Republic of Tunisia
Ministry of Higher Education, Scientific Research and Information and Communication Technologies

Sousse University
Higher Institute of Applied Sciences and Technology of Sousse

Computer Engineering & Applied Mathematics Department

German commercial register documents mobile application SUMMER INTERNSHIP REPORT

presented at

Higher Institute of Applied Sciences and Technology of Sousse

by

Aslene OUAZE

Summary

This work was developed as part of a summer internship project which was achieved within the Glocal IT company. This project involves designing and developing a dynamic mobile application to enable customers to benefit from the German commercial register, locate their desired company, and obtain its coordinates and details.
Our application also gives the possibility of automatically receiving notifications about followed companies as well as buying specific documents about said companies.

Keywords
Mobile application, Android, IOS, Online commercial register, Clean Architecture, Flutter

Contents

List of Figures

List of Tables

General Introduction

The field of mobile development has evolved very well in recent years. years gave the increasing use of Smartphones. This area demonstrates multiple important investments thanks to the increasing number of interested developers in this field who offer an infinity of applications as useful than entertaining.

Nowadays, every possible product is available online even legal documents and sometimes private information and even classified documents.

These products are accessible everywhere and through multiple devices as long as that device is connected to the internet, such availability is provided thanks to the robust web and mobile applications.

That's why we wanted to create a similar mobile solution for individuals and enterprises who want to access information about commercial companies and we took the German people as a base point. Our application will provide multiple information and statistics about each company as well as provide purchasable documents about its commercial and legal state.

Our summer internship project entitled "German commercial register documents mobile application" concludes our summer training as a computer engineer.

The project was carried out over two months, within the company Glocal IT. This report summarizes the stages of realization of this project. Its purpose is to situate the context of the project, to describe the resulting application, the methods, and tools used as well as the results obtained.

This report follows the following organization:

The first chapter is entitled ”General framework of the project”, which is an introduction chapter presenting the host company, the problem, the solution proposed and the objectives of the project, a study of the existing and the process of development of our application.

The second chapter, ”Specification of needs”, is used to identify the actors of our application and then to specify the functional and non-functional needs. functionalities to which our application must respond, making it possible to identify its main features.

The third chapter, ”Design”, serves to describe the conceptual diagrams and the architecture applied to our proposed solution.

The fourth and final chapter, ”Achievement”, illustrates the achievement of our project through the presentation of the environment and the development tools as well as the visualization of the results of our work through the main application interfaces.

Finally, we end the report with a general conclusion in which we recapitulate the work carried out and we present the prospects.

Chapter 1

General framework of the project

Introduction

We begin this chapter with a general presentation of the organization of the reception. We will then detail the problem of our internship and the proposed solution, which will attempt to resolve the inconveniences that already exist. We will finish this chapter by describing the schedule of our internship through the Gantt chart as well as the development process.

1.1 *Presentation of the host organization*

Glocal IT Solutions is an IT development center based in Trittau in Germany, and recently created a branch in Mahdia, Tunisia, it specializes in the design and production of Web and Mobile Applications.

Its main partners are major accounts in Europe. Maturity and the competence of its teams as well as the synergy with its foreign partners can bring them complete satisfaction.

Glocal IT Solutions advises, designs, and develops tailor-made mobile and web solutions, adapted to the needs of any type of business.

Depending on the needs of its customers, it will offer them Web/Mobile solutions or native applications on the most popular platforms on the market: iOS (iPhone / iPad / iPod), Android. Customers can also entrust it to:

- Design and develop custom web solutions using modern frameworks (Angular, NodeJS, etc).
- Desing and develop custom mobile applications with the modern framework Flutter.
- Create and maintain powerful and robust API and microservices to support their solutions.
- Creation and integration of customers' services and adaptation to their business strategy.

Figure 1. *Host organization Glocal IT Solutions*

1.2 *Project presenation*

In this part, we put our work in its general context. first, we present the problem and the reasons that led Lordroid to suggest this topic. Second, we present the solution proposed to solve the problem. Finally, we will describe the objectives of this project.

1.2.1 Problem

We have found that people find it difficult to surf through multiple web pages to find specific relevant information about their wanted company, and the difficulty becomes even bigger when they try to get official or semi-official documents about it from a trustworthy source.

1.2.2 Proposed solution

We offer a mobile application for the online German Commercial Register. This application must also allow consulting precise details about German commercial companies, to purchase documents about companies that have it remotely. Our application must also allow us to automatically receive real-time notifications about followed companies.

1.2.3 Objectives

Ensure user satisfaction by ensuring:

Functional objectives:

1. Search company by its name, legal form, state, branch, capital, and zip code.
2. Follow the selected company to receive real notifications about it.
3. Purchase legal documents about the selected company.

Non-functional objectives:

1. Availability.
2. Ergonomics.
3. Reliability
4. Extensibility

Technical objectives:

1. Organization of the application according to the clean architecture.
2. Dependency injection with the Bloc framework.
3. Using the programming language ”Flutter”.

1.3 *Study of the existing*

In this part, we analyze and criticize the existing applications currently, through the following table propose a solution that solves their drawbacks.

Applications	Pros	Cons
contactdetails.info	- Huge variety of companies types - Includes companies from all around the world - Customers can add their own company	- Vague details about each company - Lack of cross-platform access (web only) - No access to business documents
business-yellowpages.com	- Possibility of adding a specific company to own favorites - Relatively deep information about each company - Possibility of sending a direct message to the company	- Non-Modern U/I - Lack of cross-platform access (web only) - No access to business documents
cybo.com	- Extremely deep and detailed information about each company - Statistics and advanced charts about each company's numbers - Possibility to rate and review each company	- Lack of cross-platform access (web only) - No access to business documents

Table 1. *Study of the existing*

1.4 *Development process*

A software development process is a set of related activities followed by a team led to the production of the software within the organization. It consists of a detailed plan describing how to develop, design, test, deploy and maintain the product.

1.4.1 Incremental development

In this context, we adopt the process of incremental development as an approach to the realization of our project. According to this process, the customer's needs are specialized, the software is globally designed, then the realization is done by an increment of functionalities.

Each increment is considered an executable part of the final system. These increments are successively integrated into the final product and at each stage, the software is tested, operated, and maintained as a whole.

Implement the software by increment makes it possible to take into account the risk analysis to facilitate the detection of errors at the earliest according to customer feedback and to reduce time and cost of production, which helps in the realization of software quality.

1.4.2 Provisional schedule of tasks

A Gantt chart is a graphical tool that represents the management of the project over time, which facilitates its implementation.

Indeed, the following figure represents the progress of activities and tasks performed throughout our project.

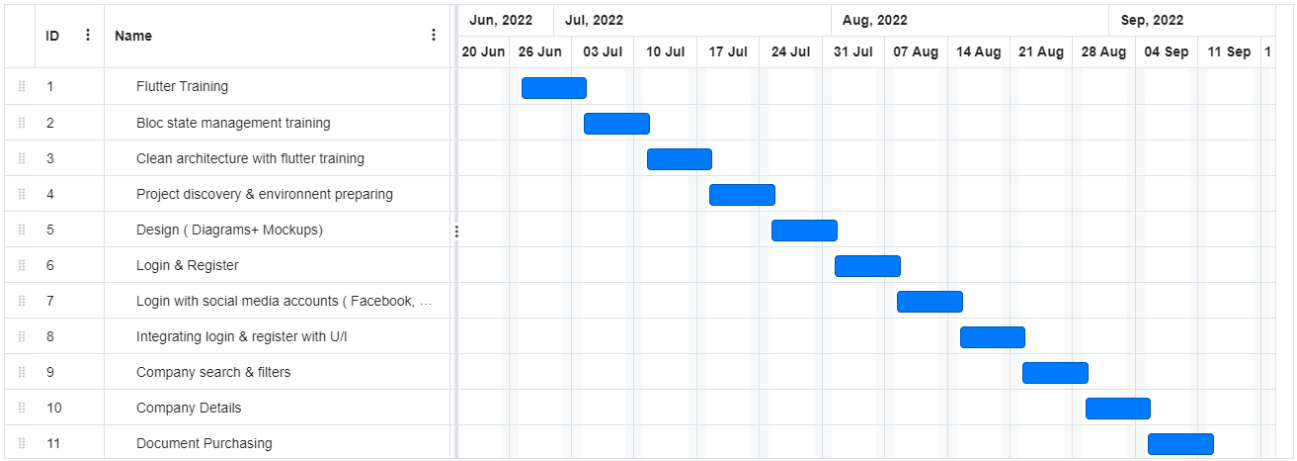

Figure 2. *Gantt diagram*

Conclusion

In this chapter, we have introduced the context of our project by representing the host organization in the first place. Secondly, we have cleared up the problem. Then, we described the proposed solution and the objectives to be achieved. Thirdly, we analyzed the existing applications. Fourthly, we have depicted the advancement of activities throughout the project according to the adopted development process. In the next chapter, we will specify the functional requirements and the non-functional needs.

Chapter 2

Specification of needs

Introduction

In this chapter, we will identify the actors, then we will specify the functional and non-functional needs that the proposed solution must meet. Finally, we present the use case diagrams explaining our application's main functionalities.

2.1 *Identification of actors*

An actor represents an external entity that interacts directly with the system. It can be either a human person or a system. We distinguish two types of actors, the main actor, and a secondary actor. Indeed, a principal actor obtains an observable result of the system while a secondary actor is asked for additional information.
In our application we have 4 actors:

Main actors

Customer:

The customer is the main user of our application who should be able to :

- Register to the application using e-mail

- Login using the Sofort-Handelsregister account
- Login using social media account (Facebook, Google, LinkedIn, Xing)
- Search for companies by name
- Filter search results by zip code, state, legal form, capital or branch
- Consult the wanted company's details

In case when the customer is logged in he will be also able to :

- Follow a specific company and receive notifications about its activity
- Purchase a specific company's documents
- Consult previously purchased documents and download them as PDF

Secondary actors

Sofort-Handelsregister remote API

This actor is a service, previously developed and deployed in a remote server by the host organization, interactable through our mobile application, and it's responsible for the:

- Registering the client information in the database
- Providing login token on successful login
- Connect the customer's social media account with the database
- Providing the latest version of the company list
- Providing updated details and statistics about each company
- Providing documents download links
- Sending notifications to customers about their tracked companies

2.2 *Functional Needs*

Functional needs are expressed by the user of the application which makes it possible to identify the functionalities of this application.
In our cases, the functional needs are:

- Register
- Login
- Account verification
- Company search
- Filter search results by zip code, state, legal form, capital, or branch
- Consult company details and statistics
- Follow the company and receive notifications about it
- Purchase company documents
- Consult purchased documents and save them as PDF

2.3 *Non-functional Needs*

Non-functional requirements represent the characteristics of the system. They relate to the constraints to be taken into consideration to set up an adequate solution.
For our application, the non-functional requirements are:

- Ergonomics:

  The interfaces of our application give the user what he is looking for quickly and easily.

- Extensibility:

  The application can be extended by adding other scenarios and other features.

- Performance:

  Response time to user queries should be reasonable.

- Availablity:

  Since the application relies heavily on its corresponding remote API, it must cache the data when it's possible to use them when the API is not available

- Adaptability:

  Our application must adapt to different platforms (Android/IOS) and screen sizes (Smartphone / Tablet).

- Security:

  All users must authenticate before purchasing documents.

2.4 *Use case diagrams*

In this section, we will highlight the system's functionalities to be from the functional needs mentioned above based on the UML(Unified Modeling Language) diagrams which group together all the use cases of the system.

2.4.1 Global use case diagram

The following figure illustrates the global use case diagram of our application

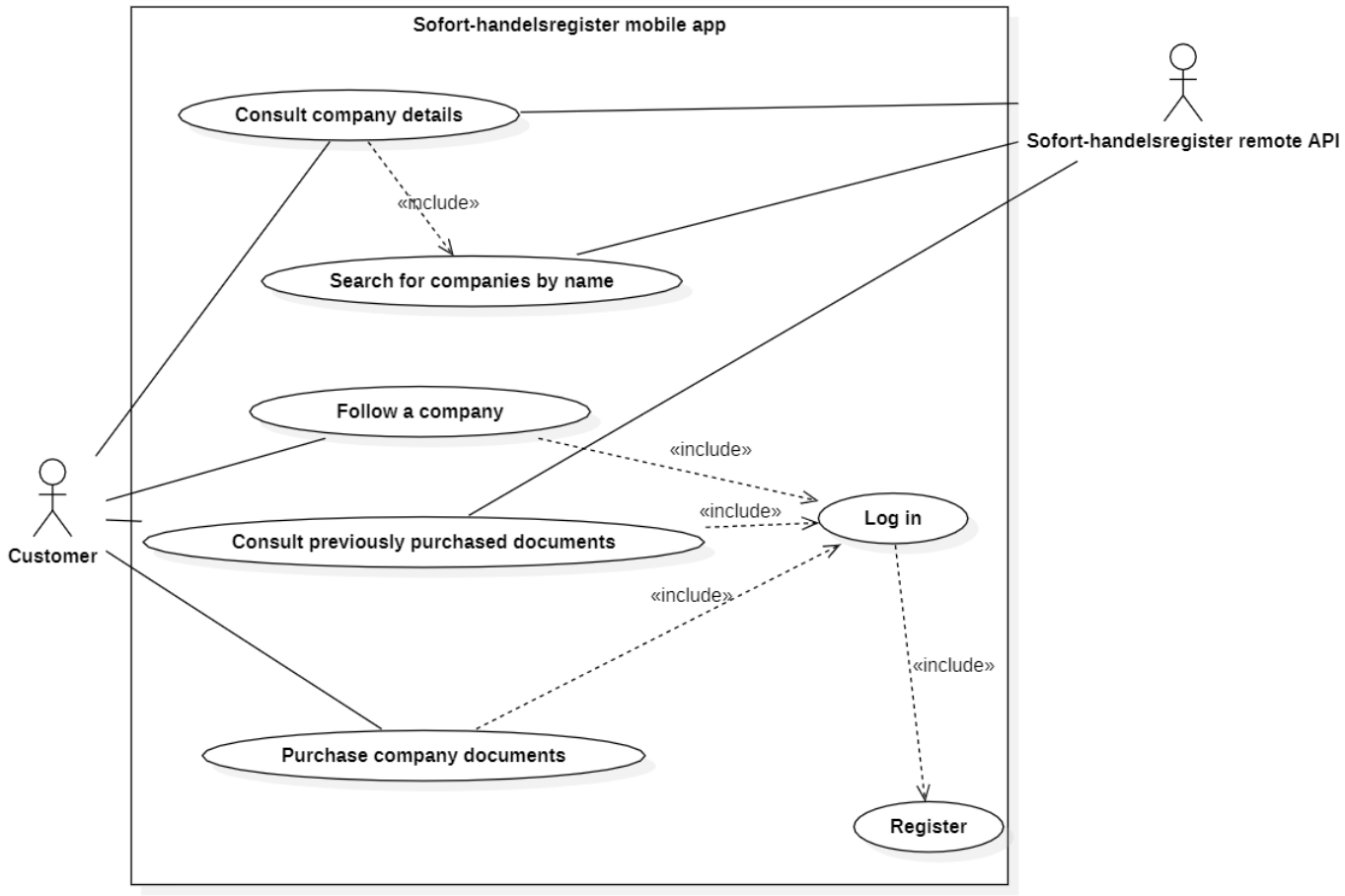

Figure 3. *Global use case diagram*

2.4.2 Use case refinement

In this section, we will detail the main use cases.

