

# Design and implementation of a virtual stylist as a software application



Víctor Juez Cañellas

Supervisor: Claudia P. Ayala Martínez

Department of Service and Information System Engineering

Barcelona School of Informatics

Polytechnic University of Catalonia – BarcelonaTech

January 2020

# Resum

Aquest projecte es basa en el desenvolupament d'un prototip per una futura aplicació anomenada Mixties.

Mixties pretén ser un marketplace<sup>1</sup> de roba amb un estilista virtual integrat el qual, segons les característiques i preferències de l'usuari, aquest li suggereix outfits<sup>2</sup> personalitzats.

Lluny d'aquest objectiu, el propòsit d'aquest prototip és crear un formulari interactiu a través del qual puguem obtenir informació rellevant sobre els usuaris que possiblement estiguin interessats en la nostra aplicació. Això ens permetrà validar el mercat i poc a poc anar definint el nostre model de negoci.

Concretament, el sistema que es desenvolupa es pot dividir en dues parts. Per un costat, tenim una aplicació pels administradors, en aquest cas som nosaltres com a membres de Mixties. Mitjançant aquesta aplicació podem entrar roba i crear outfits en el sistema, així com analitzar les dades rebudes del formulari interactiu. Per altra banda, tenim el formulari interactiu, pel que mostrem a l'usuari cinc outfits dels que haguem creat nosaltres prèviament per a que ell o ella els valori. Amb les seves valoracions i un conjunt de preguntes afegides que se li fan, podem definir el seu perfil per tal d'analitzar-lo posteriorment.

---

<sup>1</sup> Plataforma online creada per una empresa que actua com un tercer neutral per posar en contacte compradors i venedors.

<sup>2</sup> Conjunts de roba

# Resumen

Este proyecto se basa en el desarrollo de un prototipo para una futura aplicación llamada Mixties.

Mixties pretende ser un marketplace<sup>3</sup> de ropa con un estilista virtual integrado el cual, según las características y preferencias del usuario, este le sugiere outfits<sup>4</sup> personalizados.

Lejos de este objetivo, el propósito de este prototipo es crear un formulario interactivo a través del cual podamos obtener información relevante sobre los usuarios que posiblemente estén interesados en nuestra aplicación. Esto nos permitirá validar el mercado y poco a poco ir definiendo nuestro modelo de negocio.

Concretamente, el sistema que se desarrolla se puede dividir en dos partes. Por un lado, tenemos una aplicación para los administradores, en este caso somos nosotros como miembros de Mixties. Mediante esta aplicación podemos entrar ropa y crear outfits en el sistema, así como analizar los datos recibidos del formulario interactivo. Por otro lado, tenemos el formulario interactivo, por el que mostramos al usuario cinco outfits de los que hayamos creado nosotros previamente para que él o ella los valore. Con sus valoraciones y un conjunto de preguntas añadidas que se le hacen, podemos definir su perfil para analizarlo posteriormente.

---

<sup>3</sup> Plataforma online creada por una empresa que actúa como un tercero neutral para poner en contacto a compradores y vendedores.

<sup>4</sup> Conjunto de ropa.

# Abstract

This project is based on the development of a prototype for a future application called Mixties.

Mixties intends to be a clothing marketplace with an integrated virtual stylist which, according to the user's characteristics and their preferences, it suggests custom outfits.

Far from this goal, the purpose of this prototype is to create an interactive form by which we can obtain relevant information about users who may be interested in our application. This will allow us to validate the market and gradually define our business model.

Specifically, the implemented system can be divided into two parts. On the one hand, we have an application for the administrators, in this case, ourselves as Mixties members. Using this application, we can insert clothes and create outfits in the system, as well as analyze the data received from the interactive form. On the other hand, we have the interactive form, where we show the user five outfits that we have previously created so that they can rate them. With their respective rates and a set of additional questions that we ask them, we can define their profile to analyze it later.

# Table of Contents

<b>1. Context and Scope.....</b>	<b>7</b>
1.1 Context.....	7
1.1.1 Why are we here.....	7
1.1.2 What is Mixties?.....	7
1.1.3 Would people be interested in this application? .....	9
1.1.4 Who are we? .....	10
1.2 State of the art.....	10
1.2.1 Lookiero .....	11
1.2.2 TryTuesday.....	13
1.2.3 Carol Davidson .....	16
1.2.4 Viume .....	17
1.2.5 Conclusions .....	19
1.3 Scope.....	19
1.3.1 Current situation .....	19
1.3.2 Mixties Minimum Value Product .....	20
1.4 Methodology.....	23
1.4.1 Monitoring Tools.....	24
1.5 Obstacles and Risks.....	24
1.5.1 Obstacles.....	25
1.5.2 Risks.....	25
<b>2. Time Planning.....</b>	<b>28</b>
2.1 Description of tasks.....	28
2.1.1 Agile Project Inception .....	28
2.1.2 Agile Project Planning .....	29
2.1.3 Agile Project Execution .....	29
2.1.4 Resources.....	30
2.2 Estimates and Gantt .....	31
2.2.1 Time planning summary .....	31
2.2.2 Gantt Chart.....	32
2.3 Risk Management: Identification and alternative plan.....	32
<b>3. Budget and Sustainability .....</b>	<b>33</b>
3.1 Cost Estimates.....	33
3.1.1 Direct Costs .....	33
3.1.2 Indirect Costs .....	35
3.1.3 Contingency and Incidentals.....	37
3.1.4 Summary .....	38
3.1.5 Management Control.....	38
3.2 Sustainability.....	39
3.2.1 Self-assessment.....	39
3.2.2 Economic Dimension.....	39
3.2.3 Environmental Dimension .....	40
3.2.4 Social Dimension .....	40
<b>4. Requirements Specification.....</b>	<b>41</b>

4.1	<i>Stakeholders</i> .....	41
4.2	<i>Use Cases Diagram</i> .....	42
4.3	<i>Functional requirements</i> .....	43
4.3.1	Mixties Prototype App.....	43
4.3.2	Admin Dashboard .....	45
4.4	<i>Non-functional requirements</i> .....	53
4.5	<i>Mixties Prototype App sequence diagram</i> .....	56
4.6	<i>Domain class diagram</i> .....	57
4.6.1	Textual restrictions .....	58
4.6.2	Entities description .....	59
<b>5.</b>	<b>Design</b> .....	<b>64</b>
5.1	<i>General Vision</i> .....	64
5.2	<i>Graphical User Interface (GUI)</i> .....	65
5.2.1	Mixties Prototype App (GUI).....	65
5.2.2	Admin Dashboard (GUI) .....	67
5.3	<i>Frontend</i> .....	70
5.4	<i>Backend</i> .....	72
5.4.1	Middleware pattern .....	73
5.4.2	Authentication .....	74
5.5	<i>Database logical schema</i> .....	75
5.6	<i>Infrastructure design and technologies</i> .....	76
5.6.1	Technologies: which ones and why .....	78
<b>6.</b>	<b>Implementation</b> .....	<b>80</b>
6.1	<i>Frontend</i> .....	80
6.1.1	Template .....	80
6.1.2	Component .....	81
6.1.3	Service .....	82
6.2	<i>Backend</i> .....	83
6.2.1	Routes .....	83
6.2.2	Controller .....	84
6.2.3	Model .....	85
<b>7.</b>	<b>Testing</b> .....	<b>87</b>
7.1	<i>Frontend Testing</i> .....	87
7.1.1	Functionalities' Testing .....	87
7.1.2	Usability Testing .....	87
7.2	<i>Backend Testing</i> .....	88
<b>8.</b>	<b>Law considerations</b> .....	<b>89</b>
8.1	<i>General Data Protection Regulation (GDPR)</i> .....	89
8.2	<i>Our approach to comply with the law</i> .....	90
	<b>Close out</b> .....	<b>91</b>
8.3	<i>Software Engineering Approach</i> .....	91
8.3.1	Technical Competences .....	91
8.3.2	Knowledge applied.....	92

8.4	<i>Planning deviation</i> .....	93
8.4.1	Final Time Planning .....	94
8.4.2	Final Gantt Diagram .....	94
8.5	<i>Cost Deviation</i> .....	95
8.6	<i>Conclusions</i> .....	96
8.7	<i>Future work</i> .....	96
<b>9.</b>	<b>Bibliography</b> .....	<b>97</b>

# 1. Context and Scope

In this section, we are going to put the project in context to understand its purpose. Then, we provide a state-of-the-art analysis including our differentiation among the alternatives, and finally, the scope of this project.

## 1.1 Context

Here we are going to describe what is meant to be Mixties right now to have a general vision. It's important to understand that here we are explaining our long-term goals, our vision as a startup at this moment. This is not the thesis' goal.

### 1.1.1 Why are we here

In recent times, thanks to the possibilities that IT is bringing, the world of fashion is adopting new ways to take advantage of it.

One of the traditional styling services used from a long time ago is the personal stylist. A specialist that given your body characteristics and your personal taste, gives you advices on what clothes will suit you best. We are here to reinvent this service.

Knowing this situation of transformation, we want to leverage from it reinventing the traditional concept of personal stylist service, known by its exclusivity and expensive service only within the reach of a few, and making it free, immediate, intuitive and quick. This is the purpose why Mixties was born in the first place.

### 1.1.2 What is Mixties?



Figure 1.1 Mixties brand logo

*“For young girls who want to get quick and free advice on how to dress better and what suits them best, Mixties is virtual stylist application. Unlike physical or online personal stylists, our product offers a free and immediate virtual stylist service.”*

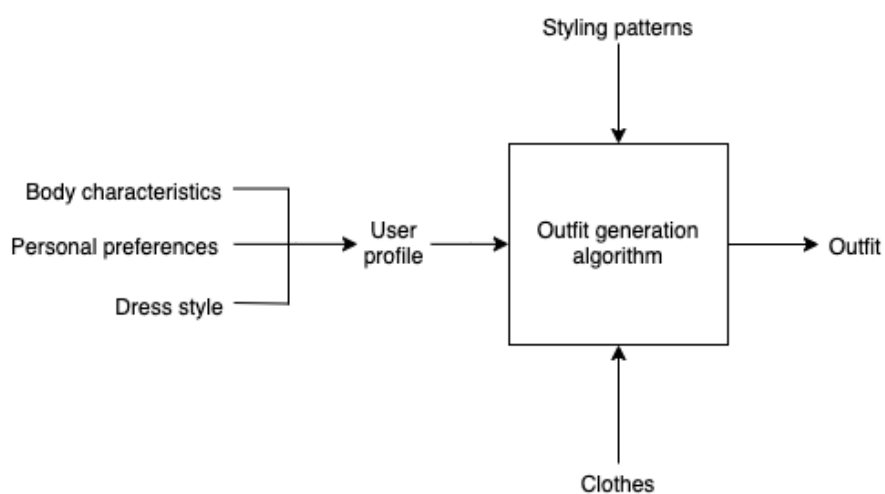
Mixties Elevator Pitch



It will be a **virtual stylist application** that brings personalized fashion style recommendations to their users. It will be fed with the products from a wide range of brands and the users' personal profile in order to provide them accurate advices.

The approach to this virtual stylist service is the **personalized outfits creation**: given the users' profile and their characteristics, the application will suggest several entire outfits that answers the question "What type of clothes they want, and what do they want to achieve?".

In order to generate these personalized outfits, it will use an artificial intelligence algorithm that as input receives the **user profile** and the already established **styling patterns**. Then, as output, using the clothes available in the system, the algorithm will generate an outfit that satisfies the user profile. See Figure 1.2.



*Figure 1.2 Outfit generation algorithm overview*

Now, for a better understanding, we are going to explain each one of these inputs.

## User Profile

All the information needed from the users to define their complete profile in order to give them the most personalized outfits possible. We have split this information in three types: body characteristics, dress style and user preferences.

- **Body characteristics** are related to objective aspects about the user's body, such as height, weight and body morphology.
- **Dress style** is simple, it is just the type of clothes that the user likes. And finally
- **User preferences** are related to the body characteristics such as the part of the body they want to hide or enhance, i.e. a user could want to hide her hips.

## Styling Patterns

The styling patterns are defined by our stylist and are a key for the system because:

- They define what type of clothes combine with each other, independently from the user profile. In other words, what type of clothes combined will make a good-looking outfit and which ones don't.
- They define the visual effect that different clothing characteristics will generate to our perception. For example, horizontal lines patterns will generate the perception of a broader shoulders and shorten the person's figure, whereas vertical lines will stylize the figure.

### 1.1.3 Would people be interested in this application?

We wanted to know if people and potential users would be interested in this product idea, that's when we contacted Natural By Lila [1], a small local women's clothing store.

Through their Instagram social network, we did a small survey to have a first impression. See the results in the following table.

<b>Introduction text</b>	
"Hello girls, today we are going to ask you some questions to know a little bit more about you. It is only 5 easy questions; it will take you only 10 seconds."	
<b>Answers</b>	<b>Votes</b>
<b>Do you like dressing in fashionable?</b>	
Yes	4488
No	78
I don't care	981
<b>How frequently do you renew your wardrobe?</b>	
Every 2 weeks	457
Every month	1275
Every 3 months	1796
2 times a year	1574
<b>Do you use to buy in retail stores or online?</b>	
Online	1344
Retail store	4219
<b>Would you like to have a personal stylist free and quick?</b>	
Yes	4232
No	427
I don't care	700
<b>Do you use other fashion related apps apart from Instagram?</b>	
Yes	2847
No	1778
I have, but I don't use them	666

Table 1.1 Initial form

As we can observe, they generally like to dress in fashionable and are interested to have a personal stylist.

#### 1.1.4 Who are we?

This project covers a lot of different areas, which means that different specializations are needed for the its successful development. That's why we are a multidisciplinary team with the following roles.

- **Paula Carceller** (Personal Stylist). In charge of defining and designing the styling patterns commented before.
- **Lucas Larripa** (Operations Manager). Responsible for overall management of the team and the business side of the project.
- **Marc Momplet** (Marketing Specialist). Responsible for branding, designing and selling aspects of the application.
- **Víctor Juez** (Software Engineer). In charge of the technical aspects of the application.

## 1.2 State of the art

Nowadays, these are the different types of personal styling services available.

- **Traditional personal stylists.** Customer and personal stylist meet in person in order to analyze the customer and give her advices. It is a highly personalized service. Its drawbacks are the high cost of the service and the high amount of time the customer has to spend to receive it.
- **Personal stylist online.** It's the same as the traditional one but instead of meeting in person, they meet online through virtual like channels (like Skype). Compared to the traditional service you can save some time and usually the service is cheaper. Even though, generally the service received is not as good as the traditional.
  - Examples that we are going to see: Carol Davidson [2].
- **"Mystery Boxes"**. Online service that after gathering your personal information through a form, they prepare a personalized box containing different clothes that will fit you. The advantage of these services is that they are so much cheaper than the traditional service. As drawbacks, the customer does not choose what products do they like, neither can see what will be sent to him.
  - Examples that we are going to see: Lookiero [3] and TryTuesday [4].

- **Virtual stylist as a service.** They sell the software to other eCommerce<sup>5</sup> in order to improve the quality of their sites, we are going to see an excellent example, Viume [5].

### 1.2.1 Lookiero

LOOKIERO

Figure 1.3 Lookiero brand logo

Lookiero is a Spanish startup that offers a personal stylist service just as we do, but in a different way.

#### How does it work?



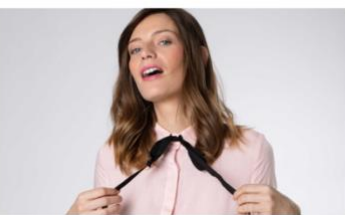
		
<p><b>Tell us about yourself</b></p> <p>Create <a href="#">your profile</a> and tell us about your style and preferences, we'll choose the clothes that will suit you best.</p>	<p><b>Order your Lookiero</b></p> <p>We'll send you 5 items selected specially for you by your Personal Shopper.</p>	<p><b>Try them on and choose</b></p> <p>You'll have 5 days to try on your selection. Return the items you don't want and only pay for what you keep.</p>

Figure 1.4 Lookiero how it works

#### **How does Lookiero works?**

*Lookiero offers you the most comfortable and personalized method of buying clothes without even leaving your house.*

1. *Fill in your style profile with information about your size, tastes and preferences. It'll only take you 5 minutes and will allow your Personal Shopper to choose the items that suit you best.*
2. *Order your Lookiero. After filling in the quiz and creating your account, click on "Order my Lookiero now" and enter your delivery and payment details. Once the order is confirmed, you'll be charged a £10 Personal Shopper service fee. That £10 will be credited toward any items you purchase.*
3. *On the date selected in the calendar, you'll receive five items selected especially for you by your personal shopper.*

---

<sup>5</sup> Electronic commerce

4. *You'll have 5 days to decide what you love and if you want to return anything. After trying everything on, go through the checkout process by logging into your account and indicate which items you're keeping and which ones you're returning. If you were charged the £10 Personal Shopper fee, that will be applied as a credit toward any items you purchase. If you buy all 5 items, we'll give you 25% off the entire purchase. It's very important that you fill out the reasons for returning an item as it will help us to learn more about your style and we'll be able to advise you better in your next Lookiero.*
5. *In the box with your order, you'll find a bag with a prepaid return label in which you can return any items. After you've checked out, follow the on-screen instructions to arrange the return with the courier. You'll also receive these instructions by email. You'll always get free delivery and returns!*

*All this with free delivery and returns!*

**Can I order specific items?**

*As Lookiero's concept consists of a customised selection made by our personal shoppers for you, we cannot give you the option to choose specific items. However, when ordering your Lookiero, you can tell us what you're looking for and we'll do our best to send it to you.*

*On the other hand, if you've received an item in your Lookiero that you like but you want it in another size or colour, leave your comments in the checkout and if it's available in our store we'll send it to you in your next Lookiero.*

Lookiero FAQ webpage [3]

Comparison

Lookiero	Mixties
Assigns a physical personal stylist to the client in order to analyse them and generate the outfits.	Generates the outfits through an algorithm without the need of a personal stylist.
More accurate and personalized service of personal styling	Less accurate and personalized service of personal styling
Sends a box with a single outfit (5 pieces of clothing)	Lets you see as much outfits as you want
Client doesn't know what will be sent to him	Client can see the outfit before buying it
Slow service, as the client has to wait until the box arrives at his house	Immediate service, as the client receive outfits suggestions just after filling his personal information
Client receive outfit suggestions only when buying a box with one single outfit	Client can receive the outfit suggestion anytime and for as long as he wants without the need of buying anything

Personal stylist service free as long as the client keeps at least 1 piece of clothing, otherwise it costs 10€	Personal stylist service completely free
--	--

Table 1.2 Lookiero and Mixties comparison

## 1.2.2 TryTuesday




Figure 1.5 TyTuesday brand logo

TryTuesday is the personal styling platform from Marks & Spencer, a company that specializes in selling high-quality clothing, home products, and food products.

### How does it work?


**How it works**



**1**

**Meet the experts**


We are a team of passionate stylists, dedicated to empowering women to have confidence with their wardrobe & individual personal style



**2**

**Your edit is curated**

We will introduce you to colour, suggest new ways to wear classic pieces and interpret new trends



**3**

**Your wardrobe revitalised**

Never be stuck for what to wear again with a wardrobe that works for you. You can rely on your stylist for each new season and year-round inspiration

Figure 1.6 TryTuesday how it works

### **Who are you?**

*We are Tuesday - an online womenswear personal styling service here to inspire you with your wardrobe and your everyday look. We know how difficult it can be to find your style, whether it's due to a lack of confidence, or just a lack of time. So, we're here to make your shopping trips, your wardrobe, and your daily style as effortless as possible by doing the hard work for you – building inspirational outfits to suit you.*

*Whether it's for your work or your day-to-day wardrobe, our stylists select pieces from this season's collection at Marks & Spencer – helping with everything from a little inspiration to full head to toe looks.*

### **What can I expect after signing up?**

*After a few initial questions on the website we'll introduce you to one of our personal stylists.*

*Your stylist will then send you monthly edits full of inspiration looks and pieces (from the M&S womenswear ranges), solving the wardrobe problems you have, all based on your lifestyle, body shape, budget, and aspired look. You can then browse your edit at your leisure and shop the items you like!*

***Is the service free?***

*Our styling service is completely free to use! We're 100% impartial and always have you and your wardrobe at the heart of what we do.*

***Which shops do you recommend clothing from?***

*Our stylists select pieces from Marks & Spencer only, using the best pieces for you from the current ranges.*

*You can view each item your stylist recommends to you on the Marks & Spencer website. Plus ordering couldn't be simpler, with free next-day delivery to your local store.*

***How frequently can I use the service?***

*Your stylist will send you an edit roughly every month but you can request to receive them less frequently if you like (you can do this on your account once you have signed up).*

TryTuesday FAQ webpage [4].

To gather all the users' personal information needed by the trytuesday's personal shoppers, they use a chatbot (Figure 1.7) that asks you a set of questions, and once it's done, a physical personal shopper is assigned to the user.

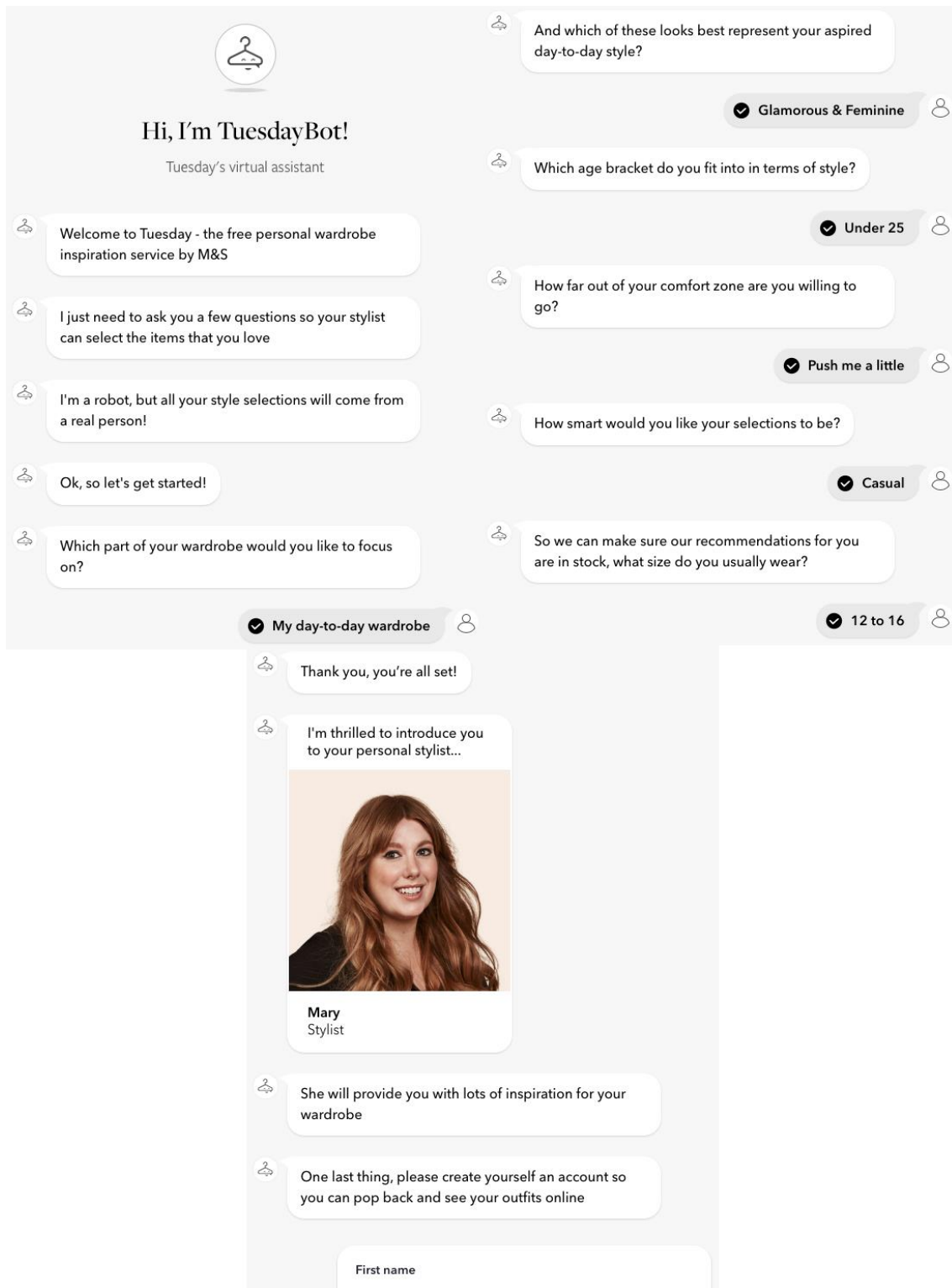


Figure 1.7 TryTuesday Chatbot

## Comparison

TryTuesday	Mixties
Assigns a physical personal stylist to the client in order to analyze them and generate the outfits.	Generates the outfits through an algorithm without the need of a personal stylist.



