A Motivational Assistant in the ActiveUP System for Frailty Prevention in Older Adults

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE ENGINEERING – INGEGNERIA INFORMATICA

Author: **Luca Leoni**

Student ID: 963082
Advisor: Prof.ssa Franca Garzotto
Academic Year: 2021-22

Abstract

The percentage of the elderly population is increasing. Consequently, living a healthy aging is one of the great challenges of the current century. Societies that adapt to this demographic change and invest in healthy aging can enable people to live longer and healthier lives [1].

More and more older people are in need of care. However, the number of potential caregivers is not satisfactory for the current demand.

Frailty is a clinically recognizable state of increased vulnerability resulting from deterioration associated with aging [2].

To ensure autonomy and a higher quality of life, it is essential to prevent the progression of frailty.

As people age, certain physiological and cognitive changes are almost inevitable. And while many people over 60 have been surrounded by technology for most of their adult lives, there is a need to compensate for these physiological and cognitive changes.

However, some older adults are not as comfortable or familiar with technology in general, or with more specific things like mobile devices. They need incentives to participate.

The goal of this master's thesis is to implement a motivational assistance component to be integrated into the ActiveUP project, a national initiative created to counteract frailty with the help of tailored exercise plans for older adults. The purpose of providing project participants with additional motivational assistance is to prevent non-adherence to exercise plans. Indeed, even if the user is aware of the benefits of exercising, maintaining regularity is the biggest challenge of intervention plans for older people.

The work is based on *"Application of Motivational Traits, Motivational State and Stage of Change Modelling for Frailty Prevention in Older Adults"* by Natalia Sokól [3]. Sokól created a first prototype of a motivational assistance software that uses information about older patients' motivational traits, their level of motivation (motivational state) and their degree of internalization of the daily exercise habit (stage of change) to adjust the user interface and initiate personalized interactions.

This paper presents new advances in the design and implementation of the Motivational Assistant to support older adults during their training program. The work began with an analysis of the component developed so far and identification of code-related problems. Also, the document describes improvements in the ActiveUP system, such as the migration to the new server and to a brand new Android application. Then, it provides a detailed description of the design of new features as well as the implementation. In particular, the document describes the Leader board, the Challenges, the User statistics, and the Alerts' mechanism Finally, the manuscript reports the usability evaluation performed by an expert in the matter.

Key-words: usability, older adults, accessibility, exercise, frailty.

Abstract in italiano

La percentuale di popolazione anziana è in aumento. Di conseguenza, vivere un invecchiamento sano è una delle grandi sfide del secolo attuale. Le società che si adattano a questo cambiamento demografico e investono in un invecchiamento sano possono consentire alle persone di vivere più a lungo e in salute [1].

Sempre più anziani hanno bisogno di assistenza. Tuttavia, il numero di potenziali assistenti non è soddisfacente per l'attuale domanda.

La fragilità è uno stato clinicamente riconoscibile di maggiore vulnerabilità derivante dal deterioramento associato all'invecchiamento [2].

Per garantire l'autonomia e una maggiore qualità di vita, è essenziale prevenire la progressione della fragilità.

Con l'invecchiamento, alcuni cambiamenti fisiologici e cognitivi sono quasi inevitabili. Sebbene molti ultrasessantenni siano stati circondati dalla tecnologia per la maggior parte della loro vita adulta, è necessario compensare questi cambiamenti fisiologici e cognitivi.

Tuttavia, alcuni anziani non si sentono a proprio agio o non hanno familiarità con la tecnologia in generale, o con cose più specifiche come i dispositivi mobili. Hanno bisogno di incentivi per partecipare.

L'obiettivo di questa tesi di master è quello di implementare una componente di assistenza motivazionale da integrare nel progetto ActiveUP, un'iniziativa nazionale creata per contrastare la fragilità con l'aiuto di piani di esercizio su misura per gli anziani. Lo scopo di fornire ai partecipanti al progetto un'assistenza motivazionale aggiuntiva è quello di prevenire la mancata aderenza ai piani di esercizio. Infatti, anche se l'utente è consapevole dei benefici dell'esercizio fisico, il mantenimento della regolarità è la sfida più grande dei piani di intervento per gli anziani.

Il lavoro si basa su *"Application of Motivational Traits, Motivational State and Stage of Change Modelling for Frailty Prevention in Older Adults"* di Natalia Sokól [3]. Sokól ha creato un primo prototipo di software di assistenza motivazionale che utilizza informazioni sui tratti motivazionali dei pazienti anziani, sul loro livello di motivazione (stato motivazionale) e sul loro grado di interiorizzazione dell'abitudine all'esercizio quotidiano (stadio di cambiamento) per regolare l'interfaccia utente e avviare interazioni personalizzate.

Questo articolo presenta i nuovi progressi nella progettazione e nell'implementazione dell'Assistente Motivazionale per supportare gli anziani durante il loro programma di allenamento. Il lavoro è iniziato con un'analisi del componente sviluppato finora e con l'identificazione dei problemi legati al codice. Inoltre, il documento descrive i miglioramenti apportati al sistema ActiveUP, come la migrazione al nuovo server e a una nuova applicazione Android. Fornisce poi una descrizione dettagliata della progettazione di nuove funzionalità e della loro implementazione. In particolare, il documento descrive la classifica, le sfide, le statistiche degli utenti e il meccanismo delle notifiche.

Parole chiave: usability, older adults, accessibility, exercise, frailty.

Contents

1 Introduction

1.1. Context

The global number of people aged 60+ years is continuously increasing. In 2019, the number of people aged 60+ years was 1 billion. This value is estimated to rise to 1.4 billion by 2030 and 2.1 billion by 2050. This increase is occurring at an unprecedented pace and will accelerate in the coming decades [1].

An increased number of seniors means coping with several different situations. Indeed, there is no prototype for older people. Some 80-year-olds have physical and mental capacities similar to many 30-year-olds. On the other hand, other people experience significant declines in abilities and need assistance. This condition turns on frailty.

Frailty is a clinically recognizable state of increased vulnerability resulting from an ageing-associated decline in reserve and function across multiple physiologic systems as the ability to cope with everyday life. Identified signals are low grip strength, low energy, slowed walking speed, and low physical activity [2].

Preventing the progression of frailty is crucial to guarantee autonomy and a better quality of life.

In Europe, several projects are currently on to ensure the prevention of frailty.

Started in June 2020, the ActiveUP project aspires to implement a personalized assistance framework that promotes active ageing activities for the elderly. Unfortunately, the adherence rate to exercise programs observed in previous projects is lower than expected. For this reason, the researchers think involving motivation may be the correct technique to prevent treatment abandonment.

1.2. Thesis objectives

The objectives of this thesis are directly related to the progress of the national ActiveUP project. Following the piloting phase of POSITIVE and the usability evaluation of the ActiveUP application, changes and improvements had to be made in order to correct the flaws identified. The chosen approach was to create a new Android application for patients to integrate the Vivifrail program into.

Also, during the POSITIVE project cycle, the team noticed how the server in NodeJS had drawbacks in terms of concurrency and reliability. For this reason, it was decided to migrate to a new server in Python, based on the FastAPI framework.

Finally, as researched by Fernández-Avilés Pedraza [4], Zorzenon [5], and Sokól [3], involving motivation in the exercise plan can prove to be the trump card for increased adherence. To achieve this, building on Sokól's work [3], the thesis proposes a refactoring of the component developed so far. In addition, to incorporate features mentioned by Sokól [3], the design and implementation of the motivational assistant able to adapt itself according to the motivational traits of the patients are described. In particular, the incorporated features are the Leader board, the Challenges, the User statistics, and the Alerts mechanism.

1.3. Structure of the document

The Master Thesis consists of ten chapters.

The first one explains the context in which the work is carried out. Chapter 2 introduces some theoretical backgrounds and the works already done in this field. Chapter 3 provides an overview of the system at the time the development of this motivational assistant component began. The fourth chapter describes the problems encountered and presents the thesis objectives. The migration to the new server, with all the details, is covered in Chapter 5. Chapter 6 explains the migration to the new version of the android application for patients. Chapter 7 describes in-depth the new features of the motivational assistant. A Cognitive Walkthrough is described in Chapter 8. Finally, conclusions and future works are explained in the ninth chapter. Chapter 10 contains the references of all the consulted manuscripts.

2 Theoretical Foundations

2.1. User-centered design

With the term User-Centered Design (UCD), we refer to a collection of techniques that place consumers at the center of product development and design. Nowadays, it is imperative to understand our users, their routines, and their requirements to continuously design and implement with these elements in mind.

The UCD acknowledges that any innovation must begin with a user understanding. The basic idea of user-centered design is that the end user's needs, abilities, and desires drive the design at every step of the process. From the beginning of the software life cycle, user scenarios, personas, and requirements are produced, assessed, and evaluated.

UCD is frequently used interchangeably with human-centered design. User-centered design necessitates a more in-depth examination of users and your target audience.

UCD considers age, gender, social position, education, and professional background, influencing factors, product usage expectations and needs, and other characteristics that may differ for various segments.

Human-computer interaction user-centered design techniques allow us to understand our users and their patterns and requirements to design iteratively with these features in mind and then evaluate the design's usability and influence on daily routines, behaviors, and activities [6].

- The design process relies on the knowledge of people, tasks, and surroundings.
- Users are involved in the design and development process from start to finish.
- User-centered assessment drives and modifies design.
- The procedure is iterative.
- The design regards the entire user experience.
- The design team consists of individuals with diverse talents and perspectives.

UCD is an interconnected set of circles encompassing eight indicators. The user joins a circle comprising "Context, Objectives, Environment, and Goals" and a layer representing "Task Detail, Task Content, Task Organization, and Task Flow." (Figure 2.1).

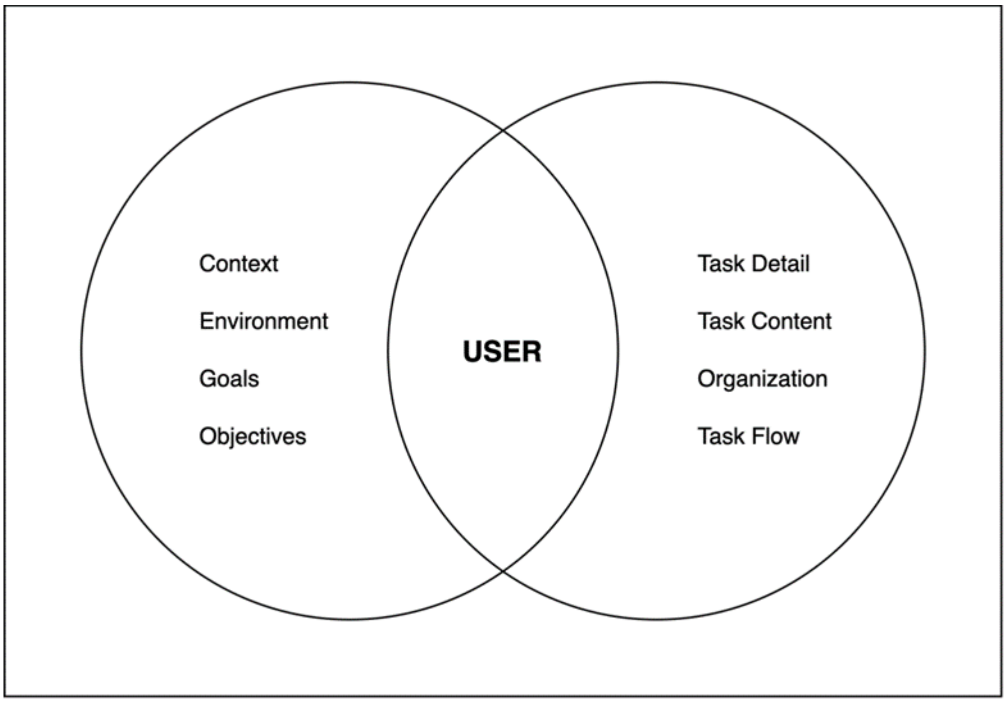


Figure 2.1: User-centered design definition

According to [7], the UCD is founded on three core ideas:

- Early focus on users and tasks: The user should be involved starting from the product's origin. The earlier the user gets involved, the less maintenance work is required at the end of the life cycle.
- Product usage metrics based on empirical data: The emphasis here is on conventional usability: simplicity of learning and effective, error-free operation through prototype usability testing.
- Iterative design: The product should be created, changed, and tested regularly. The goal behind the iterative design is to fail early; changing the user interface of an early prototype is considerably easier than changing the user interface of a deployed system.

The objective of User-Centered Design is to create usable products. According to [8], there are four general phases of the User-Centered Design process, as presented in Figure 2.2:

- Understand and specify the context of use: Establish who the product's consumers will be, why they will use it, what their requirements will be, and the context they will use it.
- Specify the user and organization requirements: At each step of UCD, product designers conduct usability testing to gather input from consumers on the product.

- Produce design solutions: Begin an iterative process based on product goals and requirements.
- Evaluate design against requirements: Check if the proposed solution matches the specifications. If yes, the process ends. Otherwise, some modifications are needed.

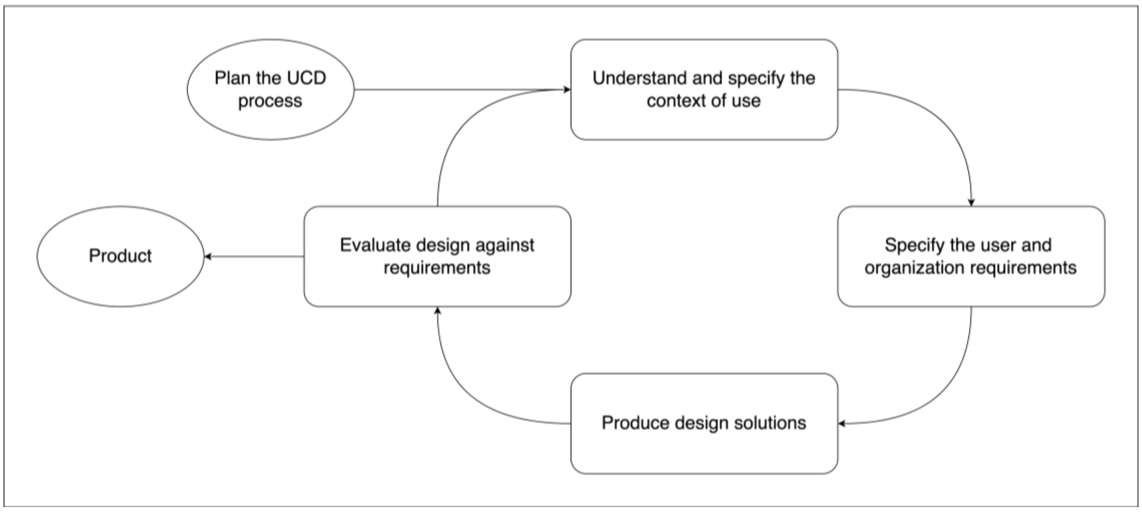

Plan the UCD process
Understand and specify the context of use
Product
Evaluate design against requirements
Specify the user and organization requirements
Produce design solutions

Figure 2.2: User-centered design process

2.2. Design for older adults

The user target of the ActiveUP project is older adults. This term refers to people aged over 65. When dealing with technology, this segment proved to be weak. According to [9], older adults face several unique barriers and challenges when it comes to adopting new technologies. These include physical challenges to using them, skeptical attitudes about the benefits, and difficulties learning to use new apparatus. Nielsen heuristics [6] were not enough to overcome these obstacles. The framework adopted in this work was based on the "*Heuristics established by a systematic review of guidelines to design mobile technology for older adults*") [10]. Since this Master Thesis is about the implementation and integration of a component and not a new application, not all the heuristics are applicable. The ones considered during the whole process are the following:

Table 2.1: Usability Heuristics for Older Adults [10]

UHOA[1]	**Heuristic**
4	Simplify the navigation, reduce the number of alternatives, and maintain the focus on the current action.

[1] Acronymous for Usability Heuristics for Older Adults

5	Provide back and exit buttons as a safe exit.
10	Reduce the number of available elements, options, and actions on the screen.
11	Use icons that are concrete and familiar images.
12	Add labels to icons.
13	Use semantically close icons.
14	Clearly show which elements are touchable.
15	Provide high contrast between foreground and background color.
16	Use simple, familiar, and unambiguous language. Don't rely on the user to remember information.
17	Don't assume familiarity with conventional symbols as "?", "+", "→".
19	Maintain instructions and messages short.
20	Favor control tapping over gesture interaction.
23	Increase the distance between interactive controls.
26	Show clear feedback after control tapping, subtle feedback might not be noticed.

2.3. Vivifrail

The Vivifrail project [11] is an exercise promotion program, an international benchmark for community and hospital-based interventions aimed at preventing frailty and falls in the elderly.

The entire project is based on the idea that health in the elderly should be measured in terms of functionality and not as a disease that determines life expectancy, quality of life, and the resources or supports each population needs. The goal is to maintain the level of functionality that preserves as much independence as possible in each case.

The evidence based Vivifrail application allows health care providers to classify older people according to their risk of loss of functional capacity, dependency, and risk of falls. The assessment consists of the Short Physical Performance Battery (SPPB or Guralnik test), the 6-meter walk speed test, the get-up and go test, and the fall risk assessment.

The SPPB is a test in which the participant is asked to perform three tasks. First, the Balance Test in three different positions (feet together, semi-tandem, and tandem). Next, the Walking Speed Test and the Chair Stand Test are performed.

One of the most widely used assessments to examine movement-related physical function involves measuring the time it takes to walk 6 meters at the patient's usual speed. Finally, the Get-up and Go Test combines the assessment of aspects related to strength, balance, and walking and is considered a good test to assess the risk of falling in a frail elderly person.

Once the participant is assessed, the algorithm proposes a tailored multicomponent physical training program. The training program consists of a series of exercises that allow, depending on the level of functional capacity (i.e., severe limitation, moderate limitation, and mild limitation), to develop arm and leg muscle strengthening, balance retraining, and flexibility. By following these simple programs, an elderly participant can improve functional capacity and avoid frailty syndrome and the risk of falls.

Depending on the elderly person's level of functional ability (severe limitation, moderate limitation, and mild limitation) and fall risk, up to six different types of exercise programs can be assigned:

- A: Disability (elderly person who cannot get up from a chair or bed),
- B and B+: Frailty (an elderly person who can walk with difficulty or with help),
- C and C+: Pre-fragility (an elderly person who has slight difficulty walking and also has difficulty getting up or maintaining balance), and
- D: Robust (older adult with minimal physical limitations).

To be effective, the program must be followed for 12 weeks. During this period, the participant should see the first signs of improvement. After the 12 weeks, the initial assessment tests should be repeated, and a new tailored program should be assigned.

Ideally, the participant should end the program in a healthier state regardless of the assigned program.

According to [12], the key factor is consistency in exercise. This is a critical aspect of Vivifrail, as lack of adherence to exercise plans tends to be high in older adults.

2.4. Positive

POSITIVE [13] is a project that integrates the Vivifrail exercise set in a comprehensive system to bring care home by constantly monitoring a patient’s intrinsic capacity and sounding an alarm when a decline in that capacity may indicate the onset of a disease or disability. The system unites a person’s full community of caregivers, bringing together the patient, caregiver, and primary and specialized care professionals (Figure 2.3).

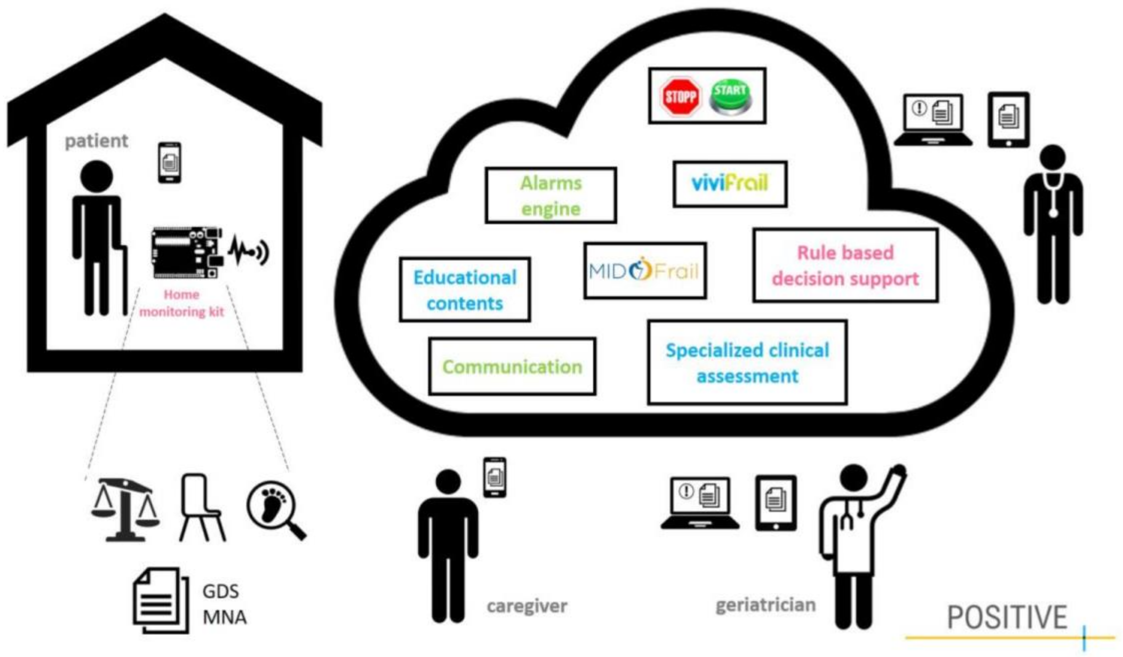


Figure 2.3: POSITIVE conceptual map

POSITIVE was led by SERMAS, through its EIP-AHA reference site (Getafe University Hospital). All partners involved are among the leading institutions in the relevant segments in their countries. They also represent a broad range of European locations: UPM and ATOS in Spain, Karolinska Institut in Sweden, and the Medical University of Lodz in Poland.

Using remote monitoring of a person's intrinsic capacity, POSITIVE provides a technological infrastructure to enable a new organizational model for elderly care that involves all the relevant actors: patient, caregiver, and primary and specialized care.

Patients are assessed constantly, and as needed, a tailored physical activity program is automatically prescribed to maintain or improve their condition. Caregivers are aware of any decline and can check whether the senior being monitored is alright at any given time. Primary care professionals can process alarms related to dangerous declines of intrinsic capacity and adjust treatments accordingly, with the assistance of a decision support system. As the remote system keeps them in regular contact, primary caregivers can involve specialized care if needed.

The pilot phase lasted 6 months. According to the findings of this experiment, most of the participants were not using the application on their own, some faced difficulties with the user interfaces and some did not see the advantages of the system.

2.5. ActiveUP

The R&D national project ActiveUP starts from the previous POSITIVE project. The findings of the pilot phase in POSITIVE suggested possible improvements to the system.

The goal of ActiveUP is to provide an unobtrusive and personalized system for assessing and managing frailty, promoting active ageing, and preventing dependency on the elderly.

The design and deployment of a customized software support system particularly targeted at senior users are part of ActiveUP. This framework should be able to construct a user model that includes important personal attributes and psychological qualities, infers motivational state from observed behavior, and determines the present stage of change. Based on this model, the system will suggest appropriate therapeutic intervention adjustments while also providing appropriate motivational and empowerment support. The ultimate aims of individualized support are to reduce treatment desertion, increase older adult empowerment to prevent frailty, and therefore promote a healthier ageing process.

The ActiveUP ecosystem (see Figure 2.4) is designed as an Internet of Things (IoT) infrastructure to evaluate frailty and functional capacity without interfering with daily life and with the least amount of pain to the person.

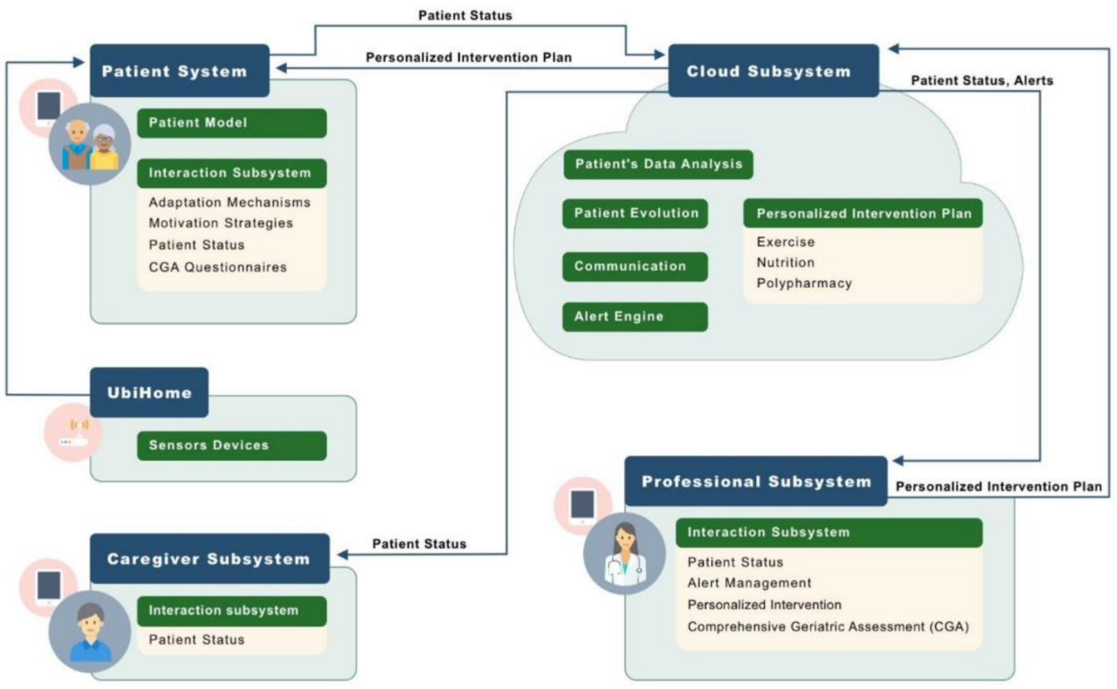


Figure 2.4: ActiveUP ecosystem

The ActiveUP system consists of a wearable sensor (IMU), central node (Raspberry Pi), Cloud Server and APP for Tablet (Patients APP).

The Raspberry Pi acts as a central node for the system at the patient's house. The wearable sensor is connected to it via WiFi.

This central node has a service running in Python inside a Docker container that establishes the connection between the sensors and the Raspberry Pi via FTP and MQTT protocol which is a standard for IoT communications. The Raspberry Pi acts as a broker and the sensor as a client. Also this service is responsible for collecting the data sent from the sensors and saving it into a ORACLE database.

The cloud server is developed using Python, more specifically FastAPI which is a standard-based framework that lets us build fast, intuitive and robust APIs easily. It is also one of the fastest Python frameworks to build APIs.

During the course of the ActiveUP project, some improvements were made, and some adjustments were implemented to the POSITIVE system.

The figure of health care professionals is not in the scope of ActiveUP because of budget constraints. For this reason, starting from the POSITIVE application for patients, the possibility to send messages to doctors is not available anymore. So, the focus is on patients, and the mobile applications for caregivers and care professionals are now discontinued.

2.6. Motivation

As stated in his work [12], adherence to the exercise plan is crucial. Lack of training will influence patients' outcomes and consequently their quality of life.

Motivation, the need, desire, or willingness to do something to achieve the desired outcome, plays a significant role in the adherence to any type of exercise activity or program [14].

Accordingly, an individual’s motivation to engage in exercise may be influenced positively or negatively by internal factors (e.g., knowledge of exercise benefits) and by external factors (e.g., support of others). Exercise self-efficacy, the confidence a person has in their ability to develop and meet physical or cognitive exercise goals, is also key to exercise motivation.

While older adults acknowledge theoretical awareness of exercise benefits and express a desire to adopt exercise regimes, their intentions often fail. Multiple motivational barriers have the potential to interfere with their ability to successfully engage in exercise programs. Documented motivational barriers to initiation of physical or cognitive exercise include an individual’s decreased insight into their own need to exercise, absence of exercise goals, and absence of specific information regarding what would constitute a beneficial exercise program [15] [16] [17].

In recent years, some students have looked in depth at the topic of motivation as applied to older people and how it can play a decisive role in adherence to exercise programs.

Daniel Fernández-Avilés Pedraza, in his doctoral thesis, proposed an ontology and architecture for motivational systems. He explained that combining motivation and technology could bring a higher adoption rate and better user experience for these applications. Identifying a person with specific motivational traits will allow understanding the behavior related to a specific challenge and taking action to prevent undesired conduct.

Luca Zorzenon, starting from the PhD manuscript [4], designed and implemented a new gamified test for the motivational traits assessment in the elderly. The game aims at extracting the user's motivational profile. This information will be consequently used to personalize and customize the ActiveUP application for patients.

Finally, Natalia Sokól set the foundations of the motivational assistant to be integrated into the ActiveUP application for patients. A detailed explanation of this component will be provided in the section Initial state of the Motivational Assistant.

3 Antecedents

The chapter aims at explaining the original architecture of the POSITIVE system, as well as the database and the so-far implemented motivational assistant component.

3.1. POSITIVE architecture

The original POSITIVE architecture (Figure 3.1) included a cloud server for data storage, measurement sensors, and a number of applications, including:

- Patient Mobile App: mobile application the patient uses at their home.
- Professionals Mobile App: mobile application used in the hospital by health professionals to check on the patients.
- Caregiver Web App: web application used by the caregiver to review the patient's evolution.

A weight scale, a chair stand sensor, and a gait speed sensor are among the Bluetooth measuring devices, directly connected to the Patient Mobile App.

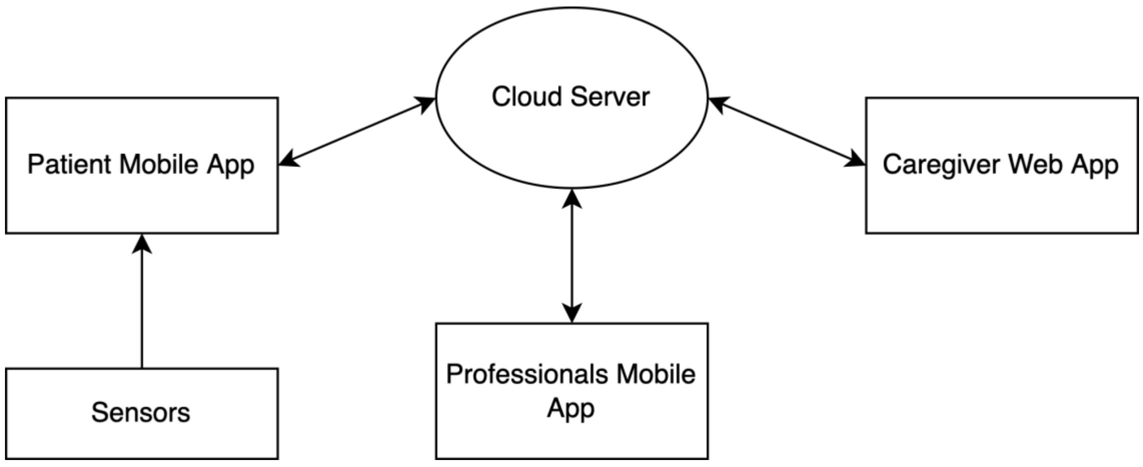


Figure 3.1: POSITIVE Architecture

3.2. POSITIVE Patient app

The POSITIVE application for patients is a tablet-based software. It counts several services:

- Perform chair stand, gait speed, and weight scale tests, using Bluetooth devices.

- Answer to questionnaires.
- Check the results of already performed tests.
- Support the performance of the Vivifrail exercises.
- Watch videos about nutrition.
- Read the manuscript of the assigned Vivifrail plan.
- Send messages to health care professionals.

The communication between the patient application and the server relies on the Internet connection. In order to avoid loss of data, all the data are temporarily stored in Shared Preferences, and uploaded to the server once the connection is stable. Shared Preferences is an interface for accessing and modifying data. This class provides strong consistency guarantees and for this reason it is used for storing data that can be accessed frequently.

Figure 3.2 presents the UML (Unified Modeling Language) representation of the architecture of the POSITIVE Patient app at the package level.

