



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



DESIGN AND DEVELOPMENT OF WIDGETS FOR A CORPORATE SECURITY APPLICATION

LAURA SALES I MARTÍNEZ

Thesis supervisor: JORGE PALACIO FRANCO (ENTRUST EU SLU)

Tutor: MARC ALIER FORMENT (Department of Service and Information System Engineering)

Degree: Bachelor Degree in Informatics Engineering (Software Engineering)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

28/06/2023

RESUM

Aquest projecte es lliura com a Treball Final del Grau d'Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona. L'objectiu és posar en pràctica els coneixements adquirits durant l'especialitat d'enginyeria del Software.

El projecte consisteix a dissenyar i desenvolupar widgets per a una aplicació mòbil corporativa de seguretat que permet als usuaris interaccionar amb una de les funcionalitats principals de la aplicació, sempre mantenint la perspectiva de la seguretat i la usabilitat.

RESUMEN

Este proyecto se entrega como Trabajo Final del Grado de Ingeniería Informática de la Facultad de Informática de Barcelona. El objetivo es poner en práctica los conocimientos adquiridos en la especialidad de ingeniería del Software.

El proyecto consiste en diseñar y desarrollar widgets para una aplicación móvil corporativa de seguridad que permite a los usuarios interaccionar con una de las principales funcionalidades de la aplicación, siempre manteniendo la perspectiva de la seguridad y la usabilidad.

ABSTRACT

This project is delivered as the Bachelor Thesis of the Informatics Engineering Degree of the Barcelona Faculty of Computer Science. The objective is to put into practice the knowledge acquired during the Software engineering specialty.

The project consists of designing and developing widgets for a corporate security mobile application that allows users to interact with one of the main functionalities of the application, always maintaining the perspective of security and usability.

Contents

1. Context and Scope	7
1.1. Introduction and contextualization	7
1.1.1. Context	7
1.1.2. Concepts	8
1.1.3. Problem to be resolved	9
1.1.4. Stakeholders	9
1.2. Justification	10
1.3. Scope	11
1.3.1. Objectives and sub-objectives	11
1.3.2. Requirements	11
1.3.3. Potential obstacles and risks	12
1.4. Methodology and rigor	13
1.4.1. Methodology	13
1.4.2. Monitoring tools and validation	15
2. Project Planning	16
2.1. Description of tasks	16
2.1.1. Task definition	16
2.1.2. Summary of tasks	19
2.1.3. Resources	20
2.1.3.1. Human resources	20
2.1.3.2. Material resources	21
2.2. Estimates and the Gantt	23
2.3. Project final status	24
2.4. Risk management: alternative plans and obstacles	25
3. Budget	27
3.1. Staff costs per activity	27
3.2. Generic costs	29
3.3. Budget deviations	30
3.3.1. Contingency costs	30
3.3.2. Incidental costs	31
3.4. Final budget	31
3.5. Management control	32
4. Sustainability	33
4.1. Self-assessment	33
4.2. Economic dimension	33
4.3. Environmental dimension	34
4.4. Social dimension	34

5. Specification	36
5.1. Conceptual Data Model	36
5.2. Use Case Diagram	36
5.3. Use Case Descriptions	37
6. Design	43
6.1. Architecture	43
6.2. Design patterns	44
6.3. User interface design	45
6.4. Flowchart	50
7. Development	51
7.1. Technologies	51
7.2. Android	54
7.3. iOS	63
8. Testing	67
8.1. Testing widgets	67
9. Experimentation and analysis	68
9.1. Usability testing	69
9.2. Widget usability test	71
9.3. Results	79
10. Legal and regulatory compliance	82
10.1. Identification of applicable laws and regulations	82
10.2. Academic regulations	83
12. Conclusions	85
12.1. Integration of knowledge	86
12.2. Future work	88

List of Figures

1.1. Entrust Identity	6
1.2. Trello board created for the project	12
1.3. Labels with each issue type	12
2.1. Gantt char	21
5.1 Conceptual data model	35
5.2 Use case diagram	36
6.1 Architecture design	43
6.2 Color palette	45
6.3 IOS small widgets	47
6.4 iOS medium widget	47
6.5 Android widget	47
6.6 Instructions view	48
6.7 Widget settings	48
6.8 Flowchart	49
7.1 Technology stack	52
7.2 Communication between React Native and native	53
7.3 Android app widget processing	54
7.4 New Android Component dialog	54
7.5 Comparison Android weather app widget API 27 vs API 31	57
7.6 Adding a Widget Extension	62
7.7 Placeholder widget	64
9.1 Steps to usability testing	69
9.2 Context screen	71
9.3 Question to know if participants have used the app	71
9.4 Mission 1 Android	72
9.5 Screen navigation flow Mission 1 Android	73
9.6 Mission 2 Android	73

9.7 Screen navigation flow Mission 2	74
9.8 Mission 3 Android	74
9.9 Screen navigation flow Mission 3 Android	74
9.10 Mission 1 iOS	75
9.11 Mission 2 iOS	75
9.12 Screen navigation flow Mission 1 iOS	76
9.13 Screen navigation flow Mission 2 iOS	77
9.14 Qualitative data questions	78
9.15 Performance indicators Mission 1 Android	79
9.16 Performance indicators Mission 2 Android	80
9.17. Performance indicators Mission 3 Android	80
9.18. Performance indicators Mission 1 iOS	80
9.19. Performance indicators Mission 2 iOS	80

List of Tables

2.1. Summary of tasks	20
3.1. Staff costs for each task	28
3.2. Amortization costs of hardware resources	29
3.3. Generic cost of the project	30
3.4. Incidental costs of the project	31
3.5. Total cost of the project	31
5.1. Use Case 1. Configure a widget	38
5.2. Use Case 2. Show usage instructions in the widget	38
5.3. Use Case 3. Reconfigure a widget	39
5.4. Use Case 4. View token information	39
5.5. Use Case 5. Copy the OTP of a token.	40
5.6. Use Case 6. Resize the widget	40
5.7. Use Case 7. Open the app from the widget	41
5.8. Use Case 8. View the remaining time for the regeneration of the OTPs	41
5.9. Use Case 9. Update the widget content periodically	42
5.10. Use Case 10. Auto refresh token information	42
7.1. Comparison between the number of cells and minWidth and minHeight	57

1. Context and Scope

1.1. Introduction and contextualization

The concept of widget has existed for centuries, although its meaning has evolved over time. In its original context, a widget referred to a mechanical device or tool used in manufacturing or industry. Nowadays, a widget is an essential component of modern graphical user interface (GUI), providing an intuitive and efficient way for users to interact with software applications and web pages [1].

Widgets are designed to provide quick access to frequently used features or to display important information to the user. They can be found on various platforms, including desktop and mobile operating systems, web pages, software applications, and even smartwatches. The use of widgets is widespread, since they are used in a variety of fields, from weather apps to news apps to social media platforms.

Additionally, widgets are available in different shapes and sizes and can be customized to fit the specific needs and preferences of individual users. Furthermore, widgets improve the user experience, make complex tasks easier to perform and help users stay informed and up-to-date on important information without having to manually search for it. Due to all this, widgets achieve to make applications and web pages more user-friendly.

1.1.1. Context

This is a Bachelor Thesis of the Informatics Engineering Bachelor Degree, specialization in Software Engineering, done in the Facultat d'Informàtica de Barcelona of the Universitat Politècnica de Catalunya, tutored by Marc Alier Forment and externally mentored by Jorge Palacio Franco, the Software Development Manager of Entrust's mobile team.

Entrust is a global technology company that provides secure digital identity and access management solutions [2]. It was founded in 1969 and is headquartered in Minneapolis, Minnesota, USA. The company offers a wide range of products and services, including authentication, authorization, encryption and public key infrastructure (PKI). These solutions are used by users and organizations to manage access to sensitive data, protect their digital assets and secure online transactions. Entrust serves clients in a variety of industries, including finance, government, healthcare and telecommunications.

During the last year of my degree, I was very lucky to have the opportunity to do academic internship formation at Entrust, where I was able to apply theoretical concepts to real-world scenarios and learn a lot from professionals in my field. Throughout my time at the company I worked in the mobile department, specifically on the Entrust Identity mobile application, which is a platform for providing strong identity credentials to both employee and consumer users. It offers authentication, advanced password reset and transaction verification capabilities, and allows users to create identities and activate unique one-time passcode soft token applications for different organizations [3]. Along with this app, Entrust aims to combine security with usability for millions of customers across the globe.

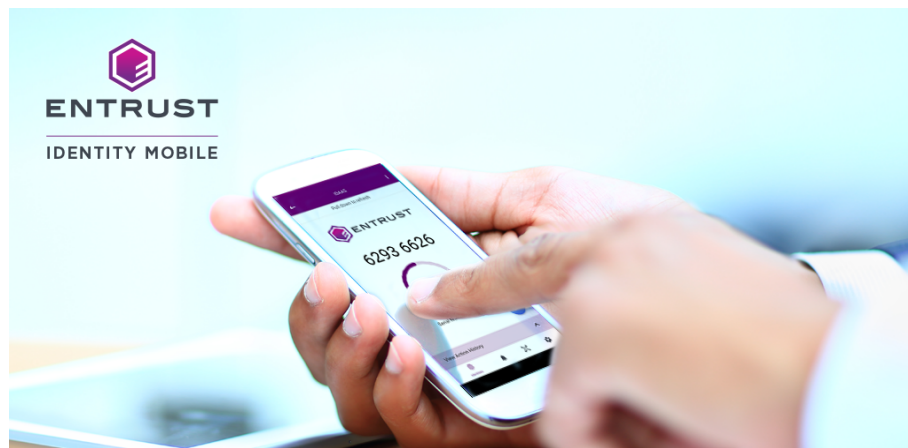


Figure 1.1. Entrust Identity. *Source: [4].*

1.1.2. Concepts

One of my goals is that this project is understandable by anyone regardless of their technical knowledge, and therefore I am going to explain some of the fundamental concepts in order to understand the project correctly.

- **Best practices:** a working method or set of working methods that is officially accepted as being the best to use in a particular business or industry [5].
- **Contingency plan:** a plan that is made for dealing with an emergency, or with something that might possibly happen and cause problems in the future [6].
- **Graphical user interface (GUI):** a way of arranging information on a computer screen that is easy to understand and use because it uses icons, menus, and a mouse rather than only text [7].
- **Kanban:** a system of communication between workers during a manufacturing process, in which they order the materials and parts they need using cards [8].
- **One-time passcode (OTP):** a one-time password is an automatically generated numeric or alphanumeric string of characters that authenticates a user for a single transaction or login session [9].

- **Public key infrastructure (PKI):** a public key infrastructure is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption [10].
- **Soft token:** a software token is a piece of a two-factor authentication security device that may be used to authorize the use of computer services [11].
- **Swift:** a high-level general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. and the open-source community [12].
- **Version control:** a class of systems responsible for managing changes to computer programs, documents, large web sites, or other collections of information [13].
- **Widget:** a piece of software that is used on a page of a website to give the user changing information of a particular type in a small area of the computer screen [14].
- **Workflow:** the way that a particular type of work is organized, or the order of the stages in a particular work process [15].

1.1.3. Problem to be resolved

The Entrust Identity mobile application has the widest breadth of mobile authentication options to enable the modern employee and many more functionalities mentioned before in Section 1.1.1. However, the application lacks the ability to offer its users the possibility to interact with widgets and, consequently, benefiting from all its advantages. Furthermore, it is worth noting the fact that some of the consumers of the mobile application have requested the incorporation of this functionality in the app, and one of our biggest goals is to ensure the best possible user experience. Therefore, the problem to be resolved is the inclusion of different types of widgets in the Entrust Identity mobile application to be able to offer users the possibility of using widgets on their devices.

In addition, it must be taken into account that since it is a mobile application focused on the field of security, another fundamental problem to solve in this project will be to provide widgets that strictly comply with the security perspective of the app, which can lead to certain limitations or restrictions.

1.1.4. Stakeholders

The project has many involved parties, which can be grouped into two different groups depending on the interaction and benefits they have with the project itself.

On the one hand, the stakeholders that have direct interaction with the development of the project are the tutor, Marc Alíer Forment, the director, Jorge Palacio Franco, the GEP tutor, Joan Sardà Ferrer, and the researcher. Both the tutor and the director have the role of leading and guiding the researcher

for the correct development of the project. In addition, the GEP tutor is in charge of helping the researcher to correctly manage the project during the first month of the project. Finally, the researcher, myself, is responsible for defining, planning, developing and documenting the project, as well as experimenting, analyzing and drawing conclusions. Therefore, the researcher will have to carry out different roles such as project manager, UI/UX designer, developer, quality assurance tester and technical writer.

On the other hand, the stakeholders that do not interact with the project development but receive direct benefits from it, since they are the ones who will use the final product of the project on a day-to-day basis, can be divided into two more groups: the employees of the Entrust company and the customers of the Entrust Identity mobile application. All these final users will be able to use the widgets of the mobile application to, for example in the case of employees, access internal company services or, in the case of consumers, authenticate themselves in other portals, authorize and verify online transactions, among others.

1.2. Justification

Widgets are an important component of mobile application design because they provide users with an easy and efficient way to interact with the app. By placing frequently used features and information on the home screen, widgets can reduce the amount of time and effort required for users to perform common tasks. Actually, it is becoming increasingly common to use widgets, so most mobile apps decide to offer widgets to their users because they provide several benefits, such as improving the user experience and increasing engagement and app usage.

However, after focusing and doing some research on the current market for mobile security apps that offer two-factor authentication (2FA) with automatically generated unique codes, I have realized that even though there are an infinite number of applications available on the market with similar functionalities only a very small number include widgets. In addition, it should be noted that the most popular mobile applications in this sector, such as Google Authenticator or Microsoft Authenticator, do not offer widgets to their users and, therefore, offering them in the Entrust Identity mobile application would provide a competitive advantage in the mobile app market.

For all these reasons, the project director proposed to me to carry out my degree final project on the development of a new extension of the Entrust Identity mobile application that allows users to interact with various types of widgets with different functionalities of the app. I delightedly accepted the proposal of the project theme since widgets are something I use on a day-to-day basis and I personally

find them very attractive and useful. At that moment, it was when the title of my degree final project was born, the “Design and Development of Widgets for a Corporate Security Application”.

1.3. Scope

1.3.1. Objectives and sub-objectives

The main objective of the project is to develop a new functionality for the Entrust Identity mobile application which consists of offering users different types of widgets related to some of the most used features of the mobile app. Once the main objective of the project have been established, it is also important to take into account the following secondary objectives:

- Provide users with an easy and efficient way to interact with the mobile application.
- Reduce the amount of time and effort required for users to perform common tasks.
- Improve the user experience of the mobile application.
- Increase engagement and usage of the mobile application.
- Provide a competitive advantage in the mobile application market.

1.3.2. Requirements

The following requirements are needed to ensure the quality of the degree final project.

- Project deliverables must meet all delivery deadlines.
- Documentation must be adequate and understandable.
- Good programming practices must be used, with a readable style and least possible complexity.
- Widgets must be available for Android and iOS.
- Widgets must display important information to the user.
- Widgets must be attractive, efficient, intuitive, useful, user-friendly.
- Users must be able to easily add widgets to their home screen.
- The type of each widget must be appropriate for its purpose.
- The content of each widget must be adapted to different sizes.
- Widgets must be customizable to fit the specific needs and preferences of users.
- Widgets must strictly comply with the security perspective of the mobile application.
- Widgets must look and feel like the company's mobile application.
- Users must be able to choose the language of the widget content, and change it as many times as they want.
- Visually impaired users must be able to use the widgets without any problem.

- There must be a support section with the main information sections, including one for frequently asked questions and one for customer service.

1.3.3. Potential obstacles and risks

Throughout the development of the thesis, some potential obstacles and risks may arise that I will have to face.

- **Bugs.** Bugs can cause issues such as crashing, incorrect results, or unexpected behavior. Also, some of the third-party libraries used in the project could contain bugs in some functions that may affect my code.
- **Deadline of the project.** There is a deadline for the delivery of the project that must be accomplished. Therefore, having a good plan is a must in order to meet the specified deadlines to be able to finish the project in time.
- **Impossibility of meeting with the tutor and director in person.** In order to advance properly in the project I have to hold meetings with the tutor and director to see if the work done is correct, and to know what the next steps to follow are. This task can be hampered, for example, by a virus (COVID-19).
- **Inexperience in the field.** A very common and possible scenario is that I may have to work with technologies that I have no prior knowledge of or experience with, so I will have to dedicate time learning the basics of these technologies.
- **Personal challenges.** Personal challenges, such as illness, family problems, or mental health issues, can affect my ability to work on my thesis. It is important to seek support and assistance when needed and prioritize self-care.
- **Technical difficulties.** It is possible to encounter technical difficulties, such as computer crashes, loss of data, software issues, or have a bad network connection. Therefore, it is important to back up the work done regularly to prevent future data loss, install an antivirus to avoid future viruses, and invest in a good network connection to prevent future connectivity issues.

In conclusion, it is extremely important to anticipate potential obstacles and risks and have contingency plans in place to address them.

1.4. Methodology and rigor

Correctly defining the work methodology to be followed, the monitoring tools and the validation methods is decisive for the achievement of the main objectives and the proper development of the thesis.

1.4.1. Methodology

The chosen development method is based on Agile. The main idea is continuous iteration of development and testing throughout the software development life cycle of the project [16]. Unlike a waterfall method, in Agile the development of a project is divided into sprints. A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. I have opted for this methodology mainly because it is the methodology used in the company but also because the Agile methodology is very suitable for projects that require flexibility and adaptability, such as my final degree project.

Regarding the planning and monitoring of project tasks, I will use the Kanban methodology because it is a project management methodology that originated in Japan and focuses on visualizing the workflow, limiting work in progress, and continuously improving the process [17]. In Kanban, work items are represented by visual cards on a board that shows the status of each task in the workflow, from “to do” to “done”. The visual representation of the workflow helps team members to quickly identify bottlenecks, prioritize work, and make decisions about what to work on next. I have decided to use the website application called Trello to apply this methodology. Within the application, I created a board for the project and started creating cards and moving them between the following 4 different columns of the board:

- **To do:** Tasks that have been defined but have not been started yet.
- **In progress:** Tasks that are being developed.
- **In test:** Tasks that have been developed but still not tested for the correct functioning.
- **Complete:** Finished and tested tasks.

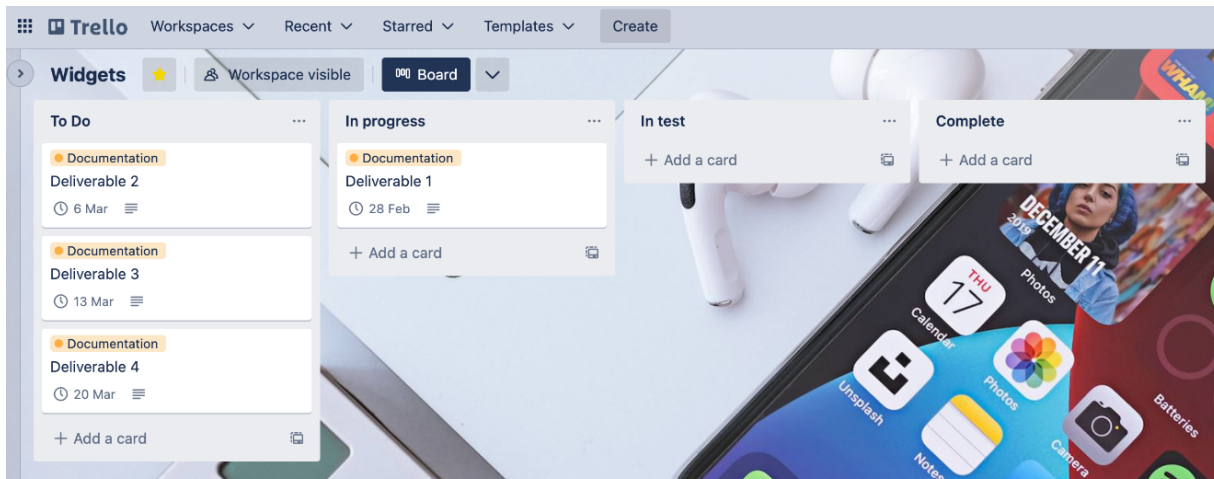


Figure 1.2. Trello board created for the project. *Source: Own compilation.*

It is worth mentioning that in some tasks I will not have to program or test anything, so these will skip the testing step. Furthermore, for each card I will define a brief description of the task and its acceptance criteria, add labels with the issue type and the due date to be well organized, assign the task to someone, create a checklist to list some subtasks, attach files and, finally, there is a field to write some comments in case it is needed.

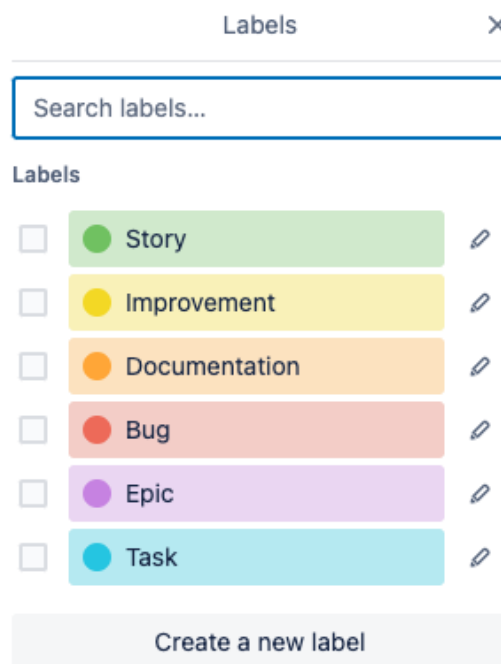


Figure 1.3. Labels with each issue type. *Source: Own compilation.*

1.4.2. Monitoring tools and validation

Repository

I will use a Bitbucket repository as a tool for version control because it is what I use in the company. BitBucket is a cloud-based service that helps developers store and manage their code, as well as track and control the changes to their code [18]. BitBucket provides a cloud-based Git repository hosting service. Its interface is user-friendly enough so even novice coders can take advantage of Git. All the code related to the widgets extension of the mobile app that I will develop in the repository will be located in a new sub-branch of *develop* called *widgets*. Therefore, different children (sub-branches) will be born from this branch for each epic, task, etc., that needs to be developed or tested.

