TÉCNICO LISBOA

Ultimate Capsule, an app that helps us dress better with less clothes

Susana Isabel Pires Machado Ferreira Guerreiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Prof. Sandra Pereira Gama

Examination Committee

Chairperson: João António Madeiras Pereira

Supervisor: Maria Luísa Torres Ribeiro Marques da Silva Coheur

Member of the Committee: Hugo Miguel Aleixo Albuquerque Nicolau

June 2018

Acknowledgements

Foremost, I want to express my gratitude to my supervisors, Maria Luísa Coheur and Sandra Gama, for making it possible the combination of two of my favourite topics, fashion and computer science, into a thesis theme. I am very grateful for all the support, guidance and advices you have given me throughout all the development of this project.

I would also like to thank my parents and family for all their support and help throughout my academic journey and especially during this project.

Finally, but not least, I would like to thank all my colleagues who helped me in this project, giving support and participating in tests.

Susana Guerreiro

Abstract

The clothes we wear can influence the impression that we give to others, making the process of choosing an outfit a considerable important task. Over the past few years, this has been more complicated, because most of people have more and more clothes in their closets.

The term capsule wardrobe was invented by Susie Faux, in 1980, with the intuition of reducing the number of pieces in the wardrobe. Several reinventions have occurred in different fashion online blogs, in terms of time period and number of clothing items of each capsule.

Despite being a concept relatively famous, that we have knowledge of, there is no application or service capable of simultaneously allow users to create capsules wardrobe and obtain outfits suggestions based on them.

Motivated by the capsule wardrobe concept, we developed an Android application, called Ultimate Capsule. It allows users to create three month capsules wardrobe and obtain outfits recommendations according to occasion and/or characteristics specified. Regarding outfit recommendation we added natural language to allow users to textually describe them.

To create this application we followed an iterative and incremental development approach. Tests with users have shown that we were able to create an intuitive application, easy to learn and use.

Keywords: Clothes Coordination, Capsule Wardrobe, Outfit Recommendation, Natural Language

Resumo

As roupas que usamos podem influenciar a impressão que damos aos outros, tornando o processo de escolher uma roupa uma tarefa crítica. Nos últimos anos, isso tem sido mais complicado, porque a maioria das pessoas tem mais e mais roupas nos seus armários.

O termo cápsula da roupa foi inventado por Susie Faux, em 1980, com o objectivo de reduzir o número de peças no guarda-roupa. Diversas reinvenções ocorreram em diferentes blogs on-line de moda, em termos de intervale de tempo e número de peças de roupa de cada cápsula.

Apesar de ser um conceito relativamente famoso, que tenhamos conhecimento, não existe nenhuma aplicação móvel ou serviço que permita aos utilizadores simultaneamente criar cápsules de roupa e obter recomendação de combinação de roupas com base nelas.

Motivados pelo conceito de cápsula, desenvolvemos uma aplicação Android, denominada Ultimate Capsule. A aplicação permite aos utilizadores criarem uma cápsula de roupa com três meses de roupa e obterem recomendações de combinação de roupas de acordo com a ocasião e/ou características especificadas. Relativamente à recomendação de combinação de roupas, adicionámos língua natural para permitir aos utilizadores descreverem-na textualmente.

Para criar a aplicação seguimos uma abordagem de desenvolvimento iterativa e incremental. Os testes com utilizadores mostraram que conseguimos desenvolver uma aplicação intuitiva, fácil de aprender e usar.

Keywords: Combinação de Roupa, Cápsula de Roupa, Recomendação de Combinação de Roupa, Língua Natural

Contents

Acknowledgements i

Abstract ii

Resumo iii

1 Introduction 1

2 Background and Related Work 3
2.1 Background . . . 3
2.2 Related Work . . . 5
2.2.1 Fashion Recommendation Systems . . . 5
2.2.1.1 Clothing Attributes . . . 5
2.2.1.2 Occasion . . . 6
2.2.2 Finding Clothes and Outfits Through Natural Language . . . 8
2.2.3 Similar Applications . . . 10
2.2.4 Summary . . . 11

3 Ultimate Capsule 13
3.1 Functionalities . . . 14
3.1.1 Profile . . . 14
3.1.2 Virtual Closet . . . 14
3.1.3 Capsule Wardrobe . . . 15
3.1.4 Outfit Recommendation . . . 16
3.1.5 Simulate Purchase . . . 17
3.2 Solution Architecture . . . 17
3.2.1 Clothes Coordination Algorithm . . . 17
3.2.2 Capsule Creation Algorithm . . . 17
3.2.3 Fashion Natural Language Algorithm . . . 17
3.2.4 Outfit Recommendation Algorithm . . . 18
3.2.5 Simulate Purchase Algorithm . . . 18
3.2.6 Database . . . 18

4 Solution Development 21
4.1 Main Tasks and Storyboards . . . 21
4.2 Low-Fidelity Non-Functional Prototypes . . . 23
4.2.1 Virtual Closet . . . 23
4.2.2 Capsule Wardrobe . . . 26

4.2.3 Outfit Recommendation . . . 26
4.2.4 Simulate Purchase . . . 27
4.3 Functional Prototypes . . . 28
4.3.1 Project Structure and Organization . . . 28
4.3.2 Project Initial Elements . . . 30
4.3.3 Profile . . . 33
4.3.4 Virtual Closet . . . 38
4.3.5 Simulate Purchase . . . 44
4.3.6 Capsule Wardrobe . . . 45
4.3.7 Outfit Recommendation . . . 48
4.3.8 Clothes Coordination Algorithm . . . 50
4.3.9 Capsule Creation Algorithm . . . 51
4.3.10 Fashion Natural Language Algorithm . . . 52
4.3.11 Outfit Recommendation Algorithm . . . 54
4.3.12 Simulate Purchase Algorithm . . . 56

5 Solution Evaluation **57**
5.1 Low-Fidelity Non-Functional Prototypes Evaluation . . . 57
5.1.1 Evaluation Process . . . 58
5.1.2 Users . . . 59
5.1.3 Evaluation Results . . . 59
5.2 Formative Functional Prototypes Evaluation . . . 61
5.2.1 Evaluation Process . . . 61
5.2.2 Users . . . 63
5.2.3 Evaluation Results . . . 63
5.3 Summative Functional Prototypes Evaluation . . . 66
5.3.1 Usability Tests . . . 67
5.3.1.1 Evaluation Process . . . 67
5.3.1.2 Users . . . 69
5.3.1.3 Evaluation Results . . . 69
5.3.2 Case Studies . . . 73
5.3.2.1 Evaluation Process . . . 73
5.3.2.2 Users . . . 73
5.3.2.3 Evaluation Results . . . 74
5.4 Fashion Natural Language Algorithm Evaluation . . . 74

6 Conclusions **75**
6.1 Contributions . . . 76
6.2 Future Work . . . 76

Bibliography **77**

Appendices **81**

Appendix A Storyboards **81**

Appendix B Semantic Grammar **85**

Appendix C Usability Time Results Collected **87**

List of Figures

2.1 Example of colour wheel with 12 colours, 3 primary, 3 secondary and 3 tertiary taken from [4]. 4
2.2 Examples of analogous, complementary, split complementary and triad colours taken from [4]. 4

3.1 Scheme of SQLite database developed to store our solution data. 19

4.1 Storyboard for upload a new clothing piece 22
4.2 Paper prototypes for virtual closet organization 24
4.3 Paper prototypes for adding a new piece into virtual closet. 25
4.4 Paper prototypes for creating a new capsule wardrobe. 26
4.5 Paper prototypes for obtaining a new outfit recommendation. 27
4.6 Paper prototypes for simulate a purchase of a new clothing piece. 28
4.7 Android project structure. 29
4.8 Functional prototypes iterations of side menu design. 31
4.9 Splash screen with our application icon and name. 32
4.10 Home screen final design, respectively when exists outfit or not for selected day. 32
4.11 First profile edition design iteration, identically to first profile creation design iteration. . . . 33
4.12 Android default calendar possible views. 34
4.13 Functional prototypes first design of errors in profile creation. 35
4.14 Functional prototypes final design of errors and warnings in profile creation and edition. . 35
4.15 Second profile edition design iteration, identically to second profile creation design iteration. 36
4.16 Profile creation third design iteration. 37
4.17 Fourth profile edition design iteration, identically to third profile edition design iteration with Gender input box with grey background and additional field, Location. 38
4.18 Fifth profile creation design iteration, identically to fourth profile edition design iteration with help button above the Location field input box. 39
4.19 Virtual closet types overview with add new piece button in the toolbar. 39
4.20 Select new clothing piece category, with both previous and cancel buttons. 41
4.21 Select new clothing piece colour. 42
4.22 Select new clothing piece occasions new design. 42
4.23 Select new clothing piece take picture problematic screens. 43
4.24 Explanatory screen where we describe simulate purchase functionality and its main goal. 45
4.25 Summer capsule layout example with information button in the right upper corner. 45
4.26 Current capsule section possible scenarios. 47
4.27 Current capsule section possible scenarios. 48
4.28 Outfit recommendation personalization screen that allow users to customize type of outfit, select specific clothing item, dominant colour and/or if must have accessories or not. . . . 49

5.1 Correct end screens for create new capsule, obtain outfit recommendation and simulate new piece acquisition 59
5.2 Evaluation results for each task of low-fidelity non-functional prototypes. 60
5.3 Correct end screens for create new capsule and select as current, obtain outfit recommendation and simulate new piece acquisition. 62
5.4 Classification results for each task of functional prototypes formative evaluation. 64
5.5 Select new clothing piece take picture problematic screens. 65
5.6 Outfit recommendation personalization screen. 66
5.7 Quartiles graphic representation, in seconds, for time spent in each task by users. 71
5.8 Classification results for each task of functional prototypes summative evaluation. 73

A.1 Storyboard for create a new capsule wardrobe 82
A.2 Storyboard for obtain an outfit recommendation 83
A.3 Storyboard for simulate the acquisition of a new clothing item 84

List of Tables

3.1 Clothes categories for male users by clothes types. 15
3.2 Clothes categories for female users by clothes types. 15

4.1 Correspondence between tags and keywords identified by Aho-Corasick during mentioned entities recognition. 53

5.1 Each task start and correct end. 58
5.2 Correct end for each task. 62
5.3 Correct end for each task. 68
5.4 Mean, median and standard deviation, in seconds, for time spent in each task by users. . 69
5.5 Quartile values, in seconds, for time spent in each task by users. 70

B.1 Semantic grammar defined by us concerning outfit description. 85

C.1 Time, in seconds, spent performing each task by each user. 87

List of Acronyms

RGB red-blue-green

SQL Structured Query Language

SVG Scalable Vector Graphics

SVM Support Vector Machine

XML eXtensible Markup Language

1

Introduction

Impressionsaabout us can be formed in a fraction of a second, without people talk or interact with us, as stated by Bar et al. [3]. According to Howlett et al. [11], one of the visual aspects that influence impression is the way we dress and the clothing items that we choose to wear and combine. Having this in consideration, making the decision of choosing an outfit becomes a considerable important task. In general, over the past few years, the number of clothing pieces in people's closets has increased significantly, transforming the task of finding an appropriate outfit into a very complex and difficult problem.

In 1980, Susie Faux created the concept capsule wardrobe [1]. It consists in the smallest possible set of clothing items, including shoes and accessories, all possible to combine with each other, producing several different outfits. Over the past few years, the concept capsule wardrobe has been reinvented and revisited by many fashion online blogs. Two example of blogs that reinvented the concept are: *Unfancy* [2] and *Be More With Less* [3]. In both the capsule time and number of clothing pieces are defined. Both consider three months capsules, however the former limits to 37 clothing pieces, while the latter limits to 33. In both, the capsule wardrobe must be functional and practical, easing the process of selecting and matching clothing items, but still being able to accommodate all the different types of events during that three months period.

Based in the concept of capsule wardrobe, that we have knowledge of, there is only one online service called *Cladwell Capsules* [4]. It supports all the process of creating a capsule wardrobe, however, it can not recommend outfits based in the clothing items of that capsule.

Our main goal was to develop an intuitive, easy to learn and use mobile application that supports the creation and management of those three month capsules wardrobe. This application also helps both male and female users select and match clothing items by suggesting complete outfits that are appropriated to a given occasion. Our solution supports as clothing items bottom, top, full-body pieces, shoes and accessories and as main functionalities:

- Virtual closet, where users can upload and visualize their clothes;
- Creation and management of capsules wardrobe for three months periods, with the option to personalize the number of pieces by type;

[1] `http://www.wardrobe.co.uk/bio.html` (last access on 26/05/2017)
[2] `http://www.un-fancy.com/` (last access on 25/05/2017)
[3] `http://bemorewithless.com/` (last access on 25/05/2017)
[4] `https://cladwell.com/capsules` (last access on 14/05/2017)

- Outfit recommendations appropriate to occasion and/or other requirements mentioned by users;
- Simulate acquisition of new clothing items, showing how many outfits are possible to be created with these new items.

In order to achieve these goals we created an Android application, called Ultimate Capsule [5], using an iterative and incremental approach. This application was structured into five modules and five algorithms. The modules encapsulate all logic behind each of five functionalities of our application, more specifically Profile, Virtual Closet, Capsule Wardrobe, Outfit Recommendation and Purchase Simulation. The algorithms were created with the intention to be able to verify if two clothing items match, calculate capsules' clothes, interpret natural language textual descriptions given by users, find the appropriate outfit according to requirements given and compute possible outfits with the new clothing item simulated.

Our main contributions with this work are:

- An algorithm capable of automatically create three month capsules wardrobe;
- An algorithm capable of process and interpret an natural language textual description about an outfit;
- An algorithm with the ability of automatically recommend an appropriate outfit, from the users' clothes in current capsule, or virtual closet if there is no current capsule, according to occasion and other requirements given by users;
- An algorithm that allows to simulate the acquisition of a new clothing item, calculating possible outfits with it;
- The creation of an intuitive, easy to learn and use Android mobile application that merges outfit recommendation system and natural language.

The rest of this document is divided into five chapters. In chapter 2, we will describe some background in how to match clothes as well as previous works in the areas of recommend outfits and find clothes through textual descriptions in natural language. Additionally, we will also describe and analyse existing mobile applications and online services similar to our proposal. In chapter 3 we will detail our application functionalities as well as define its architecture. In chapters 4 and 5, we will, respectively, narrate our iterative and incremental design approach to develop our solution and the different evaluation phases performed in terms of interface during development and concerning natural language algorithm. Finally, in chapter 6 we will make some conclusions about our work, considering evaluations results obtained, and also list some future work that can be done. In Appendix we will show some of our solution interface design iterations through development, as well as our natural language grammar and some usability results.

[5] `https://ultimatecapsule.wordpress.com` (last access on 11/05/2018)

2

Background and Related Work

Before presenting studies related to our work, we found important to understand how clothing items can be put together to achieve good outfits. With this intention, we did a research on how to coordinate clothes in terms of colours and patterns, that we present in section 2.1.

2.1 Background

To coordinate outfits is necessary to have in consideration if clothes are in harmony with each or not. According to [5], clothes are in harmony if they coordinate well with each other, if there is a balance between them, neither being confusing or tedious [1] [2]. Several aspects can be considered to verify if two clothes are in harmony, namely colour and pattern.

"Colour is a visual effect resulting from the eye's ability to distinguish the different wavelengths or frequencies of light" [1]. Normally, to define colours that result in a harmonious set it is used a colour wheel with twelve different colours [3], three of them primary, three secondary and three tertiary. In figure 2.1, we can observe a colour wheel example from [4].

Based on colours positioning in colour wheel we can define several colour schemes, in other words, colours sets that create colour harmony. Possible colour schemes are analogous, complementary/opposite, complementary split and triadic scheme. Examples of these schemes are presented in figure 2.2 from [4]. Concerning analogous scheme, sometimes named similar or adjacent, according to [5], this is composed by two or three colours that are immediately neighbours of each other in the colour wheel. In figure 2.2, we can observe that blue-green, green and yellow-green are considered analous colours. Relatively to complementary/opposite scheme, as the name say and according to according to [5], this correspond to two colours in opposite sides of colour wheel. In figure 2.2, we can observe that one example of complementary colours are green and red. In terms of complementary split scheme, this is composed by one colour and two adjacent colours of opposite colour of the first [5] [4]. In figure 2.2, we can

[1] https://www.askandyaboutclothes.com/clothing/style-tips/practical-color-help/ (last access on 27/02/2018)

[2] https://www.thedarkknot.com/pages/how-to-match-ties-to-suits-shirts (last access on 26/02/2018)

[3] http://www.fashionbeans.com/2011/mens-fashion-basics-\T1\textendash-part-41-\T1\textendash-an-introduction-to-colour/ (last access on 25/02/2018)

[4] http://www.spectrumnoir.com/an-introduction-to-the-colour-wheel/ (last access on 05/05/2018)

[5] http://attireclub.org/2014/05/05/coordinating-the-colors-of-your-clothes/ (last access on 27/02/2018)

Figure 2.1: Example of colour wheel with 12 colours, 3 primary, 3 secondary and 3 tertiary taken from [4].

observe that one example of complementary split colours scheme corresponds to red-violet, red-orange and green. A triadic scheme, according to [1] [2], means three colours equally distant in the colour wheel. One example, represented in 2.2, is violet, orange and green. Additionally, there is a mono tone scheme, called monochrome chromatic scheme, which is constituted by only one colour from colour wheel [5].

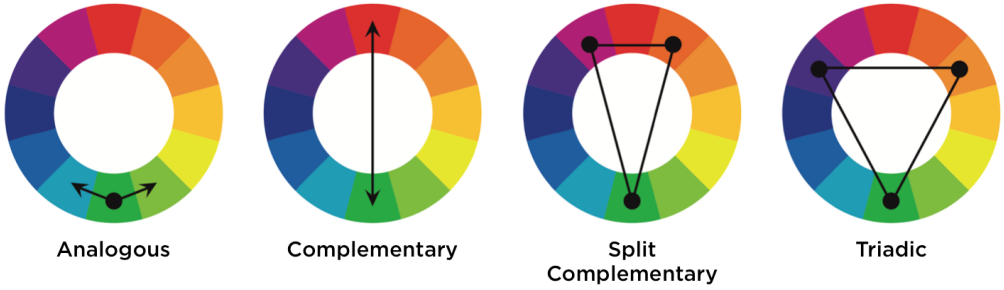


Figure 2.2: Examples of analogous, complementary, split complementary and triad colours taken from [4].

Besides colours in colour wheel, there are also colours that are defined as neutrals [1]. These correspond to black, grey, white, brown, beige, khaki and navy, colours that result from combining different percentages of all primary colours or one primary with one secondary. These colours are denominated neutrals because they can combine with any colour from the colour wheel. In respect of neutral colours, we can define neutral colour scheme. This corresponds to combine neutral colours. Similarly to colours in colour wheel, there is also a mono tone scheme with neutrals colours, denominated monochrome tone scheme, since it is composed by only one neutral colour [5].

Normally, while combining clothes into outfits, we should not have more than three different colours [5]. Ideally, we should have one primary, one secondary and one tertiary colours. Sometimes is also defined an accent colour, in other words, one colour from colour wheel is chosen and combined with only neutral colours [6].

In terms of patterns combination, identically to colours, we should not mix lot of patterns together [1]. Usually, according to [5], patterns should be coordinated with solid colours. However, this does not mean

[6] https://www.collegefashion.net/fashion-tips/a-foolproof-guide-to-matching-colors-in-your-outfits/ (last access on 27/02/2018)

that two pattern can not be used together. It is possible to combine two patterns if their colours create harmony and if they have different scales, in other words, if one has small pattern and the other has big pattern [2].

2.2 Related Work

With the intention to better understand what has been done and what could be improved in terms of our work main goals, we searched for previous works related to the capsule wardrobe concept, outfit recommendation and finding clothes through textual descriptions in natural language. Besides that, we also tried to find and analyse similar online services and mobile applications.

More precisely, in terms of fashion recommendation systems, in subsection 2.2.1, we will present several works that have the goal to make clothes suggestions and analyse them in terms of conditions in the recommendation, type of clothes suggested and approaches. Taking into account our main goal of suggesting outfits according to clothes attributes and occasion, we will focus in works that consider at least one of these two parameters in the recommendation process.

In subsection 2.2.2, we will describe and discuss some previous works that, through natural language mechanisms, are capable of finding outfits that satisfy textual descriptions in natural language, including a recommendation system mentioned in 2.2.1.

In subsection 2.2.3, we will analyse similar online services and mobile applications that have in consideration the capsule wardrobe concept, are capable of outfits recommendation and/or have a virtual closet. We will also describe their advantages and disadvantages and compare them with our solution.

Finally, in subsection 2.2.4, we will perform a overall analysis on previous works presented, comparing them with our solution.

2.2.1 Fashion Recommendation Systems

Fashion recommendation systems have the ability of suggesting outfits or clothes pairing which may be interesting to the user. This has been an area more and more explored through the years, arising different systems in terms of approach or conditions considered in the recommendation process. Generally, these systems can suggest outfits appropriate to clothes attributes and occasion.

2.2.1.1 Clothing Attributes

One aspect that our recommendation system depends on is the clothes' characteristics, for example type, colour and pattern. Some examples of systems that consider those kind of features in the recommendation are described in [12, 17, 24].

Iwata et al. [12] propose a clothing recommendation system that, using photographs taken from magazines with complete outfits as reference and machine learning mechanisms, namely topic model, learns to relate and coordinate different pieces of clothing, without needing textual descriptions associated with the images. Using the topic model, the proposed system is able to suggest upper (low) clothing pieces, present in the users' closets, from a given lower (top) clothing piece given as a query. In order to identify each clothing piece in each reference photography and its characteristics, a face detection algorithm was used. In this system, the feature colour was used as a parameter in the clothes combination. In order to suggest outfits according to a top (bottom) piece, the system searches for reference photographs with similar top (bottom) pieces and returns users' bottom (top) pieces similar to the ones in the references photographs.

Similar to [12], Lee and Lee [17] and Vartak and Madden [24] also propose fashion recommendation systems that use reference images with complete outfits. However, these images are from online fashion

stores, instead of being from magazines, and they are only used to train the systems.

The system described in [17], besides the references images with complete outfits, also uses single clothing items images. For both type of images, from its descriptions and images, four different attributes are obtained: type of piece, material, pattern and colour. In order to learn how to match clothes, a heterogeneous information network was created based on the reference photographs and attributes extracted from them. In this network, relations between outfits' pieces and between two clothing pieces' attributes were defined. Unlike the system proposed by [12], using this network, this system is capable of suggesting complete outfits, given a query clothing item, in other words, combinations of top and bottom pieces, shoes and accessories.

The system described in [24] is called Chic and, given a user's clothing piece or desired feature of the combination, it is capable of recommend several outfits that meet the initial requirement. This system recommends outfits through a prediction model and a new search and classification algorithm called C-Search. The prediction model learns how to coordinate clothing pieces through the visual features and textual descriptions of each clothing piece in each reference image and the outfits' features calculated through each clothing piece's characteristic. The new search and classification algorithm, C-Search, learns to classify outfits in terms of the coordination's quality and selects the outfits with higher quality. To be able to recommend the higher-quality outfits, it is necessary to compute, classify and compare the quality of each outfit with the initial clothing piece or desired feature. In order to reduce time of computing and comparing all possible outfits, without missing any possible outfit, this system pre-calculates some clothing pieces pairs with users' clothes. When users want a outfit recommendation, the system searches for the most similar piece or desired feature in the database, locating its pre-computed pairs. Based on the pre-computed pairs, the system creates complete outfits that are classified and compared in terms of quality. The outfits with higher quality are returned as suggestions. The query clothing item used to obtain an outfit suggestion can correspond to a picture of the clothing item or an image of a clothing item from the pre-existing database.

All the three systems mentioned in this subsection only consider clothing attributes in the recommendation process and use reference images of complete outfits to suggest their own outfits. In [12] the images are used to suggest outfits similar to them, while in [17] and [24] the images are used to train their own systems. Despite only considering clothing attributes and requiring images to train or coordinate clothes, these three systems are similar to our proposal, since all the three systems receive as input a query clothing item and in our proposal we will allow users to obtain outfits recommendations with a clothing item chosen by the user or simulate the purchase of a new clothing item. From these three, the most similar are the systems proposed by Lee and Lee [17] and Vartak and Madden [24] since they are capable of suggesting complete outfits, unlike the system in [12], which only suggests top and bottom pairs. The main difference between [17] and [24] relies on the approach followed.

2.2.1.2 Occasion

Other aspect that we will have influencing our recommendation system is the occasion, like casual, formal or others. Some examples of systems that consider occasion in the recommendation, besides clothing attributes, are described in [4, 8, 22, 14, 15, 19, 26].

Cheng and Liu [4] propose a smart clothing search system, which simulates a personal fashion consultant, suggesting outfits appropriate to occasion and desired image/style. It also provides a virtual closet where users can search clothing items through keywords or occasion. In order to obtain recommendation of outfits, users must insert all clothes in the virtual closet, taking a photography and inserting the type of fabric. Clothing features such as colour, pattern type and silhouette are extracted using digital image processing methods. To learn how to infer a desired outfit, given certain characteristics as

input, the proposed system uses a neural network. This neural network learns how to classifies the different outfits through several training sets of clothes-style examples, outfits' image and characteristics and expected/correct results.

Feng et al. [8] propose a system, called Magic Closet, which aims to explore and capture the rules of clothes-occasion and clothing-clothing combination. With these rules, the system, given an initial clothing piece, recommends users' clothes combinations or pairs of users' clothes with clothes from online stores appropriate to the occasion selected. The Magic Closet uses a machine learning algorithm, called Support Vector Machine (SVM) model, which, through several complete outfits training images provided, learns to recommend suitable outfits to occasions and according to clothes coordination rules. The training images were analysed with image processing algorithms to distinguish the top and bottom pieces and extract its visual characteristics and annotated with Amazon Mechanic Turk [7], in terms of categories, pieces' characteristics and details and appropriate occasions. After learning, the SVM model can distinguish between suitable and unsuitable outfits, according to combinations of visual characteristics vs attributes, visual characteristics vs. occasion, attributes vs. occasion and attributes vs. attributes. This system is similar to ours since it captures relations between clothing items but also between clothing items attributes. However, it has the disadvantage of requiring thousands of images to learn.

Similar to [8], Shen et al. [22] propose a fashion recommendation system that, besides the clothing attributes, also considers occasion, in this case they called it scenario, in the recommendation process. This system allows users to textually describe the scenario and/or the emotional state that users wish to feel. Having in account the occasion and/or emotional state descriptions, this system is capable to infer the right style the suggested outfit may have. To infer the style of the outfit and users' clothes, this system depends on a file, where are represented all relations between brands, types of clothing items, materials and some occasions and styles, Open Mind Common Sense [23], a collection of English sentences about common sense and ConceptNet [18], a tool that relates different concepts with each other. Unlike the system described in [8], the system proposed by Shen et al. [22] has in account users' profiles, most common styles in users' clothes, preferences and past interactions/feedback to suggested outfits in the recommendation process. Additionally, this system also allows users to create personalized outfits or ask friends for advices. This system has two main advantages: textual descriptions of scenarios or emotional states desired, instead of the usual checkboxes, and the ability to infer styles from users' clothes. However, to infer style it depends on a hand crafted file, which it is subjective and dependent on the personal opinion and mental idea of the person who created the file. This system is similar to our proposal, since it allows textual descriptions about occasions.

Additionally to these, the system proposed by [14], besides clothing attributes, has in account occasion while recommending. It consist in a fashion recommendation system that recommends outfits appropriate to the time, place, occasion and impression that the user wishes to transmit. The last attribute, impression depends on time, namely season, place's luxury, whether it is high, intermediate or low, and people invited, namely, girlfriend/boyfriend, friends and co-workers. To infer the correct impression to a certain time, place and occasion, this system has created a regressive model of impression, composed by neural networks. To train this model, several outfits' images and characteristics, impressions, times, places and occasions were provided. Through this training phase, the regressive model is capable of two things: infer the impression of users' clothes and infer the impression that the suggested outfit must have, according to a given time, place and occasion. Knowing the impression that the suggested outfit must have and impression of each users' clothing pieces, the system is capable of recommend an appropriate outfit to a given time, place and occasion, in other words, an outfit with the correct impression to those conditions. Considering the approach followed, this system has one main disadvantage, it requires a high number of images to train the neural networks. However, it is similar to

[7] `http://www.mturk.com/mturk/` (last access on 28/05/2017)

our solution proposal, because it considers clothes' attributes, occasions and time, namely seasons.

Both, [15] and [19], describe interactive mirrors that recommend outfits, not only appropriate to clothes' attributes, occasion, temperature and weather conditions, but also users' previous interactions.

In the system proposed by Kim [15], called UFC Mirror, users may or may not give initial clothing pieces to which the system has to find a combination appropriate to clothes' attributes, occasion, time, place and users' preferences. This system considers as clothing attributes, among others, colour, style, size, fabric. These characteristics are known through RFID tags placed in each clothing item. Besides clothes' characteristics, these RFID tags also allow to make relations between clothing pieces. In order to give an outfit suggestion, the system depends on the information of RFID tags. This is similar to our proposal, because our recommendation process will have in account the characteristics of each of the user's clothes.

The system proposed by Nagao et al. [19], called Mirror Appliance, can recommend outfits with top and bottom pieces and shoes. In order to obtain outfits suggestions appropriate to users' day/occasion, users should start by telling the system, through an ARToolKit marker, the type of event. While the system recognizes the marker and accesses the user database, it obtains the weather conditions and temperature of the current day. Considering occasion, temperature, weather and users' preferences, the system filters appropriate clothes present in the user database and suggests combinations, corresponding to full outfits or parts of outfits previously used under the same conditions. Additionally, this system also stores users' clothes in a virtual closet.

Zhao and Araki [26] propose a recommendation system capable of suggest, from all the user clothes, two outfits appropriated to user's profile and preferences, season and occasion. In general, the suggested outfits consists in four clothing items (hat, top and bottom pieces and shoes), unless a full-body piece is recommended. Similar to [19], besides the recommendation, this system also has a virtual closet. When users start to wear this system, they should fill in some personal data, such as nationality, age, occupation, gender, among others, and also photographs of all their current clothes, providing some characteristics such as season, type of clothing piece, brand and occasion where the user would use that piece. Optionally, users can also comment each clothing piece through natural language. In order to recommend appropriate outfits, this system takes into account occasion, users' profile and personal preferences, including previously selected clothes and favourites or outfits, and season of the year. If users do not like the suggested outfits, they can still select favourite clothing pieces. Unlike the systems described in Kim [15] and Nagao et al. [19], in addition to the virtual closet functionality and outfits recommendation, the system proposed by [26] also has the ability to suggest clothing pieces to buy, according to users' personal preferences, in terms of styles and colour, and according to what is trend, through information/images extracted from web.

All systems in this subsection consider, besides other aspects, clothing attributes and occasion in the recommendation process, being very similar to our proposal, because we will consider clothes attributes and occasion in our recommendation algorithm. From these, [4, 8, 15] and [26] are identical to our solution because also have a virtual closet. The recommendations systems proposed by Feng et al. [8] and [19] are similar to our solution because allow to give an initial clothing item that outfit must have. Additionally, the system proposed by Shen et al. [22] is very much alike our application as it allow users to textually describe occasion.

2.2.2 Finding Clothes and Outfits Through Natural Language

One of our main goals is to allow users to textually describe, in natural language, a clothing piece that they want to find in their virtual closet and the desired characteristics in the outfit suggested by the system.

