UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS
Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF AN APPLICATION TO DECREASE PROCRASTINATION AND ENHANCE PRODUCTIVITY IN THE WORKPLACE

MARTÍN RODRÍGUEZ BARROSO

2024

TRABAJO DE FIN DE GRADO

Título:	Diseño y desarrollo de una aplicación para reducir la procrastinación y aumentar la productividad en el trabajo
Título (inglés):	Design and development of an application to decrease procrastination and enhance productivity in the workplace
Autor:	Martín Rodríguez Barroso
Tutor:	Sergio Muñoz López
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

DESIGN AND DEVELOPMENT OF AN APPLICATION TO DECREASE PROCRASTINATION AND ENHANCE PRODUCTIVITY IN THE WORKPLACE

Martín Rodríguez Barroso

Enero 2024

Resumen

La procrastinación se define como la tendencia a posponer o retrasar tareas importantes o urgentes, a menudo en favor de actividades menos importantes o más agradables. Se trata de un problema bastante común al que se enfrentan tanto los estudiantes como muchas personas en el lugar de trabajo. El hábito de procrastinar tareas y actividades importantes puede provocar retrasos en los proyectos, una acumulación de trabajo y un aumento del estrés y la ansiedad de los trabajadores.

Suele ir acompañada de una falta de motivación en los trabajadores y de la ausencia de un plan de trabajo claro o de cualquier tipo de sensación de urgencia. Pero, sin duda, siempre acaba afectando negativamente al rendimiento laboral y a la satisfacción personal del trabajador.

Para hacer frente a este problema se han desarrollado enfoques terapéuticos informáticos eficaces y rentables. Sin embargo, la terapia basada en ordenador tiene varias limitaciones, como la accesibilidad limitada y su dependencia tanto del tiempo como de la ubicación. En cambio, las aplicaciones de salud mental para teléfonos inteligentes presentan ciertas ventajas, como la disponibilidad constante, la propiedad común, la interacción con el usuario y la facilidad de uso.

Por ello, en este proyecto pretendemos contribuir en la disminución de la procrastinación con el desarrollo de una aplicación móvil que combine diferentes técnicas y herramientas para ayudar a los usuarios a gestionar eficazmente su carga de trabajo. Entre algunas de ellas se encuentran: El método Pomodoro, la priorización de tareas, la partición de tareas y la gamificación. El objetivo de este proyecto es desarrollar una aplicación destinada a aumentar el rendimiento laboral reduciendo la procrastinación. Para ello, la aplicación también permitirá analizar parámetros de rendimiento, a través de los cuales los usuarios podrán organizar su carga de trabajo de forma más adecuada y eficiente a sus necesidades.

En primer lugar, se realizará un estudio y recopilación de técnicas eficaces relacionadas con la gestión eficiente del tiempo y la gestión de la carga de trabajo. Una vez realizada la selección de las metodologías a utilizar, se realizará una primera maqueta de la aplicación. Con este primer diseño de UI, se comenzará a buscar la integración de las diferentes técnicas en la propia aplicación, que será desarrollada íntegramente en Flutter. Finalmente, se desplegará y probará la aplicación.

Palabras clave: Procrastinación, lugar de trabajo, método Pomodoro, gestión de tareas, Flutter

Abstract

Procrastination is defined as the tendency to postpone or delay important or urgent tasks, often in favor of less important or more enjoyable activities. This is a fairly common problem that both students and many people in the workplace face. The habit of procrastinating on important tasks and activities can lead to delays in projects, a buildup of workload, and an increase in stress and anxiety for workers.

It is usually accompanied by a lack of motivation in workers, and a lack of a clear working plan or any kind of sense of urgency. But without a doubt, it always ends up negatively affecting worker's job performance and personal satisfaction.

Efficient and cost-effective computer-based therapeutic approaches have been developed to address this problem. However, computer-based therapy has several limitations, such as limited accessibility and its dependence on both time and location. In contrast, mental health applications for smartphones have certain advantages, such as constant availability, common ownership, user interaction and ease of use.

Therefore, in this project we aim to contribute in the decrease of procrastination with the development of a mobile application that combines different techniques and tools to help users to effectively manage their workload. Among some of these are: The Pomodoro method, task prioritization, task partitioning and gamification.

The goal of this project is to develop an application intended to increase work performance by reducing procrastination. To achieve this, the application will also allow the analysis of performance parameters, through which the users will be able to organise their workload more appropriate and efficiently to their needs.

Firstly, a study and compilation of effective techniques related to efficient time management and workload management will be carried out. Once the selection of the methodologies to be used has been made, a first mock-up of the application will be made. With this first UI design, we will start looking for the integration of the different techniques in the application itself, which will be entirely developed in Flutter. Finally, the application will be deployed and tested.

Keywords: Procrastination, Workplace, Pomodoro method, task management, Flutter

Agradecimientos

Contents

List of Figures

CHAPTER 1

Introduction

In this chapter, we will delve into the context of the project and explore all the different parts that will be covered in detail throughout the project. The introduction will provide a comprehensive overview of the project's scope, and expected outcomes. It will also outline a set of objectives to be accomplished. Moreover, it will introduce the structure of the document with an overview of each chapter.

1.1 Context

Procrastination may be defined as the tendency to postpone or delay important or urgent tasks, often in favour of less important or more enjoyable activities [6]. In another definition, procrastination can be seen as the voluntary, irrational impulse to postpone a planned action despite knowing that this delay will have a cost or have negative effects on the individual [7]. This is a fairly common problem that both students and many people in the workplace face.

A study carried out by Harriott and Ferrari [8], 1996, concluded that almost 20% of adults in the U.S. self-identified as procrastinators. Chronic procrastinators tend to protect their self-esteem by choosing environmental obstacles that make it harder for them to complete tasks successfully. In addition, they often deceive themselves by completing easy tasks that require less effort, instead of focusing on completing more challenging ones.

Moreover, their health and human relationships were affected by their tendency to work close to deadlines and their perception of work delays as a possible revenge against others. They concluded that procrastinators may use this behaviour pattern and some other negative attitudes as a way of avoiding or hiding their possible lack of capacity from themselves and others.

These behaviours are even more relevant and more prevalent in college students. Studies and surveys reflect that 80%-95% of undergraduates commit procrastination, almost 75% regard themselves as procrastinators, and close to 50% systematically procrastinate [9].

This picture has gotten even worse with the increasing usage of mobile phones. Some more recent studies reveal a link between the overuse of these devices and procrastination. Findings indicate a significant positive correlation, highlighting that increased mobile phone addiction is associated with increased procrastinatory tendencies. Students who are absorbed in their mobiles may find it difficult to manage their time and show a greater inclination to delay planned tasks. This study conducted by the University of Weifang (China) [10], suggests that the mitigation of phone addiction could potentially alleviate this issue and improve overall academic performance.

The negative effects of procrastination have led to the development of techniques to tackle it [11]. Traditional methods focus on time management, motivation, and self-discipline. However, these techniques carry a great responsibility on the user, who is responsible for the correct use of these tools.

For this reason, mobile applications have emerged as an optimal solution to improve professional and educational productivity. They offer a myriad of tools tailor-made for each person's needs and preferences. These applications provide users with efficient solutions for task management, communication, organization, and goal tracking. They contribute to

optimise the workflow and enhance productivity.

Several studies have explored the relationship between these applications and the increased productivity of their users. The article by Miguel Ángel Cardoso and Alfredo Daza [12] shows and brings together the results of various research papers on the use of mobile gamification applications and their impact. Gamification is a technique that consists of using typical game elements and applying them to work contexts. The authors conclude that these applications benefit productivity in companies as long as they are adapted to the company's employees and their interests.

Research has been also carried out on time management applications. The pilot study done by the University of New South Wales [13] aimed to investigate the relationship between students' performance and usage of time management apps. It comments that a student's ability to manage time well is strongly related to academic success. In addition, students who were not using the app were more likely to be unable to handle unplanned events and to deal effectively with tasks that required additional effort.

In addition to the features they provide, mobile applications have a positive impact on productivity by helping people avoid distractions in the workplace. By using techniques like the Pomodoro method and notification management, mobile apps can keep users focused on their work and away from unnecessary distractions, including their mobile devices [14]. This creates a healthier work environment and helps people stay on task.

Bringing all these points together, the objective of this project is to create a multiplatform application that helps in enhancing productivity and curbing procrastination of students or workers. The application will incorporate various techniques such as the Pomodoro method, task partitioning and prioritization, and it will be entirely customizable to suit each user's needs and preferences. The aim is to adapt to the specific requirements of each individual.

1.2 Project goals

The primary goal of this project is to create an application that can assist students and workers in staying focused while at work, consequently reducing procrastination and enhancing their performance. The Flutter Framework has been selected as the platform for building this application. To achieve this goal, the project has been divided into smaller, more achievable tasks.

- Identification of the different options currently available to students, as well as the most widely employed tools to increase productivity.
- Choice of tools and techniques that best fit the users' requirements

- Design of an application that implements these techniques in a user-friendly and intuitive way.
- Development and implementation of such application.
- Evaluation of the system through a case study.

1.3 Structure of this document

This section outlines the structure followed by the document, listing the different chapters and providing a brief description of their contents.

Chapter 1 - Introduction: provides an overview of the subject of study and highlights previous research related to the topic. It aims to provide an understanding of the motivation behind the project and specifies the main objectives to be achieved. Additionally, it includes an index of the project's chapters and their respective content.

Chapter 2 - State of art: offers a vision of the ecosystem of time management apps and a few examples are analyzed, highlighting their strengths and areas to be improved. The main technologies used in the development of this project are also described, to provide a better understanding of the framework used and to justify their usage.

Chapter 3 - Requirements: analyses each of the use cases and their different requirements are shown.

Chapter 4 - Architecture: details the architecture used in the design of the application through all its layers, from the views themselves to the data models employed.

Chapter 5 - Case Study: explains the realization of the different use cases from a user's point of view, to understand the implementation of each of them in the final product.

Chapter 6 - Conclusions: summarizes and discusses the different conclusions drawn during the development of the project, as well as the aspects to be improved and modified with a view to a future project.

