



Ministry of Higher Education  
and Scientific Research

ACADEMIC YEAR  
2018 / 2019

# EPI-Polytec

## POLYTECHNIC SCHOOL

INTERNATIONAL PRIVATE SCHOOL  
OF ENGINEERS

## FINAL YEAR DISSERTATION REPORT

In order to obtain the National Diploma in Engineering  
**Field of Study              Software Engineering**

### Entitled

**Web application for bakeries management**

&

**Mobile application for bakeries finder and passing orders**

| Internship place | Author          | Supervisor                        |
|------------------|-----------------|-----------------------------------|
| Anypli           | Ibtissem CHALBI | Bayrem TRIKI<br>Anis BEN BETTAIEB |

# Acknowledgment

I would like to express my deepest appreciation to all those who aide me to complete this report. A special gratitude is giving to my final year project manager, Mr. BAYREM TRIKI, whose contribution in stimulating suggestions and encouragement, helped me to coordinate this project especially in writing this report.

A special thanks goes to my supervisor Mr. ANIS BEN BETTAIEB, who guided me throughout the development of “**Cake it up !**” application.

Furthermore I would also like to acknowledge with much appreciation the crucial role of **Anypli** company, who gave the permission to use all required equipment and the necessary materials to complete this project.

Finally, I have to appreciate the guidance will be given by other supervisor, especially in my project exposition that will surely improve my presentation skills thanks to their comment and advices.

# Dedication

This report is dedicated to my mother **Fatiha**, for her kindness and devotion, and for her endless support, thanks for always being there for me. I would like to express my gratitude towards my father **Mohamed** who is my role model and mentor.

My sisters **Khawla, Imen, Ikhtiyar** and **Wafa** who were more than helpful morally, so proud to have them as my backup.

A special thanks goes to the one who is the most helpful and supporting person I know **Ahmed**, thank you so much for being there for me.

Finally I would like to thank my best friends **Sirine, Ons** and **Sarra** for standing by me at all costs, which made it easy for me to get past all the difficult situation.

# Table of Contents

|  |            |
|--|------------|
| <b>Acknowledgement</b>                       | <b>i</b>   |
| <b>List of Figures</b>                       | <b>vii</b> |
| <b>List of Tables</b>                        | <b>x</b>   |
| <b>List of Abbreviations</b>                 | <b>xii</b> |
| <b>General introduction</b>                  | <b>1</b>   |
| <b>Chapter 1: Project scope</b>              | <b>3</b>   |
| Introduction . . . . .                       | 3          |
| 1.1 Host organization Presentation . . . . . | 3          |
| 1.2 Project Presentation . . . . .           | 4          |
| 1.2.1 Analysis of the existing . . . . .     | 4          |
| 1.2.2 Criticism of the existing . . . . .    | 9          |
| 1.2.3 Solution . . . . .                     | 10         |
| 1.3 Used methodology . . . . .               | 10         |
| Conclusion . . . . .                         | 12         |
| <b>Chapter 2: Requirements</b>               | <b>13</b>  |
| Introduction . . . . .                       | 13         |
| 2.1 Requirements . . . . .                   | 13         |
| 2.1.1 Actors . . . . .                       | 13         |
| 2.1.2 Functional requirements . . . . .      | 14         |
| 2.1.3 Non-Functional requirements . . . . .  | 15         |
| 2.2 Project management with SCRUM . . . . .  | 16         |
| 2.2.1 SCRUM . . . . .                        | 16         |

|  |  |           |
|--|--|-----------|
| 2.2.2                                    | SCRUM roles . . . . .                      | 16        |
| 2.3                                      | Product Backlog . . . . .                  | 16        |
| 2.4                                      | General use case diagram . . . . .         | 20        |
| 2.4.1                                    | Use case: web . . . . .                    | 20        |
| 2.4.2                                    | Use case: Mobile . . . . .                 | 23        |
| 2.5                                      | Conceptual Data Modeling . . . . .         | 25        |
| 2.5.1                                    | Data dictionary . . . . .                  | 25        |
| 2.5.2                                    | Class diagram . . . . .                    | 28        |
| 2.6                                      | Sprint Planning . . . . .                  | 31        |
|  | Conclusion . . . . .                       | 32        |
| <b>Chapter 3: Project Initialization</b> |  | <b>33</b> |
|  | Introduction . . . . .                     | 33        |
| 3.1                                      | Choice of the architecture . . . . .       | 33        |
| 3.2                                      | Programming languages . . . . .            | 37        |
| 3.2.1                                    | Kotlin . . . . .                           | 37        |
| 3.2.2                                    | PHP . . . . .                              | 38        |
| 3.2.3                                    | JavaScript . . . . .                       | 38        |
| 3.3                                      | Used technologies and frameworks . . . . . | 38        |
| 3.3.1                                    | Android . . . . .                          | 38        |
| 3.3.2                                    | Laravel . . . . .                          | 39        |
| 3.3.3                                    | VueJS . . . . .                            | 39        |
| 3.3.4                                    | Swagger . . . . .                          | 41        |
| 3.4                                      | Used software . . . . .                    | 41        |
| 3.4.1                                    | Android studio . . . . .                   | 41        |
| 3.4.2                                    | Visual studio code . . . . .               | 42        |
| 3.4.3                                    | Laragon . . . . .                          | 42        |
| 3.4.4                                    | Star UML . . . . .                         | 42        |
| 3.4.5                                    | TexStudio . . . . .                        | 43        |
| 3.4.6                                    | SourceTree . . . . .                       | 43        |
|  | Conclusion . . . . .                       | 43        |
| <b>Chapter 4: Release 1</b>              |  | <b>44</b> |
|  | Introduction . . . . .                     | 44        |

|                   |                                 |           |
|-------------------|---------------------------------|-----------|
| 4.1               | Backlog . . . . .               | 44        |
| 4.2               | Use case . . . . .              | 47        |
| 4.2.1             | Use case diagram . . . . .      | 47        |
| 4.2.2             | Use case description . . . . .  | 51        |
| 4.3               | Sequence diagram . . . . .      | 54        |
| 4.4               | Implementation . . . . .        | 58        |
| 4.4.1             | Web . . . . .                   | 58        |
| 4.4.2             | mobile . . . . .                | 64        |
| 4.5               | Release retrospective . . . . . | 69        |
|                   | Conclusion . . . . .            | 69        |
| <b>Chapter 5:</b> | <b>Release 2</b>                | <b>70</b> |
|                   | Introduction . . . . .          | 70        |
| 5.1               | Backlog . . . . .               | 70        |
| 5.2               | Use case . . . . .              | 73        |
| 5.2.1             | Use case diagram . . . . .      | 73        |
| 5.2.2             | Use case description . . . . .  | 76        |
| 5.3               | Sequence diagram . . . . .      | 79        |
| 5.4               | Implementation . . . . .        | 82        |
| 5.4.1             | Web . . . . .                   | 82        |
| 5.4.2             | Mobile . . . . .                | 86        |
| 5.5               | Release retrospective . . . . . | 87        |
|                   | Conclusion . . . . .            | 87        |
| <b>Chapter 6:</b> | <b>Release 3</b>                | <b>88</b> |
|                   | Introduction . . . . .          | 88        |
| 6.1               | Backlog . . . . .               | 88        |
| 6.2               | Use case . . . . .              | 91        |
| 6.2.1             | Use case diagrams . . . . .     | 91        |
| 6.2.2             | Use case description . . . . .  | 94        |
| 6.3               | Sequence diagram . . . . .      | 98        |
| 6.4               | Implementation . . . . .        | 101       |
| 6.4.1             | Web . . . . .                   | 101       |
| 6.4.2             | Mobile . . . . .                | 103       |

|                                     |            |
|-------------------------------------|------------|
| 6.5 Release retrospective . . . . . | 104        |
| Conclusion . . . . .                | 104        |
| <b>General conclusion</b>           | <b>105</b> |
| <b>Bibliography</b>                 | <b>106</b> |

# List of Figures

|     |   |    |
|-----|---|----|
| 1.1 | Logo Anypli . . . . .   | 4  |
| 1.2 | Application: FACEBOOK <sup>0</sup> . . . . .                  | 5  |
| 1.3 | Google maps <sup>1</sup> . . . . .                            | 6  |
| 1.4 | Web site: Bakery days <sup>2</sup> . . . . .                  | 7  |
| 1.5 | Web application: Yelp (business owner) <sup>3</sup> . . . . . | 8  |
| 1.6 | Web application: Yelp (user) <sup>4</sup> . . . . .           | 8  |
| 1.7 | Cake It Up logo <sup>5</sup> . . . . .                        | 10 |
| 1.8 | Scrum processes <sup>6</sup> . . . . .                        | 11 |
| 2.1 | General use case: Web . . . . .                               | 21 |
| 2.2 | General use case: Mobile . . . . .                            | 23 |
| 2.3 | Class Diagram . . . . .                                       | 29 |
| 3.1 | Project architecture . . . . .                                | 34 |
| 3.2 | MVC pattern . . . . .   | 35 |
| 3.3 | MVP pattern . . . . .   | 36 |
| 3.4 | MVVM pattern . . . . .  | 37 |
| 3.5 | Kotlin vs Java <sup>0</sup> . . . . .                         | 37 |
| 3.6 | Vuex work flow . . . . .                                      | 41 |
| 4.1 | Use case Authentication . . . . .                             | 47 |
| 4.2 | Use case edit account . . . . .                               | 48 |
| 4.3 | Use case manage pastries . . . . .                            | 49 |
| 4.4 | Use case manage products . . . . .                            | 50 |
| 4.5 | Authentication sequence diagram . . . . .                     | 55 |
| 4.6 | Manage product sequence diagram . . . . .                     | 57 |
| 4.7 | Register interface web . . . . .                              | 58 |

|      |  |    |
|------|--|----|
| 4.8  | Login interface web . . . . .                    | 59 |
| 4.9  | Reset password interface web . . . . .           | 59 |
| 4.10 | Logout interface web . . . . .                   | 60 |
| 4.11 | Edit account interface web . . . . .             | 60 |
| 4.12 | List of pastries interface web . . . . .         | 61 |
| 4.13 | Edit pastry interface web . . . . .              | 61 |
| 4.14 | Product list interface web . . . . .             | 62 |
| 4.15 | Add product interface . . . . .                  | 62 |
| 4.16 | Edit product interface . . . . .                 | 63 |
| 4.17 | Profile non connected mobile interface . . . . . | 64 |
| 4.18 | Register and login mobile interface . . . . .    | 65 |
| 4.19 | Register and login mobile interface . . . . .    | 66 |
| 4.20 | Reset password mobile interface . . . . .        | 67 |
| 4.21 | Register and login mobile interface . . . . .    | 68 |
| 5.1  | Use case Manage clients . . . . .                | 73 |
| 5.2  | Use case Manage complains . . . . .              | 74 |
| 5.3  | Use case Manage reports . . . . .                | 75 |
| 5.4  | Use case Manage events . . . . .                 | 75 |
| 5.5  | Manage complains sequence diagram . . . . .      | 80 |
| 5.6  | Manage reports sequence diagram . . . . .        | 81 |
| 5.7  | List of clients interface web . . . . .          | 82 |
| 5.8  | List of complains interface web . . . . .        | 83 |
| 5.9  | Reply to complain interface web . . . . .        | 83 |
| 5.10 | List of reports interface web . . . . .          | 84 |
| 5.11 | Calender interface web . . . . .                 | 84 |
| 5.12 | Add event interface web . . . . .                | 85 |
| 5.13 | List of complains mobile interface . . . . .     | 86 |
| 6.1  | Manage reviews use case . . . . .                | 91 |
| 6.2  | Manage ratings use case . . . . .                | 91 |
| 6.3  | Statistics use case . . . . .                    | 92 |
| 6.4  | Manage cart use case . . . . .                   | 93 |

|      |   |     |
|------|---|-----|
| 6.5  | Manage orders use case . . . . .          | 93  |
| 6.6  | Manage personal orders use case . . . . . | 94  |
| 6.7  | Manage cart sequence diagram . . . . .    | 99  |
| 6.8  | Manage orders sequence diagram . . . . .  | 100 |
| 6.9  | List of reviews Interface web . . . . .   | 101 |
| 6.10 | List of orders Interface web . . . . .    | 102 |
| 6.11 | Edit order Interface web . . . . .        | 102 |
| 6.12 | List of orders mobile interface . . . . . | 103 |

# List of Tables

|      |  |    |
|------|--|----|
| 1.1  | List of disadvantages of the existing . . . . .  | 9  |
| 2.1  | List of actors. . . . .                          | 14 |
| 2.2  | Product Backlog. . . . .                         | 17 |
| 2.3  | General use case web explanation . . . . .       | 22 |
| 2.4  | General use case mobile explanation . . . . .    | 24 |
| 2.5  | Pastry class . . . . .                           | 25 |
| 2.6  | User class . . . . .                             | 25 |
| 2.7  | Product class . . . . .                          | 26 |
| 2.8  | Order class . . . . .                            | 26 |
| 2.9  | Review class . . . . .                           | 27 |
| 2.10 | Complain class . . . . .                         | 27 |
| 2.11 | Schedule class . . . . .                         | 27 |
| 2.12 | event class . . . . .                            | 28 |
| 2.13 | Address class . . . . .                          | 28 |
| 2.14 | Personal_product class . . . . .                 | 28 |
| 2.15 | Class diagram explanation . . . . .              | 30 |
| 2.16 | Sprint planning . . . . .                        | 31 |
| 4.1  | Backlog: Release 1 . . . . .                     | 44 |
| 4.2  | Use case "Authentication" description . . . . .  | 51 |
| 4.3  | Use case "Manage account" description . . . . .  | 52 |
| 4.4  | Use case "Manage pastries" description . . . . . | 53 |
| 4.5  | Use case "Manage products" description . . . . . | 54 |
| 5.1  | Backlog: Release 2 . . . . .                     | 70 |
| 5.2  | Use case "Manage clients" description . . . . .  | 76 |

|     |   |    |
|-----|---|----|
| 5.3 | Use case "Manage complains" description . . . . .       | 77 |
| 5.4 | Use case "Manage reports" description . . . . .         | 78 |
| 5.5 | Use case "Manage events" description . . . . .          | 79 |
| 6.1 | Backlog: Release 3 . . . . .                            | 88 |
| 6.2 | Use case "Manage reviews" description . . . . .         | 94 |
| 6.3 | Use case "Manage ratings" description . . . . .         | 95 |
| 6.4 | Use case "Statistics" description . . . . .             | 96 |
| 6.5 | Use case "Manage cart" description . . . . .            | 96 |
| 6.6 | Use case "Manage orders" description . . . . .          | 97 |
| 6.7 | Use case "Manage personal orders" description . . . . . | 98 |

## List of Abbreviations

**3D** Three Dimensional

**AI** Artificial Intelligence

**API** Application Program Interface

**CLI** Command Line Interpreter

**IDE** Integrated Development Environment

**MVC** Model View Controller

**MVP** Model View Presenter

**MVVM** Model View View Model

**ORM** Object Relational Mapper

**OS** Operating System

**PHP** Hypertext Preprocessor

**PWA** Progressive web app

**REST** Software Development Kit

**RTL** Register Transfer Level

**SDK** Server side rendering

**SQL** Standardized Query Language

**SSR** Server side rendering

**UCD** Use Case Diagram

**UI** User Interface

**UML** Unified Modeling Language

**U.S** User Story

**UX** User Experience

**VSC** Visual Studio Code

# General introduction

Bakeries are a popular type of food-service establishment, and they allow you to express your culinary creativity while also serving a unique market, the problem is you can spend all day and night in the kitchen creating the next best cake, but if no one knows about it, all that hard work will go in vain.

Evidently bakeries are facing difficulties attracting customers and handling their orders and products, in fact even clients find it difficult to find a pastry to their taste without having to go through the process of searching and asking around for opinions.

In order to succeed in the bakery business, some marketing and advertising can also make a big difference.

Depending on all of that, we are providing as a solution an application named "**Cake it up !**" that fills in the needs of both parts the business owner and the client. It provides an easy way for the owner to promote their business and easier way for the client to find a place that matches him.

This report contains six chapters;

The first chapter is a quick presentation of the company "**Anypli**" where the internship took place, this chapter will also contain the preliminary Study of the project, in fact we will be analyzing the existing and explaining our solution.

The second chapter focuses on the requirements of the project and the planning of this work, where we will present the breakdown of this project needed to highlight the different modules to be developed in time.

The third chapter will provide the means used to make this work possible and give a detailed description of the project's architecture.

In the fourth, fifth and sixth chapter, we will present the three releases of this work. This step is based on the AGILE SCRUM methodology for the purpose of good project management.

We conclude with a general conclusion that presents an analysis of the work done within

---

**“Anypli”** company. This section highlights not only a summary of the work done throughout the period of the end-of-course project internship but also proposals from the various perspectives.

# **Chapter 1**

## **Project scope**

### **Introduction**

This chapter will be dedicated to the presentation, in the first place, of the environment in which our project was realized while taking into account the efficiency and the quality of the work.

Secondly, we will present the project itself, analyzing the existing and proposing a solution that will allow us to provide elements of response to the shortcomings identified.

finally, we will give a brief and clear idea about what methodology was used to make this project.

### **1.1 Host organization Presentation**

**“Anypli”** is a web and mobile development agency, founded in 2011 and located at Monastir, it has over 50 employees. it is in possession of several teams of enthusiasts each mastering their area of expertise as well as related fields. **“Anypli”** follows a methodology based on listening and consulting in order to offer quality achievements perfectly suited to the objectives of its customers and the resources they want to devote to it.



Figure 1.1: Logo Anypli

**“Anypli”** offers multiple services such as :

- Mobile applications: IOS, Android
- Web sites
- Design
- UI and UX

## 1.2 Project Presentation

### 1.2.1 Analysis of the existing

We are in an era where technology leads everything, there is an application for most of the people’s needs, however most of these softwares lack some details that are not yet fulfilled.

This section will start by listing some of the applications or ways people use to get information about pastries, then criticize every one of them to show what is missing.

#### Social media

The mostly common way is using the social media such as **“FACEBOOK”** and **“INSTAGRAM”** where you can find a verity of choices and number of opinions that can guide the user to select the right place. [1]

We can also use it to figure out the cost of the meal or get direction as shown in Figure 1.2.

The social media also give you the ability to communicate with the manager of the place in private to either pass an order or to ask for more details regarding a specific product.

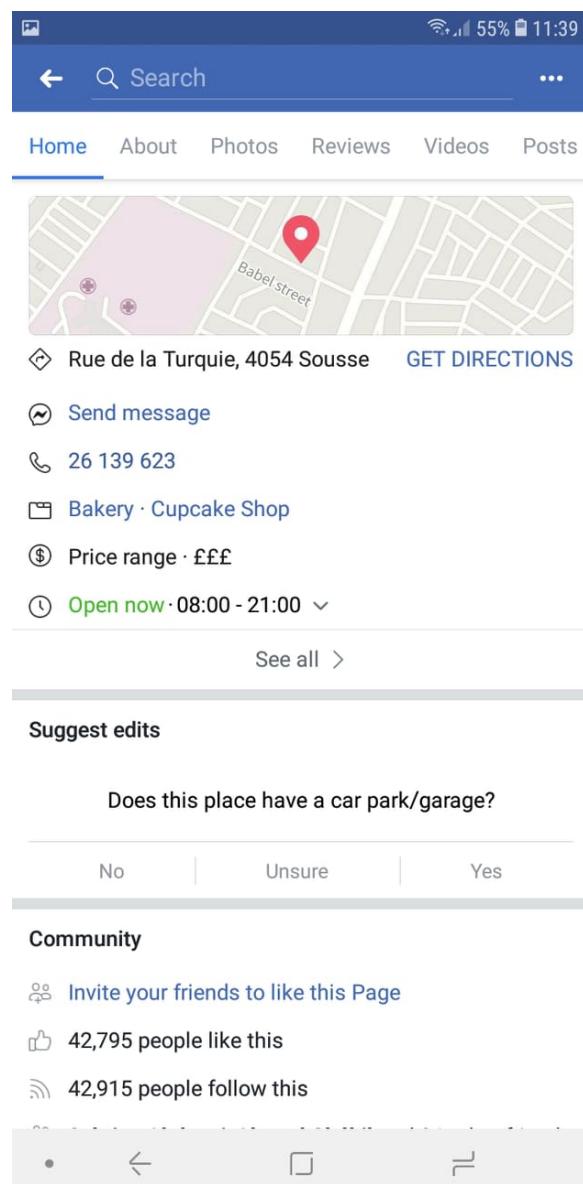


Figure 1.2: Application: FACEBOOK<sup>1</sup>

---

<sup>1</sup>Source: <https://www.facebook.com/>

## Google maps

“Google maps” has been popular for quite a while now, it can be used to figure out the nearby pastries, reviewing shop and get directions as showing in Figure 1.3.

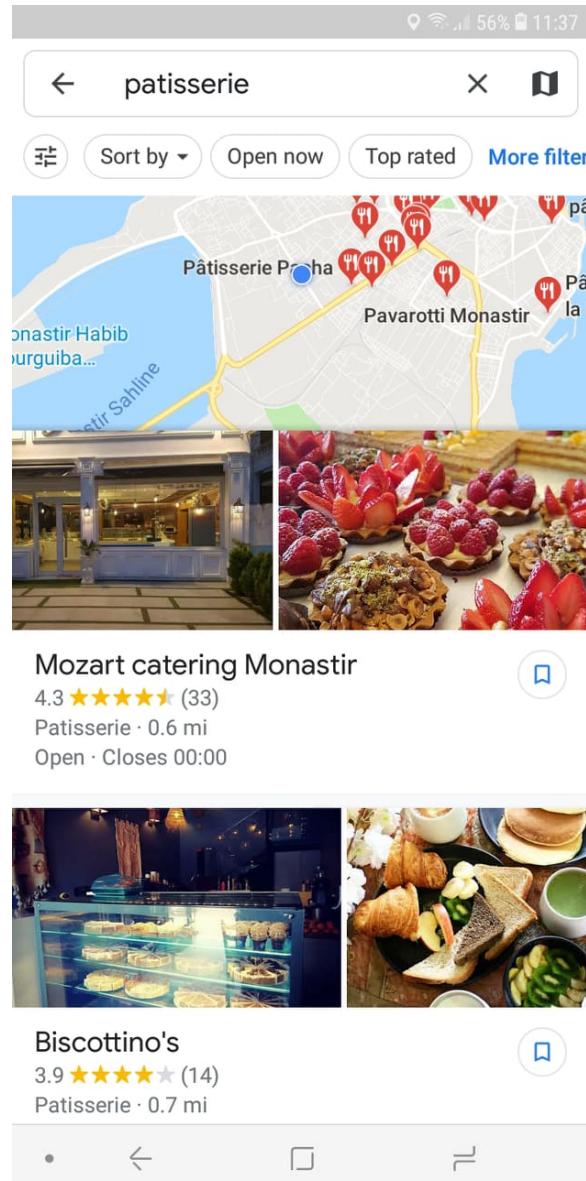


Figure 1.3: Google maps<sup>2</sup>

<sup>2</sup>Source: <https://www.google.com/maps>

## Bakery days

“Bakery days” gives you the ability to design your cake as presented in Figure 1.4:

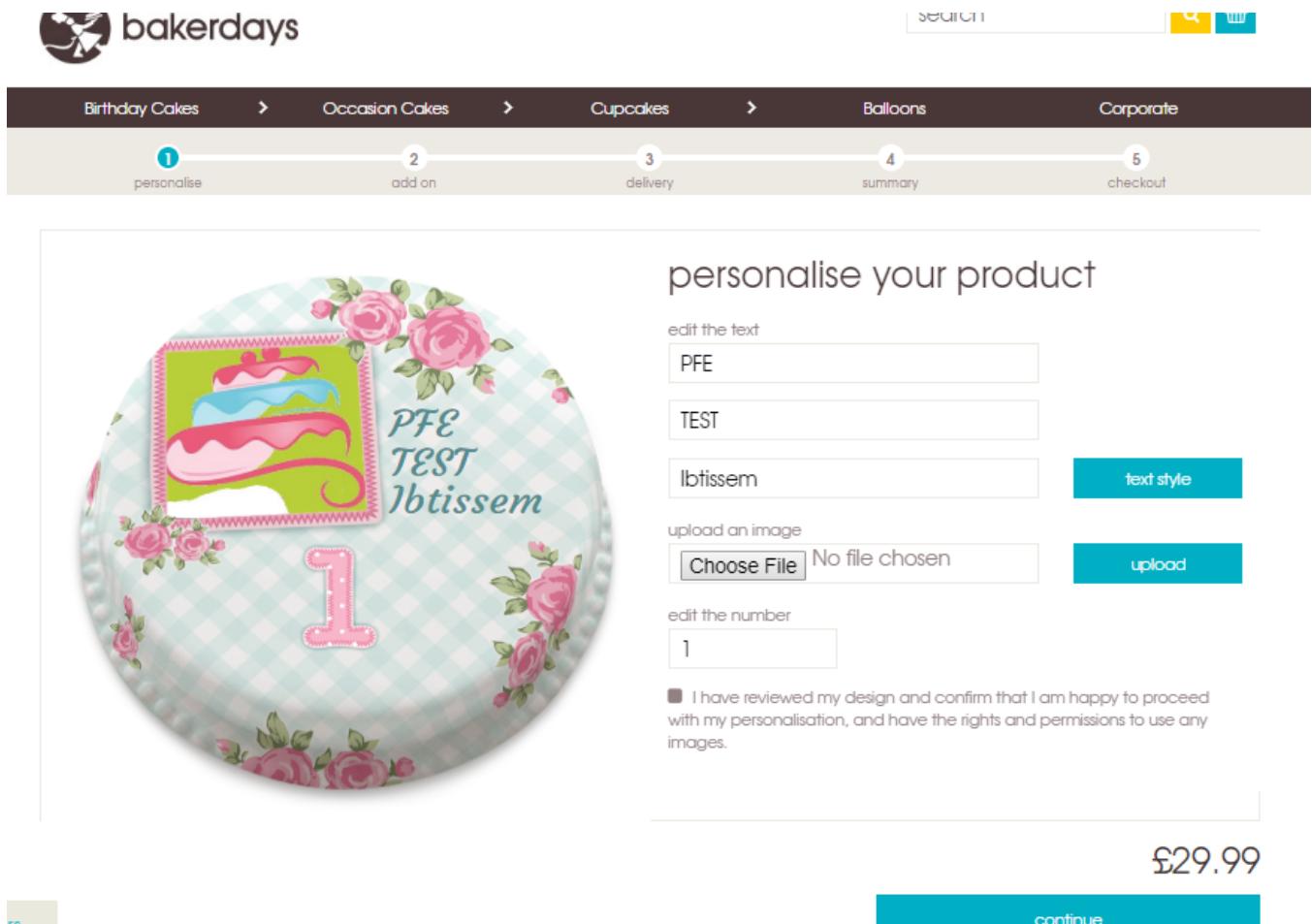


Figure 1.4: Web site: Bakery days<sup>3</sup>

## YELP

“Yelp” has tremendous power in the pastries industry, and having a strong backing of positive Yelp reviews is like having a flock of golden geese reviews from Yelp can do wonders for any business.

Advantages of “Yelp”:

- The application provide two parts, the business owner’s section presented in Figure 1.5 and the visitor in Figure 1.6.
- The user can add his own business by adding as many details as possible.

<sup>3</sup>Source: <https://www.bakerdays.com/>

- He can respond to the reviews.
- Manage his shop.
- A visitor can rate and review a shop.