More accurate and personalized service of personal styling	Less accurate and personalized service of personal styling
Sends a monthly edits (suggestions of outfits)	Lets you see as much outfits as you want
Slow service, as the client has to wait until the box arrives at his house	Immediate service, as the client receive outfits suggestions just after filling his personal information
Client receive outfit suggestions monthly.	Client can receive the outfit suggestion anytime and for as long as he wants without the need of buying anything
Personal stylist service is completely free	Personal stylist service completely free
Outfits generated with clothes only from M&S	Outfits generated with clothes from multiple brands
Only gives you personal stylist suggestions	Offers personal stylist suggestion and the possibility to buy clothes.

Table 1.3 TryTuesday and Mixties comparison

### 1.2.3 Carol Davidson



Figure 1.8 Carol Davidson brand logo

Carol Davidson is a freelancer running the business all by herself. She offers a wide range of services about life and style (Figure 1.9). Among them, the Virtual Stylist service is directly related to our project and it's the one that we are going to analyze.

Life Coaching	Personal Image	Corporate Image	Training for Stylists
Life Coaching Philosophy	A La Carte Image Services	Executive Image Consulting	Image Training
Life Coaching Services	Personal Image Packages	Image Seminars	Business Coaching for Stylists
	Virtual Stylist		Success Sessions
	Group Shopping and Styling		Consultations for Consultants
			Image Tools and Affiliates

Figure 1.9 Carol Davidson services

#### **Virtual Styling - \$375**

*After our meeting I will ask you to upload photos of your clothing and accessories to your virtual closet. Using your existing pieces I will work my magic to put together 10-15 head-to-toe outfits and add them to your online lookbook. I will also list the key items you may want to add to round out and/or update your wardrobe, suggest brands to look at, as well as offer tips for best allocating your budget.*

Carol Davidson webpage [2]

## How does it work?

Carol Davidson offers the traditional personal stylist service but instead of doing it physically, she meets their clients and analyze them through her online portal.

She also offers a “online lookbook”. A virtual space where users can upload their own clothes. This way she can generate outfits using not only new but their users’ clothes.

## Comparison

Carol Davidson	Mixties
Assigns a physical personal stylist to the client in order to analyze them and generate the outfits.	Generates the outfits through an algorithm without the need of a personal stylist.
More accurate and personalized service of personal styling	Less accurate and personalized service of personal styling
Generates outfits with both client’s own clothes and new ones.	Only generates outfits with new clothes.
Slow service, the client has to meet with the personal stylist	Immediate service, as the client receive outfits suggestions just after filling his personal information
Client receive the outfit suggestions when meeting with the personal stylist	Client can receive the outfit suggestion anytime and for as long as he wants without the need of buying anything
So expensive, 375\$	Personal stylist service completely free

Table 1.4 Carol Davidson and Mixties comparison

### 1.2.4 Viume



Figure 1.10 Viume brand logo

*Viume AI SaaS is an AI driven software for ecommerce platforms that brings the best selection of products and customers together through individualized recommendations. Leveraging on image recognition and self-adaptive algorithms, Viume accelerates business to be truly customer centric and predictive.*

*The products that they offer are:*

- **Recommender System.** *Viume’s Recommender System, with the combination of internal inputs, external activities and context data recommends the best match of products for each individual user.*

- **Automatic Product Tagging.** *Viume AI's Automatic Product Tagging easily recognizes the product in the image and automatically tags them with the most specific attributes and categories. The accurate product tags and categories help eCommerce platforms to improve their customers' product search experience and optimize the SEO. Save 40 hours per week using Viume AI's Automatic product tagging which is 14x faster than the manual tagging.*
- **AI Stylist Agent.** *Viume AI Stylist Agent is a digital stylist based on a set of styling functions and conditions designed to offer the best total looks for each individual user.*
  - *Based on the personal characteristics of the client, AI stylist agent creates product combinations, choosing the matching colors, types of products and other attributes.*
  - *The output of this mix and match activity is continuously improved by the Feedback System that learns from the eCommerce and the user.*

### **What type of algorithms do you use?**

*We are using the state-of-the-art machine learning algorithms to train our data. In our recommender system we are using an unsupervised KNN model to find the relations between all data we have. Also, we are using other machine learning techniques to predict the best styles and analyze the feedback.*

Viume FAQ webpage [5]

### How does it work?

#### **What are the different pricing models available?**

*You can contact us to know more about our pricing models. Depending on the solution you are interested in, we will offer you different plans considering your website traffic, the size of your catalog or any custom requirements for your business.*

Viume FAQ webpage

As it is answered in this FAQ question, the service is completely personalized to the client.

### Comparison

The business model of Viume is completely different from the other two and ours. They are a B2B<sup>6</sup> company, whereas our project is a B2C<sup>7</sup>. This means that their product is their software which they sell to other eCommerce. On the other side, one of their products

---

<sup>6</sup> Business to Business

<sup>7</sup> Business to Client

is exactly what we want to develop for our project, the AI Stylist Agent, which unfortunately, as we cannot know precisely what this algorithm does, we cannot compare it with what we want to create. Nevertheless, we are pretty convinced that as they are a pure software company that has an entire team of specialists, from the machine learning and artificial intelligence fields, their product has to be powerful and far better of what we can develop at the moment.

### 1.2.5 Conclusions

Given the existing solutions, we can see notorious similarities with what we want to do, but none is the same. This is why we want to build this project, because we found that there is a gap in the market that we can fill in.

In the Table 1.5 we can observe a comparison between the different services mentioned in this section. We have used a numerated range from zero to three on each category, the higher the better.

	Traditional p. stylist	Online p. stylist	Mystery Boxes	Mixties
Personalization	2	2	1	0
Price	0	0	1	2
Time to service	0	0	0	2
See what you buy	2	2	0	2
Accessible	0	0	0	2
<b>TOTAL</b>	4	4	2	8

Table 1.5 Comparison Table

## 1.3 Scope

In this section we explain what our current situation regarding the Mixties product development is and what is this thesis' goal.

### 1.3.1 Current situation

As you can already have noticed, this is an ambitious project and we are just at the beginning of its development. To guide ourselves into the startup creation process we are following the **Lean Startup Methodology** [6]; *it's a methodology for developing businesses and products that aims to shorten product development cycles (see Figure 1.11) and rapidly discover if a proposed business model is viable; this is achieved by adopting a combination of business-hypothesis-driven experimentation, iterative product releases, and validated learning.*

We can see these iteration steps in the Figure 1.12 and they are described below.

1. We start with an **idea** of what we want to create and the problem that we want to solve. In our case, our idea is what we have seen in the context section.

2. During this process we generate a set of **hypotheses** for the viability of our solution. This includes guessing about who our target clients are, guessing that they really have this problem and guessing that they are interested enough in our solution to pay for it, among others.
3. We have to validate those hypotheses doing an experiment, the **Minimum Viable Product (MVP)**: it's a product with just enough features to satisfy early customers and provide feedback for future product development.
4. At the end, we have to **measure** the data gathered from the users about the MVP and evaluate if the hypotheses have been validated. If it is the case, we continue to the next **iteration** following the same path, improving our MVP and doing the same steps again or. Contrarily, if they have not been validated, we need to **pivot**, meaning that we need to change some aspects of our business model and continue to the next iteration.

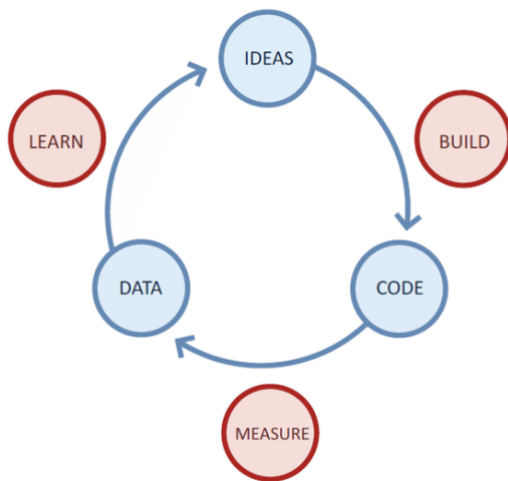


Figure 1.11 Lean Startup Cycles

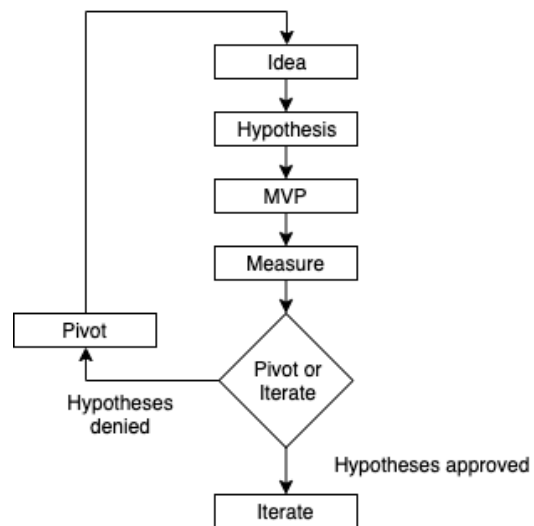


Figure 1.12 Cycle tasks

We find ourselves at the first MVP development stage. This is the thesis' goal, the development of the first **Mixties MVP**

### 1.3.2 Mixties Minimum Value Product

With this very first version of our MVP we want to create a system that consists of two different applications:

- **Admin Dashboard.** A web application for us, as a Mixties team, to manage the information about outfits and products, allowing us to create clothing products and generate outfits with them, either manually or automatically using the AI outfit generation algorithm. It's an internal use application. On the other side, it provides a set of data analysis tools to be able to see statistics about users' interaction with the Mixties Prototype App (explained in the following step)

- **Mixties Prototype App.** An interactive form as a web application. It's meant to be a very simple prototype with only few functionalities. But, it can provide us meaningful information in order to make analysis and extract conclusions. It's explained more in depth in the following sub-section.

## Mixties Prototype App

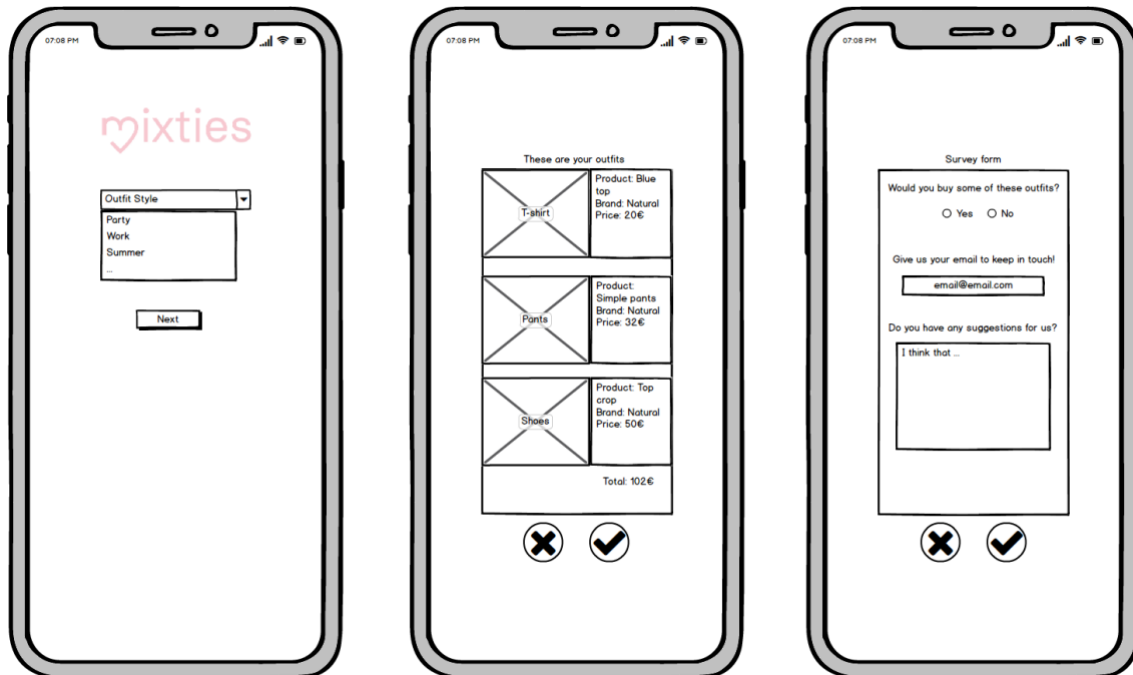


Figure 1.13 Mixties Prototype App mockup

The Mixties Prototype App works as follows.

1. User enters into the application and selects their preferred outfit style
2. The application shows five outfits that fits the style selected, one at a time.
3. User see each one of the outfits individually, with the information of all the products in the outfit. At this point the user is only able to:
  - a. Like/dislike the entire outfit.
  - b. Go to the original webpage of one of the outfits' products, so they can see more information about them.
4. After repeating the step 3 until the five outfits have been liked/disliked, a small form appears asking for personal information.

Simple and concise, we only need these small set of functionalities. The focus here is to not develop useless functionalities that don't provide us value information, nor to the user.

We think this is the best choice for our MVP because:

- It is **easily accessible**, with only one link user enters into the web application, easy path, they don't have to download anything. It's a plus also, considering we want to make ads through Instagram, they can be redirected from the ad to the web app and start using it right away.
- **Easy to do**, interaction with users is minimal, and the steps are simple and clear.
- We can gather **value data** saving all the users' interactions. This means, their geolocation, their outfits likes, dislikes, etc.

## MVP System

The big picture of the system that we are going to build is represented in the figure below.

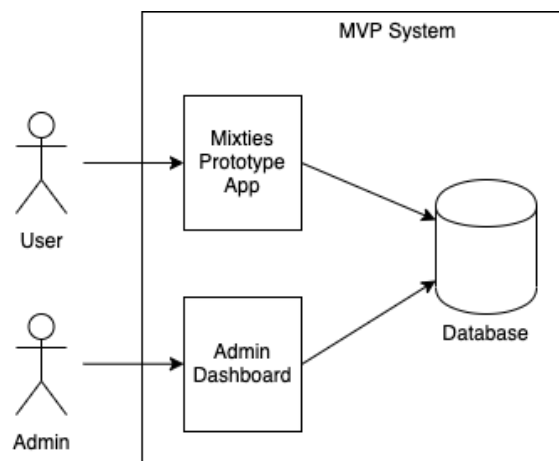


Figure 1.14 MVP System Diagram

We can divide the system workflow in these three steps:

1. **Data entry.** First of all, we have to generate all the outfits that we want to show to the users through the Mixties Prototype App. To do so, we as admins will enter all the products' information into the system using the admin dashboard, and afterward, our stylist will generate all the needed outfits with these products manually or automatically. In the latter case, using the AI outfit generation algorithm.
2. **Users' data gathering.** Once the data entry step is done, the Mixties Prototype App is ready to start showing outfits to users and gathering data from them with their interactions.
3. **Data analysis.** While users are interacting with the Mixties Prototype App, we as admins will be able to see different kind of statistics from the users in the corresponding section of the Admin Dashboard. For example, average outfits like/dislike rate, the most liked outfits, how many users have used the prototype, etc.

## Modular System

An important aspect of this system, the other goal apart of what we have already explained is the design and implementation of a modular system in which these applications will be developed. With modular, we mean that it has to be easy to develop an extended functionality to the current prototype or to develop a completely different prototype and attach it to the system.

We have to develop a system ready to hold this constant change and evolution of the product, over each iteration of the Lean Startup methodology new functionalities for the prototype will be added or a completely different prototype will be required.

### 1.4 Methodology

As we have shown before, we have adopted the **Lean Startup Methodology** for our business development, and we find ourselves at the build-code stage, starting to develop our MVP. This methodology, because of its iterations that cause to adopt constant changes of the product or the business plan in general, suggests using **agile techniques** for the product development, and that is what we have done; not only because its suggestion but because we need to be capable to adopt changes in our requirements as it will occur during the development.

In our case, we have opted for **Scrum** [7], as we are already familiar with it. The process is divided in three phases: Agile Project Inception, Agile Project Planning and Agile Project Execution. See the table below.

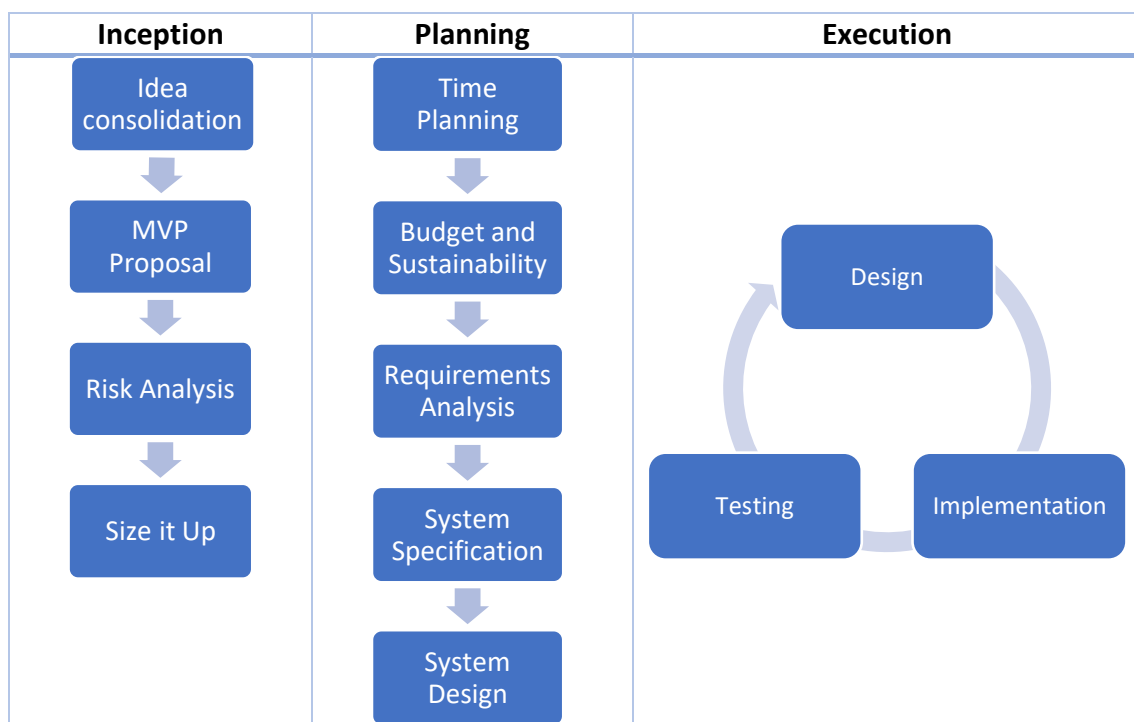


Table 1.6 Methodology diagram



### 1.4.1 Monitoring Tools

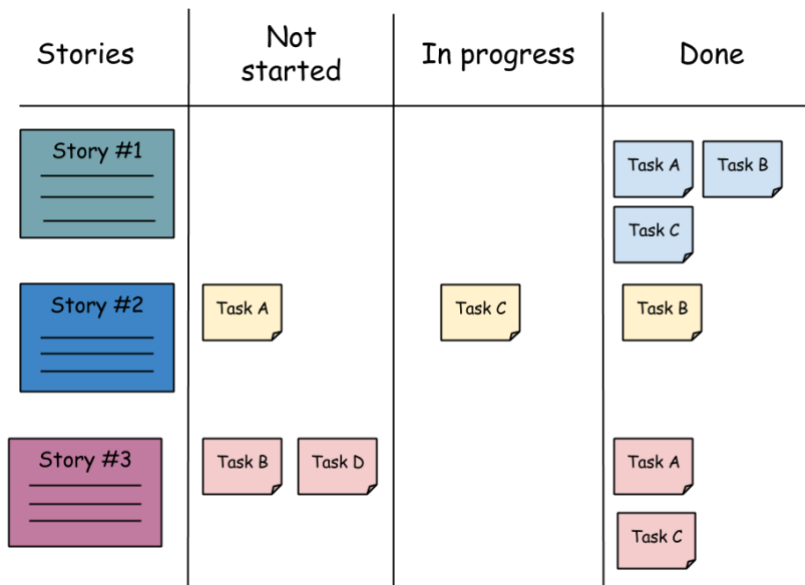
To keep track of the project’s timeline we will use different technologies and tools.

- Basic spreadsheet to write down each task done during the day, here is an example of how it is structured.

Date	Activity	Duration(h)	Comment
10/11/2019	Application Infrastructure	5	
11/11/2019	Domain analysis and design	4	

*Table 1.7 Task tracking spreadsheet*

- Sprint task board for the current iteration tracking, it is physically placed in the working environment. This is because as I’m the only Mixties member in the development team, there is no need to use digital alternatives like Jira or Taiga. Also, I personally prefer the physical option.



*Table 1.8 Scrum storyboard*

At the end of each sprint, using the information from the story board and the spreadsheet, we will generate a Burn-Down chart. It will show us how quickly we are burning through our user stories and when we can expect them to be done. Then, comparing it with the Gantt diagram we can check the deviations.

## 1.5 Obstacles and Risks

In this section we are going to describe the obstacles and risks of this project. Then, we provide the mitigation and contingency strategies for each risk found.

### 1.5.1 Obstacles

First of all, thinking about our team we encounter some obstacles:

- **Inexperienced team.** We are four young entrepreneurs with little or no experience.
- **Only one software engineer.** Only having one person in the development team lowers both the quality of the product and the scope of the project compared to if there were a team with multiple roles.

