Kotlin Android Course Project 5

Moon Store - Custom Application

The idea behind the project was to build an online platform to support physical stores, were the users can consult the existing items available in the platform and mark them as favorites, so that later, when they pass by one of the company's physical stores, they were able to be notified that their favorites are in stock in a store nearby. This way, users don't need to go to the stores and search for the items, they know beforehand if their item is available in that store they are passing by. It would also be good for sales since a user may have not even go in and buy if there wasn't for the notification reminding them.

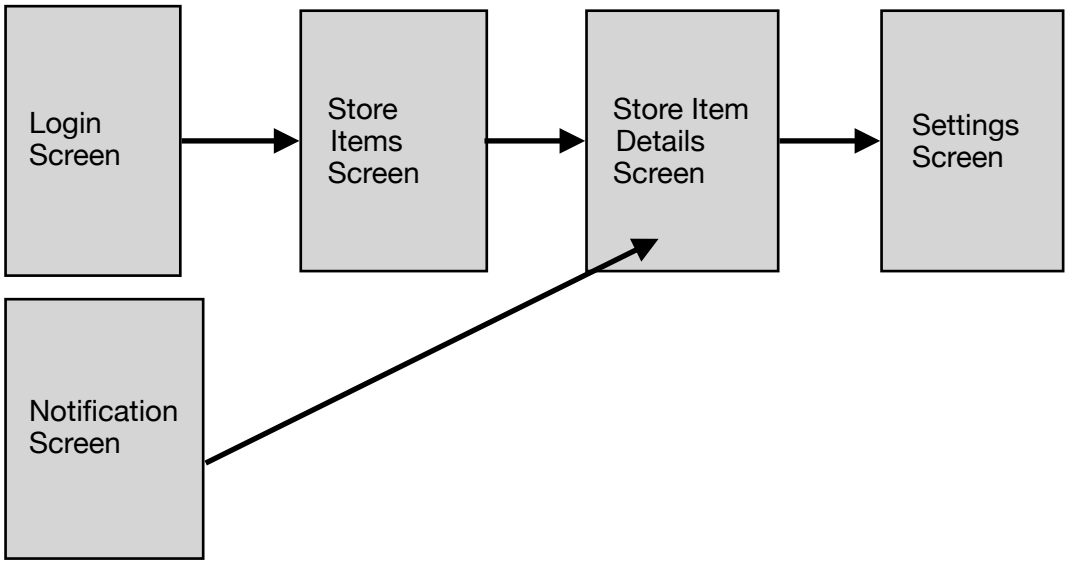
Building an API

It was necessary to build an API for the store information using Google Cloud Platform Tools. For this, the following steps were taken:

1. Create a Project for Moon Store in Google Cloud Platform
2. Create a Cloud Run instance of the store
3. Create locally a custom java project using 'Cloud Run' plugin for Intellij IDEA and build a basic api that retrieves store information. The available requests for this API are the following:
   - '/pieces' - Gets all the store items with the correspondent information
   - '/stores' - Gets all the existing stores
   - '/stores/stock/{piece_id}' - Gets all the stores that have available stock for the specified item (piece_id)
   - '/stores/{store_id}/stock/{piece_id}' - Returns either true or false, depending on if the store specified by store_id has or not in stock the piece specified by piece_id
4. Deploy local project to Google Cloud Platform API instance using plugin tools
5. Create an 'API Gateway' instance in Google Cloud Platform in order to communicate with API using an API key to make requests safer.
6. Create Cloud Storage bucket to hold the images
7. Start using API in project.

Project Overview

The general idea for the app, is to build the following structure:



Login Screen - Screen where the user can login using googles firebase auth

Store Items Screen - Screen where the user can see a list of all store items and add them to favorite. If one item image is pressed, it should navigate to storeItemDetails.
This screen also should have a menu to navigate to settings screen and to filter items to only see favorites. For first time usage, this screen should display a popup to activate stock notifications

Store Item Details Screen - Screen where the user can see all the information of one specific store item. User can also favorite this item in the screen.