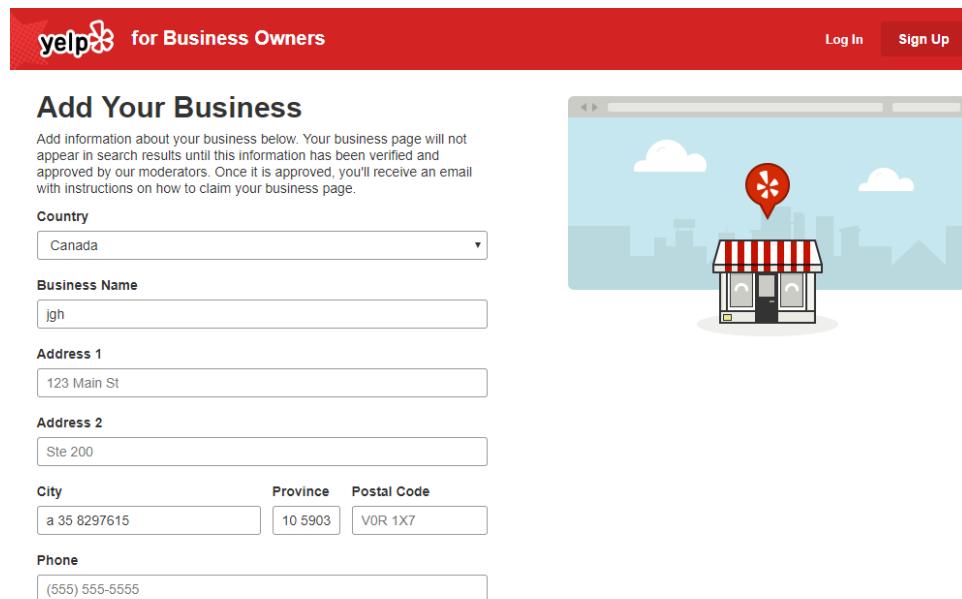


Figure 1.5: Web application: Yelp (business owner)<sup>4</sup>

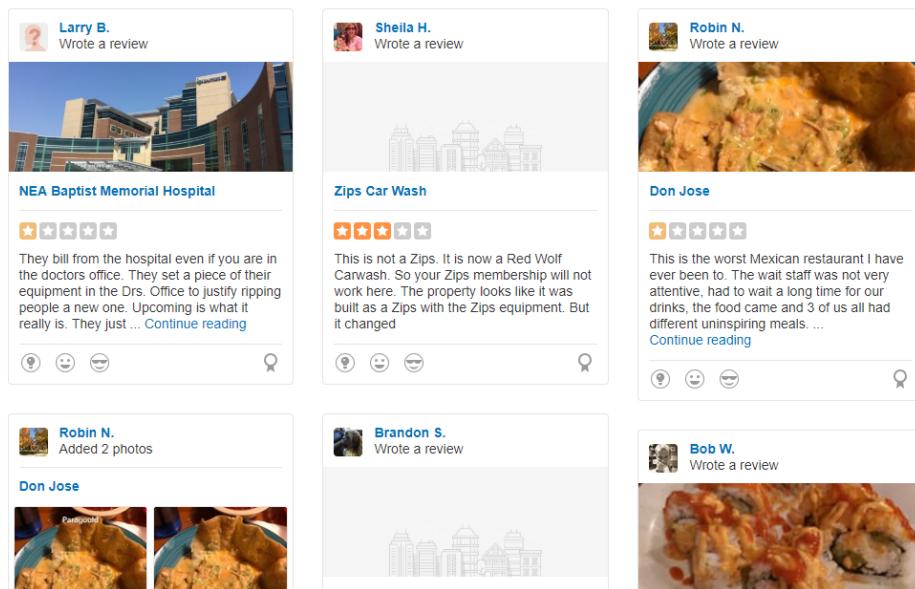


Figure 1.6: Web application: Yelp (user)<sup>5</sup>

<sup>4</sup>Source: <https://biz.yelp.ca>

<sup>5</sup>Source: <https://www.yelp.com>

### 1.2.2 Criticism of the existing

As shown in Table 1.1, there are number of disadvantages that can not be ignored.

Table 1.1: List of disadvantages of the existing.

| Existing     | Disadvantages   |
|--------------|---|
| Social media | Time consuming.<br>Risk of being wrongly informed about the place.<br>No information about the prices.<br>No security in passing orders.  |
| Google maps  | Places are added by anyone not the owner.<br>The suggestion are only based on the location of the user.<br>No on-line orders.<br>Not enough information about the products.         |
| Bakery days  | The personal design includes only the image on the cake or the writing.<br>The site is for one specific bakery.<br>Only available as a web site.<br>Covers only the United Kingdom. |
| YELP         | The site is designed only for reviews.<br>Can not pass an order.<br>Too many types of business.<br>Not available worldwide.   |

### 1.2.3 Solution

This section will give a small presentation of the solution regarding previous criticism followed by the requirements for this solution. As a solution for the previous issues, I provided a web/mobile application named “**Cake it up !**”.

In one hand, the mobile part will be dedicated to the clients everywhere. It provides an easier way to view all the pastries nearby or far away and get access to all their product’s information (price, description, ...). In addition to that he can pass orders from his phone. He also has the capability to rate and give his own opinion about the bakeries or even block/report a shop. Finally, he can create his own cake design in 3D (forms, layers, colors, perfumes, ...).

On the other hand, the web application is accessible for bakeries owners, so they can contribute by adding their place to the application with detailed description of their products (prices, ingredients, ...). The owner can also manage the orders of his clients or report any suspicious ones. The application gives also a daily statistic of the shop and its performance.



Figure 1.7: Cake It Up logo<sup>6</sup>

## 1.3 Used methodology

For this project, we used the SCRUM design methodology which is an agile project management framework used primarily for software development projects to provide a new software capability every 2 to 4 weeks , that allows us to conduct our project well while organizing our work plan. [2]

The choice of the Agile method was based on five main factors that are taken into consideration to validate the applicability of this method:

- Quick need for product availability

- Unpredictability of customer needs
- Need for frequent changes
- Customer visibility needed on the progress of developments
- Presence of the user providing immediate feedback.

For the realization of the project, the Scrum methodology used is an agile methodology dedicated to project management. The specificity of the Scrum method is its flexibility, as well as the self-organization of the development team. We chose this method because it applies well to the project because of its exploratory nature.

Instead of performing a complete and complete project planning from the start, the work is organized in increments (called sprints) planned as the work progresses. This makes it possible to concentrate efforts on the rapid production of testable and functional elements, while keeping a certain flexibility with respect to the hazards that may arise.



Figure 1.8: Scrum processes<sup>7</sup>

Sprints can last from a few hours to a month (with a two-week preference). Each sprint has a goal and is associated with a list of features to achieve. Also each sprint begins with an estimate followed by operational planning, ends with a demonstration of what has been completed and helps to increase the business value of the product as shown in Figure 1.8. Before starting a new sprint, the team performs a retrospective: it analyzes what happened during this sprint, and improve for the next.

<sup>7</sup>Source: <http://www.althea-groupe.com>

## Conclusion

In this chapter, we have briefly presented the “**Anypli**” company and elucidated the field of study of our work. Then, an existing study was developed to describe the problematic in order to identify the deficiencies to be corrected. The next chapter will focus on the requirements.

# **Chapter 2**

## **Requirements**

### **Introduction**

In the previous chapter, we chose to adopt the Scrum methodology for designing our future system. In fact, Scrum is organized into three phases, the first of which is the planning and architecture phase (also known as sprint 0 in some books). This phase is the most important in the Scrum development cycle since it directly influences the success of sprints. and in particular the first.

The work done in this period of time leads to build a good vision of the product, identify the roles of users and clear the main features to produce the initial backlog and a first sprint planning. This phase will be the subject of this chapter where we start by capturing the different needs, identify the roles of the users and prepare our release plan.

### **2.1 Requirements**

#### **2.1.1 Actors**

“**Cake it up !**” includes two main actors and one secondary actor. They are divided into two categories web and mobile as shown in Table 2.1.

Table 2.1: List of actors.

|               | Actor            | Role  |
|---------------|------------------|---|
| <b>Web</b>    | Pastry manager   | He is the owner of the pastry, he is responsible for the managing of his own bakery.          |
|               | Administrator    | This is a secondary actor, he has access to all the data of both web and mobile applications. |
| <b>Mobile</b> | Customer/Visitor | He has only access to the mobile application.   |

## 2.1.2 Functional requirements

The Functional requirements are any requirement which specifies what the system should do. The application should be able to response to all the user's needs:

- For the web application:
  - The administrator also known as the web master should be able to:
    - \* Connect into his private space of the application,
    - \* Manage reports,
    - \* View statistics
    - \* Activate/deactivate pastries,
    - \* Block clients,
    - \* Manage account,
    - \* and also Receive notifications.
  - The second actor which is the Bakery manager or the pastry owner need to have access to the flowing functions:
    - \* Create an account,
    - \* Connect,
    - \* Manage his shop,
    - \* Manage complains received from the clients,
    - \* Manage orders,
    - \* View statistics,

- \* Manage his own products,
  - \* Manage personal events,
  - \* Receive notifications.
- As for the mobile application we can find two types of users:
- The visitor who also have multiple activities that has to be fulfilled, such as:
    - \* The ability to create an account,
    - \* To connect,
    - \* Manage cart,
    - \* Consult pastries and products.
  - Once the visitor connect his role changes to a customer, which will give him more advantages:
    - \* he can manage his account,
    - \* Manage orders,
    - \* Create complains,
    - \* Manage cart,
    - \* Add reviews,
    - \* Consult pastries and products,
    - \* Create a personal order,
    - \* Report a Bakery,
    - \* Access to history of orders,
    - \* Receive notifications,
    - \* and rate a pastry shop

### 2.1.3 Non-Functional requirements

Non-Functional requirements are any requirement which specifies how the system performs a certain function. They generally specify the system's quality attributes or characteristics.

- Security: every account is secured by a encrypted password

- Performance: quick response
- Reliability: fault tolerance
- Maintainability: easy to fix and evolve
- Availability: the application can be used anytime and anywhere

## 2.2 Project management with SCRUM

### 2.2.1 SCRUM

To manage the Scrum projects, team members use several techniques. One of these techniques, which is the most answered, is to create cards and paste them on a wall or a table visible to all members of the team. Another technique is to use an Excel file containing all the necessary information for sprints, user stories, estimates, and so on. This file must be shared in read and write (so that all members of the team can change it at any time).

### 2.2.2 SCRUM roles

Scrum defines three roles, which are:

- The Product Owner: The person with the product vision, usually an expert in the field.
- The Scrum Master (product manager): it is the person who must ensure the smooth running of the different sprints of the release, and who must imperatively master Scrum.
- Development Team: The team members who execute the work.

## 2.3 Product Backlog

The backlog of the product is the most important artifact of Scrum, it is the set of functional or technical characteristics that constitute the desired product. These items can have a technical nature or can be user-centric e.g. in the form of user stories. The owner of the Scrum Product Backlog is the Scrum Product Owner. The Scrum Master, the Scrum Team and other Stakeholders contribute it to have a broad and complete To-Do list.

Table 2.2 summarizes the product backlog of our application.

Table 2.2: Product Backlog.

|          | <b>Theme</b>     | <b>ID</b> | <b>User story</b>  | <b>Importance</b> | <b>Risk</b> |
|----------|------------------|-----------|--|-------------------|-------------|
| <b>1</b> | Authentification | 1.1       | As an unauthorized User i want to create a new account.                    | High              | Low         |
|          |                  | 1.2       | As an unauthorized User i want to login.                                   | High              | Low         |
|          |                  | 1.3       | As an authorized User i want to logout.                                    | High              | Low         |
|          |                  | 1.4       | As an unauthorized User i want to reset my password.                       | High              | High        |
| <b>2</b> | Manage account   | 2.1       | As an authorized user i want to edit my own account.                       | Medium            | High        |
| <b>3</b> | Manage pastries  | 3.1       | As a pastry owner i want to edit my own pastry.                            | High              | High        |
|          |                  | 3.2       | As an Administrator i want to view the list of pastries.                   | High              | Medium      |
|          |                  | 3.3       | As an Administrator i want to block a pastry.                              | Low               | Low         |
|          |                  | 3.4       | As an Administrator i want activate or dis-activate a pastry.              | Low               | Low         |
|          |                  | 3.5       | As an client i want to view and search the list of activated pastries.     | High              | Low         |
|          |                  | 3.6       | As a pastry owner i want to dis-activate my own pastry if it is activated. | High              | Low         |

|   | Theme            | ID  | User story  | Importance | Risk   |
|---|------------------|-----|---|------------|--------|
| 4 | Manage products  | 4.1 | As a pastry owner i want to add a product.              | High       | High   |
|   |                  | 4.2 | As a pastry owner i want to edit a product.             | High       | High   |
|   |                  | 4.3 | As a pastry owner i want to view list of products.      | High       | Low    |
|   |                  | 4.4 | As a pastry owner i want to archive a product.          | Low        | Low    |
| 5 | Manage clients   | 5.1 | As an administrator i want to view list of all clients. | High       | Low    |
|   |                  | 5.2 | As an administrator i want to block a client.           | Low        | Low    |
|   |                  | 5.3 | As a pastry owner i want to view list of my clients.    | High       | Low    |
| 6 | Manage complains | 6.1 | As a client i want to add a complain.                   | Medium     | Low    |
|   |                  | 6.2 | As a client i want to view my list of complains.        | Medium     | Low    |
|   |                  | 6.2 | As a pastry owner i want to view the list of complains. | Medium     | Low    |
|   |                  | 6.3 | As a pastry owner i want to reply to a complain..       | Medium     | Medium |
| 7 | Manage Reports   | 7.1 | As a client i want to report a pastry.                  | Low        | Low    |
|   |                  | 7.2 | As a pastry owner i want to report a client.            | Low        | Low    |
|   |                  | 7.3 | As an administrator i want to view list of reports.     | Low        | Low    |

|           | Theme                  | ID   | User story   | Importance | Risk   |
|-----------|------------------------|------|--|------------|--------|
| <b>8</b>  | Manage events          | 8.1  | As a pastry owner i want to add a personal event.                | Low        | Medium |
|           |                        | 8.2  | As a pastry owner i want to view my calender.                    | Low        | Low    |
| <b>9</b>  | Manage reviews         | 9.1  | As a client i want to add a review.                              | Low        | Medium |
|           |                        | 9.2  | As a pastry owner i want to view my pastry's reviews.            | Low        | Low    |
| <b>10</b> | Manage ratings         | 10.1 | As a client i want to rate a pastry.                             | Low        | Medium |
|           |                        | 10.2 | As a pastry owner i want to view my pastry's rating.             | Low        | Low    |
| <b>11</b> | Statistics             | 11.1 | As an administrator or a pastry owner i want to view statistics. | Medium     | Low    |
| <b>12</b> | Manage cart            | 12.1 | As a client i want to edit my cart.                              | High       | High   |
|           |                        | 12.2 | As a client i want to view my cart.                              | High       | Low    |
| <b>13</b> | Manage orders          | 13.1 | As a client i want to pass an order.                             | High       | High   |
|           |                        | 13.2 | As a client i want to view my own orders.                        | High       | Low    |
|           |                        | 13.3 | As a pastry owner i want to view orders.                         | High       | Low    |
|           |                        | 13.4 | As a pastry owner i want to change the state of an order.        | High       | Medium |
| <b>14</b> | Manage personal orders | 14.1 | As a client i want to add a personal order.                      | High       | High   |
|           |                        | 14.2 | As a client i want to view my list of personal orders.           | High       | Low    |

## 2.4 General use case diagram

The purpose of a UCD<sup>1</sup> in UML<sup>2</sup> is to demonstrate the different ways that a user might interact with a system.

A use case helps represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

### 2.4.1 Use case: web

This section will give a detailed description of the general and detailed use cases that exists in the web application.

---

<sup>1</sup>A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system

<sup>2</sup>The OMG's Unified Modeling Language™ (UML®) helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of these requirements

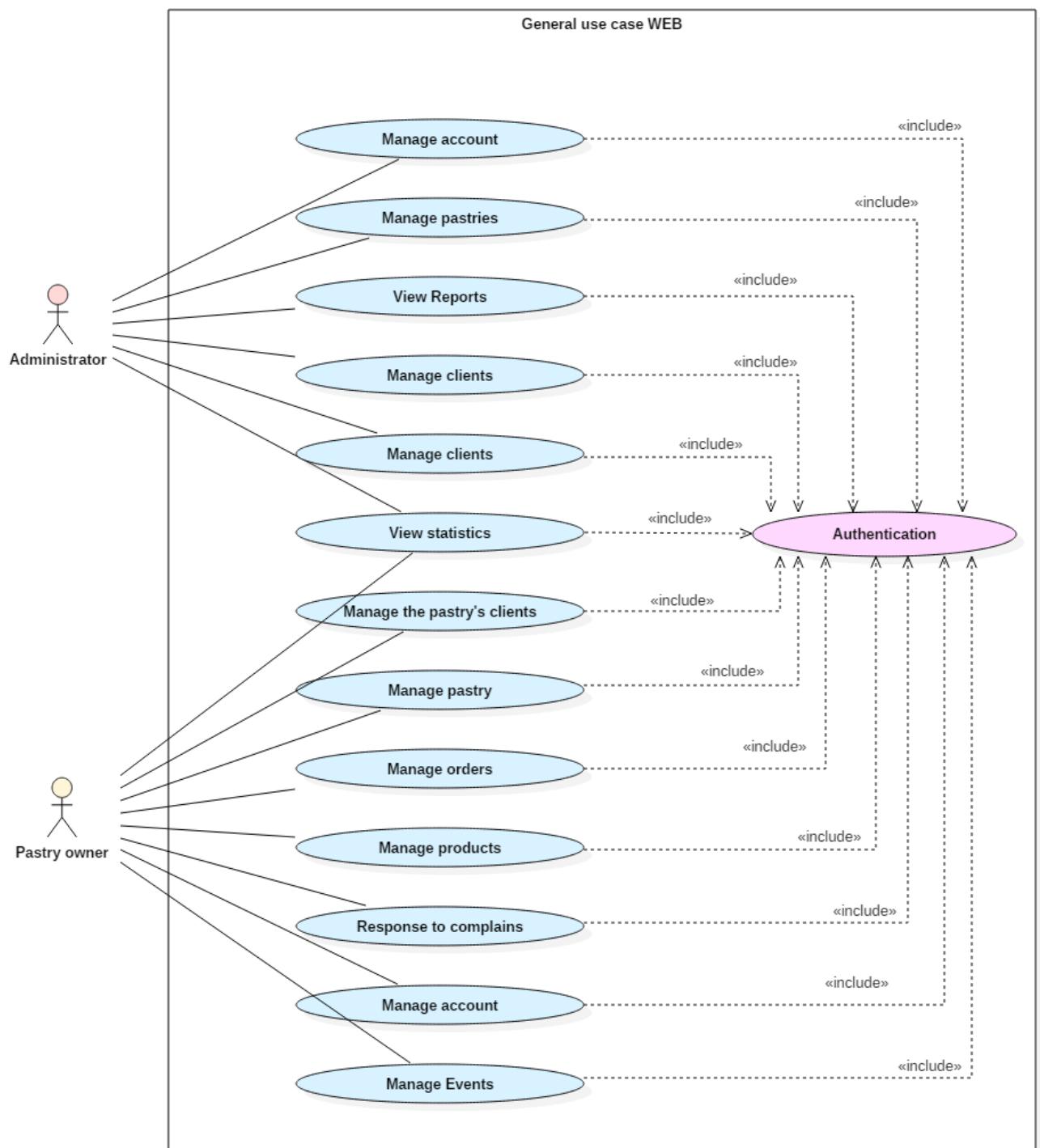


Figure 2.1: General use case: Web

Table 2.3: General use case web explanation

|                      | <b>Use case</b>        | <b>Explanation</b>  |
|----------------------|------------------------|---|
| <b>Administrator</b> | Edit account           | The administrator has access to his own private account where he can edit his personal information,   |
|                      | Manage pastries        | He can also view the list of all the pastries and manage them but with limits,  |
|                      | View Reports           | Viewing the reports is also an option, a report can be send from a client or a pastry,  |
|                      | Manage clients         | Managing the clients is one of the Administrator's tasks.   |
| <b>Patsy owner</b>   | Manage his own clients | The second user which is the pastry's owner have access to the list of his clients (the ones who made at least one order with them),                |
|                      | Manage pastry          | He can view the information of his pastry and edit or deactivate it,  |
|                      | Manage orders          | All the orders made by a client can be seen by the pastry's owner, and he has multiple actions he can use on an order(accept, refuse, cancel, ...), |
|                      | Manage products        | Every pastry has a list of products that are created and managed by the pastry's owner,   |
|                      | Response to complains  | The complains made by the clients are only visible to this user, he has the ability to response to these complain by a message,                     |
|                      | Manage account         | The pastry's owner have access to his own information (email, password)   |
|                      | Manage events          | The pastry's owner have a secondary option which is handling his private calendar   |

## 2.4.2 Use case: Mobile

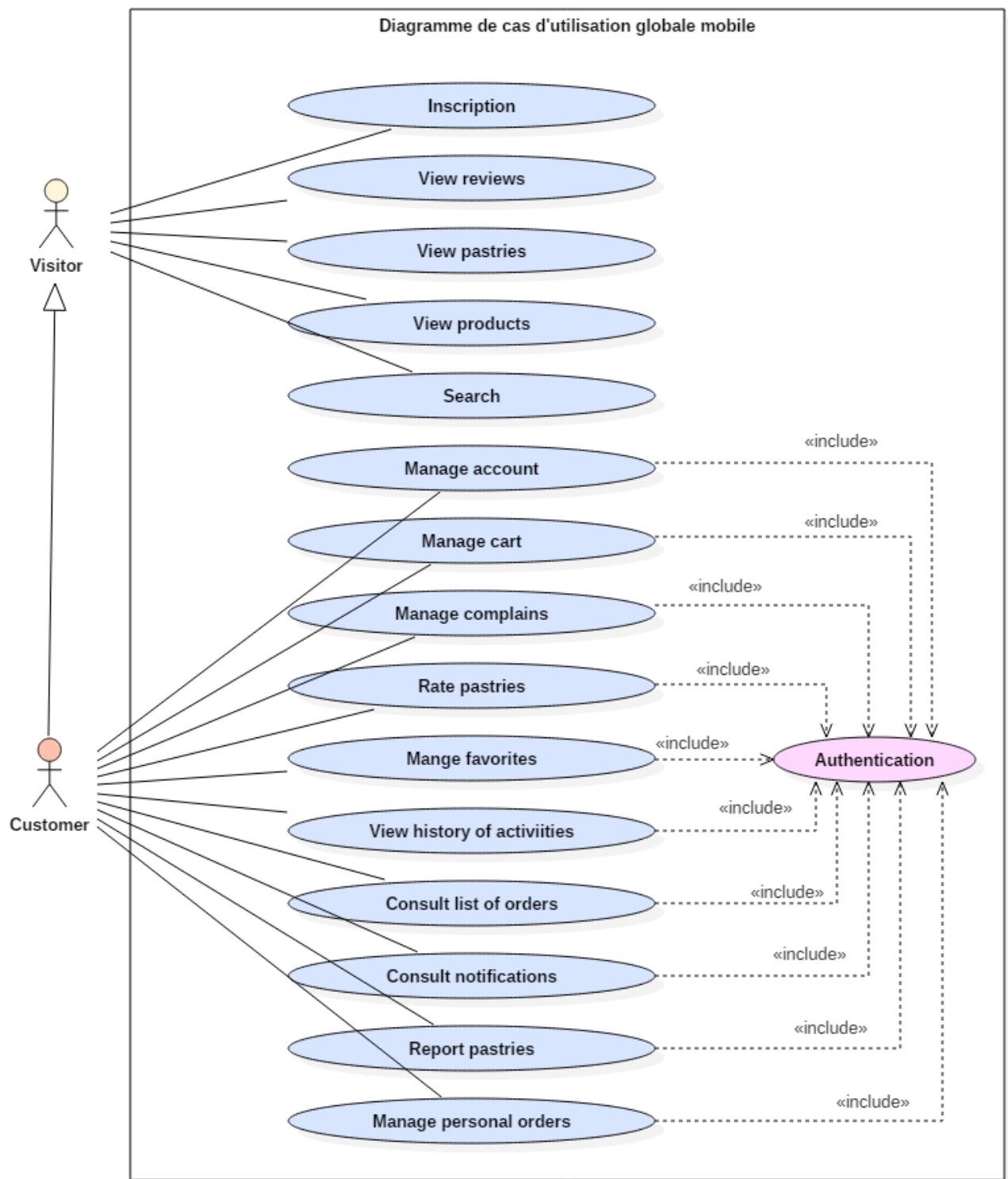


Figure 2.2: General use case: Mobile

Table 2.4: General use case mobile explanation

|                      | <b>Use case</b>        | <b>Explanation</b>  |
|----------------------|------------------------|---|
| <b>Administrator</b> | Edit account           | The administrator has access to his own private account where he can edit his personal information,   |
|                      | Manage pastries        | He can also view the list of all the pastries and manage them but with limits,  |
|                      | View Reports           | Viewing the reports is also an option, a report can be send from a client or a pastry,  |
|                      | Manage clients         | Managing the clients is one of the Administrator's tasks.   |
| <b>Patsy owner</b>   | Manage his own clients | The second user which is the pastry's owner have access to the list of his clients (the ones who made at least one order with them),                |
|                      | Manage pastry          | He can view the information of his pastry and edit or deactivate it,  |
|                      | Manage orders          | All the orders made by a client can be seen by the pastry's owner, and he has multiple actions he can use on an order(accept, refuse, cancel, ...), |
|                      | Manage products        | Every pastry has a list of products that are created and managed by the pastry's owner,   |
|                      | Response to complains  | The complains made by the clients are only visible to this user, he has the ability to response to these complain by a message,                     |
|                      | Manage account         | The pastry's owner have access to his own information (email, password)   |
|                      | Manage events          | The pastry's owner have a secondary option which is handling his private calendar   |

## 2.5 Conceptual Data Modeling

### 2.5.1 Data dictionary

Following the conceptual modeling of UML, all the data mentioned in the dictionary are organized according to classes.

Classes are the basic modules of object-oriented programming. It is a formal description of a set of objects with common semantics and features.

Table 2.5: Pastry class

| Pastry class |                                   |           |
|--------------|-----------------------------------|-----------|
| Code         | Meaning                           | Data type |
| name         | The name of the pastry            | String    |
| description  | the description of the pastry     | String    |
| mail         | the mail address of the pastry    | String    |
| web_site     | the web site of the pastry        | String    |
| date_open    | the opening date of the pastry    | DateTime  |
| logo         | the logo of the pastry            | String    |
| delivery     | The pastry does deliveries or not | Boolean   |
| isActive     | The pastry is activated or not    | Boolean   |
| isBlocked    | The pastry is blocked or not      | Boolean   |

Table 2.6: User class

| User class |                          |           |
|------------|--------------------------|-----------|
| Code       | Meaning                  | Data type |
| email      | The email of the user    | String    |
| password   | the password of the user | String    |

Table 2.7: Product class

| Product class |                                    |           |
|---------------|------------------------------------|-----------|
| Code          | Meaning                            | Data type |
| ref           | A unique reference for the product | String    |
| name          | The product's name                 | String    |
| description   | The description of the product     | String    |
| price         | The product's price                | Decimal   |
| stock         | Quantity of products in stock      | Integer   |
| image         | The image of the product           | String    |

Table 2.8: Order class

| Order class  |  |           |
|--------------|--|-----------|
| Code         | Meaning  | Data type |
| quantity     | the number of products in the order                                    | Integer   |
| description  | The description of the order if it exist                               | String    |
| status       | To describe in what state is the product (waiting, done, canceled ...) | Integer   |
| total_amount | The price of the whole order   | Decimal   |
| date_order   | The date to deliver the order  | DateTime  |

Table 2.9: Review class

| Review class  |                                      |                  |
|---------------|--------------------------------------|------------------|
| <b>Code</b>   | <b>Meaning</b>                       | <b>Data type</b> |
| content       | The comment added with the review    | String           |
| date_creation | The date when the review was created | DateTime         |

Table 2.10: Complain class

| Complain class |  |                  |
|----------------|--|------------------|
| <b>Code</b>    | <b>Meaning</b>                         | <b>Data type</b> |
| description    | Every complain has a description       | String           |
| date_creation  | The date when the complain was created | DateTime         |

Table 2.11: Schedule class

| Schedule class |                     |                  |
|----------------|---------------------|------------------|
| <b>Code</b>    | <b>Meaning</b>      | <b>Data type</b> |
| day            | The day of the week | String           |
| open_time      | Openning time       | Time             |
| close_time     | Closing time        | Time             |

Table 2.12: event class

| event class |                        |                  |
|-------------|------------------------|------------------|
| <b>Code</b> | <b>Meaning</b>         | <b>Data type</b> |
| title       | The title of the event | String           |
| date        | The date of the event  | Date             |

Table 2.13: Address class

| Address class |                              |                  |
|---------------|------------------------------|------------------|
| <b>Code</b>   | <b>Meaning</b>               | <b>Data type</b> |
| longitude     | The longitude of the address | Decimal          |
| latitude      | The latitude of the address  | Decimal          |

Table 2.14: Personal\_product class

| Personal_product class |   |                  |
|------------------------|---|------------------|
| <b>Code</b>            | <b>Meaning</b>                          | <b>Data type</b> |
| allergy                | List of allergies the client might have | String           |
| plus                   | A comment added by the client           | Decimal          |

## 2.5.2 Class diagram

Class diagrams are one of the most useful types of diagrams in UML [3] as they clearly map out the structure of a particular system by modeling its classes, attributes, operations, and

relationships between objects.