The structure of the system is as follows:

- es.upm.ctb.positive:
  - activities: this package contains all classes defined as extensions of Activity. An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI. All activity classes must have a corresponding <activity> declaration in their package's AndroidManifest.xml,
  - objects: this package contains objects that are used within the system. They are created with the statement new ObjectName(params),
  - util: contains several files in charge of different functions, such as detection algorithms,
  - rest: this package oversees the RestAPI. All the requests are declared in the RestAPI file,
  - firebaseutils: file CSVWriter and JSONFlattener are stored.

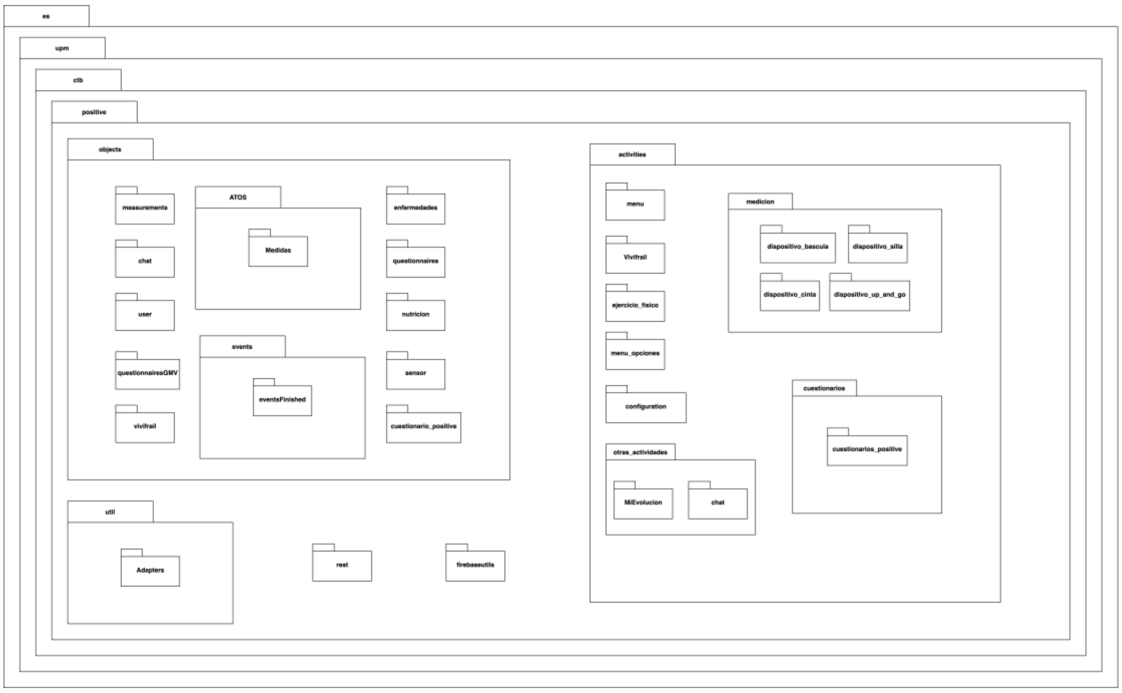


Figure 3.2: UML Positive_pacientes_android (February 2022)

3.3. POSITIVE server

The POSITIVE server is implemented in Node.js, a back-end JavaScript runtime environment that executes JavaScript code.

The server uses RabbitMQ, an open-source message-broker software [18]. It gives the application a common platform to send and receive messages, and the messages a safe place to live until received.

The server used originally in ActiveUP was the one implemented for the POSITIVE project from the beginning.

The structure (https://gitlab.com/AgeingLab/servidorpositive) is as follows:

- caidas: contains the file caidas.js, responsible for everything related to falls,
- configuration: this folder contains the sub-folders (backup_positive, f_config, and servicio_positive),
- pacientes: this folder contains all the files related to the patient's mobile application.
  - comprobar_actividades.js
  - comprobar_asistente.js
  - comprobar_vivifrail.js
  - consumidor_rabbit.js
  - pacientes.js
  - publicador_rabbit.js

- profesionales: this folder is responsible for all the files related to the professional's application.
  - actividades_aplicacion_profesionales.js
  - profesionales.js

The server uses Cron Jobs for scheduling tasks. Cron Jobs automate specific commands or scripts on the server to complete repetitive tasks automatically [19].

To specify at what time a specific job must be executed, it is important to edit the crontab file by adding a new line, using the proper syntax:

```
        m     h     dom   mon   dow   user command
        00    00    *     *     *     ageinglab  cd
/var/www/api/positive;   node   pacientes/eval_asistente.js   >>
logs/eval_asistente.log
```

Where m: minute, h: hour of the day, dom: day of the month, mon: month, dow: day of the week. For example, the command above is executed every day at midnight.

The database structure is directly defined and initialized in phpMyAdmin. phpMyAdmin is a free software tool, written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. The database server used in POSITIVE is MariaDB, which is an open-source relational database.

Figure 3.3 shows the entity-relationship diagram of the original POSITIVE database. It stores some data and information no longer in use in ActiveUP, such as the messages between patient and health care professionals.

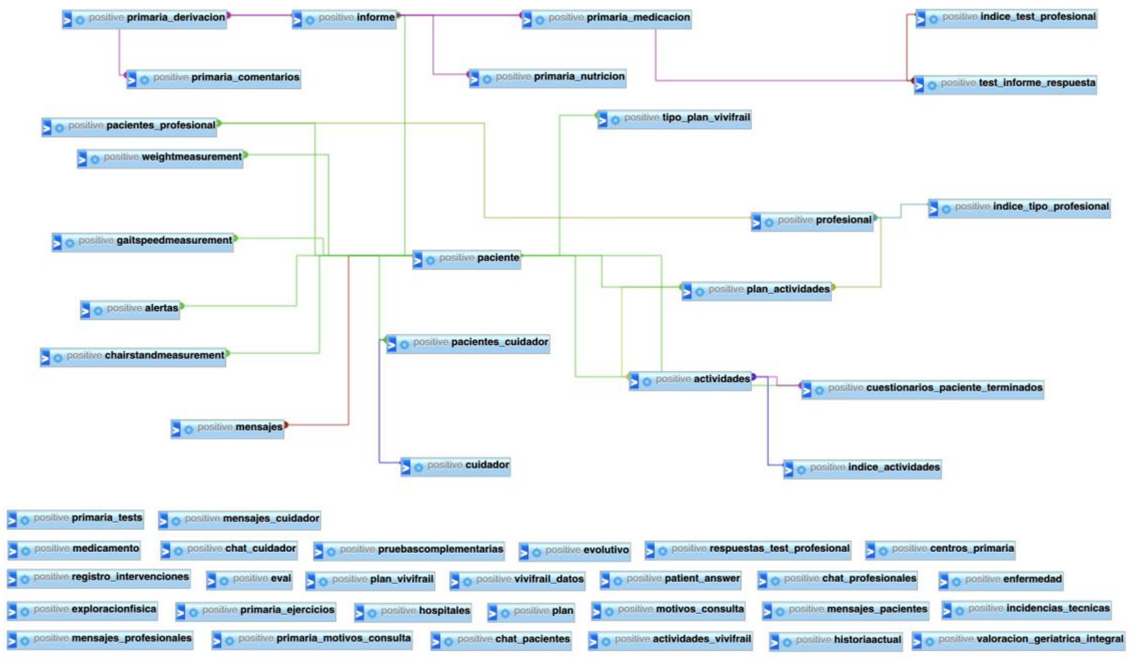

positive primaria_derivacion
positive informe
positive primaria_medicacion
positive indice_test_profesional
positive primaria_comentarios
positive primaria_nutricion
positive test_informe_respuesta
positive pacientes_profesional
positive tipo_plan_vivifrail
positive weightmeasurement
positive profesional
positive indice_tipo_profesional
positive gaitspeedmeasurement
positive paciente
positive plan_actividades
positive alertas
positive pacientes_cuidador
positive chairstandmeasurement
positive actividades
positive cuestionarios_paciente_terminados
positive mensajes
positive cuidador
positive indice_actividades
positive primaria_tests
positive mensajes_cuidador
positive medicamento
positive chat_cuidador
positive pruebascomplementarias
positive evolutivo
positive respuestas_test_profesional
positive centros_primaria
positive registro_intervenciones
positive eval
positive plan_vivifrail
positive vivifrail_datos
positive patient_answer
positive chat_profesionales
positive enfermedad
positive exploracionfisica
positive primaria_ejercicios
positive hospitales
positive plan
positive motivos_consulta
positive mensajes_pacientes
positive incidencias_tecnicas
positive mensajes_profesionales
positive primaria_motivos_consulta
positive chat_pacientes
positive actividades_vivifrail
positive historiaactual
positive valoracion_geriatrica_integral

Figure 3.3: Legacy database architecture

3.4. Initial state of the Motivational Assistant

The Master Thesis developed by Sokól [3] proposes a prototype of a motivational assistance component to be integrated into the ActiveUP project.

The designed motivational assistance software uses information about elderly patients' motivational traits, their level of motivation (motivational state), and their degree of daily exercise habit internalization (stage of change) to adjust the user interface and initiate personalized interactions.

In her document, Sokól firstly described the motivational assistance software design. It consists of three main parts:

- **Key Factors**: refer to motivational traits, motivational states, stage of change, and user history.
  - **Motivational traits** are defined as relatively stable attributes describing how individuals approach goal-oriented situations [20]:
    - **Determination (D)** – being more likely to have high levels of motivation and pursuing achievement,
    - **Personal Mastery (PM)** - being motivated by the desire of constant improvement of one's performance as well as by the need of gaining new knowledge and experience,
    - **Other-Referenced Goals (ORG)** - pursuing status and recognition in the eyes of others and using others' performance as a point of reference for the determination of one's own objectives,
    - **Competition Seeking (CS)** - striving for competition and being motivated by a possibility to outperform others,
    - **Failure Avoidance (FA)** - dislike of evaluative situations and a tendency to avoid situations with a possibility of failure.

    Motivational traits in patients can vary continuously. For this reason, each motivational trait is assigned a degree of prevalence. The used values are the following:

    - Very Low,
    - Low,
    - Medium,
    - High,
    - Very High.
  - **Motivational states** are defined as a transient characteristic related to motivation [21]. In Sokól's design the motivational state can take three values:
    - High,
    - Medium,

        - Low.
    - **Stages of change** are behavioral stages described in the Transtheoretical Model [22]:
        - **Precontemplation** – a stage characterised by a lack of awareness; the patient is lacking knowledge about the reasons that speak for the habit of daily exercise or has doubts about the benefits,
        - **Contemplation** – a stage of ambivalence, in which the patient is aware of the benefits of exercise but also concerned with the difficulties and disadvantages that they may encounter on their way to habit change,
        - **Preparation** – the stage characterised by being fully convinced about the need for daily exercise but before having started to exercise,
        - **Action** – a stage in which the patient is already realising regular exercise but is still prone to relapse,
        - **Maintenance** – a stage achieved after following the exercise programme for a longer time, when the risk of falling into old habits is considerably lower.
        - User History comprises the information recorded during interactions with the intervention and assistance software.
- **Key Resources**: refer to the resources that can be applied to influence the motivational state of the user, namely the motivational techniques and elements that can be added to the user interface to increase the overall motivational performance. Among the resources defined by Sokól, there are Awards, Challenges, Leader board, User statistics, and Messages from the motivational assistant.
- **Key Situations**: refer to significant events which may require the motivational assistant to act. Examples of Key Situations are:
    - First use of the application,
    - Application Start after a Long User Absence,
    - Application Start after an Interrupted Exercise Session,
    - End of an Exercise Session.

In the designing phase, the techniques to be applied for each motivational trait are defined. To be more precise, the Leader board is available only for Competition Seekers with a value of Failure Avoidance lower than High. Then, Challenges and User statistics are only for patients with Personal Mastery higher than Competition Seeking and Other-Referenced Goals, and Failure Avoidance lower than High.

After the initial motivational trait diagnosis and stage of change assessment, the user profile should include the level of each motivational trait and the initial stage of change. In the document, the algorithm for motivational modelling is explained in detail. Here, Figure 3.4 illustrates all possible transitions between the stages of change and motivational states.

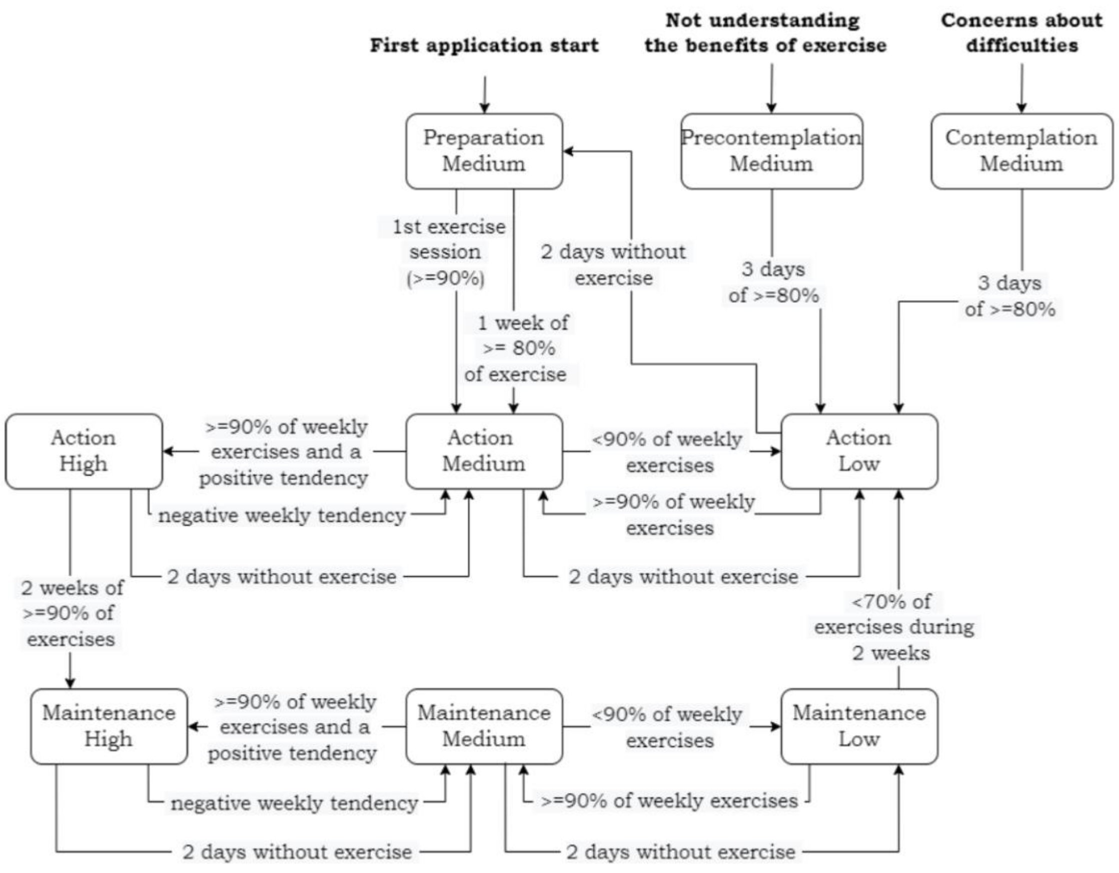


Figure 3.4: Possible transitions between stages of change and motivational states in the motivational assistance system proposed by Sokól

Furthermore, a complete and very precise list of messages to be delivered to the patient was provided. Here, Table 2 shows an example of feedback after a decision to start an exercise session, as proposed by [3].

Table 3.1: The messages designed to give the user positive feedback after a decision to start exercise depending on user's motivational traits and stage of change

	Precontemplation	**Contemplation**	**Preparation**	**Action/ Maintenance**
D	"Fantastic! Regularity means constant improvement."	"Brilliant! Regular exercise makes you feel better and have more energy."	"That is wonderful! Exercise will help you maintain your progress."	"Excellent! There is nothing better for your health than exercise."

PM	"Awesome! You are one step closer to achieving your goals."	"That is wonderful! The time spent exercising always pays off."	"Brilliant! Regular exercise is easy once you make it a habit."	"Awesome! Regular exercise prevents many diseases."
CS	"Awesome! You are one step closer to advancing in the classification of patients."	"Brilliant! Your results are remarkable."	"Wonderful! This way you will maintain your excellent results."	"Fantastic! Keep up the good job and you will be one of the top project participants."
ORG	"Awesome! Your dearest ones will be very proud of you."	"Brilliant! Your results are remarkable."	"Excellent! Getting stronger means being there for your dearest ones."	"Excellent! Being more fit means being there for your dearest ones."
FA	"It is awesome that you decided to exercise today!"	"Fantastic! You are doing a great job."	"Brilliant! I know that you can do it."	"Excellent! I believe in you."

After the design, the implementation of the motivational assistant for the ActiveUP project was started.

The first step to implement the motivational assistant logic was the creation of new tables in the database to store additional information about the user and to manage the new alert and evaluation system. These tables were added directly on the database in phpMyAdmin.

Changes introduced to the original server were saved in the branch 'qa' of the repository 'servidorpositive'.

The changes introduced to the user application were saved in the repository 'POSITIVE_Pacientes_Android' in the branch '1-recoger-mas-datos-de-vivifrail' (https://gitlab.com/AgeingLab/positive_pacientes_android). The modifications concerned the communication with the database as well as the visual and logical aspects of the user interface. The most significant changes were introduced to classes:

- RestAPI.java, in the package es.upm.ctb.positive.rest
- MenuPrincipal_nuevo.java, in the package es.upm.ctb.positive.activities
- MeasurementUploaderService.java, in the package es.upm.ctb.positive.rest
- EjercicioFisico.java, in the package es.upm.ctb.positive.activities.ejercicio_fisico

One important function developed and kept in the new version is the evaluation of the patient's performance. The table "eval" was added to store information about daily and weekly evaluations of the exercise completeness level for each patient. The evaluation is run at midnight every day. The daily evaluation type considers the exercise completeness level obtained on the previous day and the weekly evaluation takes into account the data from the just-finished week.

4 Problem Statement

4.1. Criticisms

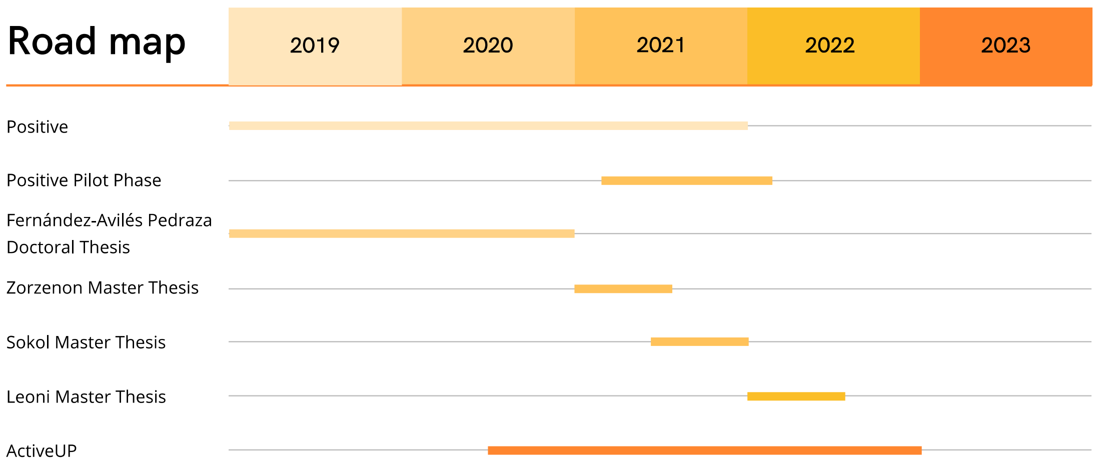


Figure 4.1: Evolution of the POSITIVE - ActiveUP projects

Figure 4.1 describes the evolution of the POSITIVE – ActiveUP projects. The POSITIVE project began in 2019. During it, Fernandez-Aviles Pedraza and Zorzenon pursued their projects related to motivation. In the last semester of the POSITIVE project, while the pilot phase was taking place, Sokól began designing the prototype of the motivational assistant to be integrated into POSITIVE. When this master's thesis began, the POSITIVE pilot phase was in its final stage. The ActiveUP project, on the other hand, was already underway.

The changes proposed by ActiveUP, combined with POSITIVE's problems identified during the pilot, made renovation necessary. In addition, this renovation made it necessary to adapt Sokól's work to the new system.

In addition, in March 2022, the Ageing Lab performed a heuristic review of the ActiveUP application. All the 27 heuristics proposed in (Gomez-Hernandez et al. 2022) were considered. The results highlight the following drawbacks:

- Some icons are not concrete enough and do not depict a real meaning.
- Some texts are not clear enough. Older adults could face problems understanding them.
- The feedback after control tapping is not properly highlighted. Older users need stronger feedback to their interaction.

To address these issues and also to cope with a new architecture and evolved technologies, the team decided to create a brand-new android application for patients, implemented in Java and Kotlin. However, the new application still did not integrate Vivifrail and the motivational assistant.

The POSITIVE server also had proved to have several issues. In particular, it was not reliable enough, and the concurrency did not work as expected. The main problems were raised when complex queries were executed. In addition, the generation of exercise plans often failed. For these reasons, the team opted for a new server, written in Python3, using the FastAPI framework and a new database.

Moving on to the motivational assistant, Sokól in her manuscript proposes Key Resources, i.e., resources that can potentially be used to influence the user's motivation state. However, these are only mentioned. The design of such resources was not addressed in the master thesis.

For what concerns the initial implementation of the motivational assistant, we identified the following drawbacks:

- The class MenuPrincipal_nuevo was used to handle the alerts from the Motivational Assistant. This mainly led to having a class of over a thousand lines of code with multiple disconnected functionalities. In addition, this class was modified so that only notifications related to "Two days of inactivity" could be shown. That is, it means that the code is neither scalable nor reusable.
- While handling notifications, multiple PUT requests were being sent. This meant that the client kept changing values in the database with each user interaction. This move was considered unnecessary, as it would suffice to have a single update request once the dialogue was finished.
- The added classes were not incapsulated in a proper structure. This makes the possibility of component expansion complex.

4.2. Objectives

Given the problems highlighted in the previous section, the following four goals were set for this thesis:

1. **Migration to the new server** - overhaul of the legacy server and integration of only the functions currently in use,
2. **Migration to the new Android application** - review of the legacy patient application and integration of features that will be preserved in the new version,
3. **Refactoring of the motivational assistant component done so far** - resolution of the critical issues highlighted above and reorganization of the code,
4. **Design of new features** – design and low-fidelity prototypes of the new features to integrate into the motivational assistant,

5. **Implementing the motivational assistant component** - development of the new features as well as the ones only specified by Sokól in her manuscript but never implemented.

5 Migration of the server

5.1. New server in Python with FastAPI

As briefly explained in the previous chapter, several issues brought the team to the choice of moving the server from Node.js to Python with FastAPI. In addition, the new server features Swagger toolset, which simplifies the API development and allows developers to check APIs through its interface.

The new server (https://gitlab.com/ageing-lab/activeup-cloud-server) is organized as follows:

- config: in this folder, you will find the files related to the configuration of the different databases.
- models: here you can find files that define the types of the data model that you are going to use as if they were objects.
- routes: CRUD operations are programmed here.
- schemas: in this folder, you define the table schema that will create the object (columns, data types, etc.).
- utils: folder where we put code that can be used generically by the rest of the files.
- main.py: is the file to be executed by FastAPI. Here we simply import the CRUD operations created in the files in the routes folder.
- docker-compose.yml: Docker container where we define the services to use: MySQL, PHPMyAdmin, and MongoDB.
- launch.sh: bash script that runs the whole system. It is in charge of installing dependencies, activating the Python virtual environment, raising the Docker container, and, finally, launching the API.
- requirements.txt: file where the Python packages to be installed are defined as dependencies.

The initial server configuration was done by another team member. Despite this, the motivational assistant was not integrated yet. Figure 5.1 shows the description of the database schema.

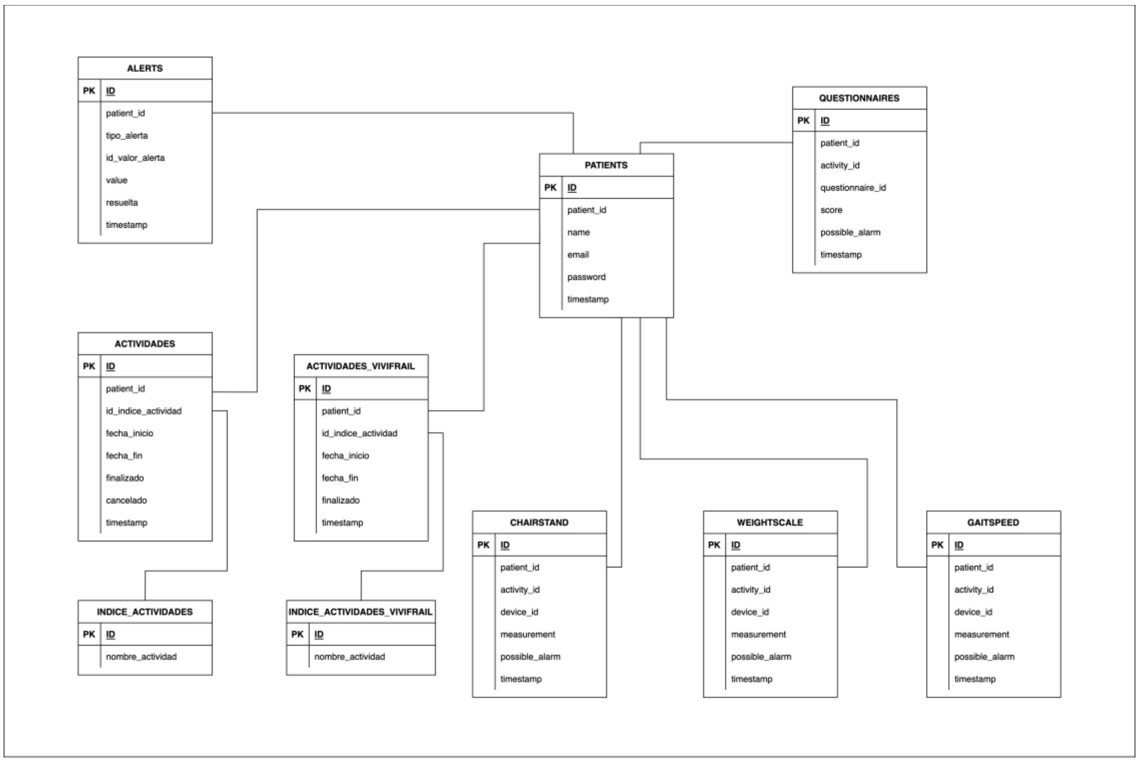


Figure 5.1: Description of the database schema (February 2022)

In order to integrate the motivational assistant component, the following files have been added:

In the **schemas** folder:

- alerts_assistant_schema.py

```
class AlertAssistant (BaseModel):
alert_id: int
content_id int
patient_id: int
notified: int
conclusion: int
```

- stage_of_change_schema.py

```
class StageOfChange (BaseModel):
patient_id: Optional[int]
stage_of_change_id: Optional[int]
motivational_state_id: Optional[int]
```

- eval_schema.py

```
class Eval(BaseModel):
patient_id: Optional[int]
eval: float
type: int
activity_id: int
status: int
```

```
timestamp: datetime
```

- motivational_profile_schema.py

```
class MotivationalProfile(BaseModel):
patient_id: Optional[int]
profile_id: Optional[str]
value: int
```

In the models folder:

- alerts_assistant_model.py

```
Column ("id", Integer, primary_key=True),
Column ("alert_id", Integer),
Column ("patient_id", Integer),
Column ("notified", Integer),
Column ("conclusion", Integer),
Column ("timestamp", TIMESTAMP)
```

- eval_model.py

```
Column ("id", Integer, primary_key=True),
Column ("patient_id", Integer),
Column ("eval", Float (6)),
Column ("type", Integer),
Column ("id_index_activity", Integer),
Column ("status", Integer),
Column ("timestamp", TIMESTAMP)
```

- index_alerts_assistant_model.py

```
Column ("id", Integer, primary_key=True),
Column ("name_alert", VARCHAR (50))
```

- index_alertas_conclusion_model.py

```
Column ("id", Integer, primary_key=True),
Column ("name_conclusion", VARCHAR (50))
```

- index_stage_of_change_model.py

```
Column ("id", Integer, primary_key=True),
Column ("name_stage_of_change", VARCHAR (50))
```

- index_motivational_state_model.py

```
Column ("id", Integer, primary_key=True),
Column ("name_state", VARCHAR (50))
```

- index_motivational_profile_model.py

```
Column ("id", Integer, primary_key=True),
Column ("name_profile", VARCHAR (50))
```

- motivational_profile_model.py

```
Column ("id", Integer, primary_key=True),
        Column ("patient_id", Integer),
        Column ("profile_id", Integer),
        Column ("value", Integer)
```

In the **routes** folder:

- alerts_assistant.py

  This file contains the GET requests to retrieve the alerts.

  During the migration, the code has been changed from MySQL query to Python. An example is provided in Table 3:

  GET Request: getAlertsAssistant

  Given the patient_id as a parameter, it returns a list of AlertAssistant. This function is called when the application is launched, in order to retrieve all the alerts to display.

  Table 5.1: Example of query in MySQL and Python

MySQL	`"SELECT id, id_alerta, id_contenido FROM alertas_asistente WHERE id_paciente = " + id_paciente + " AND  = 0 "`
Python	`alertas_asistente_data_model.select().where( alertas_asistente_data_model.c.patient_id == patient["patient_id"], alertas_asistente_data_model.c. == 0)`

  GET Request: getAlertAssistant

  Given the alert_id, it returns the object AlertAssistant.

  In addition, it has the PUT and POST requests:

  POST Request: postAlertAssistant

  This function adds a new alert in the database.

  PUT Request: updateAlertAssistant

  Given the alert_id and conclusion_id as parameters, the function updates the AlertAssistant by changing the values of notified and conclusion.

- stage_of_change.py

  This file contains the GET request to retrieve the state of change and the PUT requests to update it:

  GET Request: getStageOfChange

  Given the patient_id, this function returns an object StageOfChange.

PUT Request: updateStageOfChange

Given the patient_id and stage_of_change_id as parameters, the function updates the StageOfChange of the specified patient, by changing the stage_of_change_id.

PUT Request: updateMotivationalState

Given the patient_id and motivational_state_id as parameters, the function updates the StageOfChange of the specified patient, by changing the motivational_state_id.

- eval.py

  This file contains the GET request to retrieve the evaluations of a patient:

  GET Request: getEval

  Given the patient_id, this function returns an object Eval.

- motivational_profile.py

  This file contains the GET request to retrieve motivational traits per each user:

  GET Request: getMotivationalProfile

  Given the patient_id, this function returns an object MotivationalProfile.

5.2. Periodic Jobs

The following sections describe the changes introduced to the crontab file and explain the scripts (modified and created) to perform some required actions.

5.2.1. Cron

Cron is a tool that allows you to schedule the execution of preset tasks. It can be used in Unix-like systems to schedule jobs that must be completed over a set period or at a specific moment. Cron settings for the entire system are stored in the file etc/crontab.

The structure of the cron files has changed during the migration in order to properly accomplish the required tasks.

The already existing eval_asistente.js file has been split into eval_asistente_daily.js and eval_asistente_weekly.js.

5.2.1.1. eval_asistente_daily.js

This file is executed every day (from Tuesday to Saturday) at midnight. The main function is evalPacienteDia.

evalPacienteDia evaluates the daily performance of the user and inserts a row in the “eval” table. Since this is the per diem assessment, the ‘type’ field will contain the value 1.

5.2.1.2. eval_asistente_weekly.js

This file is executed every Saturday at midnight. The main function is evalPacienteSemana.

evalPacienteSemana evaluates the weekly performance of the user and inserts a row in the “eval” table. Since this is the per-week assessment, the ‘type’ field will contain the value 2.

6 Migration to the new Android application

As explained in Chapter 4, during the timeline of the POSITIVE project, it has been decided to make changes, including the mobile application for the patients.

The new version aims to be more efficient and in line with the results obtained during the pilot phase and the feedback received from the review.

The main difference from the predecessor version in terms of technology is the use of Kotlin, as well as Java. In addition, the new version does not work with Firebase Analytics, and it does not rely on SharedPreferences, which is what Android apps use to store simple data in an allocated space.