In the previous section, 2.2.1, we mention the system proposed by Shen et al. [22]. This allows users to textually describe scenario and/or the emotional state that the user wishes to feel with the recommended outfit. Through Open Mind Common Sense [23], a collection of English sentences about common sense, ConceptNet [18], a tool that relates different concepts with each other, and a handcrafted file with all relations between brands, types of clothing items, materials, some occasions and styles, this system is capable to infer the appropriate style that the suggested outfit must have. The main idea of allowing users to textually describe what they want is similar to our proposal, however we want to only allow simple pre-defined sentences, since we will focus on the capsule wardrobe concept and outfit recommendation process.

Other examples of works related with describe textually, in natural language, clothes/outfits or concepts in general are [27] and [21].

Zoghbi et al. [27] propose a textual and visual clothing search system, that is, a system with two main tasks:

- Given an image of a garment, obtains words or sets of words that describe it;
- Given a set of words as a description of an outfit, finds an image of a outfit that satisfies the description.

The second task of this system is very similar to our main goal, however the several approaches to represent outfits' images, possible vocabulary and relations between images and vocabulary implemented and compared, besides needing to be trained with thousands of images and textual annotations, are much more complex than our proposal.

Rao et al. [21] propose a system that, given a natural language query in English can convert it to SQL and obtain the result of applying it to existing and previously known tables. This system uses knowledge of the syntax of SQL commands and a semantic grammar with all possible words, including their synonyms, and how they can be organized to form sentences. The algorithm behind this system consists of seven steps:

1. Conversion of the query in natural language to a list of words;
2. Tokenization according to the semantic grammar, associating an order number with each token;
3. Find the connectors/criteria tokens, in order to relate attributes and values, to specify conditions in the query in SQL;
4. Associate values with their attributes and tables;
5. Replace synonyms with their table attribute names;
6. Get intermediate query;
7. Convert intermediate query to SQL query.

One example given by [21] corresponds to "What is employee name with salary greater than 2000". This sentence is converted, by system proposed by Rao et al. [21], into intermediate query ((SELECT NAME! FROM (EMP (WHERE (SALARY >= 2000))))), which is then, as final step, converted into SQL query.

From all systems mentioned in this section, the system described in [21] is the most similar to what we want to accomplished. Similarly to our solution we developed a semantic grammar to interpret textual descriptions given by user to describe an desired outfit. From this interpretation, we are able to create outfits and verify which ones comply the requirements expressed by users through textual descriptions.

2.2.3 Similar Applications

Services or applications similar to our solution proposal, that means with the ability to support the creation and management of capsules wardrobe or recommend outfits, have been created and developed in the past years. In this section we present the most similar to our solution and/or the ones with higher classification and/or number of installations.

One example is *Cladwell Capsules*[8]. This is an online service that, just like our solution proposal, helps users create and maintain capsules wardrobe for each season. Similar to our proposal, it does not limit the number of clothing pieces in the capsule. *Cladwell Capsules* allows users to personalize their own capsules, get recommendations of clothing items to buy in order to complete them and information on how to match clothing items in order to obtain the most appropriate outfit to season and occasion.

With the intention to give users daily outfits recommendations, Cladwell also created and developed *Cladwell - Daily Outfits From Your Closet*[9]. This is an iOS application capable of give to users outfits recommendations according to weather and users' clothes. Users are also capable of seeing what they worn in the past days and the number of times they worn a specific item, being able to only keep the pieces they wear. Besides that, the application can also help users buy new clothing pieces that they really need, only when they need.

Cladwell Capsules and *Cladwell - Daily Outfits From Your Closet* are very similar to our main goals of help users create capsule wardrobe and get outfits recommendation according to weather and users' clothes. The main differences/disadvantages are:

1. *Cladwell - Daily Outfits From Your Closet* is only available to iOS;
2. Both *Cladwell Capsules* and *Cladwell - Daily Outfits From Your Closet* require a monthly subscription and are only for female users;
3. In *Cladwell - Daily Outfits From Your Closet*, users can not take photos of their own clothes, they only have the option to choose, from a database of clothing items' images the most similar.

Similar to *Cladwell - Daily Outfits From Your Closet*, there are also Android applications that automatically suggest outfits. Some examples are *Fashion You*[10], *What 2 Wear?*[11] and *Oshare Weather*[12].

Fashion You is an application that allows female users to store their own clothes in a virtual closet, create personalized outfits and get outfits recommendation according to season, style and type of clothes. With clothes' characterization in terms of category, subcategory, style (elegant or casual) and season (spring/summer or fall/winter), the application can automatically match clothes and suggest an outfit appropriate. Besides the outfits automatically suggested, users can also create their own personalized outfits for a specific season and numerous occasions. Additionally, *Fashion You* allows female users to create suitcases with different clothing items and outfits inside, easing the process of planning outfits to a trip, see outfits worn each day and statistics about clothing items and outfits saved and worn, closet value and colours in it. Identically to *Fashion You*, with the same functionalities, there is also an application to male users called *Fashion You Man virtual closet*[13]

What 2 Wear? and *Oshare Weather* are two applications that recommend outfits appropriate to temperature and weather conditions. There are two main difference between these two applications:

[8] `https://cladwell.com/capsules` (last access on 14/05/2017)
[9] `https://cladwell.com/outfits` (last access on 14/05/2017)
[10] `https://play.google.com/store/apps/details?id=com.fashionyou` (last access on 14/05/2017)
[11] `https://play.google.com/store/apps/details?id=com.iloiacono.what2wear&hl=pt_PT` (last access on 14/05/2017)
[12] `https://play.google.com/store/apps/details?id=jp.co.recruit.mtl.osharetenki` (last access on 14/05/2017)
[13] `https://play.google.com/store/apps/details?id=com.fashionyouman` (last access on 14/05/2017)

1. The latter only uses examples of clothes from application database to recommend outfits, while the former can also use users' clothes;

2. The latter considers different occasions/styles (feminine, casual, office and fashion), while the former does not consider any occasions/styles;

3. The latter only considers female users, while the former considers both male and female users.

In the Google Play Store, besides these Android applications capable of automatically create suggestions of outfits, there are also some simpler applications that consist only in virtual closets where users can manually create their own outfits. Some examples are *Closfy*[14], *Smart Closet*[15] and emphYour Closet - Smart Fashion[16]. All these three applications allow users to personalize occasions/styles considered in the outfits categorization as well as clothing categories (and subcategories if existed). *Closfy* stands out from the other two applications because it has a one-time paid service of outfit recommendation, where a team of stylists and fashion assessors can recommend personalized outfits for a certain event or day, with users' clothes and/or clothes available in stores.

Similar to these online services and mobiles applications, Amazon recently launch a new product called *Echo Look*[17]. This product is based on *Alexa*[18] [19]. *Alexa* is an intelligent system, launched by Amazon in 2014, capable of learning, that through users' voice can perform/answer calls and send messages, play music, provide information example about news, traffic and weather, among other things. *Echo Look* incorporates all *Alexa*'s functionalities and adds the abilities to take photos, record videos and compare two different outfits according to which looks better on users' bodies.

2.2.4 Summary

In this chapter we presented and analysed several works similar to want to accomplish, namely in terms of the capsule wardrobe concept, outfits recommendation systems, dominant colour extraction, finding items through natural language descriptions and similar online services and mobile applications available.

Several fashion recommendation systems have been developed in the past year considering only one or several of the following parameters in the recommendation process. Numerous combinations of those parameters have been explored in the previous works, as well as different approaches to learn combination rules. In section 2.2.1, we described some examples of recommendation systems. The most similar to our proposal correspond to systems described in [4, 8, 14, 15, 19, 22, 26], because, like our proposal, they consider clothing attributes and occasion in the recommendation process. From these, [4, 8, 15] and [26] also have a virtual closet like our solution. Furthermore, recommendations from systems proposed by Feng et al. [8] and [19] allow to give an initial clothing item that outfit must have. Additionally, the system proposed by Shen et al. [22] is also similar to our proposal in terms of allowing users to textually describe occasion. Additionally to these and although the systems described in [12, 17, 24] only consider clothing attributes and require images to train or coordinate clothes, they are similar to our proposal, because all receive as input an initial clothing item and in our proposal we will allow users to obtain outfits recommendations with a clothing item chosen by the user or simulate the

[14] `https://play.google.com/store/apps/details?id=com.agba.closfy&hl=pt_PT` (last access on 14/05/2017)
[15] `https://play.google.com/store/apps/details?id=com.rkk.closet` (last access on 14/05/2017)
[16] `https://play.google.com/store/apps/details?id=com.yourclosetapp.app.freecloset` (last access on 14/05/2017)
[17] `https://www.amazon.com/Echo-Hands-Free-Camera-Style-Assistant/dp/B0186JAEWK` (last access on 15/05/2017)
[18] `https://www.amazon.com/dp/B00X4WHP5E/ref=twister_B01KIOU214?_encoding=UTF8&psc=1` (last access on 15/05/2017)
[19] `https://techcrunch.com/2014/11/06/amazon-echo/` (last access on 15/05/2017)

purchase of a new clothing item. From these three, systems proposed by Lee and Lee [17] and Vartak and Madden [24] suggest complete outfits, while system in [12] only suggests top and bottom pairs.

In section 2.2.2, we highlighted one fashion recommendation system, described in 2.2.1, the system proposed by Shen et al. [22], that allows users to textually describe occasion, instead of using checkboxes. Besides this system we also described two relevant works, one related to e-commerce search, [27], and other related to natural language queries processing, [21]. In our proposal we will use a similar approach to the system proposed by Rao et al. [21] by letting users textually describe an outfit suggestion that they want to find/obtain.

Finally, in section 2.2.3, we describe and analyse some of already existent online services and mobile applications similar to what we want to create and accomplish. That we have knowledge of, the only online service or mobile application that, like our proposal, allows users to create and maintain capsules wardrobe is *Cladwell Capsules*[20]. In terms of automatically recommend outfits there are several online services and mobile applications, like *Cladwell - Daily Outfits From Your Closet*[21], *Fashion You*[22], *What 2 Wear?*[23] and *Oshare Weather*[24].

In this chapter, we mentioned and described several related works, similar online services and mobile applications, however neither of them simultaneous accomplish all our goals. In other words, a mobile application that allows to create and manage capsules wardrobes, suggest complete outfits appropriated to a given occasion and other requirements selected by users, visualize all the outfits worn or planned to be worn and simulate acquisition of new clothing items. This application will help optimize the process of combining clothes into appropriated and harmonious outfits, complying occasion and users specifications, making this task more simple and satisfying.

[20] `https://cladwell.com/capsules` (last access on 14/05/2017)
[21] `https://cladwell.com/outfits` (last access on 14/05/2017)
[22] `https://play.google.com/store/apps/details?id=com.fashionyou` (last access on 14/05/2017)
[23] `https://play.google.com/store/apps/details?id=com.iloiacono.what2wear&hl=pt_PT` (last access on 14/05/2017)
[24] `https://play.google.com/store/apps/details?id=jp.co.recruit.mtl.osharetenki` (last access on 14/05/2017)

3

Ultimate Capsule

As a final result of this work, our goal was a mobile application, Ultimate Capsule, capable of supporting the creation and management of three month capsules wardrobe and suggestions of complete outfits appropriated to a given occasion and/or other restrictions defined by users. Although we are limiting to a single platform, and there are some disadvantages related to native applications, we implemented this application as an Android native application, since, according to previous works, [6] and [13], a web app would require a constant internet connection (so would require more data consumption, having a negative impact on battery usage), have limited performance and access to device features, and existing platforms for creating a hybrid application still have limitations.

This application supports both male and female users and they can:

- Upload and characterize all current clothing items into the virtual closet;
- Visualize, edit and remove any clothing item in the virtual closet;
- Create a new minimalist capsule wardrobe for a specific season;
- Visualize all clothing items in some capsule wardrobe;
- Obtain personalized outfits recommendations of bottom and top pieces, or full-body, shoes and optionally accessories, according to colour and pattern matching, occasion and optionally according to a given clothing item or predominant colour;
- Early planning of outfits;
- Save and visualize outfits worn or to be wore in a specific date;
- Simulate purchase of new clothing items.

In order to support these tasks, our solution was divided into four main functionalities: **Virtual Closet**, **Minimalist Capsule Wardrobe**, **Outfit Recommendation** and **Purchase Simulation**. Additionally, with the intention to provide a more personal experience and obtain users information, we added a fifth functionality, **Profile**.

In the section 3.1, we will describe in more detail each functionality and, in 3.2, our solution architecture and algorithms.

3.1 Functionalities

As mentioned before, we divided our application into five main functionalities. In the following sections, 3.1.1 to 3.1.5 we will described each one in more detail.

3.1.1 Profile

In our solution we depended on some users information, more precisely gender and location for, for example, determine clothes categories and seasons start and end dates. Having this objective in mind and the purpose to support a more personal experience, we added a profile functionality. In this context, we allow users to create, see, edit and delete their profiles.

We defined the profile creation as the first step before starting using our application, since the remaining functionalities depend directly on information collected. For example, to define clothes categories or seasons start and end dates.

In our application, users are characterized by name, date of birth, gender and location. The first two, name and date of birth, are not entirely needed in our application, because we do not influence our solution functioning by them, but we decided to have them since it could be helpful in the future, for example in our recommendation or simulation algorithms. At this moment, the only informations that can influence our application are gender and location. The former, gender, has impact on clothes categories considered by our application, while the latter, location, determine each season start and end dates. Regarding location, we decided to ask users in which hemisphere they are, instead of country. We opted by this, since we only really needed to know in which hemisphere users are and presenting the entire list of countries, even if they were alphabetically ordered, would lead to a very exhaustive list of possible countries, being very inefficient, time wasting and frustrating to users, and also would imply the implementation of a function that matched selected country with correspondent hemisphere.

3.1.2 Virtual Closet

In order to create minimalist capsules wardrobe and give outfits recommendation with users' clothes, we had to acquire knowledge about them, meaning we needed to save each clothing item information. In other words, we needed to support the creation and management of a virtual closet, so we added the Virtual Closet functionality to our solution. In this, users are able to see all their clothes and add new, modify or remove existing ones.

The users' clothing items in the virtual closet were defined by their own image, type (and category if applicable), colour, type of pattern (and pattern scale if applicable), season and occasion. In terms of clothes images we offer two options to users: the possibility to take a photography of the clothing item or select a image from device's gallery. In terms of clothes types, either it is male or female user, we considered **top**, **bottom**, **full-body**, **shoes** and **accessories**. We did not consider workout a clothing type since there was no consensus between previous works and applications analysed. In terms of clothes categories, we have in account if it is a male or female user, having different possible categories for each one. The clothes categories that we considered for male users can be seen in table 3.1 and for female users can be seen in table 3.2. The clothes types and categories choices were based on previous works and similar applications mentioned in section 2.2 and also some research on online shopping stores like Massimo Dutti[1], Stradivarius[2] and Zara[3]. We do not allow clothes types and categories

[1] `https://www.massimodutti.com/pt/en/` (last access on 15/05/2017)
[2] `https://www.stradivarius.com/pt/en/` (last access on 15/05/2017)
[3] `https://www.zara.com/pt/en/` (last access on 15/05/2017)

customization, like some similar applications, because these are used in our recommendation algorithm while calculate appropriate outfits.

Table 3.1: Clothes categories for male users by clothes types.

Types	Tops	Bottoms	Full-body	Accessories
Categories	Jackets	Jeans	Suits	Bags and Backpacks
	Shirts	Trousers		Ties and Bow Ties
	Long sleeve tops	Shorts		
	Short or sleeveless tops			
	Sweaters			

Table 3.2: Clothes categories for female users by clothes types.

Types	Tops	Bottoms	Full-body	Accessories
Categories	Jackets	Jeans	Dresses	Bags and Backpacks
	Shirts and Blouses	Shorts	Jumpsuits	Scarves
	Long sleeve tops	Skirts	Suits	
	Short or sleeveless tops	Trousers		
	Sweaters and Cardigans			

Regarding colours, we had in account the twelve colour wheel that is normally used for combine clothes. As mentioned in section 2.1, this colour wheel has three primary colours, three secondary colours and three tertiary colours that are: **yellow**, **yellow-orange**, **orange**, **orange-red**, **red**, **pink**, **violet**, **purple**, **blue**, **cyan**, **light-green** and **green**. Additionally, we used the colours considered as neutrals in several websites mentioned also in section 2.1, namely **black**, **grey**, **white**, **brown** and **beige**. In terms of patterns, we considered: **plain/no pattern**, **striped**, **checkered**, **plaid**, **dotted**, **floral**, **animal** and **other**. We decided to have a pattern option named **other**, with the intention to be able to support all patterns that may exist but were not considered as more commons in the links mentioned in section 2.1. Concerning patterns, we additionally defined the attribute pattern scale, as **small** or **big**, to all patterns except **plain/no pattern**, with the intention of being capable to combine different patterns. Regarding occasion, each clothing item can have more than one of the following occasions associated: **casual day**, **work day**, **meeting**, **party/dating** and **formal event**. Similar to clothes types and categories, occasions and type of patterns were based on previous works and similar applications mentioned in section 2.2 and also some research on online shopping stores. Just like clothes types, we also did not consider workout as an occasion, since there was no consensus between previous works and applications analysed. In terms of seasons, since the capsules wardrobe created by users are valid for three months, we considered **spring**, **summer**, **autumn** and **winter**. Just like occasions, users will have the option to choose several seasons for each clothing item.

3.1.3 Capsule Wardrobe

As mentioned before one of our main goals was to support the creation and management of three month capsules wardrobe, so we created the Capsule Wardrobe functionality. In this, users are able to see all their capsules and also create new, edit and delete existing ones. Besides, since we allow to have multiple capsules for the same season, users can select which capsule should be considered as current for the present season.

The capsules in our solution are characterized by name, season, start and end dates and number of clothing items by type. Since in our solution we allow capsules with the same season and number of clothing items, we needed an unique attribute. With the intention to distinguish between capsule with same season and number of season, we added the name attribute to capsules. Concerning season, this attribute was defined because our intention was to create capsules with three month duration. In terms of capsule start and end dates, we realised that there were two options. We could let users define each capsule start and end dates, allowing to have capsules with less or more than three months and with the same season but different dates, or define ourselves the specific start and end dates of seasons. We chose the last option, because our main goal from start was to allow create capsules dedicated to each season in other words, with no more and no less than three months duration. With a little research [4][5][6], we come to realised that, from all possible ways to define each season start and end dates, there were two that are more commonly used in real world. These two are:

- meteorology method, where seasons start and end days always correspond, respectively, to the first day and end day of specific and clearly defined months
- astronomic method, where seasons are determined by equinoxes and solstices days

We decided to chose the meteorology method, over astronomic, since the former only required to know if users live in north or south hemisphere, while the latter would required to know both the country where user live and equinoxes and solstices dates for current year and for selected country, since they vary from year to year, depending on location. In respect to capsule clothes, we also had two possible scenarios, ask users the total number of clothes and we define how we divide this total into the several types, in other words, which are tops, bottoms, full-body pieces, shoes and accessories, or ask users how many clothing pieces they want of each type. We opted for the last scenario, since from the research made we observed there are no consensus about which clothes actually count to total number of clothes in capsules neither about how to divide this total for the several clothes types. So, we allow users to customize the number of capsule clothes in terms of tops, bottoms, fullbody pieces, shoes and accessories, ranging from zero to the number of existing clothes for each type.

From information specified by our users about new capsules, more specifically season and number of capsule clothes, our solution is capable to create capsules with clothing pieces that maximize the number of possible outfits, complying with number of pieces asked by users. When season or number of pieces of a existent capsule changes, our solution recalculates capsule's clothing items.

3.1.4 Outfit Recommendation

Similarly to the module Capsule Wardrobe, in order to accomplish the goal of give outfits suggestions appropriate to occasion and users' restrictions, we added the Outfit Recommendation functionality. In this, users can obtain an outfit suggestion for a specific day, through textual description or attributes selection. Regarding attributes selection, we allow users to specify occasion, clothes types of the outfit, one specific clothing piece that the outfit must have, dominant colour and if it should have accessories or not. With information specified by our users, through textual description or attributes selection, and clothes from current capsule wardrobe, or, if there is none defined, from virtual closet, we obtain a list of appropriated outfits, from which we choose one. From the suggested outfit, users can accepted and save it to a database or ask for another suggestion with same characteristics. The suggested outfit saved into database for current day or selected day can be seen in the initial screen.

[4] `https://www.timeanddate.com/calendar/aboutseasons.html` (last access on 16/02/2018)
[5] `http://www.bbc.com/news/uk-northern-ireland-29014886` (last access on 16/02/2018)
[6] `https://www.livescience.com/25202-seasons.html` (last access on 16/02/2018)

3.1.5 Simulate Purchase

Identically to Capsule Wardrobe and Outfit Recommendation, to achieve our goal of helping users to buy only clothing items that combine with maximum number of clothes possible, our solution has a functionality where users can verify how a new piece, that they are considering to buy, combines with their virtual closet clothes. In other words, we created the Simulate Purchase functionality. In this, through a process similar to adding a new clothing piece into virtual closet, users can observe the number of outfits possible with this new piece and a small sample of those outfits. Considering these informations, users are more capable of making a conscious decision about buying (or not) a certain clothing item. With the intention to avoid duplicate and redundant actions, at the end of this process, we allow users to add the simulated piece into the virtual closet.

3.2 Solution Architecture

To develop our solution we created and developed five modules and five algorithms. The modules store the correspondent logic for each functionality mentioned before, **Profile**, **Virtual Closet**, **Minimalist Capsule Wardrobe**, **Outfit Recommendation** and **Purchase Simulation**, while the algorithms correspond to **`ClothesCoordination`**, **`CapsuleCreation`**, **`FashionNaturalLanguageQueries`**, **`OutfitRecommendation`**, and **`SimulatePurchase`**. Additionally, we have created a SQLite database where we store all users' clothes, capsules wardrobe created and outfits worn and to be wore. This database is locally stored in users devices.

In sections 3.2.1 to 3.2.5, we will describe all five algorithms implemented and in section 3.2.6 we will detail implemented database.

3.2.1 Clothes Coordination Algorithm

In the `CapsuleCreation`, `OutfitRecommendation` and `SimulatePurchase` algorithms, we needed to know if two clothing items combine or not, considering their colours and patterns types and scales. With the intention to avoid duplicate code through these three algorithms, we abstracted this task into one algorithm called `ClothesCoordination`. In this, we implemented the rules mentioned in section 2.1 used normally to combine clothes in terms of colours and patterns. To abstract these rules and allow to be used by other modules, we implemented two functions that tell us if two clothing pieces combine in terms of colours and patterns, respectively functions `clothesPatternCoordination` and `clothesColoursCoordination`.

3.2.2 Capsule Creation Algorithm

In our Capsule Wardrobe functionality, to abstract our capsule creation methodology, we implemented an algorithm, called `CapsuleCreation`, which is capable of choosing the clothing pieces that maximize the number of possible outfits, complying season and number of clothes by type. The result of this is a list of clothing items that constitute a capsule.

3.2.3 Fashion Natural Language Algorithm

One of our goals as functionalities in our application was to be able to process textual descriptions in natural language regarding outfit suggestions. In other words, our goal was to be able to, given an outfit description in natural language, transformed it into a useful structure that we can used to calculate the most appropriated outfit. In order to accomplish this we implemented an algorithm called

FashionNaturalLanguage, which corresponds to a natural language parser related to fashion, which input is a natural language description of an outfit and the result corresponds to pairs of attribute values predefined by us. In these, we store outfit general attributes, namely dominant colours, patterns and/or appropriated occasions, as well as outfit specific pieces descriptions.

3.2.4 Outfit Recommendation Algorithm

Another of our main goals was to calculate the most appropriated outfit for specified requirements by users, so we implemented an algorithm called OutfitRecommendation. This is subdivided into two, depending if recommendation is through natural language textual description or attribute selection. In the former scenario, natural language textual description, the first step correspond to call FashionNaturalLanguage algorithm to interpret textual description, and then, based on result calculate the most appropriated outfit. In the latter scenario, attributes selection, the algorithm based on required attribute occasion and optional attributes outfit pieces types, specific piece, dominant colour and if it should have accessories or not, calculate the most appropriated outfit.

3.2.5 Simulate Purchase Algorithm

Additionally to capsules creation and outfit recommendation, our goal was to help users decide if they should buy a clothing piece or not based on possible outfits with it. To calculate possible outfits with specified clothing item, we created an algorithm called SimulatePurchase. This, with knowledge of new clothing piece type, category, colours, pattern type (and scale if applicable) and appropriated seasons and occasions, determines possible outfits with specified item. Contrarily to OutfitRecommendation that allows users to determine if it should have accessories or not, this algorithm returns always outfits with accessories.

3.2.6 Database

As mentioned above, we developed a SQLite Database to store all our application data, from users personal information to their clothes, capsules wardrobe created and outfits worn or to be wore. We organised our database in several different tables, as can be observed in figure 3.1.

Regarding users personal information, we create table users, responsible to store users names, birth dates, gender and locations. In other words, the table users has five columns, one for the unique identifier and the other four to store respectively name, birth date, gender and location. Since in our application context we do not allow to logout/login using different users profiles, this table consists, when a user is registered, to one single row. Besides storing users information, this table is also helpful while deciding to which screen should the application go after start, since it allows to verify if there is a user created or not.

Concerning clothes, we created clothes. In this, we store all clothes attributes with single value, in other words, clothing piece type, category, image source, image data, pattern type and pattern scale. Besides these, we also associate an unique identifier to each clothing item. Concerning clothes attributes with multiple values possible, we had two options: concatenate all values into a string and stored it in clothes table or create additional tables to store one row for each value. Having in account that we would need to query clothes in terms of one or several seasons, colours or occasions, we opted for the second alternative, since it would be more efficient this way. So, we created tables for colours, seasons and occasions, respectively occasions, colours and seasons, and tables to relate seasons, colours and occasions with clothes, correspondingly clothes_occasions, clothes_colours and clothes_seasons. For the first three tables, occasions, colours and

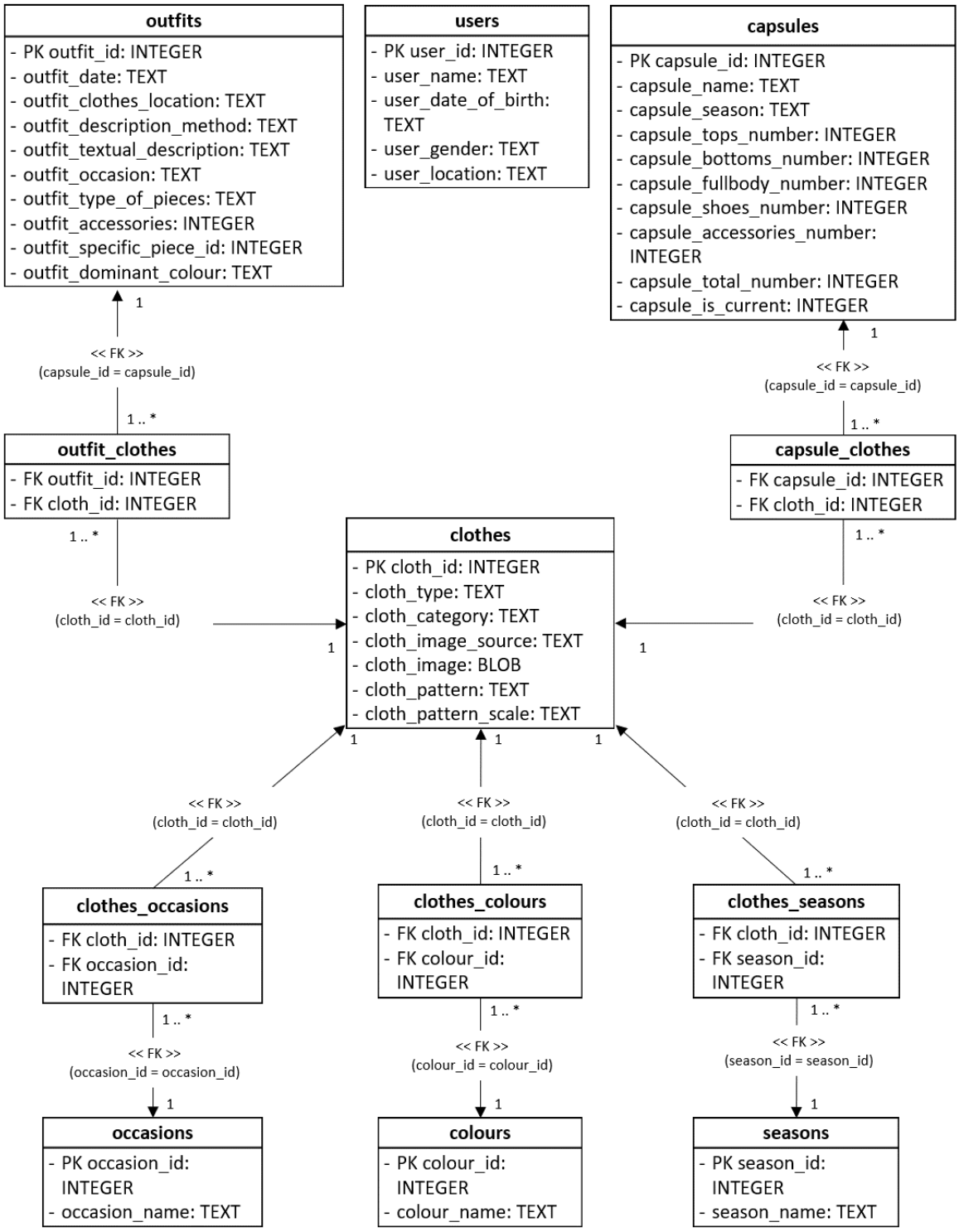


Figure 3.1: Scheme of SQLite database developed to store our solution data.

`seasons`, have as columns the attribute identifier and attribute name. In these, we store, respectively, all possible names of colours, seasons and occasions, associated with an unique identifier. The three latter tables, `clothes_occasions`, `clothes_colours` and `clothes_seasons`, allow to make relation between clothes and their occasions, colours and seasons, correspondingly. In these table we have

as columns clothing piece unique identifier and respective attribute unique identifier. These identifiers correspond to foreign keys from tables `clothes`, `occasions`, `colours` and `seasons`. In other words, to store each clothing piece season, colour and occasion, a row with clothing piece unique identifier and correspondent attribute identifier is added in one of these latter three tables, depending on the attribute.

With the intention to save informations about the capsules created, we developed a `capsules` table. This table have as columns the capsule unique identifier, name, season, number of clothes by type, namely tops, bottoms, full-body pieces, shoes and accessories, and also the total number of pieces. Additionally to these informations regarding capsule characterization, since in our application we defined the existence of a current capsule, we store an attribute, in column `capsule_is_current`, which is zero if the capsule is not current and one, otherwise. With intention of not producing redundant data in database, similarly to clothes seasons, colours and occasions, we created a table `capsule_clothes` to store capsule clothing items. For each clothing piece in a capsule, it is added a new row in this table with capsule identifier and clothing piece identifier.

To save outfits suggested by our application, we created `outfits` table. The first two columns of `outfits` table correspond to unique identifier and date. In our application, our goal is to recommend outfits based on current capsule clothes. However, if users do not have a current capsule selected, they still can have outfit suggestion with virtual closet clothes. To distinguish these two scenarios, we added the third column, `outfit_clothes_location` into `outfits` table.