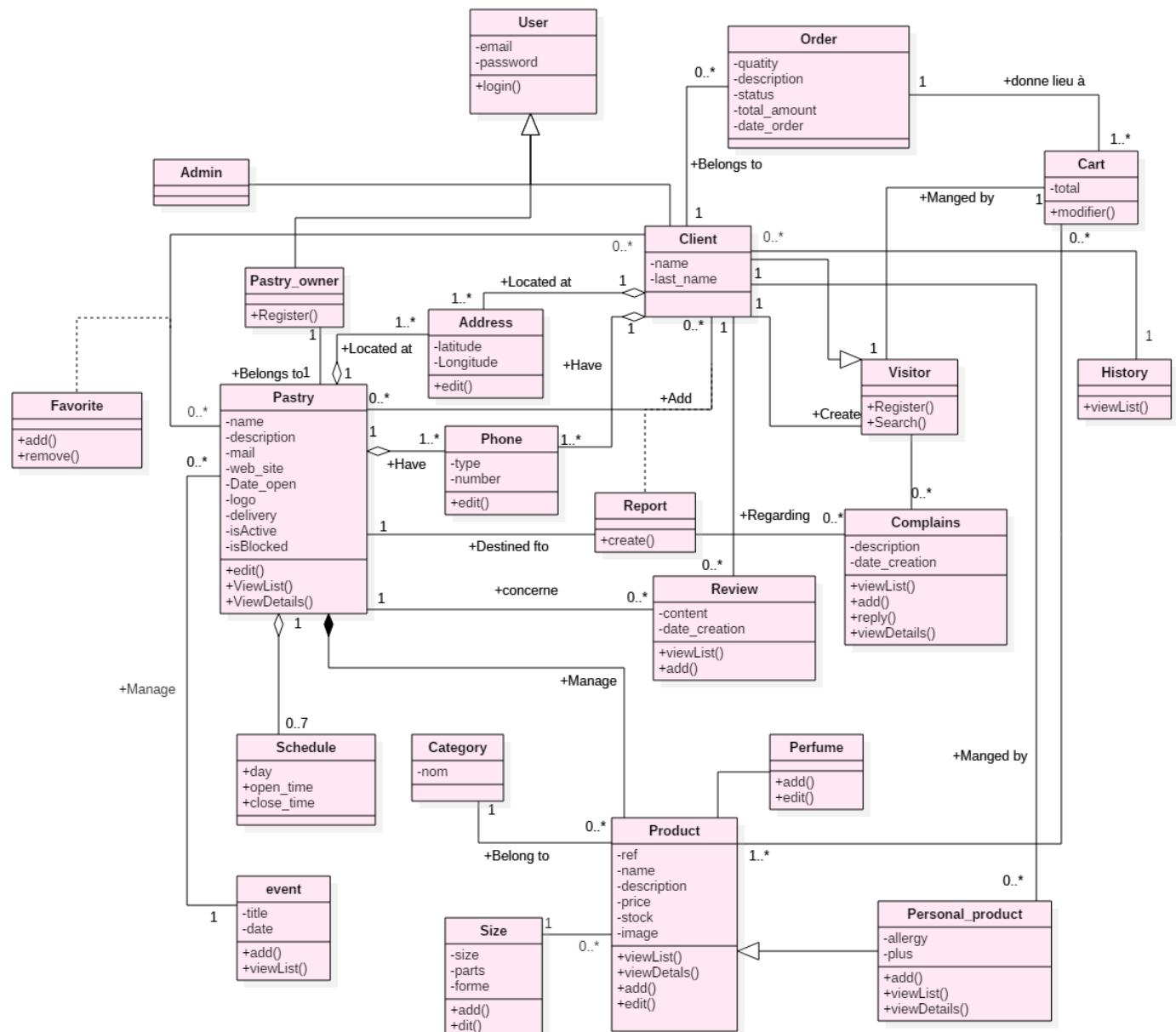


Figure 2.3: Class Diagram

Table 2.15: Class diagram explanation

| <b>Entities</b>   | <b>Definition</b>  |
|-------------------|--|
| User              | The user can be the administrator, the pastry owner, the visitor or the client |
| Pastry owner      | The person who manages the pastry.   |
| Administrator     | The person who manages the application.  |
| Client            | The person who will be passing orders.   |
| Visitor           | Is a client who does not have an account                                       |
| Pastry            | The pastry added by the owner  |
| History           | Any order or complain or a block done by a client is saved in this table.      |
| Review            | The client has an option to write a review about a pastry.                     |
| Product           | Every pastry has a number of products.   |
| Event             | An event can be created by a pastry owner.                                     |
| Favorites         | The client can add a pastry to his favorite list                               |
| Address           | Both the client and a pastry have an address.                                  |
| Personal_products | The client can create his own personal product.                                |
| Complains         | A complain can be sent from a client to a pastry owner.                        |

|          |   |
|----------|---|
| Cart     | Is where the orders are saved.                        |
| Category | Each product belongs to a specific category.          |
| Schedule | The pastry have an opening and closing time.          |
| Report   | Both pastry owners and clients can report each other. |
| Phone    | A phone number can belong to a pastry or a client.    |

## 2.6 Sprint Planning

The planning stage is very important for better organization and management of the project.

Table 2.16: Sprint planning

| Release 1              | Release 2           | Release 3              |
|------------------------|---------------------|------------------------|
| 18 February - 22 March | 23 March - 05 April | 06 April - 31 May      |
| Authentication         | Manage clients      | Manage reviews         |
| Manage account         | Manage Complains    | Manage rating          |
| Manage pastries        | Manage reports      | Statistics             |
| Manage products        | Manage events       | Manage cart            |
|                        |                     | Mange orders           |
|                        |                     | Manage Personal orders |

## Conclusion

This chapter allowed us to cover the different functional and non-functional needs of the actors of our application. We provided a representation of a use case diagram for actors reacting with the application and the sprint planning. The next chapter will deal with the realase 1.

# **Chapter 3**

## **Project Initialization**

### **Introduction**

We dedicate this chapter to initializing the project and setting up the development environment. The architecture as well as the software and technologies will be explained thereafter.

#### **3.1 Choice of the architecture**

The definition of the application architecture is an important step in the design process. It depends on a number of factors including performance requirements, prospects and scalability. Figure 3.1 illustrates a representation of the software architecture of the project.

The choice of the architecture of the application is considered as an important step in the process of making the application. Our application consists of three parts that are, the front-end part, the back-end part and the mobile app. The first part is done with the VueJS framework which consumes the REST API implied by the second part that is done with Laravel, Also the third part consumes the REST API implied by the second part also.

We chose to split the application architecture into two major parts, one layer as the back-office that contains Laravel app and VueJS app, and the other part as the front-office contains the Android mobile application which also uses the Laravel app as a back-end. That of Laravel communicates with the MYSQL database using the ELOQUENT ORM that is to say everything in the database has a representation as manipulable objects to simplify access and operations On the base. With the Eloquent principle, a table is represented by a PHP class that extends the class Model. And the controller contains the logic concerning the actions per-

formed by the user. It defines all the methods that represent the actions. In the Laravel part (Server side), we have implemented our API. The latter will specify all the logic that will be exploited by the VueJS part and the mobile Android app (Client side) using AXIOS in VueJS and Retrofit in Android. Quite simply, we tell them the method HTTP that we would like to use (eg GET / POST / etc.) and to which URL the request must be called.

Axios is a popular HTTP client based on promises that sports an API ‘a use. It allows you to make HTTP requests to retrieve or save data that is one of the tasks that a client-side JavaScript application will need to perform. As well as Retrofit for the Android Rest client.

In the VueJS part, we chose to use the VUEX State Management Library whose purpose to manage the state of the components of our application, and to facilitate their sharing. it separates the logic within the application to ensure better structuring and maintenance.

All these elements are grouped in the figure: 3.1 store of Vuex. The possibility of spreading the state in several modules allows to manage several different states in the single store of the application. For the android part, we have decided to go with MVP along with Retrofit.

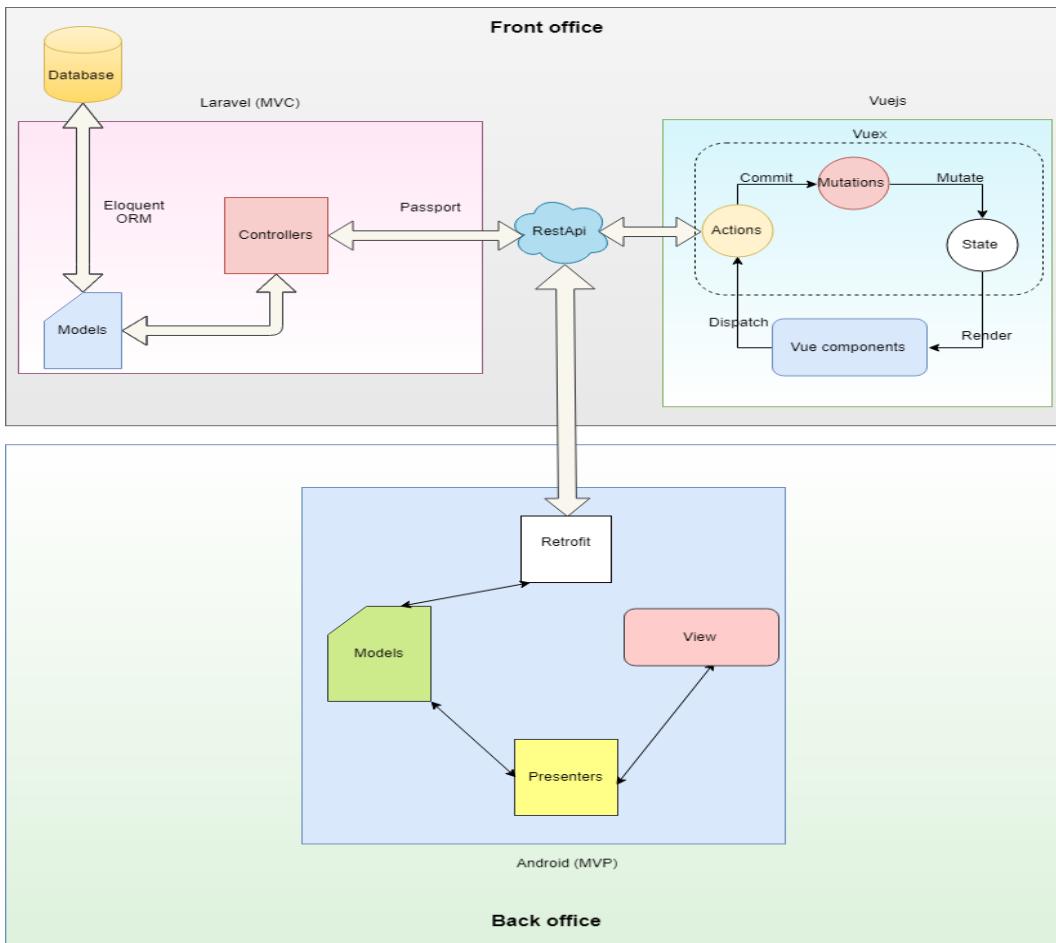


Figure 3.1: Project architecture

We will be explaining each and every design pattern used in this application:

**MVC:** This pattern was used for the back-end of the application, this pattern is actually used in Laravel which we will be using to create our APIs. MVC stands for Model-View-Controller as shown in Figure 3.2. [4]

- **Model:** is the object carrying the data.
- **View:** the data in the model will be visible to the user via this part.
- **Controller:** as for the controller is where we can find all the updates done on the model and sends them to the view.

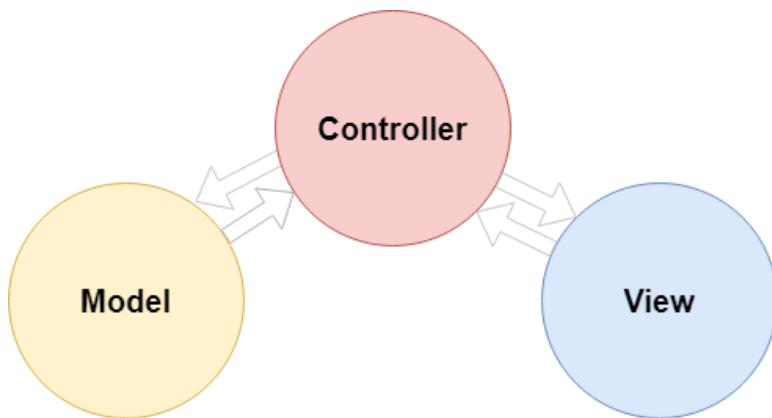


Figure 3.2: MVC pattern

**MVP:** For the android application we decided on using the MVP pattern to make our code cleaner and easier to understand.

- **Model:** Model is where the business logic and application data is stored.
- **View:** View is basically a passive interface cum UI which is responsible for routing user's action to the presenter. In Android, View can be your Activity, fragment or RecyclerView Adapters.
- **Presenter:** Is the middleman or mediator between View and Model which hold responsibilities of everything which has to deal with presentation logic in your application.

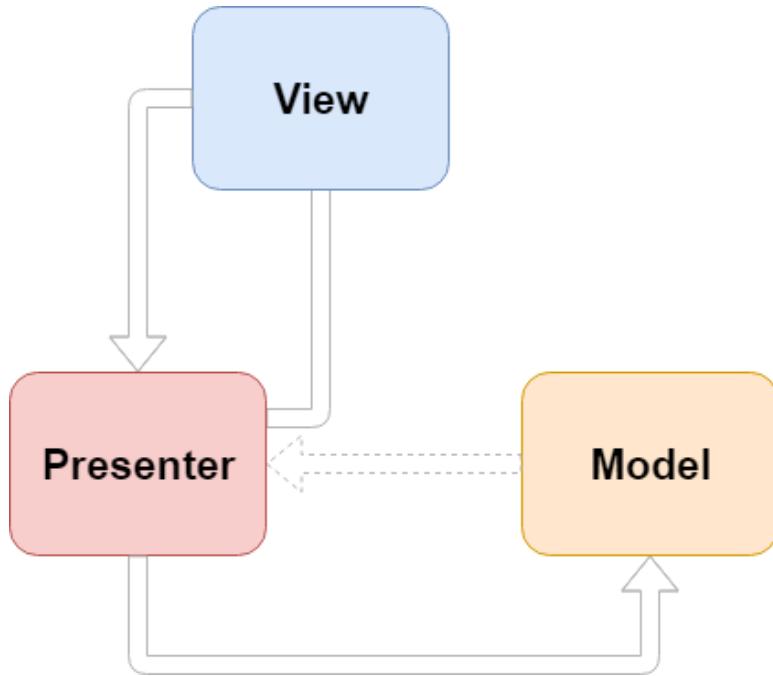


Figure 3.3: MVP pattern

**MVVM:** The MVVM design pattern (model, view, view-style) is a derivative of the MVC design model. The principle of this design pattern is to separate into three layers (model, view, view-style) where the view always receives the actions of the user and only interacts with the ViewModel, and the model communicates with the server and notifies the View-Model of the changes made.[5]

- **Model:** The model contains the data. Typically, this data comes from a database or an external service such as an API.
- **View (Vue):** The view corresponds to what is displayed (the web interface). The view contains the different graphic components (buttons, links, lists) as well as the text.
- **ViewModel (Model View):** This component is the link between the model and the view. It takes care of managing data links and eventual conversions. This is where the binding comes in.

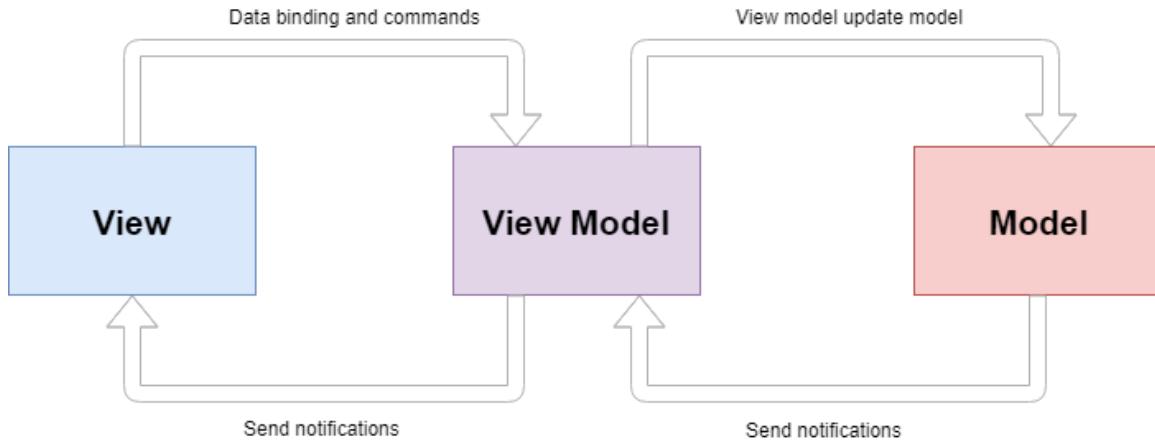


Figure 3.4: MVVM pattern

## 3.2 Programming languages

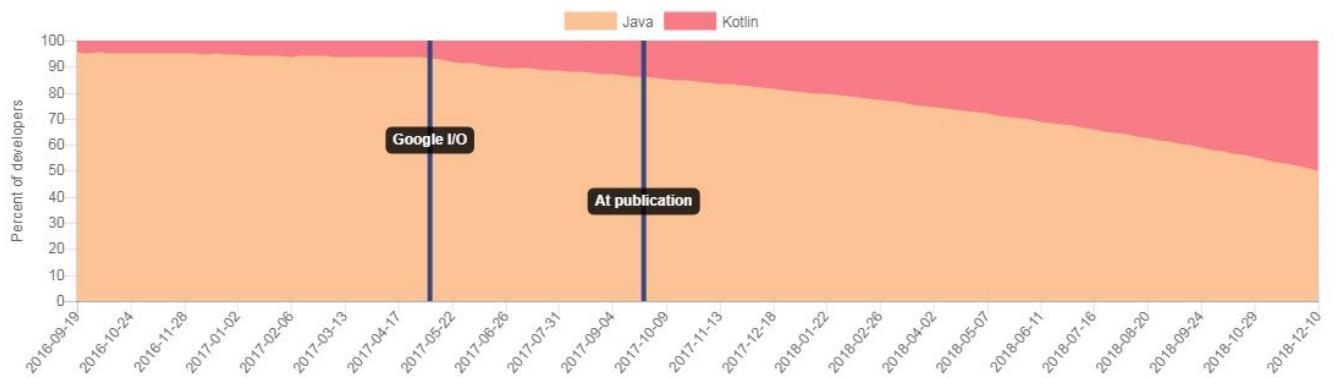
### 3.2.1 Kotlin



We decided on using “**Kotlin**” as an Android development language, seen it is the a perfect fit and considering the multiple advantages “**Kotlin**” has to offer.

“**Kotlin**” is the new open source programming language supported by Google. Its main goal is to improve the productivity of developers, while remaining compatible with the existing code. It is an object-oriented and functional programming language that can also be called a multi-paradigm language.[6]

As we can see in Figure 3.5, “**Kotlin**” has experienced strong growth in Android application development since May, after Google announced its official support in Android Studio. This is revealed by the Realm barometer, as part of its latest quarterly report.

Figure 3.5: Kotlin vs Java<sup>1</sup>

### 3.2.2 PHP



As for backend it is natural that we will be working with “**PHP**” since we chose Laravel as our backend framework. [7]

“**PHP**” is a server scripting language, specially designed for the development of web applications. It can be easily integrated with HTML.

It is common for this language to be associated with a database, such as MySQL. Executed on the server side there is no need for visitors to have particular software or plugins.

### 3.2.3 JavaScript



We gonna be using Vuejs ergo “**JavaScript**” will be one of the programming languages we are using.

Javascript is an object-oriented scripting language mainly used in HTML pages. Unlike server languages that run on the site, “**JavaScript**” is executed on the user’s computer by the browser itself. Thus, this language allows interaction with the user according to his actions.

## 3.3 Used technologies and frameworks

### 3.3.1 Android



“**Android**” is a mobile Operating OS developed by GOOGLE, Written primarily in Java and based on the Linux operating system [?].

Native Android apps are developed using the Java programming language, and can easily be ported to other mobile operating systems like Blackberry, Symbian and Ubuntu.

*“The Android platform enables device makers to compete and innovate. App developers can reach huge audiences and build strong businesses. Consumers now have unprecedented choice in devices, at ever-lower prices.”* -Hiroshi Lockheimer, SVP, Android, Chrome OS and Play.

---

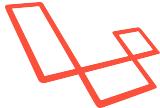
<sup>1</sup>Source: Rapport Realm

## Retrofit Android

“**Retrofit**” is type-safe REST client for Android and Java which aims to make it easier to consume Representational State Transfer (RESTful) web services.

It will save our development time, And also we can keep our code in developer friendly. Retrofit has given almost all the API’s to make server call and to receive response.

### 3.3.2 Laravel



“**Laravel**” is a framework that has existed since 2011 but has exploded recently and seems to attract a lot of developers, is a PHP framework that offers tools to build a web application. Laravel’s creator, Taylor Otwell, has simply grouped together the best libraries for every feature needed to create a website. [8]

### 3.3.3 VueJS



“**VueJS**” was our chose for the front end of the we application, since it is Flexible and easy to use. [9]

“**VueJS**” is a JavaScript UI library. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries.

When comparing it with its competitors, including Angular, React, Ember, Aurelia, etc., Vue boasts of beating some of them in certain aspects. These aspects include simple API, size, performance, learning curve, etc.

## Vuetify



We can find multiple frameworks that can be used with Vue.js but we decided on one which is “**Vuetify**”.

“**Vuetify**” is a progressive framework that attempts to push web development to the next level. It complies with the Material Design specification. This means the core features of both Vue and Material will be available by default and can be improved by both communities. In addition, Vuetify offers:

- Compatibility with Vue CLI-3 and RTL
- Templates for various frameworks (Cordova, webpack, etc.)
- Internationalization
- SSR and PWA

## Vuex



“**Vuex**” is our choice to use as a state management pattern and library for Vue.js. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion.

It helps us deal with shared state management with the cost of more concepts and boilerplate. It's a trade-off between short term and long term productivity.

As shown in Figure 3.6 it is a self-contained app with the following parts:

- The state, the source of truth that drives our app
- The view, a declarative mapping of the state
- The actions, the possible ways the state could change in reaction to user inputs from the view.

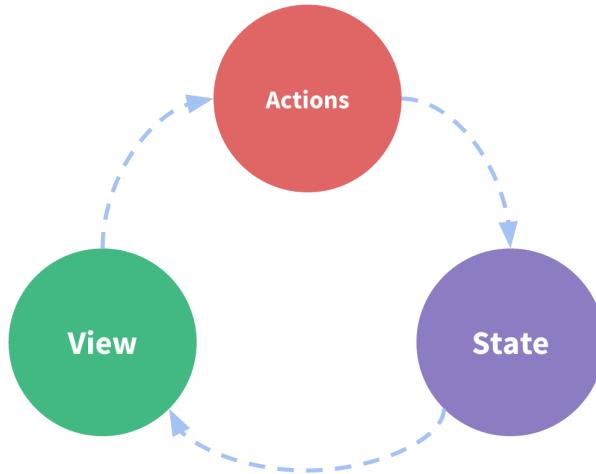


Figure 3.6: Vuex work flow

### 3.3.4 Swagger

 “**Swagger**” tools were used to document the APIs of our application, which made it easier to test our APIs before using them. “**Swagger**” is a specification for documenting REST API. It specifies the format (URL, method, and representation) to describe REST web services. The methods, parameters, and models description are tightly integrated into the server code, thereby maintaining the synchronization in APIs and its documentation.[10]

## 3.4 Used software

### 3.4.1 Android studio



We chose “**Android Studio**”; also developed by “**JETBRAINS**” for GOOGLE’s operating system, as an IDE to develop our android application. Android Studio is fully featured IDE designed specifically for Android development. Automatically, it integrates the SDK that enables developers to create applications for the Android platform<sup>2</sup>.

---

<sup>2</sup><https://developer.android.com/studio/>

### 3.4.2 Visual studio code



For the back-office code we decided on visual studio code as an IDE.

“**VSC**” is a code editor redefined and optimized for building and debugging modern web and cloud applications.

Visual Studio Code combines the simplicity of a code editor with what developers need for their core edit-build-debug cycle. It provides comprehensive code editing, navigation, and understanding support along with lightweight debugging, a rich extensibility model, and lightweight integration with existing tools.

### 3.4.3 Laragon



“**Laragon**” is a software that proposes to solve these problems by grouping all the tools you need for your development in a single installer.

The decision on using “**Laragon**” as a development server depend on various reasons, it is easy to use, it is a modern, maintained and rich-featured local development environment.

### 3.4.4 Star UML



We used “**Star UML**” to concept our different application’s diagrams. Star UML is developed by “**MKLAB**”. It is an open source project to develop fast, flexible, and extensible UML conceptions’ diagrams<sup>3</sup>.

---

<sup>3</sup>[staruml.io/](http://staruml.io/)

### 3.4.5 TexStudio

We used “**TexStudio**”<sup>4</sup> as an IDE for creating our “**Latex**” report. As we used “**JABREF**”<sup>5</sup> to include the bibliography in our document. **LATEX** is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation as it is the de facto standard for the communication and publication of scientific documents.[11]



### 3.4.6 SourceTree

A Git GUI that offers a visual representation of your repositories. “**Sourcetree**” is a free Git client.



It simplifies how we interact with the Git repositories which offers an easy and fast way for us to save our progress in the code.

## Conclusion

We have detailed the general architecture detailing the used technologies and framework to make this work possible. In the next chapter we will pass part 1 of the SCRUM methodology, the release 1 which will detail and present its user story.

---

<sup>4</sup><https://www.texstudio.org/>

<sup>5</sup><http://www.jabref.org/>

# Chapter 4

## Release 1

### Introduction

In the previous chapter, we discussed in detail the architecture of the “**CakeitUp**” application followed by all the used technologies to make this work.

This chapter is concerned with the presentation of the release 1, where we will detail the different stages of creation of this system.