The new application features some optimizations that allow developers to speed up the implementation. The BaseActivity class extends the AppCompatActivity bult-in class provided by Android. As the name suggests, it is a basic activity with specific functions available to all the activities that extend it. An example of those is the function inicializarMenuInferior. By calling this method, the navigation bar at the bottom is automatically added to the user interface. In addition, the methods desactivarBoton, activarBoton, pulsarBotonAtras, pulsarBotonSalir, and pulsarBotonSiguiente are in charge of the behavior of each button.

In order to integrate the Vivifrail system, it was necessary to add to the new application the ability to perform the exercises in the treatment plan, see the nutrition advice, and consult the Vivifrail manual. The entire Vivifrail package can be found under My treatment in the main menu (see Figure 6.1). Upon accessing this section, the user will find the three subsections: My exercises, My nutrition, and My advice (see Figure 6.2).

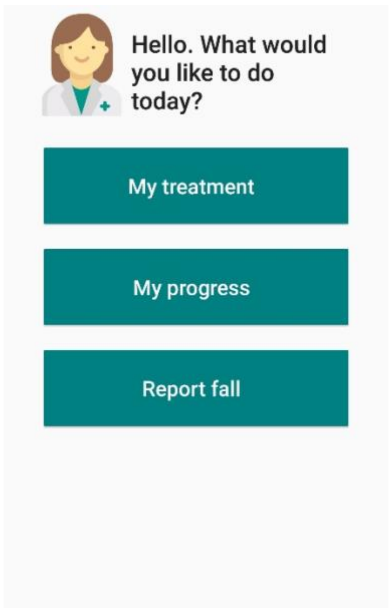


Figure 6.1: Home page of the new application

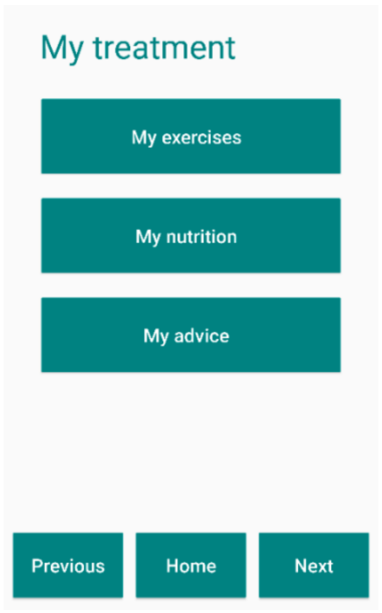


Figure 6.2: My treatment section

Upon access to the specific section, the system calls respectively the methods cargarEjercicios, cargarMiDieta, and cargarMiEducacion. In the following three subchapters, all the details of the above-mentioned sections are provided.

6.1. My Exercises section

My Exercises section is encoded in the Java class MenuVivifrail that extends BaseActivity. This is where the patient can see the exercises to be performed that day. Exercises are associated with the user through the Vivifrail plan previously set up. The exercises are retrieved from the database through the function ObtenerPlanActividadesVivifrail. If no exercises are scheduled, a notification message is shown (see Figure 6.5). The user interface presents a grid with a number of buttons equal to the number of exercises (see Figure 6.3). By pressing on them, the user interface changes and displays the specific exercise (see Figure 6.4).

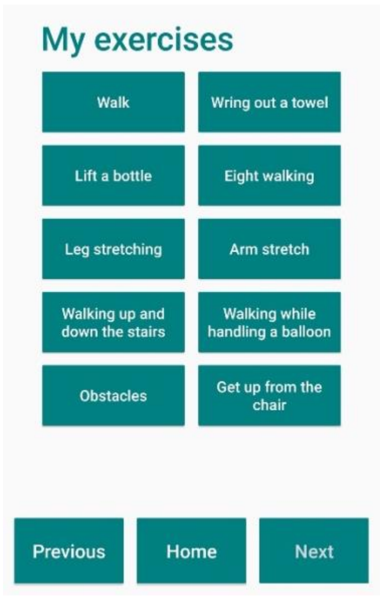


Figure 6.3: My exercises menu

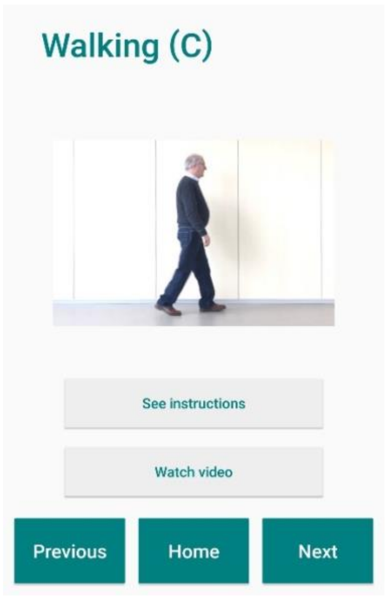


Figure 6.4: Exercise interface

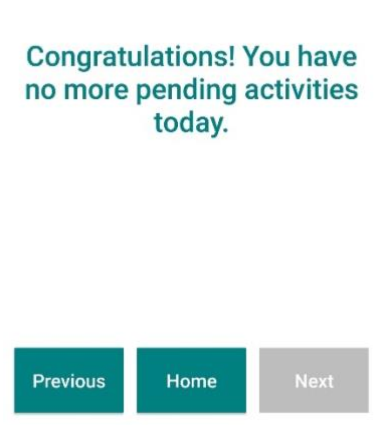

Congratulations! You have no more pending activities today.

Figure 6.5: Interface in case there are no pending exercises

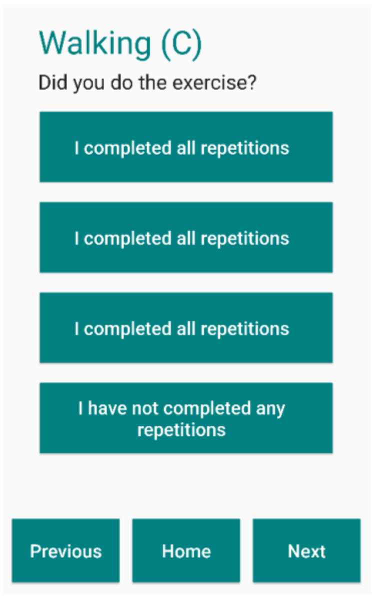

Walking (C)
Did you do the exercise?
I completed all repetitions
I completed all repetitions
I completed all repetitions
I have not completed any repetitions
Previous
Home
Next

Figure 6.6: Interface that asks the user how many repetitions they did

From this point, users can start with the execution of the activity. Furthermore, they can ask to read the instructions and see a video tutorial.

Once the user has finished the activity, by pressing the Next button, they are asked how well they performed the exercise. As Figure 6.6 shows, four options are available (I did all the repetitions, I did almost all the repetitions, I did a few repetitions, and I did not perform the exercise). The answer to this question is crucial in computing the patient’s daily evaluation.

6.2. My Nutrition section

My Nutrition section starts with the Java class MiDieta. The class extends BaseActivity and sets the content to the layout element activity_mi_dieta.xml. From here (see Figure 6.7), the user can go to the video section, where a collection of videos related to nutrition are displayed (see Figure 6.8). The video playback is handled by the class MiDietaVideoPlayer that opens a window with the corresponding YouTube video.

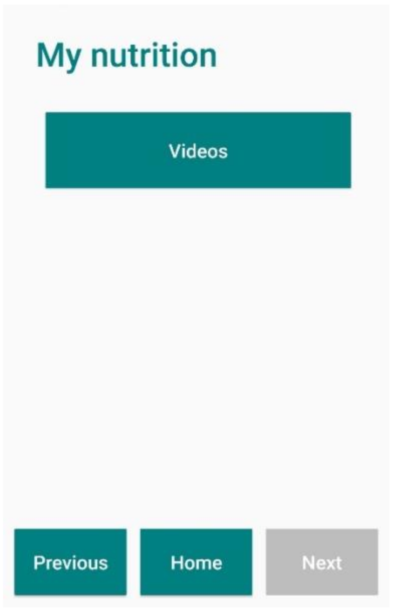


Figure 6.7: My nutrition menu

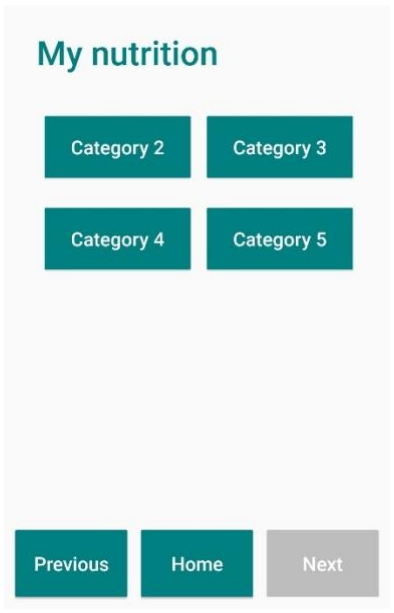


Figure 6.8: My nutrition recipes

6.3. My Advice section

My Advice section allows the user to consult the exercise manual and get general tips. This section starts with the Java class MiEducacionMenu. The menu (Figure 6.9) shows two buttons, the Exercise Manual one starts a new activity within the class ManualVivifrail. On the other hand, the General advice one starts a new activity within the class ConsejosGenerales.

The ManualVivifrail activity class shows the Vivifrail plan assigned to the user (Figure 6.10). First, through the GetVivifrailTask, the system retrieves the information from the database. Then, it computes which manual has to be displayed to the user. To display this information, the application leverages on the WebView component. WebView objects allow the developer to display web content as part of the activity layout.

The ConsejosGenerales class provides tips about healthy ageing and why it is important in order to improve the quality of life (Figure 6.11).

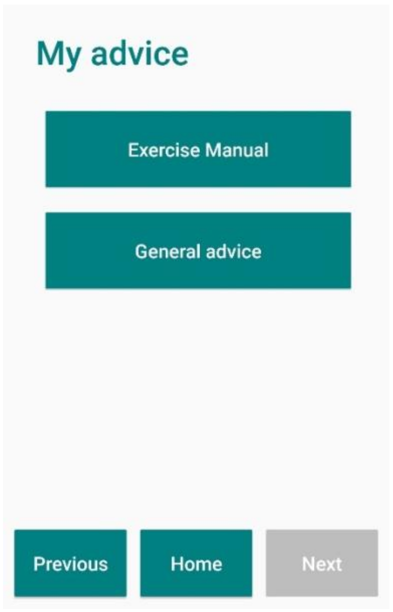


Figure 6.9: My advice menu

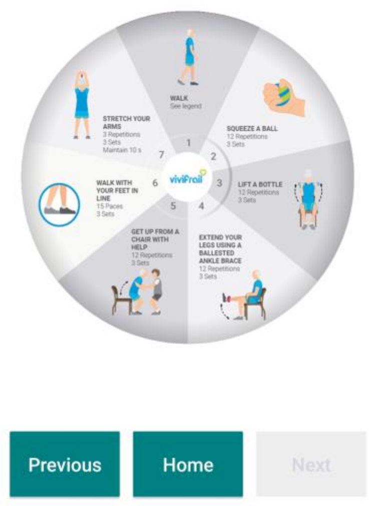


Figure 6.10: Exercise Manual

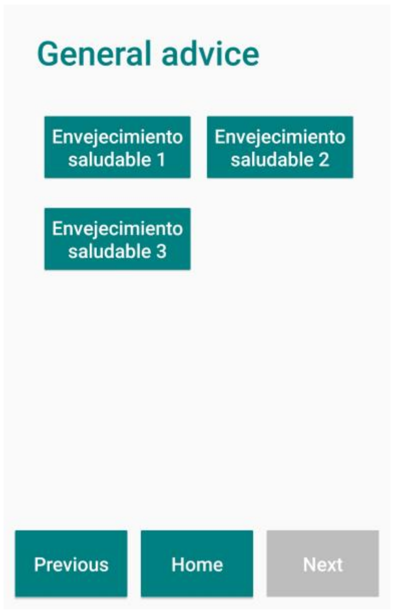

General advice
Envejecimiento saludable 1
Envejecimiento saludable 2
Envejecimiento saludable 3
Previous
Home
Next

Figure 6.11: General Advice

7 Development of the new features

As explained in Chapter 4, one of this Master Thesis objectives is the implementation of new features to be integrated into the ActiveUP Patient application. The set of these new features comprehends the leader board, challenges, users' statistics, and a collection of notifications to be sent to the user.

7.1. Leader board

7.1.1. Description

The leader board has been identified as a Key Resource by Sokól (2022), but its design was pending. It is a ranking showing the user's performance in comparison to the performance of other project participants in terms of exercise completeness level.

This resource is used to motivate patients with High or Very High levels of Competition Seeking. They are classified as striving for competition and being motivated by a possibility to outperform others. The user ranking will be based on adherence to the program, measured as the percentage of the performed exercises' repetitions and the total number of exercises performed. So, the patients with the highest percentage of performed exercises will appear at the top of the ranking.

The user classification will be shown to all users with Very High or High levels of Competition Seeking, unless they also have High or Very High levels of Failure Avoidance.

The leader board will include daily, weekly, and global classifications. Since the evaluation of the performance is computed at midnight, the daily leader board will rank users based on the performance of the day before. Equally, the weekly leader board will rank users based on the performance of the previous week. Finally, the global leader board will rank users starting from the first day of the program until the last one.

To meet some design requirements for older adults, having a leader board with a long list of participants was undesirable, so we decided to create groups and divide the participants. Each group should have a maximum of 6 members. The main reason for this choice is related to the difficulty of older adults to perform the scrolling gesture on digital devices. In fact, if we had considered a ranking based on all users in the system, we would have generated a very long list. And it would have meant forcing users to navigate through this list. This way, each patient will be included in a group of participants that have initiated their exercise program in the same day.

Also, since it could happen that less participants than 6 initiate their program in a given week, we decided to create fake users to complete groups with less than 6 real patients. For every real user, there will be a maximum of five fake users. This way, we will make sure that each user has a group of people to compete with.

7.1.2. Low fidelity design

To access the leader board two steps are required. First, the user will reach the Leader board menu (Figure 7.1) from *Homepage / My Progress / My Results*. In this menu, the user will find three buttons with an icon and a title explaining the section. Then, from there, by clicking on one of the three options, they will reach the specified leader board (daily, weekly, or global). Depending on the type of leader board, at the top of the interface, there will be the date of yesterday, the interval of the previous week, or the text 'Global'. Each row on the ranking will have the position, the username, and the evaluation (Figure 7.2).

To be consistent with the ActiveUP application UI, the navigation bar will be available on all the screens.

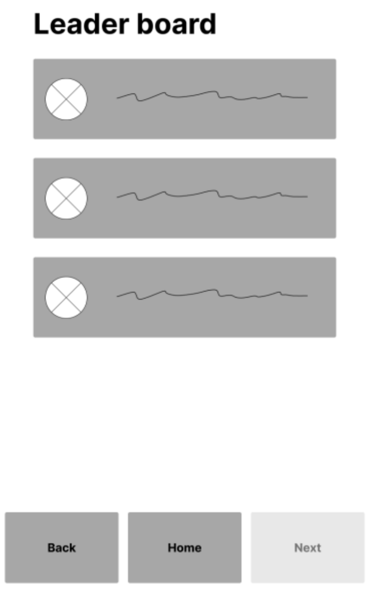


Figure 7.1: Wireframe of Leader board menu

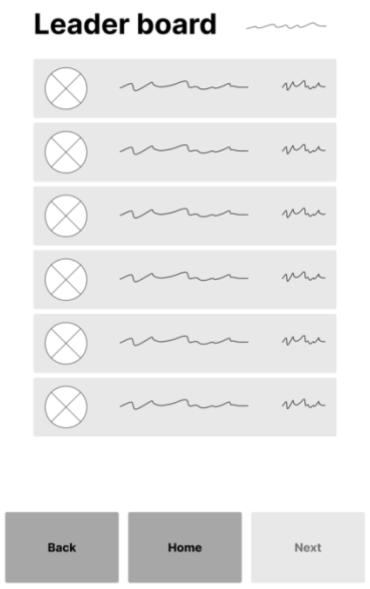


Figure 7.2: Wireframe of Leader board activity

7.1.3. Implementation

7.1.3.1. Server-side

Figure 7.3 represents the diagram of the tables added for the leader board feature into the database.

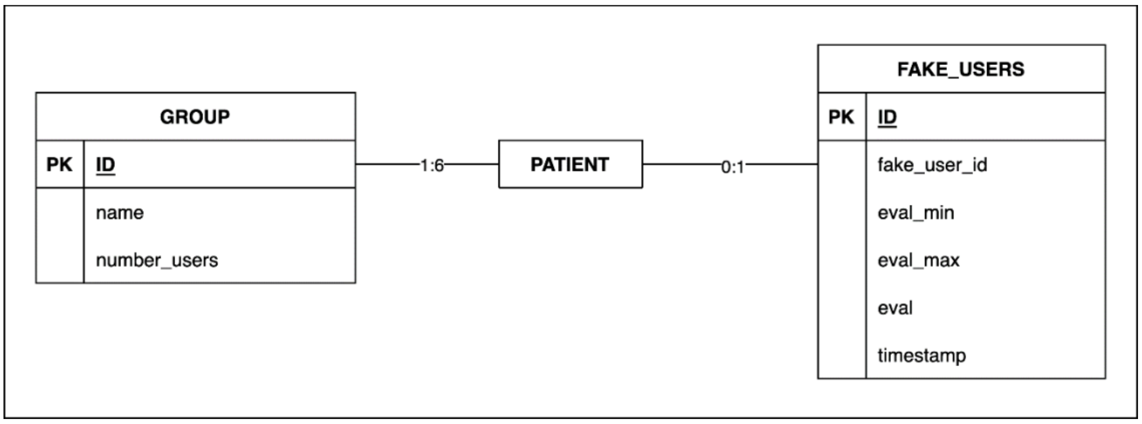


Figure 7.3: New tables added into the database for the Leader board feature.

In order to implement the leader board component, the following files have been added to the server:

In the **models** folder:

- fake_users_model.py

```
Column (“id”, Integer, primary_key=True),
Column (“fake_user_id”, Integer),
Column (“eval_min”, Float (6)),
Column (“eval_max”, Float (6)),
Column (“eval”, Float (6)),
Column (“timestamp”, TIMESTAMP)
```

- group_model.py

```
        Column (“id”, Integer, primary_key=True),
        Column (“name”, String (100)),
        Column (“number_users”, Integer)
```

In the **schemas** folder:

- fake_users_schema.py

```
        class FakeUser(BaseModel):
             fake_user_id: Optional[int]
             eval_min: float
             eval_max: float
             eval: float
             timestamp: datetime
```

- group_schema.py

```
        class Group(BaseModel):
             id: Optional[int]
             name: text
             number_users: int
```

In addition, the patient.py has been modified. A new column has been added to associate each user with a specific group. `Column (“group_id”, Integer)`.

In the crontab, the function `evalFakeUsersDia` has been added to eval_asistente_daily.js.

`evalFakeUsersDia` updates the evaluation in the *“fake_users”* table. The generated value is a random number between the values eval_min and eval_max defined for each fake user. As Table 7.1 shows, every user has a different value for eval_min and eval_max. These amounts allow us to have users along all the positions in the leader board. For instance, user 60 will be always at the bottom, while user 64 will be always in the top positions.

Table 7.1: Table "fake_users" to store daily evaluation of the fake users

id	id_fake_user	eval_min	eval_max	eval	timestamp
1	60	0.0010	0.29	0.2358	2022-06-22 00:00:01
2	61	0.1	0.49	0.4384	2022-06-22 00:00:01
3	62	0.2	0.59	0.5612	2022-06-22 00:00:01
4	63	0.4	0.79	0.7233	2022-06-22 00:00:01

5	64	0.6	1	0.8943	2022-06-22 00:00:01

After the random generation, the UPDATE the '`insert_fake_user_eval`' trigger is launched. It simply adds into the *"eval"* table, the eval for each fake user.

To generate the ranking, the system implements three different GET Requests, respectively `getClasificacionDiaria`, `getClasificacionSemana`, and `getClasificacionGlobal`, passing as parameter the `patient_id`. The response is a list of users from the same group with the evaluation.

`getClasificacionDiaria` performs a query to obtain the daily ranking (To see the code Appendix 10.1). It returns an object with patient_id, patient_name, and eval.

`getClasificacionSemana` performs a query to obtain the weekly ranking (To see the code Appendix 10.1). It returns an object with patient_id, patient_name, and eval.

`getClasificacionGlobal` performs a query to obtain the global ranking (To see the code Appendix 10.1). It returns an object with patient_id, patient_name, and eval.

The object returned is then managed in the Android app, as explained below.

7.1.3.2. Client-side

On the ACTIVE-UP application, the files related to the leader board are in the packages (as reported in Figure 7.4):

- **com.activeup.activities.leaderboard**
- **com.activeup.objects.asistente_motivacional**.

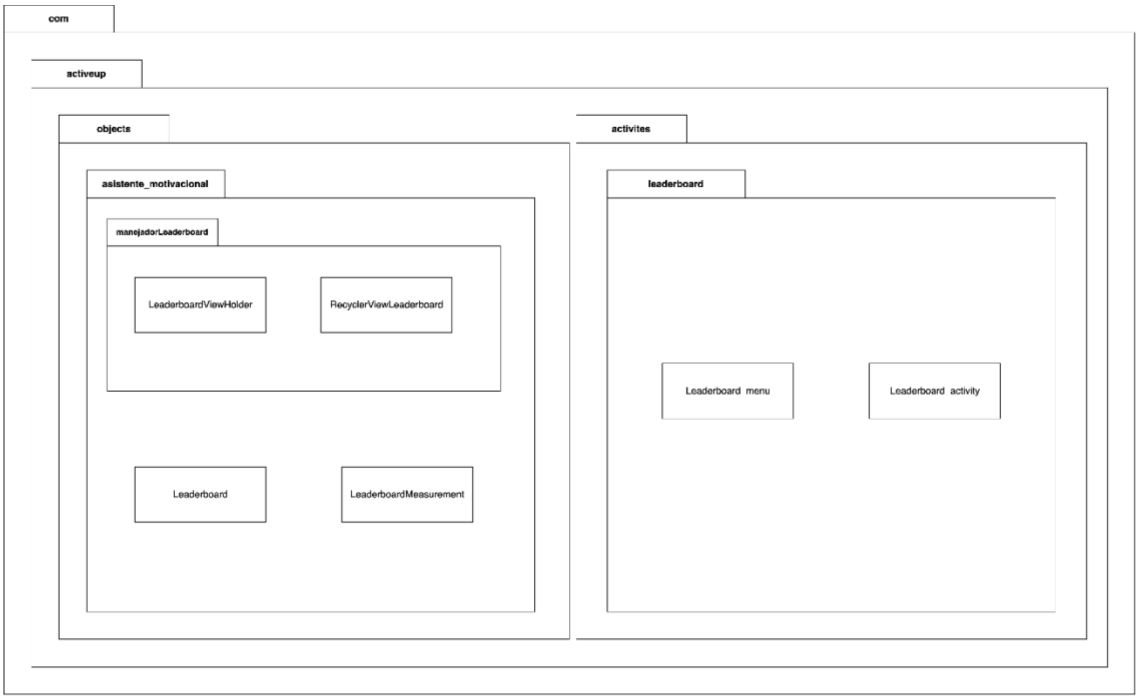


Figure 7.4: UML for the Leader board feature

Leaderboard is the main object. To create an instance of the object, the system does a `new Leaderboard(Context context, RecyclerView recyclerView, String token, String user_id, int activity_id)`. The class has the constructor and three private classes `ObtenerClasificacionDiaria`, `ObtenerClasificacionSemana`, and `ObtenerClasificacionGlobal`. The first performs the GET Request to the server to retrieve the daily leader board. The second performs the GET Request to the server to retrieve the weekly leader board. The third one performs the GET Request to the server to retrieve the global leader board. Once the data are downloaded from the database, they are stored in a list and passed to the *RecyclerViewLeaderboard*.

RecyclerViewLeaderboard is a RecyclerView. RecyclerView makes it easy to efficiently display large sets of data. You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed. Each element in the list is defined by a view holder object (in our case *LeaderboardViewHolder*). It is responsible for how the rows in the ranking are displayed. In addition, it is in charge of replacing the first three icons on the list, putting an icon representing a trophy. The RecyclerView requests those views, and binds the views to their data, by calling methods in the adapter.

LeaderboardViewHolder is a RecyclerView.ViewHolder, which describes an item view. In particular, *LeaderboardViewHolder* has three TextView (position, username, and evaluation) and one ImageView (the trophy icon).

LeaderboardMeasurement is the return type of the GET requests.

Leaderboard_menu extends BaseActivity. The Activity class takes care of creating a window for you in which you can place your UI with #setContentView. This class takes care of creating the leader board menu, where the three buttons (Daily, Weekly, and Global) are shown. In the following section, the UI of this activity will be presented.

Leaderboard_activity also extends BaseActivity. It is responsible for each specific leader board. In the following section, the UI of this activity will be presented.

7.1.4. Final User Interfaces

Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow you to build the graphical user interface for your app. Each developed layout contains Android components, such as LinearLayout, RelativeLayout, ImageView, TextView, and Button. LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. RelativeLayout is a view group that displays child views in relative positions. ImageView displays image resources. TextView contains the text to be displayed. A Button is an element the user can tap or click to perform an action (specified in the onClick attribute).

The user interfaces created for this feature are XML files stored in the layout folder. In particular, the files are `leaderboard_menu.xml, leaderboard_activity.xml, and row_leaderboard.xml`.

Concerning the low-fidelity design, some changes have been made. In particular, in the Leader board menu (Figure 7.5), all the icons have been removed.

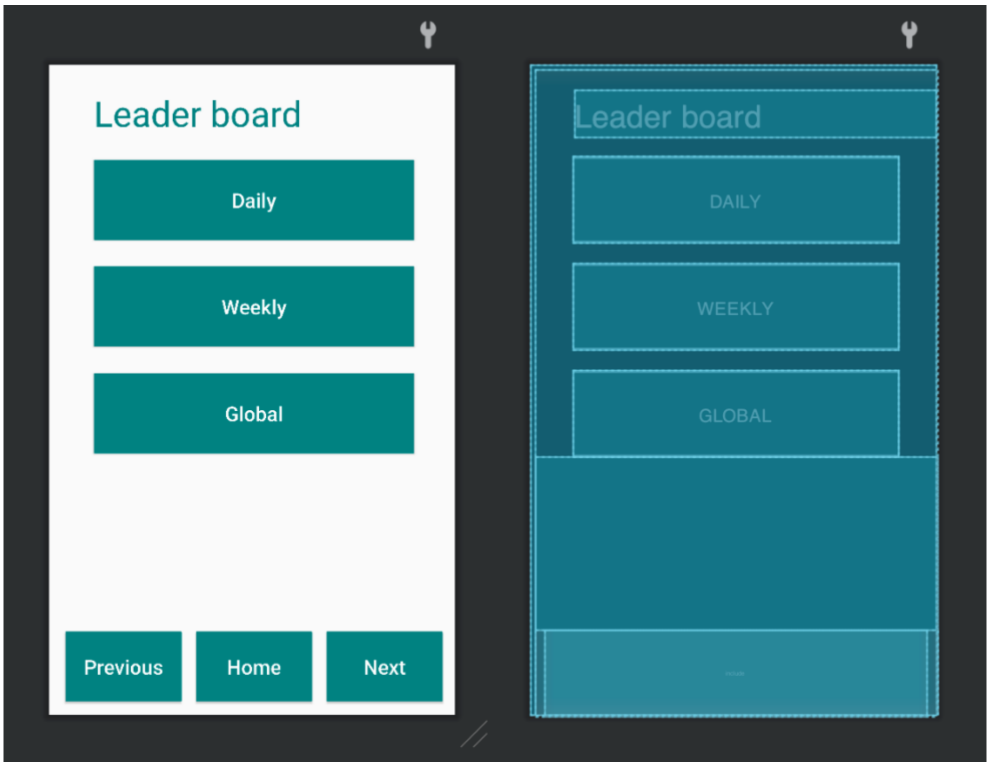


Figure 7.5: UI for Leader board menu

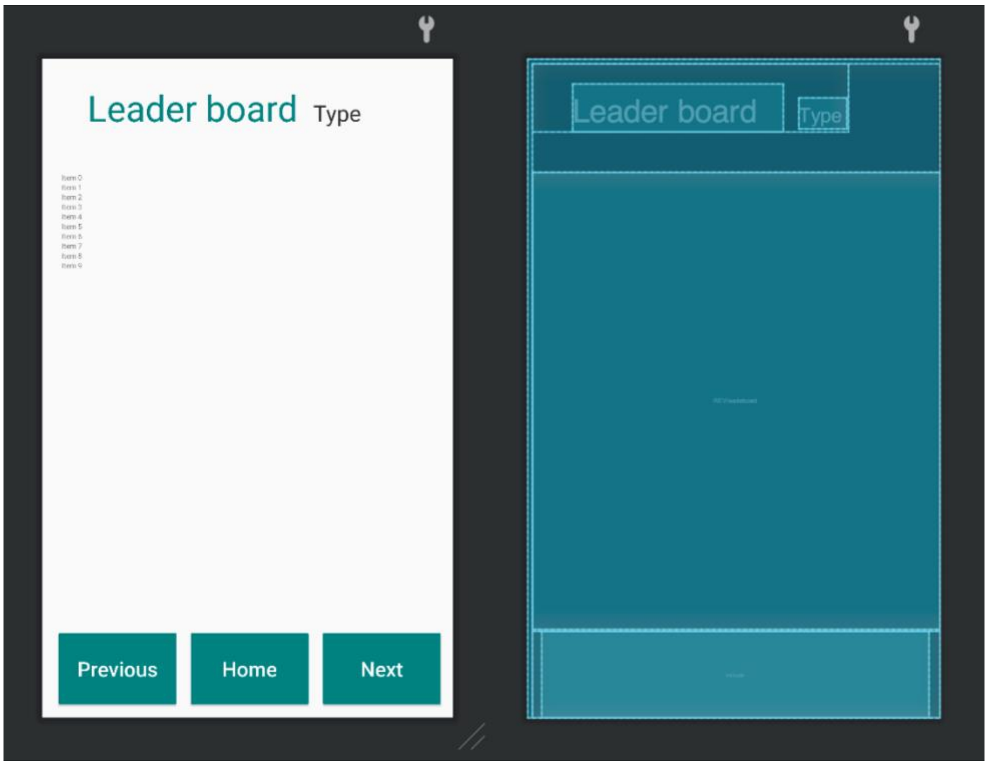
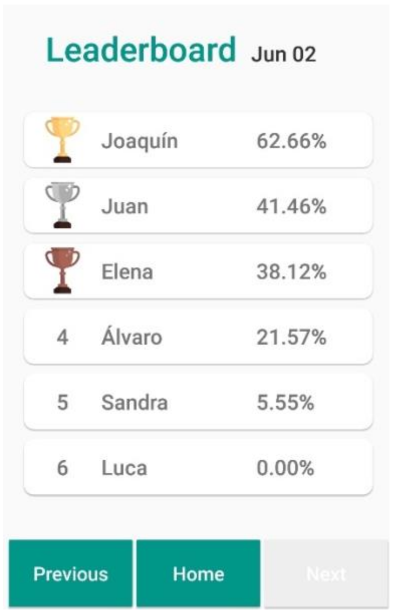


Figure 7.6: UI for Leader board Activity

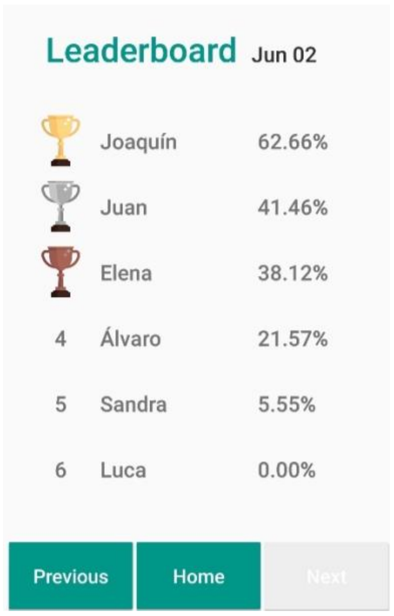
For what concerns the Leader board activity, instead of using the CardView, apparently the most straightforward option, to contain each row, a simple LinearLayout has been used (Figure 7.6). The reason behind this choice lies on the possibility of confusion among the elderly. Actually, with CardView, the interface is more aesthetically pleasant (Figure 7.7). However, the rows appear as if they were buttons and therefore might create the illusion of being clickable. Thus, the final choice was to provide a ranking closer to a simple list (Figure 7.8). In this way, the system matches the UHOA 14 (Section 2.2), which recommends increasing the difference between touchable and non-touchable elements [10].