On the other side, regarding on the development of the MVP we have another obstacle:

- **Data entry.** We are manually entering the products into our system with their corresponding tags. This is a slow and tedious process.

### 1.5.2 Risks

In the following table we have all the risks with their probability to occur and their impact if it happens.

Description	Probability	Impact
<b>AI doesn't generate well-combined outfits</b> This could happen because of multiple factors <ul style="list-style-type: none"><li>• <b>bad product classification</b>, we are tagging the products in a wrong way (style patterns)</li><li>• Bad AI algorithm</li><li>• Small data set of products</li></ul>	High	Low
<b>External services non available</b> Our system is connected to some external services to gather information about user's geolocation (explained later in the design section). They could not be available.	Low	Medium
<b>Databases down</b>	Low	High
<b>Servers down</b>	Low	High
<b>We don't obtain valuable data</b>	Low	Low
<b>Unused by users</b>	Medium	High

Table 1.9 Risks table

### Mitigation strategies

#### **AI doesn't generate well-combined outfits**

- Previously with the team, analyze which are the useful characteristics to use for product tagging and establish the relations between them
- Choose a simple AI algorithm just to get the job done and don't overcomplicate the solution.

#### **External services non available**

- Research about the existing solutions and choose one with high availability rate.
- Make tests with the external service to check its availability and performance.

#### **Databases down**

Look for a cloud DB provider with low fault rates

#### **Servers down**

Look for a cloud provider with low fault rates

#### **We don't obtain valuable data**

- Gather the maximum data possible
- Previously discuss with the team which are the most useful indicators to gather.

#### **Unused by users**

Try to make an interesting web application with an easy and attractive interface.

*Table 1.10 Mitigation table*

## Contingency strategies

These strategies minimize the damage caused by the risks in case we couldn't avoid them.

#### **AI doesn't generate well-combined outfits**

Make outfits manually without the use of AI. It is the famous Mechanical Turk strategy, that even though we sell that we are automatically creating the outfits we are doing it manually.

#### **External services non available**

Don't use the external service at all, even though it will cause obtaining fewer valuable data.

#### **Databases down**

If at the moment to access the database it is not working, show an error message and securely stop the server operations to avoid further problems.

#### **Servers down**

If at the moment to access the server it is not working, show an error message to the user.

#### **We don't obtain valuable data**

- Talk with data specialist to have a deeper insight about the data we have and what we can extract from it.
- If we can't extract anything, pivot and change the MVP accordingly.

**Unused by users**

Talk with our customer target and try to find out what we have done wrong.

*Table 1.11 Contingency table*

## 2. Time Planning

The project starts on the 1<sup>st</sup> of June and it is expected to be defended on the 23<sup>rd</sup> of January. But, as the memory has to be delivered one week before the defense date, it has to be finished by the 15<sup>th</sup> of January.

On the other side, the total amount of hours estimated to develop the entire project is 450h. I got this estimation considering the following factors. First of all, the workload of the bachelor thesis is 18 ECTS and each ECTS implies 25h of work, that means that by doing 18ECTS x 25h/ECTS we get a total of 450h of workload. This is the reference that we will use for the time planning.

### 2.1 Description of tasks

As explained at the Methodology section (1.4 Methodology page 23) this project has adopted agile methodologies, and we have split the project in three phases: the Agile Project Inception, Agile Project Planning and Agile Project Execution.

#### 2.1.1 Agile Project Inception

This phase is where the general idea of the Minimum Viable Product is defined. The tasks done in this phase are to align the team into the same goals, sketch up the MVP and briefly estimate its development duration.

##### **I.1 Idea Consolidation**

Consolidates the current business model idea of Mixties with the team.

##### **I.2 MVP Proposal**

Sketch up the technical solution proposed with diagrams and mockups, discussed with the team to ensure everyone agrees. Is a way to have a general vision of what is going to be the final solution.

##### **I.4. Risk Analysis**

Expected risks are analyzed in early stage.

##### **I.5. Size it Up**

First approach to the time estimation and scope of the project

*Table 2.1 Agile Project Inception tasks*

### 2.1.2 Agile Project Planning

In this phase we create the time planning, budget and sustainability impact of the project. It is based primarily on the deliveries done in the Project Management course (GEP) during the thesis' initial phase.

On the other side, here we also gather the requirements, define the system specification and the system design.

#### **P.1. Context and Scope**

Contextualizes the thesis and describes its scope, defines its general objective, relevance, justification and how will be developed. The Product Backlog is defined during this stage.

#### **P.2. Time Planning**

Temporal planning for the thesis' total execution. There is a description of all the phases of the project and the resources and requirements related to each of them.

#### **P.3. Budget and Sustainability**

The budget of the project, including the identification of costs, the cost estimates, and the management control.

On the other side, sustainability is analyzed by the environmental, economic and social impacts.

#### **P.4. Final Document (GEP)**

After the feedback of the I.1, I.2 and I.3, this task is used to correct these parts and prepare the final document.

#### **P.5. Requirements Analysis**

Functional and non-functional requirements gathering for the system, use case diagrams, user stories and sequence diagrams for their definition.

#### **P.6. System Specification**

Specification of the system's domain with a UML class diagram and it's corresponding restrictions.

#### **P.7. System Design**

Design of the system, the architecture applied, technologies and data structure.

*Table 2.2 Agile Project Planning tasks*

### 2.1.3 Agile Project Execution

This phase is for the product development. Based on the Scrum iterations. We have split it in five Sprints of two weeks each except the last one of one single week.

Each sprint is identified by “S.1, S.2, . . . S.n” where n is the number of the corresponding sprint.

Each Sprint includes the following tasks:

1. **Sprint Plan.** Set the Sprint Goal and the Sprint Backlog for the current iteration.
2. **Sprint Execution.** Design, development and testing of Sprint Backlog stories.
3. **Sprint Review and Retrospective.** Once the Sprint is done, we will meet with the team to evaluate the current Sprint, update the Product Backlog and prepare the next one.

Also, we set three releases (a group of user stories that together sets a feature).

- **Release 1: Mixties Prototype App (MPA).** In this release we expect to have the frontend for the Mixties Prototype App up and running. Released in the S.1
- **Release 2: Backend (B).** In this release we expect to have the whole backend finished for both the Mixties Prototype App and the Administrators Dashboard. Developed during the S.2 and S.3. Released at the S.3
- **Release 3: Administrators Dashboard (AD).** In this release we expect to have the Administrators Dashboard finished. Developed during S.4 and S.5. Released at the S.5

#### 2.1.4 Resources

In order to develop the tasks defined above, different kind of resources are needed, specified as follows

Software		
R.1	Visual Studio Code	The code editor used to develop the application
R.2	Google Docs	Office applications used to keep track and document the project
R.3	Github	Cloud repository of the application source code and version control.
Hardware		
R.5	Macbook Pro 15”	The laptop used to develop the entire project
R.6	Oneplus 5T	Smartphone used to test the application
Human		
-	Me	As a single developed project, I will be in every task
R.7	Mixties Team	The rest of the Mixties Team

*Table 2.3 Resources table*

## 2.2 Estimates and Gantt

Considering that the start date is the 16th of September and the deadline is on the 15th of January, we have a total of 82 working days to dedicate to the project. As working days, we consider from Monday to Friday and excluding the following festivity days during the two dates: 23 September, 1 November, 6-25-26 December, 1-6 January.

Consequently, as the total workload of the project is estimated with 450h, divided to the 82 working days, we have an average of 5.5h/day which can be considered acceptable.

### 2.2.1 Time planning summary

Name	Days	Hours	Resources	Predecessors
<b>Agile Project Inception</b>			R.2, R.5, R.7	
I.1 Idea Consolidation	6	10		
I.2 State of the Art	2	10		I.1
I.3 MVP Proposal	9	11		I.2
I.4 Risk Analysis	3	5		I.3
I.5 Project Size Up	3	5		I.3
<b>Agile Project Planning</b>			R.2, R.5	
P.1 Context and Scope	8	24,5		I.5
P.2 Time Planning	5	8,25		P.1
P.3 Budget and Sustainability	6	9,25		P.2
P.4 Final Document	6	18,25		P.3
P.5 Requirements analysis	4	10		I.5
P.6 System specification	7	14		P.5
P.7 System design	7	15		P.6
<b>Agile Project Execution</b>			R.1, R.2, R.3, R.4, R.5, R.6	
S.1 Sprint 1   Release 1: MPA	12	60		P.3
S.2 Sprint 2   B	12	60		S.1
S.3 Sprint 3   Release 2: B	13	60		S.2
S.4 Sprint 4   AD	13	60		S.3
S.5 Sprint 5   Release 3: AD	6	30		S.4
<b>Memory Documentation</b>		40	R.2, R.5	I.5
<b>TOTAL</b>		450,25		

*Table 2.4 Time planning summary*



## 2.2.2 Gantt Chart

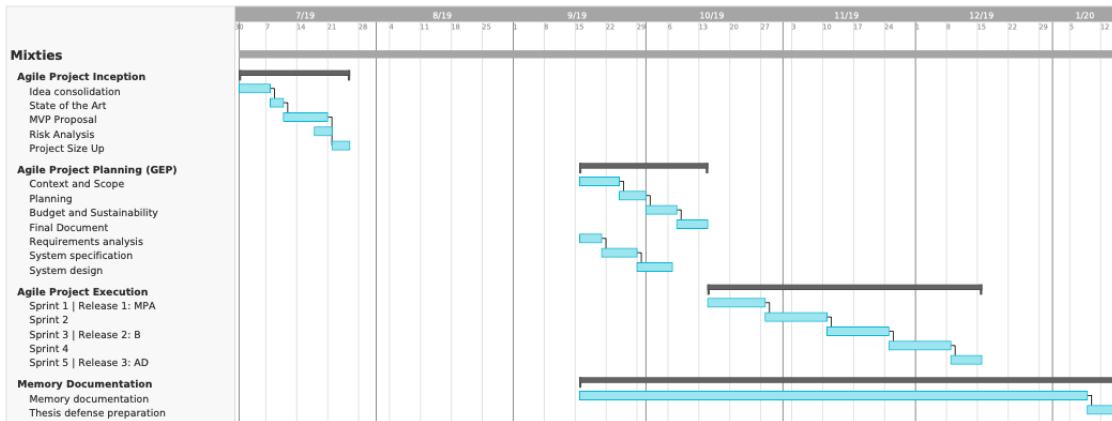


Figure 2.1 Gantt Chart diagram

## 2.3 Risk Management: Identification and alternative plan

Given the specified risks, there is such an amount of uncertainty about how much complexity and workload will be generated over the Agile Project Execution phase of the project, in consequence, the estimated project planning timeline will probably be altered in some sort.

As this is expected, the scrum methodology used for this project comes up. It's flexible on the plan change and it's tackled after each sprint. Given the situation that we are advancing slower than expected, one solution is doing fewer user stories, hence reducing the scope of the project. On the other side, we can postpone the deadline to finish it, which in our case, as we have a fixed deadline, it is not possible.

To prevent any of this, we will overestimate the story points assigned to each Sprint in order to overcome the unexpected situations. Moreover, in the worst of the cases, where the project cannot be finished in time in any way, the reduction of the scope is contemplated by eliminating as many user stories as needed from the product backlog. As they are sorted by priority, the last ones would be the first to eliminate.

## 3. Budget and Sustainability

### 3.1 Cost Estimates

We have 2 types of costs, direct and indirect. The direct costs are human costs, the salary that will be paid to the team for developing the project. The indirect costs, on the other side, are the costs that are not directly connected to the development of the project itself. These ones are the software, hardware and generic costs (like electricity).

#### 3.1.1 Direct Costs

Regarding this topic, as the project is developed only by one person and in order to make this economic management more realistic, it has been considered all the roles that would normally be involved in a similar project.

The point of this part is to obtain the cost of developing each activity of the Gantt's chart. This cost will be based on the total hours assigned to each activity, and how many of these hours will be done by each of the team roles, as it depends on their salaries.

First, in Table 3.1 we have calculated the net salary for each one of the roles. For a better understanding we are going to introduce some concepts used in the mentioned table.

- **Gross salary (€):** the yearly gross salary of each role, the data has been obtained from Michael Page's study [8].
- **Gross salary(€/h):** the gross cost per hour of work of each role. It has been considered a total of 1764 working hours a year. Defined by the Workers' Commissions official website [9].
- **Net salary (€/h):** adding the costs of Spain's social security to the gross salary per hour. Multiplying the gross cost by a factor of 1.35.

Role	Gross salary (€)	Gross cost (€/h)	Net cost (€/h) x 1,35
Project Manager	40.000,00	22,68	30,61
Architect	40.000,00	22,68	30,61
Developer	18.000,00	10,20	13,78
UX/UI designer	18.000,00	10,20	13,78
Quality Assurance	18.000,00	10,20	13,78

Table 3.1 Costs(€/h) by role

Secondly, Table 3.2 represents the percentage of participation in each activity by each one of the roles.

Name	Project Manager	Architect	Developer	UX/UI	Quality Assurance
<b>Agile Project Inception</b>	100%				
<b>Agile Project Planning</b>					
P.2 Time Planning	100%				
P.3 Budget and Sustainability	100%				
P.4 Final Document	100%				
P.5 Requirements analysis	100%				
P.6 System specification		79%	7%	7%	7%
P.7 System design		79%	7%	7%	7%
<b>Agile Project Execution</b>					
Sprint planning	20%	20%	20%	20%	20%
Software architecture		100%			
Development			80%	20%	
Testing					100%
Documentation	50%	25%	8,3%	8,3%	8,3%
Sprint Review	20%	20%	20%	20%	20%
Control meeting	20%	20%	20%	20%	20%
<b>Memory Documentation</b>	100%				

Table 3.2 Activity participation by role

Finally, in Table 3.3 we got the total amount of money each role would be paid for the entire project duration (450h). It is obtained by calculating the total hours they contribute to the project. To do so, we have applied the contribution percentages of the above table to the total time duration of each activity as shown in the Time Planning section, page 31.

Role	Net cost (€/h) x 1,35	Hours worked	Total salary (€)
<b>Project Manager</b>	30,61	165,65	5.070,92
<b>Architect</b>	30,61	55,31	1.693,16
<b>Developer</b>	13,78	152,92	2.106,61
<b>UX/UI designer</b>	13,78	44,92	618,85
<b>Quality Assurance</b>	13,78	31,42	432,88
		<b>450,23</b>	<b>9.922,42</b>

Table 3.3 Salary by role

There is another useful way to represent the direct cost, and this is by calculating the amount of money each activity of the Gantt's Chart (Figure 2.1 in page 32) costs. Their

values are calculated again using the number of hours each of the roles participated into the total of each activity and how much money does that hours costs respective to the roles. We can see the result as follows.

Activity	Amount (€)	Observations
<b>Agile Project Inception</b>		
I.1 Idea Consolidation	397,96	
I.3 MVP Proposal	428,57	
I.4 Risk Analysis	214,29	
I.5 Project Size Up	214,29	
<b>Agile Project Planning</b>		
P.1 Context and Scope	750,00	
P.2 Time Planning	252,55	
P.3 Budget and Sustainability	283,16	
P.4 Final Document	558,67	
P.5 Requirements analysis	306,12	
P.6 System specification	379,07	
P.7 System design	406,15	
<b>Agile Project Execution</b>		
Sprint planning	184,59	
Software architecture	688,78	
Development	2.479,59	
Testing	309,95	
Documentation	475,01	
Sprint Review	184,59	
Control meeting	184,59	
<b>Memory Documentation</b>	<b>1.224,49</b>	
<b>Total CPA</b>	<b>9.922,42</b>	· Total CPA: Total personnel costs by activity (Gantt activities)

Table 3.4 Direct costs by activity

### 3.1.2 Indirect Costs

As indirect costs, we can differentiate over software, hardware, and generic costs.

#### Software

It's detailed in the Table 3.5. There is no cost at all as every software resource used is either open source or free versions of the products.

Product	Price
Visual Studio Code	0
Google Docs	0
Github	0
Taiga	0
Heroku	0
Postman	0
<b>TOTAL</b>	<b>0</b>

*Table 3.5 Software Costs*

## Hardware

Secondly, in the following table are the hardware costs. The main two resources used are the MacBook Pro 15" (laptop) and the OnePlus 5T (smartphone). To calculate their cost, we're not considering their entire price, but the proportional part of the price used in the project. This means that we divide the total cost of the product to its expected lifespan and then we count the proportional usage time for the project (considering there are 250 working days per year, 8h working hours a day and a total of 450h for the project). Technically, we have used the following equation.

$$\text{Amortization} = \text{Total cost} / \text{Lifespan} / 250 \text{ days} / 8\text{h} * 450\text{h}$$

Name	Cost	Lifespan	Amortization
MacBook Pro 15"	3,500€	4 years	197€
OnePlus 5T	550€	2 years	62€
<b>TOTAL</b>			<b>259€</b>

*Table 3.6 Hardware Costs*

## Generic Costs

Lastly, as generic costs, we have the electricity needed to power up the laptop and internet access.

- **Electricity.** We considered only the waste of energy needed to power up the laptop as it is the only tool used for the entire development of the project. In the worst case, the consumption is 87W (is the charger's maximum). We consider the laptop is working the entire 450h of the project at this consumption rate and that the average cost of electricity is 0,15€/kwh. As a result, we obtain the following equation.

$$\text{Electricity} = 87\text{W} * 450\text{h} = 39.15\text{kWh} * 0.15\text{€/kWh} = 5.87\text{€}$$

- **Internet.** We considered 53€ as the monthly cost of an average internet plan. Then we have to calculate the proportional part for the development of this project. We have done it calculating the cost per working hour and then multiply

it for the total hours of the project (450h). 53 € each month is a total of  $53 \times 12 = 636\text{€}$ /year. As before, considering there are 250 working days per year and 8 working hours per working day, we get  $636/250/8 = 0.318\text{€/hour}$ . Finally, as the project is 450h long, we get a total of  $0.318 \times 450 = 143.10\text{€}$  as the internet cost.

$$\text{Internet} = 53\text{€} * 12 \text{ months}/250/8 * 450 = 143.10\text{€}$$

Therefore, the total generic cost is:

$$\text{Generic Cost} = \text{Electricity} + \text{Internet} = 5.87\text{€} + 143.10\text{€} = 148.97\text{€}$$

### Total Indirect costs

In the following table we have a summary with all the indirect costs detailed.

Resource	Cost (€)
<b>Software</b>	
Visual Studio Code	0
Google Docs	0
Github	0
Taiga	0
Heroku	0
Postman	0
<b>Hardware</b>	
MacBook Pro 15"	196.88€
OnePlus 5T	61.88€
<b>Generic Costs</b>	
Electricity	5.87
Internet Access	143.10
<b>Total IC</b>	<b>407.72</b>

*Table 3.7 Indirect Costs*

### 3.1.3 Contingency and Incidentals

Apart from the Direct and Indirect costs, we have to consider a contingency level and the incidentals that might occur. Contingency is nothing more than a security margin to mitigate the unexpected and unpredictable, a percentage of the total of direct and indirect costs, it usually goes from 5% until 15% in software project.

On the other side, incidentals are unexpected but at the same time known issues that could appear during the project. They are measured by the probability they could occur, and their cost is the cost to overcome the incidental.

- As contingency, we will consider 15%, because even though the budget is well detailed, as the project will be continuously evolving during its development, we prefer to keep it safe. Knowing that Direct Costs + Indirect Costs = 1,563.79. The 15% of that is 1,563.79. This is the contingency cost.

- Regarding incidentals, we are considering only the possibility of hardware failure. Basically, if either the laptop or the smartphone breaks, causing an interrupt of the project's development. We are considering a 20% of probability to occur any malfunction for the smartphone and an average of 100€ to repair it. For the laptop, a 10% of probability plus an average cost of 500€ to repair it.

Name	Cost (€)	Description
<b>Contingency</b>	1.549,52	15% of IC+DC
<b>Incidentals</b>		
Broken OnePlus 5T	20.00	20% probability, 100€ avg repair
Broken MacBook Pro	50.00	10% probability, 500€ avg repair
<b>TOTAL</b>	1,619.52€	

*Table 3.8 Contingency and incidentals cost*

### 3.1.4 Summary

Finally, the total cost of the project is the sum of the direct and indirect costs plus the contingency and incidentals. We can see it in the following table.

Name	Amount (€)
Direct Costs	9,922.42
Indirect Costs	407.72
Contingency	1,549.52
Incidentals	70
<b>TOTAL</b>	11,949.66

*Table 3.9 Total Cost*

### 3.1.5 Management Control

In order to keep control of the actual costs and being able to contrast with the cost estimates, we have a spreadsheet where we will constantly write down all the costs appearing during the execution of the project, so in the sprint review, it can be contrasted with the estimates and supervise cost deviations.

The most important thing to keep track of is the timeline. This is because human resources are the most expensive cost of all the project and the time deviation is a common problem. Therefore, a deviation of the time will result in a different cost. We will use the following indicators

- Human resources deviation = (Expected cost - Real cost).

Secondly, we will also set an indicator for the indirect costs, as it could happen that during the execution of the project, we decided to adopt a new technology which could not be free or increase the current server's plan in order to have more power. Anyway, we will use the following indicator for this purpose.

- Technology resources deviation = (Expected cost - Real cost).

Finally, to unify all the cost deviation we will use the following indicator.

- Total cost deviation = (total expected cost - total real cost).

## 3.2 Sustainability

In this section we are analyzing the environmental, economic and social impact of our project in the process of putting it into production. But before to do so we include a self-assessment about the subject by myself.

### 3.2.1 Self-assessment

After completing the sustainability form about software and general informatics projects, I realized that I do not have a lot of background in the subject and that there are a lot of things to learn, for example, when I thought about sustainability I was only aware of the environmental dimension.

I was thinking that during the bachelor, even though we have talked about some aspects of sustainability in different courses and done some activities, we should do more about this matter. It is true that we also have some elective courses about the topic, but I think it is not enough to cover all the important aspects of sustainability. I hope to see it in the near future.

Apart from that, doing the form I realized how much “power” we have in our hands as a software engineers and how easily we can use it. An infinite number of possibilities to change all the people’s lives. And given this, we have to be conscious about it to all the repercussions we can generate.

On the other side, regarding the environmental dimension, nowadays is becoming a topic more and more important, as the recent events like the burnings in the Brazilian Amazon region, the melting of the ice of the North Pole or even the technology residuals dumped into African countries like Ghana, and all the risks that carry with it are only a small set of examples. We only have one world and we have to take care of it. That’s why us, as software engineers when starting a new project have to be conscious about it and try to leave the world better than how we have encountered it.

### 3.2.2 Economic Dimension

Regarding the economic dimension we have done a detailed cost estimation for the development of the project and it seems reasonable when thinking about the size of the project. Most of the costs comes from human resources, the team needed to develop the project is the most expensive part compared to the small expenses from other general costs, this is because the pure software nature of the project.



Given that the project is developed only by myself and in order to generate a more realistic invoice, we decided to calculate it assuming it was developed by a team with the different roles needed. Similar to what we can encounter in the market right now.

### 3.2.3 Environmental Dimension

Regarding the environmental dimension, our impact during the development of the project is minimal as we can only consider the power consumption of the hardware and the corresponding working environment installation. We have calculated this at the cost estimations section.