Branch naming is a relevant aspect of software development because it helps team members understand the purpose and status of a particular branch. Regarding the naming conventions, I have decided to use the same as in the company which consists of using prefixes to indicate the type of branch (e.g. task/, bugfix/), followed by a brief description of the changes. Moreover, I will apply best practices such as being descriptive so that the branch name conveys its purpose, keeping branch names short, avoiding special characters, and deleting branches once they have served its purpose in order to keep the repository clean [19].

Meetings

A biweekly meeting will be scheduled with the director of the project, and occasional meetings with the tutor will be arranged, both with the intention of discussing the project status, any doubts, problems or blockages that may arise, and checking the tasks to be accomplished the following weeks. However, in case an urgent meeting is needed for any reasonable cause, I will notify the tutor or director of the project, depending on the issue, to schedule an extraordinary meeting as soon as possible.

2. Project Planning

2.1. Description of tasks

This project will have a duration of approximately 540 hours, distributed in 129 days starting on February 14, 2023 until June 22, 2023. It should be noted that of the total of 129 days, there will only be 90 working days. Regarding the number of hours that I plan to work each day, it will be 6 hours/day and, therefore, I will dedicate 30 hours per week to the project. The date for the oral defense of the project is not yet scheduled, but it is known that the TFG presentation period will take place the last week of June 2023, from 26 to 30. Therefore, the project is expected to be finalized at least the week before the date of the oral presentation.

2.1.1. Task definition

In this section, I am going to define all the tasks that will be carried out throughout the project, including a brief explanation, the duration in hours and the dependencies with other tasks. Table 2.1 summarizes all the information presented in this section and Figure 2.1 shows the project schedule through a Gantt chart.

Project management

Project management is probably one of the most important task groups in any project. It defines the scope and objectives of the project, the temporary planning of the tasks to be carried out, the calculation of the costs that it will entail and a study on its relationship with sustainability, as well as the planning and monitoring of tasks and meetings.

- **[T1.1] Context and scope (15h).** Definition of the scope of the project in the context of its study, indicating the general objective of the TFG, the context, the reason for selecting the subject area, how the project will be developed and using which means.
- **[T1.2] Project planning (10h).** Planning of the entire execution of the TFG, providing a description of the project phases, and the tasks, resources and requirements associated with each one.
- **[T1.3] Budget and sustainability (10h).** Elaboration of the project budget and analysis of the sustainability of the project.
- **[T1.4] Integration in final document (5h).** Final written document summarizing the project, considering the feedback received.
- **[T1.5] Meetings (20h).** Face-to-face and online meetings with the tutor or the director of the project where we will discuss the status and the following tasks to carry out, as well as any

doubts, problems or blockages that may arise. Extra time has been added due to possible extraordinary meetings.

Research

Before starting the development phase of the project, it is essential to do a thorough research on the widgets environment. This task consists of researching the different types of widgets that exist, their main objectives and characteristics, the design principles that must be taken into account, how they work and what technology is involved, in addition to investigating how they are developed on both Android and iOS operating systems, and the constraints and limitations they have, etc. In addition, some research regarding usability testing methods will be needed in order to carry out the experimentation and analysis phase. This task is quite large and therefore it will take approximately 90 hours.

Project development

The development phase of the project depends on the research task, but it must be taken into account that throughout the project research will continue on different topics, both related to widgets and other concepts.

- **[T3.1] Define widgets (10h).** Before starting to work with the widgets, I will have to think and define the different possibilities of widgets that could be integrated into the mobile application.
- **[T3.2] Design widgets (20h).** Design mockups and prototypes of the widget's interface, including the visual appearance (UI) and the functionality it will provide (UX).
- **[T3.3] Develop widgets (165h).** Develop widgets on Android and iOS. This task has two subtasks for each operating system, the Android one has a duration of 60 hours and the iOS one of 105 hours.
- **[T3.4] Test widgets (30h).** Thoroughly test the developed widgets to ensure that it works correctly and does not cause errors in the mobile application. This task has two subtasks for each operating system, each with a duration of 15 hours.
- **[T3.5] Optimize widgets (5h).** It is important to optimize widgets to ensure they load quickly and don't affect the overall performance of the mobile application.

Experimentation and analysis

In the experimentation and analysis phase, a usability test will be carried out with the aim of evaluating how easy and efficient it is to use the developed widgets, identifying areas in which users

may have difficulties or encounter frustration, and discover possible improvements to create a better user experience. This phase depends on the development phase, which means that the first task of this phase will be started when the last task of the development phase finishes.

- **[T4.1] Define the purpose and goals (3h).** Determine the purpose of the usability test and what I hope to achieve from it.
- **[T4.2] Select appropriate testing methods (5h).** Determine the testing methods that I will use, such as qualitative or quantitative, moderated or unmoderated, remote or in-person.
- **[T4.3] Identify target audience (2h).** Determine the specific group of users that I want to test widgets with, considering factors such as age, gender, education, and experience.
- **[T4.4] Develop test scenarios and tasks (20h).** Create test scenarios and tasks that will help me measure how users interact with widgets, which will be realistic and relevant.
- **[T4.5] Recruit participants (5h).** Find participants who fit the profile of my target audience and are willing to take part in the usability test.
- **[T4.6] Conduct the usability test (15h).** Manage test scenarios and tasks to participants while recording their interactions and collecting feedback.
- **[T4.7] Conclusions (10h).** Analyze the data collected from the usability test, including quantitative data such as task completion rates and qualitative data such as user feedback, and finally draw conclusions.

Project documentation

Documentation is a task that must be done throughout the project because it is important to document each task during its course to avoid having work overload during the last weeks before delivering the complete memory.

- **[T5.1] Collect all the information obtained (10h).** Collect all the information obtained throughout the project to be able to write it afterwards in the final document.
- **[T5.2] Write the documentation (60h).** This task consists of, as its name indicates, writing all the documentation of the memory. It depends on the task described above, the collection of all the information obtained throughout the project.

Bachelor thesis defense preparation

Finally, the last task of the project will be the preparation of the oral defense of the project. In order to carry it out successfully, I will consider the most important parts of the thesis that must be presented and also possible questions or doubts that may occur to the members of the tribunal. This task depends on the project documentation and is very important so therefore it will take approximately 30 hours.

2.1.2. Summary of tasks

The following table summarizes the tasks explained above that will be carried out in the project, showing their unique identifier (ID), a descriptive name, the number of hours dedicated to each one, the dependencies between them and the resources needed for each one.

ID	Task	Time (h)	Dependencies	Resources
T1	Project management	60		
T1.1	Context and scope	15		R, D, GEPT, PC, GD
T1.2	Project planning	10		R, GEPT, PC, GD, GP
T1.3	Budget and sustainability	10		R, GEPT, PC, GD
T1.4	Integration in final document	5	T1.1, T1.2, T1.3	R, PC, GD
T1.5	Meetings	20		R, T, D, GC, GM, MT
T2	Research	90		R, PC
T3	Project development	230	T2	
T3.1	Define widgets	10		R, D
T3.2	Design widgets	20	T3.1	R, PC, F
T3.3	Develop widgets	165	T3.2	R, PC, B, S, P, IDE
T3.3.1	Develop Android widgets	60		
T3.3.2	Develop iOS widgets	105		
T3.4	Test widgets	30	T3.3	R, PC, M, E, P, IDE
T3.4.1	Test Android widgets	15	T3.3.1	
T3.4.2	Test iOS widgets	15	T3.3.2	
T3.5	Optimize widgets	5	T3.4	R, PC, P, IDE
T4	Experimentation and analysis	60	T3	
T4.1	Define the purpose and goals	3		R
T4.2	Select appropriate testing methods	5		R
T4.3	Identify target audience	2		R
T4.4	Develop test scenarios and tasks	20	T4.1, T4.2, T4.3	R
T4.5	Recruit participants	5	T4.3	R, UTP

T4.6	Conduct the usability test	15	T4.4, T4.5	R, UTP
T4.7	Conclusions	10	T4.6	R
T5	Project documentation	70		
T5.1	Collect all the information obtained	10		R
T5.2	Write the documentation	60	T5.1	R, PC, GD
T6	Bachelor thesis defense preparation	30	T5	R, PC, GD, GS
Total		540		

Table 2.1. Summary of tasks. *Source: Own compilation.*

R: Researcher, T: Tutor, D: Director, GEPT: GEP tutor, PC: computer, M: mobile phones, E: emulators, GD: Google Documents, GS: Google Slides, GC: Google Calendar, GM: Google Meet, MT: Microsoft Teams, GP: GanttProject, F: Figma, B: Bitbucket, S: Sourcetree, P: programming languages, IDE: Integrated Development Environments, UTP: usability test participants.

2.1.3. Resources

Every project needs different types of resources, human and material, in order to be carried out and it is crucial to allocate these resources effectively to ensure a successful organization and development of the project.

2.1.3.1. Human resources

In this project we can find mainly four human resources. First of all, the researcher is responsible for the development of the project. In addition to defining, planning and documenting the project, the researcher will have to carry out different roles such as project manager, UI/UX designer, developer, quality assurance tester and technical writer. On the other hand, both the tutor and the director of the project are in charge of leading and guiding the researcher for the correct development of the project. Finally, the GEP tutor is in charge of helping the researcher to correctly manage the project, through the resolution of doubts and the feedback of the different deliverables, during the first month of the project.

2.1.3.2. Material resources

Material resources such as books, articles and documentation, both in physical and online formats, are essential for conducting the research phase of the project. Additionally, I will require some software and hardware resources to carry out the practical part and the management of the project.

Hardware resources

The essential hardware resources needed are a computer, a monitor and at least two mobile phones of different operating systems. In this project the devices from the following list will be used, which includes the model, operating system and RAM memory.

- **MacBook Pro:** macOS Ventura 13, 512GB SSD, 16GB of RAM.
- **Monitor:** Monitor HP X24c, 23.8" Full HD, 144 Hz.
- **iPhone 11:** iOS 15, 128GB, 4GB of RAM.
- **Samsung Galaxy A53 5G:** Android 12, 128GB, 6GB of RAM.

In addition to physical devices, it is very important to have testing resources available because testing tools, such as emulators and simulators, are required to ensure that widgets will work properly on different configurations, platforms and sizes.

Furthermore, it is worth mentioning the need of having connectivity resources, such as Wi-Fi and mobile data, since nowadays are essential to be able to access the Internet and communicate easily, as well as being required for various tasks, such as downloading updates, uploading builds and performing tests. Therefore, the availability, reliability and speed of the network must be considered in the planning and management of the project.

Software resources

Several software resources will be needed during the development of the project. Regarding project management, I will need tools such as Trello to track project tasks and Google Calendar to plan monitoring and control meetings with the project tutor and, if we cannot meet in person, we will use Google Meet to be able to hold online meetings. On the other hand, monitoring and control meetings with the project director will be held in person at the office or online through Microsoft Teams. And let's not forget the Atenea and Racó tools, which are essential for communicating with GEP teachers and delivering assignments.

During the development of the project, I will need resources like Bitbucket, a version control software that I will use as a code repository, and Sourcetree, which allows viewing and managing the repository through its simple Git GUI. Additionally, I will use Figma for the design phase. During the development phase, I will use the React Native framework and some programming languages, such as TypeScript, JavaScript, Swift, Objective C, Java, as well as the following Integrated Development Environments (IDEs) for the development phase: Visual Studio Code, Android Studio and Xcode. Moreover, I will need a developer account on both Android and iOS platforms. Also, I will need a developer account for both Android and iOS, i.e. Google Developers and Apple Developers account respectively.

Finally, regarding project documentation, it is worth noting that all the documentation that I will do throughout the project will be through Google Documents and the oral defense presentation through Google Slides. Moreover, the Gantt chart shown in Figure 2.1. has been created using GanttProject.

2.2. Estimates and the Gantt

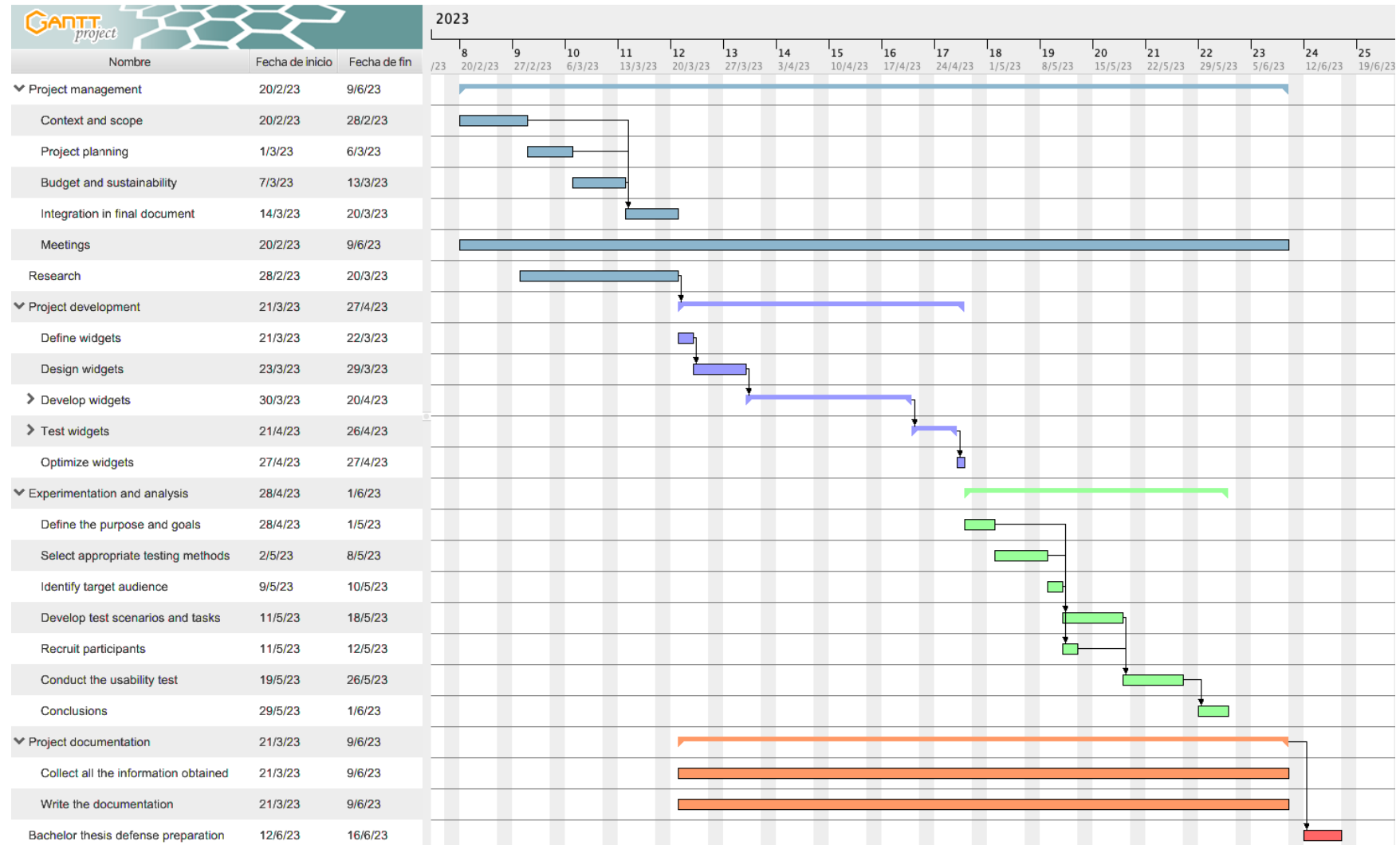


Figure 2.1. Gantt chart. *Source: Own compilation.*

2.3. Project final status

The project has made significant progress since its inception, albeit with some deviations from the initial project planning that have slightly altered the project tasks and consequently the estimated hours and budget for those affected tasks.

Throughout the project, it became apparent that certain tasks were larger or more complex than expected, requiring additional time and resources to ensure their successful completion. This is the main case of the task for the development of the widgets (T3.3) of the development phase of the project (T3). On the Android side, the number of hours has increased slightly. On the contrary, on the iOS side, it has been significantly affected due to the complexity and restrictions included in the development of widgets in this operating system. This also applies to some tasks in the experimentation and analysis phase of the project (T4). In this case, the tasks to develop test scenarios and tasks (T4.4) and to conduct the usability test on the participants (T4.6) have had a slight increase in duration. This is because it was not initially thought that such a dedication would be needed and therefore it was underestimated.

Furthermore, the project encountered unexpected blockages that hindered the planned schedule for specific tasks. In some cases, the delays required the inclusion of new tasks to address emerging requirements or mitigate potential risks. As previously mentioned, in the case of iOS, the development of widgets has been much more complex than in Android due to certain restrictions and limitations of the operating system, and therefore it has meant a variation in the initial planning, increasing the number of hours needed to complete the task. But it has also involved unexpected blockages that have given rise to new tasks to manage them. This has been reflected in the fact of having to change the way in which one feature was to be carried out to another so that it could work as expected.

Otherwise, there were cases where certain tasks turned out to be shorter than initially planned. These achievements were welcome as they provided an opportunity to offset the setbacks. Regarding these cases, it is worth mentioning the design (T3.2) and testing (T3.4) tasks of the project development phase (T3) have suffered a small decrease in the number of hours for each task because they had been initially overestimated and could really be carried out successfully in less time than expected.

In conclusion, the project has experienced several unforeseen events and challenges that have impacted the initial planning time. It is worth noting that despite these adjustments, the project has remained firm in its commitment to achieve its objectives within the given constraints. Through proactive management and a lot of dedication, progress has been made and the project remains focused on its initial objectives.

2.4. Risk management: alternative plans and obstacles

Throughout the development of the thesis, some potential obstacles and risks may arise that I will have to face and it is extremely important to anticipate and have contingency plans to deal with them. In this section, I will explain how the potential obstacles and risks presented in Section 1.3.3. can be solved by introducing new tasks and rearranging the planning. In addition, the level of risk that would be involved if it occurs is indicated and also the additional resources needed.

- **Bugs [Medium risk].** To mitigate these risks, good programming practices should be followed such as testing code thoroughly and using version control. Regarding bugs in third-party libraries used in the project, the ideal solution would be to wait for a library update to fix the bug, but since the project has very strict deadlines, what I will have to do in these cases is to program the necessary functions of the library from scratch and test that they work as expected. Consequently, the total duration of the project will be affected and therefore I will have to create a new task to fix the risk which will take approximately 5 hours depending on different factors like the difficulty of the bug. In this new task the researcher, the computer, Bitbucket, the IDEs and the programming languages will be the necessary resources.
- **Deadline of the project [Medium risk].** One of the main problems of planning a project before starting it is not knowing exactly the tasks that are going to be carried out throughout the project, but also underestimating or overestimating the number of hours dedicated to each task. This can be solved by defining a second project planning at a more advanced point in the project. In case I run out of time for the project deadline, I can solve this problem by increasing the number of working hours per day or taking advantage of weekends and national holidays to advance work. In any case, it is estimated that the resolution of the incident will take approximately 25 hours. The researcher, the computer and Ganttproject will be the main resources in the new task.
- **Impossibility of meeting with the tutor and director in person [Medium risk].** In case it is not possible to hold the control and monitoring meetings with the tutor and director of the project in person, the solution will be to hold those meetings online. Luckily, after suffering from the global COVID-19 pandemic, nowadays we are generally more used to holding online meetings than a few years ago and the number of tools available on the market does not stop growing. The new task to schedule the meeting is estimated to last approximately 1 hour, because we first have to communicate via email and agree on how we will conduct the meetings. In this situation, human resources such as the researcher, the tutor or the director of the project, and software resources such as Google Calendar, Google Meet or Microsoft Teams will be key to successfully overcoming the obstacle.
- **Inexperience in the field [High risk].** In case I have to work with a technology that I have no prior knowledge or experience of, such as a new programming language, then I will need to

create a new task that will take approximately 30 hours to complete before starting the programming part and therefore a new dependency will be born. The researcher, the computer and the technology involved will be the main resources in the new task.

- **Personal challenges [Low risk].** Dealing with personal challenges while working on a project can be a difficult situation. Some of the steps that can help to manage these challenges may be to seek support from the tutor or the director of the project and explain the situation to them for advice on how to handle the situation while moving forward with the project, if possible. Another possible solution to this obstacle would be to set realistic goals, i.e. break the project down into smaller, more manageable tasks to help me stay focused and motivated, and also allow me to progress even if I'm not feeling my best. The new task that would be included in the project schedule to solve this obstacle will take approximately 6 hours. In this risk, the four main human resources will be involved: the researcher, the tutor and the director of the project.
- **Technical difficulties [Low risk].** Depending on the technical difficulty that has occurred, there are different troubleshooting steps that can be followed. For example, if the computer crashes, try to restart it and run a virus scan. If there has been data loss, try to recover it using a data recovery tool. If the network connection is bad, try to reset the router or contact the Internet service provider. However, if I am unable to resolve the issue on my own, I will have to seek help from technical support. The researcher, the computer and the connectivity resources will be the main resources in the new task which will take approximately 12 hours to resolve the risk.