Since, in English version, we allow to describe outfits with natural language textual descriptions or through attribute values selection, we needed to store both methods informations. We started by addind the `outfit_description_method` column into `outfits` table. Regarding textual description method, to save outfit natural language description given by user, we added the column `outfit_textual_description`. Concerning attribute values selection method, we had as required characteristic occasion and as optional if it is with or without accessories, type of outfit pieces, specific clothing item identifier, dominant colour and if it is with or without accessories. To save these informations, respectively, we added the following columns: `outfit_accessories`, `outfit_occasion`, `outfit_pieces`, `outfit_specific_piece_id` and `outfit_dominant_colour`. The first column, `outfit_accessories`, similarly to `capsule_is_current` column in `capsules` table, stores zero if outfit has no accessories and one, otherwise.

Identically to `capsule_clothes`, with the intention to save outfit clothes without creating redundant items in the database, we developed `outfit_clothes` table. For each clothing piece that constitute the outfit, it is added a new row in this table with outfit identifier and clothing piece identifier.

4

Solution Development

Our intent was to build an application with an intuitive graphical user interface, easy to use and learn. In order to explore and compare different design alternatives, in a short period of time with a low modification cost, we followed an iterative and incremental design approach. With this we started by creating some storyboards of the main tasks, in order to define the different states of our application, and then, incrementally, we added more complexity, going from a low-fidelity and non-functional prototype until a final product. More specifically we have:

1. Defined main tasks and created storyboards for them;
2. Created low-fidelity non-functional prototypes (paper prototypes);
3. Evaluated low-fidelity non-functional prototypes;
4. Created functional prototypes;
5. Performed a mid evaluation on our functional prototypes;
6. Improved our functional prototypes considering feedback from mid evaluation;
7. Evaluated functional prototypes.

4.1 Main Tasks and Storyboards

The first step in our solution development process consisted of defining main tasks and creating storyboards to them. We started with storyboards, since, according to Fonseca et al. [9], they can show the sequence of steps required to support some tasks, allowing to see and understand how our application will flow and react to users' interaction.

We consider that the most important tasks that our solution would support were: upload a new clothing piece, create a new capsule wardrobe, obtain a outfit recommendation and simulate purchase of a new clothing item. We consider these because are the most critical: either they are more complex or more commonly performed by the user.

In figure 4.1, we can see the several steps that will be required to upload a new clothing item: take new picture, choose from gallery or from collection of pre-existing clothes' images, choose pattern type,

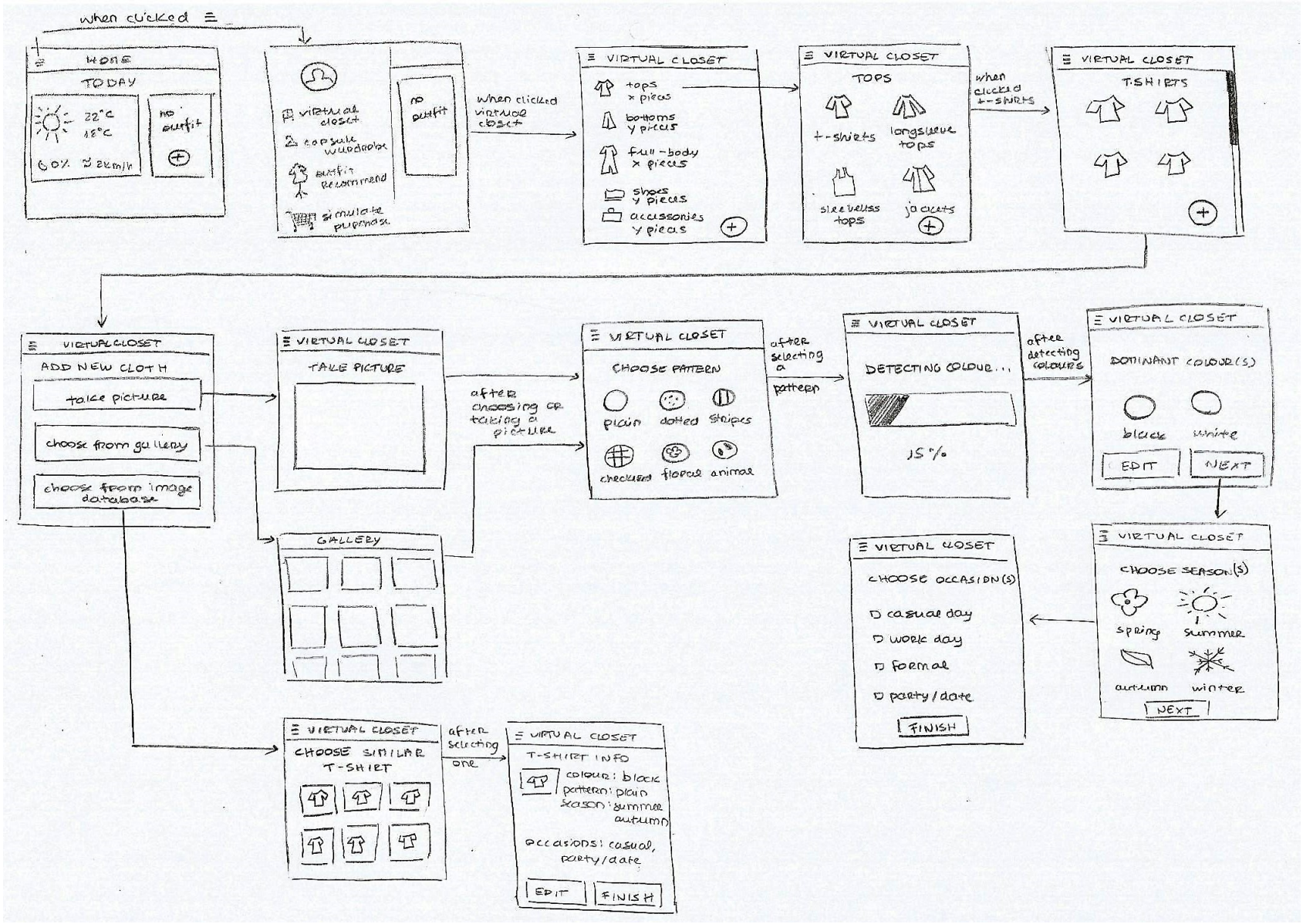


Figure 4.1: Storyboard for upload a new clothing piece

verify if dominant colours extracted are correct, select season(s) and occasion(s). Despite not being represented, users will also be able to add a new clothing item from closet overview or from clothes types, where type and/or category are not known. In this case, users must select the type (and category if applicable) of the clothing item.

In appendix figure A.1 we represent the steps to create a new capsule wardrobe, namely choose season, number of clothing items and name of the capsule. After creating the capsule, users will be able to visualize, edit or remove clothes and outfits of the capsule wardrobe.

In appendix figure A.2, we present the flow of both options to obtain a outfit suggestion, textual description or attributes selection. In terms of attributes selection, users are forced to choose an occasion, however they can skip dominant colour and clothing item selection. In both cases, as a result, users will get an outfit suggestion, which they can accept or ask for another.

In appendix figure A.3, we present the several steps users must do to simulate the acquisition of a new clothing item. Similarly to upload a new clothing item, users must select the type (and category if applicable) of the clothing item, take new picture, choose from gallery or from collection of pre-existing clothes' images, choose pattern type, verify if dominant colours extracted are correct, select season(s) and occasion(s). Additionally to the process of uploading a new clothing item, in the case of simulating a new purchase, users will also able to choose from match with clothes from current capsule wardrobe or virtual closet. In the end, users will be able to see how many outfits are possible to make with the initial clothing item.

4.2 Low-Fidelity Non-Functional Prototypes

As second step in our solution development process we created low-fidelity and non-functional prototypes. Since our main goal was an android application to be used in daily life of our users, we choose to use a smart phone device frame.

We started by designing our side menu and home screen low-fidelity and non-functional prototypes. After those, having in account our main tasks defined in section 4.1, we have chosen to implement four of our five main functionalities, namely Virtual Closet, Capsule Wardrobe, Outfit Recommendation and Purchase Simulation. More specifically, we decided to implement one example of each main task associated to these functionalities:

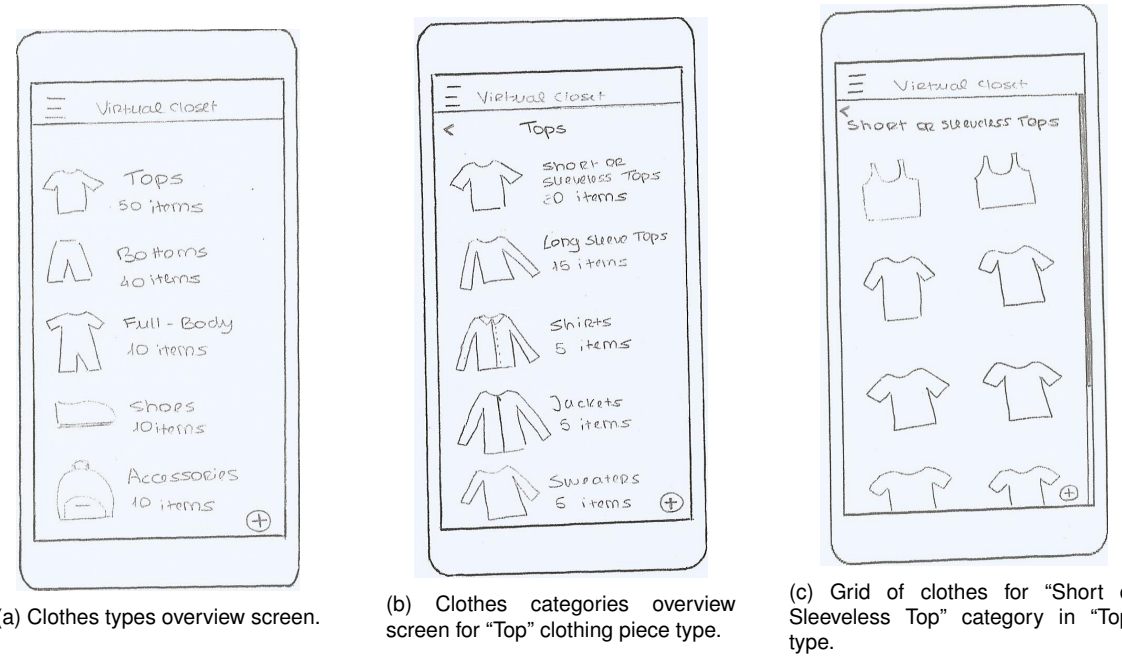
1. Add a new short sleeve top, with black and white stripes, taking a new picture. This top can be used in spring or summer, in casual days;
2. Create a new summer capsule wardrobe with 30 clothing pieces, called Summer Capsule;
3. Obtain a outfit recommendation to a party, today, June 26th, having as dominant colour blue, without any piece in particular;
4. Simulate the acquisition of a new long sleeve top, plain, pink, taking a new picture, into the virtual closet. This long sleeve top can be used in spring, in casual days and parties.

In this stage, having in account that we wanted our application to be available to everyone, independently from where users are, we decided to design our low-fidelity and non-functional prototypes in English.

4.2.1 Virtual Closet

For the Virtual Closet functionality we implemented screens related to how the user clothes would be organized in virtual closet and related to adding a new clothing piece to it.

In terms of virtual closet organization, just like in storyboards in section 4.1, we considered that all users' pieces would not be shown all together, but instead will be organized and filtered by types and categories. In other words, we organized the virtual closet hierarchically in terms of types and categories, so users must select the type and category of clothes that they want to see. In the paper prototypes, we implemented this organization by creating three screens as shown in figure 4.2. When virtual closet functionality is selected from side menu, the first screen shown corresponds to sub figure 4.2a where available clothes types are shown. After choosing the type, it is shown the screen corresponding to sub figure 4.2b with all available categories in the selected clothing type. Finally, after selecting clothing piece category, it is shown the sub figure 4.2c, with a grid view of users' clothes.



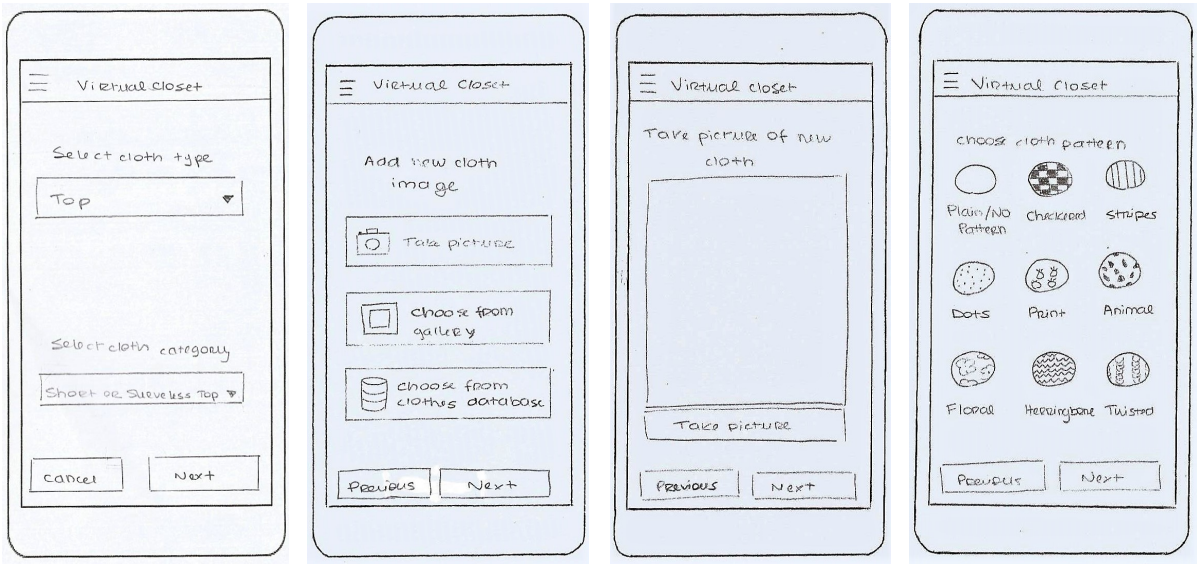
(a) Clothes types overview screen.

(b) Clothes categories overview screen for "Top" clothing piece type.

(c) Grid of clothes for "Short or Sleeveless Top" category in "Top" type.

Figure 4.2: Paper prototypes for virtual closet organization

In order to allow users to add a new piece in the virtual closet from different places, in all screens of the virtual closet organization, sub figures 4.2a, 4.2a and 4.2a, we added a plus icon that links to the screen corresponding to sub figure 4.3a or 4.3b, depending if user has already chosen both clothing piece type and category or not. In case of user has not chosen clothing piece type and/or category and presses the plus icon, we redirect to screen corresponding to sub figure 4.3a, where user can choose the new piece type and category from two drop downs lists. We decided to have these two characteristics in the same screen because they are directly related. If user was seeing the clothes that possesses in the virtual closet for a certain clothing piece type and category and presses the plus icon, we redirect to screen corresponding to sub figure 4.3b, where user has the possibility to take a new picture of the new piece, select a image from gallery or choose one clothing piece from a default clothes database. In this step, we have choose to have these three options represented by their names and related icon to be more easily identified each option by the user. In our example, while implementing these prototypes, we only considered the case of taking a new picture. For this purpose, we designed the screen corresponding to figure 4.3c with a space reserved to the image taken and a button called "Take Picture". After this step, user is asked about new piece pattern, in sub figure 4.3d. In this phase, is presented all patterns that we defined as possible in subsection 3.1.2, represented by theirs names and a small example image. After choosing the pattern, as defined in section 4.1, our solution would extract dominant colours from new clothing item image using an algorithm. While our dominant colour

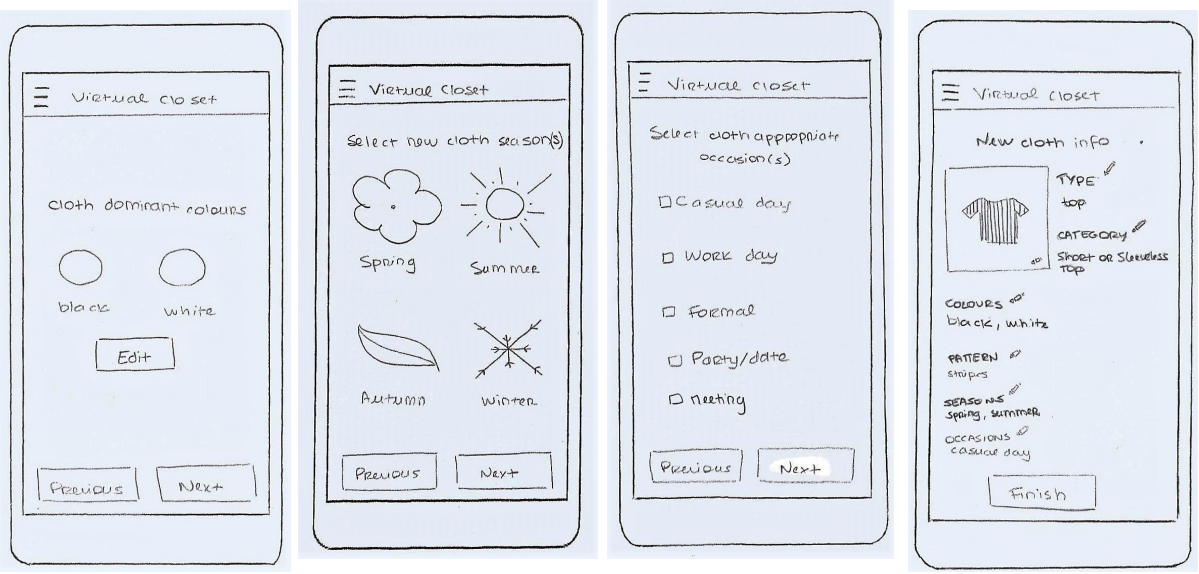


(a) Choose new clothing piece type and category.

(b) Choose new clothing piece image source.

(c) Take picture of new clothing piece.

(d) Select new clothing piece pattern.



(e) Confirm new clothing piece dominant colours.

(f) Choose new clothing piece appropriated season(s).

(g) Choose new clothing piece appropriated occasion(s).

(h) Confirm new piece to virtual closet.

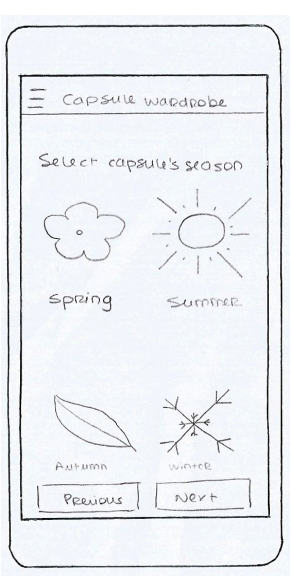
Figure 4.3: Paper prototypes for adding a new piece into virtual closet.

algorithm is working, we decided to have a progress bar in order to indicate the progress to users. After the algorithm is finished, our application would show the screen corresponding to sub figure 4.3e, where the dominant colours of the new clothing piece are shown, allowing users to edit them if necessary. After selecting new clothing piece pattern and confirm dominant colours, users must select appropriated season(s) and occasion(s). For this, we designed the screens corresponding to sub figures 4.3f and 4.3g. In case of selecting the clothing item appropriate season(s) we choose to have a grid view with a icon and name for each possible season that can be selected. Since for occasion there is no defined icon for each, in case of select appropriated occasion(s) screen we choose to only have each name with a check box before. After all these steps, just like designed in storyboards, we considered important to have a confirmation step like figure 4.3h, where users would be able to see every information for the new clothing piece to be added in the virtual closet and if necessary edit it. When everything is correct, users

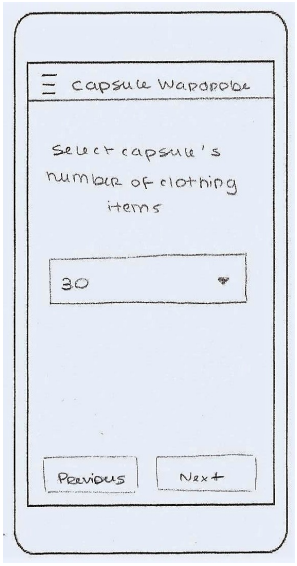
can add the piece, or cancel, and application returns to the screen from where the process begins.

4.2.2 Capsule Wardrobe

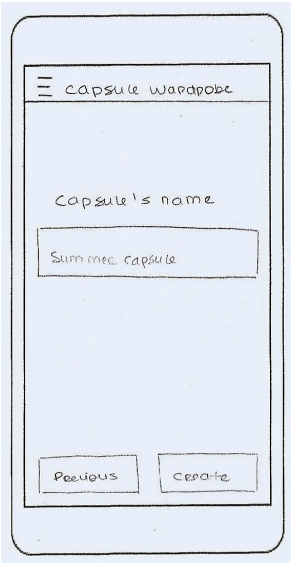
For the Capsule Wardrobe functionality we implemented screens related with the task of adding a new capsule and exploring the capsule created. In terms of capsule creation, when users select Capsule Wardrobe from the side menu, they will be redirected to one of the two possible screens, depending if they have already create a capsule or not. As defined in storyboards, we decided to simulate the case where user does not has any capsule created yet. In this, we present a message saying that there is no capsules created yet and an add button that allows the user to create a new capsule. By clicking in this button, users are redirected to the screen corresponding to sub figure 4.4a where they can select the season that will be associated to the new capsule. After this step, users are redirected to sub figure 4.4b where they can choose the maximum number of pieces of the new capsule from a dropdown list. We decided to have a dropdown list instead of a simple text field, in order to show immediately all possible values, instead of letting users try to write a number and get an error saying that it is too low or too high. As a final step, sub figure 4.4c is shown and users can write capsule's name in the text box and press the button "Create". With the intention to show progress to users, while the application is creating the capsule, it is shown a screen with a progress bar. After the capsule wardrobe has been created, users are presented with a screen where they can chose to explore which clothing pieces are part of the capsule as well as the possible outfits or finish the process. In the latter case, by finishing the creation of a new capsule through the "Finish" button, users will be redirected to the screen where it can be seen the new capsule created.



(a) Capsule wardrobe overview.



(b) Select new capsule wardrobe max number of pieces.



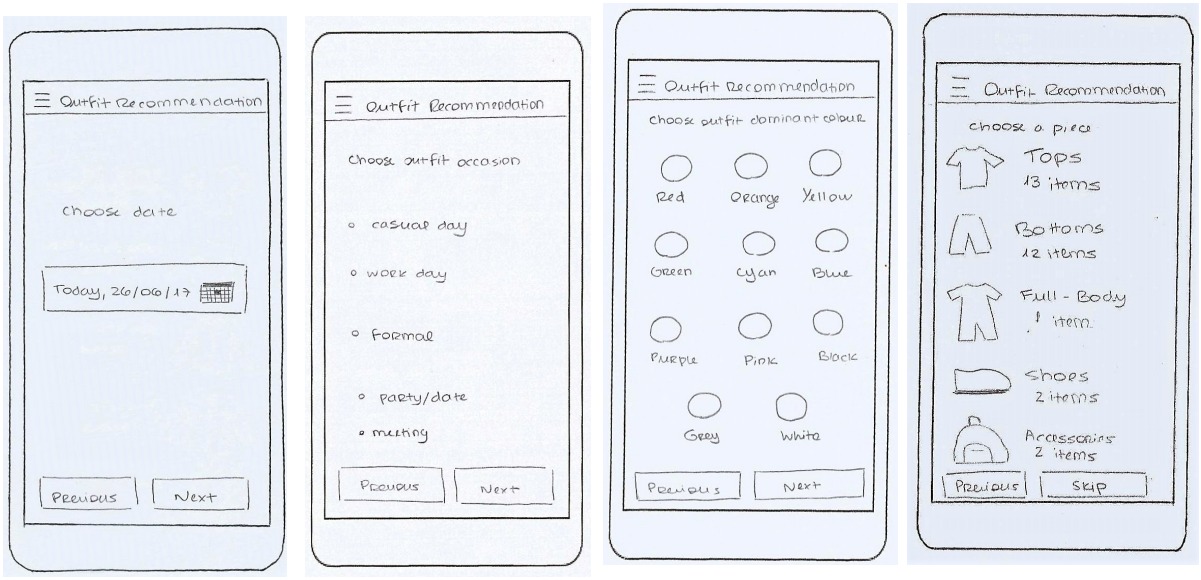
(c) Write new capsule wardrobe name.

Figure 4.4: Paper prototypes for creating a new capsule wardrobe.

4.2.3 Outfit Recommendation

For the Outfit Recommendation functionality we implemented screens related to obtain a new suggestion, namely selecting the date, occasion and, if the users want, dominant colour and/or specific piece. In terms of selecting the date we, like in 4.1, decided to have a screen dedicated to that corresponding to sub figure 4.5a, having a field where is presented the date selected, including weekday,

day, month and year. After this step, we would ask users to choose how they want to describe the outfit, through textual description or attribute selection. Since our main goal in this phase was to test our design, and the textual description method consists in having a text field where users can use their own words to describe the outfit, we choose to skip this step, not representing it in the low-fidelity and non-functional prototypes. Instead, after selecting the date, our prototypes transit to the step of selecting outfit occasion, in other words, to sub figure 4.5b. In this, we show to users a list of occasions from where they must choose the occasion where they want to use the outfit to be suggested. As mentioned in section 3.1.4 in Chapter 3, this is a mandatory field that can not be skipped like the two next attributes: dominant colour and specific piece. For the dominant colour attribute, we designed, as mentioned sub figure 4.5c, where we present a grid of colours with a small portion of the colour and its name and, most important, a button that allows users to skip this step. Similarly, for a specific clothing piece we designed sub figure 4.5d with a skipping button and a layout resembling to virtual closet organization where users can navigate, selecting type, category and then a clothing piece. From this step, whether selecting a specific piece or skip, users are redirected to a screen with a progress bar while the application is calculating the most appropriated outfit. Soon as our application finds the perfect outfit, it is shown the screen where users see the outfit suggested, having the option to edit, accept or ask another. If users decide to accept it, in background, our application would save the outfit in the database and users would be redirected to the home screen of the application. If users ask another outfit by clicking the button "Other", the application would return to the progress bar screen while calculating another appropriated outfit.



(a) Select outfit date.

(b) Select outfit occasion.

(c) Select outfit dominant colour(s).

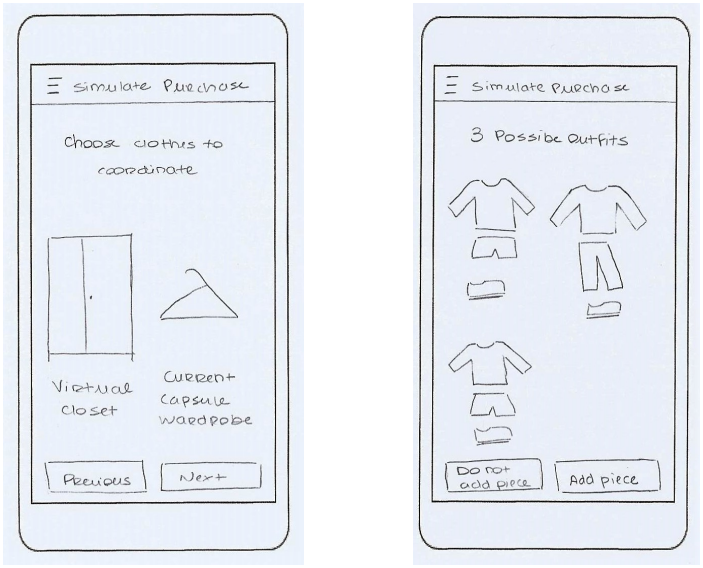
(d) Select outfit specific piece.

Figure 4.5: Paper prototypes for obtaining a new outfit recommendation.

4.2.4 Simulate Purchase

For simulating the acquisition of a new clothing piece, we implemented screens related to describe the new clothing piece and choose where to simulate the new piece. In terms of describing new clothing item, we maintained a similarly look to add a new piece to the virtual closet in order to have consistency in our application. In other words, the screens to select clothing piece type, category, image and image source, pattern, dominant colour(s) and appropriated season(s) and occasion(s) of the simulation

clothing piece are practically identical to respective screens from virtual closet with the exception of the title at the top of the screen, which says “Simulate Purchase” instead of “Virtual Closet”. In terms of choosing the simulation location we designed the screen corresponding to sub figure 4.6a. In this, it is shown two options to users, simulate the new piece in virtual closet or current capsule. After this, users are redirected to a screen with a progress bar while the application is calculating the possible outfits. When finished, sub figure 4.6b is shown with all possible outfits to the piece simulated. Based on the outfits returned, users can decide to either add, or not, the piece to the location selected by clicking one of the respective buttons, “Add piece” and “Do not add piece”.



(a) Choose simulation location. (b) Simulation result.

Figure 4.6: Paper prototypes for simulate a purchase of a new clothing piece.

4.3 Functional Prototypes

After having created and evaluated the low-fidelity non-functional prototypes, we started to develop our functional ones. Considering that our goal was to create an intuitive, easy to learn and use application, we chose to start implementing interface related to our solutions functionalities iteratively, and only after having part of it implemented, started to create our algorithms described in sections 3.2.1 to 3.2.5 in chapter 3.

In the following sections,4.3.1 to 4.3.7 we will describe our functional prototypes development in terms of functionalities and their interfaces, while from 4.3.8 to 4.3.12 we will describe algorithms we developed.

4.3.1 Project Structure and Organization

As mentioned at the beginning of chapter 3, to create our solution we have chosen to implement an Android application. So our first step to start implementing our functional prototypes was to create a new project in the Android Studio platform [1]. We defined our application name, company domain, package name and project location as well as platforms and android versions supported by our solution. In terms of platforms, since our goal was an application for mobile devices like android phones and tablets, we

[1] `https://developer.android.com/studio/` (last access on 02/05/2018)

selected the option Phone and Tablet. Regarding android versions and APIs, we have chosen to support API 16 and latter, since, according to Android Studio platform, this way our solution would be compatible with more than 90% of devices. Besides those settings, we had to chose our first Android activity. We had chosen a Navigation Drawer Activity since this way it would create automatically our side menu layout and logic. After these steps, we had an empty Android project with a default side menu created.

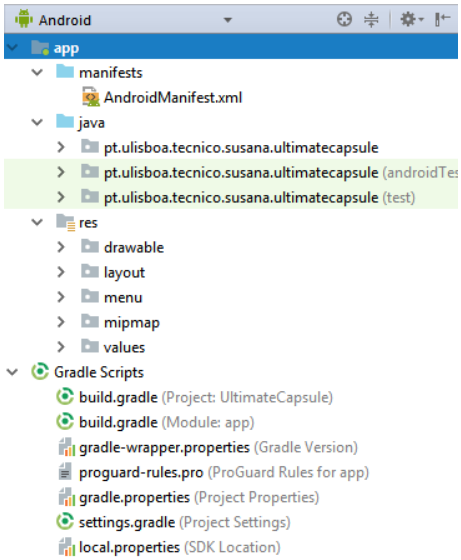


Figure 4.7: Android project structure.

The structure of our project in Android project view can be seen in figure 4.7. We can observe that there are two main modules: app and Gradle Scripts.

According to Queirós [20], the former module, app, is responsible to store all our solution information, java classes and resources, respectively in manifests, java and res folders, while the latter, Gradle Scripts module, is responsible to store all our solution configuration files.

In manifests folder exists the AndroidManisfest.xml that, as said in [20], stores our application information relatively to, among other things, components, permissions used/required and libraries used. In our case, in this file we have defined camera and gallery permissions as required and our application activities.