### 4.1 Backlog

Table 4.1: Backlog: Release 1

| Sprint                         | ID<br>U.S | User story  | ID<br>task | Task   |
|--------------------------------|-----------|---|------------|--|
| Sprint 1<br><br>Authentication | 1.1       | As an unauthorized User i want to create a new account. | 1.1.A      | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|                                |           |   | 1.1.B      | Develop the inscription model.   |
|                                |           |   | 1.1.C      | Test the inscription model.  |

|   |     |  |       |  |
|---|-----|--|-------|--|
|   | 1.2 | As an unauthorized User i want to login.             | 1.2.A | Make the diagrams needed for this U.S.   |
|   |     |  | 1.2.B | Develop the login model.   |
|   |     |  | 1.2.C | Test the login model.  |
|   | 1.3 | As an authorized User i want to logout.              | 1.3.A | Make the diagrams needed for this U.S.   |
|   |     |  | 1.3.B | Develop the logout model.  |
|   |     |  | 1.3.C | Test the logout model.   |
|   | 1.4 | As an unauthorized User i want to reset my password. | 1.4.A | Make the diagrams needed for this U.S.   |
|   |     |  | 1.4.B | Develop the reset password model.  |
|   |     |  | 1.4.C | Test the reset password model.   |
| <b>Sprint 2</b><br><b>Manage account</b>  | 2.1 | As a authorizer user i want to edit my account.      | 2.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |     |  | 2.1.B | Develop the edit pastry model.   |
|   |     |  | 2.1.C | Test the edit pastry model.  |
| <b>Sprint 3</b><br><b>Manage pastries</b> | 3.1 | As a pastry owner i want to edit my own pastry.      | 3.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |     |  | 3.1.B | Develop the edit pastry model.   |
|   |     |  | 3.1.C | Test the edit pastry model.  |
|   | 3.2 | As an administrator i want to block pastries.        | 3.2.A | Make the diagrams needed for this U.S.   |
|   |     |  | 3.2.B | Develop the block pastry model.  |
|   |     |  | 3.2.C | Test the block pastry model.   |
|   | 3.3 | As an administrator i want to view list of pastries. | 3.3.A | Make the diagrams needed for this U.S.   |
|   |     |  | 3.3.B | Develop the list of pastries model.  |
|   |     |  | 3.3.C | Test the list of pastries model.   |

|   |     |   |                         |   |
|---|-----|---|-------------------------|---|
|   | 3.4 | As an administrator i want to activate or dis-activate a pastry.        | 3.4.A<br>3.4.B<br>3.4.C | Make the diagrams needed for this U.S.<br>Develop the activation/dis-activation pastry model.<br>Test the activation/dis-activation pastry model.           |
|   | 3.5 | As a client i want to view and search list of activated pastries.       | 3.5.A<br>3.5.B<br>3.5.C | Make the diagrams needed for this U.S.<br>Develop the list of pastries for the client.<br>Test the list of pastries.  |
|   | 3.6 | As a pastry owner i want dis-activate my own pastry if it is activated. | 3.6.A<br>3.6.B<br>3.6.C | Make the diagrams needed for this U.S.<br>Develop the pastry dis-activation model.<br>Test the pastry dis-activation model.                                 |
| <b>Sprint 4</b><br><b>Manage products</b> | 4.1 | As a pastry owner i want to add a product.                              | 4.1.A<br>4.1.B<br>4.1.C | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram).<br>Develop the add product model.<br>Test the add product model. |
|   | 4.2 | As a pastry owner i want to edit a product.                             | 4.2.A<br>4.2.B<br>4.2.C | Make the diagrams needed for this U.S.<br>Develop the edit product model.<br>Test the edit product model.   |
|   | 4.3 | As a pastry owner i want to view list of products.                      | 4.3.A<br>4.3.B<br>4.3.C | Make the diagrams needed for this U.S.<br>Develop the list of products model.<br>Test the list of products model.   |

|     |  |       |  |
|-----|--|-------|--|
| 4.4 | As a pastry owner i want to archive a product. | 4.4.A | Make the diagrams needed for this U.S. |
|     |  | 4.4.B | Develop the archive a product model.   |
|     |  | 4.4.C | Test the archive a product model.      |

## 4.2 Use case

### 4.2.1 Use case diagram

We will be deviding the use case of the release 1 depending on each sprint, starting by the first sprint “**Authentication**” presented in Figure 4.1.

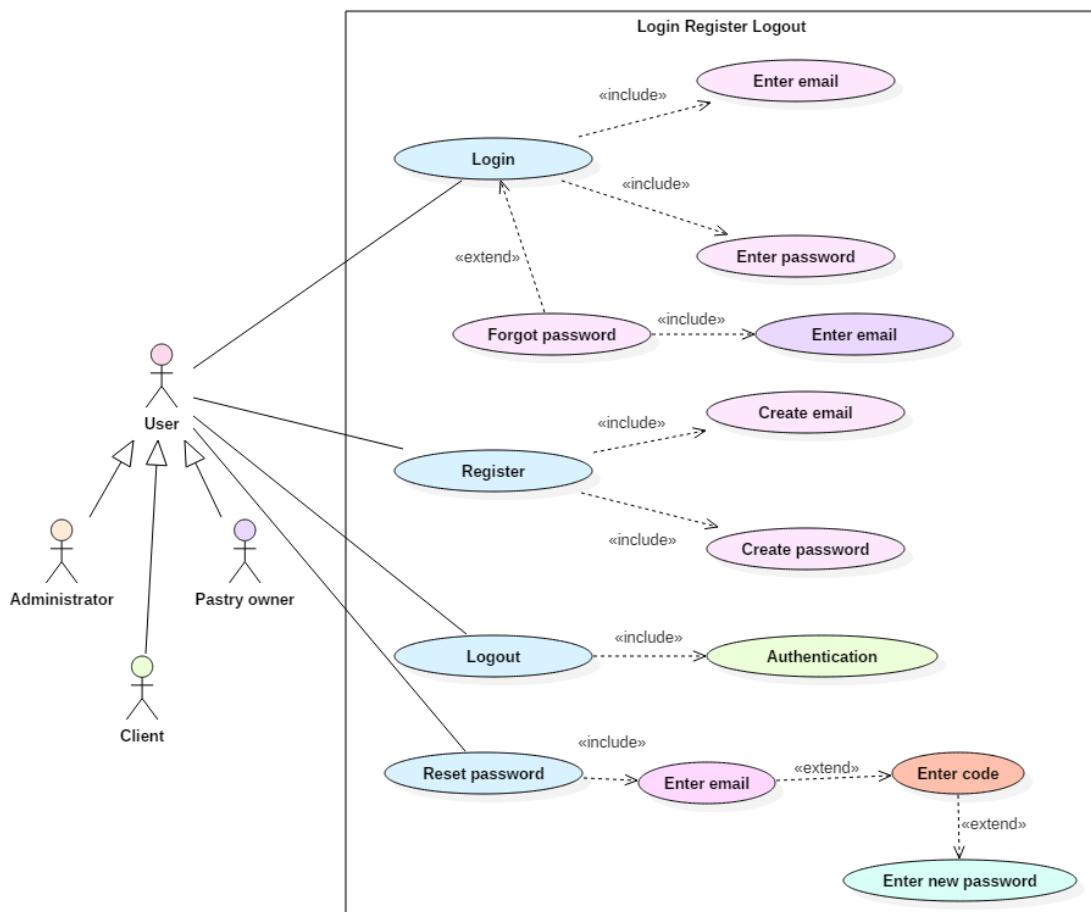


Figure 4.1: Use case Authentication

In Figure 4.2 we introduce the different use case of the sprint 2 “**Manage account**”.

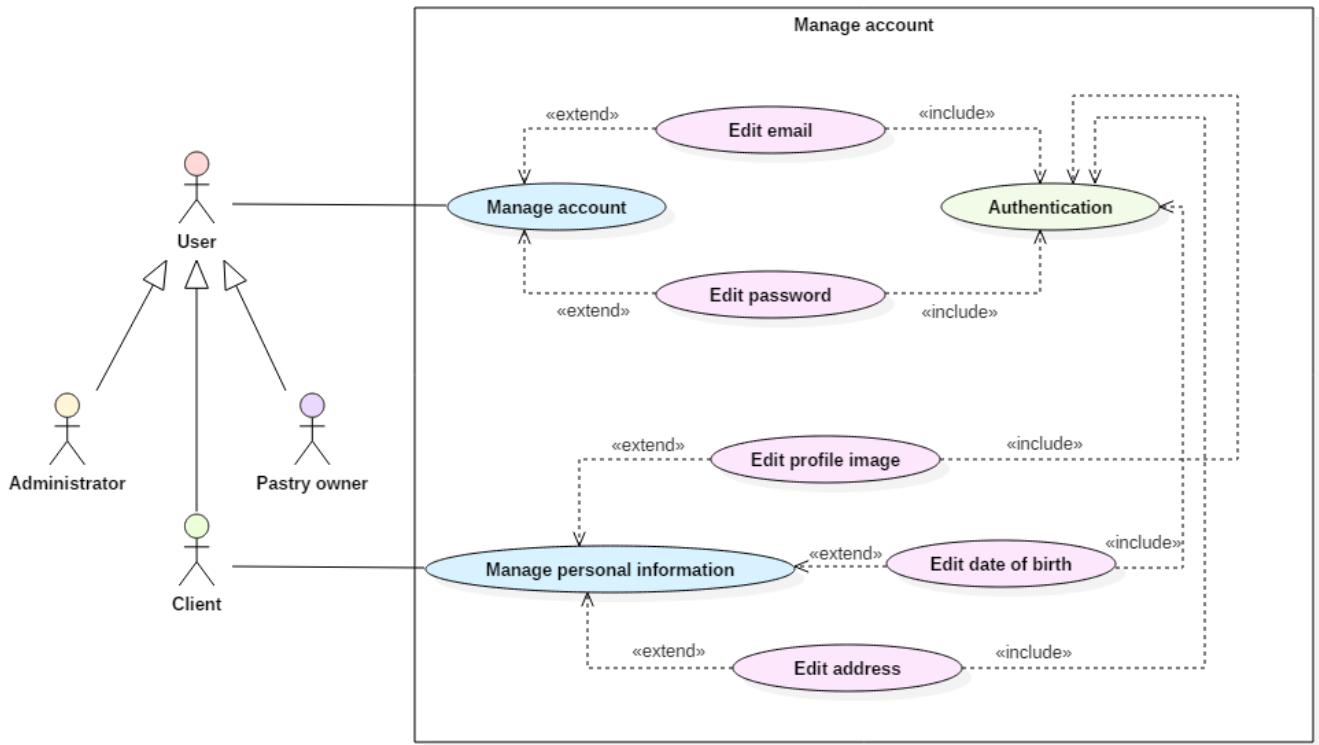


Figure 4.2: Use case edit account

We will be giving an idea about the sprint 3 “**Manage pastries**” via Figure 4.3.

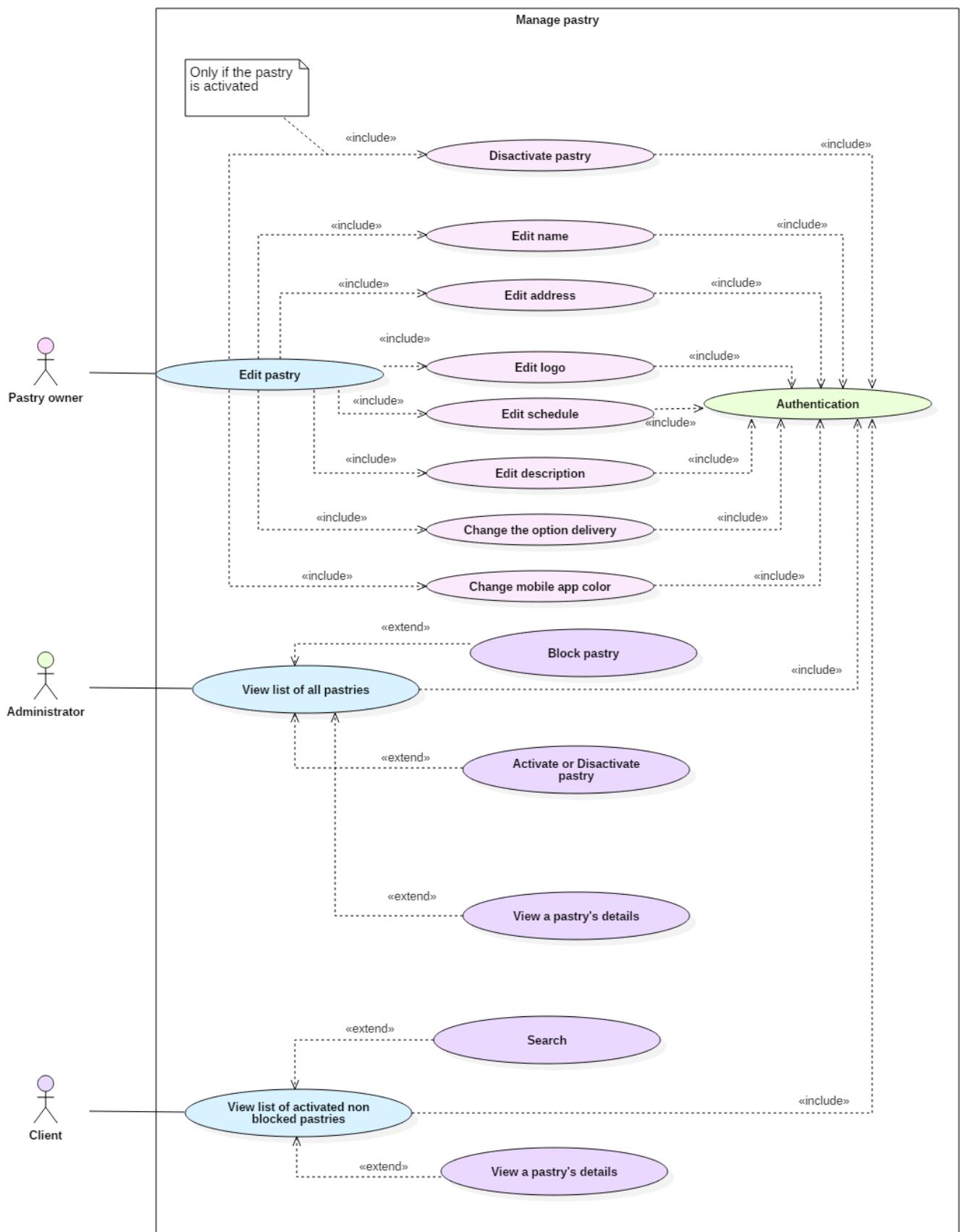


Figure 4.3: Use case manage pastries

Figure 4.4 is a graphic presentation of sprint 4 “**Manage products**”.

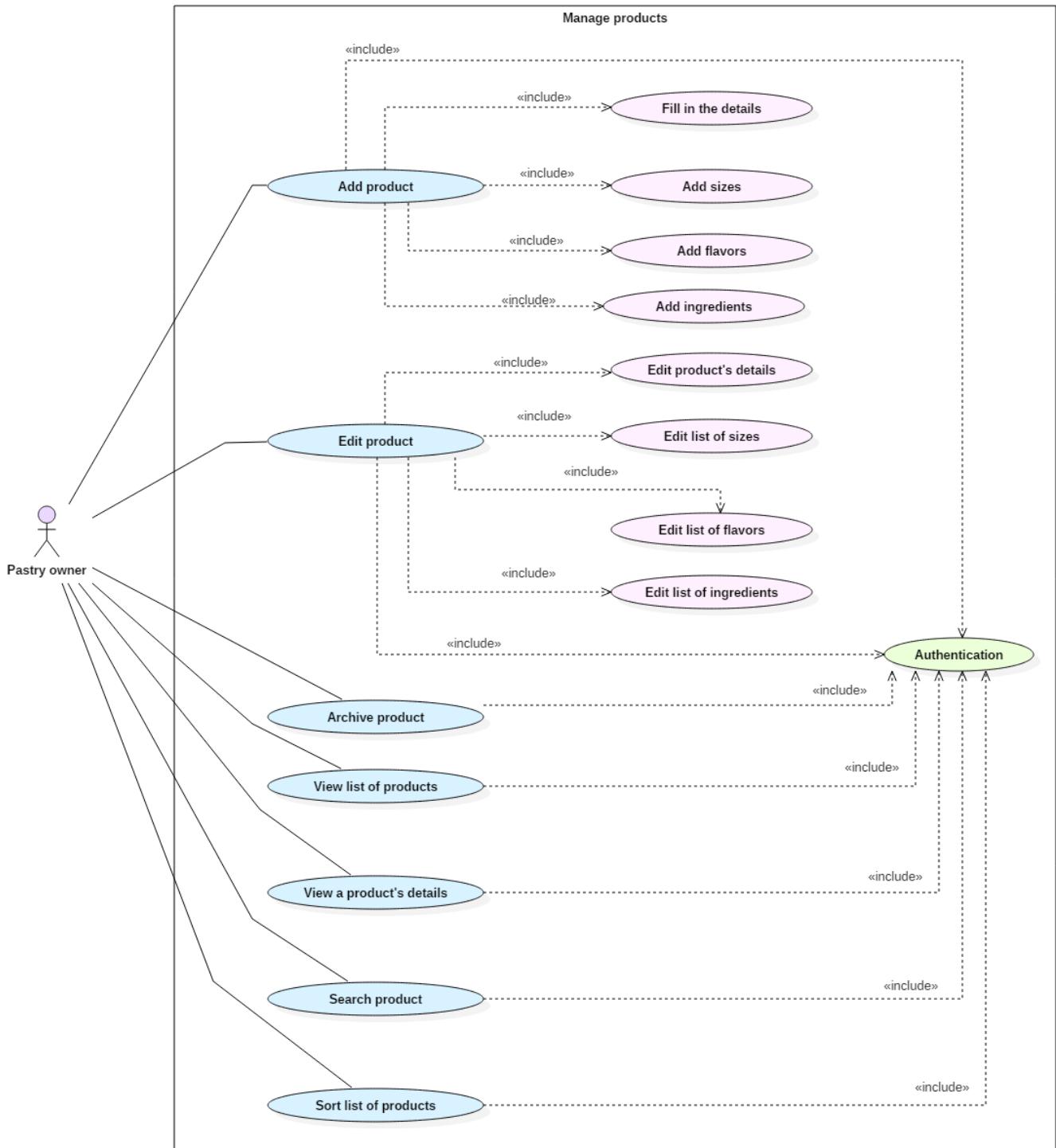


Figure 4.4: Use case manage products

## 4.2.2 Use case description

### Use case "Authentication" description

Table 4.2: Use case "Authentication" description

| Use case "Authentication" |  |
|---------------------------|--|
| <b>Actors</b>             | User   |
| <b>Precondition</b>       | Have or will create an account   |
| <b>Postcondition</b>      | The user have ,will create an account or is already logged in  |
| <b>Main scenario</b>      | <ul style="list-style-type: none"> <li>• The user does not have an account           <ul style="list-style-type: none"> <li>– The system will require the used to fill in the email, password and confirm password,</li> <li>– The user fill in the form,</li> <li>– The system verify the information,</li> <li>– the system shows the home of the application whether it is the web or mobile.</li> </ul> </li> <li>• The user have an account and is not logged in           <ul style="list-style-type: none"> <li>– The system will require the used to fill in the email and password,</li> <li>– The user fill in the form,</li> <li>– The system verify the information,</li> <li>– the system shows the home of the application whether it is the web or mobile.</li> </ul> </li> </ul> |

|                             |  |
|-----------------------------|--|
|                             | <ul style="list-style-type: none"> <li>• The user have an account and is logged in           <ul style="list-style-type: none"> <li>– The system will ask for logout confirmation,</li> <li>– The user will confirm,</li> <li>– The system will log out the user and show the login page.</li> </ul> </li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

### Use case "Manage account" description

Table 4.3: Use case "Manage account" description

| Use case "Manage account"   |   |
|-----------------------------|---|
| <b>Actors</b>               | User  |
| <b>Precondition</b>         | The user is authenticated   |
| <b>Postcondition</b>        | account edited  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• The user access his account ,</li> <li>• The user his account,</li> <li>• The system will save the changes.</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

### Use case "Manage pastries" description

Table 4.4: Use case "Manage pastries" description

| Use case "Manage pastries"  |   |
|-----------------------------|---|
| <b>Actors</b>               | User  |
| <b>Precondition</b>         | Pastry exists ,user connected and authorized  |
| <b>Postcondition</b>        | Pastry edited   |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• The user access the specific pastry to edit, deactivate, activate or block,</li> <li>• The user edit the pastry,</li> <li>• The system will save the changes.</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

### Use case "Manage products" description

Table 4.5: Use case "Manage products" description

| Use case "Manage products"  |  |
|-----------------------------|--|
| <b>Actors</b>               | User   |
| <b>Precondition</b>         | The user is authenticated and authorized   |
| <b>Postcondition</b>        | Product added or edited  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• The user access the list of products,</li> <li>• The user make changes,</li> <li>• The system will save the changes.</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

## 4.3 Sequence diagram

We will be presenting multiple sequence diagrams for some of the sprints we have in this release.

### Sequence diagram "Authentication"

Starting by the first sprint **Authentication** showing in Figure 4.5. The use whether it is an administrator, pastry owner or even a client if he is not logged in he has two option either create an account or login in he already have an account, To create an account all he has to do is to enter the required information (email, password and password confirmation) if all is good the controller will ask the database for the information needed to create the session such as a token and also send a notification and a mail to the administrator and will redirect the use to the home interface, else if there is any kind of an error the system will show an error message describing the error and ask the user to try again.

In case the user already have an account he can login by giving the email and password

on condition they are valid and match a record on the database, if not the user will receive an error message.

Last but not least, the user is logged in and wants to end his visit, he can simply logout.

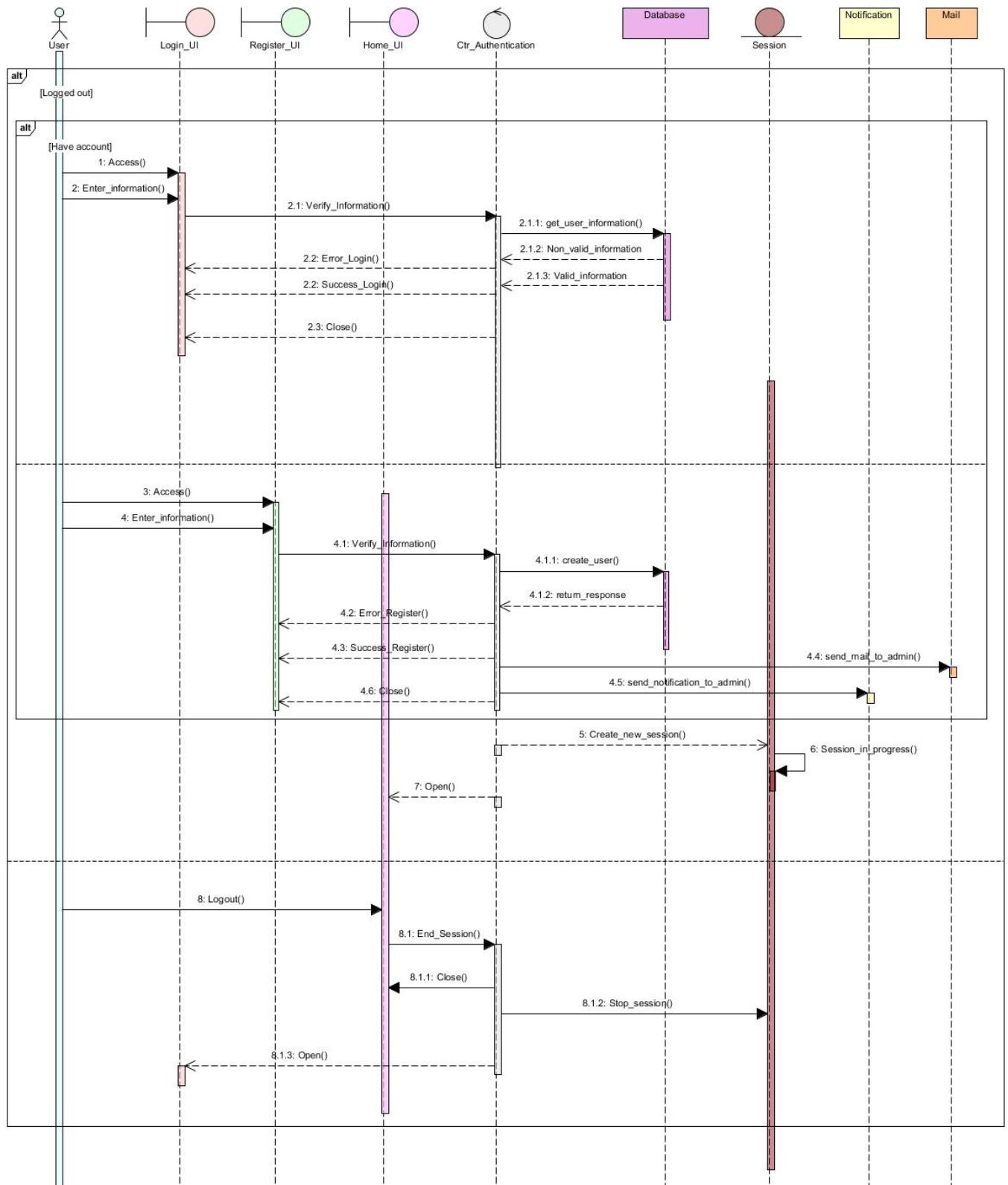


Figure 4.5: Authentication sequence diagram

### Sequence diagram "Manage product"

Figure 4.6 represent the sequence diagram for the last sprint of this release **Manage product**.

As we described before this sprint has four U.S:

- Add product:

To add a product the pastry owner should access the add product interface and fill in the required data for it (name; image, description, price, ingredients ...), the next step is to confirm his addition, the controller will receive the data and verify them, in case all is good, the product is saved to the database and the controller will wait for a response from it, after that the user will be redirected to the list of products including the new product that was just added.

- Edit product:

Editing a product will take basically the same steps as adding one, first of all the user will access the edit interface of a specific product, he will make the changes that he wants to make and confirm them. The controller will receive those changes and verify every one of them, if the verification is a success, the controller will update the requested product and redirect the user to the show product interface.

- View list of products:

As for viewing the list of products, the pastry owner just needs to navigate to the list of products interface, and the controller will request the list from the database and send it back to the view.

- Archive a product:

In order to archive a product the user needs to access the list of products list first and select the archive button, the controller will make a request to the database to add this product to the archived list then it will show a success message or an error message depending on the database response.