In case any potential obstacle or risk appears that is not included in the contingency plan, I will be able to schedule an extraordinary meeting with the tutor or the director of the project in order to discuss and find a possible solution, and it is for this reason that I have decided to overestimate the number of hours dedicated to the meeting task.

3. Budget

In this section I will discuss the economic cost of the project. To do so, I will describe all the elements of the estimated project budget, including staff costs by activity, generally calculated costs, amortization, contingencies and incidentals. In addition, I will explain the mechanisms to control potential budget deviations, including numerical indicators.

3.1. Staff costs per activity

To make an adequate estimate of personnel costs, it must be taken into account that this project is carried out only by a student who earns €10/hour according to agreement 2022/05/13, which approves the internship regulation of UPC [20]. For this reason, the student will have to assume different roles such as project manager, researcher, UI/UX designer, developer, quality assurance tester and technical writer.

The following table summarizes the staff costs per activity, also known as PCA, that will be carried out in the project, showing their unique identifier (ID), a descriptive name, the number of hours dedicated to each task and the cost of each one.

ID	Task	Time (h)	Cost (€)
T1	Project management	60	600
T1.1	Context and scope	15	150
T1.2	Project planning	10	100
T1.3	Budget and sustainability	10	100
T1.4	Integration in final document	5	50
T1.5	Meetings	20	200
T2	Research	90	900
T3	Project development	230	2,300
T3.1	Define widgets	10	100
T3.2	Design widgets	20	200
T3.3	Develop widgets	165	1,650
T3.3.1	Develop Android widgets	60	600
T3.3.2	Develop iOS widgets	105	1050

T3.4	Test widgets	30	300
T3.4.1	Test Android widgets	15	150
T3.4.2	Test iOS widgets	15	150
T3.5	Optimize widgets	5	50
T4	Experimentation and analysis	60	600
T4.1	Define the purpose and goals	3	30
T4.2	Select appropriate testing methods	5	50
T4.3	Identify target audience	2	20
T4.4	Develop test scenarios and tasks	20	200
T4.5	Recruit participants	5	50
T4.6	Conduct the usability test	15	150
T4.7	Conclusions	10	100
T5	Project documentation	70	700
T5.1	Collect all the information obtained	10	100
T5.2	Write the documentation	60	600
T6	Bachelor thesis defense preparation	30	300
Total		540	5,400

Table 3.1. Staff costs for each task. *Source: Own compilation.*

Since the staff salary is calculated from the gross salary, the cost of social security payments must be included. In general terms, this means multiplying the gross salary by 1.3 and, consequently, the staff cost per activity is increased by a total of $€5,400 * 1.3 = €7,020$.

In addition, to this cost must be added the management expenses of the educational cooperation agreement for doing external academic internships in the company, which implies adding an amount of €847.80. Thus, the final cost is $€7,020 + €847.80 = €7,867.80$.

3.2. Generic costs

Material resources cost

One aspect to take into account in the project budget is the amortization of the necessary material resources. The hardware amortization time is 4 years, taking into account that 2023 has 251 working days and that the project lasts 90 working days, the amortization cost can be calculated with the following formula:

$$\text{Amortization} = \frac{\text{price}}{\text{lifetime}} * \text{project time} = \frac{\text{price}}{4 * 251 \text{ days}} * 90 \text{ days}$$

Hardware resource	Price (€)	Amortization (€)
MacBook Pro	2,499	224
Monitor	149	13.35
iPhone 11	739	66.25
Samsung Galaxy A53 5G	449	40.25
Total		343.85

Table 3.2. Amortization costs of hardware resources. *Source: Own compilation.*

On the other hand, the cost of most of the software resources are free but the cost of the licenses to publish and maintain the mobile application in the stores must be considered. To be able to have an app in the Play Store, only a Google Developer account is required with a one-time fee of €25. However, to have it in the App Store an Apple Developer account is required with an annual subscription of €99. Hence, taking into account that the duration of the project is 5 months, the cost of an Apple Developer account is €99 / 12 months * 5 months = €41.25.

Therefore, the total amortization cost of material resources, both hardware and software, is €343.85 + €66.25 = 410.1€.

Electric cost

The cost of electricity is around €70 per month. Therefore, the total cost would be (€70/month) * (5 months) = €350.

Internet cost

The cost of the internet is around €60 per month. Therefore, the total cost would be (€60/month) * (5 months) = €300.

Travel cost

To be able to get to the office I need the T-Jove card for public transport and it costs €40 for three months. Therefore, I will need 2 cards that add up to a total of €80 in travel expenses.

Workspace cost

Regarding the cost of the workspace, the furniture that has been necessary during the development of the project must be taken into account, consisting of a desk, a chair and a lamp that cost €1000. The furniture amortization time is 10 years, taking into account that 2023 has 251 working days and that the project lasts 90 working days, the amortization cost can be calculated with the following formula:

$$Amortization = \frac{price}{lifetime} * project\ time = \frac{1000}{10*251\ days} * 90\ days = 35.85\text{€}$$

Generic cost of the project

Table 3.3 summarizes all the generic costs (CG) of the project introduced in the previous sections.

Concept	Cost (€)
Material resources	410.1
Electric	350
Internet	300
Travel	80
Workspace	35.85
Total	1,175.95

Table 3.3. Generic cost of the project. *Source: Own compilation.*

3.3. Budget deviations

3.3.1. Contingency costs

During the development of the project unforeseen events may appear, which take part of the final budget. For this reason, it is necessary to have a contingency margin in order to overcome these

possible events. Thus, to the total cost (CPA + CG = 9,043.75€), a 15% contingency margin is added. With this, the computed contingency cost is 1,356.56€.

3.3.2. Incidental costs

Finally, the incidental costs of the project that would imply applying the alternative plans defined in the previous deliverables are added. Table 3.4 shows the cost of resolving each potential risk or obstacle that may occur, which is calculated by multiplying the estimated price it would cost by the probability risk of occurrence.

Incident	Time (h)	Estimated cost (€)	Risk (%)	Cost (€)
Bugs	5	65	35	22.75
Deadline of the project	25	325	40	130
Impossibility of meeting with the tutor and director in person	1	0	30	0
Inexperience in the field	30	390	80	312
Personal challenges	6	78	5	3.9
Technical difficulties	12	156	10	15.6
Total				484.25

Table 3.4. Incidental costs of the project. *Source: Own compilation.*

3.4. Final budget

The final budget of the project is presented in Table 3.5, computed using all the justified costs calculated in the previous sections.

Activity	Cost (€)
Costs per activity (CPA)	7,867.80
Generic costs (CG)	1,175.95
Contingency costs	1,356.56
Incidental costs	484.25
Total	10,884.56

Table 3.5. Total cost of the project. *Source: Own compilation*

3.5. Management control

Next, I will explain the procedures needed to control the budget and also I will define control indicators that help monitor cost deviations during the execution of the project.

A control mechanism to monitor potential budget deviations is to obtain the difference between the actual resources consumed and the estimated resources consumed. Using the following formula, the cost deviation in human resources is calculated for each task described in the Gantt chart (CPA):

$$\text{Human Resources Deviation} = (\text{Estimated cost per hour} - \text{Real cost per hour}) * \text{Total real hours}$$

Moreover, the same applies to generic costs (GC). In the case of material resources, such as hardware or software, the cost variance is calculated through the amortization process:

$$\text{Amortization Deviation} = (\text{Estimated usage hours} - \text{Real usage hours}) * \text{Price per hour}$$

Finally, the total budget deviation is obtained by adding the previously calculated deviations. Neither contingency costs nor incident costs are taken into account.

$$\text{Budget Deviation} = \text{Human Resource Deviation} + \text{Amortization Deviation}$$

In summary, with all these control indicators it is possible to have a control on where the deviations from the budget are and how much they will cost. In case the budget deviation is negative, which means that the estimated costs of resources have been underestimated, the contingency margin must be applied.

4. Sustainability

4.1. Self-assessment

The self-assessment survey of the current domain of sustainability competence has made me reflect on the importance of public awareness, behavior and priorities related to sustainability issues. It has also made me realize that the word sustainability does not only consist of preserving and contributing to improve the environment, but also focuses on two fundamental pillars: the economic and social dimensions. In addition, the survey has made me think about the causes, consequences and possible solutions of the economic, environmental and social problems that currently exist. Moreover, it has also made me analyze the relevance of these problems in my professional field.

Furthermore, I want to highlight the fact that I was completely unaware of the existence of the large number of indicators to evaluate the three different dimensions, which surprised me a lot while doing the survey. Honestly, I have learned the importance of these indicators as tools used in order to measure and monitor progress towards achieving sustainable development goals. This is due to the fact that they provide a way to assess the impact of human activities on the economy, environment and society, as well as identify where sustainability issues exist and how they can be addressed.

In conclusion, sustainability is crucial for any project in any sector because it ensures that the project benefits the economy, environment and society while minimizing negative impacts. Therefore, it is an essential component of responsible project management and a key factor for success.

4.2. Economic dimension

Regarding PPP: Have you estimated the cost of carrying out the project (human and material resources)?

The reflection on the estimated cost for the project can be found in Section 3, which also includes the management control. The estimate of the cost of carrying out the project takes into account human and material resources, such as hardware and software, as well as indirect costs, such as electricity, internet, transportation and workspace. In addition, potential deviations from the final budget have been taken into account through the calculation of contingency and incident costs.

Regarding Useful Life: How will your solution improve economically over existing ones?

With the aim of reducing the economic cost of the project, the reuse of material resources, both hardware and software, the reduction of energy consumption and the minimization of waste will be taken into account. In addition, my project improves the execution flow of a routine process at the business level, which results in an economic improvement.

4.3. Environmental dimension

Regarding PPP: Have you estimated the environmental impact that the project will have? Have you considered minimizing the impact, for example, reusing resources?

I have not estimated the environmental impact of the project. However, this project does not waste material or have a high electricity consumption during its development.

How the hardware was made and the resources it consumes, such as electricity, is not in my control. Nevertheless, it is in my hands to reduce the impact on the environment as much as possible by properly recycling all hardware once it is no longer useful. And yes, I have considered minimizing the environmental impact by reusing resources, as mentioned in the economic dimension section.

Regarding Useful Life: How will your solution improve the existing ones environmentally?

IT projects often rely on hardware and energy-intensive infrastructure, which can have a significant environmental impact. Therefore, the project will aim to reduce its carbon footprint by using good practices, such as cloud computing, which can reduce energy consumption.

4.4. Social dimension

Regarding PPP: What do you think the realization of this project will bring you on a personal level?

I believe that the realization of this project will bring me a range of personal benefits, such as career advancement, personal satisfaction and a sense of accomplishment, especially because it is a challenging and long-term project. In addition, it is a great opportunity for learning and personal growth, such as developing new skills or expanding knowledge in this professional field.

Regarding Useful Life: How will your solution improve socially (quality of life) over existing ones? Is there a real need for the project?

The result of the project will have a direct impact on society, especially in company employees and consumers of the mobile application. The project represents a potential opportunity for improvement in terms of user experience, functionality and other key factors, since it aims to develop an

user-friendly and intuitive interface and also offer custom configurations. Therefore, there is a real need for the project because it aims to meet and successfully achieve the requirements of a new functionality of the mobile application in order to increase user engagement, user experience and other indicators, as well as make life easier for users and that they can spend time on other tasks.

5. Specification

The specification of a project is a key process that helps to understand and define the requirements and functionalities of the system to be developed. In this section, a detailed vision of the specification of this project will be provided, covering both the conceptual data model and the use cases, in order to have a clear understanding of how users will interact with the system and what functionalities and behaviors it is expected to fulfill.

5.1. Conceptual Data Model

The following conceptual data model, shown in Figure 5.1., is a very simple, high-level representation of the key entities and relationships within the system. These relationships indicate that a widget can display 0 or more tokens, a token can be associated with 0 or more widgets, and each token inherits from a single identity. Therefore, the diagram provides a comprehensive understanding of the structure and meaning of the data without technical implementation details.

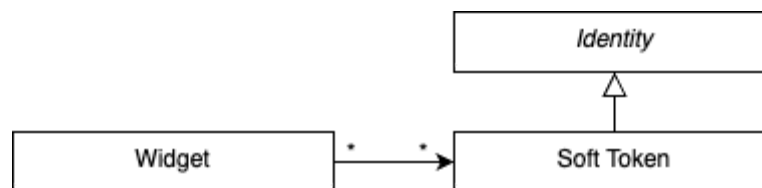


Figure 5.1. Conceptual data model. *Source: Own compilation.*

5.2. Use Case Diagram

The easiest way to graphically represent the actors, the actions and the interactions that occur between them and the system is with a use case diagram. The use case diagram of this project is presented in Figure 5.2., which provides a visual overview of the different user roles and the functionalities that the system is expected to provide.

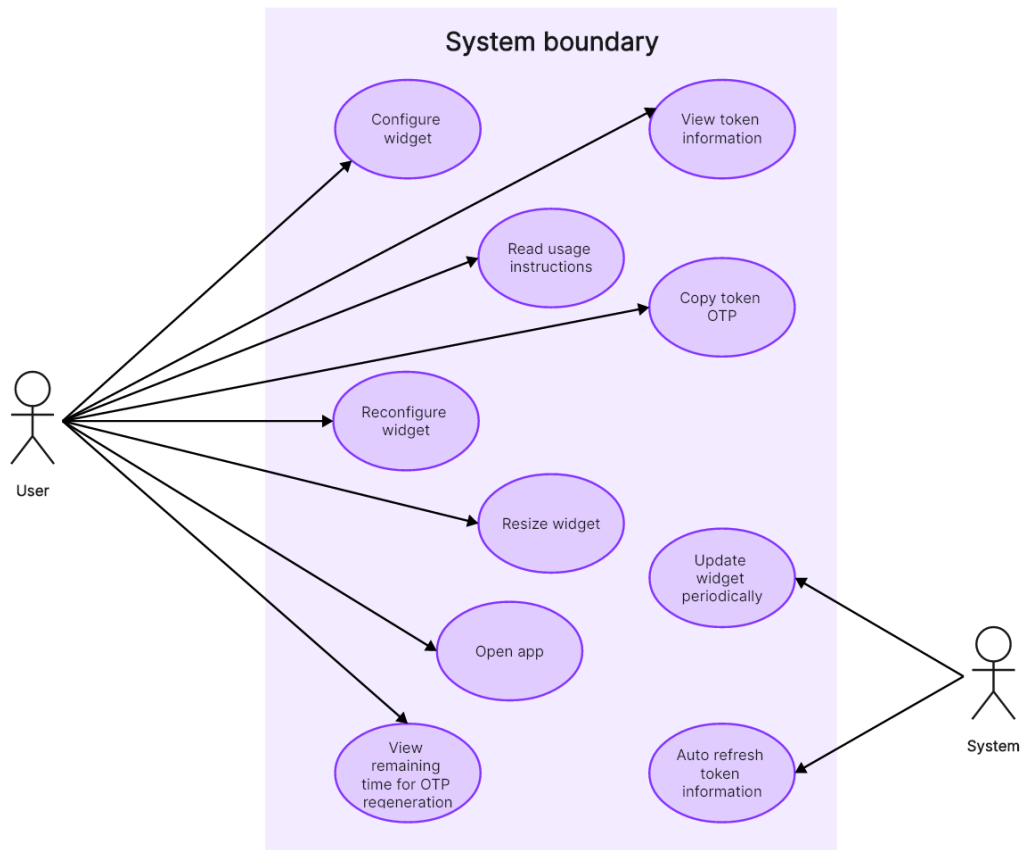


Figure 5.2. Use case diagram. *Source: Own compilation*

5.3. Use Case Descriptions

The description of the use cases focuses on providing specific details about each of the scenarios in which users interact with the system. Each use case describes a particular situation, specifying the actions that the actors can perform and the expected responses of the system, and consists of the following main elements:

- **Actors:** They represent users, external systems or devices that interact with the system. Actors can be real people, roles within an organization, or even other systems.
- **Description:** Provides an overview of the use case, including its name, main purpose, and a brief description of the task being performed.
- **Event flow:** Describes the specific steps that are followed in the use case, including the actions performed by the actors and the corresponding system responses. The event stream can contain main actions, alternative actions, and exceptions.
- **Preconditions and postconditions:** Specify the conditions necessary for the use case to start (preconditions) and the conditions that are met at the end of the use case (postconditions).
- **Extensions (optional):** Represent additional situations or actions that can occur within the use case.

Next, the descriptions of the use cases identified in the project are presented.

Use Case	1. Configure a widget
Actors	User
Description	Allow the user to configure a widget with selected tokens
Preconditions	The mobile app is installed and the user has at least one registered token
Event flow	<ol style="list-style-type: none"> 1. The user selects the option to add widget 2. The system displays a settings screen with the list of tokens available in the app 3. The user selects one or more tokens from the list 4. The user confirms the selection 5. The system saves the widget configuration with the selected tokens 6. The system adds the widget on the home screen and displays the selected tokens
Extensions	3.a. If the user has not selected any token and confirms the selection, then the system adds the widget on the home screen and displays instructions on how to add tokens in the widget
Postconditions	The widget has been successfully configured and shows the selected tokens

Table 5.1. Use Case 1. Configure a widget. *Source: Own compilation*

Use Case	2. Read usage instructions in the widget
Actors	User
Description	Provide the user with clear instructions on how to configure the widgets
Preconditions	The widget is displayed on the home screen but no token is shown
Event flow	<ol style="list-style-type: none"> 1. The user views the widget on the home screen 2. The system displays instructions on how to add and configure tokens in the widget 3. The user follows the instructions on how to add tokens in the widget
Postconditions	The user has received clear instructions on how to configure the widget to display tokens

Table 5.2. Use Case 2. Show usage instructions in the widget. *Source: Own compilation*

Use Case	3. Reconfigure a widget
Actors	User
Description	Allow the user to change the tokens displayed in a widget
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The user taps the settings icon displayed at the top of the widget 2. The system displays a settings screen with the list of tokens available in the app 3. The user selects one or more tokens from the list 4. The user confirms the selection 5. The system saves the widget configuration with the selected tokens 6. The system updates the widget and displays the selected tokens
Postconditions	The widget has been successfully reconfigured and shows the selected tokens

Table 5.3. Use Case 3. Reconfigure a widget. *Source: Own compilation*

Use Case	4. View token information
Actors	User
Description	Allow the user to view basic token information directly in the widget
Preconditions	The widget has been previously configured, is displayed on the home screen and shows at least one token
Event flow	<ol style="list-style-type: none"> 1. The user views the widget on the home screen For each token configured in the widget: 2. The system displays the token name 3. The system displays the serial number of the token 4. The system displays an icon representing the status of the token 5. If the token is “unlocked”, the system displays the OTP of the token 6. If the token is “locked”, the system displays the message “Tap to unlock” 7. If the token is "blocked", the system displays the message "Tap to unblock"
Postconditions	The user has viewed the basic information of a token from the widget

Table 5.4. Use Case 4. View token information. *Source: Own compilation*

Use Case	5. Copy the OTP of a token
Actors	User
Description	Allow the user to copy the OTP of a token from the widget
Preconditions	The widget has been previously configured, is displayed on the home screen and shows at least one token
Event flow	<ol style="list-style-type: none"> 1. The user views the widget on the home screen 2. The user identifies the token from which wants to copy the OTP 3. The user taps the token to copy the OTP 4. If the token is “unlocked”, the system copies the OTP to the clipboard 5. If the token is “locked”, the system opens the app and shows the authentication screen <ol style="list-style-type: none"> 5.1. If the user authenticates successfully, the token is unlocked, the token details screen is shown and the system copies the OTP to the clipboard 6. If the token is “blocked”, the system opens the app and shows the block screen
Postconditions	The user has copied the OTP of a token from the widget if it is unlocked

Table 5.5. Use Case 5. Copy the OTP of a token. *Source: Own compilation*

Use Case	6. Resize the widget
Actors	User
Description	Allow the user to adjust the size of the widget on the home screen to suit his/her preferences and available space
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The user presses and holds the widget on the home screen 2. The system allows the user to resize the widget by dragging the edges or corners in or out 3. The user adjusts the size of the widget according to his/her preferences 4. The system resizes the widget based on user actions and shows more or less information depending on the new size.
Postconditions	The widget has been resized based on user preferences

Table 5.6. Use Case 6. Resize the widget. *Source: Own compilation*

Use Case	7. Open the app from the widget
Actors	User
Description	Allow the user to quickly access the application from the widget
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The user touches the upper area of the widget where the logo and the name of the app are displayed 2. The system opens the app
Postconditions	The user has accessed the application from the widget

Table 5.7. Use Case 7. Open the app from the widget. *Source: Own compilation*

Use Case	8. View the remaining time for the regeneration of the OTPs
Actors	User
Description	Allow the user to see the time remaining for the OTPs to regenerate in the widget
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The user views the widget on the home screen 2. The user observes the timer that shows the seconds remaining for the regeneration of the OTP codes 3. The user waits until the timer starts a new cycle to see that a new OTP is generated. 4. The system updates the widget content with the new OTPs generated
Postconditions	The user has observed the remaining time for the regeneration of the OTPs in the widget

Table 5.8. Use Case 8. View the remaining time for the regeneration of the OTPs. *Source: Own compilation*

Use Case	9. Update the widget content periodically
Actors	System
Description	Ensure that widget content is automatically updated at regular intervals
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The system starts a timer with a defined interval 2. When the specified interval is reached, the widget refreshes the displayed content, such as token status and current OTP if applicable
Postconditions	The widget content has been automatically updated at regular intervals

Table 5.9. Use Case 9. Update the widget content periodically. *Source: Own compilation*

Use Case	10. Auto refresh token information
Actors	System
Description	Automatically update the information of the tokens displayed in the widget when there are changes in the app that affect them to reflect the updated information
Preconditions	The widget has been previously configured and is displayed on the home screen
Event flow	<ol style="list-style-type: none"> 1. The system detects changes in the tokens stored in the app that affect the widget 2. The system updates the information of the tokens in the widget 3. The widget reflects the changes made to the tokens automatically and without user intervention
Postconditions	The widget content has been automatically updated to reflect changes in token information

Table 5.10. Use Case 10. Auto refresh token information. *Source: Own compilation*

6. Design

Once the requirements and functionalities of the system have been defined, the design stage can begin. Next, the architecture of the system, the design of the user interface, along with some mockups, and the flowchart will be detailed in order to understand the internal functioning of the system.

