é passado no intent que inicializa o serviço, no field com nome "Gender". Um timer é inicializado que ocorre a cada segundo, marcando assim o tempo decorrido. É enviado um broadcast a cada segundo com os campos calculados a partir da variável *newStepCounter* que garda o número de passos dados.
    b. Broadcasted intent:
        i. Action: "sensorMovement"
        ii. Bundle Fields:
            1. "Steps" – valor do objeto *newStepCounter*
            2. "Elapsed Time" – valor incrementado a cada execução do timer
            3. "Distance" – calculado matematicamente:
                a. Distance = newStepCounter * distancia_passo

ViewModel

Nesta aplicação recorremos a ViewModels para guardar dados que eram essenciais para o bom funcionamento da App.

MainViewModel

ViewModel comum a todas as classes. Contém atributos partilhados por todos os fragmentos e em toda a aplicação. Inclui:

- "currentSpeed" – Velocidade atual obtida pelo LocationService quando atividade é do tipo "E" ou "B"
- "mCurrentLocation" – Localização atual obtida pelo LocationService
- "mCurrentActivityRoute" – Pontos LatLng da atividade a ser gravada
- "airData" – informação sobre a qualidade do ar em mCurrentLocation
- "mCurrentUserEmail" - Email do utilizador atual
- "mGender" – género do utilizador
- "mRecording" – boolean que identifica caso estejamos a gravar uma atividade

ActivityDetailsViewModel

Guarda dados sobre a atividade a ser visualizada no momento.

- "mCurrentPath" – objeto Path a ser visualizado
- "mImages" - Lista de URI de images associadas à atividade
- "uri_name_images" – Associação entre URI e nome das Images
- "isPrivate" – boolean que indica se uma atividade é privada ou não

MapsFragmentViewModel

- Guarda dados relativos a uma atividade que está a ser gravada no momento.
  - "routePoints" – lista de objetos LatLng que representam as difrentes localizações ao longo do percurso
  - "steps" – número de passos dado, fornecido pelo SensorService
  - "airData" – lista de dados relatiovos à qualidade do ar num percurso, fornecido pelo APIService
  - "speed_history" – lista de todas as velocidades registadas pelo método Location.getSpeed() no serviço LocationService. Só regista dados se a atividade for do tipo "E" ou "B"
  - "time" – tempo decorrido em segundos, fornecido por SensorService
  - "avg_speed" – velocidade média até agora em km/h
  - "distance" – distancia percorrida em km
  - "activity_type" – tipo de atividade ("E", "B", "W", "R")
  - "date" – timestamp de início da atividade
  - "priv" – boolean que indica se a atividade é privada ou não
  - "mCurrentLocation" – localização atual

PathViewModel

Utilizado para obter os dados do repositório local implementado utilizando RoomDB. Possui as funções suportadas pelo DAO: insert(Path p), clear() e getAllRoutes() já explicadas anteriormente.

Jetpack

Utilizámos o jetpack navigation componet para estabelecer a navegação na nossa aplicação. Assim sendo falaremos melhor disto no tópico seguinte em que explicaremos a UI bem como as

interações implementadas. Como já falamos anteriormente, utilizámos o MainViewModel para guardar informações comuns a toda a aplicação. Assim sendo, este view model foi fundamental para manter o estado de fragmentos e podermos navegar para outros enquanto que eles permanecem em execução.

Implemented interactions

Activities

SignInActivity

Realiza uma conexão ao Firebase e permite fazer login utilizando os serviços Google.

SettingsActivity (PreferencesActivity)

Permite alterar o tema da app (Modo Noturno) e a orientação da mesma.

Figura 2 - SignIn Activity

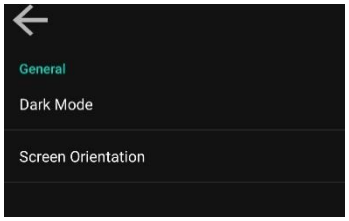

Figura 1 - SettingsActivity

MainActivity

Atividade Principal, responsável pela inicialização da aplicação. A partir desta activity, o utilizador é enchaminhado para a SignInActivity ou, caso já o tenha feito logIn, pode navegar livremente pela app.

A MainActivity é reponsável fornece o layout geral da aplicação e um fragmento que será utilizado pelo JetpackNavigation para mostrar as diferentes abas . Inicializa uma bottomnavigationview para permitir a navegação.

