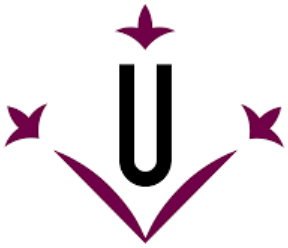

Universitat de Lleida

Sprint 3

Commed

Made by
Oriol Alàs Cercós, Nicolas Arias, Yassine Elkihal, Emina Hanzic
Joaquim Picó Mora, Sergi Simón Balcells

Delivery
15th of December, 2021

Universitat de Lleida
Escola Politècnica Superior
Grau en Enginyeria Informàtica
IT Project Management

Professorate:
Juan Enrique Garrido Navarro
Josep Escribà Garriga

Contents

List of Figures

List of Tables

1 Relevant links

- GitHub backend repository
- GitHub Mobile App repository
- GitHub Web client repository
- GitHub docs
- GitHub project management
- Slides of the presentation
- Spreadsheet documentation

2 Planification

2.1 User Stories

The first thing that was done regarding the planification of the project was to define the behavior of the application in a list of user stories. The next list exposes all of the actions that the user can do with it as well as different ways of interacting with it. It has to be noted that for this Sprint there has been no changes on the user stories.

- **AUTH1**: As a guest, I want to register in the application
- **AUTH2**: As a user, I want to log in to the application.
- **AUTH3**: As a registered user, I want to create a profile of my company.
- **PROD1**: As a guest, I want to search for services or products so that I receive a list of services or products.
- **PROD2**: As a guest, I want to have a detailed view of the product/service.
- **PROD3**: As a registered user who has a company profile, I want to create services/products.
- **CHAT1**: As a user, I want to connect to a company because of a publication.
- **CHAT2**: As a user, I need to chat with the company that I connected with.
- **CHAT3**: As a company, I want to chat with the users that have sent a message.
- **FO1**: As a company, I want to send the Formal Offer which contains the contract pdf through the chat.
- **FO2**: As a company, I want to digitally sign contracts.

- **FO3**: As a user, I want to digitally sign contracts.
- **FO4**. As a company, I want to have a list of my sent formal offers.
- **FIN1**: As Commed, I want to get a 5% commission on each contract.
- **FIN2**: As a company, I want to publish the first 3 announcements freely.
- **FIN3**: As a company, I want to have the possibility to pay to promote my announcements.

2.2 Scrum organization and planification

Afterwards, a meeting was held in order to fulfill the product backlog with all the tasks that had to be done. These tasks were related to User Stories, but they were divided so that the product backlog had a small granularity in the given tasks. Some tasks were leftovers from the second sprint, as it was intended to have the integration with the API done by the end of it, but the team didn't have enough time to implement them.

According to the kanban, it has been created a milestone named *Sprint 3*, which groups all the issues from this Sprint. It has also been migrated the leftovers from the *Sprint 2* to the new milestone.

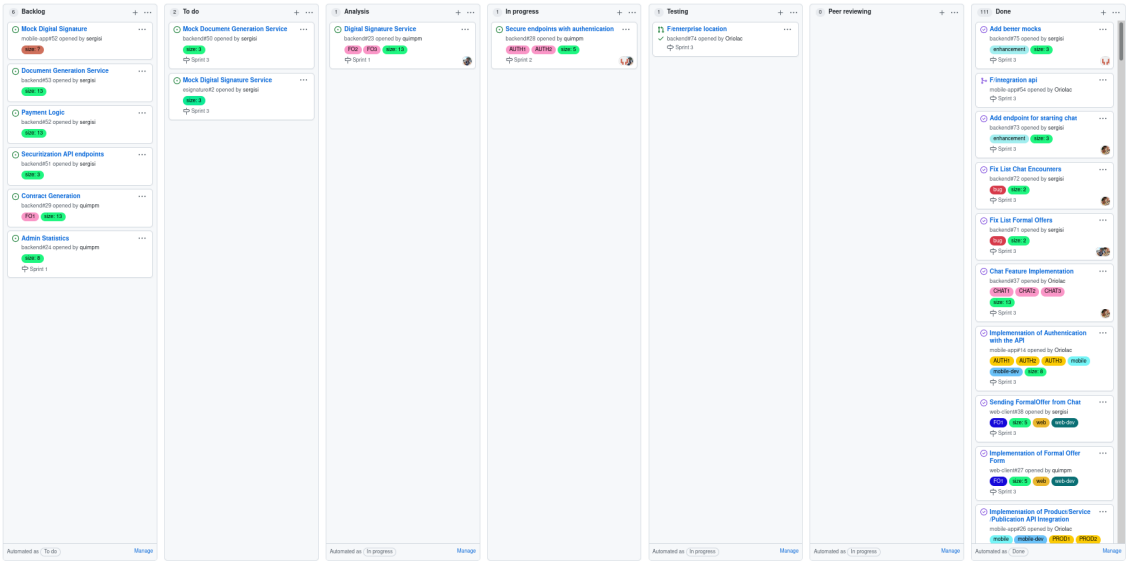

Figure 1: Screenshot of the last days of the project kanban.

2.2.1 Issues

Backend

```
- Upgrade Docker
- size: 1
```

- Chat Feature Implementation
- size : 13
- Fix Product Images CRUD
- size: 13
- FormalOffer in Admin
- size: 1
- Enterprise Profile Image and Banner
- size: 1
- Fix List Formal Offers
- size: 2
- Fix List chat encounters
- size: 2
- Add Endpoint for starting chat
- size: 3
- Add better mocks
- size: 3

Web client

- Sending FormalOffer frm Chat
    - size: 5
- Implementation of Formal Offer Form
    - size: 5
- Implementation of Chat
    - size: 8
- WebClient NavBar fixes
    - size: 3
- WebClient Labels
    - size: 2
- Recommendations algorithm Location
    - size: 2

Mobile application

- Implementation of Authentication with the API
    - size: 8
- Implementation of FormalOffer API backend Integration with flutter
    - size: 5
- Implementation of Chat API Integration
    - size: 13

```
- Implementation of Search feature with the API in mobile
    - size: 8
- Implementation of Product/Service/Publication integration
    - size: 5
- Implement application labels
    - size: 1
- Application NavBar fixes
    - size: 3
```

2.3 Scrum analytics

As Github does not support milestones for a group of repositories but for single one, it has been created three different milestones for each project: `backend`, `web-client` and `mobile-app`.

backend

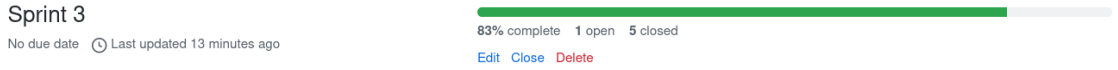

Figure 2: Backend Sprint 3 milestone

As it can be seen, the backend has only a few issues but some big ones. The implementation of Chat was one of the biggest use cases that was implemented, and integrating it with the web client and the mobile application was not an easy task. Also, it was thought at first that solving a bug regarding the product images would be an easy task, but it took as much as the chat implementation itself, as a lot of tasks regarding the analysis was needed before being able to implement a correct way.

web-client

Figure 3: Web client Sprint 3 milestone

Regarding the web client, all of the tasks assigned for this Sprint are done. The main topic in the web client was polishing things and making the chat work, as well as the formal offers.

mobile-app

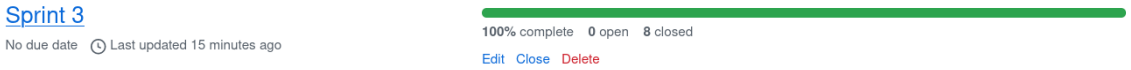

Figure 4: Web client Sprint 3 milestone

In this sprint, the API integration was done in he mobile application. It was also polished some things for the user experience.

3 Requirements

In this section the list of requirements that the application has to offer to the user are on the list below. As matter of fact, the requirements were left as they were in the previous Sprint, as no new ones or refinements were found.

Functional Requirements:

- The application has to let all kinds of users search for products or services.
- The application has to let users register into the application and it will create automatically an enterprise profile.
- The application has to let users log in to the application if they have an active account on the system.
- The application has to let logged users publish products or services in the web application.
- The application has to let logged users interested in either a product or a service to start a chat with the owner of it.
- The application has to let logged users who are owners of a given product to chat with said interested users through a chat.
- The application has to let logged users send a commercial transaction contract when an agreement has been reached .
- The application has to let logged users download a commercial transaction contract sent by the owner of a product that they are interested.
- The application has to let logged users sign a commercial transaction contract sent by the owner of a product that they are interested.
- The application has to generate the evidences for both sides of the commercial agreement.
- The application has to let logged users to view a list of the formal offers they are in.