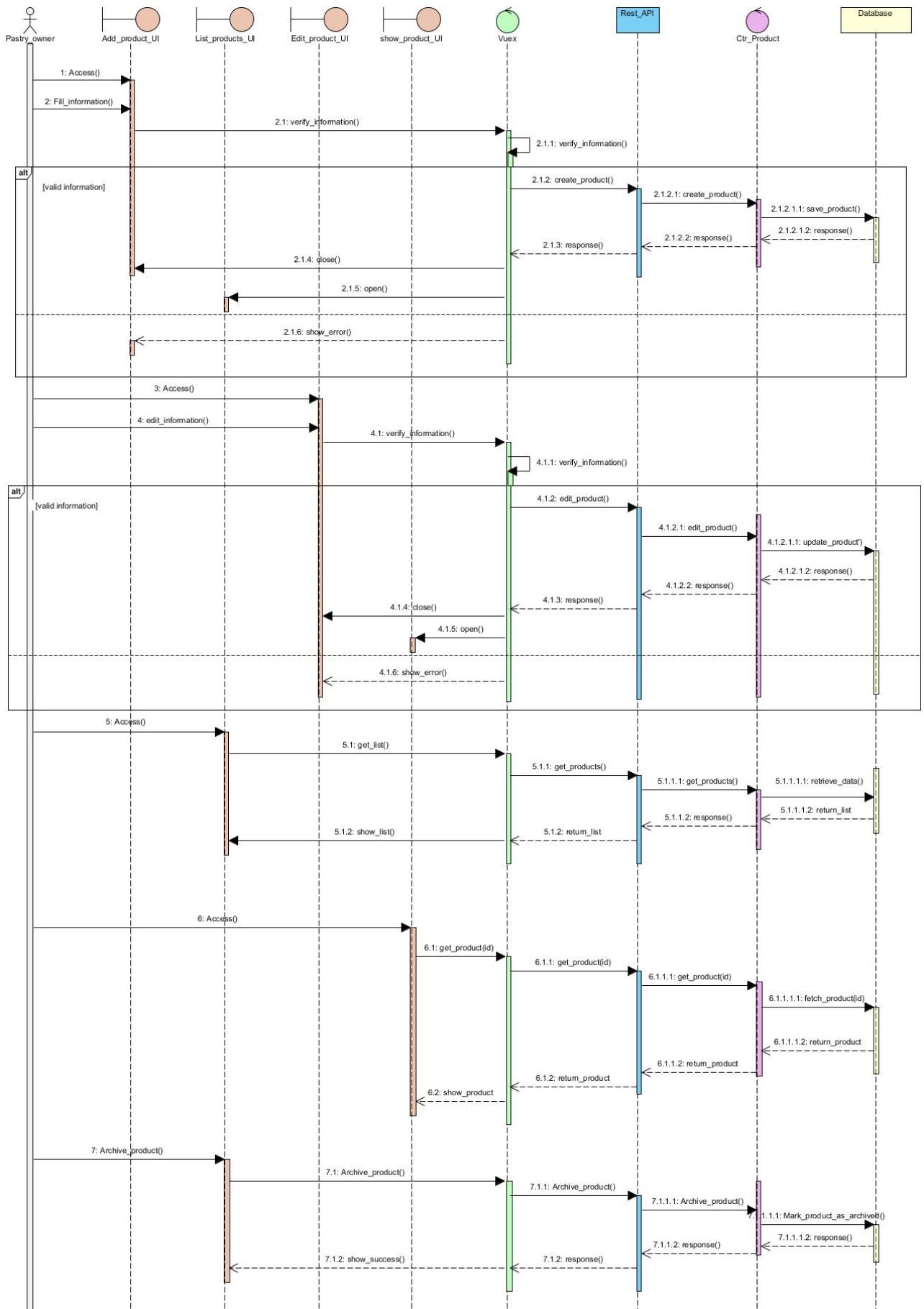


Figure 4.6: Manage product sequence diagram

## 4.4 Implementation

We will be presenting the different interface of our mobile and web application for this release.

### 4.4.1 Web

#### Sprint "Authentication"

Starting by the authentication (login, register, reset password, logout), Figure 4.7 shows the interface for the register task for the web application where the user have to fill in the form in order to create a pastry owner account, all the entered data has to be valid, in case of an error the system will show the user where is the mistake.

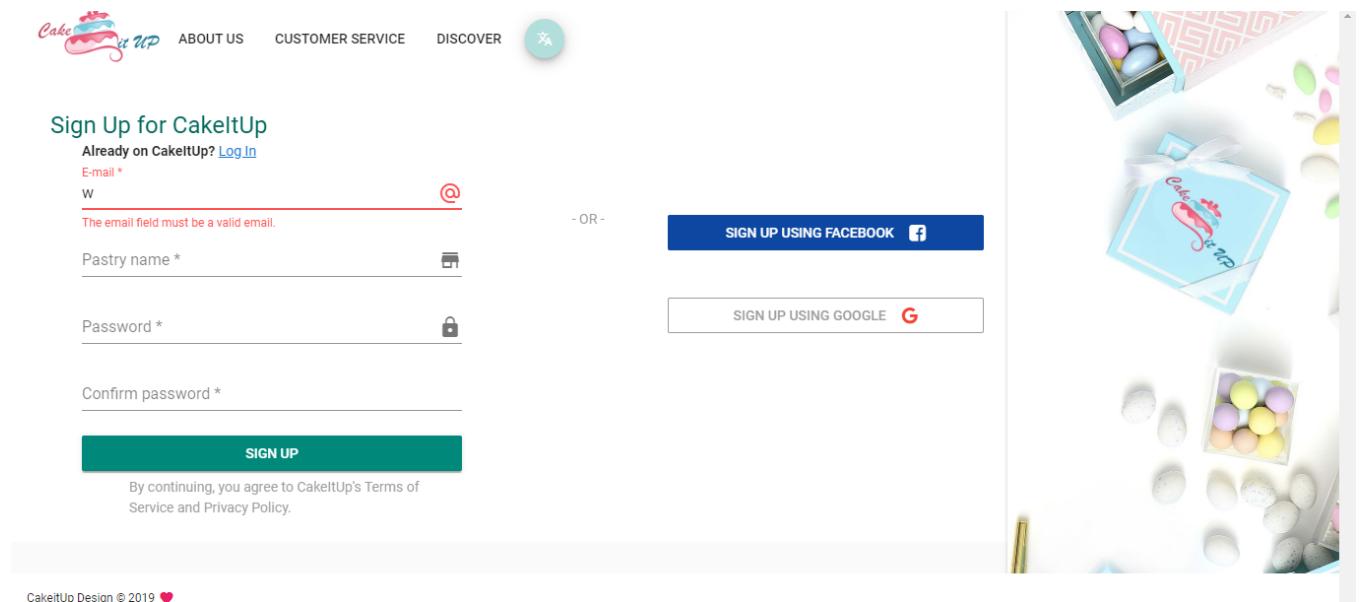


Figure 4.7: Register interface web

As shown in Figure 4.8, the interface of login is simple and elegant, the user will fill in the data and login if no error has been displayed.

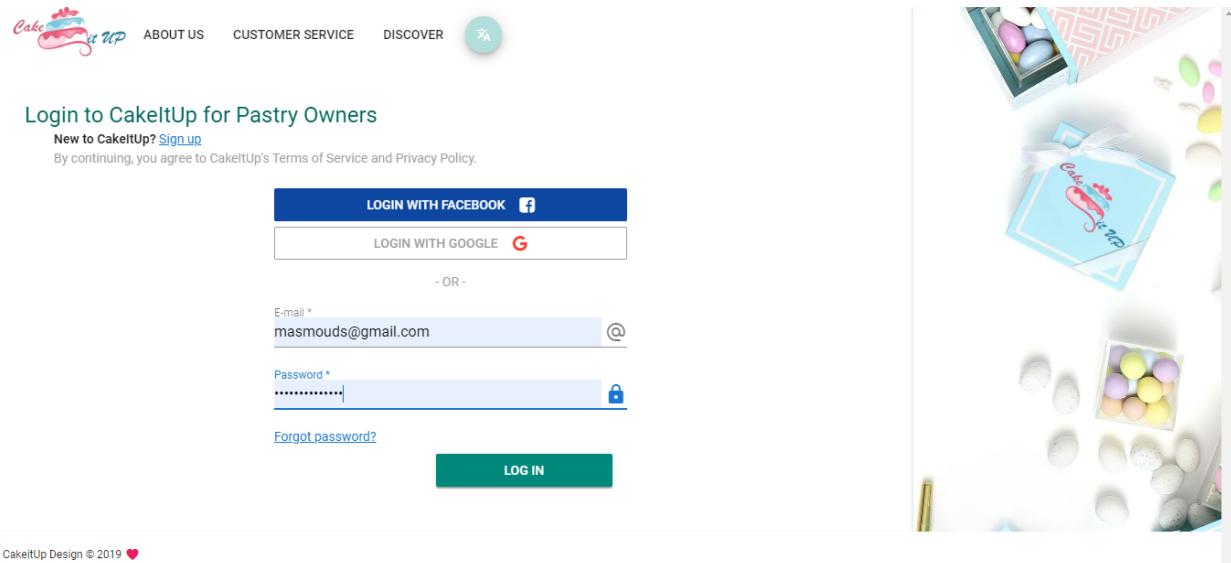


Figure 4.8: Login interface web

In case the user forgot his password, he can reset it by entering his email as shown in Figure 4.9, a mail will be sent to him with the code needed to reset the password, once he entered the code, it will be verified if it is the right one the system will show the next step which is resetting the password.

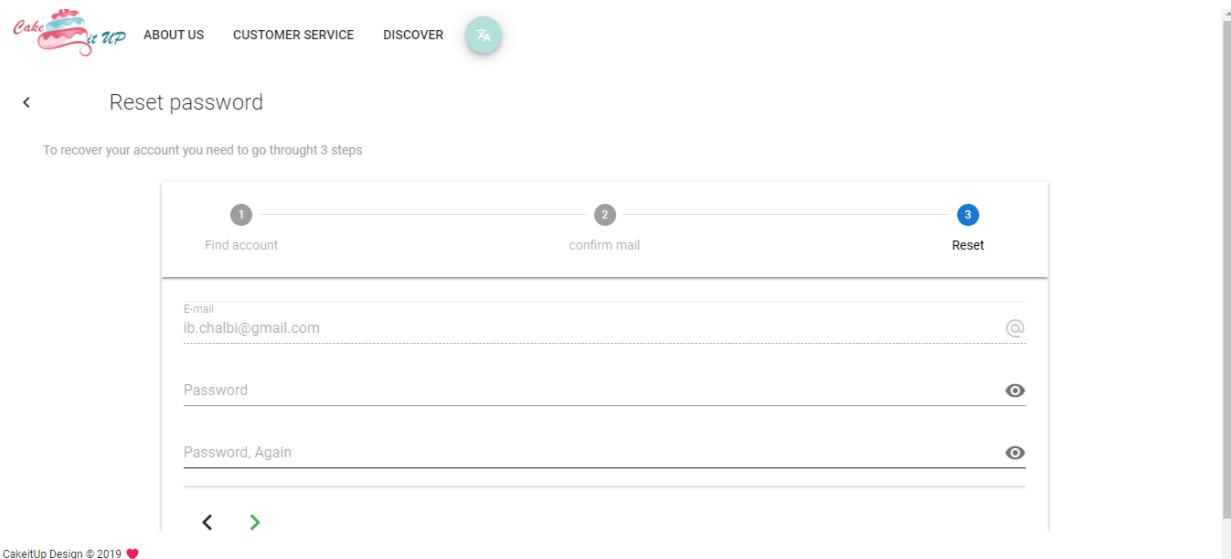


Figure 4.9: Reset password interface web

Once the user has been logged in he has the ability to logout by a simple click as presented in Figure 4.10.

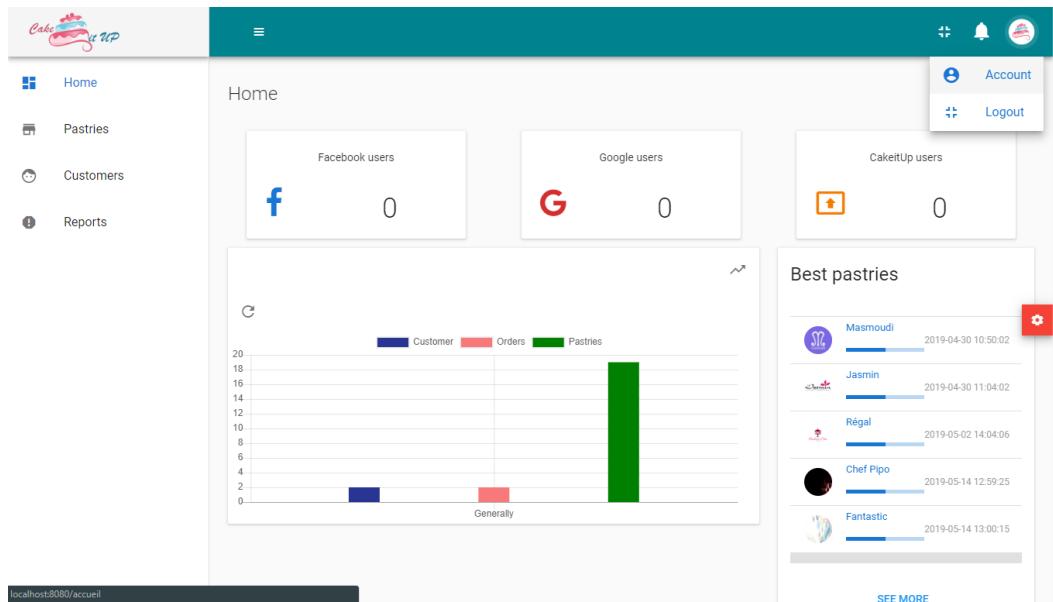


Figure 4.10: Logout interface web

### Sprint "Manage account"

Both the web users (Administrator and pastry owner) have the permission to edit their account, they can change the email and their password as well.

The screenshot shows the 'Edit account' interface. On the left, there is a sidebar with icons for Home, Pastries, Customers, and Reports. The main area displays the user's profile information:

| Information | Value                                 |
|-------------|---------------------------------------|
| Phone       | 55 358 310<br>Mobile                  |
| Email       | cakeitup.app.ib@gmail.com<br>Personal |
| Address     | Riad Sousse,Tunisie                   |

On the right, there is a 'Settings' sidebar with the following sections:

- Language**: Shows 'en' selected. A dropdown menu is open.
- Account**: Shows 'Password \*' and 'E-mail' fields with 'MODIFY' buttons.

Figure 4.11: Edit account interface web

## Sprint "Manage pastries"

The administrator can view the list of all pastries where he can access to just one of them, block them, activate or dis-activate them as we can see in Figure 4.12.

| Name         | Creation date       | Account level                                   |
|--------------|---------------------|---|
| ahmed pastry | 2019-05-17 14:18:33 | <div style="width: 100px; height: 10px;"></div> |
| Chef Pipo    | 2019-05-14 12:59:25 | <div style="width: 100px; height: 10px;"></div> |
| Dream pastry | 2019-05-14 13:00:31 | <div style="width: 100px; height: 10px;"></div> |
| Fantastic    | 2019-05-14 13:00:15 | <div style="width: 100px; height: 10px;"></div> |
| hech pastry  | 2019-05-17 14:20:30 | <div style="width: 100px; height: 10px;"></div> |

Figure 4.12: List of pastries interface web

We have already gave a detailed description of the task edit pastry, only the pastry owner can edit his own pastry, as shown in Figure 4.13 the user can make a lot of changes depending on his choice.

**Masmoudi**

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritas et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni

adresse: Tunisie

Tunisie

Rue Montri, Rue Lépine, Rue Garibaldi, Avenue Habib Bourguiba, Rue Radha Haddad, Avenue Farhat Hached

**Reviews**

ibti ch this pastry is the best 5/2/2019

**Mobile**

You have the ability to change the mobile app's color

Please choose a color:

[Color swatches]

Figure 4.13: Edit pastry interface web

## Sprint "Manage products"

Figure 4.14 gives us an idea about what the list of product look like in the web application, the list is click-able, a click will take you to the specific product, you can also access the edit mode and archive a product from it.

Figure 4.14: Product list interface web

Adding a product can be found only on the web side where a pastry owner can describe his product and fill in all the forms.

Figure 4.15: Add product interface

The pastry owner can edit his product and change any information he wants as long as it is valid.

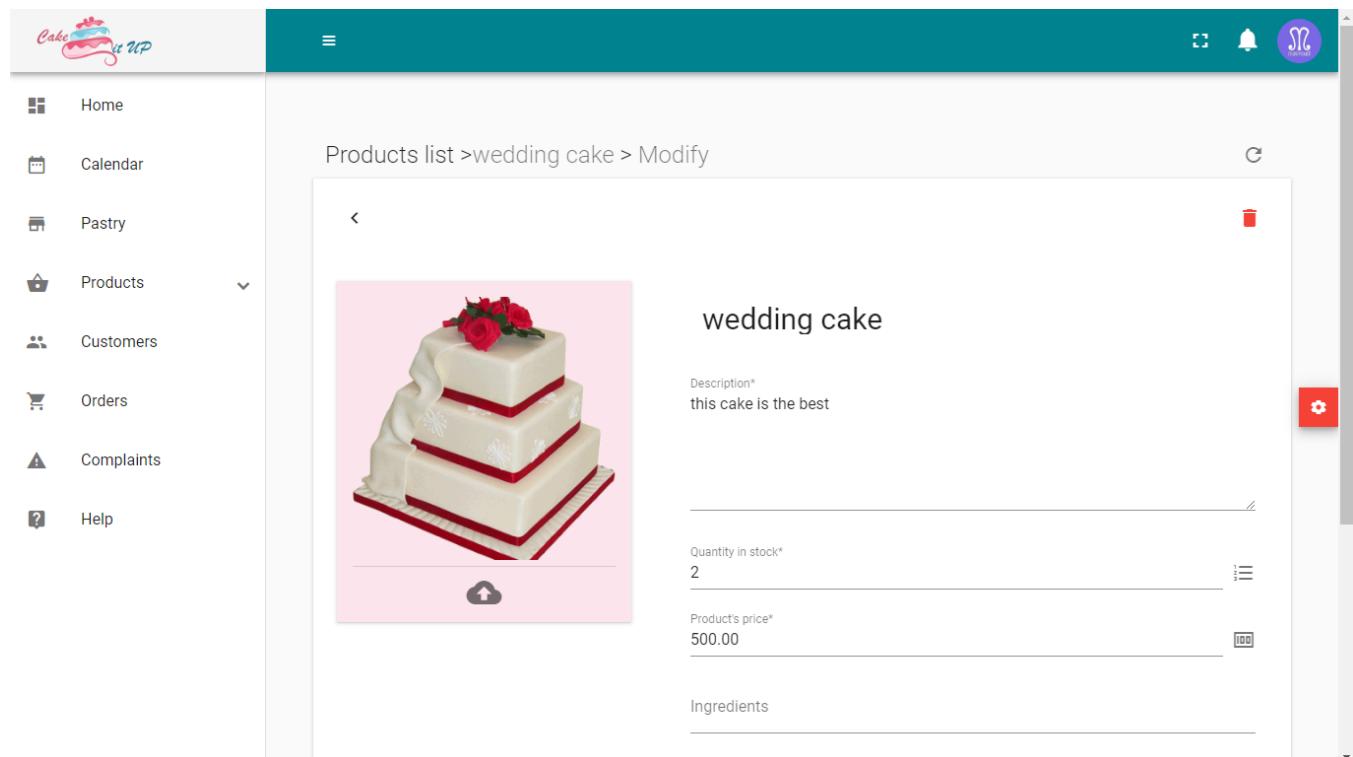


Figure 4.16: Edit product interface

#### 4.4.2 mobile

##### Sprint "Authentication"

The client won't have access to his profile unless he logs in, the interface shown in Figure 4.17 presents the layout showed when the user is a visitor.

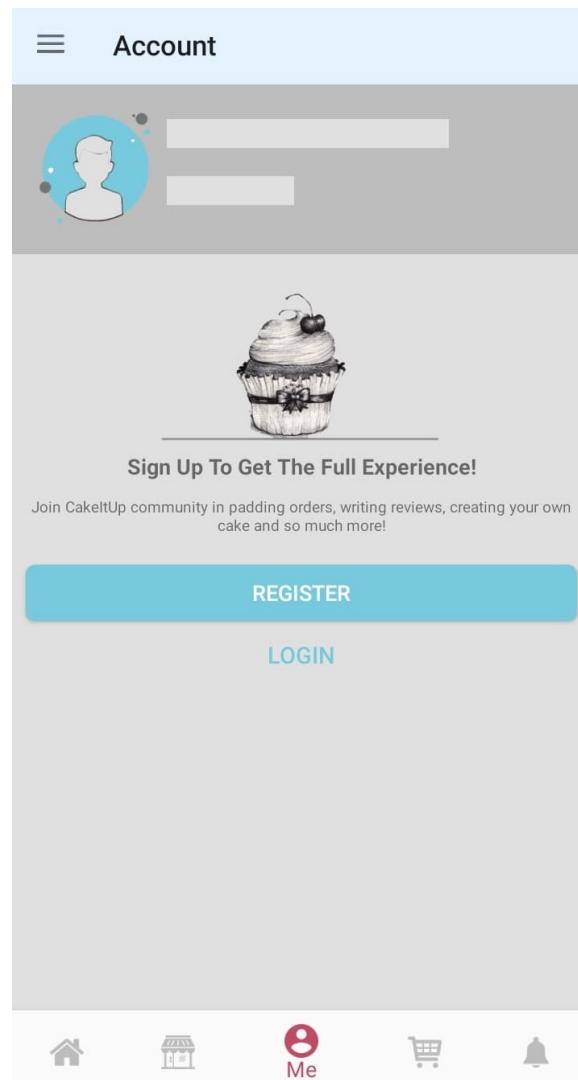


Figure 4.17: Profile non connected mobile interface

To create an account, the client just needs to fill in the form show in Figure 4.18.

Once the account is created, the client can login by filling in the valid information.



CREATE YOUR ACCOUNT NOW

Already on CakeltUp? [Login](#)

@ Email

Name

Password

Confirm password

**Register**

(a) Register interface



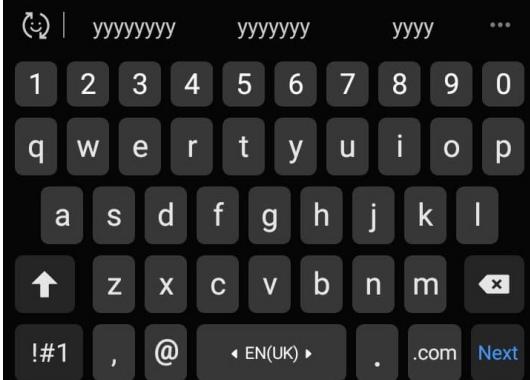
LOGIN

@ yy ! The email should be valid

Password

**Login**

Don't have a cakeltUp account yet? [Sign up](#)



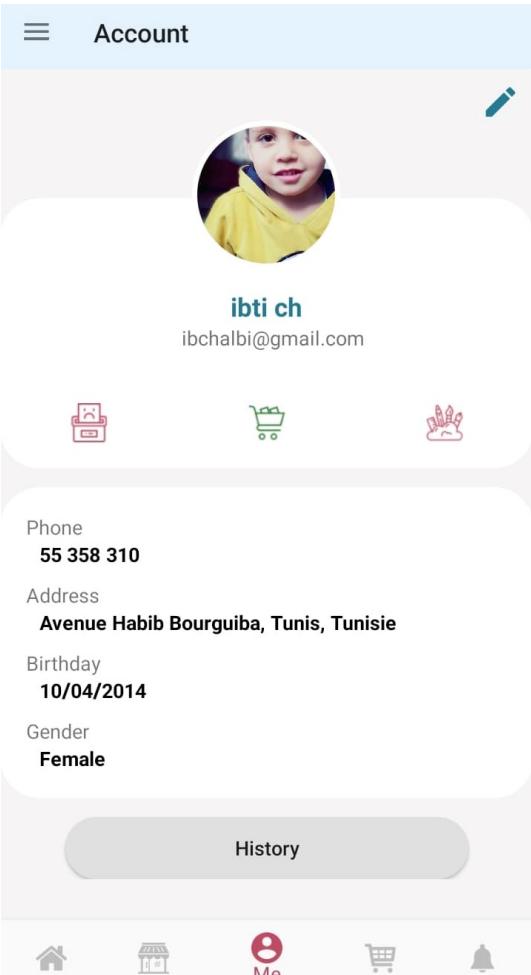
(b) Login interface

Figure 4.18: Register and login mobile interface

### Sprint "Manage account"

Once the user is logged in he has access to his own profile where he can find his personal information.

The client can edit his profile as shown in Figure 4.19.



**ibti ch**  
ibchalbi@gmail.com

Phone  
**55 358 310**

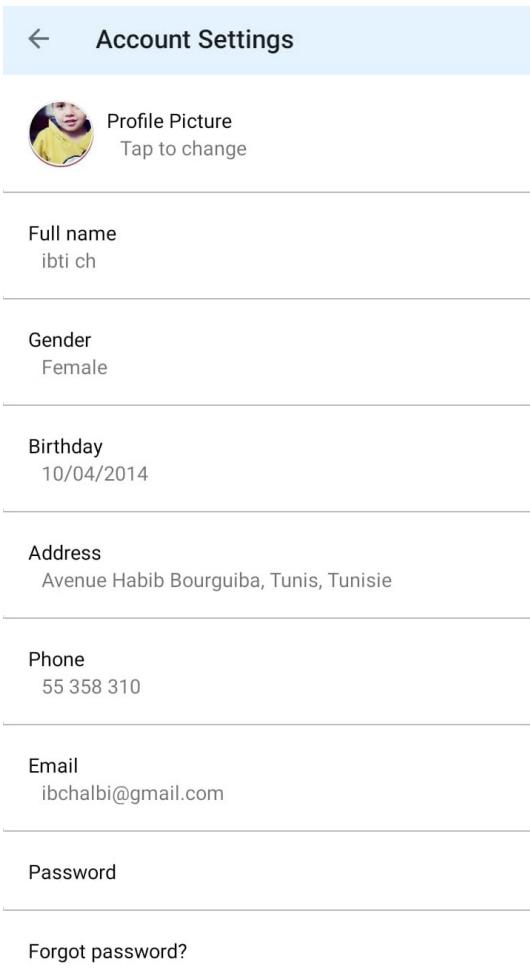
Address  
**Avenue Habib Bourguiba, Tunis, Tunisie**

Birthday  
**10/04/2014**

Gender  
**Female**

**History**



**Account Settings**

Profile Picture  
Tap to change

Full name  
**ibti ch**

Gender  
**Female**

Birthday  
**10/04/2014**

Address  
**Avenue Habib Bourguiba, Tunis, Tunisie**

Phone  
**55 358 310**

Email  
**ibchalbi@gmail.com**

Password

Forgot password?

(a) Register interface      (b) Login interface

Figure 4.19: Register and login mobile interface

Resetting a password has the same process as the web application.

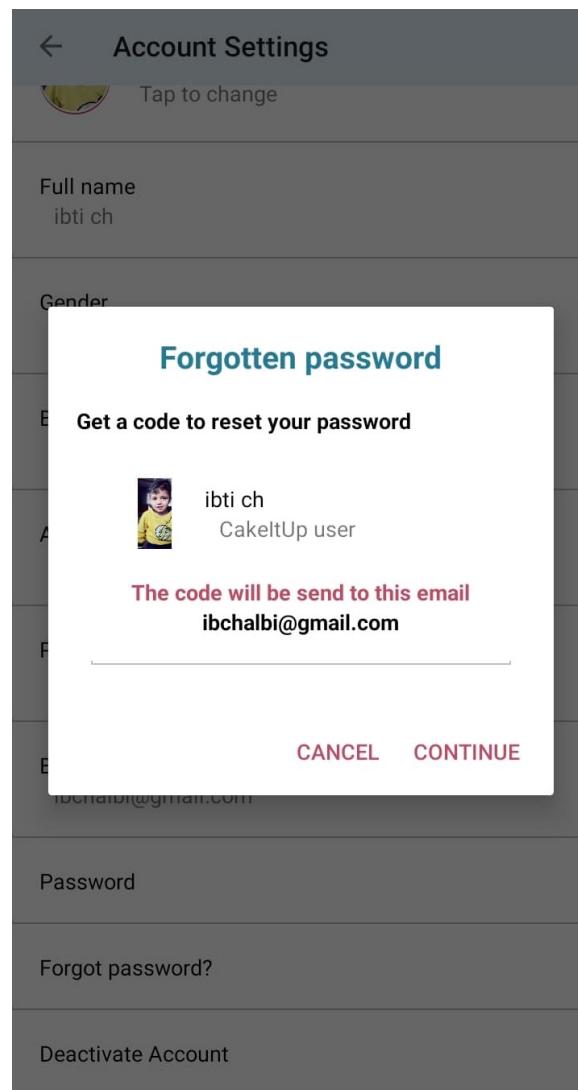


Figure 4.20: Reset password mobile interface

### Sprint "Manage pastries"

The list of pastries is accessible to all users the only difference is that some function are not allowed if you are not logged in.