In the java folder, we have created sub folders to divide our future classes into activities, algorithms, helper, model and sql. As the names indicate the first two, activities and algorithms, are to, respectively, store all classes responsible for applications screens and algorithms that we implemented for our application (for example the natural language algorithm). The remaining sub folders, helper, model and sql, store all our classes related to our database. More specifically, helper is responsible to check data before putting into database, model stores the characterization of each element saved in the database and Structured Query Language (SQL) stores the SQLiteOpenHelper class accountable to create and manage our database content mentioned in 3.2.6.

Concerning res folder, this is divided, by Android, into five sub folders, in particular in drawable, layout, menu, mipmap and values.

In Queirós [20], drawables are described as an abstraction to something that can be sketched and one of their potential uses is to create scalable images, without loosing quality. We took advantage of this and we used the drawable folder to store our application icons. To create our icons as drawables we followed the tutorial mentioned in this user guide [2]. More precisely, we created our own icons, in Scalable Vector Graphics (SVG) format, and then converted them into eXtensible Markup Language (XML) files

[2] `https://developer.android.com/studio/write/vector-asset-studio.html` (last access on 02/04/2018)

using Vector Asset Studio, that is embedded in Android Studio. To create our icons in SVG we have chosen the Inkscape[3] tool because it allow us to create and edit SVG files. We decided to design our own icons, instead of using existing ones from websites like Flaticon[4] or Iconfinder[5], mainly due to the fact that it would be very difficult to find all icons we needed in our application, specially clothes related icons, with the same design. This way, by creating our own icons we guaranteed that all icons would be consistent.

Regarding layout sub folder of res folder, we stored all our XML files with our screens layouts, while in the mipmap sub folder, we saved our application launcher icon, in other words, the icon that represents our application and is seen during launcher screen. Besides the application launcher icon, we also used mipmap sub folder to save some icons used in our application, in particular icons related to patterns, due to the fact we could not convert those icons SVG files into XML, due to limitations of Vector Asset Studio. Additionally, in the mipmap sub folder, we also stored there images of clothes that were used during evaluation, in order to simulate our application state after user inserted some clothes in the virtual closet and created some capsules wardrobe.

Concerning menu sub folder, this is not a default characteristic of Android projects. This folder was created because, as mentioned before, the side menu logic and resources were automatically generated while creating our Android project. So, the menu resources is used to store menu content, namely to configure menu items.

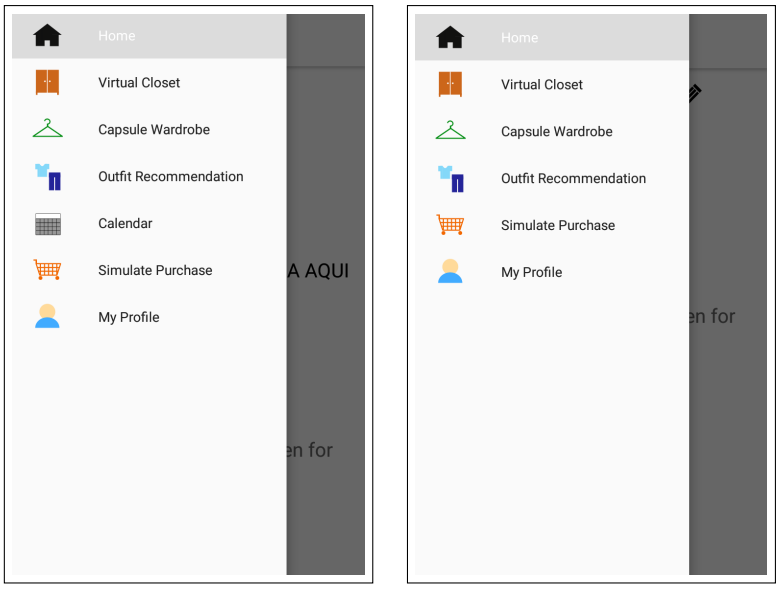
Finally, the values sub folder, of folder res, was also organised into several folders to store the XML files related to colours, dimensions, integers, strings and styles used in our application. The colour XML file, as the name says, stores all colours names used in our application, with corresponding red-blue-green (RGB) values associated to it. For dimensions and integers, with the intention to support Android devices with different dimensions and having in consideration user guide [6] from Android and tutorial [7], we created multiple dimensions and integers files based on smallest width. We did not implement custom layouts according to the smallest width. Instead, we created different dimensions and integers files, because we wanted to provide the same overall interface design independently on Android device size, but with different values to, for example, text size, specific field height or width, among other dimensions. Following this approach, in total we created four files for dimensions and four files for integers, more specifically we defined dimensions and integers files considering 320dp, 480dp, 600dp and 720dp as smallest widths. We considered these ones because were the ones mentioned in guide [6]. Similarly to dimensions and integers, we also created multiple strings XML files, more precisely two, with the purpose of supporting our application in Portuguese and English. We decided to create our solution with these two language possibilities since in low-fidelity non-functional prototypes evaluation, with Portuguese native users, we observed and users told us that some mistakes and difficulties were due to language barriers with English.

4.3.2 Project Initial Elements

After having our Android project structure created, we started by configuring our side menu content and implement a splash screen.

The side menu was the first thing that we implemented in our functional prototypes. We had decided to create an Android project with a Navigation Drawer Activity because this was the only way to had access to all functionalities and also because it determined how we would implement our solution screens

[3] `https://inkscape.org/en/` (last access on 02/04/2018)
[4] `https://www.flaticon.com` (last access on 02/04/2018)
[5] `https://www.iconfinder.com` (last access on 02/04/2018)
[6] `https://developer.android.com/training/multiscreen/screensizes.html` (last access on 09/04/2018)
[7] `https://medium.com/oceanize-geeks/supporting-different-screen-sizes-andorid-98d93493e50b` (last access on 09/04/2018)



(a) Clothes types overview screen in functional prototype.

(b) Clothes types overview screen in functional prototype.

Figure 4.8: Functional prototypes iterations of side menu design.

definitions. More precisely, since we had chosen to have a side menu, our screens had to be Fragments instead of Activities. The difference between these two objects, according to [20], is that Fragments are implementations of partial portions of an Activity. In our case, by selecting Navigation Drawer activity type, the Android Studio created a project with a class of that type that is our solution main activity. Our remaining screens are Fragments that implement part of screen layout and are replaced by others Fragments according to our solution state and actions performed by users. To customize our side menu just like we designed in section 4.2, we substitute the default menu items and corresponding icons created by Android Studio by our owns. Additionally, for each menu item, we associated one empty screen fragment with a title bar, where each corresponding functionality name appears. In a first stage, our menu design was as illustrated in sub figure 4.8a, with six items, home screen, our application functionalities and profile. However, during our functional prototypes, we have decided to prioritize implementing simulate purchase functionality over calendar, so we removed Calendar item from menu, resulting in sub figure 4.8b.

After customizing our side menu, we created a splash screen. We decided to implement a splash screen for two reasons. The first because we needed to have a step where we could verify our database user table and determine to which screen should we go based on if there is any user created. The second reason because, in a first run, we needed to initialize some tables like colours, patterns, seasons and others with default values. Similarly to menu, the design of our splash screen changed very little during our functional prototypes. We started by having the design with a place holder icon and our application name while we did not created our application final icon. Once we designed our application icon, based on our main functionalities, Virtual Closet, Capsule Wardrobe and Outfit Recommendation, respectively represented by closet, clothes hanger and t-shirt icon, our splash screen looked like sub figure 4.9.

After creating our side menu and splash screen, we implemented the home screen in our functional prototypes. We started by designing it as in our low-fidelity non-functionality prototypes, in other words, with one section for meteorology and another to outfit recommended or message saying that there is no outfit yet.

During virtual closet functionality development, we modified this layout, removing the meteorology

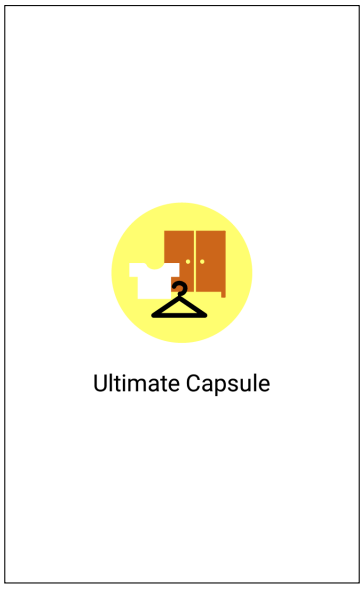
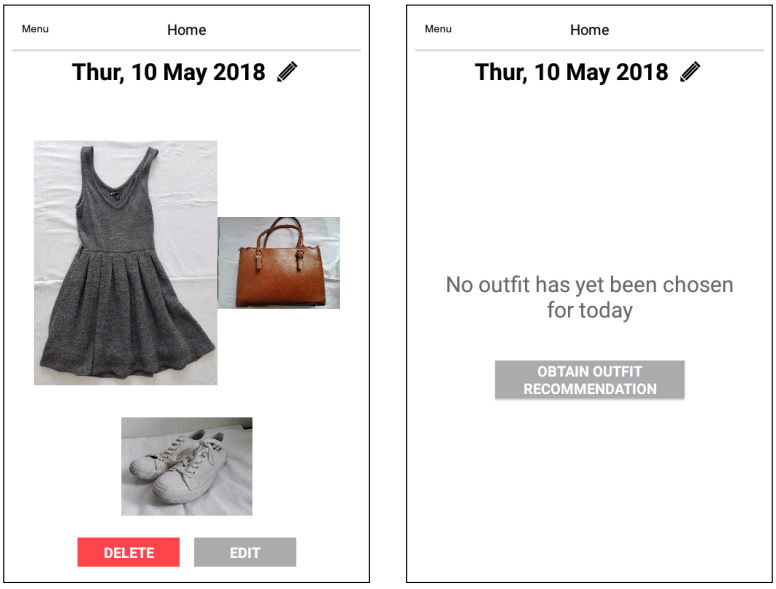


Figure 4.9: Splash screen with our application icon and name.

section, because we gave up of having meteorology as one of the many aspects considered while suggesting an outfit. We opted by removing this, due to the fact that we would have to change clothes, and consequently capsule characterization regarding appropriated time to wear. More precisely, we would have to substitute season by temperature range and if it was possible to wear while raining or not. Having this obstacles in mind, we decided to not consider meteorology while recommending outfits and, consequently, we did not found this information relevant to be in our home screen.

Additionally to this change, when we gave up of implementing calendar functionality, we decided to allow users to change date selected in home screen, by adding an calendar icon right next to today date. This way, we still allow users to see previously used and future outfits, even without calendar. At the end, our home screen design corresponded to sub figure 4.10a or 4.10b, depending if exists outfit for selected day or not, respectively.



(a) Home screen final design with chosen outfit.

(b) Home screen final design without chose outfit.

Figure 4.10: Home screen final design, respectively when exists outfit or not for selected day.

4.3.3 Profile

As mentioned in section 3.1.1 of chapter 3, we considered important to have a profile associated to each user, since it would allows us to provide a more personal experience. We considered this a required step, before using our application, since our solution depends on users personal informations, namely gender and location. We opted to start by implementing profile functionality in our functional prototypes, instead of Virtual Closet, Capsule Wardrobe or other functionality, after creating the splash screen and side menu, because we were aware of those dependencies in our application.

In this functionality, as mentioned in section 3.1.1, we defined that it would be four possible tasks: create, see, edit and delete profile. We decided to allow users to see, edit and delete their profiles in order to give them the possibility of modifying or deleting their profile.

To implement the process of creating a user profile, since this is the first screen of our application and for that reason there is no menu/navigation drawer yet, instead of implementing a fragment, we had to create an activity with a toolbar. We implement a toolbar in this layout with the intention to maintain consistency with remaining application interface.

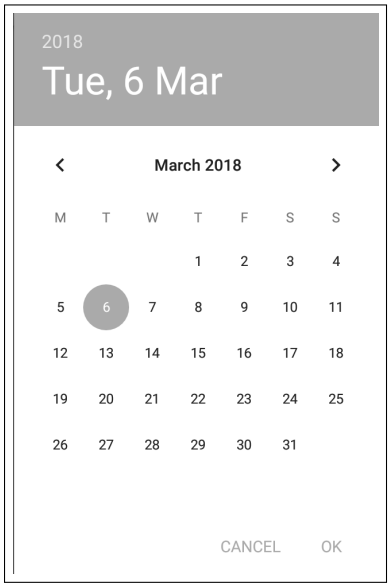
This was a functionality that we did not prototyped in our low-fidelity non-functional prototypes, since we did not considered it as main functionality of our application in 4.1. As a result, we had several iterations through implement it, in terms of interface design and functioning. During these iterations, all changes in interface and functioning were always simultaneously applied to create, see, edit and delete profile.

When we started implementing the profile functionality, we had chosen to only ask name, birth date and gender of user. The first two were not entirely required in our application, because they do not influence our solution, but we decided to have them since it could be helpful in the future. At that moment, the only information that would influence our application was gender, since as mentioned before it would define possible clothes categories in virtual closet.

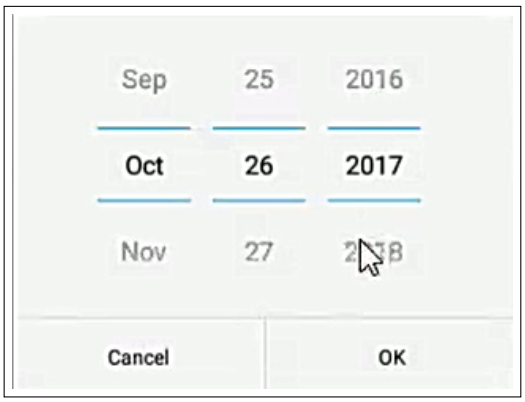
Figure 4.11: First profile edition design iteration, identically to first profile creation design iteration.

We started by implementing a simple version of it, showed in sub figure 4.11, where we would have title field followed at right by the corresponding field text box or radio buttons. More specifically, for both Name and Date of Birth we had input boxes and for Gender we decided to have two radio buttons, one for female and other for male. The main difference between the two input boxes for Name and

Date of Birth was its input type. For name we defined it as text and for Date of Birth we chose date. These specifications make difference because, in case of Name field, when we select the input text, the keyboard is shown and in terms of Date of Birth, when we select the input text, a Calendar is shown. Since the input type date is a default of Android, the view of calendar shown also is. In this stage, while testing our application with emulators we realised that there were two options to calendar view. If nothing was said the default was just like as sub figure 4.12a. Since we considered this was not an intuitive way to select a date corresponding to birth of date, we searched and found that we could obtain the view like 4.12b by setting up as spinner. Regarding the Gender field, we had chose radio buttons because, according to Fonseca et al. [9], this sends a message of only one option can be selected and that was the idea we intended to pass on to users.



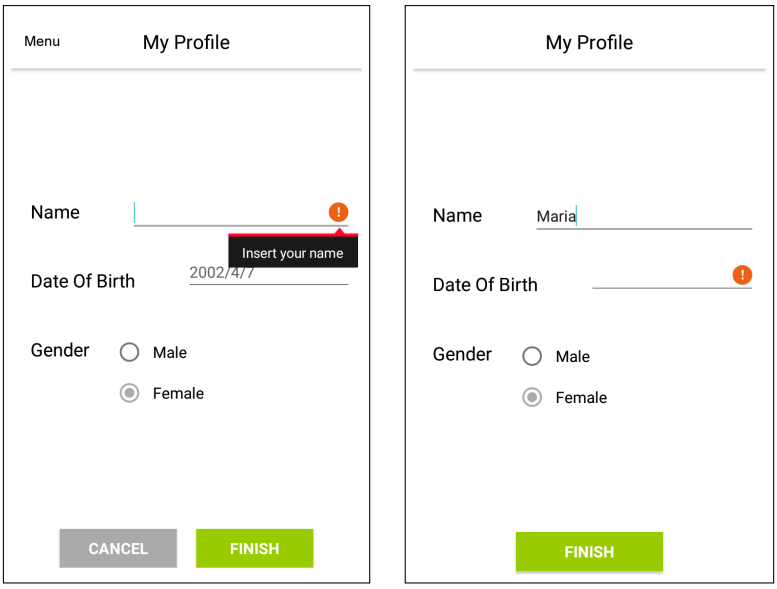
(a) Android default design in calendar view type.



(b) Android default design in spinner view type.

Figure 4.12: Android default calendar possible views.

Having our layout decided, we started to implement verifications while creating and editing our profile. More precisely, we check if the name was not empty or only spaces, if the date is not null and valid we do not let get through creating or editing profile until everything was right. We did not check gender value, since we would have one option already selected from the moment the application is open and radio buttons by default do not allow to unchecked an option. In the beginning we started by using the errors attached to each field, default by Android, as shown in sub figures 4.13a and 4.13b. As we can observed by these two sub figures, the errors function differently depending on input type. We can observe that for the Name field, in which input type is text, the error message is shown, while for the Date of Birth field, in which input type is date, that does not occurred. For this reason, we decided to implement our own error pop up dialogues. We started by implementing pop ups with Android default layouts, however, when we implemented warning dialogues about delete profile and cancel profile edition, with the intention to prevent unintentionally actions, we come to realise that those were not the best option, since it would not allow users to distinguish between warning and error dialogues. Having this in consideration, we implemented our own dialogues layouts. We decided to have three components in our dialogues pop ups layouts, title, message and buttons. As both names say, in the title, we define dialogue type as error or warning, and in the message layout, we show the error ou warning description message. The layout of last part of our dialogues, buttons, depend if we want the user to only comprehend the message, in this case we have only one button, or if we expect an answer from user. To emphasize the difference

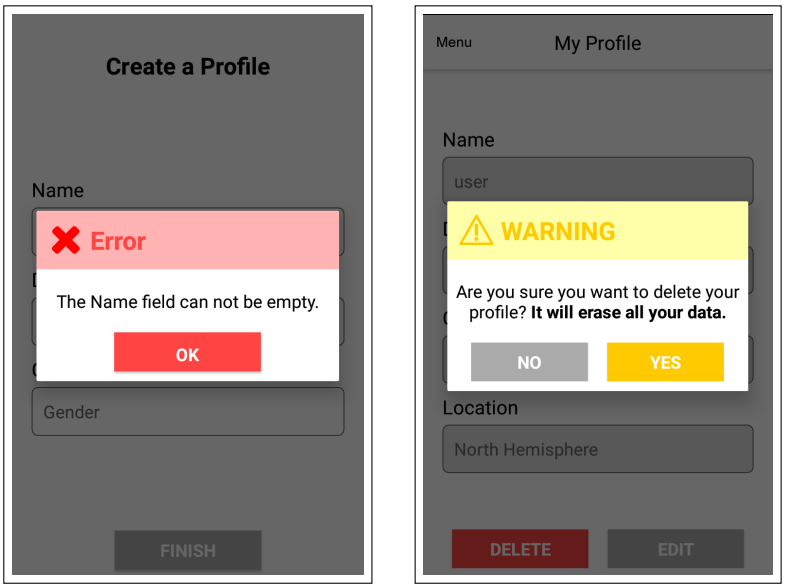


(a) Profile creation with error showing Name field is empty.

(b) Profile creation with error showing Date of Birth field is empty.

Figure 4.13: Functional prototypes first design of errors in profile creation.

between warning and error dialogues, in the title component, in addition to mention the type of dialogue, we added a representative icon created by us and defined the title background colour as light red or light yellow, respectively to error and warning dialogues. Aside from that, in our profile functionality, more specifically in error pop ups while creating and editing user profile, we set the only button background to colour red. In sub figures 4.14a and 4.14b we can observe, respectively, our error dialog pop up for Name field empty and our warning pop up for delete profile.



(a) Custom error pop up dialogue, showing Name field is empty.

(b) Custom warning pop up dialogue, alerting user about deleting profile.

Figure 4.14: Functional prototypes final design of errors and warnings in profile creation and edition.

We quickly abandoned this first profile layout arrangement, where we had title field followed at right by the corresponding field text box or radio buttons, and switched to the design represented in sub figure 4.15, since it would result in having limited space to write users name, specially on smaller screens.

Figure 4.15: Second profile edition design iteration, identically to second profile creation design iteration.

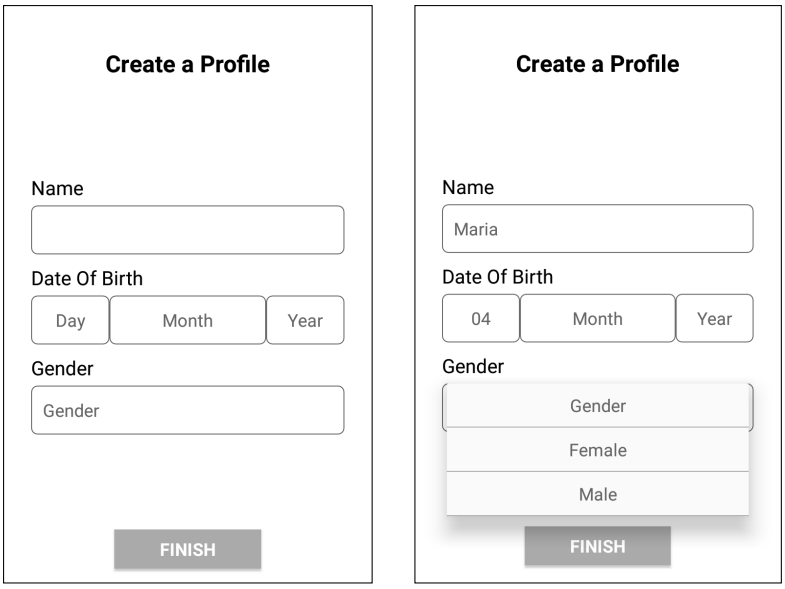
In this iteration, we decided to place each field input text or radio buttons below corresponding field titles, horizontally centred in our screen. With this change we resolved the problem of having insufficient space for the Name field.

At this moment, we thought that we had already finished implementing profile so we started to implement other functionalities. We soon realised that there was improvements and changes that we would need to perform with the intention to improve our design or due to necessitation of more information about our users.

One of the improvements based on other functionalities was surround input text with border. We did not had this idea until we realised the input text in date screen for outfit recommendation would be very discrete, possibly causing confusion from not seeing the input. So we created an file in drawable folder, composed by an white rectangle with dark grey rounded border, and applied to all input boxes that we had, more specifically the Name and Date of Birth fields. With this change we could better enhance each field input location.

At this development stage, we decided to test our application, namely profile functionality, in several physical Android devices. While doing this, we stated the calendar view to select date of birth did not function how we expected. We assumed that for calendar, in every single Android device, a view similar to sub figure 4.12b will be shown because we had added the spinner definition, however in some devices we stated that did not happened. As creating an intuitive, easy to learn and use application was our main goal, we decided to replace Android default date input and implement date of birth as three text fields that when pressed, instead of opening the keyboard, open a drop down list with days/months/years, depending if user pressed the first, second or third text field. In terms of drop down lists, the day list contains number from 1 to 31, months list has each month abbreviation name and years list contains 1900 to current year, ordered by reverse order. This design feature, drop down list with possible options, was also applied to the Gender field with the intention to keep consistency in our overall layout aspect. In sub figure 4.16a, we can observe new design after surround Name, Date of Birth and Gender fields input with a border, and in sub figure 4.16b, we can observe how drop down list is shown with Female and Male gender options.

Another change that was necessary to make, while implementing other functionalities, namely virtual closet, was the inability to change gender while editing user profile. If we would allow users to change



(a) Third profile creation design iteration.

(b) Gender edit in third profile creation design iteration.

Figure 4.16: Profile creation third design iteration.

gender, specially after adding some clothes into virtual closet, it would cause inconsistency in database. This would happen since clothes categories shown are based on user gender and consequently some previously added clothes would not have a valid category according to the updated gender and because of that, those would be ignored when retrieving clothes from database and would not be considered as pieces of closet while creating capsules, recommending outfits or simulating new items acquisitions. To distinguish between fields that could be edit, more precisely Name and Date of Birth fields, and Gender that could not be edit, we opted by giving a grey colour to background of Gender input box.

Additionally, while implementing the process of creating a new capsule, we realised that there were two options in terms of define new capsule start and end dates. We could let users define each capsule start and end dates, allowing to have capsules with less or more than three months and with same season but different dates, or define ourselves the specific start and end dates of seasons. As mentioned in section 3.1.1, from these two we chose the last option, because, as defined in chapter 1, our main goal from start was to allow create capsules dedicated to each season, in other words, with a three months duration. With a little research [8] [9] [10], we come to realised that, from all possible ways to define each season start and end date, there were two that are very commonly used in real world. These two are:

- meteorology method, where seasons start and end days always correspond, respectively, to the first day and end day of specific and clearly defined months
- astronomic method, where seasons are determined by equinoxes and solstices days

We decided to chose meteorology method, over astronomic, since the former only required to know if users live in north or south hemisphere, while the latter would required to know both the country where user live and equinoxes and solstices dates of current year to the country selected, because they vary from year to year, depending on location. Having in mind that we need to know users location, more precisely if they live in north or south hemisphere we added a Location field to profile. We started by

[8] `https://www.timeanddate.com/calendar/aboutseasons.html` (last access on 16/02/2018)
[9] `http://www.bbc.com/news/uk-northern-ireland-29014886` (last access on 16/02/2018)
[10] `https://www.livescience.com/25202-seasons.html` (last access on 16/02/2018)

implementing the interface of this field as a drop down list of countries but we realised that this would lead to a long exhaustive list of possible countries, making this task very inefficient, time wasting and frustrating to users, even if they were alphabetically ordered. We also stated this would be more complex to have, since it would obligate us to implement correspondence between countries and hemispheres. So we opted by a more simple approach to us and users. Instead of presenting a countries list, we decided to ask directly to users in which hemisphere they are. As final result, our profile design looked like figure 4.17.

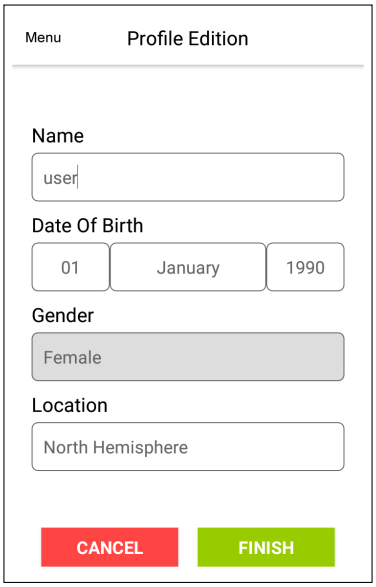


Figure 4.17: Fourth profile edition design iteration, identically to third profile edition design iteration with Gender input box with grey background and additional field, Location.

From our formative evaluation phase, observations and users feedbacks, we could conclude that there were two issues in our functional prototypes regarding this functionality. In terms of profile, users though the year drop down list of birth date was too big and this last field, Location, could be a little tricky to some users, causing some doubts and errors. Regarding birth year we considered this a minor problem, since this will only be a one time task, so we decided to not perform any change. Concerning location, we considered this an important and relevant obstacle since could cause actually errors in the rest of our application, for example if location is wrongly selected, seasons dates in capsules will also be wrongly calculated. So, with the intention to help users to correctly select location we added a help icon. When pressed, this icon opens an information dialogue, that let users know the reason why we need this kind of information and the current seasons for each location options. This way, users do not have to rely only on their geographic knowledges, they are capable of selecting the correct one based on the current season on their location. In figure 4.18, we can visualize the addition of the help button above the Location field input box.

4.3.4 Virtual Closet

The following steps in our functional prototypes development, after implementing our user profile, was adding virtual closet organization and new clothing piece process into it.

Regarding virtual closet organization overall layout, we maintained the interface design sketched in non-functional low-fidelity prototypes with the exception of small details.

The main changes between non-functional low-fidelity and functional prototypes corresponded to:

- Change location of add new piece button;

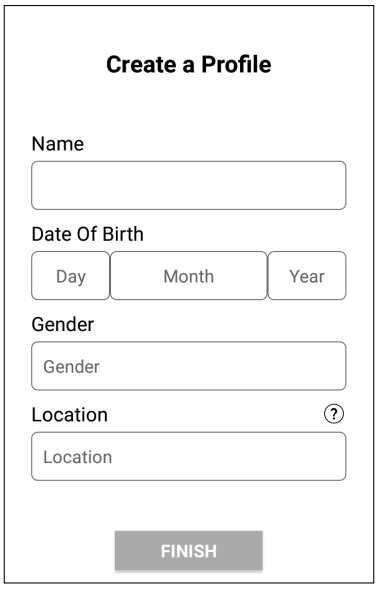


Figure 4.18: Fifth profile creation design iteration, identically to fourth profile edition design iteration with help button above the Location field input box.

- Remove “Tops” word from top clothes categories;
- Change columns number while showing users clothes;
- Add an information dialogue while updating changed information about one clothing item.

Regarding the add new piece button, while implementing our functional prototypes, we become aware that the new piece button, characterized by the plus icon, was not in the best place possible. We concluded that being close to clothes types/categories could lead to accidentally press it. With the intention to prevent those accidents, we placed the add new piece button in the toolbar, at right of virtual closet functionality title. This change can be seen in figure 4.19.

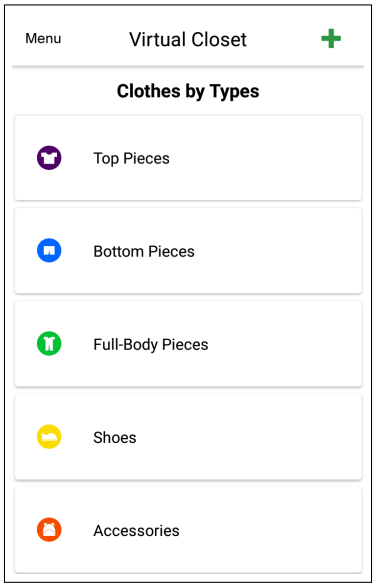


Figure 4.19: Virtual closet types overview with add new piece button in the toolbar.

Having in consideration limited width, specially in smaller screens, from non-functional low-fidelity prototypes to functional prototypes, in our solution English version, we removed the word “Tops” from

top clothes categories in two different scenarios. The first while showing all top possible categories and the second while presenting all users clothes from one top category.

In terms of columns number while presenting users clothes, at the beginning we followed our non-functional low-fidelity prototypes sketches and had two columns, independently of device dimensions. However, when we tested on physical devices, we recognized that it was possible and beneficial to have one more column in larger screens, namely 7"and 10"tablets, since it would allow users to see more items in the same space. So, for these bigger devices, we defined three columns instead of only two.

The last change in terms of virtual closet organization was related to giving users feedback about our application status. With this purpose in mind, we added an asynchronous task that is executed whenever a clothing item is edited. This asynchronous task allows us to show a dialogue pop up, saying "Updating Clothing Item...." while performing the necessary changes in database.

As mentioned in section 4.3.1, we implemented our own icons. The main reason to do that was precisely due to clothes types and categories. We did some research in popular icons websites, as mentioned in subsection 4.3.1, however none of those had all the icons we needed. With the intention to better distinguish the various clothes types possible, we decided to designate a different colour to each clothing piece type. For clothes categories, we followed the opposite thinking. Regarding categories, we remained to the same colour of each corresponding clothing piece type, changing the luminosity between categories. We decided to have same hue and different luminosity for distinct categories icons of same type, because it would help users to understand make connection between types and categories. Since we implemented this same hue but different luminosity rule, we reordered the categories, in order to have a gradual increment of luminosity from top to bottom.

