TÉCNICO LISBOA

Vithea

Development of a Mobile Application for Remote Speech Therapy

Guilherme Jorge de Lemos Marques

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Dr. Alberto Abad Gareta
Anna Maria Pompili

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Dr. Alberto Abad Gareta
Member of the Committee: Prof. José Alberto Rodrigues Pereira Sardinha

Dezembro 2019

Acknowledgments

Countless acknowledgments to all the work, research, and support that helped me complete this work. This thesis also would not have been possible without the kind support and help of many individuals.

I cannot express enough appreciation and gratitude to my thesis supervisors Dr. Alberto Gareta and Anna Pompili for their continued and endless expertise guidance in making this thesis possible.

I am also highly indebted to all the faculties and colleagues of Informatics Department for providing necessary knowledge and support towards the successful completion of my studies in Instituto Superior Técnico.

I must also thank the people who participated in the usability testing of the application to guarantee that the system is working properly, and also for giving their valuable suggestions to improve the application within a limited time frame.

I also want to thank my brother, Duarte Marques, for his constant and valuable feedback on the mobile application, and for his words of encouragement.

Finally, my special thanks go to Pema Chentsho, my girlfriend, without whom this thesis would not be possible. Her advice, support and encouragement were invaluable to the completion of this work.

Abstract

Communication is an essential part of human interaction. Aphasia is an acquired speech disorder that disrupts communication, depending on the type, it can affect reading, writing and even speech comprehension. With the help of technology, patients suffering this condition can have an easy and reliable way to complement their therapy. Currently there are two different prototypes available in Portuguese medium, which will be used as a baseline for this project. They already include a raw implementation of the required functionalities, however, they will need to be re-engineered to comply with current trends of software architectures and also suit today's mobile user experience. The objective of this project is to build an Android mobile application for patients with aphasia (the target audience would be composed mainly by elderly people), so the application developed is user-friendly, responsive, and adaptable to different screen sizes.

Keywords

Mobile Application; Remote Speech Therapy; Aphasia; VITHEA.

Resumo

A comunicação é uma parte essencial da interação humana. A afasia é um distúrbio da fala adquirido que interrompe a comunicação, dependendo do tipo, pode afetar a leitura, a escrita e até a compreensão da fala. Com a ajuda da tecnologia, os pacientes que sofrem desta condição podem realizar um tratamento mais fácil e confiável, podendo assim complementar a sua terapia. Atualmente, existem dois protótipos disponíveis em português, estes seram utilizados como base para este projeto. Ambos incluem uma implementação básica das funcionalidades necessárias, no entanto, precisam de ser modificados para ter em conta as tendências atuais de arquitetura de *software* e para que se possam adequar à experiência de utilizador móvel atual. O objetivo deste projeto é criar uma aplicação móvel em Android para pacientes com afasia (o público-alvo seria composto principalmente por pessoas idosas), devido a estas condições, a aplicação desenvolvida deverá ser fácil de usar, ter uma capacidade de resposta rápida e ser adaptável a diferentes tamanhos de ecrã.

Palavras Chave

Aplicação Móvel; Terapia da Fala; Afasia; VITHEA

Contents

List of Figures

List of Tables

Listings

Acronyms

AAC Augmentative and Alternative Communication

AI Artificial Intelligence

API Application Programming Interface

APK Android Package

ASD Autism Spectrum Disorder

ASR Automatic Speech Recognition

CSS Cascading Style Sheets

DAO Data Access Object

DSL Domain Specific Language

EU European Union

EEA European Economic Area

GDPR General Data Protection Regulation

HQL Hibernate Query Language

HMM/MLP Hidden Markov Models/ Multi-Layer Perception

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure

IDE Integrated Development Environment

IoC Inversion of Control

JSON JavaScript Object Notation

JSP Java Server Pages

MVC Model View Controller

ORM Object Relational Mapping

OS Operating System

POJO Plain old Java object

POM Project Object Model

REST Representational State Transfer

RDBMS Relational Database Management System

RPC Remote Procedure Call

SOA Service-Oriented Architecture

SOAP Simple Object Access Protocol

SLT Speech and Language Technology

SQL Structured Query Language

SUS System Usability Scale

TDD Test Driven Development

TTS Text-To-Speech

URI Uniform Resource Identifier

URL Uniform Resource Locator

UI User Interface

UX User Experience

VITHEA Virtual Therapist for Aphasia Treatment

VITHEAWS VITHEA Web Services

XML Extensive Markup Language

1

Introduction

Contents

This thesis presents a native Android mobile application called **Virtual Therapist for Aphasia Treatment (VITHEA) 2.0**, this application will include two modules, one for the patient and one for the clinician- It will be developed based on existing mobile and web applications for aphasic patients and their clinicians.

Aphasia creates an inability to produce or comprehend spoken or written language, although the level in which it is presented differs. People who suffer from aphasia need to rely on some form of gesture or image communication [4]. Aphasia is a chronic condition and thus cannot be cured. However, with constant speech-language therapy, partial spontaneous recovery is possible. The extent of the recovery can vary, some abilities return after a few days or up to a month, but usually some residual disorders remain.

To obtain the best results, frequent and constant speech therapy sessions are recommended for patients with aphasia [5]. Hence, the application developed aims to complement standard speech therapy sessions by providing many speech exercises, to practice remotely with a virtual therapist. The virtual therapist is an animated character, that presents exercises and provides feedback on how the exercises were performed. The application is designed to be simple, user friendly and offer a personalized experience. These new features and functionalities will make the application an easy and accessible tool for remote speech therapy for aphasic patients.

1.1 Aphasia

Aphasia is a neurological condition caused by a brain injury, which damages the portions of the brain that are responsible for language [1]. Speech rehabilitation depends on the severity of the brain injury, some people may require months to recover, while others may need years. Therefore, it is very important to constantly do speech therapy to improve language skills.

Aphasia can be categorized by speech fluency into fluent and non-fluent [6] and further categorised depending on aphasic patient's ability to comprehend spoken messages and repeat words or phrases.

1. Fluent

    - **Wernicke's aphasia**: This type of aphasia affects speech causing difficulty in producing coherent sentences. The words pour out of the patient's mouths but the sentence has no meaning for the listener. Wernicke's aphasia also affects reading and writing capabilities.
    - **Anomic Aphasia**: Those who suffer from anomic aphasia cannot find the words they want to use. They use similar words or fillers such as "stuff" or "thing", because of this the patients also have problems finding the correct word when writing. They understand speech and can usually read.

2. Non-Fluent

- **Broca's Aphasia**, this type requires an extraordinary amount of effort to say words or form sentences coherently. Patients with this kind of aphasia have limited vocabulary and due to this they have problems in finding the words they want to use, but they can understand speech.
- **Transcortical aphasia**, its categorized by reduced speech, but good auditory comprehension. Individuals who suffer from it can repeat long and complex sentences.
- **Global Aphasia**, the most severe form of aphasia, patients cannot read or write, they have an extremely limited vocabulary and sometimes don't understand speech.
- **Primary Progressive Aphasia**, it is a form of dementia, where people lose the ability to speak, write, and read over time.

The following Figure 1.1 illustrates a layout of different types of aphasia as explained above.

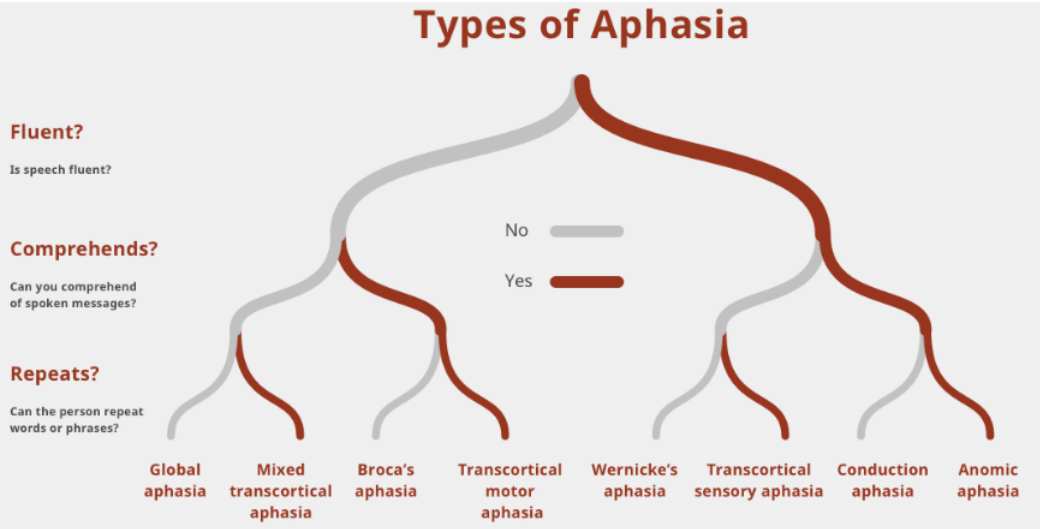

Figure 1.1: Types of Aphasia [1]

1.2 Motivation

Portugal's resident population has been aging continuously, as a result of a decline in fertility and an increase in longevity [7]. Life expectancy of Portuguese population is increasing due to a rise in advanced age of survival. Life expectancy after 65, in the 2015-17 period, has maintained a positive trend, at 65 the average life expectancy is 19.45 years. Due to the aging population, and a lack of healthy lifestyle habits, there is an increasing percentage of the population suffering from strokes, especially those over

65 years old [8]. Around 30% of patients who have suffered from stroke are diagnosed as being aphasic [9]. Consequently, aphasia will most likely become more prevalent in aging population. As mentioned earlier, speech therapy is the best way to help patients with aphasia and technology could be a valuable resource, to provide a fast and reliable way of complementing the therapy remotely.

There are plenty of web and mobile applications that complement therapy, but the majority of them are in English language. VITHEA is a Portuguese project started with the aim of providing easy remote speech therapy through a web application system to Portuguese aphasic patients [2]. However, such web applications cannot leverage device utilities which provide easier user input and output resulting in an immersive and personalized user experience. Additionally, it is browser dependent and also takes an extra effort from the user to type in the URL of the web application, resulting in a bad user experience. Therefore, to extend and optimize the productivity of VITHEA's noble cause, the system was extended to the Android platform via mobile application. Nonetheless, the existing mobile application is regarded as a proof of concept and it only incorporates the basic features that are important for a correct and complete interaction. It needs re-engineering to keep up with new software architectures and the functionalities. Hence, a mobile application will be developed based on the existing VITHEA web application to address all the shortcomings, as well as, to add new features to current prototype of Android mobile application. The new mobile application VITHEA 2.0, intends to be more interactive, intuitive, personalised and to run seamlessly.

VITHEA has a functioning web based system, but the new native mobile application for Android platform will facilitate VITHEA to extend its services to aphasic mobile users. Nowadays, smartphones are widely available; in 2018, 52.2% of all worldwide online traffic was generated through mobile phones [10], besides it is estimated that users spend 87% of their time on mobile apps and just 13% on mobile websites [11]. Smartphones are preferred over desktop computers due to their touch-screen gestures, features, portability and ease of use, even if the tasks were performed more efficiently on the computers [12]. VITHEA 2.0 application will implement all the functionalities and modules supported by the web-based VITHEA platform and make the process of virtual speech therapy available in Android devices, thus taking advantage of the mobile features and encouraging a frequent engagement.

1.3 Objectives

The main objective of this thesis is to develop a native mobile application for Android platform that resembles the functionality offered by an on-line browser dependent system called VITHEA, with the following main functionalities:

- Develop an application with two modules, a patient module, where the patient will be able to exercise and a clinician module, where the clinician will be able to monitor his/her patients.

- Integration of a virtual therapist with speech synthesis capabilities. The virtual therapist is developed as an animated character and is provided as an external library.
- Reproduction of different types of stimuli (audio, video, images) and easy navigation among them.
- Acquisition of the voice signal through the device microphone and communication with the server where the automatic speech recognition engine is located.
- Adaptation to screen rotation and different screen sizes including tablets.
- Add common functionalities such as settings menu, feedback prompts and other features to design a simple and suitable application to meet the current trend of user experience.