A single pastry can be viewed by clicking on one of the pastries in the list.

**(a) Register interface**

Pastries

JASMIN

3.9 ★★★★☆

Avenue Habib Bourguiba,  
Tunis, Tunisie

dolore magnam aliquam quaerat voluptatem. Ut enim  
ad minima veniam, quis nostrum exercitationem ullam  
corporis suscipit laboriosam, nisi ut aliquid ex ea commo...

**(b) Login interface**

Jasmin

Le Jasmin

JASMIN

3.9 ★★★★☆

Avenue Habib Bourguiba, Tunis, Tunisie

dolore magnam aliquam quaerat voluptatem. Ut enim  
ad minima veniam, quis nostrum exercitationem ullam  
corporis suscipit laboriosam, nisi ut aliquid ex ea commo...

Top products

Show all >

No top products available for now

Reviews

IBTI CH  
2019-04-24 00:00:00  
i did not like the popcakes

IBTI CH  
2019-05-05 00:00:00  
my review

Figure 4.21: Register and login mobile interface

## 4.5 Release retrospective

After the first sprint, we did the tests to see if we were able to achieve the goals set out. And for that, we rely on the sprint retrospective:

- What went well:
  - The tasks of this module went well. There were no problems in this sprint.
- What went wrong:
  - Nothing.
- What was not realized:
  - Nothing.
- What can be improved:
  - the presentation of the interfaces can be improved,
  - the validation of the form authentication can be enriched more.

## Conclusion

At the end of this chapter, we have managed to produce a work with sufficient value for the customer and can be used in a production environment. In the following chapter, our effort will be devoted to the production of a new release.

# Chapter 5

## Release 2

### Introduction

According to the established plan, the second Sprint focuses on the use cases: "Manage clients", "Manage complains", "Manage reports" and "Manage events".

### 5.1 Backlog

Table 5.1: Backlog: Release 2

| Sprint                         | ID<br>U.S | User story  | ID<br>task | Task   |
|--------------------------------|-----------|---|------------|--|
| Sprint 9<br><br>Manage clients | 9.1       | As an administrator i want to view list of all clients. | 9.1.A      | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|                                |           |   | 9.1.B      | Develop the view list of clients model for the administrator.                              |
|                                |           |   | 9.1.C      | Test the list of clients model.  |
|                                | 5.2       | As an administrator i want to block a client.           | 9.2.A      | Make the diagrams needed for this U.S.   |
|                                |           |   | 9.2.B      | Develop the block client model.  |

|  |     |   |       |  |
|--|-----|---|-------|--|
|  |     |   | 5.2.C | Test the block client model.   |
|  | 5.3 | As a pastry owner i want to view list of my clients.    | 5.3.A | Make the diagrams needed for this U.S.   |
|  |     |   | 5.3.B | Develop the list of clients model for the pastry owner.                                    |
|  |     |   | 5.3.C | Test the list of clients model.  |
| <b>Sprint 6</b><br><b>Manage Complains</b> | 6.1 | As a client i want to add a complain.                   | 6.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|  |     |   | 6.1.B | Develop the add complain model.  |
|  |     |   | 6.1.C | Test the add complain model.   |
|  | 6.2 | As a client i wnat to view my list of complains.        | 6.2.A | Make the diagrams needed for this U.S.   |
|  |     |   | 6.2.B | Develop the complains list model.  |
|  |     |   | 6.2.C | Test the complains list model.   |
|  | 6.3 | As a pastry owner i want to view the list of complains. | 6.3.A | Make the diagrams needed for this U.S.   |
|  |     |   | 6.3.B | Develop the list of complains model.   |
|  |     |   | 6.3.C | Test the complains list model.   |
|  | 6.4 | As a pastry owner i want to reply to a complain.        | 6.4.A | Make the diagrams needed for this U.S.   |
|  |     |   | 6.4.B | Develop the reply model.   |
|  |     |   | 6.4.C | Test the reply to complain model.  |
| <b>Sprint 7</b><br><b>Manage reports</b>   | 7.1 | As a client i want to report a pastry.                  | 7.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|  |     |   | 7.1.B | Develop the add report model.  |
|  |     |   | 7.1.C | Test the add report model.   |

|   |     |   |       |  |
|---|-----|---|-------|--|
|   | 7.2 | As a pastry owner i want to report a client.        | 7.2.A | Make the diagrams needed for this U.S.   |
|   |     |   | 7.2.B | Develop the add report model.  |
|   |     |   | 7.2.C | Test the add report model.   |
|   | 7.3 | As an administrator i want to view list of reports. | 7.3.A | Make the diagrams needed for this U.S.   |
|   |     |   | 7.3.B | Develop the list of reports model.   |
|   |     |   | 7.3.C | Test the reports list model.   |
| <b>Sprint 8</b><br><b>Manage events</b> | 8.1 | As a pastry owner i want to add a personal event.   | 8.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |     |   | 8.1.B | Develop the add event model.   |
|   |     |   | 8.1.C | Test the add event model.  |
|   | 8.2 | As a pastry owner i want to view my calender.       | 8.2.A | Make the diagrams needed for this U.S.   |
|   |     |   | 8.2.B | Develop the calender model.  |
|   |     |   | 8.2.C | Test the calender model.   |

## 5.2 Use case

### 5.2.1 Use case diagram

We will be following the same steps as the previous release, starting by the use case of the first sprint of this release “**Manage clients**” presented in Figure 5.1.

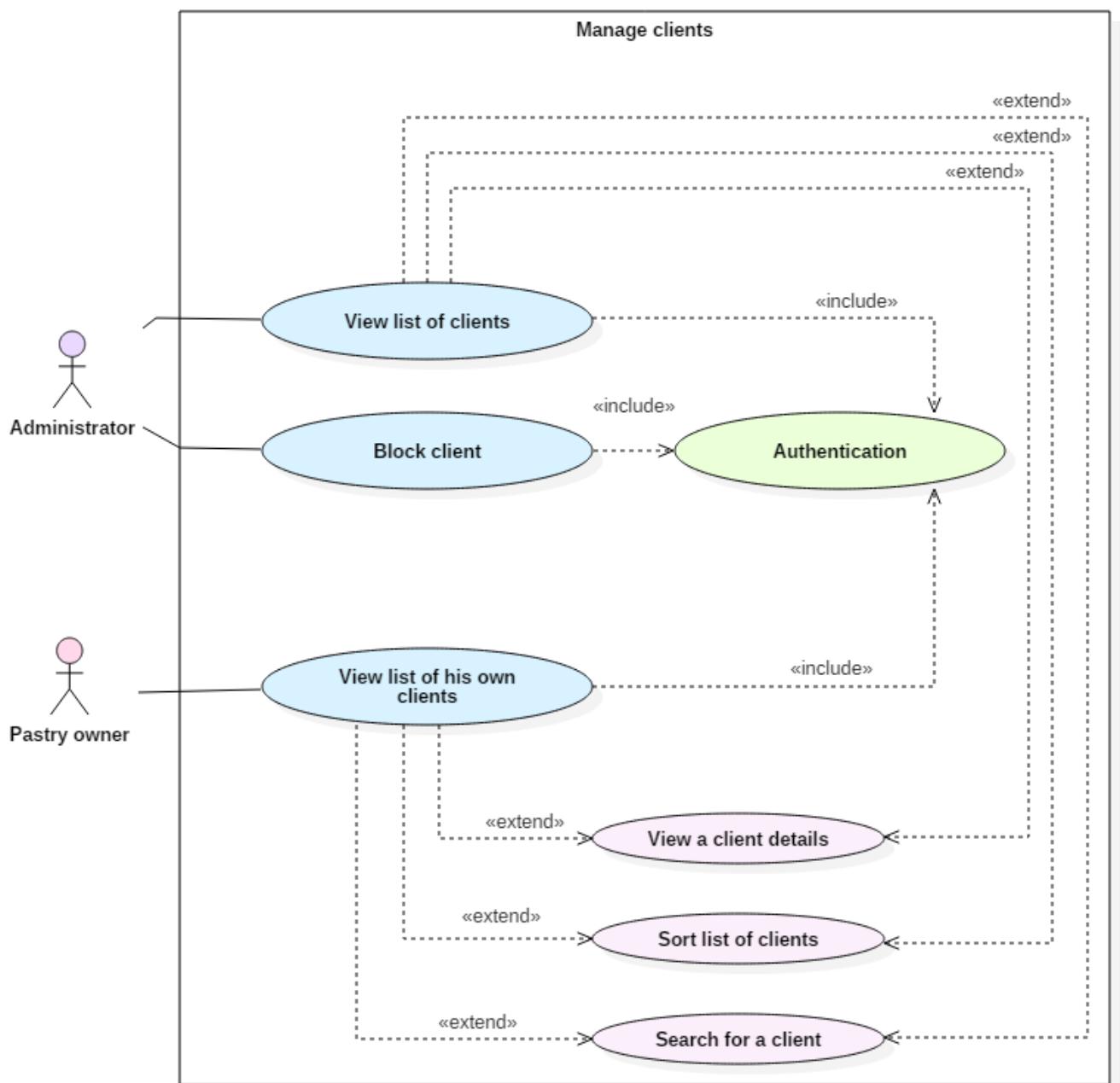


Figure 5.1: Use case Manage clients

As for the second sprint “**Manage complains**”, we created another use case diagram showing in Figure 5.2

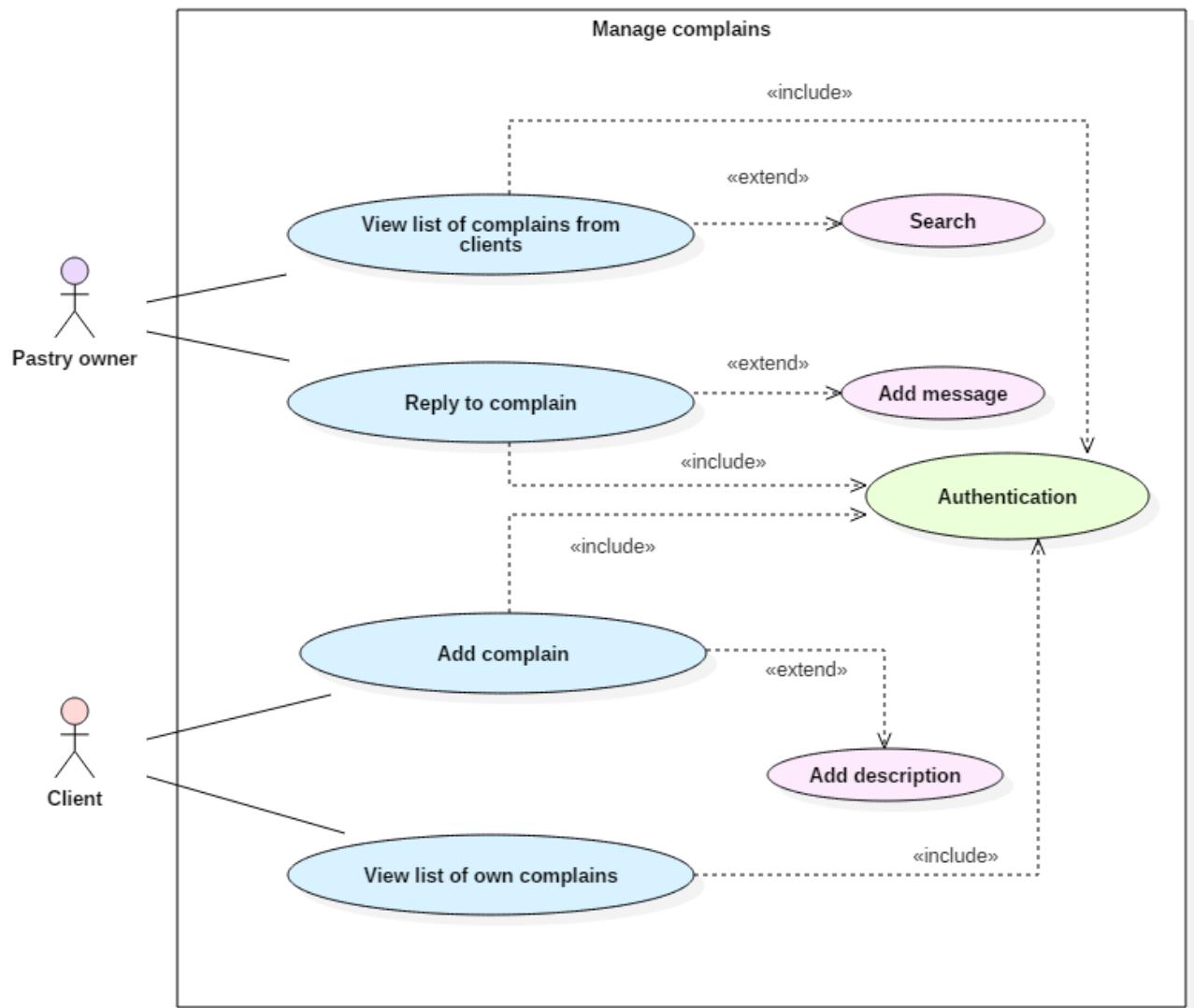


Figure 5.2: Use case Manage complains

Sprint “**Manage complains**” which is the third sprint this this release is presented in Figure 5.3.

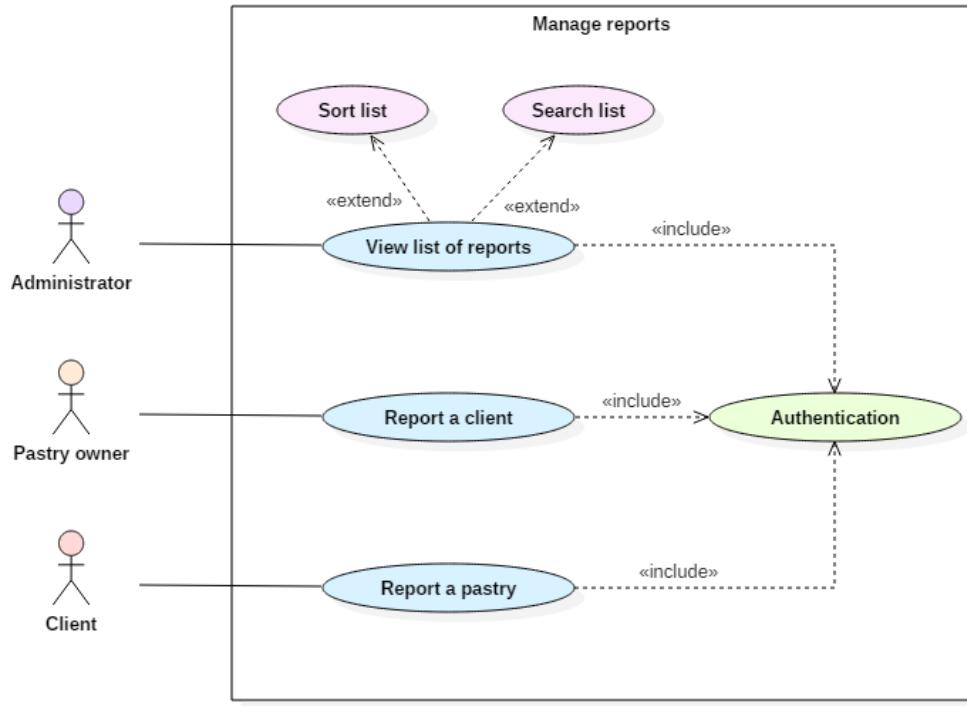


Figure 5.3: Use case Manage reports

The last sprint in this release is “**Manage events**”, Figure 5.4 is a use case diagram that describes the different cases of this sprint.

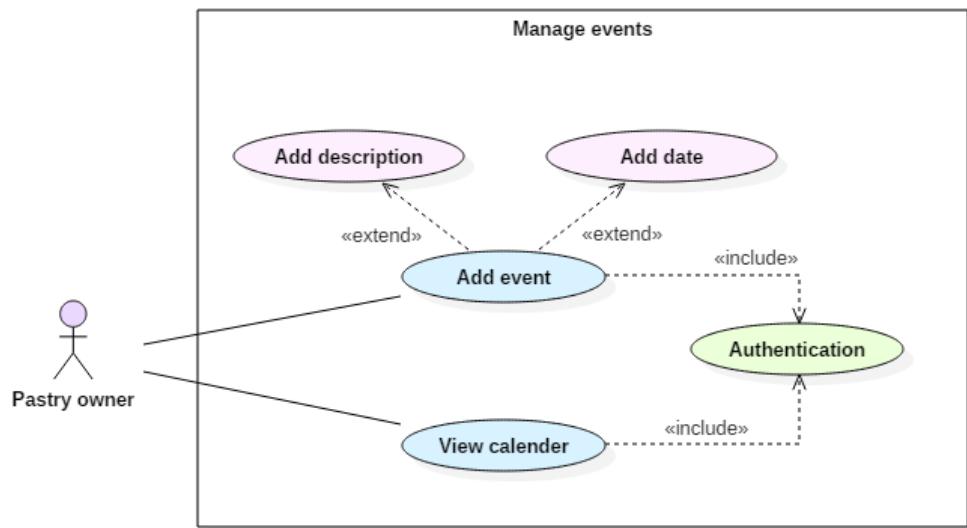


Figure 5.4: Use case Manage events

## 5.2.2 Use case description

We will be description each and every use case of this release in this section.

### Use case "Manage clients" description

Table 5.2: Use case "Manage clients" description

| Use case "Manage clients"   |  |
|-----------------------------|--|
| <b>Actors</b>               | Administrator and pastry owner   |
| <b>Precondition</b>         | Clients exists and the actors are authenticated and authorized   |
| <b>Postcondition</b>        | Clients viewed or blocked  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• block client,           <ul style="list-style-type: none"> <li>– the administrator block a client,</li> <li>– The system save changes.</li> </ul> </li> <br/> <li>• view list of clients.           <ul style="list-style-type: none"> <li>– the actor access the list of clients,</li> <li>– The actor sort, search or select a client from the list.</li> </ul> </li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

### Use case "Manage complains" description

Table 5.3: Use case "Manage complains" description

| Use case "Manage complains" |   |
|-----------------------------|---|
| <b>Actors</b>               | The pastry owner and the client   |
| <b>Precondition</b>         | the actors are authorized and authenticated   |
| <b>Postcondition</b>        | A complain is created or updated  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• Add complain,           <ul style="list-style-type: none"> <li>– the client will send a complain to a specific pastry,</li> <li>– The system save the complain.</li> <li>– The pastry owner will receive the complain.</li> </ul> </li> <br/> <li>• Reply to a complain.           <ul style="list-style-type: none"> <li>– the pastry owner will reply to the complain,</li> <li>– The system update the complain.</li> <li>– The client will receive the reply.</li> </ul> </li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

### Use case "Manage reports" description

Table 5.4: Use case "Manage reports" description

| Use case "Manage reports"   |   |
|-----------------------------|---|
| <b>Actors</b>               | The user  |
| <b>Precondition</b>         | the actors are authorized and authenticated   |
| <b>Postcondition</b>        | A report is created or viewed   |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• Add report,           <ul style="list-style-type: none"> <li>– a client or a pastry owner can report each other,</li> <li>– The system save the report.</li> </ul> </li> <li>• view reports           <ul style="list-style-type: none"> <li>– the administrator view the list of reports</li> </ul> </li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

### Use case "Manage events" description

Table 5.5: Use case "Manage events" description

| Use case "Manage events"    |   |
|-----------------------------|---|
| <b>Actors</b>               | The pastry owner  |
| <b>Precondition</b>         | the pastry owner is authenticated   |
| <b>Postcondition</b>        | An event is created or viewed   |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• Add event,           <ul style="list-style-type: none"> <li>– The pastry owner create an event,</li> <li>– The system save the event.</li> </ul> </li> <br/> <li>• view reports           <ul style="list-style-type: none"> <li>– The pastry owner view his calender with the list of events</li> </ul> </li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

## 5.3 Sequence diagram

The same as the previous release will be dedicating this section for the sequence diagrams.

We have four sprint and we will be presenting only two of them to give an idea about the work flow of our application.

### Sequence diagram "Manage complains"

In the third sprint of this release which is **Manage complains** we have two actors with two different applications but the same database.

Figure 5.5 is giving us a detailed description of the way our system works. Since we have two different application, means we have two different work flow, for the web part it is

MVC where the user have an interaction with the view and the view with the controller and the controller with the database. The mobile application is based on MVP where the user interacts with presented and this last with the repository and the repository with the database.

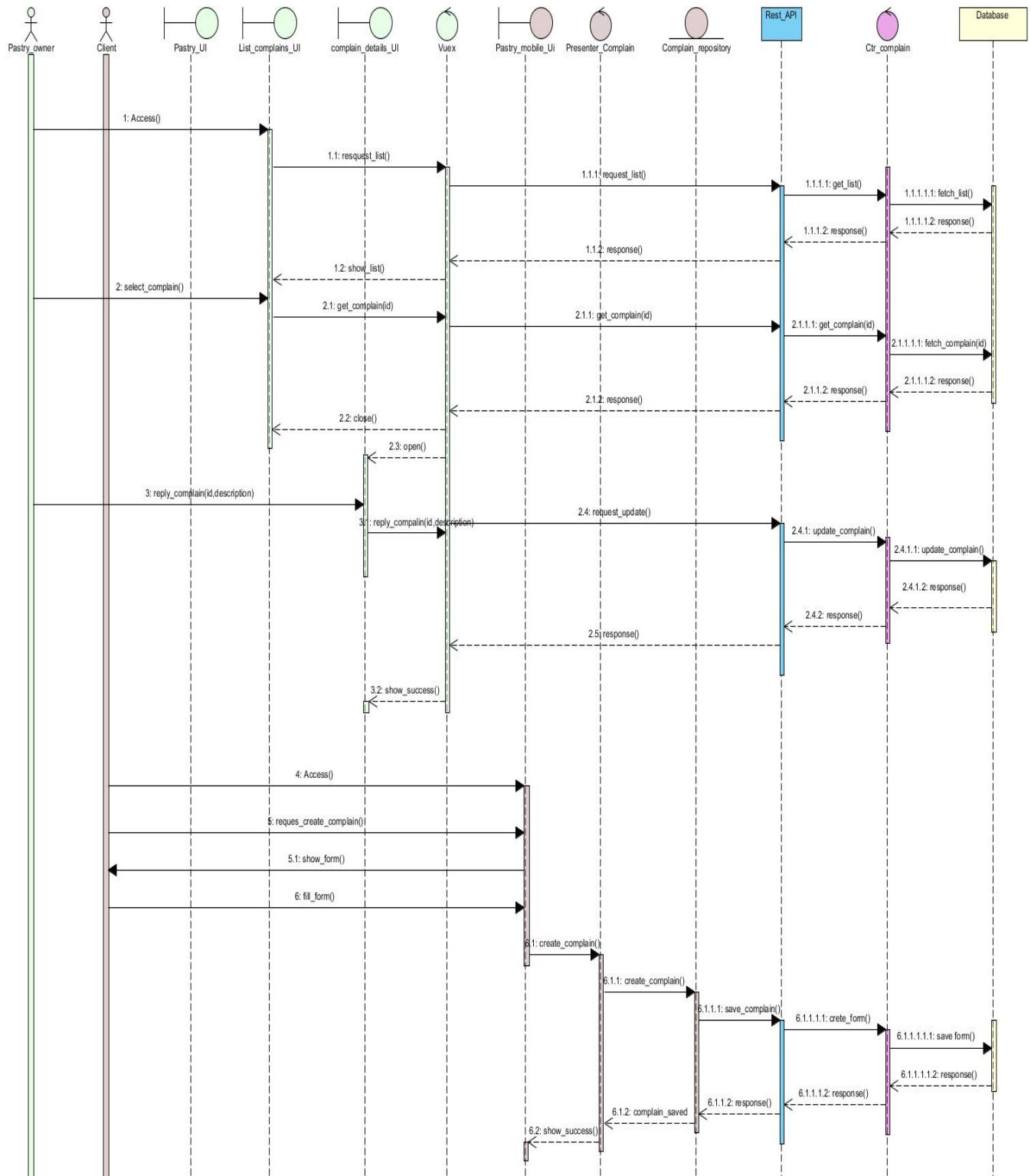


Figure 5.5: Manage complains sequence diagram

### Sequence diagram "Manage reports"

Figure 5.6 is a representation of the needed steps to the third sprint of this release, the actors are all involved in this sprint to either view list of reports for the administrator, or reporting a pastry for the client, or reporting a pastry for the pastry owner.

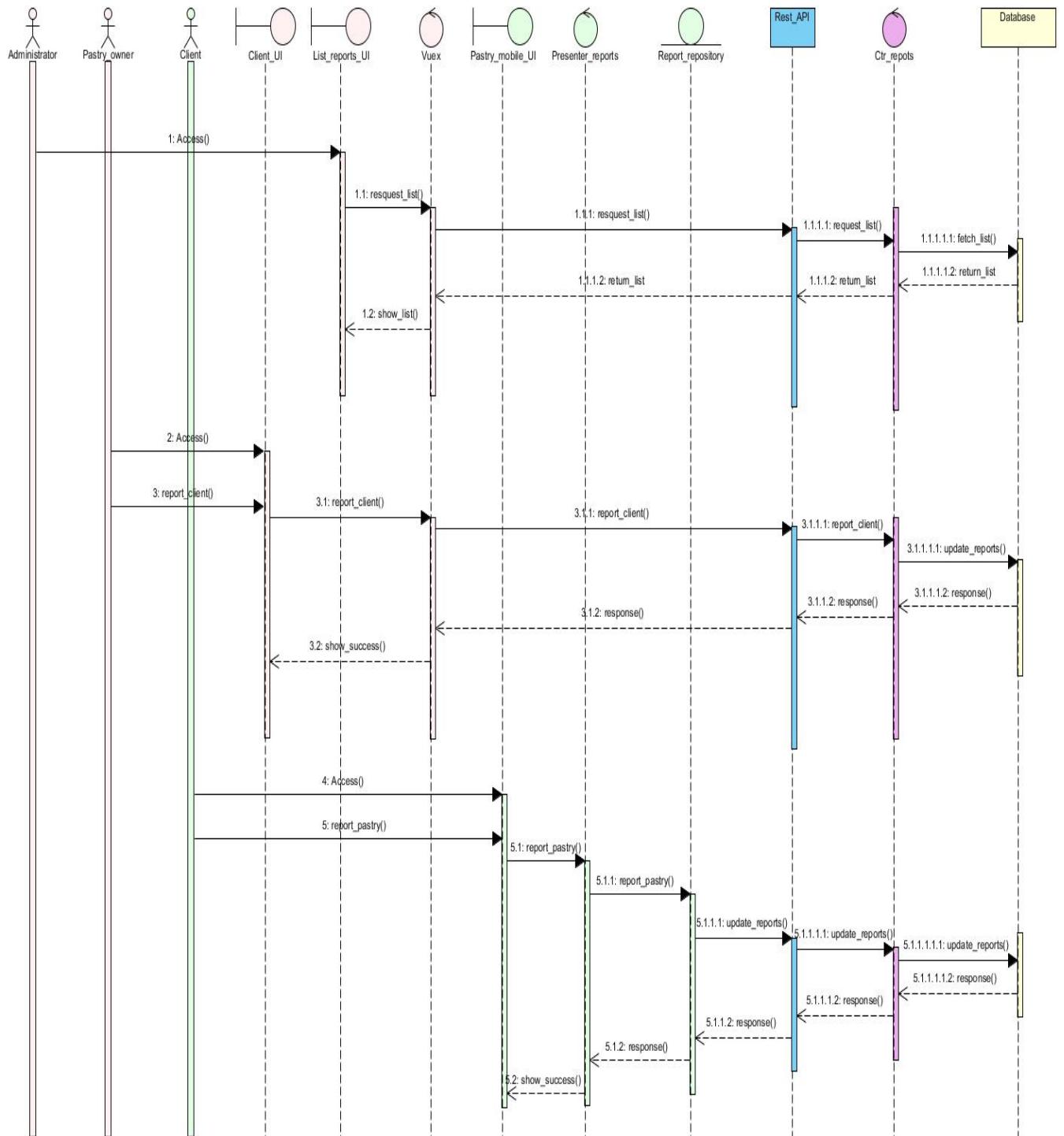


Figure 5.6: Manage reports sequence diagram

## 5.4 Implementation

In this section we will present our interfaces for the four sprints in this release.