Non-Functional Requirements:

- The application has to be the most usable possible.
- The application has to be compliant and respect the laws that run in each country that it's available in.

- The application mustn't have large waiting times for the client.
- The application has to be portable and easy to deploy.
- The application has to be scalable and always leave the code open to the possibility of adding new features in the future.

4 Main use cases

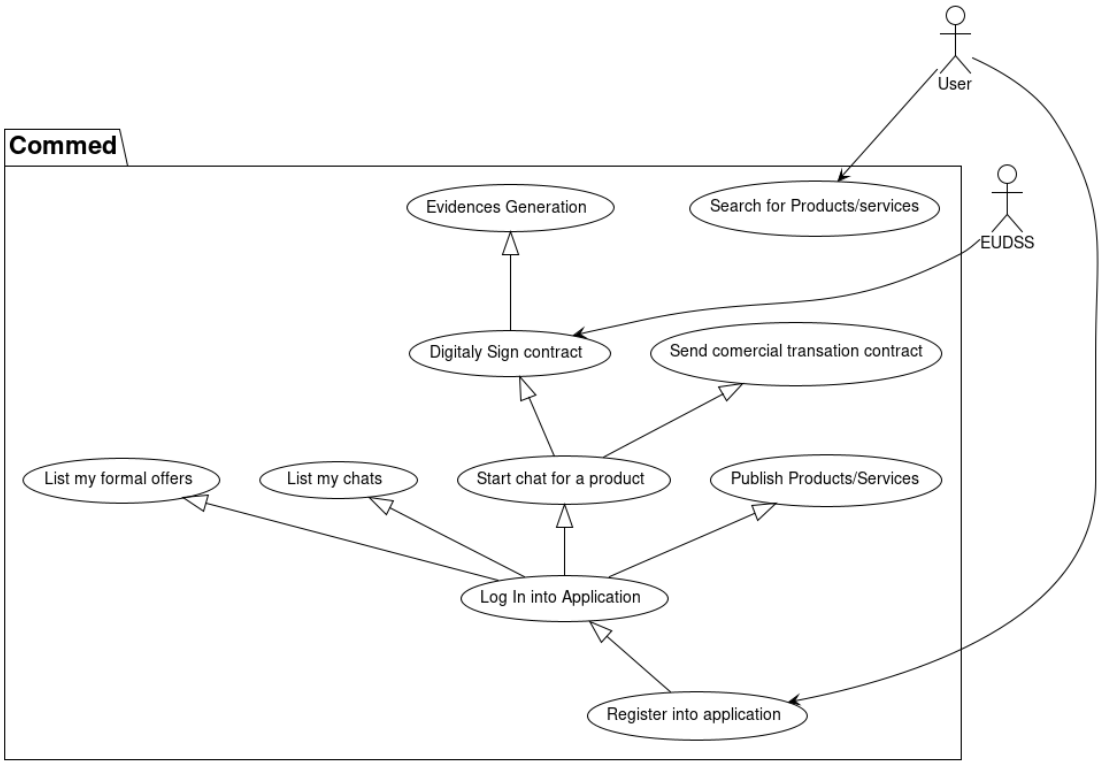

Figure 5: Use case diagram of the application.

The use cases will be described as:

- **Register into application**
  - **Actors:** User
  - **Purpose:** Let a user register into the application system
  - **Description:** Provides a screen with a form in which the user is able to fulfill it and send the information to the system in order to be registered.
- **Log In into Application**
  - **Actors:** User

  - **Purpose:** Log in to the application to be able to use some of the application services.
  - **Description:** Provides a screen with a form in which the user will put its email and password. Then, they will log into the application so that they can start using the services that it provides.

- **Search for Products/services**
  - **Actors:** User
  - **Purpose:** Search for any product or service the user is interested in.
  - **Description:** Provides a searcher for every user so that they can look up the products or services that they are interested in.

- **Publish Products/Services**
  - **Actors:** User
  - **Purpose:** Publish services or products in order to be sold to other users.
  - **Description:** Lets a logged user publish the products and services that they offer in order for them to be sold to other interested users.

- **Start chat for a product**
  - **Actors:** User
  - **Purpose:** Users can start a chat when they are interested in a product
  - **Description:** Lets a logged user start a chat with the owners of either a product or a service that they are interested in, so that they can start a negotiation.

- **Send commercial transaction contract**
  - **Actors:** User
  - **Purpose:** Send a formal offer with a commercial transaction contract.
  - **Description:** Lets the owners of given products or services send a formal offer containing a compliant commercial transaction contract within the chat in which the negotiations are taking place.

- **Digitally Sign contract**
  - **Actors:** User EUSSD
  - **Purpose:** Sign a commercial transaction contract sent within a Formal Offer.
  - **Description:** Lets the users of the application sign digitally the contract that was sent as a Formal Offer in the chat in which the negotiations took place.

- **Evidences Generation**
  - **Actors:** User
  - **Purpose:** Provide users with evidences and the billing of a business transaction
  - **Description:** The system will generate for both parts the commercial transaction with all the evidences and the billing of the contract.
- **List my chats**
  - **Actors:** User
  - **Purpose:** Provides the user a list of the current opened chats with other users.
  - **Description:** The system will query the current chats for a user, either if they are the product owner or the client.
- **List my formal offers**
  - **Actors:** User
  - **Purpose:** Provides the user a list of the current formal offers with other users.
  - **Description:** The system will query the current formal offers for a user, either if they are the product owner or the client.

5 General architecture

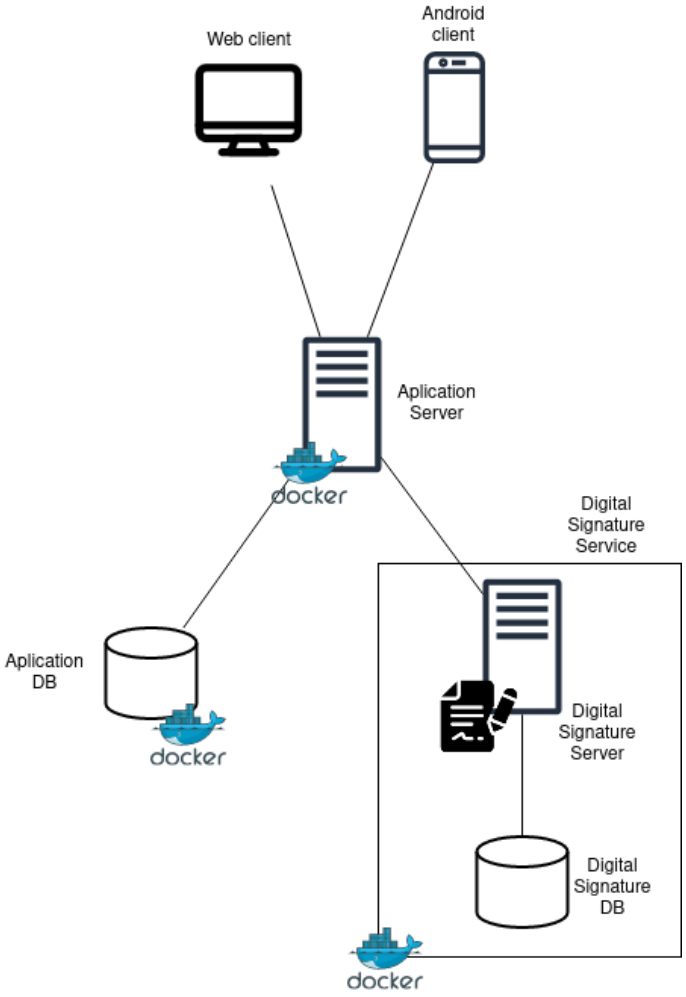

Figure 6: General architecture of the application.

5.1 Chat implementation

In this subsection it shall be described how and why the Chat implementations was done as it is today in the backend. The technologies that were used for providing the Chat was a simple WebSocket. They have been a standard since 2015 and have been widely adapted to browsers, as well as in flutter, so it is an easy development on both web and mobile app.

As for the server, this is done with an official Django library called *django-channels*, but the requirements on the server change to be able to hold this application. The websockets are connected to a *room chat*, that resents every message to all the others clients that are connected on this room.

This happens asynchronously in the server, which makes the integration for this server so different from the other services, as the server runner has to be changed accordingly to be able to

hold the *asgy* application.[1]

Another problem that we had upon the application was that websockets don't have memory, so you can't ask for the messages from a chat in the websocket. At first, it was thought that this would be managed by the library itself, as it uses Redis for escalating the websockets, but it was found out that it cannot achieve this use case. For this reason, the database model was changed to hold the messages from a chat.