1.4 Organization of the Document

The remaining of this document is organized as follows - Chapter 2 discusses the related works and the state-of-art web and mobile applications addressing aphasia. Then, the different methodologies for software development are compared to choose the most suitable methodology and technologies for this work. Chapter 3 explains in detail the architecture, User Interface (UI) and features of the client application as well as the software dependencies of VITHEA 2.0. The developed client application required a re-engineering of the server application, which is described in Chapter 4. Chapter 5 examines the evaluation methods that will be used for measuring the performance and quality of the application developed and also presents the results achieved to verify if VITHEA 2.0 application is adequate or not for aphasic patients. Finally Chapter 6 reports on the conclusions of this work and also mentions possible future works.

2

Related Work and State of Art

Contents

This chapter presents the related work and the state of art that addresses aphasia, by providing a remote virtual speech therapy. Firstly, it discusses the most related work such as web or mobile applications that address aphasia in European Portuguese language. Then, it also explores the state of art mobile applications for adult aphasic patients from Google Play Store, ordered by number of downloads and classification (regardless of their language). Finally, it explains different state of the art methodologies to develop mobile applications and which methodology will be chosen for this thesis.

There are several types of mobile applications for aphasia with Augmentative and Alternative Communication (AAC), which help the patient to communicate on a daily basis by providing; for example providing flash cards with words they can use. However, for this work only speech therapy applications will be considered and analysed.

2.1 VITHEA Web Application

"VITHEA is the first prototype of an on-line platform that incorporates Speech and Language Technology (SLT) for the treatment of Portuguese speakers with aphasia [2]." VITHEA is a web-based platform aimed to behave like a virtual therapist to help the user complement their training sessions by leveraging the predetermined exercises online. The user will just have to access the VITHEA online platform through any web browser, thus allowing them to practice exercises at home [2].

The following Figure 2.1 depicts an overview of the VITHEA platform and how it functions to enhance the speech recognition therapy for Portuguese speaking patients with aphasia. It is based on a client-server architecture, where the client side of VITHEA's platform uses Adobe Flash technology to take inputs from patients as they answer the exercises presented. These answers are then encoded and sent to a web application server (Java Server Pages (JSP)/Servlet) for processing.

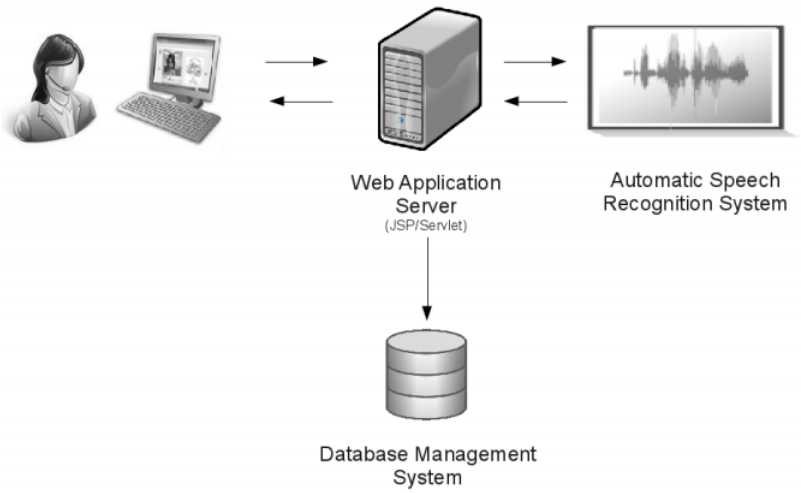

Figure 2.1: Overview architecture of VITHEA online platform [2]

The virtual therapy process with VITHEA platform begins by showing a series of visual and auditory stimuli using the display and the speaker of the user's device respectively. Then, the user records the answer using the microphone by recognizing and naming the content that is presented to them. The recorded input of user is then encoded and sent to the server side in order to verify the correctness of the user input. Once the server receives the encoded input, server sends it to Automatic Speech Recognition (ASR) system for decoding and then generates textual representation of the user's encoded answers. Next, it compares the decoded answers with the set of predetermined textual answer for a given set of questions stored in the database system. When ASR finishes verifying the answer, it will send the feedback to the user [2].

The ASR system is the backbone for VITHEA, as it can decode the user's encoded input, validate the answers and then finally sends the feedback. This system has two major modules as follow:

(i) **The baseline speech recognize** VITHEA uses a version of AUDIMUS, a platform for an Automatic Speech Recognition System, which includes acoustic model trained in Portuguese, over 56 hours of sampled broadcast news data, and 58 hours of mixed telephone data.

(ii) **Background speech modelling** with Hidden Markov Models/ Multi-Layer Perception (HMM/MLP) recognizer, computes the posterior probability of the background speech as an average in the likelihood domain [2].

Virtual speech therapy in VITHEA platform is done through a three-dimensional (3D) game environment with speech synthesis capabilities in a web-based application where users can try to recognize and name the content they are asked. There are two types of client modules in the web application, one for the patient and another for the clinician to exercise remote therapy. Clinician's module is the administrative module where a therapist can create and manage exercises and see how the users are progressing with privileges to choose the resources for the exercises. In the patient's module, there are a set of exercises to answer presented by an animated character, called the virtual therapist. The answers are recorded with the help of the device's microphone and uploaded to the server. The answer is verified by comparing and matching it to a set of predetermined answers stored in the database, and the feedback (”very good” or ”try again”) is sent to the user depending on the correctness of the answer.

Nonetheless, web application technology limits the extension of the framework to mobile devices to scale up. Web applications need constant updating to comply with the browsers updates. When building a mobile application there is a minimum version required, but from that point on, the need for updates is smaller and easier to manage. A Web-app based system is restricted to the browser's features such as the *back* and the *refresh* buttons to perform even the most fundamental functionalities, as it cannot leverage the advance mobile gestures such as *tab*, *double tab*, *swipe*, *pinch*, *hold*, etc. Therefore, VITHEA extended its web-based system to mobile application which is discussed in next Section 2.1.

VITHEA mobile application was designed and developed to enhance and scale up the user experience, taking advantage of mobile based features and functionalities. Being a mobile application means that the user can take his phone with him and practice speech therapy anytime, anywhere, making the application more accessible. However, the mobile application is still in a early stage of development and it needs software re-engineering, which is the exact motivation of this thesis.

2.2 VITHEA Mobile Application

A native VITHEA Android mobile application was developed to complement and overcome the shortcomings of the web based platform and also to extend its service to mobile users. However, the Android application that was developed is just a proof of concept and not fully operational at this point. The application uses version 2.3.3 of the Android mobile operating system which was released on September of 2011, as of May of 2019 this version is only used by 0,3% of the users [13]. Due to the use of an old version and, lack of functionalities, it is necessary to improve it by producing a new mobile application that could be used by a broader user base and have more functionalities.

The application uses Representational State Transfer (REST) architectural style. This kind of architecture defines a set of constraints to be used for creating web services, such as the identification of resources, manipulation of resources through representations, self-descriptive messages, and hypermedia as the engine of the application. Any information can be a resource, each resource and interconnection are uniquely identified and can be addressed with a Uniform Resource Identifier (URI). REST components communicate through the transfer of data representation in the standard data type format. Self-descriptive messages requires a request from any client ought to contain all of the information necessary to fulfill the task.

Hypermedia as the engine of application state, which means clients can move from state to state via URIs. In RESTful, application data is exchanged through Hyper Text Transfer Protocol (HTTP). Due to the need of exchanging complex data types that represents the system state information, the data objects need to be serialized into text, usually represented in JavaScript Object Notation (JSON) or Extensive Markup Language (XML). JSON was chosen as the data format for the exchanging the information between client and server. The serialization process takes place upon sending and receiving the data.

The **Authentication** is implemented by a simple basic authentication over Hyper Text Transfer Protocol Secure (HTTPS). When accessing the system, the user writes his credentials which are then sent to the server. In each of the following requests, data is then stored in the client application for the entire execution time. The information sent to the server is encrypted, and added to the authorization header field. On the server side it is compared with the encrypted version of the same data that resides in the

persistent storage support. If the credentials are correct, the user is granted the access. The access restriction to a given resource is done at the configuration file level.

In the **Server Side**, Spring Security and Spring Web Model View Controller (MVC), were implemented. Spring Security is a non-intrusive framework that focuses on providing both authentication and authorization to Java applications, it is easily extendable and can meet custom requirements. Spring MVC is a framework that helps in the development of web applications and REST services.

In the **Client application**, Spring for Android has been used. This is an extension of the Spring Framework that aims at simplifying the development of native Android applications. It includes a REST client that provides higher level functions and several conversion functionalities for the various data representations supported. It also provides support for integrating Spring Social functionality, which includes an OAuth based authorization client, although this part was not explored. The audio response from user is acquired and recorded using the microphone; when the recording stops this audio is sent to the server through a RESTful POST request. Then the speech engine AUDIMUS processes the file and verifies if the answer provided was correct or not, then the result is returned to the user [14].

The developed prototype only incorporates features that are important for a correct and complete interaction, such as integration of the recognition process, the virtual therapist character, and authentication. The native module of the virtual therapist, which is done in the game engine Unity, is exported and then integrated into the Android application by plugins. The exercises include some video, audio and text, but the application cannot display some of the videos and audio files due to them not being supported by the Android platform.

The following Figure 2.2 depicts the UI of the application.

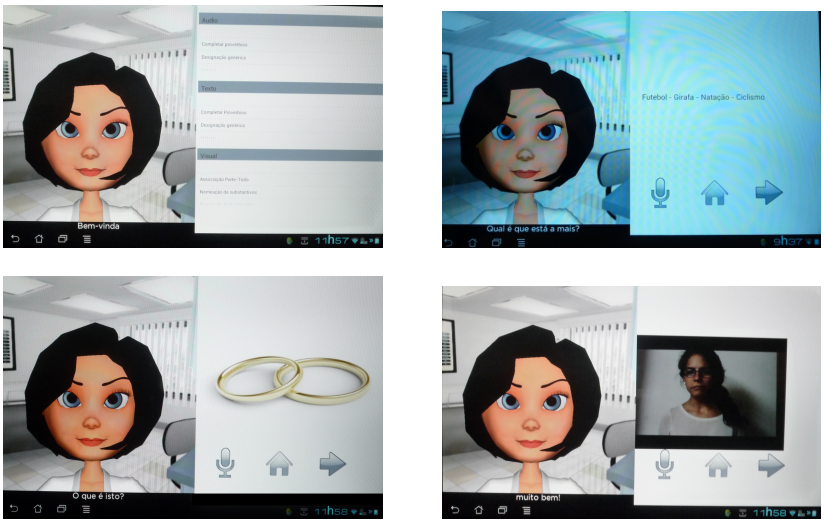

Figure 2.2: Stimuli for exercises (top left), Text exercise (top right), Picture exercise (bottom left) and Video exercise (bottom right) [2]

On the top left of Figure 2.2 is the main menu where a patient would select which category of exercises he would like to do. On the top right there is an example of an exercise from the text category. On the bottom left and bottom right are examples of the visual category, the first being an image and the second a video exercises.

VITHEA 2.0 application will be developed based on this application and architecture, it will be improved by adding new features and functionalities to suit today's mobile user experience, which are discussed thoroughly in Chapter 3. The new application will have two modules, one for the patient and one for the clinicians, while the previous version of VITHEA was developed only for the patients.

2.3 VITHEA Kids - Virtual Therapist for Autism Treatment on Children

VITHEA also has another similar application called VITHEA Kids which was re-engineered to help children with communication disorders caused by Autism Spectrum Disorder (ASD) by providing a low cost solution to improve access to therapy. It aims to provide enough exercises and custom options to both the users (kids with communication disorder) and their caregivers, such as therapist, doctors, parents or family members who can assist and help to personalize it. The caregiver can interact by creating personalized exercises in Portuguese as per the kid's need. Figure 2.3 and Figure 2.4 show how two options for the caregiver's and the child modules of the VITHEA Kids web-app looks like respectively [3].

VITHEA kids is based on VITHEA's framework, and it has both web-app and an Android mobile-app to enhance its functionality and optimize the features. The mobile application is able to run on 5.x Android versions and above. VITHEA Kids main focus is to help children with ASD to improve their language and communications skills by practicing the exercises such as image identification, matching an image with expression/words, and naming the content. The exercises are created on the caregiver's module and the user practices and solves the problem on child's module.