CHAPTER 2

State of Art

This chapter presents the state of the art of the project considering various aspects. Firstly, the different solutions already in place are analysed, addressing the different approaches each can bring to the project. The tools and frameworks used are detailed below, providing a basic understanding of how they work and why they were chosen for this project.

2.1 Related work

Nowadays, time management has become a crucial aspect of our daily routine. Fortunately, a wide range of programs and applications exist to help users better organise their time and maximise their productivity. Each one utilises a diverse range of techniques and features to cater to various user needs and preferences, creating a diverse ecosystem with a myriad of options.

Software solutions such as Clockify [1] or Trello have become an integral part of companies, enabling employees to organize and manage their workgroups in a visual and efficient way. These tools are especially useful for remote teams, where collaboration and communication are crucial to success.

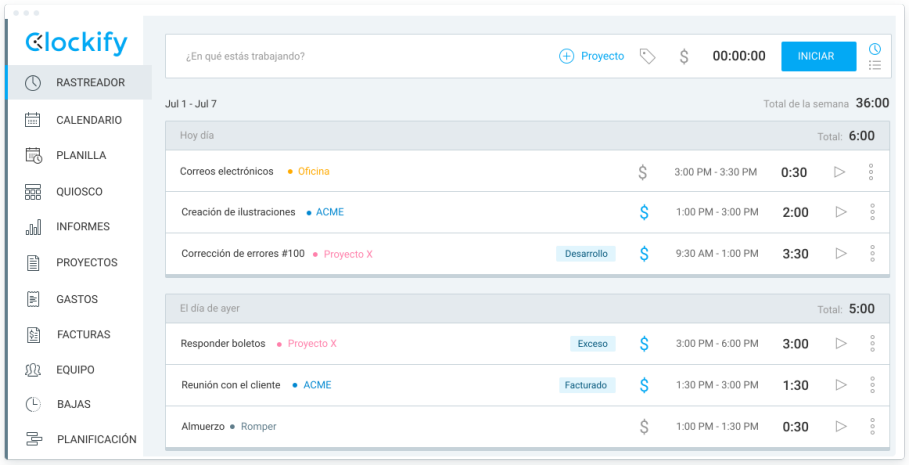


Figure 2.1: Clockify [1].

However, while these software solutions offer multiple benefits, they come with a cost. Many of them have subscription plans for premium features, and most do not even offer a free version. This means that students and individuals with fewer resources may find their options limited and may not be able to afford these premium features. Moreover, most of these tools do not have a dedicated mobile application, which can be a significant disadvantage for those who need to stay productive on the go. This is why we'll be focusing on exploring different alternatives specifically designed for mobile devices.

Taking a closer look at the main app stores available today, we can see that the vast majority of them tend to employ gamification techniques to engage their users. This is often done by implementing one or more tools, such as the popular Pomodoro method or task prioritization, and then complementing it with an extensive reward system to incentivize users to continue using the app. Typically, these rewards come in the form of in-app purchases, which can provide users with access to additional features, more content, or

other exclusive benefits.

This model has become the norm across most app stores, leaving few options for those who are looking for open-source or fully free alternatives.

One of the most popular apps is “Forest: Stay focused, be present”. It has more than 10 million downloads on Google Play and an average rating of 4.7 stars. The app combines the Pomodoro method with gamification, offering rewards for each successfully completed clock cycle. In this case, it allows the user to plant a tree for each time interval that has stayed focused without leaving the app, letting them grow their own personal ”forest”. It also has customisation options and events depending on the different times of the year, varying the existing models and adding variations to the trees.

Figure 2.2: Forest App [2]

Another trend in productivity-enhancing apps is the “To Do List”. These apps allow users to break their tasks down into manageable chunks, and keep track of their progress. However, while these apps are useful for organising tasks, they do not necessarily help users focus on the task at hand or avoid procrastination. This is because most “To Do List” apps simply provide users with a list of tasks to complete, without offering any techniques or strategies to help them stay focused and motivated. As a result, users are left to manage their time and productivity on their own, without any real guidance or support from the app. While these apps are certainly helpful in terms of organising tasks, they may not be the best option for users who are looking to improve their productivity and concentration.

Finally, there are several alternative time management techniques, which implement different time intervals for both concentration and rest. Traditional intervals are usually 25 minutes, and this can be too short for more complex tasks. Among the most popular

alternatives is the so-called 52/17 rule [15]. This is based on the results obtained by the DeskTime application, which monitored the workflow of employees in various companies. The employees who were at the top of the productivity rankings within the application had in common that they had longer work sprints, and took breaks to get away from the work environment and clear their minds. Extrapolating the results, the method proposes 52 minutes of intense and dedicated work, and 17 minutes of breaks in which the user disconnects completely.

Another alternative is based on the well-known Ultradian Rhythms [3]. These are the biological cycles of the human body, and it proposes to follow the natural work/rest cycle of the human being. According to psychologists, our body tends to perform best when we work for about 90 minutes, followed by a 20-minute break. These rythms are represented in Figure 2.3. This method is commonly used by professional musicians, who divide each session into 90 minutes and take an extended break between each session.

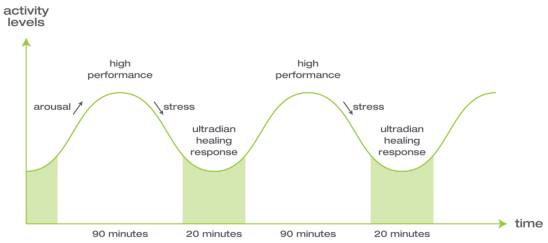


Figure 2.3: Ultradian Rythms Diagram [3].

2.2 Enabling technologies

This section highlights the technologies used in the development of this project. Frameworks and programming languages will be explained, its structure, and main features, in order to understand the reason for their usage.

2.2.1 Dart

Dart is an open-source programming language, originally designed by Lars Bak and Kasper Lund and developed by Google. The motivation behind its creation is the rise of web browsers applications and mobile devices. The way software is written has been significantly influenced by both of these patterns.

Dart was born with this idea in mind as a client-optimised language, made specifically to help developers create fast and intuitive applications for all kinds of platforms. It aims to shield developers from the low-level complexity of the devices while providing easy access

to its powerful features. As its main features stand [16] [17]:

- **Pure object-oriented:** all values a Dart program deals with at run-time are considered objects, including elementary data such as numbers and Booleans. This implements a uniform treatment for all data that permits, for example, the use of a collection for all kinds of data. This uniform object model simplifies and solves many issues for programmers.

- **Type-safe:** Dart is an optionally typed language. While types are mandatory, the use of type annotations is optional due to the language's robust type inference capabilities. Dart also uses static type checking, ensuring that a variable's value always matches the variable's static type.

- **Null safety:** prevents potential runtime errors resulting from unintentional access to null values. With null safety, all variables are considered as non-nullable, requiring a value of the declared type. Dart analyzer flag if a variable has either not been initialized with a non-null value or has been assigned a null value.

- **Libraries:** Dart offers a diverse set of core libraries that cater to numerous programming tasks such as encoders, mathematical functions or asynchronous programming. Also, many APIs are available through their packages, and third-party companies and some users are constantly uploading new packages, with new or evolving features.

- **Native development:** the Dart VM offers a just-in-time compiler (JIT), that allows hot reloads, accelerating the possible iterations during development. Also the Dart ahead-of-time (AOT) compiler can compile to machine code in a short startup time.

2.2.2 Flutter

Flutter is an open-source SDK (Software Development Kit) launched by Google in May 2017. It is widely used for cross-platform development thanks to its compatibility with up to 6 different operating systems. Despite being a newcomer in the market, has managed to attract a lot of attention from many app developers, thanks to its user-friendly interface and utilization of an easy-to-use language such as Dart.

It has been gaining more and more popularity over the years with more than 36,889 projects being built in Flutter and a user base of about 2 million developers [18]. Flutter has about 84K followers on Reddit, 134K GitHub stars, and about 111,456 questions asked in Stack Overflow [18]. The following graph helps to understand his popularity compared to similar programming platforms:

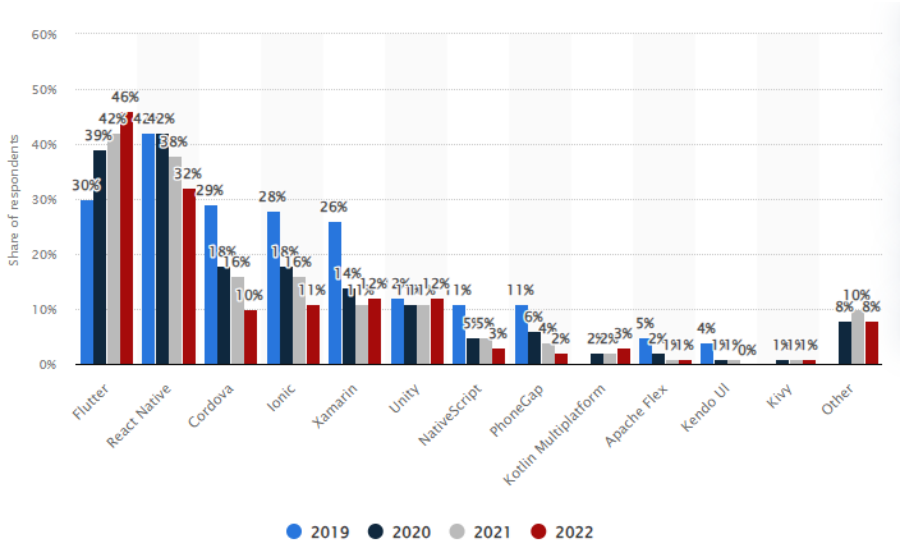


Figure 2.4: Growing popularity of Flutter [4].

Before going into the architecture behind Flutter, some of the general functionalities that have made it so popular are [19]:

- **Efficiency:** Hot reload instantly reflects code changes on the UI, speeding up the process and saving costs.
- **Cross-platform compatibility:** Flutter allows for cross-platform development, making it easier to maintain a single codebase for multiple applications.
- **Reusability of Widgets:** In Flutter, everything is a widget. Widgets can be customized for the app's needs, and there are plenty of libraries and packages full of built-in ones to help the development process.
- **Native performance:** Flutter provides platform-specific widgets that can be used to access native features like camera and geolocation by integrating existing Java, Swift, and Objective-C codes. Flutter also supports easy integration with third-party APIs.
- **Open source:** Flutter is an open-source platform for creating user-friendly applications. It's free of cost and has detailed documentation available online.