2.4.2.1 Register use case refinement

In our application, a customer must register before benefiting from the service of purchasing a document through our application.

The following figure shows the Register use case diagram.

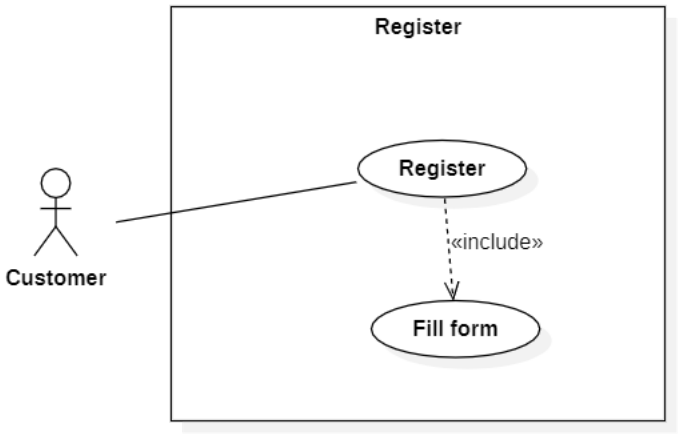

Figure 4. *Register use case diagram*

The following table details the tasks to be performed by the customer to register to our app.

Summary	
Title	Register
Objectif	Allowing the user to register to be able to purchase documents and follow companies through the application
Actors	Customer
Description of sequences	
Pre-condition	User should start the application and go to the registration interface.
Post-condition	The user is registered and his contact details are saved in the database.
Normal scenario	1. The user accesses the registration interface. 2. The user completes the form and complies. 3. Registration is completed.
Alternative scenario	1. Empty fields: The system sends an error message: You must complete all fields 2. The email entered or the password entered is not valid: The system displays an error message describing the validation conditions for these fields. 3. The email is already in use: An alert is sent by the system
Non functional constraints	1. The interface must be ergonomic. 2. Error messages should be understandable and clear.

Table 2. *Register use case*

2.4.2.2 Login use case refinement

In our application, a customer must log in before benefiting from the service of purchasing a document through our application and tracking a specific company through notifications.
The following figure shows the Login use case diagram.

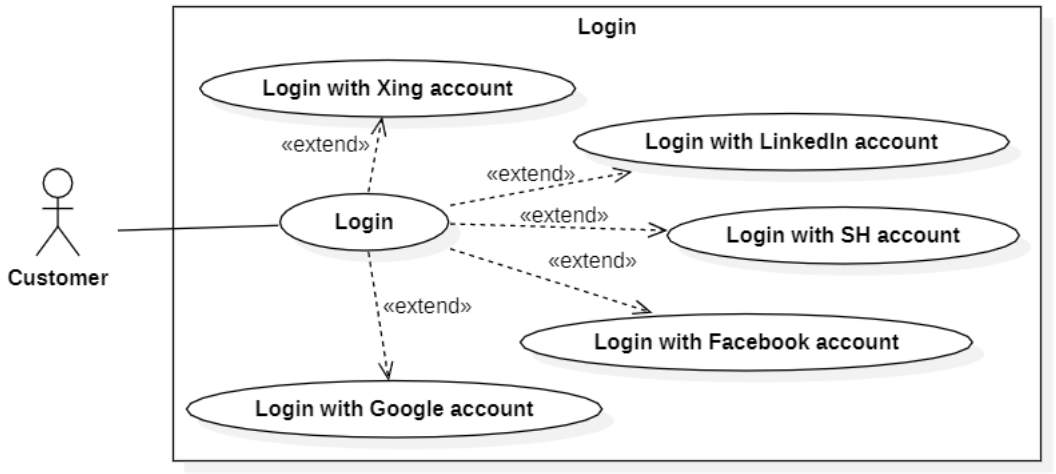

Figure 5. *Login use case diagram*

The following table details the tasks to be performed by the customer to log in to our app.

Summary	
Title	Login
Objectif	Allowing the user to login to be able to purchase documents and follow companies through the application
Actors	Customer
Description of sequences	
Pre-condition	User should start the application and go to the log in interface.
Post-condition	The user is logged in and his session is cached in the application.
Normal scenario	1. The user accesses the login interface. 2. The user completes the form and complies or he clicks on one of the social media buttons and completes the external login process. 3. Login is completed. 4. The login token is cached within the application.
Alternative scenario	1. Empty fields: The system sends an error message: You must complete all fields 2. The email entered or the password entered is not valid: The system displays an error message describing the validation conditions for these fields.
Non functional constraints	1. The interface must be ergonomic. 2. Error messages should be understandable and clear.

Table 3. *Login use case*

2.4.2.3 Company search use case refinement

The following figure illustrates the use case diagram ”Company Search”

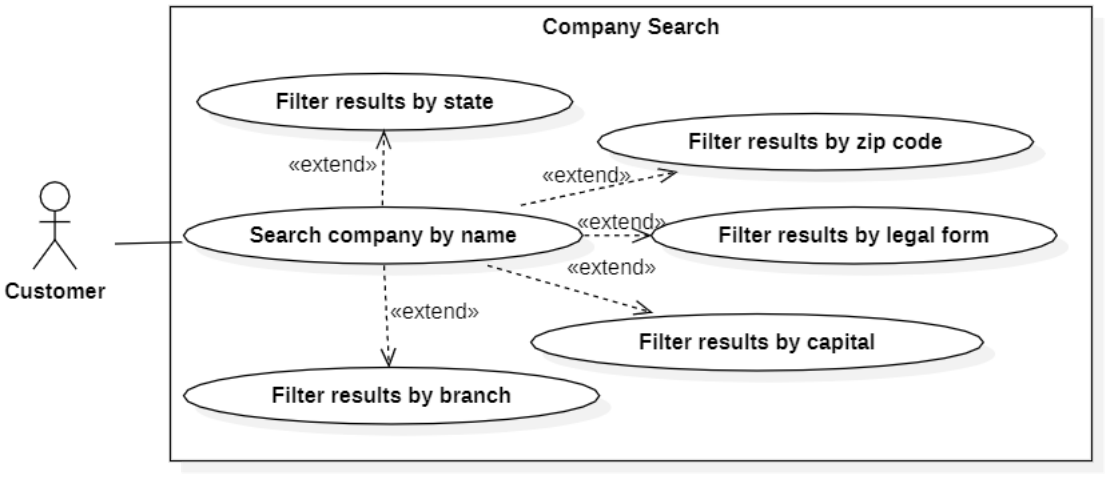

Figure 6. *Company Search use case diagram*

The following table details the tasks to be performed by the customer to search for companies within our app.

Summary	
Title	Company Search
Objectif	Allowing the user to search for specific companies through the application
Actors	Customer
Description of sequences	
Pre-condition	User should start the application and go to the search company interface.
Post-condition	The user chooses a company from the list of results and consults details about it.
Normal scenario	1. The user accesses the company search interface. 2. The user types the company name or the first letters of it. 3. The user may choose a specific state, a legal form, or a branch to filter the results. 4. The user may also type zip code or capital as a filter. 5. The user consults details about companies from the list of results.
Alternative scenario	1. No results: The system sends a message: No results were found with the given parameters 2. No internet: The system sends an error message: No internet connection found.
Non functional constraints	1. The interface must be ergonomic. 2. Error messages should be understandable and clear.

Table 4. *Company Search*

2.4.2.4 Follow Company use case refinement

The following figure illustrates the use case diagram ”Follow Company”

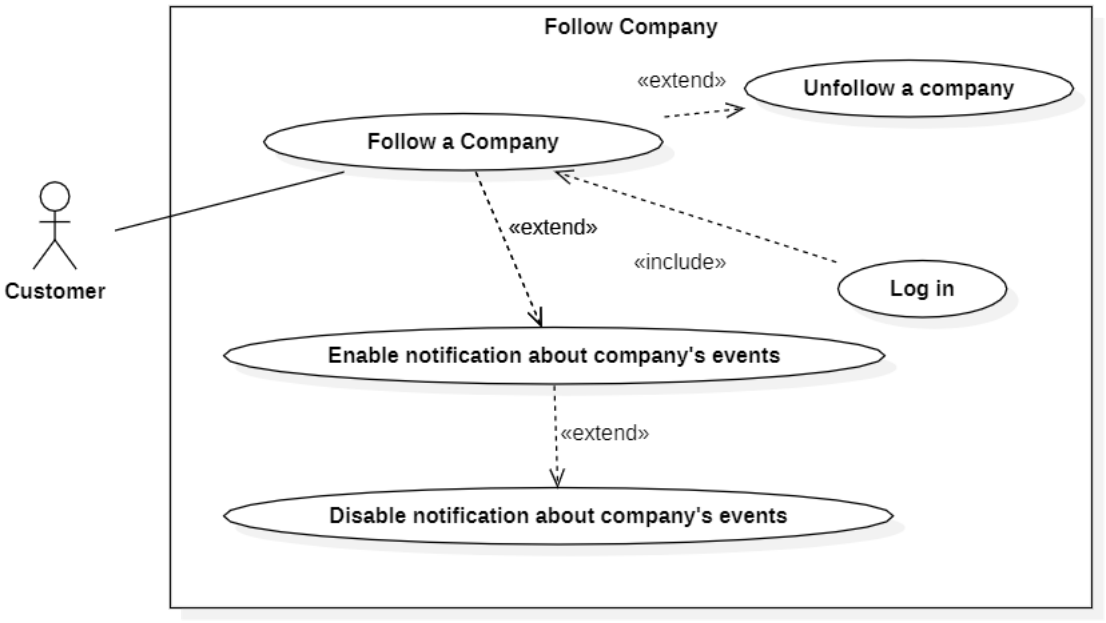

Figure 7. *Follow Company use case diagram*

The following table details the tasks to be performed by the customer to follow a company within our app.

Summary	
Title	Follow Company
Objectif	Allowing the user to follow specific companies to receive notifications about its events through the application
Actors	Customer
Description of sequences	
Pre-condition	User should start the application, be logged in, search for a specific company and access the consult company details interface.
Post-condition	The tracked company is added to the user's follow list.
Normal scenario	1. The user accesses the consult company interface. 2. The user taps the follow button. 3. The user may choose to unfollow the company or enable event notifications. 4. Upon activating notifications, the user will receive notifications through the application when a notice is published about said company. 5. The user can tap the notification to access the notice.
Alternative scenario	1. Not logged in: The system sends an error message: You must be logged in to follow a company.
Non functional constraints	1. The notification must be as close to real-time as possible

Table 5. *Follow Company use case*

2.4.2.5 Purchase Documents use case refinement

The following figure illustrates the use case diagram ”Purchase Documents”

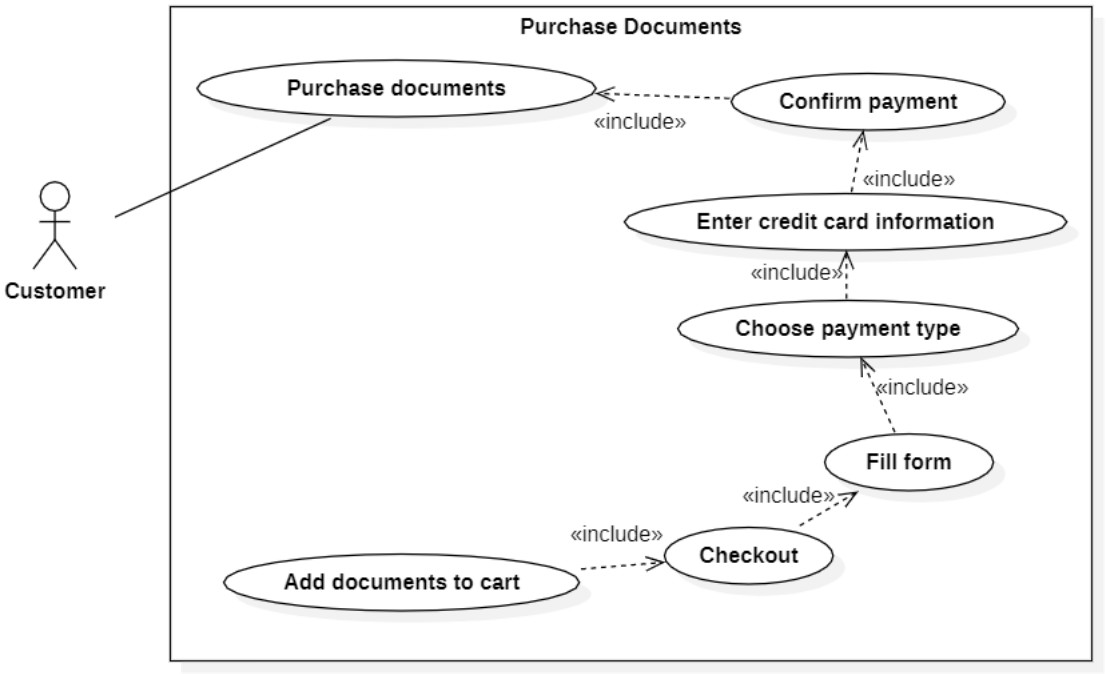

Figure 8. *Purchase Documents use case diagram*

The following table details the tasks to be performed by the customer to purchase documents within our app.

Summary	
Title	Purchase Documents
Objectif	Allowing the user to purchase official documents about the consulted company, if they are available.
Actors	Customer
Description of sequences	
Pre-condition	User should start the application, be logged in, search for a specific company, access the consult company details interface and scroll to the documents section.
Post-condition	The purchased documents are added to their user's purchased documents interface where he can download them on demand.
Normal scenario	1. The user accesses the consult company interface. 2. The user taps the wanted documents. 3. The documents will be added to the shopping cart 4. Upon checkout, the user will be redirected to the payment interface 5. The user fills in the contact information form 6. The user chooses the payment method 7. The user enters credit card details 8. The user confirms the payment 9. Purchase is completed
Alternative scenario	1. Not logged in: The system sends an error message: You must be logged in to purchase documents.
Non functional constraints	1. The interface must be ergonomic. 2. Error messages should be understandable and clear. 3. The payment must be as secure as possible

Table 6. *Purchase Documents use case*

Conclusion

In this chapter, we have described the specification phases of the needs of our developed application to identify the different actors as well as the features and services that our application must provide.

We have detailed these features with use case diagrams. The next chapter will be devoted to the design phase.

Chapter 3

Design

Introduction

After identifying the functional and non-functional needs and the main functionalities of our application. We begin this part of the conceptual study by describing the general architecture of our system and its detailed internal modeling through class and sequence diagrams.

3.1 *Global Architecture*

In this section, we will give an overview of the definition of the the architectural pattern was chosen to model our application and we will detail the projection of this pattern on our application.

3.1.1 Definition of the Clean Architecture pattern

Clean architecture is a software design philosophy proposed by Robert C. Martin, aka Uncle Bob. This architecture is based on:

- The SOLID principles from which derive the principle of inversion of control and the principle of dependency injection (a framework external to the application creates the components of the application and injects the dependencies between these components

- The principle of separation of ”business rules” (independent of the applications) and the ”application rules”.
- The principle of independence from the core of the application of necessary technologies for its operation.

Thanks to these principles, the components of the application become easy to manage, test, modify and reuse.
The clean architecture pattern is divided into 4 layers (upcoming figure) whose meaning dependency is incoming:

- **Entities**

  Entities that represent application domain objects with their methods (Business Rules).