So, seen it as a sequence diagram, the chat has to be implemented following the next calls:

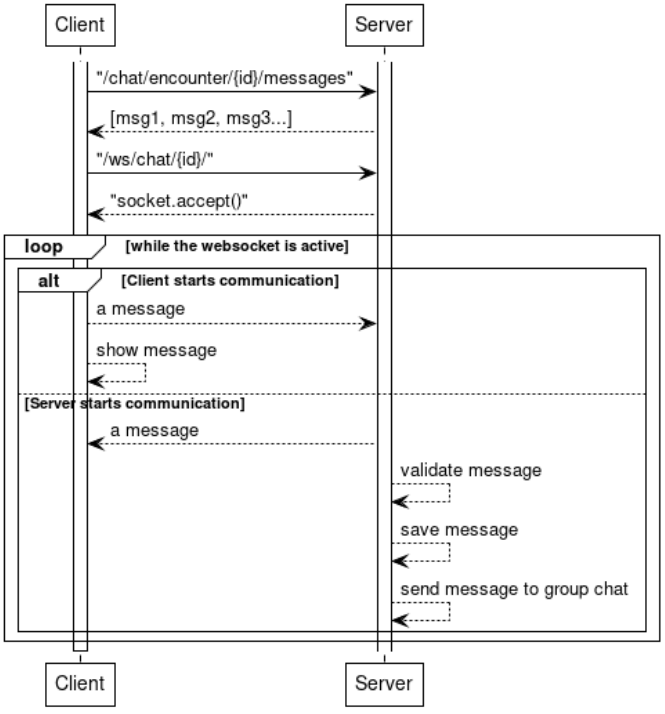

Figure 7: General architecture of the application.

5.2 Mobile app architecture

In this subsection, the decisions regarding the architecture of the mobile app will be explained, but not those that regard the user experience of the mobile. This will be explained in more detail at 7.

As it was said in the former spring, the mobile application development is still done with Flutter. The reason of why the app is implemented by flutter and not other frameworks, as Android or React Native, will be explained in 5.2.1. Then, it will be introduced the Redux for the state management of the app in **??**. Finally, the changes of the mobile application architecture for this sprint will be in **??**.

[1]Python has two standards for web servers, one called *wsgi* for synchronous servers, and the other called *asgi*. Django was done with *wsgi* in mind, but, in recent years, development has been made so that it can use *asgi*.

5.2.1 Flutter

Flutter is a multiplatform framework aiming to become a standard for building apps that have to work on either Android or iOS, as well as the web, desktop, and embedded operative systems.

As a tool, it has been proven that it can build more resilient and optimized apps than its counterparts, like React Native, while still being multiplatform. Even though it was taught how to work with native Android Applications, it was decided to choose this tool against it as the team had already some experience with it.

Even if the framework is young, the community behind flutter is active and has several repositories that have some application examples, as well as a libraries showcase that has and will help the development of the app.

5.2.2 Redux

State management in applications is a hot topic, and although both frameworks used by the clients in this project have some way to implement it, it was preferred to use a third-party option that is well consolidated, easy to use, and easy to manage.

Redux is an abstraction that provides a way to implement easier a reactive pattern, that is, instead of having the typical Model-View-Controller, it has a way of drawing a State, the buttons can launch actions, that is taken by reducers along the actual state, and return the next state, as shown at 8.

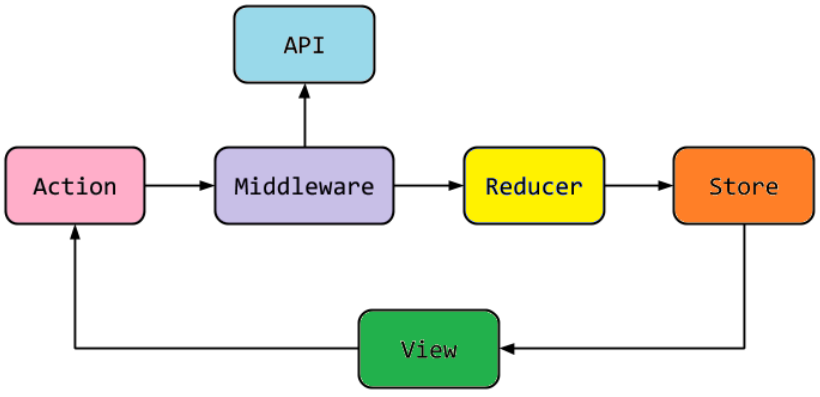

Figure 8: Redux Diagram.

This makes the components stateless, which makes it easier to work with them and provides a

way of mesmerizing them so they aren't computed that many times, making them faster than their counterparts.

This also helps to have a way of monitoring the changes that the state has suffered, as you can keep the states and the actions in a list for debugging, so it can be traced easily the changes. This adds when you have to have a shared object that won't be modified easily, but has to be read by most of the components, as an API token or a theme object. In order to do that, the different functionalities of the mobile application have been splitted and sorted. Therefore, it has been created in every package a `store` and `action` dart files which contain all the logic of the management of the store globally.

Finally, as it was also used in the web applications, it makes the team be able to change quickly between the two frameworks, as the most difficult part is done the same way.

5.2.3 Service Package

Regarding the connection to the backend, it has been created a package to manage all the data received and turning the data transfer objects into data models, which interacts to the logic and design of the application. In *Figure 8*, it can be seen how this package is connected through the app.

5.2.4 Asynchronous Actions

In order to connect to the backend API, it is compulsory to use asynchronous actions. Even though, Redux only provides pure functions dispatches and the state can not wait for asynchronous calls. In order to do that, it was needed a higher layer to do this communication. A good way is implementing a middleware between dispatching and reducing the action to a new state.

The middleware would catch the new middleware actions, which could be asynchronous, and then dispatches the other logical actions. A library called `ReduxThunk` was imported to make the middleware easier. The thunk actions provides making asynchronous calls returning the store as the effect of the action instead of the state.

A clear example is `clearSearchAndReload`, which you can dispatch lower level actions and do asynchronous calls inside the actions.

```
ThunkAction<AppState> clearSearchAndReload() {
    return (Store<AppState> store) async {
        store.dispatch(ClearSearch());
        final HashMap<int, Product> products =
```

```
        await store.state.commedMiddleware.getProducts();
        store.dispatch(SetProductList(products));
    };
}
```

5.2.5 WebSocketChannel

The part of connecting to the WebSockets in order to send and receive messages without doing gets and posts is more or less trivial. The library . One of the most challenges in this was connecting correctly and getting the messages correctly through the channel and then, reload the state and dispatching the store to display the new information. The library used for connecting to the WebSocket is called `web_socket_channel`.

5.3 Web Architecture

5.3.1 Reason on implementing a Web Client

As the application has its target user in the administrative world, it was decided that a web will be created alongside the mobile application that will make the usage of our services easier to our customers in administrative environments. It has been decided that the mobile app can be really useful for doing little tramits, for chatting with some customers, for quick searches of products/services and enterprises...
But it has been felt that in order to make it more usable and accessible for administrative porposes a web client should also be created.

Furthermore, the implementation of the web client shined some light on how to communicate with the backend and has lead us to iterate a little bit on the API endpoints that were implemented on the last sprint. This is nice because now on the third sprint the integration of the mobile app with the backend will be a lot more straightforward.

5.3.2 React

For implementing this Web Client React has been used, which is a library built in javascript for creating User Interfaces. To be more precise React was used because it gives the opportunity of working with Hooks. Hooks are functions that comprehend inside them inmutable components with independent states, wich makes the code a lot more clean and structured and simplifies the amount of chaos that has to be dealt with usually when it's programmed with vanilla javascript.

Also React has a huge community, and it has a lot of libraries that have been key for making a lot more easier the programming task. Some examples can be:

- react-router-doom
- react-modal
- redux-react-session
- react-bootstrap
- ...

5.3.3 Redux

In this part of the project, redux has also been used. In fact a react library called redux-react-session that builds a store to maintain the state of our session is being used. Also it creates the correspondig Session Cookie to maintain the value of the authorization token provided by our backend. This way, every time a request that needs authorization has to be made, it will be possible to pass the value of the token stored in the browser cookie within the request .

5.3.4 Web Client Architecture

By the moment the Web Client is still in a development phase, so for now the default testing server that React provides to us is being used. Although, a nginx+docker is expected to be used when deploying our application to a production enviroment.

Right now the application is able to fully connect with the backend, beeing able to Register, SignUp and interacting with all the different features that our application provides.

