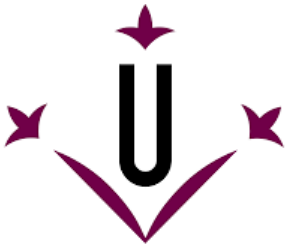

Universitat de Lleida

Sprint 2

Commed

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

Delivery
17th of November, 2021

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

Professorate:
Juan Enrique Garrido Navarro
Josep Escribà Garriga

Contents

1 Relevant links **3**

2 Planification **3**

2.1 User Stories . . . 3

2.2 Sprint 2 . . . 4

3 Requirements - Product **4**

4 Main use cases **5**

5 General architecture **8**

5.1 Mobile app architecture . . . 9

5.1.1 Flutter . . . 9

5.1.2 Redux . . . 9

6 Database model **9**

6.1 Users model . . . 10

6.2 Enterprise Module . . . 10

6.3 Product Module . . . 11

6.4 Formal Offer module . . . 11

7 Main Screens **12**

8 Web application **12**

8.1 Authentication . . . 14

9 Financial Case **15**

9.1 Monetization Strategy . . . 15

9.2 Marketing Strategy . . . 15

9.3 Speculation and flow chart . . . 16

9.4 Pessimistic Scenario . . . 17

9.5 Realistic Scenario . . . 18

9.6 Optimistic Scenario . . . 19

9.7 Economic indices comparison between scenarios . . . 20

List of Figures

1 Github project sceen . . . 4

2 Use case diagram of the application. . . . 6

3 General architecture of the application. 8
4 UML diagram of the models. 10
5 Admin login. 12
6 General dashboard. With this you can look all the tables and the last updates. . . 13
7 Detail of a Table. In this image , all the records in product can be looked on. . . . 13
8 Creation of a product as a form. 14
9 When a record is created, we can see it in the table page along with message. . . . 14
10 Cash flow of the different revenue function from scenarios and cost function 22

List of Tables

1 Pessimistic Cash Flow 18
2 Realistic Cash Flow 19
3 Optimistic Cash Flow 20
4 ROI Index comparison between the three scenarios 21
5 Indices comparison between the three scenarios 21

1 Relevant links

- GitHub backend
- GitHub docs
- GitHub project
- 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.

- **US1:** As a guest, I want to register in the application
- **US2:** As a user, I want to log in to the application.
- **US3:** As a registered user, I want to create a profile of my company.
- **US4:** As a guest, I want to search for services or products so that I receive said list.
- **US5:** As a guest, I want to have a detailed view of the product/service.
- **US6:** As a registered user who has a company profile, I want to create services/products.
- **US7:** As a user, I want to be able to *connect* with a company that has made a publication.
- **US8:** As a user, I need to be able to speak in a chat with the company that I connected.
- **US9:** As a company, I want to respond in the chat with the users that have sent messages.
- **US10:** As a company, I want to send a Formal Offer which contains a contract as a PDF through the chat.
- **US11:** As a company, I want to digitally sign contracts.
- **US12:** As a user, I want to digitally sign contracts.
- **US13:** As a user, I want to receive evidences of the process when a contract is signed.
- **US14:** As a user, I want to receive billing and invoices regarding the signed commercial transaction contract.

2.2 Sprint 2

The next step was creating a Scrum Board inside a Github project with the following structure:

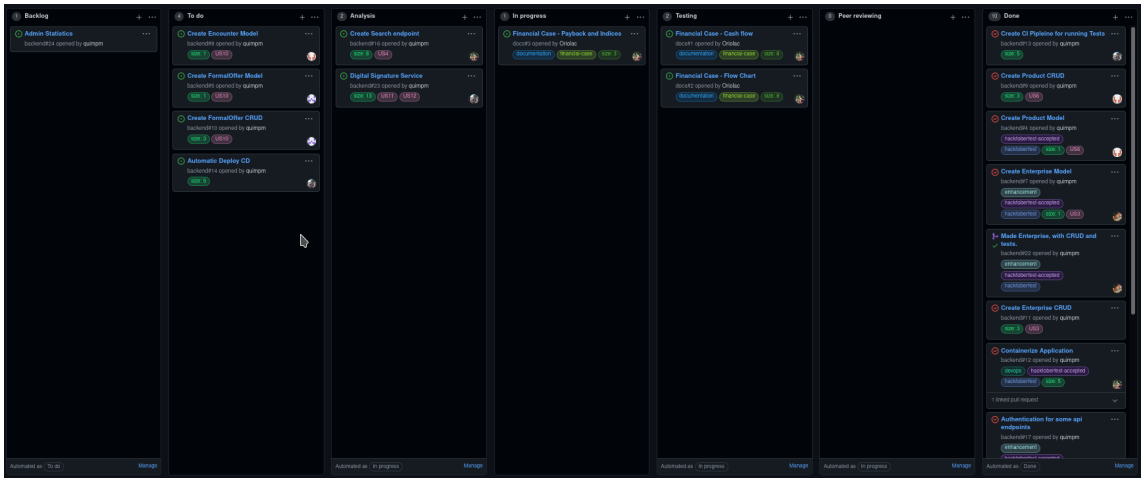

Figure 1: Github project sceen

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.