2.2.2.1 Flutter Architecture

Taking now a closer look at the general architecture [5], Flutter is a modular system that comprises independent libraries that rely on one another, with no layer having privileged access to the layer below. Parts are also optional and replaceable.

Flutter applications are packaged following the standard packaging process like other native apps. A platform-specific embedder provides an entry point and manages the message event loop. The Flutter engine is responsible for rendering composited scenes and supports all Flutter apps. It provides graphics, text layout, plugin architecture, and a Dart runtime. Developers interact with Flutter through its framework, which includes foundational classes, a rendering layer, a widgets layer, and Material and Cupertino libraries. Many higher-level features are implemented as packages, including platform plugins and platform-agnostic features.

Figure 2.5 provides an overview of the components of a typical Flutter app. It shows the position of Flutter Engine in the stack, indicates API boundaries, and identifies component repositories. The Dart App is responsible for composing the desired UI and implementing the business logic of the application. The Framework provides the necessary higher-level API and composites the app's widget tree. On the other hand, the Engine provides a low-level implementation of Flutter's core APIs and integrates with the needed platform using the embedder. This embedder manages system access, event loop, and API integration. Finally, the Runner creates an app package for the target platform using the platform-specific API exposed by the Embedder.

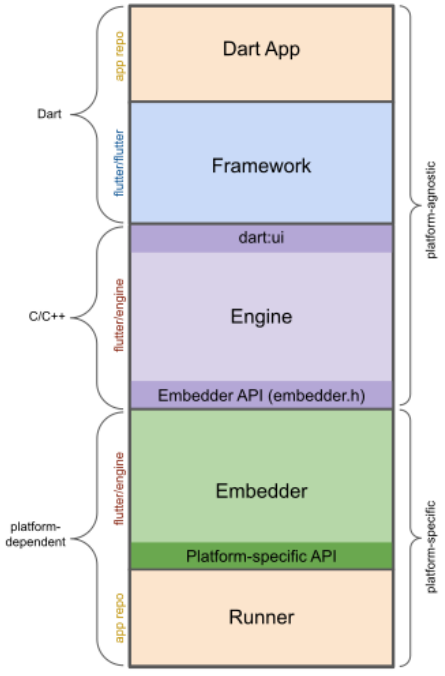


Figure 2.5: Anatomy of a Flutter App. [5]

2.2.2.2 Widgets Overview

The user interface for every Flutter app is built by widgets [5]. Each of them is an immutable declaration of part of the user interface and forms a hierarchy based on composition. Each widget is nested inside its parent and can receive context from it. This structure continues to the root widget, which is the container that hosts the Flutter app.

When an event occurs, such as a user interaction, apps update their user interface by instructing the framework to replace a widget in the hierarchy with another widget. After this, the framework compares the new and old widgets efficiently to update the user interface. There are pure Dart implementations specifically designed for Flutter to control the UI, regardless of whether it's on iOS or Android.

Flutter widgets are made of small, single-purpose components that work together to create powerful effects. Core features are abstract, and even basic features such as padding and alignment are implemented as separate components instead of being built into the core. There are widgets for padding, alignment, rows, columns, and grids. The Container widget is widely used and it consists of various other widgets that are responsible for layout, painting, positioning, and sizing. These widgets include Align, Padding or DecoratedBox. Although these layout widgets do not have any visual representation of their own, their purpose is to control some aspect of another widget's layout.

Figure 2.6 shows an example of the composition of different widgets creating a widget tree that represents a coloured card with an image and a caption next to it.

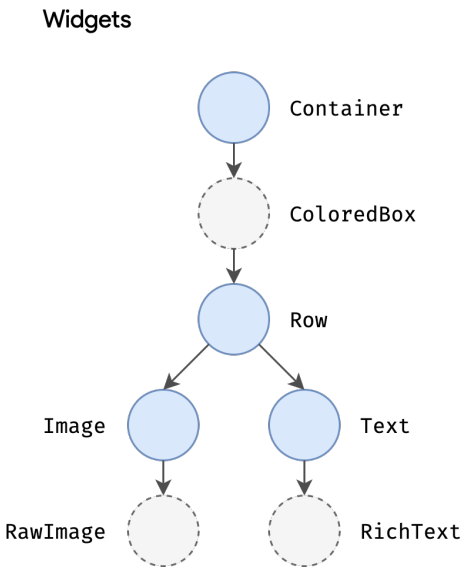


Figure 2.6: Widget tree. [5]

Widgets can also contain a state. Flutter manages this by having stateful and stateless

widgets. Many widgets need to vary based on user interactions, for example, tapping a button, and when that input appears, the widget needs to be rebuilt. Because the widget itself is immutable, this state is stored in a separate class. When the method setState is called, the framework updates the interface and rebuilds the widget with its new state. Widgets whose properties doesnt vary over time are StatelessWidgets.

Separating state and widget objects allows other widgets to treat stateless and stateful widgets the same way without losing their state. Parents can create new instances of the child without losing its persistent state, and the framework reuses existing state objects when needed.

CHAPTER 3

Requirements Analysis

This chapter analyses the application requirements and user needs according to the different use cases. This analysis is necessary in order to describe the users and their different objectives, anticipating how they will interact with the system in different scenarios and being able to design the system accordingly.

3.1 Use Cases

A use case is a simple textual description that examines a system by representing the interactions between the system and an outside entity [20]. They are totally oriented to the final user, being easy to understand for all the implied parts and they aim to elicit user needs and the system functional requirements in terms of the user's goals.

They often describe a scenario, which is shaped by a set of actions, where the user utilizes the system to perform or reach a desired output of value to a particular actor. Actors are users or external entities by role. Each scenario is a particular use case. It is possible to have alternative flows for the same use case, representing alternative scenarios of failure of the actions.

Use Case modelling [21] is commonly used due to its facility to identify the actors and the behaviour of a system. Dependencies among use cases may be represented if needed. These layouts are not able to express static structures and characteristics of systems though and must be complemented later on in the project. Figure 3.1 displays a UML Use Case diagram that represents the use cases of our project.

The different use cases proposed are:

1. **Start and control timer cycles:** the user starts the timer an goes through the focus phase, where concentrates on the corresponding task and stays productive. Then, the rest phase starts, which the user uses to rest up to be ready for the next focus interval.

2. **Visualize tasks:** the user to visualise their tasks, helping them to prioritise and tackle the most important ones.

3. **Manage tasks:** the user can efficiently manage their tasks, adding new ones, modifying existing tasks or deleting them if necessary.

4. **Manage timer settings:** the user has the option to modify the duration of the intervals. This can be done using pre-configured Timers, or setting up the durations as the user whises.

5. **Visualize user statistics:** the user access to the record of their focused intervals. Days and hours with peaks of productivity can be studied to improve the user's workflow.

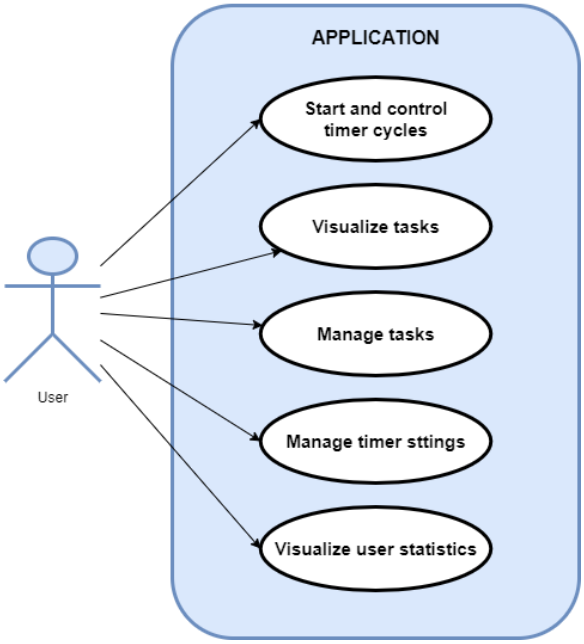


Figure 3.1: UML Diagram

In the characterisation of each of the use cases described, the structure proposed by *"Use Case Description of Requirements for Product Lines"*[22], University of Florence, has been followed. This has been adapted to the needs of our project, omitting some characteristics to fit better in our system. In this project, the following characteristics have been included:

- **Goal:** is the indicator that the use case has been completed successfully.
- **Primary actor:** the actor who initiates the system to achieve the goal.
- **Trigger:** the interaction between the actor and the system that functions as the start of the scenario
- **Scenario:** is an instance of a Use Case, and represents a single path of a set of actions through the Use Case.
- **Extensions:** refers to alternative sequences that may also satisfy the goal, or possible extensions to the previous sequence.

3.1.1 Start and control timer cycles

- **Goal:** stay focused during a certain value of time to be chosen by the user, with a subsequent rest time afterwards.
- **Primary actor:** user

- **Trigger:** press the start button on the main screen.
- **Scenario:**
  1. The user plays the start button and the timer starts to run.
  2. The user keeps focused during the duration of the Pomodoro interval.
  3. The user may pause the timer every time needed until the timer expires.
  4. Each time the timer is paused the application registers and stores a focused interval.
  5. When the timer concludes the system toggles to the rest screen.
  6. A focused interval is also stored when the timer finishes.
  7. The user is given a time-out to rest.
  8. When the rest time expires the system toggles back to the initial Pomodoro timer
  9. The case repeats itself cyclically.
- **Extensions:**
  1. The user can restart the Pomodoro timer each time needed by pressing the restart button.
  2. Restarting the timer doesn´t store a focused interval. This may be used when the user starts the timer but not necessarily start being focused.
  3. The rest time can be elongated by pausing for the time needed.

