laSalle

Universitat Ramon Llull

Table of contents

Introduction 3
SDKs used and why 3
Chosen API 4
Design of the app 5
Problems found 9
Conclusions and reflections 9

Introduction

Learnizer is an application that allows students to organize their tasks and notes among different subjects. Each subject has its own folder. He can personalize its interface, upload an instant picture taken with the Camera or upload an image from the gallery. Moreover, he is able to solve his doubts by displaying different Youtube videos and being able to play them to see its content.

For the implementation, we have decided to use Dart as the main programming language and Flutter as the framework. Regarding the authentication and storage of information, we have used Firebase. Finally, for Youtube videos, we have used the Youtube API given from Google APIs.

SDKs used and why

1. Flutter SDK

We have decided to implement our app with Flutter SDK since it allows us to create multi platform applications, such as Android, iOS, web and desktop version. Also, for developers it is quite useful to see the changes in real time meanwhile developing the app.

2. Firebase SDK

Firebase offers a real-time database that synchronizes data in real time between clients. It eases the authentication with real emails. For the different pictures of the tasks, Firebase provides a cloud storage service that enables us to store and retrieve those images in a simple and scalable manner.

3. YouTube API SDK (Google API Client Library)

Since our objective was to configure a window to show different videos according to what the user inputted on the search bar, we connect it to the Youtube API SDK from Google API. It is quite easy to use, generates the curl before using it and enables developers to do some tests with the requests before using them.

Chosen API

We have used Youtube API to allow the user to find videos that could solve his doubt. From all the methods, we have chosen the Search get method, which returns an array of json objects containing information about a video, channel or playing list that coincides with the search parameters stated in the request. In our case, we have filtered by videos and ordered the result by rating to facilitate the work.

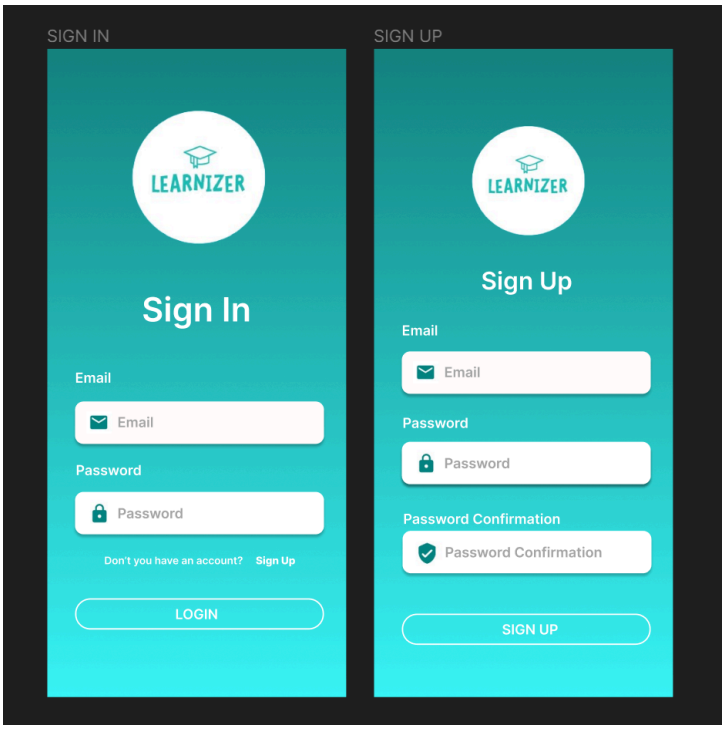
Returned array

```
GET https://www.googleapis.com/youtube/v3/search
```

