Last edited by **Luis Eduardo Perez** 7 months ago

Sprint 3 – Deliverable

Implemented features

- Smart Feature: A movie recommendation system has been partially implemented. This smart feature takes into account the movies that a specific user likes. This is visible by clicking the random button on the navbar.
- Authentication: The authentication system has been implemented using Firebase. The app checks if a user is logged in the session. If not, it will show the login and register screen. Otherwise, the home screen will be shown.
- External Service: The app communicates with firebase functions services. The storage is used to save lists, users, and likes. On the other hand, the functions are used to integrate the data stored in firebase storage with the recommendation system, which is hosted in an AWS EC2.
- Context-Aware: The context-aware feature has not been implemented yet, but the team had in mind to ask the user how he was feeling and, depending on the answer, the app would show certain movies and not others.

Business Questions Sprint 2

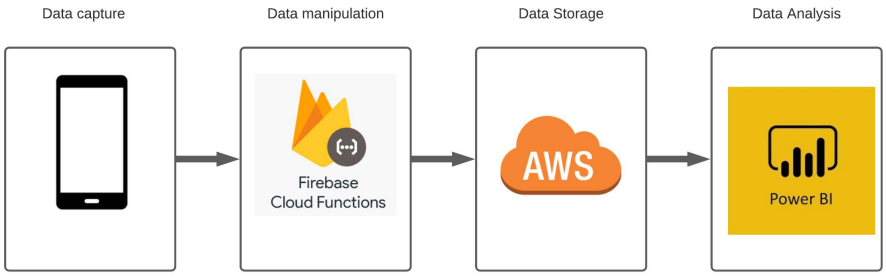
1. What is the percentage of the recommended movies liked by the user? [Type 3] (Juan Camilo Villamarin)
2. How long do users stay on the app?[Type 4] (Julian Oliveros Forero)
3. What is the distribution of genres liked by the user ? This will be presented on a Pie chart [type 2] (Luis Perez)
4. What are the top 5 genres mostly liked by the app's users? [Type 4] (Sebastian Baquero)
5. How many suggestions does the user pass before the first like on average? [type 3] (Kevin Blanco)
6. How much time does the system take to generate a recommendation, This will be presented in a scale by time [type 1] (Tobia Gasparoni)

Business Questions Sprint 3

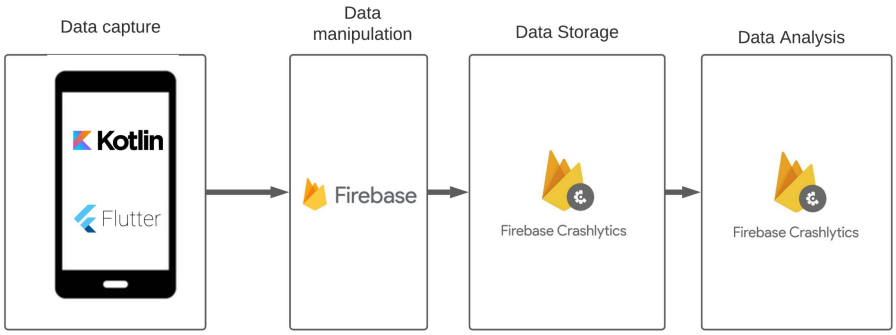
1. Which is the most used feature by the users? [Type 3] (Juan Camilo Villamarin)
2. Which part of the code where the app constantly crashes? [Type 1] (Julian Oliveros Forero)
3. What is the distribution of genres liked by the user ? This will be presented on a Pie chart [type 2] (Luis Perez)
4. What are the top 5 genres of the most liked movies on the app? [Type 4] (Sebastian Baquero)
5. How many suggestions does the user pass before the first like on average? [type 3] (Kevin Blanco)
6. How much time does the system take to generate a recommendation, This will be presented in a scale by time [type 1] (Tobia Gasparoni)