3.1.2 Visualize tasks

- **Goal:** display the task the user has to do, and help them prioritise and focus on certain tasks.
- **Primary actor:** user
- **Trigger:** press the task list icon in the lateral menu
- **Scenario:**
  1. The user slides the screen to access the lateral menu
  2. They navigate to the task screen, by pressing the pertinent button.
  3. A scrollable list is displayed showing every task the user has registered
  4. Tasks that must be done according to the deadline, are displayed with their date in red, to help the user identify and prioritise them.

5. Done tasks are also marked, to show the user the corresponding progress.

- **Extensions:**

1. The user can easily detect what task has to be done immediately by the colour of the deadline
2. The user can organise their workload by managing their task with the corresponding filters.

3.1.3 Manage tasks

- **Goal:** manage easily and efficiently the user's tasks.
- **Primary actor:** user
- **Trigger:** add, modify or delete an existing task
- **Scenario:**

1. The user access to the Task List page.
2. The user can add a new task by pressing the floating button located at the bottom of the screen.
3. A questionnaire appears where the user can specify the name and description of the task
4. A deadline for the task can be added by choosing on the calendar the desired date.
5. By saving the task, the system registers and storages the given data and automatically adds the new task at the top of the list.
6. The user can set tasks to be done or not done yet.

- **Extensions:**

1. Done task can be marked by clicking on the checkbox. By crossing out the name, the system advises the user which task has been done.
2. Any introduced task can by erased from the application and database by dragging the container of the task to the left.

3.1.4 Manage timer settings

- **Goal:** set focus and rest intervals according to the needs of each user.
- **Primary actor:** user
- **Trigger:** press the settings icon in the lateral menu
- **Scenario:**

  1. The user slides the screen to access the lateral menu
  2. They navigate to the settings screen, by pressing the pertinent button.
  3. In the first text field, the user inputs the desired duration for the focus time timer.
  4. In parallel, the user also inputs the desired duration for the rest time.
  5. The user can select an option in the drop-down menu to use a pre-configured setting.
  6. By pressing the save button, the values are introduced in the application.
  7. Navigating back to the timer page, the user sees the desired duration in the focus timer.

- **Extensions:**

  1. These values can also be changed by dragging the circular slider around the timer. However, this value only is saved for the dragged slider. The next interval will be back to the original value introduced.
  2. Pre-configured Pomodoros can be used by the user following alternative techniques to the clasic Pomodoro intervals.

3.1.5 Visualize user statistics

- **Goal:** visualise the duration of the different time intervals in which the user has been focused, and thus be able to track the user's productivity peaks on both a daily and hourly basis.
- **Primary actor:** user
- **Trigger:** press the history icon in the lateral menu
- **Scenario:**

1. The user slides the screen to access the lateral menu
2. They navigate to the history screen, by pressing the pertinent button.
3. A list with all of the intervals the user has been focused is display
4. Intervals are organized chronologically by date, and show the duration and the hour on the day which it was recorded
5. The user can analyse which hours were more focused by looking at their durations, due to every time the chrono is paused a new entry is recorded

- **Extensions:**

1. The user can also dismiss some intervals if they consider was an error (Ex: by clicking the timer but pausing immediately after)

CHAPTER 4

Architecture and Methodology

This chapter provides a comprehensive overview of the application's architectural design, including its implementation. It starts with an overview of the application's architecture, followed by a detailed description of each layer. This chapter also covers the different screens of the app, the logic behind them how they relate to each other, and the different data models used in the development process.

4.1 Overview

The system has been designed to follow a layered architecture, consisting of the application layer, domain layer, and data layer. This model provides several benefits in terms of design, maintainability and scalability. While the application layer focuses on the UI, the domain layer contains the business logic.

This modularity permits either modifying the user interface without affecting the underlying logic or changing the logic rules without impacting the user interface. This approach enhances maintainability and scalability because also scaling one functionality can be done isolated. Also, this separation allows for the reuse of modules or components across different parts of the application. Finally, separating the data layer from the application and domain layers allows you to incorporate different data storage systems without impacting the other layers, creating a more flexible and manageable solution.

The objective of the project is to create an efficient, maintainable and scalable system, by following this layered architecture approach.

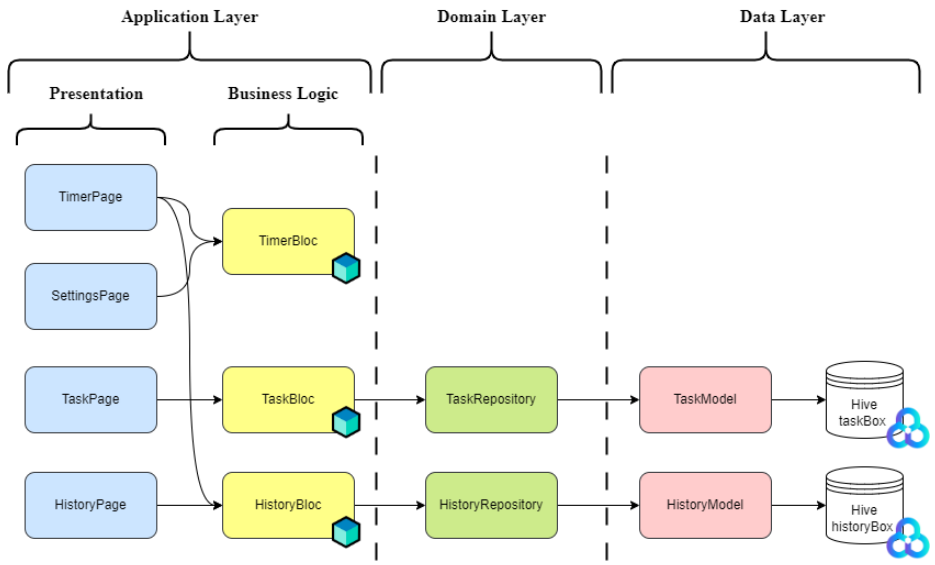


Figure 4.1: Architecture Diagram

Figure 4.1 shows the layered architecture as mentioned before. The relations between each component have been simplified in order to provide a visual and eye-catching representation. Now we will now take a deeper look at each of the layers. In the application layer the different screens of the application will be explained and how they conform the user interface. The domain layer holds the set of rules and actions that manages the events and data of the application and we will develop on how it communicate the user inputs with the data layer. Finally, in the data layer we will see the data models used and how they

are implemented using the Hive package [23], an efficient and lightweight nosql database written in Dart.

4.2 Application Layer

The main function of this layer is to display the application data on the screen and serve as the main point of interaction with the user. This forms what is known as the UI (user interface).

Each time an input is received, either from the user or externally, such as a response from the network, the user interface must be re-rendered to reflect these changes. This layer can therefore be considered as the visual representation of the state of the application as retrieved from the data layer.

However, sometimes the information you receive from the data layer of an application may not be in the desired format for display purposes. This means that you may need to extract only a portion of the data or combine multiple data sources to present the relevant information to the user. This data formatting is also carried out by the application layer. Regardless of the logic you apply, you must pass the UI all the information it needs to render completely. So this layer acts as the pipeline that converts changes in the data into a format that the UI can be rendered and then displayed.

In this context, the application layer can be divided into 2 separated parts, that combine and communicate with each other to conform to a smooth user interface and a seamless flow of data.

4.2.1 Presentation

This first layer division is in charge of representing all the parts of the application with which the user will be able to interact. It is formed by each of the views that make up the application, each with its respective functionalities and objectives.

As seen above, in a flutter application, each view is made up of a tree of widgets. These widgets have two main functionalities, to represent each of the visual components of the application, and to capture all user interactions on the screen.

An overview of the main pages of the application is provided below:

- **Timer Page:** This is the main view of the application. It is the home page and the page from which the other screens are accessed. It shows the Pomodoro cycles and the timer with the countdown clock. To know the duration desired by the user, it communicates with the “TimerBloc”, which provides the page with the necessary

configuration. Its implementation is divided into two parts, one interface for the work cycle, and one for the rest cycle:

  - **Focus Page:** This page includes the timer for the concentration time. It also has two buttons, one to reset the timer and one to pause it. In addition, it communicates with the "HistoryBloc", to which it sends the user's concentration interval records. The logic behind this system is described in more detail in Chapter 5. Finally, from this view, you can access the side menu that allows you to navigate to the rest of the application's functionalities.
  - **Rest Page:** Is distinguished from the previous interface by the absence of the restart button, it also changes the palette of colours so that the user can clearly differentiate the cycles of concentration from those of rest. It does not incorporate any other elements, in order to take the user away from the phone and allow them to take a break from both work and the screen.

- **Settings Page:** From this page, the user can adjust the concentration and rest intervals to the desired time. The view has a menu with several pre-configured Pomodoros, each with different durations and based on different methods. In addition, in case none of them suits the user's needs, they have the option to set each cycle to the exact duration of their choice. Once the values have been selected, the view communicates with the "TimerBloc", sending it the necessary data, so that it updates the application and the corresponding timers with the set values.

- **Tasks Page:** his view incorporates the use cases of viewing and managing tasks. Once the user accesses it, it displays the tasks previously entered by the user. To do so, it establishes communication with the "TaskBloc", which is in charge of all the logic related to the tasks. This Bloc communicates with the "TaskRepository" to obtain the tasks stored in the database. From this view, the questionnaire to add new tasks is also accessed through a button at the bottom of the screen. Once the user has filled in the necessary data and clicked on save the task, the view sends the data entered to the "TaskBloc", which communicates again with the "TaskRepository" to update the database with the new task entered. This process is similar to the one followed to delete a task, except that in this case, the Bloc provides the repository with the index of the task to be deleted. Finally, this communication flow also allows the user to update the tasks. By checking or unchecking the checkboxes of each task, the user can notify the system that a task has been performed. Each time one of these events occurs, the "TaskBloc" repaints the interface, adjusting it with the necessary filters implemented for task prioritisation, or adding or removing the necessary tasks.

- **History Page:** The last view shows the list of all the user's concentration ranges recorded by the application. For this purpose, it establishes communication with the "HistoryBloc". This links to the "HistoryRepository" in charge of retrieving the user's history from the database. The user can see in chronological order how much time he has spent each time concentrating on his task. The intervals are sorted by days, and also show the exact time at which they ended, so that the times of the day when the user is most productive can be studied.