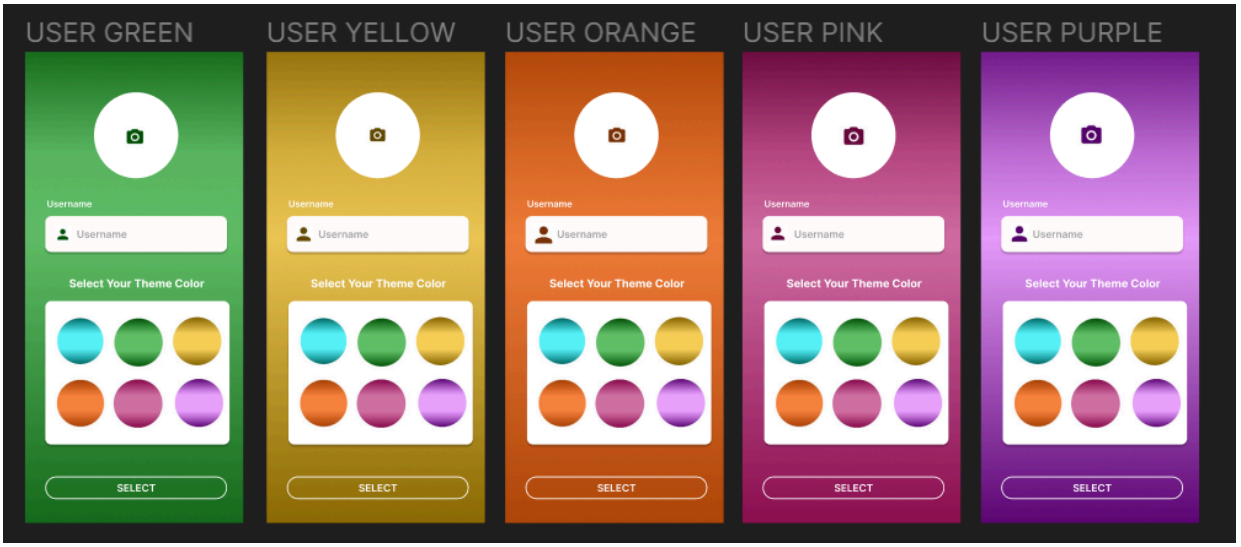
Design of the app

Initial design in Figma

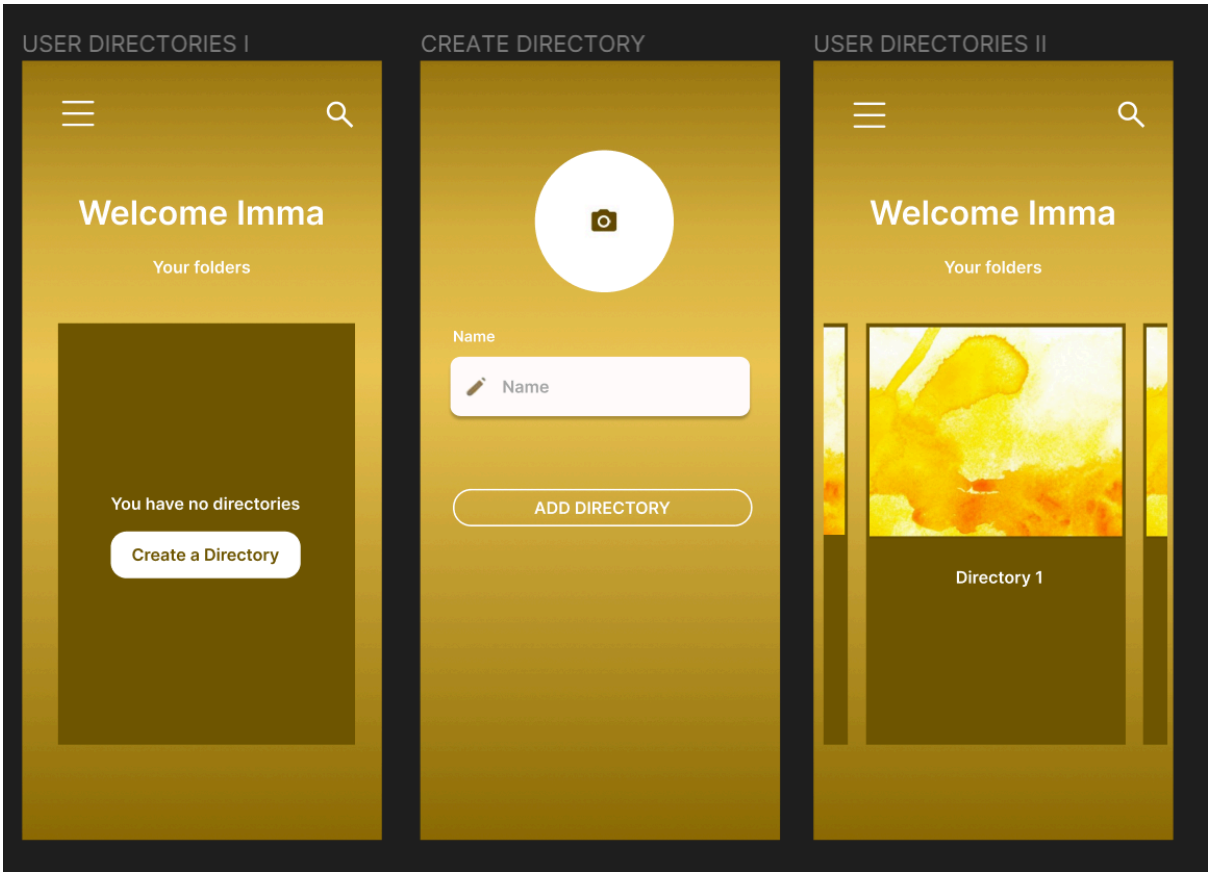
Sign Up / Log In connected to Firebase SDK