Analytics pipeline

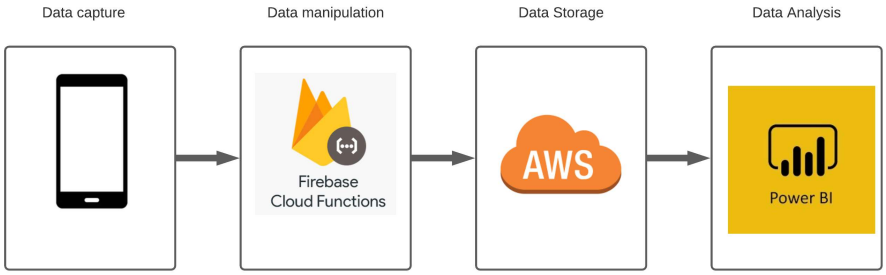
1. Which is the most used feature by the users? [Type 3] (Juan Camilo Villamarin)



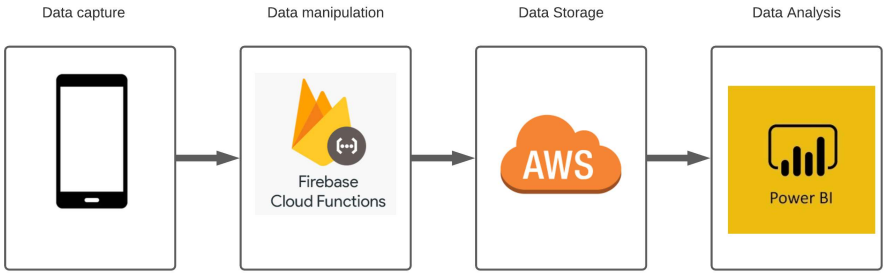
2. Which part of the code where the app constantly crashes? [Type 1] (Julian Oliveros Forero)



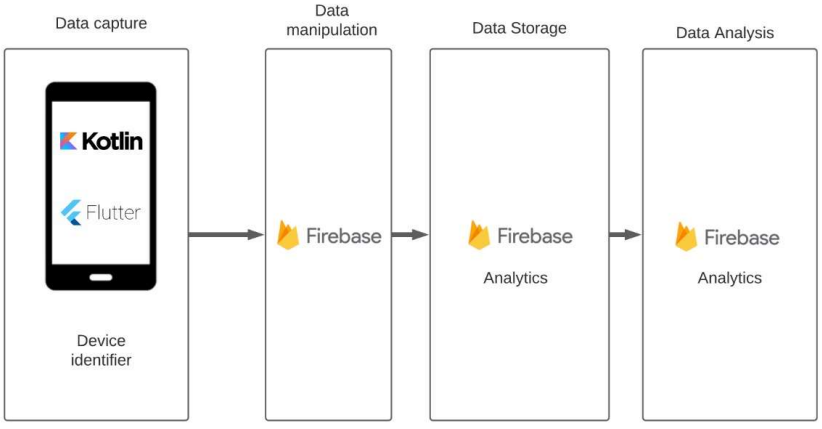
3. What is the distribution of genres liked by the user ? This will be presented on a Pie chart [type 2] (Luis Perez)



4. What are the top 5 genres of the most liked movies on the app? [Type 4] (Sebastian Baquero)



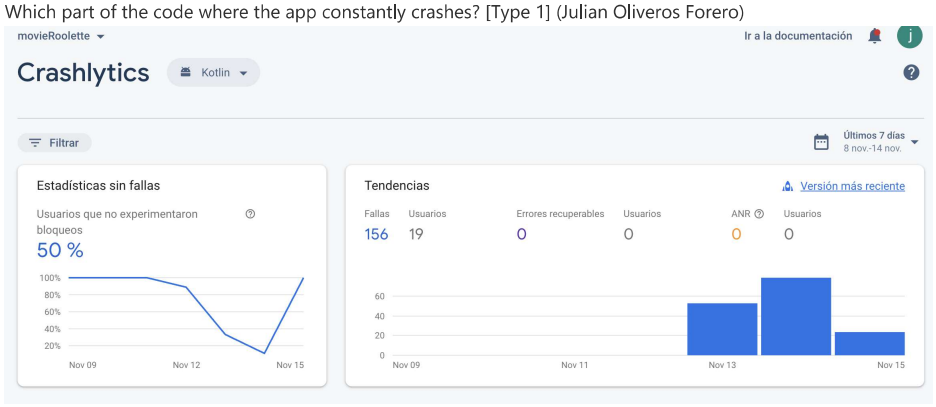
5. How many suggestions does the user pass before the first like on average? [type 3] (Kevin Blanco)
6. How much time does the system take to generate a recommendation, This will be presented in a scale by time [type 1] (Tobia Gasparoni)