Also, the chat has been implemented using websockets. For doing so, we used a React library called react-use-websocket that has been used as a wrapper of the complexity of connecting with the client websocket to the server.

5.3.5 Challenges

The management of files and images in the front and in the communicatios with the backed has been a really challenging topic. For doing this, different approaches had been tryed and finally we decided to pass the images encoded in base64 with in the json request.
The first aproach that was tryed was using the form-data Content-Type. We didn't succeed on parsing in the backed the information that was beeing send using this protocol so we decided to switch. After some more attempts and a lots of ours of search for solutions. We found that a possibility was sending the images in b64 and then decode those images and parse them into Django FormFields.

Another Challenging topic has been the chat. As we have different kinds of messages, wee

had to agree in a protocol with the backend for retrieving the informations of the messages. Also, the implementation of the comunication through the web socket was not trivial, keeping the chats updated at real time with all the events that happen.

Finally, one of the biggest challenges has been making the web usable. This includes making the web responsibe, with all the components that had been developed separetly nicely connected. As we said in our idea presentation, we want to build a really confortable expirience to the user, showing there that our application really makes it's live easyer. Eventhough we put a lot on effort on this, a lot of work is still remaining in this part.

6 Database model

The database model can be seen at figure 9.

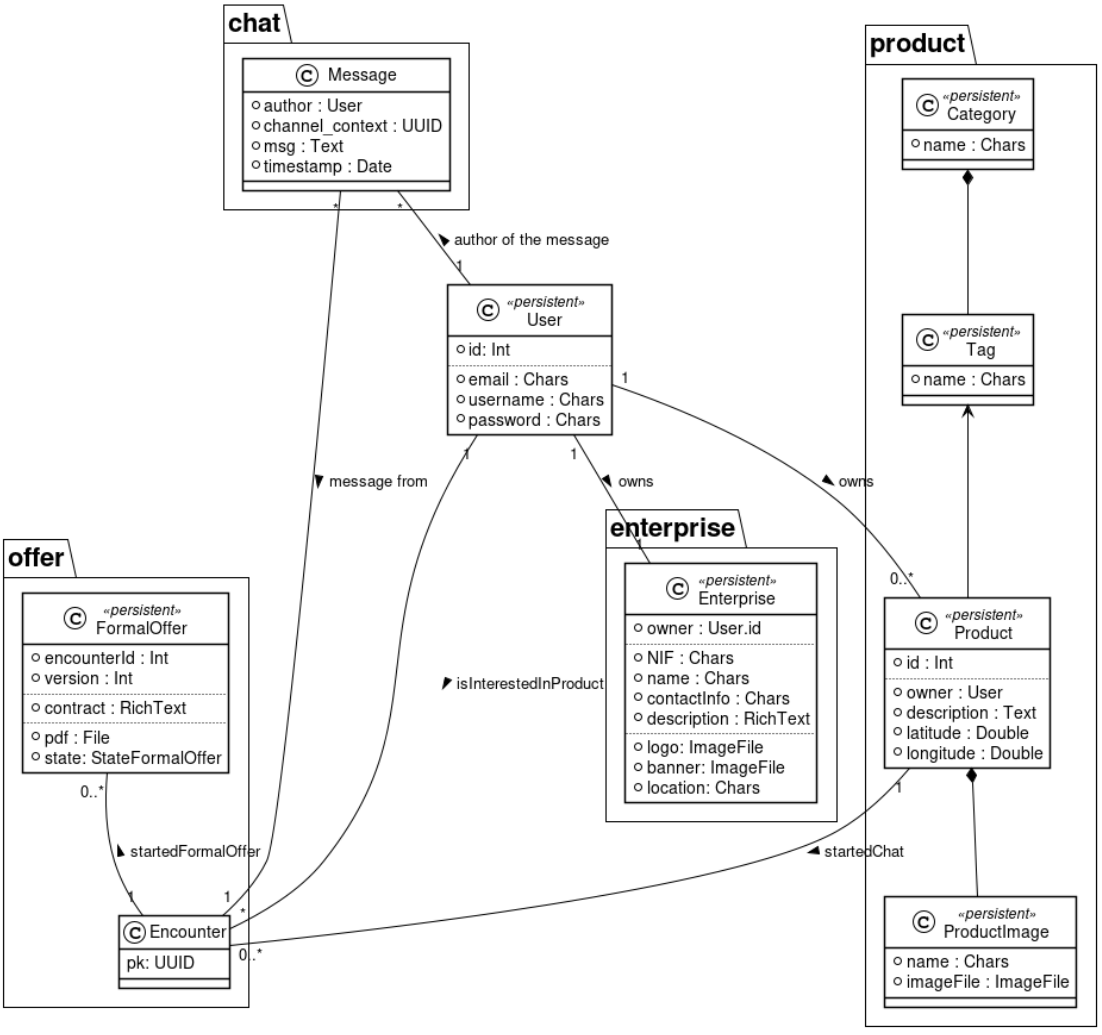

Figure 9: UML diagram of the models.

6.1 Modifications on the models

The logo and the banner for an `Enterprise` had been finally added to the enterprise model, as well as location of it. It has also been changed the `FormalOffer` model, changing the name of `signedPDF` to `PDF`, and allowing nulls on it, as some formal offers won't have any documents. It was added a field for storing the state of the formal offer, and although now it has only two states possible, some detected use cases have more than the naive thought that a document can either be signed on unsigned. It has been made it an immutable model, that is, it can only create instances and it cannot modify any of them, as it was thought that adding in the version of the document with another state was better than updating the database. This lets the chat mesmerize the state of a formal offer, so it doesn't modify it overtime if they patch the formal offer.

Finally, it has been added a Message model, that holds the information on the messages, and having fields regarding the author, the time of creation, the author and the message itself. This last field stores a JSON with the values that were sent on it, as it will only be needed in a JSON format and won't be queried against its parameters. Moreover, it simplifies the querying of a normalized database, as it holds different types of messages, which for now it's only `FormalOffer`s and text messages.

7 Main Screens

7.1 Mobile Application

7.1.1 Application

In this section, it will be shown the changes in the views of the mobile application. It has been made some changes regarding authentication state in order to make the application more usable and accessible.