Leaderboard Jun 02

	Joaquín	62.66%
	Juan	41.46%
	Elena	38.12%
4	Álvaro	21.57%
5	Sandra	5.55%
6	Luca	0.00%

Figure 7.7: Leader board activity with CardView



Leaderboard Jun 02

	Joaquín	62.66%
	Juan	41.46%
	Elena	38.12%
4	Álvaro	21.57%
5	Sandra	5.55%
6	Luca	0.00%

Figure 7.8: Leader board activity without CardView

7.2. Challenges

7.2.1. Description

Challenges have been identified by [3] as a *Key Resource* for patients with *High* or *Very High* levels of the Personal Mastery trait. Personal Mastery includes both Desire to Learn and Mastery Goals, the motivational techniques referring to this trait frequently mention user goals and model the exercise sessions as an opportunity to gain new experiences, skills, and knowledge.

So far, seven challenges have been implemented. They are explained in Table 7.2.

Table 7.2: Name and description of the challenges

Challenge	Name	Description
1	Regular day	The user is asked to do all the exercises for one day, regardless of the repetitions.
2	Regular week	The user is asked to do at least one exercise every day for a week, regardless of the repetitions.
3	Perfect week	The user is asked to do all the exercises with all repetitions every day for a week.

4	Two regular weeks	The user is asked to do at least one exercise every day for two weeks, regardless of the repetitions
5	Two perfect weeks	The user is asked to do all the exercises with all repetitions every day for two weeks.
6	Perfect month	The user is asked to do all the exercises with all repetitions every day for an entire month.
7	Perfect final week	The user is asked to do all the exercises with all repetitions every day for the last week of the program.

The user is notified of a new challenge via alerts. In addition, they can always check their collection of awards in the specific section. Furthermore, if they want to know what the challenge is about, they can check it in the Next Challenge section, where the description is provided.

Ideally, all the challenges should be faced during the 12 weeks of the program according to Figure 7.9. During the first week, the user receives an alert from the motivational assistant explaining the Challenge n°1 (Regular Day). If they overcome the challenge, the next Monday, the motivational assistant will launch the Challenge n°2 (Regular week). The same pattern is followed for the remaining challenges. Notice that Challenge n°7 (Perfect final week) will be offered only to users who accomplished all the previous challenges in their due time.

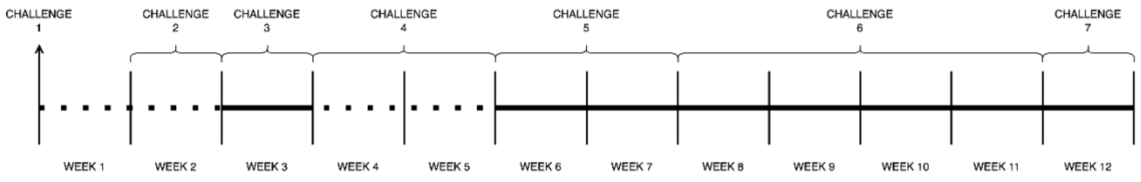


Figure 7.9: Schedule of the challenges during the 12-weeks programme

If the user always wins each challenge in time, at the end of the program they should have gained all the awards. It is crucial to take into account the possibility of losing a challenge. In this case, the idea is to re-launch the same challenge in the following week. For instance, if the user does not complete the Challenge 2 (Regular Week) during the second week, the next Monday they will receive again the alert to fight for this challenge.

7.2.2. Low fidelity design

To access the challenge section at least two steps are required. First, the user will reach the Challenges menu (Figure 7.10) from *Homepage / My Progress / My Results*. In this menu, the user will find two buttons with an icon, and a title explaining the section.

Then, from there, by clicking on one of the two options (Won challenges, or Next challenge), they will reach the specified challenge activity (Won or Next), as shown in Figure 7.11, and Figure 7.12. From these interfaces, by clicking on a specific challenge, the Challenge description activity appears (Figure 7.13). This screen provides all the information related to the challenge, such as name, description, icon, and date of achievement (if achieved).

To be consistent with the ActiveUP application UI, the navigation bar will be available on all the screens.

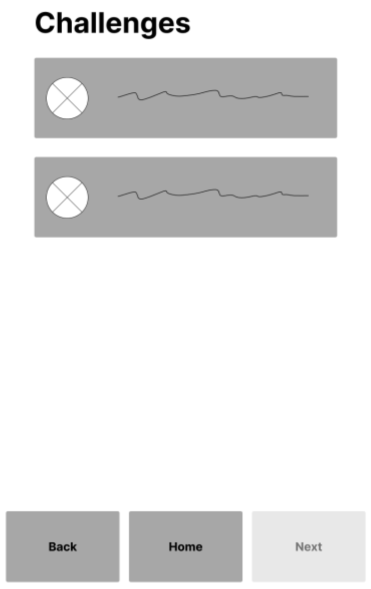


Figure 7.10: Wireframe of Challenges menu

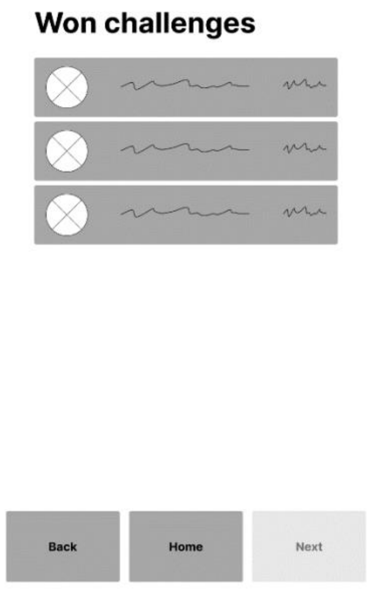


Figure 7.11: Wireframe of Challenges activity (Won challenges)

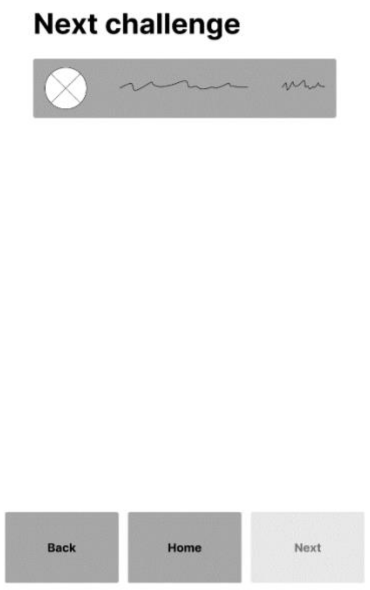


Figure 7.12: Wireframe of Challenges activity (Next challenge)

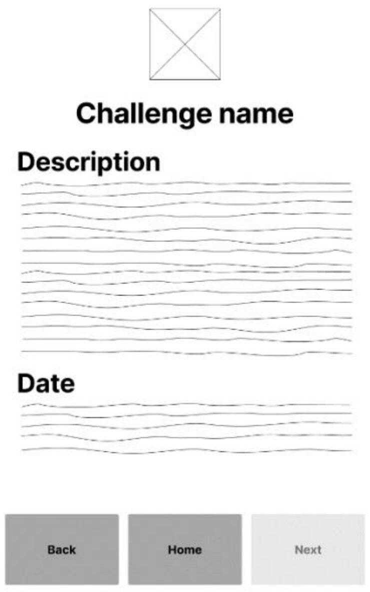


Figure 7.13: Wireframe of Challenges description activity

7.2.3. Implementation

7.2.3.1. Server-side

To implement this feature, two new tables have been added to the database (Figure 7.14).

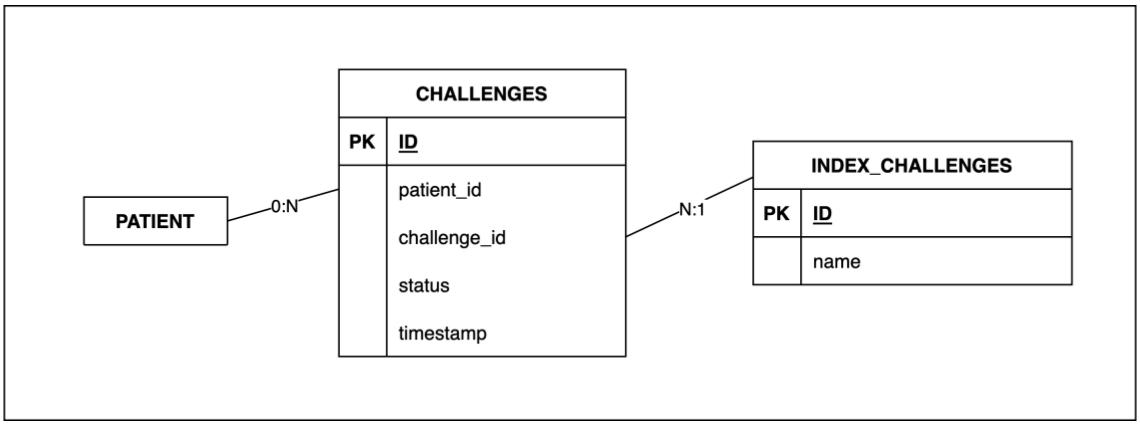


Figure 7.14: New tables added into the database for the Challenges feature

The following files have been added to the server:

In the **models** folder:

- index_challenges_model.py

```
Column (“id”, Integer, primary_key=True),
Column (“name”, VARCHAR (50)),
```

- challenges_model.py

```
            Column (“id”, Integer, primary_key=True),
            Column (“patient_id”, Integer),
            Column (“challenge_id”, Integer),
            Column (“status”, Integer),
            Column (“timestamp”, TIMESTAMP)
```

In the **schemas** folder:

- challenges_schema.py

```
            class Challenge(BaseModel):
                patient_id: Optional[int]
                challenge_id: Optional[str]
                status: int
```

CHALLENGES (id, patient_id, challenge_id, status, timestamp)

The field `‘id’` refers to a unique id of each row. The id of the patient is stored in the field `‘patient_id’` and it is taken from the already existing table *“patient”*. The id of the challenge is stored in the column `‘challenge_id’`. Every value has a description in the table *“index_challenges”*. The column `‘status’` contains a value that could be -2, -1, 0, or 1, depending on the status of the challenge. -2 means that the challenge is not available yet. -1 means that the challenge has to be notified to the user by the motivational assistant. 0 means that the user is currently facing that challenge. Finally, if the challenge has been achieved, the value is 1. This column is crucial in order to display the correct content in the section Won Challenges and Next Challenge. The field `‘timestamp’` contains the timestamp of the insertion of the challenge. This timestamp is updated when the field `‘status` changes. Once the patient is added to the database and the motivational profile has been defined, this table is filled with all the 7 challenges (`‘status’` = -2). The Challenge 1 is the only one with `‘status’` = -1.

INDEX_CHALLENGES (id, name)

As stated above, so far, it has been implemented 7 types of challenges. The following table (Table 7.3) shows the contents of the table in the database defining the challenges.

Table 7.3: Types of Challenges

Id	Name
1	Regular day
2	Regular week
3	Perfect week
4	Two regular weeks
5	Two perfect weeks

6	Regular month
7	Perfect final week

The field `'id'` identifies the challenge, it is referenced in the table *"Challenges"*. The column `'name'` explains the specific challenge type.

To download the challenges, the system implements two different GET Requests, respectively `getLogros`, and `getRetos`, passing as parameter the `patient_id`.

`getLogros` performs a query to obtain the already won challenges. It accesses the table *"Challenges"* and selects all the rows with `'patient_id'` = `patient_id` and `'status'` = 1.

`getRetos` performs a query to obtain the challenge the user is facing at the moment. It accesses the table *"Challenges"* and selects all the rows with `'patient_id'` = `patient_id` and `'status'` = 0.

The management of the challenges is implemented using Cron Jobs. Two files were added to the server: `check_status_challenges.js` (Appendix B.1.1) and `manejador_challenges_to_be_notified.js` (Appendix B.1.2).

`check_status_challenges.js` is executed every Monday at midnight. It consists of the following twelve functions:

- `computePatients()` – retrieves the patients with Personal Mastery trait.
- `checkChallengeInProgress(patient_id)` – checks whether the patient associated with the `patient_id` has a challenge in progress. According to the result, one of the following `checkChallenge()` is called:
    - `checkChallenge1(patient_id, challenge_id)` – checks if the requirements of the Challenge 1 are satisfied. If so, `notifyWonChallenge()` is called.
    - `checkChallenge2(patient_id, challenge_id)` - checks if the requirements of the Challenge 2 are satisfied. If so, `notifyWonChallenge()` is called.
    - `checkChallenge3(patient_id, challenge_id)` - checks if the requirements of the Challenge 3 are satisfied. If so, `notifyWonChallenge()` is called.
    - `checkChallenge4(patient_id, challenge_id)` - checks if the requirements of the Challenge 4 are satisfied. If so, `notifyWonChallenge()` is called.
    - `checkChallenge5(patient_id, challenge_id)` - checks if the requirements of the Challenge 5 are satisfied. If so, `notifyWonChallenge()` is called.

        - `checkChallenge6(patient_id, challenge_id)` - checks if the requirements of the Challenge 6 are satisfied. If so, `notifyWonChallenge()` is called.
        - `checkChallenge7(patient_id, challenge_id)` - checks if the requirements of the Challenge 7 are satisfied. If so, `notifyWonChallenge()` is called.
- `updateWonChallenge(patient_id, challenge_id)` – this function is executed if the `checkChallenge()` retrieves a Won Challenge. It updates the field "`status`" on table *"challenges"*, by setting it equal to 1 (won).
- `notifyWonChallenge(patient_id, challenge_id)` – inserts a new alert of type 5 into *"alerts_assistant"* with "`id_content`" set to the value of the challenge. In addition, if the `challenge_id` is different from 7 (which is the last available challenge) the `updateNewChallenge()` is executed.
- `updateNewChallenge(patient_id, challenge_id)` – it updates the field "`status`" on table *"challenges"*, by setting it equal to -1 (to be notified).

`manejador_challenges_to_be_notified.js` is executed every Monday at 00:05. It is structured as follows:

- `computePatients()` – retrieves the patients with Personal Mastery trait.
- `checkNewChallenge(patient_id)` – checks whether the patient associated with the `patient_id` has a challenge to be notified. According to the result, `notifyNewChallenge()` is called.
- `notifyNewChallenge(patient_id, challenge_id)` – inserts a new alert of type 4 into *"alerts_assistant"* with "`id_content`" set to the value of the challenge.
- `updateChallenge(patient_id, challenge_id)` - it updates the field "`status`" on table *"challenges"*, by setting it equal to 0 (to be notified).

The entire code of these files is attached in sections B.1.1 and B.1.2.

7.2.3.2. Client-side

On the ACTIVE-UP application, as shown in Figure 7.15, the files related to the challenge are in the packages **com.activeup.activities.challenges** and **com.activeup.objects.asistente_motivacional.**

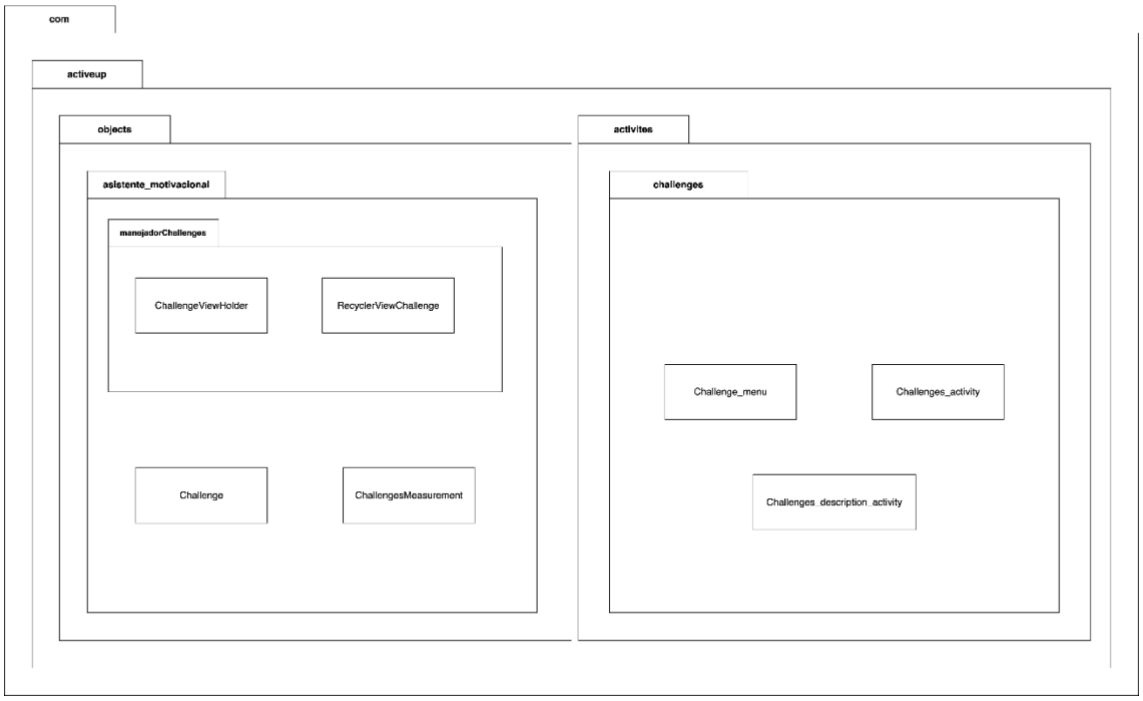


Figure 7.15: UML for Challenges

Challenge is the main object. To create an instance of the object, the system executes a `new Challenge(Context context, RecyclerView recyclerView, String token, String user_id, int activity_id)`. The class has the constructor and two private classes `ObtenerReto` and `ObtenerCollecionLogros`. `ObtenerReto` is in charge of performing the GET request to the server to obtain the current challenge the user is facing. On the other hand, `ObtenerCollecionLogros` performs the GET request to the server to retrieve the already won challenges. Once the data are retrieved from the database, they are passed to the *RecyclerViewChallenge*.

RecyclerViewChallenge is a RecyclerView. It dynamically creates the element defined by the *ChallengeViewHolder*.

ChallengeViewHolder is a RecyclerView.ViewHolder, which describes an item view. It has two TextView (name and date), one ImageView (challenge icon), and one CardView (to simulate the appearance of a button).

ChallengeMeasurement is the return type of the GET requests.

Challenge_menu extends BaseActivity. It creates the window responsible for the menu. In this interface the two buttons (Next and Won) are displayed.

Challenges_activity extends BaseActivity. In this activity the view set with #setContentView represents the list of the won challenges or the next challenge.

Challenges_description_activity extends BaseActivity. It shows the details of each challenge, providing name, icon, description, and date of achievement.

7.2.4. Final User Interfaces

The user interfaces created for the challenges are XML files stored in the layout folder. Specifically, `row_challenges.xml` `challenges_menu.xml`, `challenges_activity.xml`, and `challenges_description_activity.xml`. With respect to the low-fidelity prototypes, the left icons on the buttons have been deleted (Figure 7.16).

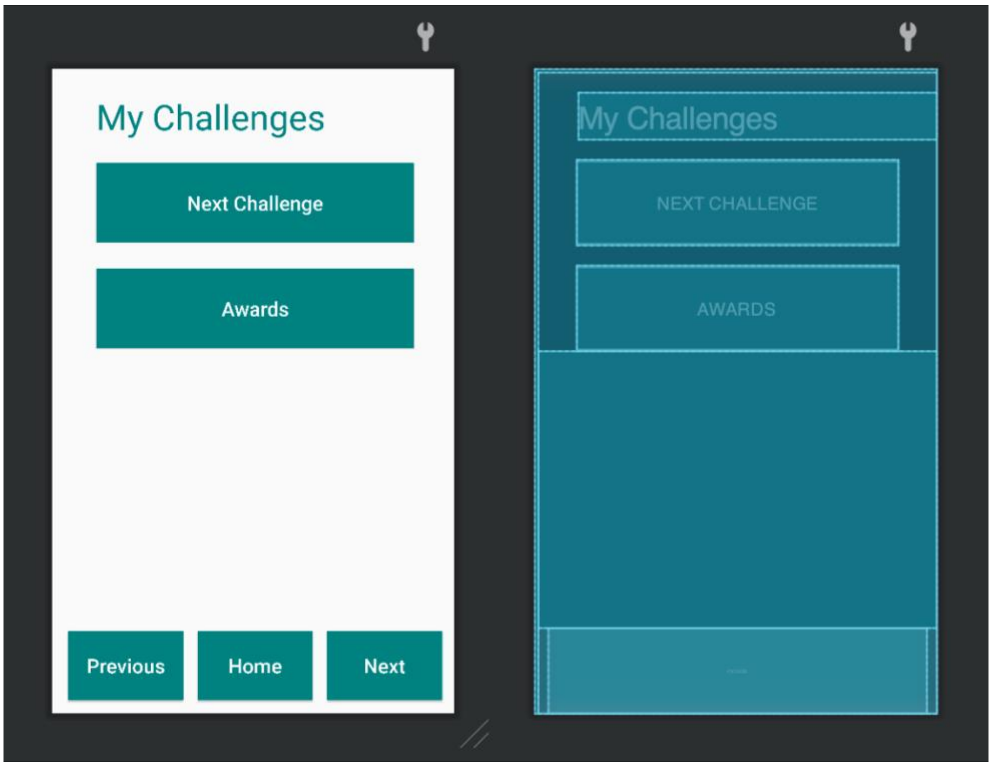


Figure 7.16: UI of Challenges Menu

The Challenges activity has been implemented as expected (Figure 7.17). By using the CardView component, the rows simulate buttons and behave as such (Figure 7.18).

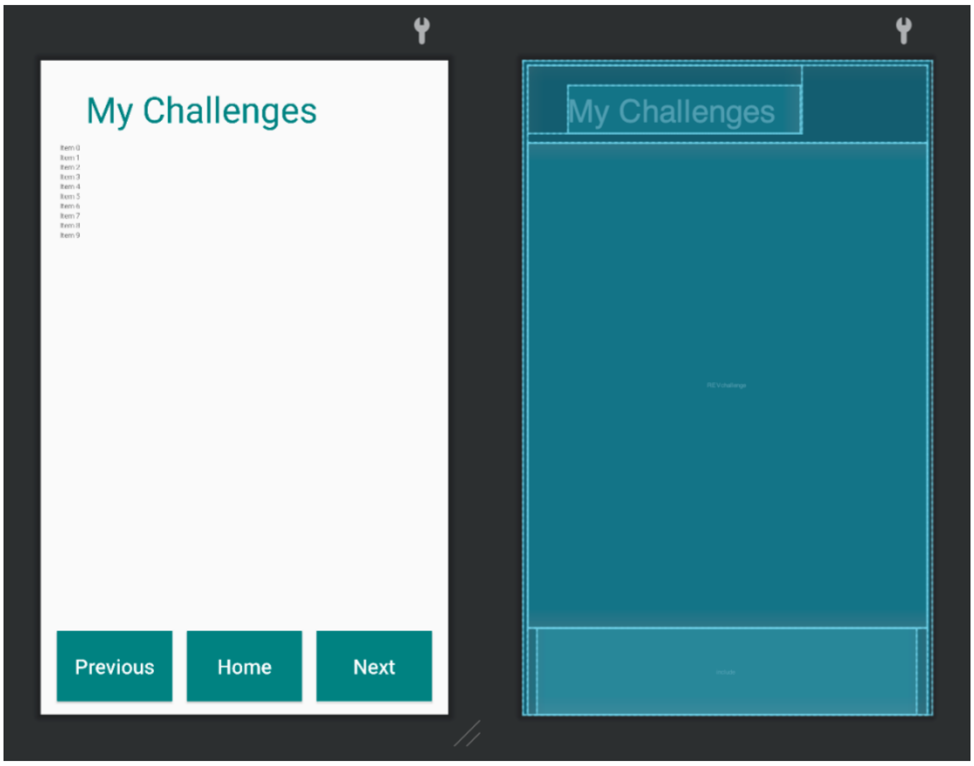


Figure 7.17: UI for Challenges Activity

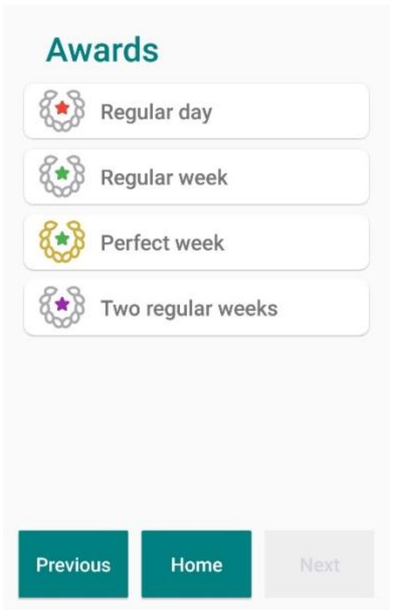


Figure 7.18: UI for the awards collection

On the other hand, the Challenge description activity has been changed, to be more consistent with the whole system.

Figure 7.19 shows the first user interfaces that were implemented. The idea was to put the icon and the name of the challenge at the top-center. Below that, the description of the challenge and then the date of achievement. However, after review we realized this design was not in line with the already existing application.

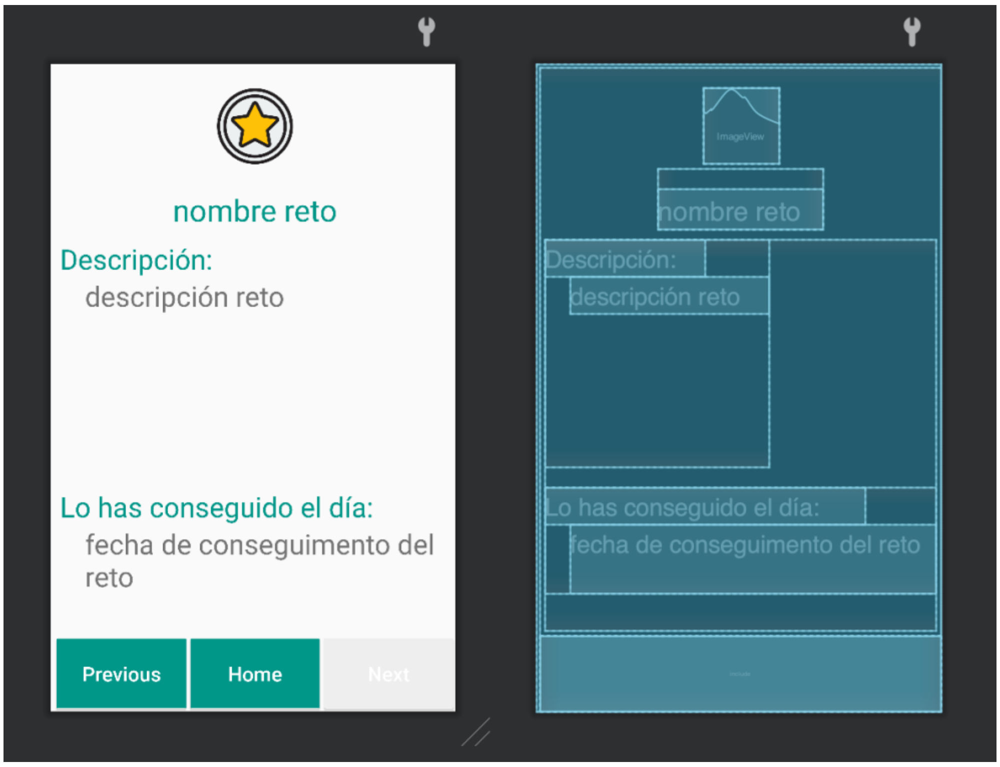


Figure 7.19: UI for Challenge description

We agreed to differentiate two interfaces for Challenge Won and Next Challenge. Figure 7.20 represent the Won Challenge interface. It shows the icon of the challenge and the name. Then, the date of the achievement. Finally, a short motivational message is provided.

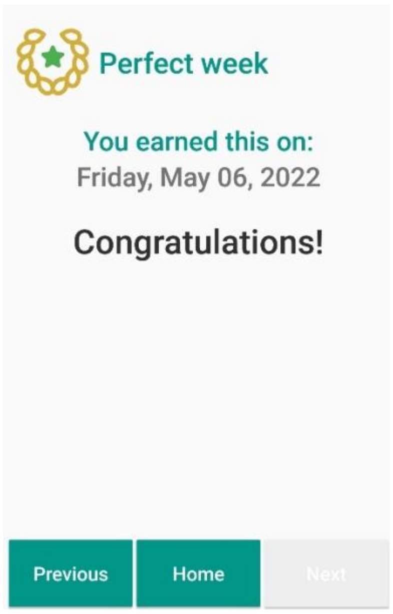
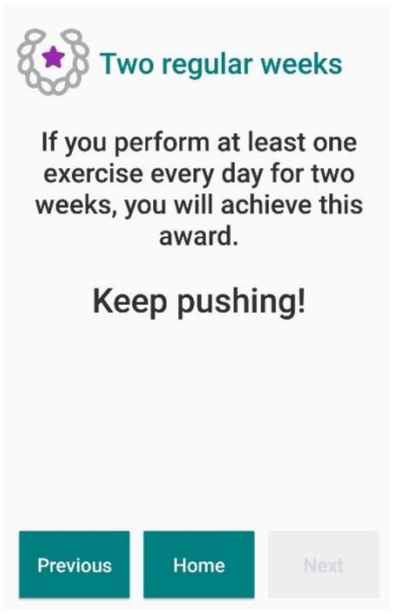


Figure 7.20: UI of specific won challenge

The description of the challenge has been kept only for the Next Challenge (Figure 7.21). In addition, a relevant motivational message is shown.


Two regular weeks

If you perform at least one exercise every day for two weeks, you will achieve this award.

Keep pushing!

Figure 7.21: UI of specific next challenge with description

An additional difficulty encountered was the decision about the design of the icons. Indeed, the icons might prove difficult for older people to understand. As suggested by UHOA 11 and 13 (Section 2.2), icons should be concrete and semantically closed. Initially, thought was given to how to represent each challenge through an image. A small calendar was thought of to represent monthly challenges, or a box with a number inside for daily ones. However, the meaning was not obvious. As a second approach, we were directed toward an icon representing a laurel wreath with a star inside (Figure 7.22). To differentiate the different challenges, different colors were used. Red was selected to identify daily challenges, green for weekly challenges, purple for 2-weeks challenges, blue for monthly challenges. Finally, gold for the last challenge. In addition, silver color in the wreath was used to differentiate the Regular category challenges, and gold color was used for the Perfect ones.

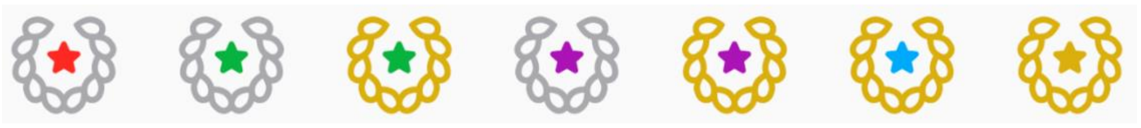


Figure 7.22: Icons of each award

7.3. User Statistics

7.3.1. Description

User Statistics have been identified by [3] as a *Key Resource* for patients with *High* or *Very High* levels of the Personal Mastery trait.

7.3.2. Low fidelity design

Sokól proposed representing user statistics as a diagram showing the evolution of users' own performance over time. However, the topic of statistics for older adults is a highly contested one. In fact, many people currently over 65 have not received a comprehensive mathematical education. For this reason, graphs and numbers could require too much cognitive effort. Also, an expert pointed out that a two-axis diagram is a foreign concept to this user target.

Following the suggestions received, the low-fidelity prototypes have been designed. Figure 7.23 shows the Statistics Menu, reachable from *Homepage / My Progress / My Results*, in which 12 buttons are available. These buttons will represent the 12 weeks of the program. By clicking one of them, the interface changes to Figure 7.24. It will display the daily evaluation statistics for that specific week.

To be consistent with the ActiveUP application, the navigation bar will be always available on all the screens.

