FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Development of a multimodal management platform for patients in physical rehabilitation

Tiago Luís Salgueiro dos Santos

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Bruno Giesteira

Co-Supervisor: Pedro Cardoso

July 23, 2021

Development of a multimodal management platform for patients in physical rehabilitation

Tiago Luís Salgueiro dos Santos

Mestrado Integrado em Engenharia Informática e Computação

July 23, 2021

Abstract

Increasing mobile apps accessibility is paramount to reaching more users, namely people with special needs. Assistive technologies enable users to overcome obstacles when interacting with smartphones. Multimodal input/output combinations increase the options offered to users, reducing obstacles present in single-modality applications.

In order to aid clients with special needs at Centro Profissional de Reabilitação de Gaia, a mobile application was designed as a platform to manage their rehabilitation process. This project's first goal was to implement a mobile app helpful to clients' daily lives. The second goal was to determine whether and how multiple modalities of interaction increase accessibility and inclusiveness.

The design phase of this project was performed by students of the Specialization in Interaction Design, Web and Games of the Faculty of Fine Arts of the University of Porto, of which a non-functional prototype resulted. The conclusions and requirements gathered in this phase were used as a starting point for the development of our application. Then, a User-Driven Development method was adopted, with end-users providing continuous feedback. Usability tests with CRPG clients were conducted to test our hypotheses.

The application features several integrated modalities of interaction and features related to daily tasks the clients perform, such as selecting meals, consulting transport schedules or meditating. Users can interact with the application by using the touchscreen or voice commands, and audio hints are also provided. Contextual notifications remind users of events and provide shortcuts to specific features of the app. The available interaction modalities were well received by the users. Testing suggested that the multiple modalities are adequate in allowing participants to fulfill their tasks. Younger participants, more experienced in smartphone usage, performed better with the touchscreen. Older participants, who reported less self-perceived dexterity, seemed to benefit from voice commands and audio hints. Some tasks were executed faster and with less requests for help when compared to touchscreen use.

Including users in the development process allowed the application to meet the needs of the CRPG community with more efficacy, addressing issues that would not have been identified otherwise. The test results suggest the viability and efficacy of multiple interaction modalities in improving the accessibility of the application, ensuring a broader reach to the CRPG community. The study was limited by the small and homogeneous group. The five participants used their smartphones daily and did not use other assistive technologies, so everyone had the dexterity needed to interact without external assistance. In the future, other modalities of interaction can be integrated to ensure even more clients at the CRPG can use the application.

Keywords: Accessibility, Mobile Health, Mobile Development, Rehabilitation, Assistive Technologies

Resumo

Aumentar a acessibilidade de dispositivos e aplicações móveis é fundamental para aumentar o seu alcance a um maior número de utilizadores, nomeadamente pessoas com necessidades especiais. As tecnologias assistivas permitem ao utilizador superar obstáculos na interação com estes dispositivos. Combinações multimodais de input/output aumentam as opções disponíveis ao utilizador, reduzindo os obstáculos que podem ser encontrados nas aplicações monomodais.

Para ajudar clientes com necessidades especiais do Centro Profissional de Reabilitação de Gaia, foi desenhada uma aplicação móvel como plataforma de gestão do processo de reabilitação dos clientes. O primeiro objetivo desta dissertação foi implementar a aplicação útil para o dia a dia dos clientes. O segundo objetivo era determinar se e como as múltiplas modalidades de interação aumentam a inclusividade e acessibilidade.

A fase de concepção e design do projecto foi realizada por alunos da especialização em Design de Interacção, Web e Jogos da Faculdade de Belas Artes da Universidade do Porto, tendo culminado com a criação de um protótipo não funcional. As conclusões e requisitos recolhidos nesta fase foram utilizados como ponto de partida para o desenvolvimento da aplicação. Durante o processo de desenvolvimento, foi adotado o método User-Driven Development, com feedback contínuo por parte dos utilizadores finais. A validação do protótipo funcional pelos utilizadores ocorreu durante a fase de desenvolvimento, permitindo a identificação de obstáculos e possíveis melhorias. Testes de usabilidade foram realizados com cinco clientes do CRPG para testar as nossas hipóteses.

A aplicação móvel integra múltiplas modalidades de interação, e apresenta funcionalidades relacionadas com tarefas diárias que os clientes realizam, como selecionar refeições, consultar horários de transporte ou meditar. Os utilizadores podem interagir com a aplicação usando o ecrã tátil ou comandos de voz, e sugestões de áudio também foram implementadas. Notificações contextuais servem de lembretes para eventos e fornecem atalhos para recursos específicos da aplicação. As modalidades de interação integradas foram bem recebidas pelos utilizadores. Os testes realizados sugerem que as múltiplas modalidades são adequadas para permitir que os participantes cumprissem as tarefas. Os participantes mais jovens, mais experientes em aplicações para smartphones, tiveram melhor desempenho ao interagir usando o ecrã tátil. Os participantes de maior idade, que autodiagnosticaram a sua destreza no uso dos dispositivos como sendo menor, pareceram beneficiar com o uso das sugestões de áudio e dos comandos de voz. Algumas tarefas foram realizadas em menos tempo, e suscitando menos dúvidas quando comparadas ao uso do ecrã tátil do dispositivo.

A inclusão dos utilizadores no processo de desenvolvimento da aplicação permitiu dar uma resposta mais eficaz às necessidades da comunidade do CRPG, levantando obstáculos e questões que não seriam identificadas de outra forma. Os resultados dos testes sugerem a viabilidade e eficácia de múltiplas modalidades de interação na melhoria da acessibilidade da aplicação, garantindo que a aplicação consiga abranger o máximo de pessoas na comunidade. O estudo foi limitado pelo grupo pequeno e homogéneo. Todos os participantes utilizam smartphones diariamente, e não

usavam nenhum outro tipo de tecnologia assistiva ou de assistência para ajudar à execução das tarefas. No futuro, outras modalidades de interação podem ser integradas para garantir que ainda mais clientes do CRPG possam utilizar a aplicação.

Keywords: Acessibilidade, Mobile Health, Desenvolvimento para dispositivos móveis, Reabilitação, Tecnologias Assistivas

Acknowledgements

I would like to thank my mother, Clementina Silva, father, José Vieira dos Santos, and my brother Pedro Santos for supporting me not just during these trying months, but throughout the years so far.

I would also like to thank my supervisor, Bruno Giesteira and my co-supervisor, Pedro Cardoso, for their guidance. I would also like to thank Cecília and Sandra of CRPG for their cooperation, as well as the clients who took part in this process, for their availability and kindness. Last but not least, thank you to everyone who contributed by either providing advice or by sparing some words of encouragement and motivation.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
BaaS	Backend-as-a-Service
CRPG	Centro Profissional de Reabilitação de Gaia
EMA	Ecological Momentary Assessment
EU	European Union
FBAUP	Faculdade de Belas Artes da Universidade do Porto
HCI	Human Computer Interaction
ICT	Information and Communication Technologies
IDC	Inclusive Design Cube
IDE	Integrated Development Environment
ISO	International Organization for Standardization
iX	Specialization in Interaction Design, Web and Games
OOD	Object-Oriented Design
OS	Operating System
SDK	Software Development Kit
SR	Speech Recognition
TTS	Text-To-Speech
UCD	User-Centered Development
UDD	User-Driven Development
UI	User Interface
UX	User Experience
WAI	Web Accessibility Initiative
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
WCAG2ICT	WCAG Applied to Non-Web Information and Communications Technologies

Chapter 1

Introduction

The management of healthcare and self-care by rehabilitation patients has been the subject of significant changes over the years. Technology advancement is part of that change, by enabling new ways of monitoring and delivering healthcare to patients. One of the key elements to the innovation in healthcare is related to the widespread adoption of *smartphones*.[1]

The practice of medicine and public health supported by mobile devices (also known as mobile health or *mHealth*), aims to improve quality-of-life for the clients and caregivers. These aid individuals in their management of chronic medical conditions, allowing for early self-identification of symptoms, monitoring of their condition and contributing to a better adherence to treatment, through self-care [21].[2]

In the context of the Specialization in *Interaction Design, Web and Games* (also known as *iX*) from *Faculty of Fine Arts of the University of Porto* (FBAUP), a partnership was made with *Centro de Reabilitação Profissional de Gaia* (CRPG).[3] CRPG requested a platform which would help the clients in their rehabilitation process.

This dissertation aims to continue the development of the previously accomplished work in this project, at this stage primarily focused on the implementation of the platform, using the non-functional prototype and its documentation as the starting point. The expected result is an inclusive mobile application to be used by CRPG clients, resulting in a positive impact on their daily lives and rehabilitation process.

[1] A cell phone that includes additional software functions (such as e-mail or an Internet browser).

[2] Self-care is defined as the capacity of an individual to monitor health state independently of the contact with the caregivers [71].

[3] Center of Professional Rehabilitation of Gaia.

1.1 Context

CRPG started operations in 1992, as the result of an agreement between three entities (IEFP, CERCIGAIA and ADFA). Increasing the social and economical autonomy of people with disabilities is CRPG's mission. The institution offers several services and solutions for individuals with various levels of incapacity or deficiency (both physical and cognitive) with the aim of integration into the labour market. Some of these include rehabilitation and training of patients consultancy support for organizations/companies, insurance companies and hospitals, and prescription and sourcing of support products [18].

The focus of this specific proposal is on the rehabilitation and reintegration of victims of labor accidents and occupational diseases. CRPG uses the term "clients" to identify participants in their ongoing programs, and this nomenclature is used throughout this document. Clients in this condition enroll into a program designed by CRPG named *RAC-LCA*. The five month program consists of an assessment of the condition of each client, with a formative offer and development of the skills of each client depending on the characteristics of the physical or cognitive limitation.

As mentioned previously, there is a high variance between the capabilities and skills of each specific client. Some of the clients may need the assistance of a caregiver, or may even need specific hardware (such as peripherals) to allow them to interact with devices such as computers and mobile phones. Such circumstances provide additional challenges to the autonomy and require detailed assessments.

In order to improve the autonomy and quality life of its clients, CRPG requested the creation of a mobile application with several requirements, inserted in the scope of the program "Recovery and Updating of Skills – Acquired Brain Injury". A group of students of the specialization in "Interaction Design, Web and Games" were responsible for the first phase of the project. The design stage of this application, in which a User Centered Design methodology was adopted, relied on close collaboration with the CRPG and its clients during a three month period. During this phase, users' specific needs and expectations on the management of their everyday tasks were gathered through testimonials and meetings with clients and staff from CRPG.

Contacts with CRPG in the first and second phase are mediated by Cecília Carvalho, *R&D* assistant coordinator, and Sandra Guerreira, psychologist. Before, during and after the development of the project, the establishment of a communication channel for exchange of ideas with CRPG collaborators was necessary for the project's success.

Along with answering the needs of CRPG and its clients, it is important to mention the restrictions and obligations present in the Portuguese and European Law regarding public-sector accessibility in websites and mobile applications. CRPG is an institution that works closely and is related to Government-owned entities, and therefore can be included in this sector.

The most relevant policy at the European level is the *European disability strategy 2010-2020*, presented in 2010 [17]. Under the umbrella of this policy, the European Parliament, to harmonize varying standards within the EU, reduce barriers for developers of accessibility-related products and services, issued the *Directive (EU)2016/2102* [24]. This directive aimed at making public

sector bodies websites and mobile applications "perceivable, operable, understandable and robust". This strategy is the follow-up to the *eEurope* action plan.[4] The critical ideas mentioned in this Directive serve as the basis for the considerations made in the Portuguese Government's decree which is mentioned below.

Another crucial objective of the decree is the diminishing of inconsistencies of national measures across several European countries. The aim to end the internal market fragmentation is made possible by agreed accessibility requirements. As a result, uncertainty is reduced for developers and interoperability is fostered.

Regarding the Portuguese Law, under the *Decreto-Lei n.º 83/2018* [80], a proposal is presented with regards to the "standardization of open standards in the State's computer systems, the local Public Administration and the functions considered essential, concerning accessibility of the content of websites web and mobile applications". The decree definition commands the obligatory use of the *WCAG* (more details in section 3.1) standard *WW WCAG 2.1/EN 301549* on mobile applications and other comparable standards in websites. Guidelines advising for WCAG by the Portuguese Government had been proposed back in 1999, albeit not being as far-reaching.[5]

The combination of the guidelines presented in the mentioned legislation with the requirements identified in CRPG's proposal is necessary to ensure the adequacy of the proposed solution. It is also likely that the legislation contemplates some requirements that were not included in the proposal but which foster the inclusivity of the platform.

[4] A list of initiatives and guidelines to promote inclusion and extend Internet and digital platforms access for people with disabilities. This plan marked a milestone in the *European Union* (EU) guidance on *Information and Communication Technologies* (ICT) accessibility[85].

[5] Decree *RCM 97/99* recognizes that the adequate access of citizens with special needs to the information society is prioritary importance and is a factor of social inclusion. The decree also highlights the necessity of adequate processes of search and consultation of documents [79].

1.2 Questions, Goals and Challenges

The clients of CRPG have difficulties interacting with devices such as smartphones and therefore need specific solutions that cater to their needs. By analysing the report and the existing literature, the presented hypothesis is based on the claim that the autonomy and quality of life of CRPG clients can be improved by means of an inclusive management platform, adapted to their needs of accessibility and multimodality. In order to validate this hypothesis the following research questions should be answered:

1. How can inclusive design methodologies and approaches contribute to an adequate implementation of the non functional-prototype?
2. How can assistive technologies and multiple interaction modalities be integrated in the mobile application in a way that specifically benefits the CRPG clients?
3. How do the multiple modalities of interaction between user and system benefit the clients of CRPG?

To answer the first question, a study of the literature regarding existing methodologies for inclusive design and software development allowed for the optimization of the development process to address the characteristics of this project. Regarding the second question, by understanding the needs of the users and by validating the results during the development phase, it is possible to assess the adequacy of the multimodality and interoperability with assistive software to increase user satisfaction with the support tools at every iteration. To answer the third question, usability tests were conducted at the end of the development phase, in order to allow CRPG clients to interact with the different existing modalities and gather feedback regarding the interaction between user and application. With this into consideration, the goals of this dissertation are the following:

1. Develop a mobile application that can respond to the needs of the CRPG community, based on the non-functional prototype that was created in the design phase, following inclusive design methodologies and continuous feedback from the CRPG clients and representatives during development.
2. Identify requests and common necessities amongst the CRPG community, in order to increase the accessibility and expand the reach of the application to as many clients as possible.
3. Evaluate the application with a group of participants from CRPG who interact with the system, to assess the efficacy and adequacy of the multiple integrated modalities in the context of the CRPG community.

The main challenges are related to the high degree of variance in each client's autonomy and capabilities, which means that the app needs to be flexible and adaptable to different circumstances and needs. Overcoming these challenges translates the highly innovative component of the platform, as it made use of the advances in technology in general, and more specifically of the accessibility tools available for mobile developers, which rapidly increase, to improve the non-functional prototype and be perfectly adjusted for the necessities of the clients at CRPG. Compared to other existing platforms, this solution has several differentiating and innovative factors: multimodality, high level of accessibility and its focus on a group of users with particular needs, requiring a specific approach.

1.3 Expected Results

The study regarding the technological approach to respond to the specific accessibility needs should allow for the implementation of innovative tools. The focus on multimodality and interoperability with assistive technologies also contributes to assuming a specific type of development which is, first and foremost, for the users. The contributions made by this work can also be expanded to other communities with similar needs, given that the features of the application are mostly based on common daily tasks which are transversal across different socioeconomic status.

By removing any obstacles to the usability of the app and including all of the requested tasks, it was possible to increase its reach to as many CRPG clients as possible. The finished application is expected to have a positive impact and benefit the clients and staff of CRPG, by being a complement to the rehabilitation process and possibly even increasing the percentage of success in the adherence to the proposed regimen (such as medication intake and activity participation) by the clients.

The continued validation of the app, by comparing the results obtained from the usability tests done at the end of each phase of the project, contribute to a deeper understanding of the needs of the users and can also serve as a point of reference for future work, by allowing an understanding of what was done adequately, but also identifying points which can be improved.

1.4 Document Structure

In chapter 2, a literature study is conducted on accessibility and methodologies for software development. Guidelines for inclusive design are presented, as well as an overview of the provided non-functional prototype. A study of related platforms is also presented. Chapter 3 provides an overview of development tools, assistive technologies and accessibility guidelines.

In chapter 4, considerations regarding the proposed solution are made, including an overview of the application and its functionalities, software architecture, and intended approach. Details of the tasks, approaches to development and usability testing are explained. Limitations and challenges which led to disruptions in the scheduled are also identified.

Chapter 5 describes the technical aspects of the development of the application. The chapter starts with a description of libraries and development tools used to develop the application. Restrictions to development, such as technical limitations or requirements, are detailed. An overview of both the first and second prototypes is presented. After the prototypes are described, the file hierarchy and structure of the application are presented. The implementation chapter also highlights approaches to technical challenges that were identified during development, and how the integration with assistive technologies was achieved.

Chapter 6 provides insights into the different types of tests performed, and the results which were obtained, both in tests performed during development and also with clients at CRPG premises. Chapter 7 presents a summary of the previous chapters, limitations to the development, future work proposals and the conclusions.

Chapter 2

State of the Art

This chapter presents a study of the literature in different relevant areas to this dissertations. Section 2.1 is based on the report that accompanies and documents the non-functional prototype. Some remarks are made on the development process and the impact of the guidelines on the implementation process.

Section 2.2 provides an overview of health management platforms, and a description of their uses. In the same section, related work regarding similar management platforms and their main features is presented. In section 2.3, insights into the design for multimodal software, along with a discussion regarding their adoption in the context of this project are introduced.

An analysis and discussion of several approaches to software development is presented in section 2.4. Section 2.5 presents methodologies and approaches specific to inclusive software design. These approaches were used both in the design and implementation phases of the development of the application.

2.1 Previously developed work

As mentioned in chapter 1, the first phase of the project aimed at requirements gathering and subsequent proposition of the application's design. The process and findings of the first phase are documented in a report titled *Creating a Mobile Application for people with brain injuries*, which serves as the basis for the work on this dissertation [73].

Subsection 2.1.1 provides an overview into the user research and requirements gathering phase. Subsection 2.1.2 details the workflow and wireflow of the application. Subsection 2.1.3 presents the main elements of the high fidelity prototype. Lastly, in subsection 2.1.4, insights obtained during the usability tests are detailed.

2.1.1 Requirements

A survey was conducted with CRPG staff, which identified as an issue the cognitive deficits that some of the users presented related to memory, planning and daily life management. As a result, CRPG requested a mobile application that would allow each user of the institution to access information regarding their specific activities, as well as other key-functionalities. Although similar platforms for rehabilitation management have been developed, the focus on the needs of CRPG clients and use of innovative assistive technologies are differentiating factors for this specific solution.

The report is supported by user research, involving the cooperation of CRPG staff and testimonials of clients for a better understanding of each individual's daily routines and level of dexterity regarding smartphone use. This process relied on the communication with clients of the institution in order to integrate the necessities of the end-user and to ensure accessibility. This process relied on the communication with clients of the institution, in order to integrate the necessities of the end-user and to ensure accessibility.

The descriptions that each client provided regarding tasks in their daily life, along with the requirements of CRPG's first proposal, led to the definition of six significant features, which correlate with clients tasks and needs: *Agenda*, *Reminders*, *Meals*, *Transports*, *Notes*, and *Meditation*. The following list details each feature:

- **Agenda**: allows users to consult the schedule and information of the activities part of their rehabilitation program.
- **Reminders**: allows users to set reminders, such as alerts announcing when is the time to take the medication.
- **Meals**: a menu with the meal options for the following day is shown, and clients can choose one of the available options.
- **Transports**: users can access the timetables of public transportation which have stops near CRPG, as well as seeing on a map the path between their current location and CRPG. requesting CRPG's own transport is also possible.
- **Meditation**: users can access audio files for meditation purposes.
- **Notes**: users can store notes in different formats (text, images, audio files).

The priority associated with each feature was assessed for reasons related to the planning of the first phase. However, the priority hierarchy was also used to define which features should be implemented first. The priority of each feature regarding its impact in the user's daily life was discussed with Sandra Guerreiro from CRPG, and after consideration the following priorities were defined:

- **High priority**: Agenda and Reminders.

- **Medium priority**: Transports and Meals.
- **Low priority**: Meditation and Notes.

After the User Research, the team discovered also that CRPG client disabilities were related with cognitive and movement issues. In order to represent a wide range of needs specific to different clients of CPRG, three personas[1] were created. For each persona, several indicators were defined: *Goals*, *Motivations*, *Pain Points and Skills* and quantified their *Focus*, *connection to Technology* and *Anxiety*. By attributing different characteristics to the personas, each one ends up representing an individual with a unique set of characteristics. Scenarios were also created, in the form of a description for each one of the personas. This process allowed the team to have a deeper understanding of the daily lives and needs of the clients, and, as a consequence, of the requirements which the platform would have to meet.

2.1.2 Workflow and wireflow definition

With the requirements and features defined, the team drew the workflow and a preliminary interface based on the priority of each task and possible interactions. After logging in, the main screen is presented. The initial screen is the one that was defined by users when they start the activities at CRPG. From there, users can navigate to each feature screen.

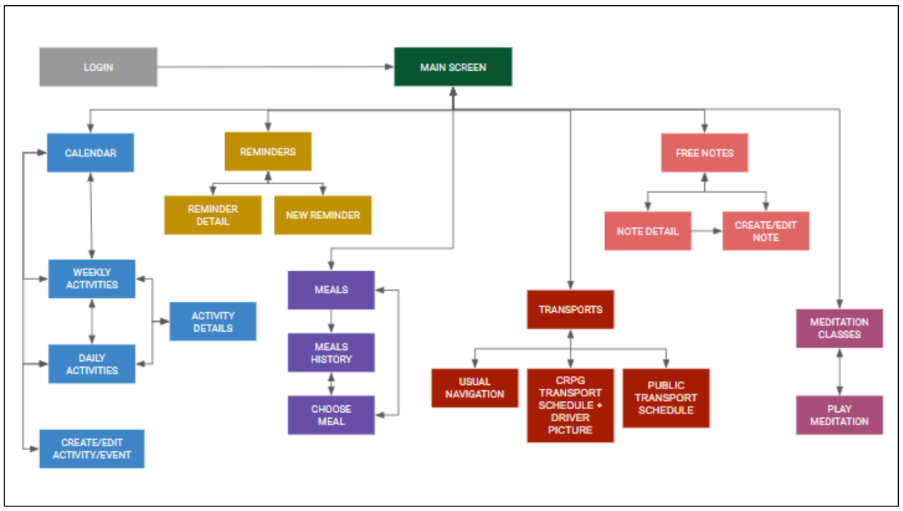

Figure 2.1: The proposed workflow (extracted from [73]).

From the Agenda screen, users start in the daily view and can opt to go to the weekly view. They can also see activities details and create/edit a new activity. In Reminders, the initial screen is the list of the reminders. Here, reminders can be individually selected in order to consult specific information regarding each one. It is also possible to go to a screen to create/edit a reminder. In Transports, the option to choose between the public transport screen and CRPG transport screen is offered. Similar to Reminders screen, in Notes users can go to the note's details or create a New

[1] A fictional character created to represent a user type that might use a site, brand, or product in a similar way [54].

Note. Lastly, the Meditation screen gives access to meditation details and plays the audio file for the meditation classes.

After the definition of the workflow, the team defined the wireflow documenting the interactions such as layout changes and system feedback. Fig.2.2 presents an excerpt from the platform wireflow, regarding the wireflow of the Notes feature. The screens are designed to reduce the number of clicks and inputs given by the user to finish the task.The platform is also adapted to let users quickly correct errors, by providing buttons which cancel the task or return to the previous screen, depending on the feature.

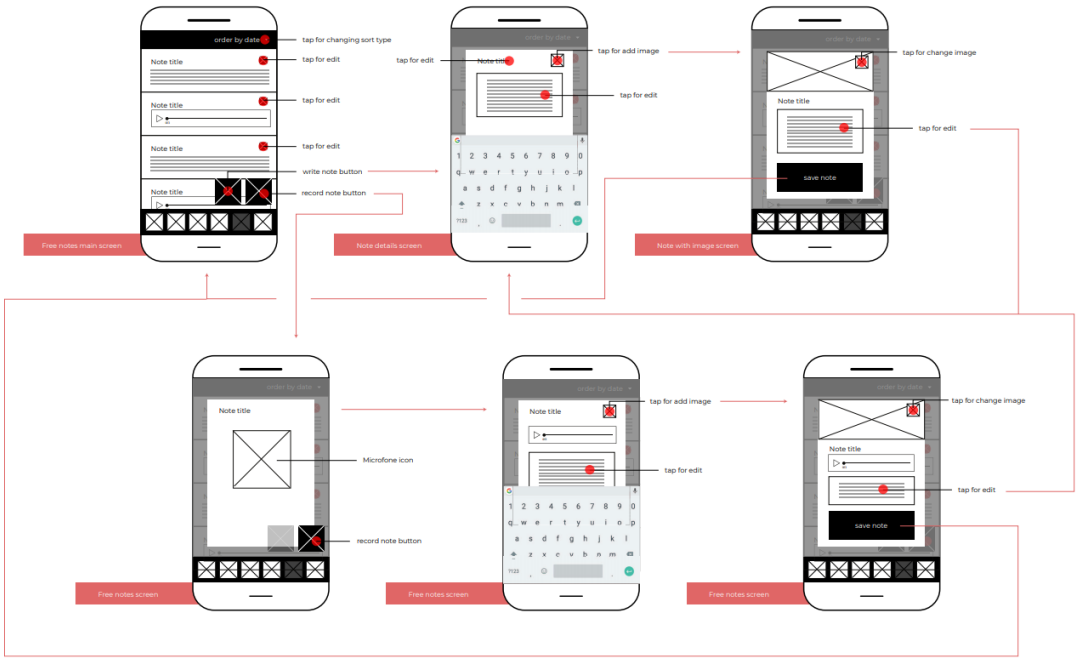

Figure 2.2: Wireflow of the Notes feature (excerpt from the platforms wireflow document, extracted from [73]).

2.1.3 High-fidelity prototype

A request was made for the brand identity of CRPG to be consistent with the institution's updated website. The colour palette of the website presented some problems to the users, given its high contrast. The colour palette was modified to reduce the overall disparity between colours, resulting in improved readability. The final prototype also features guidelines from *Mobile Accessibility Principles* (w3.org) [90] and *Material Design* (Google) [6]. As a result, elements such as the elevated cards concept[2] and hierarchy of information are present.[3]

[2] A snapshot-like display that groups related information in a flexible-size container visually resembling a playing card[72].

[3] Visual design principle which designers use to show the importance of each page/screen's contents by manipulating characteristics such as size and color of visual elements[23].

The report also dwells on the need to consider unexpected situations and what use cases may be more prone to the occurrence of an alternate flow of action (i.e. allowing the user to complete a task even if mistakes are made during the process). Several iterations were conducted, each one improving on the existing work. The screen for the Agenda with regards to the final iteration can be seen in Fig.2.3.

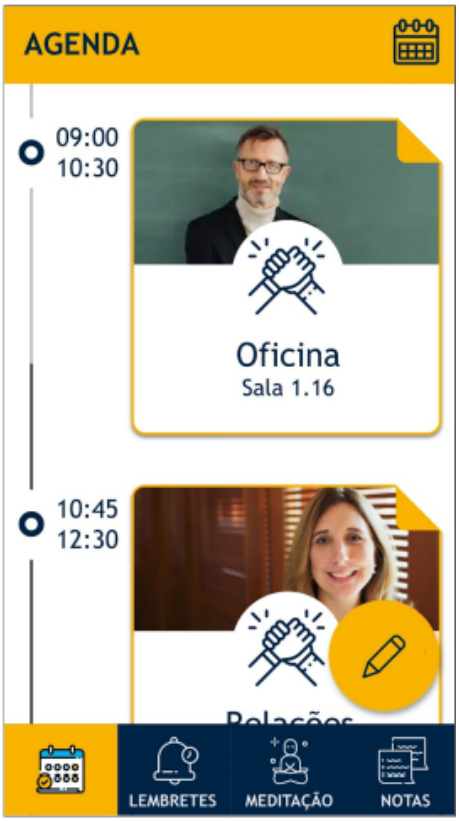

Figure 2.3: The design for the non-functional prototype for the Agenda functionality (extracted from [73]).

The screens available in Fig.2.3 and 2.4 present a visual representation of some of the features and guidelines used. The use of high contrast between colors, simplicity and lack of clutter, as well as appropriate font and button sizes, are some of the concepts used to improve the readability of the data on display.

Using concepts that are familiar to the clients also helps them in understanding the visual cues. CRPG clients make frequent use of Post-Its to write their reminders. Since Post-Its are a known concept, the platform adopts a visual style for the reminders akin to a Post-It, making the task of setting a reminder more understandable.

The app also provides feedback regarding the validation of the data introduced by the user. Using the check symbol next to each field, the user gets visual feedback that the field is complete. While not possible to see in the image, the built auto-fill system automatically fills some of the fields depending on the type of reminder that is set by the user.

Figure 2.4: Example of the different screens for the Reminder feature (extracted from [73]).

Other features of the application share the same visual elements and concepts. The wireflow may vary, but from the starting screen when opening the app, finishing any task can be done with a small number of interactions by the user.

2.1.4 User feedback and proposed improvements

In order to assess the effectiveness of the non-functional prototype, usability tests were conducted. The main points of interest were *Easiness*, *Simplicity*, *Experienced Comfort* and *Readability*, *Recognition of graphical elements* and *Responsiveness* [73].

Participant feedback on the final prototype regarding the rehabilitation process was positive. However, the users also suggested that there was room for improvement, and thus the following suggestions were made:

- Highlight save function to make it standout and easy to spot.
- Use of different colors for events on the agenda.
- Including a component related to physical exercise would add value to the app.

The team who developed the report also did a self-assessment on what improvements could be made from their work. Some considerations:

- It was difficult for users to recognize activities happening at the same time.
- The user data should be stored as soon as the minimum requirements are inserted, in order to reduce work for the user (akin to an autofill system, seen in web browsers).

- The current time of day in order must be known by the platform to present the correct activities.
- The app should allow to choose a meal between four options available.
- Initial guides and pop-ups with tips should be added.
- Message screens for errors should be implemented.

2.1.5 Discussion

The extensive documentation (including a report) provided by the team responsible for the first phase allowed to a deep understanding of the intricacies of the project and the thought-process behind some of the design decisions.

The assessment present on the report demands reflection regarding improvement to some of the suggestions made by the users. Since the first phase was mostly focused on the design aspects of the platform, areas such as interoperability and peripheral/support software use were not subject of extensive documentation in the report.

These insights allowed the functional prototypes to be more adequate to the needs of clients at CRPG. Validation by the users was prioritized during development, by conducting usability tests in which obstacles to using the platform were identified and improved.

2.2 Related Work - Health Management Platforms

Healthcare and the medical area, in general, have been significantly impacted by the development of technology. The changes are very significant, enabling new approaches to patient care, as well as allowing the patients themselves to access more information and be more active in the management of their condition. One of the most impactful changes in the way society used technology is the mass adoption of smartphones globally, which led to a significant part of the global population having access to mobile apps with a broad range of uses.

As with many other areas, applications that allow patients to manage their conditions have become increasingly common. Estimates point to over five thousand mobile health apps as of 2017, but an accurate number is difficult to pinpoint as more apps are released every year. The range of functionality between these applications varies considerably, and they are generally categorized in six groups. The groups are as follows:

- *Lifestyle-oriented apps*: allow individuals to track their progress in regimens such as diets and exercise programs, leading to an healthier lifestyle.
- *Patient-oriented apps*: provide tools for early self-identification of symptoms along with management and adherence to treatment, allowing for better management of chronic medical conditions.

- *Clinician-oriented apps*: aimed at clinicians, allow patient management by providing reference or educational information as well as aiding decision making.
- *Disease management systems*: allows for the monitoring of patients with chronic conditions by clinicians. the systems (which can be web based or a mobile app) may be integrated into electronic medical records and can include decision support tools.
- *Traditional telehealth systems*: provide and deliver information and services remotely using electronic communications.
- *mHealth systems*: akin to telehealth applications but usually assume the form of a mobile application rather than being accessed via a computer.

The main focus of this dissertation is in the group of *mHealth* apps. These applications aid individuals with medical problems in managing their conditions by offering information about their specific circumstances, as well as allowing them to manage and incentivize treatment adherence.

These applications cannot replace the indications of medical personnel but can be used as an increment to an already defined process. Medical personnel should always supervise the use of these platforms. Despite the obvious limitations and precautions regarding the use of these platforms, some literature claims benefits on their moderated use [21]. The overwhelming majority of these platforms are implemented as applications for smartphones, mainly due to the following advantages that this option enables:

- Individuals are more prone to carrying smartphones with them at all times, allowing for improved accessibility.
- The devices are able to collect *Ecological Momentary Assessment* (EMA) data.[4]
- The communication capabilities of these devices, for example in a case of distress or need of urgent care; the platform can provide the necessary contacts, and the user can use the smartphone to contact a caregiver instantly.

One of the significant points of interest in the design of these applications is their adoption of user-centered, high accessibility interfaces, which have to take into account individuals with a range of different physical and psychological frames. A review of papers concluded that human factors approaches are nearly always adopted in the design, development and evaluation of mobile applications. The approach to the implementation of the platform results in a more engaging and positive experience for the users, improving the rehabilitation process.

One of the most used methods for the human factor approach is the use of interviews and surveys, which allow an assessment on the use of information technology, including mobile apps. Technical literacy is an obstacle in the development and conceptualization of accessible software, as some of the users are people of an older age and which may have less exposure to the use of mobile applications.

[4]Data obtained in real-time regarding the user well-being and their current location[53].

A considerable number of mHealth solutions are present in the market, requiring a selection of notable examples. The research for this dissertation focuses on applications released during 2010 or after. The research could not find a significant number of solutions subject to extensive usability testing, and as such three examples were chosen. The decision to use these examples is due to them sharing one or more functionalities with the platform that is proposed by this dissertation, as well as having literature published which includes usability assessment.

2.2.1 iMHere (interactive Mobile Health and Rehabilitation)

iMHere is an application for mobile phones which can be used to support self-care and adherence to self-care regimens for patients with chronic conditions [25]. Through the smartphone, users are able to access a number of functionalities aimed at improving their rehabilitation process, such as setting up alarms and confirming medication intake, as well as being able to report skin issues.

This application is the subject of several articles regarding the usability and accessibility of mHealth apps. The problem which iMHere aims to solve is similar to the one faced in this dissertation, not only regarding the functionalities, but also because it focuses on clients with overlapping characteristics. Given the extensive research which used performance metrics to assess iMHere usability, it appears to be the most relevant subject of study regarding the several mHealth apps under scrutiny in this section.

Adding to the extensive research and usability tests done, it is worth noting that different versions of the app, which try to improve upon the initial version, allow the readers to understand the weak points of the first release, and how incremental improvements contribute to expanded accessibility in later releases.

The first version of the app, iMHere 1.0, was released in 2013, and is the subject of study for the article *Perspectives on the Evolution of Mobile (mHealth) Technologies and Application to Rehabilitation* [21]. In this article, an overview of the scope of the app, as well as possible uses, is presented. A visual representation of the screen allowing the user to see its medication list can be seen on Fig.2.5.

The first version, while having received positive feedback from its users, still presented a number of barriers which undermined its usability. Several articles tackled the weak points shown in iMHere 1.0, leading to improved versions of the app. One of the most notable examples is *Accessibility needs and challenges of a mHealth system for patients with dexterity impairments*, in which points of improvement were identified for the original version of iMHere (1.0). In order to create a new, improved version (2.0), accessibility needs and preferences of individuals with dexterity impairments were identified, and usability tests carried out to assess their implementation [96]. In the discussion section of the article, some of the main accessibility issues with the design of the application were identified:

- Screen size is the main difficulty.
- Screen cluttered with irrelevant clutter, making text difficult to read.

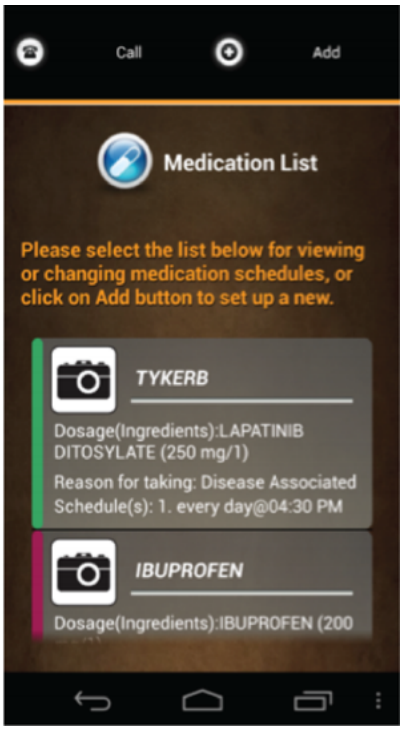

Figure 2.5: Medication list screen of the iMHere 1.0 mobile application. (extracted from [21]).