6.1. Architecture

Due to the fact that the widgets have been designed for a big project of an existing mobile application, the system architecture is integrated into the mobile application architecture and consequently follows a three-layer architecture. For simplicity, the architecture design, shown in Figure 6.1, covers only the elements affected by the inclusion of widgets in the mobile application.

The three-layer architecture is a common architectural pattern used in software development. It provides a structured approach to separating the concerns of an application into distinct layers, promoting modularity, scalability, and maintainability. The three layers found in this architecture are the presentation layer, business layer, and data layer.

Presentation Layer

The presentation layer is responsible for handling the visual presentation and user interaction of the mobile application. It focuses on delivering an engaging and intuitive user experience. This layer encompasses all the UI components, such as screens, assets (icons, images, drawables...) and styles (themes, fonts, colors...). The presentation layer communicates with the underlying layers to fetch and display data, as well as to trigger business logic operations.

Business Layer

The business layer represents the logical operations and processes that affect the behavior of the mobile application. In this layer we can find the models, controllers, managers, the bridge files necessary for the communication between the React Native side and the native side of the app, as well as the interaction with the SDK in order to fetch data from the IDaaS API. The business layer acts as an intermediary between the presentation layer and the data layer. It communicates with the presentation layer to receive user input and update the UI accordingly and it interacts with the data layer to retrieve or update data as required.

Data Layer

The data layer manages the storage, retrieval, and persistence of data used by the mobile application. This layer handles tasks like data access, caching, synchronization, and data transformation. The data layer communicates with the business layer to provide or receive data for processing and storage. It ensures that data is available when needed and handles data synchronization between the application and external data sources, such as with Async Storage.

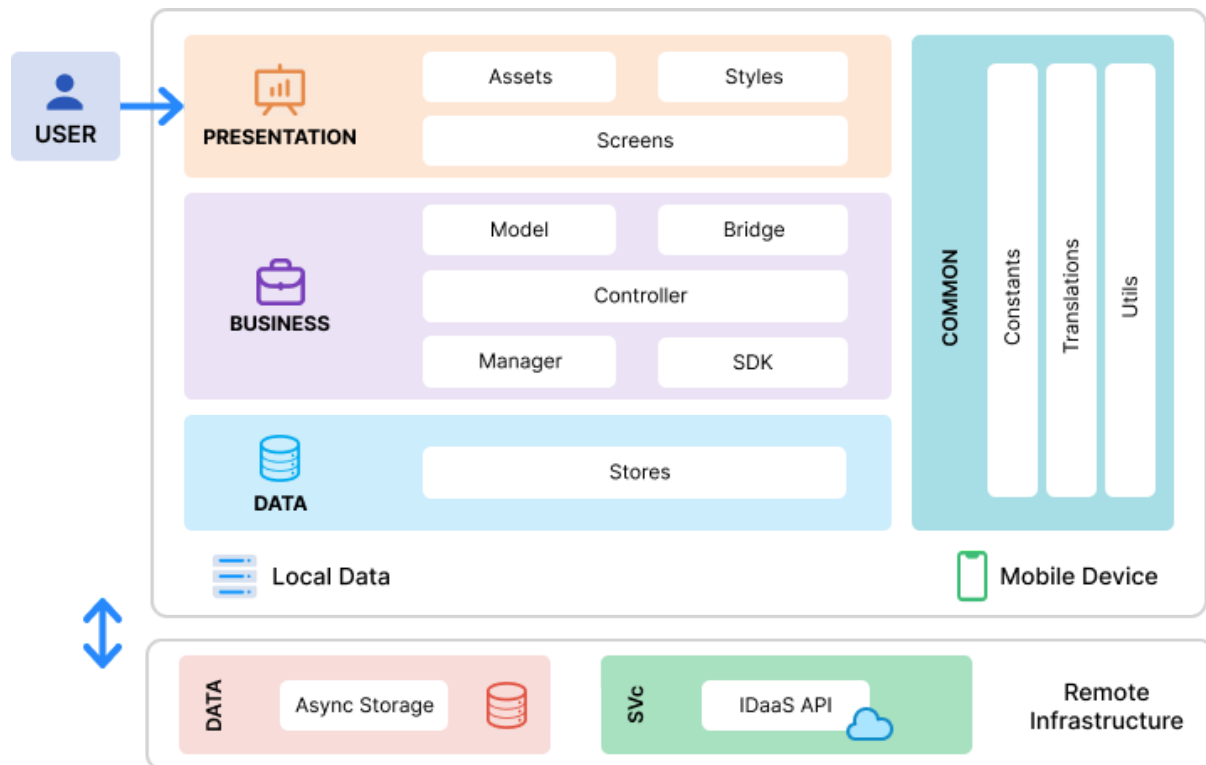


Figure 6.1. Architecture design. *Source: Own compilation.*

6.2. Design patterns

Design patterns in software are reusable solutions to common problems that developers encounter when designing and implementing software systems. They provide proven approaches to address specific issues and help improve the structure, flexibility, and maintainability of software. The design patterns that will be applied to the system are mentioned below:

Singleton

The Singleton pattern ensures that only one instance of a class is created throughout the application's lifecycle. It provides global access to this instance and is commonly used when a single object needs to coordinate actions across the system. This creational pattern will be useful for scenarios such as managing a shared resource.

Adapter

The Adapter pattern allows objects with incompatible interfaces to work together. It acts as a bridge between two interfaces, converting the interface of one class into another interface that clients expect. By doing so, it enables objects with different interfaces to collaborate seamlessly. This structural pattern will be useful to translate the format of the data to be sent or received.

Observer

The Observer pattern establishes a one-to-many dependency between objects, where multiple observers are interested in the state of an observable. When the observable's state changes, it automatically notifies all the registered observers, ensuring they are updated. This behavioral pattern will be useful for the graphical user interfaces.

6.3. User interface design

The user interface (UI) design is one of the most important aspects because it directly impacts the competitiveness, engagement and usability of a mobile application. A well designed widget enhances user experience by providing convenient and quick access to information and functionality, ensures intuitive interactions, and supports accessibility for a wider range of users.

Best practices

This section presents some of the principles, best practices and considerations that have been taken into account to create a visually appealing widget experience.

- **Look for a simple idea that is clearly related to your app's main purpose.** The first step in the design process is to choose a single idea for your widget. Throughout the process, use that idea to keep the widget's content focused and relevant.
- **In each size, display only the information that is directly related to the widget's main purpose.** In larger widgets, you can display more data or more detailed visualizations of the data.
- **Offer your widget in multiple sizes when doing so adds value.** In general, avoid simply expanding a smaller widget's content to fill a larger area. It is more important to create one widget in the size that works best for the content you want to display than it is to provide the widget in all sizes.
- **Avoid creating a widget that only launches your app.** A widget should display meaningful content and offer useful actions and deep links to key areas of your app. A widget that behaves like an app icon offers no additional value, which means people are likely to remove it from their screens.

- **Prefer dynamic information that changes throughout the day.** If a widget's content never appears to change, people may not keep it in a prominent position. It is important to find ways to keep their content fresh to invite frequent viewing.
- **Let people know when authentication adds value.** If your widget provides additional functionality when someone is signed in to your app, make sure people know that.
- **Help people recognize your widget by including design elements linked to your brand's identity.** Design elements like brand colors, typeface, and stylized glyphs can make a widget instantly recognizable.
- **Aim for a comfortable density of information.** When content appears sparse, the widget can seem unnecessary; when content is too dense, the widget isn't glanceable. Seek ways to curate the content so that people can grasp the essential parts instantly and view relevant details at a longer look.
- **Use standard margins to ensure your content is comfortably legible.** The standard margin width is 16 points. If your widget displays content like text, glyphs, and graphs, use the standard margins to avoid crowding the edges and creating a cluttered appearance.
- **Consider using the system font and text styles.** Using the system font helps your widget look at home on any platform, while making it easier for you to display great-looking text in a variety of weights, styles, and sizes. Additionally, avoid using very small font sizes.
- **Support Dark Mode.** Ideally, a widget looks great in both the light and dark appearances. In general, avoid displaying dark text on a light background for the dark appearance, or light text on a dark background for the light appearance.

Color palette, layout and icons

The design of the widgets incorporates the following color palette that aligns with the app's branding and the company's overall identity. This helps maintain brand consistency while enhancing the user experience.

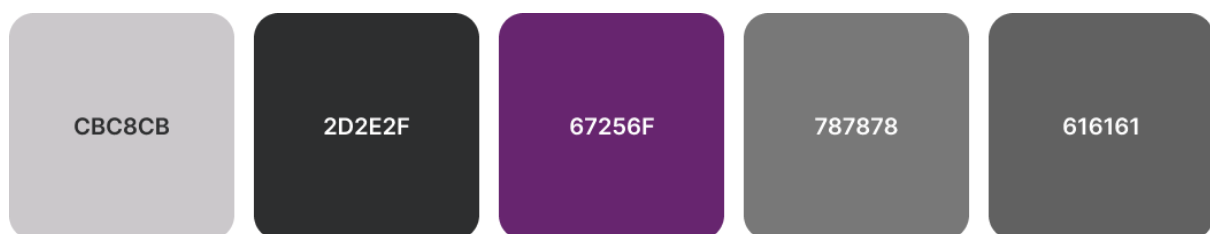


Figure 6.2. Color palette. *Source: Own compilation.*

On the iOS side, circular shapes and SF symbols are used in order to ensure familiarity and alignment with Apple's design language, providing a seamless and intuitive experience for users. However, in the design for Android devices the layout and distribution of the elements is different to adapt to the usual design that users find in Android. Also, custom icons reflecting the company's branding are used, establishing a unique identity. By adapting the icons and design elements to each operating system, the widgets integrate smoothly with the visual language of iOS and Android, providing an interface that feels native and intuitive to users.

Screens

The mockups created in task “T3.2. Design widgets” are presented below. Mockups are prototypes that allow you to have a clear idea of how the widget will look and behave. Furthermore, they help to detect and solve design problems, avoiding costly changes at later stages. Therefore, making mockups before entering the development phase is a valuable practice that helps to ensure design quality and end user satisfaction.

Figures 6.3. and 6.4. show the mockups for the design of the widget interface for iOS. Apple offers a wide range of widget sizes: system small, system medium, system large, system extra large, accessory corner, accessory rectangular and accessory inline. Since I want to develop widgets for iPhone on the home screen, the above list is reduced to the following available sizes: system small, system medium and system large. Following design guidance and best practices, I have decided to only provide the small and medium sizes because these are the sizes that work best for the content I want to display in each widget.

The idea and main objective of the small size widget is that the user can have quick access and be able to interact with his favorite token, which he/she will have previously chosen. In this way, the user will be able to consult basic information about the token such as its name and serial number to identify it, as well as the status of the token through the colors and icons displayed, and its OTP, if applicable. As we can see in Figure 6.3., in the event that a token is unlocked, an open lock icon will be displayed along with its primary color, in addition to showing the OTP in a large size so that the user can easily view it and copy it to use where needed. On the other hand, if a token is locked or blocked, a closed lock icon or a shield with an exclamation mark will be shown, respectively, along with a gray color and a message to indicate to the user what they must do to view and be able to copy the OTP of the token. In both cases, the color is a clear indicator of the state of the token and allows the user to easily understand the state of the token and what to do.

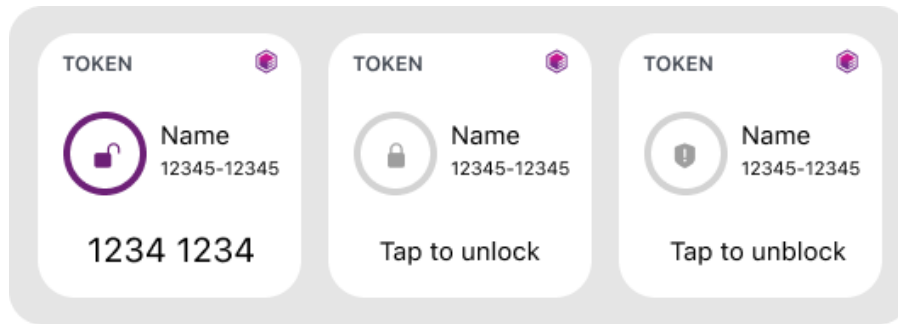


Figure 6.3. iOS small widget. *Source: Own compilation.*

Likewise, the idea and main objective of the medium-sized widget is that the user can have quick access and be able to interact with their favorite tokens, which he/she will have previously chosen. The advantage this widget provides is the ability to display a larger token name due to its larger size. In this case, the user will be able to have their tokens compactly grouped on their home screen.

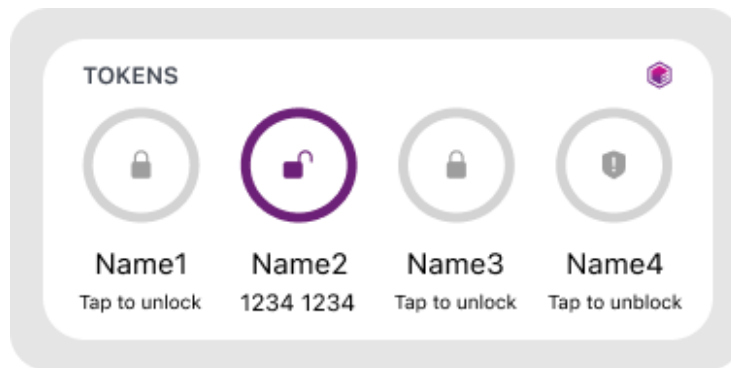


Figure 6.4. iOS medium widget. *Source: Own compilation.*

On the other hand, Figure 6.5. shows the mockups for the design of the widget interface for Android. As we can see now, only one size is considered since, unlike iOS that provides predefined sizes, Android allows you to change the size of the widgets freely according to the user's preferences. In any case, depending on the size chosen by the user, the widget will show more or less content to adapt to the available space.

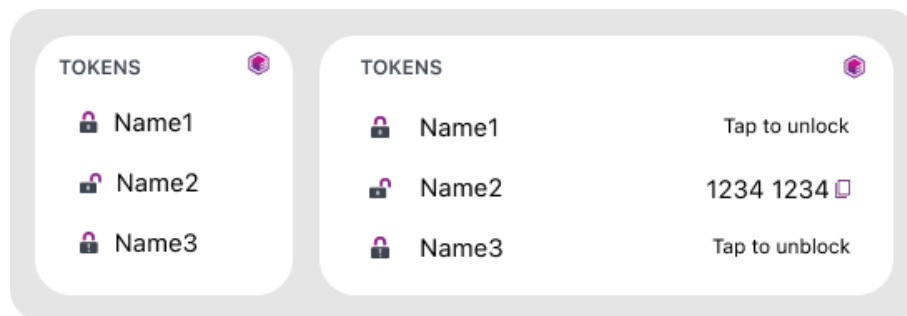


Figure 6.5. Android widget. *Source: Own compilation.*

Finally, in Figure 6.6. we can see an “instructions” view that will be displayed in the widget in case the user has no tokens registered in the app or has not yet chosen the tokens he/she wants to display in the widget. The main objective of this view is to guide the user with some simple instructions so that they can easily configure the widget if necessary and without any problem or doubt. This view will also be shown in the small size widget but with the text size reduced.

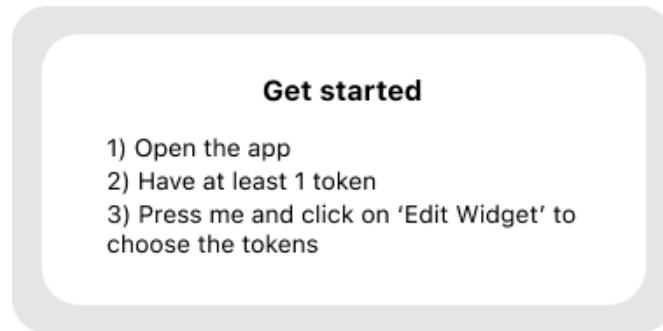


Figure 6.6. Instructions view. *Source: Own compilation.*

During the explanation of the different views of the widgets it has been mentioned that the tokens that are shown have been previously chosen by the user. This happens in the widget settings screen, shown in Figure 6.7., which allows the user to easily and intuitively choose the token or tokens he/she wants to display in the widget. This screen is displayed when a user adds a new widget to the home screen or wants to reconfigure a previously added widget to change the tokens that are displayed.

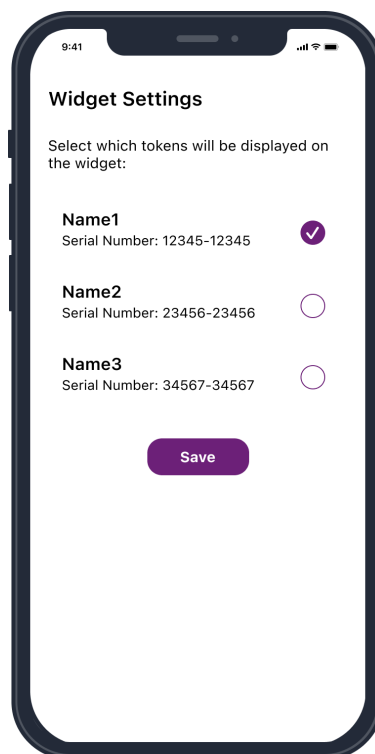


Figure 6.7. Widget settings screen. *Source: Own compilation.*

6.4. Flowchart

The flowchart in Figure 6.8. represents the navigation flow between the different views of the widget, which in some cases involves interaction with the mobile application. These cases include clicking on the widget or on a token displayed on it. In the first case, the app will open and show the Identities screen, which is the main screen of the app. In the second case, the app will be opened if the token is in lock status, the authentication screen will be displayed to unlock the token through the pin or biometrics (face ID/touch ID), or in block status, the block screen of a token will be displayed. This behavior is due to the fact that one of the objectives of this project is to maintain the security perspective of the app and, therefore, the widgets will only show the sensitive information of the tokens, such as the OTP, when possible.

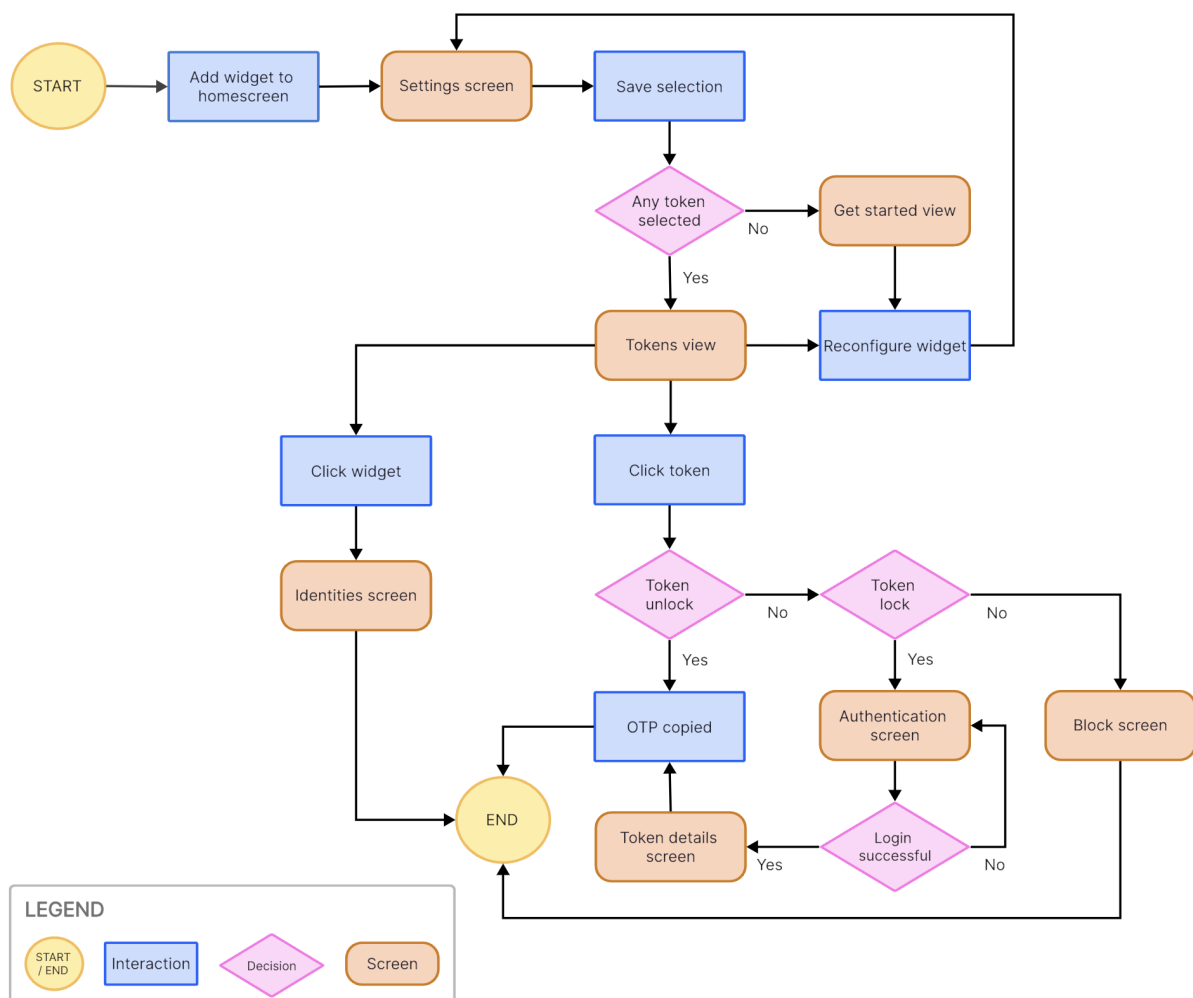


Figure 6.8. Flowchart. *Source: Own compilation.*

7. Development

Having designed the system, considering the architecture design, the patterns to be applied, the look and feel and the behavior that it will have through the mockups and the flowchart, respectively, we can now start the development stage. During this phase, the design will be implemented and converted to functional code. First of all, I will introduce the technologies involved in the project, then I will explain the development of widgets on Android and iOS, and finally I will make a comparison between the two operating systems and draw conclusions about it.