The edit exercise option on the **caregiver's module** as shown in Figure 2.3 (left), allows them to create exercises by providing a topic (*e.g. Fruits*), difficulty level (*introductory, intermediate or advanced*), questions/instructions, options for answers and also the correct answer(s). The caregiver can change the child's information such as name, date of birth and gender, as well as, list and edit the existing exercises by pressing the edit children information option as shown in Figure 2.3 (right). Apart from editing and choosing exercises, caregivers can also edit the animated character (the virtual therapist) used for greeting, congratulating and presenting the exercises to the user [3].

The **caregiver's module** is a web application which was developed using Apache Tomcat with a MySQL database to store information and follows a MVC architecture. The child's module is an Android application and it uses HTTP to send requests to the Application Programming Interface (API) [3].

Figure 2.3: Edit Exercises (left) and edit child information (right) options on a Caregiver's Module [3]

On the **Child's module** as shown in Figure 2.4, there is an option to select the type of exercise a child wants to learn or play. On selection, the virtual therapist presents the question to the user, with a set of options to choose the correct answer from. If the user chooses the correct answer in the first attempt, the reinforcement image will show up, all the distractions surrounding the answer will disappear to help the user choose the right answer as seen in Figure 2.4.

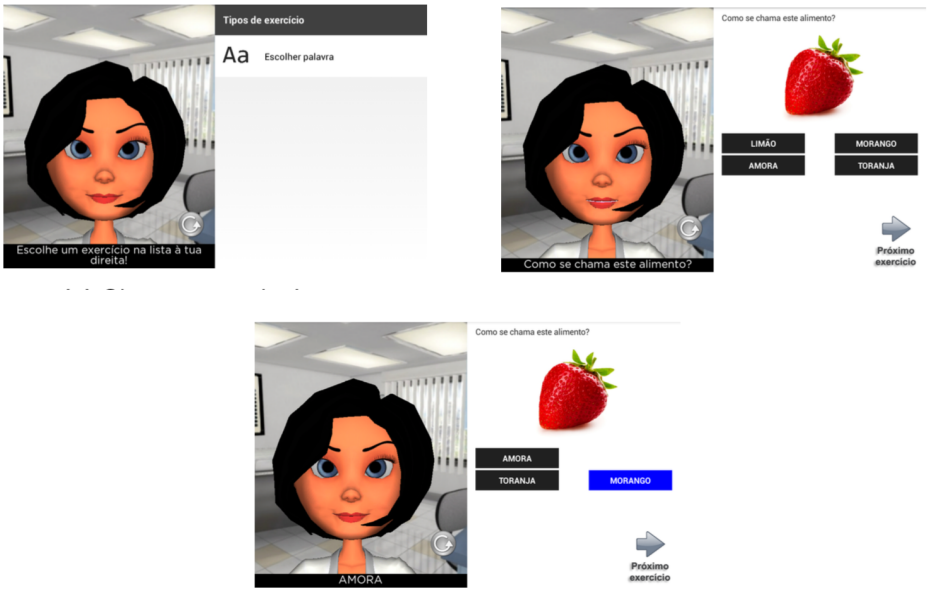

Figure 2.4: Choosing the exercises (top left), the exercise (top right) and prompting (bottom) options on a child's Module

VITHEA Kids mobile application is similar to VITHEA 2.0 application developed for this work, as both of the applications are based on VITHEA system and architecture. Therefore, both of them have a client-server architecture in order to deliver a virtual remote speech therapy to help patients with communication difficulties. While both applications have two modules, VITHEA Kids clinician module is a web application and in this project the module was developed as one mobile application with two different modules, making it easier for the clinician to access and monitor his patients.

The applications are developed for patients with different kinds of communication disorder, hence the UI development focuses on their particular patient and their needs. Some of the differences between the two applications are:

- VITHEA Kids application has a reinforcement feature to help the child identify the correct answer using animations and audio, where as for VITHEA 2.0 application has visual feedback prompts with changing background colour.

- VITHEA Kids mobile application was developed only for children, the caregiver is a web application, while in VITHEA 2.0 the clinician is a module within the same application.

- VITHEA Kids does not have the functionality to record voice as input from the patient, while in VITHEA 2.0 all the exercises require recording of the patients voice to complete.

- In the server side VITHEA Kids uses Play Framework, this is an open-source web application framework which follows the MVC architecture. This project uses Spring Framework, a comparison between Spring Framework and Play Framework performance was inconclusive [15]. Play is a better option for Scala developers, besides the previous version of VITHEA web server used Spring, and this framework is still widely used, due to all this reasons Spring Framework was maintained as one of the main components of the server.

2.4 State of Art Mobile Applications for Speech Therapy

This section will draw attention to the most popular mobile application for aphasic patients available in Google Store, ordered by number of downloads and classification, regardless of the language used. These applications will be evaluated according to its UI - it must be simple and user friendly, which means the applications must be easy to learn, use, and understand. The interactiveness with the user will be another factor to evaluate, along with the types of exercises provided. The application will be compared with VITHEA 2.0, plus some suggestions will be provided on how to improve the applications.

2.4.1 Constant Therapy

This application has more than 100,000 downloads, a rating of 4.2 and has been voted 443 times [1]. It provides several exercises to help the patient with speaking, writing and comprehension to patients with aphasia, dementia, and Alzheimer's disease. The application is a free for 15 days trial, and then a subscription mode must be chosen.

The application begins with an assessment of the patient, where it asks patients goal, what he would like to improve upon and it also provides metrics on each of the exercises performed, as depicted in Figure 2.5. The application tracks patient's progress to help therapist and the patient know the improvements and also where they might need an extra effort. Constant Therapy exercises adapt to the patient's ability and get harder or easier based on their improvements to keep challenging the patients to meet their rehabilitation goals.

In regards to UI, it's very user friendly - the simple mode has everything the user would want, as well as some therapy options, and the advanced mode would be useful for the therapist to set or change exercises. It also provides reports on the tasks completed, and many more metrics as shown in Figure 2.5 below. However, the advance mode would be hard for an inexperienced user to navigate.

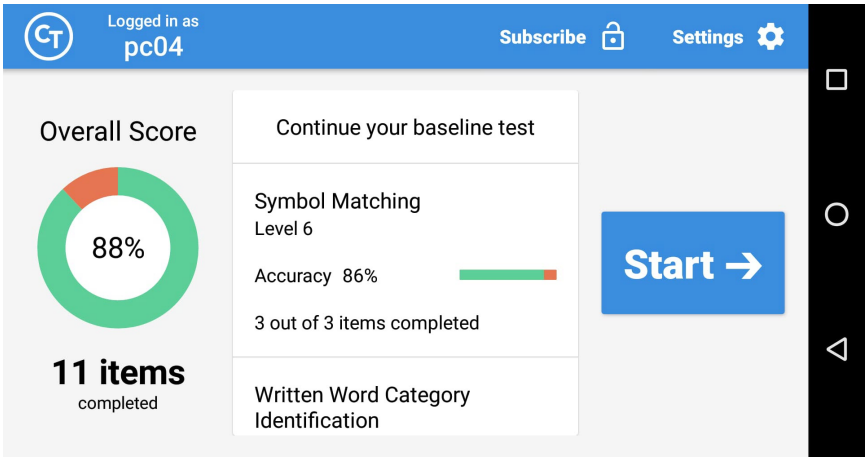

Figure 2.5: Main Menu for Constant Therapy application

Constant Therapy also has a built-in speech recognition system, and has a useful volume meter that allows the patient to know if he is audible. It also contains different kinds of featured exercises such as alternation symbol matching, picture N-Back memory, Functional Math, Auditory Command, Sentence Picture Description and Functional Reading for improving patient's attention, visual memory, arithmetic, auditory processing, speaking and reading respectively.

The list of exercises types is very extensive, so only the main categories will be presented here as following:

[1] Information obtained from Google Play in 25th October 2019

- **Language**: In this category there are exercises for reading, speaking, writing, word retrieval and more.

- **Cognitive**: This category includes exercises for attention, memory (auditory and visual), arithmetic and reasoning.

- **Everyday Skills**: This category is a more general category amongst others, that contains exercises like clock reading and math, currency math, reading exercises, picture naming, and map reading.

- **Assessment**: This is the first task the user does when he opens the application for the first time, the exercises are selected by what the patient wants to improve and then its modified according to the results obtained.

Clearly Constant Therapy is the result of many hours of development work, VITHEA 2.0 application will have some of the mobile features like Constant Therapy in-terms of personalising as well as customizing the mobile application to the patient's needs and also meet current trend of mobile user experience. However, it would be unfeasible to try to match this application, besides VITHEA 2.0 will focus only on patients with aphasia and it will provide speech therapy with a limited number of categories. Another distinction is that VITHEA 2.0 will be focused in the Portuguese medium while Constant Therapy it is only available in English.

2.4.2 Language Therapy

This application has two versions one lite (trial version) and another with a complete version. Combined they have more than 10,000 installations, the rating for the trial is 3.9 with 48 votes, and the complete version has a total of 1000 downloads, a rating of 4.9 but only 15 users voted [2].

The UI is user-friendly, the menu is well designed as well as the exercises, but all of them seem very similar and there is not much distinction between them. Before each exercise, there is an explanation of why that particular exercise will help the patient. Language Therapy targets the improvement of reading, writing, speaking and listening skills using the four categories as explained below, which are also depicted in their main menu, as illustrated in Figure 2.6.

- The **comprehension** module has exercises for matching pictures to written words and matching pictures to words that are previously recorded.

- The **naming** module has flashcards describing objects, and naming practices.

[2] Information obtained from Google Play in 25th October 2019

- The **writing** module has different types of difficulties, the exercises consist of filling out the blank or spelling what the patient is currently seeing.

- The **reading** module has exercises to match the picture to a word or a sentence. The patient does not interact by reading, just by selecting the correct word or sentence.

Figure 2.6: Main Menu for Language Therapy application

The exercises are well done, and even have hints if the patient requires some extra help, but there is no speech recognition like in Constant Therapy application discussed in section 2.4.1. The only option is to record the voice and then compare what was said to what the device has recorded. Language therapy operates in English, Spanish, French and German.

In comparison VITHEA 2.0 will have a speech recognition system, and will allow the user to customize the application. For now VITHEA 2.0 will be in European Portuguese, although there is an experimental language feature with English, but it is not complete, for example the virtual therapist speaks in Portuguese, this opens the application so that in the future it might be possible to have more languages.

2.4.3 Aphasia Speech Therapy

This application has more than 10000 device installations, and has a rating of 4.0 with 52 votes [3]. The UI is simple and easy to use, but since the target audience of this application are mainly elderly people, the letters would be too small on mobile phones for reading and there is no option to change the font size.

There are 3 modules in this application: auditory, vocal and recognition as shown in Figure 2.7 below, which are then explained later.

[3] Information obtained from Google Play in 25th October 2019

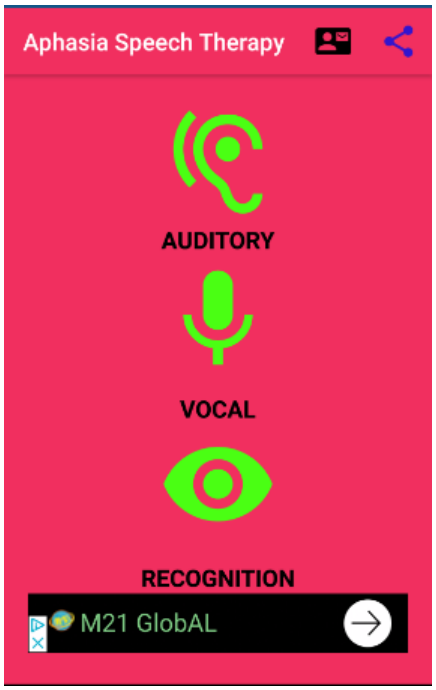

Figure 2.7: Main menu for aphasia speech therapy

- The **vocal module** has several options - the patient could write a sentence and then say it back, or repeat the pre-made statements and expressions. Other functionalities of this module are conversations and stories but in these options there is no interaction from the user part, s/he can only just read or listen to what is being said.

- In the **auditory module** the user can listen to alphabet letters, numbers, colors, feelings and shapes.

- In the **recognition module** the user has to recognize and write letters, numbers or sentences shown in the screen. In this module there are a couple of problems regarding UI, first the letters, number and sentences disappear from the screen after some time. The other issue would be that in order to get another word you need to press the refresh button on the page, this could be done automatically and would significantly improve user experience.