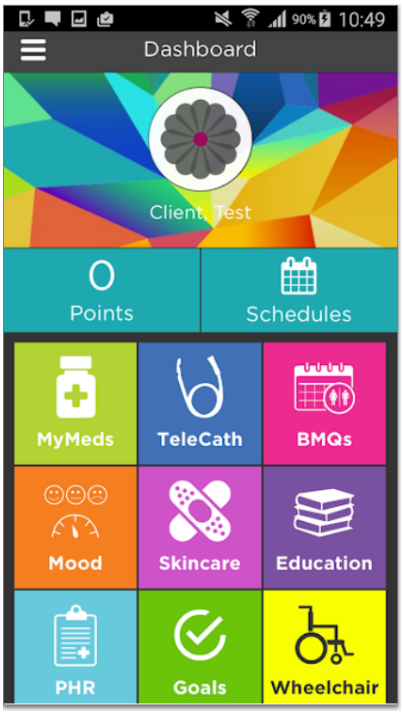

Figure 2.6: Dashboard of the iMHere 2.0 mobile application (extracted from [69]).

- Low contrast of colors.
- Inappropriate text size and font.

The results of this study appointed a number of key accessibility tools that improved the experience of the users considerably:

- Thematic colors.
- Instructive guidance.
- Simpler cognitive process.
- Alternative camera button (one of the functionalities allows the user to take a picture of the skin).
- Simple and streamlined applications.

Yu and colleagues [96] did an assessment of the needs and obstacles that impaired users faced when using the iMHere app. A one-week field trial using the iMHere app was conducted. In this trial, a group of nine participants participated. Among the group different levels of dexterity impairment allowed for a more accurate assessment of the usability. As might be expected, the degree of impairment of each individual user was shown to have a strong correlation with the amount of errors that happened when using the app interface.

Given the fact that the usability tests conducted included individuals with a broad range of capacity, it is logical to assume that there is an overlap between the necessities of the users of iMHere and the users of the rehabilitation platform. The guidelines and ideas present on the results section of the article should therefore be considered of great relevance when making decisions regarding the implementation of the app.

2.2.2 Medication Plan

Medication Plan (*Medikamentenplan* in the original German name) is an app for the iPad, developed in 2010. It was specified by the Department of Nephrology, Essen University Hospital, Essen, Germany; three German companies were in charge of its development [12]. The app allows the users to set reminders for medication intake. The app is targeted mostly at elderly patients or that suffer from chronic conditions.

Medication Plan is the subject of the scientific paper *A mobile application improves therapy-adherence rates in elderly patients undergoing rehabilitation: A crossover design study comparing documentation via iPad with paper-based control* [63]. The study aims to compare the effectiveness of advanced technological measures for increasing adherence to medication.

The usability tests allowed the work group to assess the main difficulties and points of improvement to the application's usability. The users (N = 24) were asked to perform several predefined tasks and then answer questions regarding their satisfaction with the different methods and characteristics of the user interface. Some of the key considerations made using the feedback from the users:

Figure 2.7: Scheduling screen of the Medication Plan mobile application (extracted from [63]).

- Eleven elements of the group considered the font size to not be comfortable.
- Twenty-four elements (all of the individuals) were satisfied by the use of the touch screen interface.

Mertens and colleagues classify the software as being successful in its goal. The iPad intervention improved medication intake, which was a significant achievement, as the initial recorded therapy adherence was already high, which correlates to a smaller margin of improvement. Even though other methods of improving adherence were studied, such as using a pen and paper journal, the digital solution resulted in greater adherence and surpassed the non-digital alternatives.

2.2.3 UbiMeds

UbiMeds is a German mobile application and architecture that aims at supporting prescription medication adherence. Like the previously mentioned solutions, it serves specific sectors of the population, such as older or disabled individuals[84]. The article *UbiMeds: A Mobile Application to Improve Accessibility and Support Medication Adherence* describes the design process guided by the accessibility issues that arise from the characteristics of the users. Fig.2.8 shows the View Screen of the application in which the user can get an overview of the schedule and adherence status. The authors identify four types of limitations regarding accessibility which contributed to significant levels of non-adherence:

- **Medication related**: the amount of different medication to remember, schedules which can be hard to memorize, drugs with similar names or similar containers.
- **Patient related**: trouble remembering events or tasks, health literacy, understanding the different types of conditions and medications, incapacities (of any type).

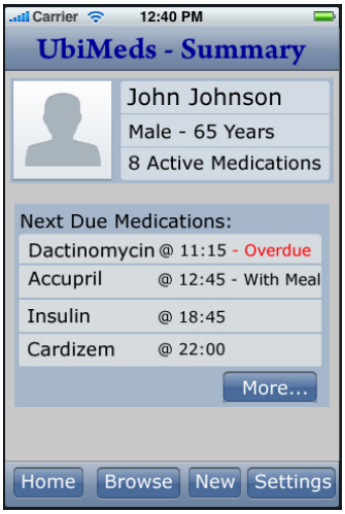

Figure 2.8: Scheduling screen of the UbiMeds mobile application (extracted from [84]).

- **Prescription related**: lack of communication or incapacity of transmitting ideas and directions to the patient, negligent patient monitoring, illegible handwriting on prescriptions, complex technical jargon.
- **Pharmacy related**: difficulty reading type font, inconsistent formats in the instructions, printing errors which cause misunderstandings on correct dosages and other crucial information.

Following the identification of the problems faced by the patients, the authors realized a list of implementations that were found to remove the obstacles raised by the mentioned issues:

- **Automated Reminders**: the system should proactively remind patients of scheduled medicine intake hours, both by visual and audio cues.
- **Intake tracking**: a tracking system verifies the adherence to the drug prescription by the user, and notifications are sent if the system identifies lack of compliance.
- **Visual Aids**: information on the screen should be easily comprehensible, by using adequate visual elements such as icons, font type and color selections.
- **Text to speech**: the system dictates the instructions on how to take the medicine.
- **Separation of information**: interface views are designed with the intent of avoiding overload of information. The screens should be as free of clutter and non-essential information as possible.

The architecture of the system is also explained, as this solution includes not only the mobile app, but also a centralized module and a web app which is used by the medical team. In the section *Conclusions and Future Work*, the authors indicate that a usability study would be needed

to correctly assess the adequacy of the taken measures in order to mitigate any usability obstacles for the users. A mention is also made regarding the privacy of the users. In the UbiMeds system this problem is exacerbated due to the reliance on a third-party system like *Google Health* and *Microsoft Vault* to store the health records.

2.2.4 Discussion

The field of health management platforms was introduced, and the different existing categories detailed. After consideration of the capabilities and purposes of each category, the proposed platform was identified to be part of the mHealth category of health platforms.

Three platforms were the subject of analysis, due to their relevance to the scope of this dissertation. The first analyzed platform is iMHere, which is the subject of different articles and had several versions.

The second and third platform, MedicationPlan and UbiMeds, had less literature available. However, looking at the conclusions and recommendations published in both cases, there is a clear overlap on the topics which are considered important. Elements such as visual aids, automation, use of voice commands and adequate text and button size are valid concerns across all platforms, and were considered to be helpful when trying to increase the usability of the platforms. Due to the similarity in the approaches and expected results of the platforms, the concerns mentioned in their development were also addressed when developing the platform for CRPG.

2.3 Design guidelines for multimodal applications

Due to the unique challenges of developing a platform for individuals with disabilities present, a study regarding multimodal applications is relevant. Reeves[82] presents some key ideas regarding this topic:

- Simplicity is of high priority, in order to avoid cognitive load.
- Flexibility and customization to answer the needs and requirements of the users and to the context.
- Users should receive system feedback, that is, understand which modalities are being detected and interpreted.
- In order to prevent errors, the option regarding which modality to use should be given to user; undoing or canceling the ongoing tasks should also be allowed.

The user should be aware of what is happening at all times during the use of the platform. The visual or sound cues that are used to achieve that purpose are necessarily different from the ones used on software designed for people without any disabilities. Other possible approaches must also be considered:

- **Application of *Human Computer Interaction* (HCI) techniques in the following points**: independence between the interface and the application (e.g. use of a language/framework enabling a cross-platform consistent interface); advanced user interface design (study of the user profile leads to interfaces that are better suited to respond to particular needs); inclusive design (user diversity should always be considered during the design phase).

- **Adopting specific privacy measures for the users**: adequate storage (encryption use and reliance on third-party servers needs to be assessed in order to avoid security risks) and the duration of storage (it should only be stored as long as strictly necessary), must be specified. There may be additional concerns, such as location tracking by the phone, and others.

- **Designing specific feedback actions that actively improve usability**: active feedback system should be implemented, allowing the user to understand which modalities are being detected and interpreted, as well as continuously monitoring the application usage (by analyzing parameters for each user and suggesting changes).

- **Removing barriers and obstacles to access to the application**: cost and hardware requirements constitute a barrier to software adoption; software distributed for free and which can be used regardless of hardware limitations improves access for the users; It should also be independent of third-party applications, i.e. it should not require the installation of any other software in order to be operational.

Some of these concepts are also corroborated and object of further research later in 2009, in the paper *Multimodal human–computer interaction: A survey. Computer Vision and Image Understanding*[44]. The document provides a review on the major approaches to multimodal human computer interaction. The authors focus mostly on body gestures, gesture, gaze, as well as facial expression recognition and recognition of emotion in synthesized voice. System integration architectures, modeling, and adaptability are also included in the scope of the paper. Due to its increased relevance in the scope of this project, the focus was on emotion in the use and adequacy of synthesized voices, system adaptability and system integration.

The authors identify the perceived emotion of speech software as necessary to establish in the user a sense of trust and to improve its effectiveness. It is explained that the vocal aspect carries different kinds of information. Disregarding the manner in which a message spoken may create misunderstandings and does not allow the receiver to fully understand the meaning of the message. The limitations also have an effect on the user's speech, due to the technical shortcomings of language processing and the inability to perform a context sensitive analysis. If the user has speech impediments, the software may also be unable to properly interpret the conveyed message.

The paper also classifies software adaptability and customization as priorities. Adaptability is especially important when there is a significant difference in skill level, culture, necessities between the users of the system. The idea behind adaptive human-computer interaction is that more sophisticated natural input and output can be achieved. By refining the settings of the software,

users with varying levels of dexterity and capacity can perform complex type tasks more quickly and with greater accuracy.

One of the topics which is also relevant is the response to input and output from hardware devices such as peripherals, and how the interaction techniques change accordingly. Ideally the system itself should be able to take into account not only support software and hardware, but also store user preferences and dynamically present an adaptive interface which is suitable for that specific user. This translates, for example, into the interface handling errors and interruptions elegantly.

Despite some concepts which seem relatively consensual among researchers in this area, there are also initiatives which aim to unify concepts from different areas, and, by doing so, contributing to the development of more adequate multimodal software. In the report *Universal Accessibility as a multimodal design issue*, some topics regarding multimodal design are addressed. The authors claim that frequently interconnections between concepts of universal Accessibility and Human-Computer Interaction are not evident. To show the connection between universal accessibility and multimodal design the proposed solution is a unified generic framework where these areas overlap[67].

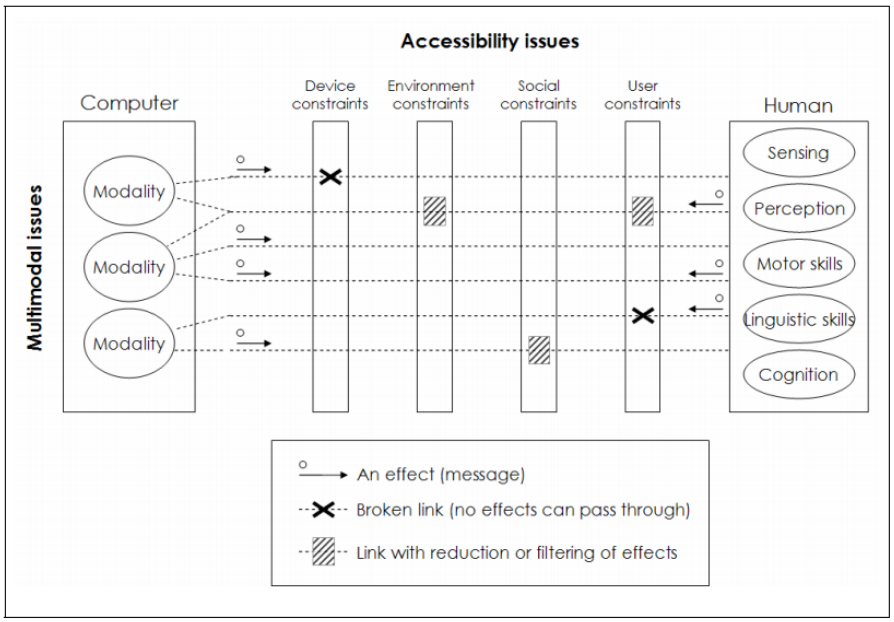

Figure 2.9: Visual representation of the modalities, constraints, and effects and their relation (extracted from [67]).

The main idea is that user interfaces can be described as a set of communication channels. If those descriptions can be linked with the profiles that represent the user's environment and devices, the adequacy of a multimodal interface can be assessed for a specific situation.

Initially a unified modeling framework was formally defined, with the objective of describing multimodal human computer interaction and various user and environment characteristics using standard terms which would minimize ambiguity. Using a table which maps the modalities and

effects it is possible to describe concrete user interfaces using the high-level descriptions of these modalities.

The investigators defending this approach identify several advantages. First, it allows for more flexible and reusable solutions to be used by developers. On the other hand, it enables the same treatment for different situations. The use of the term universal solution also discards the use of the term disability which often is a negative connotation. In this case, the constraints are not necessarily due to the physical limitations of the users.

2.3.1 Discussion

In section 2.3 several key ideas and elements of multimodal application design are presented. Most mentioned strategies are relevant in the development of the platform. By integrating elements of the design into the application, it will be possible to reach more CRPG clients, regardless of the limitations they may face.

Interoperability with assistive technologies and use of sound both as input/output are relevant for the implementation of the platform. Despite the identified limitations which are still relevant today, it is also true that the technology behind *Text-To-Speech* systems and language processing has improved significantly. As a consequence, the use of speech recognition and synthesized voice both as input and output for the platform were deemed suitable and helpful given the wide range of autonomy of the users.

Regarding the presented unified framework approach, despite appearing useful in several contexts, it was not considered adequate for the development process and as such its use was discarded. The concept of the communication channel does not adequately represent the implications of using assistive technologies such as support applications and peripherals, which have specific characteristics not mentioned in the paper.

2.4 Methodologies and approaches to Software Development

This section describes and compares approaches and methodologies to software development. Subsection 2.4.1 provides an overview of the Agile methodology.

Subsections 2.4.2 focuses on the Waterfall Approach and 2.4.3 on the Incremental Approach. For each one of the approaches, considerations regarding the adoption of each for the scope of this project are included.

2.4.1 Agile Methodology

The Agile approach allows for the development of solutions like software through a collaborative effort of self-organizing teams in collaboration with the end-users. It was introduced by the *Manifesto for Agile software development*. It borrows several principles from other software development methodologies such as *Scrum* and *Kanban*.

There are several key concepts that are used with the Agile approach. The first consists in the breakdown of the development work into smaller segments which minimize planning and design time. Using development sprints or iterations, which are usually in the time frame of one to four weeks, is also common.

Agile also advocates for the physical coexistence of coworkers or team members. As a result, face to face interaction is encouraged, contributing to reduce the time that is typically used for answering questions. In order to allow for very short feedback, daily stand-ups are used. These sessions consist in the reporting of what each member did in the previous day, what is intended to do on the same day and what challenges or impediments they face to reach their goal.

Along with the stand ups, Agile encourages tools and techniques which improve quality and enhance development agility. This translates among other elements, the ability to show the end user a version of the software which typically happens at the end of every iteration [11].

While not all of the characteristics of the Agile methodology can be applied to this project, namely the ones regarding team work, the use of iterations and functional prototypes for demonstration and validation by the end user will be adopted.

2.4.2 Waterfall Approach

The Waterfall model is one of the most common software development methodologies. Similar to the manufacturing process in an assembly line, it follows a structured sequential path. Each phase of the sequence is comprised of specialized tasks. At each phase, milestones must be accomplished before the next stage can begin [13]. A visual representation of this methodology can be seen in Fig.2.10.

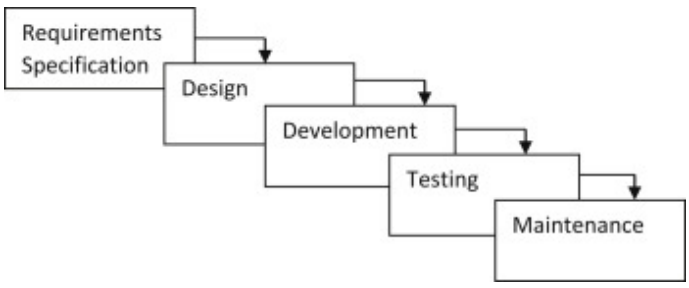

Figure 2.10: A visual representation of the Waterfall Approach (extracted from [28]).

This approach is limited because there is little flexibility or margin for corrections if new necessities arise during the development of the project. For this reason, many developers consider this approach to be outdated [62]. Due to the limitations of this approach and the necessity of

being able to adapt to new circumstances and feedback during development, using the waterfall approach does not seem to be the most adequate solution regarding this project.

2.4.3 Incremental Approach

In the incremental development approach, user needs and the overall architecture are defined. A series of increments, known as software builds, is then used to divide the system. The first build incorporates a part of the total planned capabilities, the next build adds more capabilities, and so on, until the entire system is complete [42].

The incremental approach allows partial utilization of the product and avoids a long development time. One of the main advantages is the ability to present functional releases soon in the development process, speeding up assessment of the current work and allowing for validation. Another benefit is that, when changes are necessary, it is not necessary to develop a new system from scratch. The current project should have some modularity that can allow for changes and flexibility. A visual representation of this methodology can be seen in Fig.5.1.

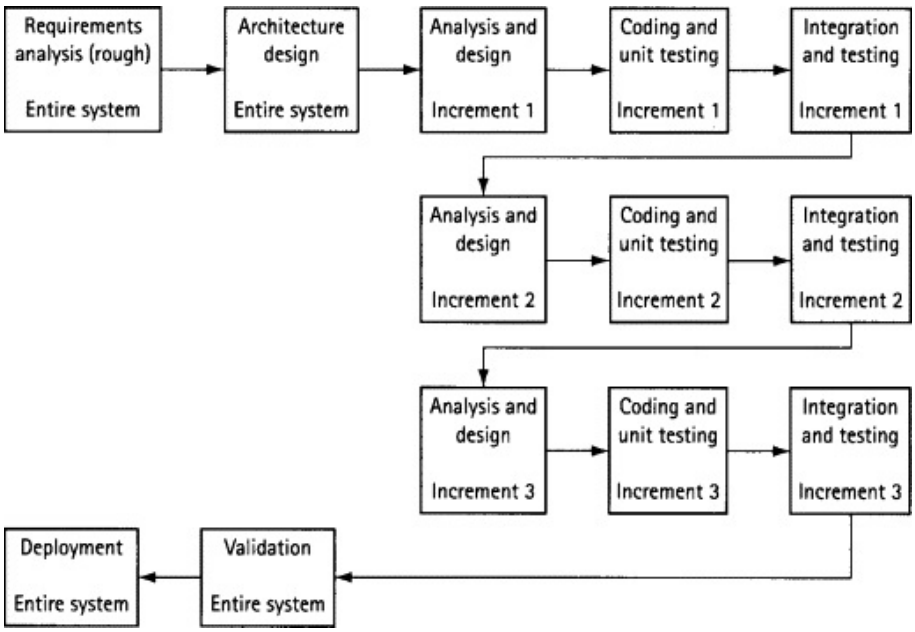

Figure 2.11: A visual representation of the Incremental Approach (extracted from [55]).

The incremental approach seems to tackle some of the uncertainties that are to be expected in the process of developing software. It is not realistic to expect to know and predict what changes could arise during development, since the development is a flexible endeavour.Given the circumstances of the platforms, this type of approach seems to be more suited for the development phase.

2.4.4 SOLID Principles

The *SOLID* principles are an acronym for the first five object-oriented design (OOD) principles by Robert C. Martin. Despite the evolution of software development and the surge of different approaches and principles, *SOLID* principles are frequently cited in articles.[5] These principles are also commonly used in IT companies globally. The following list summarily describes each of the principles:

1. **Single-responsibility principle**: "There should never be more than one reason for a class to change." In other words, every class should have only one responsibility[61].

2. **Open–closed principle**: "Software entities ... should be open for extension, but closed for modification." A program's modules should have responsibility over a single part of that program's functionality and be responsible for its encapsulation[60].

3. ***Liskov* substitution principle**: "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it". In other words, semantic consistency is maintained in complex inheritance hierarchies, making classes easier to understand and modify[59].

4. **Interface segregation principle**: "Many client-specific interfaces are better than one general-purpose interface." This principle protects objects from depending on the behavior they do not need. This principle aims to keep a system decoupled and thus easier to refactor, change, and redeploy [58].

5. **Dependency inversion principle**: "Depend upon abstractions, [not] concretions." In other words, high-level modules should not depend on low-level modules[57].

The main idea behind these principles is that using them leads to avoiding bad design decisions. There are several principles and approaches from different authors, so different projects may require different solutions. In the case of the application for CRPG, it should be expected that future work will be done on the platform. This is due to CRPG's IT department wanting to continue the project and adding more features and integration with the existing infrastructure. Using the *SOLID* principles allows the code to be more modular and easier to read. This allows modifications to the code to be done faster, and less time is spent trying to understand the responsibility of a specific class or module.

2.4.5 Discussion

Section 2.4 introduced several approaches to software development. Using both Agile and Incremental approaches seems justified, due to the possible change in requirements over the development phase and need for early validation by end users. The Waterfall approach was also

[5] Using the keywords "software SOLID principles" in the *Google Scholar* search engine returns 2.250.000 results (as of 08-06-2021).

introduced, but due to its limitations such as lack of flexibility and adaptability to unforeseen circumstances its viability is compromised. The SOLID principles were also found to be adequate in the context of this project, as they should improve the code quality and make future work on the application easier.

2.5 Inclusive Design Methods

This section provides an overview of design concepts with a focus on inclusivity. The first subsection, 2.5.1, introduces the concept of Participatory Design. Subsection 2.5.2 presents the characteristics and insights into User-Centered Design. Subsection 2.5.3 details the concept of Universal Design. Subsection 2.5.4 explains the concept of the Inclusive Design Cube, and establishes its strong points and limitations.

2.5.1 Participatory Design

Participatory design is an approach that aims to include all the stakeholders (clients, users, or other entities which may be impacted by the product) in the conception of the solution and allowing them to actively contribute to the design process. The aim is to ensure that the proposed solution meets the requirements and is suitable to the end users [87].

Two main types of participatory design can be defined: one attributes power to the user to provide insights for the professional designers to consider, but the final decision is the responsibility of the professional. This type is named consultative design. The second type gives users full power and responsibility in the final outcome, known as consensus design.

This approach has characteristics that overlap with the protest from software programming such as Extreme programming.[6] Both these approaches put the user at the center of the design of the software. However, participatory design focuses on the involvement of a broad group of individuals and not a small number of user representatives which is the case in a lot of software projects [49].

This project employs the use of participatory design, which started in the first phase of the project. When user research was conducted, the clients were questioned regarding features they liked to see on the interface and also identifying points of improvement which would make their interaction easier. Surveys and usability tests were used for validation of the design in each iteration.

[6] Software development methodology whose purpose is to improve software quality and responsiveness to changing customer requirements [10].

2.5.2 User-Centered Design

The development of an application that has the purpose of social inclusion has, necessarily, to have the necessities of the final users as its utmost priority. The concept of development and design focused on the user is discussed in *User-Centered System Design: New Perspectives on Human-Computer Interaction*, published in 1986 [66]. Edited by D. A. Norman and S. W. Draper, it is the result of collaborative work by *UCSD* group on Human-Machine Interaction and colleagues. The book lays out a map of central issues, questions, and complex trade-offs facing designers, creators, and users of interactive systems.

The authors highlight the importance of improving information flow between human and machine, employing a unified approach. Under some conditions, it is argued that the system should volunteer information to the users, without the users even knowing they needed the information.

The stance that error is an inevitable aspect of human action must be considered. Accounting for the user's fallibility allows for setting on the platform the burden of adapting to the user and not the contrary, removing obstacles in the interaction. In practical terms, software must be capable of monitoring task completion by the users. If erroneous or undesired actions are detected, it should allow for rapid and graceful recovery and return to a previous state.

Another critical idea mentioned is integrating the users into the development process by gathering information on how they progressively understand the system. The authors make the case that, with better knowledge about how users come progressively to understand a system, we may be able to build better learning support into it.

After the book's publication, extensive documentation regarding the development with this approach was released, and *ISO* (International Organization for Standardization) standards. While different authors and publications have different interpretations of what means to adopt a User Centered Design (also known as User Driven Development), some key ideas seem consensual.

The main focus is to research and discuss each stage of the design process from a user perspective. Several phases of the process need to be reviewed frequently, such as usability goals, user characteristics, environment, tasks, and workflow to guarantee adequacy at each change to the software.

In other product design philosophies, it may be expected that the user will adapt to the product, and not the other way around. Obviously this approach has an impact on the scope of the social inclusion of the software, so a good understanding of the users is necessary in order to assess its adequacy. In user-centered design, the focus is on optimizing the product around how users can, want, or need to use it. By doing so, users do not change their behaviour and expectations to accommodate the solution, contributing to more inclusivity.

2.5.3 Universal Design

The aim of universal design is to allow for the design of a concept which can be a product or a service to be used and accessed to the greatest extent regardless of the characteristics of the user such as age, ability, or disability [1]. In 1997 a group of seven principles for universal design were developed. These principles can be used to evaluate existing designs and guiding design processes [7]. The following list presents the different principles:

1. **Equitable Use**: Useful and marketable to users with different needs and capacities.
2. **Use Flexibility**: Allows for different individual preferences and abilities.
3. **Simple and intuitive use**: Easy to understand design, regardless of user characteristics.
4. **Perceptible Information**: information is conveyed adequately to the user.
5. **Margin of Error**: allows users to regain control after accidental or unintended actions.
6. **Low physical effort**: design does not provoke fatigue and can be used comfortably.
7. **Appropriate Conditions for Use**: Size and space is provided, and no barriers are presented to the manipulation, regardless of users body size, posture or mobility.

2.5.4 Inclusive Design Cube

One of the possible approaches is related to the use of an *Inclusive Design Cube*. This methodology highlights different design approaches that can be used to develop inclusive products [46]. The graphical representation of this cube is shown in Figure 2.12. In each the five different layers, With each one highlighting a different approach or necessity.

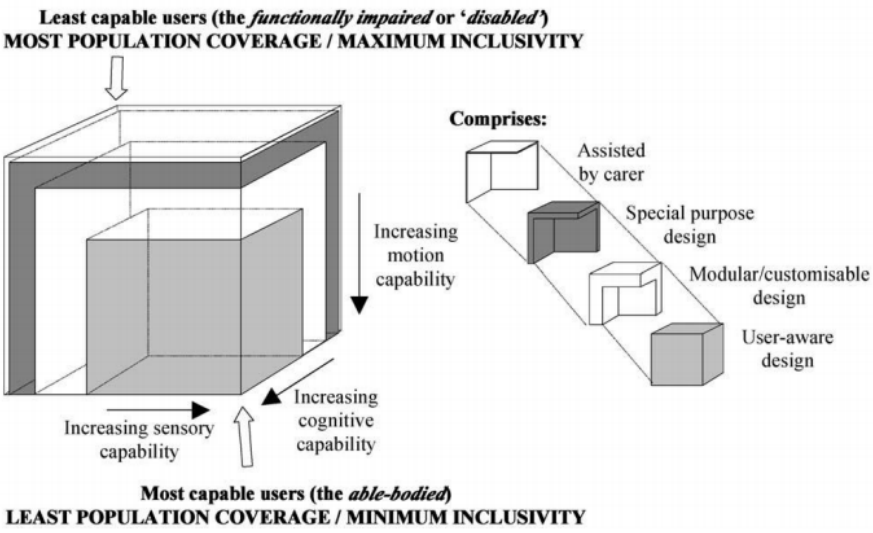

Figure 2.12: A visual representation of the Inclusive Design Cube (extracted from [46]).

Each axis of the cube represents beneath our capability, whether it is regarding motion, sensory input or cognitive limitations. The necessities of non-disabled users are obviously fewer than those of users who may have some incapacity, which leads to the most capable users to occupy a small volume of the cube.

In order to include other groups of users with varying levels of capability, the cube needs to expand in the three axis. Each axis length represents the capability of one area, respectively motion, cognitive and sensorial. The expansion of the cube, in practical effects, translates into going beyond focusing only on user-aware design and incorporating a customizable, special purpose design, as well as accounting for the intervention of a carer in the use of the software.The five levels of the IDC can be summarized as follows:

1. **User needs**: specification of system requirements.
2. **User perception**: identification of the output modalities that may steer users away from interacting.
3. **User cognition**: system clarity is defined, given possible cognitive impairments.
4. **User motor function**: definition of the most adequate input modalities.
5. **Usability**: system evaluation.

One of the main limitations of this methodology is that it does not take into account the necessities of individual users. This approach is more suitable when it is not possible to know who is the final user, or the number of users is considerably high, such that it does not allow for fine-tuning for specific individuals. Because of the characteristics of the current project, and also because there is no mention of the use of this method in the non-functional prototype report, its use does not seem justified here.

2.5.5 Discussion

In section 2.5, several methods for assuring inclusive design are presented. While these methods may not be applied directly in the implementation of the solution, their comprehension provides an insight into some of the design decisions, and how to continue refining the interface in a way that does not compromise the work done so far. Concepts such as participative design and user-centered design, which were applied during the design phase, were also used in the implementation phase.

2.6 Summary

In this chapter, the background of the areas related to the project were presented. Section 2.1 gave an overview of the first phase of the project, and introduced the main specifications of the non-functional prototype.

Section 2.2 provided an overview of health management platforms. It also includes an analysis of three relevant platforms for the scope of this dissertation. The absence of more examples in this section is due to a lack of platforms which were in conformity with the requirements. However, some conclusions were reached during the study of the presented examples, such as common points considered to be barriers to the user experience, and what measures can be taken to mitigate these problems.

The state of the art of the design guidelines for multimodal software development and guidelines that can be used or serve as a reference in the development of the solution were also reviewed in section 2.3.

Section 2.5 provided an insight into some of the inclusive design methods used during the first phase of the project. While the main focus of this project is on the development and not on the design of the platform, the exposition of the different concepts allowed for a better understanding of the reasoning behind some of the decisions, translating into additional knowledge during the implementation of the platform.

Chapter 3

Technologies

This chapter features an overview of the standards and technologies that are relevant to the development of the application. Section 3.1 provides an insight into web and mobile accessibility guidelines. An overview of development tools for mobile development is presented in section 3.2.The strong and weak points of each framework is highlighted and their adequacy for the development analyzed in section 3.2.8. An overview regarding assistive technologies is included in section 3.3. Section 3.4 presents a summary of the ideas and conclusions.

3.1 Accessibility standards and guidelines

The *Web Content Accessibility Guidelines* (WCAG) are part of a series of web accessibility guidelines published by the *Web Accessibility Initiative* (WAI) of the *World Wide Web Consortium* (W3C). This consortium is the leading international standards organization for the Internet, which aims to foster compatibility and agreement among industry members in the adoption of new standards [91, 89]. The WCAG are a set of recommendations for making Web content more accessible. Due to the difficulties that people with disabilities face when using the Web, this group is the most in need of accessibility guidelines, which also apply to highly limited devices, such as mobile phones.

After the proposal of WCAG, which was primarily targeted for web content, a W3C working group presented the document titled *Guidance on Applying WCAG 2.0 to Non-Web Information and Communications Technologies* (WCAG2ICT) [48]. It provides informative guidance, meaning that it is not normative and does not set requirements. However, since it is developed by experts in accessibility, it is considered a good starting point for developers who need to meet specific accessibility requirements in their projects. The document encompasses a wide range of topics and presents some ideas on implementing the accessibility tools mentioned in the non-functional prototype report. The technologies and development tools used in the application's development were in compliance with the standards, resulting in increased usability.

3.2 Frameworks/Programming Languages

In this section, an overview of different solutions for mobile development is presented. Several native and hybrid development tools for Android are introduced. An overview for each one is provided; characteristics that distinguish each one are also detailed.

3.2.1 Overview

Mobile development development tools can be in three categories: native, mobile web app frameworks, and hybrid, which combine the features of both native and mobile web app frameworks. As the application's intended use is on Android smartphones, there was flexibility on whether to opt for a native or hybrid solution.

Some of the functionalities (such as increased color contrast and voice assistant) are present natively on Android and iOS. Frameworks that allow for the integration of the existing resources of the OS were preferred, as some of these tools are too complex and time-consuming to be implemented independently within the time-frame available for development. In this regard, native development is usually preferred when the objective is to run the app on a single operating system, taking advantage of its full capabilities [64].

Frameworks used for hybrid development can usually be more difficult to maintain and their design can be rendered obsolete if is not updated frequently. Additionally, hybrid frameworks make it more difficult to take advantage of the in-built tools that each specific operating system offers. When developing for different operating systems, hybrid frameworks are usually time-savers, as it enable to develop for several operating systems in a single project.

Progressive Web Apps (PWAs) are a type of application software delivered through the web, built using common web technologies including HTML, CSS and JavaScript. The advantage of using PWAs is that they do not require installation of software, and run across a wide range of devices. On the other hand, PWAs have limited access to the OS resources, so no true integration with the device is achieved. This limitation makes PWA usage inappropriate in the context of the application as full access to OS resources was needed, such as support software built-in to the system [14].

In software development, one of the biggest time-consumers is the need of frequently compiling and recompiling the code, which can take, depending on the processing power of the system, anything from several seconds to several minutes. Since this is an operation that has to be done very frequently, the cumulative amount of time spent on this is considerable and amount to loss productivity. One of the mentioned characteristics is "*Hot Reload*", and aims to solve that problem. Hot Reload translates to the possibility of keeping the app running and to inject new versions of the files edited at run-time. This way, the state of the application is preserved. Development tools which support Hot Reload or similar should be preferred.

3.2.2 React Native

React Native is a framework for developing web and mobile apps. Developed and maintained by Google, React is described as a "*JavaScript* library for building user interfaces". The programming language used is JavaScript. The library is updated regularly, and React Native allows developers to use React to create cross-platform native Android and iOS apps. "React components wrap existing native code and interact with native APIs via React's declarative UI paradigm and JavaScript". In other words, the app adapts to the operating system where it is running, using its specific functionalities [68].

Both Android and iOS provide APIs for integrating apps with assistive technologies like the bundled screen readers that are built into the operating systems. React Native has complementary APIs that allow for the integration of these resources[65].

3.2.3 Flutter

Flutter is a software development kit created by Google. With Flutter, applications for Android, iOS and other platforms can be developed. While other options rely on web browser technology or the set of widgets specific to each platform and device, Flutter uses a proprietary rendering engine to draw widgets [38]. The majority of Flutter's system, which includes the visual components, animations, among other elements, is implemented in *Dart* (an object-oriented language).

Flutter offers several accessibility widgets, allowing to create a highly accessible application. One of these, Semantics, allow the creation of a widget tree which allow the addition of annotations to give audio cues to visually impaired users [40]. Given the reasonable amount of documentation available for Flutter, the characteristics of this framework also make it a suitable option for the approach to the development.

3.2.4 NativeScript

NativeScript is an open-source framework to develop mobile apps. Like React Native and Flutter, development for iOS and Android is possible, resulting in native apps. Progressive is responsible for its creation and updating. NativeScript apps are built using JavaScript; other languages, like *TypeScript*, can also be used, if they can be transpiled to JavaScript. NativeScript supports the *Angular* and *Vue* JavaScript frameworks. Third-party libraries can also be re-purposed from several sources (such as *CocoaPods*, *Maven*, and *npm.js*) in their mobile applications, discarding the use of wrappers [81].

The NativeScript framework produces native apps. Thus, all vendor tools for impaired users work right out of the box. Most automation tools and frameworks should work with no additional bridging code or programming required. In addition to the entire set of native accessibility capabilities, NativeScript also provides cross-platform APIs which speed up the process of incorporating accessibility tools in both operating systems [20].

3.2.5 Ionic

Ionic is an open-source *SDK* for hybrid mobile app development.[1] It was created by three developers at *Drifty Company*. Released originally in 2013, it has been frequently updated and new releases have allowed for more options regarding framework choice. The first release was based on *AngularJS* and *Apache Cordova*. Later releases allowed for more freedom of choice regarding interface framework, such as *Angular*, *React* or *Vue.js*. Ionic components without user interface framework are also allowed [22].