Settings Screen - Screen where the user can turn on and off the stock notifications. It also should have a brief explanation on how they work.

Detailed Project Code Overview

API
This package contains the retrofit service that ables the app to communicate with the created API. All of the api existing requests are mapped in the “StoreApService” interface that will be used in the Repository layer to perform requests.

Authentication
The authentication package contains the authentication activity and fragment that allow the user to login into the app with the support of firebase google platform, where both gmail and regular email logins/registrations are allowed into the app.

Data
The goal of this module is to hold all data related functionalities, namely android Room related operations and a repository layer to serve as a data source interface between fragments and room+api data.
This package contains several layers such as: dao, dto and local.
The dao module contains the online store dao interface, that allows the operations within the apps room database to take place. This includes inserting the items and stores to our local database, saving the favorite status for each item, and several others.
The dto module contains both entity classes of the objects stored into our room database, StoreItemDto, that represents an item on the store and the current favorite status for that user, and PhysicalStoreDto, that represents an actual store and its location
Finally, the local module, contains the actual database initialization, along with the repository implementation, the necessary layer so that fragments can have access and change local/api data.

Store Items
This package contains all store list related elements. It is composed by the fragments for both store item list and store item details and the main flow activity, store activity.
The list fragment contains the app’s main page, and it shows all the store items that were retrieved from the API previously and stored into the apps room database. Within this page, we are able to add each item to our favorites by clicking the heart icon, navigate to the details of a specific item by clicking on the item’s image, or navigate to settings using the menu. It is also possible to filter the list so that we can only see the items we marked as favorites, instead of all of them. For a first time user, this is also where we will prompt the user with the possibility of being

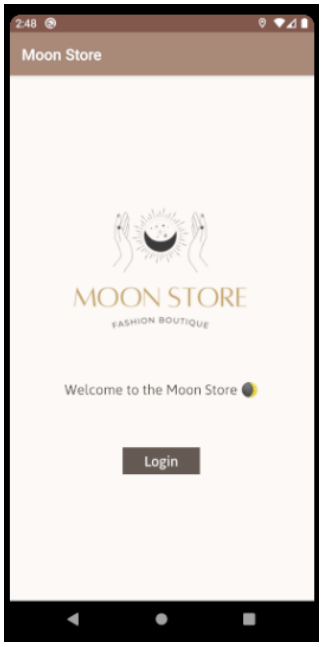
notified if he passes a store that has one of his favorite items in stock, and save his preference. He can then change this setting later in the settings fragmeent.
The details fragment appears when the user, through the store list fragment or notification fragment, clicks on the image of a specific element of the list. This fragment contains a more detailed view of the store item, including the image, favorite icon (in which they can also click and add the item to favorites like the previous screen) title, price, and stores that have the item in stock. To go back to the list, the user just needs to press the back button.

Settings
The settings fragment is accessible through the store items list fragment, by clicking on the side menu -> settings. This fragment allows the user to enable/disable the stock pass-by notifications. This fragment also contains a detailed explanation of how this notifications work, so that the user can be aware of the way it functions, together with a descriptive image.

Notification
The notifications package contains all elements related to the notification flow. The geofence module contains a broadcast receiver, that is triggered when the user enters a geofence (meaning a physical store area), and enqueues work on the GeofenceJobIntentService, that checks what of the users's favorite items are available at that geofence store, and sends a notification to the user informing the availability of said items.
When the user clicks on the notification, a new flow starts, represented in the nav_graph_notification, starting the ItemsInStockNotificationActivity. This activity is responsible for the retrieval of the elements sent in the intent (the items in stock and the store) and starts the StockNotificationFragment, that shows a screen that contains the store and a list of the elements in stock, where the user can click any element and navigate to the details screen of that store item.