7.1. Technologies

In this section, the technologies and programming languages used during the development of the project are described, along with a diagram of the technology stack for a better visual understanding of the technologies involved in the system and how they communicate. It should be noted that the development of widgets for mobile applications is not yet possible through a framework, and consequently, they must be implemented in native code for both operating systems, Android and iOS.

React Native: React Native is an open-source framework developed by Facebook for building mobile applications using JavaScript and React. It allows developers to write code once and deploy it on multiple platforms, including iOS and Android.

JavaScript: JavaScript is a widely-used programming language primarily used for web development. It is also the main programming language for React Native. JavaScript is known for its versatility and runs on the client-side (web browsers) as well as the server-side (Node.js).

TypeScript: TypeScript is a superset of JavaScript that adds optional static typing and additional features to the language. It provides a more structured and scalable approach to JavaScript development, allowing developers to catch errors early during development and enhance code maintainability. TypeScript is often used in React Native projects to improve type safety and developer productivity.

Visual Studio Code: Visual Studio Code (VS Code) is a lightweight and highly popular source code editor developed by Microsoft. It supports a wide range of programming languages, including JavaScript and TypeScript. VS Code offers features such as syntax highlighting, intelligent code

completion, debugging support, version control integration, and an extensive library of extensions that enhance the editor's functionality

Objective-C: Objective-C is a programming language used for developing applications primarily on Apple's platforms, including iOS and macOS. It was the primary language for iOS app development before the introduction of Swift. Objective-C is an object-oriented language that adds Smalltalk-style messaging to the C programming language, making it suitable for developing apps with the Cocoa and Cocoa Touch frameworks.

Swift: Swift is a modern programming language developed by Apple for building applications on iOS, macOS, watchOS, and tvOS. It aims to be safe, fast, and expressive, providing a more concise and readable syntax compared to Objective-C. Swift is designed to work seamlessly with existing Objective-C code, allowing developers to gradually migrate their projects. It has become the preferred language for new iOS app development.

Apple Xcode: Xcode is the official integrated development environment (IDE) for iOS and macOS app development. It provides a complete set of tools for coding, debugging, and deploying applications for Apple platforms. Xcode includes features like a powerful source code editor, interface builder, simulator for testing apps, performance analysis tools, and integrated documentation.

iOS SDK: The iOS Software Development Kit (SDK) is a collection of tools, frameworks, and libraries that developers use to create applications for iOS devices. It provides a rich set of APIs and services for building user interfaces, handling user input, accessing device capabilities (such as camera and sensors), networking, data storage, and more.

Java: Java is a widely used programming language known for its versatility and platform independence. It is the primary language for Android app development and is supported by the Android platform. Java provides a robust and object-oriented approach to building Android applications.

XML: XML (Extensible Markup Language) is a markup language used to define the structure and content of data. In the context of Android development, XML is commonly used for creating user interfaces using layouts and defining resources such as strings, colors, and dimensions.

Android Studio: Android Studio is the official integrated development environment (IDE) for Android app development. It provides a comprehensive set of tools and features to streamline the development process. Android Studio offers a rich code editor, debugging capabilities, layout designers, and tools for building, testing, and deploying Android applications.

Android SDK: The Android Software Development Kit (SDK) is a collection of software tools, libraries, and resources that developers use to create Android applications. It includes the necessary APIs, development tools, emulator system images, and documentation required to build Android apps. The Android SDK provides developers with the foundation to access device features, create user interfaces, handle data, and perform various other tasks in Android app development.

The following diagram shows the technology stack of the system, which is the set of technologies used to develop the application, including programming languages, toolkits, SDKs, and frameworks.

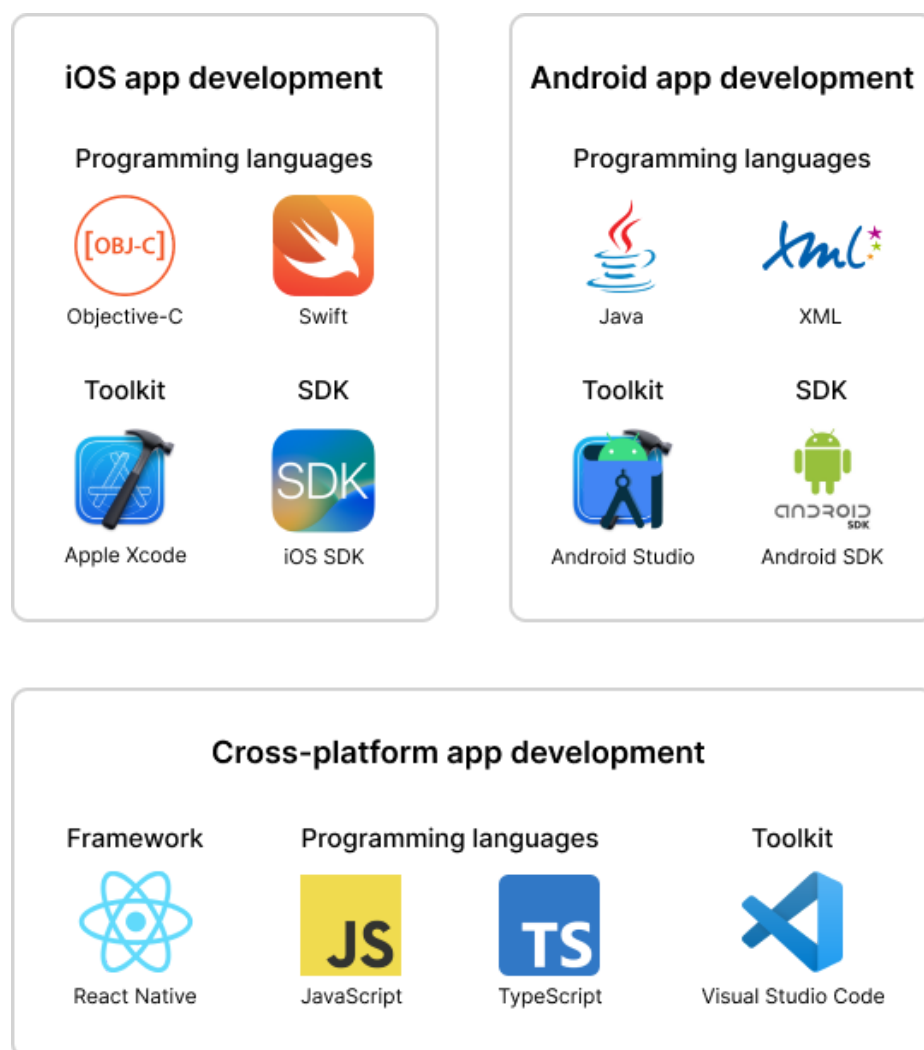


Figure 7.1. Technology stack. *Source: Own compilation.*

Since I am developing the widgets in native and most of the logic of the app is in React Native, there is a need for communication between these two worlds. Fortunately, React Native provides a bridge that enables interaction between JavaScript code and native code. This communication allows to utilize platform-specific features while benefiting from cross-platform development. The diagram below represents the communication between React Native and native through the bridge.



Figure 7.2. Communication between React Native and native. *Source: Own compilation.*

7.2. Android

As the Android documentation says, app widgets are miniature application views that can be embedded in other applications, such as the home screen, and receive periodic updates. These views are referred to as *widgets* in the user interface. Before going into the details of the development process, it is important to know and understand the main components that surround Android widgets.

- **Provider info file:** Widgets include an XML file that describes the metadata and initial properties for a widget, such as the widget's layout, initial or minimum size, update frequency, configuration activity, and preview image.
- **Widget provider class:** The widget provider contains the Java code for the widget. This class defines the basic methods that allow to programmatically interact with the widget. Through it, it is possible to receive broadcasts when the widget is updated, enabled, disabled, or deleted.
- **Layout:** The user interface of a widget is defined in an XML layout file. The same goes for native Android apps, but it should be noted that widgets support a more limited set of layouts and views than regular apps.
- **Configuration activity:** The configuration activity is listed in the widget's provider-info file and is automatically launched when the user adds a widget to the home screen.
- **App widget manager:** The app widget manager manages widget updates and sends the broadcast intents that the app widget provider receives to do the actual work of updating.
- **App widget host:** The app widget host is an app component that can hold and display other app widgets. The Android home screen is the most frequently used app widget host, although it is possible to create your own app widget host. In any case, in this project it was not necessary to implement an app widget host.

The following figure shows the processing flow of an Android widget, where we can observe that the operating system reads the Android manifest file that points to the provider info file to obtain the metadata, such as the initial layout that will be displayed in the widget when it is added to the home screen. In addition, the widget provider is declared as a broadcast receiver in order to be able to update the widget via the app widget manager when necessary, as well as defining the widget's behavior and using RemoteViews and XML layouts to define the widget's user interface.

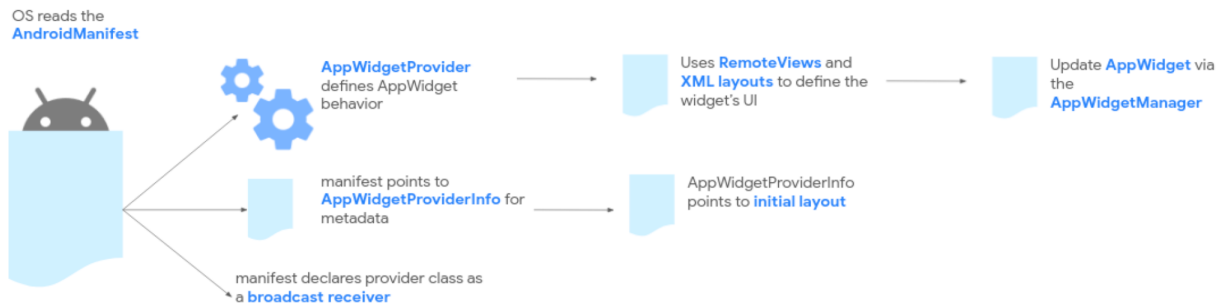


Figure 7.3. Android app widget processing flow. *Source: [31].*

Once the concepts and the processing flow of a widget are understood, the development phase begins. Android Studio allows to easily create a widget for an app project by simply selecting **File > New > Widget > App Widget** on the menu, and then configuring it in the New Android Component dialog that is opened, shown in figure 7.4. In this dialog, I entered the name of the widget, selected the placement to be the home screen, decided to make the widget resizable horizontally and vertically, chose the minimum sizes for the widget, checked the Configuration Screen checkbox to include an initial widget configuration activity, and finally selected the source language as Java.

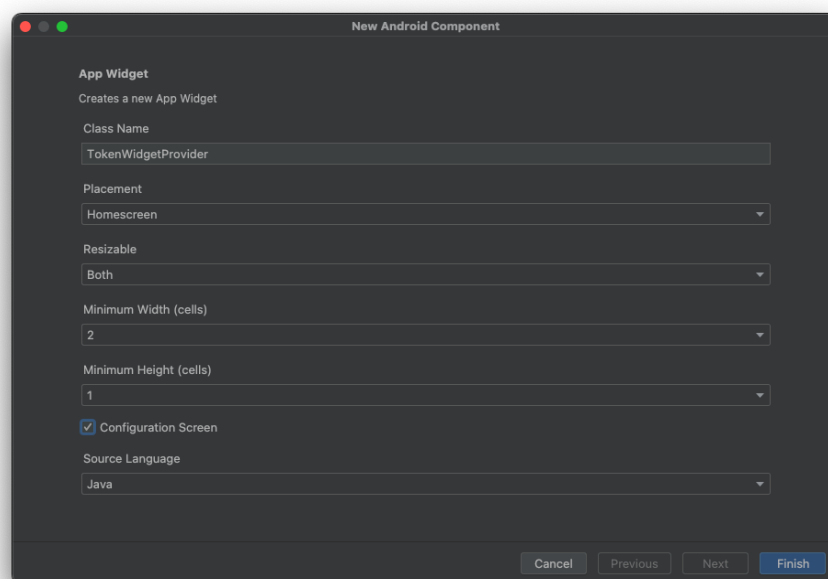


Figure 7.4. New Android Component dialog. *Source: Own compilation.*

After clicking the Finish button, Android Studio generates all the template files needed for adding the widget to the app and several files that already existed in the project are modified:

- A provider info file is added in `res/xml/token_widget_info.xml`.
- A new layout file for the widget is added in `res/layout/token_widget.xml`.
- The Android manifest is updated to include the provider and configuration activity classes.
- A provider class called `TokenWidgetProvider.java` is added to the project.
- A configuration activity class called `TokenWidgetConfigureActivity.java` is added.
- A new layout file for the configuration is added in `res/layout/token_widget_configure.xml`.

Declaring the widget provider info file

Firstly, I opened the provider info file generated and updated some of the metadata contained in it. This XML resource contains a single `<appwidget-provider>` element with the following attributes.

```
XML
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:minWidth="250dp"
  android:minHeight="110dp"
  android:targetCellWidth="4"
  android:targetCellHeight="2"
  android:minResizeWidth="110dp"
  android:minResizeHeight="40dp"
  android:resizeMode="horizontal|vertical"
  android:updatePeriodMillis="1800000"
  android:description="@string/token_widget_description"
  android:previewLayout="@layout/token_widget_preview"
  android:previewImage="@drawable/token_widget_preview"
  android:initialLayout="@layout/token_widget"
  android:configure="com...TokenWidgetConfigureActivity"
  android:widgetCategory="home_screen">
</appwidget-provider>
```

Code block 7.1. Widget provider info file. *Source: Own compilation.*

The first five attributes are part of the group of widget sizing attributes. The `minWidth` and `minHeight` attributes define the default size of the widget in dp. Android Studio provides these values based on the number of cells specified when the widget was created. The rule for how many dp fit into a grid cell is based on the equation $(70 \times \text{grid_size} - 30)$, where `grid_size` is the number of cells you want the widget to occupy. The following table helps to determine the `minWidth` and `minHeight` values depending on the number of cells it takes up space.

Number of cells (columns or rows)	minWidth or minHeight values
1	40dp
2	110dp
3	180dp
4	250dp

Table 7.1. Comparison between the number of cells and minWidth and minHeight. *Source: Own compilation.*

Starting in Android 12, the `targetCellWidth` and `targetCellHeight` attributes specify the default size of the widget in terms of grid cells. These attributes will be ignored in Android 11 or lower, and may be ignored if the home screen does not support a grid-based layout. For this reason, I defined both the `targetCellWidth/targetCellHeight` and `minWidth/minHeight` sets of attributes due to the fact that the mobile app supports versions from Android 9. The `minResizeWidth` and `minResizeHeight` attributes specify the widget's absolute minimum size. These attributes have allowed me to define the minimum size of the widget when the user resizes it, which is 2x1. Finally, the `resizeMode` attribute specifies the rules by which a widget can be resized. This value was specified when the widget was created and in this file it can be changed at any time. The possible values are "horizontal", "vertical", "horizontal|vertical", and "none".

The following attributes belong to the widget qualities attribute group. The `updatePeriodMillis` attribute defines how often the widget framework should request an update, which is set to 1800000 milliseconds because I will handle widget updates via the `AlarmManager`, it will be explained later. In order to improve the widget picker experience for the app, a description and dynamic widget previews have been added. The `description` attribute specifies the description for the widget picker to display for the widget. The `previewLayout` and `previewImage` attributes have the same objective, to specify a preview of what the widget will look like after it is configured, which the user sees when selecting the widget in the widget picker. The difference between them is that `previewLayout` is supported starting in Android 12 so for the same reason as mentioned above with the `targetCellWidth/targetCellHeight` and `minWidth/minHeight`, both attributes are specified. Then, the `initialLayout` attribute points to the layout resource that defines the widget layout. The `configure` attribute defines the activity that launches when the user adds the widget, allowing him/her to configure widget properties. Finally, the `widgetCategory` attribute declares that the widget can only be displayed on the home screen.

Creating the widget layouts

Secondly, I opened the layout file for the widget generated, as well as the widget preview layout, and started defining and designing the widget layout. Android Studio provides a Design tab that allows you to easily create the layout in the design editor by adding and dragging components (texts, buttons, images...) to it. However, I have preferred to create the layout of the widget programming the XML and observing the result at the same time with the Split tab, which divides the screen between the code and the design editor.

Due to the fact that the widget layouts are based on RemoteViews object hierarchies, and not View object hierarchies, fewer types of layouts and views are supported compared to standard views. A RemoteViews object can support the following layout classes: `FrameLayout`, `LinearLayout`, `RelativeLayout`, and `GridLayout`. And the following view classes: `AnalogClock`, `Button`, `Chronometer`, `ImageButton`, `ImageView`, `ProgressBar`, `TextView`, `ViewFlipper`, `ListView`, `GridView`, `StackView`, and `AdapterViewFlipper`. But taking into account that custom views and descendants of these classes are not supported. For the development of the widget layout, only the following components have been used: `RelativeLayout`, `LinearLayout`, `Button`, `ImageView`, `TextView`, and `ListView`.

During the creation of widget layouts, it is very important to take into account a series of elements such as margins, padding, alignments, dimensions, colors, styles, themes, and much more. They say that a picture is worth a thousand words... Figure 7.5. shows the comparison between the weather widget in API 27 and in API 31. A simple detail such as the inclusion of rounded corners can make a big difference in the visual aspect of a widget, as well as the first impression of the users.



Figure 7.5. Comparison Android weather widget API 27 vs API 31. *Source: Own compilation.*

Implementing the widget provider class

Next, I am going to explain how to implement the widget provider class, which is responsible for handling widget broadcasts and updating the widget in response to widget lifecycle events. But before it is important to declare the widget provider class in the application's `AndroidManifest.xml` file, so I added the following code block inside the `<application>` element.

```
XML
<receiver
    android:name=".TokenWidgetProvider"
    android:label="Soft Tokens"
    android:exported="false">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>
    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/token_widget_info" />
</receiver>
```

Code block 7.2. Android manifest file. *Source: Own compilation.*

The `<receiver>` element requires the `name` attribute, which includes the class name for the widget provider. In addition, I added a `label` attribute, which is displayed in the widget picker. The component is not exported because no separate process needs to broadcast the widget provider. Then, we find the `<intent-filter>` element that includes an `<action>` element with the `name` attribute. This attribute specifies that the widget provider accepts the `ACTION_APPWIDGET_UPDATE` broadcast, and it is the only broadcast that I explicitly declared because the app widget manager automatically sends all other widget broadcasts to the widget provider as necessary. Finally, the `<meta-data>` element specifies the widget provider info resource and requires the `name` attribute, which specifies the metadata name that is defined as `android.appwidget.provider`, and the `resource` attribute, which specifies the resource location.

Once the widget provider was declared as a broadcast receiver in the app manifest, I started implementing the widget's behavior in the widget provider class. The `TokenWidgetProvider` class extends `AppWidgetProvider` that extends `BroadcastReceiver` as a convenience class to handle widget broadcasts. It receives only the event broadcasts that are relevant to the widget lifecycle. Therefore, I decided to implement the following broadcast events:

- **onEnabled():** This method is called when an instance of the widget is created for the first time, which allows me to activate the `AlarmManager` to trigger the widget's timer, which will be explained later.

- **onDeleted():** This is called every time a widget is deleted from the widget host. In this method I defined the logic to clean up the preferences of the widget, which will be explained later.
- **onDisabled():** This method is called when the last instance of the widget is deleted from the widget host, which allows me to clean up any work done in onEnabled, such as stopping the widget's timer.
- **onUpdate():** This method is called when the user adds a widget, so it is very important to perform the essential widget setup, such as creating the widget layout, loading the data to be displayed, or defining event handlers for some views. This event is also called every time the widget receives an update broadcast intent from the app widget manager or from the app, and at intervals defined by the `updatePeriodMillis` attribute in the widget provider info file.
- **onReceive():** This method is called for every broadcast and before each of the preceding callback methods. Its implementation has been key to being able to handle click events, such as opening the app or copying the OTP of a token, and the alarm updates, which will be explained later.
- **onAppWidgetOptionsChanged():** This is called when the widget is first placed and any time the widget is resized. One of the advantages that using this method has allowed me is to show or hide content depending on the size of the widget, which will be explained later.

Updating a widget

In each update, the widget's layout needs to be reconstructed by creating a new `RemoteViews` object and updating any data that the remote views contains, such as the list of tokens displayed, the countdown timer and setting the different pending intents. Finally, the app widget manager is notified to update the widget with the new `RemoteViews` object.

