Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes - RABAT

End-Of-Year Project

Cross-Platform Currency Converter

Realized by :

Abderrahim JAMIAI
Mountasser LABCHIRI

Supervisor :

Pr. Mohamed NAOUM

Academic year 2021/2022

Dedication

We dedicate this effort first and foremost to those who have made sacrifices for our education and well-being: our parents, who have made sacrifices to care for us throughout our education and who are the source of our success; may God keep and protect them.

To our families and beloved friends who have stood by us in the most trying times, as well as to all those who are dear to us, to you all.

Thank you.

Acknowledgment

We would like to convey our heartfelt gratitude to Mr. Mohamed NAOUM, our dear supervisor, for his availability, support, and recommendations. We also want to express our gratitude to him for such a beneficial endeavor that has aided us on both a personal and technical level. We'd also like to express our gratitude to everyone who contributed to the project's success by their advice, assistance, and knowledge. Finally, we thank all of the instructors and administrative personnel at the National School of Computer Science and Systems Analysis (ENSIAS), as well as all of the people who contributed to the construction of this project from far and near.

Cordially.

Abstract

This report summarizes the work carried out as part of our project at the end of the 1st year of our engineering path at ENSIAS in the IDF sector.

Our end-of-year project consists of a cross-platform application that provides currency conversion. A finance-oriented application that runs on all operating systems in which we decided to add more features such as currency variation compared to the USD data chart and trending currencies.

The idea of the project is to create a platform for the conversion of currencies and such finance-related services to facilitate access to current exchange rates and conversion and the diffusion of financial information for professional finance experts and finance passioned enthusiasts.

To do this, we all first analyze the features we should achieve . Then we developed a design based on the main diagram of the UML , and finally we proceeded to the realization of the project using the different technologies described in this report.

Key words : UML, API, Flutter, Firebase

Résumé

Ce rapport résume les travaux réalisés dans le cadre de notre projet de fin de 1ère année de notre parcours d'ingénieur à l'ENSIAS dans le secteur IDF.
Notre projet de fin d'année consiste en une application multiplateforme qui permet la conversion de devises.
Une application orientée finance qui fonctionne sur tous les systèmes d'exploitation dans laquelle nous avons décidé d'ajouter des fonctionnalités supplémentaires telles que la variation des devises par rapport au graphique de données USD et la tendance des devises.
L'idée du projet est de créer une plate-forme pour la conversion de devises et de tels services liés à la finance afin de faciliter l'accès aux taux d'échange et de conversion actuels et la diffusion d'informations financières pour les experts financiers professionnels et les passionnés de la finance.
Pour ce faire, nous allons tout d'abord examiné les fonctionnalités que nous devions réaliser. Puis, sur la base du diagramme principal de l'UML, nous allons créé une conception, et enfin, nous allons mettre en œuvre le projet en utilisant les différentes technologies abordées dans ce rapport.

Mots clés : UML, API, Flutter, Firebase

List of Abreviations

Abreviation	Stands For
API	Application Programming Interface
MCD	Modèle Conceptuel des Données
UML	Unified Modeling Language

Table de matières

List of Figures

General Introduction

During our first year at ENSIAS, we took our first steps in the field of computer science and learned a lot of basic notions. During this process, we were assigned a project in order to refine our skills in know-how.

Our goal is to develop a cross-platform financial application that can provide different financial services at the same time. To illustrate this point, instead of googling all finance-related stuff or using various finance tools' providers, also to allow users to have a fully functional one that is easy to use, clean, fast. This will be achieved mainly using Flutter, Firebase and Yahoo API.

In this report, we will first talk about the general context of the project, treat the part of the analysis and the design then move to the realization by showing you the tools and the final appearance of our application.

In the first chapter, we will shed the light on the general aspects of the project so as to clarify the process, then we will move to the analysis and design in the second chapter. The latter will contain a description of the actors who interact with the application and the needs of each of them, the Use Case Diagram, the MCD, the flowchart being the main actor. And finally, in the 3rd chapter, we will mention the tools used and insert screenshots of the application.

Chapter 1

General Context

Starting with the project aim, specifications, work methodology, and planning, this chapter presents the project in its overall context.

1.1 Project objective

1.1.1 General project objective

IT developing remains one of the most important technology aspects in the world. nowadays, and so if Finance expertise. Therefore, the demand for online finance services is increasing. We will dive next into the details and the content of the specifications and the solutions found to meet these objectives during our first approaches to the project.

1.1.2 Educational objective

The project is intended for finance professionals but also for anyone who is interested in finance or just wants to stay in touch with its updates. For this purpose, we must fulfill objectives for our different users. But, also to invest in our technical skills and perfect our academic knowledge in developing, in object-oriented programming, and in software engineering.

1.2 Cahier de charges

1.2.1 Périmètre

Our application targets all people who want to know the cost of one currency compared to another as well as those who need a particular object or service like trending currencies in market.

1.2.2 Stakeholders

Stakeholders are all those who have a direct (direct) or indirect (indirect) stake in the project's success. As a result, the following are the stakeholders in our cross-platform:

The administrator part : which controls the accounts of the users registered to the application Ainsi que la convenance des publications.

The user part : an area where the user can convert currencies,visualize the variation of multiple currencies compared to "USD"and inspect trending currencies.

1.3 Methodology

1.3.1 V-Model (Cycle en V)

The V-model is a graphical representation of the lifetime of systems development. It's used to create detailed project management and development lifecycle models.

The German V-Modell, a general testing model, and the US government standard are the three primary categories of the V-model.

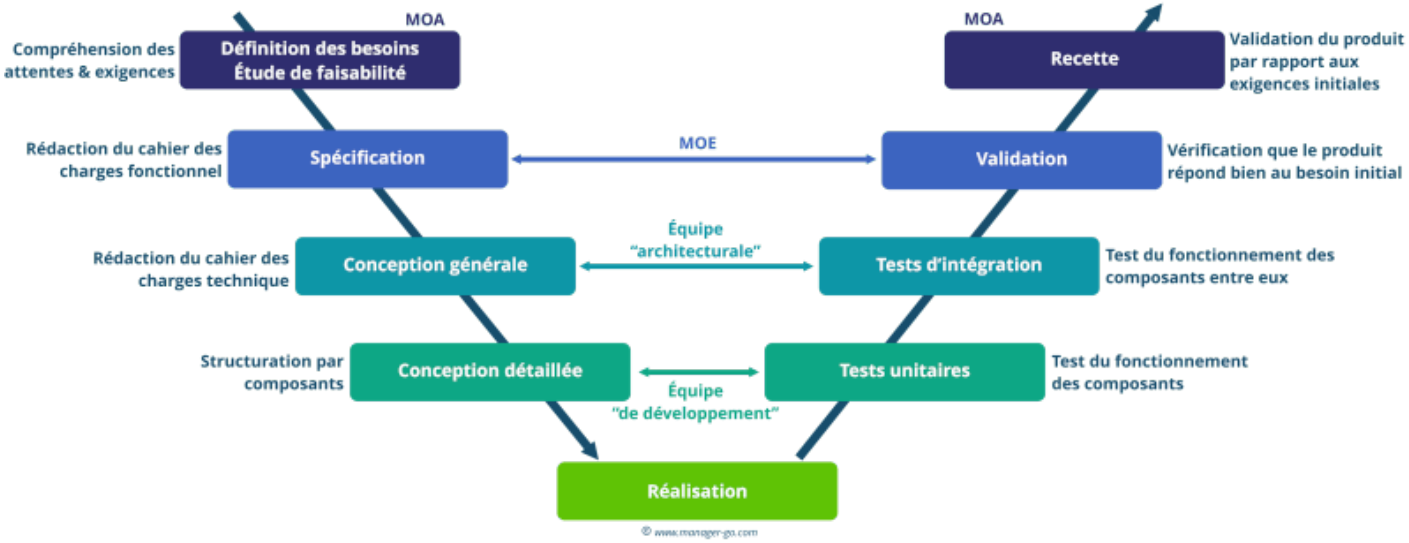

Figure 1.1: V-Model Scheme

1.3.2 Model choice justification

These are the benefits of the V-model over other systems development models:

- **The V-users model**'s are involved in the development and maintenance of the model. The V model is openly maintained using a change control board. Every day to weekly, the change control board meets to process all change requests received throughout system development and testing.

- **The V-model** gives precise guidance on how to execute an activity and its work phases by explicitly specifying the events required to complete each work step: each activity schema includes instructions, recommendations, and extensive explanations of the activity.

1.4 Planning

1.4.1 Gantt chart

Our project's administration is based on a chronological breakdown (phases) of the project, which specifies what has to be done (tasks) and by whom (resources).**Gantt chart**, a sort of bar graph that depicts a series of tasks over a set period of time, is used to simulate the scheduling. A Gantt chart is a visual illustration of a project's schedule that displays a list of activities that must be accomplished within a project, along with their start and conclusion dates.

It is presented as follows :