Furthermore, it has some bugs when the user tries to rotate the screen as it does not scale, and the scroll does not work in this mode. Google speech recognition is used in order to process the audio. On the plus side it is a free application, although it has some advertisement. VITHEA 2.0 adapts to screen rotation, has more exercises type and features compared to Aphasia Speech Therapy application. Both the applications are used to improve the speech capabilities of aphasic patients.

2.4.4 Afasia Anómica

This application has more than 1000 device installations, and has a rating of 4.2 stars, this rating can be misleading because there are only 6 voters [4].

Regarding the UI, it is very easy to use, but the colors could be better, only white and grey are used throughout the application. This application also does not have any speech recognition system. A list of the category of exercises can be seen below and the main menu of the app is presented in Figure 2.8.

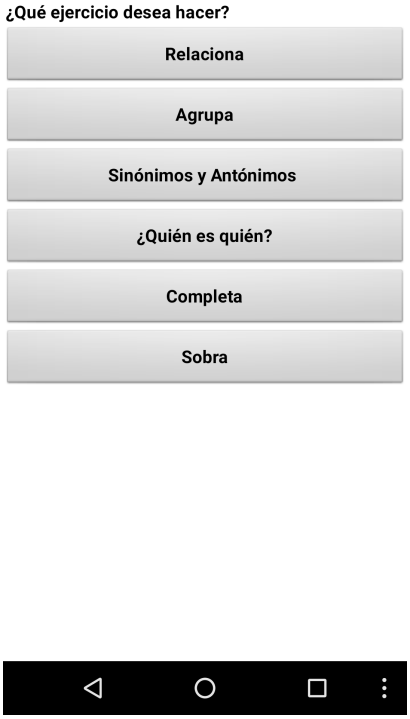

Figure 2.8: Main menu for afasia anómica

- **Relational Exercises**: There are several levels of difficulty in this category and the patient needs to relate the images with the text.
- **Group Exercises**: In this category, there are columns with different group/category and the words are scrambled throughout the columns and the patient needs to place each word to the correct column. For example, days of the week and months.
- **Synonyms and Antonyms**: This category consists exercises to find the synonyms or antonyms of words.
- **Who is Who**: A description is given on how a person looks like, and the patient needs to select the image that resembles the person described in this category.

[4] Information obtained from Google Play in 25th October 2019

- **Surplus Exercises**: This category unlike Group Exercises category, the patient needs to select in each column a word that does not correspond to the category.

This application also does not resize well, and all the options (categories) are very close together leaving a lot of space in the end of the menu as shown in Figure 2.8, and there are also not many exercises under each category. It plays a very unpleasant sound when the patient makes a mistake or give a correct answer, and it cannot be deactivated.

2.5 Comparison of mobile application development

It is important to explore different methodologies for mobile development, so that the best option can be selected according to the requirements. By performing a trade-off between them, it can be ensured that the optimal method is chosen based on the objectives of the project. Four different types of mobile applications are discussed and compared first, and after a trade-off between objectivity and relevance, a methodology will be selected for the development of VITHEA 2.0.

Native Applications are smartphone applications that are built for an operating system, with a specific programming language, such as Swift and Objective-C for iOS or Java for Android. These native applications are commonly used because they provide fast performance and a high degree of reliability. The main disadvantage is that they are tied to one operating system, and if the developer wants it to work in another platform then he needs to create a duplicate version for that particular platform.

Web Applications are software applications that uses a browser to run and are usually written in Hyper Text Markup Language (HTML)5, JavaScript or Cascading Style Sheets (CSS). This type of application do not require a minimum device memory as all the databases will be on a server, and users would also be able to access it from any device. This can be a disadvantage because in the case where there is no internet the application would not work, or if the internet is slow it can create a bad user experience. Another drawback is that the developer cannot access the device's APIs, with exception of geolocation and a few others. Furthermore it does not provide a fully native experience due to cross-platform UI and the User Experience (UX) design.

Hybrid Applications are built using multi-platform web technologies such as JavaScript, HTML and CSS. This type of applications are web applications wrapped in a native shell, this is done because this way they can be downloaded from the correspondent application store. This has an advantage because with few modifications, the same code can be used in different platforms, saving time and money. One of the disadvantages of this type is that it is slower than native apps, less interactive, and has a limited access to platform specific features.

Cross-Platform Native Apps, some frameworks like Xamarin [16] and Flutter [17] provide the ability to share the same code base between a number of platforms. Nevertheless, when another layer is

added it can be a source of bugs, and it also has to be updated constantly with every new update from each of the platforms it supports. The performance wouldn't be the same as the native applications and it wouldn't have access to the same APIs.

The choice of type of development depends on the requirements, for multiple platforms the best choices would be either cross-platform development or hybrid development. Native application development is clearly superior if the app is only for a specific platform or if its highly complex. To establish a mobile presence for VITHEA, a native application would be developed, since it requires game like environment for interactivity, there is a need to use the microphone of the device, optimization is required for encoding and decoding the recorded answers, and due to interactiveness the response time should be as low as possible.

2.6 Summary

VITHEA 2.0 aims to provide a native re-engineered mobile application (Android) to support the current functionalities of VITHEA's platform, as well as extend the features of the existing VITHEA Android prototype mobile application to provide a personalized remote virtual speech therapy for patients with aphasia in Portuguese medium.

In this Section 2.4 several applications were reviewed in terms of user interface, interactiveness with the user and exercises. Having considered all the factors mentioned above, the Constant Therapy application is clearly superior among all of them. It is very easy and intuitive to use, and provides over 100,000 exercises dedicated to cognitive training and speech and language therapy. It also has two modules, one for the therapist and another for the patient. Furthermore, there are several metrics that can be used to track the state of the patient in real-time and the exercises adapts to patient's ability. In the advanced mode, it is even possible to create detailed reports about each exercise the patient has done. A major downside for this app is that, it is a subscription based model which makes it expensive for some patients. Furthermore this application only functions in English.

The majority of the applications reviewed are in English, except for Afasia Anómica that is in Spanish. However, this application is very limited in terms of exercises, and does not provide any verbal interaction with the user regarding speech therapy. VITHEA 2.0 would be the only remote speech therapy in European Portuguese that could help patients who have Portuguese as their native language.

The following Table 2.1 summarises the different features between all the analyzed applications and VITHEA 2.0.

Table 2.1: Features comparison in the analyzed applications

Applications	Features					
	Portuguese	User Friendly UI	Speech Recognition	Word Naming Exercises	Writing Exercises	Multiple Choice Exercises
VITHEA 2.0	✓	✓	✓	✓		
Constant Therapy		✓	✓	✓	✓	✓
Language Therapy		✓	✓	✓	✓	✓
Aphasia Speech Therapy				✓	✓	✓
Afasia Anómica				✓		

3

VITHEA 2.0 Mobile Application

Contents

In the previous chapters, the problem that this thesis is trying to solve was introduced and the related works were presented, as well as different state of art mobile applications for remote speech therapy. This chapter discuses the implementation of VITHEA 2.0 to help patients with aphasia.

The implementation includes the development of the client and the server applications. The client application includes two modules, one for the patients (focused on the usability and resilience), and one for the clinicians, that are now able to add patients, which are discussed thoroughly in this chapter. With all the changes in the client side, the server application was re-engineered to meet today's industry standards of web applications and to adapt to the new mobile application.

As mentioned in Section 1.3 and Section 2.5, the objective of this work is to develop a native mobile application for a single platform (Android) as it allows direct access to the APIs. This choice allows to efficiently use features of the mobile device, such as the microphone, while also performing faster. Performance is an important aspect since the application needs to have a low response time so that the interactions and the user experience match the desired expectations.

The Android platform architecture is a software stack based on Linux. The Linux kernel is stable, secure and enables Android to use key security features such as user-based permissions model, process isolation, and the ability to remove unnecessary and potentially insecure parts of the kernel. User-based permissions model guarantees that user resources are isolated from other user resources. The process isolation is achieved by the application sandbox, which isolates applications from each other and protects applications and the system against malicious uses. The Android Operating System (OS) is available through APIs written in the Java language, and it is open-source, which means that it is publicly available for use or modification.

The application will be built with Android Studio, this is the official Integrated Development Environment (IDE) for the operating system. Some of the features of the IDE include, a fast and featured packed emulator, and a single environment where its possible to develop code for all Android devices without having to build an Android Package (APK).

Although the previous version of VITHEA application was just a prototype, it included the possibility for the user to perform exercises for speech rehabilitation. Overall, the application was functioning, however there have been many developments in Android over the years and by today's standards the application lacks features to improve the UX and functionalities to personalise the application, like for example showing the progress to engage the user.

3.1 Software Practices and Requirements

This section will first discuss the software practices implemented in this thesis while developing the server and the client application. Next, it discusses the minimum requirements and software dependen-

cies, without which VITHEA 2.0 will not function as it is supposed to. These dependencies are external library modules that are used to architect the new features and functionalities of the application.

3.1.1 Software Practices

The implementation of the following software practices will help the development of future functionalities of VITHEA 2.0:

- **Code Readability**, this is an important practice, code must be readable in order to be easy to understand, this implies naming variables and methods with the intention of revealing their purpose.
- **Project Organization**, as show in Figure 3.1, the code of the project is organized by functionality, which makes finding a class easier and organization clearer.

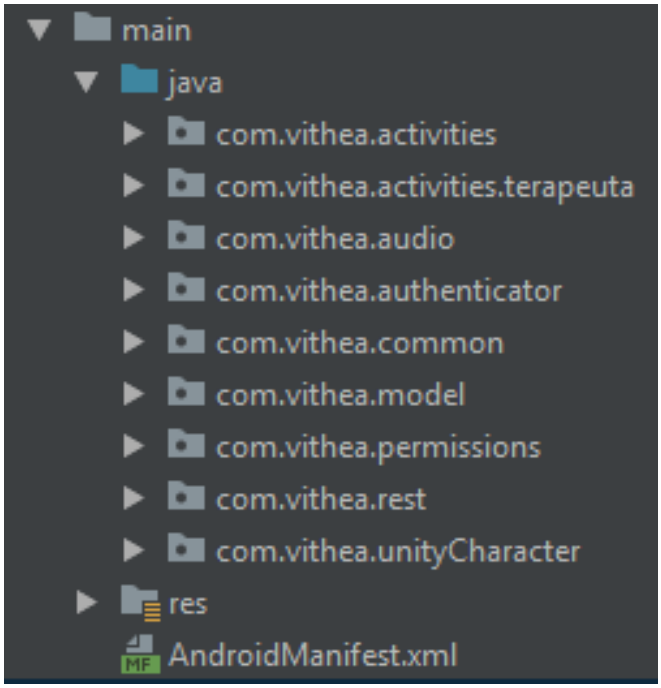

Figure 3.1: Project Organization

- **Refactoring**, which means restructuring existing code without changing its behavior, in order to reduce complexity, eliminate duplicate code, thus improving maintainability.

3.1.2 Minimum Requirements

The **minimum Android version** required, specifies the oldest version supported by the application, this is an important factor to consider as the aim of this project is to produce an application that can be used by the largest amount of users. Table 3.1 shows the Android version distribution in October of 2018 [13].

Table 3.1: Android version distribution in October of 2018

Version	Codename	Distribution
2.3.3 -2.3.7	Gingerbread	0.2%
4.0.3 -4.0.4	Ice Cream Sandwich	0.3%
4.1.x-4.3.x	Jelly Bean	3%
4.4	KitKat	7.6%
5.0-5.1	Lollipop	17.9%
6.0	Marshmallow	21.3%
7.0-7.1	Nougat	28.2%
8.0-8.1	Oreo	21.5%

As it can be seen in Table 3.1, version 5.0 would be a good compromise between recent technologies, APIs, and percentage of users above that threshold. Around 90% of user base could use the application, and with the constant updates of the Android operating system that number would only increase.

3.1.3 Dependencies

This section will discuss important external library modules. Libraries facilitate the process of complex functionality into a few simple lines of code. The client and server applications require them in order to function properly.

The criteria used to select the dependencies on the client application were:

1. **Open source**: There are many advantages if the source code is open source, for example, if the source code is closed, the only developers that can potentially detect, diagnose, triage and resolve software bugs are those that happen to be employed by the company that publishes the software. On the other hand, in the case of open source there will be more people looking at the code, therefore more bugs will be found and fixed in a shorter period of time. Open source projects tend to be more modular in terms of architecture, improving both the flexibility, and the robustness of the code.