- **Use Cases**

  This layer represents the logic of the application, i.e. the rules specific to an application (Application Rules).

- **Inerfaces / Adapters**

  This layer represents the interface between systems external to the application (framework, databases, UI...) and the core of the application (entities + use case). This layer allows formatting, structuring, and adaptation of incoming and outgoing data.

- **External technologies**

  This layer represents the layer external to the application grouping the technologies necessary for its operation (APIs, frameworks, Databases, UI, network, drivers, devices, etc.).

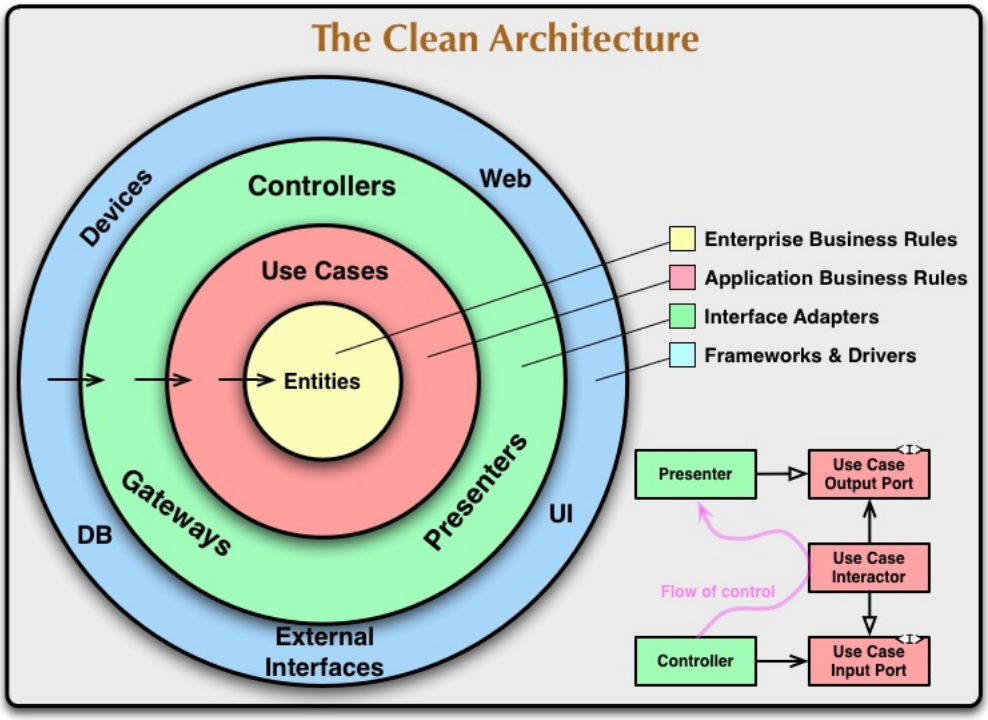

Figure 9. *Clean Architecture*

3.1.2 Application of the ”Clean Architecture” pattern

We explain in this paragraph the details of the projection of the pattern Clean Architecture on our app.

The following figure represents the class diagram which illustrates the binding between entities (classes). Indeed, this diagram translates the content of the **Entities** layer.

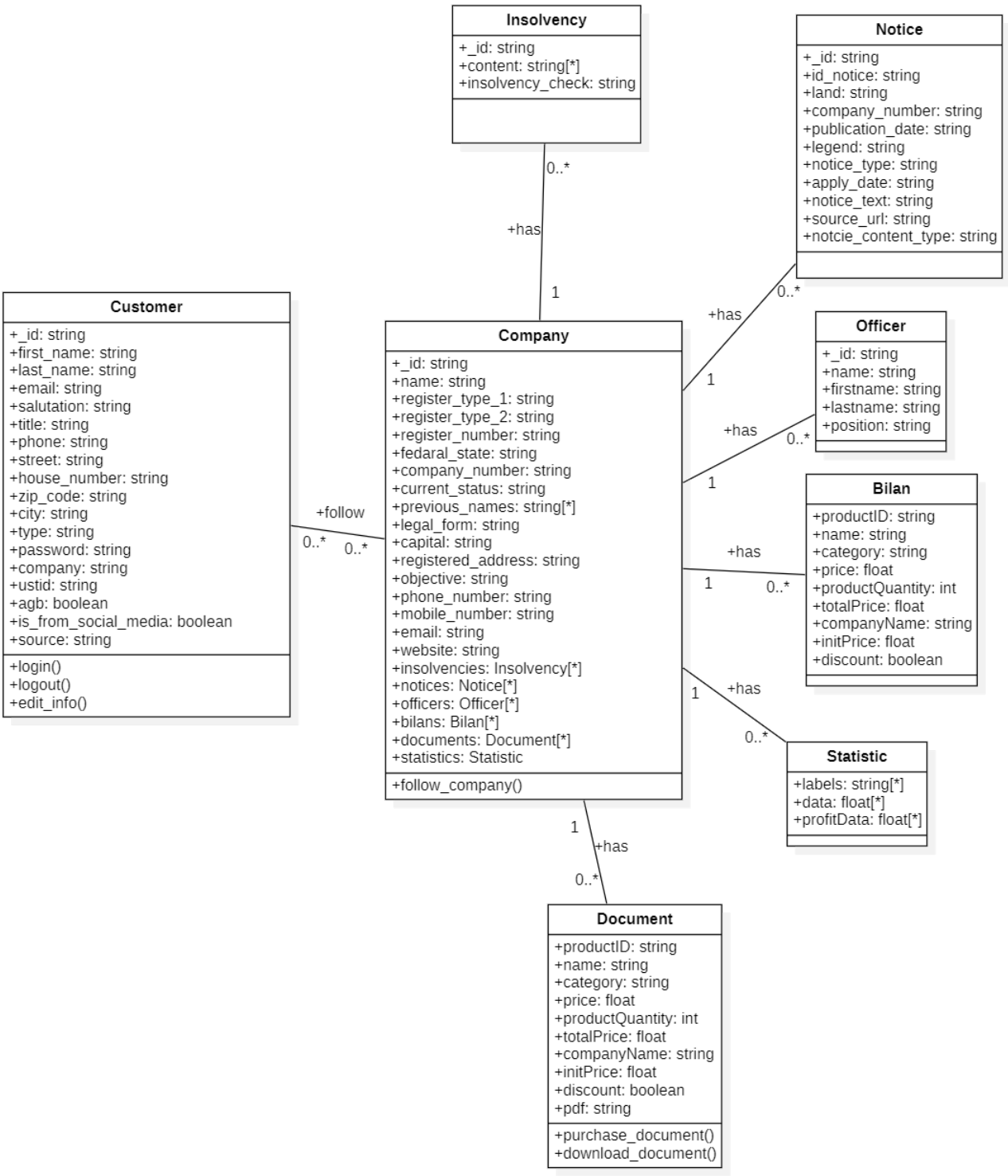

Figure 10. *Class diagram representing the ”Entities” layer*

Description of classes representing our entities:

- **Customer:**

  This class represents our application's customers who may log in, log out, and edit their account information. This class's attributes are:

  - _ id: the identifier of the customer
  - first_ name: the first name of the customer
  - last_ name: the last name of the customer
  - email: the email of the customer
  - salutation: the salutation of the customer (Herr, Frau)
  - title: the title of the customer (Prof., Dr.)
  - phone: the phone number of the customer
  - street: the street part of the customer's address
  - house_ number: the house number part of the customer's address
  - zip_ code: the zip code part of the customer's address
  - city: the city part of the customer's address
  - password: the password of the customer
  - company: the company of the customer
  - ustid: the value-added tax identification number (VAT ID) of the customer's company if he has one
  - agb: boolean variable presenting if the customer has accepted the agreements of the platform
  - is_ from_ social_ media: boolean variable presenting if the customer's account was created using a social media account
  - source: the social media of the customer account (Facebook, Google, LinkedIn, Xing, empty value)

- **Company:**

  This class represents the companies in our application that can be followed by the customer. This class's attributes are:

  - _ id: the identifier of the company
  - name: the latest name of the company
  - register_ type_ 1: the first part of the company's German type
  - register_ type_ 2: the second part of the company's German type
  - register_ number: the German commercial register number of the company
  - federal_ state: the German federal state name of the company
  - company_ number: the number of the company in our database
  - current_ status: the current status of the company (active, shutdown)
  - prevoius_ names: list of the company's previous names
  - legal_ form: the legal form of the company (GmbH, Limited & Co. KG, etc)
  - capital: the capital of the company in euros
  - registerd_ address: the latest address of the company
  - objective: the objective of the company
  - phone_ number: the latest phone number of the company
  - mobile_ number: the latest mobile phone number of the company
  - email: the latest email of the company
  - website: the latest website of the company
  - insolvencies: list of the company's insolvencies notices
  - notices: list of the company's change notices
  - officers: list of the company's office workers and individuals
  - bilans: list of the company's balance sheets
  - documents: list of the company's available purchasable documents

- statistics: visualization data of the company's business performance

- **Insolvency**

  This class presents the insolvencies notices in our application. This class's attributes are:

  - _ id: the identifier of the company's insolvencies
  - content: the list of notices from the company
  - insolvency_ check: variable representing if the company has a bad record in insolvencies (positive, negative)

- **Notice**

  This class presents the change notices in our application. This class's attributes are:

  - _ id: the identifier of the notice
  - id_ notice: the identifier of the notice regarding each company
  - land: the location of the release of the notice
  - company_ number: the number of the company owning the notice
  - publication_ date: the date of publication of the notice
  - legend: the label of the notice
  - notice_ type: the type of the notice
  - apply_ date: the date of application of the notice
  - notice_ text: the content of the notice
  - source_ url: the source of the notice
  - notice_ content_ type: the type content of the notice

- **Officer**

  This class presents the officer(office worker/individuals) in our application. This class's attributes are:

  - _ id: the identifier of the officer
  - name: the full name of the officer

- firstname: the first name of the officer
- lastname: the last name of the officer
- position: the position of the officer

- **Bilan**

  This class presents the balance sheets in our application. This class's attributes are:

  - productID: the identifier of the sheet
  - name: the name of the sheet
  - category: the category of the sheet
  - price: the price of the sheet
  - productQuantity: the available quantity of the sheet
  - totalPrice: the full price of the sheet
  - companyName: the name of the company owning the sheet
  - initPrice: the initial price of the sheet
  - discount: boolean variable presenting if the sheet is on discount

- **Statistic**

  This class presents the statistics data in our application. This class's attributes are:

  - labels: the values on the X axis on the graph (usually year values)
  - data: the values on the Y axis on the graph (usually turnover values)
  - profitData: the data for profit values on the graph

- **Documents**

  This class presents documents in our application, the customer can purchase these documents and download them as pdf files once purchased. This class's attributes are:

  - productID: the identifier of the document
  - name: the name of the document

- category: the category of the document
- price: the price of the document
- productQuantity: the available quantity of the document
- totalPrice: the full price of the document
- companyName: the name of the company owning the document
- initPrice: the initial price of the document
- discount: boolean variable presenting if the document is on discount
- pdf: the link to the pdf file corresponding to the document

The classes we have defined in the ”Entities” layer are related to the german business field, employable in various types of commercial applications, and independent of our application. Thus, we made the principle of independence between the business rules and the application rules.

For the rest of the layers of the Clean architecture pattern, we will describe the dependencies between these layers by giving an example of a connection between a set of classes belonging to these layers. The following figure represents the general dependence in our case between the layers of the clean architecture pattern.

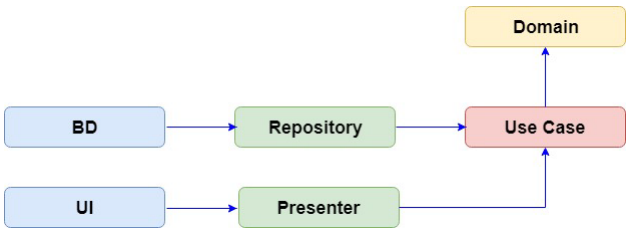

Figure 11. *General dependency diagram*

- The domain layer (Entities) is independent of all layers.
- The application layer (Usecase) depends on the domain layer and is independent of other layers. This layer prepares responses to user queries using entities.
- The adapters interface layer depends on the application(Usecase) layer. This layer transmits the user request and the necessary data to its processing at the Use Cases layer and retrieves the response and structure for the User Interface component.

- The user interface (UI) component of the ”External technologies” layer depends on the interface layer adapters through the Presenter an n pass the user’s request to it and retrieves the response.
- The database component of the ”External technologies” layer allows the Use Case layer to retrieve data through a Gateway(Repository) belonging to the Interface Adapter layer.

The following figure shows an example of a detailed dependency between layers of the clean architecture pattern in the case of our application. These dependencies link a set of classes belonging to the different layers. These classes realize the example of the functionality of displaying the list of users.

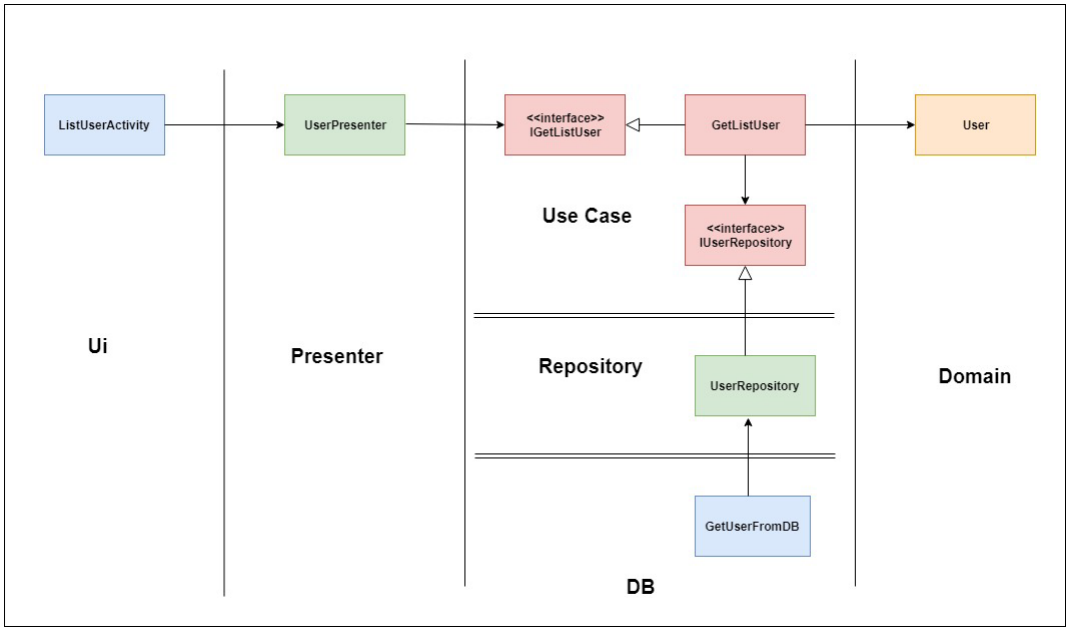

Figure 12. *Detailed dependency diagram in case of functionality print the list of users*

According to the figure, the Interface Adapter layer (UserPresenter and UserRepository) communicates (passing and retrieving data) with the Use layer boxes by implementing or instantiating its IGetListUser and IUser- Repository. These interfaces represent the ”ports” of the Use Cases layer. This example shows how we realized the principle of independence from the core of the application of the technologies used. No instance of the adapters interface layer or outer layer exists in the app’s core.