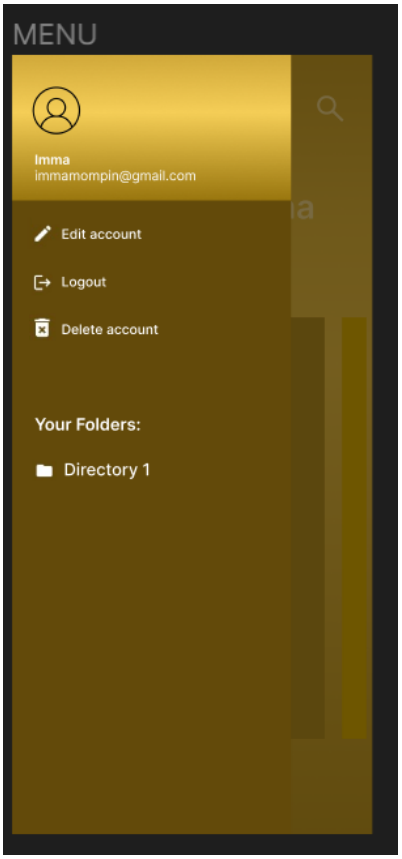
User settings



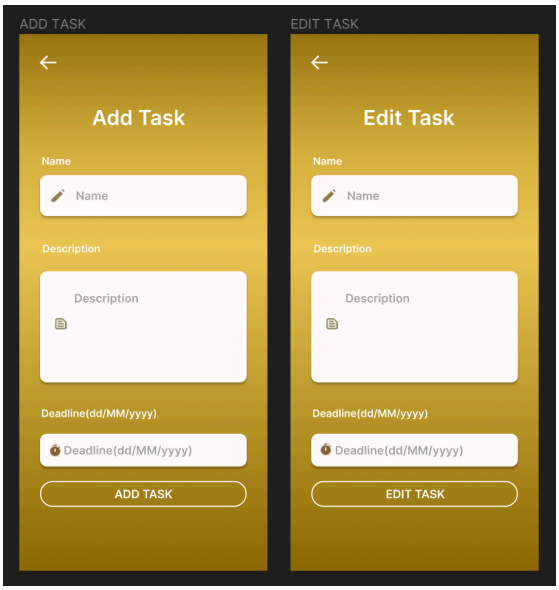
User Directories / Create Directory



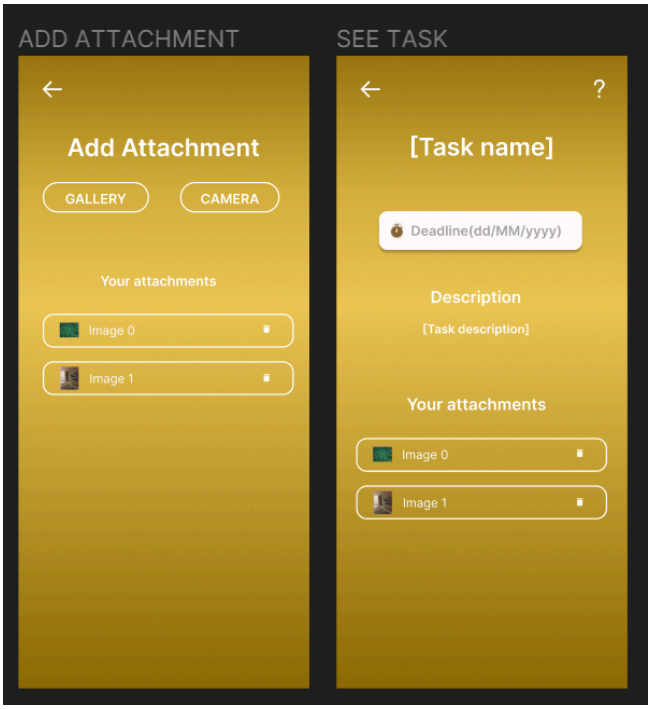
Menu



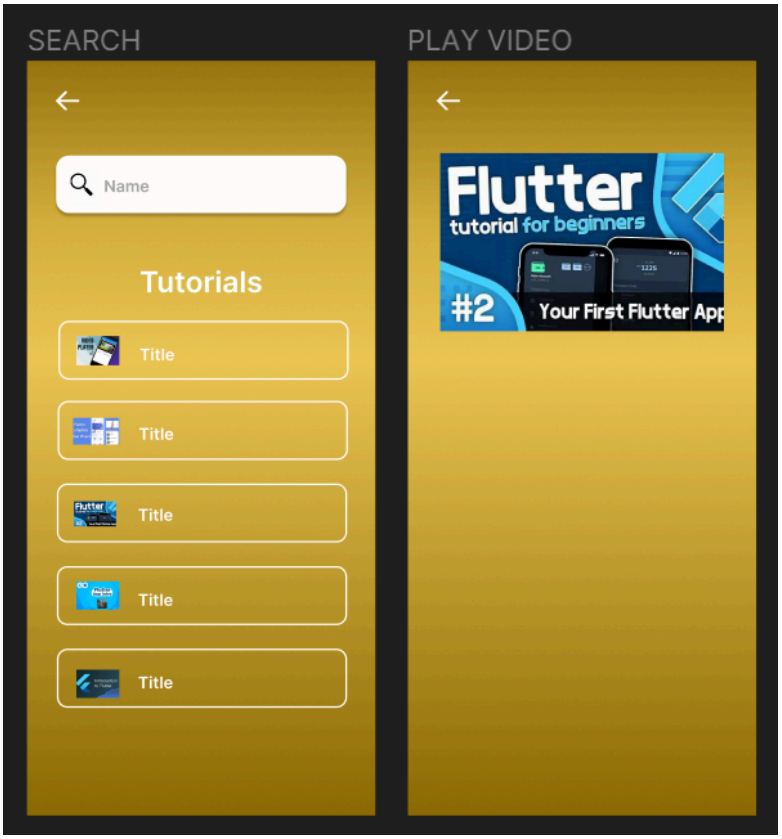
Add / Edit Task



Add Attachment / See Task



Search videos and Play video



Link to the Figma:

https://www.figma.com/file/YsgiXeBdbpktEHBLw442KN/learnizer?type=design&node-id=0%3A1&mode=design&t=rDcO6XOq6Ms1uX4g-1

Problems found

In what comes to the problems, we have found it a little bit difficult to configure Firebase and Flutter for the first time. Specially, adding dependencies that were deprecated and balancing other versions, that is usually diminishing the versions used because of the requirements of the new dependencies. To solve this, we have learned the different commands that flutter has such as *flutter pub get*, *flutter pub outrated*, *flutter pub upgrade*, etc.

Moreover, we could say it was not easy to transform data from Firebase to Flutter, since Firebase has *timestamp* and Flutter *datetime*. Similarly, the different list of objects (tasks and directories in our case) had the same issue, and it had to be transformed into maps. Regarding the app bar, we had the problem that it interfered with the touch interaction of the elements and all effects like the slide of the directories were canceled. It was fixed by configuring it manually with *aligns* and as if they were normal buttons.

Another problem we had is the loading of the image corresponding to the thumbnail link given by the Youtube API. At first, we looked at the Internet and found that maybe it wasn't loading because of the CORS policy, meaning that a server must indicate any domain with an origin other than its own to allow resource loading. We tried with chrome and explorer but then discovered that with Bing it was properly loaded. We solved it by executing on the phone emulator of Android Studio.

Conclusions and reflections

Overall, we have had the intention to apply the different concepts learned in class into our final application. Some of them are the display of lists of Objects, screen replacements, directly implementing Flutter in Android Studio, and improving the User Interface. We have introduced a new API, learned about the different requests possible and got experience programming dart files.

Now that we have finished the project we could have implemented more functionalities such as a calendar with the timeline of the tasks, added the possibility to add a comment to a video, or implement a chat between students.