Similarly to virtual closet organization, the process of adding a new clothing piece into the virtual closet also preserved the overall design with some changes, namely:

- Change location of previous button and add cancel button;
- Highlight selected options at current moment and in previous steps;
- Divide step of selecting clothing piece type and category into two steps;
- Abandon automatic colour extraction and creation of a set of basic clothes images;
- Change possible clothing piece colours and patterns;
- Add pattern scale step;
- Redesign occasions screen;
- Add subtitle to help users distinguish between one or multiple value selection steps;
- Add error and warning dialogue pop ups;
- Add border to fields in confirmation screen;
- Add an information dialogue while adding new clothing item into database.

With the intention of allowing users to cancel the process of adding a new item, we substituted the previous button at the bottom of all screens with a cancel button. The new previous button passed to the top left corner and changed to a left arrow image button. With these changes, we were able keep the ability to move to previous state, while adding the capability to cancel/drop out at any point of the process. This new design can be seen in figure 4.20.

Regarding highlight selected options, this was something that we already hat though during our non-functional low-fidelity prototypes, however we were not able to design it in them. In our functional

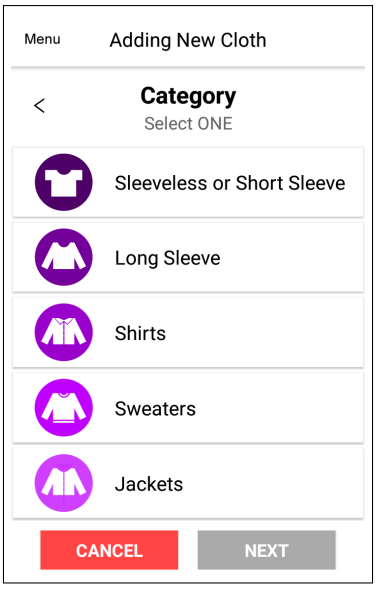


Figure 4.20: Select new clothing piece category, with both previous and cancel buttons.

prototypes we added this feature. The intention with this was to allow users see the not only selected options at the current step, but also in the previous steps. To be able to show previously collect data, and also have knowledge for the new clothing item step at the final phase of adding a new clothing piece, we decided to save the information in an object of type `ClothingItem` and passed it on through all steps. We opted by a `ClothingItem` object, and not strings or other data structures available, since using this object would allow us to more easily modify clothes information or new clothing piece process steps, if necessary.

Concerning clothing piece type and category selection step, we decided to split in two distinct steps for two reasons:

- We realised, in our non-functional low-fidelity prototypes evaluation, users were a little bit confused about difference between clothing piece type and category;
- To increase consistency in our solution interface, from virtual closet organization to process of adding a new piece.

In respect to prioritize of implemented features, we opted for abandon the idea of having a set of basic clothes from which users could chose as well as automatic colour extraction, for being able to implementing simulate purchase functionality in our solution. This decision was made based on our main work goal, to develop an application that allowed to create capsules with the minimum number of clothes that maximize the number of possible outfits. Having this target in consideration, we believed that the simulate purchase was a functionality that would go meet better our goal, if comparing to automatic colour extraction or the creation of a basic clothes set. So, regarding our functional prototypes, instead of three possible clothing piece images sources, we had two, take picture at moment or select one from gallery, and in terms of colours, instead of asking users to confirm extracted colours, we asked them to indicate us which colours the new clothing piece has. For asking new clothing piece colours, we decided to have a grid with all possible colours, showing a portion of colour and its name, as showing in figure 4.21.

The colours and patterns considered were also different from the ones originally though in non-functional low-fidelity prototypes, based on research made about clothes coordination. In terms of colours, in our functional prototypes we considered the ones from 12 colour wheel and also neutrals

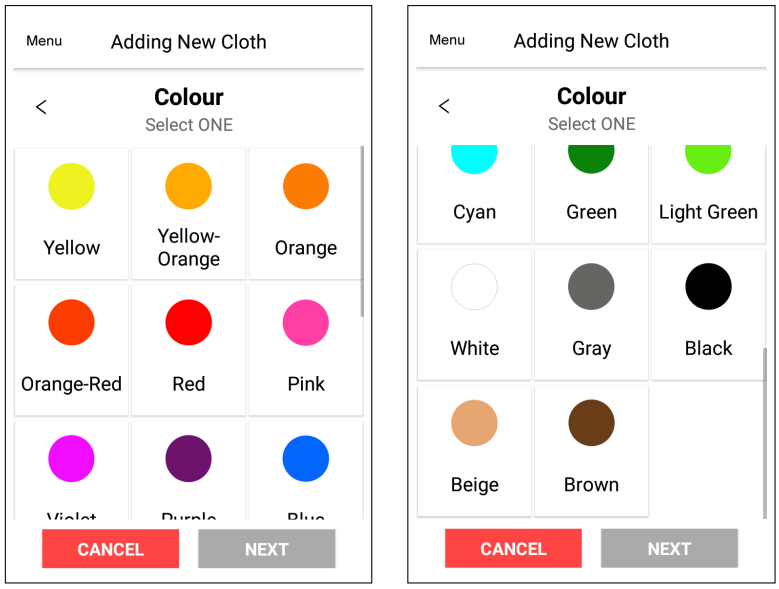


Figure 4.21: Select new clothing piece colour.

colours, like white, black, grey, brown and beige, while for patterns we contemplate plain/no patter, striped, checkered, plain, dotted, floral, animal and, to keep our application open to less used patterns, other. Additionally to change colours and patters, from our research on how to combine clothes, we also concluded that we needed to know pattern scale, so we added a step, right after define clothing piece pattern, to specify this new characteristic. This new step is skipped if pattern selected if plain/no pattern.

Concerning our overall design in adding new clothing item, we decided to reformulated occasions layout design, with the intention to make the interface more cohesive and consistent. Similar to other steps, we replaced the check boxes by a list of possible occasions, from where users can select one or more. The result of this change resulted in screen shown in figure 4.22.

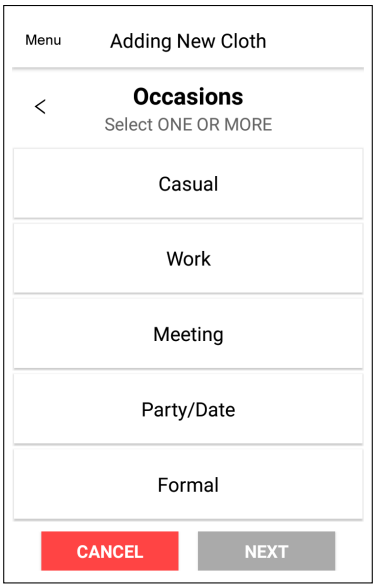


Figure 4.22: Select new clothing piece occasions new design.

For all steps of adding new piece, with intention to explicitly say to users how many options can be selected, we added a subtitle. Depending on multiplicity of each attribute defined in section 3.1.2 of chapter 3, this can have one of the three following possible values: one, one or more options or two or

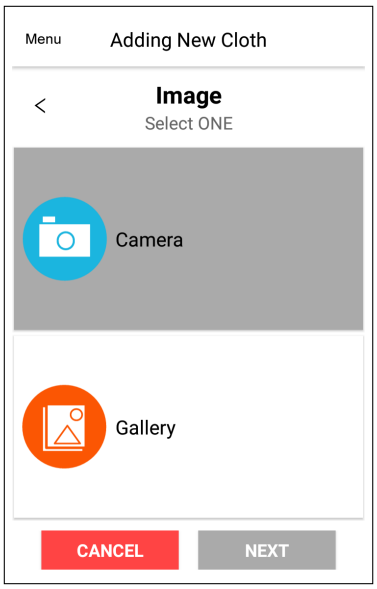
more options.

Similarly to profile functionality, we added our own dialogue error and warning pop ups. In respect of error pop ups, we added them in every step, in the next button, with the intention to only allow users to move on to next step if everything is correct. Regarding warning pop ups, similarly to warning dialogues in profile functionality, we added them in the cancel button, with the purpose to avoid accident actions, in this case to avoid cancel the process of adding a new clothing item by accident.

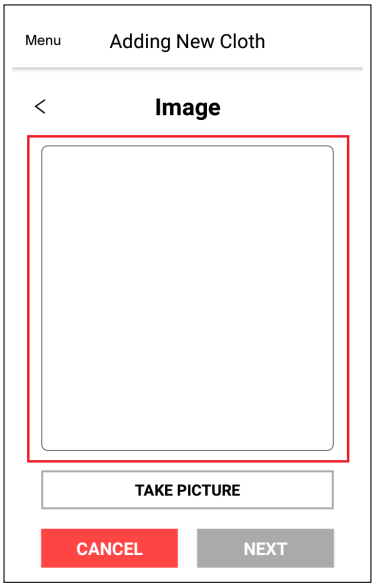
Also identical to profile, in our confirmation layout we added a border to surround each clothing piece attribute value, to better distinguish highlight each field and limit value area of each characteristic.

Analogously to editing clothing item, with the intention to show users our application state, we implemented an asynchronous task that is executed when a new clothing item is added into database. When this asynchronous task is executed, it creates a dialogue pop up, saying “Adding Clothing Item....”, that is shown during adding a new clothing piece to database.

In our functional prototypes formative evaluation, regarding this functionality, from second task, *add a new clothing item to virtual closet*, we concluded that users had shown confusion with take picture of new clothing item. The problem was that users though once they selected “Camera” option from screen shown in figure 4.23a and continued for next screen, the camera would automatically be open and/or the area highlighted by the red rectangle in figure 4.23b was the camera preview and that was not already working. In that moment, what we had implemented was that users should pressed “Take Picture” button in figure 4.23b to open the camera. With users thoughts and comments during and after performing this task, we came to the realization this was not the most intuitive and would actually cause problems. In order to correct this, we chose to change the flow in these steps. We decided that by pressing “Next” button in figure 4.23a, the camera or galley would automatically open, before showing screen corresponding to figure 4.23b. Additionally to this issue, there was one user that comment it was not intuitive in which screens it was possible to select only one or multiple options, even though we had subtitles saying how many could be selected. In his opinion, the application interface was very consistent, however there was this detail that cause him some apprehension when, adding a new clothing piece to virtual closet, users had to select clothing piece appropriated seasons. Regarding this interface obstacle, with the intention to maintain consistency, we opted by leaving as it was.



(a) Select New Clothing Piece Image Source with Camera Option Selected.



(b) Take Picture of New Clothing Item Screen.

Figure 4.23: Select new clothing piece take picture problematic screens.

4.3.5 Simulate Purchase

After implementing virtual closet functionality, we opted by adding simulate purchase. We chose simulate purchase because overall interface and functioning is very identical to the process of adding a new clothing item into virtual closet. In fact, we started this functionality implementation in our functional prototypes by creating distinct XML layout files, nevertheless we realised this was not necessary, the differences, in terms of design, between adding a new clothing item into virtual closet or simulating were very small or even non-existent for some steps. So, the design of simulate a new clothing purchase from non-functional low-fidelity to functional prototypes, suffer the same changes as the process of adding a new piece into virtual closet. More specifically,

- Change location of previous button and add cancel button;
- Highlight selected options at current moment and in previous steps;
- Divide step of selecting clothing piece type and category into two steps;
- Abandon automatic colour extraction and creation of a set of basic clothes images;
- Change possible clothing piece colours and patterns;
- Add pattern scale step;
- Redesign occasions screen;
- Add subtitle to help users distinguish between one or multiple value selection steps;
- Add error and warning dialogue pop ups;
- Add border to fields in confirmation screen;
- Add an information dialogue while calculating possible outfits with specified clothing item.

Additionally, we added a screen, as first step before start simulate purchase process, where we explain this functionality, with intention to jump to it if users decide to cancel the process. This screen is shown in figure 4.24.

Similarly to virtual closet functionality, through each step, to be able to show previously collected data and also save knowledge about clothing item to simulate, we opted to saved options selected in each step in an object of type `ClothingItem` and passed it on through all steps. For the same as in Virtual Closet functionality, we opted by an object, instead of strings or other data structures available, since this way would be more easy to modify clothes necessary informations or add new simulation steps, if necessary.

After our functional prototypes formative evaluation, regarding this functionality, we concluded that, just like in virtual closet functionality, users had shown confusion with take picture of clothing item to simulate. The problem resolution applied in the virtual closet was the same implemented to this functionality, instead of only open camera or gallery when “Take Picture” button is pressed, now this occurs right after selecting image source and transitioning to next step. Additionally to this issue, we also concluded the explanation in first screen was too long and users did not read, or when they did try, they gave up halfway through it. We opted by reducing the text, and not remove this screen by complete, because otherwise, when users cancelled the process, they would be redirected to first step of the process, and that did not seemed intuitive and correct to us.

After functional prototypes formative evaluation and during development of simulate purchase algorithm, we decided to omit the step of selection simulation location. We opted to remove this step

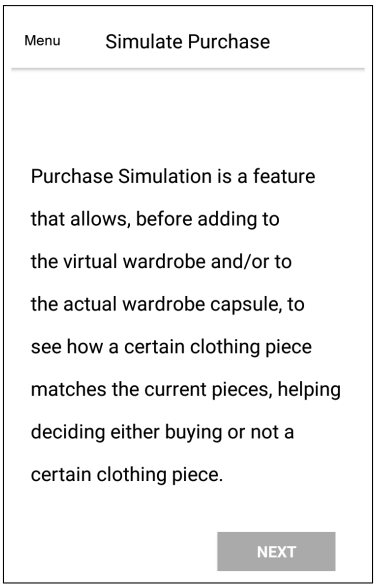


Figure 4.24: Explanatory screen where we describe simulate purchase functionality and its main goal.

because, by simulating with clothes from virtual closet, we guaranteed that the outfits shown can be incorporated, not only in some specific capsule, but also in future capsules created. This modification manifested in, instead of transition to simulation location step from confirmation screen, to directly execute asynchronous task that shows dialogue while calculating possible outfits with specified piece. Additionally, since we observed the number of possible were very high and would take very long to load them into our application, we decided to add, before outfits grid view, the number of possible outfits and limit the outfits shown. In devices with screens up to 7"we limit to ten outfits, while in devices with bigger screens than that we limit to fifteen outfits shown.

4.3.6 Capsule Wardrobe

After implementing virtual closet and simulate purchase, we added capsule wardrobe functionality to our functional prototypes, in terms of managing existent capsules wardrobe and creating new ones.

Regarding managing existent capsules wardrobe, we started to implement design from our non-functional low-fidelity prototypes, one list of existent capsules wardrobe. Similarly to virtual closet and simulate purchase functionalities, the new capsule icon was not in the best place possible, so we moved it to toolbar, at right of capsule wardrobe functionality title, in order to prevent accidentally pressing it. In terms of each capsule element list, we implement a layout similar to the one non-functional low-fidelity prototypes, with season icon, capsule name and number of clothing pieces.

Figure 4.25: Summer capsule layout example with information button in the right upper corner.

From our storyboards and low-fidelity non-functionality prototypes, regarding seeing and managing existent capsules we did one major modification. This corresponded to add an information icon to each capsule element list, with the intention to allow users see and modify capsules information. This way, if users press capsule element list, they can see capsule clothes, but if they press information icon, they

see, an can change if they want to, capsule name, season and total of clothing items. In figure 4.25, we can see a summer capsule layout example with this new button in the right upper corner.

In terms of create a new capsule wardrobe we did some changes, comparing to our low-fidelity non-functionality prototypes. More specifically, based on previously implemented functionalities, we:

- Change location of previous button and add cancel button, just like virtual closet and simulate purchase functionalities;
- Highlight selected season, similarly to virtual closet and simulate purchase functionalities;
- Save collect data from each step into an object, in this case `CapsuleWardrobe`;
- Add subtitle in select season screen to enhance that only one season can be chosen;
- Add error and warning dialogue pop ups, identically to profile, virtual closet and simulate purchase functionalities;
- Add confirmation screen, similar to confirm clothing item in virtual closet and simulate purchase;
- Add an information dialogue while calculating clothes that maximize number of outfits possible.

Besides these changes based on previously implemented functionalities, we applied the same design of select birth date in profile, to the action of selection maximum number of clothing items of new capsule.

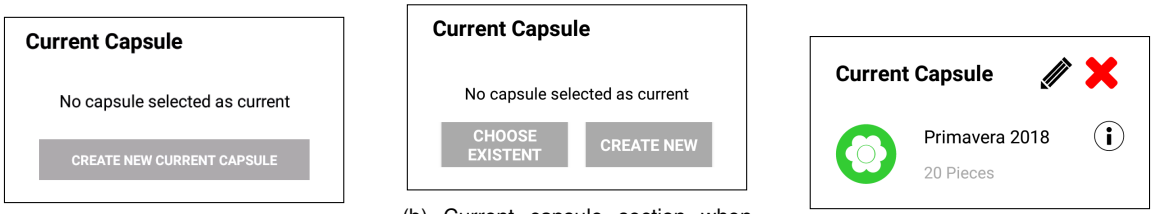
Additionally, regarding to create a new capsule, from our storyboards and low-fidelity non-functionality prototypes we changed the flow when creating a new capsule wardrobe. Previously to functional prototypes, we defined that we would allow users explore capsule clothes and possible outfits, before adding new capsule to database, with the intention to edit capsule clothes and outfits. However, in functional prototypes, we decided to not permit to edit capsules clothes, neither to show capsule possible outfits. We opted by, in our functional prototypes, after confirming new capsule information, in confirmation screen, return to the screen where we can see and manage all capsules.

As mentioned in section 4.3.3, through developing capsule creation process, we realised that had two options to define capsule start and end dates. We could let users define each capsule start and end dates, allowing to have capsules with less or more than three months and with same season but different dates, or define ourselves the specific start and end date of seasons. We opted by last option, since we wanted to allow capsules with three months duration, dedicated to a season. As mentioned, with a little research [11] [12] [13], we come to realised that, from all possible ways to define each season start and end dates, there were two that are more commonly used in real world. These two are meteorology method and astronomic method. As said in section 4.3.3, we chose the meteorology method, over astronomic, since the former only required to know if users live in north or south hemisphere, while the latter would required to know both the country where user live and equinoxes and solstices dates for current year and for country selected, since they vary from year to year, depending on location. These two dates, capsule start and end dates, are shown in confirmation screen, without allowing users to edit it.

Having both features implemented, managing existent capsules wardrobe and creating new ones, we though that we had finish implementing this functionality, however, when we started to implement outfit recommendation, we stated that by allowing to have multiple capsules for the same season, we created one problem in our application. We did not know from which capsule we should find appropriated outfits. With the purpose of resolving this problem, we added to our solution the possibility to select, from existent capsules for present season, the capsule that should be considered as current. We also

[11] `https://www.timeanddate.com/calendar/aboutseasons.html` (last access on 16/02/2018)
[12] `http://www.bbc.com/news/uk-northern-ireland-29014886` (last access on 16/02/2018)
[13] `https://www.livescience.com/25202-seasons.html` (last access on 16/02/2018)

allowed users to create a new current capsule. With the intention to implement current capsule, we added a current capsule wardrobe section, above capsule list. This current capsule section can vary, depending on existent capsules. The different possible designs of current capsule wardrobe section are presented in sub figures 4.26a to 4.26c. In a first scenario if there is no current capsule and there is no capsule at all, or all existent capsules are not for present season, in current capsule section, we only allow users to create new current capsules, we do not allow to chose from existent because there are any. We decided to have a "Add Current Capsule", besides the normal add new capsule button in the toolbar, to not making users guess which season is considered by our application as present. The main difference between these two buttons is that the button to add new current capsule skips the step of selecting season, since it automatically sets it to present season. This first scenario is presented in sub figure 4.26a. In a second scenario, if there is no current capsule, but exist capsules for present season, we allow users to chose from existent or to create a new current capsule, respectively through buttons to choose from existent and add current capsule. The latter button, "Add Current Capsule", has the same logic as "Add Current Capsule" button from first scenario, while the former button, "Choose from Existent", redirects users to a screen where are presented all capsules for present season as well as a none option. This none option was added with the intention to allow users to unselected current capsule as current. In sub figure 4.26b, we represent this second scenario. Finally, in a third scenario, if there is a current capsule selected, we show current capsule in current capsule section, with same layout as remaining capsules, and we show, in capsules list, all capsules previously created by users, except current capsule. This is presented in sub figure 4.26c.



(a) Current capsule section when there is no current capsule selected neither capsules to present season.

(b) Current capsule section when there is no current capsule selected, but exists capsules to present season.

(c) Current capsule section with a current capsule selected.

Figure 4.26: Current capsule section possible scenarios.

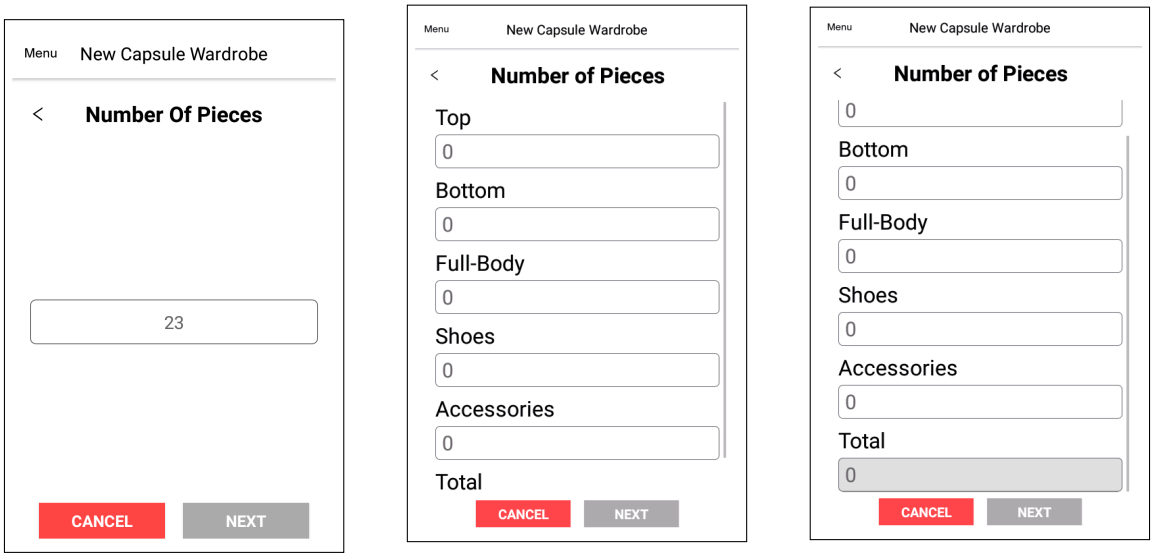
In our functional prototypes formative evaluation, concerning this functionality, from fourth and fifth task, respectively *create a new capsule wardrobe* and *delete an existent one*, we concluded that there were some obstacles. More precisely,

- The majority of users pressed "Create New" button without realizing that this would add a new current capsule and not simply a new one like the plus icon at toolbar;
- Users were confused about start and end dates on confirmation screen, they looked random numbers and not capsule start and end dates;
- One user mentioned the list of capsules should be ordered by most recently created;
- Users did not associate information icon with seeing, editing and delete a capsule.

With the intention to make our application interface more intuitive we corrected the pointed obstacles. Namely, concerning the first problem, difference between "Create New" button and plus icon at toolbar, we enhance them by renaming "Create New" button to "Create New Current Capsule". In terms of start and end dates on confirmation screen, we substitute months numbers by their abbreviation names and spaced a little these two dates. Regarding capsule list, we implemented the suggestion of showing

capsules ordered by most recently created. Relatively to information icon, we substituted it with two icons, an edit button that, when pressed, transition to edit capsule screen, and a delete button that, when pressed, opens a warning pop up, to help users not deleting capsules by accident.

After our functional prototypes formative evaluation, and while implementing our capsule creation algorithm, we encountered a problem. How we divide the number of total clothing items into the several clothes types. In respect to this, we had two possible scenarios, ask users the total number of clothes and define ourselves how this total is divided into the several types, in other words which are tops, bottoms, full-body pieces, shoes and accessories, or ask users how many clothing pieces they want of each type. We opted for the last scenario, since from research made we observed there are no consensus about which clothes actually count to total number of clothes in capsules neither about how to divide this total for the several clothes types. Having this in mind, we decided to allow users to customize the number of capsule clothes in terms of tops, bottoms, fullbody pieces, shoes and accessories, ranging from zero to number of existing clothes for each type. So, we replaced the input for defining the total number of clothes in new capsule by six inputs, one for each clothing piece type and one final for total number of clothes. We chose to have this last one, total number of clothes, not to allow users to define total number of clothes, but with the intention to show users how many clothes in total the new capsule will have. In sub figure 4.27a we can see how before we would let users chose the number of clothes in the new capsule wardrobe, only through total. In sub figures 4.27b and 4.27c, we present the new design, where users must select the number of clothes by type, and then the last input is a non edit input box, with grey background that shows the total of pieces of the new capsule to be created.



(a) Current capsule section when there is no current capsule selected neither capsules to present season.

(b) Number of clothes of new capsule divided into pieces types, more precisely Tops, Bottoms, Fullbody, Shoes and Accessories.

(c) Number of clothes of new capsule divided into pieces types, more precisely Bottoms, Fullbody, Shoes, Accessories and Total.

Figure 4.27: Current capsule section possible scenarios.

4.3.7 Outfit Recommendation

The final functionality to add in our functional prototypes was outfit recommendation, more specifically the process of allowing users to ask for an outfit suggestion according to attributes selected or textual descriptions in natural language.

We started to implement the design from our non-functional low-fidelity prototypes with some modifications similarly to the ones applied to previously implemented functionalities. More specifically,

- Added a screen, as first step before start outfit recommendation process, where we explain this functionality;
- Change location of previous button and add cancel button,;
- Highlight selected options;
- Save collect data from each step into an object called `OutfitRecommendation`;
- Add subtitle in each step to enhance how many options can or must be selected before continuing;
- Add error and warning dialogue pop ups;
- Change dominant colours based on colours considered in virtual closet and simulate purchase functionalities;
- Add confirmation screen;
- Add an information dialogue while calculating appropriated outfit.

Additionally to these changes, regarding our attribute selection method, in our low-fidelity non-functional prototypes, we received feedback that would be also interesting to allow users to also personalize which pieces should constitute, besides specific clothing item personalization. In other words, let users determine if they want an outfit with top and bottom pieces without accessories, or an outfit with full-body piece and accessories. Considering this and the fact that for some users, in our low-fidelity non-functional prototypes, was not very clear which steps were optional or required, we decided to join all optional attributes into a single screen, from which users can skip if they do not want to personalize outfit, or select one or more to customize it. In figure 4.28 is possible to see this customization screen with all possible options of customizations.

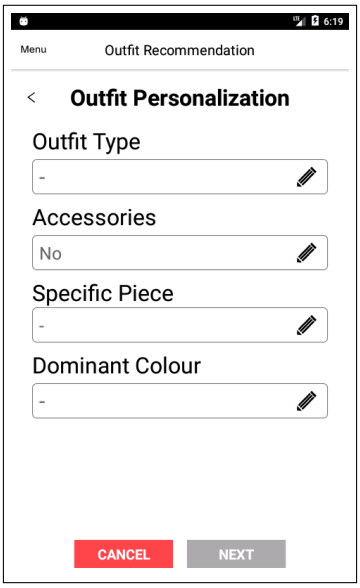


Figure 4.28: Outfit recommendation personalization screen that allow users to customize type of outfit, select specific clothing item, dominant colour and/or if must have accessories or not.

As mentioned in section 4.3.6, while implementing this functionality we encountered a problem related to capsule wardrobe. We defined that our goal was to allow create several capsules for the same season, however this mean that while recommending an outfit for a specific date we need to know from which capsule we should retrieve clothes. For solving this problem, we added the feature of choosing,

from possible capsules with present season, the current one. We though that by doing this we had all scenarios covered out regarding capsules, namely when there is only one capsule for the present seasons, as well as when there are several capsules for the current season. However, we realised that we did not defined what our application should do when there is no current capsule selected. In this case, we decided that our solution should retrieve clothes from virtual closet. To implement this, we could only had done the necessary logic, but we wanted to let users know that we would use clothes from virtual closet, due to the fact there is no capsule selected as current. To inform users about this, we decided to add an information dialogue that pop ups when this occur, while transitioning from selecting outfit date.

In our functional prototypes formative evaluation, concerning this functionality, from sixth task, *ask for an outfit recommendation*, we concluded that there were some obstacles. More precisely, we observed that users correctly selected all outfit attributes until the outfit personalization step. In this step, users hesitate concerning which option they should select in outift pieces and were confused with difference between outfit pieces and outfit specific piece fields, and what actually meant this last personalization option. Concerning outfit pieces options, since users ended up to select the correct one, we opted by not modifying. Relatively to distinguish better between outfit pieces and outfit specific piece personalization options, we decided to rename both for, respectively "Outfit Pieces Type" and "Outfit Specific Item". Additionally to these two obstacles, similarly to simulate purchase functionality, we observed the explanation in first screen was too long and users did not read, or when they did try, they gave up halfway through it. We opted by reducing the text, and not remove this screen by complete, because otherwise, when users cancelled the process, they would be redirected to first step of the process, and that did not seemed intuitive and correct to us.

After our functional prototypes formative evaluation, and while implementing our natural language algorithm to process textual descriptions given by users, we decided to limit the possibility to chose between attributes selection and textual description method only when application language is English. We did this because our natural language algorithm was only implemented to English. So, in Portuguese version, after selecting outfit users automatically transition to select occasion.

4.3.8 Clothes Coordination Algorithm

As mentioned in section 3.2.1, in algorithms `CapsuleCreation`, `OutfitRecommendation` and `SimulatePurchase`, we needed to know if two clothing items combine or not, considering their colours and patterns types and scales. With the intention to avoid duplicate code through these three algorithms, we created `ClothesCoordination` algorithm, which abstracts rules mentioned in section 2.1, normally used to combine clothes in terms of colours and patterns.

Regarding colours, we created two arrays with colours names, called neutrals and nonNeutrals, to respectively storing colours defined by us as neutrals and colours from colour wheel. Concerning colours from colour wheel, we implemented functions to calculate remaining colours for analogous, oppositve, triadic and split oppositve colours schemes, having in consideration each scheme definition in section 2.1. In all these functions, we received an colour name and returned one colour for opposite scheme or a list of colours for the remaining ones. Additionally, we implemented two functions, both with two arguments but different types. The first function, called `coloursCoordination`, receives two colours names and returns true, if at least one of the following scenarios verify, and false if none occur:

- The two colours are equal, in other words, the same;
- If at least one colour is neutral;
- If the two colours correspond to opposite colours;

- If the two colours correspond to analogous colours;

The second function, called `clothesColoursCoordination`, receives two clothing items and iterates through both clothes colours and see if they all combine, calling `coloursCoordination` function. If one of them do not coordinate, in other words, if `coloursCoordination`coloursCoordination returns false in some moment of iteration, this function, `clothesColoursCoordination`, immediately returns false too.

Concerning patterns, according to knowledge in section 2.1, clothing items can have different patterns and still combine if one of them is plain or if they have contrasting scales. We implemented these two rules in one function called `clothesPatternCoordination`, that receives two clothing items and returns true of false, depending if one the mentioned before rules verify.

4.3.9 Capsule Creation Algorithm

As referred, to chose the clothing pieces that maximize the number of possible outfits, complying season and number of clothes by type, and create our capsules, we implemented a class called `CapsuleCreation`.

Our first step in our capsule creation algorithm is to obtain all clothes, by type, for specified season.