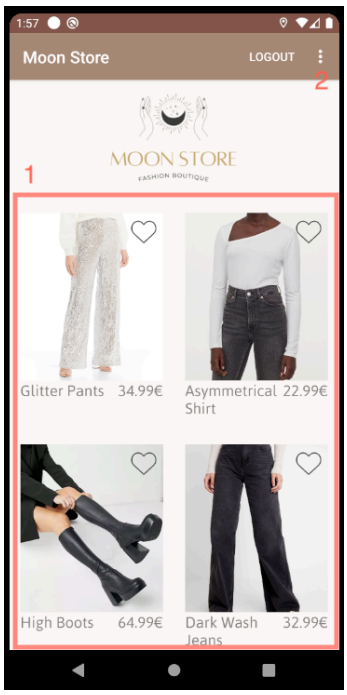
StoreActivity and StoreViewModel
These elements are the core of the main flow of the app. The StoreActivity is the activity for the nav_graph, and is responsible for creating the geofences for the stores, checking and requesting for the location permissions to the user, and also keeping track of the available physical stores. The view model serves as support for the activity and some other fragments.

Screen UI Overview



Login Screen

This is the screen where the user is able to login, the first screen of the app if the user is not authenticated.
This screen contains the store logo, a welcome text and a Login button that starts the firebase login process



Store List Screen

This Screen is the main app screen and contains the list of all items.

1- List of all store items retrieved from API

2-Side menu for settings and List filtering + Logout button

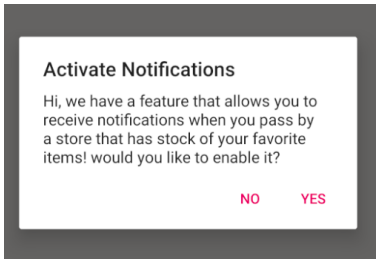
Store List Item

Each item on the list contains 4 elements:
The Image of the item, that when clicked, navigates to the details screen.

1- The favorite icon that, when clicked, adds the item to favorites

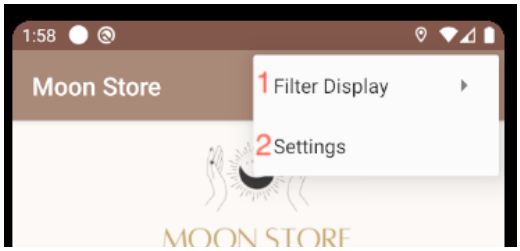
2- Items title

3- Items price



Activate Notifications Alert Dialog

This is an alert dialog that appears the first time the user is opening the app, suggesting the user to turn on his notifications for the stock. If he selects ‘yes’, geofences will be created.

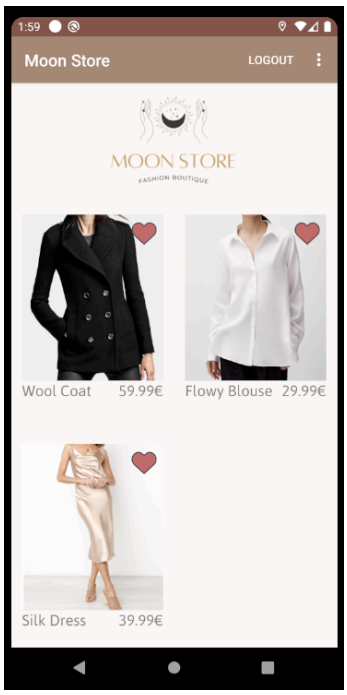


Side Menu

This menu allows the user to access both settings and filter the store items list

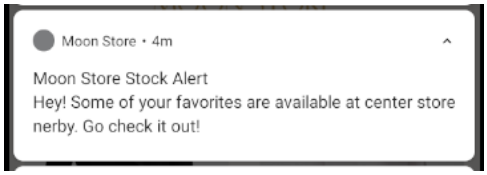
1- This menu item allows the user to filter the list by: favorites, all items