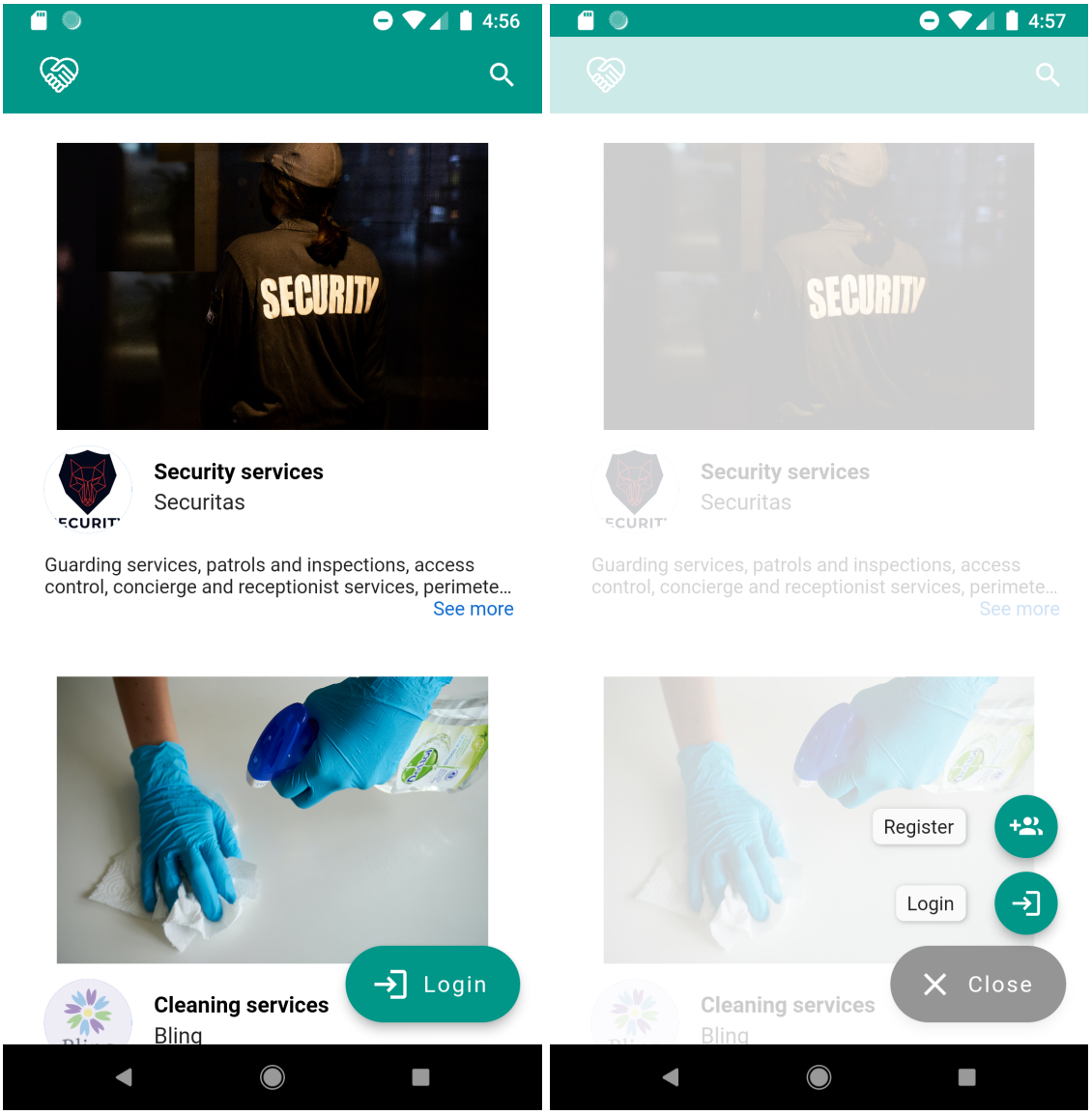

Figure 10: Home mobile screen, it lists the products

It has been put the profile button in the bottom menu bar as it must appear a label next to the icon. Also, it has been created the label next to the connect button to understand what means.

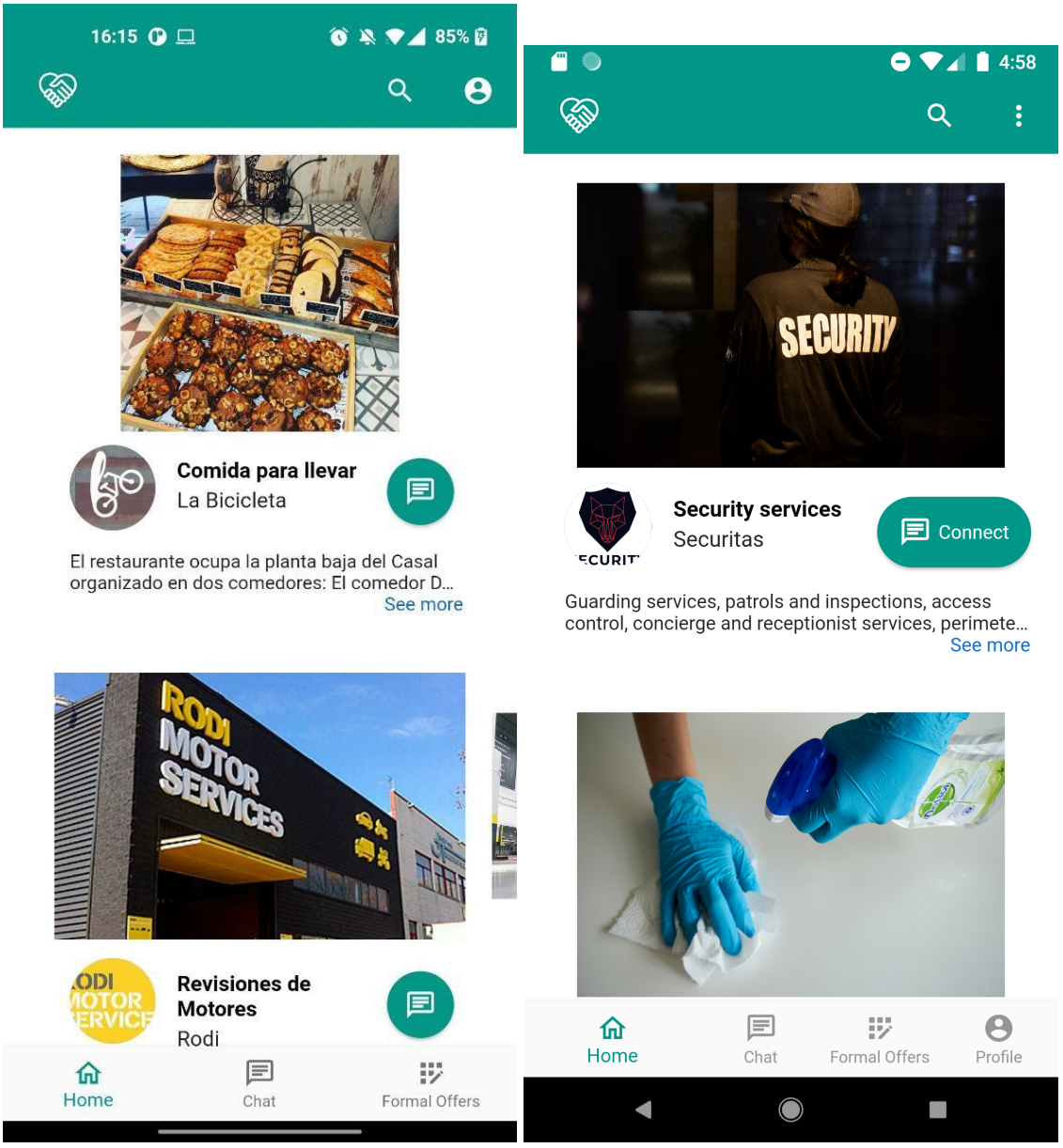

Figure 11: Home mobile screen, it lists the products. Former / New

It has been added new information in the enterprise profile. For the next sprint, it will be required to be a button to access to a GPS application.

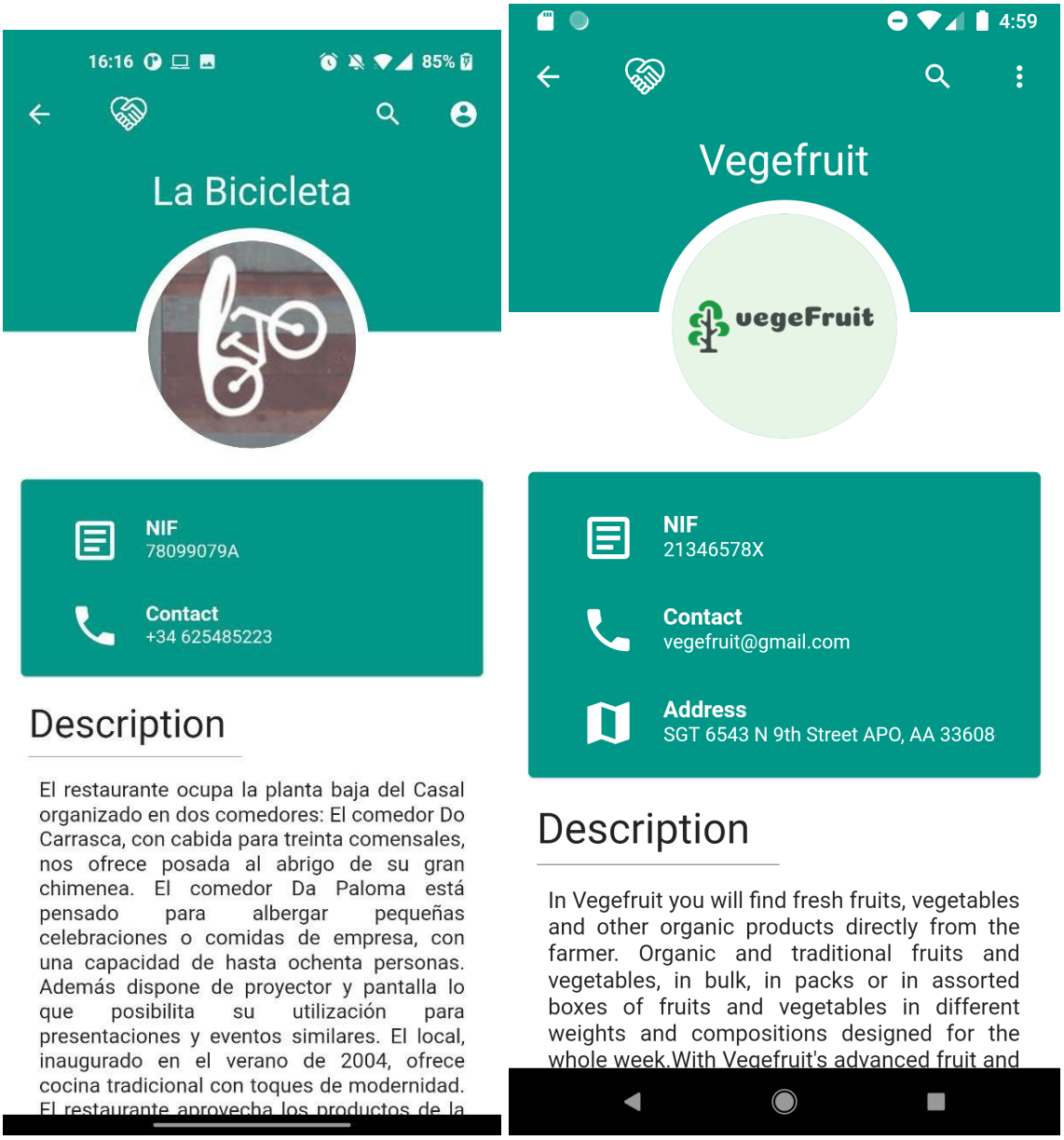

Figure 12: Detail of an enterprise, it can be found when clicking the logo of it. Former / New

It has been made some tiny design changes in the chat view to fit more in the application.