The second step corresponds to match clothes and chose the ones which maximizes the number of possible outfits. This is accomplish by iterating through two lists of clothing items and registering the number of possible outfits with each clothing piece from the first list, using the algorithm `ClothesCoordination` to see if two pieces combine or not. This information is stored in a map of integers keys, corresponding to clothing items ids from first list, and integers values, corresponding to number of possible combinations for clothing item with id saved in key. At the end, to decide which clothes coordinate best, we order the map by value, by reverse order, and chose the first n clothes, being n the selected number of clothes for that type by user. We defined that for coordinating clothes we would first calculate tops and bottoms, then shoes, full body pieces and finally accessories. We chose to start from tops and bottoms and then shoes, instead of full-body pieces first, since this way we could know which shoes go best with tops and bottoms. Concerning tops and bottoms, to decide which are the first ones to be selected, we compare both clothes number in capsule definition. The clothes type with less quantity is the ones to be first selected. For example, if number of bottoms defined in the capsule is smaller, the bottoms will be chosen first. In other words, we iterate through bottoms for season, saving and updating the number of possible outfits with each bottom piece. In the end, we order the map by value, by reverse order. Considered the map ordered, we choose the number of selected bottoms by user in capsule creation process and store it in capsule clothes list. Having the selected bottoms in account, we repeat this process to tops, but instead of using all bottoms retrieved from database to specified season, we use the selected bottoms. We order the tops map by value, by reverse order, and choose the number of selected tops by user in capsule creation process, saving them to capsule clothes list. These actions of combining pieces and saving number of possible outfits, order by reverse number of possible outfits and chose the first n pieces are repeated to all the remaining clothing types, more precisely to shoes, full-body pieces and accessories. In this second step, calculate capsule clothes, there are two scenarios where we can skip trying to chose the best clothes for a specific type:

- If selected number of clothes is zero;
- If selected number of clothes corresponds to the maximum number of clothes for that type and season.

In both cases, our algorithm simply passes on for the next type, without trying to calculate which clothes maximize the number of possible outfits.

This algorithm stops when all capsule clothes types have been calculated, returning a list of clothing items.

4.3.10 Fashion Natural Language Algorithm

As specified in section 3.2.3, to be able to, given an outfit description in natural language, transformed it into a useful structure that we can used to calculate the most appropriated outfit, we implemented an algorithm called `FashionNaturalLanguage`. This corresponds to a natural language processor related to fashion, which input is a natural language description of an outfit and returns a set of pairs attribute value predefined by us. In these pairs attribute value we store outfit general attributes, namely dominant colours, patterns and/or appropriated occasions, as well as outfit specific pieces descriptions.

To implement this algorithm, we needed to have a development corpus, in other words, a set of phrases that exemplified users descriptions of outfits. Our development corpus was created by asking to various users to pretend to have different occasions or at morning, choosing their outfits, and write phrases that would describe the outfits they would imagine or like to receive, to these imaginary scenarios. Since we did not intended to have synonyms or functions to relate unknown words in our context with words in our context, we asked users to use the clothes types and categories, colours, patterns and occasions defined by us in our application. At total, we collected 135 phrases to our development corpus. Some example of phrases from our corpus are: “formal outfit”, “jeans with t-shirt” and “casual outfit mainly with white shirt”.

Our algorithm, `FashionNaturalLanguageQueries`, is constituted by four stages: pre-processing phase, mentioned entities recognition, syntactic and semantic analysis.

In the first stage, pre-processing phase, we receive the textual description about an outfit and we:

- Lower case all words in textual description;
- Substitute . and ; with line break;
- Substitute symbols & and / with, respectively, “and”, “or”;
- Remove punctuation;
- Remove stop words;

In terms of stop words, we used the array of strings defined in [14], however we removed from this the following words from it: “and”, “or” and “with”. We decided to not remove this stop words from input phrase because they were relevant to us while parsing phrases, according to grammar created by us, shown in table B.1. Additionally to these commonly used stop words, we add our own stop words based on the sentences collected. More specifically we defined as stop words: “night”, “day”, “small”, “matching”, “big”, “colourful”, “dinner”, “piece”, “pieces”, “outfit”, “outfits”.

In the second step, mentioned entities recognition, given our textual description without punctuation or stop words, we used an implementation of Aho-Corasick [15] to find words related to clothes types and categories, colours, patterns and occasions and replace them by corresponding tags. In table 4.1 we can observed which tag was associated to each word.

We used Aho-Corasick because, citing Aho and Corasick [1], this is a “simple, efficient algorithm to locate all occurrences of any of a finite number of keywords in a string of text”. While recognizing

[14] `https://github.com/harryaskham/Twitter-L-LDA/blob/master/util/Stopwords.java` (last access on 24/04/2018)

[15] `https://github.com/robert-bor/aho-corasick` (last access on 24/04/2018)

Table 4.1: Correspondence between tags and keywords identified by Aho-Corasick during mentioned entities recognition.

Tag	Keywords
TYPE	top, bottom, fullbody, full-body, shoes, accessories
CATEGORY	t-shirt, no sleeve or short, shirt, blouse, jacket, long sleeve, sleeveless, short sleeve, sweater, cardigan, jeans, trousers, shorts, skirt, skirts, suit, dress, jumpsuit, scarf, scarves, bag, backpack, tie, bow tie, dresses, cardigans, sweaters, suits, jackets
OCCASION	casual, work, meeting, party, date, formal
PATTERN	plain, no pattern, checkered, plaid, dotted, floral, flower, animal, striped, other
COLOUR	yellow, yellow-orange, orange, orange-red, red, pink, violet, purple, blue, cyan, green, light green, white, gray, grey, black, beige, brown

mentioned entities from one input sentence, using the class Trie implemented in Aho-Corasick [15], we returned two separate phrases. One phrase with keywords replaced by correspondent tags and a second phrase with each tag value surrounded by square brackets.

After having our keywords identified, to interpreter textual description of an outfit given by users, after pre-processing it and identifying keywords, we needed to analyse it syntactically. To do this, we developed our own syntactic parser. We opted to develop a semantic grammar, over more recent technologies like neural networks, because our corpus was very small. According to Gavaldà and Waibel [10], a semantic grammar corresponds to a grammar specifically develop to a certain defined domain, so, in this case, a grammar specific to describing outfits. We had several iterations though our corpus phrases to created our grammar as shown at table B.1.

To implement our syntactic parser, based on our semantic grammar, we opted by a bottom-up parser, because we had a very small corpus and our grammar was very simple. According to Crespo [7], a bottom-up parsers corresponds to a parser that receives input phrase and attempts to obtain a specific non terminal symbol, in our case OUTFIT non terminal symbol. Also, according to Crespo [7], bottom-up parsers are characterized by two different type of actions: shift and reduce. Shift correspond to read the next symbol, while reduce corresponds to substitute a list of symbols by a non terminal. In our parser, one example of reduce action is to substitute COLOUR terminal symbol by ATTR_COLOUR non terminal symbol. In our parser we also have shift-reduce operations, in other words, a shift operation that is followed by a reduce operation. One example is when we read TYPE terminal symbol as next symbol, we try to look for ATTR_ATTRIBUTES in our last read symbols and reduce it to ATTR_PIECE non terminal object. To know our previously read symbols, we use a stack, called `tagsStack`, where each time we read a new symbol we perform a push operation, and each time we want to reduce it, we perform one or more pop operations followed by one push action. We were able to implement all correspondent shift and reduce operations to all rules mentioned in table B.1, with exception of defining OUTFIT non terminal symbol as ATTR_PIECES ATTR_ATTRIBUTES. If at some point of our parser, we encountered an unexpected symbol, we immediately return error.

As a final step, to comprehend the semantics behind the textual description given by users, we needed to add some semantic to encountered tags and syntactic structures. As a simplification we opted by joining semantic analysis to the syntactic parser. To perform this we added a second stack to our parser, `valuesStack`. In this stack the correspondent values read are stored as well as objects that

correspond to non terminal symbols in our grammar. For example, when we read TYPE or CATEGORY terminal symbol and we perform necessary actions in `tagsStack`, these same actions are applied to `valuesStack`. In other words, in this case, we pop all characteristics of clothing piece previously stored in `valuesStack`, save them in a set of pairs attribute value and push these set into `valuesStack`. When we obtain OUTFIT non terminal symbol in our `tagsStack`, in our `valuesStack` we have the final set of pairs attribute value will all characteristics of outfit textually described by the user.

One concrete example of phrase in our development corpus is "White date outfit with Dress". In our algorithm first step, pre-processing phase, in this case, we lower case all words and remove stop word "outfit", obtaining "white date with dress". In our second step, mentioned entities recognition, we receive the output from pre-processing phase and obtain two phrases: "COLOUR OCCASION with CATEGORY" and "[white] [date] with [dress]". These two phrases are received in our third and final step of our algorithm, syntactic and semantic analysis, where, while reading "COLOUR OCCASION with CATEGORY", when we read:

- "COLOUR" we push "ATTR_COLOUR" into `tagsStack` and "white" into `valuesStack`
- "OCCASION" we push "ATTR_OCCASION" into `tagsStack` and "date" into `valuesStack`
- "with" we push "with" into `tagsStack` and "with" into `valuesStack`
- "CATEGORY" we push "ATTR_PIECE" into `tagsStack` and "dress" into `valuesStack`

After reading all our input, we evaluate our stacks elements and perform the necessary actions. In this case, we end up with white stored as outfit dominant colour, date as outfit occasion and dress as a clothing item that must constitute our recommended outfit. This information is then used by `OutfitRecommendation` algorithm to calculate the most appropriated outfit.

4.3.11 Outfit Recommendation Algorithm

Another of our main goals was to calculate the most appropriated outfit for specified requirements by users, so, as mentioned in section 3.2.4, we implemented `OutfitRecommendation` algorithm. Since, in English version, we allow users to chose between describing outfit through attribute selection or natural language textual description and how we calculate outfits depends on it, we subdivided this algorithm into two major functions. One, `attributeSelectionOutfitRecommendation`, for dealing with outfit description through attribute selection and another, `textualDescriptionOutfitRecommendation`, for outfit description with natural language.

Through attribute selection, users must select occasion and can, if they want, personalize outfit regarding pieces type, specific clothing piece, dominant colour and if outfit should have accessories or not. In `attributeSelectionOutfitRecommendation` function, we verify these attributes and calculate outfits based on them. In terms of occasion, since this is required we always know to which occasion the user wants to wear the outfit to. So, we are able to only consider clothes that have the chosen occasion as one or only occasion possible to wear them. Concerning outfit pieces type optional argument, it allow users to specify if they desire an outfit with top and bottom pieces or an outfit with a full-body piece. Since this is one of the optional attributes, this attribute can be undefined. If this attribute, outfit pieces type, is undefined, we will try to calculate appropriated outfits with top and bottom pieces as well as outfits with a full-body piece. Alternatively if this attribute is defined we will calculate the outfit accordingly, in other words, if is selected as top and bottom pieces we will try to match top and bottom pieces that make an appropriated outfit, while if is selected as full-body piece, we will try to find an outfit with a full-body piece. Regarding specific clothing piece, this attribute allows users to select a specific item from their current capsule or virtual closet, depending if there is a current capsule

selected or not. If users specify a item to be part of the outfit, we will try find outfits with that piece, otherwise we will try to combine all clothes with each other. If select to simulate a top item as specific piece, we try to combine this with bottoms pieces and shoes. If the selected top piece is defined as shirt in terms of category, we additionally try to combine it with suits, if there is any. Alternatively, if users select to a bottom item as specific piece, we try to combine with tops and shoes. If users select a certain pair of shoes as a specific piece, we try to combine with one piece and two pieces outfits, in other words, with tops, bottoms and full-body pieces. If users select to an accessory as specific piece, we try to combine it, similar to shoes, with one piece and two pieces outfits, with tops, bottoms, full-body pieces and shoes. In terms of dominant colour, we allow users to select one colour to be present in our outfit, in other words, if this attribute is not null, we try to have at least one piece with the chosen colour. Since we defined this attribute as dominant, we considered that this means the top, bottom or full-body piece in our recommended outfit should be this colour, or if piece has multiple colour, this should be one of them. Respecting to have accessories or not, this determines if we should try to match top and bottom pieces and/or full-body piece with accessories from users closet. From these optional attributes, users can customize all, some or none of these, with the exception of having accessories, that if users do not customize its default value is "No". If none of these optional attributes are customized, our outfit recommendation algorithm try to calculate outfits with one piece and two pieces without accessories, meaning match tops, bottoms, full-body pieces and shoes. After calculating possible outfits, complying occasion and optional attributes, outfit pieces type, specific clothing piece, dominant colour and if outfit should have accessories or not, we obtain an list of outfits. If this list is not empty and there is more than one possible outfit, we randomly select one. If there is only one, that the outfit returned. If the list is empty, our algorithm returns an empty list, which is then interpreted, by our Outfit Recommendation functionality, as not existing an appropriated outfit.

Concerning finding an outfit described through natural language, our first step corresponds to call our natural language algorithm, `FashionNaturalLanguage`, to interpret textual description. If our natural language algorithm is not capable of interpret the description given by users, it returns null and `OutfitRecommendation` algorithm immediately stops. Otherwise, if it is able to understand description given by user, it returns a set of pairs attribute value with all characteristics related to outfit in general, as well as outfit pieces descriptions. Regarding outfit pieces, if there is no information about what pieces should constitute outfit, we try to match all clothes from current capsule wardrobe or virtual closet, depending if there is a current capsule selected. Meaning that we match tops, bottoms, full-body pieces, shoes and accessories. Otherwise, if there is information about what pieces should constitute outfit, meaning one or more pieces described in textual description given by user, we try to complete the outfit, similarly to outfit recommendation through attribute selection when there is a specific clothing item selected. In this case, since we can have more than one piece specified, we first obtain from database all described pieces and only them, considering the missing ones, complete the outfit. In this case, we only calculate outfits with accessories when this are described as one of outfit pieces. After having calculated possible outfits with pieces described or, if there are no pieces descriptions, after matching all clothes retrieved from database, we check which ones comply with outfit general attributes mentioned, if they exist. Concerning outfit general attributes, having in account the grammar defined in table B.1, users can customize outfit occasion, colour and/or pattern. For all of these, in our grammar we allow users to define them as various coordinated disjunctive sentence of copulative coordinated sentences, in other words, for example, in terms of colour, users can describe them like "blue and green", "blue and green or black" or "blue or green or black". For all of these attributes, occasion, colour and patten, if they are defined by coordinated disjunctive sentences, in other words, if there is at least one "or", we split by the "or" and check if at least one of the resulting clauses verify. Regarding occasions, for each clause to be true, all outfit clothes should be have as appropriated occasions all occasions mentioned. Relatively

to colours, for each clause to verify, at least one of clothing items that constitute the outfit should have all mentioned colours. Concerning patterns, for each clause to verify, there must at least one clothing piece with each pattern described. Identically to `attributeSelectionOutfitRecommendation` function, after calculating possible outfits regarding pieces described and outfit general attributes mentioned, we obtain an list of outfits. If this list is not empty and there is more than one possible outfit, we randomly select one. If there is only one, that the outfit returned. If the list is empty, our algorithm returns an empty list, which is then interpreted, by our Outfit Recommendation functionality, as not existing an appropriated outfit.

4.3.12 Simulate Purchase Algorithm

Additionally to capsules creation and outfit recommendation, our goal was to help users decide if should buy a clothing piece or not based on possible outfits with it. As mentioned in section 3.2.4, to calculate possible outfits with specified clothing item, we created an algorithm called `SimulatePurchase`. This, with knowledge of new clothing piece type, category, colours, pattern type (and scale if applicable) and appropriated seasons and occasions, determines possible outfits with specified item. Contrarily to `OutfitRecommendation` that allows users to determine if it should have accessories or not and clothes can be from current capsule or virtual closet, this algorithm returns always outfits with accessories and with clothes from virtual closet.

The first step of this algorithm is to retrieve users clothes, by type, from virtual closet, that can create complete outfits with specified piece. In other words, for example, if users simulate purchase of a new bottom piece, we will retrieve all tops, shoes and accessories. After having all clothes retrieved, we create an empty array where we will store possible combinations and start to create outfits. If users select to simulate a new top piece, we try to combine the new top with bottoms pieces, then shoes and finally accessories. Through these three combinations stages we only store the pieces that match, in other words, while matching bottoms pieces, we only save bottoms that match with top and while match shoes, we only save shoes that match with both top to simulate and bottoms selected in previous stage. Additionally, if the top piece simulated is defined as shirt in terms of category, we try to combine it with suits, if there is any. Alternatively, if users select to simulate a new bottom piece, we try to combine with tops pieces, then shoes and finally accessories. If users select to simulate a new full-body piece, we try to combine with tops pieces with category shirt, then shoes and finally accessories. If users select to simulate a new pair of shoes, we try to combine with one piece and two pieces outfits, in other words, with tops, bottoms, full-body pieces and accessories. If users select to simulate a new accessory, we try to combine it, similar to shoes, with one piece and two pieces outfits, with tops, bottoms, full-body pieces and shoes. As final step, this algorithm returns the outfits calculated to Simulate Purchase functionality.

5

Solution Evaluation

Our solution evaluation was consisted by two different kind of evaluations. One for evaluating our solution interface in terms of usability and utility and other for assess our developed natural language algorithm, `FashionNaturalLanguage`.

Concerning usability and utility, with the intention to obtain early feedback from the users about how to build our application in an intuitive way, easy to use and learn, and reduce the number of modifications in our final application, we had three phases of evaluation. Two were during development, evaluation of our low-fidelity non-functional prototypes and formative evaluation of our functional prototypes, and the third one was after finishing our solution, to evaluate sumatively it in terms of usability and utility.

In sections 5.1, 5.2 and 5.3, we will describe the three evaluation phases performed regarding usability and utility in more detail and in section 5.4, we will discuss how we evaluated our natural language algorithm and its results.

5.1 Low-Fidelity Non-Functional Prototypes Evaluation

Motivated by the goal of having early feedback from users, our first evaluation phase consisted on assessing our interface usability through low-fidelity non-functional prototypes. As mentioned in section 4.1 of chapter 4, we developed these prototypes based on examples of what we considered as our main tasks at that moment, more specifically: add a new piece into the virtual closet, create a new capsule wardrobe, obtain a new outfit recommendation and simulate the purchase of a new piece. After implementing these prototypes, we proceed to evaluate them by asking six users to perform some tasks defined by us, while we simulate our application behaviour towards their actions, using Wizard of Oz methodology. This evaluation was performed between June 24 and 26, 2017, at Instituto Superior Técnico. In this phase, since was non-functional prototypes and our main goal was to find improvements in our application interface, we used a think aloud methodology. According to Fonseca et al. [9], this approach, comparing to heuristic or predictive evaluations, values more users opinion and experience on using the interface and also allows to see which aspects of are more difficult or less intuitive to understand in terms of users' perspective.

5.1.1 Evaluation Process

Each evaluation had three steps: introduction, tasks execution and brief conversation. In the first step, we introduced ourselves, our work and the evaluation process, namely maximum duration and number of tasks that we would ask users to do, the tasks itself and our main goal with this evaluation, evaluate our solution and not our users. We also mention that in order to better understand what users thought about our application, including good or bad decisions that we had taken, we would need that users to speak aloud their thoughts during tasks. Before asking users to execute the tasks, we asked if they had any question, since we could not answer any during task execution. In terms of tasks, we asked users to perform four tasks in total, more specifically to:

1. Add a new short sleeve top, with black and white stripes, taking a new picture. This top can be used in spring or summer, in casual days;

2. Create a new summer capsule wardrobe with 30 clothing pieces, called “Summer Capsule”;

3. Obtain a outfit recommendation to a party, today, June 26th, having as dominant colour blue, without any piece in particular;

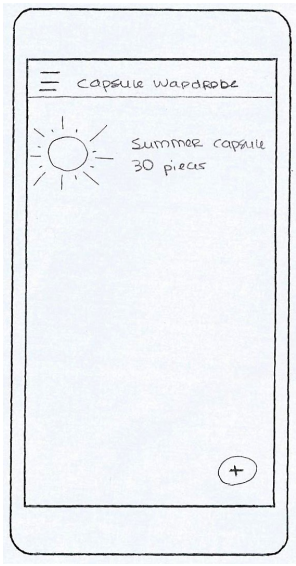
4. Simulate the acquisition of a new long sleeve top, plain, pink, taking a new picture, into the virtual closet. This long sleeve top can be used in spring, in casual days and parties.

Table 5.1: Each task start and correct end.

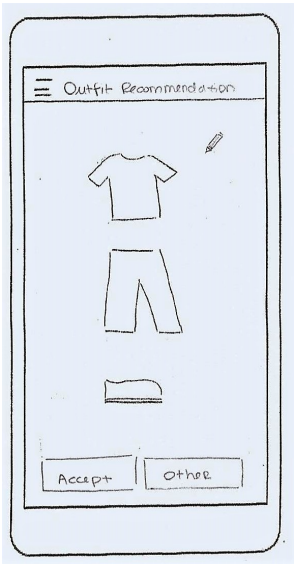
Task	Start	Correct End
Add piece in virtual closet	Home Screen	Finish Process of adding new clothing piece into virtual closet
Create new capsule		Screen with capsule listed
Get outfit recommendation		Outfit suggested
Simulate new clothing piece acquisition		Get possible outfits with piece simulated

For each task we defined the start and correct end. This can be observed in table 5.1. In terms of start, we defined that all tasks should begin in the home screen. Regarding adding a new clothing item, we defined the correct end as users finish the process of adding a new clothing piece in virtual closet. For creating a new capsule, we decided that the correct end was the user reaching the screen where there is the asked capsule listed, as shown in figure 5.1a. Relatively to ask an outfit recommendation, the correct end was to reach the screen represented in figure 5.1b, where is shown an outfit with top, bottom pieces and shoes. Finally, for simulate the acquisition of a new clothing piece, the correct end was to have three possible outfits with the item to simulate, in other words the screen corresponding to figure 5.1c.

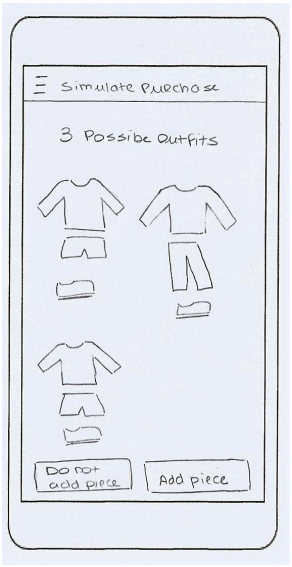
During execution, we provided a written task definition and did not answer any question or give any help, however we have taken notes about users comments or actions during each task. After each task had being executed, we asked users to classify task complexity from 1 to 5, being 1 very difficult to 5 very easy. At the end of evaluation, we asked users for general comments and suggestions, what they would change and what they liked and disliked in the overall.



(a) Correct End Screen for Create New Capsule.



(b) Correct End Screen for Obtain Outfit Recommendation.



(c) Correct End Screen for Simulate New Piece Acquisition.

Figure 5.1: Correct end screens for create new capsule, obtain outfit recommendation and simulate new piece acquisition

5.1.2 Users

With the intention to have in consideration our future users, we attempted to have the same number of female and male users, as well as from different ages. We had in total six users, three of them female and other three male, with ages between 20 and 60, more precisely two users with ages between 18 and 29, three users with ages between 30 and 59 and one user above 60 years old. All users were native Portuguese speakers.

5.1.3 Evaluation Results

From our low-fidelity non-functional prototypes evaluation we could do an analysis on our graphical interface in terms of efficiency, efficacy and ease of learn and use, namely through notes taken from task realization and comments from users and each task classification.

As observed in figure 5.2, in terms of tasks classification, with exception of task number 3, users classified all tasks as 4 or 5, in a scale from 1 to 5, being 1 very difficult and 5 very easy. More specifically, task 1 and 4 were classified by four users as 4 and by two users as 5, task 2 were considered by all users as very easy and for task 3, one user classified as 3, two users as 4 and three users as 5.

We can understand each task classification from the notes taken by us and comments received. In general, we observed that there are two major problems: the application language is an obstacle while understanding some terms related to fashion (eg. difference between clothing item and outfit or clothing piece top type confused with up/down) and the menu button was not easily identified. In terms of the first task, *add a new clothing item to the virtual closet*, besides the problems with the menu button and the language barrier, users had shown difficulties with finding the button to add a new clothing item, to understand that it was supposed to confirm the colours automatically detected and not to select them and to realize that it was possible to choose more than one season. In the following tasks, after knowing which and where the menu button was, users did not have any problems with it. So, in the second task, *add a new capsule wardrobe*, they did not had any obstacles to performed it. However in the third and fourth tasks, respectively *ask for an outfit recommendation* and *simulate a new piece in the virtual*

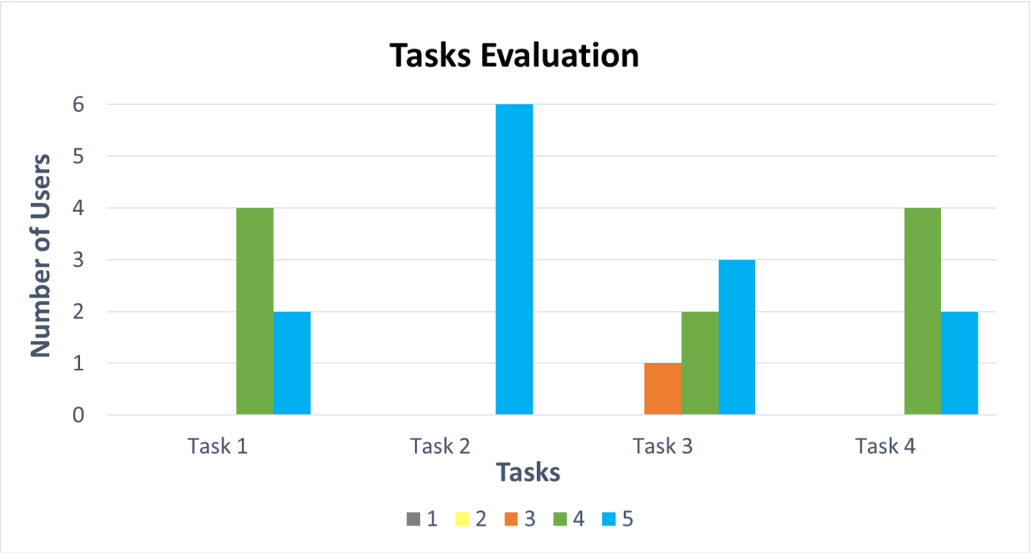


Figure 5.2: Evaluation results for each task of low-fidelity non-functional prototypes.

closet, there were some difficulties. More specifically, in the third task users had some problems finding the correct colours from the grid view and understanding that some steps were optional. Besides, it was suggested that in the final screen buttons Other and Accept were changed sides. In terms of the fourth and last task, the main handicaps were related to distinct between add a new piece to the virtual closet and simulate the purchase of a new clothing item and confusion between clothes types and categories, more specifically that the category depends directly on type, which means that in a first step type they should have selected the correct type and only then category.

Based on the results got from our low-fidelity non-functional prototypes evaluation, we could concluded that our interface needs some improvements, namely:

- Replace the menu button with one more intuitive, to help users to quickly identify it;
- Build our application with multi language support, at least Portuguese and English, since we want to make available our application to everyone;
- Review terms used in our applications, namely terms related to fashion, in order to have content more easy to understand;
- Increase size of the add a new clothing piece button;
- Divide clothing piece type and category setting step in two, in adding a new clothing item in the virtual closet and simulate a new purchase, in order to avoid confusion between clothing piece type vs category;
- Also in the process of adding a new piece and simulate a new purchase process, in the step of confirming the dominant colours detected, add a title to inform users that they must confirm the colours detected, or in case of necessity edit them;
- Distinguish better from single to multiple selection options and from mandatory to optional steps;
- Add a small explanation in the simulate purchase functionality with intention to better show differences between this and adding a new clothing piece in the virtual closet.

5.2 Formative Functional Prototypes Evaluation

With the goal to find possible improvements in our application interface and/or obstacles in it that could confuse or cause users to make mistakes, we performed a formative evaluation while we were implementing our solution. This evaluation was performed during the week of March 19-23, 2018, and, similarly to low-fidelity non-functional prototypes evaluation, was also performed at Instituto Superior Técnico. The formative evaluation consisted on performing specific tasks defined by us using our application on a 5.0"Android smartphone. We had some data stored in it, namely clothing items and wardrobe capsules, to better simulate what our application would like after users had already put some clothes and created some capsules wardrobe. In this stage, we decided to use a think aloud methodology instead of heuristic or predictive one because we valued more users opinion and experience on using our application and also this way we could see which aspects of our application would be more difficult or less intuitive to understand in terms of users' perspective. Besides finding improvements in our application interface, we also had the goal to build our natural language evaluation corpus, so we asked users to write ten sentences describing clothes coordination examples.

In this phase, since it would already involve our functional prototypes and there would be more tasks to perform, we decided to do a pilot test to make sure our solution was correctly functioning and would not crash during evaluation and also to make sure our tasks were clearly defined, without any ambiguity or obstacle to test users understanding. This pilot test was performed with a 42 years male user, the day before tests began.

5.2.1 Evaluation Process

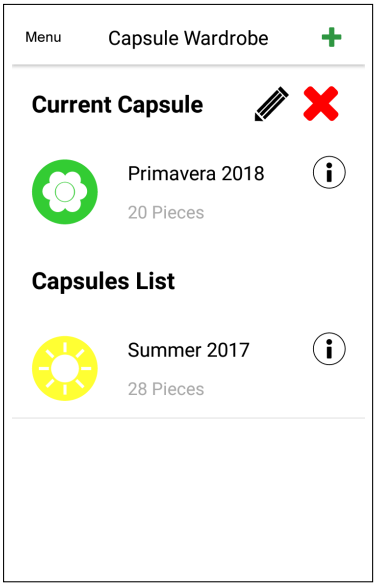
The process of evaluating the functional prototypes was also very identical to low-fidelity non-functional prototypes evaluation. We had three steps: introduction, tasks execution and brief conversation. In the first step, we introduced ourselves, our work and the evaluation process, namely maximum duration and number of tasks that we would ask users to do, the tasks itself and our main goal with this evaluation, evaluate our solution and not our users. We also mention that in order to better understand what users thought about our application, including good or bad decisions that we had taken, we would need users to speak aloud their thoughts during tasks. Before asking users to execute the tasks, we asked if they had any question, since we could not answer any during task execution. In terms of tasks we asked our users to perform seven tasks:

- Create Maria profile. She lives in Portugal and she was born in June 5th, in 1970.
- Add a casual black pants to virtual closet that can be worn all year.
- Delete beige shirt from virtual closet.
- Create a new Spring capsule wardrobe, named "Spring 2018" with maximum of 20 clothing pieces and defined it as current.
- Delete "Spring 2017" capsule wardrobe.
- Ask for dress and shoes recommendation for a party on April 1st.
- Simulate the acquisition of a new t-shirt with big black and white striped pattern into the virtual closet that can be worn at work during Spring, Summer and Autumn.

For each task, just like in low-fidelity non-functional prototypes evaluation, we defined the start and correct end. This information can be observed in table 5.2. In terms of start screen we defined that for

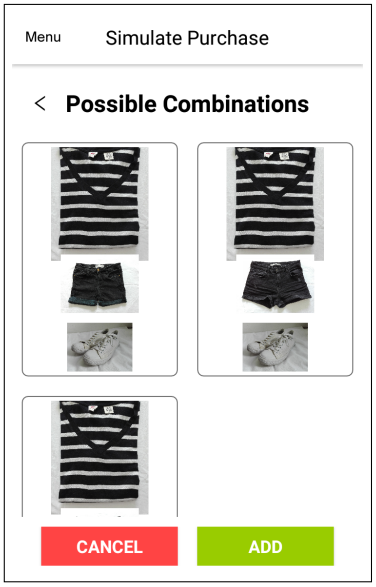
Table 5.2: Correct end for each task.