4.2.2 Business Logic

To implement the design described above, the BLOC architecture pattern has been used. This allows the user interface to be separated from the business logic through the use of different components, in the case of this project, Blocs.

Each Bloc communicates with the corresponding interface, receiving a flow of events caused by user interactions. This, in turn, returns to the interface a flow with updates on the state of the application, which allows the view to be kept up to date in the face of possible changes to the application. In addition, they act as an intermediary between the interface layer and the domain layer, preparing the interface for changes in the application's data flow. Each of the components used, with their main functionalities, is detailed below.

The "TimerBloc" is in charge of managing the state of the timer cycles, as well as modifying the interval durations according to the settings introduced by the user.

It is responsible for pausing and restarting the stopwatch in case the user triggers the corresponding event, as well as for switching the interface between focus and rest time at the end of each cycle. It also communicates with the settings page. Depending on the values entered by the user, it will emit the corresponding status to the timer page, so that it reflects the relevant changes.

The "TaskBloc" is in charge of managing everything related to the tasks. It establishes communication with both the "TaskRepository" and the Task page. Every time the user adds, modifies, or deletes a task, the Bloc receives an event, processes it, and emits the corresponding information to the "TaskRepository", which is in charge of updating the database. It then takes care of receiving the updated task list and emits it to the interface so that the application reflects the modifications received. In case it does not receive the list of tasks, it emits an error status, used to warn the user that there has been a failure in receiving the tasks.

Finally, the "HistoryBloc" has a similar function, but in this case, it manages the user's interval history. It communicates with the timer page, which emits an event whenever a new interval needs to be saved. The bloc retransmits it to the "HistoryRepository", and

it adds the new entry to the database. It is also in charge of receiving and updating the history page with the intervals corresponding to the user.

4.3 Domain Layer

The domain layer is located between the application layer and the data layer, and its main purpose is to encapsulate the business logic that drives the application interface. It is an optional layer but is widely used because of all the benefits it offers. It is used to manage complexity, as it allows responsibilities to be divided between different classes. It also avoids code duplication and improves the testability of the app.

Two repositories form the domain layer of this project. These serve as a bridge between the business logic and the data layer, enabling communication between the two. They implement specific methods to deal with each of the data models, in addition to the main CRUD operations.

Using these repositories simplifies the application logic since the data arrives at the layer already formatted as required by the user interface. Furthermore, being independent of the data layer allows the databases to be migrated to another storage system, without needing to modify any application layer component. This increases the scalability and reusability of the code.

4.3.1 Task Repository

The "TaskRepository" is responsible for managing and storing the tasks that the user enters into the application. For this purpose, it establishes communication with the "TaskBloc", which registers all events issued by the user related to task handling. It interacts with the Hive database that stores the tasks and implements the necessary methods to create, modify or delete them. It is also in charge of receiving all the tasks stored from previous user sessions, and delivering them to the interface, thus allowing the rendering of the task list view. The methods mentioned are detailed below:

- ***getTasks():*** This method is used to obtain the tasks introduced by the user.
- ***addTask(Task task):*** This method is used to add a task to the database. It receives the Task object with the data entered by the user.
- ***updateTask(Task task, Bool isDone):*** This method updates the status of a task. It indicates whether the user has marked the task as done or not. It receives the task to be modified, as well as the field that controls whether the task is registered as *"done"* or not.

- ***deleteTask(Task task):*** This method is used to delete a task. Receives the task to be deleted selected by the user.

4.3.2 History Repository

The "HistoryRepository" is responsible for managing the concentration ranges registered by the application. It establishes communication with the "HistoryBloc", ensuring the data flow between it and the database corresponding to the user's history. It is in charge of storing each of the intervals registered by the application, and of retrieving them when displaying the user's concentration history. Finally, it implements a method to delete the records, in case the database needs to be cleaned. This functionality is not accessible to users. It requires the following functions:

- ***getUserHistory():*** This method is used to obtain all the user's concentration interval registers.

- ***addHistoryInterval(History history):*** This method allows a new time interval to be recorded in the database. It can be called either when pausing the concentration timer, or at the end of the timer cycle.

- ***deleteHistory():*** This method allows the deleting of all records in a user's database. It is implemented for the developer, in case it is necessary to purge the database, and the user of the application does not have access to it.

4.4 Data Layer

The data layer is composed of the data sources. These sources are responsible for storing and providing the application with the data necessary for its operation. In this project, two databases implemented in Hive have been used as sources.

Hive is a local database written in Dart. It is light and fast, which makes it ideal for the development of this project. It follows the key-value model, so it is not a relational database. It allows the registration of the different data models used, which gives access to predefined functions to access and manage the data efficiently on each user's device.

For this project, two different data models have been registered, one for registering tasks and one for user concentration intervals. These models will be detailed below indicating the use given to each one and how they are integrated in the application.

4.4.1 Task Model

This model is responsible for representing the tasks entered by the user, as well as all their corresponding information. Figure 4.2 shows a visual representation of this model, as well as its corresponding attributes, which are explained below.

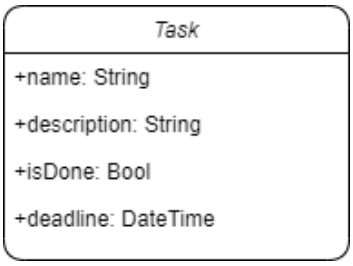


Figure 4.2: Task Model

- ***name:*** Is the name given by the user to the task.
- ***description:*** Any additional information the user may give to the task, such as additional work, observations, or anything else.
- ***isDone:*** This boolean points out whether the task has been done or not. It´s necessary to appropriately represent the Task in the list. Is settled to false by default, and when the user finalises the task, must update the task by checking the corresponding box.
- ***deadline:*** Indicates the date by which the task must be completed. If not, the system advises the user in order to help them prioritise the necessary tasks.

4.4.2 History Model

This model represents each of the time intervals in which the user has been concentrating. The elapsed time is calculated from when the stopwatch starts until it is paused. This may be because the user has deliberately paused it or because the concentration cycle is over. The same happens at the start, it can start counting from the beginning of the cycle or from when the user resumes a cycle that was paused. Figure 4.3 shows the representation of this model:

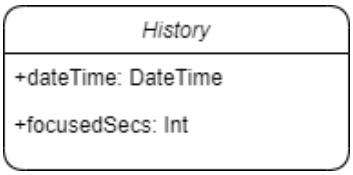


Figure 4.3: History Model

- ***dateTime:*** Indicates the date on which the interval has ended. Saves both the day and the exact time
- ***focusedSecs:*** Saves the exact number of seconds during which the user has been focused.

CHAPTER 5

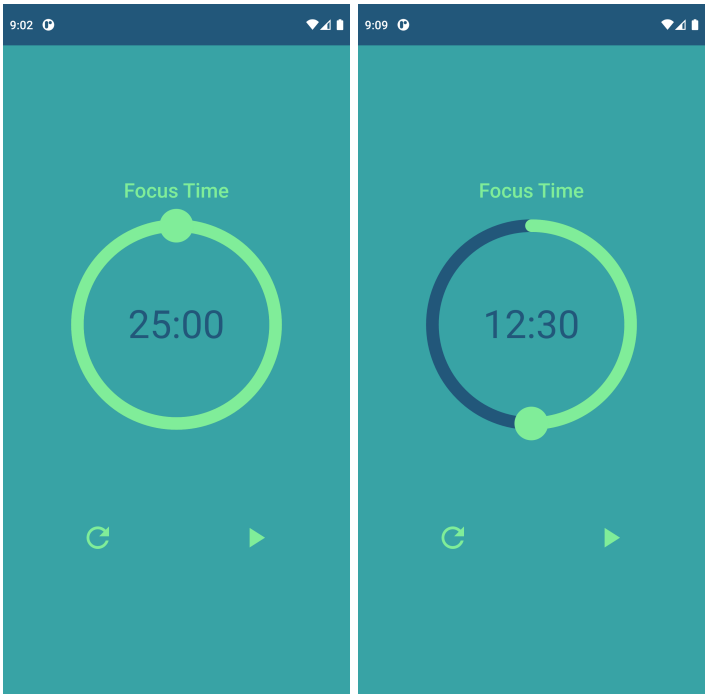
Case study

This chapter aims to show the full functionality of the system in a real environment. To do so, it will be studied from the perspective of a final user, who accesses the application and makes use of every use case. Through this case, the real usability of the tools provided will be verified for a student in his daily life, with the purpose of increasing his productivity and concentration, and reducing his levels of procrastination in his daily work.

5.1 Pomodoro method

The first thing the user sees when starting the application is the main screen with the Pomodoro method timer, as shown in Figure 5.1a. The duration of an interval will correspond to the settings previously configured by the user. If the default values have not been modified, the classic Pomodoro settings will be displayed.

The user can also adjust the countdown duration by dragging the circle surrounding it. Thus, if is desired to reduce the timer's duration by half, is only needed to position the pointer at an angle of 270^{o} concerning the circle. This example is shown in Figure 5.1b.



(a) Full Pomodoro (b) Half Pomodoro

Figure 5.1: Pomodoro Page

This interface follows a simple and minimalist design, providing the minimum necessary elements. The aim is that the user receives as few inputs as possible, allowing him to concentrate on his task and removing distractions. Apart from the stopwatch and the elapsed time indicator, the display contains only two clickable icons, but with clearly distinguishable functionalities.

The "pause" icon is not only used to pause the stopwatch if necessary. Its main purpose is to keep a record of the time interval during which the user has been focused. Both the elapsed time and the end date are recorded, as we have seen previously in Chapter 4. This makes it possible to keep track of both the maximum focus intervals (the maximum

amount of time the user can keep working on the same task) and the hours and days with the highest productivity peaks. This concept will be discussed in more detail later in this chapter, looking at the section dedicated to these records.

The "restart" icon, on the other hand, does not store or create any entries in the database. Its purpose is to restart the stopwatch, with the aim of not storing "misleading" intervals, in which the user has left the time running by mistake without having undertaken the corresponding task. It is in users' hands to use these two functionalities properly, in order to be able to correctly study the subject's productivity.