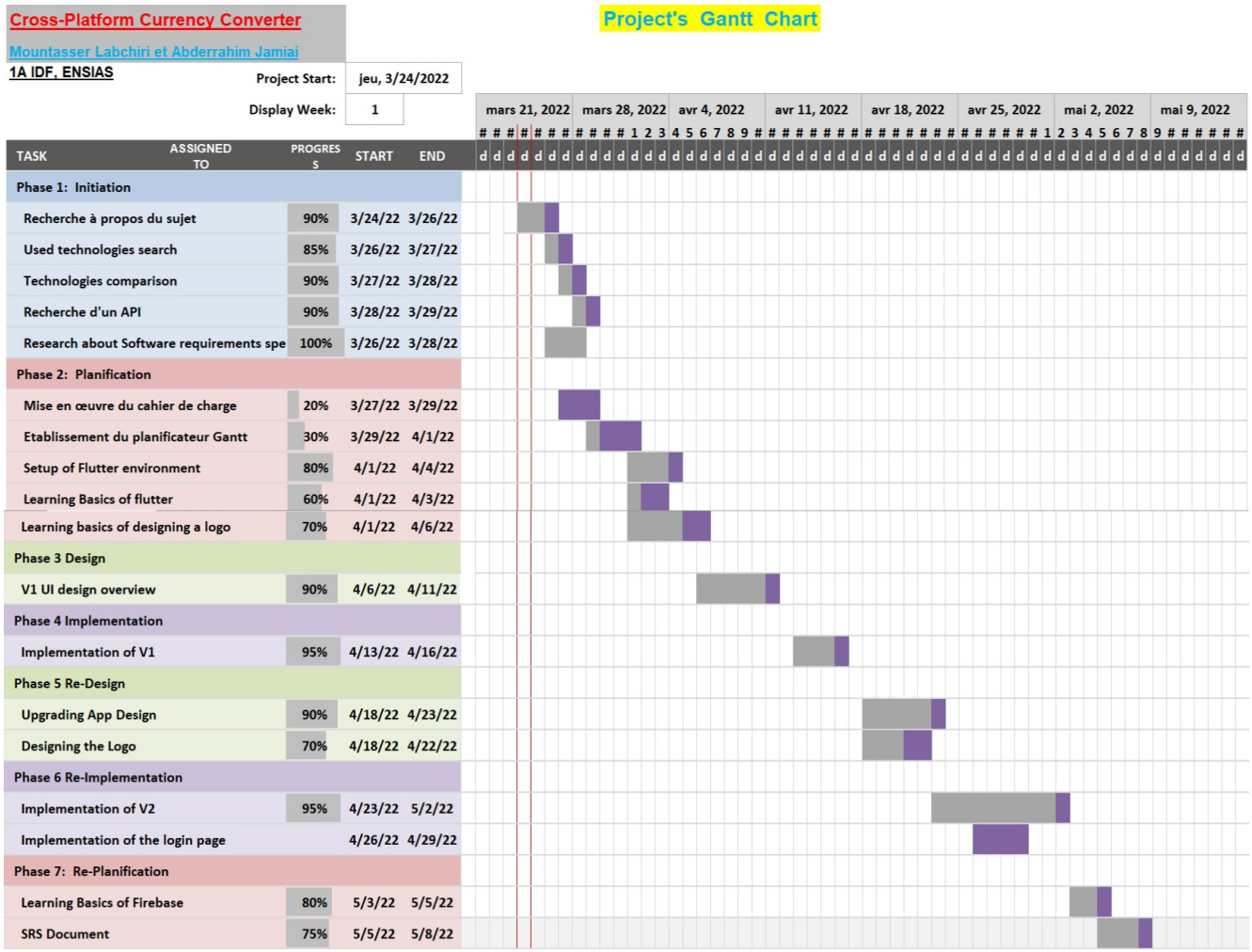

Figure 1.2: Gantt chart

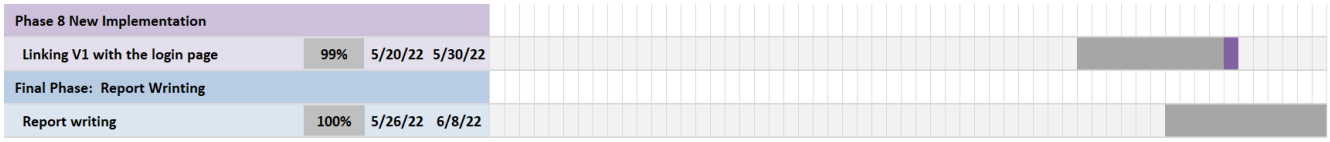

Figure 1.3: Gantt chart (continuation)

1.5 Conclusion

This chapter served as the foundation for the project's development, as it outlined the overall goal to be achieved, the project's specifications, as well as the methodology of the life cycle methodology employed and the project's planning.

Chapter 2

Analysis and Conception

In this section, we'll go over how to identify all of our system's functionalities for each type of user, starting with a list of functional needs, then understanding the list of requirements translated into non-functional needs, and finally looking at the various use cases that our system has to offer.

2.1 Analysis

2.1.1 Qualification du besoin

In this section, we'll go over how to identify all of our system's functionalities for each type of user, starting with a list of functional needs, then understanding the list of requirements translated into non-functional needs, and finally looking at the various use cases that our system has to offer.

- **Who does it serve? :** customer or intended user
- **What does it act upon?:** elements on which the subject acts, the work material
- **For what purpose? :** main need to satisfy

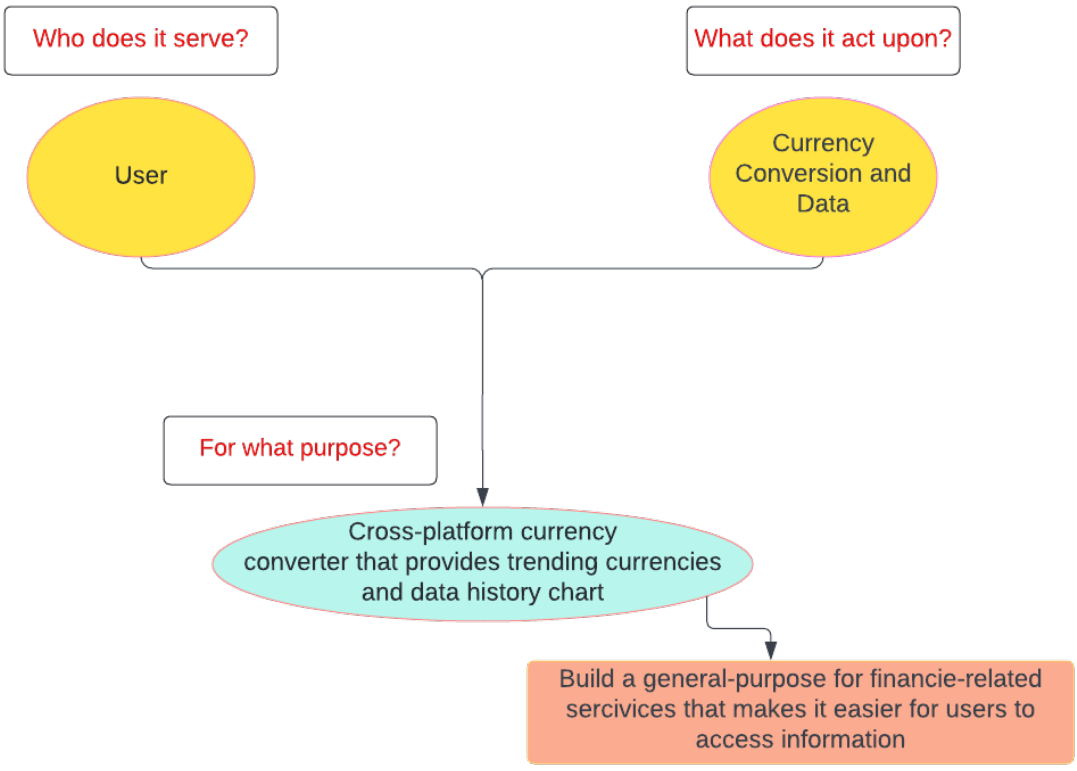

Figure 2.1: La bête à corne

2.1.2 Identifications of functional requirements

- **Our application will contain two spaces:**

1. **User space :**
   - **Create account :** The user can create an account by inputting his informations.
   - **Convert currencies :** The user must be able to convert between currencies of over 150 countries .
   - **Visualize the variation of currencies :** The user must be able to visualize the variation of multiple currencies compared to USD past-over a previous period area.
   - **Inspect trading currencies :** It shows all trading currencies that had significant variations in the world .
2. **Administrator space :**
   - **Create account :** By providing his details, the administrator can create an account.
   - **Manage users' accounts :** The administrator must have access to a list of users and their information, as well as the ability to delete accounts.

2.1.3 Identifications of non functional requirements

Non-functional requirements are defined as quality requirements. A non-functional requirement is not directly related to the business. It is rather related to the quality standards in force and

also to the constraints of realization. Thus, the non-functional requirements of our project are summarized in :

- **Security:**

  Each user, to access the application, is obliged to authenticate himself by a user name and a password. He will only be able to access the pages that are accessible to everyone, and not to other customers' data, like an administrator, since it is their privacy. Also, there won't be room for fake or spam accounts.

- **Compatibility**
  - Our cross-platform application must be compatible with all browsers for web, and all operating systems: Android, iOS, Fuchsia (Google Operating System), Linux, Windows, and macOS.
  - Our application should be responsive with all devices: phones, tablets, and laptops.

- **Maintainability** The source code of our application must be readable and understandable if needed for future improvements.

- **Ease** The application must contain easy-to-use interfaces, i.e: it has to be user-friendly.

2.1.4 Description of Cases

The use case diagram was created to investigate the needs of the actors as well as the expected functionalities of our system. To do so, we first identified the actors and defined the various use cases that our system provides.

Identification of actors

It is strongly advised to first identify the different actors of the system before developing any use case diagram. A part played by an external entity, such as a person, an object, or another system, is represented by an actor. a different system

The following table shows us the different actors of the system, each with its own type and its own description:

Actor	Type	Description
User	Main	User that wants to convert, visualize variation of currencies and trending ones
Administrator	Main	Super administrator of the application, in charge of managing the accounts

Figure 2.2: Table of actors

2.1.5 Use Case Diagram

The use case diagram is a visual representation of the functional links between the actors and the system under investigation. As we've seen, there are two actors who interact with the system depicted in the use case diagram below.

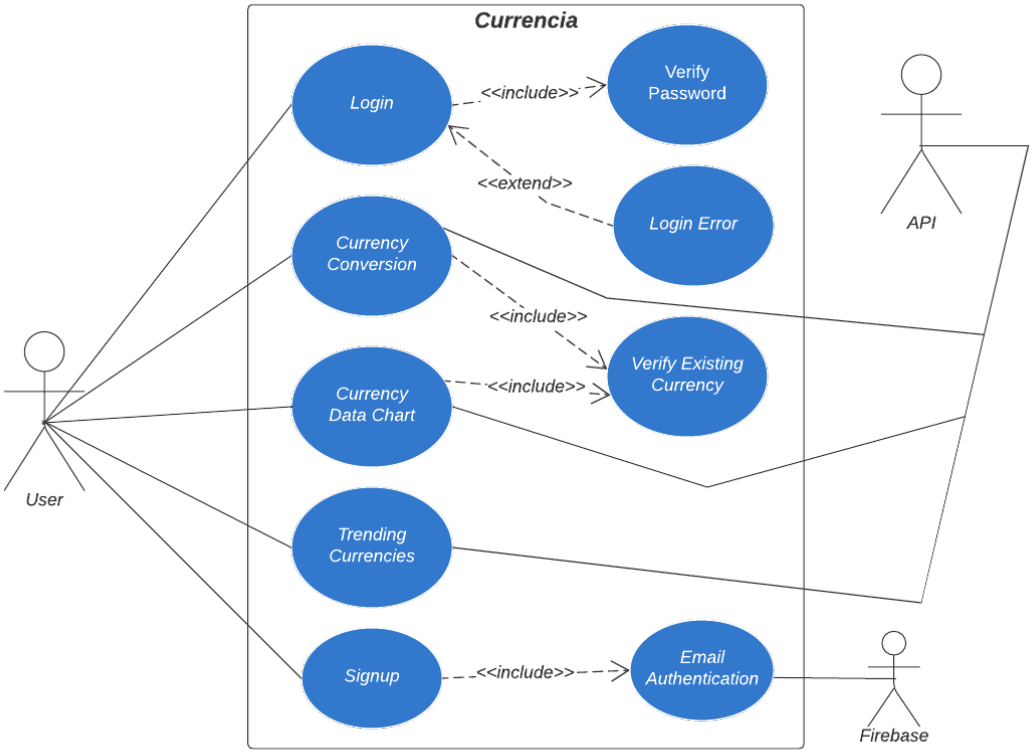

Figure 2.3: UML Use Case for the user

- **For the admin case, there would be nothing less than managing the user's database such as: activity, login values, and email.**

2.2 Conception

2.2.1 Sequence diagram

In the world of software engineering, a sequence diagram or system sequence diagram depicts process interactions in time order. It displays the processes involved as well as the messages transferred between them in order to complete the functionality.