Figura 3 - BottomNavigationView - Maps, History, Home, Profile

Os fragmentos que são possíveis de aceder a partir da bottomnavigationview são os seguintes:

1. MapsFragment:
    a. Fragmento com os dados sobre uma atividade a ser gravada. Mostra a qualidade do ar no momento, velocidade média, distância, tempo decorrido e passos caso a atividade englobe o ato de andar. Possui também um switch que permite alterar a privacidade da atividade. Responsável por iniciar um SensorService. Contém uma mapView com uma representação do percurso atualizada a cada alteração de localização.

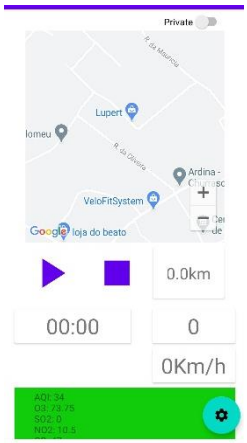

Figura 4 - Maps Fragment

2. HomeFragment & HistoryFragment
    a. Fragmentos muito semelhantes. Ambos com uma recyclerView com atividades. O Historyfragment só contem as atividades de um utilizador enquanto que o HomeFragment contém todas as atividades públicas. Cada item das recyclerViews tem um ícone que representa o tipo de atividade, o nome do user e data, distância percorrida, tempo decorrido, passos, velocidade e air quality index (AQI). O botão no canto inferior direito do HistoryFragment permite eliminar os dados na RoomDB.
    A cor dos itens depende do AQI de acordo com a norma universal.

Figura 5 - HomeFragment

Figura 6 - HistoryFragment

Ao carregar em cada item,

navegamos para o fragmento:

- ActivityDetails
  - Contém os detalhes de uma atividade, incluindo uma representação no mapa, velocidade, tempo decorrido, distância e passos dados. Contém também uma recycler view de imagens que, ao serem clicadas, aparecem com maior dimensão à direita da mesma.
  - Permite ao dono da atividade fazer upload de fotos que capturou no percurso ou eliminá-las.

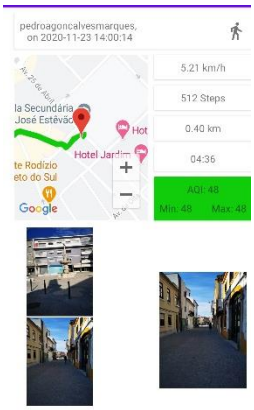

Figura 7 - ActivityDetails

3. ProfileFragment
   a. Fragmento com dados sobre o user. Inclui imagem, nome, distância total percorrida, género e qualidade do ar no momento, bem como recomendações. Vai buscar os dados a firebase database.

Figura 8 - ProfileFragment

Dialog Fragment

Existem dois dialog fragments:

GenderDialog: Permite, caso ainda não tenha sido escollhido, definir o género de um utilizador para cálculos. Inicia após o SignIn.

Figura 9 - GenderDialog

ActivityTypeDialog: Permite determinar o tipo de atividade a gravar. Abre quando carrgamos no floating action button em MapsFragment.

Figura 10 - ActivityTypeDialog

Limitations

Embora a aplicação tenha várias funcionalidades, não conseguimos implementar algumas que seriam do nosso agrado. Devido ao pouco tempo que tivemos disponível, tivemos que prioritizar certas funcionalidades sobre outras. Infelizmente, não chegámos a colocar uma aba de recompensas na qual os utilizadores trocassem pontos acumulados pelas mesmas. Por outro lado, desde a apresentação implementá-mos uma preferência que permite alterar entre Dark mode e Light Mode, escolher a orientação do ecra e partilhar a aplicação. Infelizmente não nos conseguimos focar em aspetos como o design.

Outro detalhe é o facto de, ao alterarmos a orientação, a aplicação voltar à página inicial, o fragmento **Home**.

3 References and resources

Key project resources

- **Code repository:** https://github.com/PedroMarques27/ICM_Ciclogreen
- APK: https://github.com/PedroMarques27/ICM_Ciclogreen/blob/main/GreenWalk.apk

Reference materials

Retrofit2 https://square.github.io/retrofit/

Firebase https://firebase.google.com/docs/android/setup

RoomDB and ViewModels: https://developer.android.com/codelabs/android-room-with-a-view#11

Android JetPack Navigation https://developer.android.com/guide/navigation

Other Codelabs Resources: https://developer.android.com/codelab

API Weatherbit: https://www.weatherbit.io