Later, all the tasks were described as a list of things to implement. Additionally, a weight was given for the complexity of the task and said weight has to be a number on the Fibonacci Sequence.

After that, all the tasks that had to be done in the First Sprint were moved in a *To Do* column in the project, which was the name that was given to the commonly said **Sprint Backlog**.

Finally, all of the tasks were self assigned, depending on how every member of the team wanted to contribute. By having each member choose their own work, it makes it easier to stay motivated.

Sprint 2 Backlog:

```
Add all the backlog for this sprint
```

3 Requirements - Product

In this section the list of requirements that the application has to offer to the user will be detailed:

Sprint 1:

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.
- 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 users create an enterprise profile if they are logged in.
- The application has to let logged users publish products or services.
- 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 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.

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.

Sprint 2:

Functional Requirements:

4 Main use cases

- **Register into application**
  - **Actors:** User

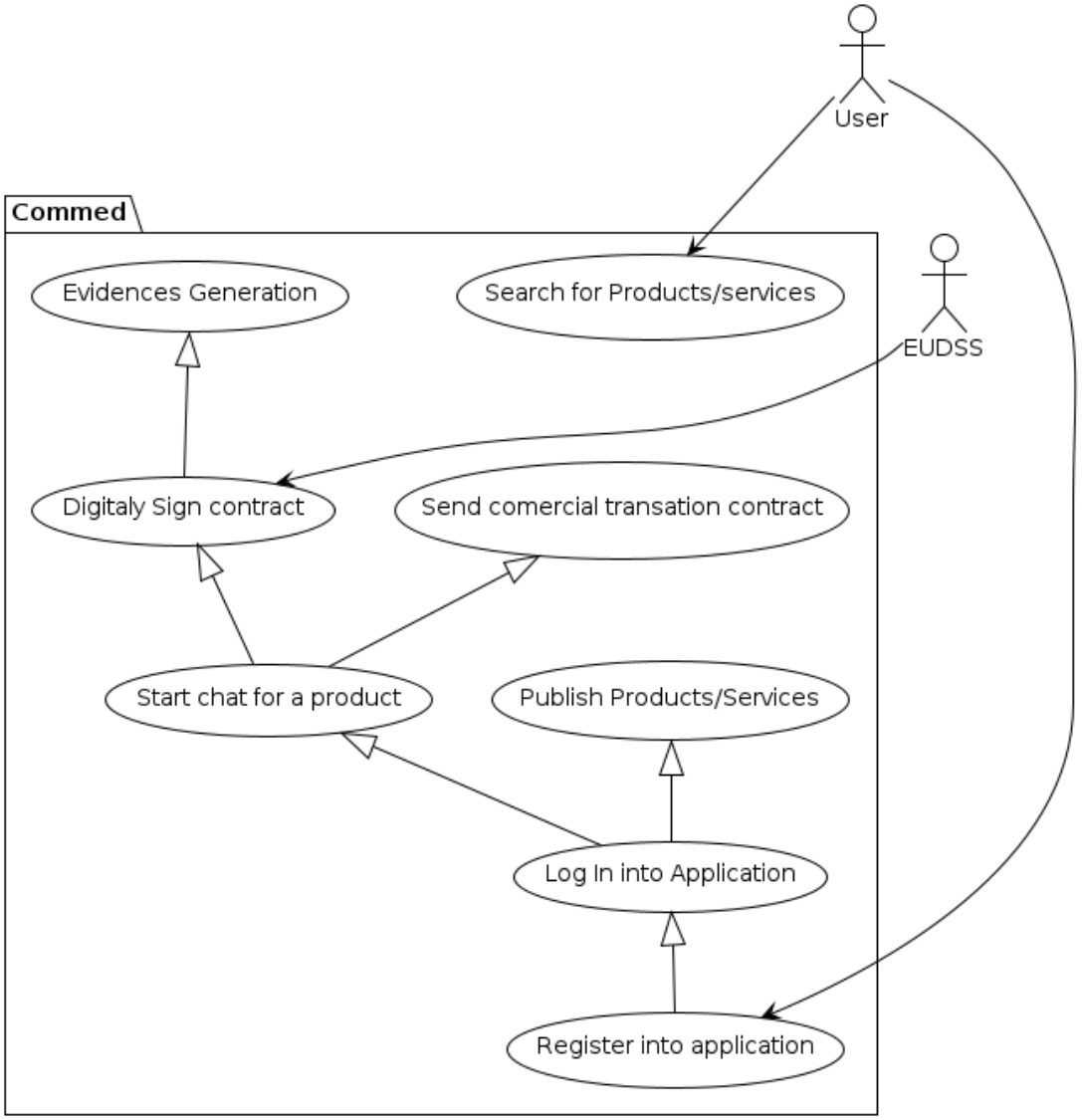

Figure 2: Use case diagram of the application.

- **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.

5 General architecture

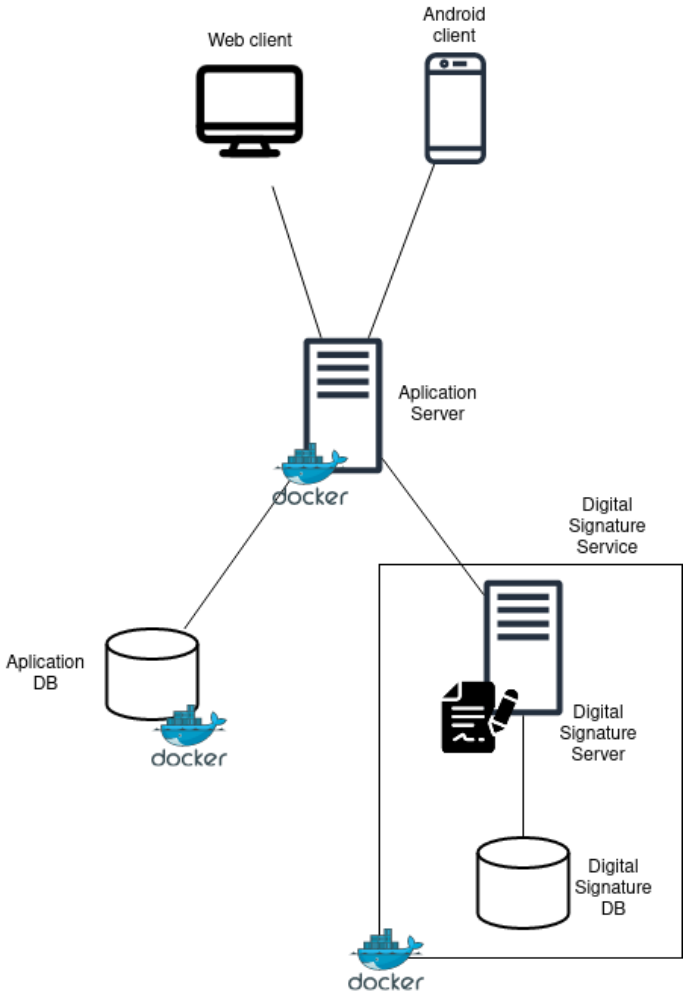

Figure 3: General architecture of the application.

The main item in the architecture is the Application Server. This will run in the backend, and furthermore, it will be responsible for most of the business logic.

This Service will consume resources for another two services. One of them is its own database, that will be isolated in a single container and will store all the application data. The other one is the digital signature service.

As digital signature is a complex thing by itself, it was thought that it should be isolated in another service, as the whole implementation of it has nothing related with the business logic held in the Application Service. So a Digital Signature service with its own database will be made, isolated in a container in which the Application server will consume resources.

Finally, the application has two clients that will be consuming resources from our Application Server.

The first one will be the Android client, which will have the whole user interface. Secondly, a Web client will be made with a similar user interface, but also administration pages for the database.

5.1 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.

First, why we chose flutter and not other frameworks, as Android or React Native, will be explained. Then, it will be introduced the Redux for the state management of the app.

5.1.1 Flutter

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

As a tool it has proven that it can build more resilient and optimized apps that its counterparts, as React Native, while still being multiplatform, as React Native. However, it was decided to chose this tool against a normal Android application as the team had already some experience with it. As a matter of fact, it was chosen to use React for the web counterpart instead of trying to use the same components made for the mobile for the same reason.

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

5.1.2 Redux

6 Database model

The database model can be seen at figure 4. In the following sections, it will be reviewed which models shape the application and which purpose inside the logic do they have.