2. **Popularity**: If a library is popular it has been used in different environments and use cases, making it more robust and less prone to bugs. Usually popular libraries have an active community, this is an advantage because then if there is any problems while using it, there will be resources to help the developer.

3. **Good documentation**: The library should have good documentation, it is an essential feature for any application, having good documentation makes it easier to develop.

3.1.3.A Dependencies used in Client Application

Gradle is an open-source build automation system, it uses the Groovy-based Domain Specific Language (DSL) instead of the XML for the project configuration. In this project Gradle is used as a dependency manager, it also manages the life cycle of the application.

The main advantages of using Gradle are [18]:

- **High performance**, Gradle runs only the tasks that need to run because their inputs or outputs have changed. In this way there is no need to compile and run all the tasks, increasing performance.
- **JVM foundation**, Gradle runs on the JVM, allowing access to standard Java APIs in the build logic, while also makes it easier to run Gradle on different platforms.
- **Insight**, these are build scans, they can provide useful information about a build run, it can be used to identify build issues, like for example build performance issues.

The client application depends on following dependencies to function:

- **Butterknife**, version 10.1.0 : Android ButterKnife library is a view injection library that injects views into android activity / fragments using annotations. This makes the code much more readable and it can be better structured .
- **Constraintlayout**, version 1.1.3 : A layout defines the visual structure for a user interface, such as the UI for an activity. The ConstraintLayout allows the positioning and sizing of android widgets in a flexible way. In this way, it is easier to provide support for different screen sizes, in fact, the layout will take care of all the dimensions of the objects in the view according to the constraints defined.
- **Shimmerlayout**, version 2.1.0 : Shimmer for Android is also a layout as explained above. It is a efficient and well documented library to add some shimmer effect while the application is loading.
- **Gson**, version 2.8.5 : Gson is a Java library that can be used to convert Java objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. The advantages of Gson include the simplicity of toJson/fromJson and for deserialization, without the need to access Java entities.
- **OkHttp**, version 4.1.1 : OkHttp is an open source project designed to be an efficient HTTP client. It supports both HTTP and HTTP/2, it also implements synchronous or asynchronous calls. If the call is synchronous the project will wait until the message is processed. Thus, since this behaviour could result in an unresponsive Android application, only asynchronous calls are used.

- **Retrofit**, version 2.6.1 : Retrofit is a high-level REST abstraction built on top of the OkHttp library. One of the main advantages of this library is the ease to retrieve and upload JSON via a REST based web services. The implementation of the Uniform Resource Locator (URL) based schema for REST is done with annotations, which makes it simple and readable. It uses modularity, due to this, it is possible to select which converter (XML, JSON) the developer wants to use, in order to process the data serialization and deserialization.

- **Converter-gson**, version 2.6.1 : This library is used by Retrofit, it is a converter that uses Gson for serialization and deserialization of data. As explained previously retrofit can use several converter, this converter was chosen because the project already uses JSON, making it consistent.

- **Converter-scalars**, version 2.6.1 : This library is used by Retrofit, it supports converting strings, as well as both primitives and their boxed types to plain text.

- **Logging-interceptor**, version 4.1.1 : This is a useful library, used in debugging that allows the developer to inspect what is being sent or received by OkHttp. It intercept's and logs HTTP request and response data.

- **MPAndroidChart, v3.1.0** : MPAndroidChart is a library that provides a visual representation of statistics. It provides pie charts, which are used to display statistics related to the exercises performed by patients, as shown in Figure 3.18.

- **Preference v1.1.0** : This library simplifies the process of creating a settings menu, the code is simpler and the developer can define all the attribute in an XML.

3.1.3.B Dependencies used in Server Application

The server application depends on following dependencies to function:

- **Hibernate**, version 5.3.10: Hibernate is an Object Relational Mapping (ORM) for Java programming language. An ORM is a framework for mapping an object-oriented domain model to a relational database.

- **Spring Framework**, version 5.1.9: It is a powerful, lightweight, open source framework Inversion of Control (IoC) container used for application development.

- **Spring Framework Security**, version 5.1.6: This is a non-intrusive framework providing a set of authentication and access-control services.

- **Jackson**, version 2.9.9: Is a high-performance JSON and XML processor for Java. It is a fast and lightweight library.

- **Mysql-connector**, version 8.0.17: It is a library that helps the database connection.
- **Log4j**, version 1.2.16: Log4j is a Java logging library.

3.2 Core Architecture

Figure 3.2 illustrates the system architecture and the interactions between its different component. VITHEA 2.0 Android mobile application has a client-server architecture where the client side uses Andrdoid APIs to take the input from patients and sent it to the server, this architecture uses Spring and Hibernate as the main components. The mobile application is developed in Android studio and has different modules. The different components of the architecture will be briefly explained in this section.

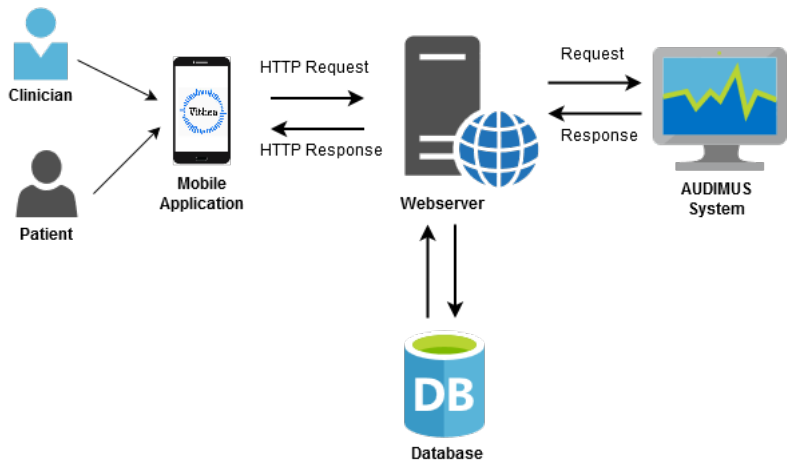

Figure 3.2: Overview of VITHEA mobile application architecture

The **Server** is based on Apache Tomcat, an open source software implementation of a Java HTTP web server. It will be discussed in detail in Section 4.2.

The **Database** used is MariaDB. This is a community-developed, commercially supported fork of the MySQL Relational Database Management System (RDBMS). It is a free and open-source software under the GNU General Public License. This was the database used in the previous version of VITHEA mobile, it will be discussed further in Section 4.2.4.

AUDIMUS is an in house ASR system developed by the Spoken Language Processing Lab of INESC-ID (L^2F), which supports automatic evaluation by recognizing speech. This system will process what the patient says in order to decide whether it is right or wrong.

The **Virtual therapist** representation is based on the Unity2 game engine and it has speech synthesis capabilities. It contains a low poly (a relatively small number of polygons) 3D cartoon model with visemes and facial emotions that receives and transmits text to the Text-To-Speech (TTS) server. When the response from the server arrives the character's lips are synchronized with synthesized speech.

An example will be given to further explain how the system works (only the patient module will be considered). When the patient starts an exercise, the virtual therapist will send a textual request to the TTS server. The response will contain the synthesized speech, which will be synchronized with the lips of the virtual therapist, in order to move according to what the virtual therapist is saying.

If the patient selected a visual exercise, then an image or a video will be loaded. In this example an image is going to be considered. A request from the application will be sent to the server to request the image, when the server replies the image will be loaded into the android device, this step is done asynchronously, so that the application keeps responding. Then the patient would use the device's microphone to record the response of the exercise, this message will be sent to the server and then to the AUDIMUS system in order to verify if what was said is correct or not. Then a reply from AUDIMUS would go through the server and then be redirected to the user's phone.

3.3 Client Application

The Android mobile application developed has two modules: a patient's module to help aphasic patients to recover their speech with VITHEA therapy exercises, and a clinician's module to assist the patients and to keep track of their progresses enabling the clinician to guide their patients properly. Each module has multiple activities, an activity is a crucial component of an Android application since it provides the window in which the application draws its UI. The Android system initiates code in an Activity, making it the entry point for the interaction with the user. Both modules will have the same authentication system, which will be explained in the next sub section.

3.3.1 Authentication

In this sub section, the authentication system and mechanism used in VITHEA 2.0 will be discussed. Both the modules (patient and clinician) from the client application need to be authenticated before accessing any VITHEA web services.

3.3.1.A Authentication system

A user email and password combination is used for the authentication mechanism. However, sending the users password in every request to the server, makes it vulnerable to possible malicious attacks or eavesdropping in non-secure connections. Therefore, to improve security and the UX without compromising the users' credentials, an authentication process was developed.

All the Android devices have an Account List, as illustrated in Figure 3.3. This list contains all the online accounts that are saved in the device.

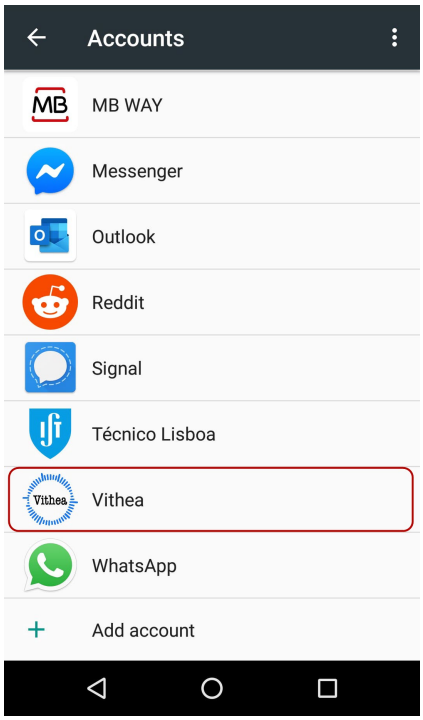

Figure 3.3: Andriod Account Manager: Accounts

To access the *Accounts*, the user can go to Settings → Accounts. VITHEA 2.0 uses this list to save the account, this is achieved by using the Account Manager APIs. The Account Manager is the built-in class for account management, it enables the application to remember and recognize a user. With this feature the username and password would only be asked once, removing the need for the user to input the email and password every time the application is opened. These APIs provide an easy way to fetch the user's credentials. For security reasons, the password is never saved in the phone, only a token with which the user's can access the services, so even if the phone is compromised the attacker would not know the users password.

The above Figure 3.3 depicts different accounts types like Google, Tecnico Lisboa and VITHEA, etc. The user can easily add an account by either 'add account', (which is an indirect way of creating account), or directly using the application.

3.3.1.B Authentication Mechanism

This section will discuss how the application will authenticate and store the user's credentials using the Android Account Manager to simplify the process. Figure 3.4 shows how an authentication process works with Android Account Manager for this particular project. All the requests and responses are sent using HTTPS to ensure security.