Using Web technologies like *CSS*, *HTML5*, and *Sass*, Ionic provides tools and services for developing hybrid mobile, desktop, and *Progressive Web Apps*. Like other hybrid frameworks, Ionic uses the concept of shared-code among different builds in order to speed up the development process. In particular, mobile apps can be built with Web technologies and then distributed through native app stores.

Compared with the rest of the analyzed development tools, Ionic provides lackluster documentation, which makes it difficult to properly access its adequacy for the development phase. In addition, complaints made by developers in websites such as *StackOverflow* indicate frequent limitations to the implementation of accessibility tools in the apps.

3.2.6 Java

Java is an object-oriented programming language. It is considered to be an high-level language, and follows a class-based programming style. Compiled Java code can run on all platforms that support Java without the need for recompilation. Java was originally released in 1995, with James Gosling being its author. Java is updated frequently, adding new features or syntax changes, as well as security updates[70].

The Android SDK uses the Java language as the basis for Android applications. However, it does not use any of its standard *Graphical User Interface* or other established Java standards. Android, built on the Linux kernel, is written largely in C. Despite the complex technical structure of Android, which features several layers, the Java language and Java libraries can be used to develop software[75]. The Java virtual machine ensures that Android can be executed in many different hardware configurations[88].

3.2.7 Kotlin

Kotlin is an statically typed programming language. It is open-source, and was released to the public by *JetBrains* in 2011. Kotlin can be used freely, and from the developer side it also benefits from having extensive documentation provided on the web [45].

The main idea behind Kotlin is being able to develop apps for Android while using a more concise syntax, when compared to writing apps in Java. Due to its interoperability with Java, every Java compatible library can be used, and the same project can share Java and Kotlin code

[1] A set of development utilities for writing software applications[29].

(though in separate files). Compatibility with libraries used with Java is relevant in the scope of this project, as it is a requirement for the full-access to the multimodal capabilities of the Android OS.

There is some debate on whether Kotlin requires more time to compile and offers worse performance than writing the entire project in Java. Some studies have shown inconsistent results, with Kotlin being faster on some instances, with Java outperforming Kotlin in others. Regardless, the competitive advantage usually boils down to a difference of milliseconds, and therefore, should only be an issue in large scale projects.

One of the cases in favor of the use of Kotlin, is the amount of Java code which, although necessary, can be considered as being boilerplate. One of these examples correlates with the need to write *getters* and *setters*, in some cases abundantly if the project has many classes.[2] The following code exemplifies what a typical class in Java may look like:

```
1  class Book {
2
3    private String title;
4    private Author author;
5
6    public String getTitle() {
7        return title;
8     }
9    public void setTitle(String title) {
10       this.title = title;
11   }
12   public Author getAuthor() {
13       return author;
14    }
15   public void setAuthor(Author author) {
16       this.author = author;
17   }
18 }
```

Listing 3.1: Data class example in Java (extracted from [4]).

And as a follow-up a fragment of code which is obtained when the same objective is tried on Kotlin is presented:

```
1  data class Book(var title: String,var author: Author)
```

Listing 3.2: Data class example in Kotlin (extracted from [4]).

This example illustrates one of the main objectives that led to the creation of Kotlin: code that is simpler to write and read, when compared to Java. Other key difference when comparing

[2]Methods used to access and update the value of private variables, ensuring encapsulation. Encapsulation is a Java programming practice that states that "sensitive" variables should be hidden from users and other classes[93].

Kotlin to Java, is the approach of both languages to *Null safety*. To explain this concept, it is necessary to mention that in many programming languages, including Java, accessing a member of a null reference will result in a null reference exception. When conditions are not put in place to check for possible null values, an error is thrown, which may cause, in most scenarios, the application to fail. Writing code that accounts for these situations on a larger scale adds up to a significant amount of conditions which have to be thrown, thus making the code harder to read and comprehend.

The way Kotlin type system is designed aims to eliminate *NullPointerException's* from the code. The type system is able to differentiate between references that can hold null (nullable references) and those that cannot (non-null references). The way Kotlin operates, is that Nulls in Kotlin do not exist unless the developer states otherwise.

3.2.8 Frameworks/Programming Languages Comparison

Table 3.1 provides information regarding each development tool and allows for comparisons between characteristics of each option. The information listed is retrieved from the websites of the companies or entities responsible for its support. No requirements regarding the use of a specific framework or programming language were specified by CRPG representatives. Development tools were selected by taking into account each option's characteristics and then opting for the ones which proved to be more adequate.

In order to document the extensiveness of the documentation available with regards to the accessibility APIs two designations are used: "Extensive", which is applied to cases where it was possible to find official documentation regarding the use of specific accessibility APIs, and "Reduced", which denotes inconsistencies in the found documentation, such as missing documentation for a specific API.

Table 3.1: Comparison between development tools.

Framework	Code	Access to Device APIs	Hot Reload	Documentation coverage on accessibility APIs
React Native [3]	JavaScript	Full	Yes	Extensive
Flutter [4]	Dart	Full	Yes	Reduced
NativeScript [5]	JavaScript TypeScript	Full	Yes	Reduced
Ionic [6]	HTML, CSS, TypeScript, JavaScript	Partial	Yes	Reduced
Java [7]	Java	Full	Yes	Extensive
Kotlin [8]	Kotlin	Full	Yes	Extensive

It is possible to conclude from table 3.1 that all of the solutions presented seem to check one or more of the requisites for its use on the project. All of the development tools are also open source and free to use, so there is no restriction due to financial constraints (although Ionic has a free version and a paid version, which offers extra security and premium native plugins, among other features).

Due to the necessity of using the in-built resources of Android, there is a preference for native development. All of the solutions for native development present in the table can make use of the entire set functionalities built into the operating system; Ionic being an hybrid framework, uses the concept of shared-code to make development for both Android and iOS easier. The downside is that the code is not optimized for Android specifically, and access to the OS resources is limited. One of the main advantages of using native solutions is that, by using UI/UX components of the OS, Android updates should still allow the UI components to be consistent between the app and Android.

Flutter, like React Native, enables access to OS features. When compared to React Native, Flutter is a less mature technology, due to being released later. It also has not reached React Native in terms of stability. The adoption between developers, assuming GitHub contribution numbers, is also not as pronounced, which contributes to a smaller number of available packages and less support by the community. NativeScript offers almost all of the capabilities that are necessary to develop the app, however performance seems to be compromised when comparisons are made to React Native.

Java has extensive documentation on accessibility and inclusion, and is frequently updated. It also has the advantage of being the native Android language, which allows for the most extensive list of both Google and third-party packages among all of the analyzed solutions, as well as having an active community of developers that discuss topics related with it. It also eliminates the risk of possible issues when trying to integrate specific Android tools.

Several reasons led to discarding the use of React Native. While most of the needed features could be implemented by using libraries specific to React Native, some questions appeared regarding limitations to the integration of the intended multiple modalities of interaction. If no libraries were available to implement all the needed modalities, it would not be possible to change the programming language in a later stage of development. Documentation was also lacking in some accessibility subjects when compared to the official Android documentation for developers. After examination of the available options, the Kotlin programming language was considered.

Kotlin, like Java, provides faster performance than hybrid frameworks, although it is not possible to declare Java or Kotlin faster due to inconsistent results that were mentioned. When compared to Java, the amount of boilerplate code is reduced, improving readability, and the amount of bugs is also reduced as a side effect of the cleaner code writing and the approach of Kotlin to null references, amongst other improvements. Developing code in Kotlin also makes the occurrence of bugs less frequent by design of the language. Since the Android documentation provides documentation for every API that can be used, including accessibility, choosing Java or Kotlin guarantees that documentation on any Android API is guaranteed, and as such it can be considered a safer choice when compared to the other solutions which have more inconsistent documentation. After analyzing the advantages and drawbacks of each specific option, the decision was made to write the application using Kotlin as the selected programming language, as it checked all of the necessary requirements.

3.3 Assistive Technologies

Individuals with a considerable degree of disability may need to resort to devices which help interact with smartphones. Assistive technologies is an umbrella term which includes these devices, as well as processes and techniques used to improve the quality of life for people with incapacities [74]. For the purpose of this dissertation the focus was on smartphone-related technology, both software and hardware, which enable complete autonomy by the user in interacting with the device. In subsection 3.3.1, an overview of physical systems is presented. Considerations are also made on approaching the integration of their use with the application. Subsection 3.3.2 introduces the concept of screen readers and explains its uses.

3.3.1 Prosthesis and physical mechanisms

The use of prosthesis and other physical mechanisms increases the autonomy of the individuals who need them. The way these systems interact with the mobile device is different from the use of a human hand, and the application design should consider those differences. It is relevant to mention that many of these solutions are created for each individual, and they may differ to the point that could reduce the usability of the application.

Physical pointers are used by people who have limited use of their arms and hands. Some of them are designed to be used with digital devices, while others can serve other purposes (like painting on canvas) [95]. Physical pointers can be divided into three main categories:

- A head pointer (also called “head wand”): mechanism attached to the head with a helmet or headband, and can be used to operate a keyboard, touch screen, or switch device.
- A mouth stick: mechanism held in the mouth by the teeth, and can be used the same way as a head pointer.
- An adapted stylus: for example, one with a wide handle and straps to secure it to the hand, which can be used with a touch screen or instead of a mouse.

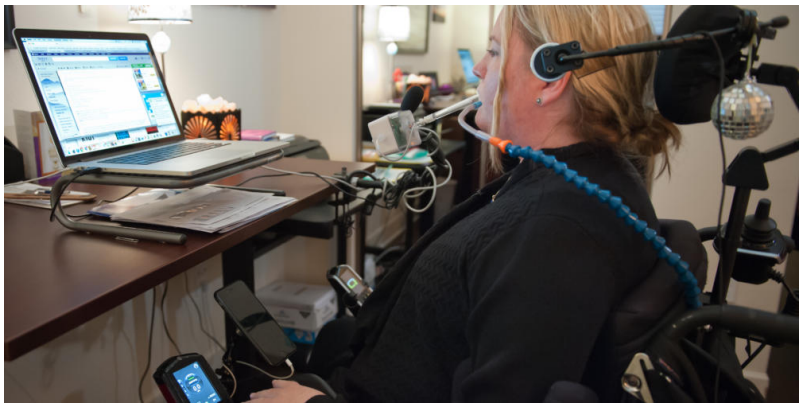

Figure 3.1: Example of mouth stick use for operation of a computer (extracted from [50]).

3.3.2 Screen Readers

A screen reader relies on software to translate on-screen information into electronic text. The information is then relayed to the user employing a text-to-speech tool or a refreshable braille display. Screen reader software is of the most importance for blind or visually-impaired users, which would otherwise have difficulty accessing the information on the screen on the device [26].

The two operating systems for mobile systems used by most smartphone users, iOS and Android, have their in-built screen reader software. *VoiceOver* is the in-built screen reader on iOS that interacts with objects in apps so users can drive the interface even if they can not see it [9]. *TalkBack* is Android's built-in screen reader. Like VoiceOver, TalkBack provides information on the status of the interface and possible commands, discarding the need to watch the screen. Google allows for the integration of this tool with third-party software, which simplifies development [41].

The application's access is also made more comfortable as the users do not need to install any additional software to have Talkback working. Talkback is available in Portuguese, which is crucial as users interact with the app in Portuguese.

3.4 Summary

The study of the state of the art found and filtered several standards related to mobile development with a focus on accessibility and inclusion. Section 3.1 detailed several standards such as *WCAG2ICT*, which improve accessibility in mobile applications and websites and could be helpful in augmenting the usability of the application.

Section 3.2 features an overview of Frameworks usage in mobile development; description of several frameworks is presented, as well as the accessibility documentation used to assess its capabilities for the project. Section 3.3 mentions some of the existing assistive technologies which may be necessary for the user interaction with the application, and how they affect and restrict the interactions between users and smartphones.

Lastly, section 3.2.8 compared the development tools, discarding the less adequate ones and allowing to identify Kotlin as the more adequate choice between the available options.

Chapter 4

Proposed Solution

This chapter describes the proposed solution and the approach to the development. Section 4.1 provides a technical overview of the application and justifies the thought process behind it. The Use Cases related to the application are introduced in subsection 4.1.1. The requirements, both functional and non-functional, are presented in section 4.2. A description of the system architecture is present on subsection 4.3. Section 4.4 provides the work plan for second phase of the project, including the tasks and an explanation of each one. Section 4.4.1 details the characteristics of each delivery and what is added at each iteration. The chapter ends with a description of the assistive technologies use survey and an analysis of the answers to the survey.

4.1 Platform Overview

The platform consists of a mobile application designed to run on smartphone devices equipped with an Android operating system release/version equal or posterior to Android 6.0 *Marshmallow*. The platform is centered on the clients at CRPG and thus all of the features aim to improve their daily life in the center. The management of real-life data and back-office features for the staff are out of scope for this project.

The main objective of the application is to aid the clients at CRPG in managing and keeping track of their daily events and serving as a memory aid through the use of visual and sound cues for notifying of the occurrence of events. As is mentioned in the survey at section 4.5, different clients have different needs which have to be accounted for. Short-term memory loss and impaired vision are among the main difficulties that several clients face. As such, the application provides tools and modalities of interaction that allows more users to interact. Figure 4.1 shows the Agenda screen with a list of the events happening at CRPG on a specific date. The application's design follows as closely as possible the visual characteristics of the non-functional prototype created by the design team, as mentioned in Section 2.1. Some technical limitations did not allow the application to fully resemble the original design, mostly due to restrictions and lack of flexibility when using the Material Design elements. The clients have access to the following features:

Figure 4.1: Agenda screen of the application.

- Selecting a date in the initial screen, through a horizontal scrolling list, which updates the information in the Agenda, Transports and Meals to be accurate for the selected date.
- Consulting CRPG-related events in the Agenda as a chronological feed along with all the events occurring on a chosen day and being able to click them to access more info about them.
- Consulting and inserting reminders, which can be customized regarding their type, frequency, alert type (sound, vibration or both). Due to the requirement which indicates that the application should delegate some tasks to already available apps featured in the operating system, the application requests the use of the native clock/alarm app present on the operating system. Editing/removing reminders after their creation is done on the native alarm app. Availability may vary depending on device model and brand, however "vanilla" Android features an app of this sort.[1] The use of an independent app for this task allows for visual and sound alarms to occur independently of the application being in use at the time of the alarm.
- Consult/request information regarding public transports or the transport options offered by CRPG, such as viewing timetables or calling the drivers of the CRPG transport for a specific date.
- Consult/choose a meal option from four different possibilities (meat, fish, diet or vegetarian), saving the selection and being able to access the chosen meal later in the agenda.
- A meditation area with the possibility of selecting different moods and playing audio files corresponding to the specific moods.

[1] Name given to unmodified versions of Android, released by Google as-is. Different smartphone brands may add/change/remove functionalities of the software developed by Google, according to their needs.

- Consulting text and voice notes in a list format, which can also have images associated with them, and having the voice notes be played directly in the list of notes.

4.1.1 Use Cases

The application has fourteen use cases, which can be accessed by CRPG clients. The use cases are separated into six main features. Table 4.1 presents the identifier, title and description for each use case.

Table 4.1: Identifier, title and description of use cases.

Use Case Identifier	Use Case Title	Description
UC01	Login	User inserts credentials and is able to sign into the app.
UC02	Select date	User selects a date from a list with the dates of the current month.
UC02	View selected date events	User sees a list of events in the Agenda screen.
UC03	Consult event details	User clicks an event in the Agenda screen, and details from the event such as name, type and additional information are shown.
UC04	Create new reminder	User adds a new reminder, defining type of reminder, frequency (single instance or repeated occurrence, alert type (auditory only, vibration only, or both) and associating a note with it.
UC05	Access public transport timetables	The user is able to access timetables of public transports which stop near CRPG.
UC06	Access CRPG bus schedules	User is able to request the CRPG transport.
UC07	Access custom schedule	User is able to consult a menu with the four meal options available for the next day.
UC08	Select meal option	User is able to choose a meal from four possible options.
UC09	Consult notes	User accesses previously stored notes.
UC10	Create text note	User inserts a new note with an associated title, image and text content.
UC11	Create voice note	User inserts a new note with an associated title, image and voice note.
UC12	Select mood	User selects the current mood from one of six options.
UC13	Play audio file associated with mood	The user can interact with a media player in order to play an audio file associated with the selected mood.

4.2 Requirements

This section presents the functional and non-functional requirements that were requested by the representatives of CRPG during the conversations. While this project was a follow-up to a finished design phase, some requirements changed over time due to requests of the CPRG representatives and clients.

4.2.1 Functional Requirements

The functional requirements of the app are described in the following list:

- **Create and view events on the Agenda:** the main feature on the application is the Agenda. The Agenda looks like a chronological feed and spans twenty-four hours, akin to the chronological feeds commonly used in other social networks. The feature welcomes the users to a screen in which a particular date can be chosen, and after confirming the chosen date the user is taken to the Agenda screen with the chronological feed for that specific date.
- **Create and Consult Reminders:** reminders are events that the users can set, and which trigger an alarm (which can take the form of a ringtone or smartphone vibration). The reminder feature allows the users to set the hours of the alarm, the frequency of the alarm (the application offers the options to set the alarm for the current day, the following day (these two options more suitable for single-instance events), for every single day (which can be useful for medication intake) and also a customized option, in which the user can select the weekdays in which the alarm should be set.
- **Access Transport Timetables:** one of the features should allow users to consult the timetables for both public transportation and CRPG's own transport system. Since the CRPG transport makes stops at Porto, Gaia, and in the residence of each one of the clients, it is necessary to display the bus stop time in each one of these locations; the name and phone number of the CRPG bus driver is also shown, and the user can contact the driver by pressing a button.
- **Consult and choose Meals:** CRPG clients can select a meal for a specific date, from a menu with four different options (meat, fish , diet and vegetarian); information regarding each dish is presented and users can select the option they prefer.
- **Meditation area:** users can select one mood between six available, and a corresponding audio file to the selected mood is played in the application.
- **Consult and Add Notes:** Notes are akin to reminders, but they are time-independent, in the sense that they do not trigger alarms and are mostly used to aid the users remember useful information. Notes can be of two types: text notes, in which the content of the note is plain text which can be edited using the keyboard, and voice notes, in which the content is a voice note recorded by the user.

4.2.2 Non-functional Requirements

Due to the special needs of the users, the non-functional requirements are comprised of issues regarding accessibility, data privacy and others as can be seen from the following list:

- **Accessibility:** the application should provide tools that help overcome the users' difficulties and obstacles, given their broad range of physical and cognitive capabilities/impairments

such as short-term memory loss and vision impairments. This objective is achieved by means of a *UI*/*UX* designed specifically with the needs of these users in mind, but also through the several modalities of communication between user and system. The modalities translate into the use of sound and visual cues to alert the user of events happening soon or to aid the user in performing an action in the application, as well as the use of Android accessibility tools such as Google's Speech Recognition software, which enable users to control the app using their voice, and TalkBack, which uses a synthesized voice to describe screens and actions on the app to the user.

- **Responsiveness:** the system should adapt the design to as many screen measurements and configurations as possible and be as fluid as possible. The system should also take advantages of the device own accessibility tools and SDKs, since developing some of the features that are present in the application, such as the alert functionality and the Speech recognition tools, would not be possible within the development time frame and would reduce the application's compatibility overall, while possibly creating slowdowns and draining the battery of the device.

- **Modular System:** the system must be capable of having new features integrated or having the existing ones adapted/removed in the future with minimal effort, i.e. the CRPG technical team may want to build a back office tailored to their needs, and as such, the system must be able to receive a new module and easily integrate it and make it available.

- **Data Security and Privacy:** data inserted by users should not be stored in a remote database and accessible by other users. In order to achieve this, data that is inserted into the app falls into one of two categories: public and private data. Public data refers to data accessible by the entire CRPG community, such as transportation schedules, meal options for a specific date and events that are aimed at all of the participants of a specific group or even all of CRPG clients. Private data refers to the reminders and notes that the user inserts into the application and which are stored on Android's internal storage (explained in more detail at subsection 4.3).

4.2.3 Technical Requirements Update

In the build-up to the start of the development process, contacts were made with representatives from CRPG, in order to confirm that all of the initial requirements were still adequate. It was also crucial that the representatives scrutinized and were in agreement to the application's technical characteristics and decisions. The main points of discussion were the following:

- whether CRPG could provide access to its existing infrastructure and data storage to the project or if a third-party storage service would be more adequate.

- implementing the *pathfinder* tool in the Transports section, which would require the use of the *Google Maps* platform.

- Integration with the *Microsoft Outlook API*, used by some members of the CRPG staff.

Regarding the topic of database hosting, since CRPG possesses an internal database, it would be sensible for the application to obtain its data from the existing database, with the added benefit that data would not need to be duplicated, saving time to the CRPG staff. After speaking to CRPG's IT department representative, the idea of accessing the existing database was discarded, given the lack of an API or adequate interface that could enable a stream of data between the application and the existing database. No documentation was provided regarding the data structure and inner workings of the information system used by CRPG. As a result, the structure of the database which was eventually implemented was designed during the development phase.

In order to prevent delays to the development of the project, it was agreed to store the data on a third-party server, but safeguarding sensitive data should not be stored in these remote servers. No additional requirements or information regarding the information system used at CRPG were provided. The solution was to use the Firebase platform and create a non-relational database from the ground up. This approach enabled a faster integration with the project and delegated data-storage and authentication responsabilites to Firebase, enabling more development time for the multimodal interaction module of the application. After discussing the possible solutions to each topic, the following decisions were made:

- the database would be stored on a server provided by a third-party service, since the information system used by CRPG did not have an API to establish communication with external systems.
- removal of the pathfinder tool, as it would require accessing *Google Maps* API, allowing an external entity to gather information regarding the user real-time location and adding financial costs which are associated with using this service.
- the Transports feature of the app was also modified: each user should be able to view the time when the CRPG transport would pass by their house and two predetermined locations (Porto and Gaia bus stops).
- integration with the *Microsoft Outlook API* was abandoned, and manual introduction of data was agreed to be the best course of action.

With this new set of information, modifications were made to the previous requirements, in order to make the application respond to the CRPG community most current needs. Other adjustments were made regarding the non-functional prototype. The text font used in the non-functional prototype, *Trebuchet MS Bold*, is trademarked and cannot be used without a paid license, which may need to be renewed after a fixed time period. In order to solve this issue, the solution was to use *Source Sans Pro*, an open source font family which can be used without restrictions. The typography is also similar to the paid font, so it was deemed to be an adequate solution. Because there were no High-fidelity designs for some of the screens, designs for screens with missing layout were based on the low fidelity prototypes or the available wireflow. An effort was made to keep the visual identity of the app consistent across all screens and features.

4.3 Software Architecture

The platform consists of a mobile application developed using Kotlin. The Firebase platform is used in the form of *Backend-as-a-Service*.[2] The Firebase platform ensures data storage through the *Firebase Realtime Database SDK*, and authentication capabilities through the *Firebase Authentication SDK*. As mentioned previously, some accessibility tools are natively built in the Android OS, such as the TalkBack text-to-speech tool. The application takes advantage of the capabilities of TalkBack, but also provides its own text-to-speech tool, as it was not possible to override Android's TalkBack code to provide more customized and contextual information to the user. Speech recognition is also integrated, enabling the users to execute contextual actions using voice commands.

Users log into the app, using personal credentials, of which there are two types, for identifying users as CRPG clients or staff. Data inserted by the staff in the database cannot be edited, only consulted, by the clients. Apart from the information which is made public by the staff, the private data that each client creates is not accessible by other clients. The staff can add or remove data from the database using the Firebase console. Due to the concerns regarding personal data, use and storage of sensitive information of each user is avoided. The data that is introduced by the users, such as reminders and notes, is stored in the form of app-specific files.[3] Media files, such as the meditation-related audio files, can be streamed from the database if necessary, or from the device's own storage. The app was also designed with the *SOLID* principles in mind, translating into a highly modular structure and easily adaptable in future implementations. A simplified overview of the system architecture can be found in Fig. 4.2.

4.4 Methodology and Work Plan

Regarding the implementation of the application, the *User-Driven Development* methodology was adopted. This translated into keeping close contact with the clients who volunteered to test the app and continuously listen to requests and feedback during the development process. Not only a channel of communication was established, in which the clients could suggest new features and voice any concerns, but a survey was conducted before starting the implementation, as well as user interview sessions following the first prototype delivery, and an usability test following the second prototype delivery.

Before starting the implementation phase, a study regarding the technologies needed for the development phase was conducted. This study coincided with the release of the survey for the clients at CRPG. Libraries necessary for the development of the app were studied, and chosen considering their adequacy in the scope of the project.

[2] Service to which some behind-the-scenes aspects of the mobile application development can be delegated, so more resources and time are available to write and maintain the frontend[16].

[3] Files that are meant for the application's use only, either in dedicated directories within an internal storage volume or different dedicated directories within external storage. These directories are recommended by the official docs to save sensitive information that other apps shouldn't access[33].

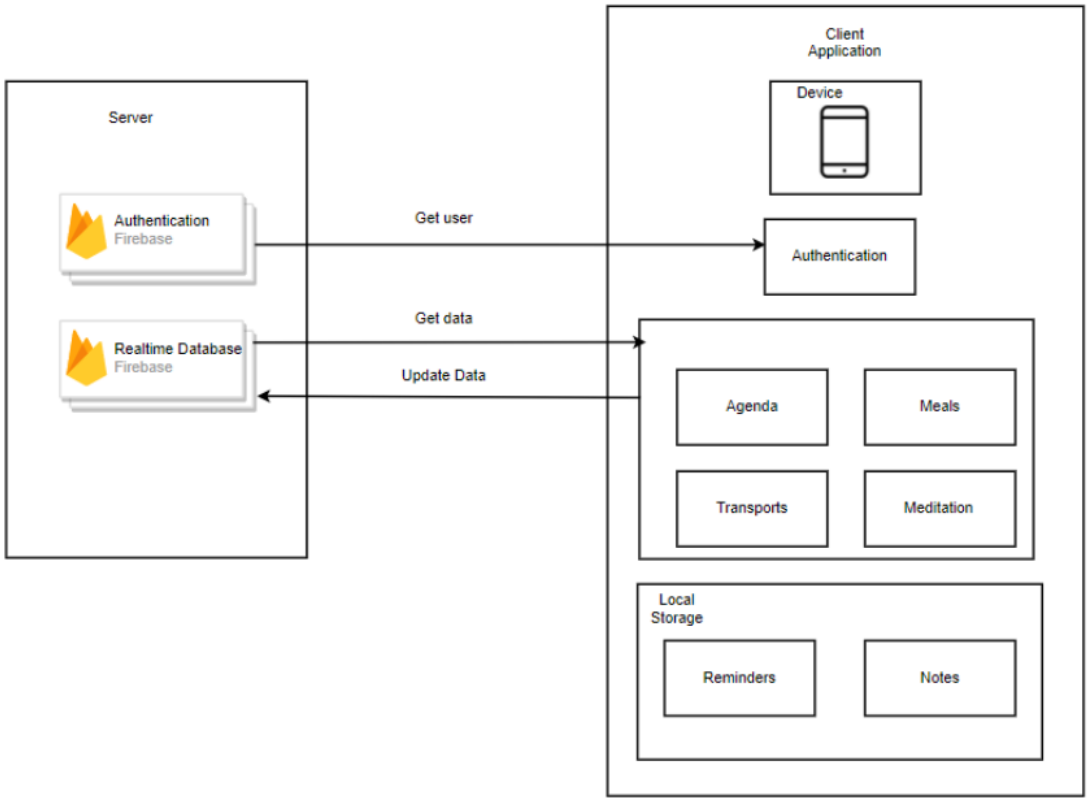

Figure 4.2: A visual representation of the architecture of the application.

The validation of the application by its users provided constructive feedback. These instigated further analysis of possible improvements to the application. Usability tests were conducted in order to quantify several parameters regarding the use of the application through different means of interaction. Metrics such as the time taken to perform specific tasks or errors made while performing the tasks were measured. More information regarding the different types of testing can be found in chapter 6.

In order to define realistic goals during development, it was necessary to break down the development of the project into smaller, more concise sections. The development of the first prototype began as soon as all of the data was gathered and the employed technologies had been studied. Even though the phases are mostly sequential, some tasks were done in parallel. For example, functional and accessibility tests were performed during all development phases, and not only after the development of code. This decision was made in accordance with what are considered to be adequate Agile development testing approaches[56]. The original scheduled work plan, with the time frame expected for each enunciated phase, can be seen in Fig.4.3.

The original work plan was modified: due to the coronavirus pandemic, no RAC program was ongoing at CRPG at the time of the originally planned first prototype delivery. It was necessary to allow users to interact with the first prototype under supervisions, in order to gather all the needed information. As a result, the first development period was extended for longer than originally

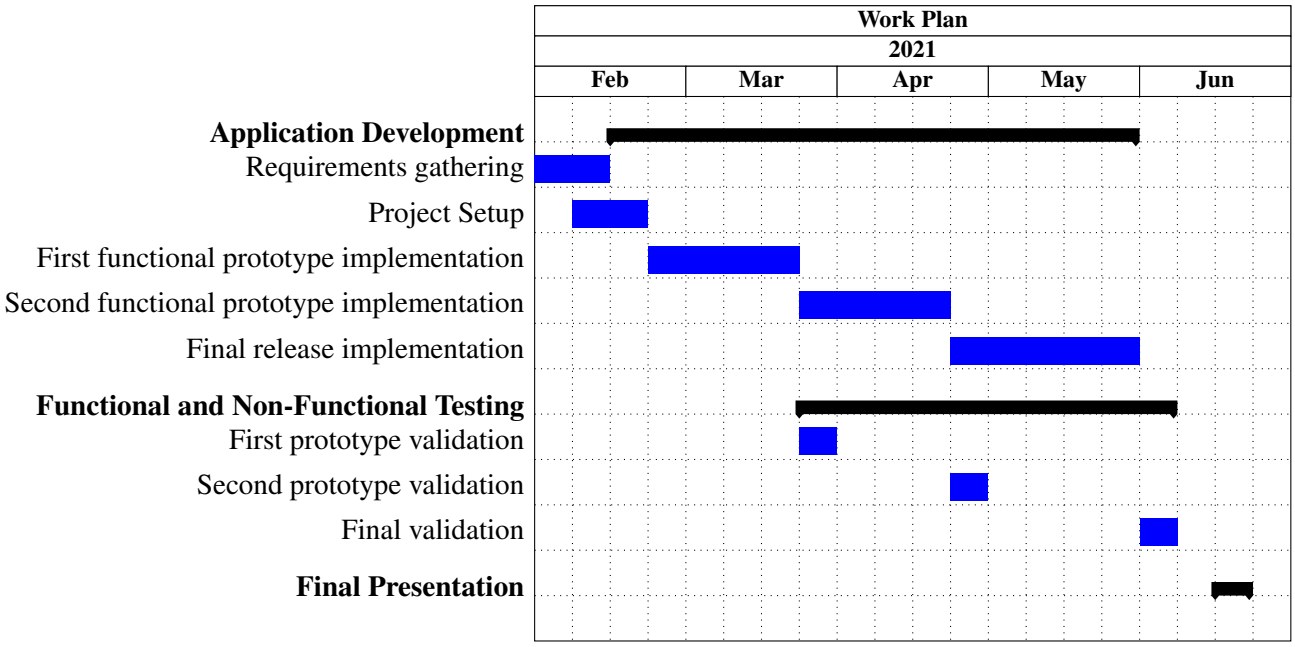

Figure 4.3: Visual representation of the original work plan (Gantt chart).

intended, and the idea of having two validations with users and an usability test was discarded, as logistically was not possible. The phases as described as follows:

- **PHASE 1 – Requirements gathering**: before the development process can begin, a survey is conducted to understand the capabilities and limitations of the mobile devices used by CRPG clients, and also identifying mechanisms used by the clients which may present additional challenges in the coding of the software.
- **PHASE 2 – Project setup**: includes initializing a working repository with version control implemented, in order to make revising changes easier, and also by installing and doing the setup of the programs and tools needed to develop the software.
- **PHASE 3 – First functional prototype implementation**: the development of the first prototype encompasses the implementation of the non-functional prototype into a minimum viable product, which features the six main features, featuring the design present in the hi-fi designs and integration with the Firebase platform.
- **PHASE 4 – First functional prototype validation**: validation of the prototype includes functional and non-functional testing. A session with CRPG clients is scheduled in order to gather feedback and thoughts on the application and on obstacles/weaknesses were faced while using the application.
- **PHASE 5 – Second functional prototype implementation**: the development of the second prototype prototype is mainly focused on the multimodal capabilities of the application,

such as contextual notifications with integrated action buttons, integration of TalkBack, customized voice messages through the use of a synthesized voice with Text-To-Speech capabilities, receiving voice commands from the user and also integration of contextual notifications with associated action buttons.

- **PHASE 6 – Second functional prototype validation**: validation of the prototype includes functional and non-functional testing. Usability testing was conducted with CRPG clients in order to perform measurements and gather quantitative information on the usability of each modality of interaction.
- **PHASE 7 – Final release implementation**: the development of the final release is mainly focused on the multimodal capabilities of the application, such as contextual notifications with integrated action buttons, integration of TalkBack, customized voice messages through the use of a synthesized voice with Text-To-Speech capabilities, receiving voice commands from the user and also integration of contextual notifications with associated action buttons.
- **PHASE 8 – Final release validation:** :validation of the prototype includes functional and non-functional testing. Usability testing was conducted with CRPG clients in order to perform measurements and gather quantitative information on the usability of each modality of interaction;
- **PHASE 9 – Final presentation**: delivery of the dissertation and of the platform in its finished state;

The first prototype included all of the functionalities and not only the first two proposed. The participants were able to interact with the application in a session at CRPG, as soon as it was ready to be used. The first session of experimentation with the app was not a usability test. Instead, it was meant to gather validation and feedback from the users regarding difficulties, obstacles and general thoughts during the interactions.

After the implementation of all the features in the first prototype, the development of the second prototype, which focused on the multimodal capabilities of the app, occurred during a much shorter development period. The development period for this phase was of four weeks, compared to the eight weeks of development time for the first prototype. Fig. 4.4 illustrates the updated work plan.

4.4.1 Deliverables description

Two functional prototypes were developed to allow for user testing as soon as possible. The first delivery implemented all of the features aimed at the users. At this phase, interoperability with other assistive technologies or alternative modalities of app interaction was not present. The second delivery featured all of the functionalities, including authentication. Adding to the application's capabilities, interoperability with other systems such as Text-To-Speech, Speech Recognition services and Android's notification system was implemented.

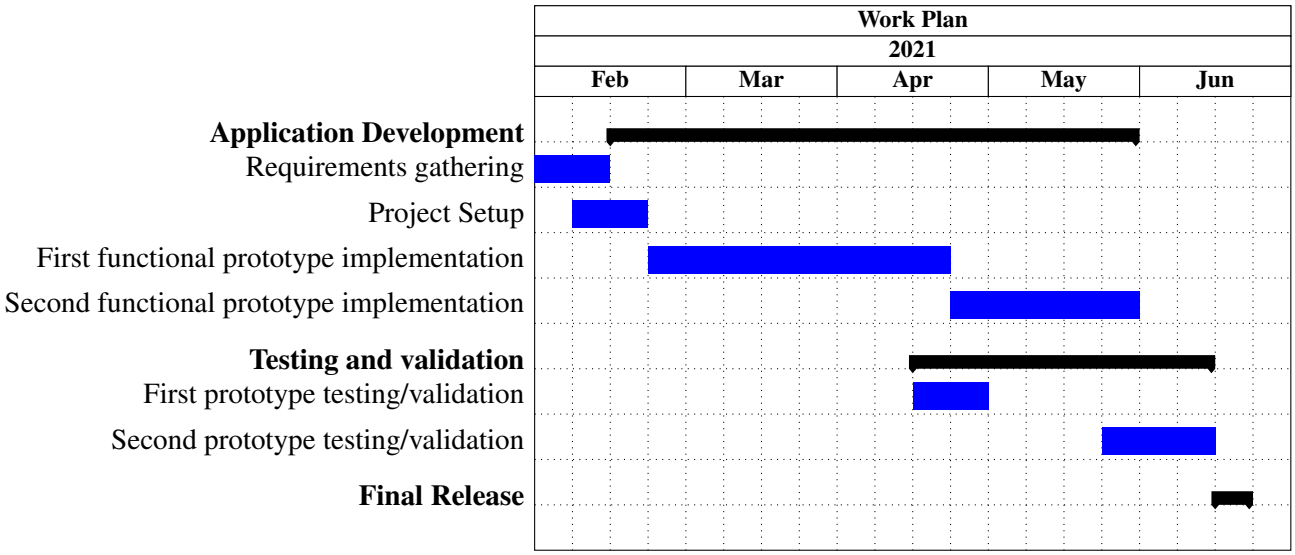

Figure 4.4: Visual representation of the updated work plan (Gantt chart).