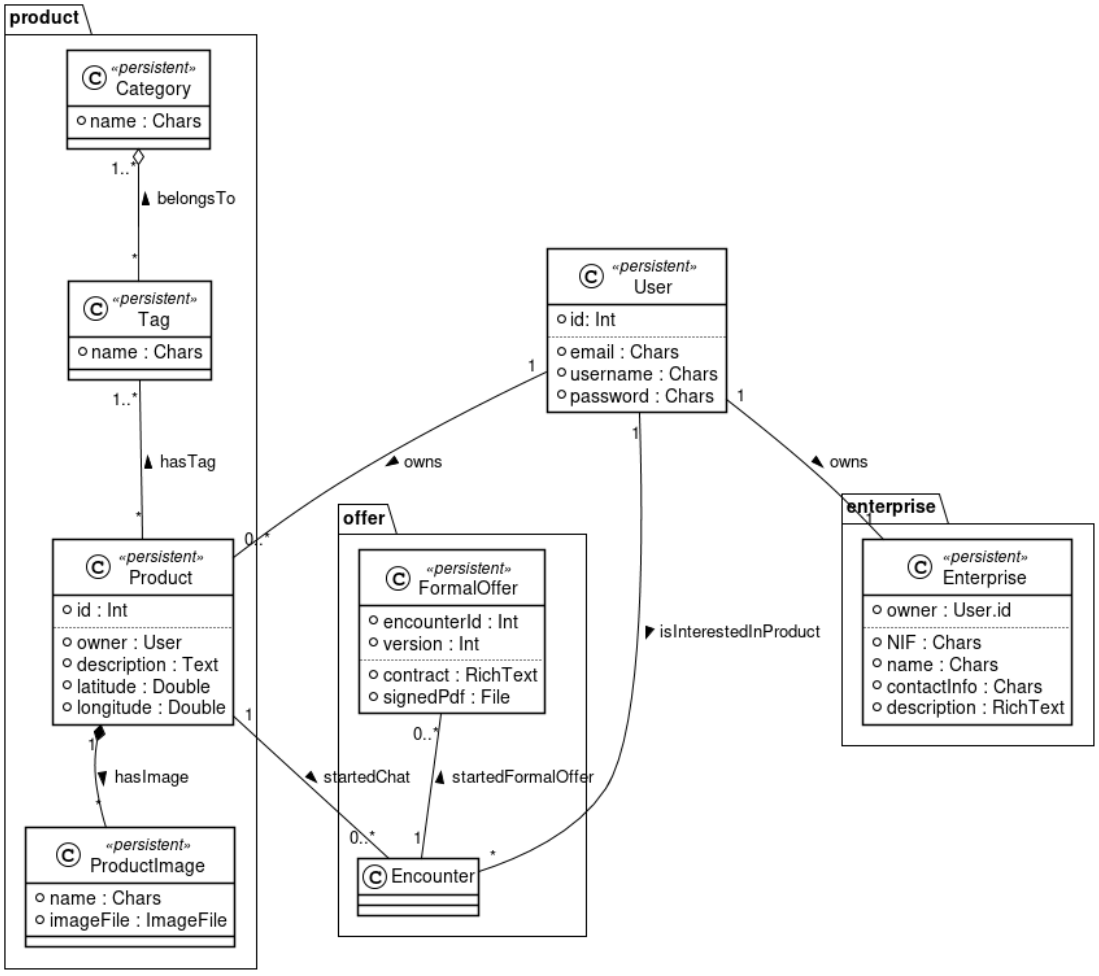

Figure 4: UML diagram of the models.

6.1 Users model

The users model is already provided by Django framework. Its purpose is to be able to store the usernames and passwords securely. Only the operations required for the authentication were needed to be done and not the CRUD operations on the User, based on a JWT Bearer schema. In 6.1 there is all the explanation regarding the way that the authentication works.

6.2 Enterprise Module

A well known practice inside the Django framework is to separate the user specific fields in another table instead of extending the User model. This way, 3rd party apps can be used, for example, for the authentication part. With this method, not only that the information such as the enterprise name, description or the fields for generating its contacts can be stored for a specific user in our app, but furthermore 3rd party apps that operate on the user can also use it.

6.3 Product Module

Each enterprise can post a number of products depending on the plans that they have subscribed to. This module models the necessary data that should be stored to do the database.

For the searcher endpoint, a way to categorize the product will be needed. Because of this, Spacy will be used, which lets us extract keywords and match against more general tags. For example, it can categorize “apple” as “fruit”. Then, when a client wants to search for a “pear”, instead of searching against the whole database, it will only search on the “fruit” tag.