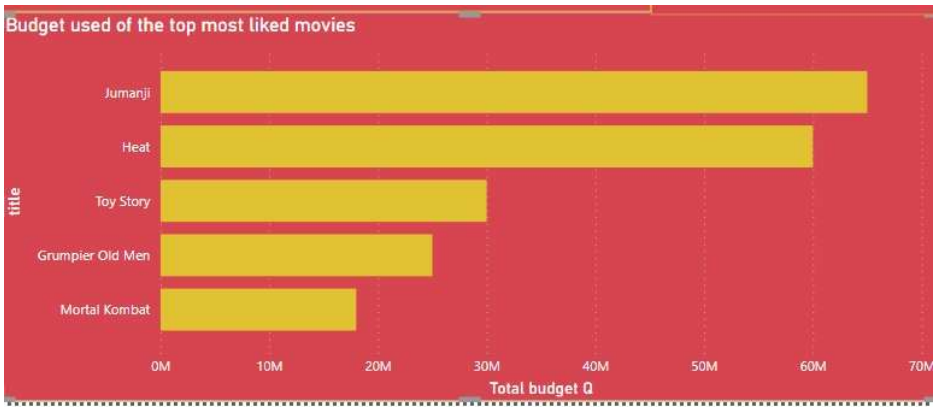
Visualization

1. Which is the most used feature by the users? [Type 3] (Juan Camilo Villamarin)

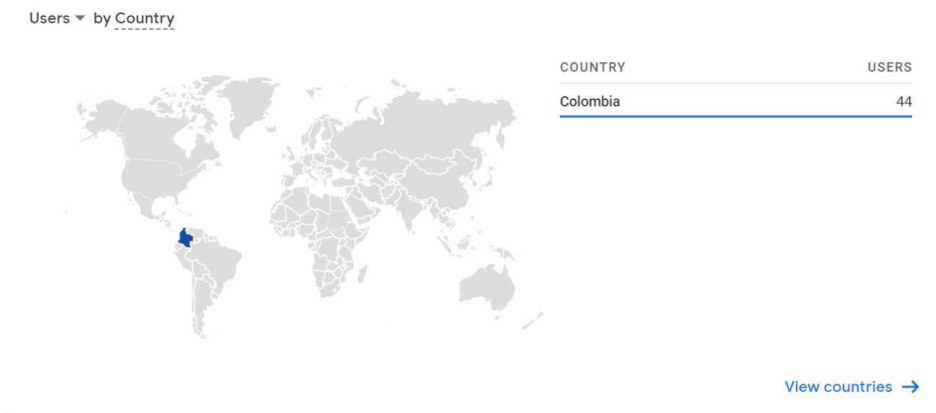
2. Which part of the code where the app constantly crashes? [Type 1] (Julian Oliveros Forero)