The feedback provided by users allowed for small adjustments and improvements to be made to the second prototype. The overall structure did not suffer any drastic changes, and no new features were implemented, as users had already interacted with the app with all of the planned features integrated. The last period of development was also focused on troubleshooting, bug fixes and minor adjustments that improve the user experience overall and increase stability on different devices.

4.5 Assistive Technologies Use Survey

A survey was conducted to understand the background of the participants and their interaction with mobile devices. The survey was shared with a total of twelve CRPG clients. Overall, eight participants answered the survey. From this group, five were a part of a RAC program which had already finished at the time of writing this dissertation. The rest were part of the program which started with the development of the application already ongoing.

As shown by table 4.2, the participants were inquired regarding the following topics: age, sex, operating system present in the participant device, identified needs, and use (or lack of) assistive technologies needed to interact with their smartphone. Five participants used phones with the Android OS, with the remaining three using iPhone devices with iOS.

The answers to the survey can be summarized as follows:

- All of the participants used the touchscreen integrated in the smartphone (no participant reported using any physical device or mechanism to facilitate interaction with the smartphone).

Table 4.2: Background of the CRPG clients who answered the survey (P).

Questions	P01	P02	P03	P04	P05	P06	P07	P08
Age (in years)	43	46	60	50	40	55	51	27
Sex (M/F)	M	F	F	F	M	M	F	M
Touchscreen/Peripheral use	Touchscreen	Touchscreen	Touchscreen	Touchscreen	Touchscreen	Touchscreen	Touchscreen	Touchscreen
Assistive software	Reminder app	No	No	No	No	No	No	No
Used Operating System	iOS	Android	iOS	Android	Android	iOS	Android	Android
Identified necessity	Vision impairment	Memory impairment	-	-	-	-	-	-

- Of the seven participants, five did not report any obstacles or difficulties regarding interaction with smartphones.
- Two participants identified specific obstacles when interacting with the devices: short-term memory loss was identified by one participant, with another mentioning an impairment to the vision.
- he participant who mentioned short-term memory loss also indicated the use of a memory aid software installed on the device.

From the total of eight participants, seven showed interest in following and contributing to the application's development. From these seven, two had to eventually cease collaborating and asked to be removed from the group of participants. The group was left with five participants. These five volunteers actively provided feedback throughout the development of the application, and participated in the validation and usability testing sessions.

4.6 Summary

This chapter provides an overview the proposed solution. The functional and non-functional requirements presented in section 4.2 aim to help the community of CRPG while maintaining the high level of accessibility that is needed in order to respond to different users' needs. The representatives from CRPG were consulted on possible requirements changes with regards to the previous requirements gathering, and the changes actually enabled to streamline some of the features of the app. Section 4.2.3 enumerates these updates to the requirements, which were adopted during the development of the application.

Regarding the architecture of the solution, an effort was made to take full advantage of the accessibility tools that the Android operating system offered. Choosing Firebase as a platform to support the back-end of the application allowed for a quick integration into the system, and reduced time spent on back-end development. This approach allowed for more available time to develop the app's multiple modalities of interaction, which increased the accessibility of the application.

In section 4.4, the different stages of the implementation phase were presented. The modifications to the plan are also described, and he updated work plan is also shown. Given the different

focus of the first and second parts of the development phase, section 4.4.1 provided an analysis of the features and highlights of each functional prototype. Section 4.5 presented the results of the survey, which allowed to understand the backgrounds and characteristics of each participant regarding smartphone usage.

Chapter 5

Implementation

This chapter provides an in-depth view into the intricacies and decisions made during the implementation phase. A further study on the applied technologies and development tools is presented in 5.1. Section 5.2 details restrictions and limitations regarding characteristics of the application which had to be accounted for. In Section 5.3, an overview is provided for the first prototype. The second prototype is described in section 5.4. Section 5.5 details the structure of the project and explains the reasoning behind implementation approaches and decisions. A summary of the content ends this chapter in section 5.6.

5.1 Technologies and Development Tools

In chapter 3, several technologies have been introduced. Conversations with CRPG staff allowed for discussing implications of the use of different technologies in the development. As a result, further conclusions were drawn, allowing for a better assessment of which technologies to adopt for this project.

Third-party libraries were used in cases where developing the features from zero would be too time-consuming and would negatively impact the time period available for the multimodality part of development. By reducing the time spent on the development of the more basic features of the application, more time was available to focus on the development of the interaction modalities.

Before starting the development, the project setup was done. The *integrated development environment* (IDE) used was *Android Studio*, due to its mainstream usage and provided support for building Android apps[31]. In order to maintain an index on changes made to the application and provide better documentation, *Github*, a software used for *version control* was used[30]. [1]

5.1.1 Firebase

Firebase is a mobile and web application development platform created and maintained by Google [34]. While Firebase is comprised of a number of various services with differentiating offers, the services to be used here are *Firebase Storage* and *Firebase Authentication*. Firebase offers the

[1] System that records changes to a file or set of files over time so that specific versions can be recalled later[2].

choice between a free version and a paid version. The paid version allows for more storage and requests to the provided services. Due to the fact that the application is expected to be used by a small number of users, it is expected that the free plan should suffice. If needed, an upgrade to a paid plan with more features can be done later.

Firebase Storage is able to store data in a remote server, accessible anywhere, to be accessed via mobile apps. The data is structured as tree of information in JSON files. The Firebase SDKs for Cloud Storage integrates seamlessly with Firebase Authentication to identify users. A declarative security language is also provided, allowing for access controls to be set on individual files or groups of files. Opting for this solution makes public data available to all users, and private data is only by the user who created it.

Firebase Authentication provides backend services, SDKs, and ready-made UI libraries to authenticate users to mobile applications. It supports authentication using different types of credentials, such as phone numbers and passwords. Using Firebase in this project offers several advantages. For instance, being supported by Google infrastructure translates into reduced downtime. Due to redundancy and load balancing the performance is expected to be better than using lower cost equipment. On the other hand, the integrated security checks and validations add layers of security to the service.

5.1.2 Material Design

One of the concepts that were used in the non-functional prototype is the Material Design principles by Google. Material Design is described as a design language developed by Google[37]. In comparison with previous iterations of visual elements provided by Google, Material Design expands on "card" motives. It also enables responsive animations, transitions and depth effects which allow for more ways of conveying information to the user in an appealing form.[2]

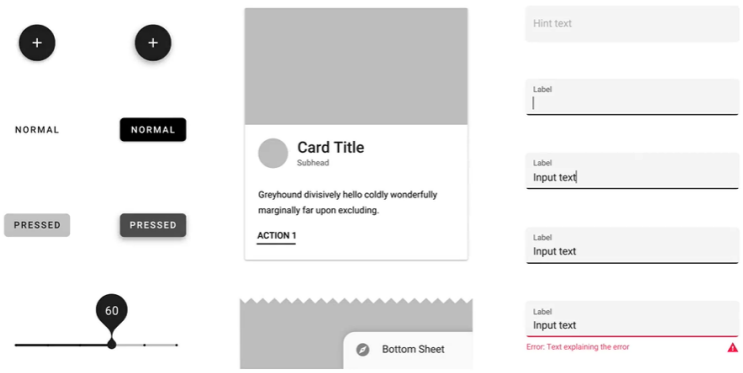

Figure 5.1: Illustration of some visual elements used in Material Design (extracted from [37]).

[2]Google published a video entitled *Making Material Design* which provides an insight into the thought-process behind this design language. Available at https://www.youtube.com/watch?v=rrT6v5sOwJg (last accessed on 12-02-2021).

5.1.3 JSON/GSON

JSON is a syntax for storing and exchanging data. It allows for lightweight data storage, and its use is less demanding than setting up and maintaining a local database [94]. *JSON* is the format that adopted by Firebase, and due to it also being a viable option to be used for local storage in the device, it was selected for use in this project. A JSON object contains data in the form of key/value pair.

Gson is a Java library that can be used to convert *Java Objects* into a JSON representation. Since it provides both serialization and deserialization, files with JSON items can also be extracted and turned into Java Objects. Gson can work with arbitrary Java objects including pre-existing objects, which facilitates working with unknown code [3]. Using Gson allows for the reduction of boilerplate code, and also makes interpreting the code easier, as less lines of code are required to achieve the same results.

5.1.4 Ktlint

During the development of large-scale software projects, it is common for some code to be written in a less than adequate way, or, for some reason, become obsolete due to new releases of libraries used in the project. Linting is the process of checking the source code for programmatic and stylistic errors. The use of linters aid developers to deliver code that complies with the official documentation, guaranteeing that the code is on par with the best available practices and making it more comprehensible to other developers to read [78].

One of the more relevant *Linters* for Kotlin is named *Ktlint* [83]. This specific linter includes an in-built formatter, and has extensive coverage for a considerable number of cases. A *wrapper* for Ktlint, *kotlinter-gradle*, was used, as it did not require any special configuration and was ready to use.[3] Ktlint set of rules include:

- Uniform identation across all files.
- Discard use of semicolons.
- Remove wildcard or unused imports.
- Consistent spacing and order of modifiers.

5.1.5 Google Voice Search

Google Voice Search is a product which allows users to say voice commands, and have them registered by the service [19]. It is an online service, which means that it is not necessary to install any language packages or download any application for it to be used. When the service is activated, the commands are sent for Google's servers for processing and the result is sent back to the device.

[3] Entity that encapsulates (wraps around) another item to hide the complexity of the underlying concept[8].

Despite the good results obtained in the commands recognition during testing, this solution has drawbacks, mainly regarding privacy. Google can have access to the words uttered by users, since these are processed by the company servers. Offline, open-source alternatives were investigated, but none had convincing results and met the requirements needed. One of the main issues with using offline speech recognition tools is the need for voice packages. These packages are usually in the hundreds of Megabytes in size, and present slower performance in comparison to the Google alternative. Adding to these drawbacks, tests with this library did not prove to be as accurate when compared to the alternative. After considering the pros and cons of each option, the decision was made to use Google's SDK. In order to facilitate this task, a wrapper for the Google Voice Search SDK called Android Speech, by Alex Gotev was used[39].

5.1.6 Testing related dependencies

Some libraries/frameworks were used specifically for testing purposes, namely unit and UI testing. Table 5.1 provides the names, authors, descriptions and URL's of each one:

Table 5.1: Description of testing-related dependencies.

Dependency Name	Author	Dependency Description	URL
AndroidX.test	Google	Basic testing package which complements *JUnit*, providing features necessary to test the application.	https://developer.android.com/training/testing/set-up-project (accessed on 10-06-2021)
JUnit	Kent Beck, Erich Gamma, David Saff, Kris Vasudevan and others.	Framework which allows writing of repeatable tests. Instance of the *xUnit* architecture targeted at unit testing frameworks.	https://junit.org/junit4/ (accessed on 10-06-2021)
Roboelectric	Michael Hoisie, Jonathan Gerrish and others.	Framework that allows unit tests to be written and run on a desktop JVM while still using Android API. Robolectric handles inflation of views, resource loading, and provides emulation for code that's implemented in native C code on Android devices.	http://robolectric.org/ (accessed on 10-06-2021)
Mockito	Szczepan Faber, Brice Dutheil, Rafael Winterhalter, Tim van der Lippe and others.	Mockito allows for the creation of test double objects (mock objects) in automated unit tests for the purpose of test-driven development (TDD) or behavior-driven development (BDD). Mockito is open-source and suitable for Java/Android development.	http://www.methodsandtools.com/tools/mockito.php (accessed on 10-06-2021)
Espresso	Google	Testing framework that helps developers write automation test cases for user interface (UI) testing. Suitable for both testing of individual components during development cycles, as well as black-box testing.	https://developer.android.com/training/testing/espresso (accessed on 10-06-2021)

5.1.7 Other tools

Along with the development tools mentioned previously, there are other tools with a smaller scope which do not justify individual sections for each. This subsection presents and details all of the libraries and dependencies. Since the total number of libraries and dependencies is in the hundreds, some libraries are described as a group of dependencies (such as *android.** and *androidx.** packages), for conciseness sake. Table 5.2 details some of the third-party libraries used and their objective in the scope of this project:

Of the libraries mentioned before, *Scroll Date Picker* is covered under a *MIT* license, and the rest are covered by the *Apache 2.0* license. Both licenses are considered to have a relatively low number of restrictions to their use, making them permissive for most uses. Among these uses are

Table 5.2: Description of non-testing-related dependencies.

Dependency Name	Author	Dependency Description	URL
ExpandableLayout	Jaewoong Eum	Allows for the integration of responsive layouts which expand and collapse when an action is performed.	https://github.com/skydoves/ExpandableLayout (accessed on 10-06-2021)
ScrollDatePicker	Harry Whewell	Allows for a way to pick dates intuitively by having items associated with each date displayed sequentially on a horizontal list. The user can then scroll the list to find the intended date.	https://github.com/GastricSpark/ScrollDatePicker (accessed on 10-06-2021)
TimelineView	Vipul Asri	Allows for the display of a timeline, with support for a dynamic amount of content.	https://github.com/vipulasri/Timeline-View (accessed on 10-06-2021)
NaraeAudioRecorder	Seo Dong Gil	Recording and playback of audio files using the device microphone as the input source. Files are stored with the *.WAV* extension, as Android's media player can play this extension without the need for any plug in or additional library.	https://github.com/WindSekirun/NaraeAudioRecorder (accessed on 10-06-2021)
ImagePicker	Daval Patel	Allows for the selection of images from the device storage by opening the gallery and allowing the user to edit the photo before selecting it.	https://github.com/Dhaval2404/ImagePicker (accessed on 10-06-2021)
ExoPlayer	Oliver Woodman, Andrew Lewis, Ian Baker, Erdem Guven and others.	Application level media player for Android. Alternative to Android's MediaPlayer API. Allows playback of audio and video, both from locally stored or remote files. ExoPlayer supports features which are not provided by the default MediaPlayer API.	https://github.com/google/ExoPlayer (accessed on 10-06-2021)

included commercial use, modification, distribution, patent use and private use [5]. Files which replicate code, or present code modified from one of these libraries, include the original library license replicated in the header of the file, as mandated by the license types. Considering the few restrictions imposed by these two licenses, the adoption of the mentioned libraries was deemed adequate for implementation on the application.

5.2 Restrictions to Development

The aim of the application is to provide support for daily tasks of the clients at CRPG plan. The app will be centered on the clients of CRPG, but the staff is also able to share events and information with the clients. Initially, the approach of developing the application for both Android and iOS was considered, as this would allow the application to be truly accessible for an even higher number of devices.

As the requirements were gathered, and given the focus on multimodality and accessibility, it became apparent that due to the architectures of the different operating systems and their approach to accessibility features, it would not be feasible to use an hybrid framework, as using shared code among the two operative systems is only possible to an extent.

Given that the period allocated to development was of three months, deciding to develop for both operating systems would eventually consume time which would otherwise be allocated into exploring the multimodal features. As a result, and given the importance of the innovative and experimental viewpoint of the dissertation, the decision to develop for one operating system was made. Android was chosen as the target OS due to the representatives of CRPG saying most clients had Android in their devices.

5.2.1 Software Compatibility

As new updates and builds of the Android OS are released, additional accessibility tools are provided, allowing for a more inclusive experience to the users. It would make sense to develop an app which would be targeted at the newer Android versions, however at the time of writing this dissertation, less than 1% of the existing mobile devices were running Android *R* 11.0, the latest version available.

Since newer versions are not easily accessible by the general audience, it is necessary to develop the application to be compatible with older versions of Android, even if some limitations are present. After careful examination of the accessibility resources present in each Android version, it was stipulated that the minimum necessary version would be Android *Marshmallow* 6.0. This version of Android features *Voice Interactions*, not available in any of the previous versions. Being able to express voice commands is necessary to ensure the desired level of multimodality and freedom of choice for the users to interact with the device.

According to data available in Android Studio, 84.9% of Android devices worldwide are on this version or a newer one, which guarantees a very significant coverage for mobile devices. While choosing any specific version of Android will necessarily cause drawbacks (either due to varying levels of device support coverage or available features), the mentioned version of Android was selected since it was deemed as an acceptable compromise between features and compatibility.

Android *Marshmallow* was released in May of 2015. It is common that smartphone devices with Android (depending on the brand/model) receive updates in the following months or years after release. In practical terms, this translates into the likelihood of a significant number of devices released before 2015 being compatible with this operating system, and as a consequence, the application. Using Android's native capabilities also answers one of the mentioned topics in chapter 1, in which a reference is made to the removal of barriers to app use, which includes not having to download additional software, therefore augmenting the reach of the application.

5.2.2 Data Storage and Security

Since CRPG possesses an internal database, it would make sense for the application to obtain its data from the existing database, with the added benefit that data would not need to be duplicated, saving time to the CRPG staff. During the first meeting with a member of the IT department of CRPG, the idea of accessing the existing database was discarded, given that the existing infrastructure did not have an API or any type of interface that could be used to fetch data. Adding to this issue, no documentation was available regarding the data structure and inner workings of the information system used by CRPG.

In order to keep track of the dynamic data that appears as the result of data inserted by the users of the app, there is a need to make this data persistent. When the user closes the app, the data is stored and accessible the next time the application is used. As a consequence, the application synchronizes the offline and online data. In the Agenda feature, some events can be accessed

by the entirety of the community, and some are only accessible on the specific device they were created on.

The chosen approach was to use JSON files to store all of the local data. The application features both public and private data: public data is data that the CRPG staff can insert into the database, and which becomes available to the entire community. Data that falls under this category include events for the CRPG community, available meals and transport information. Private data is the data that each user inserts into the application. The latter is not sent to the Firebase Storage, and is instead stored in a JSON file locally in the device, in the *assets* folder, with a different file existing for each type of data (events, meals and transports).

5.3 First Prototype Overview

The first prototype can be described as a *Minimum Viable Product* (MVP), as interaction can be done using the touchscreen, but lacks authentication capabilities the and multiple interaction modalities. This section details the layouts for each feature, using labels to identify the visual elements present in each one. A video which showcases the application in motion, as well as the different layouts, is available as a complement to the textual description.[4]

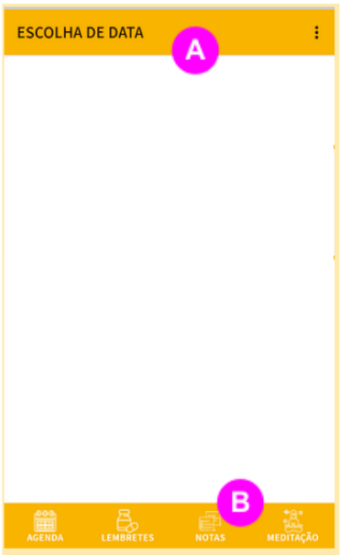

Figure 5.2: Appbar and Bottom Navigation Bar.

Some visual elements are consistent across the application (with the exception of the login screen): the *Appbar*, which is the bar that is located at the top of the screen shows the name of the current screen, as well as allowing users to return to the previous screen in certain circumstances (A). The top bar also features a three dots icon to the right, which does not serve any purpose in this release, but was integrated so that in future work a settings screen or an about page can be accessed through the three dots icon. On the bottom of the screen, the bottom navigation bar is

[4]CRPG Application video demonstration: https://drive.google.com/file/d/1RTVqFX-p5jNMjPq0flQ33R5PnLNWvfyB/view?usp=sharing (accessed on 2021-06-28)

always present (B). From the bottom navigation bar, the users can access one of four features of the application: the agenda, the reminders, notes and meditation (from left to right). The other two features of the app, choosing meals and transports, can be accessed by clicking on specific items in the Agenda screen. These items are differentiated by their title tag and differences in contrasting colours that grab the user's attention.

5.3.1 Login

The application's opening screen consists of a standardized login screen. The user must insert the credentials in two fields: one for the e-mail (A), the other for the password (B). After inserting the data, the user can click the "Enter" button to validate the credentials and access the app (C). If the validation of the credentials is successful, the Date Selection screen is shown. If the validation is not successful, an error message is shown in order to indicating the issue to the user. Fig. 5.3 presents the layout of the login screen.

Figure 5.3: Login Screen.

No sign-up screen is present in the application, as the registration of users done by CRPG staff. This was requested by the CRPG representatives, as it could not be possible for some of the users to register their own credentials. Since the number of clients is relatively small, it is feasible to have the staff inserting the users data at each new RAC program.

5.3.2 Date Selection

The date selection screen consists of a horizontal scrollable list, which the user can swipe to select a specific date (A). The user can confirm the selected date by clicking on the "Select" button (B). After the application validates the selected date, the Agenda screen is shown.

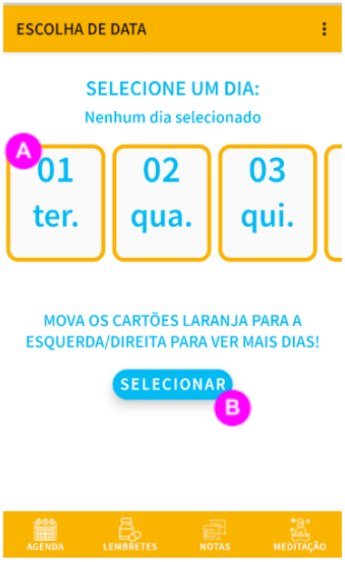

Figure 5.4: Date Selection Screen.

5.3.3 Agenda

In the Agenda screen, a list of the events, both public and inserted by the user, can be seen and clicked for more information or to access other features of the application. The events are ordered in a chronological order, starting with the earliest events in the morning, and continuing with the events that follow. To the left of the screen, a line with the hours of start and finish for each event are present (A).

CRPG caregivers commonly use *Post-Its* as a memory aid to the clients. Since *Post-Its* are a familiar concept to the users, the choice was made to make each event on the agenda list resemble the format of a *Post-It* note. Each item in this list represents an individual event and can be clicked (B). Each item has an image which is related to the event type, and also a title and preview of the content (C). Depending on the item, pressing it takes the user to another screen or launches a pop-up with more information (D).

Events can be classified into three types: related to transportation, meal selection or CRPG activities. If the user selects an activity-related item, a white pop-up is displayed, revealing all of the information associated with that activity. This information can include where the activity will take place in the CRPG premises, the person responsible for the activities, among any other info the creator of the activity may find helpful.

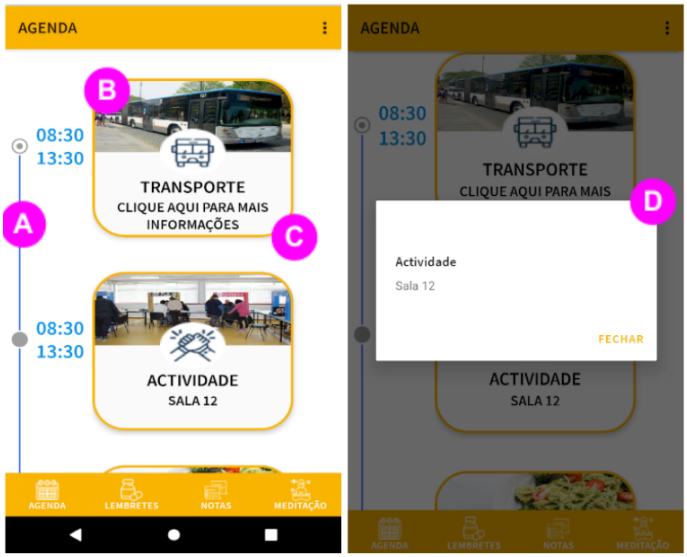

Figure 5.5: Left: Agenda Screen. Right: pop-up window with information associated with the selected item.

5.3.4 Create Reminder

When clicking on the reminder icon in the bar at the bottom of the screen, the user is shown the welcome screen for the reminders feature. After clicking the button with the pencil icon (A), the create reminder screen is shown.

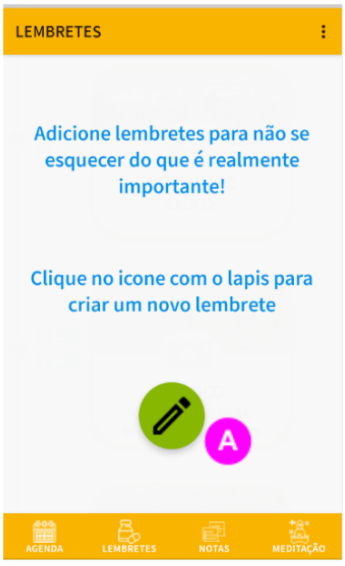

Figure 5.6: Welcome screen to the Reminder feature.

In the reminder screen, the user can set a new reminder that is accompanied by an alarm. This screen is composed of five sections, with each related to the characteristic/setting for the created alarm (B). All of the sections are expandable, meaning they can be opened or closed to facilitate

navigation on the screen. It is also possible to scroll vertically, exposing sections that may appear hidden due to the above sections being opened and taking a portion of the screen area.

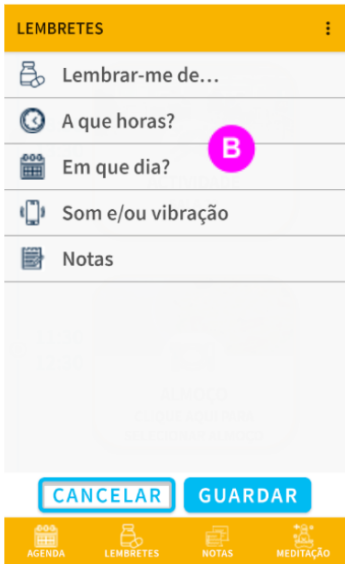

Figure 5.7: Create new reminder feature screen (all sections are shown closed/collapsed).

The reminder creation starts with the user selecting the reminder type in the corresponding section (C), from one of four options:

- Administer medication.
- Catch a bus/transport.
- Select meal option for lunch/dinner.
- Custom reminder.

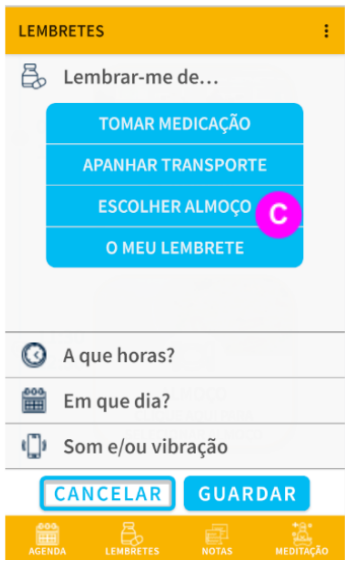

Figure 5.8: Set Reminder type section in the create new reminder screen.

After selecting the type, the application requests the user to select the time at which the alarm should be set (D). The frequency can also be defined in one of three options (E):

- One-time event.
- Daily occurrence.
- Custom alarm at specific weekdays.

Figure 5.9: Set reminder frequency section in the create new reminder screen.

The user can also define the type of alarm in the alarm type section (F): only sounds, only vibration, both sound and vibration. Finally the user can define a note for that specific reminder (G).

Figure 5.10: Set alert type section in the create new reminder screen.

The Android operating system natively features a pre-installed alarm application, which can be notified by the application. By setting the alarm in the existing application, the need for a duplicate alarm clock functionality inside the application is discarded. To confirm that the reminder was set successfully, a success screen is shown (H):

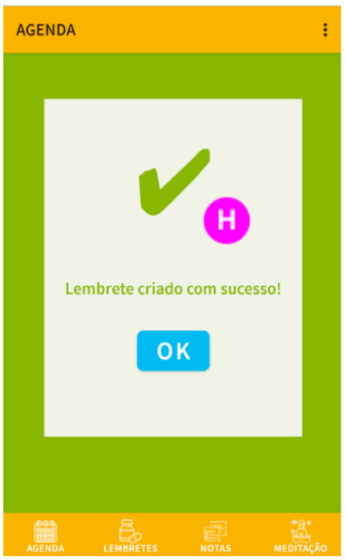

Figure 5.11: Reminder created successfully notification screen.

5.3.5 Transports

Just like the reminder screen, the transport screen can be accessed by pressing transport related items on the Agenda screen. When the user presses an item related to transports, a new screen appears allowing for the user to select the desired type of transports. The screen shows three different options:

- CRPG main bus (A).
- Users custom timetables and schedules (B).
- Information regarding public transportation (C).

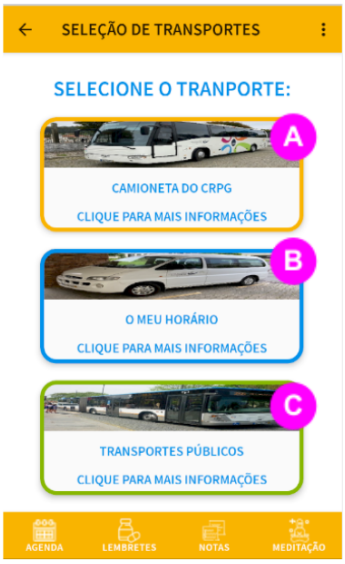

Figure 5.12: Transport Selection Screen.

If the user selects the CRPG Bus option, a new screen appears, with a dropdown list on top of the screen. From the dropdown (D), the user can choose from one of three bus stop options: the bus stop at Porto, Gaia, and the user's residence. When one of the options is chosen, the information is automatically updated. The user can then view the schedule at the selected bus stop. The screen also shows the driver's name and a button, which makes a phone call to the driver when pressed (E).

Figure 5.13: CRPG Bus Screen.

If the user selected the Public Transport option, the corresponding screen shows up. The user is greeted with a drop-down list in the public transport screen (F). The list includes four different bus lines that have stops in a one-kilometre radius, if CRPG is used as the centre. After selecting

the bus line, the information on screen is updated. Below the drop-down list, two arrows indicate the two possible directions of the line. The start and end points for each line are shown at the sides of each arrow. Two buttons, each one below each arrow, can be clicked to make the timetable specific to that line/direction appear (G).

Figure 5.14: Public Transport screen.

By clicking on one of the options, the user is shown time tables for the selected bus line (H). When the image is shown, the user can click on the image or use finger gestures to zoom in or zoom out on the picture (I). It is also possible to close the shown image by double-tapping on it. Since no API was found on the websites of the bus service providers, images with the timetables at the time of writing the dissertation were chosen to be displayed to the user as an alternative.

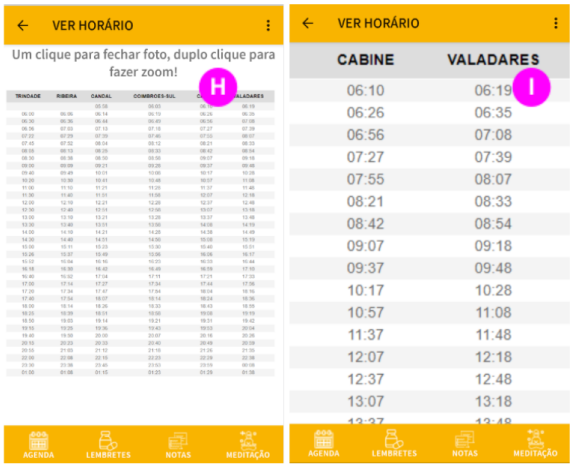

Figure 5.15: Left:Timetable viewer screen, which can be tapped once for zoom or twice to return to the previous screen. Right: Zoom on timetable.

If the user selects the Custom Schedule option, a new screen is shown, which allows the user

to see custom information that the CRPG staff inserted into the database. The screen features two groups of information: the first presents the schedules or relevant information for a specific date (J), and the second displays the contacts of the drivers which the user may need (K).

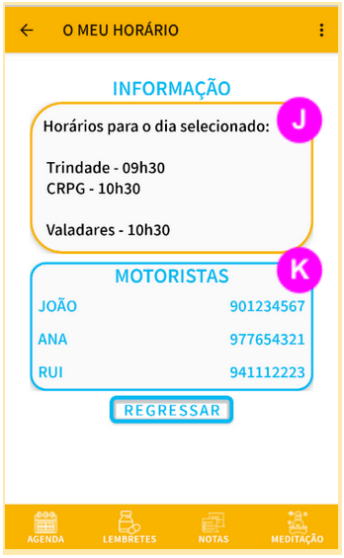

Figure 5.16: Custom schedule screen.

5.3.6 Meal Selection

If the user presses an item in the Agenda associated with a lunch or dinner event, a screen is shown with four different options: meat, fish, diet and vegetarian (A). When one of these options is pressed, a green check sign shows up inside the icon which was selected. If the user fails to select any option, a red error message shows up requesting the user to select the user before submitting.

Figure 5.17: Meal selection screen.

In order to confirm the selected option, the user presses the *Save* button (B). If the user selects a meal, saves the option and the process is completed with success, a new screen, similar to the success screen for the Create Reminder feature appears. This screen features bright colors, a green check box item and a message which reads “Meal selected successfully!” (C).

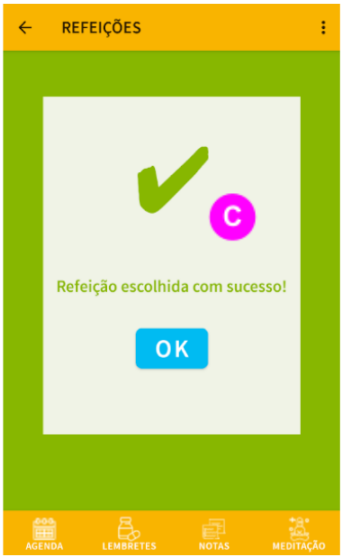

Figure 5.18: Meal selection success notification screen.

By clicking the *OK* button, the user is returned to the Agenda Screen. If the user scrolls to the item corresponding to the meal which information was updated, the item now shows the name of the selected dish, instead of the “Click to select meal” default message.

5.3.7 Meditation

In the meditation screen, the user is greeted with six options(represented by buttons) related to different moods (A). The user can select one of the following depending on current mood:

- *Confident*.
- *Happy*.
- *Relaxed*.
- *Loved*.
- *Mindful*.
- *Sleepy*.

Figure 5.19: Meditation Screen.

Figure 5.19 shows the meditation screen. By selecting one option and clicking the confirm button, the user is shown the meditation media player screen. The new screen is composed of two different sections. A centred frame at the top of the screen can be seen, containing a solid colour or an image (B). The selected mood is also shown here. Below the frame, a media player is shown with buttons, allowing the user to either play, pause, rewind or fast-forward the audio related to that mood (C). Each mood has a specific audio file associated with it. The audio file is selected by the CRPG staff and should be representative of that specific mood.

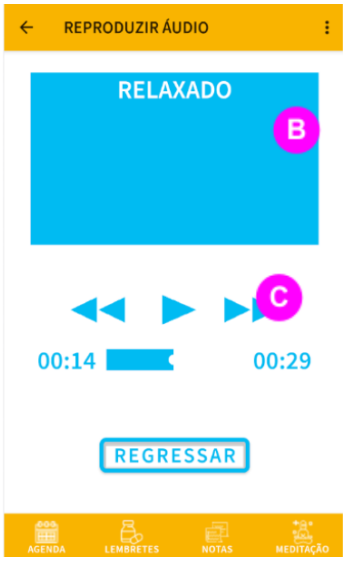

Figure 5.20: Meditation Media Player Screen.

5.3.8 Consult/Create Note

In the consult notes screen, which can be accessed via the bottom navigation bar, the user can access a list of notes added previously (A). Two types of notes exist: one is a text message, and the other is a voice message. Each type has an associated button, seen at the bottom of the screen, with each associated with the creation of a note of either type (B/C). Both types of notes can have an image file associated. When the user clicks a text note item, a pop-up dialog is shown, revealing the title and content of the note to their full extent (D). If the clicked item is of a voice note, a media player is displayed inside the item. The media player has three buttons, allowing the user to play, pause, rewind and fast-forward the voice recording associated with that item.

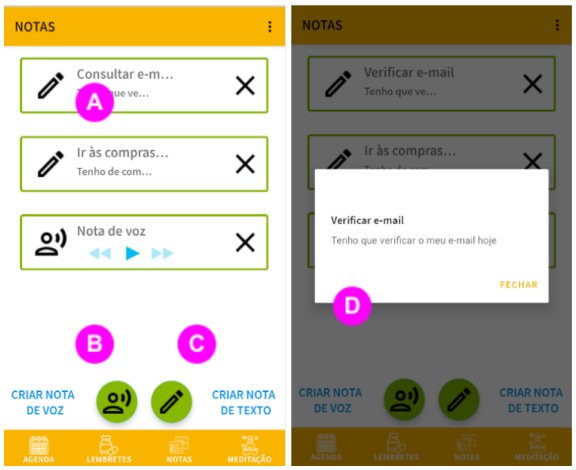

Figure 5.21: Left: View Notes feature screen. Right: Pop-up containing information regarding an item which was clicked.