3.2 *Dynamic view of the application*

In this section, we will describe the internal dynamic of our application using sequence diagrams and activity diagrams

3.2.1 Detailed sequence diagram of the ”register” use case

The following figure illustrates the detailed sequence diagram of the use case ”register”. This use case starts with opening a registration interface for the user. Then, the user enters his coordinates and confirms them. The system displays an error message if the email already exists otherwise it redirects the user to the home page.

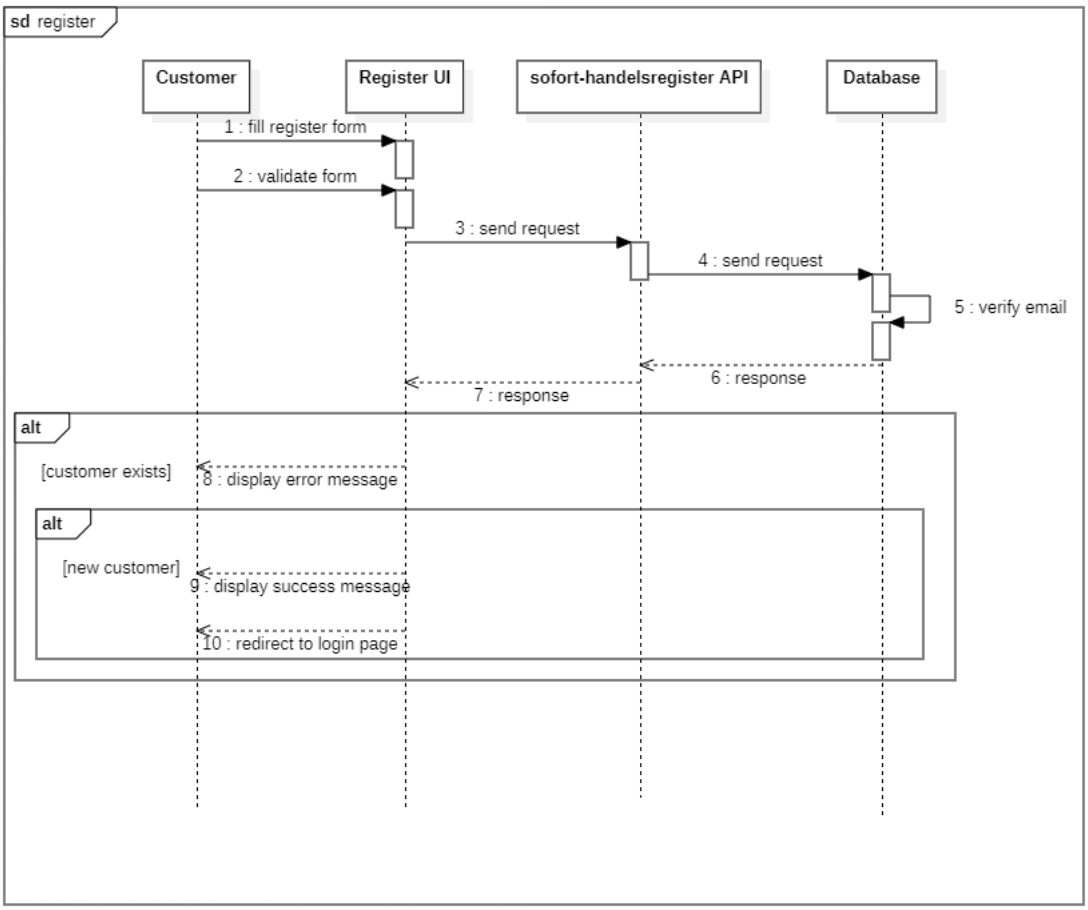

Figure 13. *Detailed sequence diagram of the ”register” use case*

3.2.2 Detailed sequence diagram of the ”login” use case

The following figure illustrates the detailed sequence diagram of the use case ”login”. This use case starts with opening a login interface for the user. Then, the user enters his email and password then confirms them. The system displays an error message if the email or the password is incorrect otherwise it redirects the user to the home page. the user may also choose to log in using a social media account (Google, Facebook, LinkedIn, Xing), by taping the dedicated button to each social media which will redirect him to the interface provided by that specific social media API, the user will be redirected to the home interface upon successful social media login.

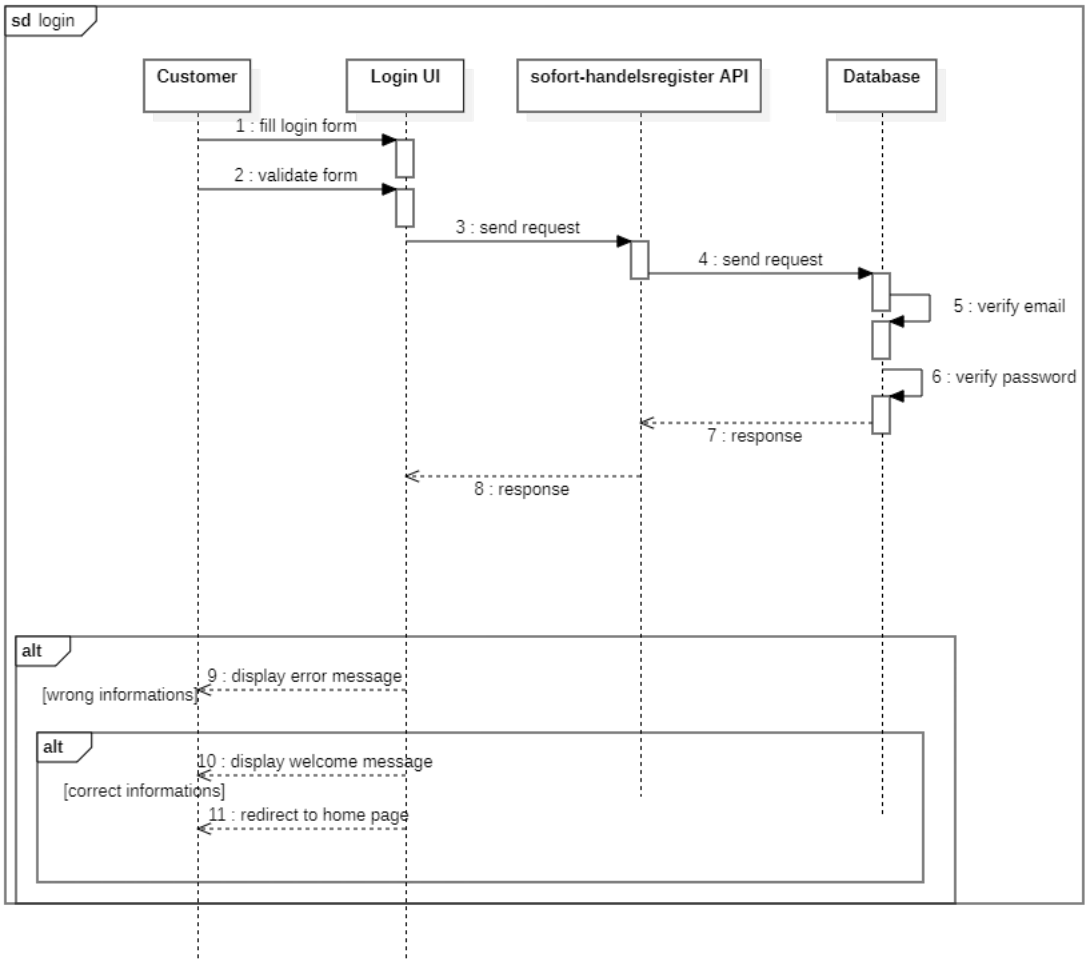

Figure 14. *Detailed sequence diagram of the ”login” use case*

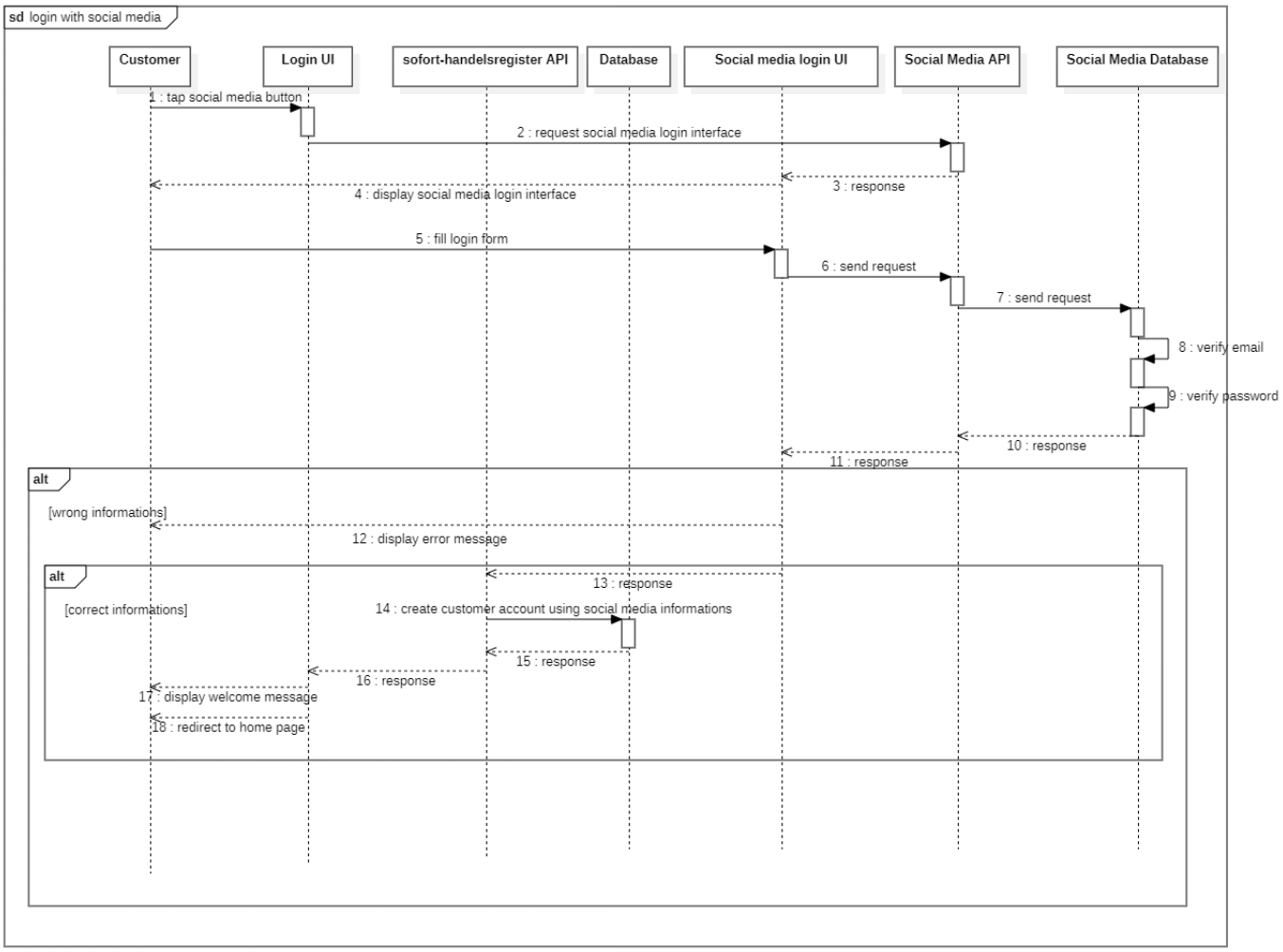

Figure 15. *Detailed sequence diagram of the ”login with social media” use case*

3.2.3 Detailed sequence diagram of the ”search company” use case

The following figure illustrates the detailed sequence diagram of the use case ”search company”. This use case starts with opening an interface for the company’s search for the user. Then the user enters the name of the wanted company’s name or the first letters of its name, he may also add some filter to the results using a different interface (by state, zip code, legal form, capital, branch), upon confirming his request the interface will prompt an interactive list of results where the user may tap one of them to access more detailed information about it.

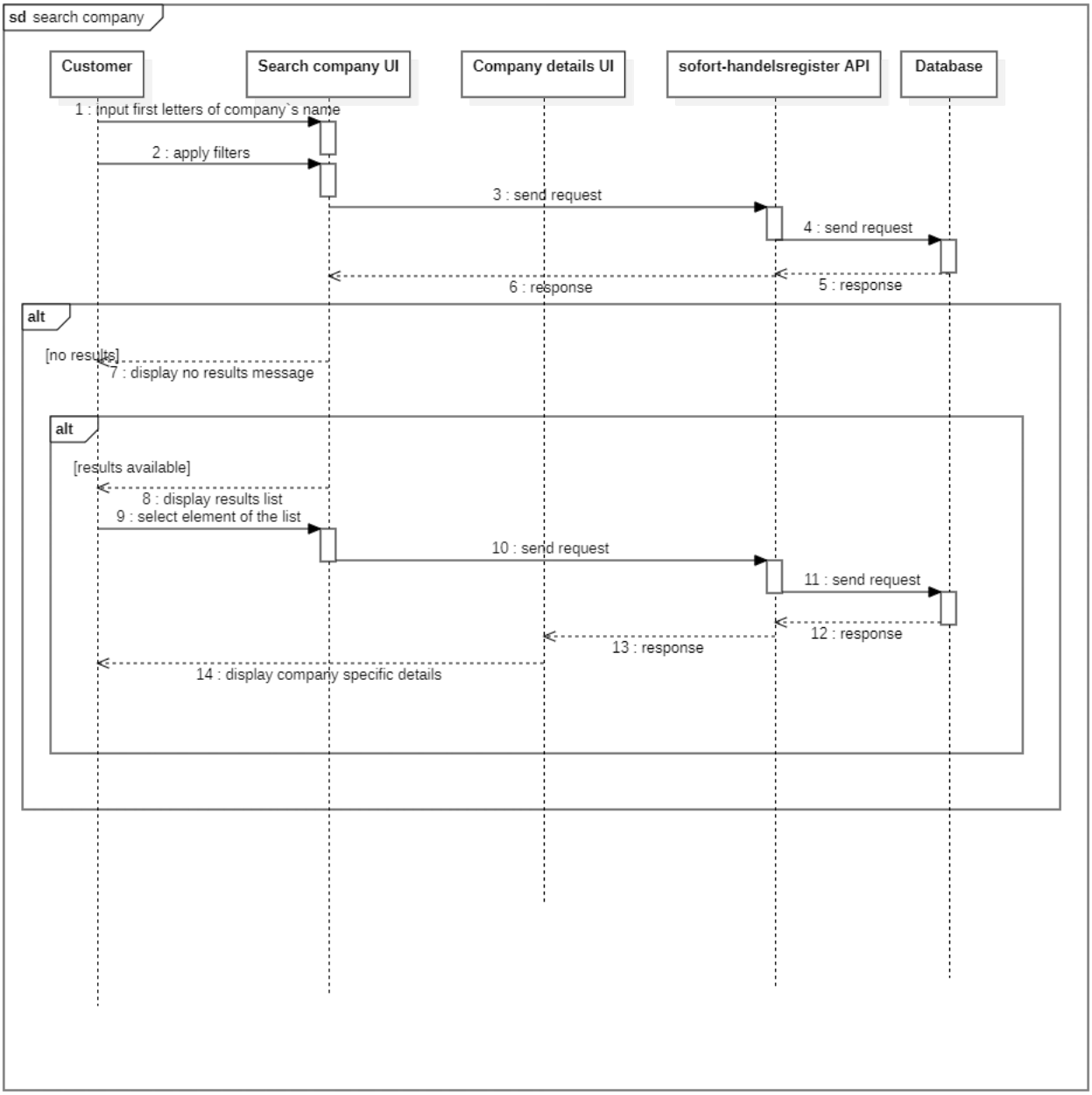

Figure 16. *Detailed sequence diagram of the ”search company” use case*

3.2.4 Detailed sequence diagram of the ”follow company” use case

The following figure illustrates the detailed sequence diagram of the use case ”follow company”. This use case starts with opening an interface containing specific company details for the user. Then the user may tap the follow button to add that company to his tracked companies list and start receiving notifications about it, these notifications may be deactivated later in the tracked companies interface.