It displays the processes involved as well as the messages transferred between them in order to complete the functionality.

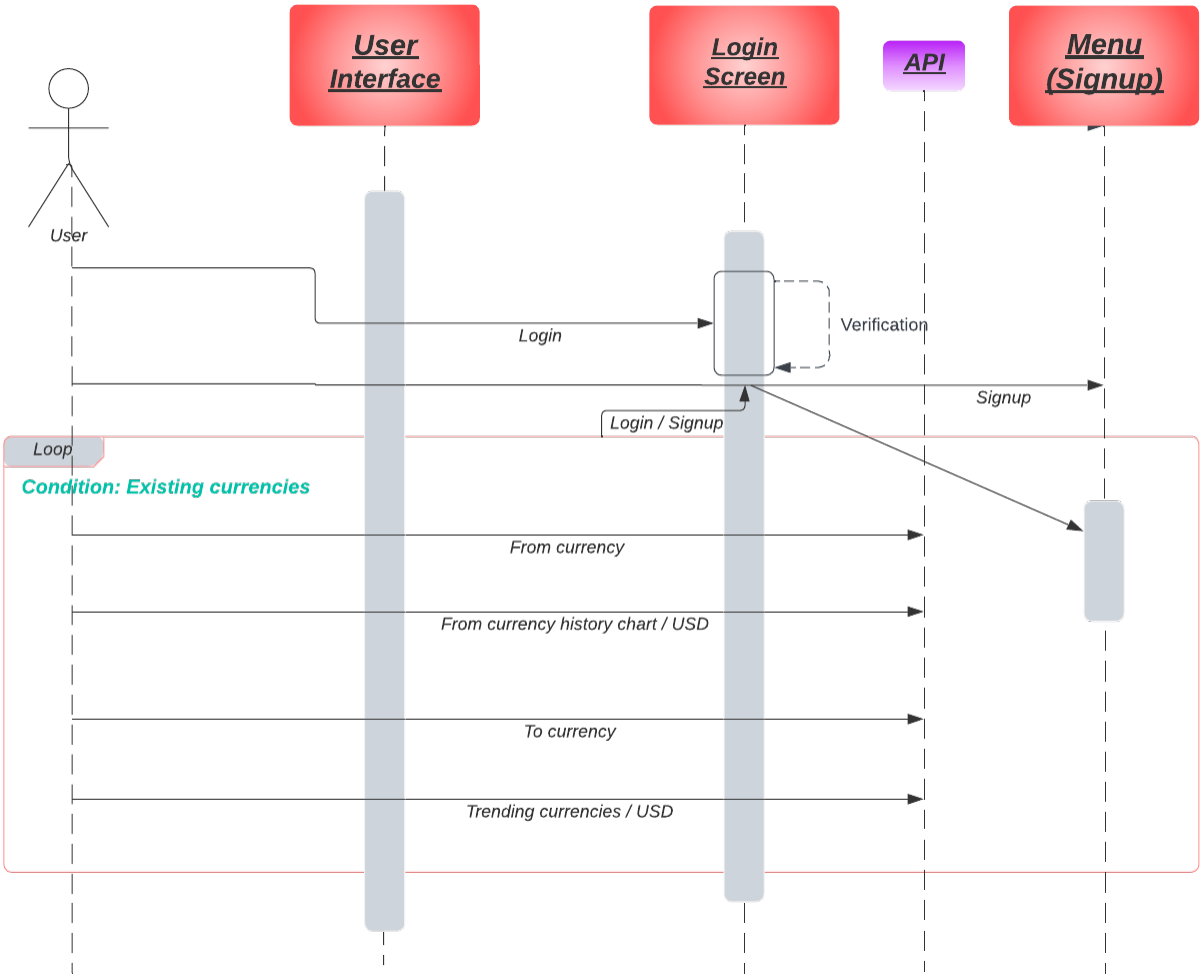

Figure 2.4: Sequence Diagram

2.2.2 Merise MCD

The purpose of the MCD (Modèle Conceptuel des Données) is to formally write down the data that will be used by the information system. It is therefore a representation of the data, easily understandable, allowing the understandable, allowing the information system to be described using entities.

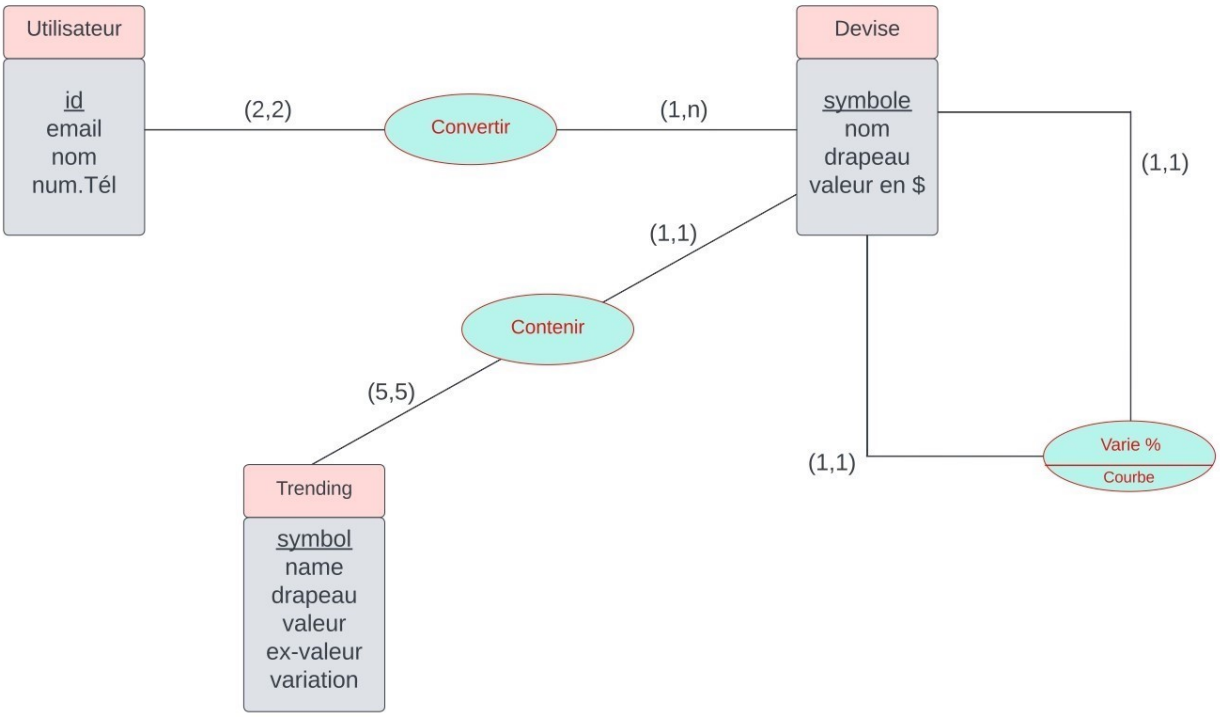

Figure 2.5: Merise MCD (Modèle Conceptuel des Données)

2.3 Technical study

2.3.1 Software architecture

Two-tier architecture characterizes client/server systems where the client requests a resource and the server provides it directly, using its own resources. This means that the server does not call on another application to provide part of the service.

An additional note on two-tier architecture is that the word ”tier” commonly refers to splitting the two software layers onto two different physical pieces of hardware. Multi-layer programs can be built on one tier, but because of operational preferences, many two-tier architectures use a computer for the first tier and a server for the second tier.

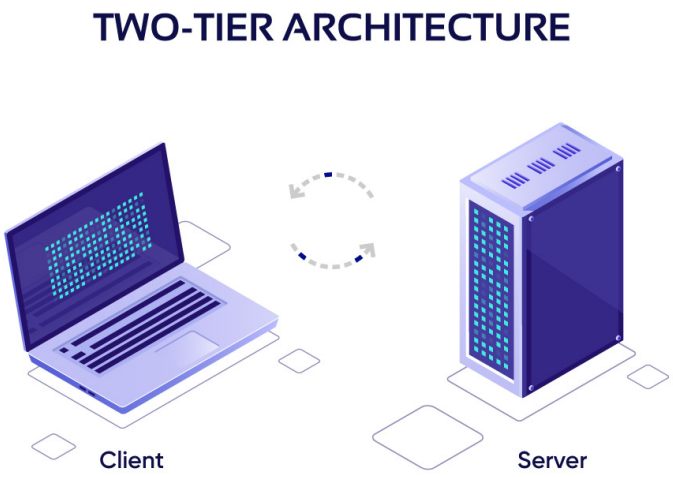

Figure 2.6: Two-tier architecture

2.3.2 Flutter architecture

Flutter provides the basic architecture that you may use to apply to your application and simply manage its state. Flutter's architecture is referred to as the Business Logic Component (BLOC). It's a state-based method that allows you to initiate events and handle state changes as a result of them. The BLOC is a useful strategy for separating business logic from the user interface and testing important business logic points. The BLOC architecture was designed with simplicity, scalability, and testability in mind, and all of these aims were met. However, this is a different topic that we may discuss at a later time.

And when it comes to app architecture, structural Model-View-ViewModel pattern can help us decide how the different parts of the app are organized.