The Create New Voice Note and Create New Text Note screens share a common module, related to the associate Image (E) and title of each note (F), since these features are the same for both types of notes. The changes between text notes and voice notes is related to the means of setting the content of the note: if the user is creating a text note, a text field shows up (G), and by pressing it the touchscreen keyboard is highlighted, allowing the user to insert the content manually. If the user is creating a voice note, three buttons appear (H), each one with a specific action associated: the button to the left of the screen initiates the recording when pressed; the center button stops the recording and the last one replays the last created recording. The user can then use the save button (I) to save the note.

Figure 5.22: Left: Create new Text Note Screen. Right: Create new Voice Note Screen.

5.4 Second Prototype Overview

For the second prototype, the main focus was on enabling the multimodality of the application through different means of interaction between the users in the system. Several modalities of communication are present between the user and the system. The list below details the different ways in which the application and the user communicate:

- The user can perform actions using the touchscreen of the device .
- The system can provide information about each visual element on all screens, through the use of TalkBack, which is mainly useful to users with impairments to their vision.
- The system can provide pre-determined audio messages indicating the user which voice commands can be said in order to perform certain tasks, which is mainly targeted at users who can see the screen but may have impaired dexterity or difficulties using the touchscreen.
- The user can utter voice commands (from a list of short predefined commands), which are then received by the app, identified, and an action is triggered depending on the uttered command.

Since Talkback and Audio Hints both use a synthesized voice, it is necessary to make a distinction between both features. TalkBack uses audio cues and feedback, as well as the device vibration feature, to inform the user on what visual elements are present on the screen, and what purpose they serve. The user can interact using specific gestures integrated into the TalkBack SDK and which are helpful in navigating or performing tasks in the app. The Audio Hints feature informs the user of the voice commands and aids in uttering the commands in a way that the software can recognize with more efficacy. The Text-To-Speech Contextual Audio Hints and Voice Recognition features that were integrated into the application are independent of the TalkBack feature and cannot be used simultaneously, since Google has strict limitations on alterations and use of the TalkBack services. As a result, the contextual information provided to TalkBack is mainly used

to aid users with impaired vision, and the Contextual Audio Hints/Speech Recognition are mainly targeted at users who can see the screen but may have limited mobility, and thus can be used to reduce touchscreen use to the minimum.

5.4.1 Speech Recognition/Voice Commands

One of the features that ensures enhanced multimodality is the use of voice commands by the user. Voice commands allow users to interact with the app, by navigating between features of the application or executing actions specific to each feature. Saying the name of a specific feature opens the fragment corresponding to that feature. When opening the app, the user is presented with a pop-up message which requests acceptance to the use of voice input in order to provide spoken commands. If the user refuses to accept this permission, the app will not capture sound and can also be fully controlled using the touch screen of the smartphone.

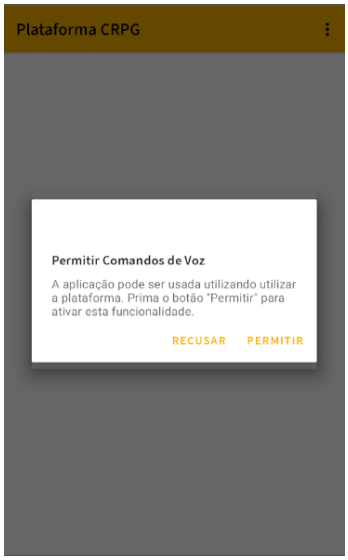

Figure 5.23: Pop-up windows requesting user preferences regarding the use of the voice commands feature.

If the user accepts the permission, the voice input is enabled whenever a feature that supports it is currently being used, in order to catch any voice commands and prevent the user from having to use the touchscreen. This option could add confusion to users and provide an additional barrier in interacting with the app. The use of voice commands is similar between the different features of the app, with slight changes which are indicated to the user through the Audio Hints feature.

It is relevant to detail what “becomes focused” means in this context. The screen is scrollable, and all of the sections, if expanded, occupy a vertical space larger than the screen dimensions. This means that the user needs to scroll to find a section if it is not visible. In order to overcome this issue, when the name of a section is uttered, the scroll occurs automatically and the section comes into view. Table 5.3 explains in more detail the usage of the service for the Create New Reminder feature.

Table 5.3: Description of an expected user flow using voice commands to create a new Reminder.

ID	Objective	Steps	Expected results	Visual Feedback	Auditory Feedback
1	Expand all of the sections.	1. The user says "Expand all".	1. All of the sections in the screen are expanded.	All of the sections which are initially collapsed are expanded, revealing the contents.	Sound beep after user command is registered and processed successfully.
2	Select the "Remember to Administer Medication"reminder type.	1.The user utters "Medication Reminder" .	1. The section becomes focused. 2.The "Remember to Administer Medication" reminder type is selected.	Corresponding button changing color.	Sound beep after user command is registered and processed successfully.
3	Set the reminder for 9h30 AM.	1.The user utters "nine and a half in the morning".	1. The section becomes focused. 2. The uttered hours are selected.	The set hour dials on the screen are updated with the correct hour and minute values.	Sound beep after user command is registered and processed successfully.
4	Set the reminder frequency to specific weekdays (Wednesday and Friday).	1.The user utters "Weekdays". 2. The user utters "Wednesday". 3.The user utters "Friday".	1. The section becomes focused. 2. The "Weekdays" option is selected. 3. The "Wednesday option is selected" 4. The "Friday" option is selected.	The buttons corresponding to the "Weekdays" option and each of the individual dates change their colors to highlight selection.	Sound beep after user command is registered and processed successfully.
5	Set the reminder alert type to "Vibrate only".	1. The user utters "Vibration only".	1. The section becomes focused. 2. The "Vibration only" option is selected.	The "Vibration only" button changes color and a check box icon appears besides the option.	Sound beep after user command is registered and processed successfully.
6	Add a note to the reminder.	1.The user utters "Note".	1.The section becomes focused. 2. The edit text field for the note is focused. 3. The keyboard becomes visible.	The exit text becomes focused and the keyboard is shown.	Sound beep after user command is registered and processed successfully.
7	Cancel current reminder creation.	1.The user utters "Cancel".	1. All of the selections are text edit fields are reset to their initial position.	All of the visible cues regarding selected elements and text in text fields, are reset or disappear.	Sound beep after user command is registered and processed successfully.
8	Confirm reminder creation.	1.The user utters "Save".	1. Validation of all of the sections occurs. 2. If validation is successful, reminder is saved.	If validation is successful, current screen stops being visible and a new screen appears, notificating the user of the success in saving the reminder. If the validation is not successful, an error message appears asking the user to correct any missing or invalid data.	Sound beep after user command is registered and processed successfully.

5.4.2 Audio Hints

After setting the voice commands options, another pop-up appears related to the Text-To-Speech Audio Hints feature. If the user selects the "Allow" settings, this feature is activated and a predetermined voice message is played when entering specific features. This audio message is only played the first time the feature is opened, in order to prevent constant repetitive audio messages to be played when using the app, which could result in a displeasing user experience and overwhelming the user.

Figure 5.24: Pop-up windows requesting user preferences regarding the activation of the audio hints feature.

The audio hints mainly serve the purpose of informing the user about the voice commands specific to that feature. It also aids users in understanding the correct words/formulation for the command in order to facilitate recognition by the service. As an example, when creating a new reminder, users are requested to select the hour of the day in which the alarm associated with the reminder should be activated. If the user wants to set the alarm to 21h30 using the touchscreen, it is expected that the user writes 21h30 because it's the natural way of doing it. When using voice commands, it is not natural for someone to say "twenty-one hours and thirty minutes". The way clients at CRPG said it during testing was "nine and a half in the afternoon", and as a result this is the expression that is recognized by the application for this specific hour setting. The transcript of every Audio Hints message in the application is shown in table 5.4.

5.4.3 Talkback

Activating TalkBack, Android's proprietary screen reader, requires enabling the associated option in the operating system settings, after making sure this feature is enabled natively. If TalkBack is enabled/installed on the device, a synthesized voice can be heard when using the app, providing

Table 5.4: Transcript of the audio hint messages.

Screen/Feature	**Audio Hint Message**
Date Selection Screen	Say the desired date as per the following example: "July 10th". Wait for the confirmation beep. After selection say "Select" out loud to confirm your option.
Transport Type Selection Screen	Say one of the following three options: "Public Transports", "CRPG Bus" or "My Timetable".
CRPG Bus Screen	Say the name of the desired bus stop. Wait for the confirmation beep and for the information on screen to be updated. Say "call outward driver" or "call return driver" in order to make a phone call to the adequate bus driver.
Public Transport Screen	Say the name of the desired bus line. After the information on screen is updated, say "Consult Schedule" to open the schedule.
Public Transport Schedule Screen	Say the name of the desired bus line. After the information on screen is updated, say "outward" or "return" in order to select the corresponding button.
Create New Reminder	Please say "expand all" to expand all sections, or the name of each section individually to expand it. If the section consists of multiple buttons, say the button's name out loud to select it. On the "define hours" section, say the desired time, as per the following example: "nine and a half in the morning" or "nine and a half in the afternoon". To add a note to the reminder, say "Notes" to make the keyboard appear. You should then use the touchscreen keyboard to add a note.
Consult Notes Screen	Say out loud "Create Text Note" or "Create Voice Note" to access the note creation feature.
Create New Text Note	Say "Upload Image" and wait for the selected image pop-up to appear. Select the desired image using the touch screen. Say "Title" or "Content" and wait for the keyboard to appear. You can then edit the fields using the touchpad. When finished, say "Select" to save the note.
Create New Voice Note	Say "Upload Image" and wait for the selected image pop-up to appear. Select the desired image using the touch screen. Say "Title". You can then edit the field title using the touchpad. To use the voice recording feature, please use the touchscreen. When finished, say "Select" to save the note.
Meditation Screen	Say the name of the mood you want to select.

information on each visual element and indicating the relevance of the visual elements for performing certain tasks inside the app. In order to request more detail on a specific item on screen, the user can use one-finger gestures. For example, if the user wants to focus on the next item, it is possible to swipe right on the screen, and the Talkback feature will read the content associated with that element. An example of the expected flow of a feature using Talkback is shown below for the Meal Selection Screen. When the screen is initially shown, the text at the top of the screen is read by Talkback. If the user swipes right using one finger, the next element is described, and this process is repeated until all of the elements are described sequentially. Fig.5.25 presents the visual elements of the screen, with a colored circle featuring the ID of each one. Table 5.5 provides the description that is uttered by Talkback when the user is swiping through each element.

Figure 5.25: Visual elements in the Meal Selection screen, each with an associated ID.

Table 5.5: Talkback description for each visual element in the Meal Selection screen.

ID	Talkback description
1	"Select your meal"
2	"Meal option"
3	"Fish option"
4	"Diet option"
5	"Vegetarian option"
6	"Save meal selection"

5.4.4 Contextual Notifications

On a specific times of the day, a notification is shown regardless of the application being currently open on the device, which can show messages related to meals, reminders and transports. To illustrate the purpose of these notification, an example for when the meal time is approaching can

be used. If the user has selected the meal option for that specific day and meal, a message is displayed with information regarding the selected dish. This feature was implemented after the clients at CRPG indicated that it was a common occurrence for some to have issues remembering what meal option had been selected, if any.

If the meal option has not been selected, the user is shown a notification reminding that one option should be selected, and that it is possible to access that feature by clicking on the notification. Whether the meal option has been chosen or not, both notifications allow clicking, which opens the application on the specific meal selection feature. This allows users to be reminded of choosing their meals or to be reminded of their selection before meal time. This approach removes the need to manually open the application and navigate to the Select Meal option menu, which would be more time consuming and inconvenient for the user.

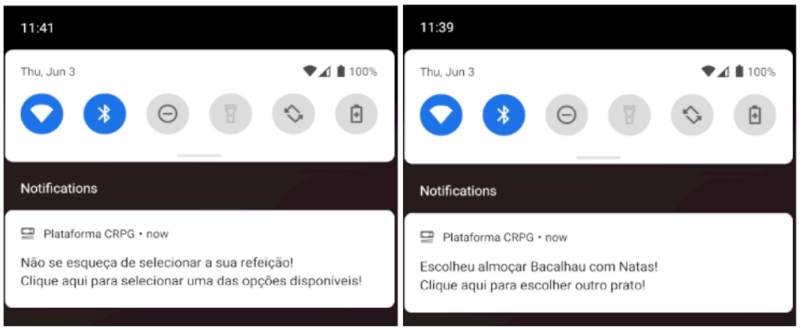

Figure 5.26: Left: Shown notification if no meal is selected. Right: Shown notification if a meal has been selected.

The same feature is also available for when the time for catching a transport is coming up. The application is proactive and sends notifications before the user may need them, and enables accessing specific screens inside the application by clicking the notification. Because the Transports area of the application has more screens and is more complex than the Meal Selection screen, three different action buttons are present on the notification. When clicked, the application is immediately opened and the screen associated with the clicked button is shown. Using these notifications is useful as they enable users to bypass both the date selection screen (because the date is obviously the current date, there is no need to have the user manually selecting it) and the Agenda/Transport Selection Screens. The user is only shown the requested screen at that specific time. The notification with contextual action buttons can be seen in 5.27.

By pressing one of the buttons, the application is opened on the corresponding screen, allowing the user to quickly access the information which is most relevant to the user at that point in time. The expected flow after pressing each of the buttons can be seen in Fig 5.28.

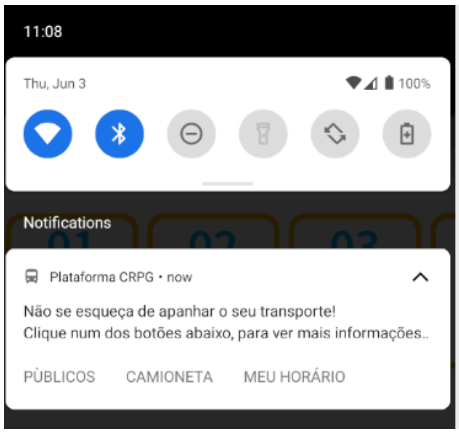

Figure 5.27: Transports notification. Clicking one of the buttons present on the notification opens a corresponding screen.

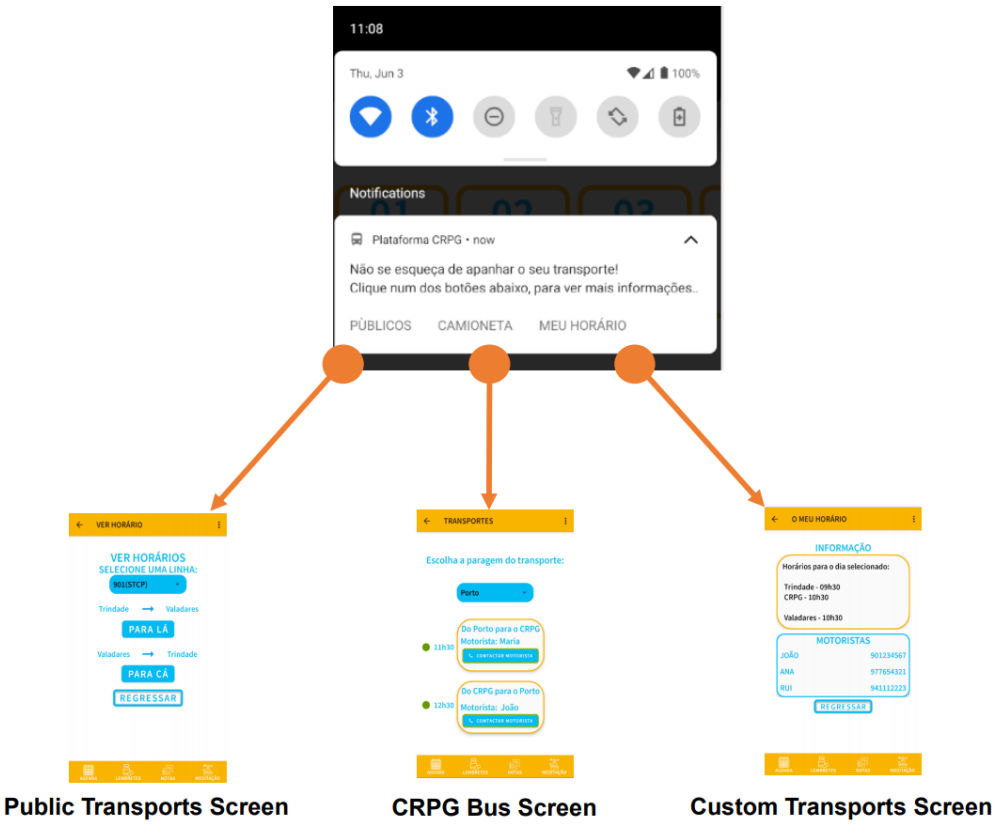

Figure 5.28: Diagram illustrating which screens are opened when pressing each of the Transports Notification buttons.

Both the transport and the meal notifications are independent of each other and of the reminders currently set by the user. While the meal selection reminder is shown at a fixed time (the default value is thirty minutes before lunch or dinner time), the transport notification will be shown thirty minutes before the user goes to or returns from the CRPG premises, if that information is available.

5.5 Project Code and Structure Documentation

This section details the decisions regarding both the structure of the project and some approaches/-solutions used from a technical perspective to reach the proposed objectives. This section focuses on the logic and approaches behind each feature; fragments of code related with each of the mentioned features can be found in Appendix A.

5.5.1 Architectural Patterns

Regarding the project structure, a *Model–View–ViewModel* (MVVM)[5] architectural pattern, in association with the Repository pattern were selected.[6] This approach separates the project into several layers, enforcing separation between the UI, front-end layer and the back-end logic, following the *SOLID* principles.[7] A visual representation of the MVVM model can be seen in Fig.5.29.

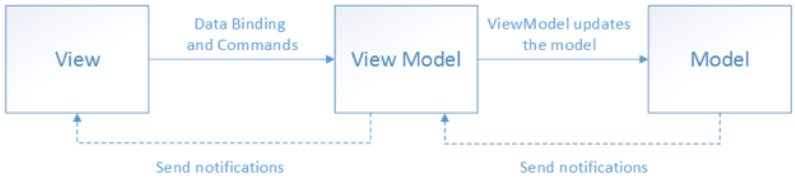

Figure 5.29: A visual representation of the *Model-View-ViewModel* architecture pattern(extracted from [15]).

The Model in MVVM refers to the domain model, which represents the structure of the data models used in the project. The platform has four models: events, meals, notes and reminders. Each one of these entities is represented by several characteristics. The View encapsulates everything that correlates to what is seen on screen. This includes associating actions to pressing specific elements on screen, updating the screen when needed, among others. Storing and and managing UI-related data is delegated to the *ViewModel*, in a lifecycle conscious way. The *ViewModel* class allows data to survive configuration changes such as screen rotations. Separating the app's UI data from the Activity and Fragment classes ensures the single responsibility principle [52].

The adoption of the Repository pattern translates into the use of repository classes. In order to isolate data sources from the rest of the app, the repository class is used. As an added benefit, it provides a clean API for data access to the rest of the app. Using a repository class adds an extra layer between the *ViewModel* and the database, and as such contributes to the centralization of the data access logic, so code maintainability is easier, among other advantages[86]. All interactions with the Firebase platform are dealt with by the Repository classes.

[5] Software architectural pattern that enforces separation of responsibilities. This approach states that the development of the graphical user interface (the view) should be kept separate from the development of the business logic or back-end logic (the model) This approach guarantees that the view is independent of any specific model platform[76].

[6] Use of a specific Repository class for abstracting data access, with the purpose of reducing duplicate code across modules and and enforcing the single responsibility rule of the *SOLID* design principles[77].

[7] See subsection 2.4.4.

5.5.2 File Structure and Description

This application can be defined as being a Single-Activity application[47]. In this approach, a sole activity is used (in this case, *MainActivity.kt*). This activity is used as a container that hosts the fragments associated with each feature of the application. For each feature available on the app, there is one or several Fragments associated. Each Fragment is associated with a screen, and has its lifecycle, independent of the Activity lifecycle. Using fragments instead of activities is advantageous in this project context as it frees up resources for the operating system when it is not being used, making the app perform better while using fewer resources. In multi-activity projects, and given the activities' more considerable demand for resources, the application may feel sluggish and slowdowns can occur more easily.

The global structure of the project can be seen in Fig.5.30. In the first folder, Manifest, the *AndroidManifest.xml* file is present. The manifest file describes essential information about the app to the Android build tools, the Android operating system, and Google Play. It also indicates the permissions which are needed in order to make specific features of the app available. Some of the features can be intrusive or harmful to the privacy of users. Every time a permission is required for a feature, a pop-up is shown to the user detailing the implications of granting permission and allowing the user to choose. If the user does not grant permission, the feature is not activated, ensuring the user is always informed about the consequences of enabling permissions. The permissions which are requested by the application are the following:

- *permission.SET_ALARM*: allows for the creation of alarms, used by the create Reminder feature.
- *permission.CALL_PHONE*: allows the application to make phone calls, used by the Transport feature, in which users can call the driver of the bus.
- *permission.RECORD_AUDIO*: allows the application to record the microphone of the device, used by the voice notes feature.
- *permission.READ_EXTERNAL_STORAGE*: allows the application to read data from the shared storage volume of the device.
- *permission.WRITE_EXTERNAL_STORAGE*: allows the application to write data to the shared storage volume of the device.

The core structure of the project is contained in the java folder, more specifically in the *com.plataforma.crpg* folder. The first two packages seen in the image, com (*androidTest*) and com (*test*) contain the test packages. The *expandablelayout* and *timelineview* modules are necessary for the correct functioning of these libraries in the project. The res folder contains several packages. The existing folders contained inside the res folder can be seen in Fig.5.31 and are described in more detail in the following list:

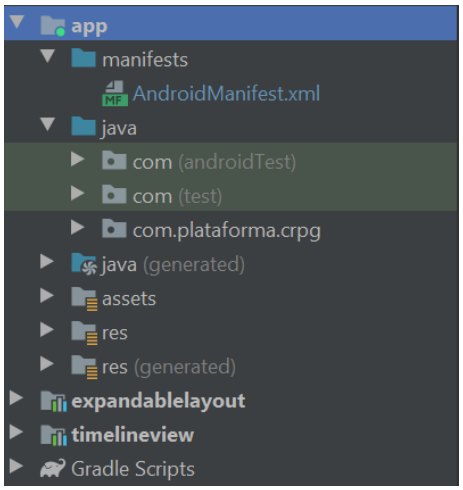

Figure 5.30: Overview of the project's structure.

- *drawable*: includes all of the visual assets, such as images, icons, *XML* definitions of several elements.

- *font*: includes *.ttf* files responsible for integrating the *Source Sans Pro* font into the project.

- *layout*: includes the *XML* layouts for all of the screens and fragments of the project.

- *menu*: includes two *XML* files, *bottom_nav_menu* and *menu_main*. The first is responsible for showing icons and descriptions of the features in the bottom navigation bar, and associating them with the correct fragment, which is crucial for the *mobile_navigation.xml* described below. The *menu_main* is responsible for the options in the top bar, which are not used but can be edited in future versions to allow for a settings tab for example.

- *mipmap*: includes icons of the CRPG logo in different resolutions and formats, in order to make the logo appear adequately in different devices on the application button in Android's application menu.

- *navigation*: contains the *mobile_navigation.xml*. This file is responsible for associating fragments with specific navigation ids, allowing for the execution of the correct transition when the user clicks on a button in the bottom navigation bar. Each fragment has a unique navigation identifier. This file can also handle transitions between fragments of the same feature. This can be illustrated with an example in which the user is currently visualizing the Public Transports Screen. By clicking the back button on a screen, the user is shown the Transport Selection Screen, which is defined in the *mobile_navigation* file.

- *raw*: contains the media files which the mediation feature may play. The platform can fetch *URLs* from the database to stream specific media files, but can also play files stored on the device.

- *values*: contains files which define several characteristics of the project, such as: the colors used in the platform, the styles (predetermined visual identities which guarantee visual consistency across the application), which colors should be predominant in certain areas, and also the *strings.xml* file, which contains all of the strings and text used in the platform.

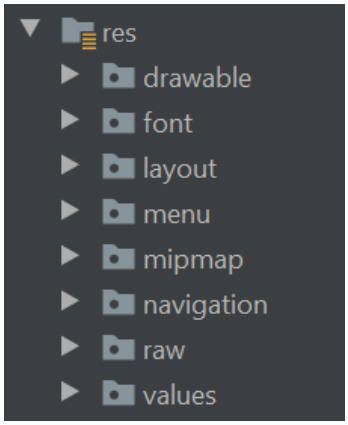

Figure 5.31: Structure of the *res* folder.

Inside the *com.plataforma.crpg* package, all of the packages containing the code, both related to the front-end and back-end, can be found. The project structure (inside the features package) consists of modules, with each one representing a feature of the application: classes are grouped by the features they represent. Every feature in the application has a corresponding package, in which a *Fragment* (View), *ViewModel* and *Repository* classes are always present. Other classes may exist depending on the feature, but the three mentioned are constant across all modules. This modular structure makes maintaining the code more manageable, as adding new features and modifying existing ones is simplified due to the separation between modules. Fig.5.32 shows the file structure of the project (excluding test packages).

The extension package has two classes: *DateTimeExtensions.kt* and *ViewExtensions.kt*. The first provides parsing capabilities for one of the used libraries, and the second provides helpful methods to be used by the *timelineview* library. In the package model, all of the data classes can be found. The purpose of data classes is to hold data while providing a structure to facilitate processing later on. Since the GSON library is being used, for some types of variables such as *EventTypes*, the *@SerializedName* annotation was used. This annotation allows the library to convert the type correctly into the JSON file and vice-versa, as without the annotation, the field would not be conveyed adequately.

Following the model package, the notifications package can be found. Since this package was implemented during the second prototype development cycle, this package is described in detail at subsection 5.4.4. The *utils* package includes two singleton classes: *CustomDateUtils* and *VectorDrawableUtils*.[8] The logic associated with the creation of these classes is related to

[8]Software design pattern in which a class is restricted to one "single" instance that restricts the instantiation of a class to one "single" instance. When one object is used in across the system, this pattern is helpful and reduces boilerplate code[27].

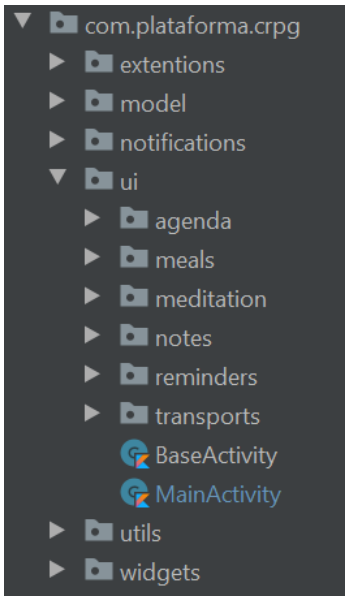

Figure 5.32: Structure of the *com.plataforma.crpg* folder.

the reduction of boilerplate/duplicate code. The first class provides classes such as *getIsLunchOrDinner()*, which indicates, based on the current time of day, if the the time of having lunch or dinner is coming up and is used by the Agenda and Meal Selection features of the app. The *VectorDrawableUtils* provides useful methods to support the *timelineview* library.

The widgets package is also related to the *timelineview* library, and provides the necessary methods and constructors to ensure the visual consistency of the visual elements provided by the library. In cases where it was necessary to share data between different fragments, a *SharedViewModel* was used. This can be seen for example in the Agenda module, where a *SharedViewModel* is used to store data between the Date Selection Fragment and the Agenda Fragment.

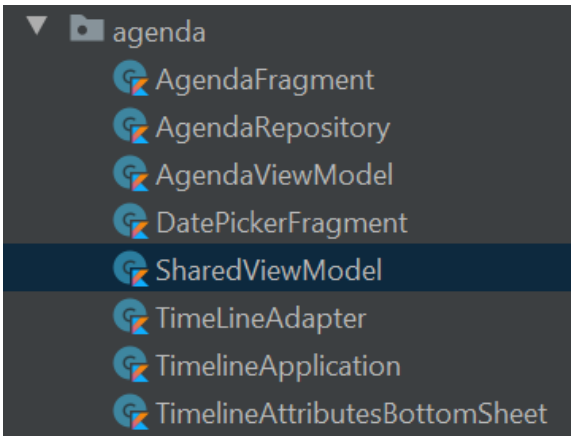

Figure 5.33: Structure of the agenda package (shared view model is highlighted in blue).

For the Agenda and Notes feature, *Recycler Views* with Adapters were used to allow for the display of large data sets in a list. These lists can be dynamically updated in real-time by creating,

editing or deleting items in real-time. Every time an item is added, modified, or deleted, the *notifySetDataChanged()* method is used, notifying any registered observers that the data set has changed. Using this method, it is possible to ensure that the list is kept up to date and will reflect any changes made by the user. When a Note is of the Voice type, a media player is attached to the item. If the item is identified as a text note, no media player is attached. Instead, a preview of the content (limited to thirty characters to fit the item) can be seen. Clicking the item opens a Dialog which exposes the full title and content of the Text Note.

5.5.3 First Prototype

The first prototype is a functional minimum viable product, as detailed in section 4.4.1. However, it does not integrate the several interaction modalities necessary to make the project more inclusive. The login screen and authentication capabilities are also not included.

5.5.3.1 Data Storage/Firebase Interaction

Reminders and notes are stored internally on the device and are not stored in the remote database. Storing data on the device ensures the safety of the data and prevents data sharing with third-party providers. Data related to transports, meals and community events on CRPG is stored in Firebase's Realtime Database servers. A setup was conducted on the Firebase console to integrate the Realtime Database features into the project (available after logging in as an administrator on the Firebase website). The setup consists of defining settings for the database and defining the data structure.The database is then associated with the project, and this is done by Android Studio automatically. The automation of this process discards the need for input by the developer. After initializing the database, it is now possible to insert, edit, remove data from it.

The database structure consists of calendar dates as the primary id, with data as its children branches such as Meals, transports and events. The data structure is relatively simple to make accessing the data as quick and easy as possible. The Firebase documentation recommends a flat hierarchy for the data. Given the relatively small scale of the app and the reduced number of users in each *RAC* program, the amount of data stored is expected to be relatively reduced. Using the relatively flat structure should make localizing and accessing the requested data faster. As mentioned previously, the MVVM with Repository pattern was used in this app. Fig.5.34 details the links between the common classes and the link between the Repository class and the Database.

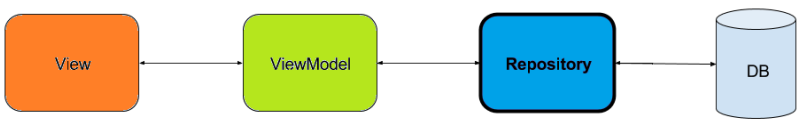

Figure 5.34: Common classes in each module and interactions between them (adapted from [51]).

The interactions with the database are handled by the repository classes present in each module. The structure of the repository classes is relatively similar between all of the existing instances: in all of them, three basic functions are shared. The first one retrieves data from the database, as a JSON, the second inserts data (parsed as a JSON entity). The third function is a listener which notifies the ViewModel in case any changes to the data have occurred while the application is open. If that is the case, the information visible on screen is updated the next time a Fragment is created.

5.5.3.2 Feature specific documentation

In the date selection screen, the application requests a date from the user. In order to do this, a horizontal scrolling list is available. The list presents all of the dates of the current month. When the user selects one date and confirms the decision, the date is then transmitted to the repository class of the agenda feature. In this class, a request is made to fetch all data available for the selected date.

The agenda, meal selection and transport information feature all use data from the remote database. In order to make sure that these features can all access the selected date information, a shared *ViewModel*, accessible by each of these modules, is used. Other features of the application are not dated, meaning they are not associated with a specific date. These features can be freely accessed even if the user has not selected a date on the initial date selection screen. After the date has been chosen, the only way to select another date is by closing the app and opening it again. It was important to not clutter the screen with buttons that could confuse the user, which led to this decision. After the date is selected, it is used to fetch the information associated with a specific date.

When the meal is selected by the users, the application then writes to the database the new order and increments a counter, depending on the dish that was chosen. This counter can then be used by the CRPG staff to know how many dishes of each type were ordered by the clients. In the agenda screen, the library *timelineview* was used. The first time the agenda is opened, events related to the selected date are loaded from the database. Regardless of the information about transports and meals having been uploaded by the staff, the *ViewModel* class automatically created items for transportation, lunch and dinner. The lunch and dinner items are set for a fixed hour of the day, usually some minutes before the specified lunch or dinner time. The transport item schedule is associated with the hours of passing by the nearest location to the user, if available.

In the meal selection feature, the user is presented with four different options for the date that has been selected. This information is stored on the user's device by pressing one of the options and selecting it. This information is helpful for the CRPG staff as well. For the staff to access this information, an increment counter request is also sent to the database. Doing so increases a counter associated with the total number of orders for each meal. This feature aims to help the CRPG staff monitor the number of orders for each meal option, which is useful when dealing with preparing meals for the RAC program clients.

Regarding the create new reminder feature, the *AlarmManager* service was used. After the user has created a reminder, the reminder is stored locally on the *JSON* file. An Intent is also sent to Android's default clock app. The advantage of using Android's resources to start the alarm is that there the alarm is not disabled (according to the documentation and testing) when the application closes, or the device is rebooted. This is an issue that can happen when defining the alarms by the application itself.

After the alarm has been set in the specific app, the application no longer has access to that specific alarm, so the user may need to disable or delete it if needed in the app itself. This approach was used because implementing an alarm feature from scratch would be too time-consuming and would not respect the guideline of using Android's resources and instead add features to the app which were already present in the operating system. The meditation scream is composed of six buttons. When the button is pressed, a bundle is created, which can be used to pass data between activities or fragments. The application then fetches the corresponding audio file, by finding the audio file or an URL in the database.

5.5.4 Second prototype

As mentioned in section 4.4.1, the second prototype is focused mainly on increasing the accessibility and interaction modalities between system and user. This section describes the implementation of each feature.

5.5.4.1 Modality settings

In order to allow the user to define the settings for the multimodal services, the *SharedPreferences* API was used. This API helps store a small collection of key values that can be accessed throughout the app and changed in real-time. According to the Android documentation, shared preferences are adequate for storing and handling user-defined settings and allow all of the fragments and activities to be aware of what services should be executed from the available communication modalities. When the application is opened, two dialog windows pop up, requesting the user to use the text to speech audio hints and the application's speech recognition capabilities. Depending on the response of the user to these dialogues, the Shared Preferences are updated. Depending on the settings that the user has determined when the application is open, fragments can access the shared preferences and determine which services will be used. The logic is the same across the fragments, but the voice messages and commands are specific to each fragment's context. If the multimodal option is selected, a predetermined voice message can be heard when entering the class. After that, a sound alarm can be heard, informing the user that the application is currently listening to the voice commands. For each fragment that supports these capabilities, there are five shared common methods related to both the Audio Hints and Speech Recognition features, detailed in the following list:

- *defineModality()*: ensuring that the correct services are deployed, by accessing the value of the SharedPreferences variables.

- *startTTS()*: handles the Audio Hints feature individually.
- *startSpeechRecognition()*: handles the speech recognition feature individually.
- *multimodalOption()*: ensures that the two services are run
- *performActionWithVoiceCommand()*: receives a string containing the words uttered by the user and performs a task depending on the content of the string.

A variable named *hasRun* flag is used to indicate if the fragment was visible/open before in the current application session. If the user has requested the audio hints to be activated, when the fragment is opened for the first time, the value is set to false, so the audio message plays. If the fragment is opened again without the activity being destroyed, the value is set to true. Because of the flag validation, only the voice recognition service is started, enabling the user to start talking as soon as he hears the beep without the voice message being played. The decision for this approach is to prevent repetitive voice messages and overload the user with information, which can be disorienting.

5.5.4.2 TalkBack

While Android provides some native voice input capabilities by default, these capabilities are limited to actions on the operating system and not inside third-party applications. Cues and descriptions regarding the existing visual elements in the platform are specific to this application. The use of content description attributes in the *XML* tag of each visual element, associated with the TalkBack functionality. The content description tag sets the description of specific visual elements' purpose. By defining the purpose of each element, the app can read it back to the user, enabling users with impairments to their vision to understand what each element on screen does. Not every element is necessary or relevant for the purpose of executing actions within a screen. Some elements serve a purely decorative purpose. For these cases, the tag "important for accessibility" allows the developer to indicate if the specific element is purely decorative or serves a relevant purpose for the correct functioning of the app.

```
1 <ImageView
2     android:id="@+id/botao_submeter_nova_nota"
3     ...
4     android:importantForAccessibility="yes"
5     android:contentDescription="@string/descricao_botao_submeter_nova_nota"
6 />
```

Listing 5.1: Code fragment associated with accessibility tags for the visual elements.

Special attention also needs to be given to the dynamic lists present in the Agenda and Notes features. Talkback is not able to identify individual items in dynamic lists, and their structure might not be easily translatable to sound descriptions. The content description can be added

programatically in the *onBindViewHolder()* method, ensuring proper description of each item and its corresponding contents.

```
override fun onBindViewHolder(holder: MyRatingViewHolder,
  position: Int) {
        val ratingData = myData[position]
        holder.ratingView.contentDescription = "Movie ${position}: " +
                "${ratingData.title}, ${ratingData.starRating} stars"
    }
```

Listing 5.2: Code fragment associated with setting content description strings to the items in a dynamic list binding.