To achieve this purpose, we store the tags that are currently used in a table, and we have a many-to-many relationship between the Product and Tag table. We made a table for the Tag instead of an enumeration as the Tag table will be increased programmatically when it encounters a new tag that does not relate to anything at all.

In the near future and given the fact that the clients won’t know about this inside feature, we will rearrange the tags to minimize the distance and provide a better support.

6.4 Formal Offer module

In this module we will have the models that have the responsibility of creating formal offers. The most important model is the FormalOffer, which contains the necessary information to create the unsigned PDF, as well as the current PDF. It iterates with different versions between the enterprise that offers the service and the one that wants to buy them, related to the times the formal offer is sent in the chat. At the end, the PDF will be agreed upon and signed by both enterprises.

A feature to provide some kind of template in the Formal Offer for the users was required. The ideal product is to have some kind of Natural Language Processing service that tries to create a Formal Offer based on previous formal offers and the messages in the chat. But, this task required a data to train the model that the application will not have in the next few years. Meanwhile, the previous contract of the same product could be given as a template. For this reason, a relation between the Product model and the FormalOffer model is needed.

The problem is that when the clients start a chat for a product, a Formal Offer is not created after some time. Additionally, there is no guarantee that a formal offer will be created for every chat that has been launched. For this reason, it was decided that creating a model that has a chat related to them and the product would be better than storing it in a FormalOffer where all the other values are Null. With this solution in mind, the Encounter model was created.

7 Main Screens

8 Web application

The administration use case for our application was done by the Django framework. This can be seen at the screenshots [5, 6, 7, 8, 9].

Additionally, some other form of documentation was provided by the endpoints, including OpenAPI and swagger docs. This lets the developers have a clear understanding of what can be done in the API, which will help later on the development of the Android and the Web application.

The relations in the dashboard are done as shown in the screenshots: the models are grouped in the application that registered them, and provide an interface to query, insert, update and delete items in the database.

It was decided that some statistics about the application should be on the administration tab, but it was left in the backlog for the third sprint.

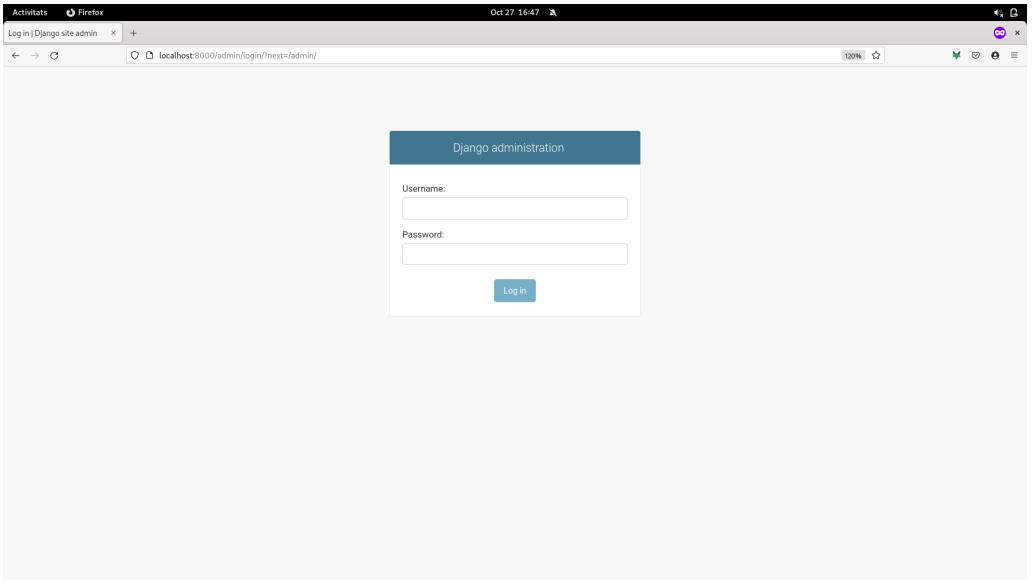

Figure 5: Admin login.