In this context, we can use the repository pattern to access data objects from various sources, such as a backend API, and make them available as type-safe entities to the domain layer of the app

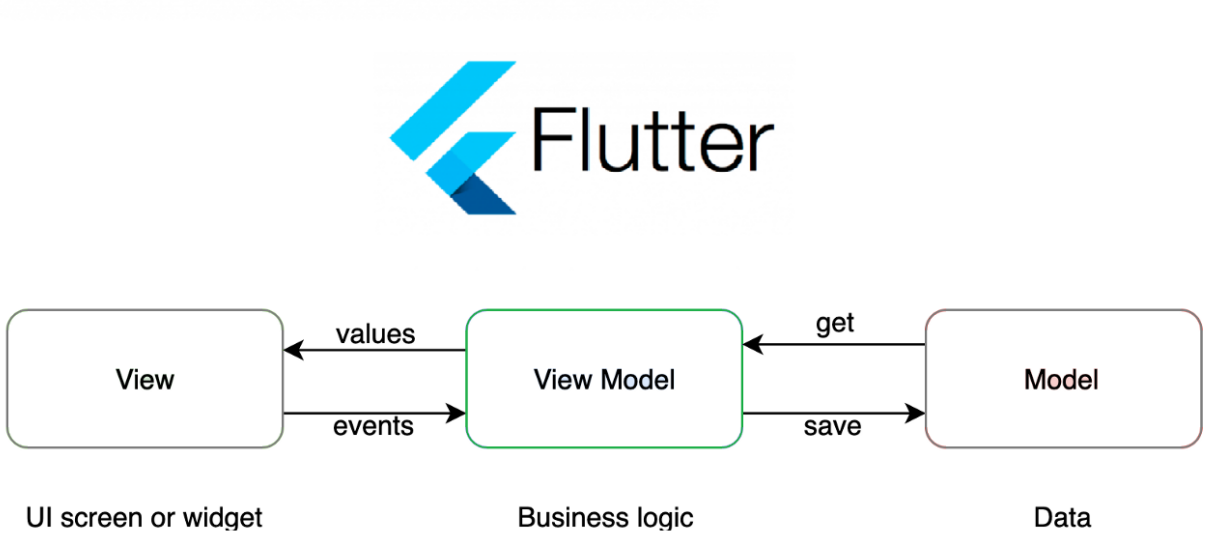

Figure 2.7: MVVM: Model-View-ViewModel pattern

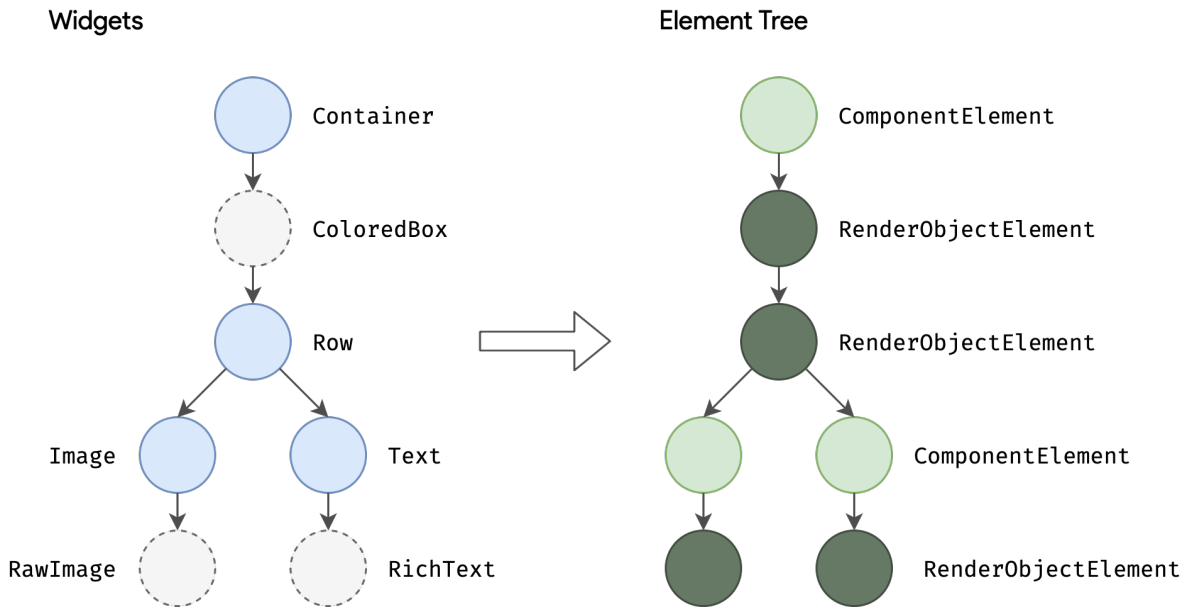

Figure 2.8: Flutter Architectural Overview

2.4 Conclusion

In this chapter, we have modeled our cross-platform application by providing answers to our modeling and design questions. Based on the analysis of the needs of our application, we were able to substantiate our project's needs and requirements, and explained how our information system is going to work by making use of diagrams: UML use case, MCD Diagram, Sequence Diagram, Unicorn Model. The following chapter will approach the realization of **Currencia** when it comes the implementation and building.

Chapter 3

Realization

This chapter introduces the project's hardware and software environment. Then, in the context of particular usage cases, we're interested in the description of some interfaces of the implemented system.

3.1 Technologies and used Frameworks

This section covers the many technologies and software used in the creation of the proposed solution, as well as a general overview of each. The latter is correct. This section discusses the programming languages used and the frameworks that support them.

3.2 Tools and used Languages

3.2.1 Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. First described in 2015, Flutter was released in May 2017.

Figure 3.1: Flutter

- Flutter: developed by Google team, allows instant changes which is better for fixing bugs, rich UX, a valid code base for all platforms and plugins (for mapping and geolocation...) unlike React Native. The reason why we chose Flutter over React Native which is one of the best frameworks that is developed by Meta Facebook and considered one of the big Flutter competitors.
- Flutter is Google's free and open-source UI framework for creating native mobile applications. It allows developers to build mobile applications with a single codebase and

programming language. This capability makes building both iOS and Android apps simpler and faster. But, it is not the best for web developement compared to JavaScript.

Regardless, Flutter runs on all operating systems, respectively in the images below: Linux, Apple, Android, Web, Windows, macOS, and Fuchsia.

Backend

- **Firebase**

  Google's Firebase technology for backend allows developers to create mobile and online applications. It started off as a stand-alone business in 2011. Google bought the platform in 2014, and it is now their main option for app creation.

Figure 3.2: Firebase

- **Yahoo Finance API**

  An API service offered by Yahoo. An application programming interface (API) is a link between two computers or programs. It's a form of software interface that provides a service to other programs. An API specification is a document or standard that explains how to create or use a connection or interface.

Figure 3.3: Yahoo Finance API

- **Dart**

  Dart is a client-side programming language that may be used to create web and mobile apps. Google created it, and it can be used to create both server and desktop apps.

  It's a garbage-collected, object-oriented, class-based language with C-style syntax. Interfaces, abstract classes, reified generics, and type inference are all supported, and it can compile to native code or JavaScript.

Figure 3.4: Dart

3.2.2 Lucidchart

Lucidchart is a web-based diagramming application that allows users to work together visually to create, revise, and share charts and diagrams, as well as improve processes, systems, and organizational structures. It is created by Lucid Software Inc., a company formed by Ben Dilts and Karl Sun and situated in Utah, United States.

Figure 3.5: Lucidchart

3.2.3 MockFlow

MockFlow is a cloud-based wireframe program that allows designers to collaborate in real time on website and software user interface prototypes.

MockFlow is a web-based wireframe program that helps designers plan, produce, and share their work. MockFlow gives users access to a wide library of mockup components, icons, stickers, and other shapes, allowing them to create professional-looking interface prototypes fast and easily.

MockFlow's built-in sharing tool and team chat make real-time design collaboration simple.

Figure 3.6: MockFlow

3.2.4 Adobe Photoshop

Adobe Photoshop is a raster graphics editor for Windows and macOS developed and marketed by Adobe Inc. It was first designed by Thomas and John Knoll in 1988. Since then, not only in raster graphics editing, but also in digital art in general, the software has become the industry standard.

Figure 3.7: Adobe Photoshop

3.2.5 Figma

Figma is a sophisticated design tool that can be used to build anything, including websites, apps, logos, and more.

We took our initial steps into User Experience and User Interface Design by learning to use Figma. It has helped us have an idea about how our application would look like.

It is a vector graphics editor and prototyping tool that is mostly web-based, with desktop apps for macOS and Windows enabling additional offline features.

Figma prototypes may be viewed and interacted with in real time using the Figma mobile app for Android and iOS.

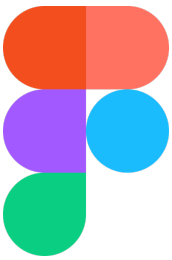

Figure 3.8: Figma

3.2.6 GitHub

GitHub is a free and open source platform for version control and collaboration. It's built on Git, a code management system that lets you store a project's source code and see a comprehensive history of all the changes you've made to it, which proved essential for code sharing and collaboration on our project.

Figure 3.9: GitHub

3.2.7 IDEs

- **Visual Studio Code**

Microsoft's Visual Studio Code, usually known as VS Code, is a source-code editor available for Windows, Linux, and macOS. Debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among the features. Users can customize the theme, keyboard shortcuts, and preferences, as well as install extensions that offer new features.

Visual Studio Code was voted the most popular development environment tool in the Stack Overflow 2021 Developer Survey, with 70 percent of 82,000 respondents saying they use it.

Figure 3.10: Visual Studio Code

- **Android Studio**

Built on JetBrains' IntelliJ IDEA software and designed exclusively for Android development, Android Studio is the official integrated development environment (IDE) for Google's Android operating system. In 2020, it will be accessible for download on Windows, macOS, and Linux operating systems, as well as as a subscription-based service. It takes the position of the Eclipse Android Development Tools (E-ADT) as the primary IDE for developing native Android apps.

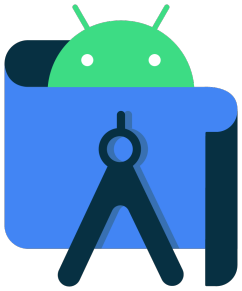

Figure 3.11: Android Studio

- **Xcode**

Apple's IDE (integrated development environment) Xcode is used to create software for macOS, iOS, watchOS, and tvOS. It is the sole officially supported tool for developing and distributing programs to Apple's app store, and it is suitable for both novice and professional developers.