Once the concentration-time has elapsed, we move on to the rest interval. In addition, a last register is stored with the duration of the time elapsed from the last pause of the stopwatch until its end. This new view can be seen in Figure 5.2.

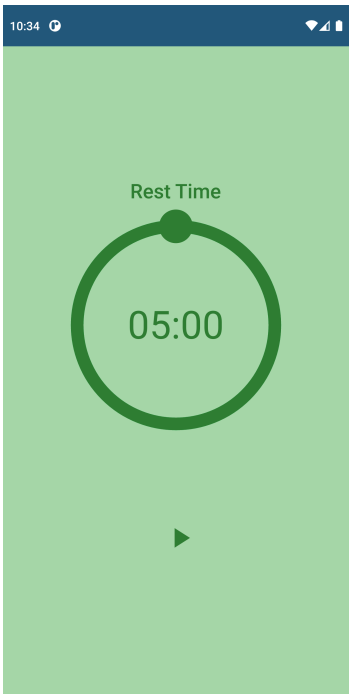


Figure 5.2: Rest Page

It keeps the same design as the previous one, changing the range of colours to clearly differentiate between concentration and rest intervals. Again, the circular indicator is draggable in case the user wants to shorten his rest time. It can be paused without entailing any additional events, since by registering the productivity intervals, it is also possible to study the time taken by the user to take a break before tackling a new Pomodoro cycle.

The view does not incorporate any additional functionality, nor does it allow access to the rest of the application's use cases. It is designed to take the user away from the mobile phone while the time is running and to provide adequate rest from studying and the screen.

All the functionalities described are then summarised in the following flow diagram, Figure 5.3. It shows all possible paths to be followed by the user in a structured and organised way.

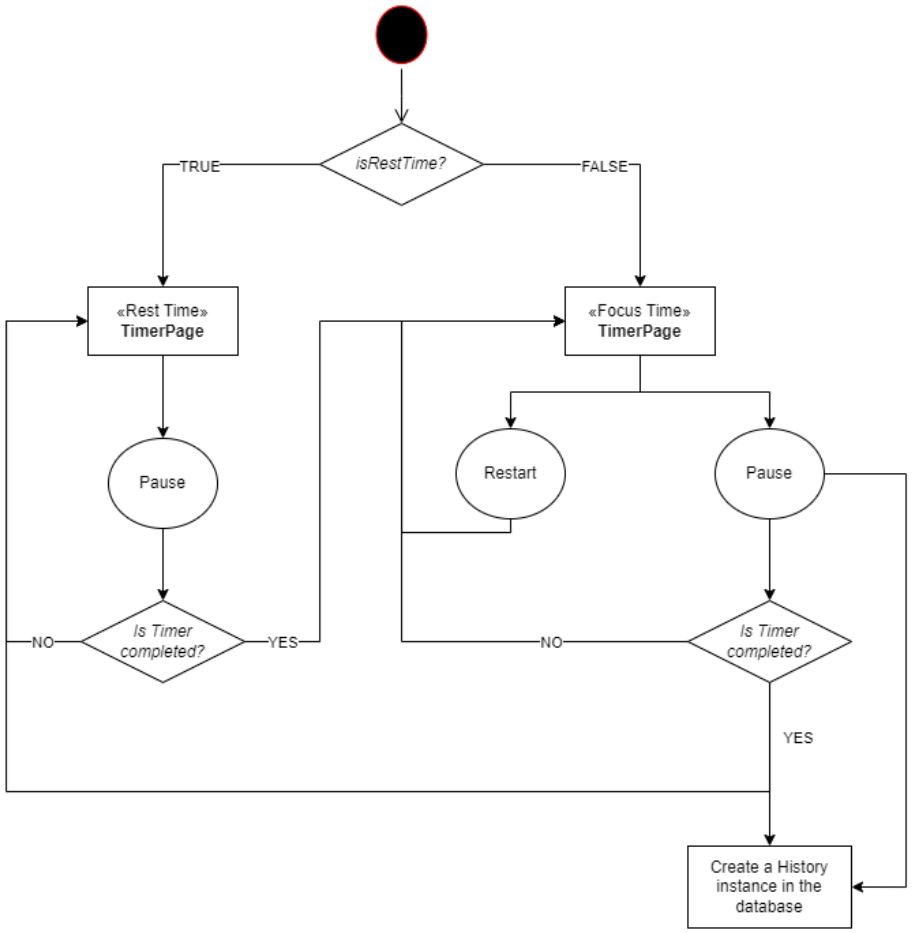


Figure 5.3: Flow Diagram

To access the rest of the application's functionalities, a navigation system has been implemented based on a side drop-down menu, known as the "Navigation Drawer". This is hidden, following the philosophy of a clean page without any stimulus for the user, and is accessed by simply sliding the screen from left to right. Figure 5.4 shows the mentioned menu.

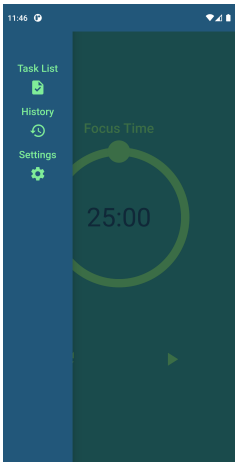


Figure 5.4: Drawer

Each use case of the application is accessible by clicking on its corresponding icon. The use cases will be explained in detail below.

5.2 Task Management

Using the "Task List" icon in the menu, the user navigates to the "Task" view, where it is possible to view and manage the different tasks that have been created. As we saw in Chapter 4, each task consists of a name, a brief description, the assigned deadline, and finally, whether it has been accomplished or not. Figure 5.5 shows the screen when the user does not have any task set, either because he has not registered them yet, or because he has deleted the previous ones.

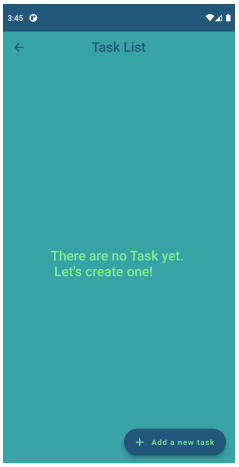
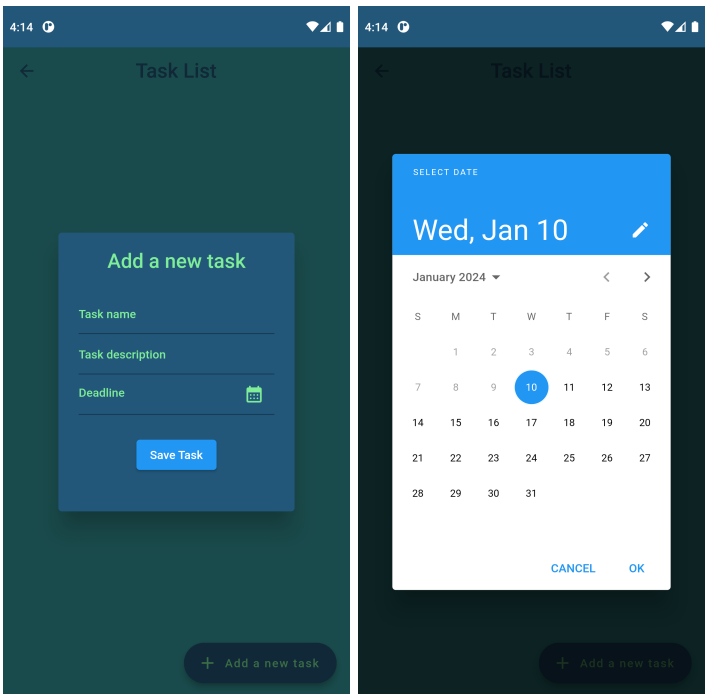


Figure 5.5: Empty Task Page

5.2.1 Creation of a Task

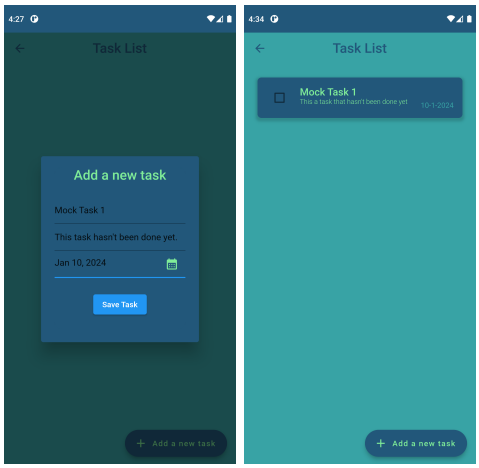
By clicking on the button in the lower right corner, the user can access the task creation menu (Figure 5.7a). This menu has two text fields to enter the name of the task and its description, as well as a third field to enter the deadline. This can be entered either manually or by clicking on the icon on the right, which will open a drop-down calendar where the user can select the desired date, as shown in Figure 5.6b.



(a) Task Creation (b) Deadline selection

Figure 5.6: Add Task Menu

We will illustrate this case study with a mock task. Once the user enters the pertinent values, shown in Figure 5.7a, pressing the save task button will create and store the task in the database. The field indicating whether the task has been performed or not is set to false by default. In Figure 5.7b you can see how the task has been created correctly.



(a) Creation of Mock Task (b) Mock Task Created

Figure 5.7: Task Creation

5.2.2 Visualization and Modification of a Task

As can be seen in the previous image, each of the task cards contains a checkbox. This checkbox allows the user to modify the value that indicates whether the task has been performed. Once checked, the system warns the user by crossing out the name and description of the task, as can be seen in Figure 5.8.