The fact that the product that we have developed in this project is just software is the reason of this minimal environmental impact. Nevertheless, we should be aware that once our system is in production the situation is completely different, this is because it would be hosted in servers working 24/7 which are constantly consuming power. And also, as the time passes, more data will be saved into the databases resulting also in bigger servers to host them consuming more power.

### 3.2.4 Social Dimension

Personally, this project will give me a wide view of all the aspects that are involved in the development of an entire software application, and above all, I will improve the project management skills which I was unaware of and it interests me a lot.

Apart from that, this dimension is the one that our project will generate the most impact compared to the other two. This is because it defines a new way to buy clothes through the internet, even though there are some similar alternatives, this project gives the immediacy that lacks on these alternatives as we do not deliver clothes to the clients, we only show them our suggestions after a couple of clicks.

## 4. Requirements Specification

In order to gather the system requirements, we will recap to the Lean Startup Methodology that we introduced before (section 1.3.1 page 19).

After consolidating our idea, we as a team discussed the best approach to our very first Minimum Viable Product and the requirements of it. This means that the requirements were gathered by consensus among the Mixties team

### 4.1 Stakeholders

#### Admins – Mixties Team

- **Software Engineer.** In charge to design and develop the system.
- **Fashion Stylist.** In charge to the define the style patterns used to generate well-combined outfits and entry the product's data into the system.
- **Marketing Specialist.** In charge to design the UI and UX of the web application.
- **Operations Manager.** In charge to coordinate the team and actively involved in team decisions.

They interact with the system through the Admin Dashboard.

#### Users

They will go through all the functionalities rating outfits, visiting products and answering the satisfaction form eventually. They interact with the system through the Mixties Prototype Application.

## 4.2 Use Cases Diagram

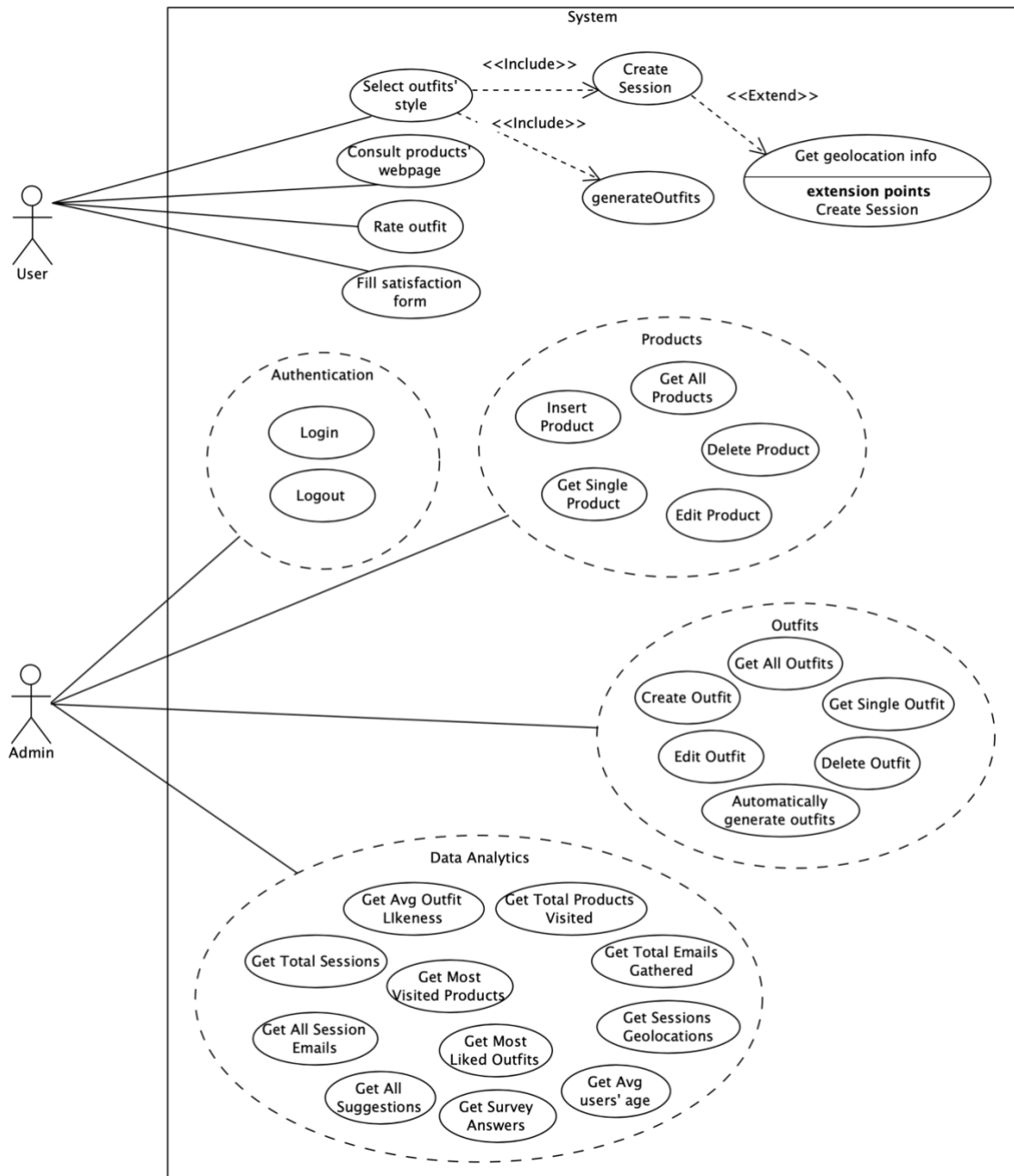


Figure 4.1 Use Case Diagram

## 4.3 Functional requirements

### 4.3.1 Mixties Prototype App

<b>#1</b>	Get outfits
User story	<b>As a user</b> <b>I want</b> to select my outfit style preference <b>so that</b> I can get five outfits of that style.
Description	As a user I want to see outfits of my selected style preference, that's the starting point of the web application. Given the user is not registering or login in any way, we have to create a session to identify them during the connection. To have more insights, their IP is saved. It is used to get their geolocation information through an external service. After the session is created, 5 outfits of the selected style preference are assigned to the user.
Tasks	<ul style="list-style-type: none"><li>• Create style preference selection page</li><li>• Create user session with selected preference</li><li>• Obtain user IP</li><li>• Get geolocation information from IP</li><li>• Save session into DB</li><li>• Get 5 outfits of the selected style preference</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li><li>• If geolocation information cannot be obtained, session is created without this value anyway</li><li>• After selecting an outfit style, a session has to be created and assigned five outfits to it</li></ul>
<b>#2</b>	Visit product official webpage
User story	<b>As a user</b> <b>I want</b> to consult a product official webpage <b>so that</b> I can see more information about it.
Description	After showing a complete outfit to the user, it's expected that maybe they are interested in checking the official webpage of a product. We have to save the source of the original product webpage of all the products in our system to provide a redirection to this when they click to the product.
Tasks	<ul style="list-style-type: none"><li>• Save original product source into DB</li><li>• Add redirection event to the original source into the outfit presentation page</li><li>• Save user visit to official webpage of a product into DB</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li></ul>

<b>#3</b>	Rate outfit
User story	<b>As a user</b> <b>I want</b> to rate an outfit <b>so that</b> I can give my opinion and/or see the next outfit.
Description	When showing an outfit to the user, their only interaction with it is to check the official webpage of a specific product and check out the next outfit. User has to rate the current outfit liking or disliking it and it is mandatory. It's the only possible way to check out the next outfit. If it is the fifth outfit, after rating the outfit, user is redirected to the form page.
Tasks	<ul style="list-style-type: none"> <li>• Create outfit presentation page</li> <li>• Save user rating into the DB</li> <li>• Move to next outfit after user rates the current one or to the satisfaction form if already rated the fifth one.</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> </ul>

<b>#4</b>	Satisfaction form
User story	<b>As a user</b> <b>I want</b> to fill the satisfaction form <b>so that</b> I can give my opinion.
Description	After user has seen their fifth outfit and rated it, is moved to the satisfaction form, it consists of a multiple question regarding their opinion of the outfits seen before. On the other side, it is important to ask for their personal information for future needs.
Tasks	<ul style="list-style-type: none"> <li>• Create form page</li> <li>• Save user response into DB</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• It is not mandatory to fill the form</li> </ul>

## 4.3.2 Admin Dashboard

### Authentication

<b>#5</b>	Login
User story	<b>As an Admin</b> <b>I want</b> to login into the system <b>so that</b> I can use de admin functionalities
Description	Admin logs into the system using their email and password credentials.
Tasks	<ul style="list-style-type: none"><li>• Create login page</li><li>• Create admin model</li><li>• Implement authentication system</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li><li>• Admin logs in if and only if their credentials are correct</li><li>• Error message is shown if bad credentials are given</li><li>• Passwords are securely encrypted into the system</li></ul>

<b>#6</b>	Logout
User story	<b>As an Admin</b> <b>I want</b> to logout from the system <b>so that</b> I can close my session securely
Description	Admin logouts from the system and stops its interaction with it.
Tasks	<ul style="list-style-type: none"><li>• Create logout functionality</li></ul>
Acceptance criteria	Tasks have to be completed

### Products

<b>#7</b>	Insert Product
User story	<b>As an Admin</b> <b>I want</b> to create a new product <b>so that</b> I have a new product into the system to create outfits with
Description	Admin inserts all the required information of a product and its characteristics and it is created and saved into the system. The id of the product is auto generated and not visible to the Admin.
Tasks	<ul style="list-style-type: none"><li>• Create Product model</li><li>• Create insert product page</li><li>• Implement functionality</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li></ul>

- Product is created only if all the required information is filled: photo, name, link, brand, price and photo. (Characteristics are optional to fill)

<b>#8</b>	Get All Products
User story	<b>As an Admin</b> <b>I want</b> to get all the products <b>so that</b> I can see all the products available in the system
Description	Admin can see all the products in the system with their corresponding information. It is important that the products can be ordered by each one of their parameters so that it has a better visibility.
Tasks	<ul style="list-style-type: none"> <li>• Create logout functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Admin gets all the products</li> <li>• They can be ordered by asc and desc by its parameters</li> </ul>

<b>#9</b>	Get Single Product
User story	<b>As an Admin</b> <b>I want</b> to get the information of a specific Product <b>so that</b> I can see their full information
Description	Admin will be able to see all the information of the specific Product including the outfits where it is used.
Tasks	<ul style="list-style-type: none"> <li>• Create logout functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• All the information about the product: photo, name, price, brand, link, product type and characteristics</li> <li>• Related outfits visible</li> </ul>

<b>#10</b>	Edit Product
User story	<b>As an Admin</b> <b>I want</b> to edit a specific product <b>so that</b> I can change its information if necessary
Description	Admin will be able to edit the information about a product: photo, name, price, product type, link, brand and characteristics.
Tasks	<ul style="list-style-type: none"> <li>• Create edit page</li> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Edit successfully if and only if all the required information is filled</li> <li>• If some required information is missing cancels the edit and show an error message</li> </ul>

- Required information: photo, name, price, brand, link and product type

<b>#11</b>	Delete Product
User story	<b>As an Admin</b> <b>I want</b> to delete a product <b>so that</b> I can manage the system products as I prefer.
Description	Admin will be able to delete a specific product if they want to. If there are outfits where this product is being used, they will be deleted too.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Product is deleted from the system</li> <li>• Product's related outfits are deleted too</li> </ul>

## Outfits

<b>#12</b>	Create Outfit
User story	<b>As an Admin</b> <b>I want</b> to create a new outfit <b>so that</b> it can be shown in the prototype app
Description	Admin will be able to create a new outfit combining three different products and selecting which style(s) this outfit relates the most. To do so, the products have to be already inserted into the system. The outfit style(s) is independent to the products styles and is up to the admin preference. No restrictions to the product combinations, up to the admin to make combinations as they prefer
Tasks	<ul style="list-style-type: none"> <li>• Create Outfit model</li> <li>• Create page</li> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Only successfully created if and only if three products are selected and at least one style.</li> <li>• If there is missing information, cancels the creation and show an error message</li> <li>• Once outfit is created it has to be visible in the prototype app</li> </ul>



<b>#13</b>	Automatically generate outfits
User story	<b>As an Admin</b> <b>I want</b> to generate outfits automatically <b>so that</b> we don't have to enter all of them manually
Description	The algorithm has to generate well combined outfits considering only the products characteristics and use the style patterns to know how to combine those characteristics are related.

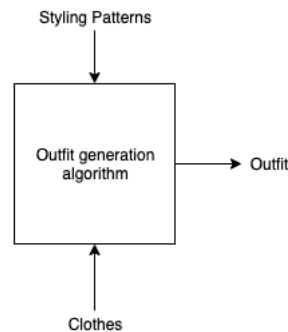


Figure 4.2 Prototype AI algorithm

Tasks	<ul style="list-style-type: none"> <li>• Design products data structure</li> <li>• Design characteristics relations</li> <li>• Data entry (products)</li> <li>• Implement AI algorithm</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Outfits generated comply with the characteristics relations established</li> </ul>

<b>#14</b>	Get All Outfits
User story	<b>As an Admin</b> <b>I want</b> to get all the outfits <b>so that</b> I can see all the outfits available into the system
Description	Admin can see all the outfits in the system with their corresponding information. It is important that they can be ordered by each one of their parameters so that it has a better visibility and user experience.
Tasks	<ul style="list-style-type: none"> <li>• Create corresponding page</li> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Admin gets all the outfits</li> <li>• They can be ordered by asc and desc by its parameters</li> </ul>

<b>#15</b>	Get Single Outfit
User story	<b>As an Admin</b> <b>I want</b> to get the information of a specific Outfit <b>so that</b> I can see their full information
Description	Admin will be able to see all the information of the specific Product including the outfits where it is used.
Tasks	<ul style="list-style-type: none"> <li>• Create single outfit page</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• All the information about the outfit: id, style, total price and total likes and dislikes</li> <li>• Related products with their own information are shown</li> </ul>

<b>#16</b>	Edit Outfit
User story	<b>As an Admin</b> <b>I want</b> to edit a specific outfit <b>so that</b> I can change its information if necessary
Description	Admin will be able to edit only the style(s) of an outfit and not the products contained in that outfit. To do so they have to create a new outfit.
Tasks	<ul style="list-style-type: none"> <li>• Create edit page</li> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Edit successfully if and only if all the required information is filled</li> <li>• If some required information cancels the edit and show an error message</li> <li>• Required information: style</li> </ul>

<b>#17</b>	Delete Outfit
User story	<b>As an Admin</b> <b>I want</b> to delete an outfit <b>so that</b> I can manage the system outfits as I prefer.
Description	Admin will be able to delete a specific outfit if they want to. Deleting an outfit breaks the relation with its products but it doesn't delete them.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Outfit deleted from the system</li> <li>• Related products detached</li> </ul>

## Data Analysis

<b>#18</b>	Get Total Sessions
User story	<b>As an Admin</b> <b>I want</b> to get the total number of sessions created <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the total number of sessions created to the moment of the functionality execution.
Tasks	<ul style="list-style-type: none"><li>• Implement functionality</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li><li>• Number calculated at the moment of functionality execution</li></ul>
<b>#19</b>	Get Average Outfits Likeness
User story	<b>As an Admin</b> <b>I want</b> to get the average outfits likeness by the users' <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the average outfits likeness represented as total outfits likes / total outfits dislikes *100 at the moment of functionality execution
Tasks	<ul style="list-style-type: none"><li>• Implement functionality</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li><li>• Average calculated at the moment of function execution</li></ul>
<b>#20</b>	Get Total Products Visited
User story	<b>As an Admin</b> <b>I want</b> to get the total number of products visited <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the total number of products visited, meaning the total number of products the users have visited their official webpage.
Tasks	<ul style="list-style-type: none"><li>• Implement functionality</li></ul>
Acceptance criteria	<ul style="list-style-type: none"><li>• Tasks have to be completed</li><li>• Number calculated at the moment of function execution</li></ul>
<b>#21</b>	Get Total Emails Gathered
User story	<b>As an Admin</b> <b>I want</b> to get the total number of emails gathered <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the total number of emails gathered, meaning the total number of sessions that have filled the email camp in the Mixties Prototype App final form.

Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Number calculated at the moment of function execution</li> </ul>

<b>#22</b>	Get Sessions Geolocations
User story	<b>As an Admin</b> <b>I want</b> to get the sessions' geolocations grouped by region <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the sessions' geolocations grouped by territorial regions and with the number of sessions from each region. It will also provide an interactive map in order to have better visibility and user experience. They are grouped by region and not by country because it is expected to have the biggest part of sessions located in Spain.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Geolocations calculated at the moment of function execution</li> </ul>

<b>#23</b>	Get Average Users' Age
User story	<b>As an Admin</b> <b>I want</b> to get the average users' age <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the average users' age grouped by four categories: <18 years, between 18 – 24, between 25 – 30 and >30 years. Each group with the number of users related
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Ages categories calculated at the moment of function execution</li> </ul>

<b>#24</b>	Get Survey Answers
User story	<b>As an Admin</b> <b>I want</b> to get the users' survey answers <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the users' survey answers grouped by possible answers and the number of users choosing that answer.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> </ul>

- Numbers calculated at the moment of function execution

<b>#25</b>	Get All Suggestions
User story	<b>As an Admin</b> <b>I want</b> to get all the users' suggestions <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the users' suggestions of the ones that have filled the corresponding prototype form camp. Each suggestion will be identified by the user's session.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Suggestions gathered at the moment of function execution</li> </ul>

<b>#26</b>	Get All Session Emails
User story	<b>As an Admin</b> <b>I want</b> to get all the users' emails <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see all the users' emails of the ones that have filled the corresponding prototype form camp. Each email will be identified by the user's session
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Emails gathered at the moment of function execution</li> </ul>

<b>#27</b>	Get Most Liked Outfits
User story	<b>As an Admin</b> <b>I want</b> to get the top 10 most liked outfits <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the top 10 most liked outfits ordered by the most liked one to the least one. Each outfit will contain its basic information and the total likes and dislikes.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Outfits gathered at the moment of function execution</li> </ul>

<b>#28</b>	Get Most Visited Products
User story	<b>As an Admin</b> <b>I want</b> to get the top 10 most visited products <b>so that</b> I have usage insights of the Mixties Prototype App
Description	Admin will be able to see the top 10 most visited products ordered by the most visited one to the least one. Each product will contain its basic information and the total number of visits.
Tasks	<ul style="list-style-type: none"> <li>• Implement functionality</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Tasks have to be completed</li> <li>• Products gathered at the moment of function execution</li> </ul>

## 4.4 Non-functional requirements

### Usability (UX)

<b>#29</b>	Attractive UI
User story	<b>As a user</b> <b>I want</b> an attractive UI design <b>so that</b> I can feel comfortable
Description	Users wants to see a good-looking interface so that they feel comfortable and receive a perception of a professional product.
Acceptance criteria	<ul style="list-style-type: none"> <li>• Average positive answer by the users in the usability test</li> </ul>

<b>#30</b>	Responsive UI
User story	<b>As a user</b> <b>I want</b> a responsive UI design <b>so that</b> I can access to the system through my smartphone, tablet or computer
Description	The interfaces have to be responsive to different screens sizes to allow users access to the system with any kind of device.
Acceptance criteria	<ul style="list-style-type: none"> <li>• All information is clear and well-structured using smartphone, tablet or laptop</li> </ul>

#31	Easy to use UI
User story	<b>As a user</b> <b>I want</b> an easy to use design <b>so that</b> I can understand everything without annoying
Description	We want the users feel comfortable interacting with the system and understands the purpose of each button/part of the interface, so we do not create confusion and discomfort
Acceptance criteria	<ul style="list-style-type: none"> <li>• Average positive answer by the users in the usability test</li> </ul>

## Reliability

#32	Availability 24/7
User story	<b>As a user</b> <b>I want</b> the system to be available all the time <b>so that</b> I can use it whenever I want.
Description	We don't know when users will be accessing to the system, therefore we have to ensure it is available 24/7. Also, it would be a big drawback otherwise, as we could lose a lot of potential value data.
Acceptance criteria	<ul style="list-style-type: none"> <li>• No major interruptions in the system access after one week of its deployment</li> </ul>

## Supportability

#33	Modular system
User story	<b>As a Mixties member</b> <b>I want</b> the system to be modular <b>so that</b> we can iteratively improve it or change it easily.
Description	The system needs to be well-decoupled and ready to attach a new component to it easily.
Acceptance criteria	<ul style="list-style-type: none"> <li>• Each application is deployed in their own infrastructure</li> <li>• No coupling between applications, communication through plain data structures</li> </ul>

#34	Testability
User story	<b>As a Mixties member</b> <b>I want</b> to do software tests <b>so that</b> we can ensure the correctness of our development.
Description	Before deploying the prototype, we have to ensure everything is working as it should, we have to have a test environment with tools like unit tests and integration tests.

Acceptance criteria	<ul style="list-style-type: none"> <li>• Each application can adopt testing frameworks/libraries</li> <li>• Each component of each application can be tested</li> </ul>
---------------------	---

<b>#35</b>	Multiple environments
User story	<b>As a</b> Mixties member <b>I want</b> to have multiple development environments <b>so that</b> we can ensure the correctness of our development.
Description	Multiple environments for our product development like development/production is important to avoid breaking the system while it is in production and to have a more organized structure of our versions. <ul style="list-style-type: none"> <li>• Development environment is used to develop new functionalities and to test its functionalities.</li> <li>• Once everything is working as it should, changes are applied into a final production version into the production environment.</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• There are two environments in each application</li> </ul>

## Legal

<b>#36</b>	General Data Protection Regulation (GDPR)
User story	<b>As a</b> Mixties member <b>I want</b> to fulfill the Europe data protection law <b>so that</b> we provide our users protection and obey the law.
Description	With our system we are treating with personal data such as the users' IP.
Tasks	<ul style="list-style-type: none"> <li>• Create popup message asking for user consent. In case of non-consent user will be able to use the application but IP is not saved nor will be able to fill the form</li> </ul>
Acceptance criteria	<ul style="list-style-type: none"> <li>• Before officially deploying the system, it has to adopt the required methods to comply with the law.</li> </ul>



## 4.5 Mixties Prototype App sequence diagram

The Mixties Prototype App has a small set of functionalities that will be executed sequentially. Therefore, to have a clearer vision about how the communication between the user and the system has to be, here we have a general sequence diagram with the described interaction.