3. What is the distribution of genres liked by the user ? This will be presented on a Pie chart [type 2] (Luis Perez)



4. What are the top 5 genres of the most liked movies on the app? [Type 4] (Sebastian Baquero)

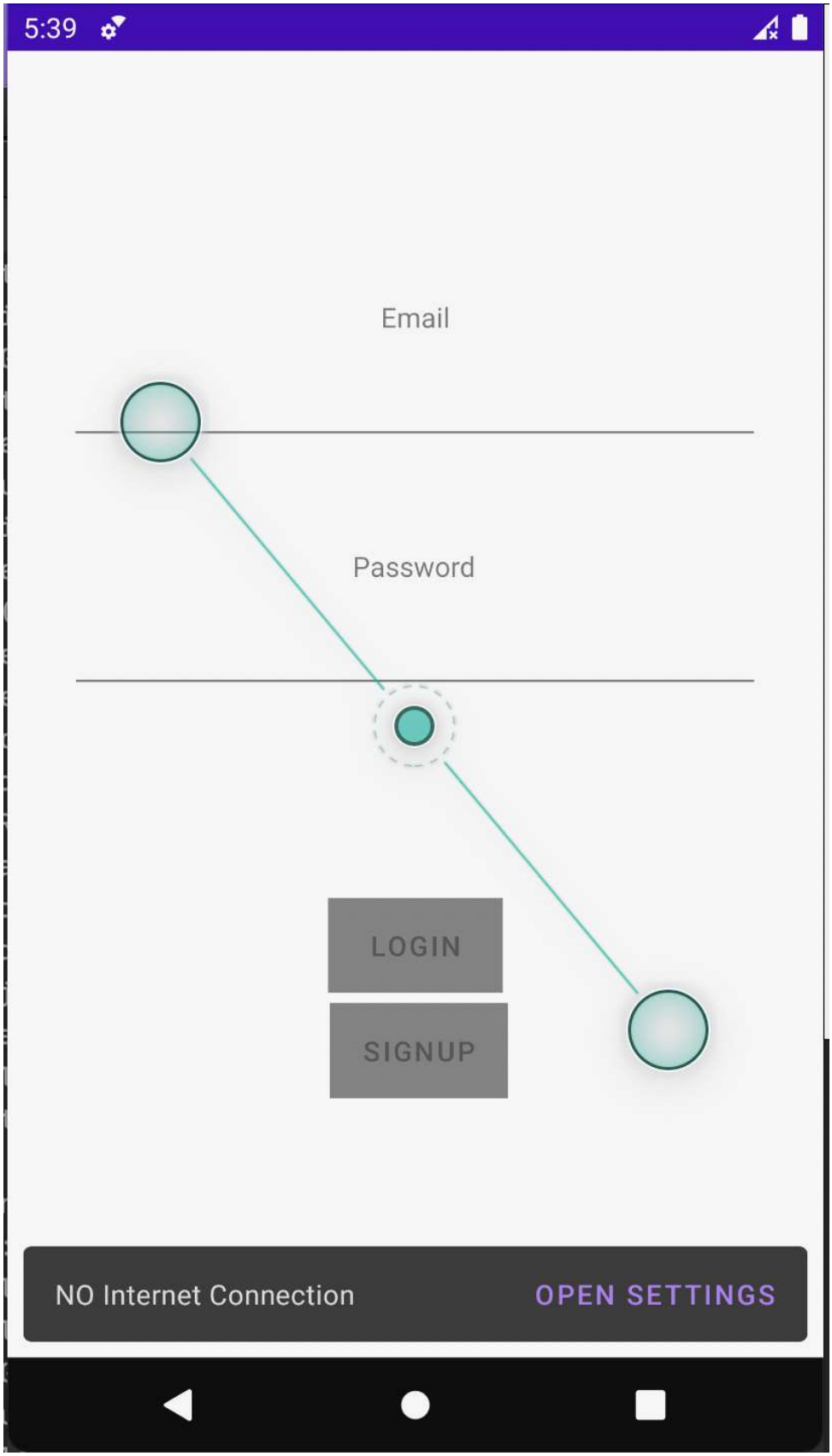


5. How many suggestions does the user pass before the first like on average? [type 3] (Kevin Blanco)
6. How much time does the system take to generate a recommendation, This will be presented in a scale by time [type 1] (Tobia Gasparoni)