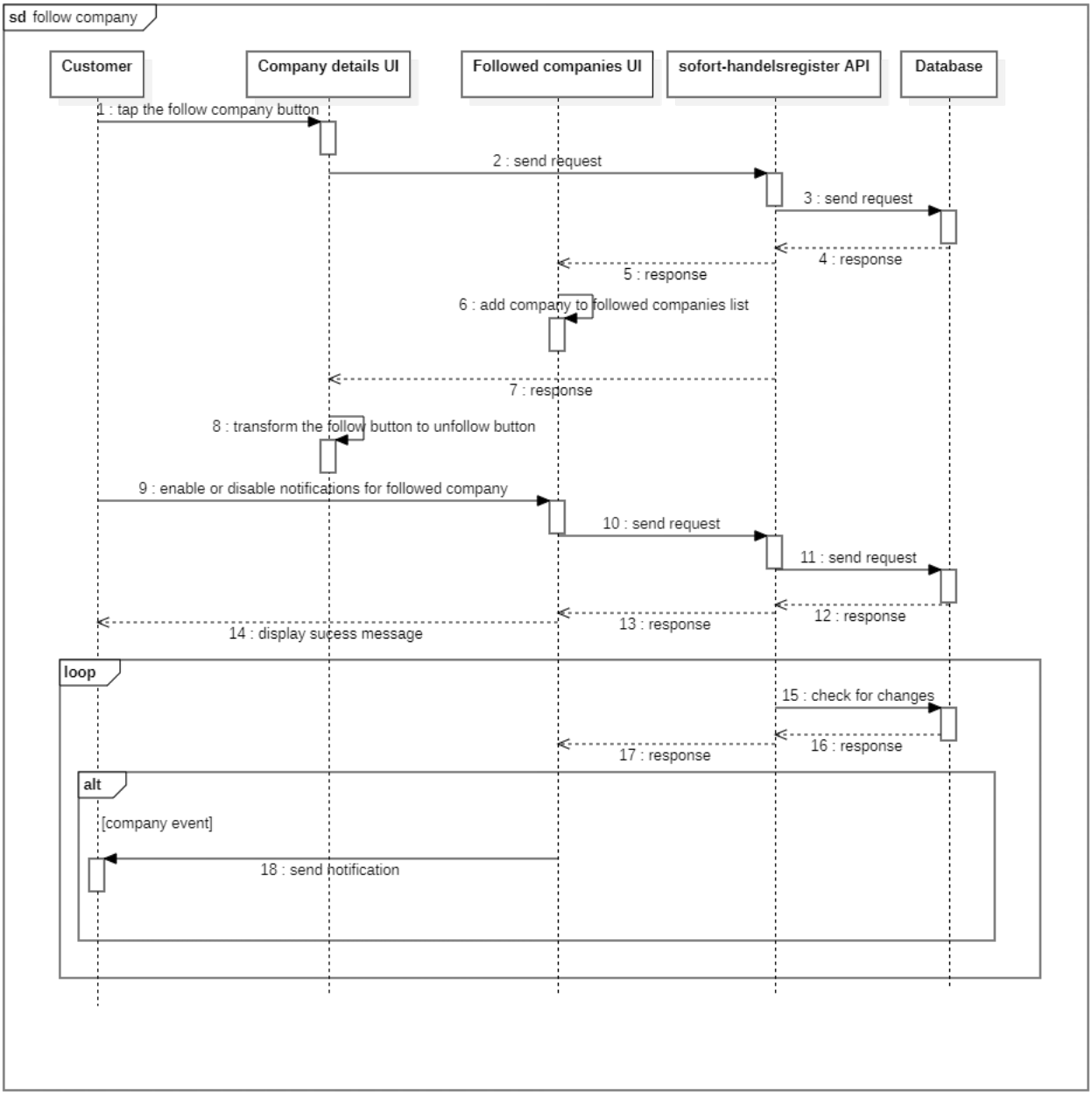

Figure 17. *Detailed sequence diagram of the ”follow company” use case*

3.2.5 Detailed sequence diagram of the ”purchase documents” use case

The following figure illustrates the detailed sequence diagram of the use case ”purchase documents”. This use case starts with opening an interface containing specific company details for the user. Then the user may scroll down to the documents section and add wanted documents to his cart, after that he may checkout and finalize his purchase process after filling in his credit

card details. As a result, the purchased documents are added to the user's inventory interface and he may download them.

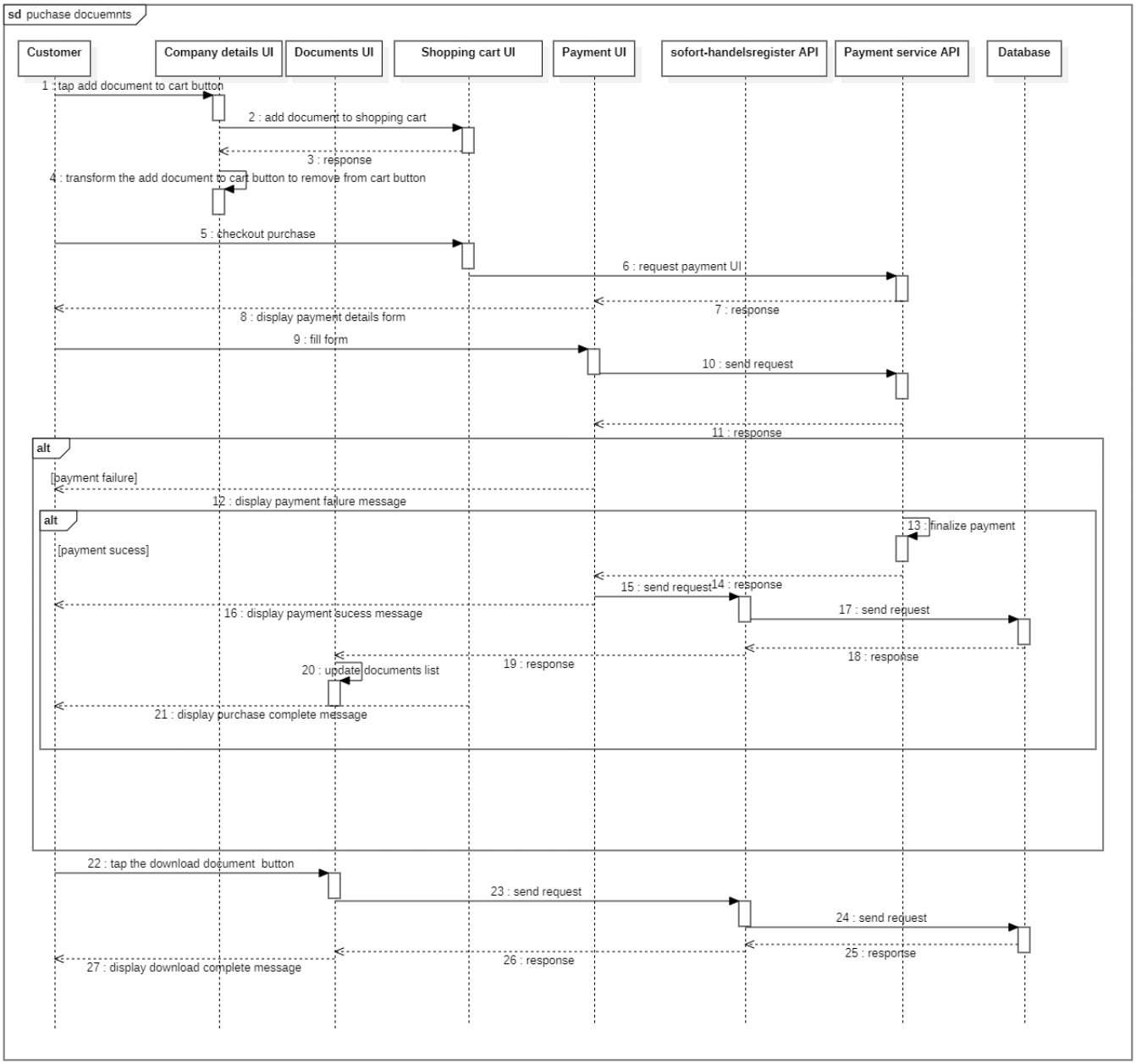

Figure 18. *Detailed sequence diagram of the "purchase documents" use case*

Conclusion

This chapter presents one of the most important phases of the process of development of a project: design subdivided into the overall design and detailed design.

We presented the Clean Architecture architecture pattern and we explained its application in our case. We have described the dynamic view of the system through a set of sequence diagrams and diagrams of activities.

The following chapter will deal with the realization of the project illustrated by screenshots of different interfaces. We will describe the environment of work and the tools used too.

Chapter 4

Realisation

Introduction

In this chapter, we describe the working environment used during the realization of our application. We will also describe its layout physical using a deployment diagram. Then, we detail the work carried out and the results obtained using a set of screenshots representing the interfaces of the different functionalities of our app.

4.1 *Technical specification*

In this section, we will present the technical choices relating to the hardware and software environment that contributed to the realization of our application.

4.1.1 Hardware environment

During the different stages of our project, i.e. documentation, code implementation and testing, we had:

- A personal computer with the following configuration:
  - Brand: Lenovo;
  - Processor: AMD Ryzen 3 3200U with Radeon Vega Mobile Gfx 2.60 GHz ;

  - RAM: 12 GB;
  - Hard disk: 1TB;
  - System type: Windows 64-bit operating system.

- A Smartphone with the following characteristics:
  - Brand: Itel;
  - RAM: 1 GB;
  - Processor: Quad Core Qualcomm Snapdragon 810 @ 2.3GHz;
  - System version: Android 9.0;

4.1.2 Software environment

Throughout the development phase, we used the tools software and the following development languages:

4.1.2.1 Software tools

During the implementation of our project we used the following software :

Android Studio

Android Studio is an integrated development environment for android, IOS, web, and Windows platforms. Developers use this environment to develop Cross platforms applications. Android Studio allows mainly editing the Dart files and configuration files of the application.

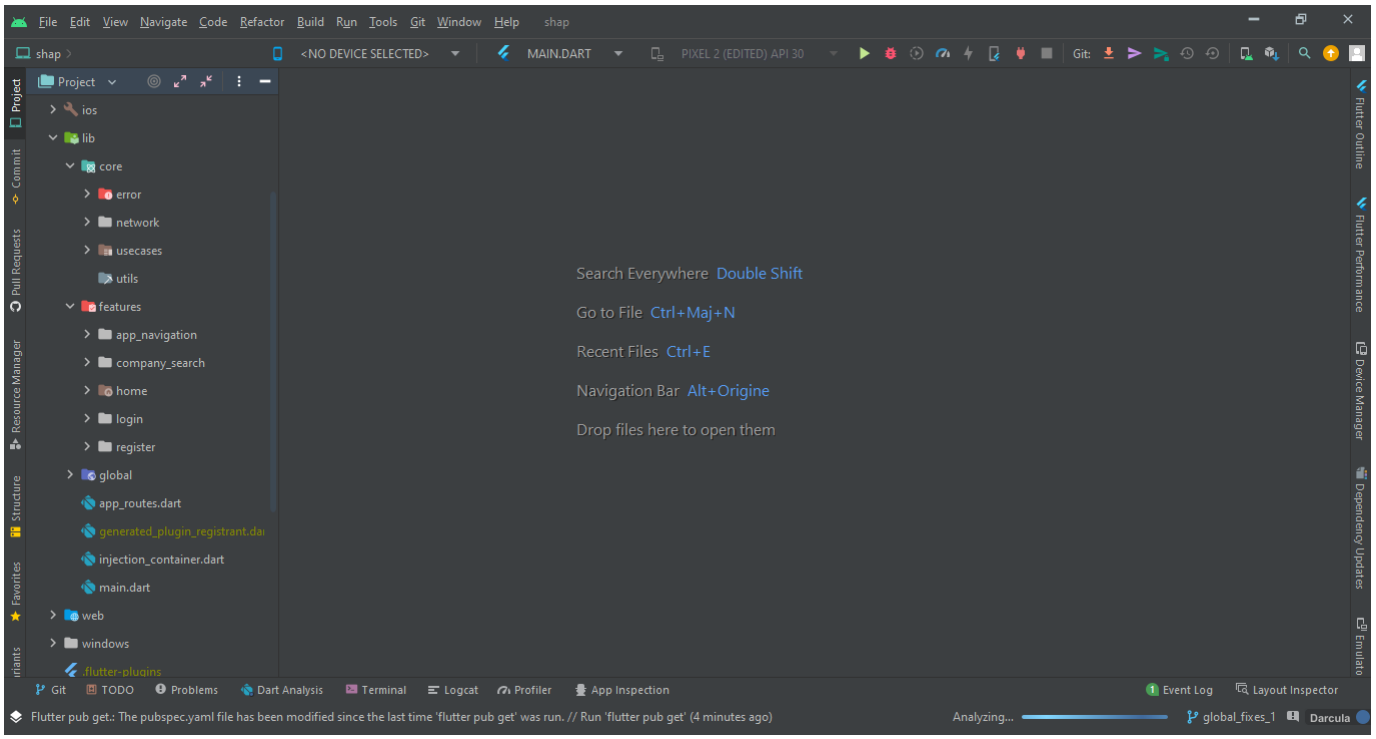

Figure 19. *Android Studio*

PostMan

Postman is an API platform for developers to design, build, test, and iterate their APIs. As of April 2022, Postman reports having more than 20 million registered users and 75,000 open APIs, which it says constitutes the world's largest public API hub.

In our application, we used PostMan to test APIs.