The main challenge during the development of the widgets has been to manage the updates and refreshes of the widget when necessary and trying to spend as few resources as possible. It has been quite a challenge because the tokens are automatically generated every 30 seconds and therefore the refresh intervals of the widgets are increased drastically in order to keep them up to date so they do not display incorrect information. To solve this problem, I have implemented a `TokenWidgetAlarm` class with a single instance that is responsible for managing an alarm that is scheduled every 5 seconds and sends the action to the `TokenWidgetProvider` via broadcast. The alarm is triggered when the first instance of the widget is added, in the `onEnabled()` method, so all the instances of the widget that are created are updated by the same alarm, and it stops when the last instance of the widget is removed, in `onDisabled()`, as mentioned previously. But just one alarm is not enough since the information of the tokens displayed in the widget is updated every 30 seconds, and not every 5,

because that is when it can really change, either because the new OTP is displayed or because its auto lock timeout has expired and the token has been locked, and therefore if a user deletes a token that is being displayed in a widget, when returning to the home screen the affected widget will continue to show the deleted token because the countdown time may not have started a new cycle and then the information of the tokens will not have been refreshed properly. To solve this problem, when the state of the app in React Native goes to "background" then through the native modules and the bridge, mentioned in Figure 7.2., React Native communicates with the Android native part of the app, specifically with a module that I have implemented for the widgets, and it requests to update all the widgets. In this way, the widgets are updated and if a user has modified any information related to a token, such as the name, or has deleted any, the changes will be instantly reflected in the widget.

Handling widget actions

Although there are a large number of app widgets on the market that only display information, I think that widgets should be interactive and perform actions when the widget as a whole, or a view of the widget, is clicked. That is why I have implemented several actions such as opening the app when the user clicks on the widget, opening the widget configuration screen when the user clicks on the configuration icon, copying the OTP when the user clicks on a token and it is unlocked, or opening the app if it is locked or blocked and display the authentication screen or block screen, respectively. It is also very important to have an adjusted number of actions that make sense, and not to do them just because, so as not to confuse the user and improve the user experience.

Resizing a widget

One of the user cases defined in the specification was to allow the user to adjust the size of the widget on the home screen to suit his/her preferences and available space. I accomplished this by getting the minimum width and minimum height of the widget, when the `onAppWidgetOptionsChanged()` method was called, and comparing them to a threshold I defined to determine if a widget is small or not. These minimum values were obtained through the `OPTION_APPWIDGET_MIN_WIDTH` and `OPTION_APPWIDGET_MIN_HEIGHT`, which are bundle int extras that contain the lower bound on the current width, and height, respectively, in dips, of the widget instance. In my case, I wanted to hide the content if the widget width was less than 3 cells so that only the name of the tokens would be displayed in the widget, as shown in Figure 6.5. Also, regarding the "instructions" view, it is worth mentioning that if the widget's height is less than 2 cells, the usage instructions are replaced by a simple "No tokens selected" in order that some text could be correctly displayed in the reduced space.

Using a configuration activity

Since one of the main purposes of the widget is to display the tokens selected by the user, it needs user configuration, so a widget configuration activity will also be implemented. Therefore, it is important to note that this activity allows users to modify the settings of a widget. But before it is important to declare the configuration activity class in the application's `AndroidManifest.xml` file, so I added the following code block inside the `<application>` element.

```
XML
<activity
    android:name=".TokenWidgetConfigureActivity"
    android:label="@string/widget_settings"
    android:exported="false">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE" />
    </intent-filter>
</activity>
```

Code block 7.3. Android manifest file. *Source: Own compilation.*

By default, the app widget host only launches the configuration activity once, immediately after the user adds the widget to their home screen. In order to enable users to reconfigure existing widgets, it is necessary to specify the `reconfigurable` flag in the `widgetFeatures` attribute in the widget provider info file.

The implementation of the configuration activity has a fundamental role since each widget has a personalized configuration and therefore must be saved in order not to lose the information of the tokens that the user has selected in each instance of the widget. To solve this problem, I have used Shared Preferences, one of the most interesting data storage options Android provides its users. It allowed me to store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as String, int, float, Boolean, that make up the preferences in an XML file inside the app on the device storage.

7.3. iOS

Unlike Android where widgets have been around for many years, Apple announced the introduction of app widgets for iOS (as well as iPadOS and MacOS) in 2020. With the new WidgetKit framework, it is possible to build widgets that can be added to the home screen of Apple devices in order to see important information at a glance. There are three key components to a widget:

- A configuration that determines whether the widget is configurable, identifies the widget, and defines the SwiftUI views that show the widget's content.
- A timeline provider that drives the process of updating the widget's view over time.
- SwiftUI views used by WidgetKit to display the widget.

To get started, I first added a Widget Extension target to the app. To do this I opened the app project in Xcode and picked **File > New > Target**, then I selected the Widget Extensions from the Application Extension group and clicked the Next button. Finally, I entered the data required for creating the extension, such as the product name and team, checked the Include Configuration Intent checkbox, and then clicked the Finish button.

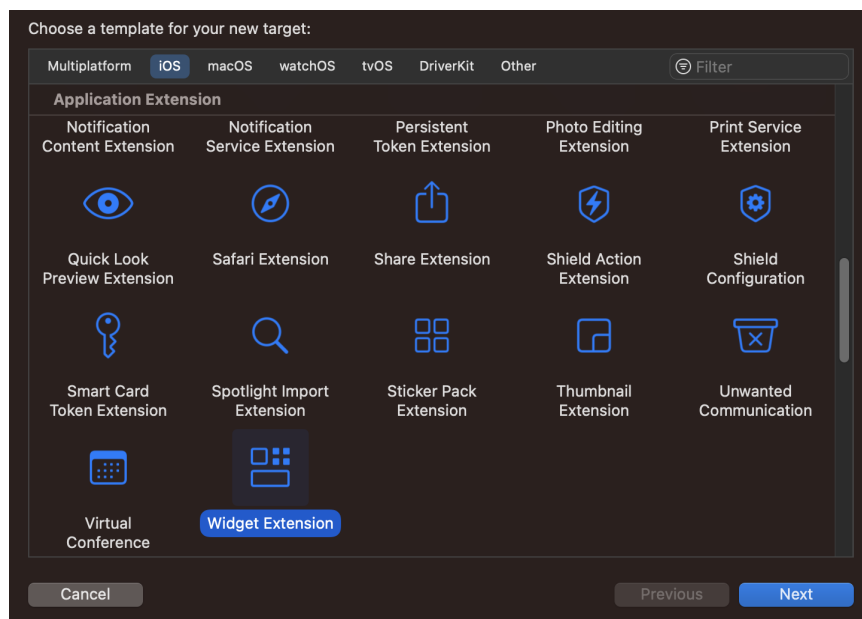


Figure 7.6. Adding a Widget Extension. *Source: Own compilation.*

The widget extension template provides an initial implementation that conforms to the Widget protocol. The block code 7.4. shows everything the operating system needs to know before adding a widget in the Apple devices. First, we find the `body` property that determines the type of content that the widget presents. As I want to give users the option to customize their widgets according to their preferences and needs, I have used an `IntentConfiguration` in the `body` property. Every widget has a unique identifier called `kind`, a string of choice. The intent in this case is a custom SiriKit intent

definition containing user-editable parameters, which will be explained later. Then, we have the `timeline provider` that is an object that determines the timeline for refreshing the widgets by providing future update dates, which will be explained later. The `content` closure contains the SwiftUI views that WidgetKit needs to render the widget's content. Finally, the modifiers specify the `name` and `description` shown in the widget gallery and the `supported families`, which are the available widget sizes presented in the widget gallery (small and medium).

```
Swift
struct TokenWidget: Widget {
    var body: some WidgetConfiguration {
        IntentConfiguration(
            kind: tokenWidgetKind,
            intent: SelectTokenIntent.self,
            provider: TokenWidgetProvider()
        ){ entry in
            TokenWidgetEntryView(entry: entry)
        }
        .configurationDisplayName("Soft Tokens")
        .description("tokenWidgetDescription".localizedStringKey())
        .supportedFamilies([.systemSmall, .systemMedium])
    }
}
```

Code block 7.4. Defining the widget. *Source: Own compilation.*

Providing timeline entries

One of the main aspects of a widget is that, in case of displaying dynamic information, they must be updated to display fresh and up-to-date data. On iOS, this is achieved via the timeline provider, mentioned before, which generates a sequence of timeline entries. Each timeline entry specifies the date and time to update the widget's content, and includes the necessary data to render in the widget's view. In my case, I included an array of Token models because it contains all the attributes of the tokens selected in the configuration intent, as well as the configuration intent itself, and a boolean indicating if the "instructions" view should be displayed or not.

Generating a preview for the widget gallery

In order for people to be able to use the widget, it must be available in the widget gallery, and for this WidgetKit asks the provider for a *preview snapshot* that displays example data via the `getSnapshot(in:completion:)` method. This preview allows users to see the type of data the widget displays and therefore it is extremely important to provide a good preview since it is the first impression that users will have of the widget, and probably the turning point on whether they decide to use it or not depending on whether it has caught their attention. So the next thing I did was create

the preview snapshot that is displayed in the gallery widget, taking into account that it must be created quickly and therefore using sample data rather than fetching information from a server or a service.

Displaying a placeholder widget

A placeholder view is similar to a preview snapshot, but instead of showing example data, it shows a generic visual representation with no specific content. This view is displayed while the widget is initializing and takes the form of the widget entry view without any data or information, as shown on the left image, and when data is fetched then the widget is updated and shows the real data, as we can see on the right image. The placeholder view has been very useful in order to increase the widget's security because I also display it when the widget is in the Today View and the phone is locked, so it is not possible to view the sensitive content of the widget till it is unlocked.



Figure 7.7. Placeholder widget. *Source: Own compilation.*

Displaying content in the widget

In iOS widget's content is defined using a SwiftUI view. Unlike Android, iOS does not allow you to choose the size of the widget freely and resize it since the sizes are previously determined. As I mentioned before, I have decided to support the small and medium sizes, and for this reason, when displaying the content in the widget, the widget's size must be taken into account to show more or less information. The 7.5. code block shows how the widget's view is defined and depending on the family size of the widget, one view or another is displayed. In addition, the boolean mentioned above that indicates whether to display the “instructions” view is also taken into account. For any other family of unsupported size, a default view is displayed indicating that the widget is not available.

```
Swift
struct TokenWidgetView: View {
    @Environment(\.widgetFamily) var family: WidgetFamily
    var entry: TokenEntry

    var body: some View {
        switch family {
        case .systemSmall:
            if entry.showInstructionsView {
```

```

        SmallInstructionsView()
    }
    else {
        SmallTokenWidgetView(entry: entry)
    }
    case .systemMedium:
        if entry.showInstructionsView {
            MediumInstructionsView()
        }
        else {
            MediumTokenWidgetView(entry: entry)
        }
    default:
        NotAvailableWidgetView()
    }
}
}
}

```

Code block 7.5. Defining the view. *Source: Own compilation.*

Responding to user interactions

User interactions with widgets in iOS, such as clicking on an element, are much more restricted because they only allow you to respond by opening the app or handling a request, but after opening the app. On the one hand, it has benefited me because clicking on the widget always opens the app, which satisfies one of the use cases. But on the other hand, when clicking on an unlocked token shown in the widget, it doesn't allow me to only copy the OTP as in Android but I'm forced to open the app to copy it, which is a bit more cumbersome. In any case, it has its positive side since it also allows the user to quickly access the token details screen, it all depends on how you look at it.

To do so, I specified a custom deep link to inform the app what content to display, using `widgetURL` and `Link`. One drawback is that small size widgets can only have one link associated with them, so I added the `widgetURL` view modifier to its whole view. However, for medium-sized widgets I have associated several deeplinks with the `Link` struct, so depending on the area where the user interacts, a different deep link will be processed.

8. Testing

The activity of software testing can be traced back down to the very beginning of computing, when in the 40's and 50's programmers manually inspected the outputs for ensuring their correctness. In the 60's, the concept of identifying and fixing defects in software emerged as “debugging” and started to be referred to as part of the software development process. In the 70's and 80's, as programs grew larger and complexity scaled, development methodologies evolved along and testing got its own section as a separate concept for ensuring correctness and fitness of purpose. Today, software testing is recognized as an important enough discipline to have its own department in almost every business that offers technological value, the Quality Assurance department.

Testing is usually divided in various levels depending on a combination of what their final goal and scope are. Some of the most known and widely accepted in the industry today are:

- **Unit Testing:** Focused on verifying the very individual components that build up an application. Functions, methods and classes are tested here in isolation from the context they work with in order to ensure they behave as expected alone in order to later be integrated into a larger system.
- **Integration Testing:** Integration testing is usually one level above Unit Testing as it focuses on verifying that the integration of the components that build an application work together as expected. Here, through system interfaces and using interactions the correct handling of the data along the flow of the system is verified.
- **System Testing:** System testing verifies that the complete and fully integrated software systems behave as they were expected to. Functionality, performance, reliability, scalability and security are some of the most important aspects that systems have to be taken care of.
- **Acceptance Testing:** In testing, not only technical aspects are verified. User expectations, business objectives and stakeholder requirements are some other software needs that need to be accepted in order to classify a system as successful.

8.1. Testing widgets

In my project, the original idea was to have some kind of testing that ensured in a programmatic way the correct behavior of the widgets, but after an extensive research of different the applications and technologies used for widget testing and taking into consideration my company policies regarding testing, I have decided to stick my widgets to the manually tested components in the application.

Most of the testing tools I found, such as UIAutomationViewer [21] for Android or the XCode tools for debugging [22] need expertise training, specific setup installations and specific language features that would be impossible to maintain for someone without the previously acquired knowledge.

As a result, after considering all the high burdens adding this testing framework and tools would introduce to the project, and moreover taking into consideration the design simplicity that my widgets aim for, introducing testing would make no sense. Following the company guidelines, with the simple use cases and test games this document provides, the widgets can be checked for consistency and correctness from top to bottom without the impediments the previously mentioned technologies would imply.

9. Experimentation and analysis

In the experimentation and analysis phase, a usability test will be carried out with the aim of evaluating how easy and efficient it is to use the developed widgets, identifying areas in which users may have difficulties or encounter frustration, and discover possible improvements to create a better user experience.

9.1. Usability testing

Usability testing is a user research method that evaluates the user's experience when interacting with a website or app in order to assess its intuitiveness and ease of use. It involves asking real users to perform tasks on the product, analyzing their success rate and paths taken, and identifying potential issues and areas for improvement, with the aim to create a user-friendly product that addresses user needs and provides a positive experience.

Before doing a usability test, it is important to take into account the different types of methods to select the one that best suits the product. There are six types of usability tests that can be grouped into the following three categories:

- **Qualitative or quantitative:** Usability testing can gather qualitative data, focusing on understanding users' experiences, feelings, and thoughts, through observation, interviews, and surveys. Quantitative data, on the other hand, involves collecting and analyzing numerical data such as success rates, task completion times, error rates, and satisfaction ratings.
- **Moderated or unmoderated:** Moderated usability testing involves a moderator guiding users through the test, answering questions, asking follow-up questions, and recording observations. However, unmoderated usability testing does not involve a moderator and users complete tasks independently using testing tools that record their actions and responses.
- **Remote or in-person:** Usability testing can be conducted remotely or in-person. Remote testing can be moderated or unmoderated and utilizes online tools or software to facilitate user participation from different locations. In-person testing is conducted in a physical location, such as a usability lab, and allows for direct supervision and physical testing.

I have decided that the widgets usability test will gather qualitative data and quantitative data, and it will be unmoderated and remote, which I justify on the next page.

Next, I will explain the steps that I have followed to conduct the usability test, which are summarized in the figure below.

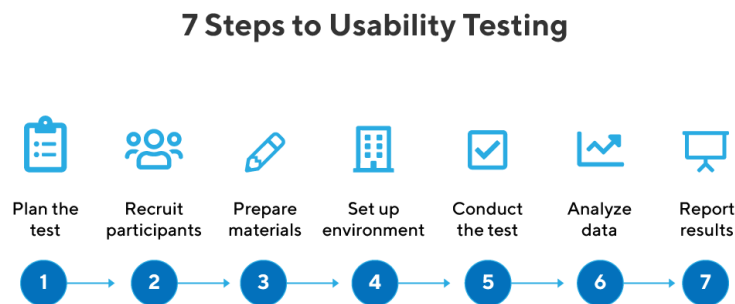


Figure 9.1. Steps to usability testing. *Source:* [32].

1. **Plan the test:** The objective of the usability test will be to evaluate the user experience and the ease of use of the widgets, as well as identify usability issues and gather user feedback. The test will focus on the ease of navigating through different widget features and completing common tasks, such as choosing the token to display in the widget and copying the OTP. The scope will include both qualitative insights on user satisfaction and quantitative metrics like task completion time and success rates.
2. **Recruit participants:** Participants will be recruited from the target user group, which consists of people over the age of 18 who have an Android or iOS device, regardless of their level of experience with widgets in general or their knowledge of the mobile app. A total of 15 participants will be selected to ensure diversity and thus gather a representative sample.
3. **Prepare materials:** I will use the Maze platform to create different test scenarios, such as "Imagine that you wanted to add a widget on the home screen with all your tokens and copy the OTP of one of the tokens" and "Reconfigure a widget". Maze is a user testing platform that uses clickable and interactive prototypes to get actionable insights from real users, and is one of the most popular tools for conducting usability testing. I have chosen this online tool because it provides multiple benefits, such as facilitating user participation from entirely different locations. I will explain to participants that the tasks and missions they will be performing have been created using Maze and I will inform them that their actions and feedback will be recorded and analyzed.
4. **Set up the environment:** Because the usability test has been created with Maze, it will be done remotely. Therefore, when preparing the test environment I will make sure that it will be done with a properly working computer and good network connectivity to avoid problems.
5. **Conduct the test:** The usability test will be unmoderated, but before conducting the test I will explain the purpose of the test and an overview of what it is about. After completing the tasks, I will encourage them to express their thoughts, any difficulties encountered, and feedback about their overall experience.

6. **Analyze the data:** The collected data will include registered paths, task completion times, success and misclick rates, and participants' feedback. Each task's paths will be reviewed to identify common usability issues, such as confusing widget features or unclear instructions.
7. **Report results:** I will summarize the findings and insights from the usability test in a report including key observations, identified usability issues, and recommended improvements.

9.2. Widget usability test

In this section, I will present in detail the usability test that I have created for the widgets using Maze, explaining the different blocks that conform the test and the goal of each question.

The widget usability test consists of a total of 12 blocks. The first block is the Welcome screen, which contains a message for the test participants. Next, we find the Context screen, shown in Figure 9.2., where I briefly explain what it is about. As you can see, the text of the test is displayed in two languages, English and Spanish, to avoid any kind of misunderstanding.

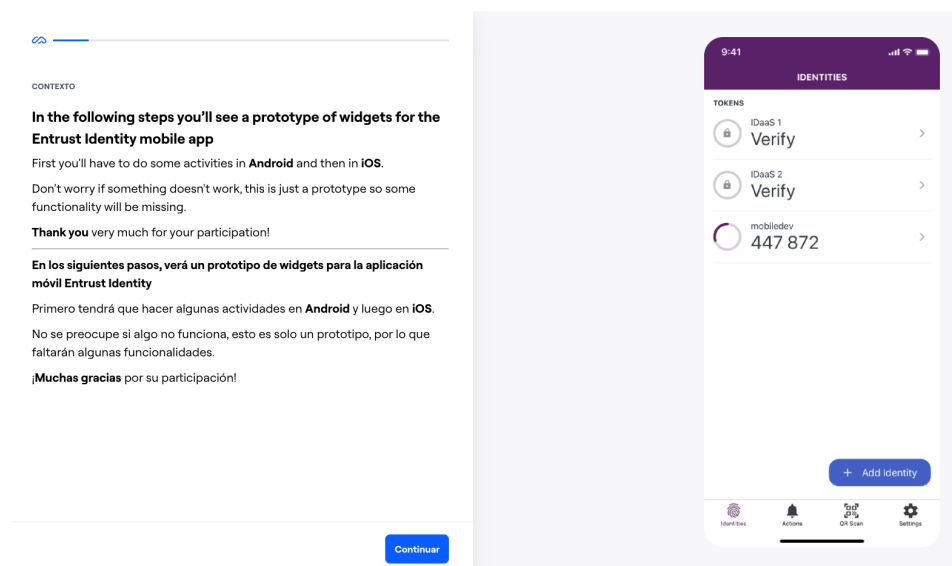


Figure 9.2. Context screen. *Source: Own compilation.*

The first question that appears says “Have you used the Entrust Identity app before?”. I have decided to ask this question because that way, in the results, I will be able to compare the success rate of a user who has previous knowledge about the app and knows its general concepts with another who has never used it.

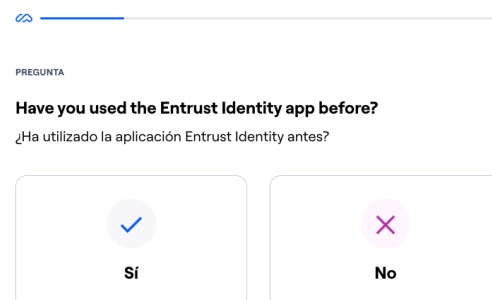


Figure 9.3. Question to know if participants have used the app. *Source: Own compilation.*

Then, the different missions that I created in stage “3. Prepare materials” are presented, which are a total of 5 tasks. The first 3 tasks are about Android widgets and the next 2 tasks are about iOS widgets. The first mission that the user will have to carry out, shown in Figure 9.4., is presented “In Android, imagine that you wanted to add a widget on the home screen with all your tokens and copy the OTP of one of the tokens”, along with some steps to guide the user, but without explicitly ordering or telling him what to do.