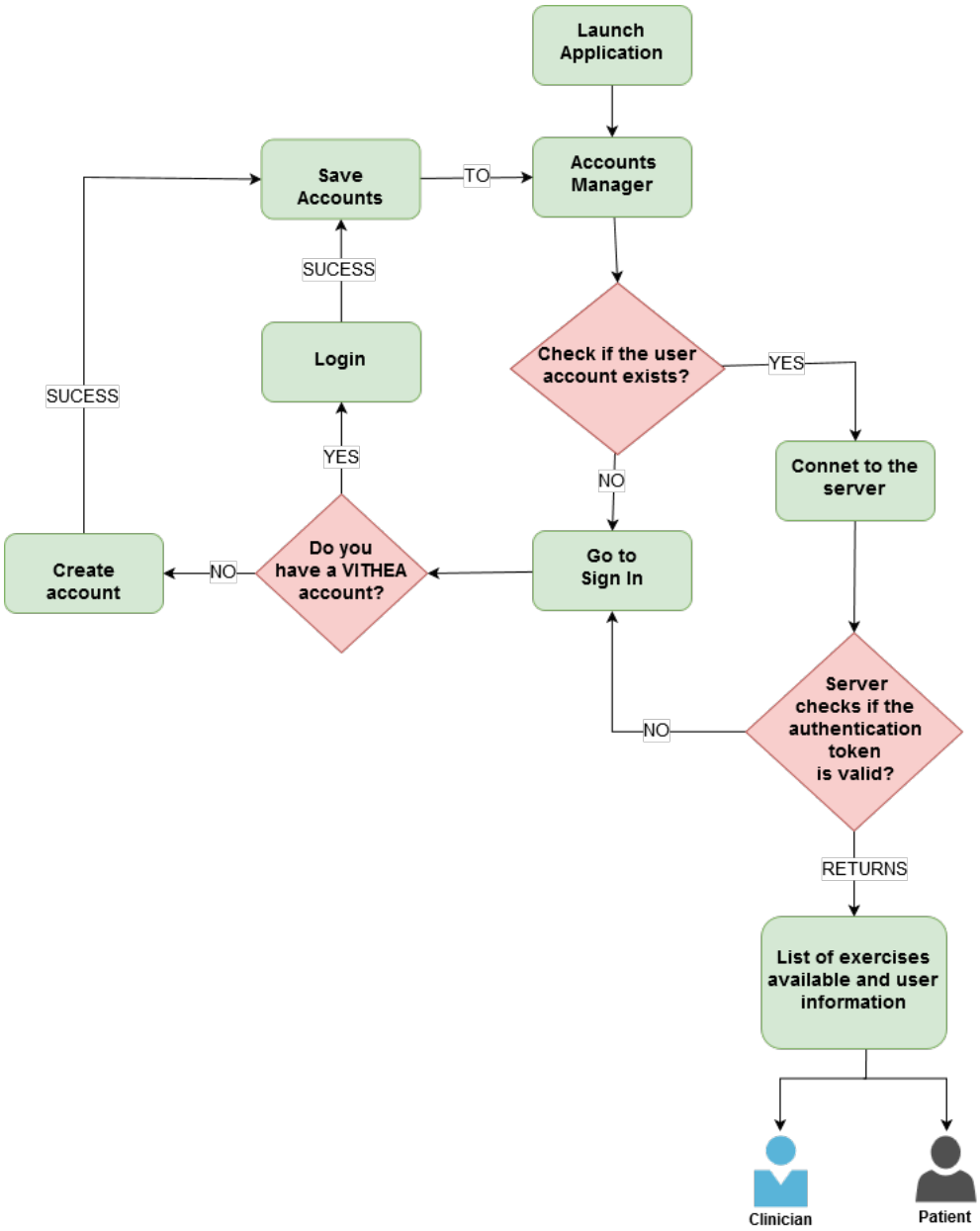

Figure 3.4: Authentication Mechanism

To successfully access any service from VITHEA Web Services (VITHEAWS), a user must be authenticated. When a user (patient or clinician) launches the application, if an account already in the *accounts list*, the application will send the token to the web server, this request is usually fast. While the application is waiting for the authentication process to finish, an activity will be shown to the user with just the logo. This is done so that the user knows the application is running, otherwise only a black screen would appear, the activity is shown in Figure 3.5. It only takes few seconds to verify and load the next activity.

Figure 3.5: Get Authentication Activity

If the user does not have an account in *accounts*, or if the user is trying to connect for the first time, the Sign-In activity will be presented, this activity can be seen in Figure 3.6.

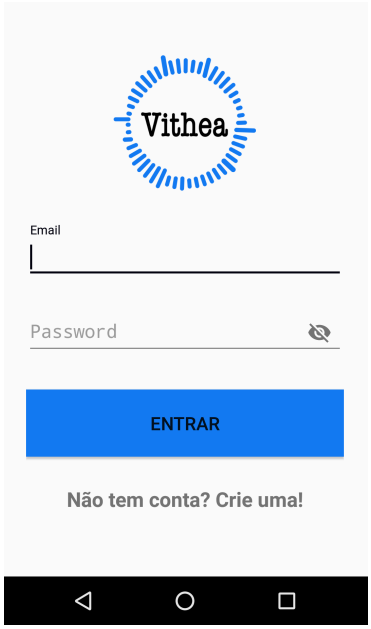

Figure 3.6: Sing-In Activity

If a user has already created an account with VITHEA but it is not saved in *accounts*, then the user needs to login using his/her credentials (email address and password) in the Sign-In activity as shown in Figure 3.6 above.

If the user does not have an account created with VITHEA, in the Sign-In activity there is a link to the Register activity, which allows to create an account as a patient or clinician, shown in Figure 3.7. When the account is successfully created, it will automatically save the email address and authentication token in *accounts*.

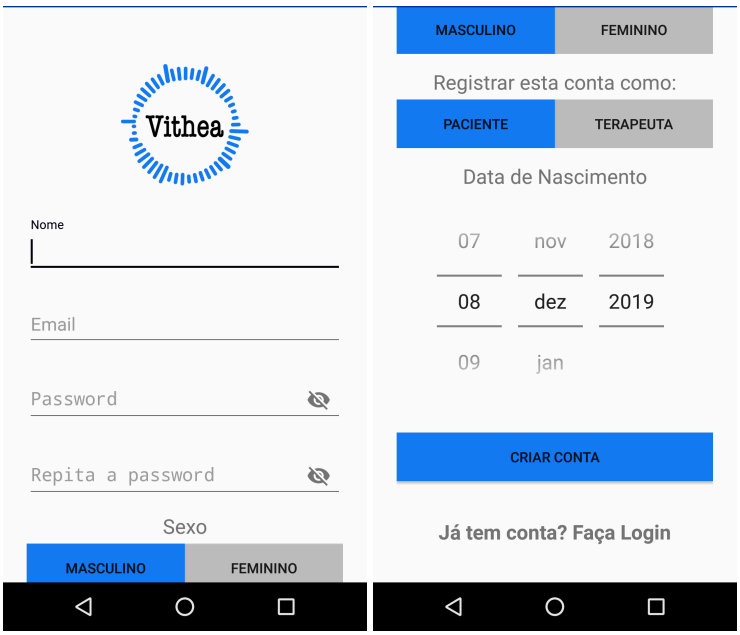

Figure 3.7: Create account Activity

If an email address and a valid token (representing the email and password) exists in the *accounts* already, for both patient and clinician, a request will be sent to VITHEAWS and the user will be granted access to the application, after the server validates its authenticity.

The server will return a list of exercises available as well as user information according to the type of user account (patient or clinician), and will redirect the user to the main screen. According to user type, the activities are bundled as patient and clinician module. Figure 3.8 shows how the client application and the server communicate in order to authenticate a user. Following sections will dwell deeper into both the modules, and its activities.

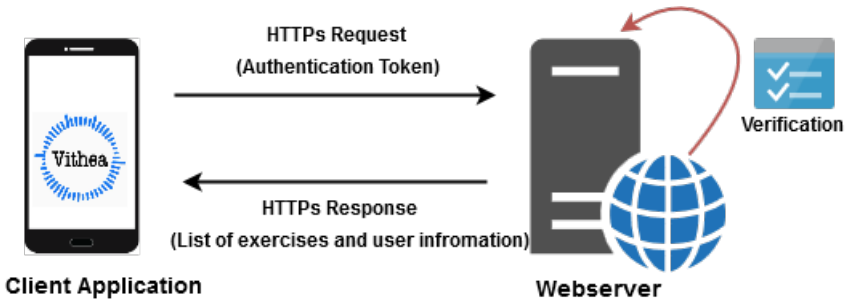

Figure 3.8: A simple overview of VITHEA authentication system

3.3.2 Patient Module

In this section, the patients module is discussed thoroughly. The following home activity, as depicted in Figure 3.9, is presented after the authentication process.

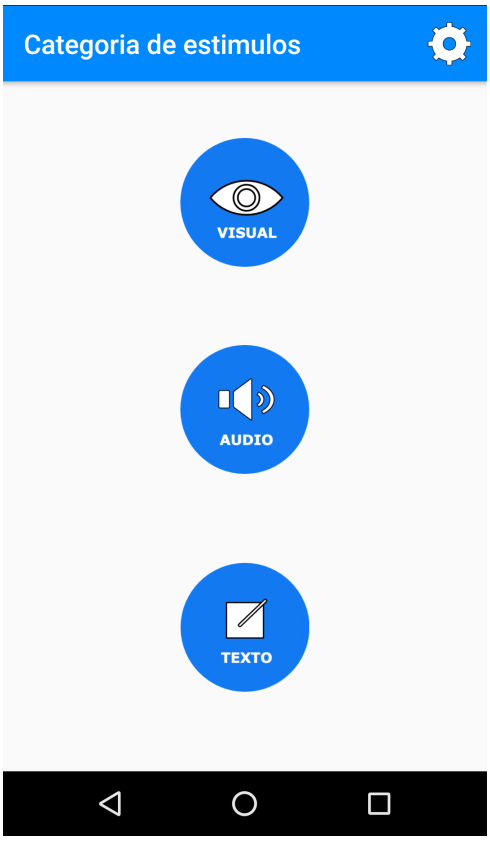

Figure 3.9: Home Activity

The patient's module contains new functionalities, each one has a different activity; this is done in order to improve UX. The users can also personalise the application to their preferences. In this way, each user can have the best experience while doing their speech recovery exercises. Gamification, meaning the use of game design elements to improve user engagement, was considered for this project, but discarded due to the fact that this application targets mainly elderly patients. The goal of this application is to be considered as a tool in the speech recovery process of the patient.

Patient's module comprises of different types of exercises such as visual (image + video), audio and text, which can be chosen depending on the patient's needs or preferences. After selecting the type of exercise, a list of stimuli (different categories of exercises to choose from) is presented to the patient as illustrated in Figure 3.20, Figure 3.21 and Figure 3.22. Selecting any stimuli would initiate and elicit the exercises accordingly, which is presented to them by the chosen virtual therapist as shown in Figure 3.10 (in this case, it is from visual category).

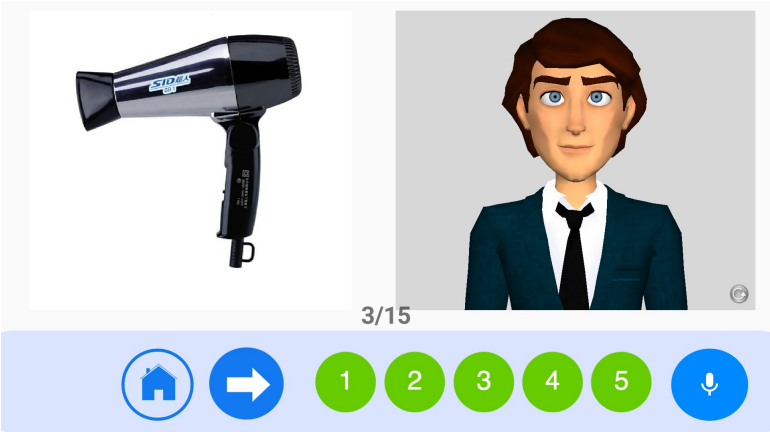

Figure 3.10: Example of an exercise from the visual category

When the patient presses the microphone button a response will be recorded using the device microphone, when the button is released, the application will create a file containing the audio, this will be sent to the server, in order to verify if the answer matches one of the multiple correct answers defined by the clinician. While recording the response, the patient can also slide left to cancel it if they do not want to deliver the response as depicted in Figure 3.11.

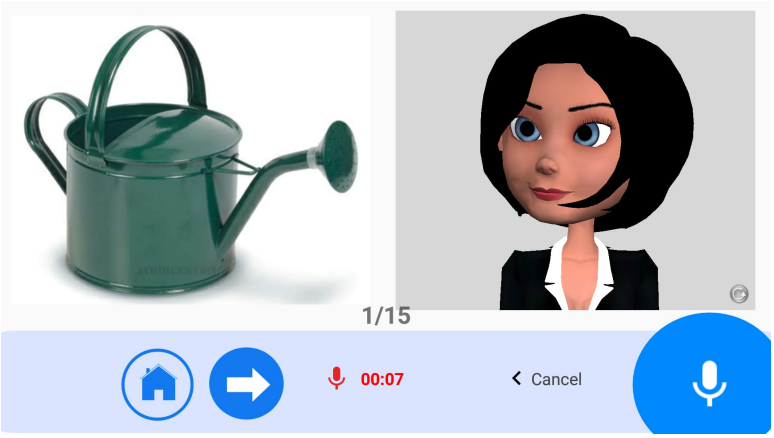

Figure 3.11: An illustration of recording and cancelling voice response

For any category and exercise, the UI for the patient module would display the stimulus on the left and a chosen virtual therapist on the right as shown in Figure 3.10 above. There are three virtual therapist a user can choose from, namely Catarina, Filipe and Edgar.