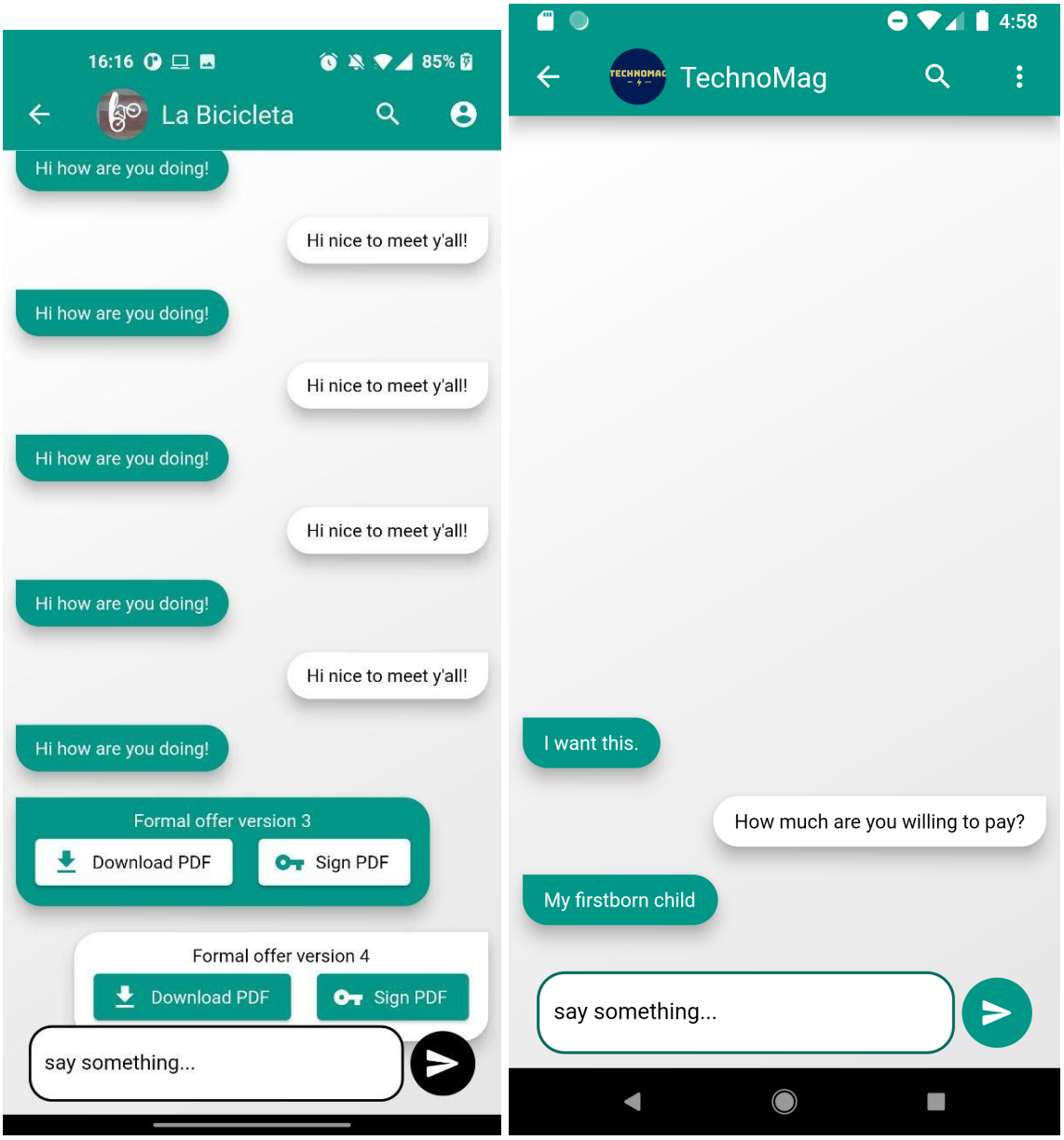

Figure 13: Chat screen. Former / New

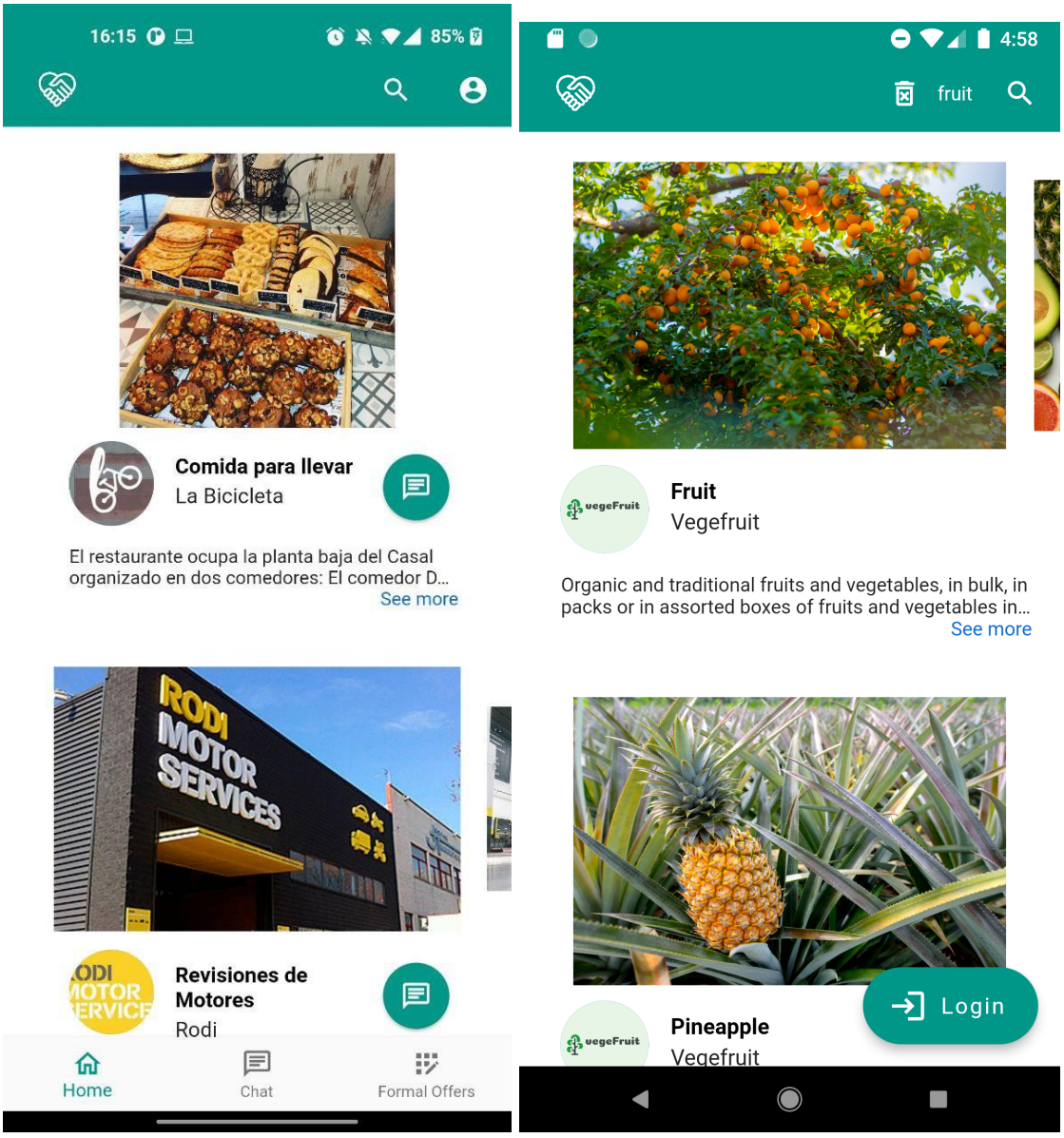

Figure 14: Search products view. Former / New

7.2 Web Client

7.2.1 Application

In this section it will be shown the changes that have graphically occurred in the views.