Xcode is a software package that contains all of the tools needed to create an app, including a text editor, a compiler, and a build system. You can write, compile, and debug your program with Xcode, and then submit it to the Apple app store once it's finished. It

includes a lot of tools to make the development process go more swiftly, so seasoned developers may create apps in record time, and newbies will have fewer questions and barriers to overcome in order to produce a wonderful app.

Figure 3.12: Xcode

3.2.8 LaTeX

Leslie Lamport, an American computer scientist, designed LaTeX as an extension to the TeX typesetting system in 1985. LaTeX was intended to make it easier for TeX users to generate general-purpose books and articles. Because LaTeX is a derivative of the TeX typesetting system, it can typeset technical papers containing complex mathematical calculations. LaTeX became popular among scientists and engineers because of this capability.

Figure 3.13: LaTeX

3.3 Currencia application

We will be showing, next, screenshots from our project's build on iPhone and Android devices.

3.3.1 Currencia's brand logo

We designed the brand's logo with its name written in such special font along with its slogan that says "Finance Made Easy"

Figure 3.14: Currencia's logo and slogan

3.3.2 Application's Wireframe

Using, the mentioned tool, MockFlow, we have created a primary wireframe of how our application would look like

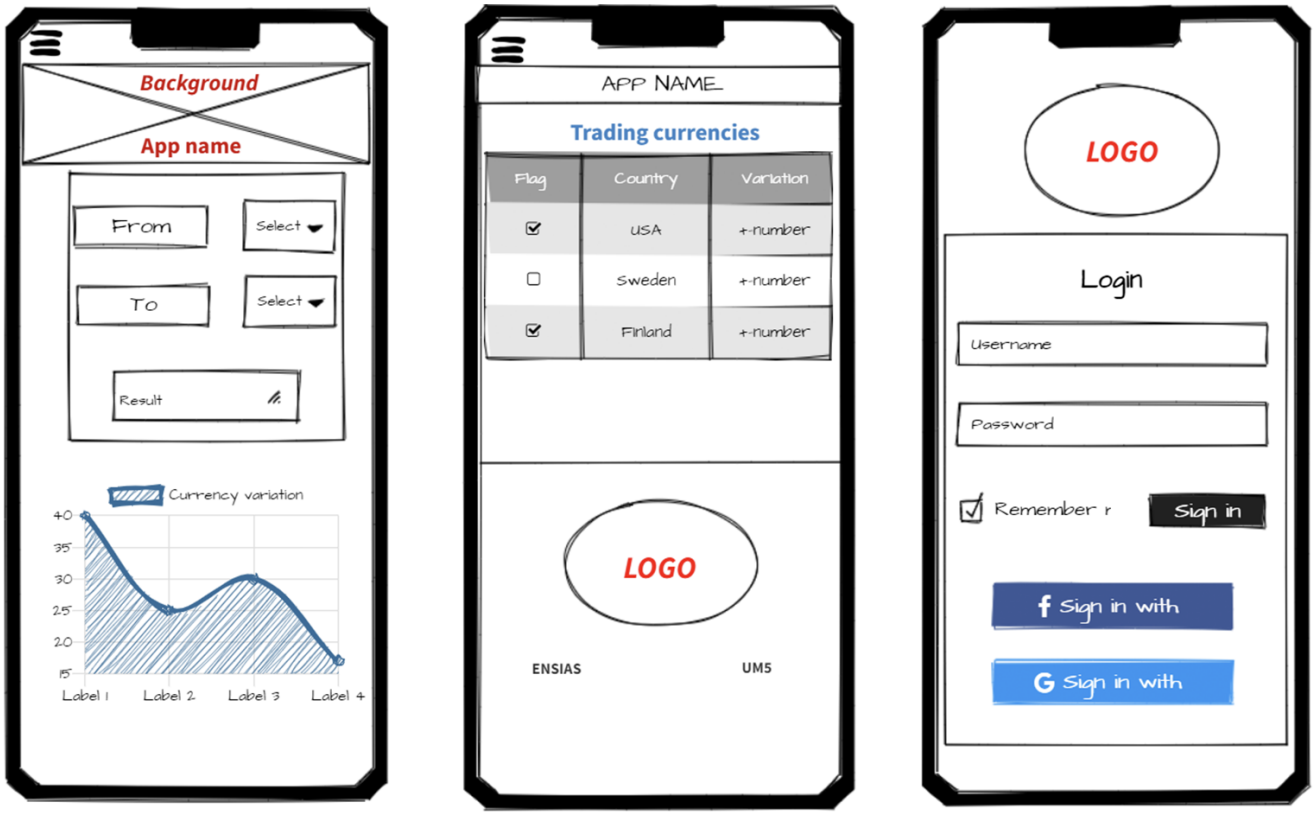

Figure 3.15: Application's wireframe

3.3.3 Application's icon

The application before starting looks this way

Figure 3.16: Application's name and icon

3.3.4 Application's launch screen

Currencia when launched, its popup screen has this appearence

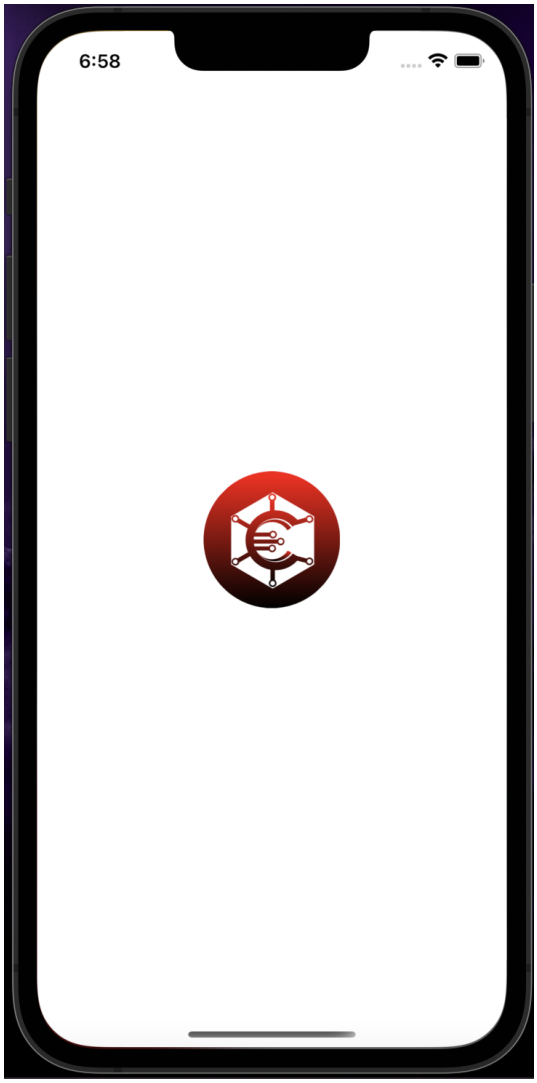

Figure 3.17: Launch popup screen

After the user starts Currencia, they'll be redirected to the home screen that, in which, there is the main currency conversion interface

3.3.5 Conversion interface

The user's currency conversion appears in this way

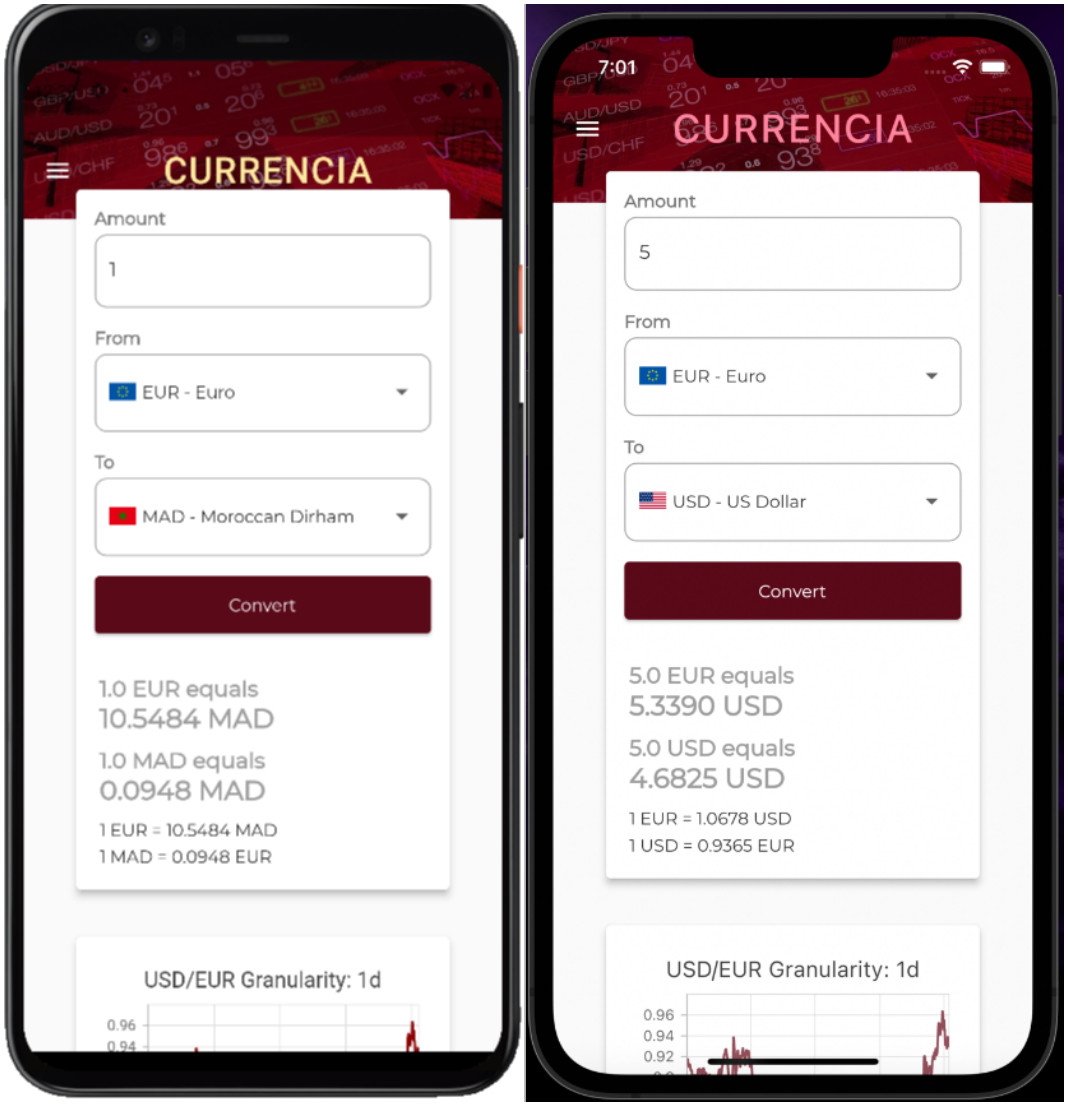

Figure 3.18: Conversion UI

3.3.6 Choosing a currency

Choosing from and to currencies remains an option that is going to be often used in our app. Therefore, it has to be a beautiful looking UI. As you can see, we added the currency's country flag.