5.5.4.3 Contextual notifications

The platform's notifications feature used several Android concepts such as Pending Intents, Broadcast Receivers, the *AlarmManager* class and lastly notification channels. Each of these concepts is explained in further detail in the following paragraphs. The module responsible for the notifications consists of four different classes/objects: *AlarmScheduler.kt*, *AlarmReceiver.kt*, *BootReceiver.kt* and *NotificationsManager.kt*. The structure of the notification module, as well as portions of the code were adapted from the *Pet Medicine Reminder* project, by Kyle Jablonski[43]. Fragments of code associated with the methods described in the below paragraphs can be seen in A. Pending Intents allow developers to define actions to be executed in the future and received by other classes of the project. In this case the Pending Intent is launched to be received by the MainActivity class.The *NotificationsManager.kt* class is the core class of the notifications module. It handles most of the logic that is needed for a notification to be sent. Responsibilities include:

- Creating the notification channels.
- Having different methods for the different types of notifications to be created.
- Creating the intents and pending intents that are needed for the notifications to be shown.

In order to provide a common visual and auditory experience for similar types of notifications, different notification channels can be set, each one with specific settings. In Android, a notification's priority is used to determine how much of the user's attention the notification should draw. The clients were receptive to the idea of the alarms using both visual and auditory feedback, and as a consequence, all the notifications are set as high priority/share the same channel.

Defining the notification priority to the highest level allows Android to show the notification on the lock screen and guarantees that a sound beep will be played to alert the user if possible. After a channel is created, the notification itself is created and a Pending Intent assigned to it. Depending on the type of notification, different actions can be executed depending on the action

buttons available or the objective that the notification serves. Each action button click or click on the notification has a unique pending intent assigned to it. Whenever the user clicks on any clickable element inside the notification, an intent is launched, which is then received by the *MainActivity* class which handles the resulting actions depending on the type of received intent.

A description of the purpose of each method is detailed in the following list:

- *MainActivity.kt*: listens for intents, and handles the actions to be executed when the user presses a notification/action button.
- *AlarmReceiver.kt*: BroadcastReceiver type class that ensures that whenever an alarm is triggered, the associated notification will be shown at the time of the alarm.
- *AlarmScheduler.kt*: allows for the notification to be shown repeatedly over time, using the AlarmManager feature present in Android was used and according to the frequency that is defined in the reminder.
- *BootReceiver.kt*: reschedules alarms after the application is closed or the device suffers a reboot.

5.5.4.4 Speech Recognition/Voice Commands

For the Speech Recognition service, the Android Speech library, by AlexGotev, was used [39]. This library is a wrapper for the package *android.speech.SpeechRecognizer* which enables voice commands to be recognized by the OS. The Speech Recognition feature requires being run on the main thread. It is necessary to declare a handler to start and stop the voice recognition feature when needed.[9] The Locale is also set to European Portuguese to make sure that both the Audio Hints and Speech Recognition services can both speak and receive voice commands in Portuguese.[10] Since Android currently supports Brazilian Portuguese but not European Portuguese, that is the language spoken by the synthesized voice. Later releases of Android may support European Portuguese, and the application is already prepared to use that functionally should it be available in the future.

After the service has been executed for the first time, it is executed again using the *handler.postDelayed()* method, inside *onSpeechResult()* method (called when the speech recognition is stopped). This allows for continuous voice recognition. Each time this service is executed, a string is created based on the assumption of words that the service assumes were spoken. The method *performActionWithVoiceCommand()* is then executed, which compares substrings of the character sequence received by the user input and compares it to a list of predetermined voice commands different for each fragment. If there is a match between the words, the associated action is started. Using a handler is necessary to use this service. The *sendEmptyMessage(0)* method is used to keep track of running tasks in the thread. The speech recognition service is stopped when the fragment is deleted/no longer visible by using the *handler.removeCallbacks(runnable)* method.

[9] A handler allows taks which require heavy processing to be relegated to specific threads, in order to not compromise perfomance and cause slowdowns to the app[35].

[10] The Locale object can be used to indicate a specific geographical, political, or cultural region[36].

5.5.4.5 Audio hints

To implement the Audio Hints, the *android.speech* package was used. In each class that used the service, a text-to-speech object was created. Each fragment has a specific predetermined voice message specific to that fragment by instructing the user on how to better take advantage of the voice commands capabilities of the application. This approach allows for contextual voice commands to be explained to the user while concise and containing only the essential commands for that context. *multimodalOption()* merges both services in the same method. However, it adds another logic layer to ensure that both services do not overlap each other and never occur simultaneously. If both services ran simultaneously, the speech recognition would pick up words spoken by the Audio Hints voice, possibly triggering unwanted events, with the added problem of having the speech recognition sound beep occur over the initial voice message. In order to overcome this issue, the *UtteranceProgressListener* (UPS) class was used. This class acts as a Listener for events relating to the progress of an utterance through the synthesis queue. By associating an ID to each Audio Hint message, the UPS tracks the message playing status. Several classes can be used:

- *onStart()*: called when utterance starts as perceived by the caller.
- *onDone()*: called when utterance stops.
- *onError()*: called when an error occurs during processing.

The *startVoiceRecognition()* method is only called inside the *onDone()* method, ensuring that the speech recognition is only started after the message has finished playing and no overlaps occur.

5.5.5 Final deliverable

After completing the usability testing, the application was nearing completion. Given the small time frame available after gathering the users' feedback, modifications done to the application were minor. In both prototypes, there were missing assets due to the unavailability of some descriptive images. For example, the images related to the CRPG transports were not available before the final user assessment. When the images were available, all of the image assets were added correctly.

Another focus during the final sprint was on the optimization of the app. Using the "Inspect Code" and "Code Cleanup" options on Android Studio, an analysis of the code is performed, and possible optimization points/improvements are identified. This analysis encountered over two hundred snippets of code which the IDE identified as the subject of possible improvements. These suggestions were analyzed individually since some of them could compromise the normal functioning of the application. The types of identified issues were, among others, the following:

- Incorrect *Android Resources* Validation.
- Unused resources.
- Deprecated API usage.

- Performance optimizations.

The analysis of these issues allowed for the optimization of the performance of the application, while reducing unused assets and cleaning up the code. The IDE also optimized the imports using the “Optimize imports” option.

5.6 Summary

This chapter presents an overview of the implementation process. Technologies such as frameworks and libraries were described, and justifications for their use were provided. Conversations with CRPG representatives and clients throughout the process led to integrating new tools and technologies not considered before.

A description of the work done for each prototype is detailed, accompanied by images representing the screens for each feature, allowing for the assessment of the faithfulness to the non-functional prototype design guidelines. While the initial design was maintained overall, several requests by the staff and clients of CRPG led to updates on several screens and features.

On a more technical side, the options made regarding the structure of the project and software design are also presented. The approach used for the project’s structure aims to help developers improve the project in the future by quickly introducing/modifying features to the project. Inclusive design methodologies, such as user-centred design, were also used, allowing users to participate in the project actively: their opinions weighted on the modifications made to each prototype.

Chapter 6

System Testing and Validation

This chapter provides an overview into the tests which were carried out throughout the implementation of the app. The tests in the chapter can be split into two major groups: user testing, and functional/non-functional testing. Section 6.1 details the procedures and results obtained in the first validation session with participation from CRPG clients. The procedures and results related to the usability testing can be seen in section 6.2. Validation of the system was done with users, but also through tests during development. Other functional and non functional tests are detailed and their results are presented in section 6.3. As mentioned in section 4.5, five participants agreed to use and validate the app in two different sessions. Table 6.1 provides a background of the participants who tested the app in the two testing sessions.

Table 6.1: Background of the participants (P).

	P01	P02	P03	P04	P05
Age (in years)	40	55	50	27	43
Sex (M/F)	M	M	F	M	M
Touchscreen/Peripheral use	Touchscreen	Touchscreen	Touchscreen	Touchscreen	Touchscreen
Assistive software	-	-	-	-	Reminder app
Used Operating System	Android	iOS	Android	Android	iOS
Identified necessity	-	Visual impairment	-	-	Vision impairment

6.1 User tests: First prototype

After the development of the first prototype, it was necessary to evaluate the current state of the application, in order to gather user feedback that would lead to improvements and problem fixes.The tests were conducted under our supervision and a script in order to ensure equal conditions and avoid inconsistencies between sessions. The sessions were initiated with an explanation of the system, followed by delivery of a script which included the tasks that the participant was expected to perform.

6.1.1 Procedures

After the development of the first prototype, it was necessary to evaluate the current state of the application, in order to gather user feedback that would lead to improvements and problem fixes. Of the seven participants which initially had agreed to take part in the validation and usability testing sessions, two participants announced that they could not participate in any of the sessions. As a result, testing of the application was done with five participants. All of the participants interacted with the application on site at the CRPG premises.

In this first session the participants were able to experiment with the app individually. This evaluation session had the following structure: an introduction was made regarding the application and its objectives. The user was asked to interact with the application, guided by a series of orders given by the supervisor. The orders were based on predetermined tasks which were created to make the participants explore all the features of the app and seek possible issues that they could find by navigating and assessing the visual elements in each screen.

During the interaction with the application, participants were incentivized to voice any thoughts or difficulties that they were finding and suggest alternatives which could improve their experience. After all of the tasks had been completed, users were asked to complete a survey which was divided into three sections.

The first ten questions are extracted and adapted from the *System Usability Score* (SUS), which, while subjective, allows for some conclusions to be drawn regarding the user's state of mind when interacting with a application and if they felt the interaction was difficult or confusing. The second section of the survey asked the participants to pinpoint the obstacles they found and if those obstacles were consistent throughout the application or were more prominent in specific features. The last three questions are related to the participants perception of the integration of future modalities and their usefulness. The answers to the questions were evaluated on a scale of one to five, where one corresponds to "fully disagree" and five being "fully agree". Table 6.2 indicates the questions that were asked to the users in the survey.

Table 6.2: Questions of the survey conducted after the users interacted with the app.

ID	Question
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.
11	I think it would be useful to be able to use voice commands in order to execute certain tasks in the application.
12	I think it would be useful for the application to present visual and auditory alerts regarding specific upcoming events, such as meals and transportation, without the need for user setting them.
13	I think it would be useful for the application to provide Audio Hints which could help me execute certain tasks or provide useful hints while using the application.

6.1.2 Results

In general, feedback regarding the user experience of the application was positive. Participants found the features helpful and, despite some points which needed further improvements, they found the interface and design to be understandable and not confusing. One of the questions in the survey allowed users to identify obstacles they had felt while using the application.

One of the main discussing points that all participants pinpointed was the structure of the transport feature. When the user clicks on the transport car in the agenda they are redirected to a screen which shows information regarding the schedules of the main CRPG bus. This was not immediately evident to the participants, showing that the flow of this feature could be improved, by allowing the user to select one of the types of transport before being shown a specific screen.

The Notes creation feature also was the subject of critique, which were mainly tied with lack of labels and confusing icons on the buttons. This issue was especially prominent in the “Create new Voice Note” feature, as the voice recording Section in the screen consists of three buttons: the first to start the recording, the second to stop the recording, as the third to replay the recording.

The icons which were used are part of the material design icons group, and as such were in conformity with the rest of the icons used in the application. However, given the difficulties in identifying the content of each image, the icons were changed and text labels would be added later in development. Other topic which the participants considered confusing was regarding the bottom

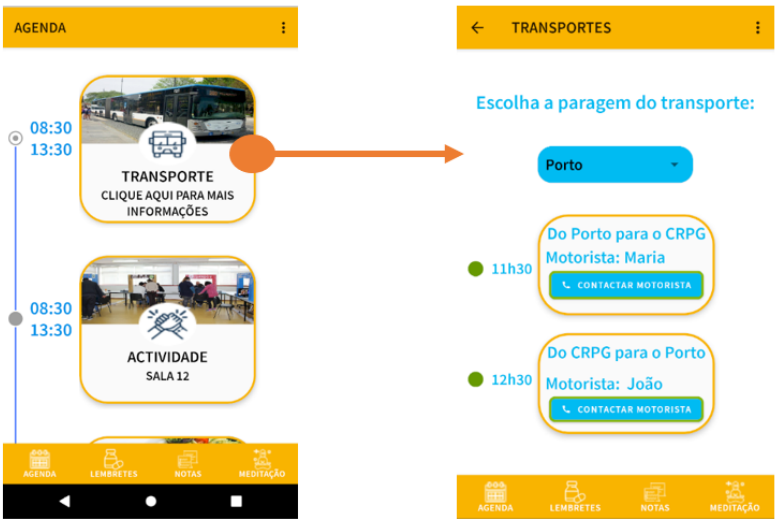

Figure 6.1: Expected flow when the user presses a transport-related item in the Agenda Screen (First prototype).

navigation bar. Two participants said navigation between features was somewhat confusing, as a result of the lack of text labels for each button in the bottom navigation bar. The size of the icons in the world was also found to be small and what the images represented was not immediately evident to the participants. integrating labels and increasing the size of the images were suggested to improve the navigation experience.

Lack of validation and visual feedback in specific areas of the application was also identified as confusing three of the users. This occurred in the Reminders feature, two groups of buttons Provided very subtle visual feedback when clicked, which made it difficult for users to understand if the application had registered their click.

One of the participants mentioned that it was difficult to read some of the visual elements and text in some screens due to reduced contrast between the text color and the background color. This was most noticeable in cases where a lighter blue text was above a white background. This participant considered the remainder of the colors and their contrast to be adequate.

After the participants identified some of their obstacles and struggles while using the application, they were asked to fill the survey in order to gather more information which will be useful in the following development stage. In the first section of the survey, Which was based on the scale disability system, the response by the participants was mostly positive. The results of the survey as summarized as follows (each list item is related to a question in the survey):

1. All of the participants fully agreed that they would like to use the application frequently.
2. All of the participants fully disagreed that they found the application unnecessarily complex.
3. All of the participants fully agreed that they found the application easy to use.
4. All of the participants fully disagreed with the idea of needing assistance from another person to use the app.

5. All of the participants fully agreed that they thought the features were well integrated.

6. All of the participants fully disagreed that there were inconsistencies in the application.

7. When the participants were asked if they thought most people would be able to learn how to use a application quickly, there were mixed results. Because the CRPG community has very different necessities, two participants said they neither fully agreed or disagreed, as some people would probably have a harder time using the app. Three participants however thought the application would be quickly assimilated by most members of the community.

8. All of the participants fully disagreed with the opinion that the application was cumbersome to use.

9. Four of the participants fully agreed that they felt confident using the application, while one participant somewhat agreed with this observation.

10. All of the participants fully disagreed with the thought of having to learn a lot of things before being able to use the app.

    In the last section of the survey, regarding the implementation of multimodal capabilities into the application in the following phases of development, participants showed interest in all of the proposed means of communication. The answers to these three questions are detailed as follows:

11. All participants fully agreed that the ability to use voice commands to execute actions and navigate to specific features inside the application would be appreciated.

12. All participants except for one (which responded with “Somewhat Agree”) said they Fully agreed that they would enjoy the integration of visual and auditory alerts for specific upcoming events.

13. All participants except for one (which responded with “Somewhat Agree”) said they Fully agreed that they would enjoy the integration of a synthesized voice which could help the user by providing suggestions and assisting the user in understanding the use of the voice commands.

When questioned about other features that the participants consider that could be useful in the app, one participant asked for real-time information regarding the transports, both of CRPG and also public transports. Due to the fact that there is no information being recorded by the drivers or the buses of CRPG, this idea was not viable to implement due to technical limitations. Regarding the panic button, the idea was proposed to the CRPG representatives in order to assess its viability. Fig.6.2 presents a graph that summarizes the results of the survey.

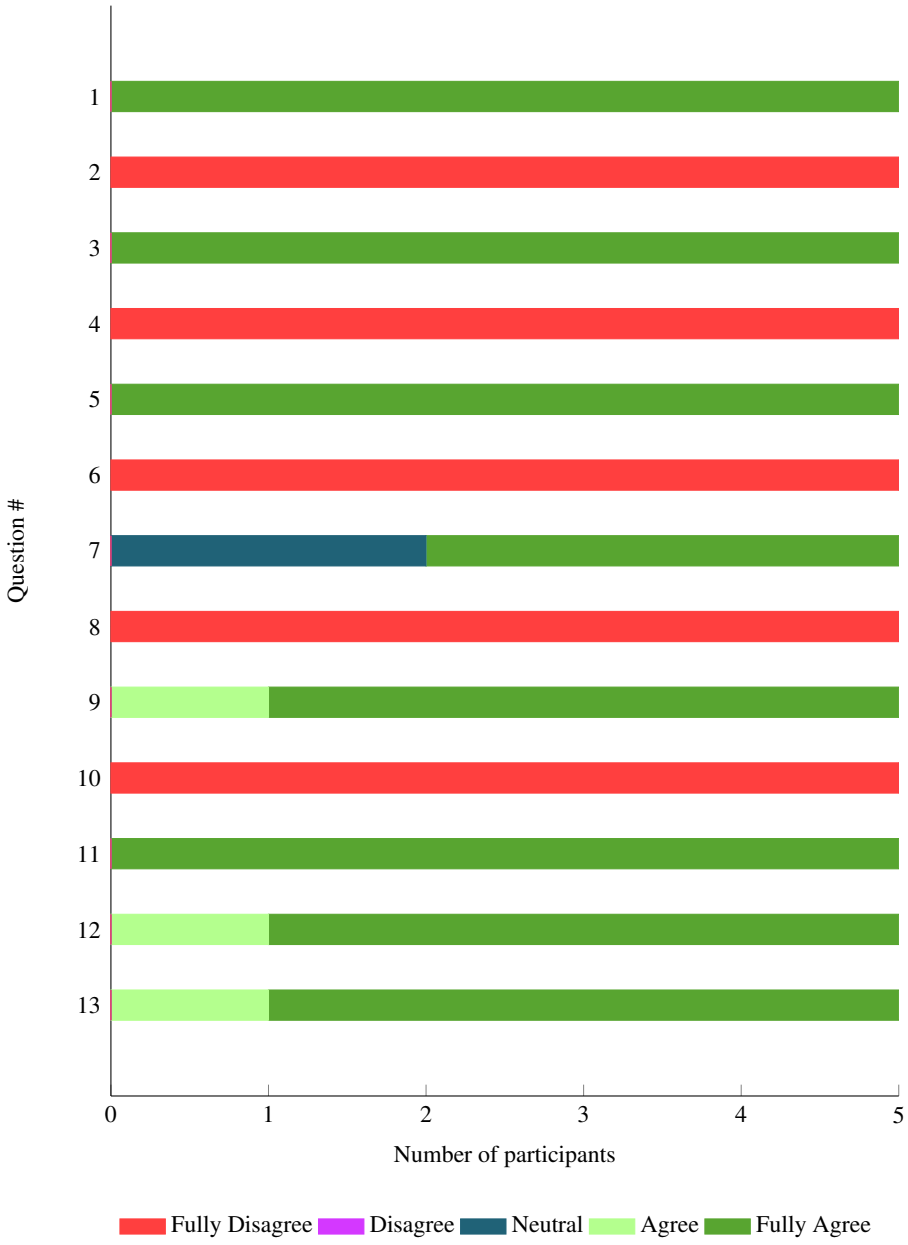

Figure 6.2: Answers to the survey (first prototype validation session).

6.2 User tests: Second prototype

After the development of the second prototype was finished, another meeting with the participants was scheduled. The objective of this sessions was to perform usability tests with all of the users who had taken part in the validation session. Subsection 6.2.1 details the procedures which were followed during the usability testing sessions. Subsection 6.2.2 provides an overview of the results of the tests.

6.2.1 Procedures

After the development of the app was finished, a more extensive phase of user testing was implemented. Users executed six test cases, using the touch screen and after that using both Audio Hints and voice commands. The sessions were initiated with an explanation of the system, followed by delivery of a script which included the tasks that the participant was expected to perform. Each user was asked to perform six test cases, using the touchscreen first, and a combination of the Audio Hints and Voice commands after. Table 6.3 details the test cases executed for each application. Each has six properties:

1. **ID**: identifies the test case.
2. **Screen/Feature**: identifies the screen/feature associated with the test.
3. **Summary**: describes the main objective that should be test case.
4. **Preconditions**: conditions that need to be met before the test case is executed.
5. **Expected user flow**: which details the minimum number of steps that the user needs to perform to complete the test case (optimal user flow).
6. **Expected results**: details the events which occur after the execution of the test case.

While the user executed each task, measurements regarding different parameters were gathered. For each test case, four parameters were measured:

- **Time needed for completion**: each case is timed from the moment the prerequisites are met, and the supervisor indicates to the user that the task starts. The timer is stopped when the user performs the last step of the flow.
- **Assists**: an assist is defined as a request for assistance by the user to the supervisor of the tests.
- **Deviations/non-critical errors**: Deviations from the expected flow that the participant can recover from. These errors result in an interruption to the normal flow of the test case, but the user is not assisted unless requested. If the user is able to recover from the deviation, the session continues until the user finishes the task.

- **Critical errors**: deviations from the expected flow which will cause the participant to not be able to finish the task. Includes navigating to a different feature, or submitting erroneous information). A critical error is usually followed by an assist, since the user needs helps from the supervisor to return to a previous state where the flow can be continued.

Table 6.3: Description of the test cases for the usability testing.

ID	Screen/Feature	Summary	Precondition	Expected user flow (touchscreen)	Expected user flow (voice commands)	Expected Results
1	Date Selection	Selection of the twenty-four of June in the horizontal scroll list and move to the Agenda screen.	1) The user is on the Date Selection Screen.	1)The user scrolls through the list until a card labelled "24" is visible. 2) The user clicks the specified card. 3) The user clicks the select button.	1) The user utters "24th of July". 2) The user utters "Select".	1) The information related to the selected date is loaded into the application.
2	Public Transports	View the schedule of public transports.	1) The user is on the Agenda Screen.	1) The user clicks on the Transport item. 2) The user clicks on "Public Transport". 3) The user clicks on "View Schedules". 4) The user clicks on button with the "Outward" label.	1) The user clicks on the Transport item. 2) The user utters "Public Transport". 3) The user utters "View Schedules". 4) The user utters "Outward".	1) The selected timetable is shown and can be zoomed in.
3	Meal Selection	Selection of the fish dish in the meal selection screen. After saving, return to the Agenda screen.	1) The user is on the Agenda Screen.	1) The user click on the "Fish" option. 2) The user clicks on the button with the "Save" label	1) The user utters "Fish". 2) The user utters "Save".	1) The meal selection is saved on the database and on the device. 2) The selected meal is shown on the adequate item in the Agenda.
4	Meditation	Selection of the "Confident" mood.	1) The user is on the Agenda Screen.	1) The user click on the button labbeled "Confident".	1) The user utters "Confident".	1)The media player screen is shown.
5	Create new text note	1) Creation of a new text note with the following characteristics: 1) No image is associated with the note; 2) The title should be the letter a". 3) The content should be the letter "b".	1) The user is on the Agenda Screen.	1) The user click on "Create New Text Note". 2) The user clicks on the title field. 3) The user sets the Title as "a". 4) The user clicks on the content field. 5) The user sets the content as "b". 6) The user click on the button labelled "save".	1) The user utters "Create New Text Note". 2) The user utters "Title". 3) The user sets the Title as "a". 4) The user utters "Content". 5) The user sets the Content as "b". 6) The user utters "save".	1) The text note is created and stored on the device storage.
6	Create new reminder	1) Creation of a new reminder with the following characteristics: 1) The purpose of the reminder is to alert the user of medication administration; 2) The alarm should be scheduled for 21h30; 3) The alarm should be triggered on a daily basis; 4) the device should vibrate, but should not play any auditory media or ringtone; 5) the reminder does not have any note associated with it;	1) The user is on the Agenda Screen.	1) The user expands all sections individually. 2) The user selects the "Administer medication" option. 3) The user sets the time to 21h30. 4) The user selects the "Everyday" option. 5) The user selects the "Vibrate" option. 6) The user clicks on the "Save" button.	1) The user utters "Expand all". 2) The user utters "Administer medication". 2) The user utters "nine and a half in the afternoon". 2) The user utters "Everyday". 2) The user utters "Vibrate". 2) The user utters "Save".	1) The reminder is created and stored on the device storage.

During the execution of the tests, interaction with the supervisor was always considered an assist request. The objective was to remove any source of ambiguity on declaring some interactions as assists or not, even if they were not requests for help. Each of the five participants performed the six tasks in both modalities; a total of sixty tests were performed. The TalkBack feature integration was not tested due to limitations in the schedule of the participants. After the tests had been performed, the users were shown the notifications feature using the following simulated examples:

- The transport notification is shown, and the user could select one of the options to open the adequate screen.
- The meal notification is shown, the user can select the meal by opening the meal selection screen directly.
- The administer medication notification is shown, and the user can select one of the options to indicate if the medication has been administered.

The session ended with the users answering a survey regarding their thoughts and feedback on the different modalities of interaction presented in the session. Some questions were adapted from the *SUS* survey done in the previous session. The purpose of the questions was to understand if the users thought they were well implemented and intuitive and obtain general feedback. The questions are described in Table 6.4.

Table 6.4: Questions of the survey conducted at the end of the usability test.

ID	Question
1	I think the several modalities of interaction with the application make interaction easier and more accessible.
2	I found the multiple modalities to be well integrated within the system.
3	I found the notifications with action buttons to improve the experience of using the application.
4	I found the notifications with action buttons simplified accessing features and performing certain tasks.
5	I believe I would use the notifications frequently.
6	I believe the notifications feature would be useful for the CRPG community.
7	I found that the Audio Hints integration improved the experience of using the application.
8	I found that the Audio Hints integration was adequate at detailing which voice commands to be uttered to perform certain tasks.
9	I felt I was faster at assimilating the existing commands due to the Audio Hints feature.
10	I believe the Audio Hints feature would be useful for the CRPG community.
11	I felt more confident when using the voice commands for the second time, due to having listened to the Audio Hints previously.
12	I believe I would use the Audio hints frequently.
13	I found that the voice commands integration improved the experience of using the application.
14	I felt confident using the voice commands.
15	I felt using the voice commands was cumbersome or that I did not possess enough information to use it.
16	I believe I would need assistance from another person to be able to use the voice commands.
17	I believe most people would be able to learn how to use the voice commands.
18	I believe the voice commands feature would be useful for the CRPG community.
19	I believe I would use the voice commands feature frequently.

6.2.2 Results

Given the considerable number of tests performed, the results presented in this subsection are summarized. Instead of presenting data regarding each test, the sum of the values for each parameter in each test is calculated. Detailed information for each individual test can be found in B. The first metric to be analyzed is the accumulated elapsed time to complete each task (in seconds). Time measurements help understand which modality allows users to perform tasks more quickly (less time spent is better). The results, organized by participant and modality, can be seen in Fig. 6.3.

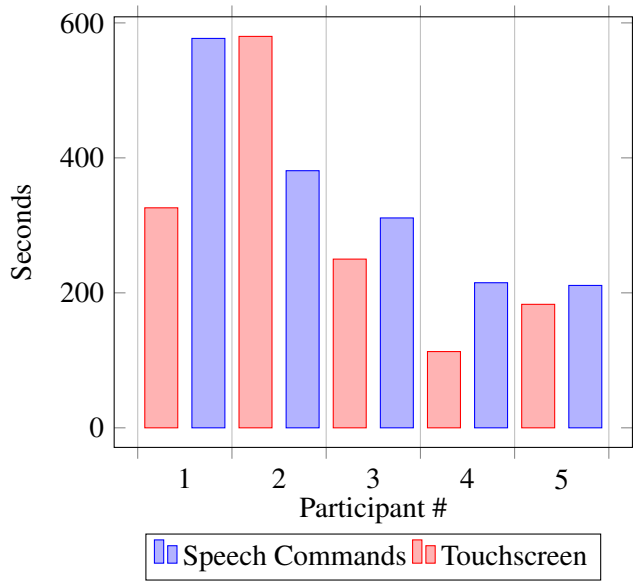

Figure 6.3: Total elapsed time (sum of every test case elapsed time) for each modality, by participant.

As shown in Fig.6.3, four participants were able to complete the tasks faster using the touchscreen. One participant was able to execute the task faster by using voice commands. The participant who needed less time to complete all of the tasks using voice commands was also the oldest in the group. The self-perceived dexterity of this participant was also considered average, with occasional difficulties in daily use of the smartphone. Younger participants assessed their dexterity as being more than sufficient to interact with their smartphones on a daily basis without any issues. This may indicate that the speech command functionality may be more helpful to older users or with lower dexterity levels in regards to touchscreen use. The following metric analyzed is deviations, also known as non-critical errors. Results from testing can be seen in Fig. 6.4.

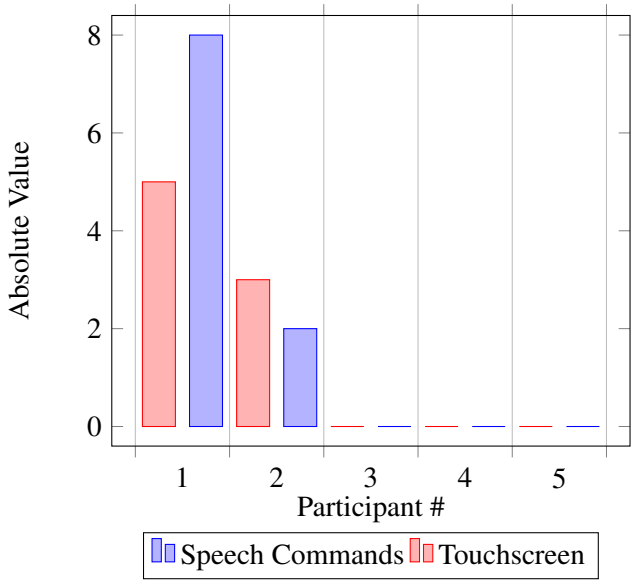

Figure 6.4: Total non-critical errors/deviations (sum of the non-critical errors from each test case) for each modality, by participant.

As can be seen in 6.4, only two participants had non-critical errors occurring during testing. One of these participants had more deviations when using the touchscreen. The other participant had more deviations when using the voice commands. The remaining three participants did not have any difficulties executing the tasks in a way that followed the expected flow. The third measurement is related to the critical errors. Results from testing can be seen in Fig. 6.5.

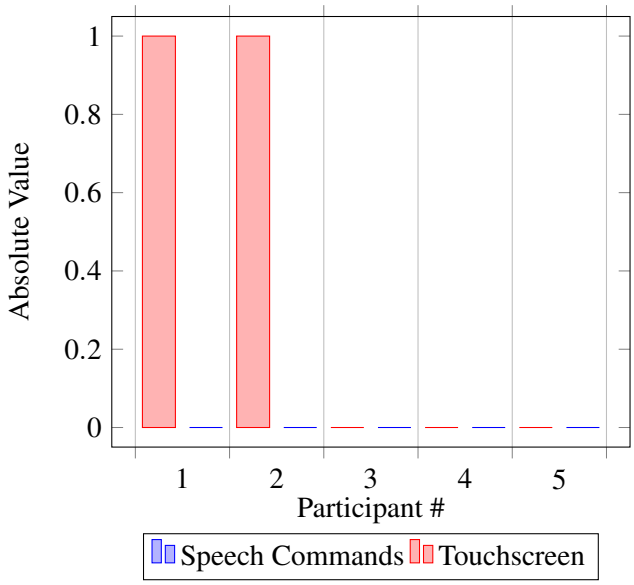

Figure 6.5: Total critical errors (sum of the critical errors from each test case) for each modality, by participant.

Only two occurrences of critical errors were found, both of them when two participants were using the touchscreen. Both instances resulted from a wrong button press which led the partici-

pants to another screen which they could not recover from. No critical errors occurred in any of the tests in which voice commands were used.

Fig. 6.6 summarizes the results of the tests regarding the assists parameter. The speech commands modality raised more questions from the participants, as compared to the touchscreen use. It is difficult to pinpoint if these questions resulted from the implementation of the feature, or the speech recognition limitations. During testing, it occurred several times that a user had said the correct command, but the system registered another word, so no action was taken. Some users repeated the command, which usually led to the correct recognition, but other participants were confused and thought their commands were wrong, leading to requesting assistance from the supervisor. It would be interesting to repeat the testing with an improved version of the software recognition in future work and measure the number of assist requests with a higher percentage of successful recognition of commands.

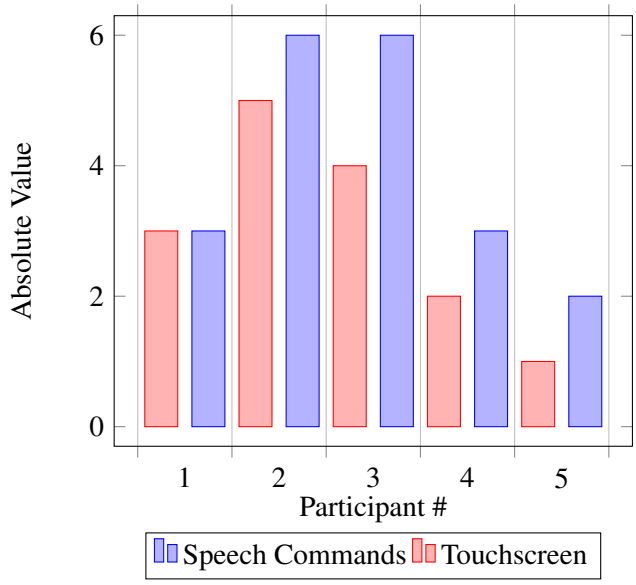

Figure 6.6: Total assists (sum of the assists from each test case) for each modality, by participant.

Regarding the results of the survey, the overall feedback was very positive. The summary of the answers can be found in the following list, organized by the ID of each question. The first two questions were related to the overall experience of the users with the different modalities.

1. All participants (except one, who answered with "neither agree nor disagree") indicated that the multiple modalities of interaction improved the user experience.

2. All participants said they fully agreed with the observation that the features were well integrated.

   The following section of the questionnaire focused on the notifications feature:

3. All participants said they fully agreed with the observation that the contextual notifications with associated actions improved the user experience.

4. All participants said they fully agreed with the observation that the notifications simplified accessing different features and performing specific tasks.

5. All participants fully agreed that they would use the notifications feature frequently.

6. All participants fully agreed that the notifications feature would be helpful for the CRPG community.

   After feedback from the notifications feature, the following answers were focused on the Audio Hints feature:

7. One participant somewhat disagreed with the observation that the Audio Hints feature improved the user experience. The other participants fully agreed with the observation.

8. There were mixed results regarding the eighth question regarding the adequacy of the Audio Hints feature in guiding users in using the voice commands. Two participants said they fully agreed with the observation, with another two participants saying they somewhat agreed; the remaining participant did not agree or disagree with the remark.

9. Three participants said they fully agreed that the Audio Hints feature made assimilating the voice commands easier. Between the other two participants, one somewhat agreed with the remark, while the other did not agree nor disagree.

10. All of the participants fully agreed that the Audio hints feature could be helpful for the CRPG community.

11. All of the participants, except one, fully agreed that they felt more confident using the voice commands a second time after using the Audio Hints before. This assessment was later confirmed, as all of the participants were asked to repeat the tasks a second time using voice commands, and in most cases, there was a significant improvement in the time needed to complete the tasks.

12. Three participants fully agreed that they would use the Audio Hints feature frequently, with the remaining two participants saying they neither agreed nor disagreed with this remark.

    Participants felt that the Audio Hints were not completely consistent, as the helpfulness of the different messages was considered inconsistent. Despite these limitations, the participants still felt that these messages helped indicate which voice commands could be uttered. The last section of the survey was related to the voice commands feature:

13. Four participants fully agreed that the voice commands improved the user experience. One participant slightly agreed.

14. All of the participants fully agreed that they felt comfortable using the voice commands.

15. All of the participants entirely disagreed with the observation that the feature was confusing or that they lacked information in order to take full advantage of the capabilities properly.

16. All of the participants except for one (which neither agreed nor disagreed) said they disagreed entirely with the observation that they would need assistance from another person in order to be capable of using the voice command feature.

17. Three of the participants said they fully agreed that they felt most people in the CRPG community would be able to use voice commands, with the remaining two participants neither agreeing nor disagreeing with the remark.

18. All of the participants fully agreed the voice commands feature could be helpful for the CRPG community.

19. When asked if they would use the voice command feature frequently, two of the participants said they fully agreed, one did not agree or disagree, and the remaining two said they somewhat disagreed with the observation.

The results are displayed in Fig.6.7. The results of the tests were satisfactory overall. Due to the small, and relatively homogeneous nature of the sample group, it is not easy to assess the actual adequacy of the modalities when transposed into the scope of the entire CRPG community. However, both modalities were well-received and considered adequate by the participants, and the feedback was positive. Some participants mentioned that they had better results using the touchscreen mainly due to their dexterity. Participants mentioned that the voice commands feature would be helpful in cases where the client's dexterity was limited or reduced. Notifications with integrated buttons were also well received, being a tool that reminds clients of upcoming events and quickly access associated features in the application.