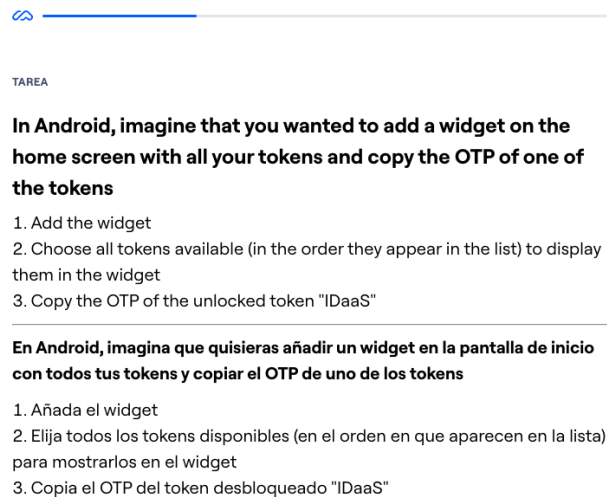


Figure 9.4. Mission 1 Android. *Source: Own compilation.*

Figure 9.5. shows the navigation flow of the screens that the user should follow to complete the task successfully, which consist of adding the widget on the home screen, selecting all the tokens shown in the list of the Widget settings screen, and finally clicking on the unlocked token names “IDaaS” with the goal to copy its OTP on the clipboard.

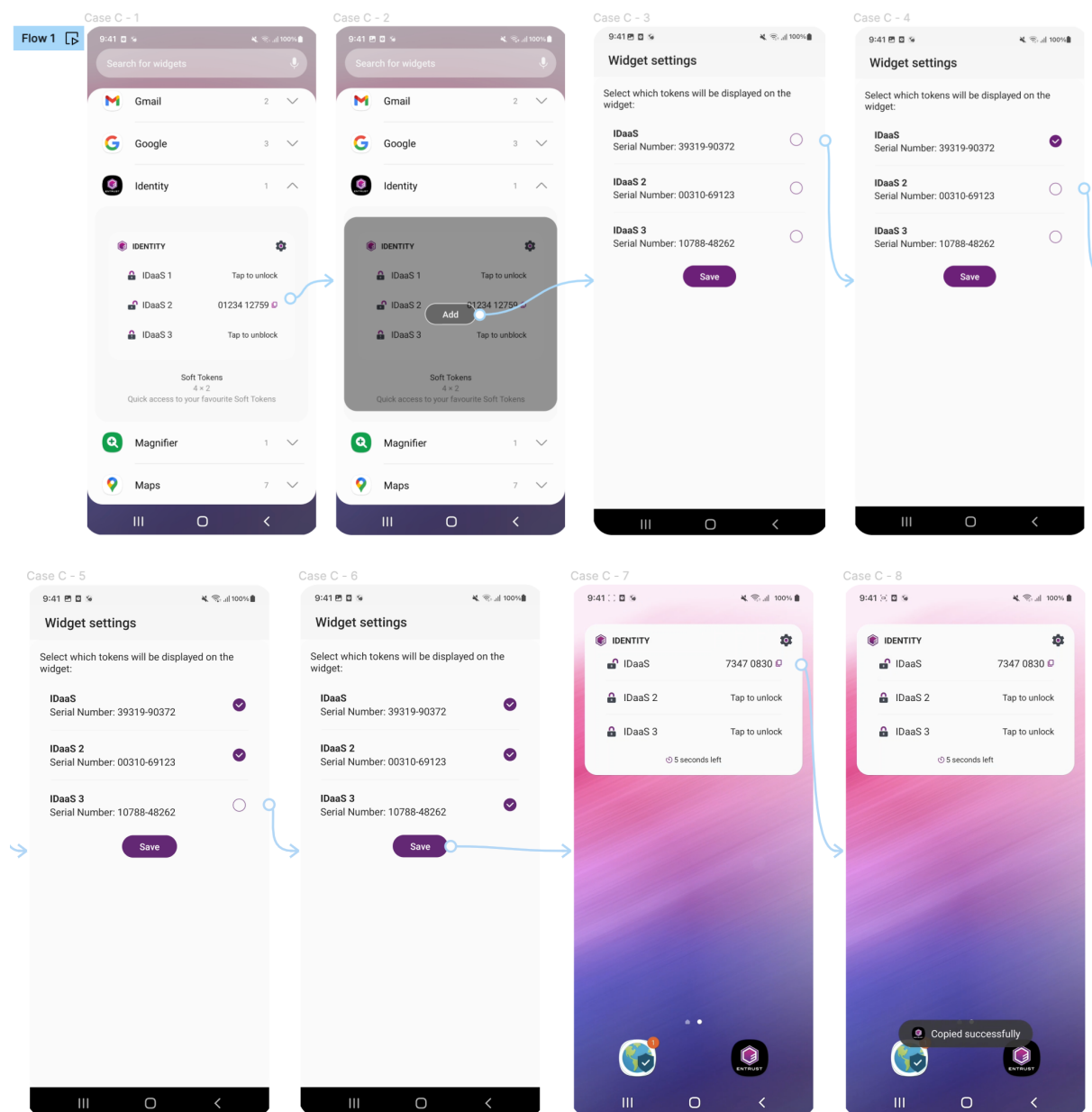


Figure 9.5. Screen navigation flow Mission 1 Android. *Source: Own compilation.*

The second mission says: Imagine that you wanted to copy the OTP of the locked token “IDaaS 3”. The goal of this task is for the user to figure out intuitively how to copy the OTP of a locked token. Figure 9.7. shows the screen navigation flow that the user should follow to complete the task successfully.

TAREA

Imagine that you wanted to copy the OTP of the locked token "IDaaS 3"

Please show the steps you would follow to do it.

Imagina que quisieras copiar el OTP del token bloqueado "IDaaS 3"

Por favor, muestre los pasos que seguiría para hacerlo.

Figure 9.6. Mission 2 Android. *Source: Own compilation.*

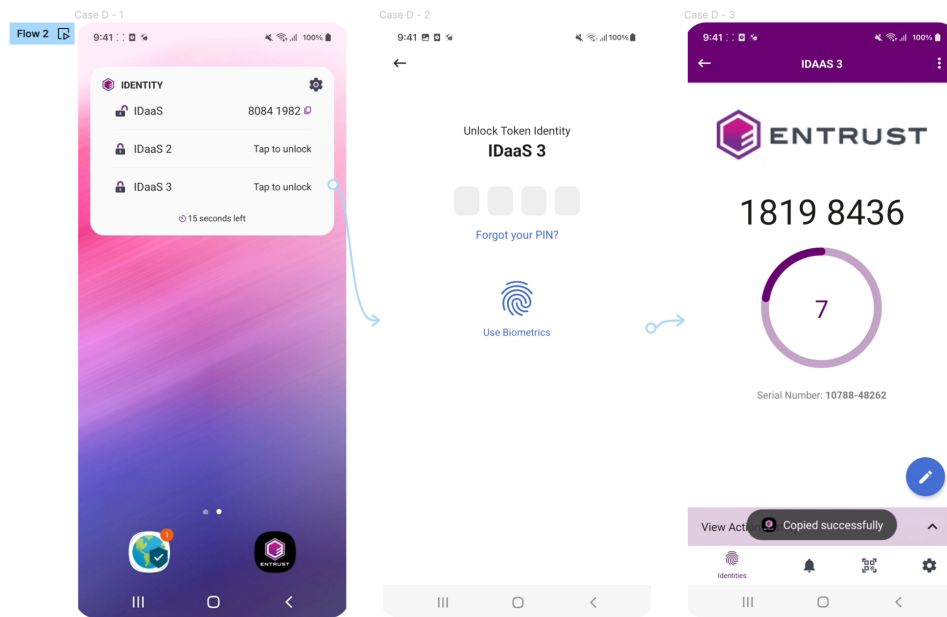


Figure 9.7. Screen navigation flow Mission 2 Android. *Source: Own compilation.*

Third mission, and the last one regarding Android widgets, is about reconfiguring a widget in order to change the tokens that are being displayed. The user should be able to do it by the description provided in figure below. The screen navigation flow that the user should follow to complete the task successfully is shown in Figure 9.9.

TAREA

Reconfigure a widget

Imagine that now you would like to save space on your home screen...

Please show the steps you would follow to reconfigure the widget to only show the "IDaaS" token.

Reconfigure un widget

Imagina que ahora te gustaría ahorrar espacio en tu pantalla de inicio...

Por favor, muestre los pasos que seguiría para reconfigurar el widget para mostrar solo el token "IDaaS".

Figure 9.8. Mission 3 Android. *Source: Own compilation.*

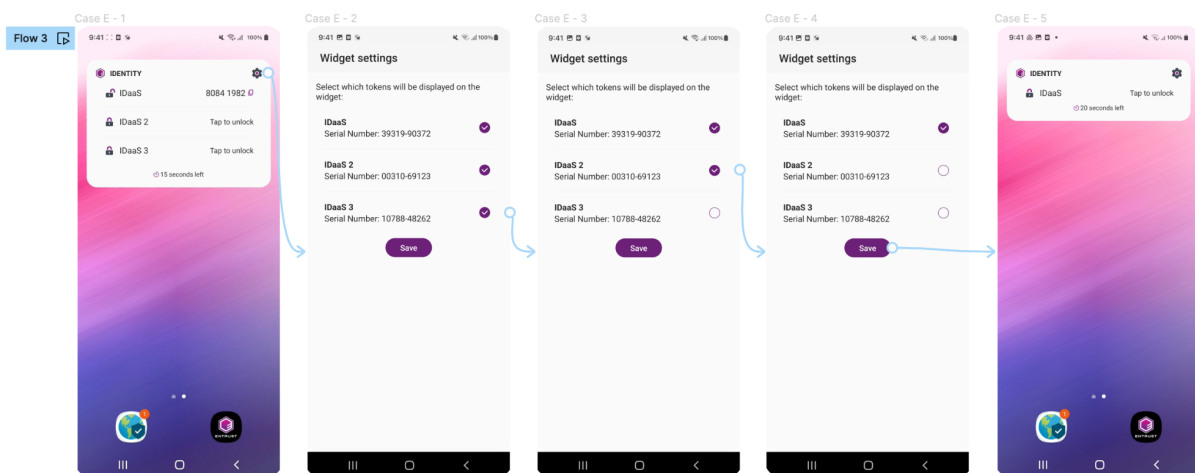


Figure 9.9. Screen navigation flow Mission 3 Android. *Source: Own compilation.*

The next block is about the first mission about iOS widgets, which says “In iOS, imagine that you wanted to add a small widget on the home screen to be able to copy the OTP of a token”. It is the same test scenario described in Mission 1 for the Android widget, but with a very different navigation flow, shown in Figure 9.12, and presenting the widget family in iOS.

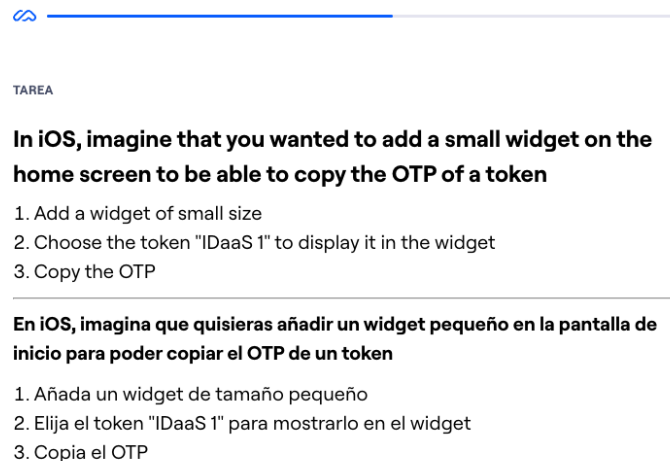


Figure 9.10. Mission 1 iOS. *Source: Own compilation.*

The second, and last prototype test block, says “Imagine that you wanted to add a medium widget on the home screen with all your tokens and copy the OTP of one of the tokens”. The aim of this mission is to evaluate if the user has learnt how to set up an iOS widget, due to its complexity compared to Android if you have not done it before. The screen navigation flow that the user should follow to complete the task successfully is shown in Figure 9.13.

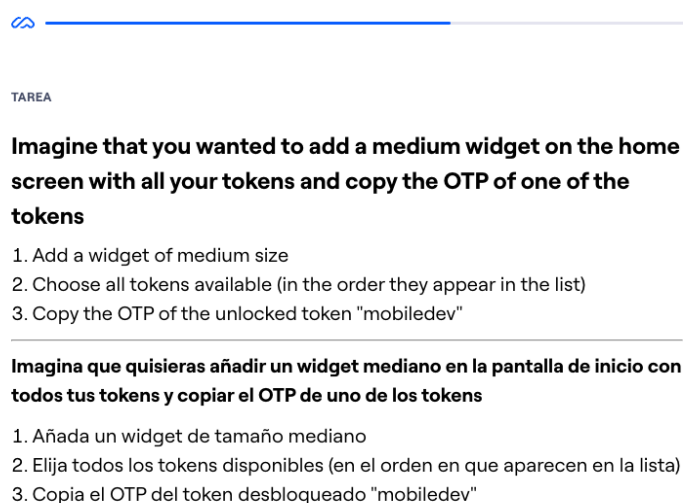


Figure 9.11. Mission 2 iOS. *Source: Own compilation.*

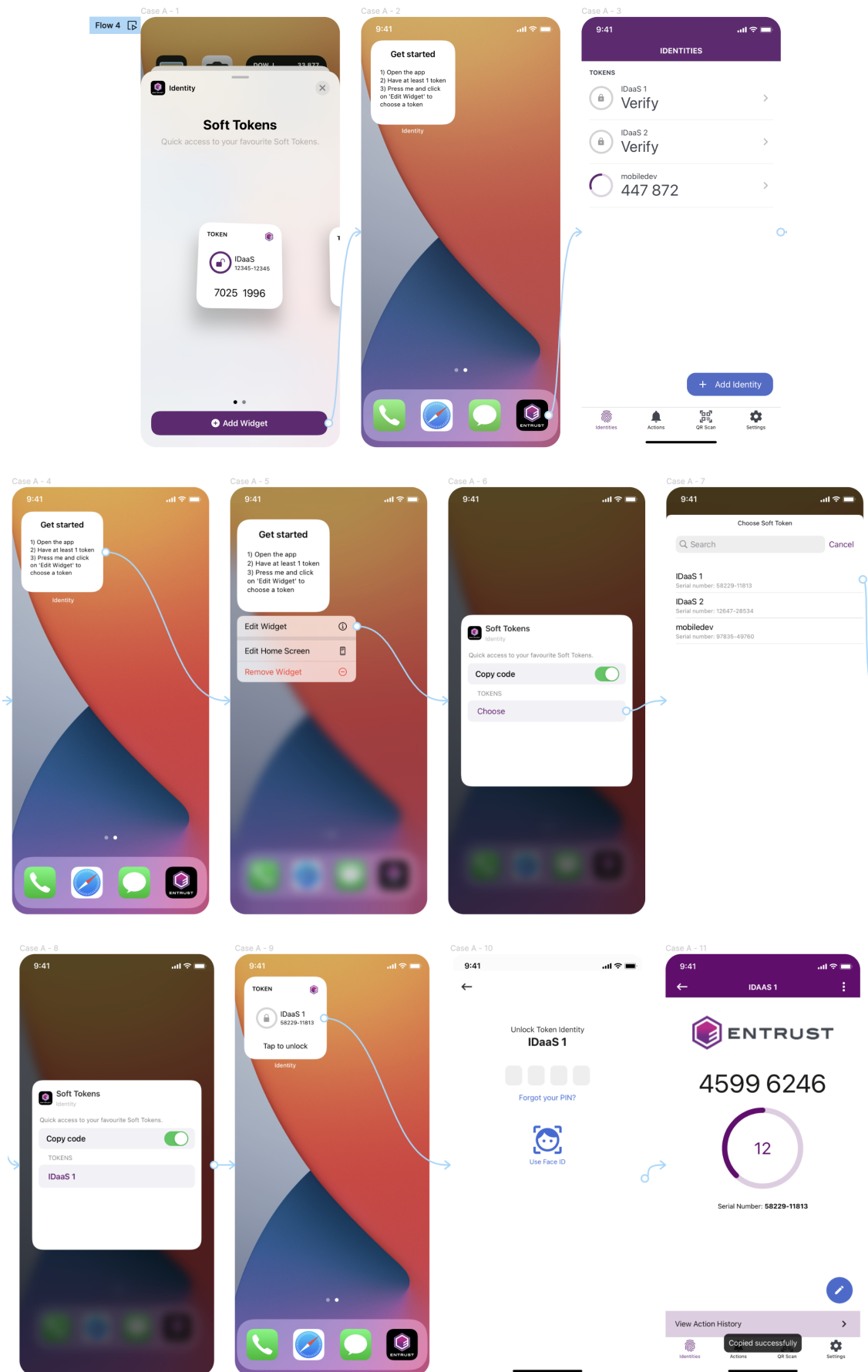


Figure 9.12. Screen navigation flow Mission 1 iOS. *Source: Own compilation.*

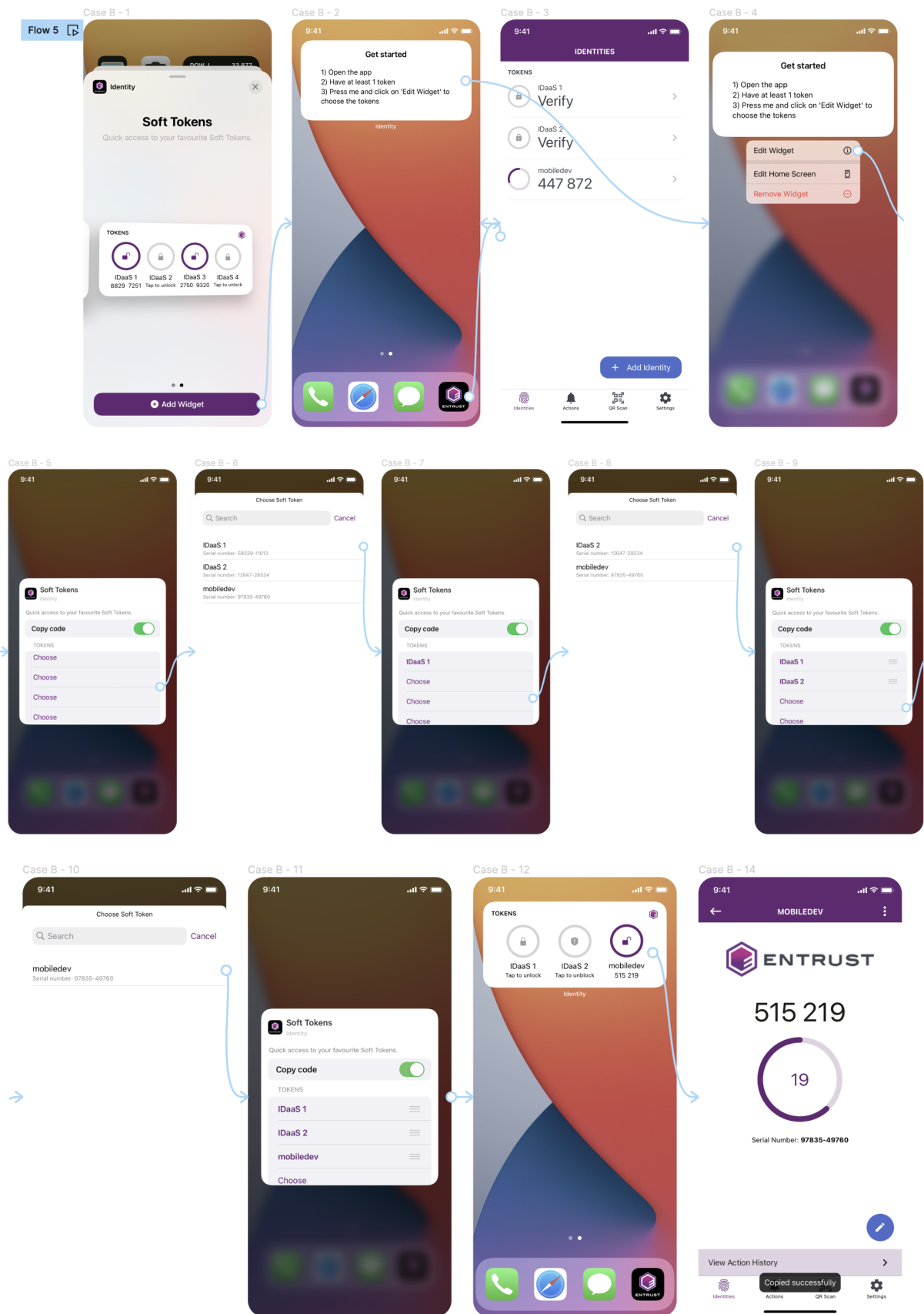



Figure 9.13. Screen navigation flow Mission 2 iOS. *Source: Own compilation.*


At this point, the user has already finished all the tasks, successfully or not. Then, they are presented with three blocks of questions in order to obtain qualitative data in the usability test, such as user satisfaction, ease of use, and user feedback. The three blocks with their respective questions are shown below.




PREGUNTA

Did the experience meet your expectations?

¿La experiencia cumplió con tus expectativas?





PREGUNTA

How did you find the ease of use?

¿Cómo encontraste la facilidad de uso?