### 5.4.1 Web

#### Sprint "Manage clients"

Sprint manage clients has three different tasks where the administrator can view the list of clients as shown in Figure 5.7, via this interface he can also block a client.

The screenshot shows the 'Customers list' page of the CakeitUp web application. The left sidebar contains navigation links: Home, Pastries, Customers (which is selected and highlighted in blue), and Reports. The main content area has a teal header bar with a back arrow icon. Below the header, the title 'Customers list' is displayed. A search bar with a magnifying glass icon is positioned above a table. The table has columns for Name, Creation date, and Orders. It lists two entries:

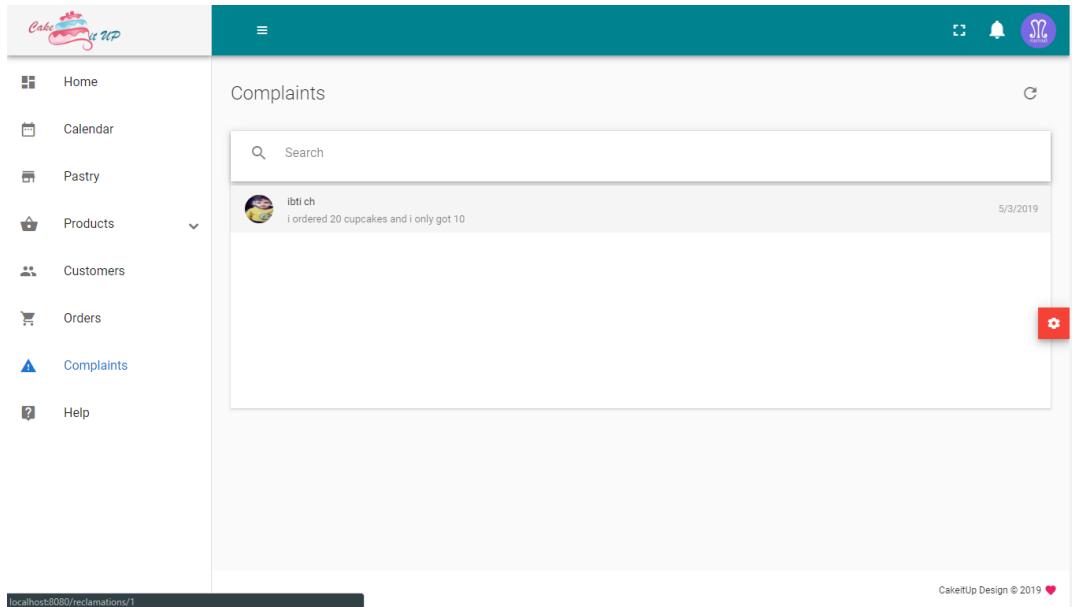
| Name     | Creation date       | Orders |
|----------|---------------------|--------|
| ibtich   | 2019-04-30 10:54:41 | 5      |
| lbtissem | 2019-05-28 11:35:57 | 5      |

At the bottom right of the table, there are icons for 'Rows per page' (set to 5), '1-2 of 2', and a gear icon. The footer of the page includes the text 'CakeitUp Design © 2019 ❤️'.

Figure 5.7: List of clients interface web

## Sprint "Manage complains"

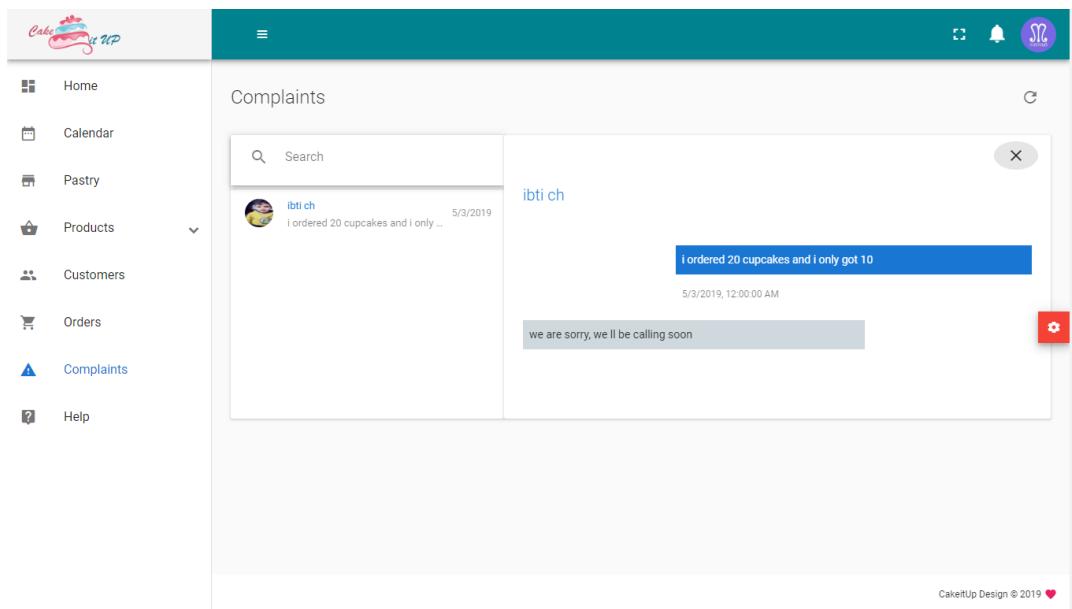
The pastry owner can view the complains that were send by their clients.



The screenshot shows the 'Complaints' section of the CakeitUp web application. On the left, there is a sidebar with navigation links: Home, Calendar, Pastry, Products (with a dropdown menu), Customers, Orders, Complaints (which is currently selected and highlighted in blue), and Help. The main content area is titled 'Complaints' and contains a search bar labeled 'Search'. Below the search bar is a list of complaints. The first complaint is from a user named 'ibti ch' with the message 'i ordered 20 cupcakes and i only got 10' and the timestamp '5/3/2019'. There is a red gear icon at the bottom right of the list area. At the bottom of the page, there is a footer bar with the URL 'localhost:8080/reclamations/1' on the left and 'CakeitUp Design © 2019 ❤️' on the right.

Figure 5.8: List of complains interface web

He can also reply to a complain as shown in Figure 5.9.



This screenshot shows the same 'Complaints' section as Figure 5.8, but with a reply added to the first complaint. The reply is enclosed in a blue box and reads 'i ordered 20 cupcakes and i only got 10'. Below the reply, a grey box contains the message 'we are sorry, we ll be calling soon'. The rest of the interface is identical to Figure 5.8, including the sidebar, search bar, and footer.

Figure 5.9: Reply to complain interface web

## Sprint "Manage reports"

The reports do not really have that much to be done on but they are really important, the administrator is the only user authorized to access the list of reports.

The screenshot shows the 'Reports list' page of the CakeItUp application. On the left, there is a sidebar with icons for Home, Pastries, Customers, and Reports. The 'Reports' icon is selected. The main area has a teal header bar with a search bar and a red 'X' button. Below the header, a card displays a report from 'Chef Pipo' dated 6/9/2019 at 12:00:00 AM. The message reads: 'from pastry:Chef Pipo' and 'Regarding client:ibti ch'. A small red 'REPORT' button is next to the name. A message below says 'Message: i think this client is prankning pastries' with a gear icon. At the bottom left, the URL 'localhost:8080/raports/1' is visible, and at the bottom right, the text 'CakeItUp Design © 2019 ❤️'.

Figure 5.10: List of reports interface web

## Sprint "Manage events"

The events or orders can be view in the calender interface as shown in Figure 5.11.

The screenshot shows the 'Calendar' page of the CakeItUp application. The sidebar includes icons for Home, Calendar (selected), Pastry, Products, Customers, Orders, Complaints, and Help. The main area has a teal header bar with a search bar and a red 'X' button. Below the header, a section titled 'All orders' shows a card for 'ibti ch [Waiting]' on 5/3/2019 with the message 'Lorem ipsum dolor sit amet, con...'. A 'SEE MORE ▶' link is at the bottom. To the right is a calendar for May 2019. A specific event for 'ibti ch' on May 3rd is highlighted with a pink background and a pink border. The event details show 'ibti ch' and a description: 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.' There are also icons for a gift, alarm, info, and checkmark. The URL 'localhost:8080/calender' is at the bottom left, and the text 'CakeItUp Design © 2019 ❤️' is at the bottom right.

Figure 5.11: Calender interface web

To add an event all the user has to do is fill in the form that is shown in Figure 5.12.

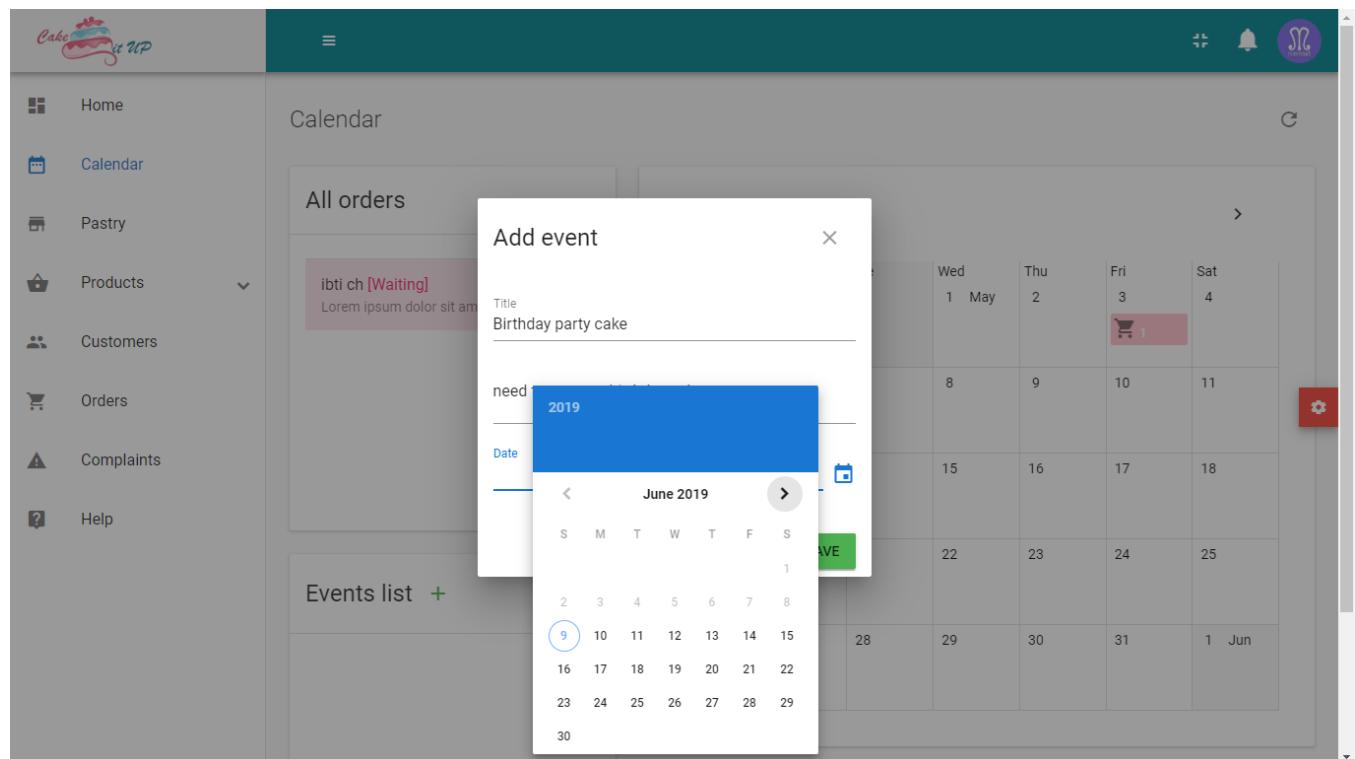


Figure 5.12: Add event interface web

## 5.4.2 Mobile

### Sprint "Manage account"

The client has access to his own list of complain that he has already sent to a number of pastries, The Figure 5.13 gives us an idea about how this layout look like.

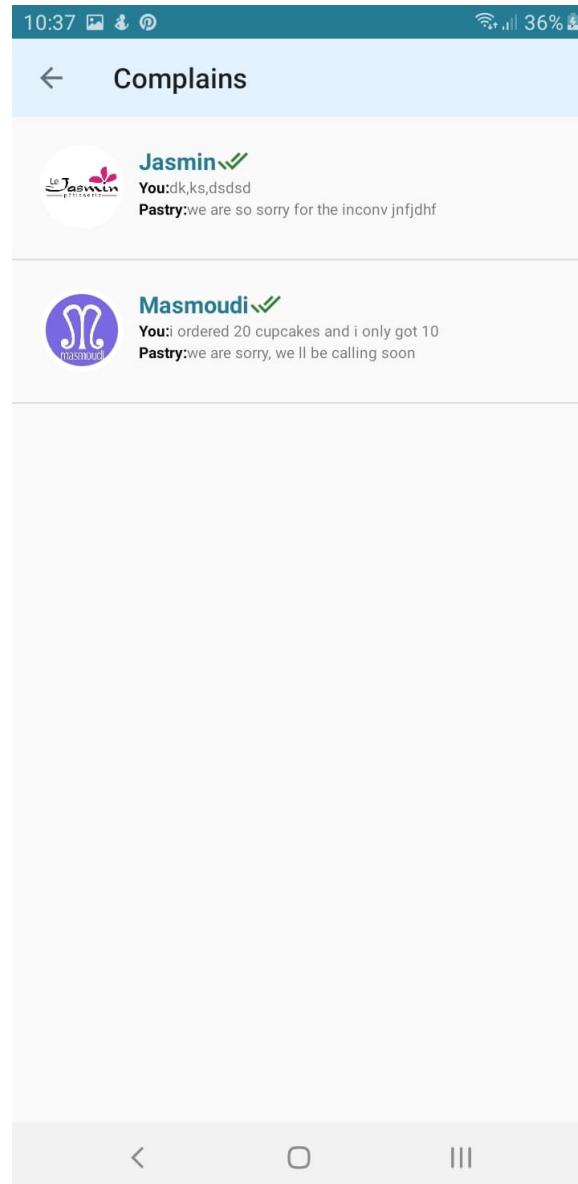


Figure 5.13: List of complains mobile interface

## 5.5 Release retrospective

Finishing this release, we had to test every sprint where we decided on these retrospectives:

- What went well:
  - The tasks of this module went well. There were no problems in this sprint.
- What went wrong:
  - Nothing.
- What was not realized:
  - Nothing.
- What can be improved:
  - the presentation of the interfaces can be improved

## Conclusion

In this chapter, we presented the second sprint result. And like the first sprint we went through specification, design, coding and retrospectives. In the last chapter our effort will be devoted to producing the third and last sprint.

# Chapter 6

## Release 3

### Introduction

We start from the same principle as the previous sprint, but this time we focus on the following use cases: "Manage reviews", "Manage rating", "Manage cart", "Manage orders" and "Manage personal orders".

Our goal is to have, at the end of this sprint, a version of the application that is stable.

### 6.1 Backlog

Table 6.1: Backlog: Release 3

| Sprint                         | ID<br>U.S | User story                          | ID<br>task | Task   |
|--------------------------------|-----------|-------------------------------------|------------|--|
| Sprint 9<br><br>Manage reviews | 9.1       | As a client i want to add a review. | 9.1.A      | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|                                |           |                                     | 9.1.B      | Develop the add review model.  |
|                                |           |                                     | 9.1.C      | Test the add review model.   |

|   |      |  |        |  |
|---|------|--|--------|--|
|   | 9.2  | As a pastry owner i want to view my pastry's review.             | 9.2.A  | Make the diagrams needed for this U.S.   |
|   |      |  | 9.2.B  | Develop the list of reviews model.   |
|   |      |  | 9.2.C  | Develop the list of reviews model.   |
| <b>Sprint 10</b><br><b>Manage ratings</b> | 10.1 | As a client i want to rate a pastry.                             | 10.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |      |  | 10.1.B | Develop the rate pastry model.   |
|   |      |  | 10.1.C | Test the rate pastry model.  |
|   | 10.2 | As a pastry owner i want to view my pastry's rating.             | 10.2.A | Make the diagrams needed for this U.S.   |
|   |      |  | 10.2.B | Develop the view rating model.   |
|   |      |  | 10.2.C | Test the view rating model.  |
| <b>Sprint 11</b><br><b>Statistics</b>     | 11.1 | As an administrator or a pastry owner i want to view statistics. | 11.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |      |  | 11.1.B | Develop the statistics model.  |
|   |      |  | 11.1.C | Test the statistics model.   |
|   | 12.1 | As a client i want to edit my own cart.                          | 12.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |      |  | 12.1.B | Develop the edit cart model.   |
|   |      |  | 12.1.C | Test the edit cart model.  |
| <b>Sprint 12</b><br><b>Manage cart</b>    | 12.2 | As a client i want to view my cart.                              | 12.2.A | Make the diagrams needed for this U.S.   |
|   |      |  | 12.2.B | Develop the view cart model.   |
|   |      |  | 12.2.C | Test the view cart model.  |

|   |      |   |        |  |
|---|------|---|--------|--|
| <b>Sprint 13</b><br><b>Manage orders</b>          | 13.1 | As a client i want to pass an order.                      | 13.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |      |   | 13.1.B | Develop the pass order model.  |
|   |      |   | 13.1.C | Test the pass order model.   |
|   | 13.2 | As a client i want to view my own orders.                 | 13.2.A | Make the diagrams needed for this U.S.   |
|   |      |   | 13.2.B | Develop the view orders model.   |
|   |      |   | 13.2.C | Test the view orders model.  |
|   | 13.3 | As a pastry owner i want to view orders.                  | 13.3.A | Make the diagrams needed for this U.S.   |
|   |      |   | 13.3.B | Develop the view orders model.   |
|   |      |   | 13.3.C | Test the view orders model.  |
|   | 13.4 | As a pastry owner i want to change the state of an order. | 13.4.A | Make the diagrams needed for this U.S.   |
|   |      |   | 13.4.B | Develop the change state of order model.   |
|   |      |   | 13.4.C | Test the change state of order model.  |
| <b>Sprint 14</b><br><b>Manage personal orders</b> | 14.1 | As a client i want to add a personal order.               | 14.1.A | Make the diagrams needed for this U.S (use case diagram, class diagram, sequence diagram). |
|   |      |   | 14.1.B | Develop the add personal order model.  |
|   |      |   | 14.1.C | Test the add personal order model.   |
|   | 14.2 | As a client i want to view my list of personal orders.    | 14.2.A | Make the diagrams needed for this U.S.   |
|   |      |   | 14.2.B | Develop the view list of personal orders model.  |
|   |      |   | 14.2.C | Test the list of personal orders model.  |

## 6.2 Use case

### 6.2.1 Use case diagrams

We are dedicating this section to present the use case diagrams for this release.

Starting by the first sprint of this release **Manage reviews** which is illustrated in Figure 6.2.

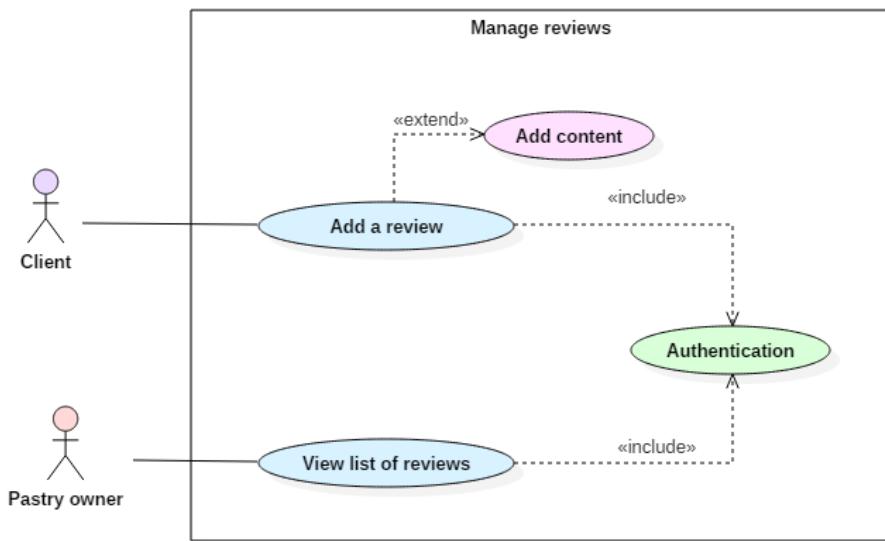


Figure 6.1: Manage reviews use case

The next use case diagram to be presented is **Manage ratings** shown in Figure 6.2.

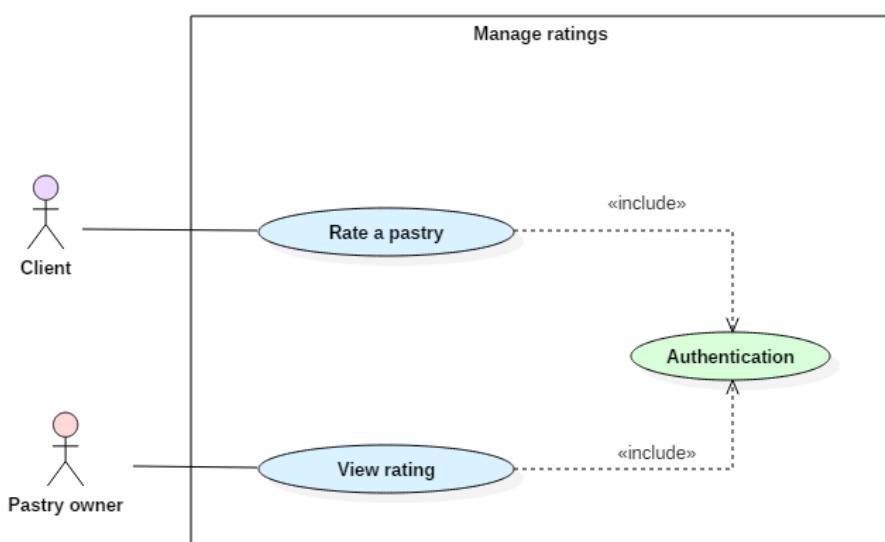


Figure 6.2: Manage ratings use case

Figure 6.3 presents the use case for the sprint **Statistics**.

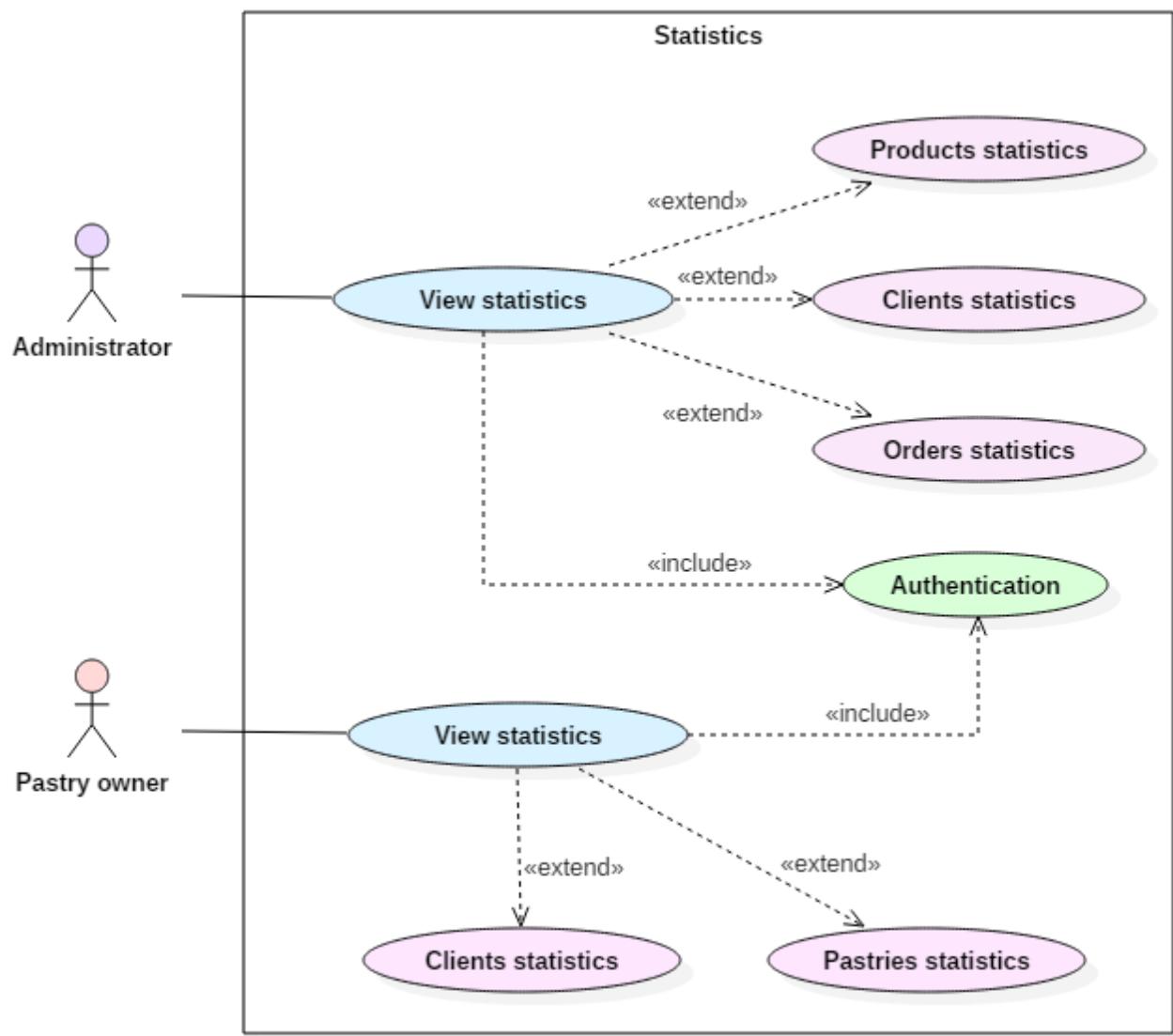


Figure 6.3: Statistics use case

As for the sprint **Manage cart** we created the use case presented in Figure 6.4

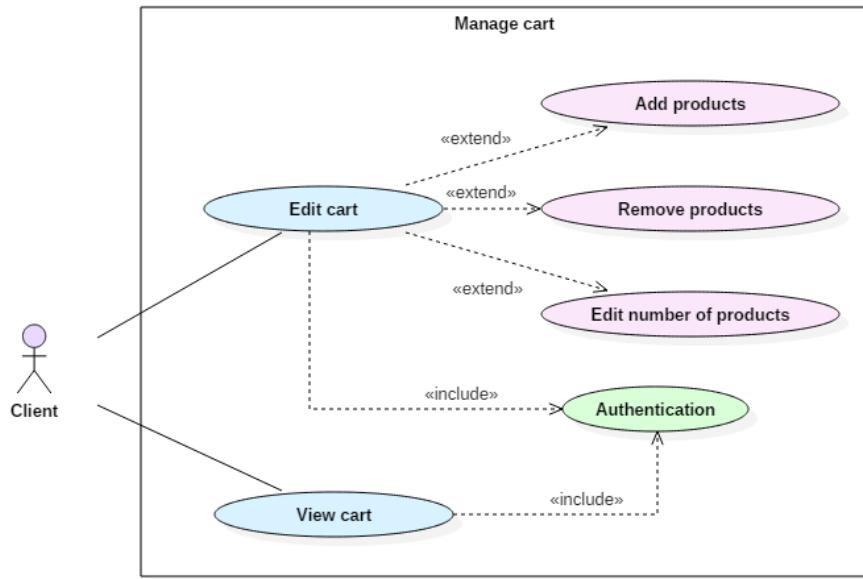


Figure 6.4: Manage cart use case

Figure 6.5 represents the use case of sprint **Manage orders**.

