Mobile App Development

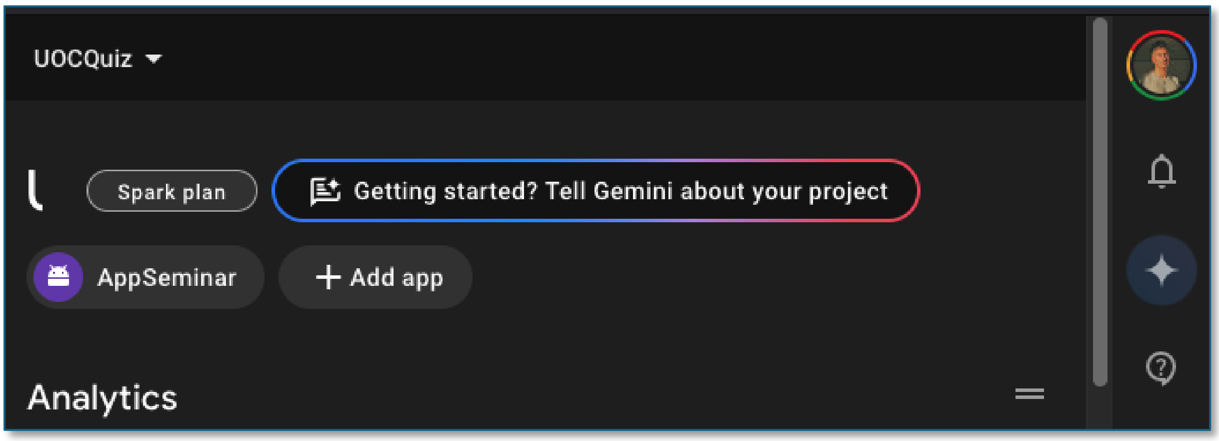
Continuous assessment Test 4 - CAT4

Nicolas D'Alessandro Calderon

Problem statement

1. Firebase environment preparation

In this first step we successfully created and connect the Firebase environment with our app:

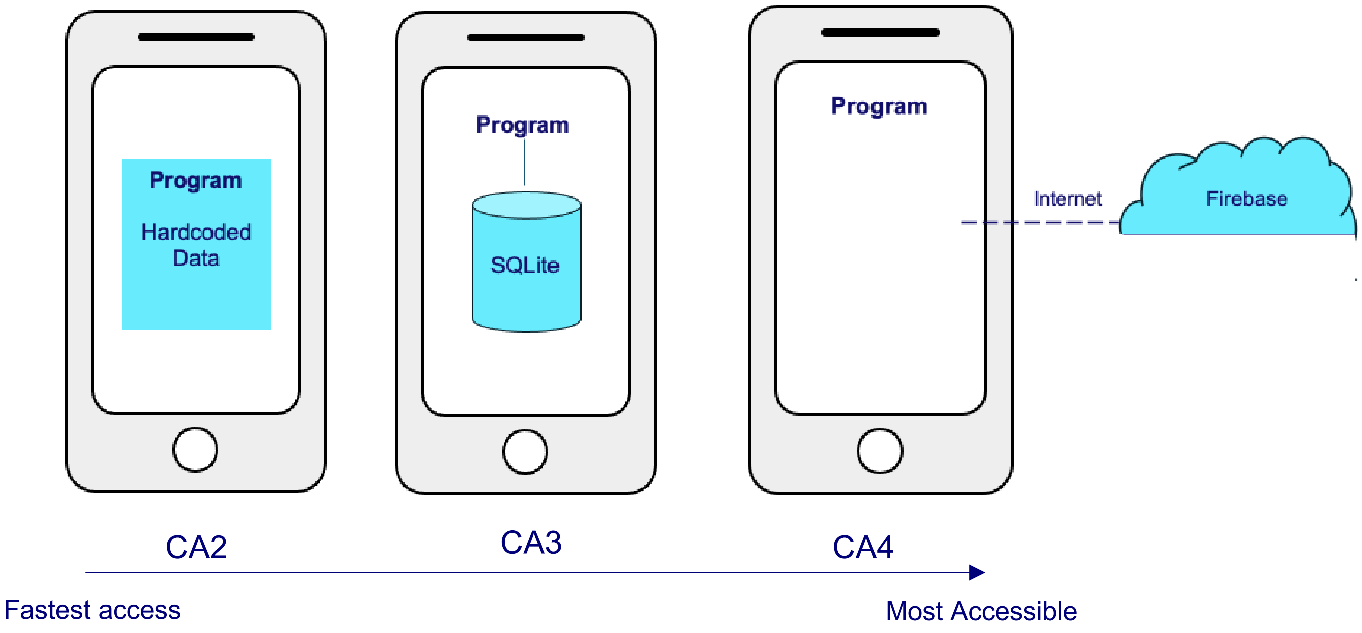


2. Back-end programming (theoretical questions)

2a) (THEORY) Answers:

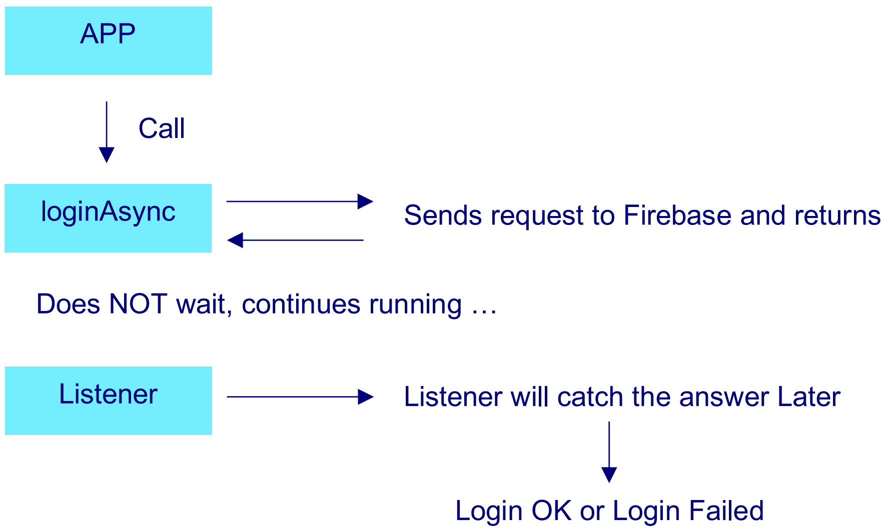
In CA2 the data where hardcoded in the code. In CAT3, we created a local database using SQLite in the same mobile. In this CAT4 the data is in the cloud in Firebase which is a BaaS – Backend as a Service platform.

- Accessing to the hardcoded data is very fast, simple to implement and no internet needed but is hard to update, the data is not persistent and is not scalable for many users.
- Having a local database offers a structured storage inside the app with data persisting between sessions and not internet is required. However, the data is still local, and the storage will be limited to the device.
- Having a Firebase connection, we now depend on the speed of the Internet connection that may result in slower data access, but we will have data available everywhere, real-time updates and user data sharing.



2b) (THEORY) Answers:

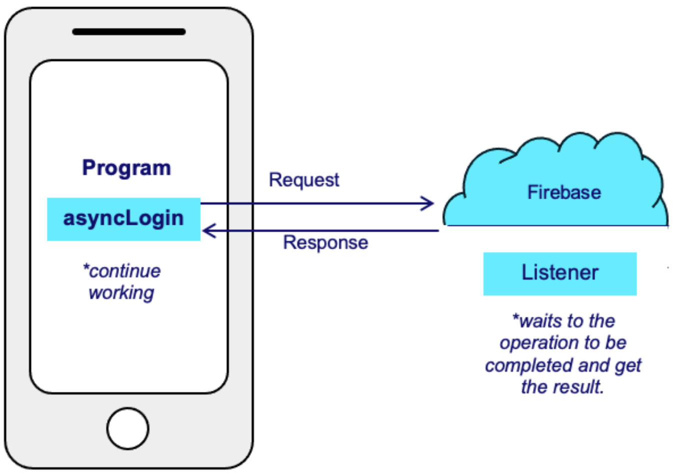
When the **loginAsync** ends, it does not know if the login attempt was correct or no. This is because Firebase use asynchronous operations, which means that it only initiates the authentication process, but it returns before receiving the answer. In asynchronous programming, the method initiates operations and return immediately, allowing other task to continue. They don´t wait the operations to be completed, so they don´t know the result of the operation.
In our case, the login result will be available when Firebase complete the operation and call the listener that has been passed as parameter.