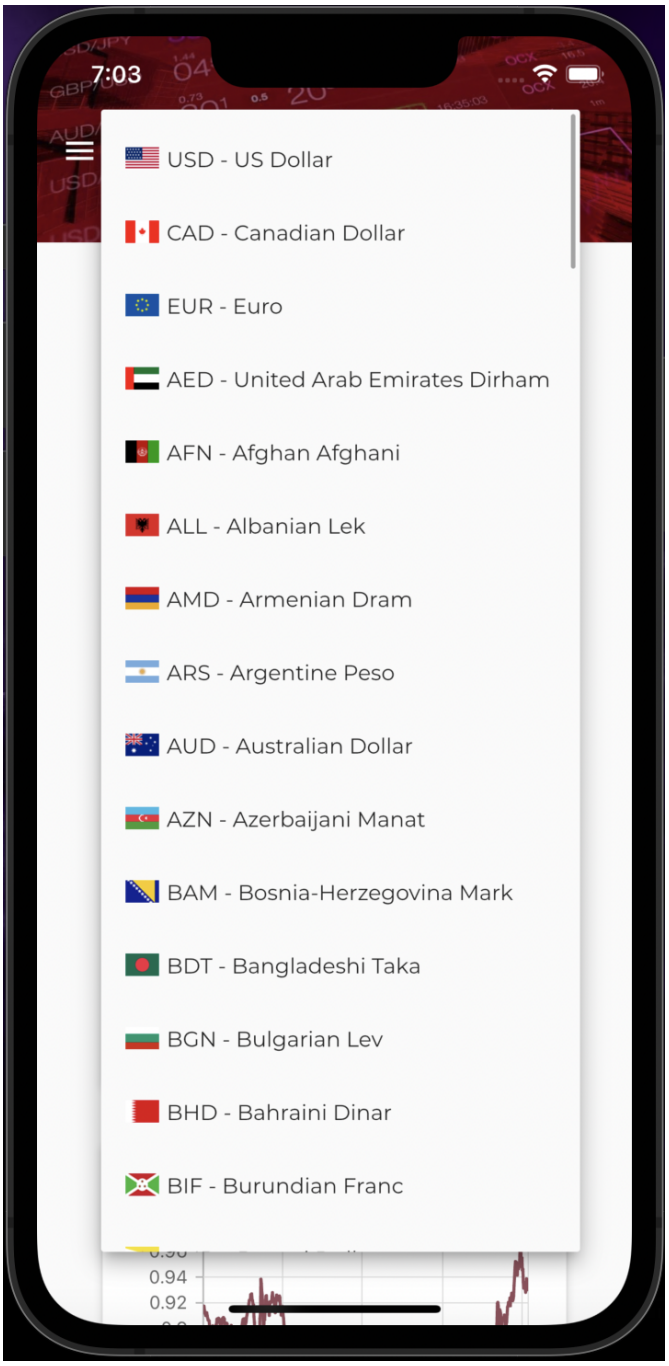

Figure 3.19: List of available currencies

3.3.7 Currency's history data chart

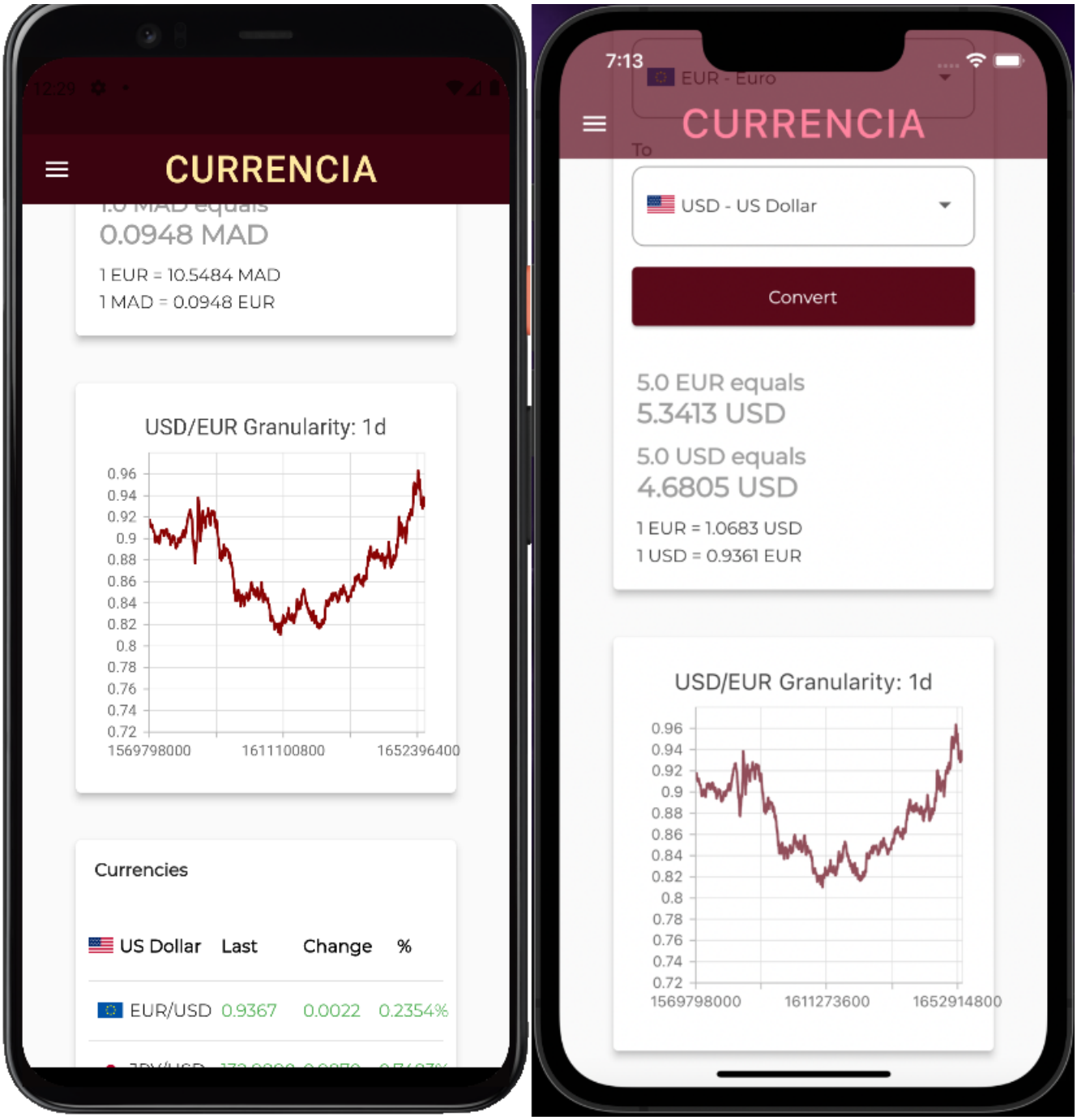

Figure 3.20: Currency's value data chart

3.3.8 Trending currencies

Figure 3.21: Trendings' category

3.3.9 Application's footer

When scrolled down, the homepage will show the footer, in which, we added our app logo, our university UM5, our engineering school ENSIAS, our social media usernames, and the app reserved copyrights.

Figure 3.22: Currencia's footer

3.3.10 Menu drawer

We planned to add the signup and login options in the sidebar, before deciding to have it at the app's launch.

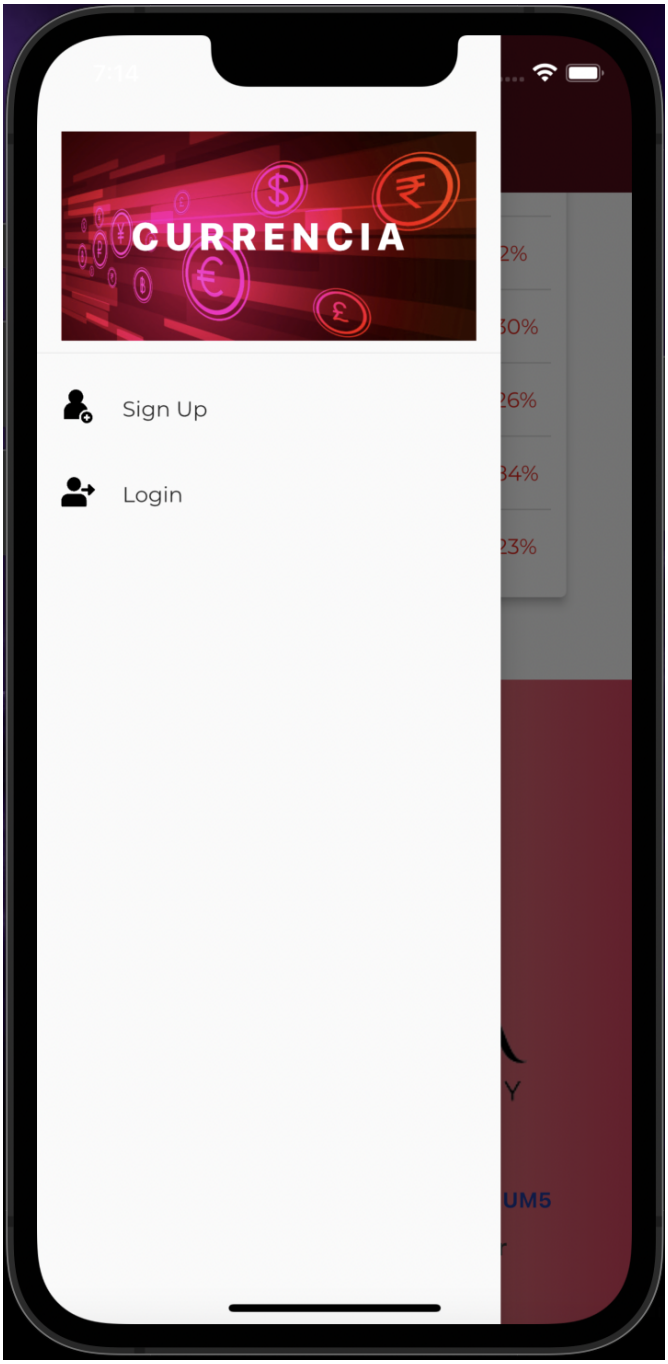

Figure 3.23: Application sidebar

3.3.11 Login

The user can log in the application. To stay able to escalate and expand the project, we can add in the future frequent finance news emails to be sent to our signed users.