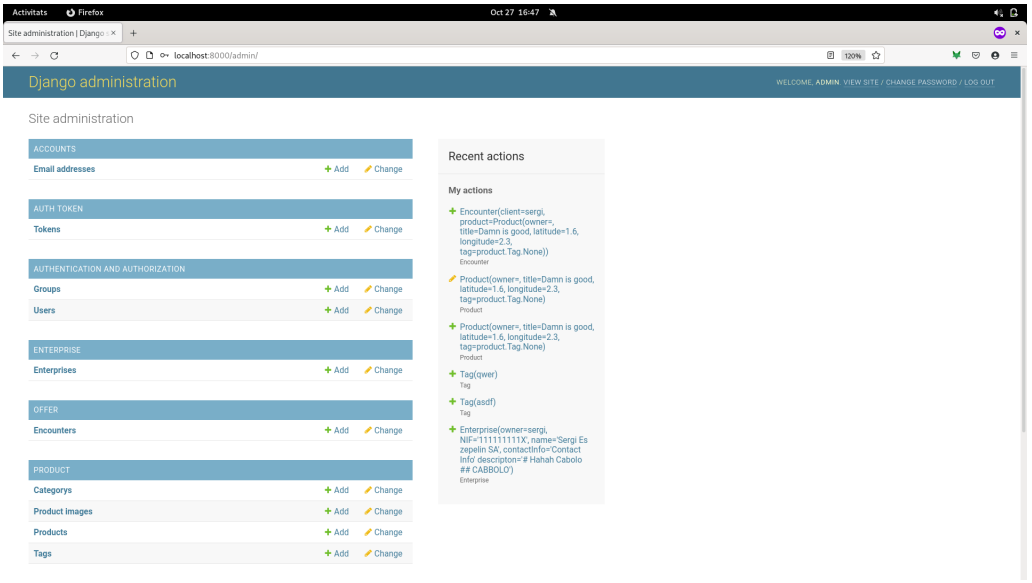

Figure 6: General dashboard. With this you can look all the tables and the last updates.

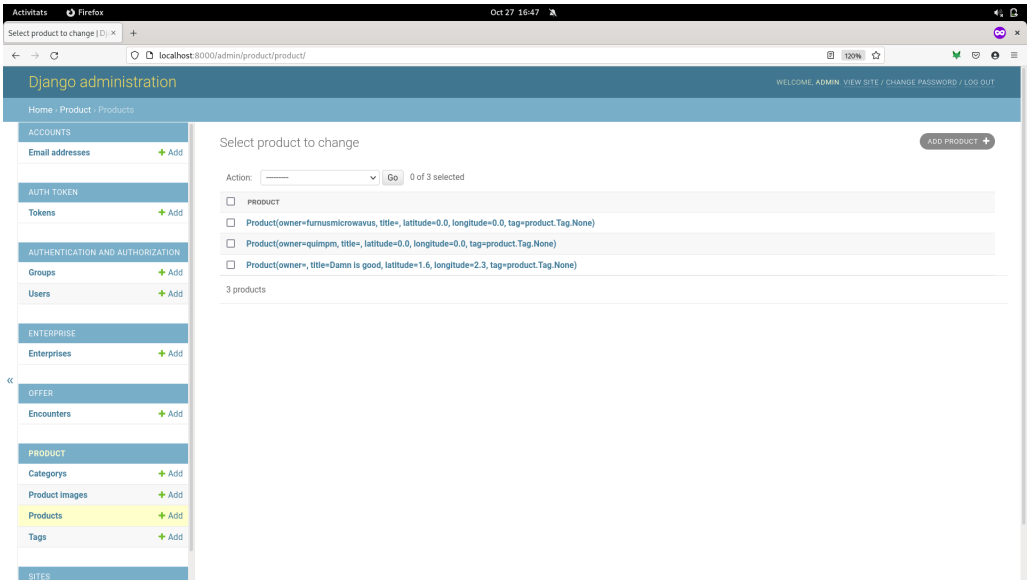

Figure 7: Detail of a Table. In this image , all the records in product can be looked on.