2- This menu item navigates to the settings screen



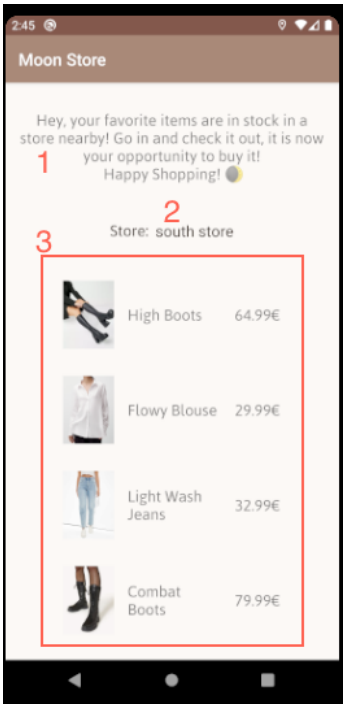
Store List Screen Filtered by Favorites

This image shows the Store List screen filtered by favorite items



Stock Notification

This notification shows up when the user passes by a store with his favorite items available



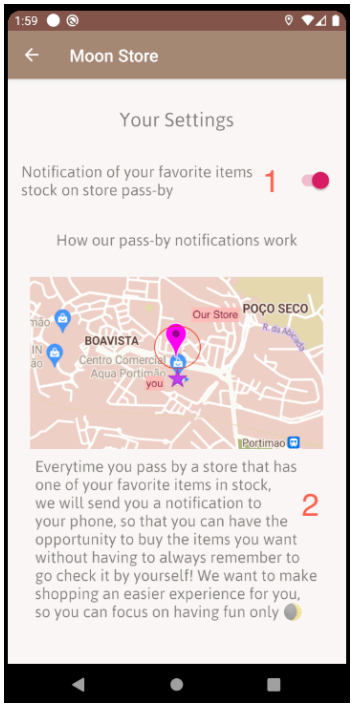
Notification Screen

This screen appears once the user clicks on the notification, informing the user of the items that are in stock in the store nearby.

1- Informative text about the notification

2- The Store nearby

3- The list of available items in the respective store (clickable and will navigate to the details screen)



Settings Screen

This screen appears once the user clicks on the settings in the side menu from the store list screen. This screen allows the user to turn on/off the stock notifications.

1- Toggle that allows the user to turn on/off the notifications

2- Long informative text about the how the notifications work

---

Criteria overview

1. Build a navigable interface consisting of multiple screens of functionality and data
   - This app contains a total of four screens and three naviagation graphs
   - Navigation controller is used for navigation of fragments and intents are used for activity navigation
   - Bundles and navargs are used throughout the app

2. Construct interfaces that adhere to Android standards and display appropriately on screens of different size and resolution.
   - Constraint layouts are used and the layout component hierarchy is as flat as possible
   - Res directory is used to store string values, drawables, dimens, etc.
   - Recycler views and adapters are used to display and handle the lists (store list screen and notification screen)

3. Animate UI components to better utilize screen real estate and create engaging content.
   - MotionLayout is used for the notification screen
   - Animator is used in several screens for the heart icon on the store item, when clicked

4. Connect to and consume data from a remote data source such as a RESTful API.
   - Retrofit is used to connect to the created api
   - Retrieved data is then handled with appropriate objects
   - Moshi library is used together with retrofit
   - All requests are performed asynchronously

5. Load network resources, such as Bitmap Images, dynamically and on-demand.
   - Glide library is used to load the images through the retrieved URL from API
   - When images are being loaded or there is an error loading, appropriate placeholders are shown

6. Store data locally on the device for use between application sessions and/or offline use.
   - Data retrieved from api is properly stored in a room database
   - Shared preferences are used to store the stock notification preference (on/off)
   - Data storage uses appropriate threads

7. Architect application functionality using MVVM.
   - Fragments and activities control the views
   - Models contain the data structures
   - View models contain the business logic

8. Implement logic to handle and respond to hardware and system events that impact the Android Lifecycle.
   - Bundles are used for data
   - External interactions (for example notification click) are handled by intents
   - Android permissions are requested

9. Utilize system hardware to provide the user with advanced functionality and features.
   - Location and Notification are used on the app
   - Permissions are requested to the user
   - Behaviors are accessed only after permissions are granted.