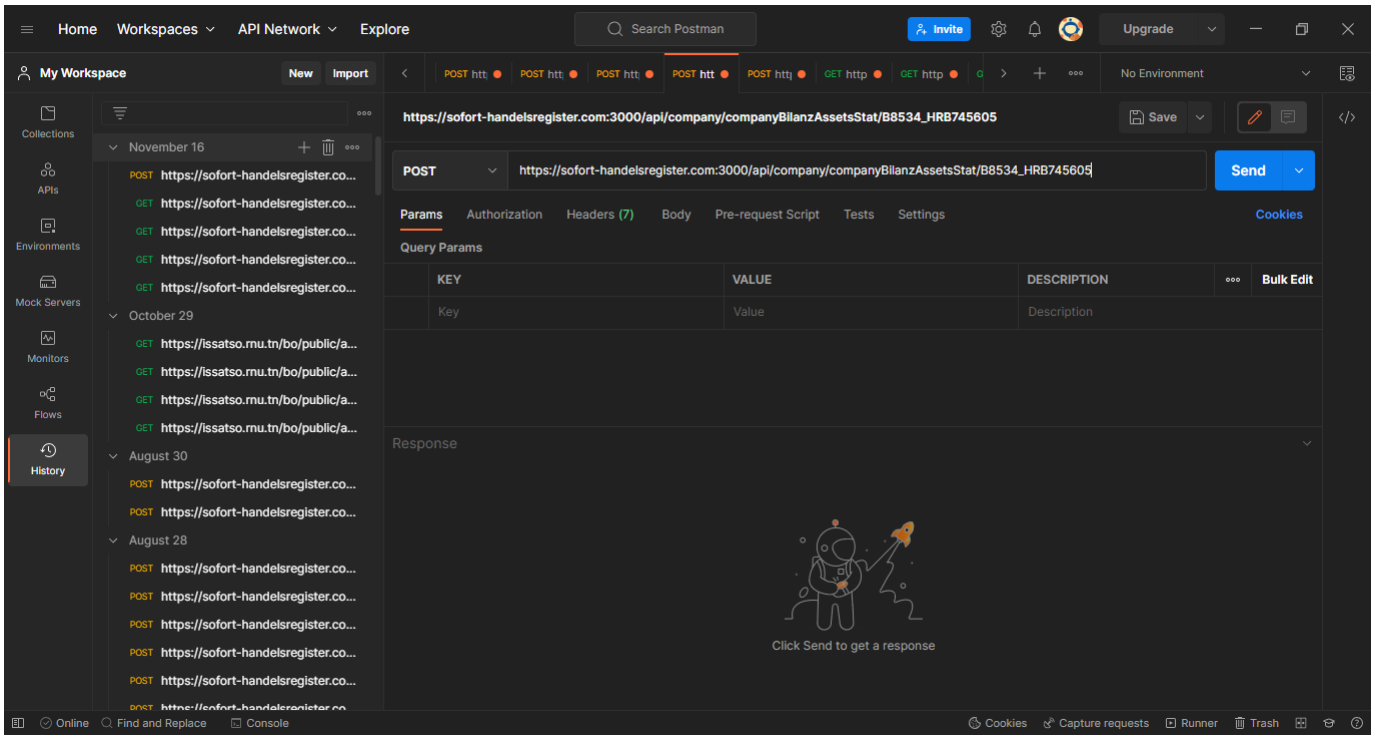

Figure 20. *PostMan*

StarUML

StarUML is a software engineering tool for system modeling using the Unified Modeling Language, as well as Systems Modeling Language, and classical modeling notations. It is published by MKLabs and is available on Windows, Linux, and macOS. Wikipedia.

In our application, we used StarUML to establish UML diagrams for this report.

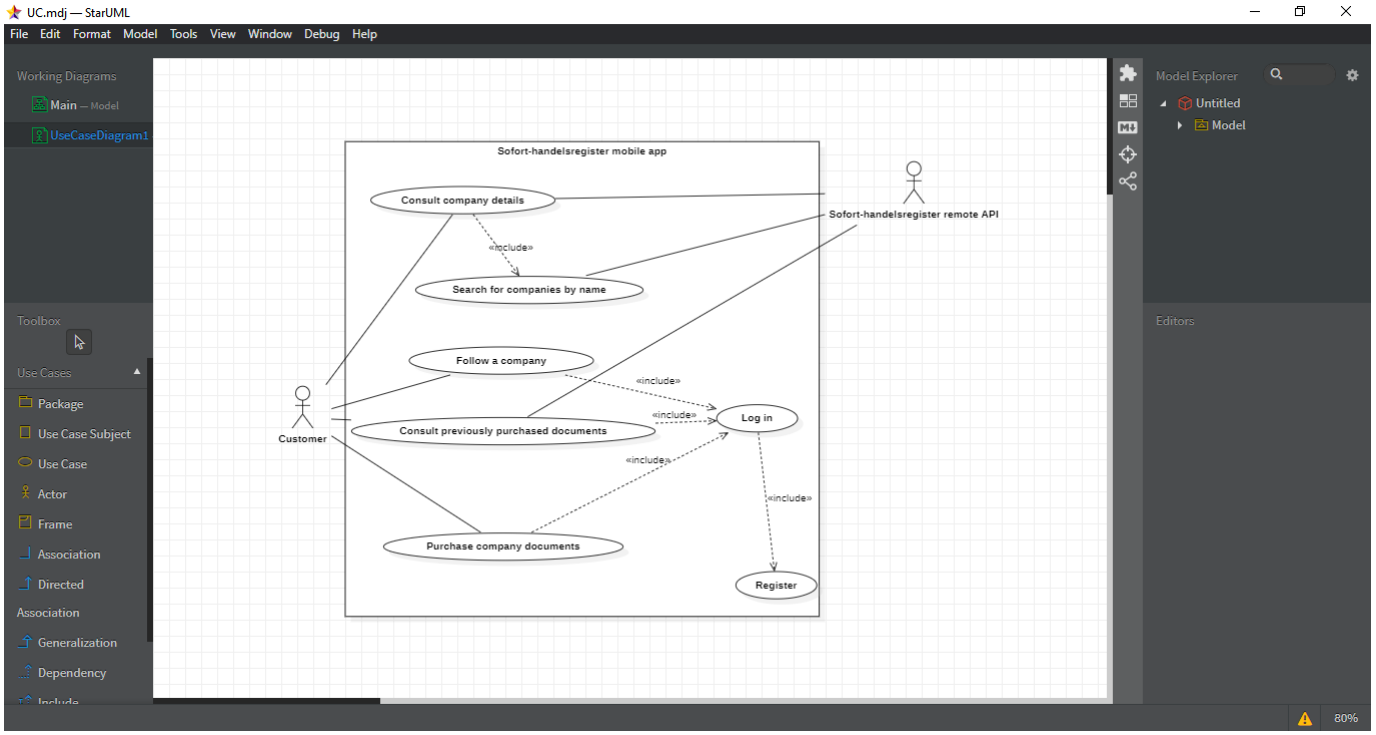

Figure 21. *PostMan*

Social media login providers

The following online platforms were used in our application to provide and configure the login process using social media accounts within our application those platforms are respectively Google Cloud, Meta for developers, LinkedIn for developers, and xing for developers as shown in the following figures.