APP
Call
loginAsync
Sends request to Firebase and returns
Does NOT wait, continues running …
Listener
Listener will catch the answer Later
Login OK or Login Failed

2c) (THEORY) Answers:

The listener that is being passed as parameter to the **loginAsync** method is acting as a messenger or callback that wait for the Firebase answer when completing the authentication operation. It has a very important role in this asynchronous operation:

- It will allow to receive the notification of operation completed,
- Obtain the result of the operation (success or failure)
- Handle the answer, updating the UI accordingly.
- Allows the app to be responsive while the network operations occur.



3. Loading a user's seminars

- BEGIN/END-CODE-UOC-3.1: File **DataSourceFirebase.kt** - **readSeminarsUserIdsAsync()**, query **user_seminar** by ID and collect the **sem_id** into a List of Long.

- BEGIN/END-CODE-UOC-3.2: File **DataSourceFirebase.kt** - **selectSeminarsUserAsync()**, query **seminar** collection using **.whereIn()** with the seminar ID, and create a **Seminary** object that update **seminarsLiveData.**

4. Creating item table

- BEGIN/END-CODE-UOC-4.3: File **DataSourceFirebase.kt** - **selectItemsSeminary()**, query **item** collection by **item_sem_id** then create **Item** object and populate the **seminarItemList.**

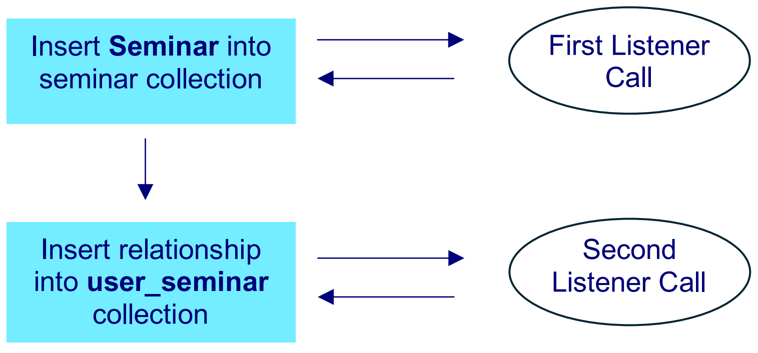
5. Inserting new Quiz

- BEGIN/END-CODE-UOC-5.1: File **DataSourceFirebase.kt** - **getNewSeminarId()**, created a method to find the seminar with the greatest id and then add 1 to it.

- BEGIN/END-CODE-UOC-5.2: File **DataSourceFirebase.kt** - **addSeminarAsync()**, add the new **Seminar** into the collection (HashMap). Then, added a new record into the **user_seminar** collection (HashMap2) to link the seminar with the user currently logged. Finally, the UI is updated to show the new seminar.

5d) (THEORY) Answers:

To perform these operations, we are using **two listener calls**, one when we add the seminar and one more listener when we link the seminar to the user.


Insert **Seminar** into seminar collection
First Listener Call
Insert relationship into **user_seminar** collection
Second Listener Call

6. Animation

- BEGIN/END-CODE-UOC-6.1: File **MainActivity.kt - Animate()**, implement the **Animate()** function that will create an animation moving the **ImageView** from the bottom to the top of the screen.

- BEGIN/END-CODE-UOC-6.2: File **MainActivity.kt - ShowAlertResult(),** modified the function to execute the Animate when the result window showing all answers correct is closed.