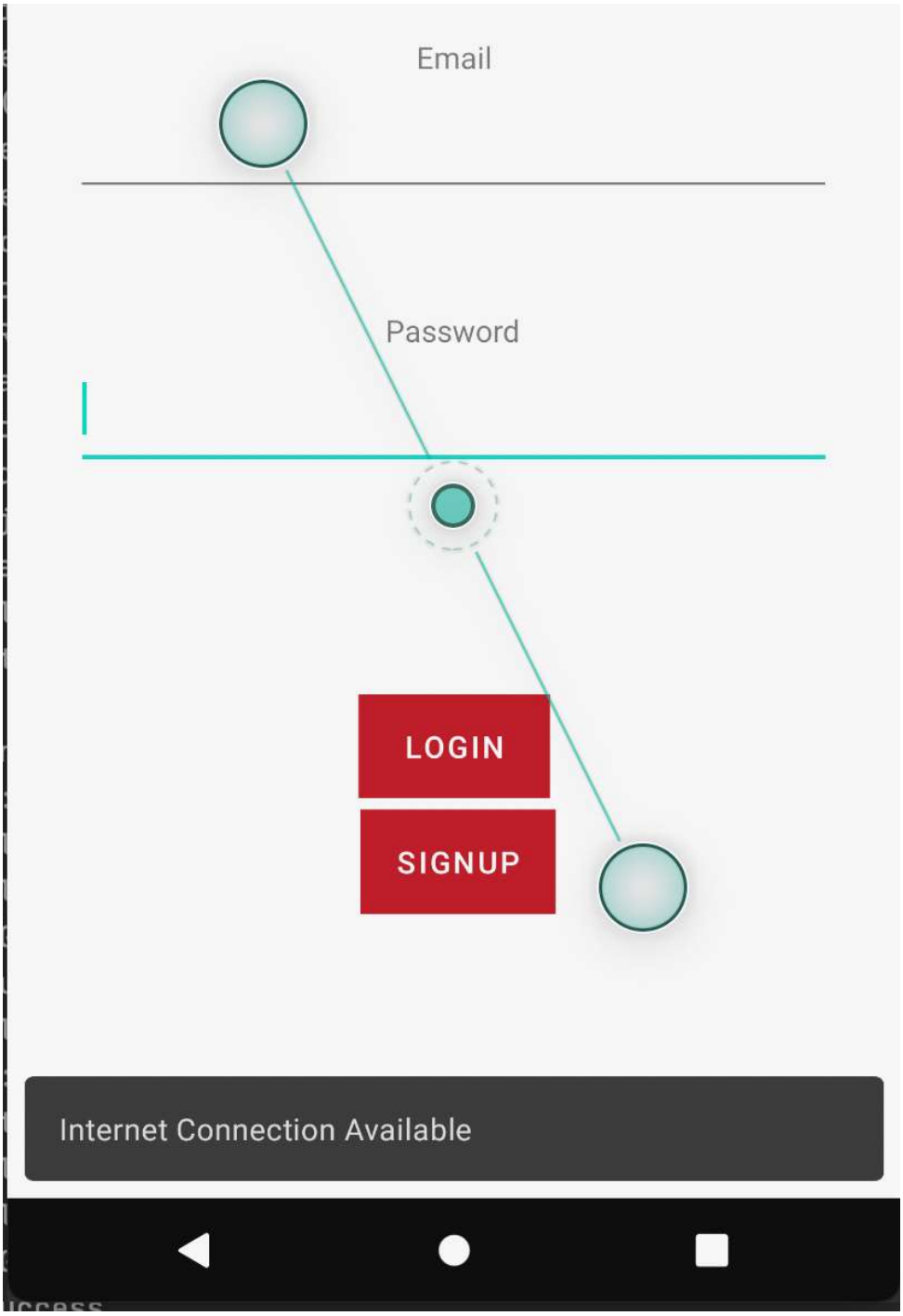
Eventual Connectivity Strategies

Kotlin

1.ECS: The first scenario is on the sign In view, here we have to manage internet connection because if a user is not login he would need to register his credentials and these credentials have to be verified of the Database and if there's no internet these credentials could not be verified so the app is checking for internet connection and if it detects that there's no wifi or Celular data it will disable the buttons and also will show a snack bar that indicates why the buttons are not enabled. Also when the internet connection is back the app will notify that there's internet connection again. You can see both cases in the images below.