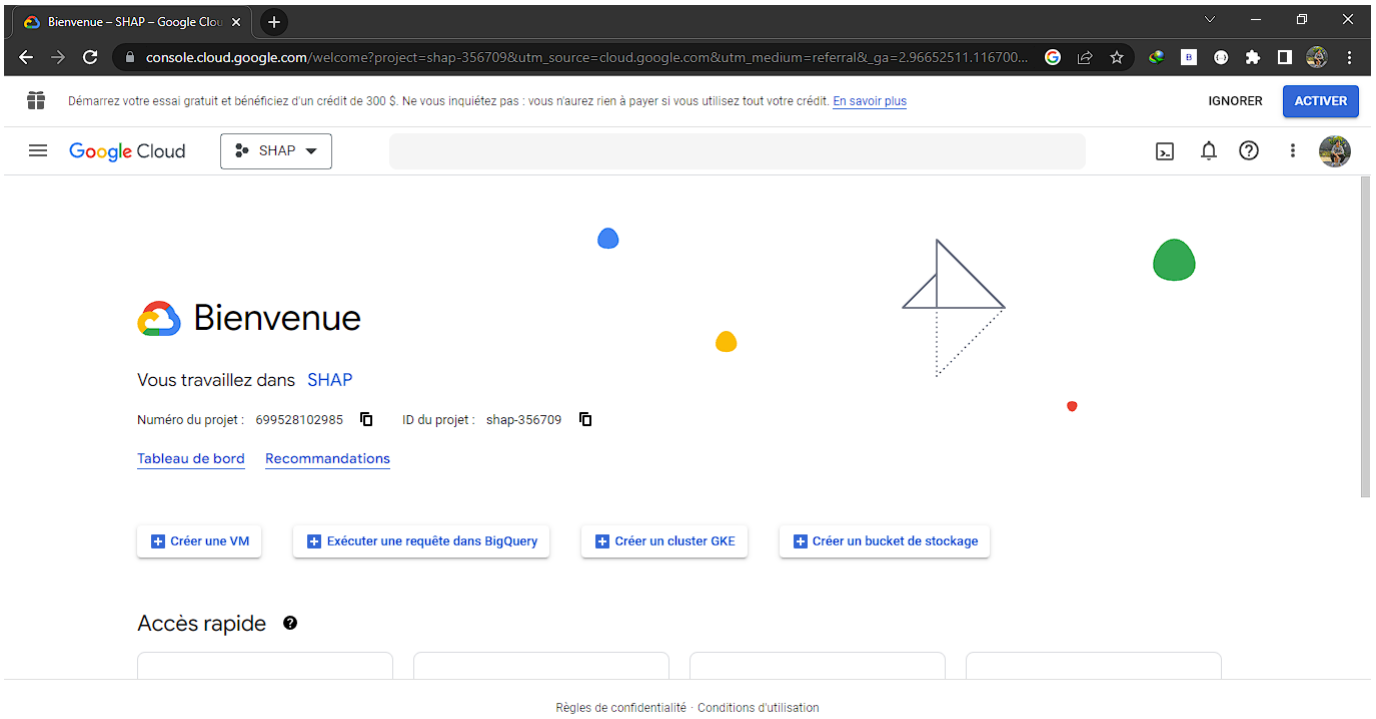

Figure 22. *Google cloud*

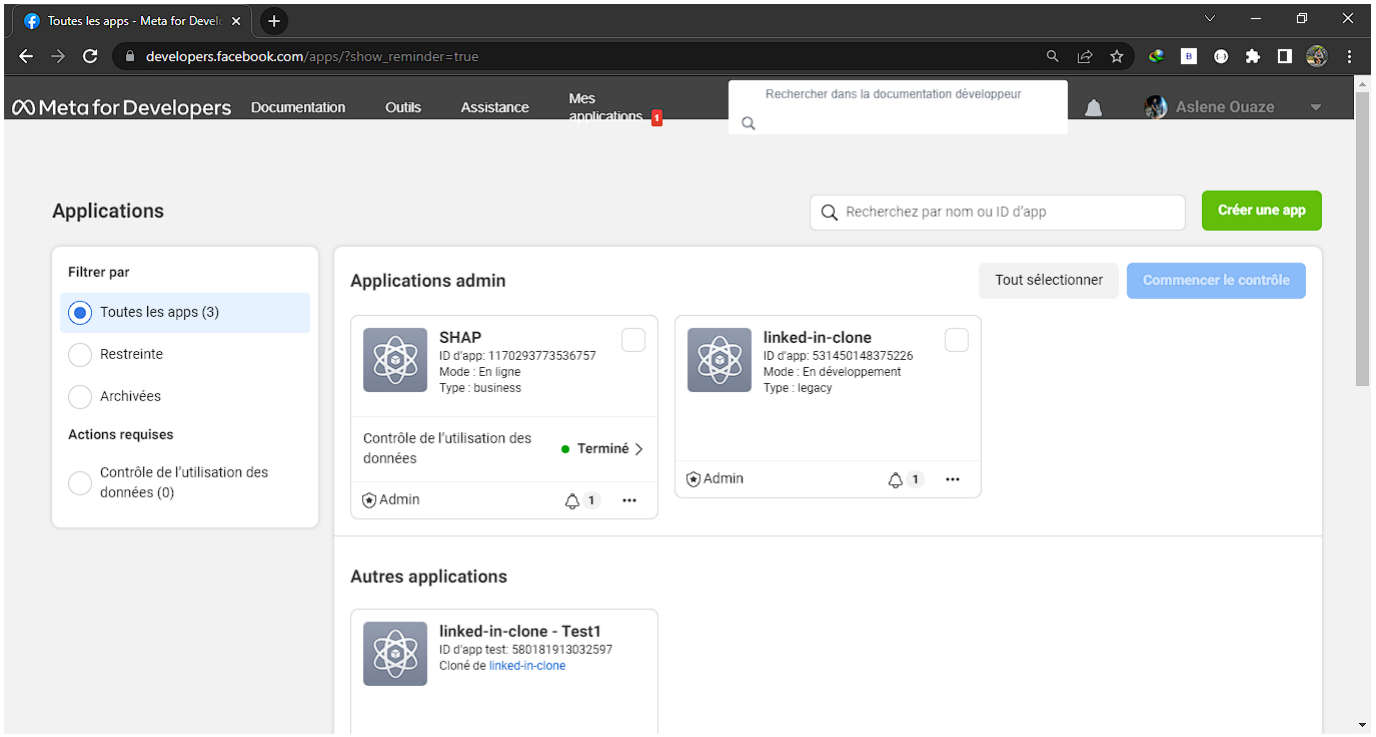

Figure 23. *Meta for developers*

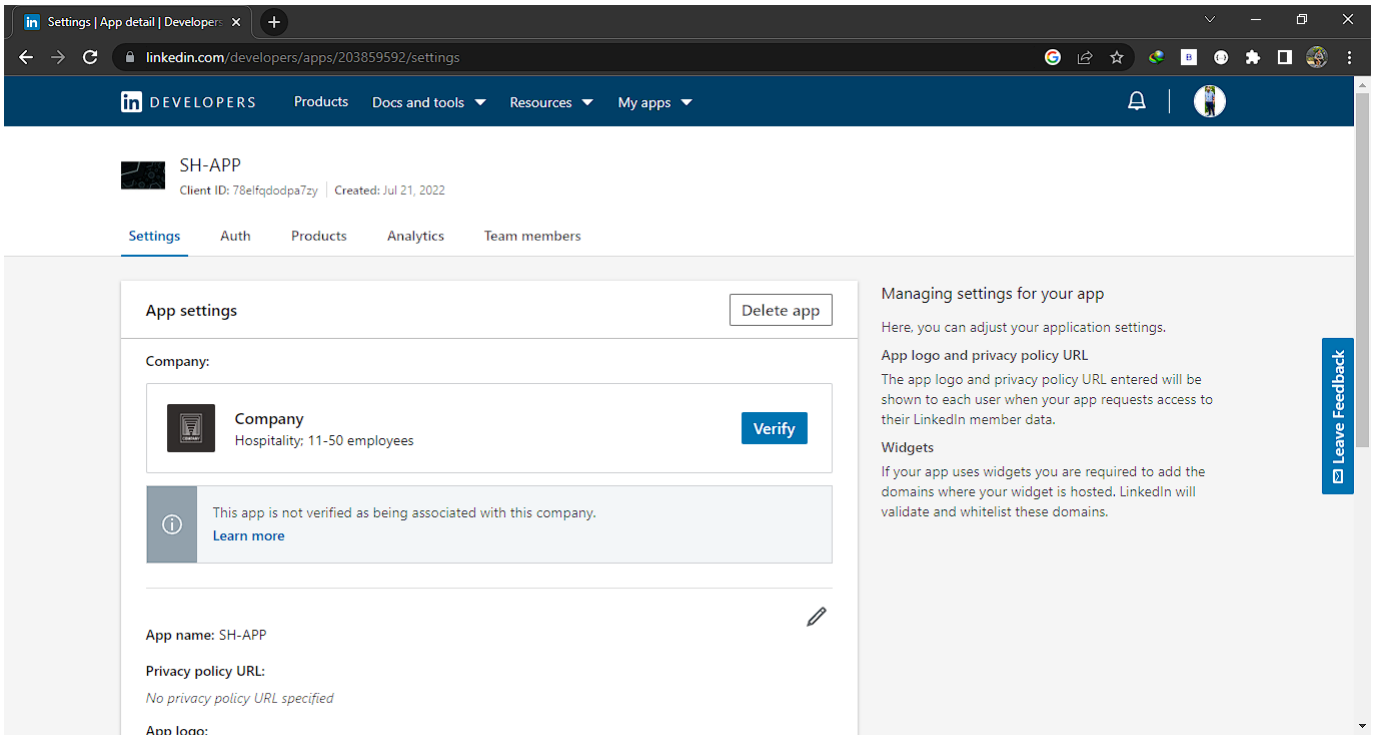

Figure 24. *LinkedIn for developers*

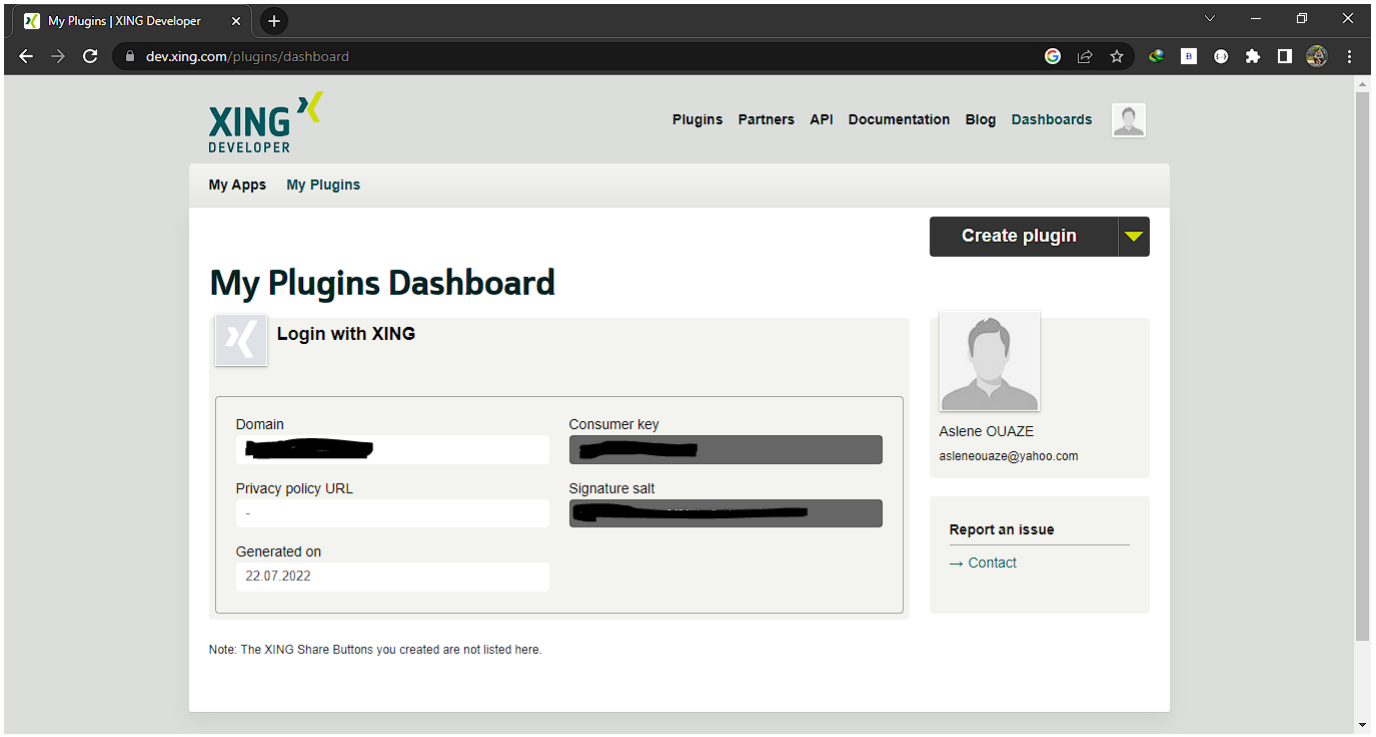

Figure 25. *Xing for developers*

4.1.2.2 APIs and libraries

Sofort-handelsregister API

The Sofort-Handelsregister API is an API developed by the host company Glocal IT, its currently used in the web platform Sofort-handelsregister. comthat offers similar services to our applications, the API allows us to communicate with the Glocal It database that contains company data, statistics, documents, and customer information as well as a robust login and register service for customers.

We can integrate it into mobile or web applications regardless of the programming language used.

Libraries

Library	Usage
flutter_facebook_auth	Add Facebook login to our flutter app. Feature includes getting user information, profile picture, and more. This plugin also supports Web and macOS.
linkedin_login	Library for login with LinkedIn OAuth V2 service. This library helps to implement authorization with LinkedIn OAuth API's.
google_sign_in	Flutter plugin for Google Sign-In, a secure authentication system for signing in with a Google account on Android and iOS.
webview_flutter_plus	An extension of webview_flutter to load local HTML,CSS and Javascript from Assets or Strings and much more.
flutter_bloc	Flutter Widgets that make it easy to implement the BLoC (Business Logic Component) design pattern. Built to be used with the bloc state management package.
email_validator	A dart class for validating email addresses
shared_preferences	Flutter plugin for reading and writing simple key-value pairs (local cache). Wraps NSUserDefaults on iOS and SharedPreferences on Android.
connectivity_plus	Flutter plugin for discovering the state of the network (WiFi & mobile/cellular) connectivity on Android and iOS.

Table 7. *Libraries*

4.1.2.3 Programming languages

Dart

Dart is a programming language designed for client development, such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop

applications. It is an object-oriented, class-based, garbage-collected language with C-style syntax.

XML

The XML (Extensible Markup Language) allows the developer to create information formats and share them via networks. This standard is used to describe data.

JSON

JSON(JavaScript Object Notation) is a lightweight data-interchange format that is easy to read and write and frequently used by developers.

4.1.3 Deployment diagram of the application

The deployment diagram models the physical architecture of a system. This diagram shows the topology of the software elements that are deployed over the material elements. The following figure represents the deployment diagram of our application.

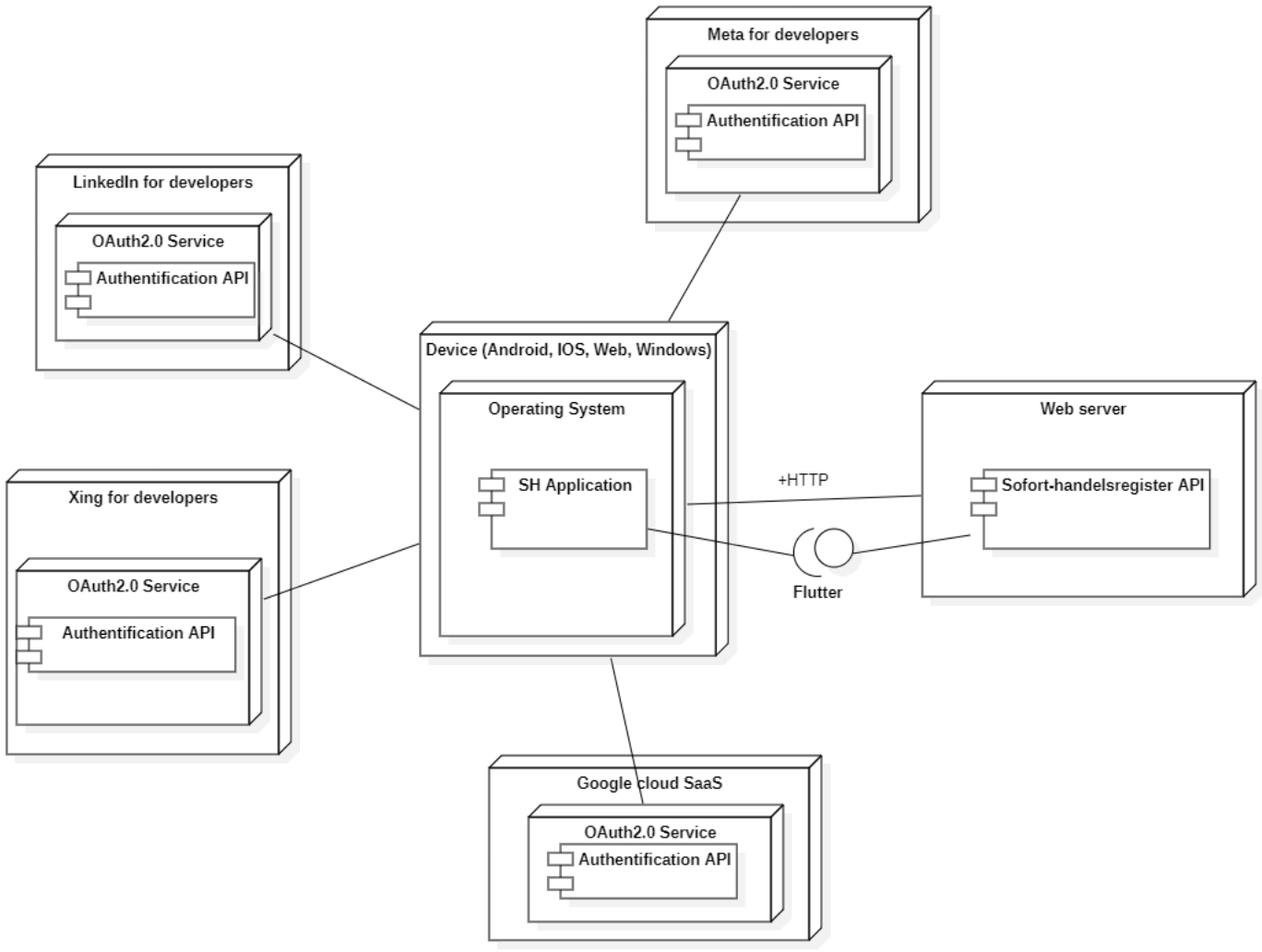

Figure 26. *Deployment diagram of the application*

4.2 *Apllication interfaces*

In this part, we present the different functionalities of our application by presenting the graphical interfaces produced:

4.2.1 Launching the application

The following figure represents the landing page of the application where the user can choose between two options login and searching for a company when choosing login the user will be

redirected to the login interface, when choosing company search the user will be redirected to the company search interface

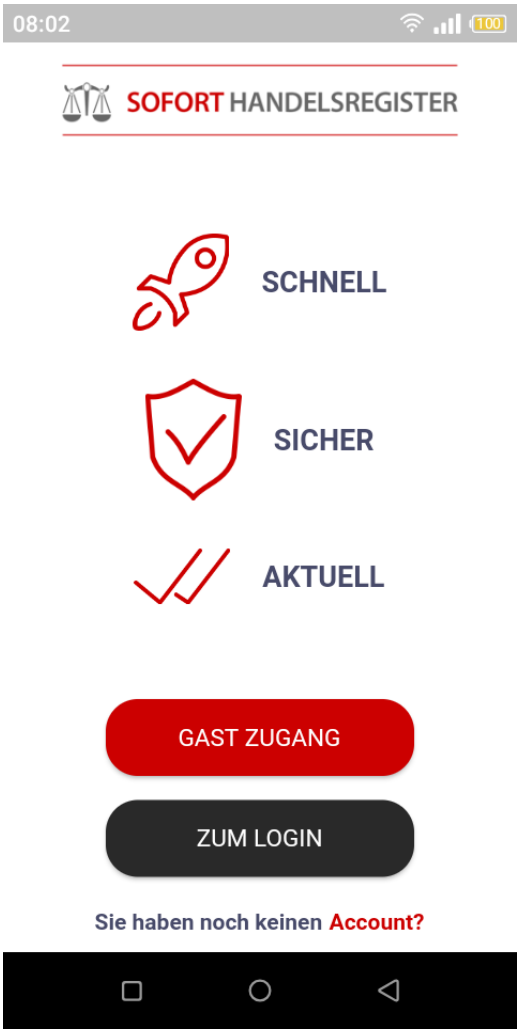

Figure 27. *Landing page of the application*

4.2.2 Authentication

The following figure explains the authentification process. The user can input his account email and password, he can also tap the forgot password option to receive an email containing a link to set up his new password. The buttons ”Facebook”, ”Google”, and ”LinkedIn” can be used to log in using social media accounts.

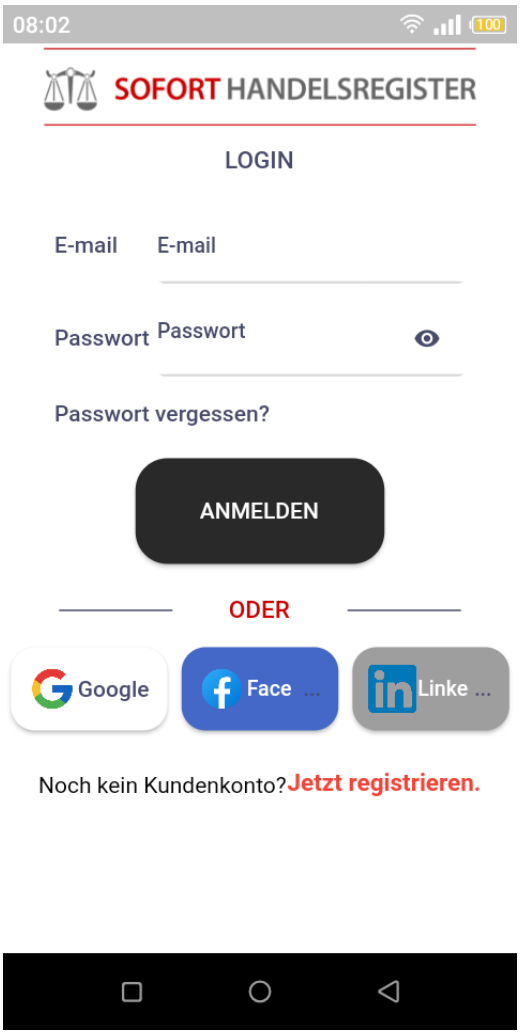

Figure 28. *Authentication page of the application*

4.2.3 Registeration

The register interface contains the input fields: FirstName, LastName, Email, Password, Password confirm, and the accept terms and conditions check box that is necessary for a successful registration. Upon validating the form the user will be redirected to the login interface.

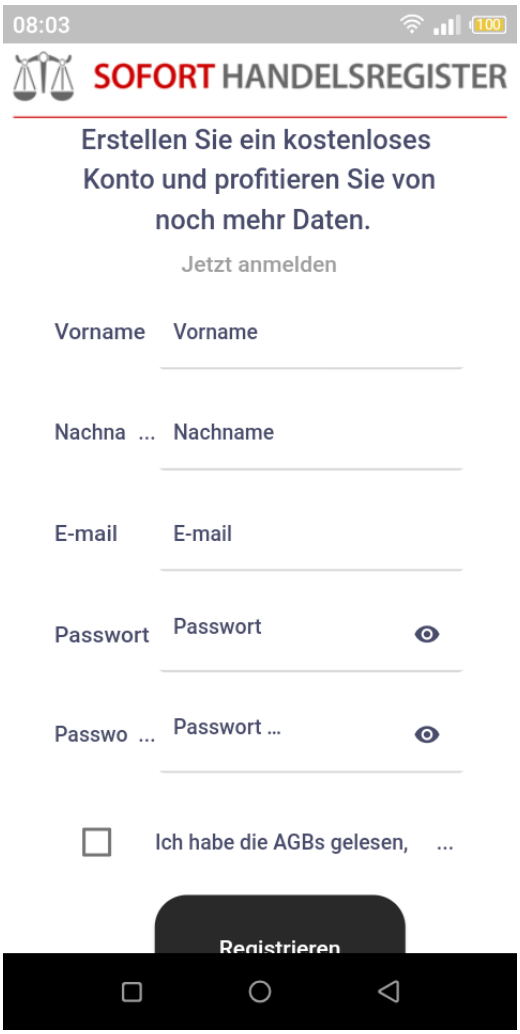

Figure 29. *Registeration page of the application*

4.2.4 Company search

The following figure represents the company search interface where the user can input the full name of his wanted company or the first letter of it using the text input field on the top of the page. The search results are presented as an interactive list showing the name of the company, its full address, its current status, and its legal form.