Task	Start	End
Create Maria profile	Smartphone home screen with application	Profile created
Add pants in virtual closet	Home screen	Clothing piece added into virtual closet
Delete specific piece		Clothing piece removed from virtual closet
Create new capsule		Capsule created and selected as current
delete specific capsule		Capsule removed from capsule list
Obtain outfit recommendation		Outfit suggestion with dress and shoes
Simulate new clothing piece acquisition		Possible outfits with piece simulated listed



(a) Correct End Screen for Create New Capsule and Select as Current.

(b) Correct End Screen for Obtain Outfit Recommendation.



(c) Correct End Screen for Simulate New Piece Acquisition.

Figure 5.3: Correct end screens for create new capsule and select as current, obtain outfit recommendation and simulate new piece acquisition.

every task, except *create Maria profile*, it would be the home screen. To create Maria profile task, we defined that the start would correspond to our smartphone home screen with our application. In terms of correct end for creating Maria profile we defined that would the our profile would be created. For second and third task, *add and remove a piece from virtual closet*, we respectively defined that the correct end would be to have the piece specified added or removed from the virtual closet. Identically, for fourth and fifth task, *create a new capsule* and *delete a capsule*, the correct end was to have a new capsule added and selected as current or a specific capsule to be removed from the capsules list. In figure 5.3a,

we can see the result of performing both fourth and fifth tasks, more specifically we can observe in the upper section of the screen, “Current Capsule”, the capsule Primavera 2018 is selected and in the lower section, “Capsules List”, there is no more a “Spring 2017” capsule listed. For the sixth capsule, ask for an outfit recommendation, the correct end was to have an outfit suggestion just like figure 5.3c. And, finally, for the seventh and final task, *simulate the acquisition of a clothing item*, the correct end would correspond to be shown three outfits as possible combinations as shown in figure 5.3c.

During execution, we provided an written task definition and, as mentioned before, we did not answer any question or give any help. However, we have taken notes about users comments or actions during each task regarding possible improvements or problems encountered. After each task had being executed, we asked users to classify the task complexity from 1 to 5, being 1 very difficult to 5 very easy. At the end of all tasks performed, we asked users for general comments and suggestions, what they would change and what they liked and disliked in the overall.

In this evaluation process, comparing to low-fidelity non-functional prototypes, we had an additional step. As for the fourth and final step, with the intention to build an evaluation corpus to our natural language functionality, we asked users to give us ten sentences describing outfits. For this phrases we had given the same rules that we give users to construct development corpus. We asked users to pretend to have different occasions or to be at morning, choosing their outfits, and write phrases that would describe the outfits they would imagine or like to receive, to these imaginary scenarios.

5.2.2 Users

Similarly to low-fidelity non-functional prototypes evaluation, we attempted to have an equal, or almost equal, amount of female and male users. In this case, we had a total of five users, three of them female and other two male, with ages between 20 and 26. Since all users were native Portuguese speakers we tested the application interface in Portuguese.

5.2.3 Evaluation Results

From our formative evaluation notes taken, users comments and each task classification we could do an analysis on our application interface in terms of usability, more specifically ease of use, and come up with a list of improvements and things to change.

As observed in figure 5.4, in terms of tasks classification only third task, *delete specific clothing item from virtual closet*, was considered very easy by all users, in other words, classified as 5 in a Likert scale from 1 to 5, being 1 very difficult and 5 very easy. We can also note that the first and last tasks, respectively *create Maria profile* and *simulate the acquisition of a new clothing piece*, were both classified as 5 by four of five users and one user quantified as 4. For tasks five and six, correspondingly delete a specific capsule wardrobe ask for an outfit recommendation, we can see that the majority of users classified both tasks as 4 out of 5. More specifically, for the sixth task four users classified as 4 and one as 5 and for fifth task three users evaluated as 4 and two as 5. For the remaining tasks two and four, respectively add a new clothing item and create a new capsule wardrobe and make it current, we can note that these were the most difficult to users. More precisely, task two was classified as 2 by two users, meaning difficult in our scale, and the three remaining users classified it as 4. Task four was quantified as 3 by one user, 4 by two users and as 5 by the remaining users.

We can better understand the reason behind each task classification from the notes taken by us and comments received.

In a more general view of our application, there was one user that highlighted the grey colour of “Next” buttons and others, like add from home screen, could be misinterpreted because of their colour, since normally grey is associated with unimportant information.

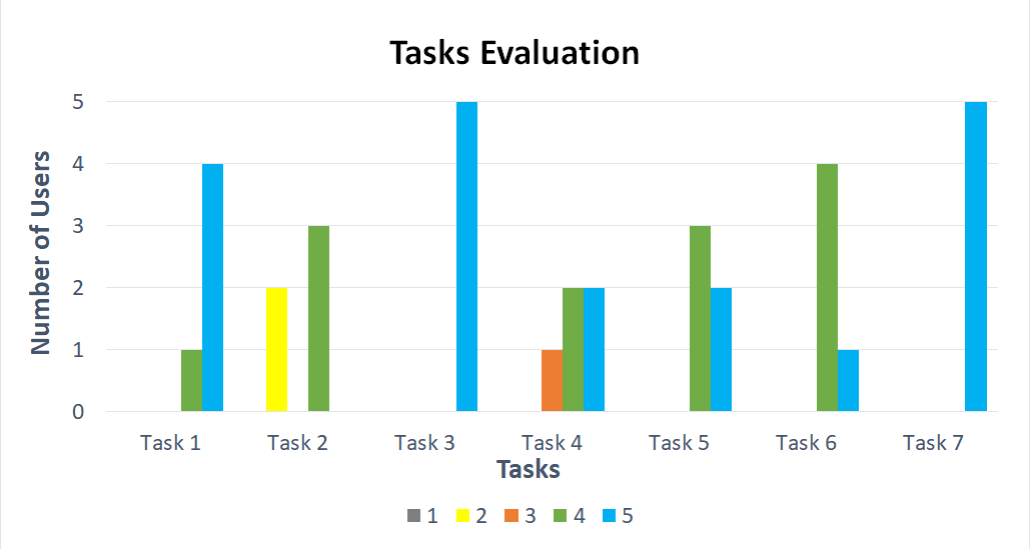


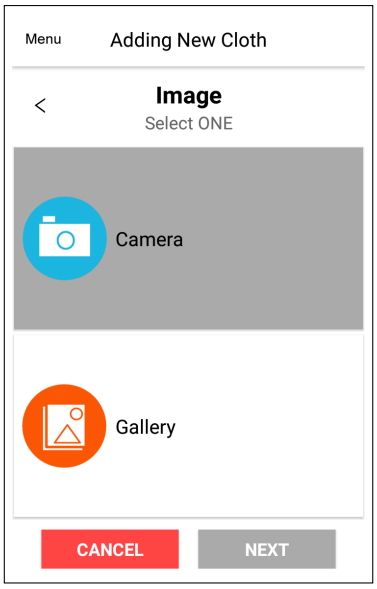
Figure 5.4: Classification results for each task of functional prototypes formative evaluation.

For the first task, *create Maria profile*, our main intention was to see if our decision for location to be north and south hemisphere would cause problems to users, so we said that Maria was currently living in Portugal, with mentioning in which hemisphere Portugal is located. We could concluded through mostly our observation notes that for some users it was not 100% clear which Hemisphere was Portugal in and for that reason, this task was not considered as very easy by all users. Besides this issue, users also considered the year drop down list too long. In users opinion, it would be more efficient and take less time to just type in the year.

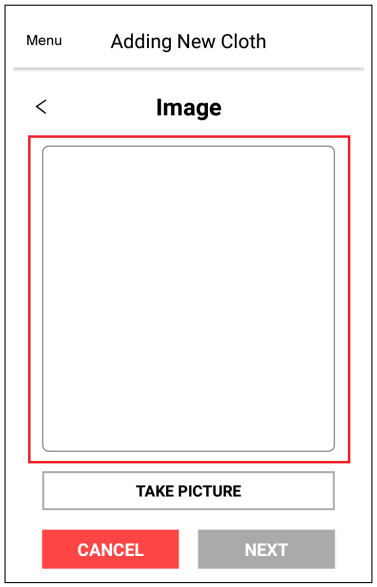
Concerning the second task, *add a new clothing item to virtual closet*, users had shown very relevant difficulties, namely confusion with home screen add button and disoriented while take picture of new clothing item. In terms of the first problem, confusion with home screen add button, this was caused because users did not read the text above button and since the task started by “Add” and the button name was “Add” they assumed that was the button to be pressed. Regarding the task of taking picture of the new clothing item, the problem was that users though once they selected “Camera” option from screen shown in figure 5.5a and continued for next screen the camera would automatically be open and/or the area highlighted by the red rectangle in figure 5.5b was the camera preview and that was not already working. In that moment, what we had implemented was that users should press “Take Picture” button in figure 5.5b to open the camera. With users thoughts and comments while and after performing this task, we came to the realization that this was not the most intuitive and would actually cause problems. Additionally to these two issues, there was one user that comment it was not intuitive in which screens it was possible to select only one option or multiple options. In his opinion, the application interface was very consistent, however there was this detail that cause him some apprehension when, in this task context, users had to select clothing piece appropriated seasons.

Regarding the third task, *delete a specific clothing item*, there was not any problems with performing it and for that reason all users quantified as very easy as shown in figure 5.4.

Relatively to task four, *create a new Spring capsule wardrobe and make it current*, there were some troubles in completing it, more specifically in selecting the new capsule as current. All users were looking for a parameter in the process to create a new capsule that would dictate this as current. We also concluded that the buttons highlighted by the red rectangle cause confusion. We observed that the majority of users pressed “Create New” button without realizing that this would add a new current capsule and not simply a new one like the plus icon at toolbar. Additionally to these issues, users



(a) Select new clothing piece image source with camera option selected.



(b) Take picture of new clothing item screen.

Figure 5.5: Select new clothing piece take picture problematic screens.

mention that start and end dates were not intuitive, they looked random numbers and not capsule start and end dates. For this aspect, users though that these two date should have month name instead of number and preferentially be a little apart from each other or even be presented in two lines. Aside from this, one user mentioned that the list of capsules should be ordered by most recently created.

Similarly to fourth task, the fifth task, *delete specific capsule wardrobe*, bring also some obstacles to users to complete it. The problems observed were the meaning of the information icon and the fact that pressing the capsule was different from information icon. In that moment we had implemented that pressing the capsule would redirect to a screen where all clothes from the capsule would be shown and pressing the information icon would lead us to see capsule characterization, where we could choose to edit or delete the capsule. We observed that the step of pressing the information icon to delete capsule was not as intuitive to users as we thought. That was verified by observing users try several hypothesis like for example click on or long press a capsule or select the red cross icon in the current capsule section.

For the sixth task, *ask for an outfit recommendation*, there were some obstacles that users faced while performing it, more precisely in terms of selecting outfit clothes. We observed that users correctly selected all outfit attributes until the Outfit Personalization screen shown in figure 5.6. This was when users started to hesitate and have problems. The first obstacle was which option they should select in Outfit Pieces. Regarding this field, all users had correctly chosen the option “Full body and shoes”, however all users hesitated a little before clicking in that option. After this, we observed the second problem. Users though that the Specific clothing piece field was meant to select outfit pieces categories (for example dress) and not a specific clothing item. They realized the difference when they notice that could go through the clothes types, categories and select a clothing item and not choose one category. Besides these problems, we also note that the initial screen, explaining the functionality, was totally ignored by users or not completely read by the ones who tried.

For the seventh task, *simulate a new clothing item into virtual closet*, similarly to the third task, there were not problems in completing it and that can be proved with task evaluation results shown in figure 5.4. Regarding this task, all users classified as very easy. However we observed that, similarly to outfit recommendation task, the explanatory text in the first screen of this functionality was completely

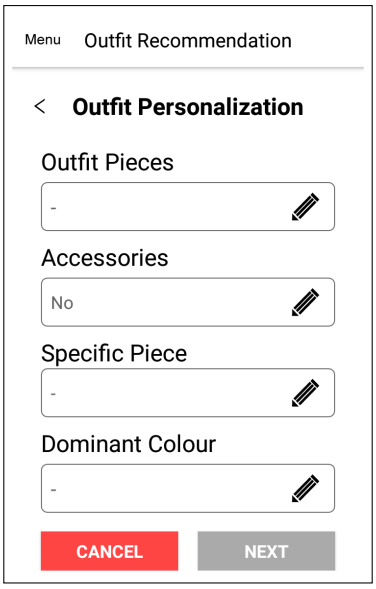


Figure 5.6: Outfit recommendation personalization screen.

ignored by users, or when they tried to read it users gave up halfway through it.

Based on the results got from our functional prototypes formative evaluation, we concluded that our interface needs some improvements, more specifically:

- Choose another colour as accent for application, to avoid have important buttons with grey colour;
- Allow to write birth year, additionally to be able to select from drop down list;
- Have a more intuitive way to users say where they live;
- Rename home screen button that has as goal redirect to outfit recommendation functionality from “Add” to “Obtain Outfit Recommendation”;
- Automatically open camera or gallery after users selected image source and clicked the “Next” button, instead of having to click the “Take Picture” button;
- Add space between capsule start and end dates and have month name instead of number;
- Allow users to select capsule as current while creating it;
- Order capsules list by most recently created;
- Rethink add new and select current capsule buttons as well as remove and replace current capsule selected;
- Substitute information button in capsules by two new ones, an edit and a delete buttons;
- Rethink outfit pieces and specific clothing piece personalization;
- Reduce explanatory text size from outfit recommendation and simulate acquisition functionalities.

5.3 Summative Functional Prototypes Evaluation

In order to assess our final solution regarding usability and utility we conducted two different tests, usability tests and case studies. We choose both because we know, from Fonseca et al. [9], that a good

solution must have high usability and high utility and these these two tests, complement each other, since it allow us to assess both usability and utility.

In the following sections 5.3.1 and 5.3.2, we describe in more detail each test process, users and results.

5.3.1 Usability Tests

With usability tests we were able to evaluate the efficiency, effectiveness and user satisfaction of our final solution. This evaluation was performed between April 16-26, 2018, and, equivalently to previous evaluations phases, it was also at Instituto Superior Técnico and consisted on performing tasks defined by us, using our final solution installed on a 7.0"Android tablet. Similarly to formative evaluation functional prototypes, we had some data stored in it, more specifically clothing items and wardrobe capsules, to better simulate what our application would like after users had already put some clothes and created some capsules wardrobe.

In this final evaluation stage, since we wanted to clearly evaluate our solution efficiency, effectiveness and user satisfaction, we chose to collect data from each test performed and treat them statistically, using direct observation. We chose to collect time taken to complete each task, number of mistakes performed and quantity of tasks completed successfully, and treat them statistically, since, according to Fonseca et al. [9], these are the most commonly used. In order to correctly collect these data, we had to clearly each task and its correct end. Besides these measurements, that allows us to assess mostly efficiency and effectiveness, we also asked ask users to answer a usability quiz, with the intention to evaluate their satisfaction with our solution.

To make sure our tests were all performed under the same conditions, we had previously defined a usability test script. Identically to formative evaluation functional prototypes, we conducted pilot tests to test and validate our evaluation script. These pilot tests were performed with two users, one male and one female, both with more than 40 years old, two days before tests began.

5.3.1.1 Evaluation Process

The process of evaluating the functional prototypes was in general identical to previous evaluation phases. We had three steps: introduction, tasks execution and usability questionnaire.

In the first step, we introduced ourselves, our work and the evaluation process, namely maximum duration and number of tasks that we would ask users to do, the tasks it self and our main goal with this evaluation, evaluate our solution and not our users. We mentioned that we would collect some data regarding users interaction with our application, but we would use these data to evaluate our solution and not the users. We also asked if they had any question, since we could not answer any during task execution. Additionally, in this phase, we allowed users to freely interact with our application during five minutes. As a final step before performing any task, we asked users to fill in a questionnaire, used with the intention to obtain personal informations about users, namely gender, sex, age and experience with Android and/or mobile applications.

Regarding tasks, we used the ones of formative evaluation with small differences to adapt to changes implemented in the application after formative evaluation. More precisely, we asked our users to:

- Create Maria profile. She lives in Portugal and she was born in June 5th, in 1970.
- Add a casual black pants to virtual closet that can be worn all year.
- Delete beige shirt from virtual closet.

- Create a new Spring capsule wardrobe, named “Spring 2018” with 10 tops, 4 bottoms, 1 full body piece, 2 shoes and 3 accessories. Define it as current.
- Delete “Spring 2017” capsule wardrobe.
- Ask for an dress and shoes recommendation for a party on May 12th.
- Simulate the acquisition of a new t-shirt with big black and white striped pattern into the virtual closet that can be worn at work during Spring, Summer and Autumn.

Table 5.3: Correct end for each task.

Task	Start	End
Create maria profile	Tablet home screen with application	Profile created
Add pants in virtual closet	Home screen	clothing piece added into virtual closet
Delete specific piece		clothing piece removed from virtual closet
Create new capsule		Capsule created and selected as current
Delete specific capsule		Capsule removed from capsule list
Obtain outfit recommendation		Press finish button on outfit recommendation last screen
Simulate new clothing piece acquisition		Press finish button on purchase simulation last screen

For each task, just like in previous evaluation phases, we defined the start and correct end. This information can be observed in table 5.3. In terms of start screen we defined that for every task, except *create Maria profile*, it would be the home screen. For create Maria profile task, we defined that the start would correspond to our tablet home screen with our application. In terms of correct end for create Maria profile we define that would the our profile would be created. For second and third task, *add and remove a piece from virtual closet*, we respectively defined that the correct end would be to have the piece specified added or removed from the virtual closet. Identically, for fourth and fifth task, *create a new capsule* and *delete a capsule*, the correct end defined was to have a new capsule added and selected as current or a specific capsule to be removed from the capsules list. For the sixth and seventh task, *ask for an outfit recommendation* and *simulate the acquisition of a clothing item*, the correct end was to press the finish button on last screen of outfit recommendation and simulate purchase processes, respectively.

During execution, we provided an written task definition and as mentioned before we did not answer any question or give any help, however we have taken notes about errors made by users. After each task had being executed, we asked users to classify task complexity from 1 to 5, being 1 very difficult to 5 very easy. Contrarily to other evaluation phases, instead of a brief conversation about overall application after tasks being performed, since was our goal to evaluate usability, and not users opinion, we asked users to fill in an usability quiz, the System Usability Scale. According to Kortum and Bangor [16] and Bangor et al. [2], the System Usability Scale is a questionnaire created with the intention to measure

subjective usability, in other words, users satisfaction and if in their opinions it is intuitive. We chose this quiz, over other alternatives, because, according to [2], is technology independent, in other words, can be used in our application since it does not depend on technology used, it does not take too long to be fill by users and its result vary from 0 to 100, directly proportional to usability.

5.3.1.2 Users

In this phase, since it was a final evaluation stage where intended to prove our solution value in terms of usability, through statistic analysis of collected data, we conducted tests with more users, comparing to low-fidelity non-functional prototypes evaluation and functional prototypes formative evaluation. As mentioned by [9], this way we can have a significant sample of how our real users would interact with our application. For this reason, we performed usability tests with 20 users, of which about 50% were female and the other 50% were male users, between 18 and 25 years. All users have a smartphone and use it daily. From these, about 80% have Android devices, while 20% have iOS devices. All users use their smartphones everyday.

5.3.1.3 Evaluation Results

As mentioned before, from these usability tests, we collected quantitative data, namely quantity of tasks completed successfully, time taken to complete each task and number of mistakes performed as well as task easiness in users opinion. Relatively to errors, we also annotate which errors were made.

Regarding tasks completed successfully, all 20 users were capable of finishing all tasks. Consequently, we had a 100% success rate for completing each task.

Concerning time spent in each task, considering data represented in appendix table C.1, we started by calculating mean and median of each task to, according to [25], respectively discover average time spent performing each task as well as “central value when all observations are sorted in order” [25]. Besides these two statistics measures, we decided to calculate standard deviation in order to analyse how parsed were our values. In other words, we computed average data distance to mean value. Additionally to these three statistics measures, we calculated the coefficient of variation for each task, which expresses standard deviation “as a percentage of the mean” [25]. The results of each of these four measures for each task can be observed in table 5.4.

Table 5.4: Mean, median and standard deviation, in seconds, for time spent in each task by users.

Task	Mean	Median	Standard Deviation	Coefficient of Variation
1	34,05	32,5	9,71	28,51%
2	82,30	81,5	24,54	29,82%
3	26,10	26	6,46	24,73%
4	61,65	59,5	8,88	14,41%
5	11,15	9,5	8,34	74,83%
6	62,10	55	16,91	27,23%
7	66,50	62,50	16	24,07%

From values obtained in table 5.4, we concluded that this was not enough to analyse our data. We know from Whitley and Ball [25] that mean, standard deviation and coefficient of variation are influenced by unusual lower and higher values, called outliers. In other words, we can obtain high values on standard deviation and coefficient of variation while still having little dispersed data if exists one or more outliers. We can clearly observe this happening in task 5, where due to an outlier, more precisely one occurrence of an extremely high time, mean and standard deviation are highly influenced by it.

To better understand distribution of collected data regarding time and the existence, or not, of outliers, we computed quartiles values for each task. These correspond to equally divide data into four sets of 25% range each. Additionally to quartiles, we also computed, from quartile 1 and 3 values, interquartile range which corresponds to the "range between the bottom and top quartiles, and indicates where the middle 50% of the data like", citing [25]. We can observe quantitative quartiles and interquartile range values for each task in table 5.5 and graphic representation of these in figure 5.7.

Table 5.5: Quartile values, in seconds, for time spent in each task by users.

Task	Minimum	Q1	Median	Q3	Maximum	Interquartile Range
1	20	26,75	32,5	40,5	51	13,75
2	42	67,75	81,5	91	117	23,25
3	15	22,5	26	29,25	32	6,75
4	50	56,75	59,5	63,25	72	6,5
5	6	7,75	9,5	11	15	3,25
6	46	49	55	72,75	100	23,75
7	48	51	62,5	79	96	28

From table 5.5 and figure 5.7, we can observe that specially for the third, fourth and fifth tasks interquartile ranges fluctuate between 3,25 and 6,75 seconds, in other words, the middle 50% results collected for those tasks are very compacted, similar between each other. For the first task the interquartile is a little higher, about 14 seconds. For the second, sixth and seventh tasks interquartile range is relatively high, about 23 to 28 seconds, in other words, the distance between minimum and maximum value of 50% results collected for this task is about 23 to 28 seconds. Analysing figure 5.7, we can also observe that second, third, fourth and fifth tasks have outliers. More precisely, in these tasks we can observe the existence of values uncommonly high comparing to remaining data.

Analysing errors registered from each evaluation we can better understand why some users took longer to finish some task than other users.

For the first task, *create Maria profile*, only one user actually made a mistake, not filling the gender field, which lead to an error dialogue appear. In this case, user closed the dialogue, added the gender and finish profile. The remaining users completed this task successfully without any errors. We believe that the spent time difference while performing this task was only due to user capably of inserting Maria's name and date of birth.

For the second task, comparing to first task, we registered more errors. In this second task, *add a new clothing item into virtual closet*, in our opinion, the most serious error registered was from one user that simulate the acquisition of the new clothing piece, instead of directly add this item into virtual

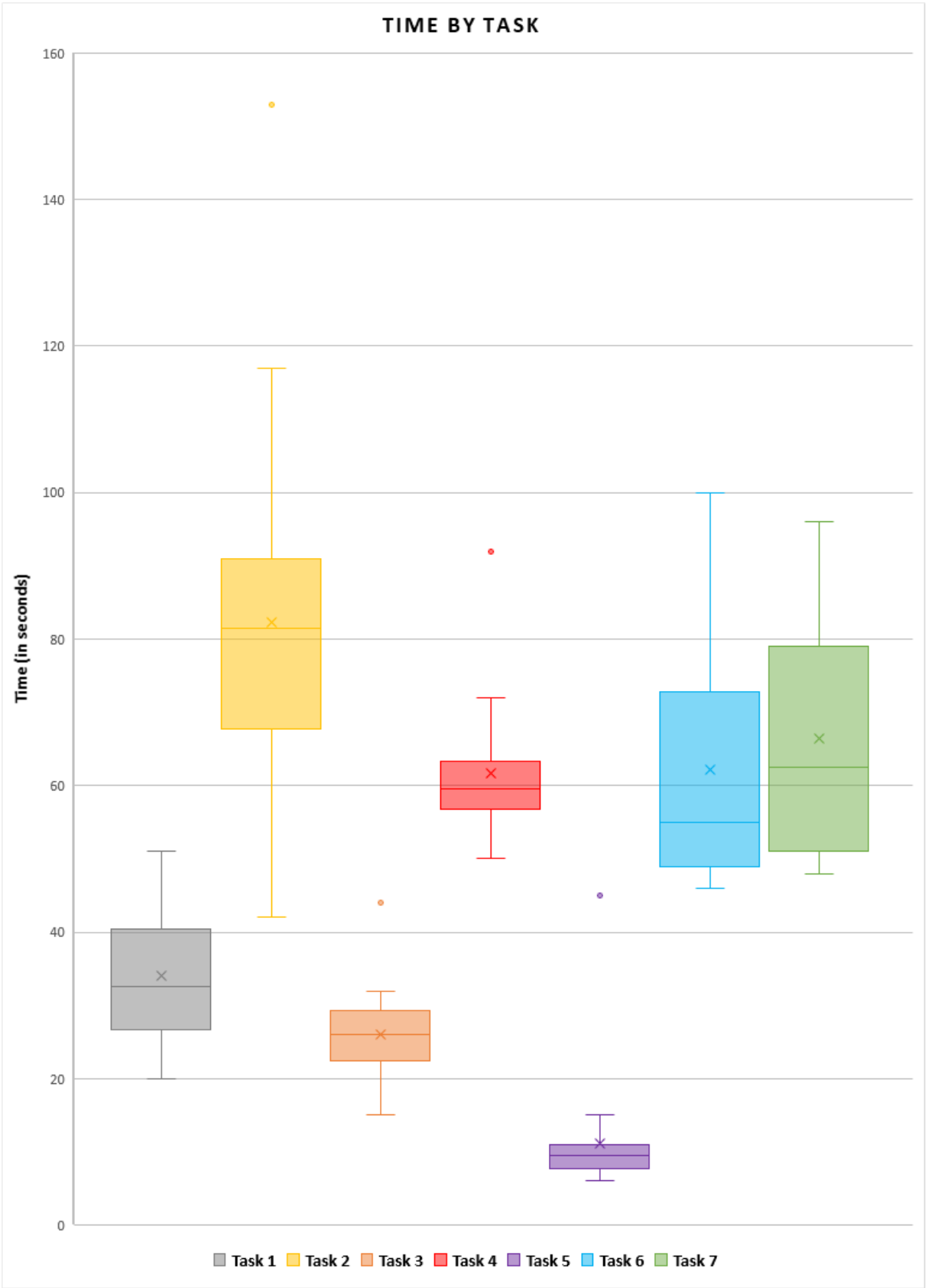


Figure 5.7: Quartiles graphic representation, in seconds, for time spent in each task by users.

closet. This user still added the clothing item to virtual closet, however took more time than remaining users, because the user had to wait for our application to calculate the possible combinations with this

new piece to then add the item to virtual closet. Besides, there were three users that wrongly clicked on "Obtain Outfit Recommendation" button on home screen. However, all these users immediately realise this was not the right functionality. Additionally to these errors that cost time to users, there were also errors that did not make users took longer, however were still mistakes. We are talking about selecting wrong clothing piece category, colour and/or appropriated occasions. This kind of mistakes we believe that were due to not reading properly the task enunciation, since in it we had all information necessary about the new clothing piece to be added. Similarly to the first task, there were users that did not commit any error but took longer than others, in this case due to reread task enunciation when selecting clothing piece information, like colour, seasons or occasions.

For the third task, *delete clothing item*, regarding errors, the majority of users did not commit any error, however there was one user that made one mistake, selected wrong clothing piece category. In this case, user went back to clothing piece categories and selected the correct one. These one users is the reason of the uncommonly high value shown in figure 5.7.

Regarding fourth task, *create a new capsule wardrobe and make it current*, all users performed this task without making any error.

Concerning the fifth task, *delete specific capsule*, there was one user that made the mistake of initially pressing capsule wardrobe instead of delete icon. This resulted in outlier presented in figure 5.7, since pressing capsule correspond to show capsule's clothes and our application had the limitation of taking time to load clothes images due to images quality and how we added them. In this case, user went back to previous screen and then clicked on delete icon.

For the sixth task, *ask for a dress and shoes recommendation*, eight users made the mistake of trying to select specific clothing item, more precisely a dress. From these, seven realise when seeing dresses from current capsule, that it was not supposed to select a specific one, however there was one user that decided to chose one. Besides these eight users, there was also one user that made a mistake in this task. In this case, user did not select the combination type of clothes, in other words, did not select that the outfit suggested should be a fullbody piece with shoes.

Relatively to the seventh task, *simulating the acquisition of a new clothing piece*, one user initially clicked wrongly on "Obtain Outfit Recommendation" button on home screen and other three users made errors while classifying new piece in terms of colour and/or occasion. In terms of colours, these three users firstly only selected one colour, which lead to an error dialogue to appear. In these cases, users closed the dialogue and added the missing colour. Regarding occasion, one user initially did not select any occasion, leading to a error pop up dialogue to appear, and other two users wrongly selected appropriated occasions, what we believed that was due to not properly read task enunciation.

Besides registering time and errors for each task, as mentioned before, we also obtained users opinion about easiness of performing each task, through asking them, after each task, how they classified it in a Likert scale from 1 to 5, being 1 very difficult to 5 very easy. In figure 5.8, we can observe results obtained concerning tasks classification. More precisely, we can observe that the first, third, fourth and fifth tasks were classified as 4 or 5 in our Likert scale by all users, in other words, respectively as easy or very easy. We also can observe that the second, sixth and seventh tasks were classified by users as 3, 4 or 5 in our Likert scale. For these tasks, the number of users that classify these as 3 in a 1 to 5 scale were relatively small comparing to the number of users that classified as 4 or 5.

Additionally to these measurements specific to tasks, we also collect overall users satisfaction through System Usability Scale quiz. Considering System Usability Scale rules to calculate result, we concluded that in average our solution has a subjective usability of 83,54 with standard deviation of 10,41.