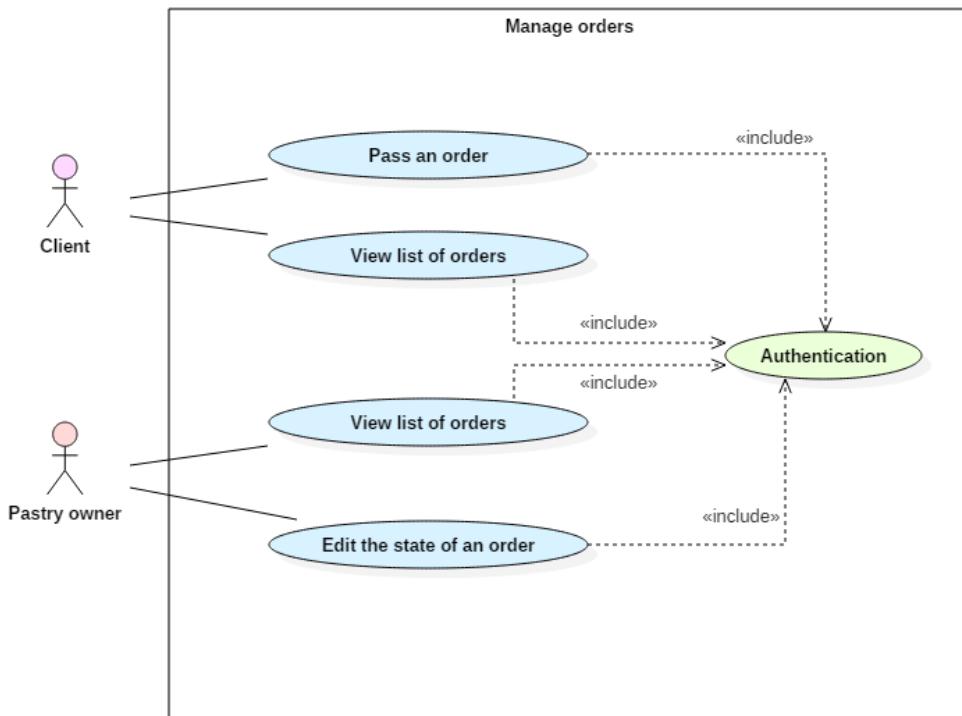


Figure 6.5: Manage orders use case

Finally the sprint cart **Manage personal orders** is shown in Figure 6.6.

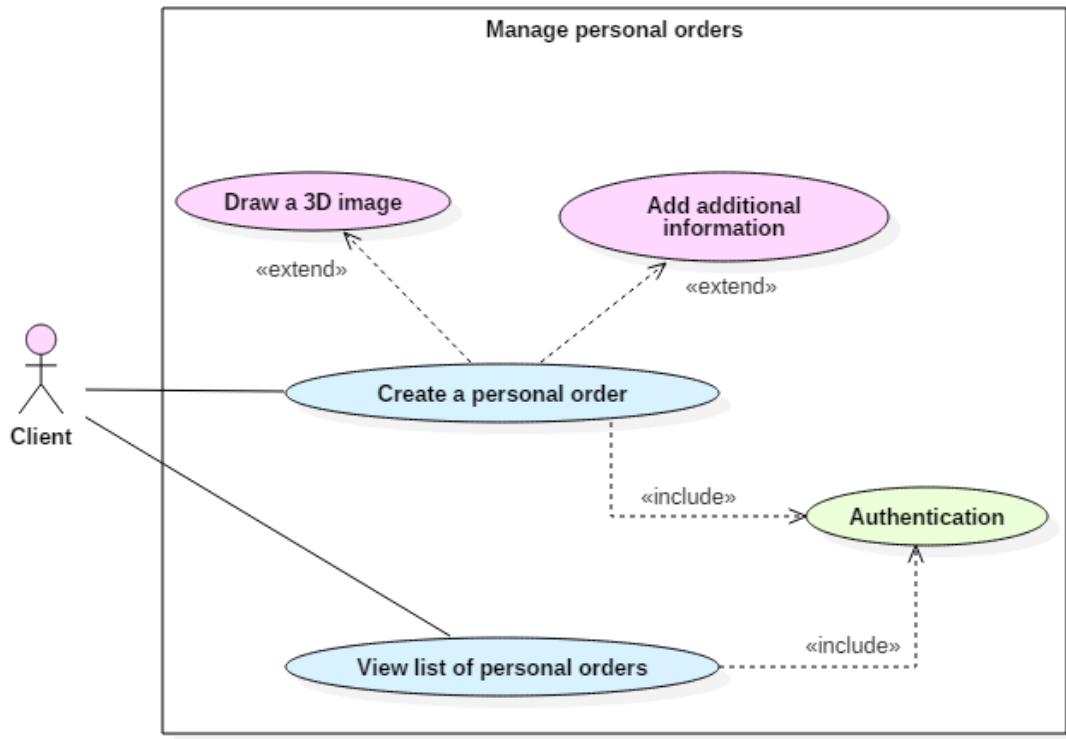


Figure 6.6: Manage personal orders use case

## 6.2.2 Use case description

Same as the previous release we now move on to describing the use case diagrams.

### Use case "Manage reviews" description

Table 6.2: Use case "Manage reviews" description

| Use case "Manage reviews" |   |
|---------------------------|---|
| <b>Actors</b>             | Client and pastry owner   |
| <b>Precondition</b>       | The user is authenticated and authorized                                      |
| <b>Postcondition</b>      | A review is added   |
| <b>Main scenario</b>      | <ul style="list-style-type: none"> <li>• The client adds a review,</li> </ul> |

|                             |   |
|-----------------------------|---|
|                             | <ul style="list-style-type: none"> <li>• The system will save the changes.</li> <li>• The pastry owner view the reviews.</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.  |

### Use case "Manage ratings" description

Table 6.3: Use case "Manage ratings" description

| Use case "Manage ratings"   |  |
|-----------------------------|--|
| <b>Actors</b>               | Client, pastry owner   |
| <b>Precondition</b>         | The user is authenticated and authorized   |
| <b>Postcondition</b>        | The pastry is rated  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• The client can rate a pastry</li> <li>• The system will save the changes.</li> <li>• The pastry owner will receive a notification about the rating</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

### Use case "Statistics" description

Table 6.4: Use case "Statistics" description

| Use case "Statistics"       |  |
|-----------------------------|--|
| <b>Actors</b>               | Administrator, pastry owner  |
| <b>Precondition</b>         | The user is authenticated and authorized   |
| <b>Postcondition</b>        | The statistics are viewed  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• both users view their own statistics</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.                               |

### Use case "Manage cart" description

Table 6.5: Use case "Manage cart" description

| Use case "Manage cart"      |  |
|-----------------------------|--|
| <b>Actors</b>               | Client   |
| <b>Precondition</b>         | The client is authenticated and authorized   |
| <b>Postcondition</b>        | The cart is managed  |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• view cart</li> <li>• Edit cart</li> <li>• The system saves the changes</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

### Use case "Manage orders" description

Table 6.6: Use case "Manage orders" description

| Use case "Manage orders"    |  |
|-----------------------------|--|
| <b>Actors</b>               | Client and pastry owner  |
| <b>Precondition</b>         | The user is authenticated and authorized   |
| <b>Postcondition</b>        | The order is added or viewed   |
| <b>Main scenario</b>        | <ul style="list-style-type: none"> <li>• Pass an order</li> <li>• The system saves the changes</li> <li>• view list of orders</li> <li>• edit state of an order</li> <li>• The system saves the changes</li> </ul> |
| <b>Alternative scenario</b> | In case of an error the system will show an error message.   |

### Use case "Manage personal orders" description

Table 6.7: Use case "Manage personal orders" description

| Use case "Manage personal orders" |   |
|-----------------------------------|---|
| <b>Actors</b>                     | Client  |
| <b>Precondition</b>               | The user is authenticated and authorized  |
| <b>Postcondition</b>              | The personal order is created or viewed   |
| <b>Main scenario</b>              | <ul style="list-style-type: none"> <li>• Create a personal order</li> <li>• The system saves the changes</li> <li>• view list of personal orders</li> </ul> |
| <b>Alternative scenario</b>       | In case of an error the system will show an error message.  |

## 6.3 Sequence diagram

### Sequence diagram "Manage cart"

Figure 6.7 is a representation of the needed steps to the sprint "**Manage cart**" where we only have the client as an actor, he have multiple choices and tasks that can be done such as view the cart's details that includes the list of products exists in the cart currently and the price of the whole order, in order to access the cart, the cart presented will ask for the cart's information from the repository and the repository will send an API call to the server asking for data, once the data is received they are send back to the repository and back to the presented , the presenter will just show them to the user and so on for all the other tasks.

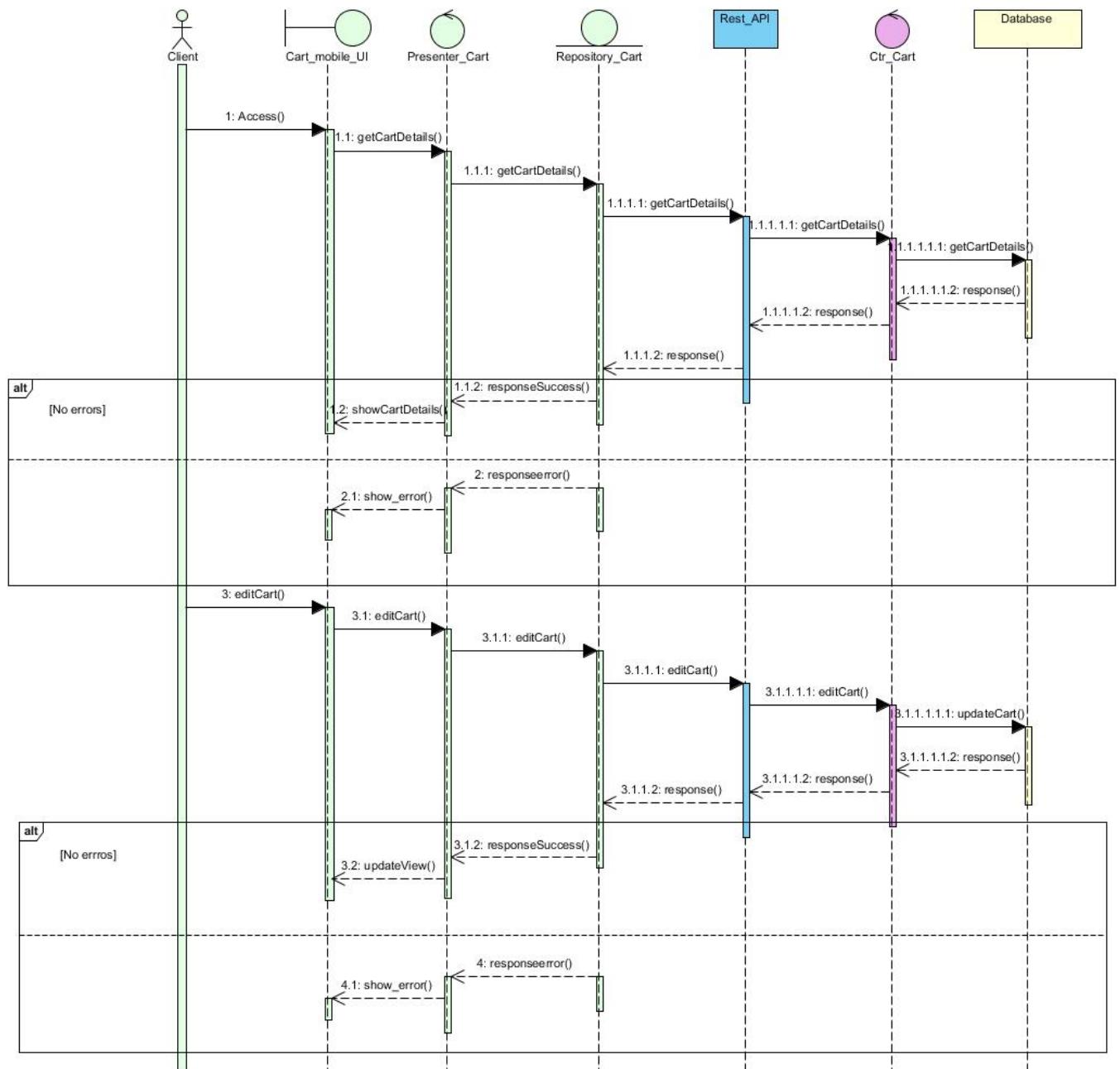


Figure 6.7: Manage cart sequence diagram

### Sequence diagram "Manage orders"

Sprint 13 is **Manage orders** where the client and the pastry owner are the main actors, the client is the one who creates the order and sends it to the pastry owner. As showing in Figure 6.8 the interaction between the web and mobile clients are not the same, the client use MVP pattern while the pastry owner uses MVVM pattern and for the back-end it uses the Mvc pattern.

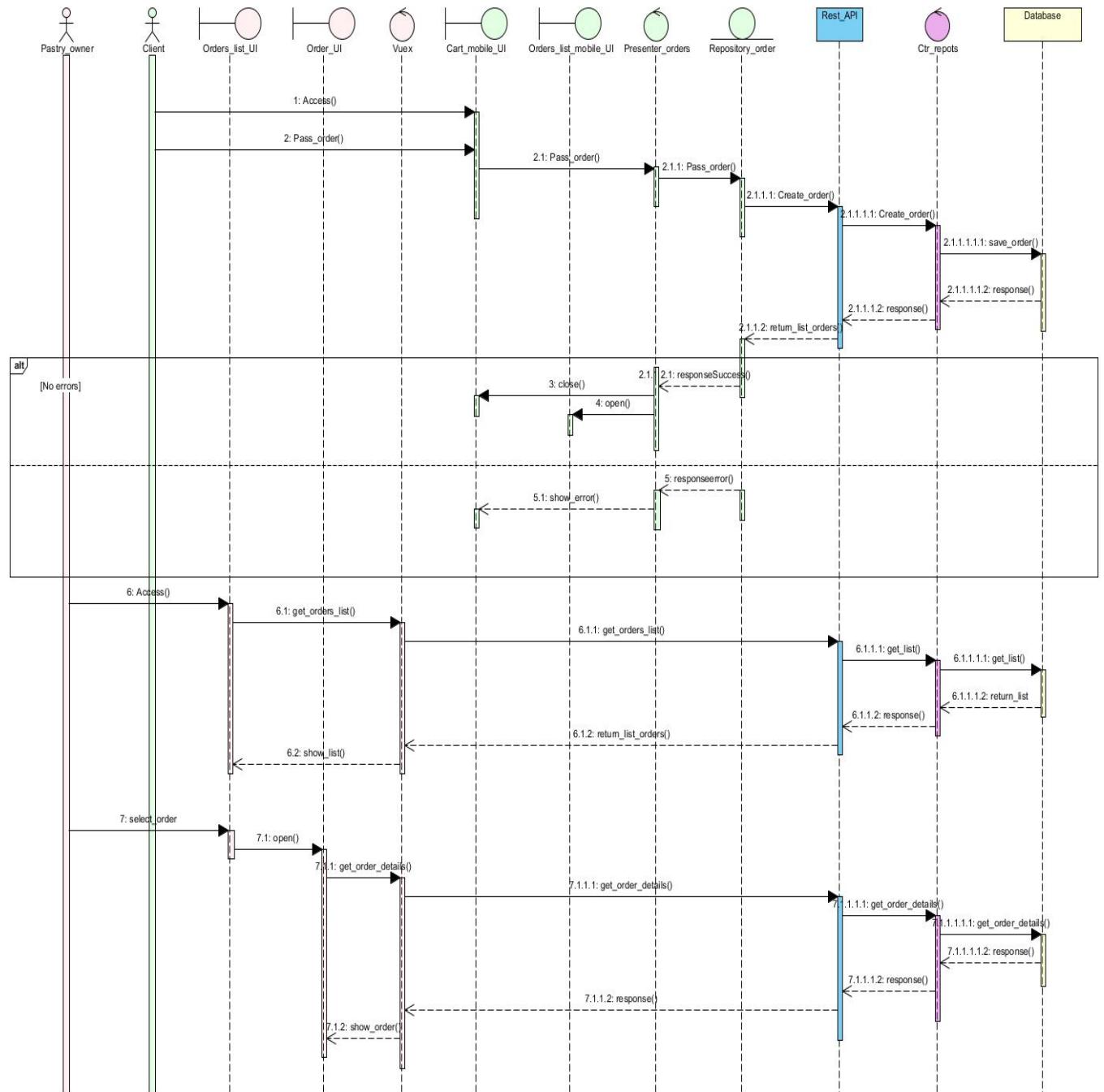


Figure 6.8: Manage orders sequence diagram

## 6.4 Implementation

### 6.4.1 Web

#### Sprint "Manage reviews"

The list of reviews giving by the clients can be seen in the pastry section.

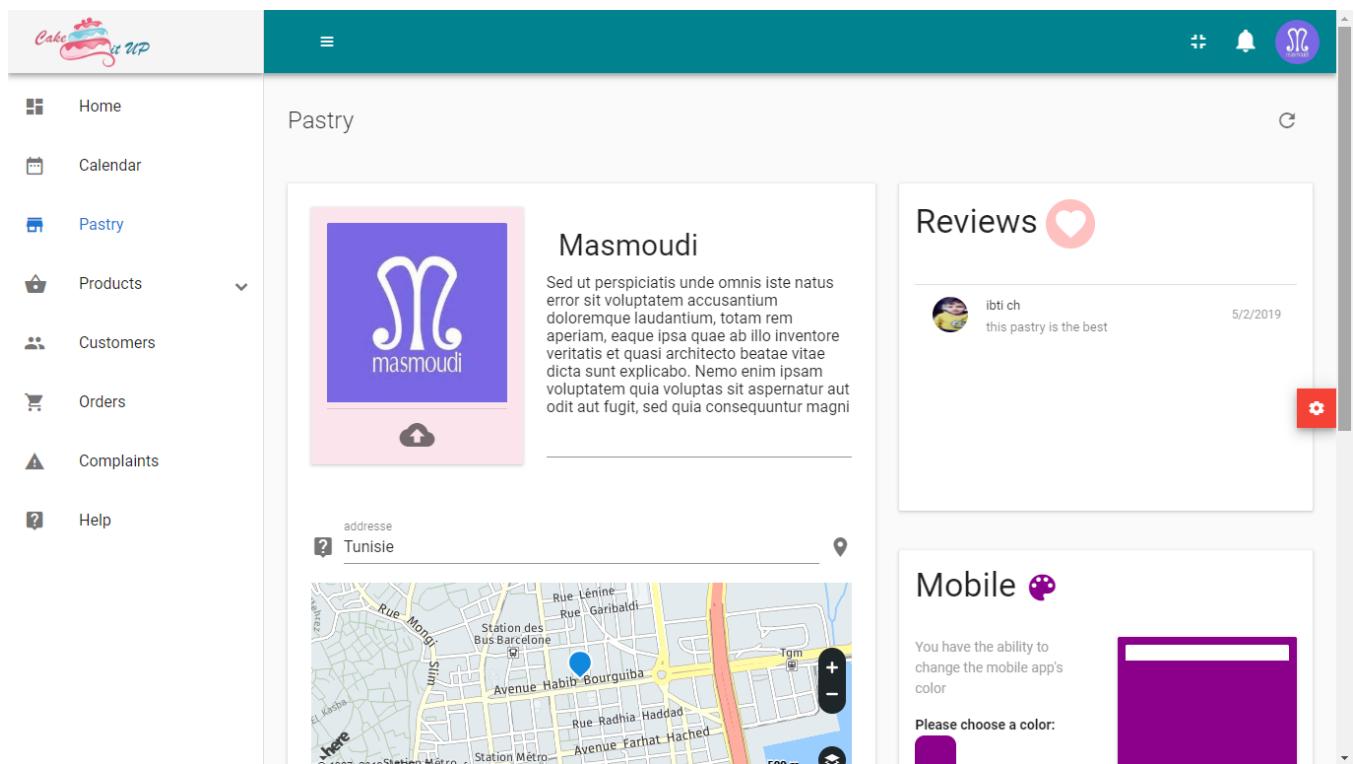


Figure 6.9: List of reviews Interface web

## Sprint "Manage orders"

The pastry owner can view his own orders as shown in Figure 6.10

|  | All orders | Waiting | Today | Late | Done | Canceled |
|--|------------|---------|-------|------|------|----------|
|  | 1          | 1       | 0     | 0    | 0    | 0        |
|  | ibti ch    |         |       |      |      |          |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt... |            |         |       |      |      |          |
| 5/3/2019   |            |         |       |      |      |          |

localhost:8080/commandes#done

Figure 6.10: List of orders Interface web

The status of an order can be change by the pastry owner but only after a confirmation as shown in Figure 6.11.

| ref   | Quantity | reduction | Category | prix totale |
|-------|----------|-----------|----------|-------------|
| #ref1 | 2        | 0%        | popcake  | 500         |
| #ref2 | 1        | 0%        | Gâteau   | 50          |
|       |          |           |          | 555         |

Are you sure you want to cancel this order?

YES CANCEL

localhost:8080/commandes#done

Figure 6.11: Edit order Interface web

## 6.4.2 Mobile

### Sprint "Manage orders"

The client has access to his own list of orders. The Figure 6.12 gives us an idea about how this layout look like.

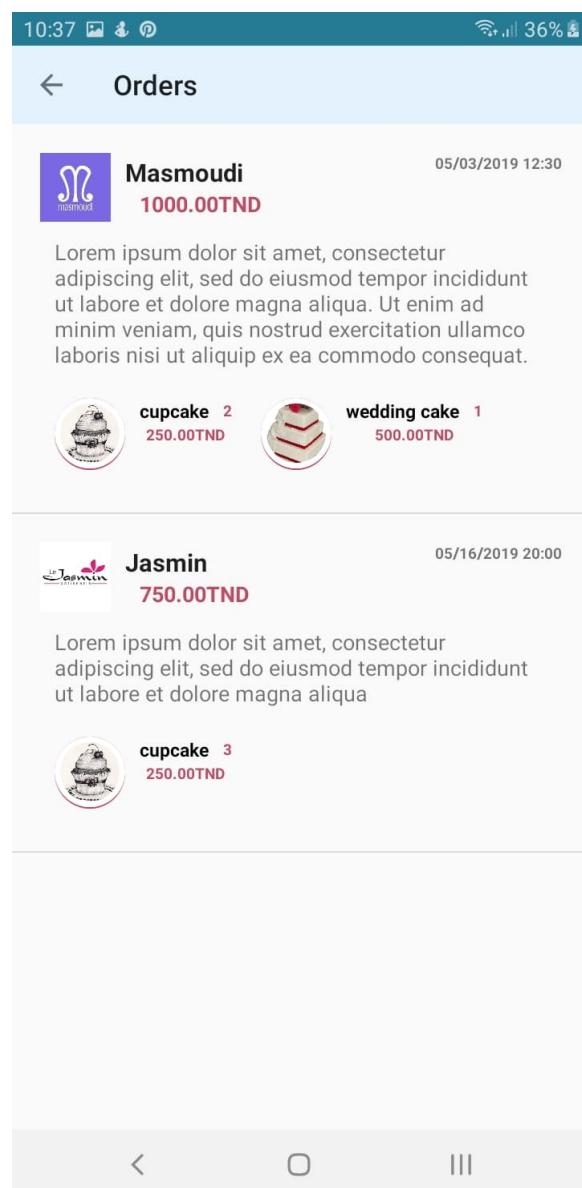


Figure 6.12: List of orders mobile interface

## 6.5 Release retrospective

Finishing this release, we had to test every sprint where we decided on these retrospectives:

- What went well:
  - The tasks of this module went well except the 3D modeling of personal order.
- What went wrong:
  - Nothing.
- What was not realized:
  - 3D personal order.
- What can be improved:
  - the presentation of the interfaces can be improved

## Conclusion

During this chapter, we developed the last release and the last phase of the system, by the passage on the sprint backlog, the design and the realization of each sprint. The next step is to close this report with a general conclusion followed by future perspectives.

# General conclusion

This project is a precious opportunity to get out of the theoretical framework and apply the knowledge acquired during university studies in a professional environment. This work, conducted within “**Anypli**” allowed us to integrate into the professional field and learn several attitudes especially the field of software development.

Teamwork and the concept of collaboration within “**Anypli**” has facilitated our integration and recognition of work procedures. The relationship between trainees and engineers has been very rewarding and user-friendly. The climate of trust and respect with the superiors was indeed favorable to achieve better productivity. Each step of our project required an effort and a thorough research.

On a technical level, this project introduced us to new concepts and new technologies. We approached the web and mobile field to develop an application that must meet the expectations of customers.

The realization of this project also allowed me to confront several constraints. At the same time time constraints, quality constraints or even constraints of technology. In addition, this internship allowed me to provide new knowledge both in the field of computer science and, in human terms, it is thanks to this work that I discover new technologies such as “**Laravel**”, “**VueJs**” and “**Android**”.

Finally, our work does not stop at this level. In fact, several perspectives are available To this project. Among the features we can consider for “**Cake it up !**”:

- 3D design
- AI
- On-line payment

# Bibliography

- [1] M. Mohsin, “10 social media statistics you need to know in 2019 [infographic],” *Oberlo*, 2019.
- [2] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.
- [3] M. Fowler, *UML Distilled*. Addison-Wesley, 2014.
- [4] A. Kumar, *Sencha MVC architecture*. Packt Publishing Ltd, 2012.
- [5] J. Kouraklis, *MVVM in Delphi*. APPRESS, 2016.
- [6] S. I. Dmitry Jemerov, *Kotlin in Action*. Manning Publications Company, 2017.
- [7] D. Powers, *PHP Solutions: Dynamic Web Design Made Easy*. Apress company, 2011.
- [8] M. Stauffer, *Laravel: Up and Running: A Framework for Building Modern PHP Apps*. O’reilly books, 2017.
- [9] K. M. Alex Kyriakidis, *The Majesty of Vue.js*. Packt publishing pvt. ltd, 2016.
- [10] C. Pietschmann, “Swagger is now the openapi specification,” *Buildazure*, 2017.
- [11] D. J. H. David F. Griffiths, *Learning LaTeX: Second Edition*. The society for industrial and applied mathematics, 2016.

# **Abstract**

This work, carried out at “EPI” school lies within the scope of the graduation project.

The purpose of this project is to develop a Web/mobile application at “Anypli”.

“CakeItUp” is an application dedicated to all the users who can not get enough of pastries and always looking for the best bakery, it is also dedicated to the pastry owners out there who wants to manage their products, clients, orders online.

**Keywords:** Laravel, VueJS, Vuex, Vuetify, Swagger, Rest api, Kotlin, Android, MVC, MVP, MVVM.

---

# **Résumé**

Ce travail, réalisé à l’école "EPI", entre dans le cadre du projet de remise des diplômes.

Le but de ce projet est de développer une application Web/mobile chez "Anypli".

“CakeItUp” est une application dédiée à tous les utilisateurs qui ne peuvent pas obtenir assez de pâtisseries et qui cherche toujours la meilleure boulangerie. Il est également dédié aux propriétaires de pâtisseries qui souhaitent gérer leurs produits, leurs clients, leurs commandes en ligne.

**Mots-clés :** Laravel, VueJS, Vuex, Vuetify, Swagger, Rest api, Kotlin, Android, MVC, MVP, MVVM.