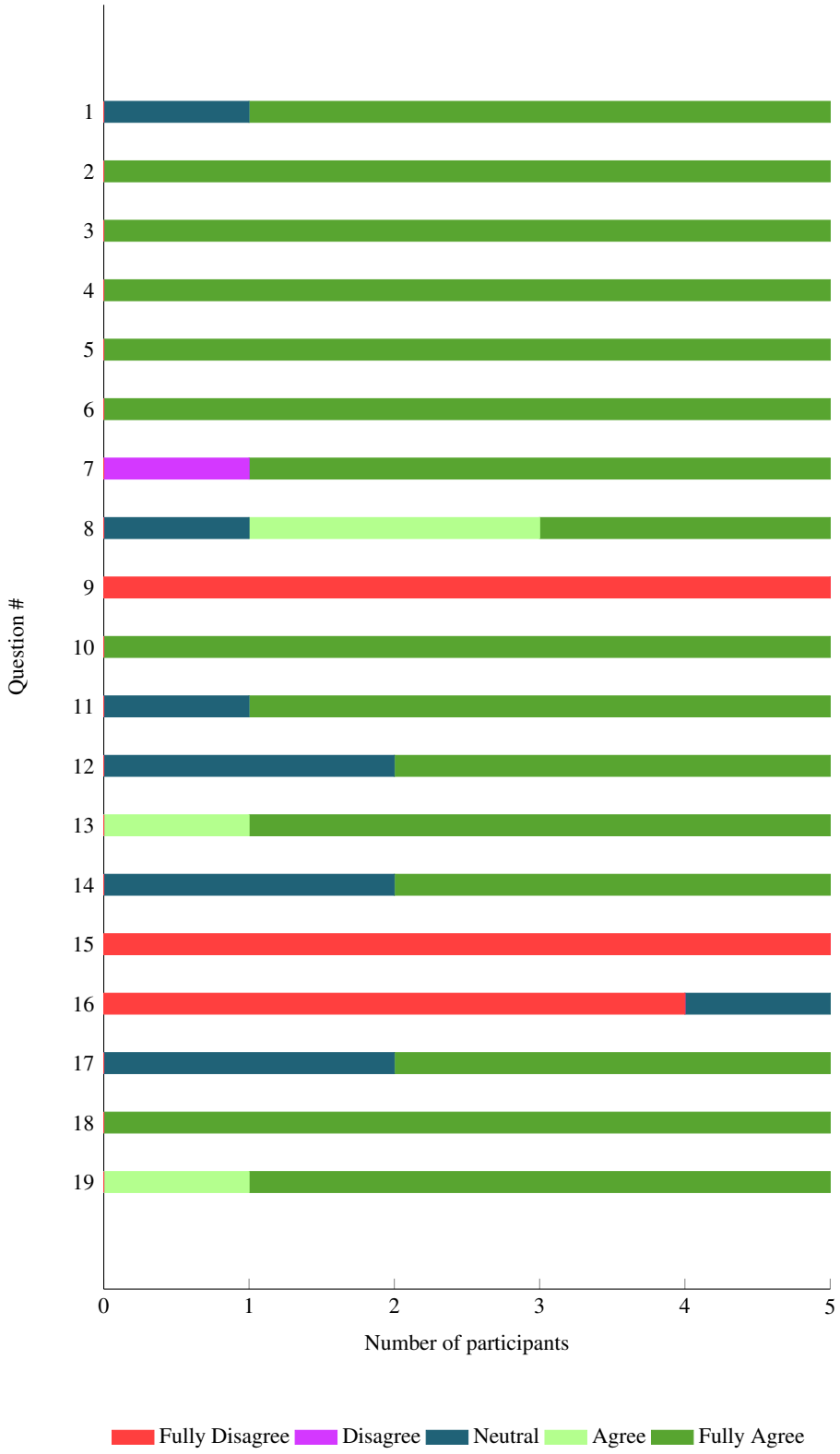

Figure 6.7: Answers to the survey (final prototype validation session).

6.3 Functional and Non-Functional Testing

This section details the functional and non-functional tests which were performed, both as system testing and accessibility testing. Regarding functional and non-functional testing, there are several ways in which a system can be tested, such as:

- **Unit testing**: testing specific actions in the app, in a very narrow scope (ie. validating that creating a reminder and clicking the save button effectively saves the reminder).
- **Integration testing**: assessment of flows in the app in a wider scope than unit tests, by associating actions performed by several unit tests.
- **System testing**: effectively using the application in order to verify if the requirements were met.
- **Acceptance testing**: testing user scenarios to verify if the system meets the customer requirements.

In order to make sure the application provided a consistent experience across a wide range of devices, three devices were used for testing. The devices spanned a five year time period of model releases, with varying levels of processing power, display resolution and aspect ratio, and Android releases. The following list details the models that were used for testing, and also details the year of model release, the Android version present on the device and the resolution of the display:

- Motorola G (2015 / Android 6.0 / 1080 x 1920).
- Xiaomi 7 (2018 / Android 8.0 / 720 x 1520).
- Xiaomi Redmi 9 (2020 / Android 11.0 / 1080 x 2400).

The focus of the functional/non-functional section was on system and acceptance testing. This focus is justified by the necessity of validating that the requirements were met, with a special focus on accessibility testing, in order to guarantee that the application reaches the highest number of clients possible. Validating the stability and performance of the application is also necessary. Table 6.5 shows in detail each type of testing that was performed.

Table 6.5: Description of functional and non-functional types of tests.

	Type	Description
Functional Tests	Functionality Testing	Consists of performing a set of planned test cases to cover all the system requirements. This ensures that the system specifications are implemented and working properly.
	Recoverability Testing	Verifying if the system is capable of dealing with invalid or inconsistent inputs.
	Ad-Hoc Testing	Trying to make an application stop working with a fatal error, and validate how the application responds in these scenarios.
	Graphical User Interface Testing	Analyzing all visual interface elements and verifying if they are working as expected.
	Exception Handling Testing	Consists of verifying if the system is capable of dealing with a fatal exception, display an error message to the user and keep working properly.
Non-Functional Tests	Compatibility Testing	Consists of testing if the system can be adequately executed and provide a consistent visual experience in multiple devices with different specifications. In this context it was necessary to test devices with multiple Android releases, processing power and displays with different resolutions and aspect ratios.
	Performance Testing	Consists of testing if the performance of the system is fluid, regardless of the device specifications and processing power.
	Accessibility Testing	Ensures that the application being tested is usable by people with disabilities, different ages and other groups.

6.3.1 Accessibility Testing

Regarding the assessment of the accessibility of the application, four areas were tested/analyzed to see if they performed as expected.

6.3.1.1 Screen Readers

In order to assess the suitability of Talkback integration with the app, a test was performed to ensure that the specified content descriptions were working as intended. Due to time constraints, the screen readers were not tested with users. The testing was done manually, by following the recommendations presented in the Android documentation[32]. The documentation recommends swiping through each visual elements, and analyze the following topics:

- Does the spoken feedback for each element convey its content or purpose appropriately?
- Are announcements succinct, or are they needlessly verbose?
- Can the main workflows be completed easily?
- Can every element be reached by swiping?
- If alerts or other temporary messages appear, are they read aloud?

After interacting with the app and following the guidelines in all of the screens, the results were satisfactory overall. The labels are succinct and clear as to describing the purpose of each element, and all of the relevant elements have their content description done. All of the workflows were completed with success, and every element could be reached. When an error message is shown on screen, it is also read by the Talkback service.

6.3.1.2 Colour Contrast Analysers

The application uses a reduced number of colors. The text in the application is mostly black, if on a white background, or white, if in a dark colored element. Apart from the background, which is always white (*#FFFFFF*), the following colors were used (color code below each example):

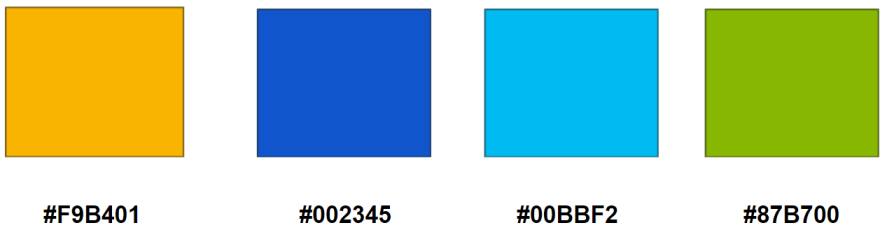

Figure 6.8: Examples of the colors present in the color scheme of the application.

In order to guarantee the suitability of the color scheme and of the contrast between colors, a *Colour Contrast Analyzer* (CCA) was used. Using this tool, it is possible to compare two different colors. A value regarding the contrast between both colors is provided and its suitability compared. Because the background color is always white, and text is always white, black or one of the above depending on the background color, four combinations of colors were analyzed (also shown in Fig.6.8). Black on white is by default the highest contrast possible so it was not tested. The contrast level values between these colors allow for the assessment of the adequacy of these colors. The following combinations were used:

- Dark blue on white background.
- Light blue on white background.
- Green on white background.
- Orange on white background.

Table 6.6 displays the results of the contrast checks. As seen in the table, the results were satisfactory overall. The reduced color scheme proved to be a good choice, despite two of the colors (light blue and green) not passing the more demanding test in which the contrast is validated based on the *WCAG AAA* more rigorous standards. In order to overcome this issue, a darker shade of each color can be used, which increases the contrast with the white background.

Table 6.6: Results of the contrast checking tool.

	Normal Text		**Large Text**	
	WCAG AA	**WCAG AAA**	**WCAG AA**	**WCAG AAA**
Orange foreground White background	**Pass**	**Pass**	**Pass**	**Pass**
Dark blue foreground White background	**Pass**	**Pass**	**Pass**	**Pass**
Light blue foreground White background	**Pass**	Fail	**Pass**	**Pass**
Green foreground White background	**Pass**	Fail	**Pass**	**Pass**

6.3.1.3 Inbuilt Accessibility Options

Android provides several native accessibility tools by default. Stock Android 11.0 release integrates the following tools (other tools are available but interoperability was focused on these tools as they were considered to be the most helpful to the CRPG clients):

- *High Contrast Text*: this feature fixes the text colour as either black or white, depending on the original text colour. High contrast should improve readability of text, improving user experience.
- *Color correction*: allows specific colors to be corrected in order to make visibility easier for colorblind users.
- *Magnification*: allows the user to zoom in and out of specific areas of the screen, improving visibility of specific visual elements.

The first tool to be tested was the High Contrast Text. All of the screens/features were explored using this tool, with a 100% success rate, as no inconsistencies or visual issues were identified. Fig. 6.9 illustrates the look of the Date Selection screen with the High Contrast Text tool activated.

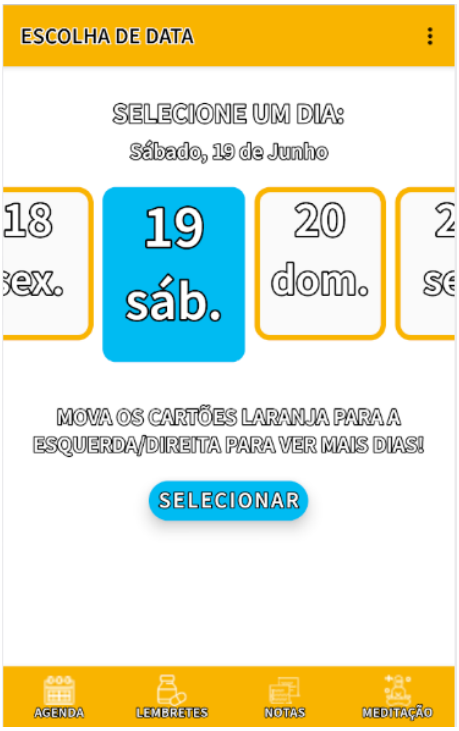

Figure 6.9: Example of High Contrast Text tool in the Date Selection Screen.

The second tool to be used was the Color Correction tool. Due to reasons that were not fully identified, usage of the tool proved to be inconsistent, as in some screens the tool could not be activated, and in other cases visual elements were not present. Because the tool would cause glitches even when the application was not being run, it is possible that the experimental nature

of this tool, or the specific release which was used for testing did not work fully as expected. In the future, these tests could be performed again using different Android releases and an improved version of the tool, in order to pinpoint the origin of the issues.

The last tool to be tested was the Magnification tool. All of the screens/features were also explored using this tool, with a 100% success rate, as the Magnification tool was able to zoom in on any specific area of the screen, and visibility was improved. Fig.6.10 illustrates usage of the Magnification tool in the Date Selection screen.

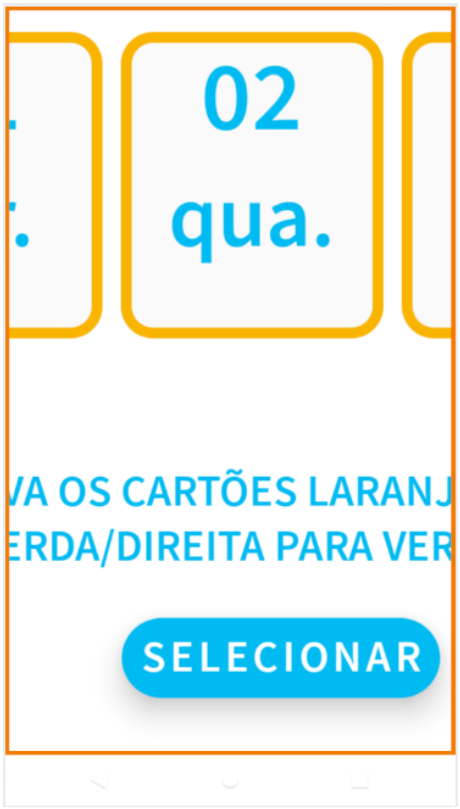

Figure 6.10: Example of the Magnification tool in the Date Selection Screen.

6.3.1.4 WCAG 2.1 Guidelines

As mentioned previously, *WCAG* 2.1 covers many different topics on accessibility in detail. It is not possible to cover every single one in detail, so a paraphrased summary from the official documentation is used to check the accessibility points of the app. In order to summarize the numerous requirements into concise, more comprehensive list, *W3C* provides a web page with thirteen topics which cover a wide range of areas of accessibility and requirements which need to be met[92]. Table 6.7 presents each of the key topics, as well as the approach during development to fulfill the requirements.

Table 6.7: *WCAG* summarized guidelines and approaches used to meet each one (adapted from [92]).

ID	Topics	Approach
1	Text alternatives for non-text content should be provided.	All buttons and icons have a description tag which describes its use.
2	Captions and other alternatives should be provided for multimedia.	The meditation media player has content description tags.
3	Created content should be presented in different ways, including by assistive technologies, without losing meaning.	All of the created notes are described in the consult reminder/note screens, following the format in which Talkback describes them is the recommended by the official documentation.
4	Make it easier for users to see and hear content.	Attention was directed to the size of visual elements and text, in order to improve readability, as well as using concise content description for the Talkback service to make comprehension easier.
5	Make all functionality available from a keyboard.	Not implemented
6	Give users enough time to read and use content.	There are no timed messages or pop-ups, so the user is free to take the time needed to read any content.
7	Content that causes seizures or physical reactions should not be used.	None of the screens/features integrate content that flickers, flashes, or blinks, which could trigger photosensitive epilepsy.
8	Help users navigate and find content.	The bottom bar icon and labels have increase size in order to help the users find the desired feature.
9	Make it easier to use inputs other than keyboard.	The use of the speech recognition commands, as well as the gesture recognition by Talkback, provide the user input alternatives.
10	Text should be readable and understandable.	Android reccomends the minimum font size to be 12px for user with no impairments to the vision. Given the context, the minimum text size in the application is 14px, but most text in the app ranges from 16-22px for normal text, and 25-35px for titles or subtitles.
11	Make content appear and operate in predictable ways.	The presentation of content follows the guidelines of accesible mobile apps design, as well as input from the end users from the CRPG community.
12	Help users avoid and correct mistakes.	Most screens allow the user to return to a previous screen using both the app bar arrow, and screens which which require considerable input from the user (such as create new Reminder) provice a button which allows all fields to be cleared.
13	Maximize compatibility with current and future user tools.	The application implementation followed the most modern Android's accessibility guidelines, and as such future Android updates should maintain or even improve the accessibility level of the app.

6.3.2 Compatibility Testing

The application ran on all three devices without any issues or reported slowdowns. Some visual inconsistencies in the screens with bigger resolution, like the Xiaomi Note Pro 9, were identified. It is not uncommon for *APK* releases to feature different configurations, allowing multiple devices to be supported. Due to time constraints, this configuration was not implemented. Still, the interface implementation followed the guidelines for multiple device compatibility, and as a result the application was always usable despite some visual inconsistencies.

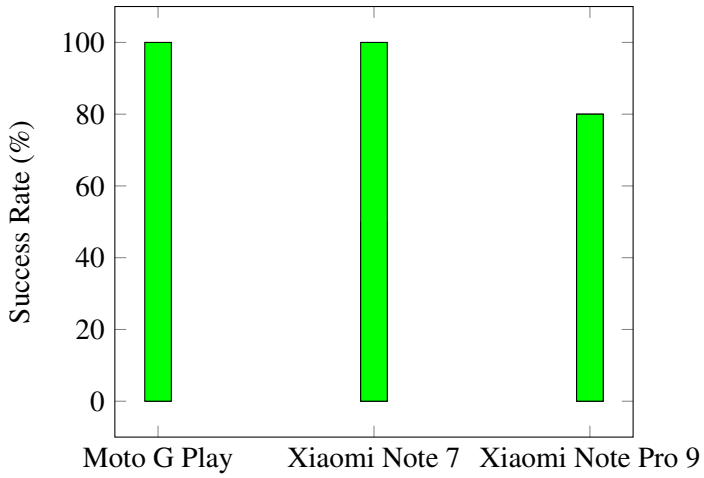

Figure 6.11: Compatibility testing results.

6.3.3 Performance Testing

The application performance was not influenced by the test environment, being uniform across the three devices. The application was less responsive on the oldest phone when the application was started, but it may be related to the fact that when the application is opened initially, a lot of data is fetched from the database. After the data is on the device memory, no further slowdowns were experienced.

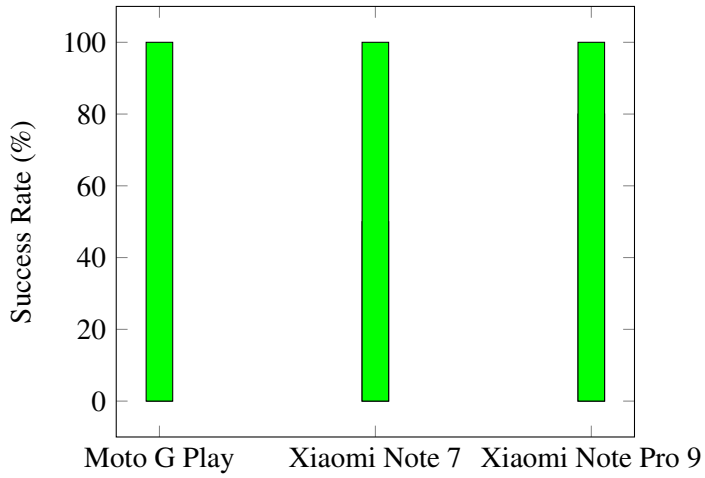

Figure 6.12: Performance testing results.

6.3.4 Recoverability Testing

The application passed all recoverability tests without any reported errors. Issues with validation of strings can happen when the edit field allows the user to have complete freedom on the type of characters and length. Since the application limits the user in the type of characters used, and a validation is being done even before the user submits the content that was written, the application presents two layers of validation, one while editing and the other when submitting, which allows the system to be less prone to errors from strings.

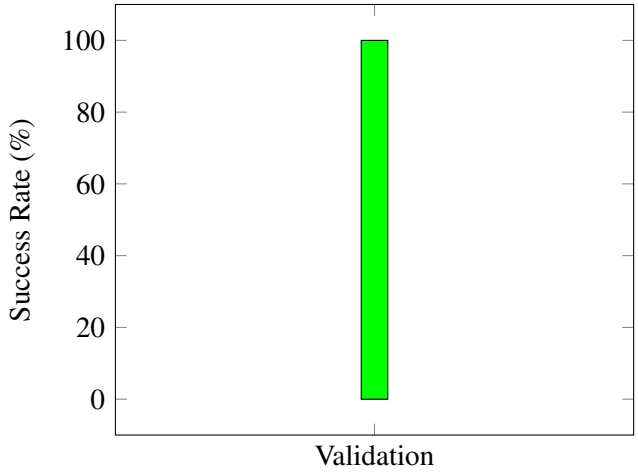

Figure 6.13: Recoverability Testing results.

6.3.5 Functionality Testing

All the functionalities worked properly when the users were experimenting with only the following minor bugs/limitations reported:

1. Since the date selection screen only shows the dates for the current month, the user is not able to see the next months dates.

2. If the user closes the Wi-fi connection when using the app, the application will still run but the Audio Hints and Speech Recognition services will stop without notifying the user, as they require an Internet connection to operate.

3. The speech recognition software is not fully accurate, as incorrect readings of the uttered words were registered during testing. This is related to the capabilities/limitations of the service which was used and is not associated with the implementation of the application.

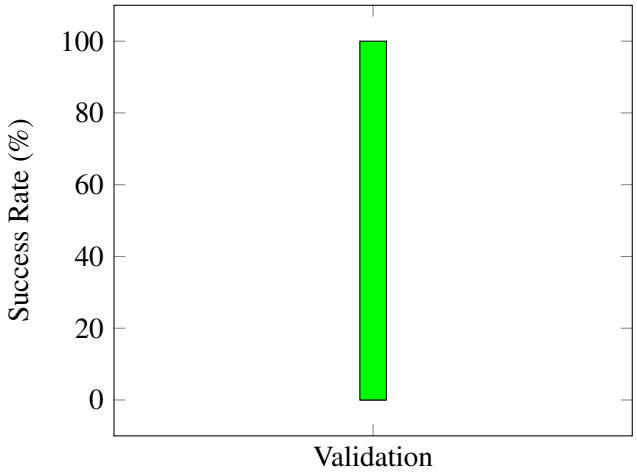

Figure 6.14: Functionality testing results.

6.3.6 Ad-hoc Testing

In the ad-hoc testing, invalid values were defined in the editable text fields, and then the reaction of the application was assessed when trying to validate these values. No crashes occurred and the application was always able to throw an error message before any issue or crash occurred.

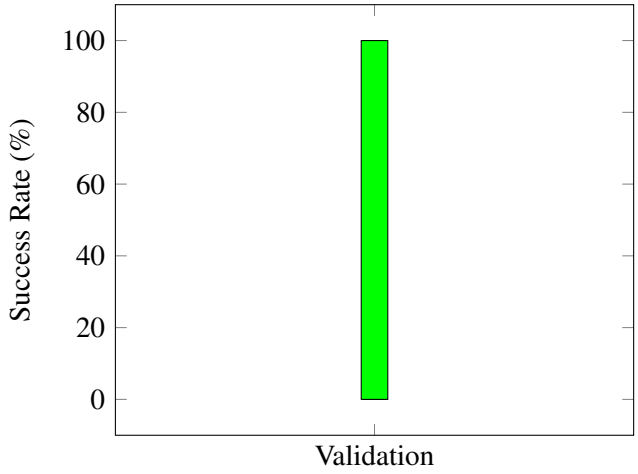

Figure 6.15: Ad-hoc testing results.

6.3.7 Graphical User Interface Testing

All the graphical user interface tests passed without any issues or visual inconsistencies in the three tested devices.

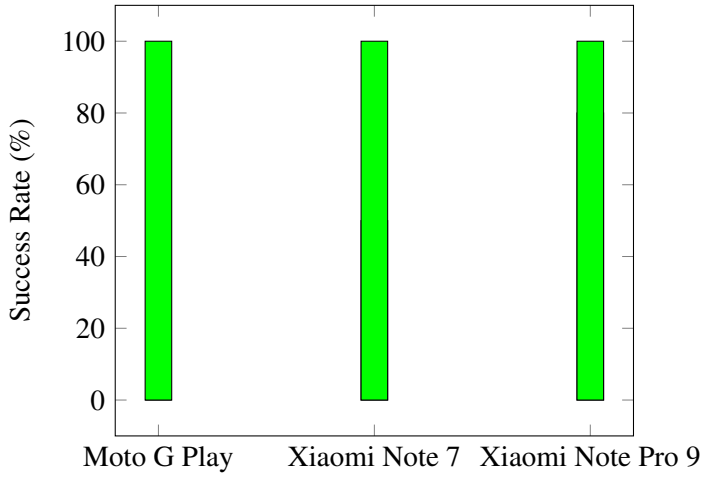

Figure 6.16: Graphical User Interface testing results.

6.3.8 Exception Handling Testing Testing

The application passed all the exception handling tests without the system stopping its execution. All the exceptions reported were dealt with smoothly. On each of these instances, the application throws an error message to the user and the system does not crash or become unusable. The exceptions reported were the following:

- Fetching data from the database or local storage and an error occurs.
- Saving a reminder or a note with invalid/missing data.
- Trying to use the voice recognition or Audio Hints features and failing.

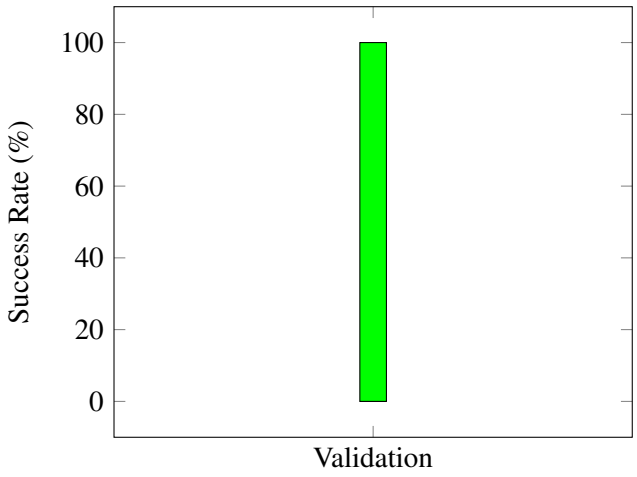

Figure 6.17: Exception Handling testing results.

6.4 Conclusions

In general, the performed tests had positive results, despite limitations to the scope of testing, such as the reduced number in available real devices for testing, and the small sample group of users. Regarding the group of user testers, despite a wide range of ages between participants, all of the participants displayed sufficient or above sufficient dexterity in the use of the smartphone, which translated into a very homogeneous group. A more numerous and heterogeneous group would allow for a more accurate representation of the CRPG community. The feedback from the users in the first user session, in which some obstacles to the use of the application were identified, allowed for several improvements to the second prototype. The users felt that the second prototype was easier to use and some issues regarding visibility of certain visual elements or obstacles to the navigation between features had been solved.

In the usability testing session, results were mixed, and did not allow to pinpoint a specific modality as being better for every participant. The participants did however agree that the multiple modalities in interaction with the app were positive and would allow the application to reach more clients. Some participants mentioned that they would not use the voice commands on a daily basis, due to the habit and comfort of using the touchscreen. Participants also found the voice hints efficacy to be inconsistent, as the messages were considered to be more or less helpful depending on the feature. The oldest participant, who mentioned occasional difficulties when using the touchscreen was able to perform some tasks faster by using the voice commands. Despite using voice commands to execute tasks not being common for the participants, they were able to quickly assimilate the commands and when they performed the tasks a second time, they were generally able to reach the goals more quickly and with less assistance when compared to the initial try.

Functional and non-functional tests were also performed. Only three physical devices were used, which makes assessing the true compatibility of the application in a wide range of models difficult. The results were satisfactory overall, with the success rate being of 100% for most tests, and a sub-par experience on one of the devices regarding the compatibility tests. The application performed as expected (no significant slowdowns or crashes) even in a mid-range device released in 2014.

Chapter 7

Conclusions

This chapter summarizes the results of the work in this project, the limitations that were identified and possible improvements that could benefit future iterations of the application. In section 7.1, the hypotheses that were made in chapter 1 are reviewed. Considerations regarding the limitations of the project and future work can be found in sections 7.2 and 7.3.

7.1 Findings

In chapter 1, three research questions were presented:

- Can the non functional-prototype be adequately implemented, by using inclusive design methodologies and approaches?
- Can assistive technologies and multimodality be integrated into the mobile application in a way that specifically benefits the clients of CRPG?
- Do the multiple modalities of interaction benefit the clients of CRPG?

In order to answer the first hypothesis, user-centered design methodology was used to ensure that the user feedback impacted the development of the application and improved following any identified obstacles. The application was successfully implemented, as a multimodal Android app. All of the initial requirements were integrated, as well as several others that appeared during the development of the application. Regarding the non functional prototype, the design was made consistent. Even on screens where high-fidelity design prototypes were not available, there was an effort to maintain the visual identity of the application consistent and without significant alterations. Due to restrictions in some of the development tools, visual fidelity was not totally achieved, but an effort was made to keep the the layouts as close to the mock-ups as possible. The feedback from both the CRPG clients and the representatives was that the application fulfilled their requirements and was suitable for daily use. Despite the limitations to the application, such as limited interoperability with assistive technologies, the first goal was accomplished, and acts as the first step of many that can expand the application's capabilities, in future work.

For the second hypothesis, the clients of CRPG were inquired regarding the obstacles they faced when interacting with their smartphones. Because vision impairments and short-term memory loss were identified, the focus on multimodality prioritized tools that help users with these types of characteristics. In addition, modalities such as the integration of text-to-speech as a synthesized voice and the ability to send voice commands to the app increased the application's reach for a higher number of users. The users praised the integrated modalities, as they felt these features would be useful in the context of the CRPG community. As with the previous point, there are limitations and it is not possible to guarantee that 100% of the clients would be able to use the application. However, given the time frame available for the development of the app, it was necessary to focus work on a limited number of modalities that were considered to be of significant relevance in the context of the CRPG community. In future work, it would be interesting to integrate interoperability with other assistive technologies such as mechanical devices, like pointers and keyboards. After considering every factor, it seems that the answer to this question is also positive, as the included modalities were based on the feedback of the CRPG clients and representatives.

In order to assess the truthfulness of the third and last hypothesis, an usability testing session was organized to allow users to experiment using the app with the different interaction modalities. In this session, clients were asked to perform a series of tasks inside the application, using both the touch screen on the phone, and afterwards using both the audio hints and voice commands. The results of the measured parameters show that depending on the user's dexterity, the use of different modalities can benefit the user experience. Adding to the results of the measurements, answers to the survey also pointed in the same direction. Participants in general said that they felt the features were well-implemented and thought the multiple options available of interaction would be beneficial for the CRPG community.

7.2 Limitations

All of the main objectives and requirements regarding this project were fulfilled. Despite the overall satisfaction in reaching the proposed objectives, some issues and limitations occurred. For example, the initial work plan was altered due to the consequences of the Covid-19 pandemic that were undergoing during the writing of this dissertation. One of the drawbacks regarding implementation of the functional prototype is the fact that it was not possible to achieve complete consistency between some visual aspects in the app and in the non-functional prototype. This was primarily due to technical limitations or obstacles to which there were no available solutions during the implementation period.

In order to accommodate interactions with both prototypes by the clients, the work plan was altered to discard the first session with users (initially three were planned), and postpone the second session at the CRPG premises. To prevent this alteration of impacting negatively later development cycles, the first prototype included all of the functionalities, lacking only the integration of multiple interaction modalities. This decision was beneficial because it allowed all of the users to interact with a prototype in an advanced state, but was also not as adequate as the initial planning, because user feedback was only gathered around one month after than initially scheduled.

Development for both Android and iOS operating systems was discarded early on, due to lack of documentation on accessibility capabilities of hybrid frameworks. The decision to focus the development just on Android allowed the study to occur on the intricacies of this specific system and to take advantage of its capabilities.

Since twelve clients of CRPG were initially asked to participate in this process, having just five participants at the end fell below the expectations of what was initially planned. Despite a wide range of ages between participants, all showed a very high dexterity in the use of the smartphone, which translated into a very homogeneous group. A more numerous and heterogeneous group, featuring users with little experience in using smartphones, or which use other assistive technologies, would allow for a more accurate representation of the CRPG community.

On the subject of functional and non-functional testing, it should be noted that testing was performed on only three physical devices. Despite the differences between the devices which suggest that the application performs well on devices with different specifications, it would be interesting to perform tests on more devices with even more differences between them.

7.3 Future Work

Although all of the method requirements have been successfully implemented, the application could benefit from the several following improvements, detailed in the following list:

- Integrating the application with the existing CRPG database was impossible, as there was no API available to fetch the data. The technical department of the CRPG could not develop an API that would allow the application to access data during the available development period. It was necessary to create a new independent database using Firebase real-time database. In future iterations of the project, integration with the existing database would reduce extra work by the staff (by not adding data to the database manually) and financial costs.

- As a result of the approach mentioned in the previous topic, the management of data has to be done using the Firebase console. CRPG already uses Excel files to organize data, some of which is then ported to the database. If the CRPG database remains inaccessible in the future, it would be advantageous to have a back-office in-built to the application. The back-office area would allow the staff to manage data or provide a feature allowing data from Excel files to be converted and uploaded to the database.

- Due to the time constraints and technical differences between Android and iOS operating systems, the application only runs on Android. It would be beneficial if the application could be adapted to run in the iOS operating system.

- As mentioned in chapter 2, the assistive technologies field is vast and complex. Many factors have an impact on the way individuals with disabilities interact with smartphones. Some assistive technologies are purpose-built to the necessity of a single user, which makes ensuring accessibility for these cases even more difficult. These circumstances make achieving total accessibility for 100% of the users virtually impossible. The focus on specific modalities of interaction was based on the assessment of common requests and necessities of CRPG clients in the RAC-LCA program. The requests were primarily related to memory and visual limitations instead of the use of hardware technologies. Future iterations of this project should include increased compatibility with more devices and other alternative assistive technologies.

- Initially, the non-functional prototype included an additional feature in the Transports section, which allowed users to find a walking path between their current location and the CRPG using the Google Maps API. After discussing this topic with the CRPG representatives, it was decided that this feature would be discarded in the final release due to security and financial costs. Alternative APIs may be studied in future iterations to allow this feature to be implemented without privacy issues.

- In the Public Transports Screen, the solution that was used in order for the users to see timetables is not ideal, as the amount of content present in the images can be overwhelming.

Due to time constraints and lack of a prototype for this specific feature, this was the selected approach, but in future iterations a more suitable solution could be implemented to improve user experience.

- While not present in the initial list of requirements, during the development of the application CRPG representatives requested using the Excel files that were already being used to organize data in the institution to insert data into the database. Because this information was only disclosed in a later stage of the development, this possibility was not included. The data has to be introduced manually in the application through the Firebase console. Future iterations of the application could benefit from data insertion using the Excel file as a source or through the back office.
- While there was an effort to support the different screens and devices, the tools used for development, such as Android Studio and its emulator, do not fully guarantee that the displays will be appropriately displayed on all devices. The application layouts were tested on resolutions and form factors common in the smartphone market. In some screens above or below a certain resolution aspect ratio, the layouts can present visual inconsistencies or have visual elements cut out. While more problematic in more extreme cases like tablets and very old phones, future work could include improved consistency across a broader range of devices.

References

[1] Definition and overview. URL: http://universaldesign.ie/what-is-universal-design/definition-and-overview/ [cited 2021-02-10].

[2] Getting Started About Version Control. URL: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

[3] GSON User Guide. URL: https://github.com/google/gson/blob/master/UserGuide.md [cited 2021-06-04].

[4] Kotlin vs Java: Which is Better for Android App Development? URL: https://www.xenonstack.com/blog/kotlin-andriod/ [cited 2021-06-04].

[5] Licenses. URL: https://choosealicense.com/licenses/ [cited 2021-06-04].

[6] Material Design. URL: https://material.io/design [cited 2021-01-21].

[7] The 7 Principles. URL: http://universaldesign.ie/what-is-universal-design/the-7-principles/ [cited 2021-02-10].

[8] Wrapper - Definition. URL: https://techterms.com/definition/wrapper [cited 2021-06-04].

[9] Apple. Vision - VoiceOver, 2021. URL: https://www.apple.com/accessibility/vision/ [cited 2021-01-14].

[10] K Beck. Extreme Programming Explained: Embrace Change. 2000.

[11] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Agile Manifesto, 2001. URL: http://agilemanifesto.org/.

[12] Stefan Becker, Andreas Kribben, Sven Meister, Clarissa Jonas Diamantidis, Nicole Unger, and Anna Mitchell. User profiles of a smartphone application to support drug adherence–experiences from the iNephro project. *PloS one*, 8(10):e78547, 2013. doi:10.1371/journal.pone.0078547.