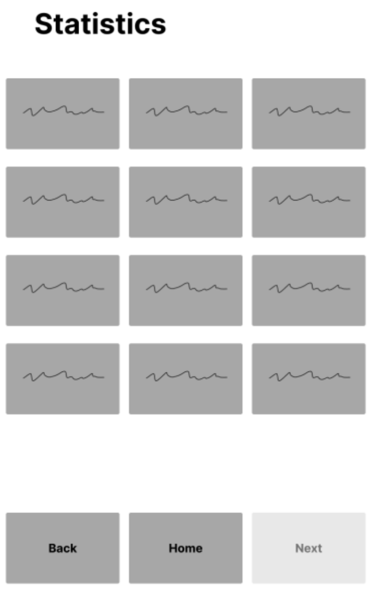


Figure 7.23: Wireframes of User Statistics menu

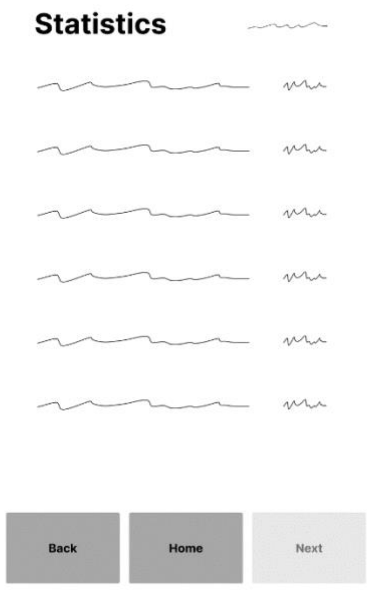


Figure 7.24: Wireframes of User Statistics

7.3.3. Implementation

7.3.3.1. Server-side

To implement the User Statistics feature a new GET Request has been created on the server. The function `getStatistics` retrieves data about the evaluation of a specified user. It receives as parameter the `patient id`, and it returns a list of *StatisticsMeasurement* objects.

So far, no new tables have been added to the database. The reason is that the desired data are already stored in the table *"eval"*. This table is updated every day.

7.3.3.2. Client-side

On the ACTIVE-UP application, as shown in Figure 7.25, the files related to the statistics are in the packages **com.activeup.activities.statistics** and **com.activeup.objects.asistente_motivacional.**

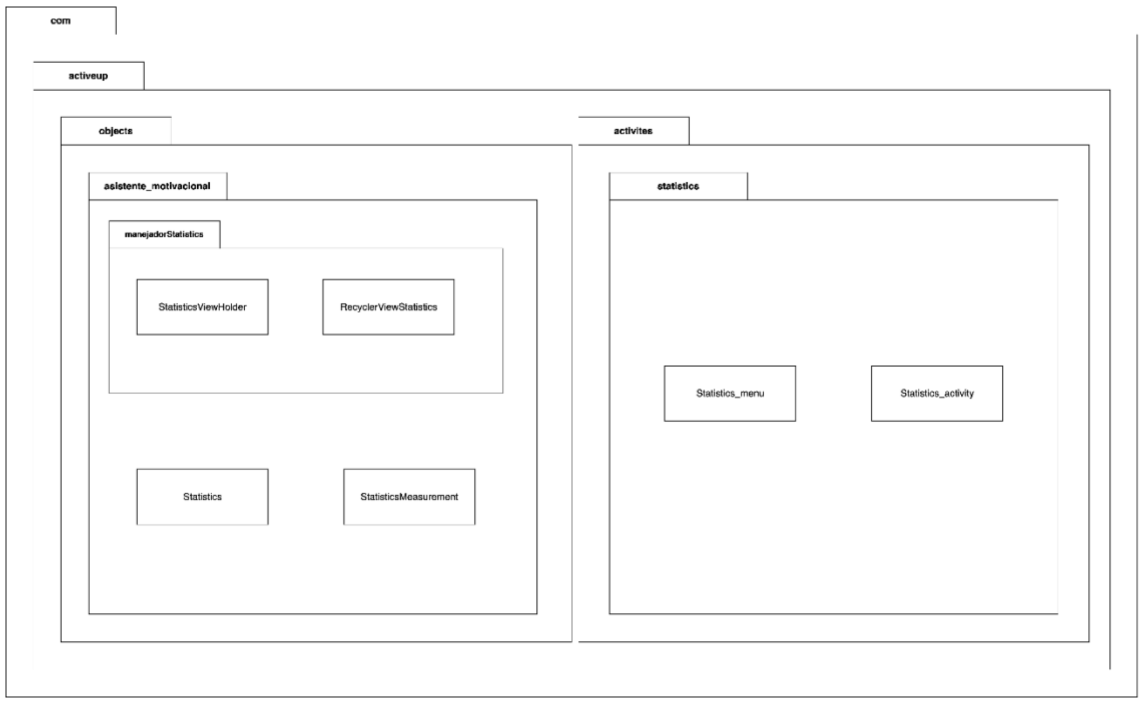


Figure 7.25: UML of User Statistics

Statistics is the main object. To create an instance of the object, the system executes a `new Statistics(Context context, RecyclerView recyclerView, String token, String user_id, int activity_id)`. The class has the constructor and a private class `ObtenerStatistics`. `ObtenerStatistics` is in charge of requesting to the server the user's statistics. Once the data are retrieved from the database, they are sorted by week, in order to be later displayed in the proper interface.

RecyclerViewStatistics is a RecyclerView. It dynamically creates the element defined by the StatisticsViewHolder.

StatisticsViewHolder is a RecyclerView.ViewHolder. It represents the view of the object. It has two TextView (day of the week and evaluation).

StatisticsMeasurement is the return type of the GET request.

Statistics_menu extends BaseActivity. In the `onCreate` method, setContentView sets the user interface to display the menu. In this interface, a button for each week of the training program is displayed.

Statistics_activity also extends BaseActivity. The class displays the `statistics_activity.xml`. The content will be a list with all the daily statistics. The statistics are shown with the day of the week and the evaluation expressed in % (of completeness).

7.3.4. Final User Interfaces

The user interfaces created for the feature User Statistics are XML files stored in the layout folder. In particular, `row_statistics.xml`, `statistics_menu.xml`, and `statistics_activity.xml`. Compared to the low-fidelity prototypes, no relevant changes have been performed.

Figure 7.26 presents the Statistics menu, which has been implemented as planned.

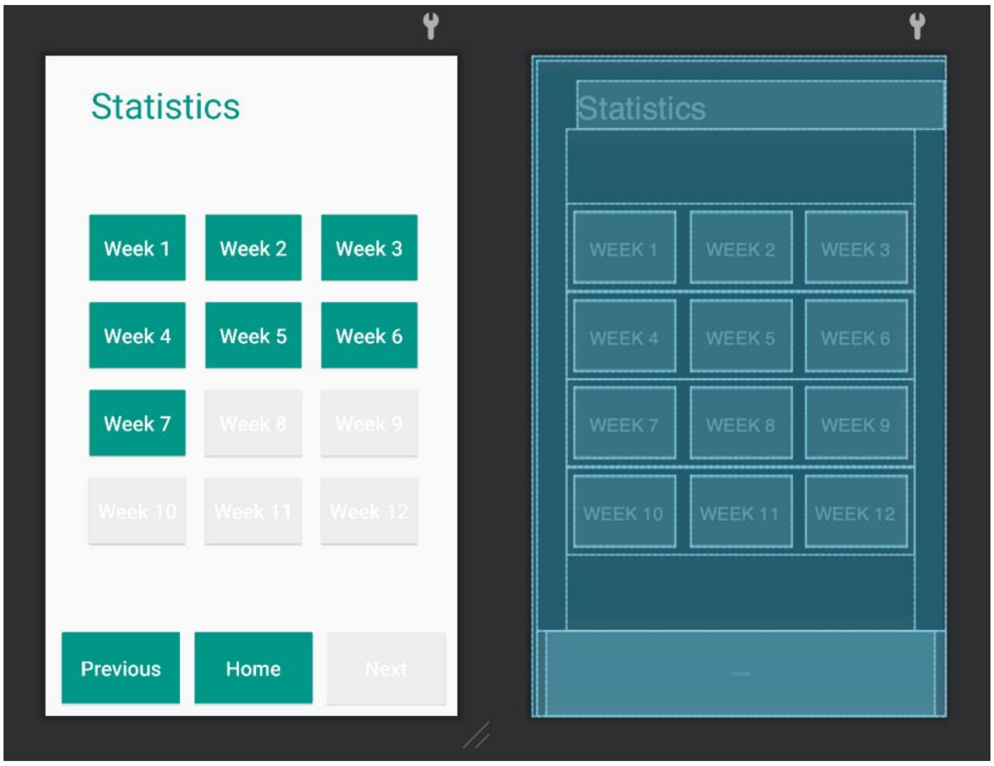


Figure 7.26: UI of Statistics Menu

The Statistics Activity, as well, has been implemented as planned (Figure 7.27). The RecyclerView takes care of the list of items and adapts the content.

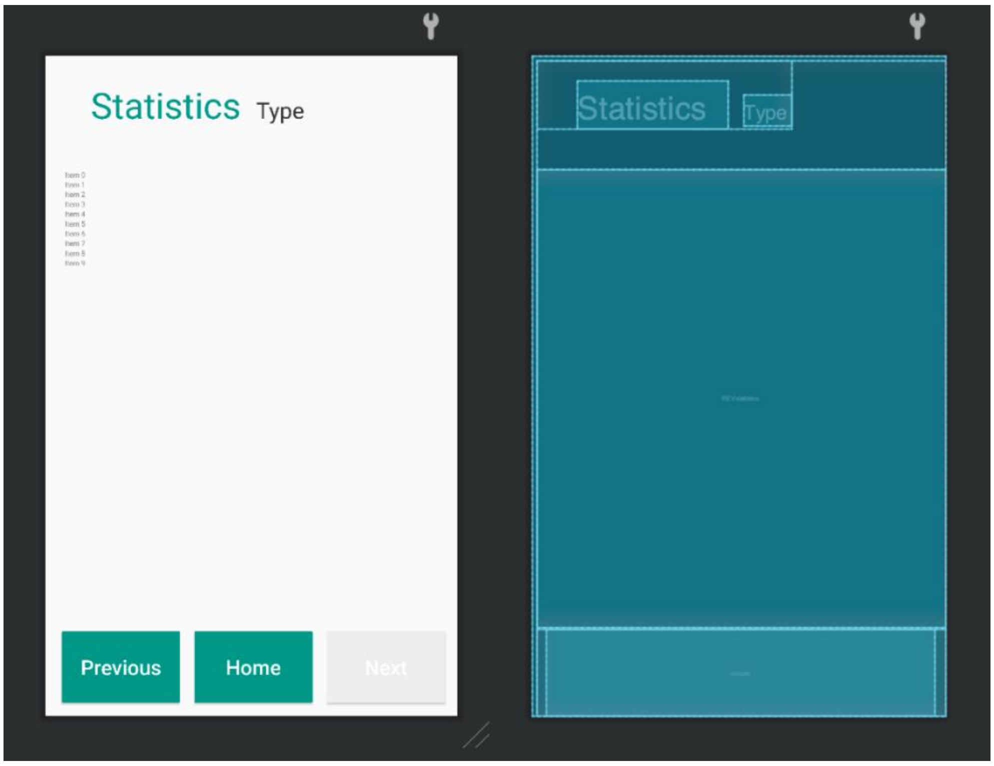
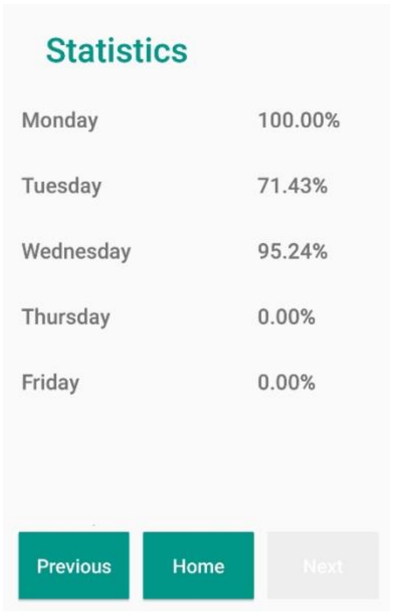


Figure 7.27: UI of Statistics Activity

Figure 7.28 represents a real example of the user interface created by Statistics Activity. On the left, the day of the week is displayed, and the evaluation is placed next on the right.


Statistics

Monday	100.00%
Tuesday	71.43%
Wednesday	95.24%
Thursday	0.00%
Friday	0.00%

Figure 7.28: UI of Statistics Activity filled in with data

This feature has been the last one to be realized. Due to time constraints, it was not possible to evaluate this functionality during the Cognitive Walkthrough later explained.

7.4. Alerts

7.4.1. Description

During the 12-weeks program, the motivational assistant sends multiple notifications and alerts to notify the user about specific situations.

So far, six different scenarios have been established, as reported in the table below (Table 7.4).

Table 7.4: Types of Alerts

Scenario	**Description**
First user connection	When the user opens the application for the first time, the motivational assistant introduces itself. Depending on the motivational traits, the dialogue is customized.
Exercise session interrupted	In the event that the user interrupts an exercise session, the next time they open the application, the motivational assistant will ask about the reasons for the interruption. In addition, it will propose to start a new exercise session.
Two days of inactivity	The user is supposed to do exercises every day (the number is based on the Vivifrail plan). If the user does not perform exercises for two days, the motivational assistant will ask about the reasons for such an absence. In addition, it will propose to start a new exercise session.
New challenge	If the user fulfills the requirements to play some challenge, at the beginning of every week they might receive an alert about a new challenge.
Challenge won	If the user is eligible for challenges, and they win one of them, the motivational assistant will congratulate the user.
Changes in the leaderboard	If the user fulfills the requirements to compete against other users, the motivational assistant will notify them every time their position on the leaderboard changes.

During these dialogues, the user could be asked to answer some questions or just read the communications.

7.4.2. Low fidelity design

A low-fidelity design has been proposed during the first phase of the development. The basic structure of the interface contains a few elements. Since elderly people are the target users of the application, some interaction methods were excluded. For example, notifications do not appear as pop-up windows. Older people often find it difficult when these components appear. For this reason, a dialogue consisting of several steps in which the user is guided was devised.

An icon representing the motivational assistant is at the top of the screen, meaning that it is the one speaking to the user. The content of the communication is under the icon, in the center of the interface. Depending on the nature of the alert, some buttons could be placed under the message. Finally, at the bottom, the navigation bar is located, to be consistent with the whole application. The user interaction provokes changes in the interface. For instance, if multiple choices are available, by clicking on one of them, the color of the others changes. In addition, once an alternative is picked, the next button in the navigation bar is enabled.

Figure 7.29 and Figure 7.30 represent an example of a low-fidelity design for this feature.

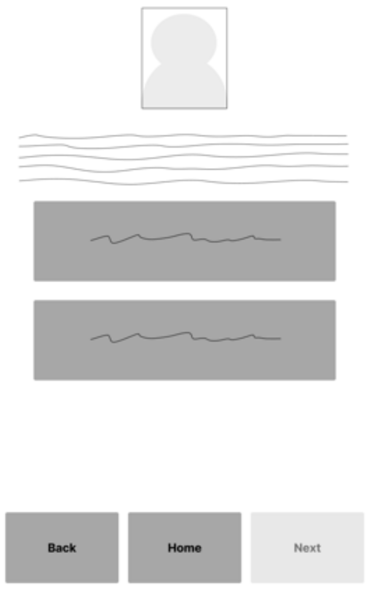


Figure 7.29: Low fidelity design of the alerts (before the user interaction)

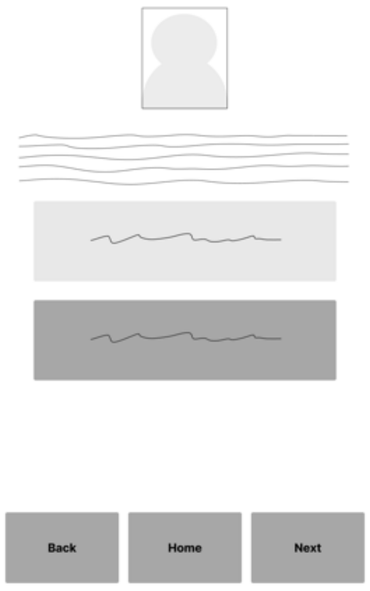


Figure 7.30: Low fidelity design of the alerts (after the user interaction)

7.4.3. Implementation

7.4.3.1. Server-side

To implement this feature, four new tables have been added in the database (Figure 7.31).

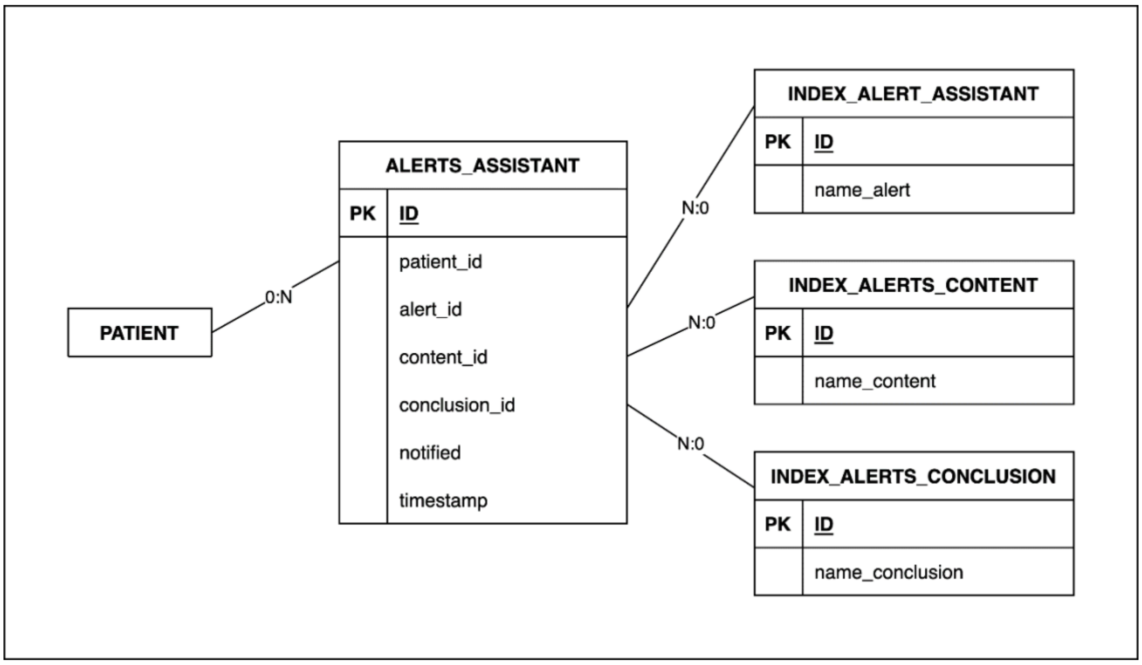


Figure 7.31: Entity-Relationship diagram for the Alert feature

The following elements have been included in the server:

In the **models** folder:

- alerts_assistant_model.py

```
Column (“id”, Integer, primary_key=True),
Column (“alert_id”, Integer),
Column (“content_id”, Integer),
Column (“patient_id”, Integer),
Column (“notified”, Integer),
Column (“conclusion_id”, Integer),
Column (“timestamp”, TIMESTAMP)
```

- index_alerts_content_model.py

```
Column (“id”, Integer, primary_key=True),
Column (“name_content”, VARCHAR(50))
```

ALERTS_ASSISTANT (id, patient_id, alert_id, content_id, conclusion_id, notified, timestamp)

The field ‘`id`’ refers to a unique id of each row. The id of the patient is stored in the field ‘`patient_id`’ and it is taken from the already existing table *“patient”*. The id of the alert is stored in the column `'alert_id'`. Every value has a description in the table *"index_alerts_assistant"*.

The column `'content_id'` contains a value that refers to the specific content for the alert. The default value is 0. For instance, if the user has gained positions on the leaderboard because they have performed better during the last exercise session, the `'content_id'` will be set to 11. Accordingly, the motivational assistant will display a congratulatory message to the user.

The column `'conclusion_id'` refers to the specific conclusion of the alert. This value can indicate multiple concepts. For instance, in the case of two days of inactivity, depending on the answers of the user, this value will change.

The default values for the columns `'notified'` and `'conclusion_id '` are 0. The 'notified' changes to 1 once the alert has been read by the user. The field `'timestamp'` contains the timestamp of the insertion of the alert. This timestamp is not updated when the fields `'notified'` and `'conclusion_id '` change.

INDEX_ALERTS_ASSISTANT (id, name_alert)

As stated above, so far, it has been implemented 6 types of alerts. The following table represents the contents of this table in the database.

Table 7.5: "index_alerts_assistant" table with id and related name of the type of alert

id	name_alert
1	First connection
2	Exercise session interrupted
3	Two days of inactivity
4	New challenge
5	Challenge won
6	Changes in the leaderboard

The field `'id'` identifies the alert, and it is referenced in the table *"alerts_assistant"*. The column `'name'` explains the specific alert type.

INDEX_ALERTS_CONCLUSION (id, name_conclusion)

Depending on the user response to the alert the conclusion changes. The following table represents the table in the database with all the possible alternatives.

Table 7.6: "index_alerts_conclusion" table with id and related name of all the possible conclusions

id	name_conclusion
1	Lack of problems
2	Problems
3	Absent caregiver
4	Disadvantages of exercise
5	Doubts about benefits
6	Lack of desire
7	Health problems
8	Technical problems
9	I prefer not to say
10	Other occupations/holidays
11	Forgotten to end
12	Lack of time
13	Lack of energy
14	Major disruption
15	First connection made

The column `'id'` refers to the type of conclusion, while the field `'name_conclusion'` describes the specific conclusion. The `'id'` is used in the table *"alerts_assistant"*. This value is established based on the answers the user provides.

INDEX_ALERTS_CONTENT (id, name_content)

For each specific alert scenario (for example New challenge), the content of the alert may change (for example the current challenge to be presented). The following table represents the table in the database with all the possible alternatives.

Table 7.7: "index_alerts_content" table with id and related name of all the possible contents

id	name_content

1	Intro Motivational Assistant
2	Intro Leader board
3	Intro Challenges & User Statistics
4	Reto 1
5	Reto 2
6	Reto 3
7	Reto 4
8	Reto 5
9	Reto 6
10	Reto 7
11	Competitive user rises in the daily rankings because they have improved
12	Competitive user drops in the daily ranking because they have worsened
13	Competitive user rises in the daily rankings because others have fallen in the rankings
14	Competitive user drops in daily rankings because others have improved

The field `'id'` identifies the different types of content. The column `'name_content'` describes the specific content. Through the `'id'` the content is referenced in the table *"alerts_assistant"*. It has been decided to add the concept of content in order to make the alert feature more scalable and easier to upgrade.

1.1.1.1.1. Generation of the alert

The rows in the *"alerts_assistant"* table can be inserted in different ways.

Alerta Asistente Tipo 1: First connection

During the first setup, every patient receives a value for the five different motivational traits. At the end of this process, a row with `'alert_id'` equal to 1 (First connection) is added to the table. Depending on the values of the motivational traits, the `'content_id'` can be set to 1, 2, or 3. 1 means that the user will not see Challenges, Leader board, or User statistics. 2 means that the user fulfills the requirements for the Leader board, so a tutorial is provided by the motivational assistant upon the first connection. 3 means that the user fulfills the requirements for the Challenges and the User statistics, so a tutorial is provided by the motivational assistant upon the first connection. Here below, an explanation and a scheme (Figure 7.32) are provided.

The leader board will be shown to all users with very high or high levels of Competition Seeking (unless they also have high or very high levels of Failure Avoidance).

The challenges and users' personal statistics will be only visible to individuals with high or very high levels of Personal Mastery.

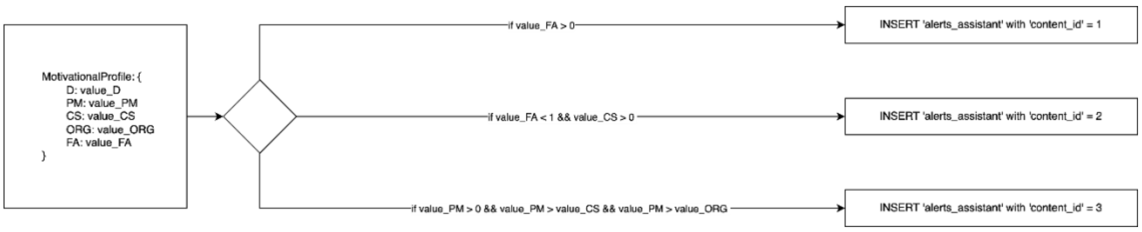
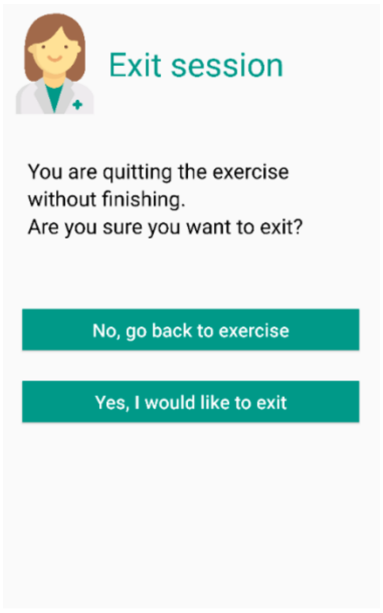

MotivationalProfile: {
D: value_D
PM: value_PM
CS: value_CS
ORG: value_ORG
FA: value_FA
}
if value_FA > 0
if value_FA < 1 && value_CS > 0
if value_PM > 0 && value_PM > value_CS && value_PM > value_ORG
INSERT 'alerts_assistant' with 'content_id' = 1
INSERT 'alerts_assistant' with 'content_id' = 2
INSERT 'alerts_assistant' with 'content_id' = 3

Figure 7.32: Algorithm that computes which features will be displayed to the user, and consequently which 'content' specify in the alert

Alerta Asistente Tipo 2: Exercise session interrupted

During an exercise session, the user always can stop it. In this case, a confirmation message is displayed (Figure 7.33). If the user decides to quit the session a row with `'alert_id'` equal to 2 is added to the table.


Exit session

You are quitting the exercise without finishing.
Are you sure you want to exit?

Figure 7.33: Window that appears to the user when they try to exit from an exercise session

Alerta Asistente Tipo 3: Two days of inactivity

Every day at midnight, the system computes the evaluation of the exercise session for every patient. If the user has not performed any exercises in the last two days (excluding the weekend), a row with *'alert_id'* equal to 3 is added to the table.

Alerta Asistente Tipo 4: New challenge

Based on the algorithm explained in Section 7.2.3, every time a new challenge has to be proposed to the user, a row with *'alert_id'* equal to 4 is added to the table. In addition, the value for the column *'content_id'* is also set.

Alerta Asistente Tipo 5: Challenge won

During the midnight evaluation of the performance, the system checks if the user completed a specific challenge. If so, a row with *'alert_id'* equal to 5 is added to the table. In addition, the value for the column *'content_id'* is also set.

Alerta Asistente Tipo 6: Changes in the leader board

Every day the leader board is re-computed to be always up to date. The file in charge of this is **leaderboard.js** (Appendix A.2). This file is executed every day at midnight. It is in charge of evaluating if a user has gained or lost positions on the leader board. In particular, it first downloads the ranking of the current date, then the one of the previous day. Finally, it compares the two lists and computes the difference between the positions. Depending on the value of this difference, a row with *'alert_id'* equal to 6 is added. The *'content_id'* value is also set according to the situation.

The alerts are supposed to be displayed once the patient opens the app.

Multiple alerts of the same type might be added to the database. A concrete example is when a patient does not do exercises for a week. In this case, the server will insert more than one row with *'alert_id'* equal to 3. Once the user reads the first notification, automatically a trigger deletes the previous alerts of the same type.

7.4.3.2. Client-side

On the ACTIVE-UP application, as shown in Figure 7.34, all the files related to the alerts are in the package **com.activeup.objects.asistente_motivacional.**

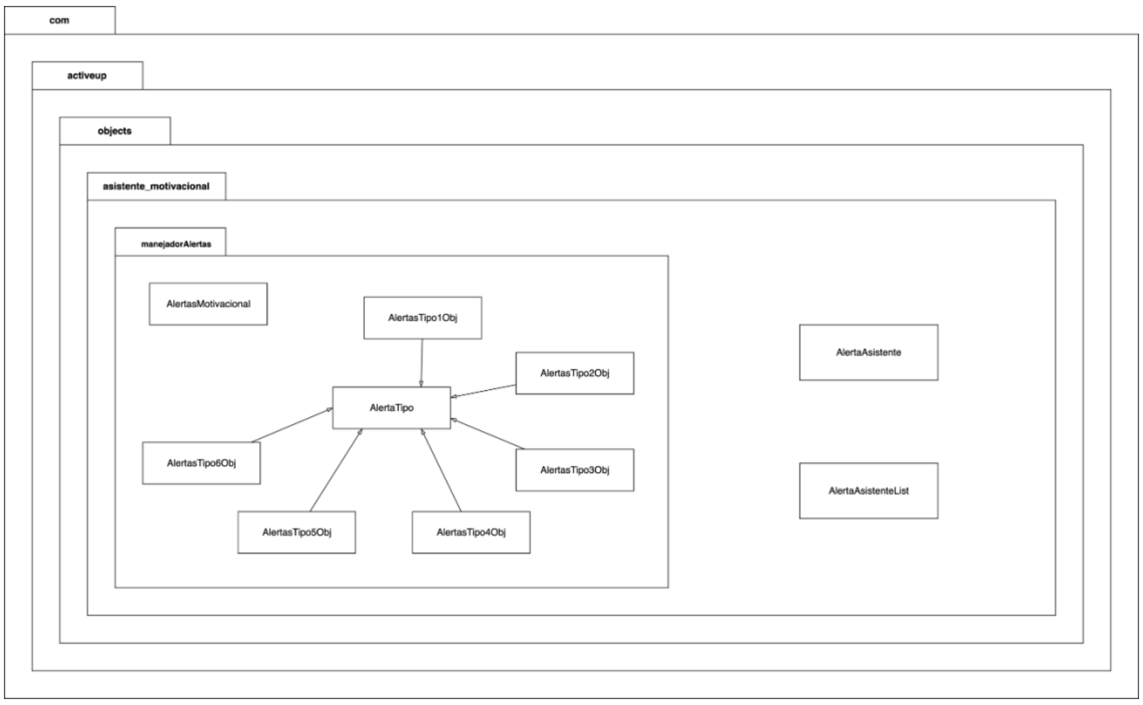


Figure 7.34: UML for the Alert feature

AlertaAsistente represents the main object. It has three attributes: alert_id, id, and id_contenido. This class only contains the constructor, getters, and setters.

AlertaAsistenteList represents a set of objects AlertaAsistente. It has the constructor, getter, and setter.

Inside the package *manejadorAlertas*, the alerts are managed.

The system, after the Login process, runs a `new AlertasMotivational(Context context, int pacient_id)`. By doing so, all the alerts related to the specified user are downloaded and managed. Once the user receives the notifications, an UPDATE request is sent to the server to update the field 'notified' to 1.

The two main methods in *AlertaMotivacional* are `obtenerAlertasServidor` and `evaluarAlertas`. The former performs a GET request to the server and retrieves the data from the database. Once the data have been downloaded, the latter evaluates every row. During the evaluation, the values of *alert_id* and *content* are checked. According to the first, one of the *AlertasTipoObj* is called. Then, depending on the *content*, the content of the notification is customized.

AlertaTipo is an object that contains the public method `cargarMotivacion()`. This method detects the dominant motivational trait, change state, and motivational status of the user. The function returns an integer value that will later be used to personalize the motivational assistant messages.

All the six *AlertasTipoObj* extend the parent object *AlertaTipo*. These child objects create and show dialogues. The methods in the parent class *AlertaTipo* can be invoked here.

7.4.4. Final User Interfaces

The user interfaces created for this feature are XML files stored in the layout folder. In particular, the files are:

- `asistente_motivacional_alerta_tipo_1.xml`
- `asistente_motivacional_alerta_tipo_2.xml`
- `asistente_motivacional_alerta_tipo_3.xml`
- `asistente_motivacional_alerta_tipo_4.xml`
- `asistente_motivacional_alerta_tipo_5.xml`
- `asistente_motivacional_alerta_tipo_6.xml`

During the implementation, some changes have been made. To keep this feature consistent with the already existing application, it has been decided to move the icon to the left and add a title at the right of the image. Under this part, the message of the motivational assistant and buttons can be displayed.