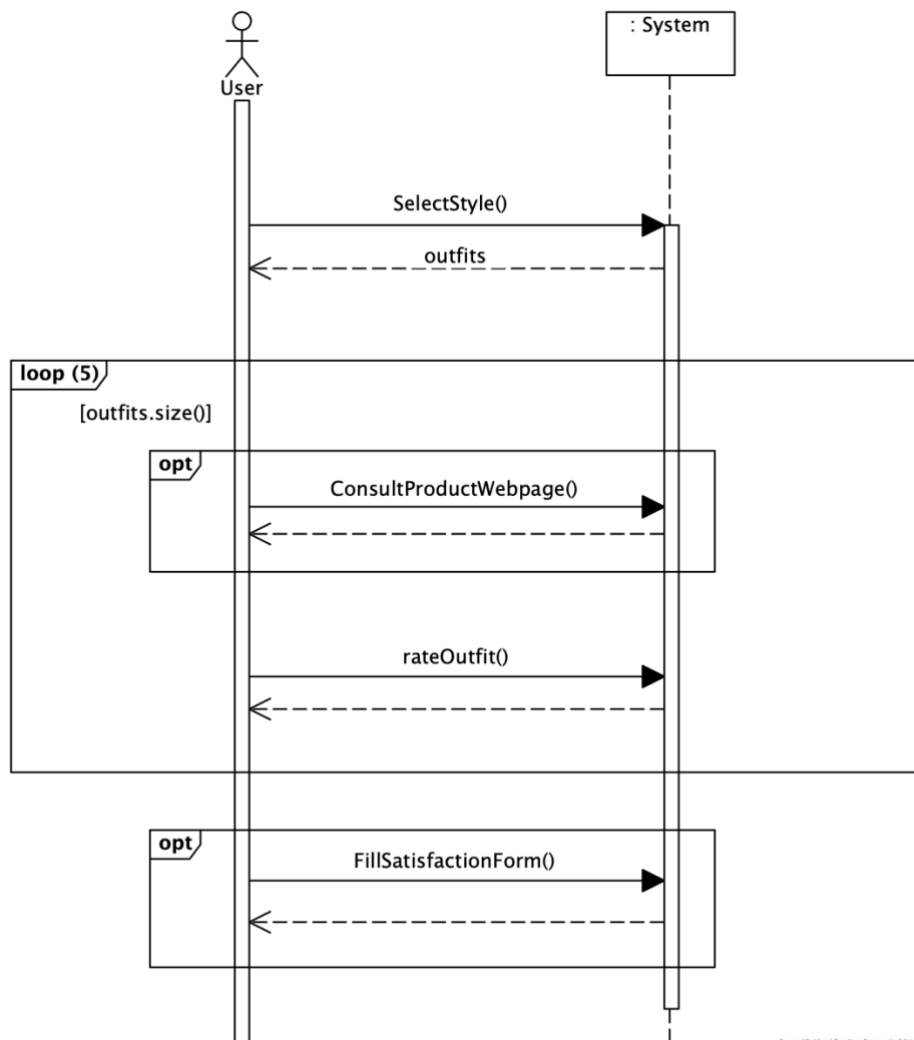


Figure 4.3 Mixties Prototype App general sequence diagram

## 4.6 Domain class diagram

Here we have the class diagram in Figure 4.4 and its enumerations in Figure 4.5. Also, the textual restrictions and the entities descriptions are included in their corresponding sub-sections.

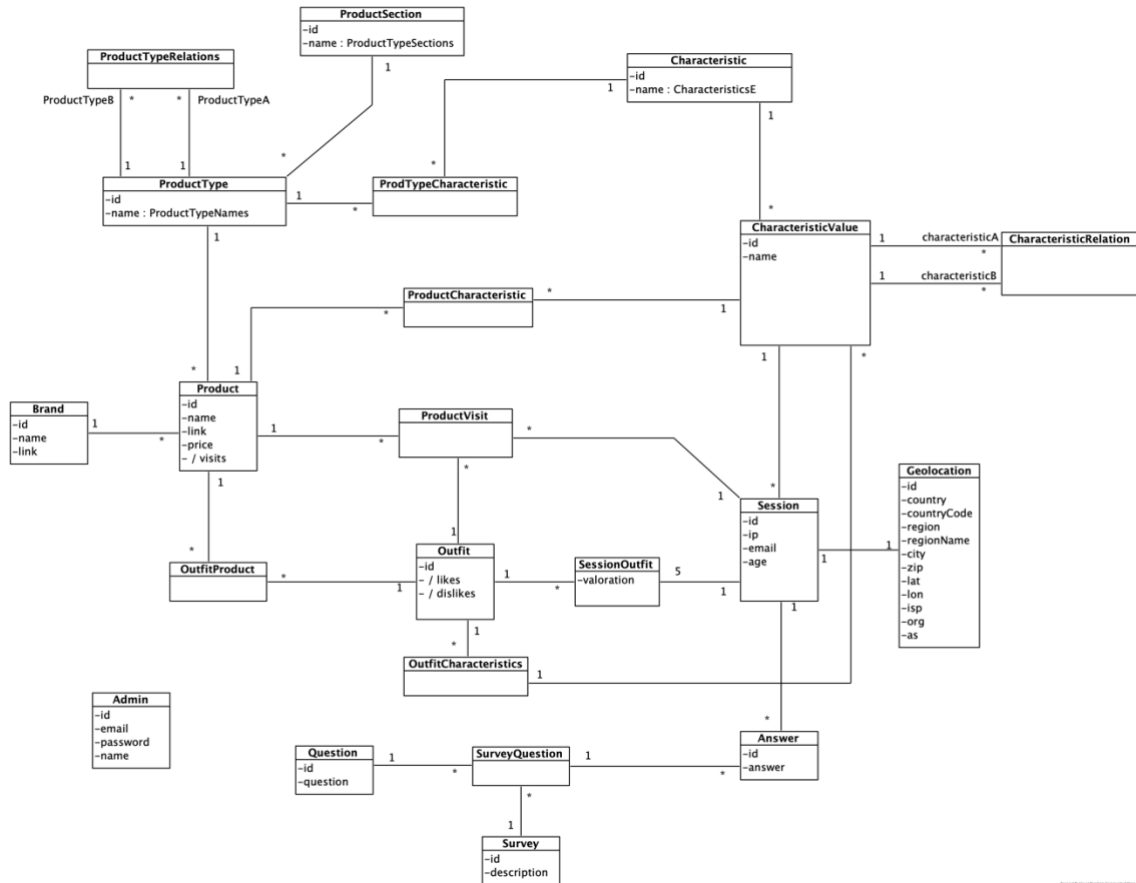


Figure 4.4 UML class diagram

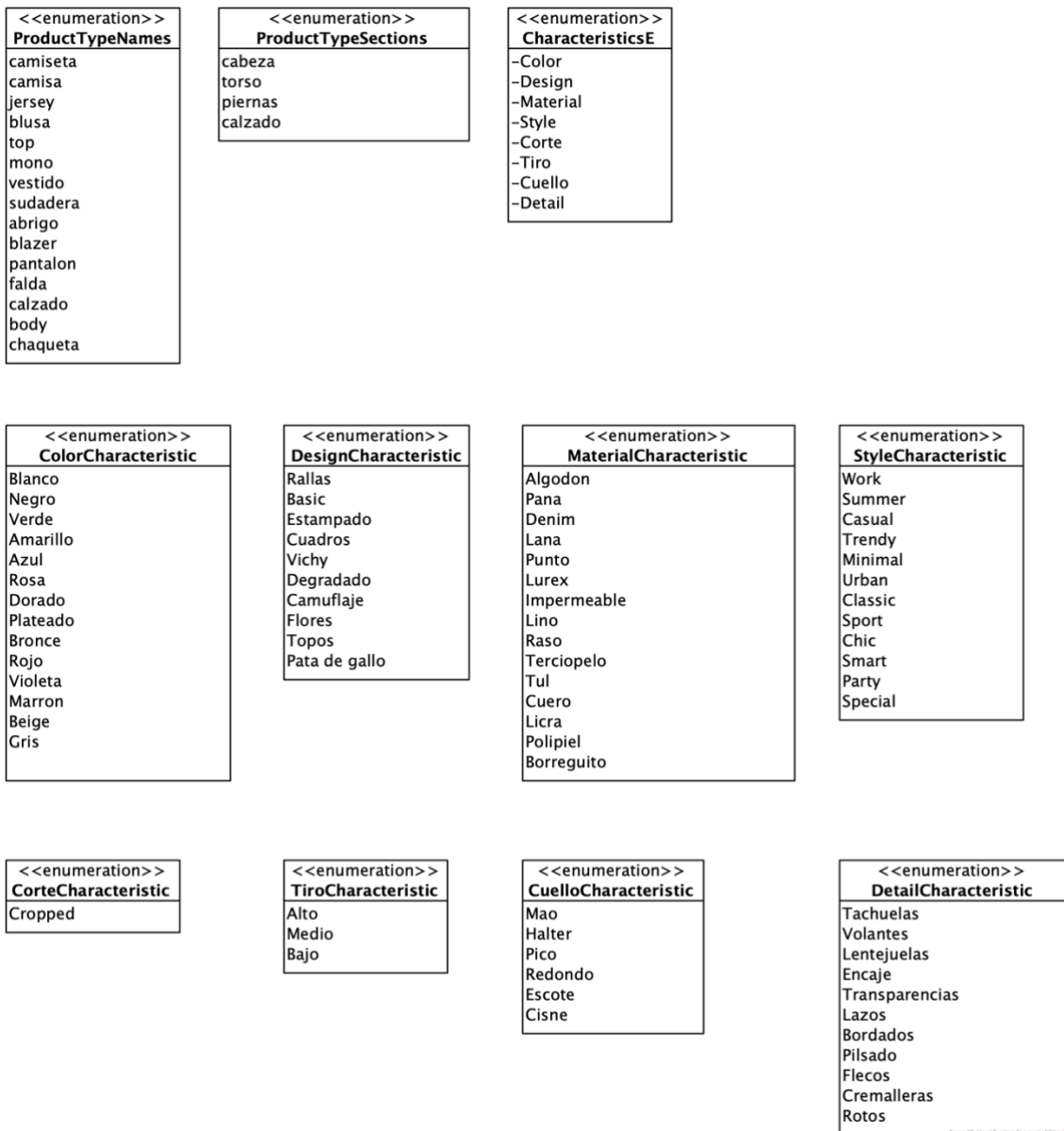


Figure 4.5 Class diagram's enumerations

#### 4.6.1 Textual restrictions

- ProductVisit -> Product has to exist in the ProductVisit -> Outfit -> OutfitProduct -> Product relation
- Session -> ProductVisit-> Outfit has to exist in the Session -> SessionOutfit -> Outfit relation
- Product -> ProductCharacteristics -> CharacteristicValues -> Characteristic must be characteristics that exists in Product -> ProductType -> ProductTypeCharacteristics -> Characteristics relation
- Session -> CharacteristicValues has to exists in the Session -> SessionOutfit -> OutfitCharacteristics -> CharacteristicValues relation

- Outfit's CharacteristicValues and its products' CharacteristicValues are independent from each other.
- ProductTypeRelations -> ProductTypeA cannot be the same as ProductTypeRelations -> productTypeB
- Session -> CharacteristicValue is not related to Session -> ProductVisit -> Product -> ProductCharacteristics -> CharacteristicValue
- CharacteristicRelation -> CharacteristicA can be the same as CharacteristicRelation -> CharacteristicB
- Session-> CharacteristicValue -> Characteristic must be the characteristic with name = style
- Characteristic->CharacteristicValue names are defined in the corresponding enumeration where the Characteristic name is equal as the enumeration

#### 4.6.2 Entities description

##### Admin

System administrators.

Attribute	Description
id	Unique key to identify session
email	Email of the admin
password	Encrypted password of the admin
name	Name of the admin

##### Session

Identifies in an anonymous way the user during the execution of the system.

Attribute	Description
id	Unique key to identify session
ip	Public IP of the device user is accessing to the system
email	Email of the session's user
age	Age of the session's user

##### Geolocation

Geolocation information of the session.

Attribute	Description
Id	Unique key to identify geolocation
country	Country name ("Spain")

countryCode	Country code ("ES")
region	Region code ("MD")
regionName	Region name ("Madrid")
city	City name ("Madrid")
zip	Zip ("28037")
lat	Latitude ("40.4332")
lon	Longitude ("-3.62431")
isp	ISP Name ("M247 Ltd")

## Outfit

Composition of single products that generate an outfit.

Attribute	Description
id	Unique key to identify outfit
/likes	Total likes the outfit has received
/dislikes	Total dislikes the outfit has received

## Product

Piece of clothing.

Attribute	Description
id	Unique key to identify the product
name	Name of the product
link	Link to the official page of the product
price	Price of the product
/visits	Total users that has visited the product in their website

## SessionOutfit

Outfit assigned to a user Session.

Attribute	Description
valuation	User valuation to the outfit (like/dislike)

## ProductVisit

Products that users have visited their official webpage among the outfits assigned to them.

## OutfitProduct

Relates the products that defines an outfit.

## Brand

Brands of the system's products.

Attribute	Description
id	Unique key to identify the brand
name	Name of the brand
link	Link of the brand's website

## ProductType

Type of the product.

Attribute	Description
id	Unique key to identify type
name	Name of the Type (t-shirt, jacket, jeans ...)

## ProductTypeRelations

Relates the product types that combines together, for example ProductTypeA = t-shirt and ProductTypeB = jeans.

## ProductSection

Section of the product type.

Attribute	Description
id	Unique key to identify outfit
name	Name of the Section (upper body, lower body, shoes)

## Characteristic

Characteristics of the products.

Attribute	Description
id	Unique key to identify characteristic
name	Name of the characteristic ("color")

## ProdTypeCharacteristic

Relates the Characteristics that each ProductType have.

## CharacteristicValue

All the values that one characteristic can have.

Attribute	Description
id	Unique key to identify the value
name	Name of the value ("red") if the Characteristic was "color"

## CharacteristicValueRelation

Defines which CharacteristicValues combines with each other, for example, CharacteristicA = "red" combines with CharacteristicB = "white". In this case we are relating two characteristicValues from the same Characteristic = "color" but it is not mandatory.

## ProductCharacteristic

Relates the CharacteristicValues that each Product have.

## OutfitCharacteristics

Defines the characteristics of one outfit.

## Survey

Survey definition.

Attribute	Description
id	Unique key to identify surveys
description	Short description about the survey

## Question

Questions to use in surveys.

Attribute	Description
id	Unique key to identify questions
question	Question text

## Answer

Question's answer by a user's session.

---

Attribute	Description
id	Unique key to identify answers
answer	Answer

---

## SurveyQuestion

Relates a survey's question with an answer.



# 5. Design

## 5.1 General Vision

In the Figure 5.1 we can observe the main components that compose our system.

- **Frontend.** The two applications, Mixties Prototype App and the Admin Dashboard. They communicate with the Backend in order to store and retrieve data.
- **Backend.** Where all the business logic is implemented. It provides a REST API [10] to communicate with the two applications and it also interacts with external services. Finally, responsible to retrieve and save information into the databases.
- **External Services.** Google Maps API [11] and ip-api.com [12] external service to gather the users' IP geolocation information.

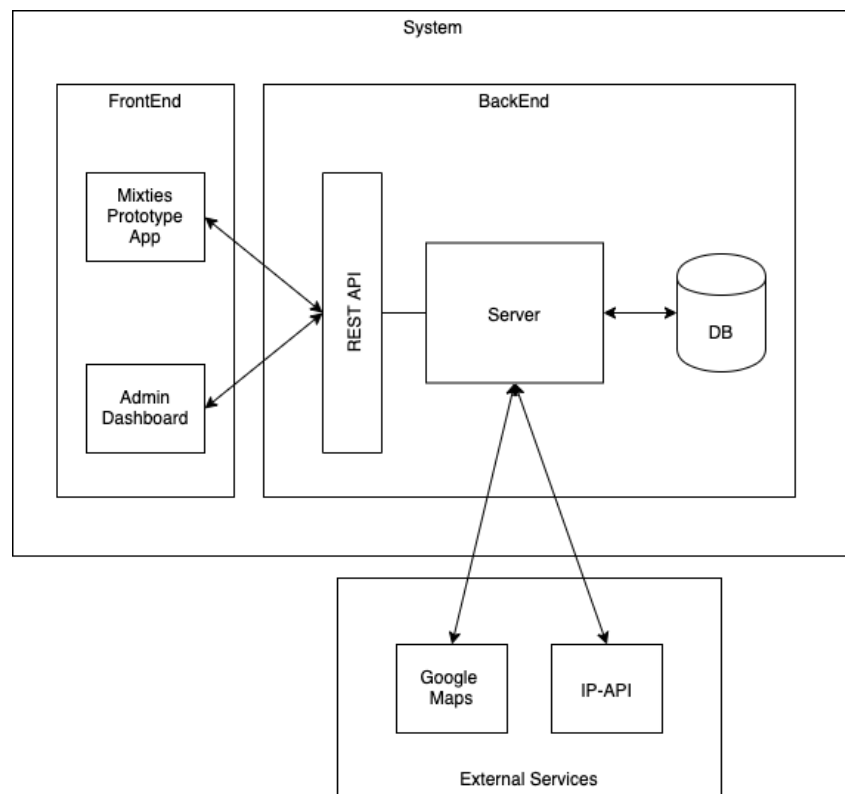


Figure 5.1 General Vision Diagram

The communication between the Frontend and the Backend is done by REST API calls. It provides us a simple interface to access all the different resources. We are going to see it more in depth in their corresponding section.

On the other side, the server communicates to both external services using API calls too. So, in general, the system has a homogeneous way to communicate between components.

## 5.2 Graphical User Interface (GUI)

We have two different interfaces, one for the Mixties Prototype App and another for the Admin Dashboard. Each one of them has different purposes and therefore their design is completely different.

The Mixties Prototype App is meant to be designed solely for smartphone devices, whereas the Admin Dashboard the opposite. This difference resides on the fact that we want and expect the prototype app to be consumed from such devices. On the other side, regarding the Admin Dashboard app, as it contains tables, graphs and maps, it is designed to be consumed from normal laptops. Although, as defined in the requirements section, both interfaces have to adopt a responsive design and adapt to any kind of device. Now some screenshots are provided for each application in their corresponding sub-section.