From collected measurements, more precisely number of tasks successfully finished, time spent performing each task, errors made, tasks classifications and users satisfaction, we can conclude that there are room for some improvements, since some mistakes were made by users and some tasks were

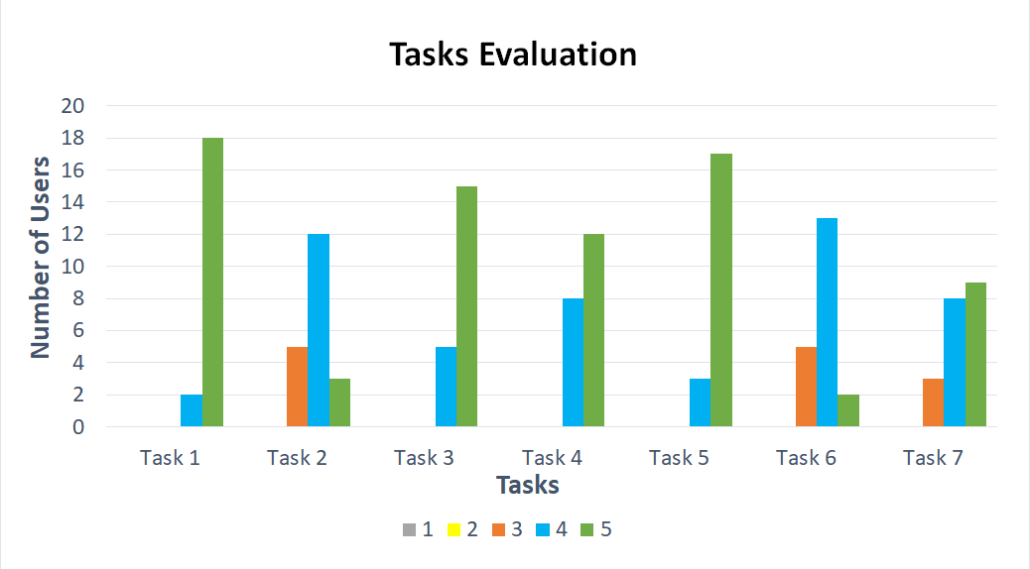


Figure 5.8: Classification results for each task of functional prototypes summative evaluation.

not classified as easy or very easy by all users. However all users successfully finished all tasks, were able to recover from errors and overall users satisfaction was relatively high, about 83,54%. So, in our opinion, we consider that we were able to create an intuitive, easy to use and learn solution that also allow users to recover from errors made.

5.3.2 Case Studies

Besides usability tests, we conducted three cases studies in order to asses our solution utility. In these tests, we asked users to perform the same tasks of usability test, while thinking aloud. Our goal with these tests was to acknowledge our application utility, in other words, real users opinions about our application while interacting with it as well as if they found something interesting that they were unaware or realize that there were more features that we did not initially found interesting. With this goal in mind, we chose users that corresponded to our users target group, users interested in fashion related application and/or with necessity of an tool to help them organize their closet into capsule wardrobes and/or reduce the task of finding an appropriated outfit for everyday.

5.3.2.1 Evaluation Process

In these studies, similarly to usability tests, we first introduced our solution, describing its main goals and functionalities, maximum duration and number of tasks that we would ask, the tasks itself and our main goal, assess our solution utility. We mentioned that we would need them to think aloud in order to be able to assess utility, but he would not judge them in any way. Additionally, in this phase, we allowed users to freely interact with our application during five minutes. After this time, asked users to perform defined tasks. During each task execution we recorded users' thoughts and comments.

5.3.2.2 Users

Unlike the previous tests performed, in the case studies we selected users that would be more similar as possible as our target users. In other words, users that would have interest in fashion related applications and/or necessitate an application to create capsule wardrobes, obtain outfit recommendations and/or simulate the acquisition of new clothing items into their closets. These were users that did not

participate in the usability tests or previous evaluation stages. The three users were all female, with ages between 18 and 25 years, and all were very interested into fashion and conjugate clothes into appropriated outfits.

5.3.2.3 Evaluation Results

From these case studies, we could collect some opinions about our application. More specifically, users expressed that although they might not use the application on a day-to-day basis, it would be "useful specially to create new and unpredictable combinations", also to "plan outfits for travels" or "when is necessary a more formal or specific kind of outfit". They also said that allowing to chose a certain piece while recommending outfits were helpful since "sometimes the major difficulty is to find outfits with a specific piece that we want wear". Additionally, users though that the possibility of see how a certain new piece can coordinate with users' clothes is "very useful because helps buy only clothes that match with a high variety of clothes" that they have.

5.4 Fashion Natural Language Algorithm Evaluation

With the intention to evaluate the natural language algorithm we developed, as mentioned in section 5.2, while performing formative usability tests on our application we ask users to create phrases describing outfits, just like we did to our build our development corpus. This phrases, however, were meant to build our evaluation corpus. At total, we collect 50 sentences.

To evaluate our natural language algorithm, without having to interact with our application interface we build an external project with natural language code from our solution. By doing this, we could create a text file with all sentences that constitute our evaluation corpus as well as functions that read this file content and outputs how many phrases were parsed and how many were not, showing the ones that were not able to interpret. From this, we could observe that, from our 50 evaluation sentences, only 29 were correctly interpreted, which means that our natural language algorithm has 58% accuracy rate. Some examples of correctly interpreted evaluation sentences were "Casual floral", "Party outfit with plain sleeveless shirt, red shorts and beige shoes." and "Casual outfit plain black white".

From the failed sentences, we can concluded that about this rate could be improved if we were able to implement our grammar rule where we define OUTFIT non terminal symbol as ATTR_PIECES ATTR_ATTRIBUTES, in other words, the last rule for OUTFIT non terminal symbol in grammar defined in appendix table B.1. One example of failed sentence due to this was "Black suit for formal occasion". We also concluded that in some cases, our algorithm failed due to unrecognised stop words, and that if we had them in account, our accuracy could also be improved. For example, in "weekend casual outfit", if our algorithm considered "weekend" as stop word this phrase would correctly interpreted. Additionally, we concluded that in some cases, our algorithm failed to some kind of attribute regarding clothing item was to right of type/category, and our grammar is not capable of recognizing this. One example of this situation is represented in sentence "party outfit with sleeveless top yellow-orange". In this case, our algorithms is not capable of understanding that yellow-orange must be colour of sleeveless top.

6

Conclusions

The task of select an outfit is very important because of the impression we give to others is influenced by the way we dress. For some people, in the past few years, finding an appropriated outfit has become a more and more difficult task due to the higher number of clothing pieces in their closets.

In 1980 it was invented the capsule wardrobe concept[1], with the purpose of reduce the number of clothing items in people's closet, selecting the ones with higher-quality and more chances to match with other clothes, simplifying the process of choosing an outfit. In the past few years, this concept has been reinvented and widely talked in many fashion online blogs. In most cases, the concept has been applied to three month periods, varying the number of clothes in the capsule.

We started by see what previous work were done and our opportunities to innovate, by doing some research in how to recommend outfits and how we could find outfits through textual descriptions in natural language. Besides that, we also analyse some existent and similar mobile applications and online services. We observed that these areas have been quite explored, however, that we have knowledge of, there is only one online service, *Cladwell Capsules*[2], that supports the creation of wardrobes capsules, without the outfit recommendation option.

Motivated by the capsule wardrobe concept and the weaknesses of previous works, we created an Android application, called Ultimate Capsule, that allows users to store their clothes, create three month capsules, obtain outfits recommendations according to several conditions, view the outfits worn in the past days and simulate the purchase of new clothing items. Besides that, we also support colour and pattern combinations rules and textual descriptions in natural language to find outfits. This application was created by following an iterative and incremental approach, with a SQLite Database, five modules and five algorithms. The five modules corresponded respectively to **Profile**, **Virtual Closet**, **Minimalist Wardrobe Capsule**, **Outfit Recommendation** and **Purchase Simulation**, while the five algorithms corresponded to **`ClothesCoordination`**, **`CapsuleCreation`**, **`FashionNaturalLanguage`**, **`OutfitRecommendation`**, and **`SimulatePurchase`**

To create our application through an iterative and incremental approach we started by defining the most critical tasks and sketching storyboards for them. After this, we created low-fidelity non-functional prototypes and evaluated them. Through users feedback, we were able to improve less intuitive aspects of our application and implemented a first version of our functional prototypes. We then performed a

[1] `http://www.wardrobe.co.uk/bio.html` (last access on 26/05/2017)
[2] `https://cladwell.com/capsules` (last access on 14/05/2017)

formative evaluation, where like low-fidelity non-functional prototypes, were able to check problems in our application, which we improved and corrected. Having our functional prototypes fully implemented, we evaluated it regarding usability and utility, through usability tests and case studies. From usability tests, more precisely collected data and statistical study made on them, we observed that there are improvements that can be made, however all users successfully finished all tasks, were able to recover from errors and overall users satisfaction was relatively high, about 83,54. So, in our opinion, we consider that we can say that we built an intuitive, easy to use and learn solution that also allow users to recover from errors made. Similarly, from case studies, we consider that we can say that our solution is useful for our users, since it helps them to plan unusual combinations, for specific occasion or with a certain piece, plan ahead and also see if how a new clothing item combines with remaining clothes. Additionally, we evaluated our natural language algorithm, `FashionNaturalLanguage`, in terms of accuracy with our evaluation corpus. In other words, we calculated the percentage of how many phrases were correctly processed by it from a set of evaluation phrases. We could concluded that our natural language algorithm has a 58% accuracy rate, however, by observing failed phrases, we realised that with some improvements, more specifically by implementing our grammar rule where we define OUTFIT non terminal symbol as ATTR_PIECES ATTR_ATTRIBUTES, adding more custom stop words and have in consideration that some clothes attributes can also be at right of type/category, this rate could be higher.

With this development approach and evaluation phases, we were able to create an intuitive application, easy to use and learn, that not only allows users to create a virtual closet and organize it into capsules wardrobe, but also eases the process of choosing an outfit and prevents users from impulsive purchases.

6.1 Contributions

Our main contributions with this work were:

- An algorithm, `CapsuleCreation`, capable of automatically create a three month capsule wardrobe;
- An algorithm, `FashionNaturalLanguage`, capable of process and interpret an natural language textual description about an outfit;
- An algorithm, `OutfitRecommendation`, with the ability of automatically recommend an appropriated outfit, from the users' clothes in current capsule or virtual closet, according to given requirements by user, like occasion, outfit pieces and/or dominant colour;
- An algorithm, `SimulatePurchase`, that allows to simulate the acquisition of a new clothing item, showing possible outfits with that piece;
- The creation of an intuitive, easy to learn and use Android mobile application, called Ultimate Capsule, that merges outfit recommendation system and natural language.

6.2 Future Work

We were able to accomplish our goals, however there are some possible improvements that can be done in our work. More specifically,

- Automatically characterize clothing items through their images, obtaining more information about clothes, for example shaping and texture/type of fabric;

- Improve `ClothesCoordination` to have in consideration clothes texture/type of fabrics and shaping;
- Improve `CapsuleCreation` algorithm to allow users to personalize clothes in terms of overall capsule dominant colour or colour pallete;
- Extend `FashionNaturalLanguage` to Portuguese language;
- Improve `FashionNaturalLanguage` by implementing it through more recent technologies, like neural networks;
- Add more information to `OutfitRecommendation` algorithm, for example temperature and weather conditions, with the intention to provide the most appropriated outfit possible.

Bibliography

[1] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[2] Aaron Bangor, Philip T. Kortum, and James T. Miller. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, July 2008. ISSN 1044-7318, 1532-7590. doi: 10.1080/10447310802205776. URL http://www.tandfonline.com/doi/abs/10.1080/10447310802205776.

[3] Moshe Bar, Maital Neta, and Heather Linz. Very first impressions. *Emotion*, 6(2):269–278, 2006. ISSN 1931-1516, 1528-3542. doi: 10.1037/1528-3542.6.2.269. URL http://doi.apa.org/getdoi.cfm?doi=10.1037/1528-3542.6.2.269.

[4] Ching-I. Cheng and Damon Shing-Min Liu. An Intelligent Clothes Search System Based On Fashion Styles. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, pages 1592–1597. IEEE, 2008. URL http://ieeexplore.ieee.org/abstract/document/4620660/.

[5] Ching-I. Cheng and Damon Shing-Min Liu. Mobile Fashion Advisor - A Novel Application in Ubiquitous Society. *Artificial Intelligence*, 2(2), 2008. URL https://pdfs.semanticscholar.org/51da/2b513e07335f789935b0656184bd3596cd38.pdf.

[6] Adam M. Christ. Bridging the Mobile App Gap. *Sigma Journal - Inside the Digital Ecosystem*, 11 (1), October 2011.

[7] Rui G. Crespo. *Processadores de Linguagens: da concepção à implementação*. IST Press, 1 edition, March 2001.

[8] Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan. Hi, Magic Closet, Tell Me What to Wear! In *Proceedings of the 20th ACM International Conference on Multimedia*, pages 619–628. ACM, 2012. URL http://doi.acm.org/10.1145/2393347.2393433.

[9] Manuel J. Fonseca, Pedro Campos, and Daniel Gonçalves. *Introdução ao Design de Interfaces*. FCA - Editora de Informática, Lda., 2 edition, October 2013.

[10] Marsal Gavaldà and Alex Waibel. Growing semantic grammars. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 451–456. Association for Computational Linguistics, 1998.

[11] Neil Howlett, Karen Pine, Ismail Orakçıoğlu, and Ben Fletcher. The influence of clothing on first impressions: Rapid and positive responses to minor changes in male attire. *Journal of Fashion Marketing and Management: An International Journal*, 17(1):38–48, 2013.

[12] Tomoharu Iwata, Shinji Wanatabe, and Hiroshi Sawada. Fashion Coordinates Recommender System Using Photographs From Fashion Magazines. In *IJCAI*, volume 22, page 2262. Citeseer, 2011. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.297.914&rep=rep1&type=pdf.

[13] William Jobe. Native Apps Vs. Mobile Web Apps. *International Journal of Interactive Mobile Technologies (iJIM)*, 7(4):27, October 2013. ISSN 1865-7923. doi: 10.3991/ijim.v7i4.3226. URL http://online-journals.org/i-jim/article/view/3226.

[14] Keisure Kastura, Momoko Kato, and Hiromitsu Simakawa. Recommendation of Fashion Coordinates Considering TPO and Impression. *International Journal of Computing, Communication and Instrumentation Engineering*, 3(2), October 2016. ISSN 23491477. doi: 10.15242/IJCCIE.U0916012. URL http://iieng.org/images/proceedings_pdf/U09160121.pdf.

[15] Si-Jung Kim. Integration of Environmental Contexts and Personal Factors for Coordinating Garments: An Environmental User Interface Paradigm to Enrich User Interactions. In *Proceedings of the 47th Annual Southeast Regional Conference*, page 59. ACM, 2009. URL http://dl.acm.org/citation.cfm?id=1566524.

[16] Philip T. Kortum and Aaron Bangor. Usability Ratings for Everyday Products Measured With the System Usability Scale. *International Journal of Human-Computer Interaction*, 29(2):67–76, January 2013. ISSN 1044-7318, 1532-7590. doi: 10.1080/10447318.2012.681221. URL http://www.tandfonline.com/doi/abs/10.1080/10447318.2012.681221.

[17] Hanbit Lee and Sang-goo Lee. Style Recommendation for Fashion Items using Heterogeneous Information Network. In *RecSys Posters*, 2015. URL https://pdfs.semanticscholar.org/1361/e67dfa07e04e88970d8783b0815849a9064d.pdf.

[18] Hugo Liu and Push Singh. ConceptNet - A Practical Commonsense Reasoning Tool-Kit. *BT technology journal*, 22(4):211–226, 2004. URL http://www.springerlink.com/index/T551Q7U40671780V.pdf.

[19] Satoshi Nagao, Shin Takahashi, and Jiro Tanaka. Mirror Appliance: Recommendation of Clothes Coordination In Daily Life. *Human Factors in Telecommunication (HFT2008)*, 8:367–374, 2008. URL http://www.iplab.cs.tsukuba.ac.jp/paper/international/nagao_hft2008.pdf.

[20] Manuel J. Queirós. *Android Desenvolvimento de Aplicações com Android Studio*. FCA - Editora de Informática, Lda., 1 edition, February 2016.

[21] Gauri Rao, Chanchal Agarwal, Snehal Chaudhry, Nikita Kulkarni, and Dr SH Patil. Natural Language Query Processing Using Semantic Grammar. *International journal on computer science and engineering*, 2(2):219–223, 2010. URL http://www.bvucoepune.edu.in/pdf's/Publications_2004-2011/Publications_9-10/International_journal9-10/IJ17-9-10.pdf.

[22] Edward Shen, Henry Lieberman, and Francis Lam. What am I gonna wear? Scenario-Oriented Recommendation. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 365–368. ACM, 2007. URL http://dl.acm.org/citation.cfm?id=1216368.

[23] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open Mind Common Sense: Knowledge acquisition from the general public. In *On the move to meaningful*

internet systems 2002: CoopIS, DOA, and ODBASE, pages 1223–1237. Springer, 2002. URL https://link.springer.com/chapter/10.1007/3-540-36124-3_77.

[24] Manasi Vartak and Samuel Madden. CHIC: A Combination-Based Recommendation System. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 981–984. ACM, 2013. URL http://dl.acm.org/citation.cfm?id=2465270.

[25] Elise Whitley and Jonathan Ball. Statistics review 1: Presenting and summarising data. *Critical Care*, 6(1):66, 2001.

[26] Ying Zhao and Kenji Araki. KA: What to Wear In Different Situations? A Content-Based Recommendation System for Fashion Coordination. In *Proc. of the Japanese Forum on Information Technology (FIT2011)*, pages 619–628, 2011. URL http://arakilab.media.eng.hokudai.ac.jp/~araki/2011(e)/2011-D-3.pdf.

[27] Susana Zoghbi, Geert Heyman, Juan Carlos Gomez, and Marie-Francine Moens. Fashion Meets Computer Vision and NLP at e-Commerce Search. *International Journal of Computer and Electrical Engineering*, 8(1):31–43, 2016. ISSN 17938163. doi: 10.17706/IJCEE.2016.8.1.31-43. URL http://www.ijcee.org/index.php?m=content&c=index&a=show&catid=81&id=1034.

A

Storyboards

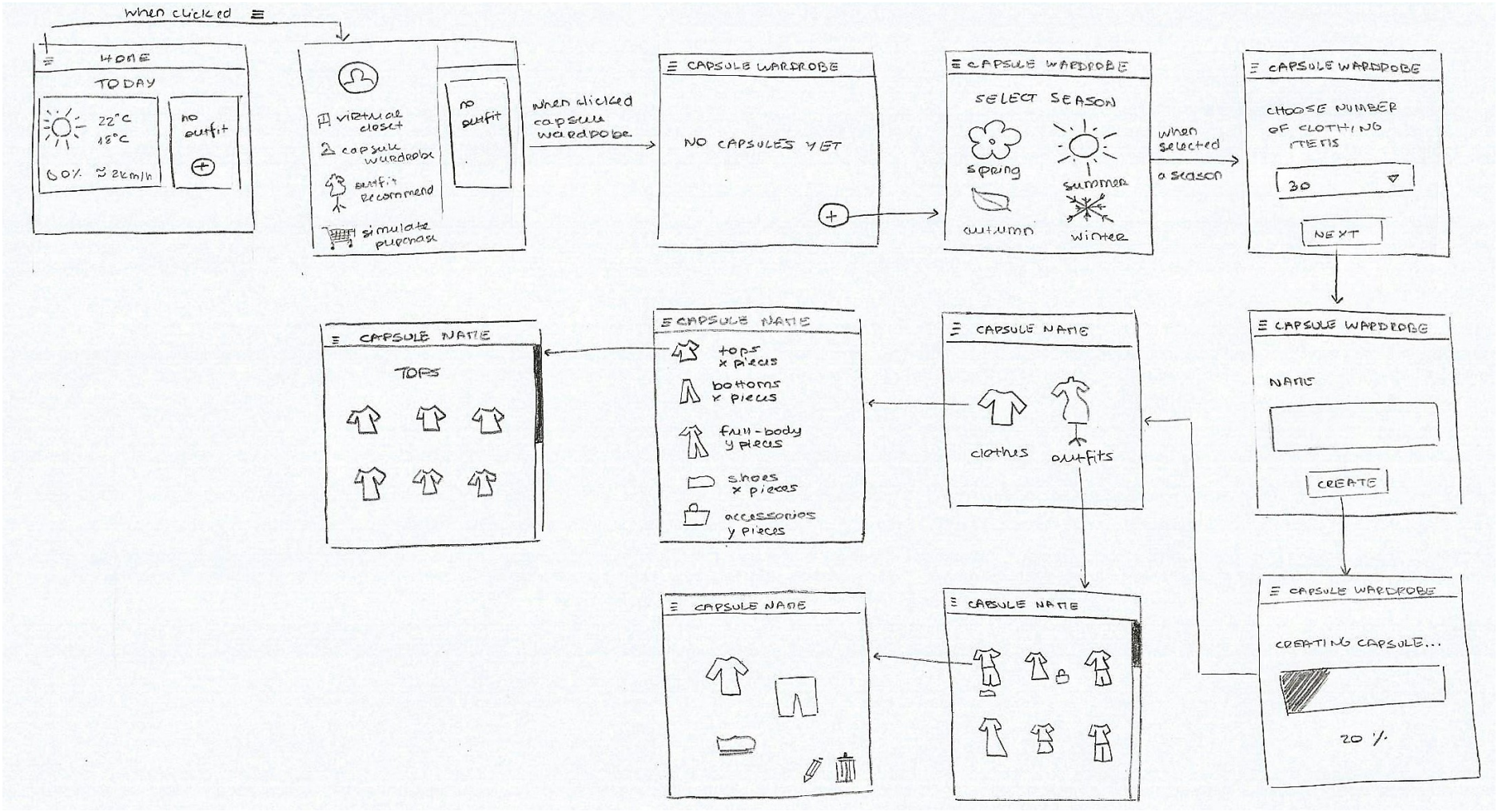


Figure A.1: Storyboard for create a new capsule wardrobe

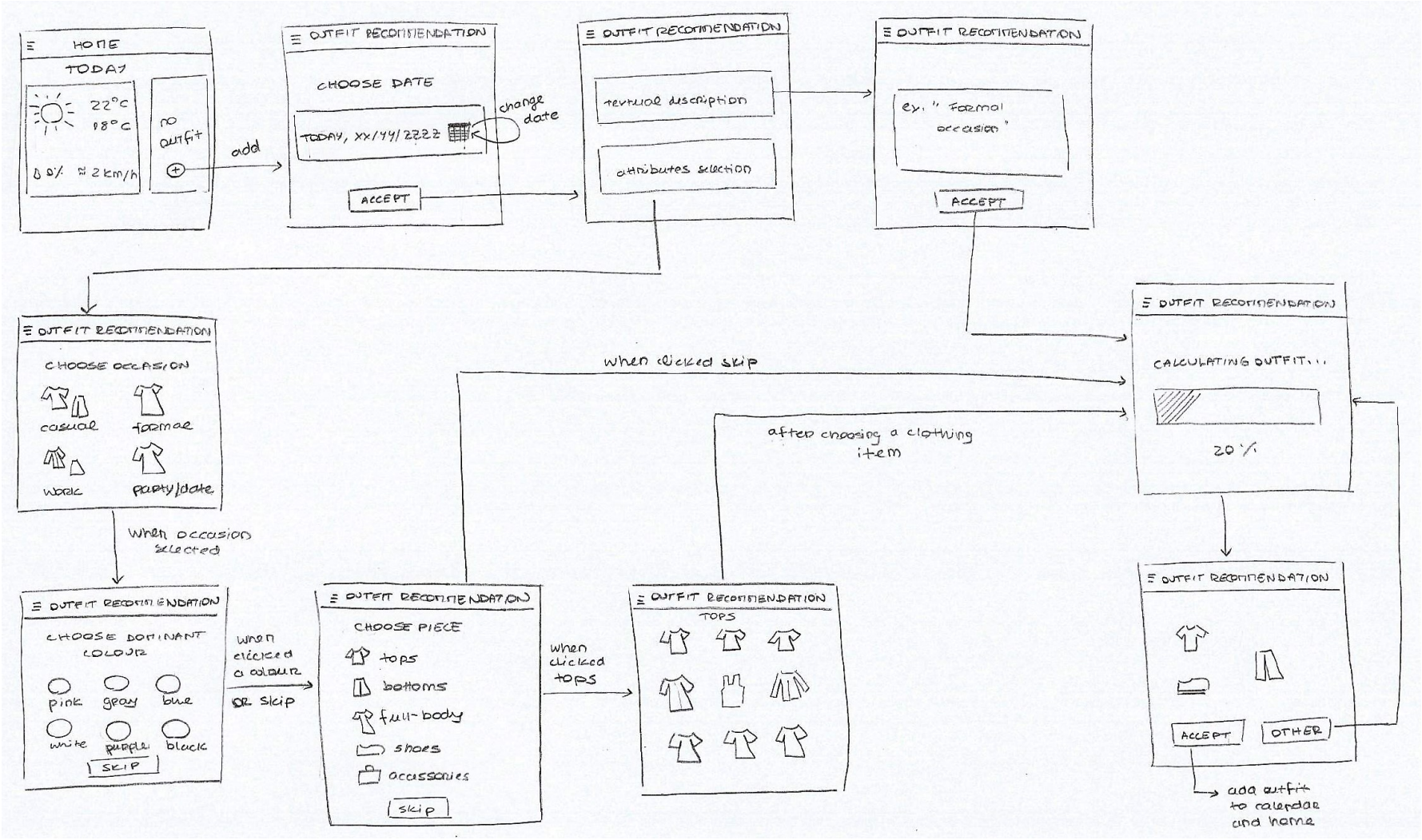


Figure A.2: Storyboard for obtain an outfit recommendation

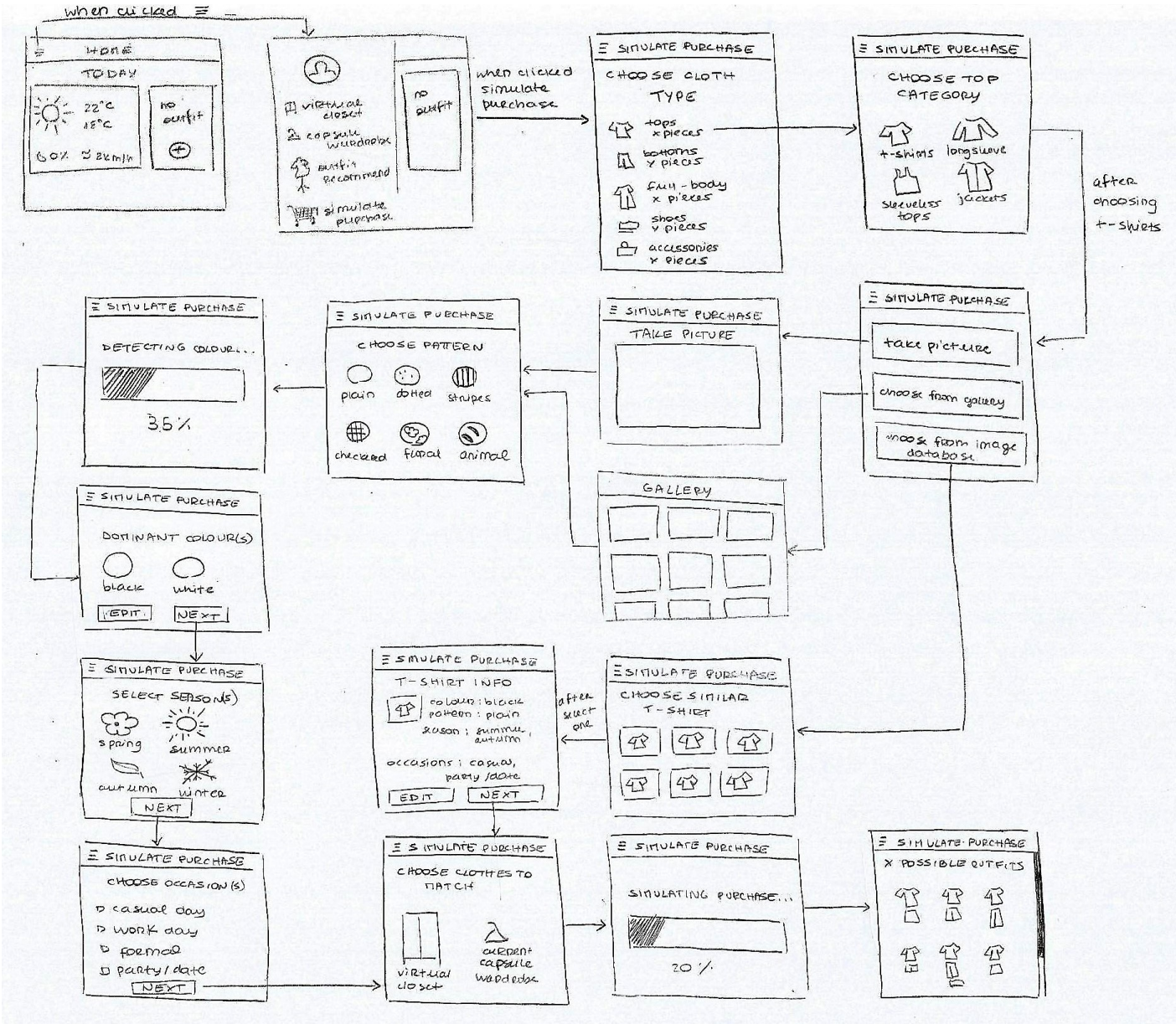


Figure A.3: Storyboard for simulate the acquisition of a new clothing item

B

Semantic Grammar

Table B.1: Semantic grammar defined by us concerning outfit description.

OUTFIT	→	ATTR_ATTRIBUTES
	→	ATTR_ATTRIBUTES with ATTR_PIECES
	→	ATTR_PIECES
	→	ATTR_PIECES ATTR_ATTRIBUTES
ATTR_PIECES	→	ATTR_PIECE
	→	ATTR_PIECES with ATTR_PIECES
	→	ATTR_PIECES and ATTR_PIECES
	→	ATTR_PIECES ATTR_PIECES
ATTR_PIECE	→	ATTR_ATTRIBUTES TYPE
	→	ATTR_ATTRIBUTES CATEGORY
ATTR_ATTRIBUTES	→	ATTR_COLOUR ATTR_OCCASION ATTR_PATTERN
	→	ATTR_COLOUR ATTR_PATTERN ATTR_OCCASION
	→	ATTR_OCCASION ATTR_COLOUR ATTR_PATTERN
	→	ATTR_OCCASION ATTR_PATTERN ATTR_COLOUR
	→	ATTR_PATTERN ATTR_COLOUR ATTR_OCCASION
	→	ATTR_PATTERN ATTR_OCCASION ATTR_COLOUR

Continued on next page

Table B.1 – *Continued from previous page.*

ATTR_COLOUR	→	ε
	→	COLOUR
	→	COLOUR ATTR_COLOUR
	→	COLOUR and _COLOUR
	→	COLOUR or ATTR_COLOUR
ATTR_OCCASION	→	ε
	→	OCCASION
	→	OCCASION ATTR_OCCASION
	→	OCCASION and _OCCASION
	→	OCCASION or ATTR_OCCASION
ATTR_PATTERN	→	ε
	→	PATTERN
	→	PATTERN ATTR_PATTERN
	→	PATTERN and _PATTERN
	→	PATTERN or ATTR_PATTERN

C

Usability Time Results Collected

Table C.1: Time, in seconds, spent performing each task by each user.

Users	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
1	38	47	15	70	6	72	50
2	40	67	20	80	7	46	60
3	32	77	23	57	15	55	50
4	30	90	29	55	7	46	96
5	28	42	24	70	11	49	48
6	50	81	44	59	10	51	63
7	20	117	27	54	45	69	70
8	51	94	26	92	6	100	90
9	28	153	32	62	10	76	79
10	35	64	18	55	11	49	88
11	39	88	21	62	14	50	87
12	26	68	23	56	8	53	62
13	27	72	18	65	8	48	51

Continued on next page

Table C.1 – *Continued from previous page.*

Users	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
14	24	94	28	74	7	52	74
15	50	86	26	76	8	56	55
16	22	90	30	64	12	75	51
17	33	82	29	63	10	95	76
18	23	78	26	59	11	85	79
19	43	56	32	62	8	68	49
20	42	100	31	57	9	51	52