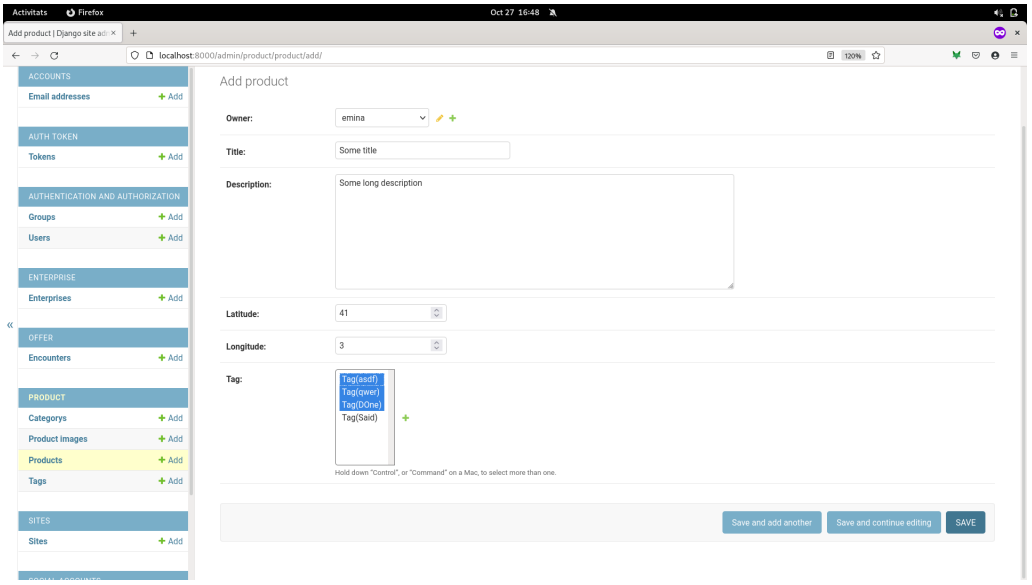

Figure 8: Creation of a product as a form.

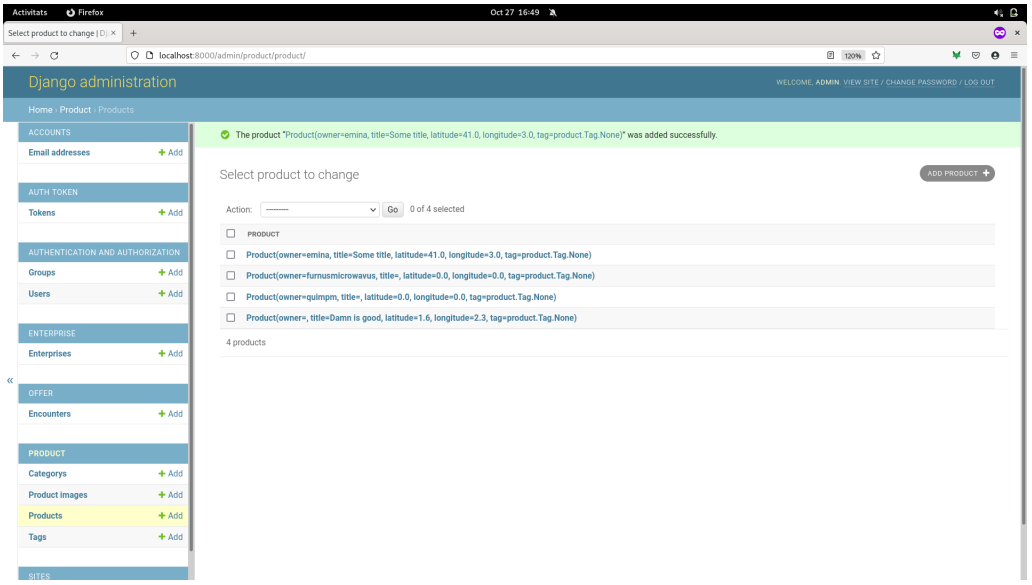

Figure 9: When a record is created, we can see it in the table page along with message.

8.1 Authentication

The problem was solved by using two well known apps in the Django community. The first is maintained by an Enterprise called *IntenCT* who specializes on doing web applications using Django. This enterprise published a library that provides an authentication system. It specializes on keeping users that come from different apps, while proving a solution for this based on tokens. With this, implementing simple JavaScript Web Tokens are just a step on the configuration, but also adding a way to integrate other services for authentication, as a **Log In with Google** button use case.

The other app that is used provides a REST entrypoint for the first one. In this way, with just a small configuration a ready-to-go service is achieved.

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.

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.
- Marketing cost.

Year	1	2	3	3
Small Business Contracts	65	100	170	280
Big Business Contracts	0	0	40	73

- 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 at the end of each year : In this scenario, the number of contracts increase linearly.

Year	1	2	3	3
Small Business Contracts	77	160	230	412
Big Business Contracts	0	0	60	137

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	26 000€	26 000€	26 000€	26 000€
Junior Employees	36 000€	36 000€	54 000€	54 000€
Marketing	18 000,00 €	18 000,00 €	28 500,00 €	36 000,00 €
Variables cost	288,75 €	690,00 €	1 371,00 €	2 475,75 €
Total Cost	**80 289€**	**80 690€**	**109 871€**	**118 476€**
Cashflow	**-55 351€**	**-25 940€**	**34 354€**	**167 224€**
ROI %	**-168,94%**	**-132,15%**	**-68,73%**	**41,15%**
Net Income				**120 287€**
PBP Years				**3,23664**
PBP Month				**39**
NVP				**109 104€**
IRR				**43%**
BEP				**75**

Table 1: Pessimistic Cash Flow

ROI = (Cashflow - Total Cost) / Total Cost

9.5 Realistic Scenario

Number of contracts at the end of each year : In this scenario, the number of contracts is the mean between the pessimistic and the optimistic scenarios.