### 5.2.1 Mixties Prototype App (GUI)



Figure 5.2 Welcome page



Figure 5.3 Welcome page

II



Figure 5.4 Welcome page

III

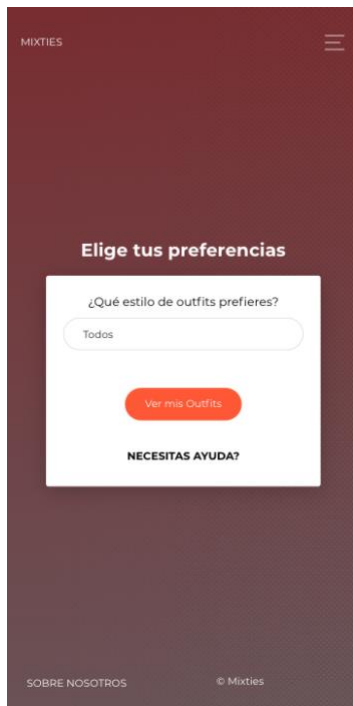


Figure 5.5 Styles preference page

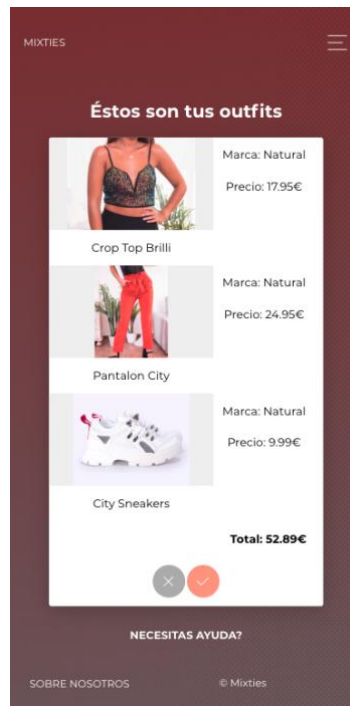


Figure 5.6 Outfit page



Figure 5.7 Form page



Figure 5.8 Form page II

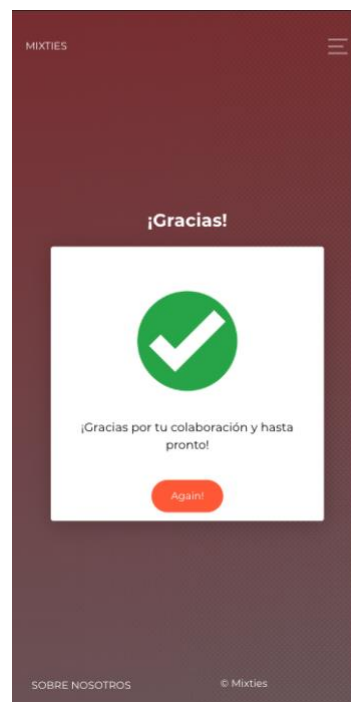


Figure 5.9 End page

## 5.2.2 Admin Dashboard (GUI)

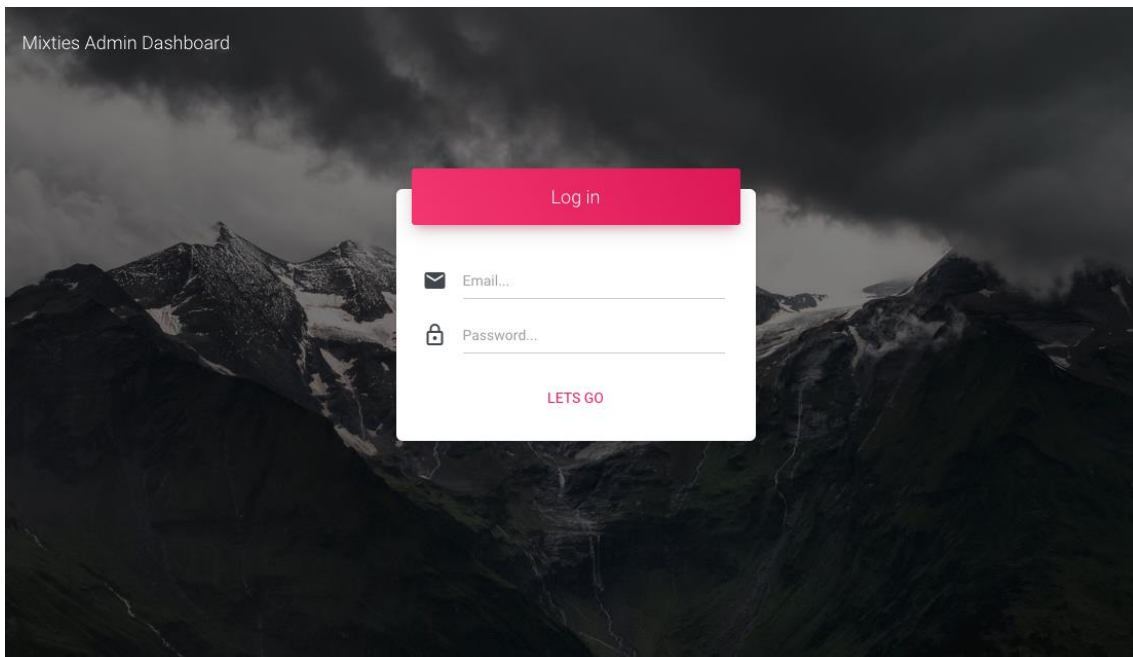


Figure 5.10 Login page

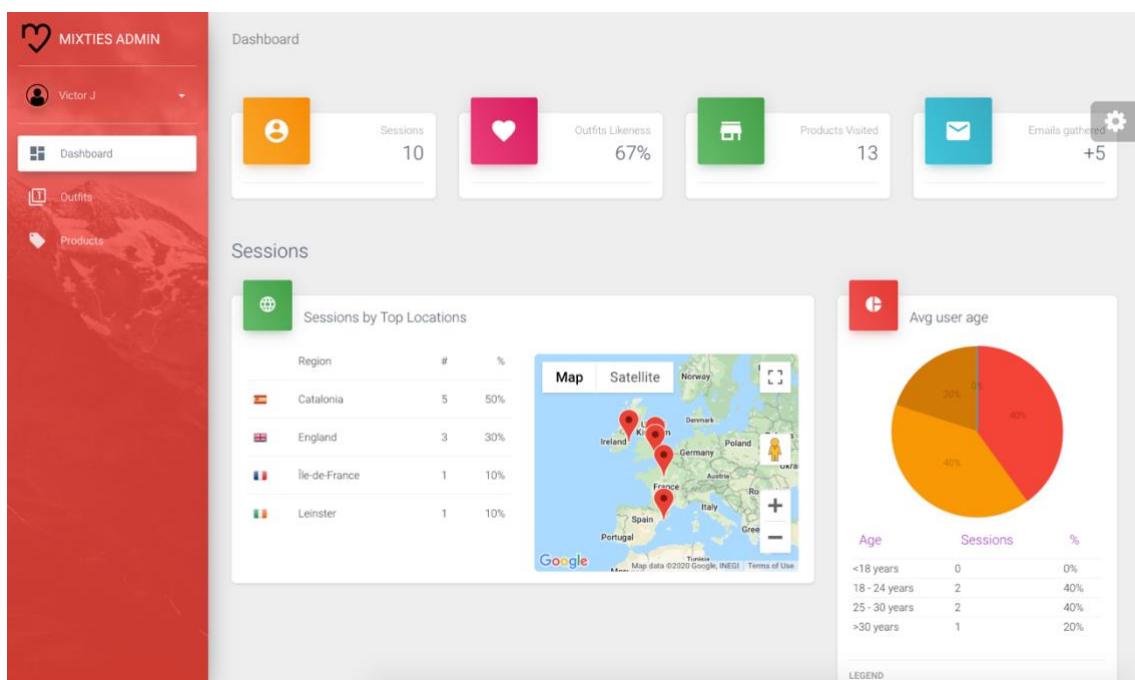


Figure 5.11 Main dashboard page

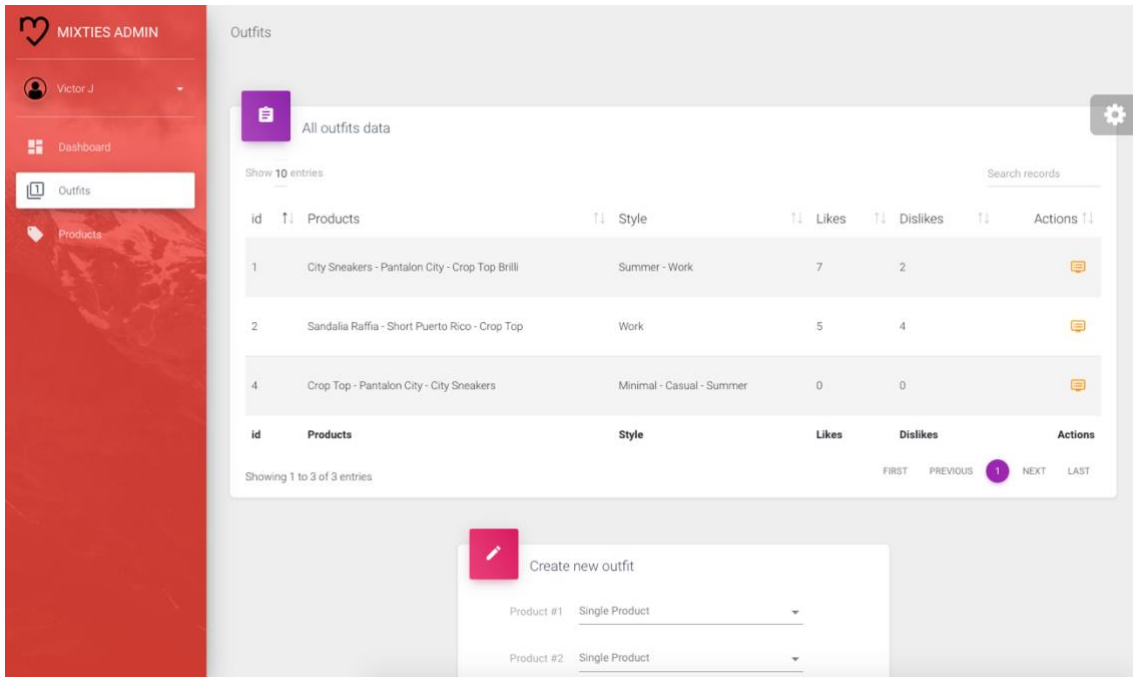


Figure 5.12 Outfits page

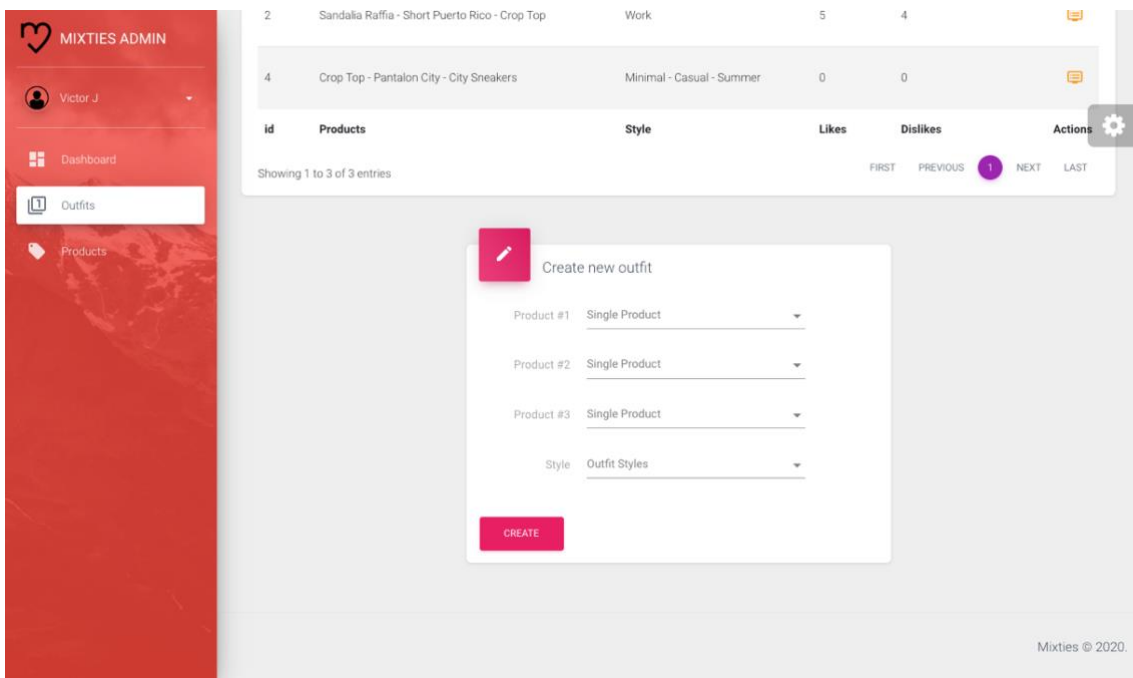


Figure 5.13 Outfits page II

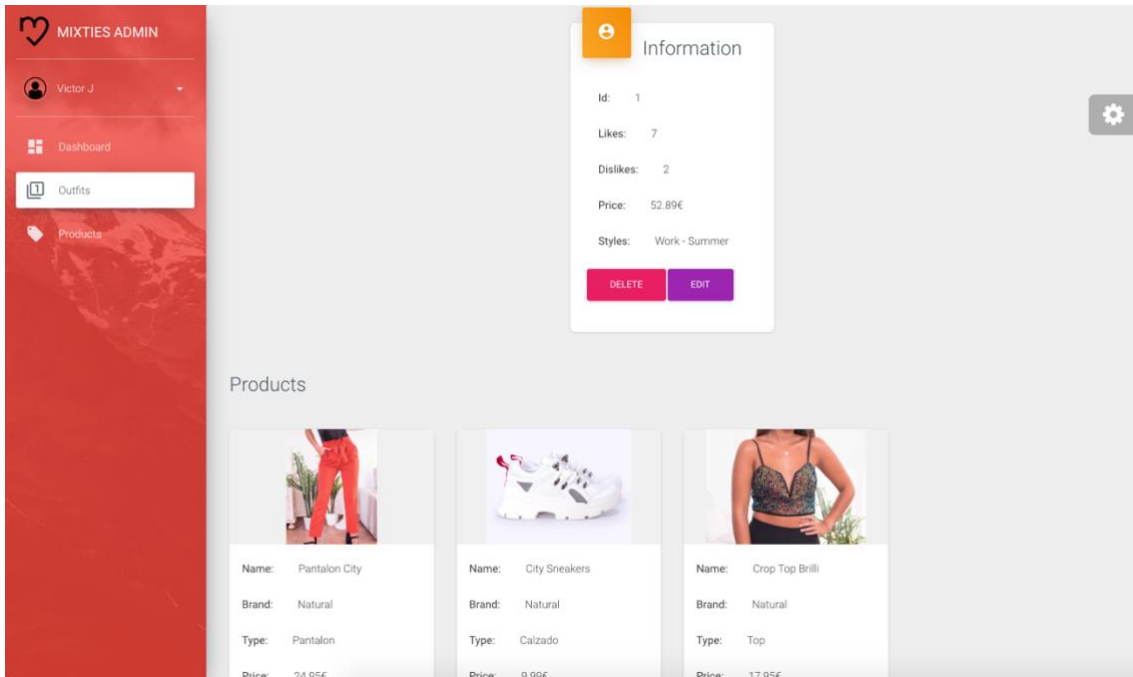


Figure 5.14 Single outfit page

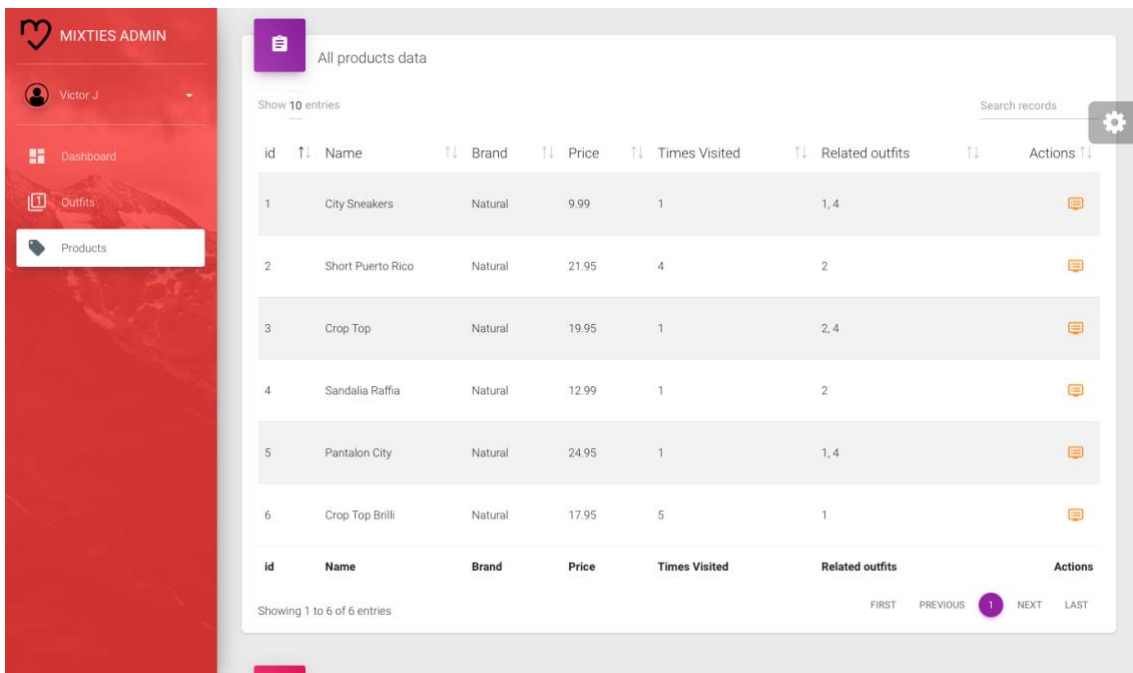


Figure 5.15 Products page

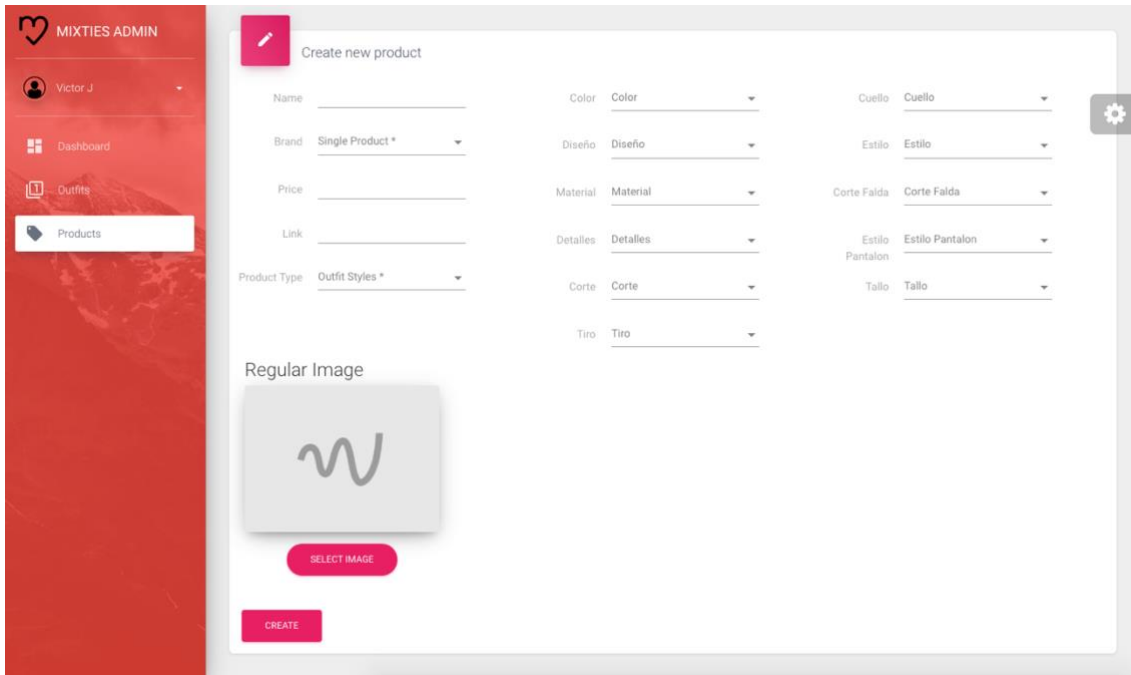


Figure 5.16 Products page II

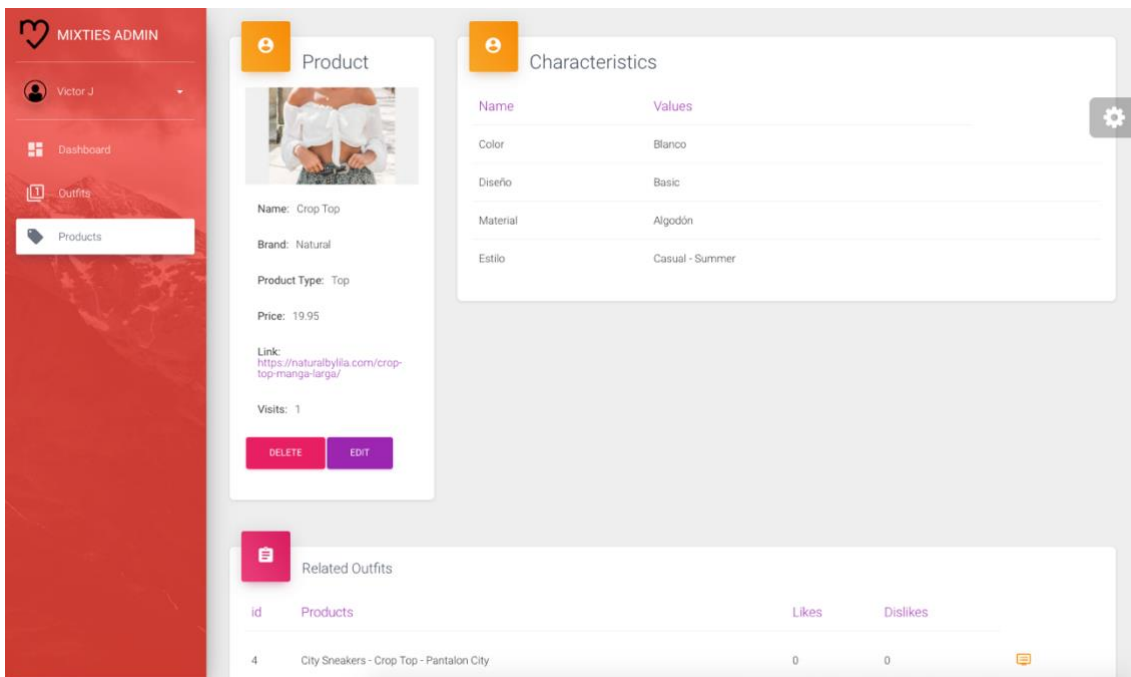


Figure 5.17 Single product page

### 5.3 Frontend

At the frontend side of the system, both the prototype and the admin dashboard applications are Single Page Applications (SPA) and the specific web framework used for their development is Angular [13]. This framework has the architecture shown at Figure 5.18 and the main parts are:

- **Module** contains a group of components under the same logical business set of functionalities, for example, in an ERP system they would be HR module, Financial, etc.
- **Component** define a class that contains application data and logic, is associated with an HTML template that defines a view to be displayed in a target environment.
- **Template** combines HTML with Angular markup that can modify HTML elements before they are displayed.
- **Service** For data or logic that isn't associated with a specific view, and that you want to share across components. In our case they are mainly used to manage the communication with the backend, specified in the Figure 5.19.

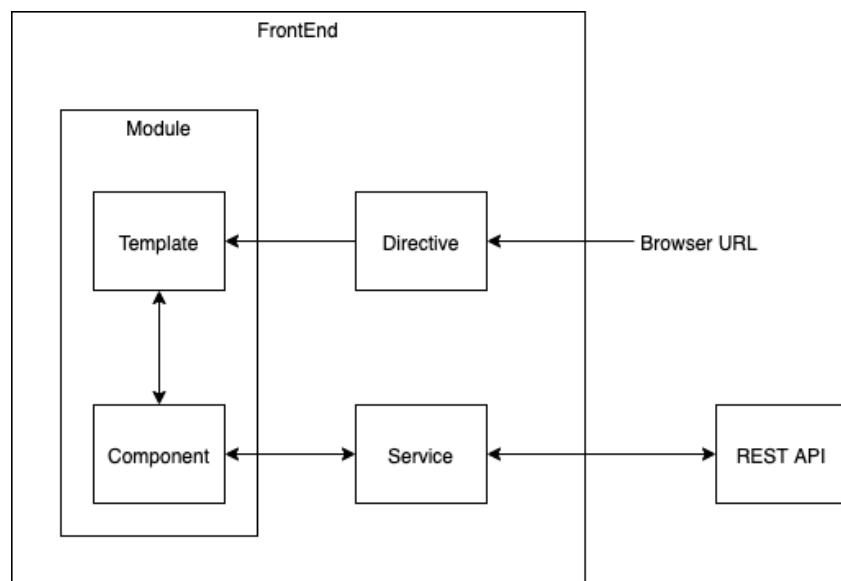


Figure 5.18 Frontend Architecture

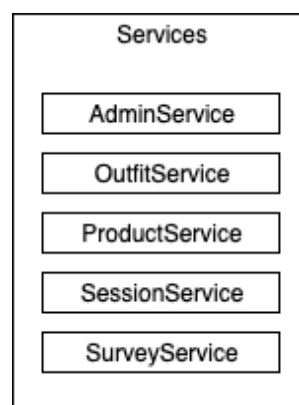


Figure 5.19 Frontend services



## 5.4 Backend

Regarding the backend, as we have seen it consists of a REST API that receives the petitions from the two applications and map them to the corresponding Controller functionalities.

In order to implement it, we have used ExpressJS [14], a JavaScript framework for NodeJS [15] runtime environment. As its documentation defines, the recommended architecture is the one that we can see at Figure 5.20 and the main components are:

- **REST API Routes.** They define the API url endpoints to be consumed by the frontend applications and maps the requests to the corresponding functionalities.
- **Helpers.** Where authentication and data validation rules are defined. They act as a middleware between a route and the corresponding controller functions to ensure their preconditions.
- **Controllers.** They implement all business logic functionalities, triggered by the routes. After their execution they generate and send the corresponding response back to the client.
- **Models.** Data models' definitions used by controllers to get/create/modify data objects from the database.
- **ORM.** Object-Relational Mapping that provides us the tools to be able to create the data models from the actual database and map the communication between them.

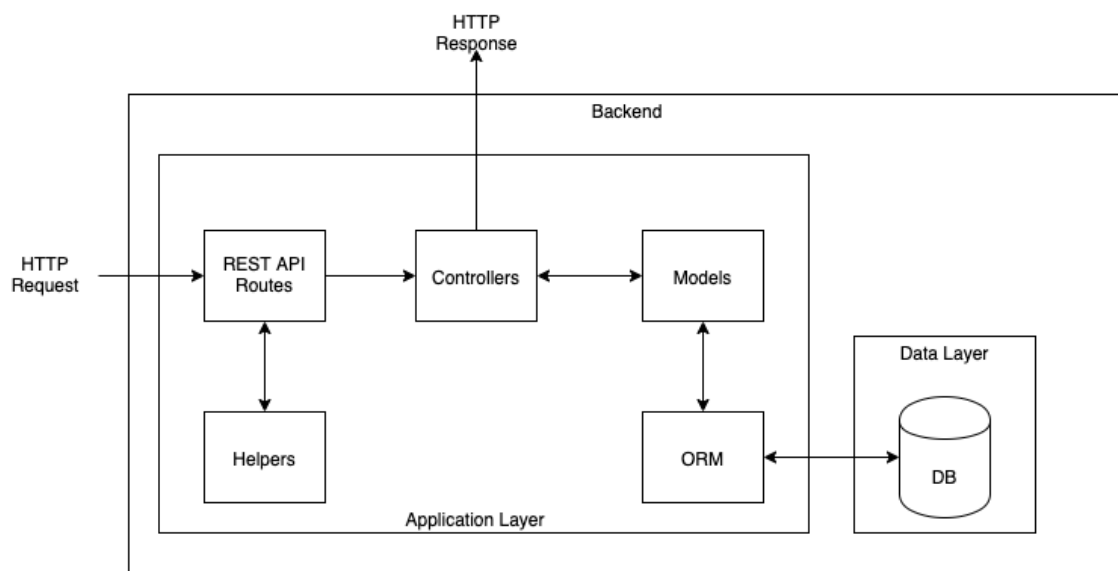


Figure 5.20 Backend Architecture

This architecture applied to our domain leaves us with the structure shown at the following figure.