Before dwelling deeper into different activities in the main menu, lets discuss the different buttons and functionalities developed to enhance the UX that are common to all the categories to avoid redundancy.

1. Home : Home button allows users to return to the main menu. It can be used if they wish to change the category or stimulus of the exercise, or just want to change any of the settings. Upon

selection, it will save all the exercises data and present a summary of the statistics as shown in Figure 3.12.

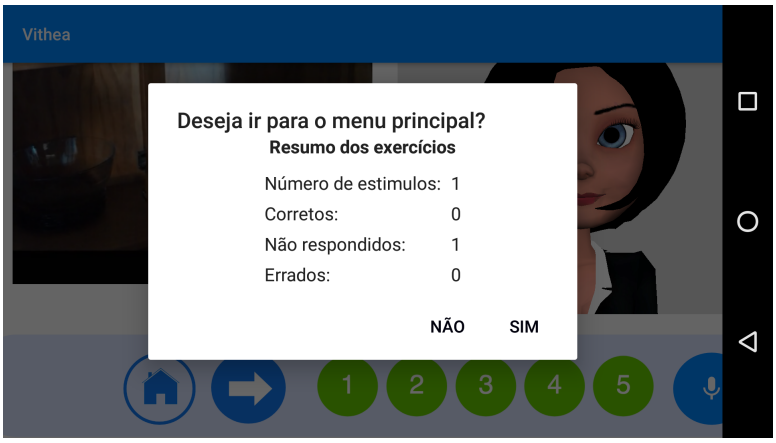

Figure 3.12: Summary Statistics when the home button is pressed

2. Forward : This button allows user to go to the next exercise.

3. Repeat/Refresh : This button appears only in the end of an exercise, it allows the user to repeat that particular an exercise again, as illustrated in Figure 3.13 below.

4. Recording : The user can tap and hold the microphone button to record the responses, or slide left to cancel the current recording process.

5. Summary Screen: When an exercise is completed, a summary is presented, as shown in Figure 3.13 below. It reports some statistics over the total number of stimuli contained in the exercise and the number of correct, incorrect, and not attempted responses.

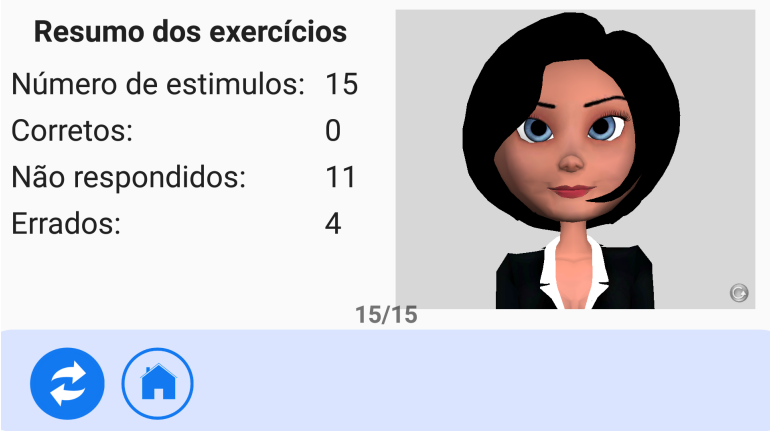

Figure 3.13: Summary statistics, presented when an exercise is completed

6. Number of Attempts 1 : This is an optional setting, patients can either turned it off or customize the number of attempts (1 to 5), by accessing the settings menu, depending on how they want to challenge themselves. When the number of attempts is deactivated, it means the patients have an unlimited number of attempts to finish the exercise.

7. Visual Feedback: After the patient finished his/her response, it will be sent to the server and validated if the answer is correct or incorrect. If the answer is correct, it is acknowledge with a green background screen, as shown in Figure 3.14, otherwise a red background screen will appear, as shown in Figure 3.15.

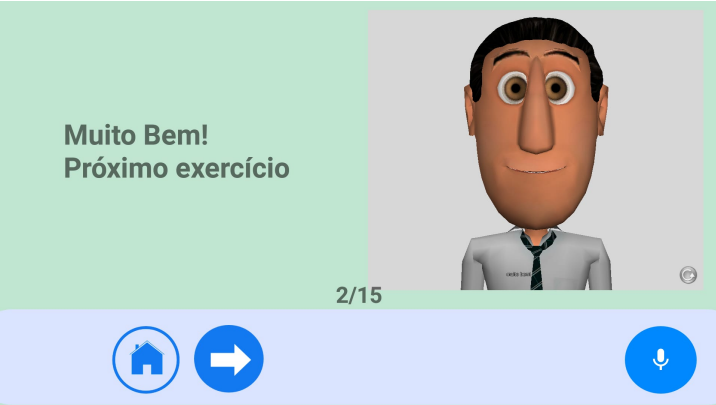

Figure 3.14: Feedback background when the response is correct

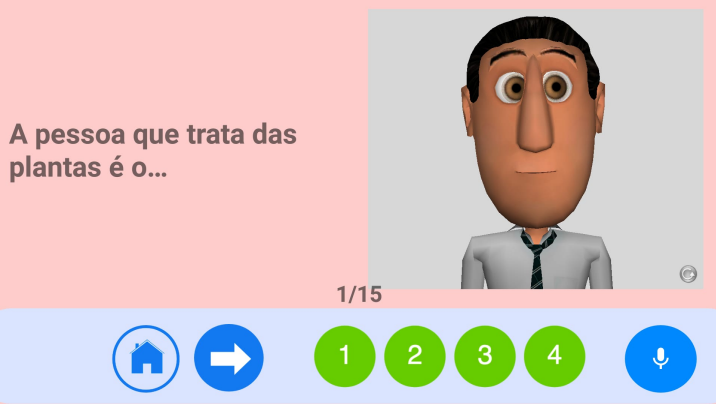

Figure 3.15: Feedback background when the response is incorrect

3.3.2.A Settings

From the home activity, shown in Figure 3.9, a patient can access the settings activity to configure, manage and customise the application to his/her preference. Settings activity provides a useful set of options to improve user retention by helping them personalise the application and record their progresses. The

full settings menu is illustrated in Figure 3.16.

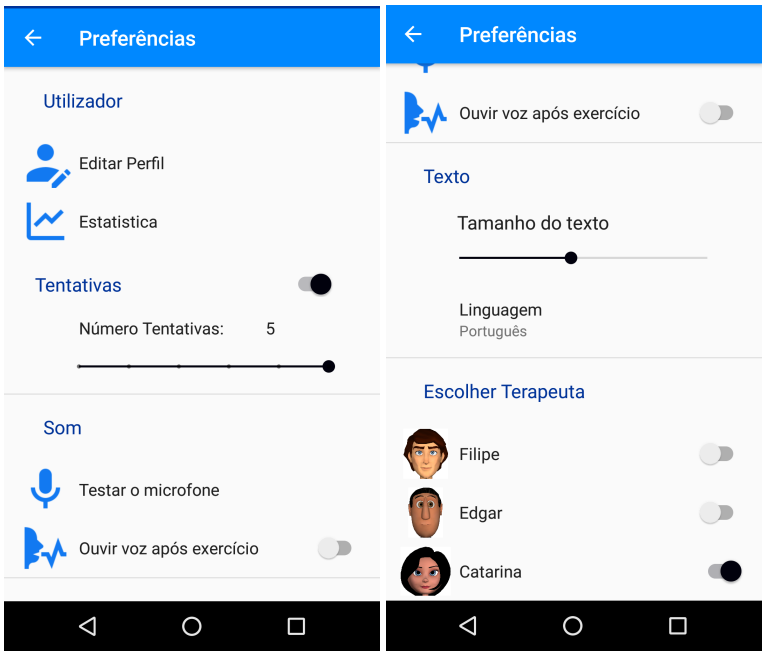

Figure 3.16: Settings Activity

The functionalities of each menu are categorised into the following menus:

1. **User**: In this sub menu, the patient can manage their existing profile by accessing the Edit profile activity as shown in Figure 3.17.

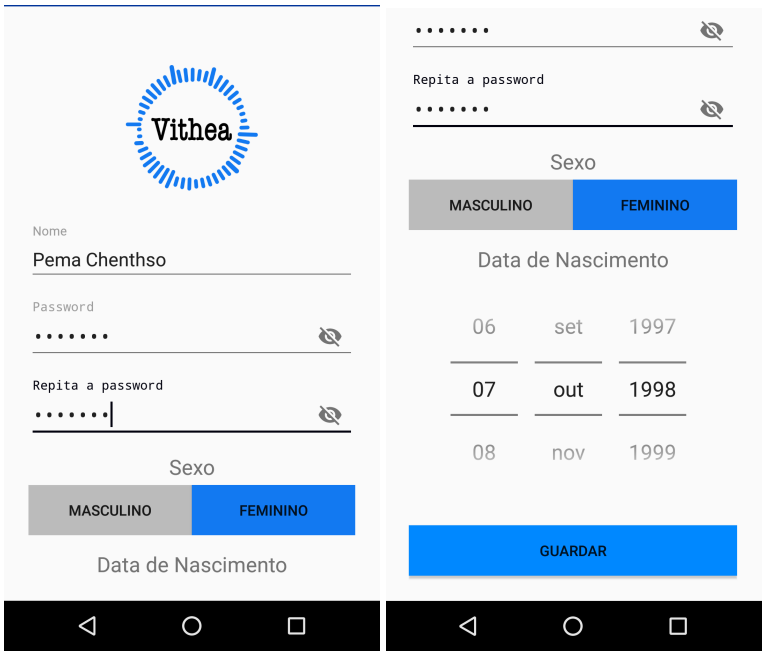

Figure 3.17: Edit Profile Activity

This activity allows users to change their password, full name, date of birth or sex. Before saving any of the changed profile settings, the user will be prompted with a message asking whether he really want to do it or not. In case of changing the password, the user will get another authentication token, which will be stored into the Account Manager replacing the old authentication token. This is done to maintain the consistency with the user authentication process, removing the need to input the email and password.

In this sub menu, patients can also view their statistics, illustrating their performance as shown in Figure 3.18 below. If the patient wants to start from scratch it is possible to erase all the statistics, this will only affect the user, the clinician will still be able to see all the statistics from his monitored patients.

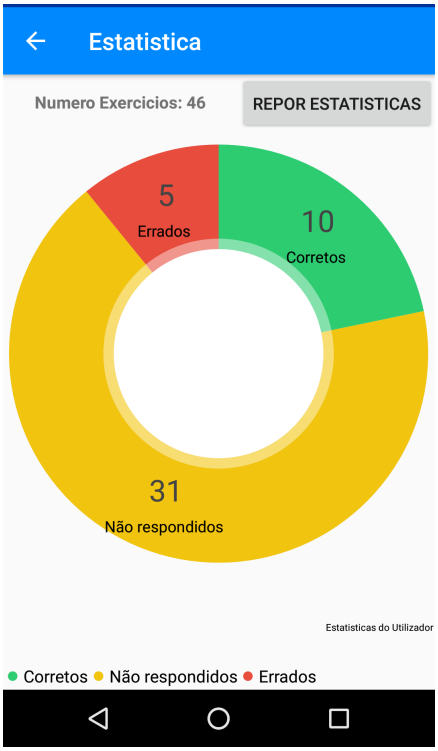

Figure 3.18: Statistics

Next in the same user sub menu, the patient can also change the number of attempts. The patient can either deactivate the number of attempts, or set a number of attempts from one to five according to their preference, by sliding on the seek-bar as depicted in Figure 3.16 (left).

2. **Sound**: This sub menu bundles features such as microphone testing, where users can test if their microphone is working or not by recording audio, which will be played back to ensure that the microphone is working correctly shown in Figure 3.19. The other option in this sub menu as shown in Figure 3.16 (left) is used to activate or deactivate the play back of the voice which was recorded,

while doing the exercises. In this way users can analyse their response and verify the audio that was recorded.

Figure 3.19: Microphone Testing Activity

3. **Text**: In this sub menu, users can change the text size as per their liking by sliding on the seek-bar as depicted in Figure 3.16 (right). This was added due to the target audience of the application being elderly patients, with age the eyesight gets worse so this option helps mitigate this problem, by proving a huge font as needed.

   Under Text sub menu, the user can choose their preferred language, English or Portuguese, this feature was added in case VITHEA ever expands to new languages. The patient module is ready to handle different languages, but for now it is just a prototype, because the database, the ASR, and the virtual therapist would also have to be changed to handle different languages.