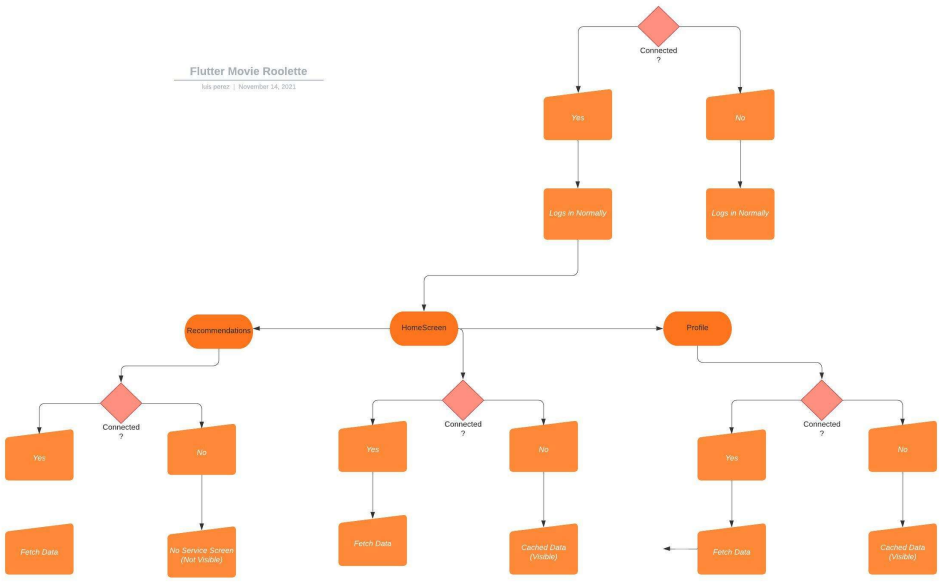

5:39
Email
Password
LOGIN
SIGNUP
NO Internet Connection
OPEN SETTINGS


5:39



2.ECS: The Second scenario its very similar to the first in how the information is showed to the user, the only difference is that the messages are implemented in the sign up view because if the user wants to create an account he will need to have internet connection so that his data can be stored in the database.

Flutter



Local Storage Strategies

Kotlin

Flutter The User is Saved to local storage through the library cache_manager. It is saved as a json file and can be accessed with a key. The data persists even on activity deletion

Multi-threading Strategies

Kotlin

1. Most of our multi-threading strategies are implemented when the app has to do I/O operations, in our case these are implemented when the app needs information from the remote database which is requested through Firebase. We implement such functionality through the coroutines, a Kotlin solution to allow the execution of asynchronous code. The way it is used is with the async/await pattern, which allows other functions of the app to keep running while executing a potentially long-running task.
2. For the eventual connectivity scenarios we send a ping to the Google primary DNS. If the function receives a ping we can know if there's an internet connection or not. Of course this is done by implementing Kotlin coroutines or multi-threading with a Dispatchers IO. This means that the coroutine runs in another thread different than the main.

Flutter

1. Isolates are used to fetch the movies from the database in order to store them into a SQLITE database locally. Also asynchronous code, Futures are implemented to deliver all the information that is not local to the app, Users, Movies and images.

Caching Strategies

Kotlin

In order to cache the movie posters we are displaying on the app, we decided to use the image loading library Glide.

Flutter

Streaming Platform Images are saved on cache while displaying movies, also the profile image, user and piechart data are stored and checked. The data is not fetched unless it is pulled by the user.

Architecture