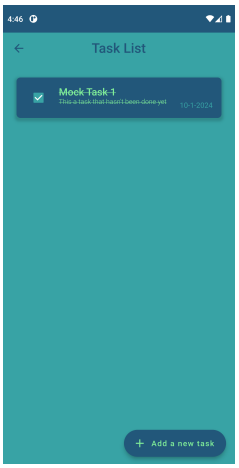
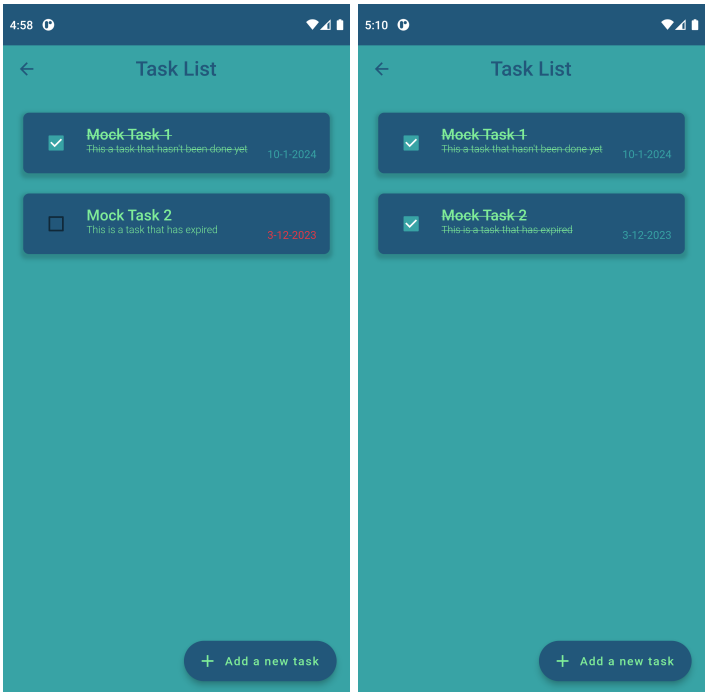


Figure 5.8: Done Task

In the case that the deadline for a task has expired, and it has not yet been completed by the user, the system will colour the date assigned to the task red. In this way, the system warns the user to prioritise the necessary tasks, according to the dates introduced by the user. Let's exemplify this by adding a new task, as shown in Figure 5.9a. Once

the user completes the task, the system will repaint the date with the appropriate colour as shown in Figure 5.9b, indicating that the task has been successfully completed. This method allows the user to manage and prioritise the necessary tasks, as well as to keep track of the completed work.



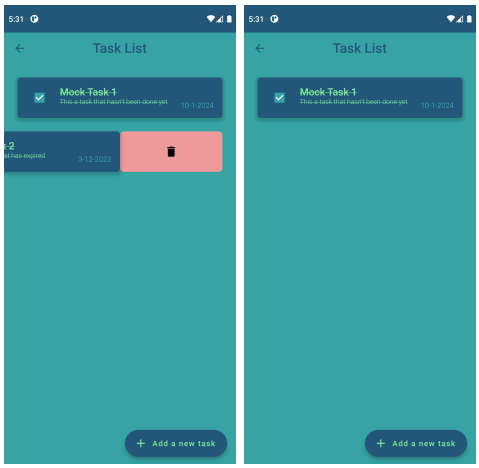
(a) Expired Task (b) Expired Task Done

Figure 5.9: Prioritization of Task

5.2.3 Deleting a Task

Finally, the user can delete tasks that are no longer required to be saved. By dragging the card for each task to the left, an icon appears that allows the task to be deleted, as shown in Figure 5.10. Immediately the task will no longer appear in the view, and any database records associated with that task will be deleted. This functionality allows the user to clean up his list of tasks, and to delete them as soon as they are completed, or to organise them by project in order to see the progress made.

It should be noted that this action is irreversible, so it will not be possible to recover the deleted task or any information associated with it. It is for this reason that the user must be sure of the information to be deleted.



(a) Deleting option (b) Deleted Task

Figure 5.10: Deleting a Task

5.3 Focused History

The next icon in the drop-down menu is the "History" icon, which leads to the history page. Here, the user can see the intervals in which he/she has been focused. How these intervals are calculated and recorded has been explained above, so we will not go into more detail in this section. The intervals are arranged in chronological order, and organised by days. They can be calculated either in minutes or seconds. The time shown is the exact minute of the end of the interval. Figure 5.11 shows two views of this page.

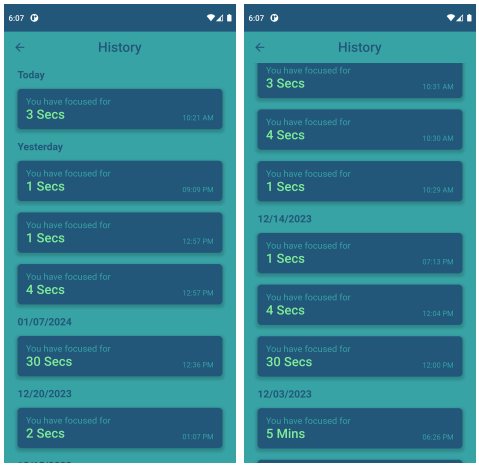


Figure 5.11: History Page

This view allows the user to keep track of the time spent. Both hours and days with productivity peaks can be studied in a simple and visual way, and work can be organised

accordingly. The results shown are experimental and do not reflect the actual work of a user.

5.4 Intervals Settings

The last icon takes the user to the settings page. Here, the interval durations can be adjusted according to the needs of each user. Everyone's work pace is different, and this allows the application to be flexible and adapt to each rhythm. Figure XX shows this page.

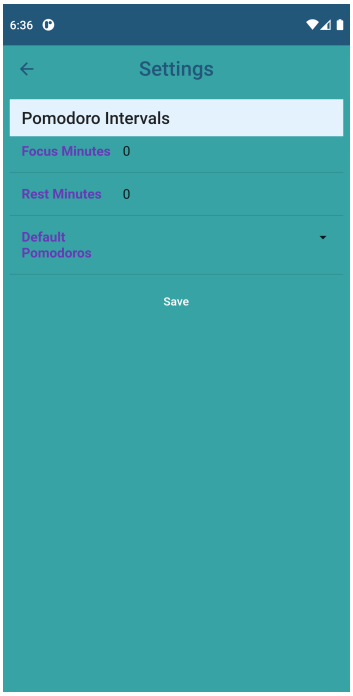


Figure 5.12: Settings Page

If the user wants to follow a specific cycle, they can fill in the "Focus Minutes" and "Rest Minutes" fields and create a completely customised Pomodoro. This gives a greater degree of freedom to the user, who can even vary the duration between consecutive cycles, depending on their energy or accumulated fatigue. Alternatively, you can select one of the default Pomodoros, using the drop-down menu just below:

- Clasic: this is the traditional Pomodoro, with its intervals of 25 minutes of concentration and 5 minutes of rest.

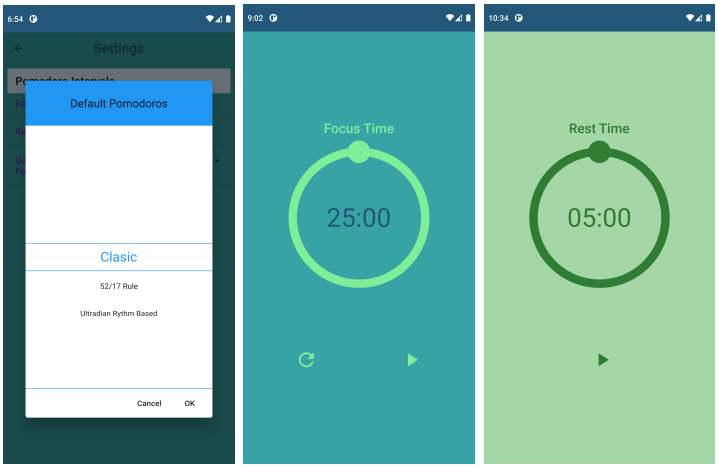


Figure 5.13: Classic Pomodoro

- 52/17 Rule: This method recommends 52 minutes of continuous work and a longer rest of 12 minutes.

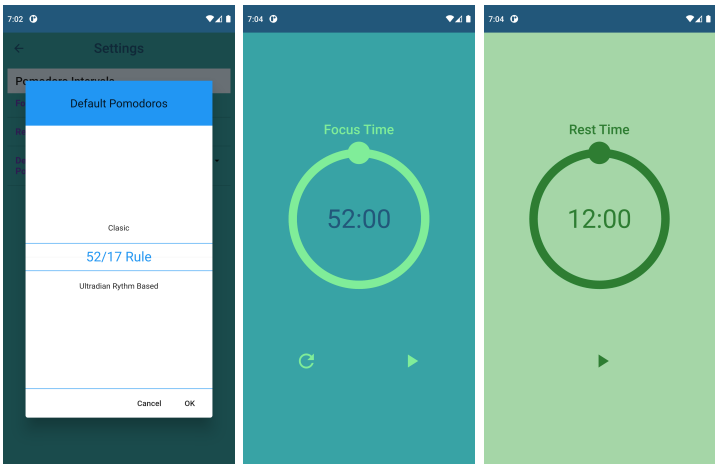


Figure 5.14: 52/17 Rule Pomodoro

- Ultradian Rythm: the latest implemented pomodoro is an extended version, which features 90 minutes of intense work and a proper 20-minute rest.

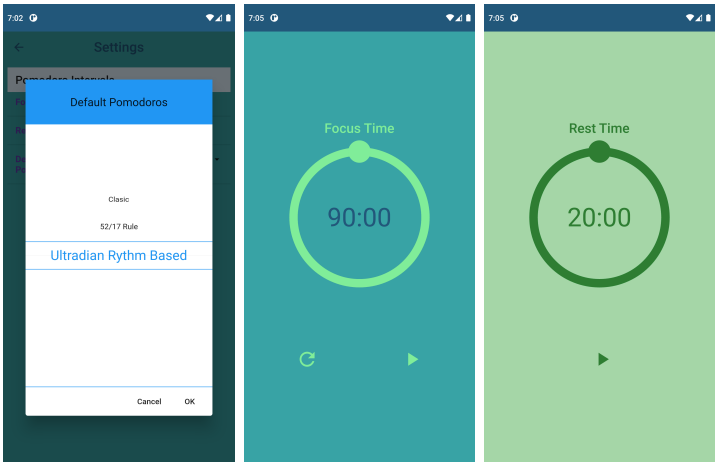


Figure 5.15: 52/17 Rule Pomodoro

CHAPTER 6

Conclusions and future work

In this final chapter, we will analyze the conclusions drawn from the project. Additionally, we will explore possible avenues for future work.