In this section, Figure 7.35, Figure 7.36, and Figure 7.37 display the layout for alerts. Figure 7.35 represents a simple notification with the message from the motivational assistant.

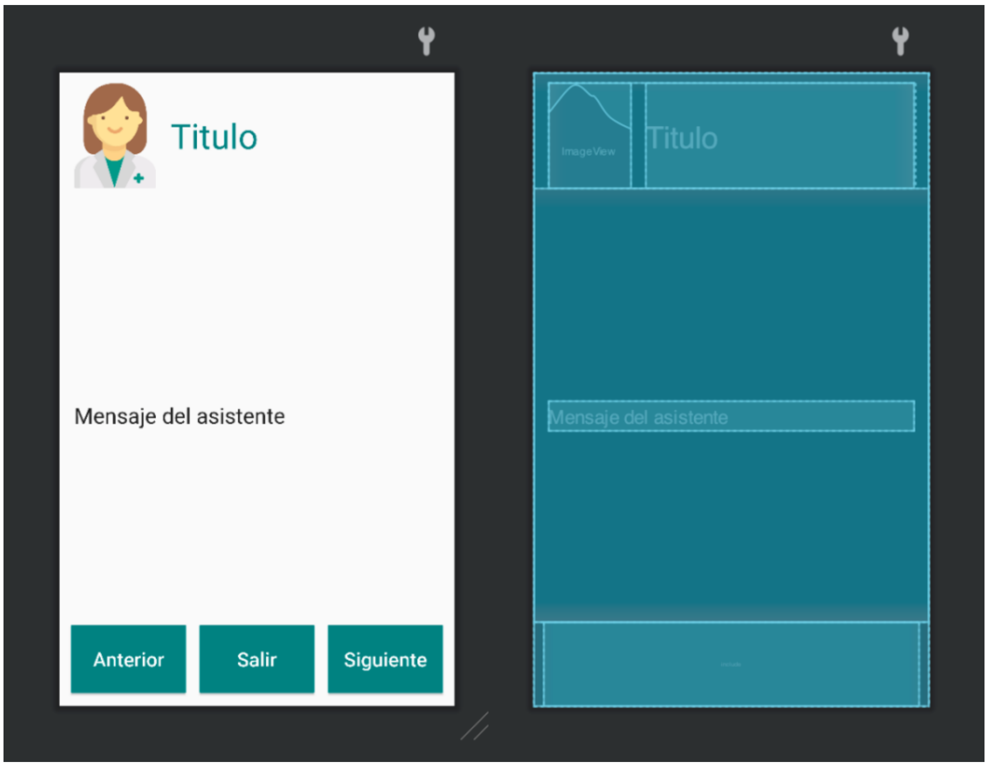


Figure 7.35: UI for Alerts with a plain text message

Figure 7.36 displays a question asked by the motivational assistant. Below that, the possible answers are provided inside buttons. In this situation, the user has not interacted with the interface. For this reason, the buttons are green with white text. Since the user has not picked an option yet, the Next button in the navigation bar at the bottom is disabled. It will be enabled after the user intervention.

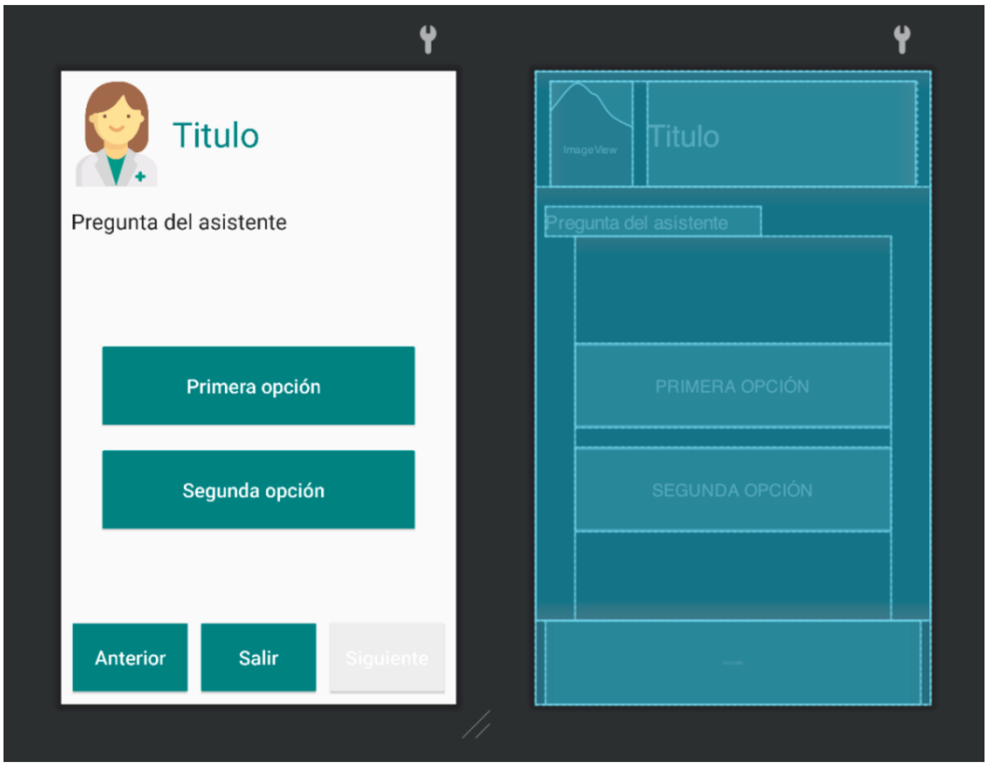


Figure 7.36: UI for Alerts with a question and possible answers

Figure 7.37 represents the user interface after the user interaction. In this case, the user has clicked on the first answer. The color of the second changes. Indeed, the button becomes gray with green text. In addition, the Next button on the navigation bar at the bottom is now enabled.

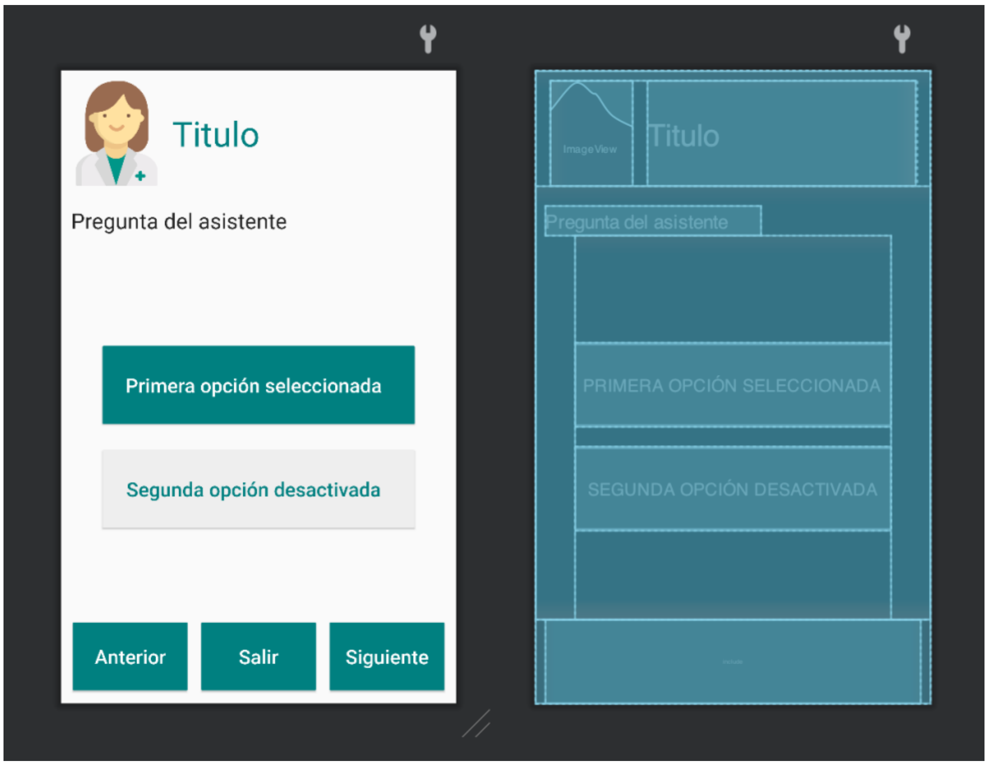

Titulo
Pregunta del asistente
Primera opción seleccionada
Segunda opción desactivada
Anterior
Salir
Siguiente
ImageView
Titulo
Pregunta del asistente
PRIMERA OPCIÓN SELECCIONADA
SEGUNDA OPCIÓN DESACTIVADA

Figure 7.37: UI for Alerts with a question and one answer selected

7.4.5. Communication from the Motivational Assistant

When older adults are the end-user, it is not enough to implement user interface measures to ensure a satisfactory level of easy-to-use. Indeed, people over the age of 75 have a very different level of education than typical users of technological devices. This fact impacts mainly two aspects, language, and level of attention.

Language, according to UHOA 16 (Section 2.2) must be familiar. This means using simple terms and clear feedback in the mobile application. In this way, older users, regardless of their cultural background, can understand it and the technology does not create anxiety.

The level of attention, (UHOA 19, Section 2.2) can be ensured by shortness. Instructions and text should be short, in order to avoid overwhelming the older user with the cognitive effort of reading extensive messages.

Here below, all the messages or the buttons' text used in the Alert feature are shown.

Alert Type 1:

Table 7.8: Messages the MA shows to the user during the first connection

Name	Message

primera_conexion_usuario	During these 12 weeks of training, I will be your assistant and I will help you to do exercises every day.
Clasificacion1	If you do your exercises every day, you will manage to stay at the top of the leaderboard.
Clasificacion2	This is how the leaderboard appears.
Retos1	From time to time, I will set you a challenge. If you complete it successfully, you will win a new award!
Retos2	Once you win the challenge, you will be able to see your awards collection.
Statistics1	I know you like to keep track of your progress!
Statistics2	Every time you want to check your statistics, just go to the Statistics section.

Alert Type 2:

Table 7.9: Messages the MA shows to the user after an interrupted exercise session

Name	**Message**
sesion_interrupida_1	I noticed that you didn’t finish your last exercise session.
Sesion_interrupida_2	Can you tell me why you didn’t finish your session?
Sesion_interrupida_3	I didn’t have the energy to finish.
Sesion_interrupida_4	I was interrupted by something important.
Sesion_interrupida_5	I didn’t have time for all the exercises.

Alert Type 3:

Table 7.10: Messages the MA shows to the user (Alert Type 3)

Name	**Message**

informarAusencia	I noticed that it has been some time since your last exercise session.
preguntaProblemas	Have you had any problems?
preguntaMotivo	Good! Why were you absent?
preguntaPorQueNo	Why couldn't you do the exercises?
tristeEmpatia	I'm sorry to hear that, can you tell me more about your problem?
botonSiProblema	Yes, I had a problem.
botonNoProblema	No, I have not had any problems.
botonPrefieroNoDecir	I prefer not to talk about it.
botonNoHePodido	I have not been able to do the exercises.
botonProblemasSalud	I have had health problems.
botonCuidadorAusente	My caregiver was not there to help me.
botonProblemasTecnicos	I have had technical problems.
botonOtrasOcupaciones	I have had other occupations.
botonPrecontemplacion	I don't understand why I need to exercise.
botonContemplacion	I think it is too difficult to exercise every day.
botonFaltaGanas	I didn't feel like exercising, but it won't happen again.
botonVacaciones	I have had other occupations or been on holiday.

Alert Type 4:

Table 7.11: Messages the MA shows to the user when explaining a new challenge

Name	**Message**
nuevo_reto	I have a new challenge for you! Do you want to know about it?

Reto_1_description	If you are going to do all the exercises for one day, you will get this achievement.
Reto_2_description	If you are going to do at least one exercise every day for a week, you will achieve this milestone.
Reto_3_description	If you are going to perform all exercises with all repetitions every day for a week, you will get this achievement.
Reto_4_description	If you are going to do at least one exercise every day for two weeks, you will achieve this milestone.
Reto_5_description	If you are going to perform all the exercises with all the repetitions every day for two weeks, you will achieve this accomplishment.
Reto_6_description	If you perform all exercises with all repetitions every day for a month, you will achieve this accomplishment.
Reto_7_description	Since you gained all the previous awards. Now, you have the opportunity to win the most important one. If you are going to perform all exercises with all repetitions every day during this week, you will get this achievement!

Alert Type 5:

Table 7.12: Messages the MA shows to the user when informing them about a won challenge

Name	**Message**
reto_conseguido	Thanks to your efforts you have earned a new achievement! Do you want to see it?
preguntaReto	Want to see your collection of achievements?

Alert Type 6:

Table 7.13: Messages the MA shows to the user when notifying them about changes in the leader board

Name	**Message**

alerta_6_tipo_1	Since your last session other people have worked hard and overtaken you in the rankings. Now you have the opportunity to regain your place.
Alerta_6_tipo_2	In your last session you managed to move up in the rankings
alerta_6_tipo_3	In the last session you were not at your usual level and lost positions in the standings. I hope that today you have the strength to try to recover your position.
Alerta_6_tipo_4	Since your last session you have moved up in the standings.
preguntaVerClasificacion	Do you want to see your position in the leader board?

Among the main features of the motivational assistant is that it is adaptable and can customize its "behavior" depending on the situation and the user with whom it is interacting. For this reason, the motivational assistant can tailor the messages to be delivered to the user based on the dominant motivation trait and the user's stage of change. The tables below provide some examples of tailored messages, proposed by [3].

Table 7.14 provides the possible answers the MA gives to the user if they decide to start an exercise session.

Table 7.14: Personalised feedback when the user decides to start an exercise session

	Precontemplation	**Contemplation**	**Preparation**	**Action/ Maintenance**
D	"Fantastic! Regularity means constant improvement."	"Brilliant! Regular exercise makes you feel better and have more energy."	"That is wonderful! Exercise will help you maintain your progress."	"Excellent! There is nothing better for your health than exercise."
PM	"Awesome! You are one step closer to	"That is wonderful! The time spent	"Brilliant! Regular exercise is	"Awesome! Regular exercise prevents many diseases."

	achieving your goals."	exercising always pays off."	easy once you make it a habit."	
CS	"Awesome! You are one step closer to advancing in the classification of patients."	"Brilliant! Your results are remarkable."	"Wonderful! This way you will maintain your excellent results."	"Fantastic! Keep up the good job and you will be one of the top project participants."
ORG	"Awesome! Your dearest ones will be very proud of you."	"Brilliant! Your results are remarkable."	"Excellent! Getting stronger means being there for your dearest ones."	"Excellent! Being more fit means being there for your dearest ones."
FA	"It is awesome that you decided to exercise today!"	"Fantastic! You are doing a great job."	"Brilliant! I know that you can do it."	"Excellent! I believe in you."

Table 7.15 shows the messages the MA sends to the user if they decide not to perform an exercise session.

Table 7.15: Personalised feedback when the user decides not to start an exercise session

	Precontemplation	**Contemplation**	**Preparation**	**Action/ Maintenance**
D	"Alright but please do not forget about your goals."	"Okay but remember that regular exercise is important for your health."	"Alright but do not forget that regular exercise makes you feel better."	"That is a shame. Exercising would help you maintain your progress."
PM	"That is a shame. Exercising would help you	"Okay but remember that regular	"That is a shame. Regular	"I am sorry to hear that. Please remember that

	maintain your progress.”	exercise prevents many diseases.”	exercise is good for your heart.”	exercise makes your bones and muscles stronger.”
CS	“That is a shame. Regular exercise would help you advance in the classification of patients.”	“I am sorry to hear that. Remember that you have a chance to be one of the top project participants.”	“Alright but remember that exercise would help you maintain your results”	“Alright but did you know that finding a fixed time to do your exercise makes it easier?”
ORG	“Alright but please remember that many people have made it to follow this programme. You can do it, too!”	“Alright but please remember that regular exercise is easy once you make it a habit.”	“Okay but keep in mind that exercise is easy once you make it a habit.”	“Alright but please remember that many people have made it to follow this programme. You can do it, too!”
FA	“Alright, do not worry. Have a rest and come back later.”	“Alright, you are still doing a great job. Come back when you feel better.”	“Okay, no problem. Take a break and come back when you feel ready.”	“Alright, do not worry. I am sure that you will feel better after a small break.”

When the user receives the alert type 3 (2 days of inactivity), the MA asks about the reasons why they were absent. First, it asks whether they had a problem or not. Table 7.16 shows the messages the MA provides if the user explains they did not have a problem.

Table 7.16: Personalised feedback when the user explains they had a problem

	Precontemplation	**Contemplation**	**Preparation**	**Action/ Maintenance**

D	"I am glad to hear that. Did you know that regular exercise improves your sleep?"	"That is good to hear but please imagine how daily exercise would change your life!"	"That is great to hear but please remember that regularity is important."	"I am glad to hear that but please remember that regularity always pays off."
PM	"That is great to hear but please do not forget about your goals."	"That is good to hear but remember that regular exercise helps you maintain your progress."	"That is great to hear. Please remember that being consistent is important to achieve your goals."	"It is great to hear that. Please do not forget that exercise makes you stronger."
CS	"I am glad to hear that. Please remember that regular exercise helps you advance in the user classification."	"That is good to hear but please imagine how daily exercise would change your life!"	"I am glad to hear that but keep in mind that exercise is easy once you make it a habit."	"I am glad. Please remember that regular exercise helps you maintain your results"
ORG	"That is good to hear but please remember that getting stronger means being there for your close ones."	"I am glad to hear that but please remember that exercising is easier after a few times."	"I am glad to hear that. Your dearest ones will be proud that you connected yourself today!"	"I am glad to hear that. Your dearest ones will be proud that you connected yourself today!"
FA	"That is good to hear. I am proud that you decided to connect yourself today."	"I am glad. You are still doing a great job connecting	"That is good to hear. Did you know that finding a fixed time to	"I am glad to hear that. I hope that you are feeling better after this break."

		yourself today.”	do exercise makes it easier to follow the programme?”	

7.5. Further Changes

During the development of the previously explained features, an important design issue was how the user should reach them. The leader board, the challenges, and the user statistics are under the path *Homepage/My Progress/My Results*, as Figure 7.38 shows.

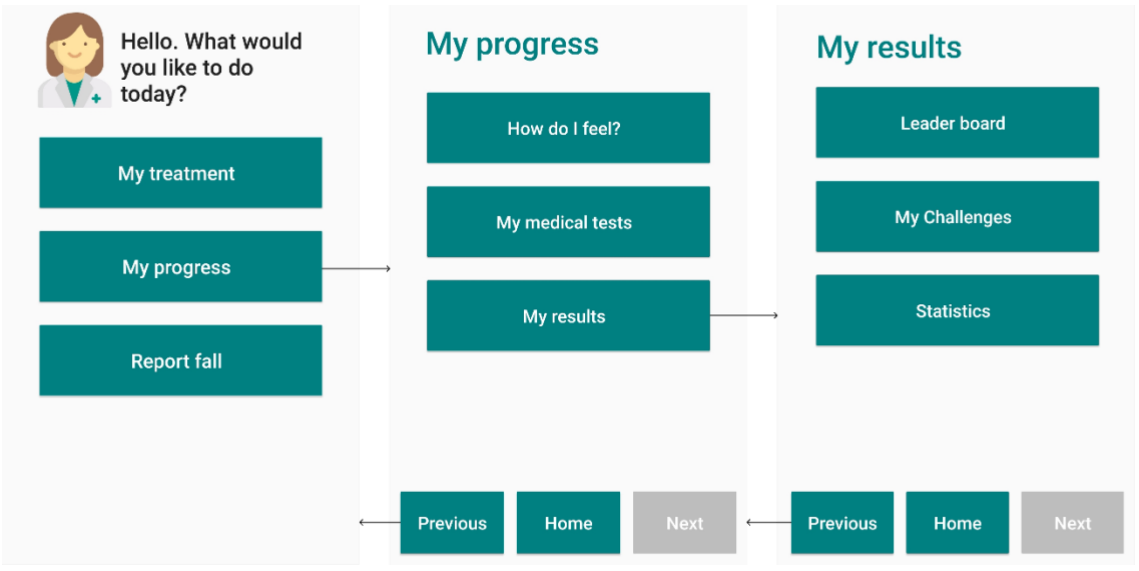


Figure 7.38: Path to reach the features

In the beginning, the possibility to reach these directly from the homepage was added, as shown in Figure 7.39. However, in order to agree with UHOA 9 and UHOA 10 (Section 2.2), this option was later removed (Figure 7.40). Indeed, the three buttons were placed at the top of the interface, which is a hard-to-reach point. In addition, older adults tend to have better user experiences with an application when layouts are simple, even at the cost of reducing the set of available functionalities.

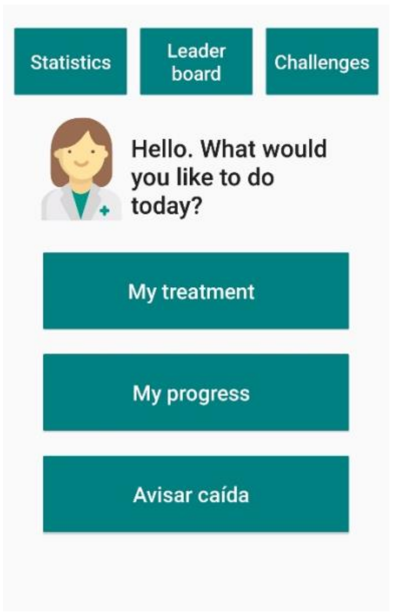


Figure 7.39: Homepage with the buttons at the top

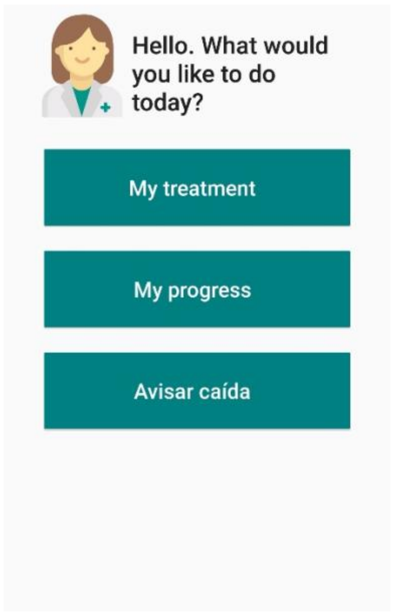
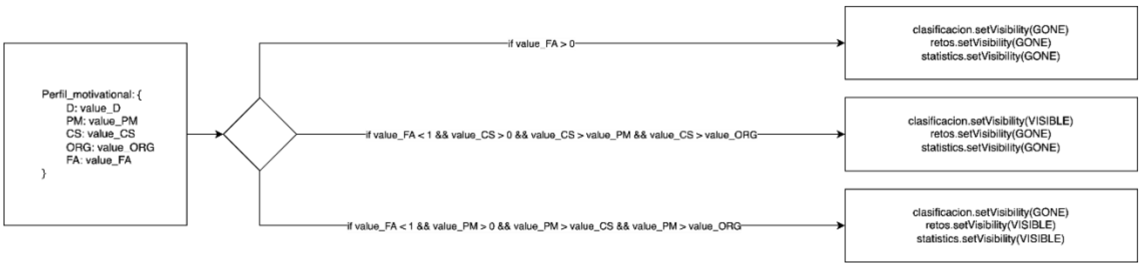


Figure 7.40: Homepage without the three buttons at the top

Nevertheless, since these features are a fundamental part of the motivational assistant, it was thought they were too hidden, so it was suggested to add a button to those already present in the homepage. So, either the Leader board button, or Challenges or Statistics, is placed below the Report fall button. The choice of which button to display is based on the dominant motivational trait. During the generation of the main menu, an object of the type `RasgoMotivacional` is created. The method `obtenerBotonesPersonalizados` is next called. This function first generates a GET request to obtain the values of each motivational trait from the server. It then checks whether the requirements to show one of the three buttons are met. If this is the case, it is set to *VISIBLE* via `setVisibility` (Figure 7.41). An example of the resulting homepage is represented in Figure 7.42.



```
Perfil_motivational: {
    D: value_D
    PM: value_PM
    CS: value_CS
    ORG: value_ORG
    FA: value_FA
}
```

```
if value_FA > 0
```

```
clasificacion.setVisibility(GONE)
retos.setVisibility(GONE)
statistics.setVisibility(GONE)
```

```
if value_FA < 1 && value_CS > 0 && value_CS > value_PM && value_CS > value_ORG
```

```
clasificacion.setVisibility(VISIBLE)
retos.setVisibility(GONE)
statistics.setVisibility(GONE)
```

```
if value_FA < 1 && value_PM > 0 && value_PM > value_CS && value_PM > value_ORG
```

```
clasificacion.setVisibility(GONE)
retos.setVisibility(VISIBLE)
statistics.setVisibility(VISIBLE)
```

Figure 7.41: Algorithm that computes which button will be shown on the homepage

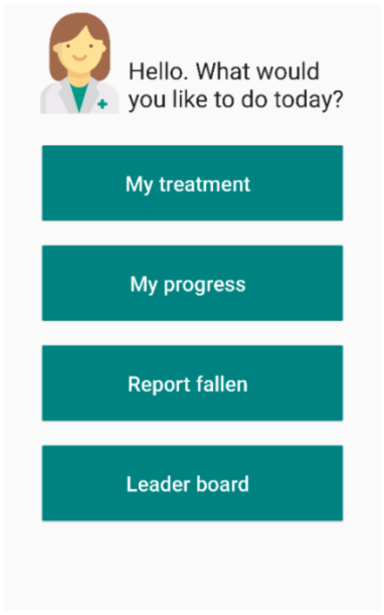


Figure 7.42: Example of the homepage, the Leader board is displayed because the user has Competition Seeking as dominant trait

8 Cognitive Walkthrough

A Cognitive Walkthrough is a technique for evaluating usability. It entails using a predesigned artifact or a functional prototype to complete a series of predetermined user tasks. This is normally done by a usability specialist who goes through the activities step by step, assessing how well a user would do at each stage, as well as which components can create confusion and which mistakes might arise [23]. It is the role of the usability expert to identify any possible deviations from the designer's ideal user task solution path [24].

The selection of user tasks to be evaluated is one of the most important things to address before doing a Cognitive Walkthrough. The chosen actions, according to [25], should be representational of the system. Furthermore, the usability analyst should answer a series of questions regarding the users' goals and knowledge, as well as the perceptible system and interface state, at each stage [25].

This usability evaluation approach, according to [23], is especially well-suited for the design of systems that may be new to the target user group. This may frequently be the case when building software applications for older persons.

The Cognitive Walkthrough was performed by a subject matter expert who works with older adults on a regular basis. A set of guiding questions were provided to the expert so that the system could be evaluated.

- *Is the cognitive load for the user too high?*
- *Is the language used understandable for older adults?*
- *Are there any usability criticisms?*
- *Will the user associate the correct action with the effect they are trying to achieve?*
- *Do changes in the user interface show an appropriate progress in the task?*

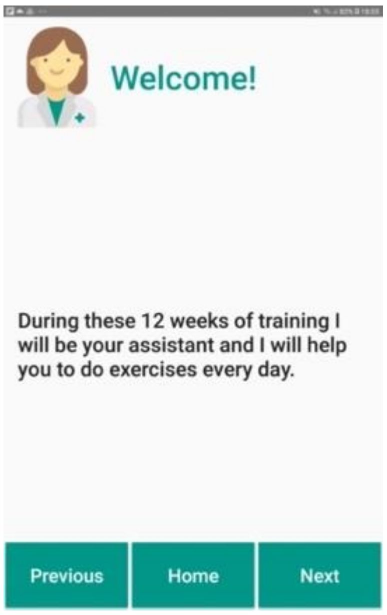
Three different scenarios have been evaluated:

1. *First connection of the user*
2. *Changes in the leader board*
3. *New challenge and achievement of it*

8.1. First connection of the user

Number	1

Scenario	First connection of the user
Description	The user is a 70-year-old patient who was prescribed with a Vivifrail exercise programme. They are robust and in an overall good health condition. They have been associated with the motivational trait Competition Seeking. They are starting the programme today. In the previous days, a technician explained them how the mobile application works.


Welcome!

During these 12 weeks of training I will be your assistant and I will help you to do exercises every day.

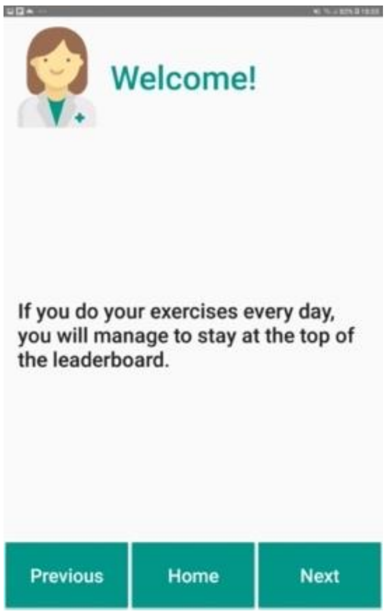
Figure 8.1: First interface of Alert type 1

The first time the user opens the application, the motivational assistant introduces itself. Figure 8.1 shows the first screen of the Alert type 1.

The expert explains that the image and the title are good, but the “Welcome!” should be placed horizontally-centered. The message is concise and understandable. Also, this text should be horizontally-centered.

He suggests increasing the padding and margin in the navigation bar.

In general, an increment of the margin should be implemented.

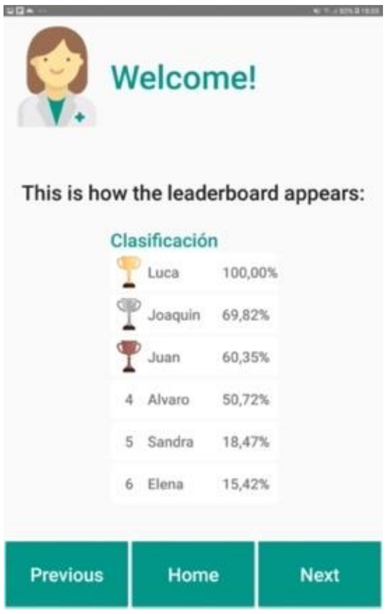

Welcome!

If you do your exercises every day, you will manage to stay at the top of the leaderboard.

Figure 8.2: Second interface of Alert type 1 (Competition Seeking) MA introducing the leader board

In order to avoid messages too long, we decided to split the communications along more screens. Figure 8.2 represents the motivational assistant introducing the leader board feature.

The expert explains that it’s always better to have short texts. He appreciated the direct tone of the messages.


Welcome!

This is how the leaderboard appears:

Clasificación

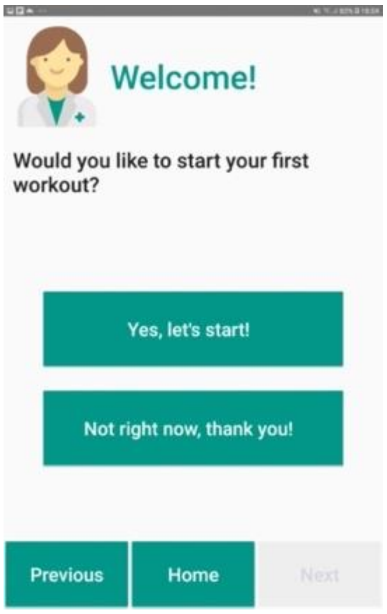
	Luca	100,00%
	Joaquin	69,82%
	Juan	60,35%
4	Alvaro	50,72%
5	Sandra	18,47%
6	Elena	15,42%

Figure 8.3: Third interface of Alert type 1 (Competition Seeking) MA showing how the leader board looks like

In this screen (Figure 8.3), the motivational assistant shows how the leader board looks like.

The expert raised concerns about the size of the image. Also, he is not sure whether the % format can be understood by elderly people (given their backgrounds). In any case, he recommends checking it during the usability testing phase.

Finally, he suggests replacing the word 'appears' with 'looks like'.


Welcome!

Would you like to start your first workout?

Figure 8.4: Fourth interface of Alert type 1, MA asking to start the first exercise session

After the explanation, the motivational assistant asks the user whether they want to start their first exercise session (Figure 8.4).