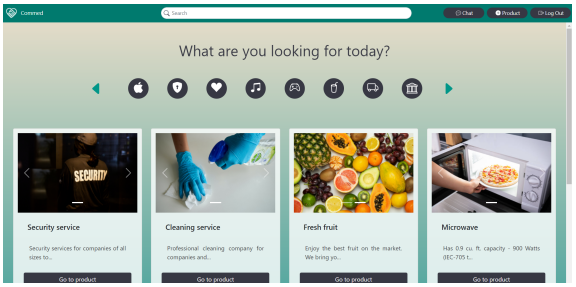

Figure 15: Home screen.

8 Dailies

8.1 20-11-2021

- Creation of the items that shall be done by the end of the sprint
- Emina will work on polishing the app.
- Yassine will work on an accesibility and usability of the app.
- Quim will work on the product images CRUD, as it has a bug that needs to be fixed.
- Oriol will work on polishing the app, and connecting some endpoints in the app.
- Sergi will work in the implementation of the Chat in the backend.

8.2 21-11-2021

- Everybody still worked on the same issues as the last time

8.3 22-11-2021

- Everybody worked on the same issues, and Oriol started focusing on integrating the application with the backend.

8.4 29-11-2021

- Everybody worked on the same issues, but Emina finished her work.

8.5 30-11-2021

- Emina and Sergi paired to solve some bugs on the chat application, as it didn't work on docker but it worked on local.
- All the other people were progressing on the same issues.

8.6 1-12-2021

- Everybody worked on the same issues.

8.7 10-12-2021

- The issues regarding the chat were discovered, so Sergi started working alone in fixing those issues.
- Emina started working on populating the database with more accurated users, to make the demonstrations better and the debugging of the application easier.

8.8 11-12-2021

- Sergi finished the chat, and began implementing custom endpoints for listing all the chats and formal offers needed by the web application and the mobile application.
- Emina finished the database and started connecting the chat with the application.
- Quim aided Emina in the integration with the chat while he was stuck debugging the product images.
- Oriol integrated most of the app.

8.9 12-12-2021

- Sergi changed some features for the chat, as they were requested to make the implementation easier for both clients.
- Emina debbuged the chat.
- Quim aided Emina in the integration with the chat, and made the product images CRUD work.
- Oriol integrated most of the app.

8.10 13-12-2021

- Sergi aided Emina, Quim and Oriol, and also did the merges on two branches.
- Emina debbuged the chat and polished things
- Quim developed the images publishing on the frontend.
- Oriol integrated most of the app.

8.11 14-12-2021

- Sergi aided some help and worked on the documentation.
- Emina finished the chat.
- Quim aided Emina.
- Oriol integrated the app completely.

8.12 15-12-2021

- Sergi Quim and Oriol worked on the documentation
- Emina worked on the slides.
- Sergi hepled Nico put the financial case on the documentation.

9 Financial Case

9.1 Monetization Strategy

The monetization strategy of Commed is mainly based on a percentage of the value of the commercial agreement contracts. When a contract has been finally set, an invoice will be generated. The value of the invoices depends on the value of the generated contract in order to make the invoices affordable for all types and sizes of companies. Therefore, in the first four years, the search of the companies and the other features will be free. After these beginning years, it is something to study and plan if some other features can become freemium.

In order to get more publications in the first few years and popularize our service quickly, the first three publications will be free for any company. The next ones will cost 250 € per year, which will not depend on the size of the company publishers. In addition, any kind of publication could get a better position in the search by paying monthly. The fee of this kind of advertising will be up to 450 € per month and the position will depend on how much the company pays for that advertising.

Therefore, there are three possible incomes:

- 5% commission of each contract generated.
- Publish a 4^{th}, 5^{th},..., n^{th} offer cost 250 € per new offer each year.
  Publishers have 3 free publications.
- Payments for advertising by month, the payment will be chosen by the customer in the same way it is done on instagram and facebook. The more the company pays for the advertising, the more the offer will appear. The maximum fee should be 450 € per month.

As the search will be free, the companies could be able to negotiate and set the contract out of our reach. In order to sort this inconvenience out, Commed will finally generate the contract making this process so easy and the invoice very affordable as the payment is only the 5% of the contract.

9.2 Marketing Strategy

According to the marketing strategy, it has been set that the main focus in the first years will be getting the small and medium sized companies, in order to quickly popularize the platform and create small business ecosystems. For example, the main ecosystems to focus on in the first year will be in order of importance:

- Restoration, butchers and other food providers.
- Food industry and supermarkets.
- Organizations for seasonal work to get temporary contracts.
- Cleaning services.
- Security services.
- Logistic services.

After creating the small ecosystem, the focus will be on strengthening the ecosystem and widening the smaller ones in order to join with the others and make them become a greater one.

This Sprint, we decide to be more specific about our Marketing Strategy. We choose to operate a door-to-door strategy to convince our first clients. Our sales mans will convince them by proposing the pros of using our service, and state our welcome offer. As for the advertising channels, we have chosen Linkedin, because we are in a B2B market, so our target are companies. As things progress, we rely on word of mouth to acquire more customers, both publishers and service seekers.

9.3 Speculation and flow chart

In order to create the simulation of the revenue, a speculation has been done on how many contracts will be held by our application in a month. This conjecture has been simulated over the first four years of the company. The revenue functions consists on these three variables:

- The income from the percentage of the contract between the entities. This variable has the main weight of the function as it is the main income generator. Mean value for small business contracts : 1000€. Mean value for big business contracts : 4000€.
- The revenue from the paid publications. In order to ease the simulation and know how many publications will be paid, a relaxation of the problem has been made. The paid publications ratio has been calculated with an approximation, which also is directly correlated with the amount of contracts held.

- The income from the payments for advertising a publication. This variable has also been approximated making a correlation from the number of contracts. We assumed that 5% of the publications are ads, for a cost of 30€ each.

According to the costs, its function depends on these variables:

- The cost of paying developers, which can be splitted into junior or senior developers. Compared to Sprint 1, we assume that the developers are not payed the first 6 months.
- Marketing cost.
- As the platform will be growing, the costs of the server will be higher. Despite that, it is thought about creating a serverless backend using AWS Lambda to minimize this cost and only pay for the requests.

Three scenarios have been made in order to show a possible but different projection of the incomes generated by the platform:

- A pessimistic scenario, where the number of contracts increases in a linear way amongst the four years.
- An optimistic scenario, in which the contracts' function has an exponential form.
- A more realistic scenario, which also has linear function in the first years but in the lasts, it gets more the way of an exponential function.

9.4 Pessimistic Scenario

Number of contracts per year :

Year	1	2	3	4
Small Business Contracts	518	1448	2262	3713
Big Business Contracts	0	0	381	1079

In this scenario, the number of contracts increase linearly. In comparison to the Sprint 1, the marketing cost are lower on the fourth year. This change is explained by the necessity to drop the costs, due to a pessimistic scenario.

Pessimistic	Years					
Project	**1**	**2**	**3**	**4**		
Income of contracts	19 250€	46 000€	129 350€	261 200€		
Income of publications	5 688€	8 750€	14 875€	24 500€		
Total Income	**24 938€**	**54 750€**	**144 225€**	**285 700€**		
Senior Employee	12 000€	24 000€	24 000€	24 000€		
Junior Employees	8 500€	17 000€	17 000€	17 000€		
Marketing	36 000,00 €	36 000,00 €	25 500,00 €	18 000,00 €		
Variables cost	288,75 €	690,00 €	1 371,00 €	2 475,75 €		
Total Cost	**56 789€**	**77 690€**	**67 871€**	**61 476€**		
Cashflow	**-31 851€**	**-22 940€**	**76 354€**	**224 224€**		
ROI %	**-156,09%**	**-129,53%**	**12,50%**	**264,74%**	**-2,10%**	
Net Income				**245 787€**		
PBP Years				**1,07339**		
PBP Month				**13**		
NVP				**222 936€**	**Interest NVP**	1,05
IRR				**107%**		
					Sales unit price	90,00 €
BEP	**53**	**72**	**63**	**56**	**Cost of unit**	0,75 €
Number of contracts	**385**	**920**	**1828**	**3301**		

Table 1: Pessimistic Cash Flow

$$\text{ROI} = \frac{\text{Total Income} - \text{Total Cost}}{\text{Total Cost}}$$

9.5 Realistic Scenario

Number of contracts per year :

Year	1	2	3	4
Small Business Contracts	518	1448	2262	3713
Big Business Contracts	0	0	381	1079

In this scenario, the number of contracts is the mean between the pessimistic and the optimistic scenarios.