1

2

3

4

5

6

7


8

9

10

Difficult
/ Difícil

Easy
/ Fácil



PREGUNTA

What could be improved?

Do you think the UI design is more attractive on Android or iOS? What did you like the most/least? Any other feedback you'd like to add?

Please indicate whether you use widgets on a daily basis and whether your personal mobile phone is Android or iOS.

¿Qué se podría mejorar?

¿Crees que el diseño de la interfaz de usuario es más atractivo en Android o iOS? ¿Qué es lo que más/menos te ha gustado? ¿Algún otro comentario que quieras añadir?

Por favor, indique si utiliza widgets en su día a día y si su teléfono móvil personal es Android o iOS.

Escribe tu respuesta aquí

Figure 9.14. Qualitative data questions. *Source: Own compilation.*

Finally, the last block contains the Thank You screen, which contains a message to thank people for participating in the usability test.

9.3. Results

The usability test of the widgets has been conducted by a total of 15 participants, out of which 8 had previously used the application and the remaining 7 were unfamiliar with it. Therefore, 53% of the testers had prior knowledge of the app, while everything was new for the remaining 47%. As for the operating system of the users' devices, 67% of the users have the iOS operating system, while 33% use Android. Additionally, users were asked if they use widgets on their mobile phones in their daily lives, and the collected results have surprised me a lot. They indicate that only 53% of the users have daily interaction with the widgets, while 47% do not have it.

On the one hand, I have obtained quantitative data, such as task completion time, success and misclick rates. A usability score has been calculated in Maze for each mission with the following formula:

$$usability\ score = avg(MIUS)$$

where **MIUS** means Mission Usability Score and is calculated with the following formula:

$$MIUS = DSR + (IDSR / 2) - avg(MC_P) - avg(DU_P)$$

which has these variables:

- **DSR** for Direct Success Rate
- **IDSR** for Indirect Success Rate
- **avg** for the average
- **MC_P** for misclick penalty = $MCR * 0.5$
- **MCR** for misclick rate
- **DU_P** for duration penalty = $(MIN(10, MAX(0, (AVGD - 5)/2)))$
- **AVGD** for Average Duration in seconds

In Mission 1 on Android, the majority of testers completed the mission via the expected path, an 86.7%, and a usability score of 82 out of 100 has been achieved.

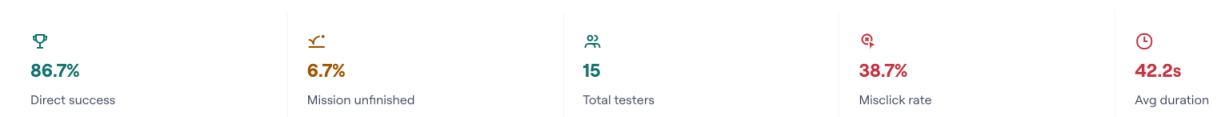


Figure 9.15. Performance indicators Mission 1 Android. *Source: Own compilation.*

In Mission 2 on Android, we can observe a great change in the performance indicators since 100% of the testers have completed the task successfully and the number of misclicks is much lower. A usability score of 94 out of 100 has been achieved. This is because the user has more easily intuited

how it works through the experience gained from the previous mission. It also influences that the task is much shorter than the previous one.



Figure 9.16. Performance indicators Mission 2 Android. *Source: Own compilation.*

In Mission 3 on Android, the majority of testers completed the mission via the expected path, almost 93%, and a usability score of 84 out of 100 has been achieved.



Figure 9.17. Performance indicators Mission 3 Android. *Source: Own compilation.*

In Mission 1 on iOS, a significant number of testers left the expected path, almost 67%, and a usability score of 70 out of 100 has been achieved. After analyzing the mission's path of each user, I have been able to conclude that users with an Android device have had a worse performance and, therefore, this is due to the fact that they are not used to the environment and to the interactions typical of iOS. Also, I've seen similar behavior from users who have confirmed that they don't use widgets regularly.



Figure 9.18. Performance indicators Mission 1 iOS. *Source: Own compilation.*

In Mission 2 on iOS, the majority of testers completed the mission via the expected path, an 93.3%, and a usability score of 92 out of 100 has been achieved. We can observe a drastic change in performance indicators. This is because the user has more easily intuited its operation thanks to the experience acquired in the previous mission.



Figure 9.19. Performance indicators Mission 2 iOS. *Source: Own compilation.*

On the other hand, I have obtained qualitative data that allows me to measure two main aspects: user satisfaction and ease of use. User satisfaction is measured alongside the results from the question "Did

the experience meet your expectations?" where participants freely rated their response on a scale from 1 to 10. The results obtained are as follows: 43% of the total participants rating it 9 and 57% rating it 10, which means that the average score is 9.6 out of 10.

Regarding ease of use, good results have also been obtained, although not as high as in the previous case. When asked "How did you find the ease of use?" I received more varied responses. In this case, 40% of the participants rated it 8, 20% rated it 9, and 40% rated it 10, resulting in an average score of 9 out of 10.

Finally, it is worth mentioning that I included a final open-ended question for participants to comment on any thoughts, doubts, or frustrations that came to mind during the test, as well as providing feedback for improvement. It is important to note that the feedback received has been excellent as it allowed me to empathize with the users and understand possible issues that I had not considered, despite being very familiar with the project, such as the "instructions" screen. Additionally, the comments reflected a sense of satisfaction from the participants, which made me reflect on the fact that widgets have a promising future.

10. Legal and regulatory compliance

Laws and regulations are legal instruments that establish rules and guidelines to regulate various aspects of society. In the broad field of technology it is of utmost importance to comply with relevant laws and regulations because they not only guarantees the protection of the rights and privacy of users, as well as maintaining the confidentiality of sensitive information, but also helps to maintain the integrity of the systems, promote public trust and avoid potential fines, penalties and legal consequences.

The development of widgets for a security corporate mobile application of a big company involves dealing with a large number of personal and sensitive data, information security and user privacy. Therefore, it is essential to identify and comply with the laws and regulations that govern these aspects, as well as the mandatory regulations established by the university.

10.1. Identification of applicable laws and regulations

Next, the main laws and regulations that affect the project are identified, along with a brief description of its definition and how they impact the project.

- **General Data Protection Regulation (GDPR):** Regulates the processing and protection of personal data, requiring explicit user consent, the implementation of appropriate security measures, and the provision of user rights regarding their data [23]. Since the widgets will be available in a mobile application that targets users in the European Union (EU) and collects personal data from EU citizens, compliance with the GDPR is crucial.
- **California Consumer Privacy Act (CCPA):** The CCPA is a national law in the United States of America that grants California residents certain rights over their personal data and establishes requirements for companies that collect and process such data [24]. Additionally, due to the fact that the widgets will be available in a mobile application also targeted at users in the United States of America and will likely collect personal data from California citizens, CCPA compliance is critical.
- **Intellectual property (IP):** Guarantees legal frameworks to safeguard various forms of intellectual property, such as patents, copyrights and trademarks [25]. Intellectual property (IP) laws play a crucial role in protecting the rights of creators and innovators in the field of technology and software. Therefore, it is extremely important not to infringe any existing intellectual property, including logos, trademarks, or copyrighted content.
- **End User License Agreement (EULA):** The EULA is a legal contract between a software supplier and a customer or end-user, generally made available to the customer via a retailer

acting as an intermediary [26]. The Entrust Mobile Application End User License Agreement specifies the rights and restrictions which apply to the use of the software, which includes the use of widgets.

- **Apple and Google Privacy Rules:** Both Apple and Google require apps to adhere to their privacy rules, which include seeking user consent to collect and process personal data, providing a clear and transparent privacy policy, obtaining permission of the user before accessing certain functions, among others. Therefore, it is essential to review and comply with these rules to ensure that the mobile application will be maintained in the app stores.
- **App Store Guidelines:** Both Apple's iOS and Google's Android platforms have specific guidelines and requirements that developers must adhere to when submitting apps or widgets to their respective app stores (App Store and Play Store). These guidelines cover various aspects, including content restrictions, privacy policies, user interface design, advertising regulations and security measures. Consequently, it is fundamental to review and comply with these guidelines to ensure acceptance and availability in the app stores.

10.2. Academic regulations

Academic regulations are the set of rules and guidelines established by educational institutions to govern various aspects of academic life, including student, faculty, and staff behavior, academic performance, assessment procedures, and overall academic integrity. These regulations serve as a framework to ensure the consistency, integrity and quality of the work produced by students and faculty. Due to the fact that the project is developed within a university, it must comply with the academic standards established by the institution, which include the general academic regulations of the UPC and the specific academic regulations of the FIB-GEI.

UPC general regulations

The academic regulations of the Universitat Politècnica de Catalunya consist of the Governing Council Decision CG/2022/04/02, of 24 May 2022, which approves the Academic Regulations for Bachelor's and Master's Degrees at the UPC, in the 2022-2023 academic year [27]. The set of regulations governing bachelor's and master's degrees is complemented by the regulation corresponding to the verification reports or the monitoring and accreditation of the degrees. The actions of staff and students are also regulated by the provisions of the Code of Ethics of the Universitat Politècnica de Catalunya (Governing Council Decision CG/2022/02/30, of 22 February), as well as current legislation regarding the rights of people with educational support needs and victims of gender-based violence and/or sexual harassment.

FIB-GEI specific regulations

The academic regulations of the Bachelor Degree in Informatics Engineering at the Facultat d'Informàtica de Barcelona consist of the Permanent Commission of the FIB Decision CP.FIB/2022/06/05, of July 1, 2022, which approves the Academic Regulations for the Bachelor Degree in Informatics Engineering [28].

Considering that this project is part of a final degree thesis, it must strictly comply with the Academic Regulations for the Bachelor Thesis of the Bachelor Degree in Informatics Engineering at the FIB, approved by the Permanent Commission on 19/09/2012 and subsequently modified in several sessions [29].

Additionally, due to the fact that the project is of type "Modality B", because it is a project carried out in a company through an Educational Cooperation Agreement, then it must comply with the regulation of external academic practices of the degree, based on the Permanent Commission of the FIB Decision CP.FIB/2022/06/06, of July 1, 2022 [30].

12. Conclusions

After conducting extensive research on the development of application widgets for Android and iOS platforms, learning their characteristics, types of applications, advantages and disadvantages, as well as limitations and restrictions of the operating system itself, I can conclude that the initial objectives have been successfully achieved and that the integration of widgets in the mobile application would provide a large number of benefits.

The developed widgets allow users to access important information and functionality at a glance from their device home screen. Moreover, they improve the user experience by allowing users to easily interact with one of the most popular functionality of the app without having to open the full application, the use of soft tokens. Therefore, the accessibility and convenience that the widgets provide will contribute to enhancing user engagement and satisfaction because users will be able to quickly access the information they want to see without even having to search the app icon on their home screens to open the app.

As mentioned in previous sections, throughout the development some issues have arisen with third-party libraries and also limitations and restrictions that the operating system establishes in the development of widgets for its specific platform, mainly highlighting iOS. But not all have been negative aspects, since it has been quite a challenge to understand the problem and very rewarding finding a solution, or workaround. Definitely, this project has allowed me to see a real mobile application project firsthand, work in real scenarios, understand the life cycle of a software project, learn and know how to deal with problems that may arise, and much more. Moreover, platform-specific considerations have been present during the project, specially in the design phase where I had to consider the design guidelines and principles specific to each platform in order to ensure that the widgets look and feel correctly adapt to the different widget frameworks, sizes and layout patterns, in order that they blend seamlessly with the overall user experience, providing a consistent experience for users.

Furthermore, the widgets offer an opportunity to customize and personalize the user experience. Users can choose the size, layout, and content of the widgets based on their preferences and needs, which enhance user satisfaction because a personalized experience is provided to them and increases the competitiveness compared to other applications.

In conclusion, widgets can provide a very positive added value to the mobile app because they will reduce the amount of time and effort required to perform common tasks, which will improve the user experience, and also increase engagement and usage of the mobile application and, consequently, a competitive advantage in the huge mobile application market.

12.1. Integration of knowledge

Throughout my university degree, the learning I have acquired has provided me with a solid foundation of knowledge and skills in specific areas. However, the benefits go beyond mastering a single discipline. This has allowed me to approach complex problems from different perspectives, develop critical and creative thinking, generate more complete and innovative solutions, strengthen my collaboration and teamwork skills, and much more. Consequently, it has enabled me to successfully integrate knowledge from various disciplines in my project.

Starting with programming subjects, such as Programming I (PRO1), Programming II (PRO2) and Data Structures and Algorithmics (EDA), where I learned from the fundamental concepts, such as understanding the process of building a program and familiarizing myself with the necessary tools (console, editor and compiler), to evaluating the efficiency of different algorithms to solve the same problem and selecting the most appropriate option. Therefore, these subjects have provided me with the necessary skills to develop programs in an effective and optimized way, present throughout the development phase of the project.

Continuing with the Introduction to Software Engineering (IES) subject which gave me an overview of software engineering, and was a major influence on my choice to specialize in it, has allowed me to use a logical layered structure and apply design patterns effectively. Moreover, the Interaction and Interface Design (IDI) subject deals with two fundamental and very present topics in my project: interface design and the study of its usability and graphic interaction. I have been able to apply knowledge and skills in the creation of effective and visually attractive interfaces, focusing on aspects such as information architecture, visual design and intuitive interaction. Furthermore, it has allowed me to evaluate the usability of interfaces using techniques such as user testing, heuristic analysis and performance metrics.

Then, in the Programming Project (PROP) subject, I had the opportunity to put into practice the knowledge I gained from previous courses while working on a Java programming project. This hands-on experience allowed me to apply appropriate algorithms and data structures to build a correct

and efficient program. All these skills have been very useful in this project, particularly the proficiency in working with the Java programming language, which also plays a significant role in my project.

Ending with the compulsory subjects of the Software specialty, which are undoubtedly the ones I have enjoyed the most, such as Requirements Engineering (ER) where I learned to determine what system needs to be built and has been key to defining the requirements and objectives of the project, as well as identifying stakeholders and understanding their interests and needs.

In the Software Architecture (AS) subject, I learned how to design and implement software systems using a layered and object-oriented architecture and using software services, starting from the specification of their requirements, and it has allowed me to apply this set of techniques, principles and patterns design in the project. Its predecessor, the Web Applications and Services (ASW) subject, provided me with a deep, critical and systematic knowledge of the most important technologies for the development of modern web applications and services, where I acquired an overview of the wide range of languages, techniques and tools available for the development of web applications and services. This subject has also been very useful to me and has somehow affected the project by acquiring more knowledge about this ecosystem.

Regarding Software Project Management (GPS), it should be noted that it has helped me to have an overview of what a software project is and what are the complexities associated with its management, as well as to know which factors affect the cost structure, which are the risk factors and the feasibility of a project, and being able to elaborate the planning of a software project.

And last but not least, we find the subject Software Engineering Projects (PES) where, together with a team of 6 students, we reproduced the development activity of a mobile application project focused on sustainable mobility as it occurs in a professional environment, that is, around a project team with differentiated roles, and taking into account all aspects of project management: planning, costs, deadlines, deliverables, daily meeting, oral presentations... By participating in this hands-on experience, we gained a comprehensive understanding of the complexities and challenges that arise in real-world software development projects, which has proven to be very beneficial to me throughout my internship at the company.

12.2. Future work

Although significant progress has been made in the development of widgets, there are several areas that warrant further exploration and enhancement. The following future work can be considered to improve the functionality and user experience of our widgets:

- **Interactive Widgets in iOS 17:** Apple announced that iOS 17 would bring a significant improvement and new features in the widget world. This could be a great opportunity to increase competitiveness in the market. Explore the integration of interactive features within widgets to provide users with more dynamic and engaging experiences.
- **Widget Suggestions:** Implement intelligent algorithms to analyze user behavior and app usage patterns to provide personalized widget suggestions. By understanding user preferences and habits, the app can recommend relevant widgets that align with their interests, saving them time and effort in discovering and setting up widgets manually.
- **Widget Analytics:** Incorporate analytics capabilities to gain insights into widget usage, such as the number of times a widget is viewed or interacted with. These analytics can help identify popular widgets, track user engagement, and guide future widget development and improvements.

By addressing these future work areas, I can further refine the widgets and enrich the user experience. Continual innovation and attention to user needs will ensure that our widgets remain valuable and indispensable tools for our users in the ever-evolving mobile landscape.

Bibliography

- [1] Wikipedia contributors. (2023, February 10). Software widget. Wikipedia. https://en.wikipedia.org/wiki/Software_widget
- [2] Entrust. (n.d.). Securing Identities, Payments, and Digital Infrastructure. <https://www.entrust.com>
- [3] Identity and Access Management. (n.d.). <https://www.entrust.com/digital-security/identity-and-access-management>
- [4] Entrust Identity - Apps on Google Play. (n.d.). <https://play.google.com/store/apps/details?id=com.entrust.identityGuard.mobile>
- [5] Cambridge Dictionary. (2023). best practice definition: 1. a working method or set of working methods that is officially accepted as being the best to use... Learn more. <https://dictionary.cambridge.org/dictionary/english/best-practice?q=best+practices>
- [6] Cambridge Dictionary. (2023). contingency plan definition: a plan that is made for dealing with an emergency, or with something that might possibly happen and... Learn more. <https://dictionary.cambridge.org/dictionary/english/contingency-plan>
- [7] Cambridge Dictionary. (2023). graphical user interface definition: 1. a way of arranging information on a computer screen that is easy to understand and use because... Learn more. <https://dictionary.cambridge.org/dictionary/english/graphical-user-interface>
- [8] Cambridge Dictionary. (2023). kanban definition: 1. a system of communication between workers during a manufacturing process, in which they order... Learn more. <https://dictionary.cambridge.org/dictionary/english/kanban>
- [9] Richards, K., & Wigmore, I. (2021, September 29). one-time password (OTP). Security. <https://www.techtarget.com/searchsecurity/definition/one-time-password-OTP>
- [10] Wikipedia contributors. (2023, February 26). Public key infrastructure. Wikipedia. https://en.wikipedia.org/wiki/Public_key_infrastructure

- [11] Wikipedia contributors. (2023, February 2). Software token. Wikipedia. https://en.wikipedia.org/wiki/Software_token
- [12] Wikipedia contributors. (2023, January 21). Swift (programming language). Wikipedia. [https://en.wikipedia.org/wiki/Swift_\(programming_language\)](https://en.wikipedia.org/wiki/Swift_(programming_language))
- [13] Wikipedia contributors. (2023, February 23). Version control. Wikipedia. https://en.wikipedia.org/wiki/Version_control
- [14] Cambridge Dictionary. (2023). widget definition: 1. any small device whose name you have forgotten or do not know 2. an imagined small product made... Learn more. <https://dictionary.cambridge.org/dictionary/english/widget>
- [15] Cambridge Dictionary. (2023). workflow definition: 1. the way that a particular type of work is organized, or the order of the stages in a particular... Learn more. <https://dictionary.cambridge.org/dictionary/english/workflow>
- [16] Wikipedia contributors. (2023, January 20). Agile software development. Wikipedia. https://en.wikipedia.org/wiki/Agile_software_development
- [17] What Is Kanban? Explained in 10 Minutes | Kanbanize. (n.d.). Kanban Software for Agile Project Management. <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban>
- [18] Atlassian. (n.d.). Overview. Bitbucket. <https://bitbucket.org/product/guides/getting-started/overview>
- [19] Bera, R. (2022, December 15). Git Branching Naming Convention: Best Practices. <https://codingsight.com/git-branching-naming-convention-best-practices/>
- [20] Pràctiques en empresa. (n.d.). Facultat D'Informàtica De Barcelona. <https://www.fib.upc.edu/ca/empresa/practiques-en-empresa>
- [21] UI Automator. (n.d.). Android Developers. <https://developer.android.com/training/testing/ui-automator?hl=es-419>

[22] Debugging Widgets | Apple Developer Documentation. (n.d.). Apple Developer Documentation.
<https://developer.apple.com/documentation/widgetkit/debugging-widgets>

[23] Wikipedia contributors. (2023, May 14). General Data Protection Regulation. Wikipedia.
https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

[24] Wikipedia contributors. (2023, May 2). California Consumer Privacy Act. Wikipedia.
https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act

[25] Wikipedia contributors. (2023, May 20). Intellectual property. Wikipedia.
https://en.wikipedia.org/wiki/Intellectual_property

[26] Wikipedia contributors. (2023, May 16). End-user license agreement. *Wikipedia*.
https://en.wikipedia.org/wiki/End-user_license_agreement

[27] Vicerectorat de Política Acadèmica. (2022, May 18). Normativa dels Estudis de Grau i Màster de la UPC, Curs 2022-2023. Universitat Politècnica de Catalunya.
https://www.upc.edu/sga/ca/shared/fitxers-normatives/NormativesAcadèmiques/NAGRAMA/normativa-academica-de-grau-i-master-2022-23_acord-cg-24-05-2022.pdf

[28] Equip deganal de la FIB. (2022, July 1). Normativa Acadèmica del grau en Enginyeria Informàtica. Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya.
<https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-academica-gei.pdf>

[29] Comissió Permanent. (2020, January 29). Normativa del treball final de grau del grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona. Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya.
<https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-mencio-addicional-gei.pdf>

[30] Vicedeganat de Relacions amb les Empreses. (2022, July 1). Normativa de pràctiques acadèmiques externes de les titulacions. Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya.
<https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-practiques-academiques-externes.pdf>

[31] App widgets overview. (n.d.). *Android Developers*.
<https://developer.android.com/develop/ui/views/appwidgets/overview>

[32] *Usability Testing*. (2021, August 12).
<https://www.productplan.com/glossary/usability-testing/>