As highlighted before, the expert thinks the text should be placed in the center.

The only suggestion is to remove the 'thank you!' in the second button.

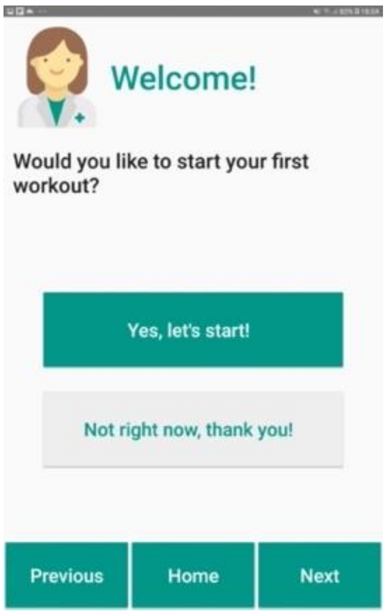

Welcome!
Would you like to start your first workout?
Yes, let's start!
Not right now, thank you!
Previous
Home
Next

Figure 8.5: Fourth interface of Alert type 1 after user interaction

The screen, after having selected an option, changes as depicted in Figure 8.5. The chosen button is highlighted and the other one is faded using a grey color. The text of the unselected button changes color, becoming green. Moreover, the "Next" button changes its color from grey to green, which draws the attention of the patient to using this option.

The expert identified these modifications as understandable by the user. The feedback is strong enough to be noticed. Anyway, he explains that depending on the users, the double confirmation of the selection could be appreciated or not.

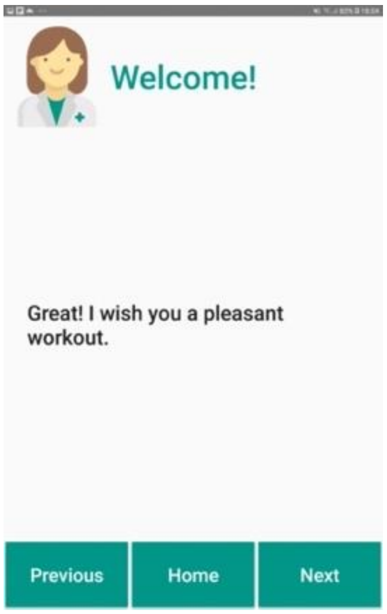
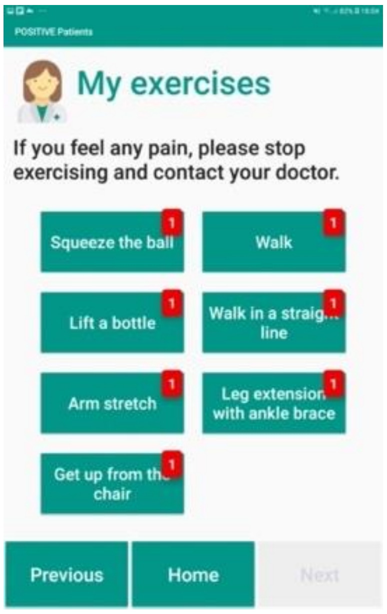


Figure 8.6: Fifth interface of Alert type 1, MA wishing a pleasant workout

Before being redirected to the exercise session, the motivational assistant sends a kind message to the user (Figure 8.6).

The expert, again, recommends putting the text horizontally-centered.



If you feel any pain, please stop exercising and contact your doctor.

Figure 8.7: My exercise menu

The My exercises menu screen (Figure 8.7) is displayed after the user has selected the option to start an exercise session and confirmed it with the "Next" button.

The expert explains that the notification boxes (red boxes with a number) are cognitively demanding for older adults. In addition, in some cases, they hide the text.

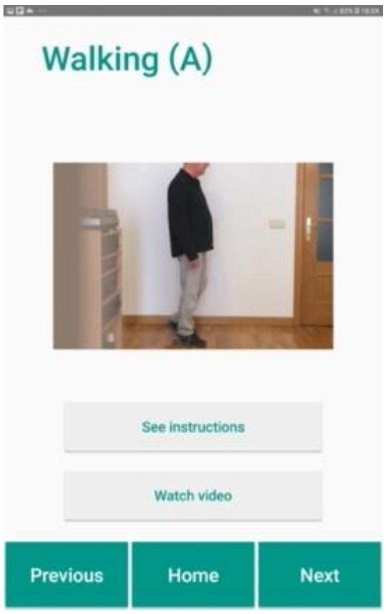
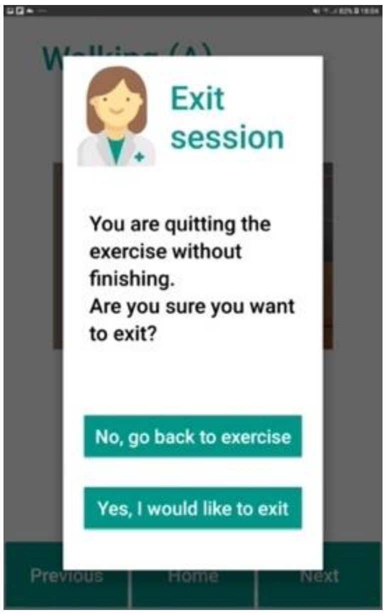


Figure 8.8: Exercise interface

The screen presented in Figure 8.8 comes from the original application for patients developed during POSITIVE. This application screen was not modified for the sake of this investigation. Nonetheless, it was included in the evaluated scenario since it is part of the predefined path to solve the user's task.

According to the subject matter expert, the screen with exercise instructions is not consistent with other parts of the application. The evaluator stated that the buttons ('See instructions' and 'Watch video') have smaller size and opposite colors. They should be modified to ensure a consistent user interface and decrease the patient's cognitive load.



You are quitting the exercise without finishing.
Are you sure you want to exit?

Figure 8.9: Pop-up window to exit an exercise session

When the user is in the middle of an exercise session, they have the option to exit it, by pressing the "Home" button. By doing so, a pop-up appears (Figure 8.9), asking for a confirmation.

The expert explains that usually pop-ups are not recommended for older adults. Nevertheless, he stated that in this case it fits with the double confirmation function. In addition, the pop-up window does cover basically the entire screen. For this reason, it does not increase the cognitive load.

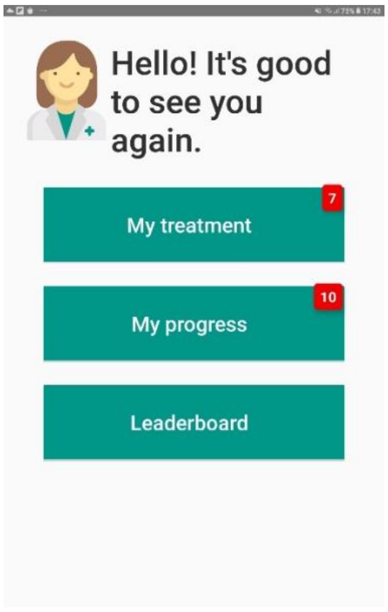
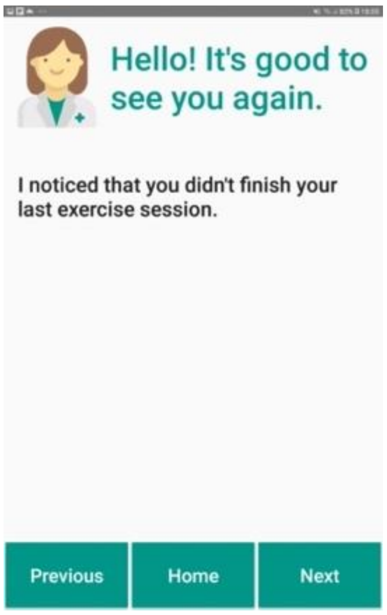


Figure 8.10: Homepage of the app

The screen presented in Figure 8.10 comes from the original application for patients developed during POSITIVE. The user comes back here every time the "Home" button is pressed.


Hello! It's good to see you again.

I noticed that you didn't finish your last exercise session.

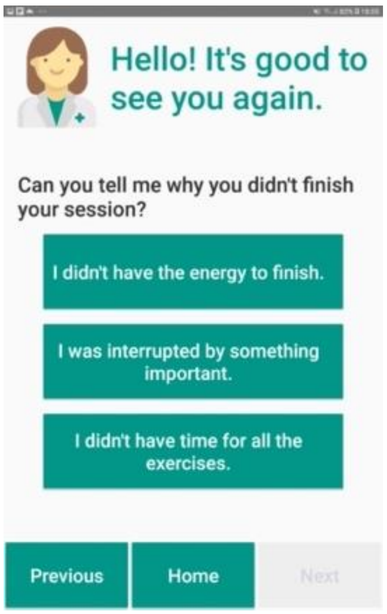
Figure 8.11: First interface of Alert type 2

Once the user selected the “Yes, I would like to exit” button (Figure 8.9), the system generates a new alert, that will be shown to the user at the following restart of the application.

Figure 8.11 presents the first screen of the alert type 2.

The expert suggests minimizing the title in green, by removing “It’s good to see you again”. He proposes to move that text to the body text below.

Regarding that text, he remarks to put the text vertically and horizontally centered.


Hello! It's good to see you again.

Can you tell me why you didn't finish your session?

I didn't have the energy to finish.
I was interrupted by something important.
I didn't have time for all the exercises.
Previous
Home
Next

Figure 8.12: Second interface of the Alert type 2, MA asking information about the interruption

When the user receives this alert, they are asked to answer a question about the reasons for interrupting the last exercise session (Figure 8.12).

The expert raises concerns about the length of text in buttons. He suggests being more concise and focusing on keywords. For example, instead of “I didn’t have the energy to finish”, a simple “Lack of energy”. Moreover, instead of “I was interrupted by something important”, just “Interrupted by something”. Finally, instead of “I didn’t have time for all the exercises”, replace it with “Lack of time”.

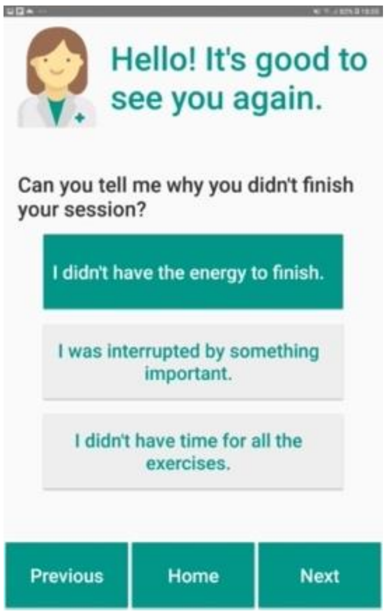

Hello! It's good to see you again.
Can you tell me why you didn't finish your session?
I didn't have the energy to finish.
I was interrupted by something important.
I didn't have time for all the exercises.
Previous
Home
Next

Figure 8.13: Second interface of Alert type 2 after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected buttons become grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.13).

The expert does not add further comments.

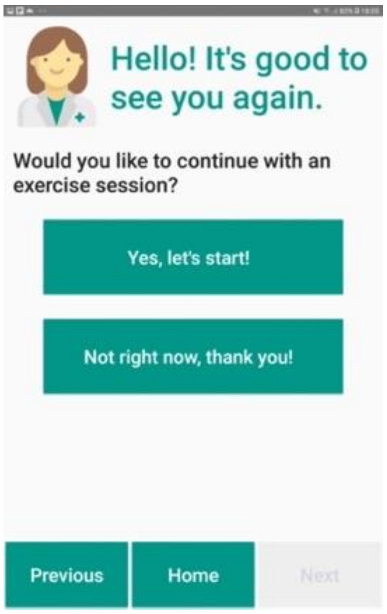

Hello! It's good to see you again.
Would you like to continue with an exercise session?
Yes, let's start!
Not right now, thank you!
Previous
Home
Next

Figure 8.14: Third interface of Alert type 2, MA asking to continue with an exercise session

As it happens in the alert type 1, also here the motivational assistant proposes an exercise session to the user. In this case, the message is slightly different. It says “continue” because the user interrupted an exercise session (Figure 8.14).

The expert explains he is not convinced that the title is explanatory enough.

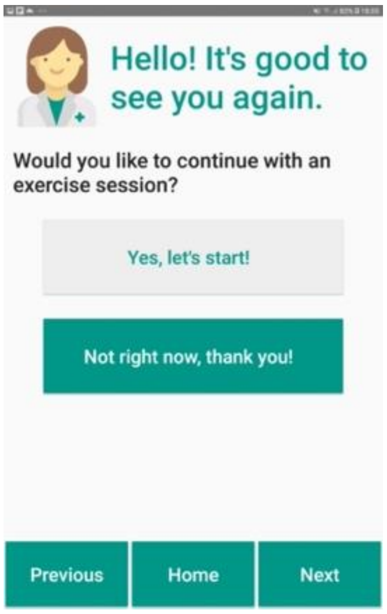

Hello! It's good to see you again.
Would you like to continue with an exercise session?
Yes, let's start!
Not right now, thank you!
Previous
Home
Next

Figure 8.15: Third interface of Alert type 2 after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected buttons become grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.15).

The expert does not add further comments.

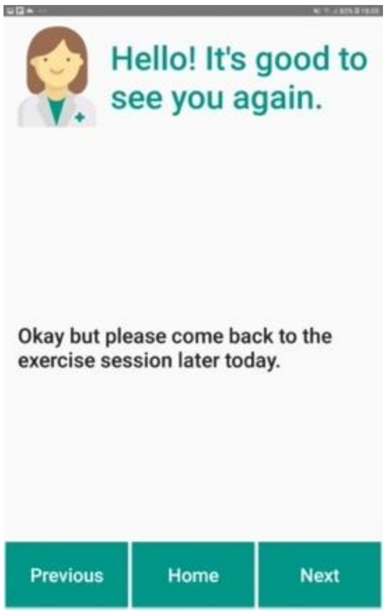

Hello! It's good to see you again.
Okay but please come back to the exercise session later today.
Previous
Home
Next

Figure 8.16: Fourth interface of Alert type 2

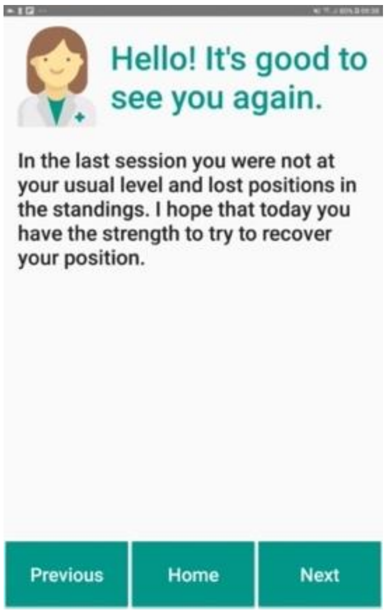
If the user decides not to continue with an exercise session, the motivational assistant reminds them to come back later (Figure 8.16).

The expert considers this message effective. The motivational assistant addresses the patient in a polite and friendly manner.

8.2. Changes in the leader board

Number	2

Scenario	Changes in the leader board
Description	The user is a 70-year-old patient who was prescribed with a Vivifrail exercise programme. They have been associated with the motivational trait *Competition Seeking*. They have been using the mobile application for a few weeks. They receive an alert from the motivational assistant related to changes in the leader board.


Hello! It's good to see you again.

In the last session you were not at your usual level and lost positions in the standings. I hope that today you have the strength to try to recover your position.

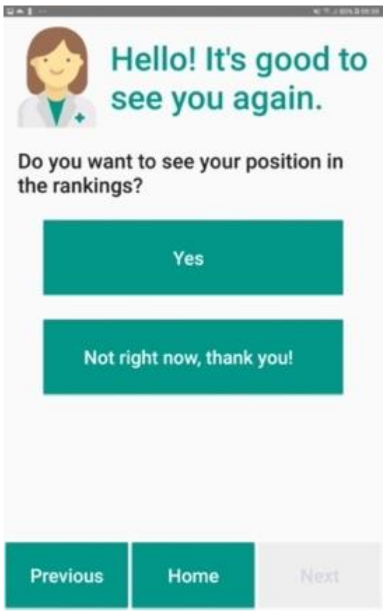
Figure 8.17: First interface of Alert type 6, MA explaining the changes in the leader board

Figure 8.17 represents the first screen of the Alert type 6 (*Changes in the leader board*). The motivation assistant first explains the changes in the ranking. Then, it gives the user a motivational message to cheer them up.

According to the expert, the quantity of the text is copious. He suggests splitting the message in more paragraphs and centering it properly.

Another option is splitting the message into more screens.

In addition, he proposes to change the title, by removing the "It's good to see you again!"

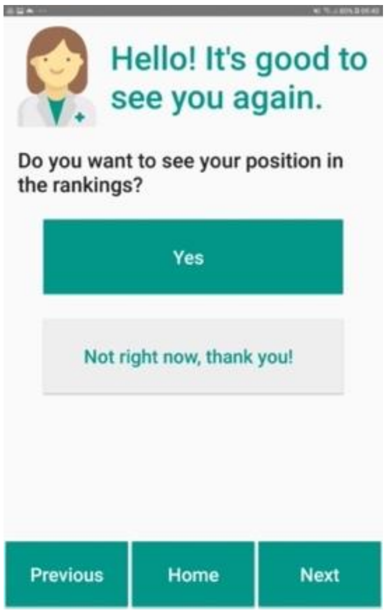

Hello! It's good to see you again.

Do you want to see your position in the rankings?

Figure 8.18: Second interface of Alert type 6, MA asking if the user wants t osee the leader board

Once the user is aware of the change in the standing, the motivational assistant checks whether the user wants to see the leader board (Figure 8.18).

The expert believes that a more concise text could be more effective. He suggests putting simple "Yes" and "No" buttons, without further text.

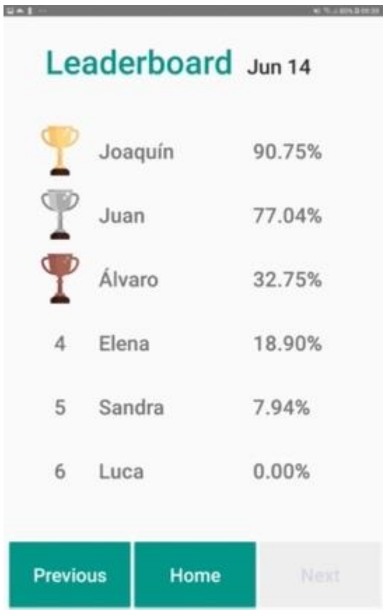


Do you want to see your position in the rankings?

Figure 8.19: Second interface of Alert type 6, after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected button becomes grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.19).

The expert does not add further comments.



	Joaquín	90.75%
	Juan	77.04%
	Álvaro	32.75%
4	Elena	18.90%
5	Sandra	7.94%
6	Luca	0.00%

Figure 8.20: Daily Leader board interface

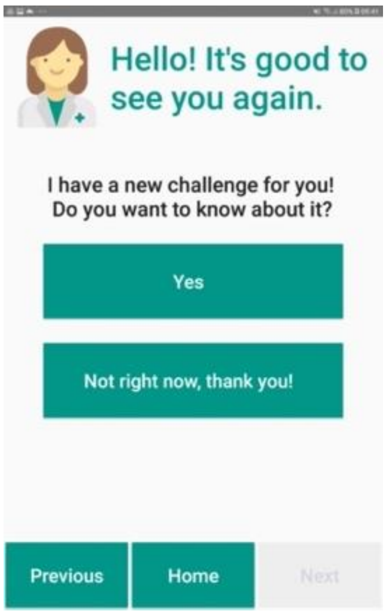
The user interface changes and displays the Daily Leader board (Figure 8.20).

According to the expert, the overall size is good, and the icons are explanatory enough. He wonders whether the % might be understood or not by older adults. In addition, he thinks that the date of yesterday could lead the patient to think that it is today's leaderboard, thus leading to confusion. He proposes further investigations and solutions, such as, instead of the date, writing "Yesterday".

8.3. New challenge and achievement of it

Number	3

Scenario	New challenge and achievement of it
Description	The user is a 70-year-old patient who was prescribed with a Vivifrail exercise programme. They are robust and in an overall good health condition. They have been associated with the motivational trait *Personal Mastery*. They started the programme some days ago. It's time for a new challenge.

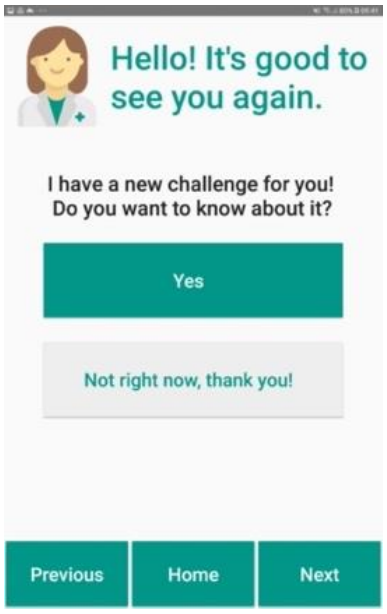

Hello! It's good to see you again.

I have a new challenge for you! Do you want to know about it?

Figure 8.21: First interface of Alert type 4, MA saying it has a new challenge for the user

As explained above, a patient with *High* profile of Personal Mastery has the Challenges feature available. When the system has to propose a new challenge, the motivational assistant sends an alert to the user. Figure 8.21 shows the first screen of the dialogue.

The expert suggests minimizing the title, removing "It's good to see you again!". The text is understandable, and the wording is familiar.

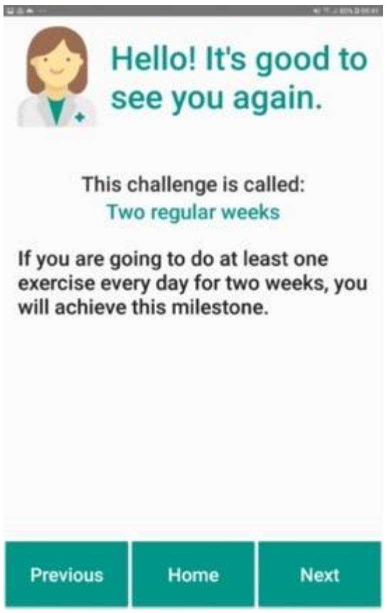

Hello! It's good to see you again.

I have a new challenge for you! Do you want to know about it?

Figure 8.22: First interface of Alert type 4, after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected button becomes grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.22).

The expert does not add further comments.


Hello! It's good to see you again.

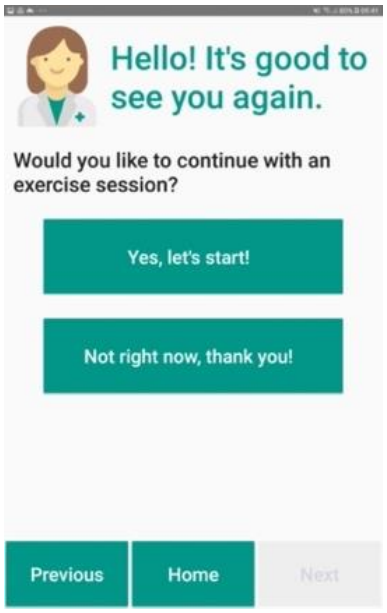
This challenge is called:
Two regular weeks

If you are going to do at least one exercise every day for two weeks, you will achieve this milestone.

Figure 8.23: Second interface of Alert type 4, MA explaining the new challenge

In order to explain the new challenge, the following interface is shown to the patient (Figure 8.23).

The expert points out the following criticism: the title of the challenge should be in black, maybe in italic (*Two regular weeks*). In addition, he suggests rephrasing the description below. Instead of "*If you are going to do at least one exercise every day for two weeks, you will achieve this milestone*", he proposes "*You will achieve this award if you exercise every day during two weeks*". Finally, he would place everything more centered.


Hello! It's good to see you again.

Would you like to continue with an exercise session?

Figure 8.24: Third interface of Alert type 4, MA proposing an exercise session

After presenting the challenge, the motivational assistant asks the patient whether they want to start an exercise session (Figure 8.24).

The expert recommends increasing the overall margin.

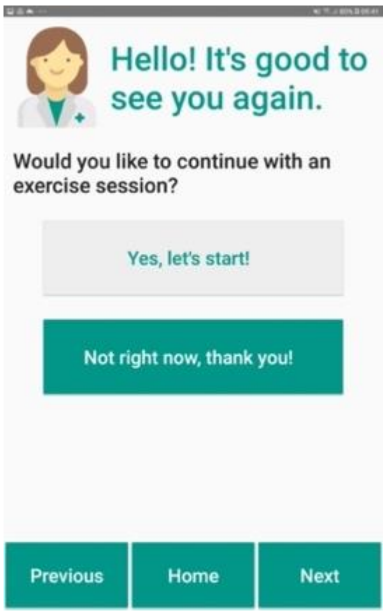

Hello! It's good to see you again.
Would you like to continue with an exercise session?
Yes, let's start!
Not right now, thank you!
Previous
Home
Next

Figure 8.25: Third interface of Alert type 4, after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected button becomes grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.25).

The expert does not add further comments.

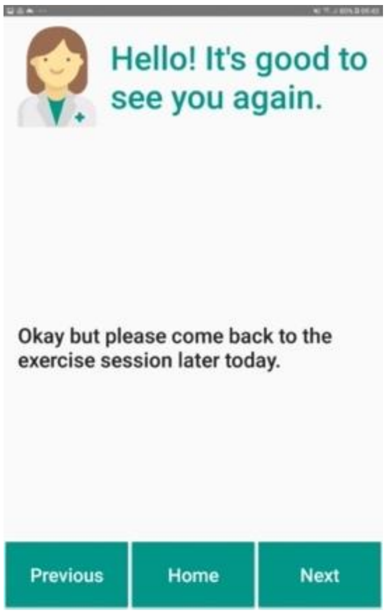

Hello! It's good to see you again.
Okay but please come back to the exercise session later today.
Previous
Home
Next

Figure 8.26: Fourth interface of Alert type 4

As seen above, if the user decides not to continue with an exercise session, the motivational assistant reminds them to come back later (Figure 8.26).

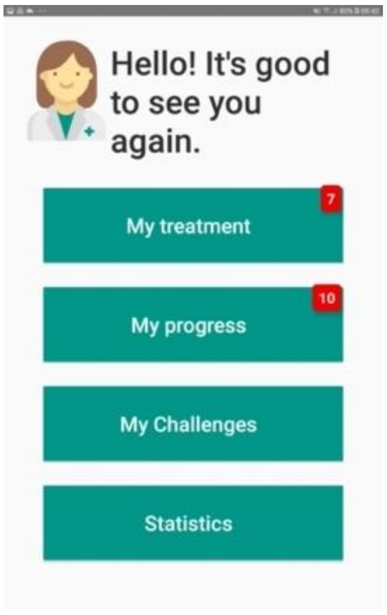


Figure 8.27: Homepage of the application

Figure 8.27 represents the homepage of the application. The buttons "My challenges" and "Statistics" have been added to allow the user to reach the sections directly from here.

According to the expert, the possibility to have a direct link is good as long as these are main functionalities. The title's color should be set to green, to be consistent with the whole system. In addition, the word "Challenges" should be in lowercase.

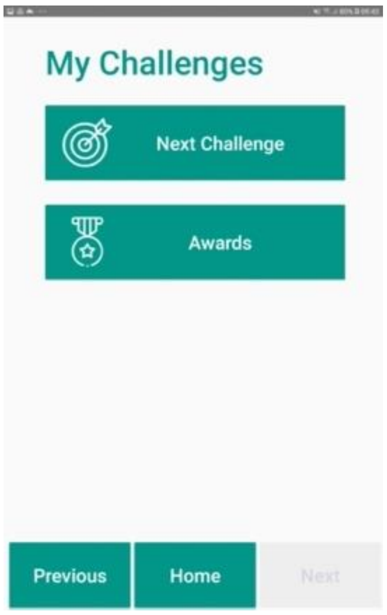


Figure 8.28: My challenge menu

When the user presses "My challenges", the interface changes and the menu is displayed (Figure 8.28).

The expert remarks to change "Challenge" to lowercase.

The icons are understandable and close to the meaning.

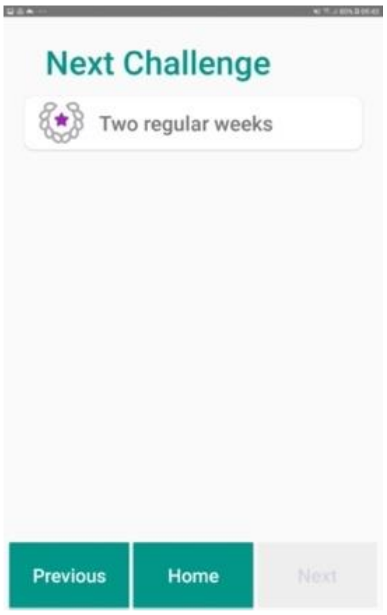
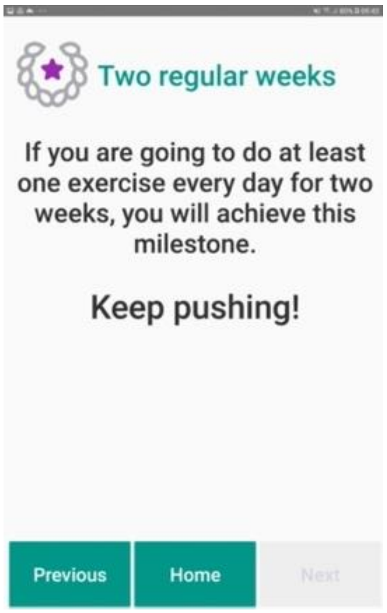


Figure 8.29: Next challenge interface

By clicking the "Next challenge" button, the interface with the pending challenge is shown (Figure 8.29).

According to the expert, the button does not seem as such. It is not consistent with the style used in the rest of the application. For this reason, he suggests replacing it with a basic button.

In addition, he looks doubtable about the icon. In his opinion, the meaning of the icon is not clear at first glance.



If you are going to do at least one exercise every day for two weeks, you will achieve this milestone.

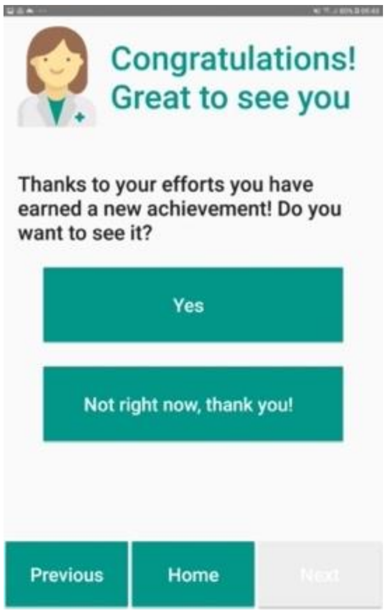
Keep pushing!

Figure 8.30: Specific challenge description interface

The expert suggests rephrasing the description provided in Figure 8.30. Instead of *"If you are going to do at least one exercise every day for two weeks, you will achieve this milestone"*, he proposes *"You will achieve this award if you exercise every day during two weeks"*.

He appreciates the motivational message below, even if he thinks the font size should be the same as the description.

Finally, he would place everything more centered.



Congratulations! Great to see you

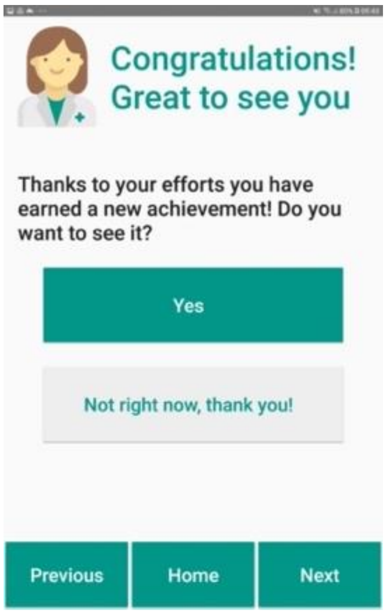
Thanks to your efforts you have earned a new achievement! Do you want to see it?

Figure 8.31: First interface of Alert type 5, MA congratulating to the user

When the user wins the challenge, the motivational assistant congratulates them. Also, it asks the user if they want to see which challenge has been won (Figure 8.31).

The expert suggests removing "Great to see you" from the title. In addition, the "Not right now, thank you!" could be more concise, by removing "thank you!".