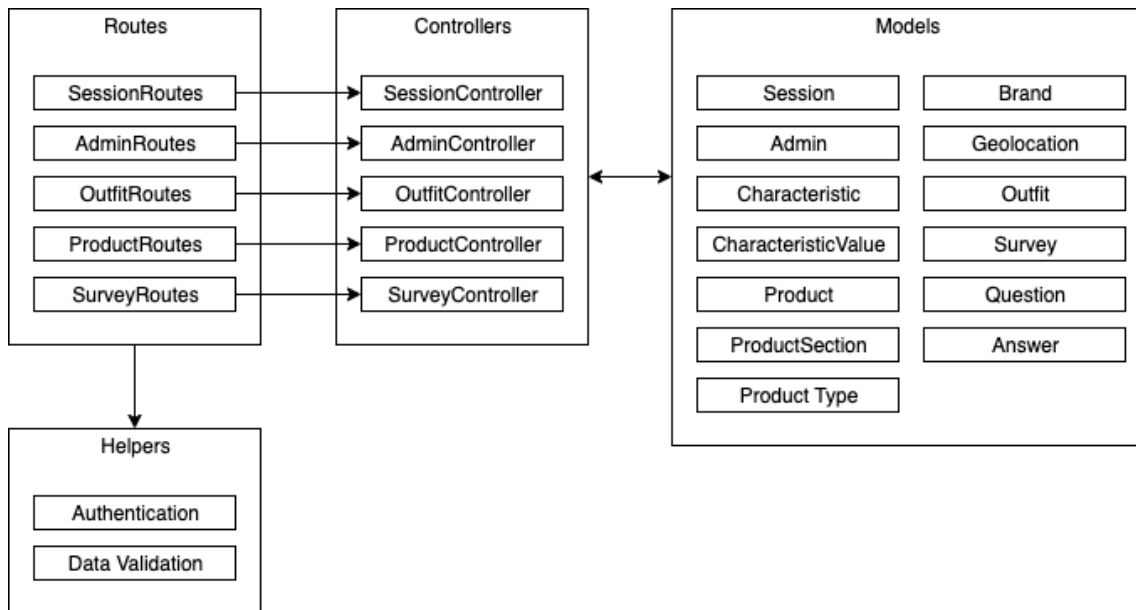


Figure 5.21 Backend detail architecture

#### 5.4.1 Middleware pattern

Middleware functions [16] are functions that have access to the request object (req), the response object (res), and the next function in the application's request-response cycle. The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware (see Figure 5.22).

In our case is a very useful pattern for our helpers' methods, specifically for authentication and data validation processes. It allows us to stop the execution of the functionalities if anything turns out different than expected. For example, if the client provides bad authentication credentials, the authentication process will throw an error and the middleware will stop the execution and return the error to the client.

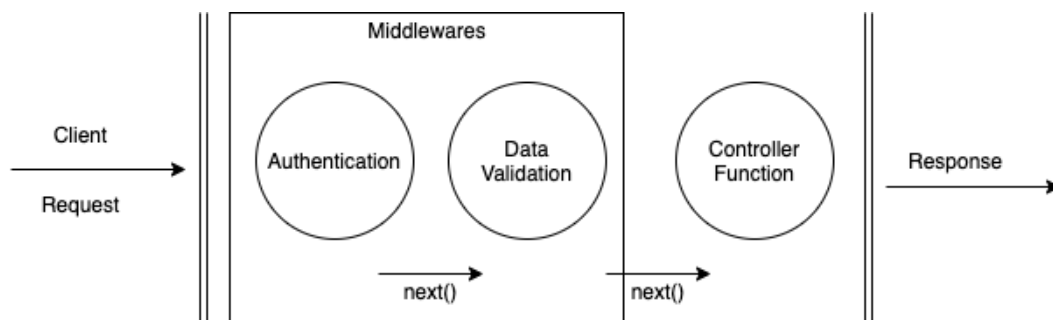


Figure 5.22 Middleware pattern

## 5.4.2 Authentication

System requires an authentication method in order to protect its data and restrict the access as preferred. As defined in the requirements section we have to distinguish between two different roles inside the system: Users and Admins.

For this purpose, we are using Json Web Tokens (JWT) [17] and it works as the following figure shows

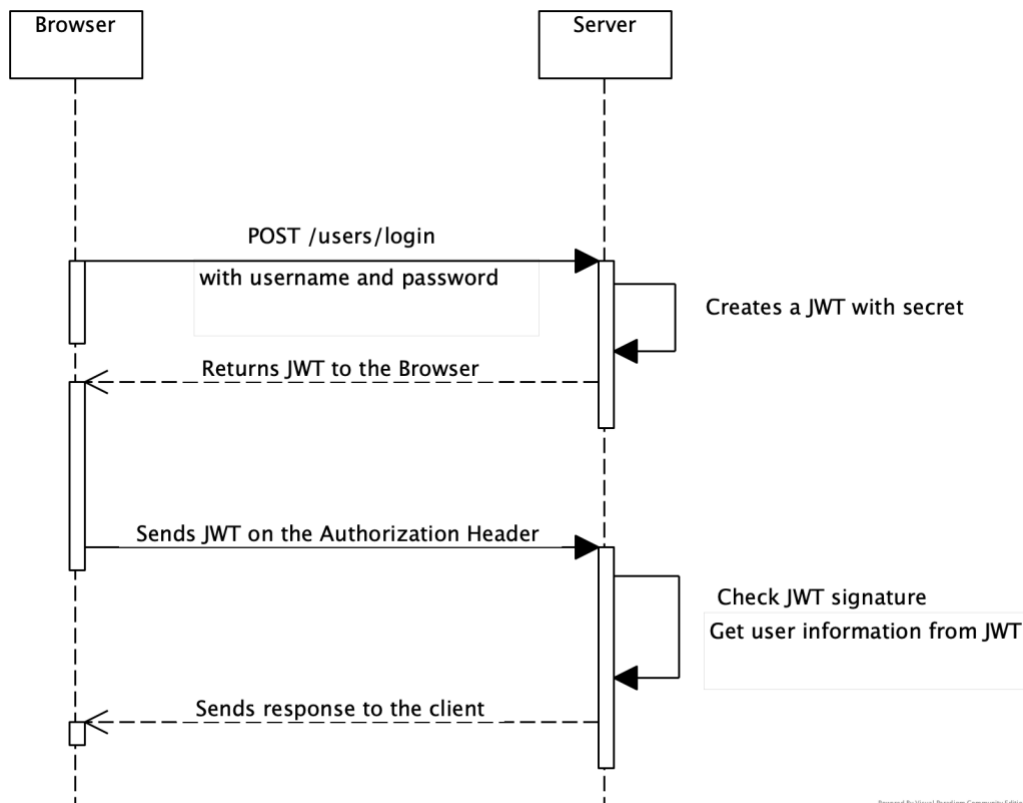


Figure 5.23 JWT authentication

Powered By Visual Paradigm Community Edition

# 5.5 Database logical schema

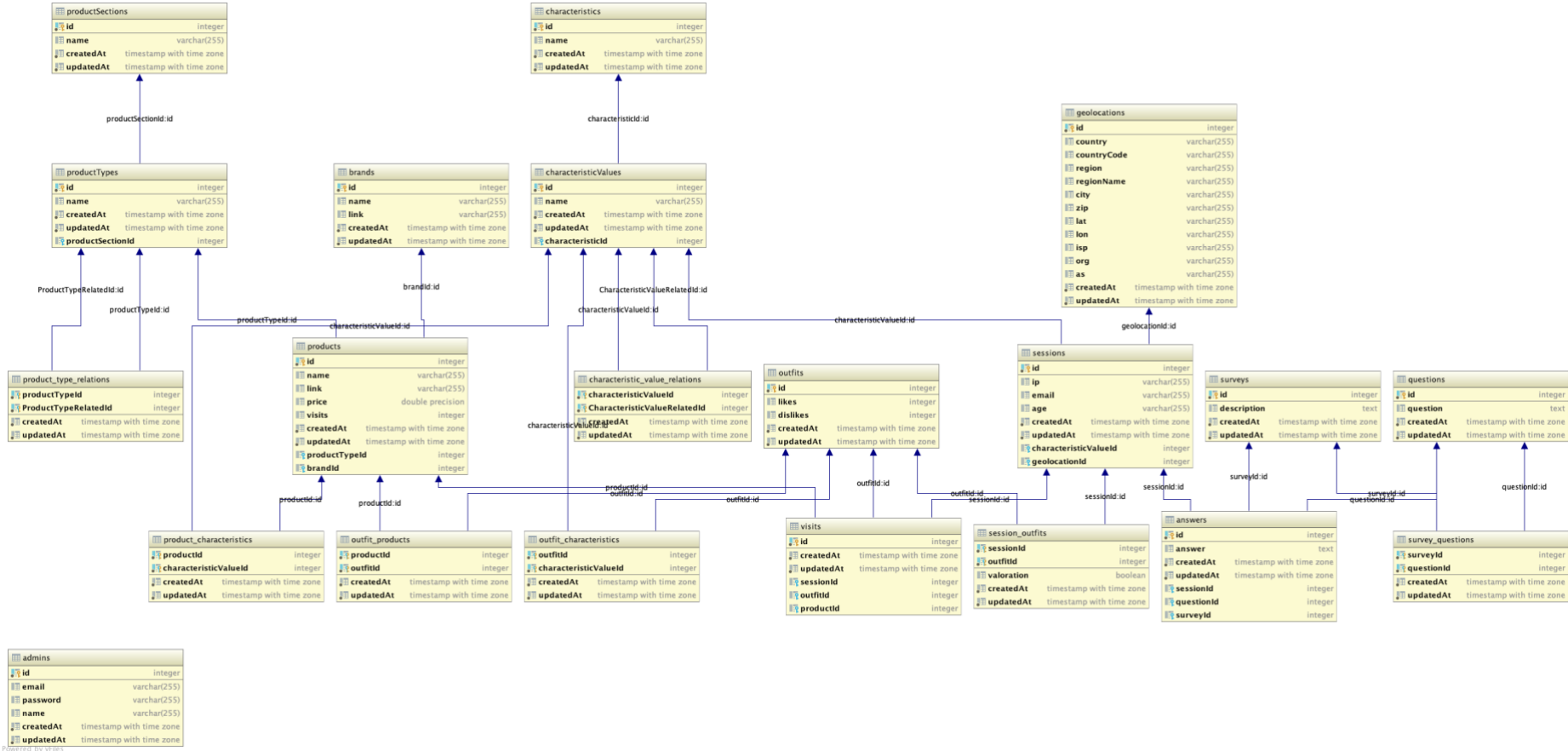


Figure 5.24 Database ER diagram

## 5.6 Infrastructure design and technologies

Now we are going to explain the infrastructure we have applied in order to hold the architecture and be able to deploy it.

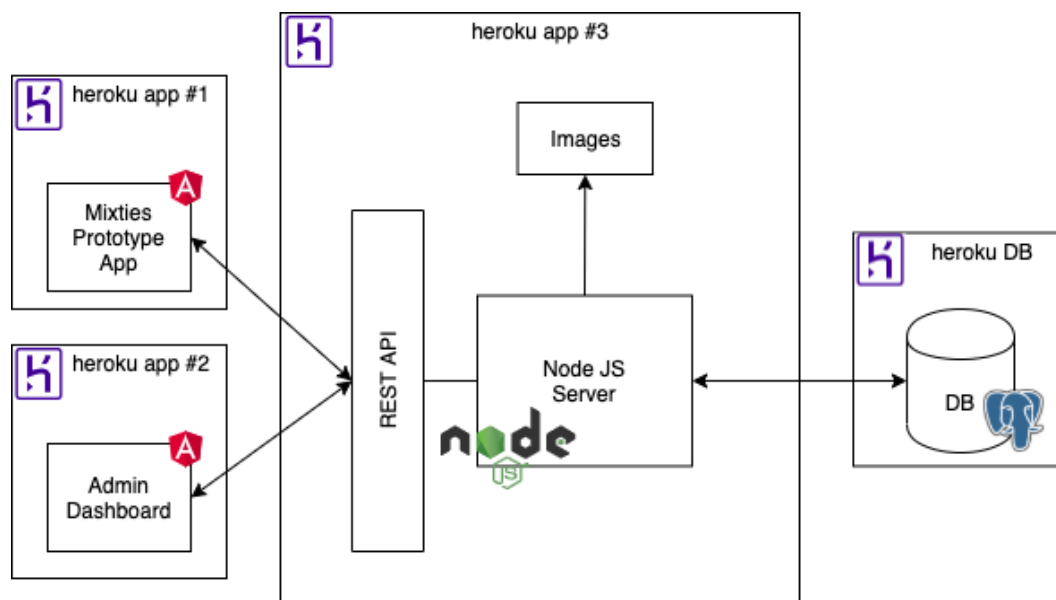


Figure 5.25 Infrastructure components diagram

As we can see in the above figure, we have implemented a decoupled infrastructure that consist of three different applications deployed all of them in the Heroku cloud platform [18]:

- **Heroku app #1.** Containing the Mixties Prototype App application  
url: <https://mixties-prototype.herokuapp.com/>
- **Heroku app #2.** Containing the Admin Dashboard application  
url: <https://mixties-dashboard.herokuapp.com/>
- **Heroku app #3.** Containing the backend for the whole system.  
url: <https://mixties-api.herokuapp.com/>

This infrastructure provides us the modular system required as each application is independently deployed and allow us to develop them separately, without compromising the others.

One of the possible future scenarios is that we have to drop the Mixties Prototype App entirely and develop a completely different prototype. Another possibility would be that we have to keep the current prototype and attach a new one into the current system. All these situations are easy to accomplish with this infrastructure.

At the following figure, we can observe a possible future scenario of the system which adds a NoSQL database into the system (MongoDB [19]) and a new prototype developed with React framework [20] instead of Angular.

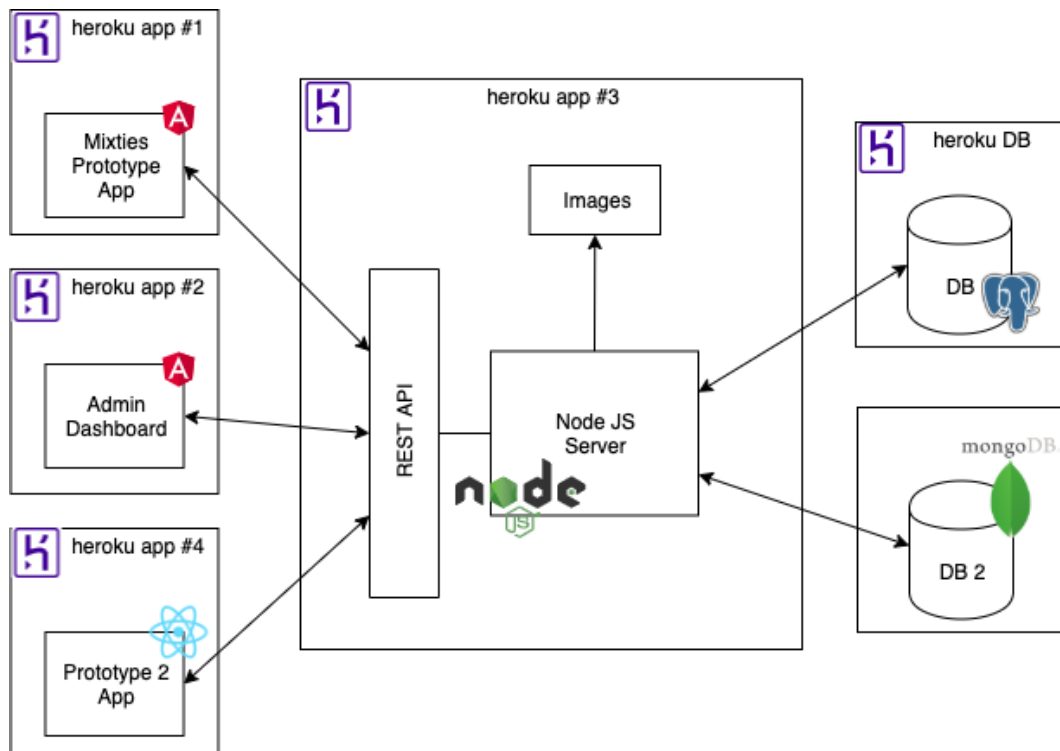


Figure 5.26 Future system scenario

### 5.6.1 Technologies: which ones and why

<b>Technology</b>	Angular [13]
<b>Description</b>	JavaScript framework used to create single page applications (applications that have only one page and change the content dynamically, so the client doesn't have to be constantly loading new pages from the server). We have used it to develop the GUI
<b>Justification</b>	<ul style="list-style-type: none"><li>• Popular with a lot of community support</li><li>• Powerful framework with a lot of tools</li><li>• Quick and easy to develop interfaces</li><li>• Personal preference, after trying alternatives like ReactJS this one feels easier</li></ul>

<b>Technology</b>	NodeJS [15]
<b>Description</b>	NodeJS is a runtime environment designed to build scalable network applications. We have used NodeJS to develop our server, the business logic layer.
<b>Justification</b>	<ul style="list-style-type: none"><li>• Uses JavaScript, same as frontend technology Angular.</li><li>• Personal preference, already familiar with this technology</li><li>• Goes along with API development</li></ul>

<b>Technology</b>	ExpressJS [14]
<b>Description</b>	ExpressJS is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.
<b>Justification</b>	<ul style="list-style-type: none"><li>• Most used, documented and supported web application framework for NodeJS</li></ul>

<b>Technology</b>	Joi [21]
<b>Description</b>	Joi is a schema description language and data validator for JavaScript.
<b>Justification</b>	<ul style="list-style-type: none"><li>• Well documented and easy to use</li></ul>

<b>Technology</b>	Passport.js [22]
<b>Description</b>	Passport is an authentication middleware for Node.js. Used to implement the JWT authentication
<b>Justification</b>	<ul style="list-style-type: none"><li>• Most used and supported authentication library for JavaScript</li></ul>

<b>Technology</b>	Multer [23]
<b>Description</b>	Node.js middleware for handling multipart/form-data, which is primarily used for uploading files. We have used this middleware to handle the image uploading process.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Most used and documented middleware for file uploading in Node.js</li> </ul>

<b>Technology</b>	Jest [24]
<b>Description</b>	JavaScript testing framework.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Minimal configuration</li> <li>• Has in-built test-runner, assertion library and mocking support</li> </ul>

<b>Technology</b>	Supertest [25]
<b>Description</b>	A library for testing Node.js HTTP servers. It enables to programmatically send HTTP requests to HTTP servers and get results.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Goes along with Jest</li> <li>• Personal preference</li> </ul>

<b>Technology</b>	Sequelize [26]
<b>Description</b>	Sequelize is a Node.js ORM for Postgres, MySQL, MariaDB, SQLite and Microsoft SQL.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Most used Node.js ORM for Postgres DB</li> </ul>

<b>Technology</b>	Json Web Tokens (JWT) [17]
<b>Description</b>	JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Easy to use</li> <li>• High amount of documentation</li> </ul>

<b>Technology</b>	PostgreSQL [27]
<b>Description</b>	PostgreSQL is a powerful, open source object-relational database system. We have used to store all our structured data information.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Personal preference, familiar with it.</li> </ul>

<b>Technology</b>	Heroku [18]
<b>Description</b>	Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.
<b>Justification</b>	<ul style="list-style-type: none"> <li>• Personal preference, familiar with it.</li> </ul>



## 6. Implementation

In this section we are going to put real examples of our code implementation of both the frontend and backend parts. It's intended to show only the basic idea behind each of the components and not to show the full implementation.

### 6.1 Frontend

Here we are going to introduce the implementation of the different parts of the Angular architecture showed in Figure 5.18 at page 71. Specifically, we are showing the implementation of getting an outfit to show to a user in the Mixties Prototype App (see Figure 5.6 at page 66). Note that even though it is real code from the application, it is only a small part of it to show just the important things.

#### 6.1.1 Template

First of all, in the following figure we can see the **Template**, it's based on HTML code together with variables' values coming from the corresponding Component. The strategy to insert Component variables into the Template is called **Interpolation** and it is done using curly braces: `{{component.variable}}`. As we can observe in this example (figure below), interpolation is used in order to insert information about the outfit and its products, such as their brand and price.

```

<div>
  <div class="card-body text-center" style="margin-top: 0px;">
    <p class="card-text" >Marca: {{outfit.top.brand.name}}</p>
    <p class="card-text" >Precio: {{outfit.top.price}}€</p>
  </div>
  <div class="card-body text-center" style="margin-top: 71px">
    <p class="card-text" >Marca: {{outfit.mid.brand.name}}</p>
    <p class="card-text" >Precio: {{outfit.mid.price}}€</p>
  </div>
  <div class="card-body text-center" style="margin-top: 71px">
    <p class="card-text" >Marca: {{outfit.bot.brand.name}}</p>
    <p class="card-text" >Precio: {{outfit.bot.price}}€</p>
  </div>
  <div class="card-body text-center" style="margin-top: 71px">
    <p class="card-text" ><b>Total: {{totalPrice()}}€</b></p>
  </div>
  <button (click)="nextOutfit(false)" class="btn disabled">
    <i class="now-ui-icons ui-1_simple-remove"></i>
  </button>
  <button (click)="nextOutfit(true)" class="btn disabled">
    <i class="now-ui-icons ui-1_check"></i>
  </button>
</div>

```

*Figure 6.1 Angular Template*

## 6.1.2 Component

Secondly, in the Figure 6.2 we have the **Component** implementation of the same functionality, so we can see how the interpolated data in the Template is processed in the Component side.

As we can see, the component implements `OnInit`, and then defines the `ngOnInit()` function. This function is the first one executed at the component creation. Therefore, in this example, `loadOutfits()` is executed as soon as the component is created.

Load outfits function is meant to load all the five outfits to show to the specific user currently using the Mixties Prototype App. To do so, it needs to gather the outfits information from the backend, and this is the `_outfitsService` responsibility. After gathering the outfits, they are saved at the “outfits” array, and the first outfit to show is saved at the “outfit” variable. Notice that this last variable is the interpolated one into the Template showed before.

```

export class OutfitCardComponent implements OnInit {

  constructor(private router: Router,
              private _outfitsService: OutfitsService,
              private modalService: NgbModal) { }

  ngOnInit() {
    this.loadOutfits();
  }

  loadOutfits(){
    this._outfitsService.getOutfits().subscribe(data => {
      this.outfits = data;
      this.outfit = this.structureOutfit(this.outfits[0]);
      this.loading = false;
    });
  }
}

```

*Figure 6.2 Angular Component*

### 6.1.3 Service

Thirdly, in the Figure 6.3, we have the implementation of the OutfitsService, where the connection with the backend is done. Here we can see the getOutfits() function called by the component showed before. What it does is generating an HTTP call to the corresponding backend endpoint. To do so, it needs to interact with two other services. On the one hand, with the \_constantsService which stores constant variables shared among all the application, in our case it is used to get the base API url. On the other hand, it interacts with the \_sessionService in order to get the currently session JWT token needed to identify and authorize the user into the backend.

```

export class OutfitsService {
  constructor(private http: HttpClient,
              private _constantsService: ConstantsService,
              private _sessionService: SessionService) {
  }

  getOutfits(): Observable<IOutfit[]>{
    const url = this._constantsService.baseAPIurl;+'/outfits/assign-session'
    return this.http.post<IOutfit[]>(url, {}, {
      headers: {'Authorization': this._sessionService.getSessionID()}
    });
  }
}

```

*Figure 6.3 Angular Service*

Lastly, this would be the main structure repeated to all the other frontend functionalities from both the Mixties Prototype App and the Admin Dashboard.