[13] Herbert D. Benington. Production of Large Computer Programs. *Annals of the History of Computing*, 5(4):350–361, 1983. doi:10.1109/MAHC.1983.10102.

[14] Andreas Biorn-Hansen, Tim A. Majchrzak, and Tor-Morten Gronli. *Progressive Web Apps for the Unified Development of Mobile Applications*, pages 64–86. 06 2018. doi:10.1007/978-3-319-93527-0_4.

[15] David Britch. Visual representation of the MVVM architecture pattern., 2017. URL: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm [cited 2021-06-04].

[16] Cloudfare. Backend-as-a-Service. URL: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/ [cited 2021-06-04].

[17] European Comission. European Disability Strategy 2010-2020: A Renewed Commitment to a Barrier-Free Europe, 2010. URL: https://ec.europa.eu/social/main.jsp?catId=1484&langId=en [cited 2021-01-12].

[18] CRPG. O CRPG - História, 2020. URL: https://www.crpg.pt/o-crpg/historia/ [cited 2021-01-11].

[19] Bertrand Damiba. Introducing Voice Actions for Android, 2011. URL: http://googlemobile.blogspot.com/2011/09/introducing-voice-actions-for-android.html [cited 2021-06-04].

[20] Dan Wilson. App Accessibility in NativeScript, 2016. URL: https://nativescript.org/blog/app-accessibility-in-nativescript/ [cited 2021-01-29].

[21] Brad E Dicianno, Bambang Parmanto, Andrea D Fairman, Theresa M Crytzer, Daihua X Yu, Gede Pramana, Derek Coughenour, and Alan A Petrazzi. Perspectives on the Evolution of Mobile (mHealth) Technologies and Application to Rehabilitation. *Physical Therapy*, 95(3):397–405, 2015. doi:10.2522/ptj.20130534.

[22] Inc. Drifty. Ionic Documentation Overview, 2016. URL: https://ionicframework.com/docs{%}0Ahttp://ionicframework.com/docs/overview/ [cited 2021-01-21].

[23] R. et al. Cugelman, B. Cugeman. Color psychology, 2019. URL: https://www.alterspark.com/color-psychology/visual-hierarchy [cited 2021-01-29], doi:10.1007/978-3-642-27851-8_228-2.

[24] European Commission. DIRECTIVE (EU) 2016/2102 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 26 October 2016 on the accessibility of the websites and mobile applications of public. Document 32016L2102 sector bodies., 2016. URL: http://data.europa.eu/eli/dir/2016/2102/oj [cited 2021-01-12].

[25] Andrea D Fairman, Erika T Yih, Daniel F McCoy, Edmund F Lopresti, Michael P McCue, Bambang Parmanto, and Brad E Dicianno. *International journal of telerehabilitation*, 8(1):11–20, 2016. doi:10.5195/ijt.2016.6194.

[26] American Foundation for the Blind. Screen Reading Technology and Refreshable Braille Displays, 2020. URL: https://www.afb.org/blindness-and-low-vision/using-technology/assistive-technology-videos/screen-reading-technology [cited 2021-01-26].

[27] Erich Gamma. Design Patterns - Repository Pattern. URL: https://archive.org/details/designpatternsel00gamm/page/127(accessedon05/06/2021) [cited 2021-06-04].

[28] Paul S Ganney, Sandhya Pisharody, and Edwin Claridge. Chapter 9 - Software engineering. In Azzam Taktak, Paul S Ganney, David Long, and Richard G Axell, editors, *Clinical Engineering (Second Edition)*, pages 131–168. Academic Press, second edition edition, 2020. URL: http://www.sciencedirect.com/science/article/pii/B9780081026946000097 [cited 2021-01-09], doi:https://doi.org/10.1016/B978-0-08-102694-6.00009-7.

[29] Gartner. SDK (Software Development Kit), 2021. URL: https://www.gartner.com/en/information-technology/glossary/sdk-software-development-kit [cited 2021-01-27].

[30] Github. Github. URL: https://github.com/ [cited 2021-06-04].

[31] Google. Android Studio. URL: https://developer.android.com/studio [cited 2021-06-04].

[32] Google. Android's "Test your app accessibility". URL: https://developer.android.com/guide/topics/ui/accessibility/testing [cited 2021-06-04].

[33] Google. Data and file storage overview. URL: https://developer.android.com/training/data-storage [cited 2021-06-04].

[34] Google. Firebase Guide. URL: https://firebase.google.com/docs/guides [cited 2021-06-04].

[35] Google. Handler. URL: https://developer.android.com/reference/android/os/Handler [cited 2021-06-04].

[36] Google. Locale. URL: https://developer.android.com/reference/java/util/Locale [cited 2021-06-04].

[37] Google. Material Design Introduction. URL: https://material.io/design/introduction [cited 2021-06-04].

[38] Google. Flutter for Android Devs, 2020. URL: https://flutter.dev/docs/get-started/flutter-for/android-devs [cited 2021-01-21].

[39] Alex Gotev. Android Speech. URL: https://github.com/gotev/android-speech [cited 2021-06-04].

[40] Muhammed Salih Guler. A deep dive into Flutter's accessibility widgets, 2019. URL: https://medium.com/flutter-community/a-deep-dive-into-flutters-accessibility-widgets-eb0ef9455bc [cited 2021-01-29].

[41] Jerry Hildenbrand. What is Google TalkBack?, 2014. URL: https://www.androidcentral.com/what-google-talk-back [cited 2021-01-12].

[42] Robert T Hughes. Project Management Software. In Robert A Meyers, editor, *Encyclopedia of Physical Science and Technology (Third Edition)*, pages 139–153. Academic Press, New York, third edition edition, 2003. URL: http://www.sciencedirect.com/science/article/pii/B0122274105008528 [cited 2021-01-12], doi:https://doi.org/10.1016/B0-12-227410-5/00852-8.

[43] Kyle Jablonski. Android Notifications Tutorial: Getting Started, 2019. URL: https://www.raywenderlich.com/1214490-android-notifications-tutorial-getting-started [cited 2021-06-04].

[44] Alejandro Jaimes and Nicu Sebe. Multimodal human–computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1):116–134, 2007. URL: https://www.sciencedirect.com/science/article/pii/S1077314206002335, doi:https://doi.org/10.1016/j.cviu.2006.10.019.

[45] JetBrains. Kotlin Programming Language, 2018. URL: https://kotlinlang.org/docs/faq.html [cited 2021-06-04].

[46] Simeon Keates, P Clarkson, and Peter Robinson. Developing a practical inclusive interface design approach. *Interacting with Computers*, 14:271–299, 2002. doi:10.1016/S0953-5438(01)00054-6.

[47] Ben Kitpitak. Reasons to use Android single activity architecture with navigation component, 2020. URL: https://oozou.com/blog/reasons-to-use-android-single-activity-architecture-with-navigation-component-36 [cited 2021-06-04].

[48] Mike Korn, Peter, Loïc Martínez Normand Pluke, Andi Snow-Weaver, and Gregg Vanderheiden. Guidance on Applying WCAG 2.0 to Non-Web Information and Communications Technologies (WCAG2ICT), 2013. URL: https://www.w3.org/TR/wcag2ict/ [cited 2021-01-09].

[49] P Kraft and Jorgen P Bansler. The Collective Resource Approach: the Scandinavian Experience. *Scand. J. Inf. Syst.*, 6:4, 1994.

[50] DON & MELBA RUGG ASSISTIVE TECHNOLOGY LAB. Assistive Technology, 2021. URL: https://craighospital.org/services/assistive-technology [cited 2021-01-27].

[51] Ovidiu Latcu. Android Repository Pattern using RX Room. URL: https://medium.com/corebuild-software/android-repository-pattern-using-rx-room-bac6c65d7385 [cited 2021-06-04].

[52] Roman Leventov. Single Responsibility Principle. URL: https://reflectoring.io/single-responsibility-principle/ [cited 2021-06-04].

[53] Lawrence Lewandowski, Whitney Wood, and Laura A. Miller. Chapter 3 - technological applications for individuals with learning disabilities and adhd. In James K. Luiselli and Aaron J. Fischer, editors, *Computer-Assisted and Web-Based Innovations in Psychology, Special Education, and Health*, pages 61–93. Academic Press, San Diego, 2016. URL: https://www.sciencedirect.com/science/article/pii/B9780128020753000036, doi:https://doi.org/10.1016/B978-0-12-802075-3.00003-6.

[54] William. Lidwell, Kritina. Holden, and Jill. Butler. *Universal principles of design*. 2003. doi:https://doi.org/10.1075/idj.14.2.11byr.

[55] Johannes Link. The Role of Unit Tests in the Software Process. In Johannes Link, editor, *Unit Testing in Java*, The Morgan Kaufmann Series in Software Engineering and Programming, page iv. Morgan Kaufmann, San Francisco, 2003. URL: http://www.sciencedirect.com/science/article/pii/

B9781558608689500249 [cited 2021-01-29], doi:https://doi.org/10.1016/B978-1-55860-868-9.50024-9.

[56] Brian Marick. Exploration Through Example. URL: http://www.exampler.com/old-blog/2003/08/22/#agile-testing-project-2 [cited 2021-06-04].

[57] Robert C. Martin. Dependency Inversion Principle.

[58] Robert C. Martin. Interface Segregation Principle. URL: https://web.archive.org/web/20150905081110/http://www.objectmentor.com/resources/articles/isp.pdf [cited 2021-06-04].

[59] Robert C. Martin. Liskov Substitution Principle. URL: https://web.archive.org/web/20150905081111/http://www.objectmentor.com/resources/articles/lsp.pdf [cited 2021-06-04].

[60] Robert C. Martin. Open-closed principle. URL: https://web.archive.org/web/20150905081105/http://www.objectmentor.com/resources/articles/ocp.pdf [cited 2021-06-04].

[61] Robert C. Martin. *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall, 2003.

[62] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. 2004.

[63] Alexander Mertens, Christopher Brandl, Talya Miron-Shatz, Christopher Schlick, Till Neumann, Andreas Kribben, Sven Meister, Clarissa Jonas Diamantidis, Urs-Vito Albrecht, Peter Horn, and Stefan Becker. A mobile application improves therapy-adherence rates in elderly patients undergoing rehabilitation: A crossover design study comparing documentation via iPad with paper-based control. *Medicine*, 95(36):e4446, sep 2016. doi:10.1097/MD.0000000000004446.

[64] Felix Mohammadi Kho'i and Jawed Jahid. Comparing Native and Hybrid Applications with focus on Features. 2016. URL: https://www.diva-portal.org/smash/get/diva2:944058/FULLTEXT02 [cited 2021-01-21].

[65] React Native. Accessibility - React Native, 2020. URL: https://reactnative.dev/docs/accessibility{#}accessibility-properties [cited 2021-01-29].

[66] Donald A Norman and Stephen W Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., USA, 1986. doi:10.1201/b15703.

[67] Z Obrenovic, Julio Abascal, and Dusan Starcevic. Universal accessibility as a multimodal design issue. *Commun. ACM*, 50:83–88, 2007. doi:10.1145/1230819.1241668.

[68] Tom Occhino. React Native: Bringing modern web techniques to mobile, 2015. URL: https://engineering.fb.com/2015/03/26/android/react-native-bringing-modern-web-techniques-to-mobile/ [cited 2021-01-12].

[69] HARI Lab University of Pittsburgh. iMHere 2.0 App. URL: https://imhere.pitt.edu/ [cited 2021-01-21].

[70] Oracle. Java - Language Environment. URL: https://www.oracle.com/java/technologies/language-environment.html [cited 2021-06-04].

[71] World Health Organization. What do we mean by self-care? URL: https://www.who.int/reproductivehealth/self-care-interventions/definitions/en/ [cited 2021-01-09].

[72] Page Laubheimer. Cards: UI-Component Definition, 2016. URL: https://www.nngroup.com/articles/cards-component/ [cited 2021-01-25].

[73] Anastasia Panfilova, Carolina Vaz-pires, and Darth Silveira. Creating a Mobile Application for people with brain injuries. Technical report, 2020.

[74] Aymeric Parant, Sandrine Schiano-Lomoriello, and Francis Marchan. How would I live with a disability? Expectations of bio-psychosocial consequences and assistive technology use. *Disability and Rehabilitation: Assistive Technology*, 12(7):681–685, 2017. doi:10.1080/17483107.2016.1218555.

[75] Troy Patrick. The Relationship Between Android and Java. URL: https://theiconic.tech/android-java-fdbd55aadc51 [cited 2021-06-04].

[76] John Smith Paul Stovell. Model-View-ViewModel (MVVM). URL: https://groups.google.com/g/wpf-disciples/c/P-JwzRB_GE8?pli=1.

[77] Joe Petrakovich. Why should you use the repository pattern? URL: https://makingloops.com/why-should-you-use-the-repository-pattern/ [cited 2021-06-04].

[78] Phabricator. Arcanist User Guide: Lint. URL: https://secure.phabricator.com/book/phabricator/article/arcanist_lint/ [cited 2021-06-04].

[79] Presidência do Conselho de Ministros. RCM n.º 97/99, 1999. URL: https://dre.pt/application/conteudo/428656 [cited 2021-01-09].

[80] Presidência do Conselho de Ministros. Decreto-Lei n.º 83/2018, 2018. URL: https://dre.pt/application/file/a/116734886 [cited 2021-01-09].

[81] Progress. What is NativeScript?, 2020. URL: https://nativescript.org/faq/what-is-nativescript/ [cited 2021-01-21].

[82] Leah Reeves, Jennifer Lai, James Larson, Sharon Oviatt, T Balaji, Stéphanie Buisine, Penny Collings, Philip Cohen, Ben Kraal, Jean-Claude Martin, Michael Mctear, T Raman, Kay Stanney, Hui Su, and Qian Wang. Guidelines for multimodal user interface design. *Commun. ACM*, 47:57–59, 2004. doi:10.1145/962081.962106.

[83] Stanley Shyiko. ktlint. URL: https://github.com/pinterest/ktlint [cited 2021-06-04].

[84] Juan M. Silva, Alain Mouttham, and Abdulmotaleb El Saddik. Ubimeds: A mobile application to improve accessibility and support medication adherence. In *Proceedings of the 1st ACM SIGMM International Workshop on Media Studies and Implementations That Help Improving Access to Disabled Users*, MSIADU '09, page 71–78, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1631097.1631109.

[85] Timothy Stephen Springer. European Union Accessibility Requirements. URL: https://www.levelaccess.com/european-union-accessibility-requirements/ [cited 2021-01-12].

[86] Tom McFarlin. Repository Pattern Benefits. URL: https://tommcfarlin.com/repository-pattern-benefits [cited 2021-06-04].

[87] Jakob Trischler, Simon Pervan, Stephen Kelly, and Don Scott. The Value of Codesign: The Effect of Customer Involvement in Service Design Teams. *Journal of Service Research*, 21:75–100, 2018. doi:10.1177/1094670517714060.

[88] Bill Venners. Inside Java virtual machine. URL: https://www.artima.com/insidejvm/ed2/index.html [cited 2021-06-04].

[89] W3. Web Content Accessibility Guidelines (WCAG) Overview. URL: https://www.w3.org/WAI/standards-guidelines/wcag/ [cited 2021-01-12].

[90] W3C. Mobile Accessibility at W3C. URL: https://www.w3.org/WAI/standards-guidelines/mobile/ [cited 2021-01-21].

[91] W3C. W3C MISSION. URL: https://www.w3.org/Consortium/mission [cited 2021-01-12].

[92] W3C. WCAG 2.1 at a Glance, 2018. URL: https://www.w3.org/WAI/standards-guidelines/wcag/glance/ [cited 2021-06-04].

[93] W3Schools. Java Encapsulation. URL: https://www.w3schools.com/java/java_encapsulation.asp [cited 2021-06-04].

[94] W3Schools. JSON - Introduction, 2018. URL: https://www.w3schools.com/js/js_json_intro.asp [cited 2021-06-04].

[95] Wunder. Physical Pointers. URL: https://wunder.io/wunderpedia/accessibility/digital-assistive-technologies/physical-pointers/ [cited 2021-01-14].

[96] D. Yu, B. Parmanto, V. Watzlaf, B. Dicianno, and K. Seelman. Accessibility needs and challenges of a mHealth system for patients with dexterity impairments. *Disability and Rehabilitation: Assistive Technology*, 12(1), 2017. doi:10.3109/17483107.2015.1063171.

Appendix A

Code Fragments

A.1 Notifications

```
1
2  /**
3   * NotificationsManager.kt
4
5   Creates a high-priority notification channel which is used by all of the
        notifications. Setting the notification as high-priority instructs Android to
        grab the user's attention using not only visual cues, but also sound signals
        if possible.
6   */
7  fun createNewNotificationChannel(context: Context, importance: Int, showBadge:
       Boolean, name: String, description: String) {
8          if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
9              val channelId = "${context.packageName}-$name"
10             val channel = NotificationChannel(channelId, name, importance)
11             channel.description = description
12             channel.setShowBadge(showBadge)
13
14             val notificationManager = context.getSystemService(NotificationManager
                   ::class.java)
15             notificationManager.createNotificationChannel(channel)
16         }
17     }
```

```
/**
 * NotificationsManager.kt

 Creates a notification associated with a meal selection event.
 */
fun createNewMealNotification(
            context: Context, title: String,
            message:
            String,
            autoCancel: Boolean,
    ){
        val channelId = "${context.packageName}-${context.getString(R.string.
            app_name)}"

        val notificationBuilder = NotificationCompat.Builder(context, channelId).
            apply {
            setSmallIcon(R.drawable.ic_notification_meal)
            setContentTitle(title)
            setContentText(message)
            priority = NotificationCompat.PRIORITY_HIGH
            setAutoCancel(autoCancel)

            val intent = Intent(context, MainActivity::class.java)
            intent.putExtra("id", "meal")
            intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.
                FLAG_ACTIVITY_CLEAR_TASK

            val uniqueID = System.currentTimeMillis().toInt()
            val pendingIntent = PendingIntent.getActivity(context, uniqueID, intent
                , 0)
            setContentIntent(pendingIntent)
        }

        val notificationManager = NotificationManagerCompat.from(context)
        notificationManager.notify(1003, notificationBuilder.build())

    }
```

```
/**
 * MainActivity.kt

 Handles the action that is performed when the user clicks on the notification or
     in one of the notification action buttons. In this case, clicking the
     notification instructs the MainActivity to replace the existing fragment with
     the Meal Selection fragment (if the platform is running), or opens the
     platform before if it is not running.
 */
private fun handleMealsReminderNotificationClick() {
        val current = intent
        val name = current.getStringExtra("id")

        if (current != null && name == "meal") {
            val bundle = Bundle()
            bundle.putBoolean("isLunch", true)
            val fragment: Fragment = MealsFragment()
            fragment.arguments = bundle
            val fragmentManager: FragmentManager = this.supportFragmentManager
            val fragmentTransaction: FragmentTransaction = fragmentManager.
                beginTransaction()
            fragmentTransaction.replace(R.id.nav_host_fragment, fragment)
            fragmentManager.popBackStack()
            fragmentTransaction.addToBackStack(null)
            fragmentTransaction.commit()
        }
    }
```

```

/**
 * AlarmReceiver.kt
 Listens to thrown Intents by any class. If the intent is of the notification type,
      executes the method defineAlarmTypeAndCreateAlarm which identifies the type
     of reminder and associates the an alarm with the reminder.
 */
class AlarmReceiver : BroadcastReceiver() {

    private val TAG = AlarmReceiver::class.java.simpleName

    override fun onReceive(context: Context?, intent: Intent?) {
        Log.d(TAG, "onReceive() called with: context = [$context], intent = [
            $intent]")
        if (context != null && intent != null) {
            NotificationsManager.defineAlarmTypeAndCreateAlarm(context,intent) }
    }
}
```

```
1
2  /**
3   * AlarmScheduler.kt
4
5   Schedules an alarm associated with a specific reminder, sharing the reminder's
        date, time and frequency.
6   */
7  private fun scheduleAlarm(dayOfWeek: Int, repeats: Boolean, reminder: Reminder,
8                                      alarmIntent: PendingIntent?, alarmMgr:
                                           AlarmManager) {
9
10
11         // Set up the time to schedule the alarm
12         val datetimeToAlarm = Calendar.getInstance(Locale.getDefault())
13
14         //sets date and hours for alarm depending
15         on the data associated with the reminder
16         datetimeToAlarm.timeInMillis = System.currentTimeMillis()
17         datetimeToAlarm.set(DAY_OF_WEEK, reminder.selectedDay)
18         datetimeToAlarm.set(HOUR_OF_DAY, reminder.hours)
19         datetimeToAlarm.set(MINUTE, reminder.mins)
20         datetimeToAlarm.set(SECOND, 0)
21         datetimeToAlarm.set(MILLISECOND, 0)
22
23         // Compare the datetimeToAlarm to today
24         val today = Calendar.getInstance(Locale.getDefault())
25
26        when(reminder.alarm_freq){
27            AlarmFrequency.TODOS_OS_DIAS -> { alarmMgr.setRepeating(
28                    AlarmManager.RTC_WAKEUP,
29                    datetimeToAlarm.timeInMillis, (1000 * 60 * 60 * 24 * 7).toLong()
                          , alarmIntent
30            )}
31            AlarmFrequency.HOJE ->
32                alarmMgr.set(AlarmManager.RTC_WAKEUP, datetimeToAlarm.timeInMillis,
                      alarmIntent);
33        }
34
35
36     }
```

```kotlin
/**
 * BootReceiver.kt
 Receives the BOOT_COMPLETED action and schedules the alarms after reboot.
 */
class BootReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context?, intent: Intent?) {
        if (context != null && intent?.action.equals("android.intent.action.
            BOOT_COMPLETED")) {
            // Reschedule every alarm here
            AlarmScheduler.scheduleAlarmsForAllReminder(context)
        }
    }
}
```

A.2 Modality Settings

```

private var textToSpeech: TextToSpeech? = null
private val handler = Handler(Looper.getMainLooper())
private var runnable: Runnable by Delegates.notNull()
private val myLocale = Locale("pt_PT", "POR")
```

```
private fun requestMultiModalitySettings() {
    val sharedPreferences = getSharedPreferences("MODALITY", Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()

    MaterialAlertDialogBuilder(this, android.R.style.Theme_Material_Dialog_Alert)
            .setTitle("Permitir Sugestoes de Audio")
            .setMessage("Prima o bot o \"Permitir\" para ativar as sugestoes de
                audio.")
            .setPositiveButton("Permitir") { dialog, which ->
                editor.putBoolean("TTS", true).apply()
            }
            .setNegativeButton("Recusar"){ dialog, which ->
                editor.putBoolean("TTS", false).apply()
            }.show()

    MaterialAlertDialogBuilder(this, android.R.style.Theme_Material_Dialog_Alert)
            .setTitle("Permitir Comandos de Voz")
            .setMessage("Prima o bot o \"Permitir\" para ativar o reconhecimento
                de comandos por voz.")
            .setPositiveButton("Permitir"){ dialog, which ->
                editor.putBoolean("SR", true)
            }
            .setNegativeButton("Recusar"){ dialog, which ->
                editor.putBoolean("SR", false)
            }.show()
}
```

```

private fun defineModality(ttsFlag: Boolean, srFlag: Boolean, hasRun: Boolean) {

        if (!hasRun){
            when{
                ttsFlag && !srFlag -> { startTTS() }
                !ttsFlag && srFlag -> { startVoiceRecognition() }
                ttsFlag && srFlag ->{ multimodalOption() }
            }
```

```
10          }
11
12          if(hasRun){
13              when{
14                  !ttsFlag && srFlag -> { startVoiceRecognition() }
15                  ttsFlag && srFlag ->{ startVoiceRecognition() }
16              }
17          }
18
19  }
```

```
1
2  private fun multimodalOption() {
3          if (!onResumeFlag) {
4              textToSpeech = TextToSpeech(context) { status ->
5                  if (status == TextToSpeech.SUCCESS) {
6                      val ttsLang = textToSpeech!!.setLanguage(myLocale)
7                      if (ttsLang == TextToSpeech.LANG_MISSING_DATA
8                              || ttsLang == TextToSpeech.LANG_NOT_SUPPORTED) {
9                          Log.e("TTS", "The Language is not supported!")
10                     } else {
11                         Log.i("TTS", "Language Supported.")
12                     }
13                     Log.i("TTS", "Initialization success.")
14
15                         val speechListener = object : UtteranceProgressListener() {
16                             @Override
17                             override fun onStart(p0: String?) {}
18
19                             override fun onDone(p0: String?) {
20                                 if(activity != null && isAdded) {
21                                     startVoiceRecognition()
22                                 }
23                             }
24
25                             override fun onError(p0: String?) {}
26                         }
27
28                     textToSpeech?.setOnUtteranceProgressListener(speechListener)
29
30                     val speechStatus = textToSpeech!!.speak("Selecione um dos
                           estados", TextToSpeech.QUEUE_FLUSH, null, "ID")
31
32                 } else {
33                     Toast.makeText(context, "TTS Initialization failed!", Toast.
                           LENGTH_SHORT).show()
34                 }
35             }
36         }
37
38     }
```

A.3 Text-To-Speech

```
1
2  private fun startTTS() {
3          textToSpeech = TextToSpeech(context) { status ->
4              if (status == TextToSpeech.SUCCESS) {
5                  val ttsLang = textToSpeech!!.setLanguage(myLocale)
6                  if (ttsLang == TextToSpeech.LANG_MISSING_DATA
7                          || ttsLang == TextToSpeech.LANG_NOT_SUPPORTED) {
8                      Log.e("TTS", "Unsupported language!")
9                  }
10                 val speechStatus = textToSpeech!!.speak("Selecione um estado",
                       TextToSpeech.QUEUE_FLUSH, null, "ID")
11             } else {
12                 Toast.makeText(context, "TTS Initialization failed!", Toast.
                       LENGTH_SHORT).show() }
13         }
14     }
```

A.4 Speech Recognition

```
fun startVoiceRecognition(){
    if(isAdded && isVisible && getUserVisibleHint()) {
        runnable = Runnable {
            handler.sendEmptyMessage(0);
            Speech.init(requireActivity())
            try {
                Speech.getInstance().startListening(object : SpeechDelegate {
                override fun onStartOfSpeech() {}
                override fun onSpeechRmsChanged(value: Float) {}
                override fun onSpeechPartialResults(results: List<String>) {
                val str = StringBuilder()
                for (res in results) {
                    str.append(res).append(" ")
                }
                performActionWithVoiceCommand(results.toString())
                Log.i("speech", "partial result: " + str.toString().trim { it <= '
                    ' })
                }

                override fun onSpeechResult(result: String) {
                    Log.d(TAG, "onSpeechResult: " + result.toLowerCase())
                    val handler = Handler()
                    if (activity != null && isAdded) {
                        handler.postDelayed({
                            try {
                                if(isAdded && isVisible && getUserVisibleHint()) {
                                    Speech.init(requireActivity())
                                    Speech.getInstance().startListening(this)
                                }
                            } catch (speechRecognitionNotAvailable:
                                SpeechRecognitionNotAvailable) {
                                speechRecognitionNotAvailable.printStackTrace()
                            } catch (e: GoogleVoiceTypingDisabledException) {
                                e.printStackTrace()
                            }
                        }, 100)}
                }})} catch (exc: SpeechRecognitionNotAvailable) {
                Log.e("speech", "Speech recognition is not available on this device
                    !")
            } catch (exc: GoogleVoiceTypingDisabledException) {
                Log.e("speech", "Google voice typing must be enabled!")
            }
        }
        handler.post(runnable) }
}
```

```
Speech.init(this)

    try {
        Speech.getInstance().startListening(new SpeechDelegate() {
            //Allows an action to be performed when the recognition starts
            fun onStartOfSpeech() {}

            //Updates the string with the user commands as they are uttered
            fun onSpeechPartialResults(List<String> results) {
                StringBuilder str = new StringBuilder();
                for (String res : results) {str.append(res).append(" "); }}

            //Receives string with commands and executes an action to ti
            fun onSpeechResult(String result) {}});

    } catch (SpeechRecognitionNotAvailable exc) {
        Log.e("speech", "Speech recognition is not available on this device!");
        // Redirects user to the Google App page on Play Store
        SpeechUtil.redirectUserToGoogleAppOnPlayStore(this);
    } catch (GoogleVoiceTypingDisabledException exc) {
        Log.e("speech", "Google voice typing must be enabled!");
    }
```

```
fun performActionWithVoiceCommand(command: String) {
    when {
        command.contains("Relaxado", true) -> button_mood_relaxed.performClick()
        command.contains("Feliz", true) -> button_mood_happy.performClick()
        command.contains("Com Sono", true) -> button_mood_sleepy.performClick()
        command.contains("Confiante", true) -> button_mood_confident.performClick()
        command.contains("Querido", true) -> button_mood_loved.performClick()
        command.contains("Mente Sa", true) -> button_mood_mindful.performClick()
    }
}
```

```
override fun onDestroy() {
        super.onDestroy ();
        handler.removeCallbacksAndMessages(null)

        if(handler.hasMessages(0)) handler.removeCallbacks(runnable)

        if (textToSpeech != null) {
            textToSpeech!!.stop()
            textToSpeech!!.shutdown() }
}
```

A.5 Recycler View Item Binding

```
when(timeLineModel.type){
    EventType.ACTIVITY -> {
        holder.itemView.card_background_image.setBackgroundResource(R.drawable.
            crpg_background)
        holder.itemView.card_center_icon.setBackgroundResource(R.drawable.maos)
        holder.itemView.text_timeline_title.text = "ACTIVIDADE"
        holder.itemView.text_timeline_info.text = timeLineModel.info.toUpperCase()
    }
    EventType.MEAL -> {
        holder.itemView.card_background_image.setBackgroundResource(R.drawable.
            background_dieta)
        holder.itemView.card_center_icon.setBackgroundResource(R.drawable.meal_icon
            )

        when (timeLineModel.title) {
            "ALMO O" -> if (timeLineModel.chosen_meal.isNullOrBlank()) {
                holder.itemView.text_timeline_info.text = "CLIQUE AQUI PARA
                    SELECIONAR ALMO O"
            } else {
                holder.itemView.text_timeline_info.text = timeLineModel.chosen_meal
            }

            "JANTAR" -> if (timeLineModel.chosen_meal.isNullOrBlank()) {
                holder.itemView.text_timeline_info.text = "CLIQUE AQUI PARA
                    SELECIONAR JANTAR"
            } else {
                holder.itemView.text_timeline_info.text = timeLineModel.chosen_meal
            }

        }

    }
    EventType.TRANSPORTS -> {
        holder.itemView.text_timeline_info.text = "CLIQUE AQUI PARA MAIS
            INFORMA ES"
        holder.itemView.card_background_image.setBackgroundResource(R.drawable.
            stcp_background)
        holder.itemView.card_center_icon.setBackgroundResource(R.drawable.bus_icon)
    }
}
```

```
1
2  if (listData[position].tipo == NoteType.VOICE) {
3              val uri: Uri = Uri.parse(listData[position].voiceNotePath)
4              val player = SimpleExoPlayer.Builder(ctx).build()
5              holder.itemView.note_item_player_view.player = player
6              val mediaItem: MediaItem = MediaItem.fromUri(uri)
7              player.setMediaItem(mediaItem)
8          }
9
10
11         holder.itemView.note_delete_icon.setOnClickListener{
12             listData.removeAt(holder.adapterPosition)
13             removedPosition = holder.adapterPosition
14             notifyDataSetChanged()
15             onChange(listData)
16         }
17
18         holder.itemView.note_card_main.setOnClickListener {
19             if (listData[position].tipo == NoteType.TEXT) {
20                 MaterialAlertDialogBuilder(ctx, android.R.style.
                       Theme_Material_Dialog_Alert)
21                         .setTitle(listData[position].titulo)
22                         .setMessage(listData[position].info)
23                         .setNegativeButton("Fechar") { dialog, which -> }.show()
24             }
25         }
```

A.6 Interaction with Firebase API

```
1
2  fun writeNewMealWithTaskListeners(data: String, carne: String, peixe: String, dieta
       : String, vegetariano: String) {
3
4          val meal = Meal(data,carne,peixe,dieta,vegetariano)
5
6          database.child("data").child(data).child("meal").setValue(meal)
7                  .addOnSuccessListener {
8                      println("Write was successful!")
9                  }
10                 .addOnFailureListener {
11                     println("Error during write to DB :(")
12                 }
13
14     }
```

```
1
2  fun addPostEventListener() {
3
4          val postListener = object : ValueEventListener {
5              override fun onDataChange(dataSnapshot: DataSnapshot) {
6                  // Get Post object and use the values to update the UI
7                  val post = dataSnapshot.getValue<Meal>()
8              }
9
10             override fun onCancelled(databaseError: DatabaseError) {
11                 // Getting Post failed, log a message
12                 Log.w(TAG, "loadPost:onCancelled", databaseError.toException())
13             }
14         }
15         database.addValueEventListener(postListener)
16     }
```

```
fun getMealFromSpecificDate(data: String,snapshot: DataSnapshot){

        database.child("meals").child(data).get().then(function(snapshot) {
            if (snapshot.exists()) {
                console.log(snapshot.val());
            }
            else {
                console.log("No data available");
            }
        }).catch(function(error) {
            console.error(error);
        });

    }
```

A.7 Event Data Class

```
data class Event(
        var title: String,
        var info: String,
        var type: EventType,
        var start_time: String,
        var end_time: String,
        var date: String,
        var notes: String,
        var chosen_meal: String,
        var isLunch: Boolean,
        var meal_int: Int
){
    override fun toString(): String {
        return "EVENT: title: ${this.title}, info: ${this.info},
        type: ${this.type},start_time:${this.start_time},end_time:${this.end_time},
        date: ${this.date}, chosen_meal: ${this.chosen_meal},
        isLunch: ${this.isLunch}, meal_Int: ${this.meal_int} \\n"
    }
}

enum class EventType {
    @SerializedName("ACTIVITY")
    ACTIVITY,
    @SerializedName("MEAL")
    MEAL,
    @SerializedName("TRANSPORT")
    TRANSPORTS
}
```

Appendix B

Usability Tests Results

B.1 Test Case #1

Table B.1: Test Case #1 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	00:12.58	0	0	0
P2	00:35.99	0	0	0
P3	00:16.28	0	0	0
P4	00:10.51	0	0	0
P5	00:20.13	0	0	0

Table B.2: Test Case #1 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	01:44.64	3	0	1
P2	00:39.49	1	0	1
P3	00:51.50	0	0	1
P4	00:14.10	0	0	0
P5	00:15.70	0	0	0

B.2 Test Case #2

Table B.3: Test Case #2 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	01:22.31	2	1	1
P2	02:26.76	2	0	2
P3	01:12.85	0	0	2
P4	00:38.45	0	0	1
P5	00:54.80	0	0	1

Table B.4: Test Case #2 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	02:10.78	1	0	3
P2	00:48.50	1	0	1
P3	00:47.50	0	0	1
P4	01:04.05	0	0	1
P5	00:40.06	0	0	1

B.3 Test Case #3

Table B.5: Test Case #3 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	00:20.50	0	0	0
P2	00:34.48	0	0	0
P3	00:31.85	0	0	0
P4	00:09.06	0	0	0
P5	00:13.56	0	0	0

Table B.6: Test Case #3 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	00:36.90	1	0	0
P2	00:39.06	0	0	0
P3	00:35.42	0	0	0
P4	00:12.72	0	0	0
P5	00:18.62	0	0	0

B.4 Test Case #4

Table B.7: Test Case #4 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	00:42.68	1	0	0
P2	01:39.42	0	0	1
P3	00:09.89	0	0	0
P4	00:07.24	0	0	0
P5	00:07.38	0	0	0

Table B.8: Test Case #4 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	00:32.66	1	0	0
P2	00:36.32	0	0	0
P3	00:09.89	0	0	0
P4	00:11.28	0	0	0
P5	00:10.93	0	0	0

B.5 Test Case #5

Table B.9: Test Case #5 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	01:11.75	2	0	1
P2	02:23.07	1	1	1
P3	01:03.60	0	0	1
P4	00:14.83	0	0	0
P5	00:32.79	0	0	0

Table B.10: Test Case #5 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	01:40.50	1	0	2
P2	02:00.76	0	0	2
P3	01:23.37	0	0	2
P4	00:47.67	0	0	1
P5	00:48.95	0	0	0

B.6 Test Case #6

Table B.11: Test Case #6 (touchpad modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	01:37.43	0	0	1
P2	02:00.67	0	0	1
P3	00:55.12	0	0	1
P4	00:34.53	0	0	1
P5	00:55.08	0	0	0

Table B.12: Test Case #6 (voice commands modality).

Participant #	Task time (min:ss.cs)	Deviations	Errors	Assists
P1	02:51.12	1	0	3
P2	01:37.96	0	0	2
P3	01:25.23	0	0	2
P4	01:05.50	0	0	1
P5	01:16.25	0	0	1