Also, he proposes splitting the text after the first sentence. In addition, to be consistent, instead of using the word "achievement", it would be better to use "award".



Congratulations! Great to see you

Thanks to your efforts you have earned a new achievement! Do you want to see it?

Figure 8.32: First interface of Alert type 5, after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected button becomes grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.32).

The expert does not add further comments.

Maybe too many colors in the same screen

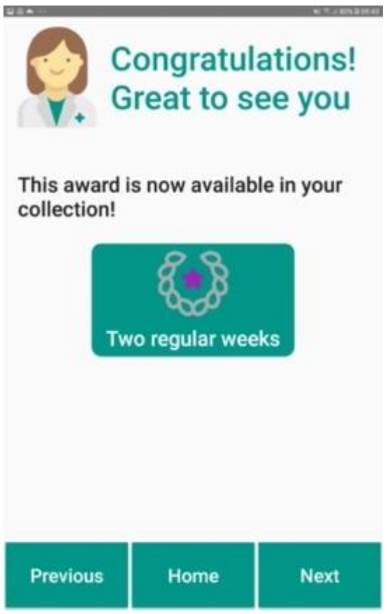


Figure 8.33: Second interface of Alert type 5, MA showing the won challenge

If the user allows the motivational assistant to show them the won challenge, the interface represented by Figure 8.33 appears.

According to the expert, the card seems to be an interactive button. The suggestion is removing the green background.

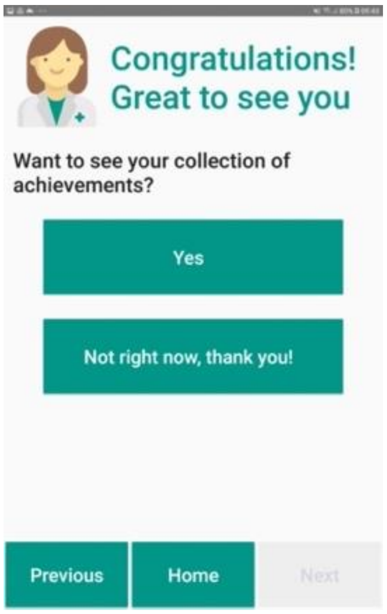

Congratulations!
Great to see you
Want to see your collection of achievements?
Yes
Not right now, thank you!
Previous
Home
Next

Figure 8.34: Third interface of Alert type 5, MA asking to the user if they want to see the award collection

Since the user has earned a new award, the motivational assistant asks them if they want to see the award's collection (Figure 8.34).

The expert reports that unlike the previous sentences, this one does not use "Do you". He suggests adding it.

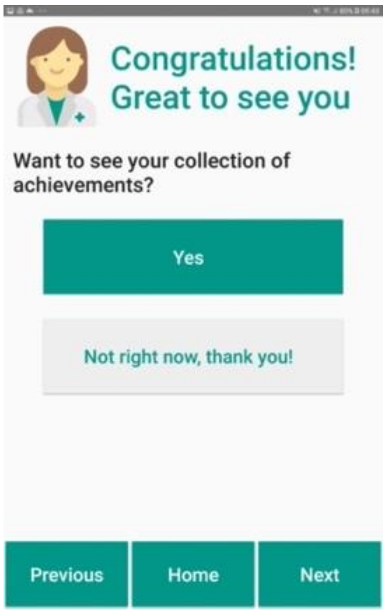

Congratulations!
Great to see you
Want to see your collection of achievements?
Yes
Not right now, thank you!
Previous
Home
Next

Figure 8.35: Third interface of Alert type 5, after user interaction

As explained above, when the user interacts with the system, by choosing among multiple options, the interface adapts itself. The unselected button becomes grey with green text. The next button becomes green, so as to mean that it is now enabled (Figure 8.35).

The expert does not add further comments.

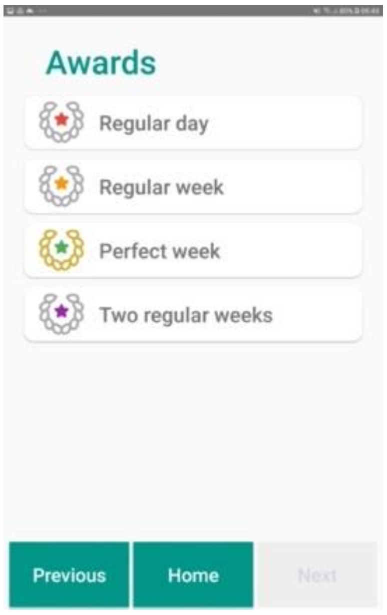
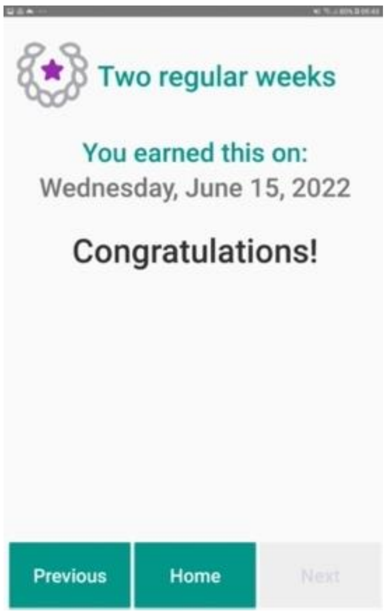


Figure 8.36: Award collection interface

Figure 8.36 represents the award collection. This interface displays all the awards earned so far. By clicking on them a new interface appears.

According to the expert, they do not seem buttons. The basic user would not understand them as such. For this reason, he proposes making them consistent with all the buttons in the application.

In addition, he is skeptical about the icons. He does not clearly see the association between the icon and the challenge.


Two regular weeks

You earned this on:
Wednesday, June 15, 2022

Congratulations!

Figure 8.37: Specific award interface

Once one of the buttons is pressed, the respective interface is shown (Figure 8.37).

The expert reports the following issues: the text “You earned this on” should not be in green. The date should not be in grey. All the text which is not title should be in black. In addition, the motivational message is bigger. He suggests adapting all the text and putting it vertically centered.

9 Conclusion and Future Work

9.1. Conclusion

The work presented in this document is part of a research line addressed by the ActiveUP project, a national initiative created to prevent frailty in older adults. The pilot results of the previous POSITIVE project highlighted that the proposed solution had a low rate of adherence to the exercise plan. According to [14], one way to solve this problem relies on the cooperation between motivation and technology.

The presented tool represents an innovative way of providing support to the elderly, by involving motivation. It adapts itself according to the user profile by giving customized messages and featuring ad-hoc functionalities. This way, every patient can receive tailor-made motivational support.

The following objectives were achieved:

1. **Migration to the new server in Python** – all the required functionalities are now implemented and available in the new server.
2. **Migration to the new android application** – the Vivifrail package is now fully integrated into the new version of the application for patients.
3. **Refactoring of the motivational assistant component done so far** – the issues identified in Section 4.1 are now fixed.
4. **Design of new features** – Leader board, Challenges, User statistics and Alerts were designed, prototyped, evaluated by an expert and improved.
5. **Implementing the motivational assistant component** – the features proposed by Sokól [3] were implemented. In particular, the component features now the Leader board, the Challenges, the User statistics, and the Alerts mechanism.

In June 2022, the Cognitive Walkthrough usability evaluation has been performed. Due to time constraints, it has not been possible to make all the changes suggested during the Cognitive Walkthrough by the expert.

The effort required for the migration to the new server and the new version of the mobile application moved the testing out of the scope of the current thesis. In addition, other features presented by Sokól [3] have not been further considered.

9.2. Future work

The following tasks were identified as future steps to be performed in the next months:

1. **Further changes to the Python server** - the new Python server is still under development. Some solutions that would optimize the server have already been identified. It is very likely that new changes will have to be made.
2. **Improvements of the Motivational Assistant** – during the Cognitive Walkthrough, the expert pointed at some minor problems and inconsistencies. These will be solved in the next weeks.
3. **Design and development of new features** – addition to the motivational assistant component of new features depicted by Sokól [3]
4. **Testing with real users of the Motivational Assistant** - in the following months, the motivational assistant is expected to be tested with real users. In particular, the Ageing Lab is planning to perform the so-called A/B test. It consists of providing two different versions of the system to determine which of them brings more benefits. A testing phase could raise problems or drawbacks to fix in order to optimize the tool.

Bibliography

[1] World Health Organization, "World report on ageing and health," 2015.

[2] Q.-L. Xue, "The Frailty Syndrome: Definition and Natural History," *Clinics in geriatric medicine, vol. 27, n. 1,* pp. 1-15, 2011.

[3] N. Sokól, "Application of Motivational Traits, Motivational State and Stage of Change Modelling for Frailty Prevention in Older Adults," 2021.

[4] D. F.-A. Pedraza, "Modelado y gestión de la motivación en sistemas computacionales," 2020.

[5] L. Zorzenon, "Design of a Digital Game for Motivational Profile Detection in the Elderly Using a Modular Architecture," 2021.

[6] J. Nielsen, "Usability Engineering," San Diego, 1993.

[7] C. L. John D. Gould, "Communications of the ACM," in *Designing for usability: key principles and what designers think*, 1985, p. 300–311.

[8] W3, "Notes on User Centered Design Process (UCD)," [Online]. Available: https://www.w3.org/WAI/redesign/ucd.

[9] A. Smith, "Older Adults and Technology Use," Pew Research Center, 2014.

[10] M. G. Hernandez, *Heuristics established by a systematic review of guidelines to design mobile technology for older adults,* 2022.

[11] Vivifrail, "Vivifrail project," [Online]. Available: https://vivifrail.com/.

[12] M. Izquierdo, "Prescripción de ejercicio físico. El programa Vivifrail como modelo," *Nutrición Hospitalaria,* p. 50–56, 2019.

[13] POSITIVE, "Maintaining and improving the intrinsic capacity involving primary care and caregivers," [Online]. Available: https://eithealth.eu/project/positive/.

[14] N. E. Miller, "“Literalization of Basic S-R Concepts: Extensions to Conflict Behavior, Modification, and Social Learning," in *Psychology: A Study of a Science*, New York, S. Koch, 1959, p. 196–202.

[15] R. M. W. G. C. P. H. a. D. E. L. Ryan, "Self-determination Theory and Physical Activity: the Dynamics of Motivation in Development and Wellness," Hellenic J. Psychol, 2009, p. 107–124.

[16] T. M. e. a. O'Neil-Pirozzi, The Importance of Motivation to Older Adult Physical and Cognitive Exercise Program Development, Initiation, and Adherence, Frontiers, 2022.

[17] S. F. T. D. a. R. M. A. Rivera-Torres, "Adherence to Exercise Programs in Older Adults: Informative Report. Gerontol," *Geriatr. Med. 5*, 2019.

[18] RabbitMQ, "Documentation: Table of Contents — RabbitMQ.," [Online]. Available: https://www.rabbitmq.com/documentation.html..

[19] cron-job.org, "Free cronjobs - from minutely to once a year.," [Online]. Available: https://cron-job.org/en/..

[20] R. &. H. E. D. Kanfer, "Motivational Traits and Skills: A Person- Centered Approach to Work Motivation”," in *Research in Organizational Behavior*, 1997, p. 1–56.

[21] A. &. P. H. De Vicente, "Informing the Detection of the Students' Motivational State: An Empirical Study”," in *Lecture Notes in Computer Science*, 2002, p. 933–943.

[22] J. O. &. V. W. F. Prochaska, "The Transtheoretical Model of Health Behavior Change," *American Journal of Health Promotion,* p. 38–48., 1997.

[23] S. B. W. R. C. N. &. R. W. A. Czaja, Designing for older adults. Principles and Creative Human Factors Approaches, Boca Raton: CRC Press, 2019.

[24] Möller, Quality Engineering: Qualität kommunikationstechnischer Systeme, Berlin: Springer, 2010.

[25] R. M. G. J. B. W. A. S. &. G. S. Baecker, "Chapter 2: Design and Evaluation," in *Readings in Human-Computer Interaction: Toward the Year 2000*, San Francisco, Morgan Kaufmann Publishers, 1995, p. 73–91.

[26] "https://support.microsoft.com/en-us/word," [Online].

A Appendix A

A.1. SQL queries to retrieve the leader board

getClasificacionDiaria performs the following query:

```
"(SELECT id_paciente, nombre, eval AS eval FROM eval JOIN
paciente ON eval.id_paciente=paciente.id WHERE id_paciente= " +
patient_id + " AND (timestamp>= curdate() OR timestamp>=
DATE_SUB(CURDATE(), INTERVAL 2 DAY)) AND tipo = 1 ORDER BY
timestamp DESC LIMIT 1)" +
"UNION" +
"(SELECT id_paciente, nombre, eval AS eval FROM eval JOIN
paciente ON eval.id_paciente=paciente.id WHERE (id_grupo IN
(SELECT id_grupo as grupo FROM paciente WHERE id = " + patient_id
+ ")) AND (timestamp>= curdate() OR timestamp>=
DATE_SUB(CURDATE(), INTERVAL 2 DAY)) AND tipo = 1 AND
id_paciente !=" + patient_id + " GROUP BY id_paciente)" + "ORDER
BY eval DESC"
```

It returns an object with patient_id, patient_name, and eval.

getClasificacionSemana performs the following query:

```
"(SELECT id_paciente AS id_usuario, nombre, eval AS semanal_eval
FROM eval JOIN paciente ON eval.id_paciente=paciente.id WHERE
id_paciente = " + patient_id + " AND tipo = 2 AND
id_index_actividades = 0 ORDER BY timestamp DESC LIMIT 1)" +
"UNION" +
" (SELECT id_paciente AS id_usuario, nombre, eval AS
semanal_eval FROM eval JOIN paciente ON
eval.id_paciente=paciente.id WHERE (id_grupo IN (SELECT
id_grupo as grupo FROM paciente WHERE id = " + patient_id + "))
AND timestamp>=DATE_SUB(CURDATE(), INTERVAL 6 DAY) AND tipo = 2
AND id_paciente !=" + patient_id + ") " + "ORDER BY semanal_eval
DESC"
```

It returns an object with patient_id, patient_name, and eval.

getClasificacionGlobal performs the following query:

```
"(SELECT id_paciente AS id_usuario, nombre,
SUM(eval)/COUNT(eval) AS global_eval FROM eval JOIN paciente ON
eval.id_paciente=paciente.id WHERE id_paciente = " + patient_id
+ " AND tipo = 1)" +
"UNION" +
"(SELECT   id_paciente,   nombre,   (SUM(eval)/COUNT(eval))   AS
global_eval    FROM    eval    JOIN    paciente    AS    P    ON
eval.id_paciente=P.id WHERE tipo=1 AND (P.id_grupo IN (SELECT
id_grupo FROM paciente AS P WHERE P.id=" + patient_id + "))
GROUP BY id_paciente)" + "ORDER BY global_eval DESC"
```

It returns an object with patient_id, patient_name, and eval.

A.2. Code to compute changes in the leader board

A.2.1. leaderboard.js

The following functions are in charge of check whether the user has gained or lost positions in the leader board. According to the result, an alerto f type 6 is generated.

```
require("dotenv").config();

// Modulo para ejecutar comandos como en el Cron de Linux, pero
desde NodeJS
var CronJob = require("cron").CronJob;

// add timestamps in front of log messages
require("console-stamp")(console, {
  format: ":date(dd/mm/yyyy HH:MM:ss)",
});

var mysql = require("mysql");

//Conexion SQL:
var usuarioDB = process.env.usuarioDB;
var passDB = process.env.passDB; //g3r14tr14 // XvzHjtzDwvtJ7Lcv
var dirDB = process.env.dirDB;
var databaseDB = process.env.databaseDB;
var portDB = process.env.puerto_db;

var db;
```

```
function conectarSQL() {
  db = mysql.createPool({
    connectionLimit: 1000,
    host: dirDB,
    user: usuarioDB,
    password: passDB,
    database: databaseDB,
    port: portDB,
  });
}

function computeCambiosLeaderboard() {
  console.log("computeLeaderBoard");
  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!
    var query = connection.query("SELECT * FROM grupo", function
(error, rows) {
      connection.release();
      if (error) {
        console.log("Error en computar numero de grupos " + error);
        numero = 0;
      } else {
        rows.forEach((resultado) => {
          var group_id = resultado.id;
          clasificacionHoy(group_id);
        });
      }
    });
  });
}

function clasificacionHoy(group_id) {
  console.log("clasificacionHoy for group: " + group_id);
  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!
    //  Generacion de la clasificación
    var query = connection.query(
      "SELECT id_paciente, eval, timestamp FROM eval JOIN paciente
ON eval.id_paciente=paciente.id WHERE tipo=1 AND id_grupo= " +
        group_id +
        " AND timestamp >= DATE_SUB(CURDATE(), INTERVAL 2 DAY) ORDER
BY eval DESC",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("clasificacionHoy: Error al actualizar " +
error);
        } else {
          console.log("clasificacionHoy: Esta: " + "\n" +
JSON.stringify(rows));
          var lista_hoy = [];
          rows.forEach(function (row) {
            lista_hoy.push(row.id_paciente);
          });
          clasificacionAyer(group_id, lista_hoy);
        }
      }
    );
  });
}

function clasificacionAyer(group_id, lista_hoy) {
  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!
    //  Generacion de la clasificación
    var query = connection.query(
      "SELECT id_paciente, eval, timestamp FROM eval JOIN paciente
ON eval.id_paciente=paciente.id WHERE tipo=1 AND id_grupo= " +
        group_id +
        " AND timestamp>= DATE_SUB(CURDATE(), INTERVAL 3 DAY) AND
timestamp< DATE_SUB(CURDATE(), INTERVAL 2 DAY) ORDER BY eval DESC",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("clasificacionAyer: Error al actualizar " +
error);
        } else {
          console.log(
            "clasificacionAyer: Esta: " + "\n" +
JSON.stringify(rows)
          );
          var lista_ayer = [];
          rows.forEach(function (row) {
            lista_ayer.push(row.id_paciente);
          });
          changes(lista_hoy, lista_ayer);
        }
      }
    );
  });
}
function changes(lista_ayer, lista_hoy) {
  lista_ayer.forEach((patient_id) => {
    console.log(
      "Change for user " +
        patient_id +
        " : " +
        lista_ayer.indexOf(patient_id) +
        " - " +
        lista_hoy.indexOf(patient_id)
    );
    var change = lista_ayer.indexOf(patient_id) -
lista_hoy.indexOf(patient_id);
```

```
    console.log("Change value: " + change);
    alertaClasificacion(patient_id, change);
  });
}

function alertaClasificacion(patient_id, change) {
  console.log("alertaClasificacion for user: " + patient_id);
  if (change > 0) {
    db.getConnection(function (err, connection) {
      if (err) throw err; // not connected!
      //Inserta una alerta nueva al eval
      var query = connection.query(
        "INSERT INTO alertas_asistente (id_paciente, id_alerta,
id_contenido, avisada, conclusion) VALUES (" +
          patient_id +
          ", 6, 11, 0, 0)",
        function (error, rows) {
          connection.release();
          if (error) {
            console.log("alertaClasificacion: " + error);
          } else {
            console.log("New alertaClasificacion added");
          }
        }
      );
    });
  } else if (change < 0) {
    db.getConnection(function (err, connection) {
      if (err) throw err; // not connected!
      //Inserta una alerta nueva al eval
      var query = connection.query(
        "INSERT INTO alertas_asistente (id_paciente, id_alerta,
id_contenido, avisada, conclusion) VALUES (" +
          patient_id +
          ", 6, 12, 0, 0)",
        function (error, rows) {
          connection.release();
          if (error) {
            console.log("alertaClasificacion: " + error);
          } else {
            console.log("New alertaClasificacion added");
          }
        }
      );
    });
  }
}

conectarSQL();
computeCambiosLeaderboard();
```

B Appendix B

B.1. Code to handle the Challenges feature

B.1.1. check_status_challenges.js

The following code is in charge of checking the status of progress of a challenge.

```
require("dotenv").config();


// Modulo para ejecutar comandos como en el Cron de Linux, pero
desde NodeJS
var CronJob = require("cron").CronJob;


// add timestamps in front of log messages
require("console-stamp")(console, {
  format: ":date(dd/mm/yyyy HH:MM:ss)",
});


var mysql = require("mysql");


//Conexion SQL:
var usuarioDB = process.env.usuarioDB;
var passDB = process.env.passDB; //g3r14tr14 // XvzHjtzDwvtJ7Lcv
var dirDB = process.env.dirDB;
var databaseDB = process.env.databaseDB;
var portDB = process.env.puerto_db;


var db;


function conectarSQL() {
  db = mysql.createPool({
    connectionLimit: 1000,
    host: dirDB,
    user: usuarioDB,
    password: passDB,
    database: databaseDB,
    port: portDB,
  });
}

function computePatients() {
  console.log("computeWeek");

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT id_paciente FROM perfil_motivacional WHERE id_perfil =
2 AND valor = 2",
      function (error_id, rows_id) {
        connection.release();
        if (error_id) {
          console.log("No obtenemos el ID de los pacientes");
        } else {
          if (rows_id.length > 0) {
            console.log("Tenemos " + rows_id.length + " usuarios");
            rows_id.forEach((resultado) => {
              var patient_id = resultado.id_paciente;
              console.log("Paciente: " + patient_id);
              checkChallengeInProgress(patient_id);
            });
          }
        }
      }
    );
  });
}

function checkChallengeInProgress(patient_id) {
  console.log("checkChallengeInProgress for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT id_reto FROM retos WHERE id_paciente = " +
        patient_id +
        " AND status = '0'",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error while retrieving data " + error);
        } else {
          rows.forEach((resultado) => {
            var challenge_id = resultado.id_reto;
            console.log("Challenge: " + challenge_id);
            switch (challenge_id) {
              case 1:
                checkChallenge1(patient_id, challenge_id);
                break;
              case 2:
                checkChallenge2(patient_id, challenge_id);
                break;
              case 3:
                checkChallenge3(patient_id, challenge_id);
                break;
              case 4:
                checkChallenge4(patient_id, challenge_id);
                break;
              case 5:
                checkChallenge5(patient_id, challenge_id);
                break;
              case 6:
                checkChallenge6(patient_id, challenge_id);
                break;
              case 7:
                checkChallenge7(patient_id, challenge_id);
                break;
            }
          });
        }
      }
    );
  });
}

function checkChallenge1(patient_id, challenge_id) {
  console.log("checkChallenge1 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval > 0 AND tipo = '1' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 5 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if (rows.length > 0) {
```

```
          console.log("Challenge1 won");
          updateWonChallenge(patient_id, challenge_id);
        }
      }
    }
  );
 });
}

function checkChallenge2(patient_id, challenge_id) {
  console.log("checkChallenge2 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval > 0 AND tipo = '1' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 5 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if ((rows.length = 5)) {
            console.log("Challenge2 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}
```

```
function checkChallenge3(patient_id, challenge_id) {
  console.log("checkChallenge3 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval = 1 AND tipo = '2' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 5 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if (rows.length > 0) {
            console.log("Challenge3 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}

function checkChallenge4(patient_id, challenge_id) {
  console.log("checkChallenge4 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval > 0 AND tipo = '1' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 12 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if (rows.length > 10) {
            console.log("Challenge4 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}

function checkChallenge5(patient_id, challenge_id) {
  console.log("checkChallenge5 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval = '1' AND tipo = '2' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 12 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if ((rows.length = 2)) {
            console.log("Challenge5 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}

function checkChallenge6(patient_id, challenge_id) {
  console.log("checkChallenge6 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval = 1 AND tipo = '2' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 26 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if ((rows.length = 4)) {
            console.log("Challenge6 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}

function checkChallenge7(patient_id, challenge_id) {
  console.log("checkChallenge7 for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT * FROM eval WHERE id_paciente = " +
        patient_id +
        " AND eval = '1' AND tipo = '2' AND
timestamp>=DATE_SUB(CURDATE(), INTERVAL 5 DAY) AND
timestamp<CURDATE()",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          if ((rows.length = 1)) {
            console.log("Challenge7 won");
            updateWonChallenge(patient_id, challenge_id);
          }
        }
      }
    );
  });
}

function updateWonChallenge(patient_id, challenge_id) {
  console.log(
    "updateWonChallenge : " + challenge_id + " for user: " +
patient_id
```

```
  );

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "UPDATE retos SET status = '1' WHERE id_paciente = " +
        patient_id +
        " AND id_reto = " +
        challenge_id +
        "",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          notifyWonChallenge(patient_id, challenge_id);
        }
      }
    );
  });
}

function notifyWonChallenge(patient_id, challenge_id) {
  console.log(
    "notifyWonChallenge to user: " +
      patient_id +
      ", challengeId: " +
      challenge_id
  );

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!
    var content_id;
    switch (challenge_id) {
      case 1:
        content_id = 4;
        break;
      case 2:
        content_id = 5;
        break;
      case 3:
        content_id = 6;
        break;
      case 4:
        content_id = 7;
        break;
      case 5:
        content_id = 8;
        break;
      case 6:
        content_id = 9;
        break;
      case 7:
        content_id = 10;
        break;
    }

    var query = connection.query(
      "INSERT INTO alertas_asistente (id_alerta, id_paciente,
id_contenido, avisada, conclusion) VALUES ('5', " +
        patient_id +
        ", " +
        content_id +
        ", '0', '0')",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error al insertar alerta " + error);
        } else {
          console.log("Alerta insertada.");
          if (challenge_id != 7) {
            var challenge = challenge_id;
            var new_challenge_id = challenge + 1;
            updateNewChallenge(patient_id, new_challenge_id);
          }
        }
      }
    );
  });
}

function updateNewChallenge(patient_id, challenge_id) {
  console.log(
    "updateNewChallenge: " + challenge_id + " for user: " +
patient_id
  );

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "UPDATE retos SET status = '0' WHERE id_paciente = " +
        patient_id +
        " AND id_reto = " +
        challenge_id +
        "",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error");
        } else {
          console.log("Alert updated correctly!");
        }
      }
    );
  });
}

conectarSQL();
computePatients();
```

B.1.2. manejador_challenges_to_be_notified.js

The following code manages the queue of the challenges. In particular, it checks whether a challenge is in progress or not. If not, it changes the status of the next challenge to 'to be notified'.

```
require("dotenv").config();

// Modulo para ejecutar comandos como en el Cron de Linux, pero
desde NodeJS
var CronJob = require("cron").CronJob;

// add timestamps in front of log messages
require("console-stamp")(console, {
  format: ":date(dd/mm/yyyy HH:MM:ss)",
});

var mysql = require("mysql");

//Conexion SQL:
var usuarioDB = process.env.usuarioDB;
var passDB = process.env.passDB; //g3r14tr14 // XvzHjtzDwvtJ7Lcv
var dirDB = process.env.dirDB;
var databaseDB = process.env.databaseDB;
var portDB = process.env.puerto_db;
var db;

function conectarSQL() {
  db = mysql.createPool({
    connectionLimit: 1000,
    host: dirDB,
    user: usuarioDB,
    password: passDB,
    database: databaseDB,
    port: portDB,
  });
}

function computePatients() {
  console.log("computeWeek");

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "SELECT id_paciente FROM perfil_motivacional WHERE id_perfil =
2 AND valor = 2",
      function (error_id, rows_id) {
        connection.release();
        if (error_id) {
          console.log("No obtenemos el ID de los pacientes");
        } else {
          if (rows_id.length > 0) {
            console.log("Tenemos " + rows_id.length + " usuarios");
            rows_id.forEach((resultado) => {
              var patient_id = resultado.id_paciente;
              console.log("Paciente: " + patient_id);
              checkNewChallenge(patient_id);
            });
          }
        }
      }
    );
  });
}

function checkNewChallenge(patient_id) {
  console.log("checkNewChallenge for user: " + patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query_challenge_to_be_notified =
      "SELECT * FROM retos WHERE id_paciente = " +
      patient_id +
      " AND status = -1";

    var query = connection.query(
      query_challenge_to_be_notified,
      function (error_id, rows_id) {
        connection.release();
        if (error_id) {
          console.log("Error in retrieving data");
        } else {
          if (rows_id.length > 0) {
            console.log("Tenemos " + rows_id.length + " usuarios");
            rows_id.forEach((resultado) => {
              var challenge_id = resultado.id_reto;
              console.log("Challenge: " + challenge_id);
              notifyNewChallenge(patient_id, challenge_id);
            });
          }
        }
      }
    );
  });
}

function notifyNewChallenge(patient_id, challenge_id) {
  console.log(
    "notifyNewChallenge to user: " +
      patient_id +
      ", challengeId: " +
      challenge_id
  );

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var content_id;
    switch (challenge_id) {
      case 1:
        content_id = 4;
        break;
      case 2:
        content_id = 5;
        break;
      case 3:
        content_id = 6;
        break;
      case 4:
        content_id = 7;
        break;
      case 5:
        content_id = 8;
        break;
      case 6:
        content_id = 9;
        break;
      case 7:
        content_id = 10;
        break;
    }

    var query = connection.query(
      "INSERT INTO alertas_asistente (id_alerta, id_paciente,
id_contenido, avisada, conclusion) VALUES ('4', " +
        patient_id +
        ", " +
        content_id +
        ", '0', '0')",
      function (error, rows) {
        connection.release();
        if (error) {
          console.log("Error al insertar alerta " + error);
        } else {
          console.log("Alerta insertada.");
          updateChallenge(patient_id, challenge_id);
        }
      }
    );
  });
}

function updateChallenge(patient_id, challenge_id) {
  console.log("updateChallenge: " + challenge_id + " for user: " +
patient_id);

  db.getConnection(function (err, connection) {
    if (err) throw err; // not connected!

    var query = connection.query(
      "UPDATE retos SET status = '0' WHERE id_reto = " +
        challenge_id +
        " AND id_paciente = " +
        patient_id +
        "",
      function (error, rows) {
        if (error) {
          console.log("Error al actualizar alerta " + error);
        } else {
          console.log("Alerta actualizada.");
        }
      }
    );
  });
}

conectarSQL();
computePatients();
```

List of Figures

List of Tables

Acknowledgments

Eccomi qui, giunto alla fine di un percorso iniziato nel settembre 2020, un periodo che difficilmente ci scorderemo. Questi due anni mi hanno permesso di crescere sia dal punto di vista accademico sia dal punto di vista professionale, ma soprattutto come persona. Come i miei amici sanno, la concisione non è la caratteristica che più mi distingue. Tuttavia, ritengo giusto ringraziare le persone che mi hanno accompagnato e supportato lungo questo percorso.

Innanzitutto, un ringraziamento alla mia relatrice, la Professoressa Franca Garzotto, e a Federico Schiepatti.

Grazie a mio papà Massimo e mio fratello Marco. Papà, sei la mia roccia, il mio supereroe, la mia ispirazione. Spero di averti reso fiero di me, di continuare a farlo, e di diventare un grande uomo come te. Marco, sei il miglior fratello che potessi desiderare. Nonostante le litigate quotidiane, so che ci sarai per sempre. Vi voglio bene.

Giuditta, la mia migliore amica da che io possa ricordare. Grazie per essere sempre al mio fianco, per capirmi anche senza troppe parole, e per supportarmi soprattutto nei momenti difficili.

Sara, la mia partner in crime. Grazie per il tuo supporto costante, i tuoi preziosi consigli, e per le avventure incredibili che mi hai regato, specialmente nell’ultimo anno a Madrid.

Jacopo, l’amico da una vita. Grazie per le chiamate di sfogo, i tuoi messaggi motivazionali, e per spingermi sempre a pensare positivo.

Giulia, la mia dog buddy. Grazie perché mi fai sempre sorridere e divertire, per le mattinate a passeggiare e per i pomeriggi passati a studiare.

Giorgio, il mio compagno di Formula 1. Grazie per le domeniche di gara, per ascoltarmi sempre e darmi consigli.

Chiara, Filippo, Martina, Sofia. Grazie per la vostra preziosa amicizia e per i momenti che mi regalate.

Grazie a Andrea, Greta, Luca, Monica, e Silvia, i miei compagni di Milano, che avete arricchito questi anni di momenti meravigliosi e indimenticabili.

Infine, ringrazio Abel, Dani, Elena, Jose, Manu, e Nate. Sono grato di avervi incontrato e spero di rivedervi spesso in futuro.

E a voi, mamma e nonna Fausta. Senza di voi, non sarei diventato nemmeno un quarto della persona che sono ora. Non passa un giorno che non pensi a voi.