## 6.2 Backend

At this section we are going to see the implementation of the Routes, Controller and Model components of the backend shown in the Figure 5.20 at page 72.

We will keep using the same example seen at the Frontend implementation section about getting the five user's outfits.

### 6.2.1 Routes

In the Figure 6.5 we can observe all the routes related to outfits. They define for each url endpoint and HTTP method (GET, HEAD, POST, PUT, DELETE) which controller functionality has to be executed.

To clarify how the structure of a route is, in the Figure 6.4 we have all the information contained in the route `'/outfits/assign-session'`, the one that we have seen in the Frontend implementation.

First of all, the HTTP method and the url endpoints are defined, after this, all the following functions separated by commas are executed from left to right, one after the other and stopping the execution if any error arises. This is the middleware pattern explained in section 5.4.1 at page 73. As we can see, before executing the controller function, it checks it is an authorized user and then validates the data inserted into the body of the HTTP request, then and only then, the controller function is executed.



*Figure 6.4 Backend route structure*

```

app.get('/outfits/styles', outfits.getAllStyles);
app.get('/outfits', passportAdmin, outfits.getAllOutfits);
app.post('/outfits', passportAdmin, validateBody(schemas.outfitSchema), outfits.createOutfit);
app.post('/outfits/:id/rate', passportUser, validateParam(schemas.idSchema, 'id'),
validateBody(schemas.rateSchema),outfits.rateOutfit);
app.post('/outfits/assign-session', passportUser, validateBody(schemas.assignSessionSchema),
outfits.assignSession);
app.get('/outfits/total-rates', passportAdmin, outfits.getTotalRates);
app.get('/outfits/most-liked', passportAdmin, outfits.getTopOutfits);
app.get('/outfits/:id', passportAdmin, validateParam(schemas.idSchema, 'id') ,outfits.getOutfitById);
app.post('/outfits/:outfitId/products/:productId/visit', passportUser, validateParam(schemas.idSchema,
'outfitId'), validateParam(schemas.idSchema, 'productId'), products.visitProduct);
app.delete('/outfits/:id', passportAdmin, validateParam(schemas.idSchema, 'id'), outfits.deleteOutfit);
app.put('/outfits/:id', passportAdmin, validateParam(schemas.idSchema, 'id'),
validateBody(schemas.outfitSchema), outfits.editOutfit);

```

*Figure 6.5 Backend outfits' routes*

## 6.2.2 Controller

Here we are going to see the implementation of the assignSession() functionality of the Outfit controller. We can see it at the Figure 6.6 and what it does basically is: first it loads five outfits of the style the user has selected. Then, assigns them into the user and finally it returns these outfits as a HTTP response.

In order to load the five outfits from the database, it is defined a query using the Sequelize ORM syntax. After waiting for this process to complete, as it is an asynchronous call to the database, each outfit is assigned to the user's session and finally returned as a HTTP response.

```

const assignSession = async (req, res) => {
  session = req.user;
  const outfits = await Outfit.findAll({
    limit: 5,
    attributes: ['id'],
    include: [{
      attributes: [],
      model: CharacteristicValue,
      where: {id: session.characteristicValueId}
    },{
      model: Product,
      include: [{
        attributes: ['id', 'name'],
        model: ProductType,
        include: [{
          attributes: ['id', 'name'],
          model: ProductSection
        }]
      }]
    },{
      attributes: ['id', 'name', 'link'],
      model: Brand
    }]
  ]}
  });

  outfits.forEach(outfit => {
    session.addOutfit(outfit);
  });

  res.status(200).json(outfits);
}

```

*Figure 6.6 Backend controller*

### 6.2.3 Model

Now we are going to see the model definition of the outfits in the backend. see Figure 6.7. There are only two parameters: like and dislike. Both are set as integers and initialized to 0.

Notice that there is no id parameter, this is because Sequelize does it internally. Also that there is no relation to the outfit products specified. This because at the model definition only the properties purely from the model are filled.

The relations between models are specified in a Sequelize configuration file. At the Figure 6.8 we can observe part of this file where the relations between the outfits and products are defined.

```

module.exports = (sequelize, type) => {
  return sequelize.define('outfit', {
    likes: {
      type: type.INTEGER,
      defaultValue: 0
    },
    dislikes: {
      type: type.INTEGER,
      defaultValue: 0
    }
  });
}

```

*Figure 6.7 Model backend*

```

Product.belongsToMany(Outfit, {through: OutfitProduct});
Outfit.belongsToMany(Product, {through: OutfitProduct});

Session.belongsToMany(Outfit, {through: SessionOutfit});
Outfit.belongsToMany(Session, {through: SessionOutfit});

Outfit.belongsToMany(CharacteristicValue, {through: OutfitCharacteristic});
CharacteristicValue.belongsToMany(Outfit, {through: OutfitCharacteristic});

```

*Figure 6.8 Models' relations*

# 7. Testing

In this section we are going to explain the testing process for both the frontend and the backend of the system. Given the tight time, it has not been as extensive as we would have liked. That is why we prioritized the backend testing as it is the system's main component. Both the Mixties Prototype App and the Admin Dashboard depend on it.

## 7.1 Frontend Testing

Regarding the frontend testing, it has been the same for both the Mixties Prototype App and the Admin Dashboard. We have applied two different testing methods for this part of the system: Functionalities' Testing and Usability Testing.

### 7.1.1 Functionalities' Testing

We have used the monkey testing methodology [28]. It consists in putting random values at all the entry fields available, so that we can check the system behavior. With this method we could find some unexpected behavior and fix them.

It is important to say that the data validation is done at the backend level, so the frontend depends on the backend responses in order to process them and show the corresponding error messages if needed.

### 7.1.2 Usability Testing

During the product development process, both applications have been tested by our Mixties teammates in an informal way so that we could validate that the system GUI is:

- **Attractive.** It looks professional and they feel comfortable using the system.
- **Easy to use.** Without effort they are able to pass through the system's functionalities.
- **Understandable.** Each part of the interface is easily understandable, the information is organized and well-structured.

With their immediate feedback it was easy to correct the different aspects about the interface that were not clear enough and generated confusion.



## 7.2 Backend Testing

Regarding the backend testing, we have implemented integration tests for each one of the REST API endpoints, which cover all the use cases of the system (see Figure 4.1).

These tests have been implemented with the Supertest framework, introduced at the technologies section. It allows us to generate correct and incorrect HTTP requests to the backend and check the expected results of these calls. It has been very useful to ensure we were handling all the different possibilities and avoid the system failures.

We considered these types of tests mandatory, as doing them we are ensuring that API consumed by the other two applications is working as it should and handles all the possible requests, either they are correct or not.

For each one of the endpoints, we have implemented four different types of tests, adapted to the specific endpoint. They are defined in the following table.

Test type	Description	HTTP status code
<b>Bad authorization permission</b>	Authorization token missing or incorrect	401
<b>Missing or incorrect data</b>	Invalid or missing data in the body or in the URL parameters	400
<b>Endpoint not found</b>	Inexistent API endpoint	404
<b>Successful request</b>	The request contains all the information required	200

*Table 7.1 Integration tests types*

## 8. Law considerations

Before officially deploying our application, we have to comply with the GDPR law. In this section we define this law and explain our approach to comply with it.

### 8.1 General Data Protection Regulation (GDPR)

With our system we are treating with personal data, therefore we have to fulfill the set of user rights the GDPR defines [29].

- **Right to be informed.** User has the right to ask a company for information about what personal data is being processed and the rationale for such processing.
- **Right to access.** User has the right to get access to their personal data that is being processed. This request provides the right for data subjects to see or view their own personal data, as well as to request copies of the personal data.
- **Right to rectification.** This right provides the data subject with the ability to ask for modifications to their personal data in case the data subject believes that this personal data is not up to date or accurate.
- **Right to withdraw consent.** User has the right to withdraw a previously given consent for processing of their personal data for a purpose. The request would then require the company to stop the processing of the personal data that was based on the consent provided earlier.
- **Right to object.** User has the right to object to the processing of their personal data. A specific scenario would be when a customer asks that his or her personal data should not be processed for certain purposes while a legal dispute is ongoing in court.
- **Right to object to automated processing.** User has the right to object to a decision based on automated processing.
- **Right to be forgotten.** User has the right to ask for the deletion of their data. This will generally apply to situations where a customer relationship has ended.
- **Right for data portability.** User has the right to ask for transfer of their personal data. As part of such request, the data subject may ask for their personal data to be provided back or transferred to another controller. When doing so, the personal data must be provided or transferred in a machine-readable electronic format.

## 8.2 Our approach to comply with the law

At this point we have not implemented any measures to ensure we comply with the GDPR law. But we are aware of it and these are the measures that we will take before officially deploying our system.

- Develop a privacy policy with all the user's rights identified above related with our system, explaining clearly what information we are gathering from them and how are we treating them.
- Force users to agree with our terms and conditions before letting them interact with our system.
- Contact with a lawyer as an extra reviewing process in order to ensure we are able to deploy our system.

# Close out

In this last section of the thesis, we are going to analyze the technical competences of software engineering applied in this project. Then, we check out the planning and costs deviations occurred during the development of the project. Finally, I provide my personal conclusions along with the future work to do.

## 8.3 Software Engineering Approach

Software engineering is the systematic application of engineering approaches to the development of software. Among all the fields involved into this specialization, in this project's development we can highlight the software requirements, software design, software development and software testing.

### 8.3.1 Technical Competences

- **CES1.1: To develop, maintain and evaluate complex and/or critical software systems and services.** [Little]

In this project, we have developed an entire system with its own service. Far from being a critical or complex software but enough to apply this competence.

- **CES1.2: To solve integration problems in function of the strategies, standards and available technologies.** [Little]

We overcome integration problems related to the communication with external services that we have used to complement our system.

- **CES1.3: To identify, evaluate and manage potential risks related to software building which could arise.** [Enough]

We have completed a risk analysis regarding the things that can break during the software building, mitigation strategies to avoid possible damages and a contingency plan to minimize its effects.

- **CES1.5: To specify, design, implement and evaluate databases.** [In deep]

In this project, we have designed and implemented a database with more than just simple relations. The process of its design has been challenging and took a big part of the work dedicated to the project.

- **CES1.7: To control the quality and design tests in the software production.** [Little]

We have applied different types of tests to our system so that we can guarantee its correct behavior.

- **CES1.9: To demonstrate the comprehension in management and government of software systems.** [In deep]

This project involved the development of an entire software system with different applications to interact with. This technical competence has been deeply developed in order to manage all the system and govern its parts.

- **CES2.1: To define and manage the requirements of a software system.** [In deep]

All the required steps to specify a software system are included in this project, gathering requirements both functional and non-functional, using use cases and diagrams to support their definition.

### 8.3.2 Knowledge applied

These are the software engineering courses that have been more relevant for the successful development of this project.

- **Requirements Engineering (ER).** We learned about correctly analyze the state of the art and define the goals of the system to develop. Also to gathering both functional and non-functional requirements.
- **Software Architecture (AS).** Highly relevant for this project, in order to end up with a well-designed software architecture for the system following the software engineering methodologies.
- **Database Design (DBD).** Used for the correct design and management of our database.
- **Web Application and Services (ASW).** Highly useful the knowledge acquired during this course, as at the end we were developing both a web application and a service.
- **Software Project Management (GPS).** Both waterfall and agile methodologies learned during the course were useful to manage this project. Also, the different software development tracking and testing techniques.

Out from the specialization but also relevant for the project.

- **Artificial Intelligence (AI).** Knowledge about how to analyze and classify an AI problem to solve it using the different existing strategies.

## 8.4 Planning deviation

As expected, during the development of the project, we faced some problems tackling the AI outfit generation algorithm. Given that the time is fixed, we could not postpone the project's deadline. Therefore, we ended up reducing the scope, as explained in Section 2.3 Risk Management: Identification and alternative plan, page 32.

The scope reduction has been the elimination of the **AI outfit generation algorithm**. This happened because:

- **Data entry** has been slower than expected, causing the lack of data needed for the development of the algorithm.
- **Style patterns** not completely defined yet.
- We prioritized to be able to create the outfits manually.

On the other side, even though the time is fixed, we set a deadline before the actual one in order to overcome problems like this. Therefore, we have redistributed the time, adding one more sprint into the Agile Project Execution. We had to do it because reducing the scope was not enough to finish with the rest of the user stories, and we found ourselves with the need of more time to finish them properly.

### 8.4.1 Final Time Planning

Name	Days	Estimated Hours	Resources	Predecessors
<b>Agile Project Inception</b>			R.2, R.5	
Idea Consolidation	7	13		
MVP Proposal	9	14		I.2
Risk Analysis	3	7		I.3
Project Size Up	3	7		I.3
<b>Agile Project Planning</b>			R.2, R.5	
Context and Scope	8	24,5		I.4
Time Planning	5	8,25		P.1
Budget and Sustainability	6	9,25		P.2
Final Document	6	18,25		P.3
Requirements analysis	4	10		I.5
System specification	7	14		P.5
System design	7	15		P.6
<b>Agile Project Execution</b>			R.1, R.2, R.3, R.4, R.5, R.6	
Sprint 1   Release 1: MPA	12	60		P.3
Sprint 2   B	12	60		S.1
Sprint 3   Release 2: B	13	60		S.2
Sprint 4   AD	13	60		S.3
Sprint 5   Release 3: AD	13	60		S.4
Sprint 6   Review	13	60		S.5
<b>Memory Documentation</b>		60	R.2, R.5	I.4
<b>TOTAL</b>		<b>560,25</b>		

Table 8.1 Final Time Planning table

### 8.4.2 Final Gantt Diagram

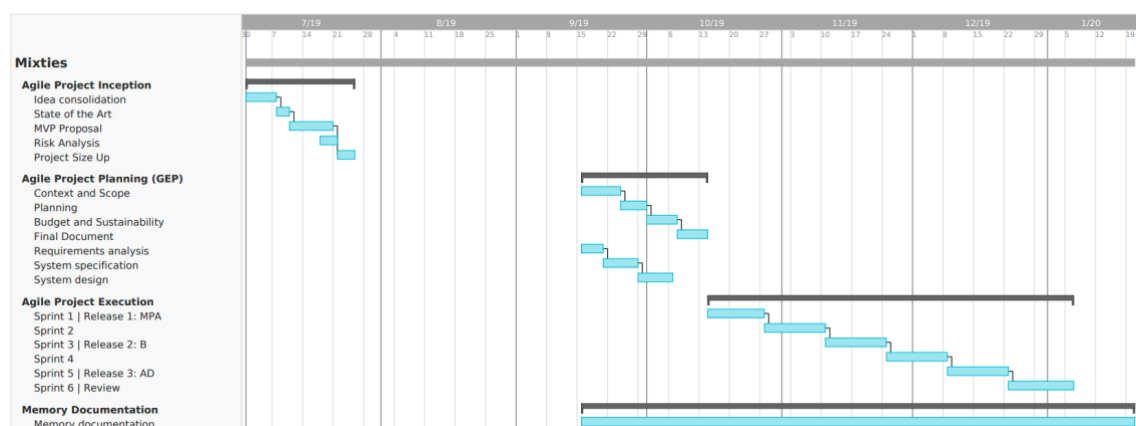


Figure 8.1 Final Gantt Diagram

## 8.5 Cost Deviation

As a consequence of the planning deviation where we ended up doing more hours than expected at the beginning, it has had an impact both to the direct and indirect costs. First of all, it has increasing the direct costs because as the time has been increased our supposed development team would have worked more hours. We can see the final direct costs at the Table 8.2.

Secondly, the indirect costs have been also increased because we have spent more time using the internet plan and electricity. We can see the final indirect costs at Table 8.3.

Finally, as the contingency costs depends on the direct and indirect costs, it has also ended up increasing. We can see all these costs together at the Table 8.4. Notice that all the values have been calculated exactly the same way as the original ones.

Role	Gross salary (€)	Gross cost (€/h)	Net cost (€/h) x 1,35	Hours worked (h)	Total salary (€)
Project Manager	40.000,00	22,68	30,61	190,45	5.830,10
Architect	40.000,00	22,68	30,61	66,11	2.023,78
Developer	18.000,00	10,20	13,78	203,22	2.799,49
UX/UI designer	18.000,00	10,20	13,78	59,22	815,81
Quality Assurance	18.000,00	10,20	13,78	41,22	567,85
<b>TOTAL</b>				<b>560,23</b>	<b>12.037,03</b>

Table 8.2 Final direct costs

Resource	Cost (€)
<b>Software</b>	
Visual Studio Code	0
Google Docs	0
Github	0
Taiga	0
Heroku	0
Postman	0
<b>Hardware</b>	
MacBook Pro 15"	245,00
OnePlus 5T	77,00
<b>Generic Costs</b>	
Electricity	7,31
Internet Access	178,08
<b>Total IC</b>	<b>507,39</b>

Table 8.3 Final indirect costs



Name	Amount (€)
Direct Costs	12.037,03
Indirect Costs	507,39
Contingency	1.881,66
Incidentals	70,00
<b>TOTAL</b>	<b>14.496,08</b>

*Table 8.4 Final total costs*

## 8.6 Conclusions

This project has resulted to be more challenging than expected. The fact of not having a specific path to follow given that the project was freely chosen by me, ended up with a continuous redefinition of the scope and the design of the system. But personally, I consider that I have learned a lot struggling with this.

Regarding the technical aspects of the project, its development has helped me to consolidate and bring together knowledge learned in different courses. At this point, I can understand the importance of the abilities developed in each area of this bachelor's degree.

Finally, I have learned new technologies such as the Angular framework. Also, I could see all the parts implied in an entire web application development. From the database design, the development of both the frontend and backend, to the deployment process of the whole system.

## 8.7 Future work

We ended up with a system that fulfills our needs and we can start using it to prepare our near deployment, but there are still important things to do that we would have to consider such as:

- Increase the system security. At the moment we have focused more on just the development of the system's functionalities, but we have to ensure the whole system's security from possible attacks.
- Comply with all the laws. As explained in the corresponding section, we have to ensure we comply with the law. Not only for our customers' protection, but also our own.
- Develop the artificial intelligence algorithm for the outfits' generation that we could not do during this project.

## 9. Bibliography

- [1] Natural By Lila, "Homepage: Natural By Lila," [Online]. Available: <https://naturalbylila.com/>. [Accessed 21 10 2019].
- [2] Carol Davidson, "Virtual Stylist: Carol Davidson," [Online]. Available: <https://caroldavidson.com/services/virtual-stylist/>. [Accessed 21 10 2019].
- [3] Lookiero, "FAQ: Lookiero," 19 10 2019. [Online]. Available: <https://lookiero.co.uk/faq.html>.
- [4] TryTuesday, "FAQ: TryTuesday," [Online]. Available: <https://trytuesday.com/frequently-asked-questions>. [Accessed 21 10 2019].
- [5] Viume Co, "FAQ: Viume Co," [Online]. Available: <https://viume.co/faq>. [Accessed 21 10 2019].
- [6] The Lean Startup, "The Lean Startup," [Online]. Available: <http://theleanstartup.com/>. [Accessed 10 12 2019].
- [7] Scrum , "The scrum guide," [Online]. Available: <https://www.scrum.org/resources/scrum-guide>. [Accessed 21 10 2019].
- [8] Michael Page, "Tecnología: Michael Page," [Online]. Available: [https://www.michaelpage.es/sites/michaelpage.es/files/PG\\_ER\\_IT.pdf](https://www.michaelpage.es/sites/michaelpage.es/files/PG_ER_IT.pdf). [Accessed 10 21 2019].
- [9] CCOO, "Jornada: CCOO," [Online]. Available: <https://www.ccoo-servicios.es/ilunioncontactcenter/pagweb/2872.html>. [Accessed 21 10 2019].
- [10] "API," Wikimedia, [Online]. Available: [https://en.wikipedia.org/wiki/Application\\_programming\\_interface](https://en.wikipedia.org/wiki/Application_programming_interface). [Accessed 28 12 2019].
- [11] Google, "Google Maps Platform," Google, [Online]. Available: <https://cloud.google.com/maps-platform/?hl=es>. [Accessed 12 12 2019].
- [12] ip-api.com, "ip-api," ip-api.com, [Online]. Available: <https://ip-api.com/>. [Accessed 12 12 2019].

- [13] Angular, "Angular," Angular, [Online]. Available: <https://angular.io/>. [Accessed 12 12 2019].
- [14] Expressjs, "Express," Express, [Online]. Available: <https://expressjs.com/>. [Accessed 13 12 2019].
- [15] OpenJS, "nodejs," OpenJS, [Online]. Available: <https://nodejs.org/en/>. [Accessed 13 12 2019].
- [16] Express, "Express," [Online]. Available: <http://expressjs.com/en/guide/writing-middleware.html>. [Accessed 20 12 2019].
- [17] Auth0, "JWT," Auth0, [Online]. Available: <https://jwt.io/>. [Accessed 13 12 2019].
- [18] Salesforce, "Heroku," Salesforce, [Online]. Available: <https://www.heroku.com/>. [Accessed 14 12 2019].
- [19] MongoDB, "mongoDB," mongoDB, [Online]. Available: <https://www.mongodb.com/>. [Accessed 13 12 2019].
- [20] Facebook, "React," Facebook, [Online]. Available: <https://en.reactjs.org/>. [Accessed 14 12 2019].
- [21] Hapi, "Github joi," Hapi, [Online]. Available: <https://github.com/hapijs/joi>. [Accessed 14 12 2019].
- [22] Auth0, "Passport," Auth0, [Online]. Available: <http://www.passportjs.org/>. [Accessed 14 12 2019].
- [23] multer, "github multer," multer, [Online]. Available: <https://github.com/expressjs/multer>. [Accessed 14 12 2019].
- [24] Facebook, "Jest," Facebook, [Online]. Available: <https://jestjs.io/>. [Accessed 14 12 2019].
- [25] visionmedia, "github supertest," visionmedia, [Online]. Available: <https://github.com/visionmedia/supertest>. [Accessed 14 12 2019].
- [26] Sequelize, "Sequelize," Sequelize, [Online]. Available: <https://sequelize.org/>. [Accessed 14 12 2019].
- [27] PostgreSQL, "PostgreSQL," PostgreSQL, [Online]. Available: <https://www.postgresql.org/>. [Accessed 14 12 2019].
- [28] "Monkey testing," Wikimedia, [Online]. Available: [https://en.wikipedia.org/wiki/Monkey\\_testing](https://en.wikipedia.org/wiki/Monkey_testing). [Accessed 22 12 2019].

[29] Intersoft Consulting, "General Data Protection Regulation," Intersoft Consulting, [Online]. Available: <https://gdpr-info.eu/>. [Accessed 18 12 2019].