Realistic	Years					
Project	**1**	**2**	**3**	**4**		
Income of contracts	25 900€	72 400€	189 300€	401 450€		
Income of publications	6 738€	14 000€	20 125€	36 050€		
Total Income	**32 638€**	**86 400€**	**209 425€**	**437 500€**		
Senior Employee	12 000€	24 000€	24 000€	24 000€		
Junior Employees	8 500€	17 000€	17 000€	17 000€		
Marketing	36 000,00 €	36 000,00 €	36 000,00 €	36 000,00 €		
Variables cost	362,60 €	1 013,60 €	1 850,10 €	3 354,40 €		
Total Cost	**56 863€**	**78 014€**	**78 850€**	**80 354€**		
Cashflow	**-24 225€**	**8 386€**	**130 575€**	**357 146€**		
ROI %	**-142,60%**	**-89,25%**	**65,60%**	**344,46%**	**44,55%**	
Net Income				**471 882€**		
PBP Years				**0,62321**		
PBP Month				**8**		
NVP				**428 011€**	**Interest NVP**	1,05
IRR				**231%**		
					Mean revenue per u	90,00 €
BEP	**53**	**72**	**72**	**72**	**Cost of unit**	0,75 €
Number of contracts	**518**	**1448**	**2643**	**4792**		

Table 2: Realistic Cash Flow

9.6 Optimistic Scenario

Number of contracts per year :

Year	1	2	3	4
Small Business Contracts	610	1801	2720	4415
Big Business Contracts	0	0	472	1372

In this scenario, the number of contracts increase exponentially. In comparison to the Sprint 1, the marketing cost are higher from the third year. This change is explained by the possibility to raise the costs, thanks to an optimistic scenario.

Optimistic	Years					
Project	1	2	3	4		
Income of contracts	30 500€	90 050€	230 400€	495 150€		
Income of publications	7 438€	17 500€	23 625€	43 750€		
Total Income	**37 938€**	**107 550€**	**254 025€**	**538 900€**		
Senior Employee	12 000€	24 000€	24 000€	24 000€		
Junior Employees	8 500€	17 000€	17 000€	17 000€		
Marketing	36 000,00 €	36 000,00 €	41 250,00 €	45 000,00 €		
Variables cost	396,50 €	1 170,65 €	2 074,80 €	3 761,55 €		
Total Cost	**56 897€**	**78 171€**	**84 325€**	**89 762€**		
Cashflow	**-18 959€**	**29 379€**	**169 700€**	**449 138€**		
ROI %	**-133,32%**	**-62,42%**	**101,25%**	**400,37%**	**76,47%**	
Net Income				**629 259€**		
PBP Years				**0,49130**		
PBP Month				**6**		
NVP				**570 756€**	**Interest NVP**	1,05
IRR				**361%**		
					Sales unit price	90,00 €
BEP	**53**	**72**	**77**	**81**	**Cost of unit**	0,75 €
Number of contracts	**610**	**1801**	**3192**	**5787**		

Table 3: Optimistic Cash Flow

9.7 Economic indices comparison between scenarios

All information related to the simulation and prediction about the first four years of the platform can be found in the spreadsheet attached to this document. Even though, the cash flows of each scenario can be found in *Tables 1, 2 and 3.* In this section, we will begin by comparing the three scenarios using the different indexes calculated above.

First, we want to show you the `ROI` index between the different scenarios, which can be found in Table 4. The `ROI` has been calculated every year, as it is more significant to us to analyze this information. Although it can be found that the `ROI` index in the first year is similary between the scenarios, the pessimistic scenario has some challenges to increase the ROI so as to be positive while optimistic and, actually, the realistic appear to have a better return on investment.

	ROI				
Project / Year	**1**	**2**	**3**	**4**	**Mean**
Pessimistic	-168,94%	-132,15%	-68,73%	41,15%	-82,17%
Realistic	-159,39%	-93,35%	-10,22%	166,56%	-24,10%
Optimistic	-152,81%	-67,50%	29,73%	249,98%	14,85%
Mean	-160,38%	-97,67%	-16,41%	152,56%	-30,47%

Table 4: ROI Index comparison between the three scenarios

According to the other indices, mostly all of them have better values in the optimistic scenario. One of the most valuable indices is the `PBP` in terms of months. Here, we can see that the optimistic has only 9 months of `PayBack Period`. On the other hand, realistic scenario is a promising case,

as in only one year and a month we would be able to recover the cost of the investment.

Although `ROI` gave us bad values, specially in the pessimistic scenario, we can see that in all the scenarios the `IRR` is positive. For the interest rate, we chose 1,05, that is quite similar to the inflation rate. That gave us good information to know if someone is going to invest on the company. The only one index that has the same behaviour in all the cases it is the `BEP`, which is calculated monthly. That is because the costs are fixed, as the lifecycle of the software would be large.

Indices	PBP (year)	PBP (month)	NVP	IRR
Pessimistic	1,07	13	222 936€	107%
Realistic	0,62	8	428 011€	231%
Optimistic	0,49	6	570 756€	361%
Mean	0,73	9	407 234 €	233%

Table 5: Indices comparison between the three scenarios

	BEP (monthly)				
Project / Year	1	2	3	4	Avg
Pessimistic	53	72	63	56	61
Realistic	53	72	72	72	67,25
Optimistic	53	72	72	72	67,25
Mean	53	72	69	66,66666667	

Table 6: BEP comparison between the three scenarios and years

In Figure 16, we can see the contracts' function of all the scenarios and the total cost function, which is the same in all scenarios. In the optimistic scenario, we will start to recover the initial investment at the beginning of the first year whereas in the worst case, we will start generating benefits at the beginning of the third year.

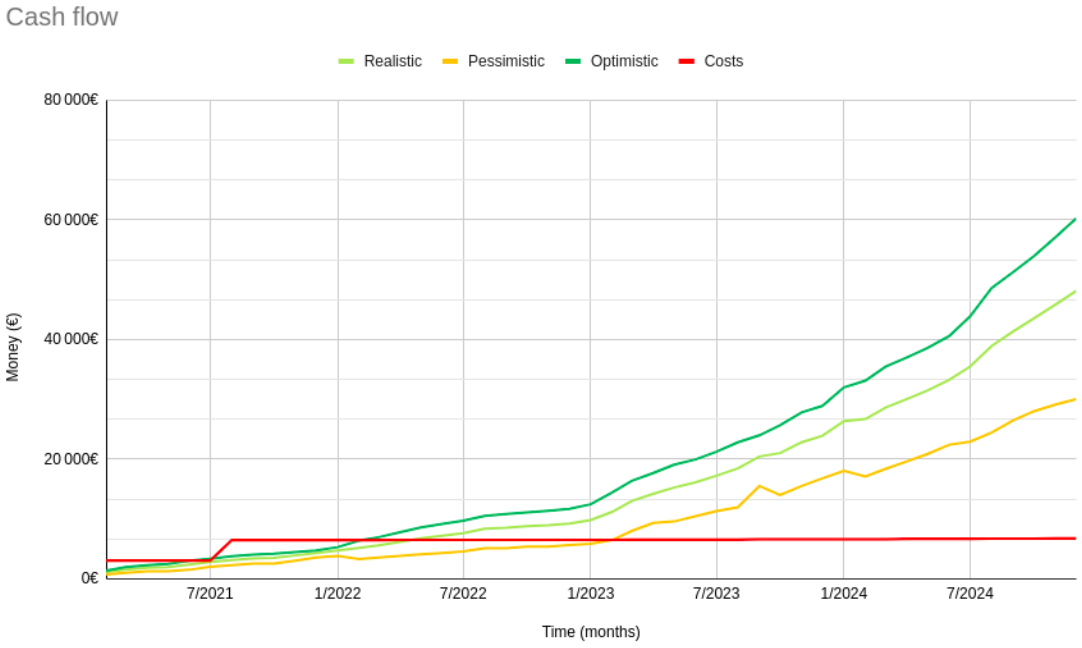

Figure 16: Cash flow of the different revenue function from scenarios and cost function