Year	1	2	3	3
Small Business Contracts	85	200	270	500
Big Business Contracts	0	0	74	180

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	26 000€	26 000€	26 000€	26 000€
Junior Employees	36 000€	36 000€	54 000€	54 000€
Marketing	18 000,00 €	18 000,00 €	28 500,00 €	36 000,00 €
Variables cost	362,60 €	1 013,60 €	1 850,10 €	3 354,40 €
Total Cost	**80 363€**	**81 014€**	**110 350€**	**119 354€**
Cashflow	**-47 725€**	**5 386€**	**99 075€**	**318 146€**
ROI %	**-159,39%**	**-93,35%**	**-10,22%**	**166,56%**
Net Income				**374 882€**
PBP Years				**1,04321**
PBP Month				**13**
NVP				**340 029€**
IRR				**129%**
BEP				**75**

Table 2: Realistic Cash Flow

9.6 Optimistic Scenario

Number of contracts at the end of each year : In this scenario, the number of contracts increase exponentially.

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	26 000€	26 000€	26 000€	26 000€
Junior Employees	36 000€	36 000€	54 000€	54 000€
Marketing	18 000,00 €	18 000,00 €	28 500,00 €	36 000,00 €
Variables cost	396,50 €	1 170,65 €	2 074,80 €	3 761,55 €
Total Cost	**80 397€**	**81 171€**	**110 575€**	**119 762€**
Cashflow	**-42 459€**	**26 379€**	**143 450€**	**419 138€**
ROI %	**-152,81%**	**-67,50%**	**29,73%**	**249,98%**
Net Income				**546 509€**
PBP Years				**0,71710**
PBP Month				**9**
NVP				**495 700€**
IRR				**193%**
BEP				**75**

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**	**BEP (monthly)**
Pessimistic	3,24	39	109 104€	43%	75
Realistic	1,04	13	340 029€	129%	75
Optimistic	0,72	9	495 700€	193%	75
Mean	1,67	20	314 944 €	122%	75

Table 5: Indices comparison between the three scenarios

In Figure 10, 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.

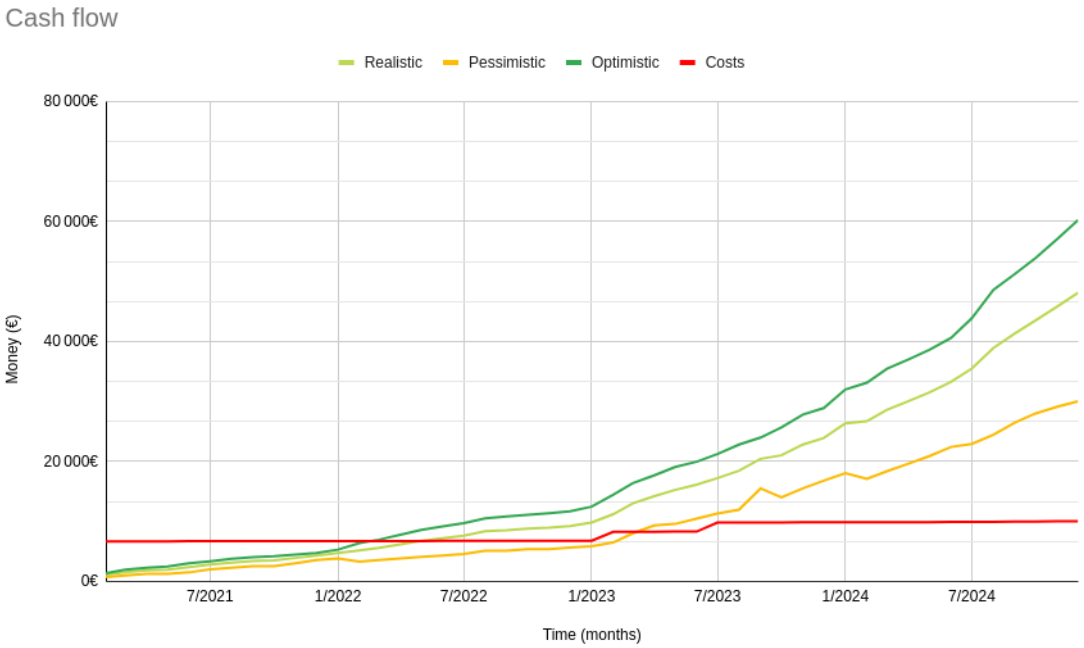

Figure 10: Cash flow of the different revenue function from scenarios and cost function