6.1 Conclusions

Procrastination is a common pattern that plays a crucial role in a person's work or academic performance. Despite the false comfort it brings to the individual, it can have disastrous long-term consequences, such as avoidance of responsibilities or accumulation of stress. Numerous techniques and programmes can help to manage time more effectively, but not all of them incorporate the necessary tools, and even fewer are completely free and accessible.

It is in this context that the idea of this project arose, to design and develop an open-source, cross-platform application that helps users to manage their time and workload effectively.

The application is based on the well-known Pomodoro method. This consists of 25 minutes of work without any distractions, followed by 5 minutes of rest. This is repeated cyclically until the task at hand is completed. This method is not suitable for all tasks or all users, so the application offers the user the possibility to set their own customised intervals. There is also the option to use different cycle-based methods, such as the 52/17 rule or intervals based on Ultradian Rhythms.

To manage the tasks to be undertaken, the application also implements various techniques such as task partitioning and task prioritisation, which serve to better manage the workload. The application allows the user to keep track of his tasks, adding or deleting them when necessary. It also implements various filters, to alert the user when to prioritise a task, or to indicate that a task has been completed. They are displayed in a visual and simple way, so that the user can focus on the task, rather than managing it.

The application also has a system that allows the user to see how long they have been concentrating on the task at any given moment. Each time the timer is paused, or a cycle is completed, a record is created of the time the user has spent. This gives you an easy and visual way to see the time spent, as well as being able to look at your productivity peaks and work accordingly.

To design all the functionalities, a study of the main techniques and how they are implemented in related works has been carried out, concluding which were the most effective. Finally, the application has been developed following a modular architecture that allows its scalability, and offers a familiar and intuitive interface that allows the user to focus on their work.

6.2 Achieved Goals

The main objective of this project was to develop an open-source, cross-platform application that implements the necessary techniques and tools to help users reduce procrastination and

increase their productivity. In order to achieve this, a series of goals have been carried out that together have enabled the overall success of the project.

- **Study of the behavioural aspect of procrastination and its impact on the productivity of the subject:** Research has been carried out to understand the sociological component of procrastination in people, and the reasons behind it. The impact of procrastination on productivity has also been studied, as well as the most effective tools to tackle it.
- **Choice of tools and design of the application:** The application has been designed following a previous study of the most used techniques in similar works, as well as the most successful ones. Research has also been carried out on the most suitable framework and the tools to be used.
- **Development and implementation of the application:** A scalable and maintainable application has been implemented following a robust architectural pattern, which facilitates the implementation of future functionalities. In addition, it follows an intuitive and user-friendly design, which favours concentration and study. This idea has been a fundamental pillar during the development of the whole project.

6.3 Future work

The following are a series of ideas for future work streams on which the application could be further developed, expanded and new functionalities added:

- **Integration of gamification techniques:** As seen above, most existing applications incorporate gamification techniques, encouraging the user to use the application. This could be used to motivate the user to stay focused and make proper use of the techniques already implemented.
- **Analysis and graphical representation of user productivity peaks:** By incorporating various diagrams, the user could be helped to better understand their work cycles and how to increase their productivity. By studying the hours of the day with the highest concentration, or the days of the week with the highest workloads, the user could adapt and adapt the workload to his or her person more appropriately.
- **Integration of machine learning techniques:** The implementation of models trained with the work intervals would allow greater flexibility to the application, adapting it specifically to each user. These models could design Pomodoro intervals with the values adjusted to each particular person, as well as efficient management of the tasks, dimensioning them appropriately to the user's work capacity.

- **Inclusion of notifications:** adding notifications or alarms every time a work or rest cycle ends would allow users to be notified without having to look at the screen. In addition, during break time, it would allow for deeper relaxation, as they do not need to keep an eye on the stopwatch to know when to start working again.

APPENDIX A

Impact of this project

This appendix shows the impact in different aspects of the project

A.1 Social impact

The time and workload management application developed in this project has a social impact by helping users to better develop their work or study. Providing tools and raising awareness of the importance of effective time management helps people to perform better in their work environment, with a subsequent increase in performance. This can also lead to improvements in the health of the subject, reducing the levels of stress and pressure to which the user is subjected, and proportionally increasing their rest and health.

A.2 Economic impact

The economic impact of the application can come from increased user productivity. By effectively improving the performance of users in their respective jobs, this can have a beneficial impact on the business. By enabling users to work better, they can undertake greater workloads and increase the profits generated by their respective jobs.

A.3 Environmental impact

Regarding the environmental impact, being a mobile application it has hardly any negative effects on the environment. It does not require any specific resource and is not necessarily polluting. It is worth mentioning that these devices do have some energy consumption that may be affected, but as it is a lightweight application, no additional battery or resource consumption should be noticeable.

A.4 Ethical impact

The application does not require any private or confidential information about its users, so it has little ethical impact. The data is processed and stored on each mobile device, and the user has access to delete it whenever they wish.

APPENDIX B

Economic Budget

This appendix details the economic budget of this project in terms of human and physical resources.

B.1 Human resources

It has been estimated that this project has required a total of 384 hours over a period of 4 months taking into account various tasks such as researching, designing and implementing the application, and finally writing the documentation.

Based on scholarships related to similar jobs within the same school, a salary of €600 has been estimated, with a required effort of about 80 hours per month. This gives an hourly wage of 7.5€. Multiplying this by the estimated total hours spent, gives a human cost of 2,880€.

B.2 Physical resources

During the development of this project, various resources have been used, with their consequent associated costs:

- This computer has been used during the development of the project, thanks to its power and capacity to efficiently move the necessary mobile simulators. The estimated

cost of the laptop is €1150. Taking into account a 5-year amortisation period, the cost of the laptop comes down to €230. Therefore the cost of use for 4 months is 76.6 €.

B.3 Licenses

Due to the limited processing capacity of the free version, a subscription to overleaf's Student plan has been contracted, at a cost of €9.68.

The rest of the software used during the project is open source.

B.4 Table Summary

LABOR COST (direct cost)	**Hours**	**Hourly Rate**	**Total**
	384	7.5	**2880**

MATERIAL RESOURCES COST (direct cost)	**Purchase Price**	**Months of Use**	**Amortization (in years)**	**Total**
Personal Computer (Software included).....	1.159,68	4	5	77,31

TOTAL COST OF MATERIAL RESOURCES	**77,31**

GENERAL EXPENSES (indirect cost)	15 %	over DC	**443,59**
INDUSTRIAL PROFIT	6 %	over DC + IC	**122,43**

SUBTOTAL BUDGET		**3532.33**
APPLICABLE VAT	21 %	**741.78**

TOTAL BUDGET	**4274.11**

Table B.1: Economic Budget

Bibliography

[1] Clockify. https://clockify.me/es/.

[2] Forest App. https://www.forestapp.cc/.

[3] Ultradian Rhythms: How to Achieve Peak Productivity. https://www.kosmotime.com/ultradian-rhythm/.

[4] Cross-platform mobile frameworks comparative. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/.

[5] Flutter Architectural Overview. https://docs.flutter.dev/resources/architectural-overview.

[6] Conquering Procrastination: A Guide to Overcoming the Time Thief. https://www.linkedin.com/pulse/conquering-procrastination-guide-overcoming-time-jaki-wasike-

[7] W Kyle Simpson and Timothy A Pychyl. In search of the arousal procrastinator: Investigating the relation between procrastination, arousal-based personality traits and beliefs about procrastination motivations. *Personality and Individual Differences*, 47(8):906–911, 2009.

[8] Joseph R Ferrari. Procrastination as self-regulation failure of performance: effects of cognitive load, self-awareness, and time limits on 'working best under pressure'. *European journal of Personality*, 15(5):391–406, 2001.

[9] Piers Steel. The nature of procrastination: a meta-analytic and theoretical review of quintessential self-regulatory failure. *Psychological bulletin*, 133(1):65, 2007.

[10] LQ Liu, G Min, ST Yue, and LS Cheng. The influence of mobile phone addiction on procrastination: a moderated mediating model. *J Ergon*, 8(232):2, 2018.

[11] Joseph R Ferrari, Judith L Johnson, and William G McCown. *Procrastination and task avoidance: Theory, research, and treatment*. Springer Science & Business Media, 1995.

[12] Miguel Angel Cardoso Miranda and Alfredo Daza Vergaray. Mobile gamification applied to employee productivity in companies: A systematic review. *TEM Journal*, 10(4):1869, 2021.

[13] Patti Shih, Lisa Watts, Suzanne Mobbs, Helen Scicluna, Boaz Shulruf, and Rachel Thompson. Time management apps and student success: Pilot study and focus group findings, 2014.

[14] Luis Leiva, Matthias Böhmer, Sven Gehring, and Antonio Krüger. Back to the app: the costs of mobile application interruptions. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 291–294, 2012.

[15] 52/17 Rule. https://www.themuse.com/advice/the-rule-of-52-and-17-its-random-but-it-

[16] Gilad Bracha. *The Dart programming language*. Addison-Wesley Professional, 2015.

[17] Dart Overview. `https://dart.dev/overview`.

[18] How Flutter is All Set to Redefine App Development Trends?. `https://www.biztechcs.com/blog/flutter-app-development-future-trends/`.

[19] Flutter Features. `https://www.educative.io/answers/what-are-the-features-of-flutter`.

[20] Daryl Kulak and Eamonn Guiney. *Use cases: requirements in context*. Addison-Wesley, 2012.

[21] Thomas von der Maßen and Horst Lichter. Modeling variability by uml use case diagrams. In *Proceedings of the International Workshop on Requirements Engineering for product lines*, pages 19–25, 2002.

[22] Antonia Bertolino, A Fantechi, Stephania Gnesi, G Lami, and A Maccari. Use case description of requirements for product lines. In *Proceedings of the international workshop on requirements engineering for product lines*, pages 12–18, 2002.

[23] Hive package documentation. `https://pub.dev/packages/hive`.

[24] The growth of Flutter development. `https://medium.flutterdevs.com/the-growth-of-flutter-development-3years-after-the-birth-of-alpha-78baee809dff`.