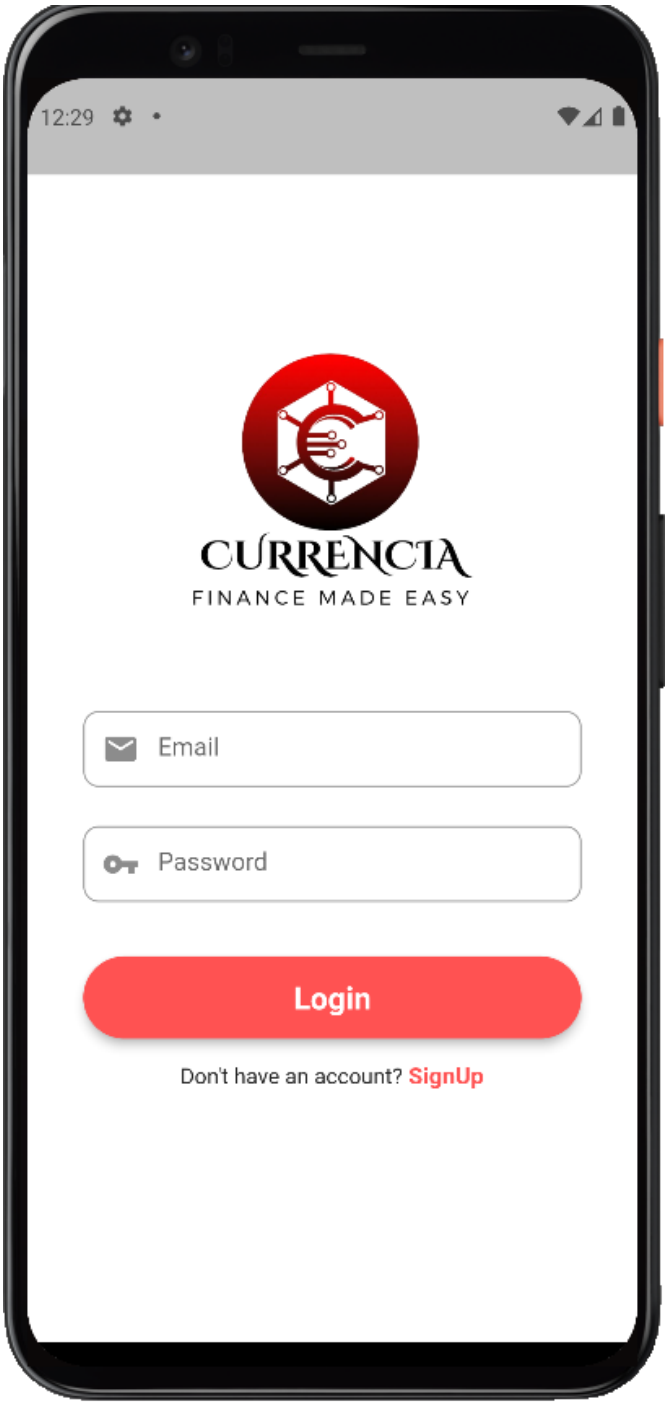

Figure 3.24: Login User Interface

3.3.12 Signup interface

The user can signup to have an account in our database. An account to use in order to log in or to receive mails or for premium members, for example.