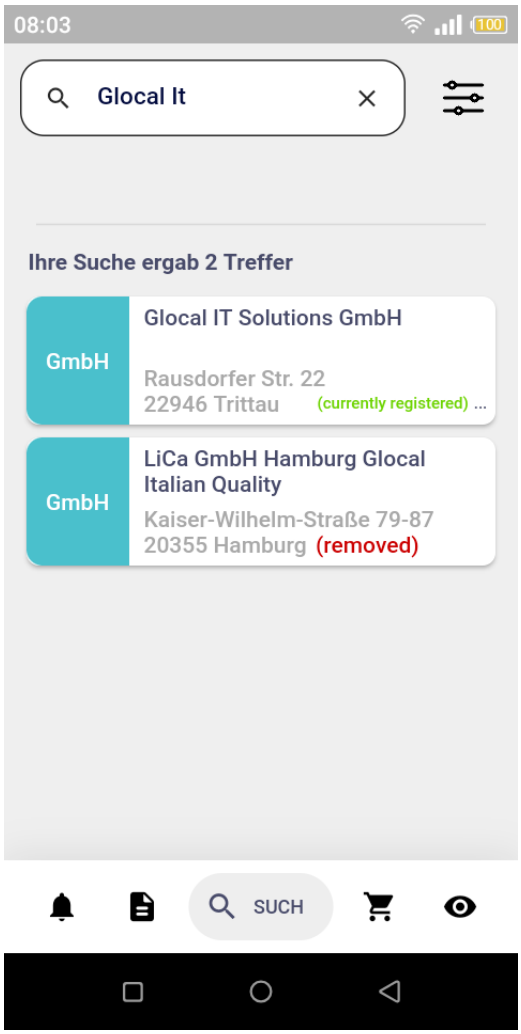

Figure 30. *Company search page of the application*

4.2.5 Results filtering

The filter interface allows the user to filter his search results using state, zip code, legal form, capital, and branch as shown in the following figure.

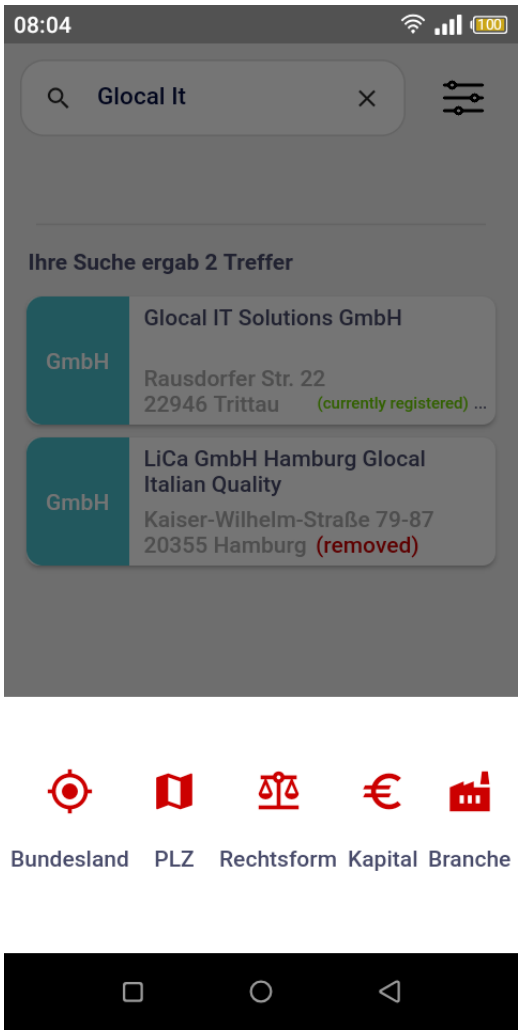

Figure 31. *Filter interface of the application*

The filter interface presents two main types of filtering the first one represents the custom filter where the user can type the filter in the text input field as shown in the following figure.

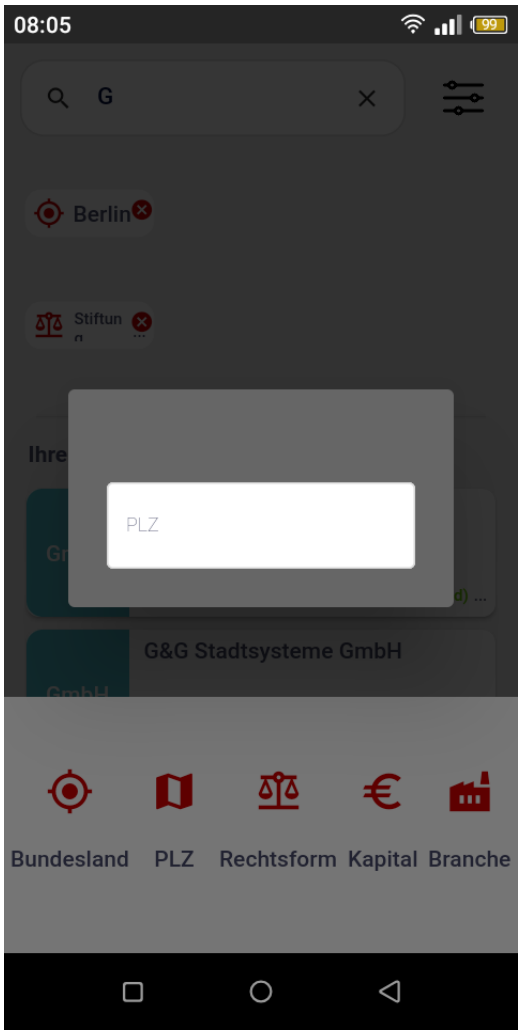

Figure 32. *Custom filter interface of the application*

The second filter type represents the predefined options filter where the user can choose one or many pre-defined options as shown n the following figure.

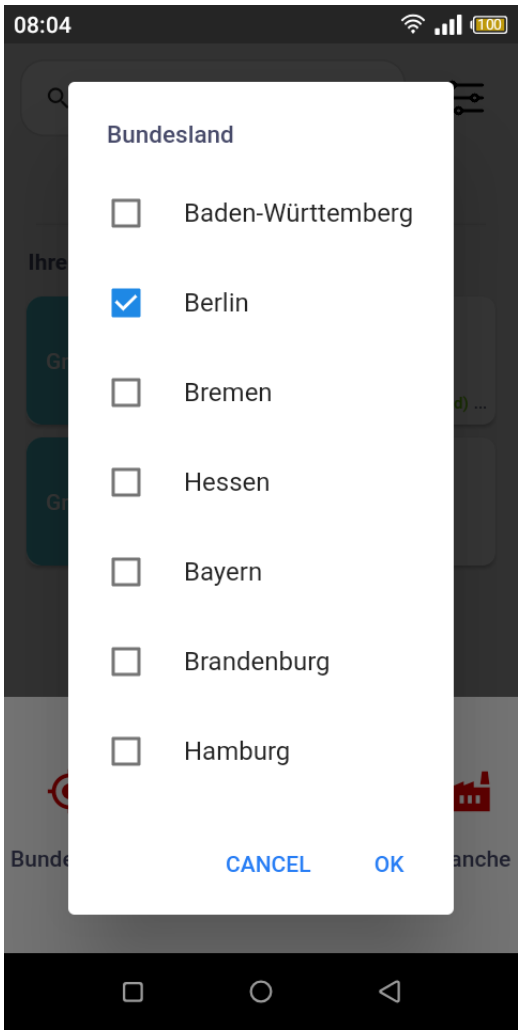

Figure 33. *Pre-defined filter interface of the application*

After choosing all his filters the user may come back to his check his filtered results where he may find his applied filters displayed as shown in the following figure.

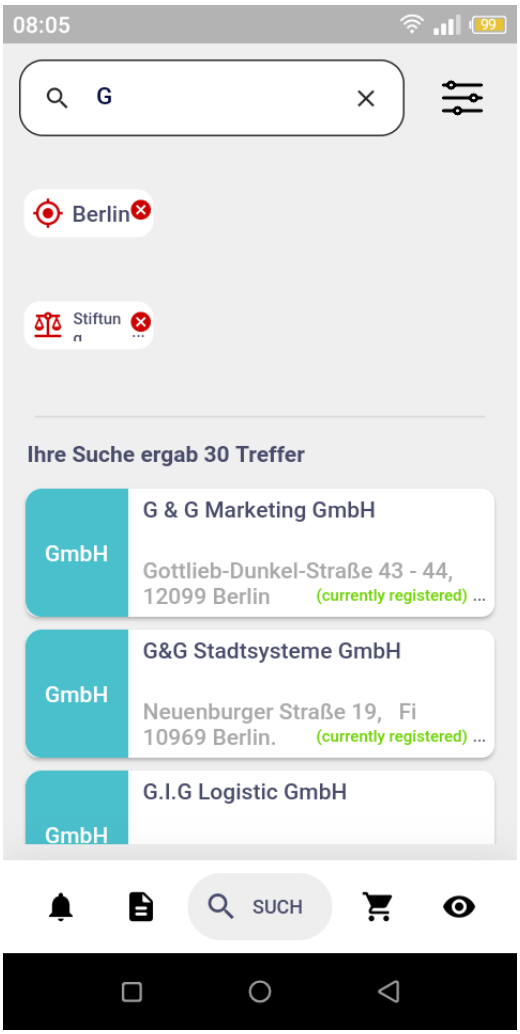

Figure 34. *Filtered search results interface of the application*

4.2.6 Company Details

After choosing an element from the list of results, the customer will be redirected to the company details interface where he may consult the company name, address, current status, legal form, officers, branch, phone, mobile phone, email, and website as shown in the following figure.

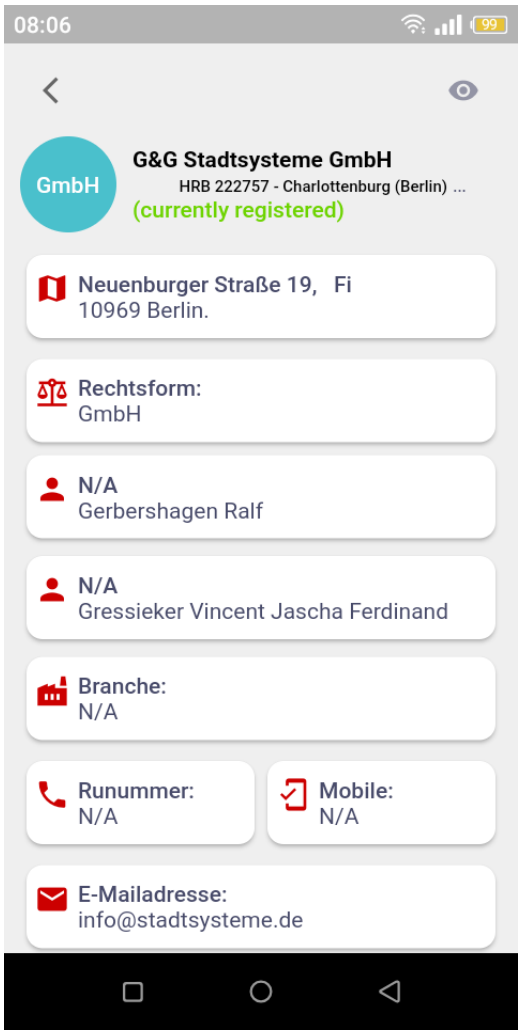

Figure 35. *Company details interface of the application*

The same interface presents the documents section where the list of documents is displayed and can be selected by the user to be added to the shopping cart using the toggle button in front of each element as shown in the following figure.

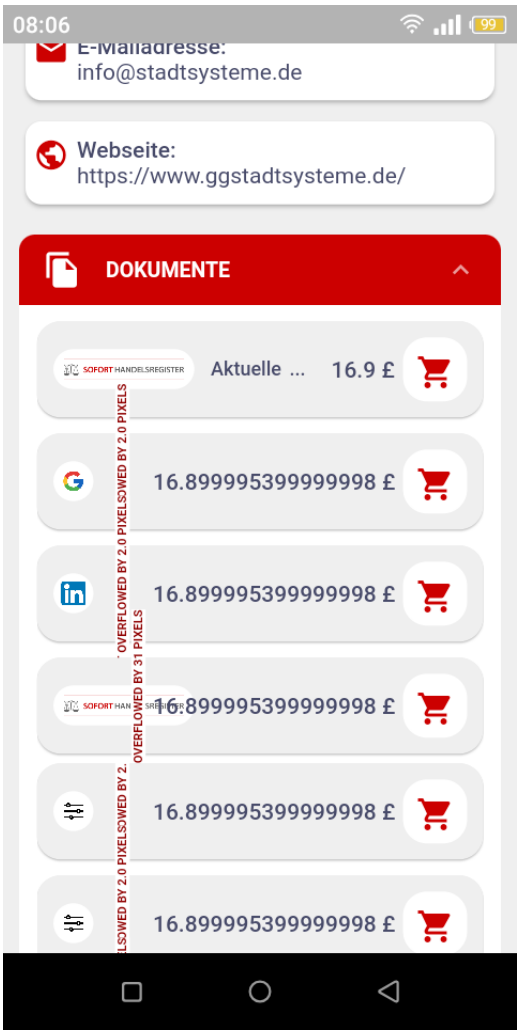

Figure 36. *Document section interface of the application*

The last part of the company details interface presents the notices section, insolvency check section, and statistics section available for free for the customer as shown in the following figure.

Figure 37. *Extra section interface of the application*

Conclusion

In this chapter, we presented the technical specification of our application through the introduction of hardware and software environments. Ultimately, we have described the functionalities of our application, illustrated by screenshots of its user interfaces. We end our report with the general conclusion and the different prospects.

Conclusion and perspectives

Admittedly, the diversity of mobile applications is an enrichment world of mobility, and social life, where everything goes through these applications to facilitate daily tasks and bring more comfort, gain in time and money.

The objective of this report is to present the application carried out during our summer internship project within the company Glocal IT. Our application allows customers to track specific companies by receiving notifications about them and consulting deep details about them. The app carried out also integrates a document marketplace where customers can purchase specific company documents and download them to their local storage.

This internship allowed us to improve our technical knowledge given that we used a variety of technologies. On the other hand, during this internship, we got used to professional life and teamwork. To conclude, our realized application can be extended by developing payment functionality planned in the design, as well as the notification system. It is also possible to make our application multilingual.

Bibliography

[1] Belgacem, Dr. Selma: Les patrons d'architecture (2) Clean Architecture. (ISSAT de Sousse)]. (35), 2021.

[2] Oumayma LAZREG : Rapport PFE, Conception et développement d'une application mobile intelligente de e _ consulation et d'estimation des maladies. (ISSAT de Sousse), 2019.