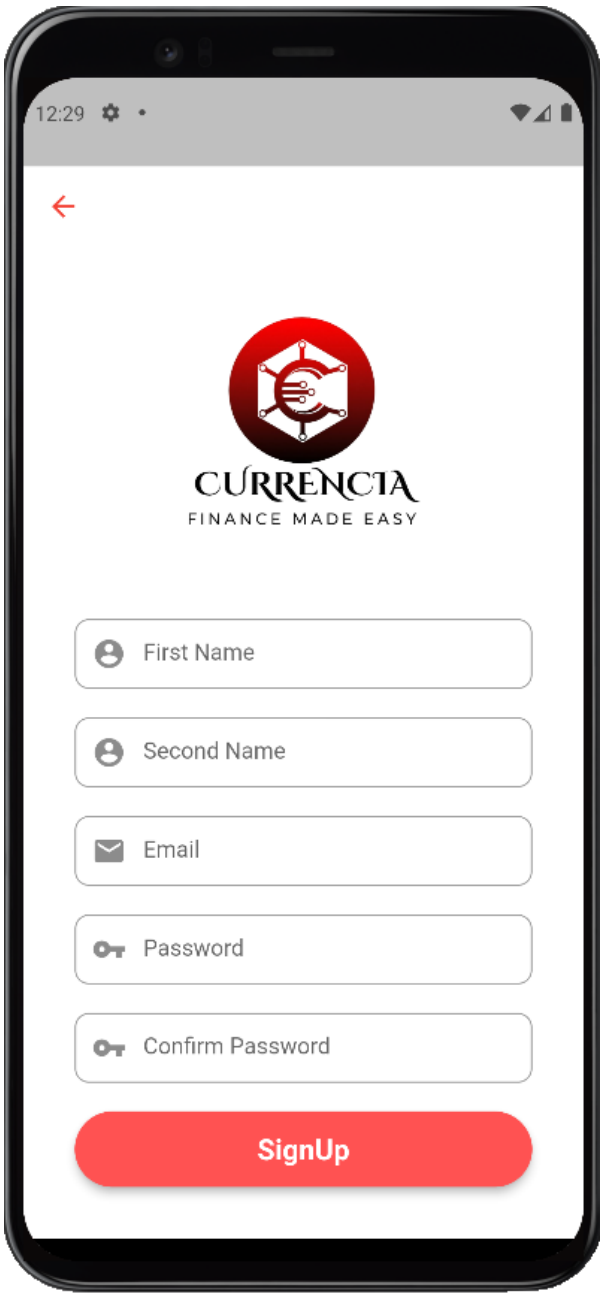

Figure 3.25: Signup interface for new users

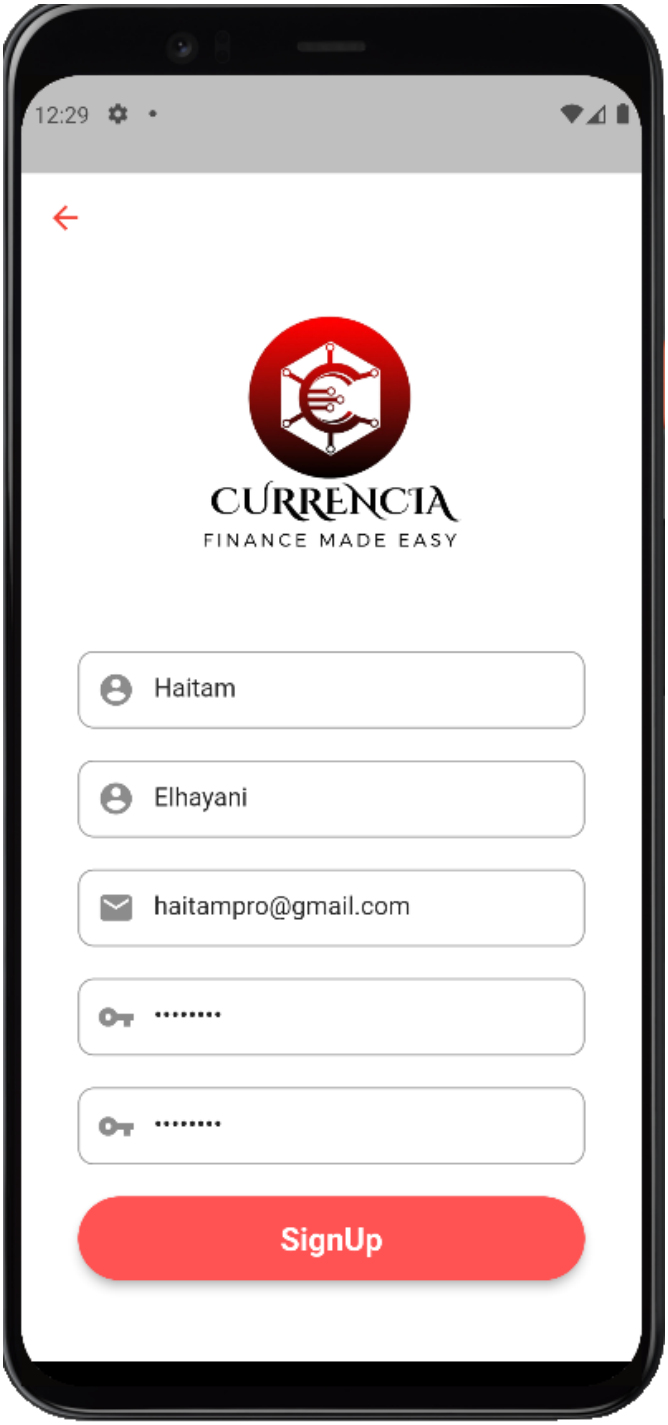

Figure 3.26: Example of a new user signup

3.3.13 Firebase database management

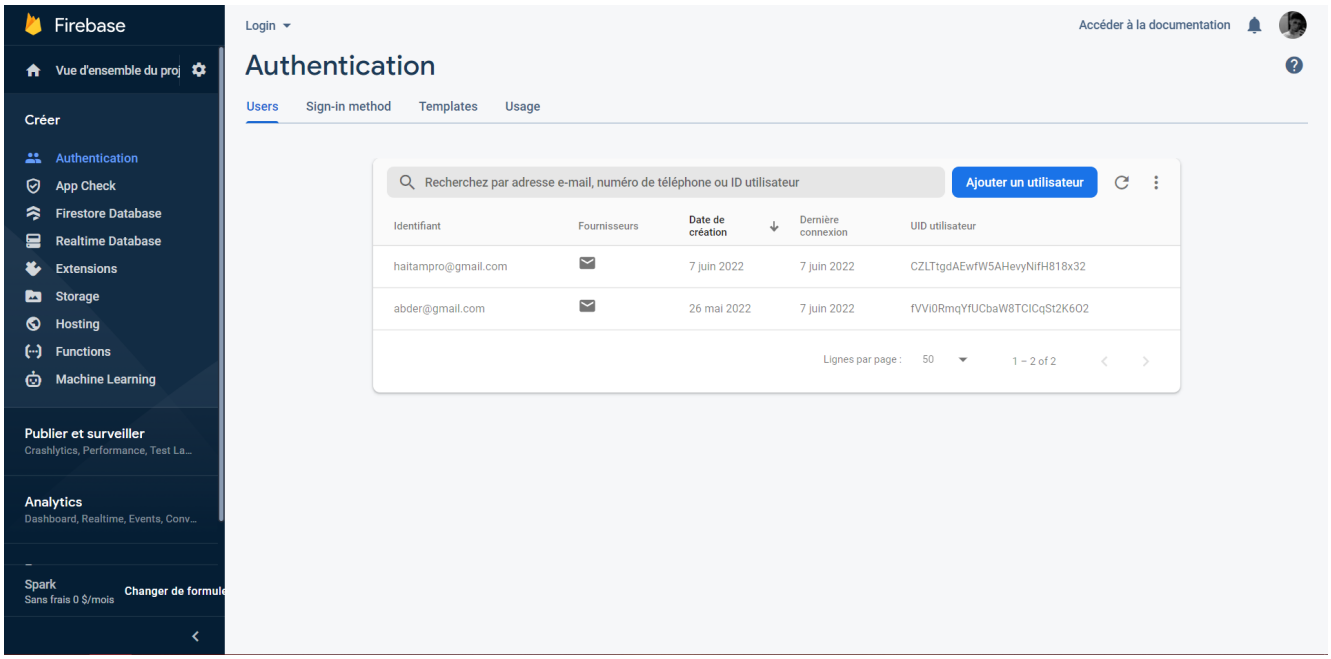

Figure 3.27: Users Database Authentication

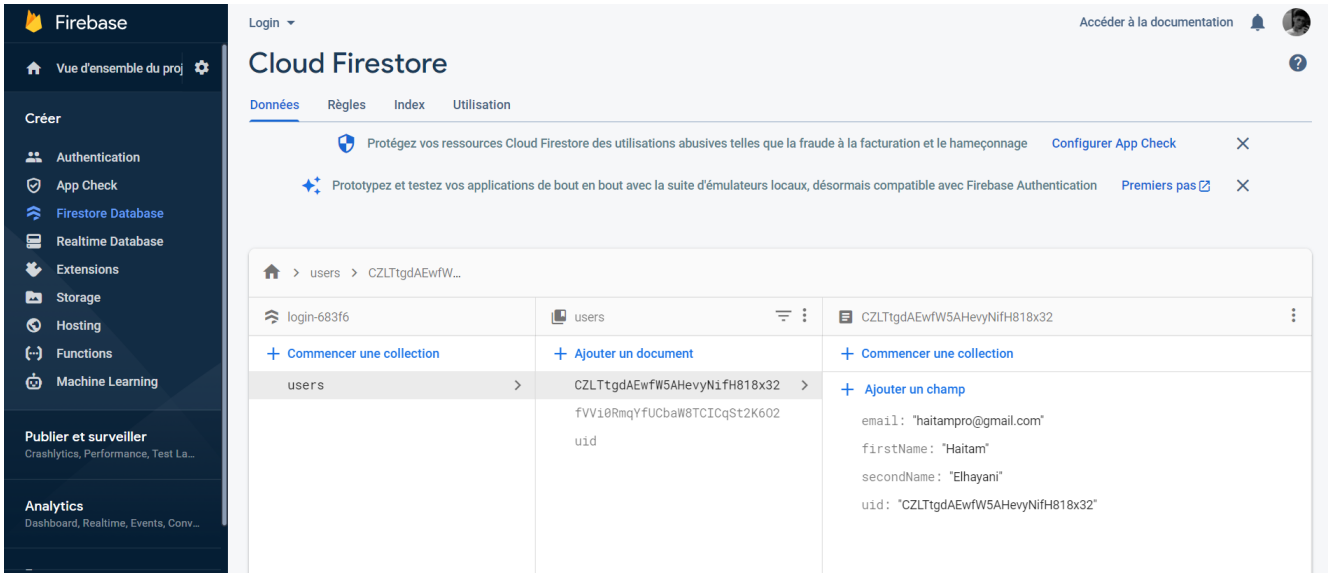

Figure 3.28: Cloud Firestore: Logged users details

3.3.14 Web/Desktop Currencia Build

As mentioned before, Flutter supports various operating systems. To illustrate: here's the web build on Google Chrome browser.

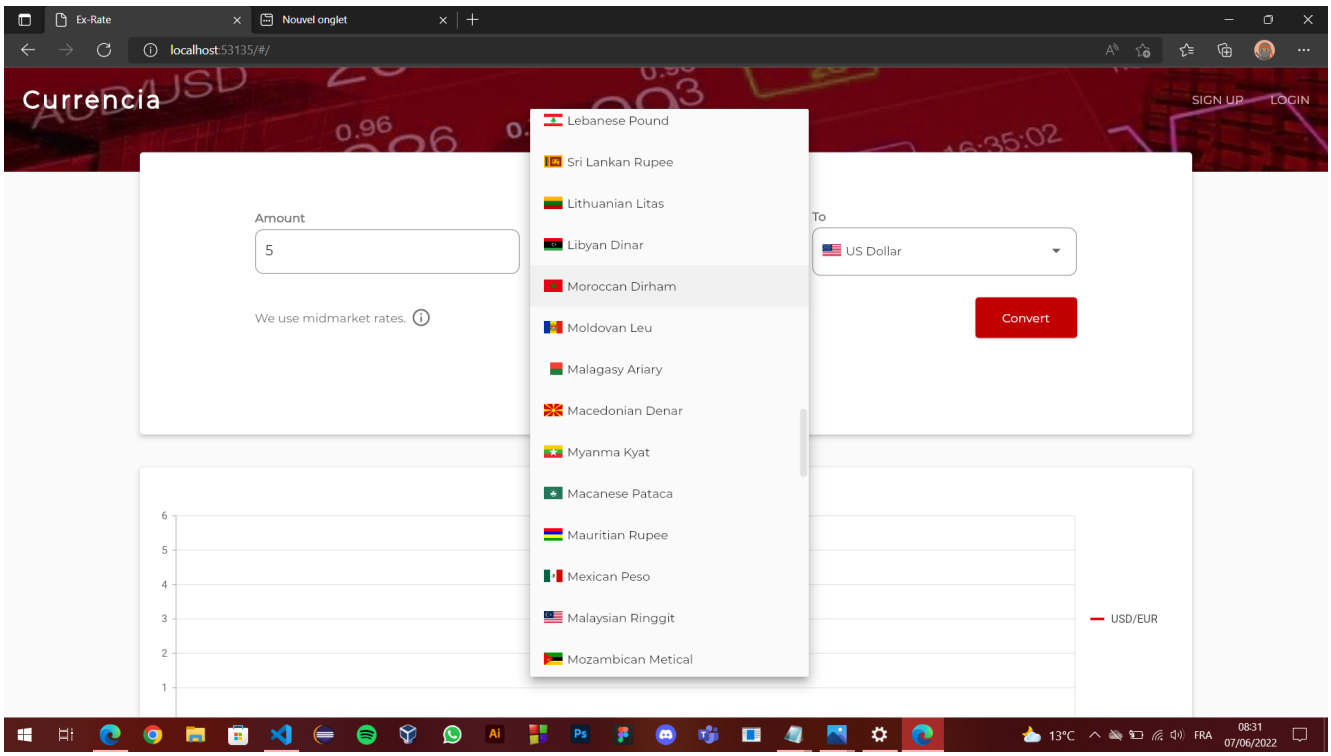

Figure 3.29: Currencia web build

3.4 Conclusion

With this last step, we concluded the realization of the project by implementing the analytical and conceptual studies already presented in the first chapter. We have presented the tools deployed for its development as well as the the different interfaces realized.

Overall Conclusion

In summary, we have presented the process we used to construct this application throughout this report, beginning with the presentation of the subject, specifications, and progressing to the section of the application that we developed. The application's conception, analysis, and final implementation.

The objective of our work was to design and implement a cross-platform application for currency conversion at current exchange rates, along with data charts and trending currencies. **Currencia** allows users to have fast and easy access to financial services on different operating systems.

In our Moroccan society, we barely find meaningful education when it comes to financial fields. Also, finance is an expertise that is necessary in every entreprise, startup, or company. Therefore, it's a great opportunity for us to work on a project that gathers IT and finance together.

Throughout this project, we were able to learn how to master Flutter based on the MVVM architecture.

From the study and design of an application to its implementation, this project allowed us to gain new knowledge that we had not gained during the year. Furthermore, it helped us to improve our cooperation, communication, and problem-solving skills as a result of issues that arise during the course of a project.

We are about giving value to finance and its passions, including us. Moreover, we are building an expandable project in which we can add many more features such as:

- Building profiles for signed users and have premium members for special services to help sponsor the application's evolution.

- Cryptocurrencies, NFTs, Decentralized Fintech wallets, and Blockchain-related services such as sending/receiving money.

- Trading charts and everything that goes with Forex and Stocks.

- Financial news through mailing our database users.

Bibliography

[1] Official Flutter documentation

Link: `https://docs.flutter.dev/`

[2] Official Dart documentation

Link: `https://dart.dev/guides/`

[3] Official Firebase documentation

Link: `https://firebase.google.com/docs/`

[4] Official Dart documentation

Link: `https://dart.dev/guides`

[5] freeCodeCamp's Flutter/Dart/Firebase Courses

Link: `https://freecodecamp.org/news/tag/flutter/`

[6] Official GitHub Documenation

Link: `https://docs.github.com/en`

[7] StackOverFlow

Link: `https://stackoverflow.com/`

[8] Meiller, Dieter, (2021), Modern App Development with Dart and Flutter 2: A Comprehensive Introduction to Flutter.

[9]Tyagi, Priyanka, (2021), Pragmatic Flutter: Building Cross-Platform Mobile Apps for Android, iOS, Web Desktop.

[10]Mike Katz, (2021), Flutter Apprentice: Learn to Build Cross-Platform Apps, 2nd Edition.