4. **Choose Virtual Therapist**: This application is updated to accommodate three virtual therapists. Users can personalise their application by changing the virtual therapist to their favourite one (Figure 3.16 right).

3.3.2.B Categories of Exercises

Different categories of exercises, as shown in Figure 3.9, are discussed in this section. There are three categories of exercises a user can practice from - namely visual (video and image), audio and text. Each category comprises of different stimuli which are presented with a corresponding example image of the exercise in Figure 3.20, Figure 3.21 and Figure 3.22 respectively.

Figure 3.20: Visual exercises (left) and an example of the stimuli contained (right)

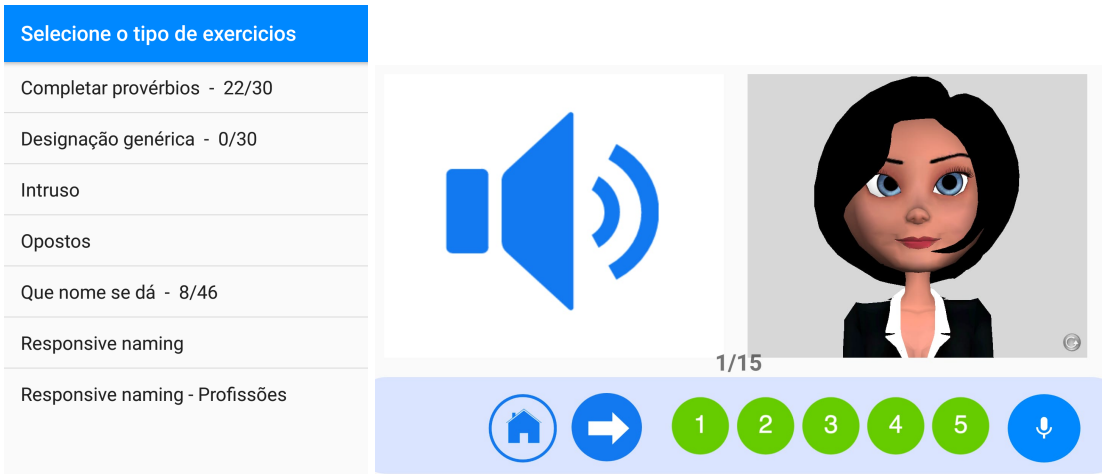

Figure 3.21: Audio exercises (left) and an example of the stimuli contained (right)

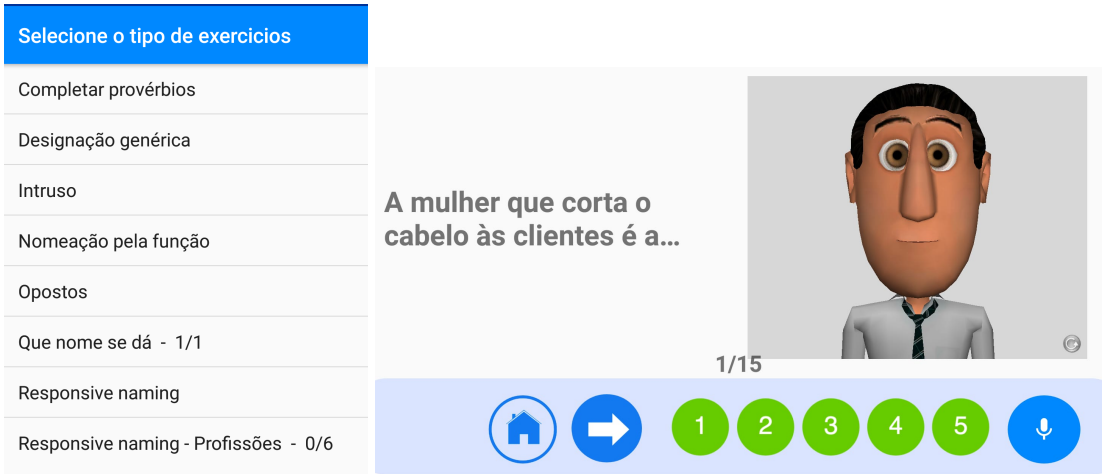

Figure 3.22: Text exercises (left) and an example of the stimuli contained (right)

3.3.3 Clinician Module

In this section, the clinician module will be discussed thoroughly. As in the patient module, if a clinician wants to access the application he will have to authenticate through the process explained in Section 3.3.1. It will store the credentials using Android Account Manager to avoid inputting credentials every single launch of the application, and it also improves security and the UX without compromising the users' credentials. After a successful authentication, the clinician will be presented with the following home activity as depicted in Figure 3.23 below.

Figure 3.23: Home Menu for the Clinician

From the home activity, there are 4 activities that the clinician can select from, as discussed in the following sub-sections to help and encourage his aphasic patients as well as edit his profile.

3.3.3.A Send Exercises To Patients

The clinician can view and analyze if his patient's outcome from the remote speech therapy is as expected or not in a given area, and accordingly send a specific number and category of exercises to help his patients practice more in the area where they need extra effort.

To send exercises to patients, the clinician needs to select patients from following activity depicted on Figure 3.24. Only the patients that accepted being monitored by the clinician will appear on this

selection from the activity shown Figure 3.33 (left). Adding patients to clinician module is discussed in next list item Section 3.3.3.C.

The clinician will first select associated patients that he would like to send the same category of exercises. In order to ease the selection process, a check box was added on top of the list, which has three states - the first state will select all the patients, the second state will deselect all patients that were selected, and the third state will only be reached if the number of selected patients is between 1 and n-1, with n being the number of elements in the list. Pressing the check box again will lead it to the first state (all the users will be selected). It is mandatory to select at least one patient to go to the next activity, as seen in the Figure 3.24 (center) the button will be disabled if none are selected.

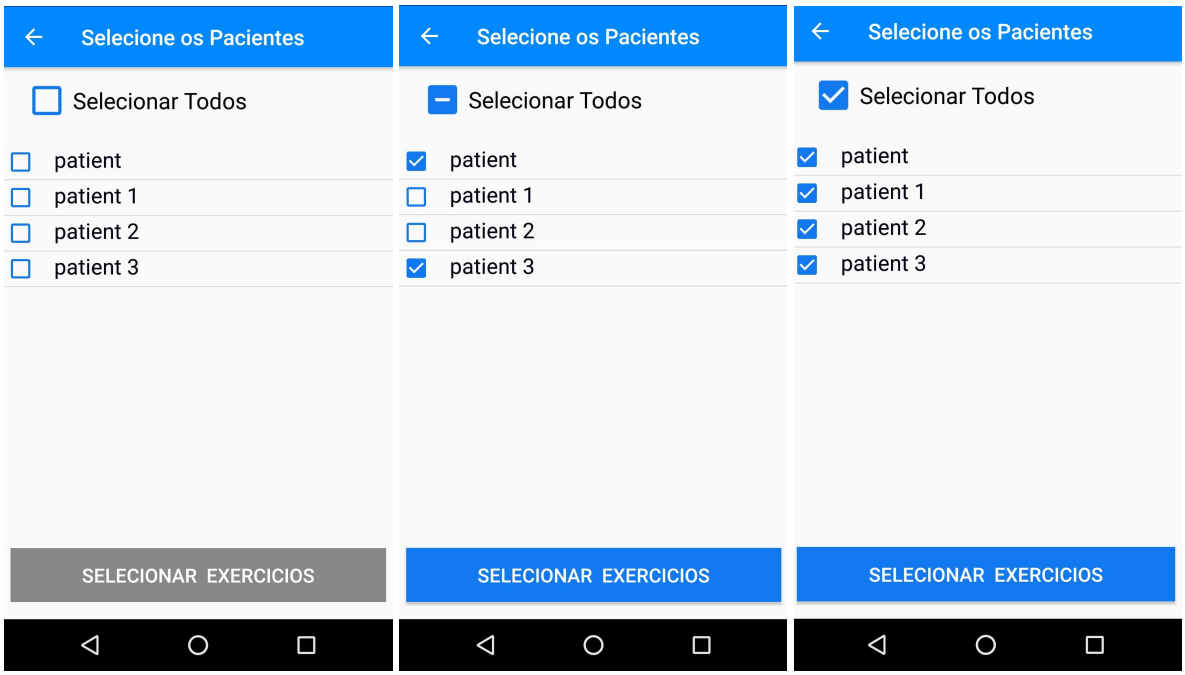

Figure 3.24: Three states of the selection process to send the exercises to patients: first state (left), second (center) and third state (right)

After selecting the patients, the following activity will be presented as shown in Figure 3.25. In here, the clinician will be able to select which exercises he wants his patients to focus on, to enhance their speech therapy depending on the patient's performances.

As it can be seen in Figure 3.25 below, the exercises are divided into categories and sub categories as discussed in Section 3.3.2.B. By pressing a category a list of subcategories will be shown, in this example the Visual category was pressed, therefore all the subcategories associated with it are shown as well. In each subcategory, the clinician can choose how many exercises from one or many subcategories he wants to suggest and send it to his patient(s). If the clinician does not want to specify an exercise he can simply write a number in the label near "Qualquer Exercicio" (Any Exercise).

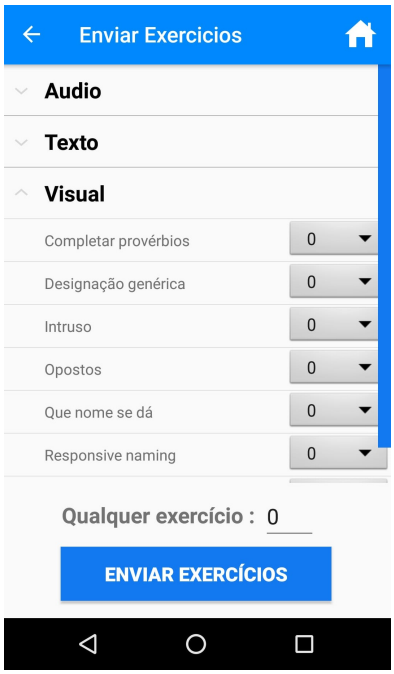

Figure 3.25: Activity where the clinician can choose the number and type of exercises to send to the selected patients

If the clinician chooses "Qualquer Exercicio" (Any Exercise), the selection of both the category and the number of stimuli will be random as explained in Figure 3.26. For example if the clinician decides to send 50 random stimuli, the algorithm will pick a random sub category and assign a random number of stimuli between 1 and 50, it will end when there are no more stimuli to distribute.

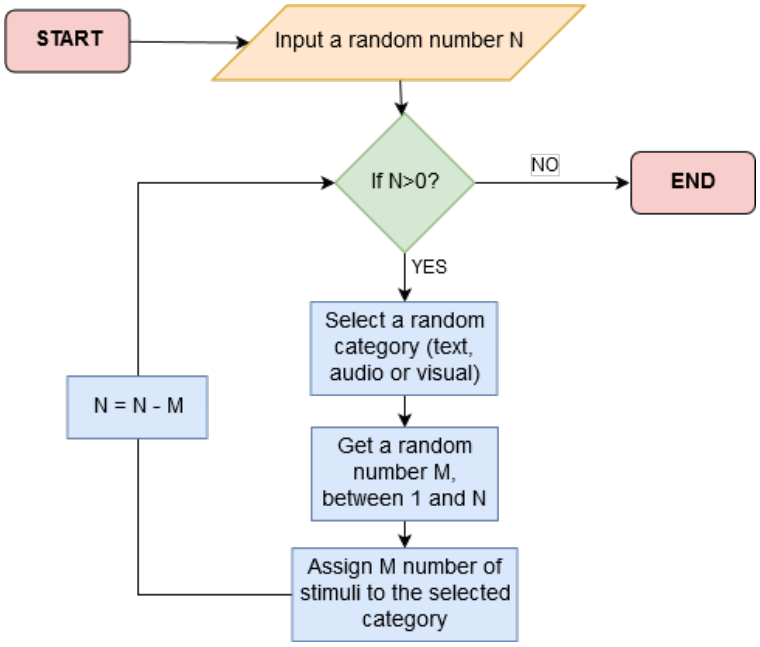

Figure 3.26: Random Selection Algorithm

After the types and number of exercises are selected, the clinician can send exercises suggestion by hitting ”Enviar Exercicios” button, i.e. if the server was able to execute and store all the values or it failed. Figure 3.27 below illustrates the server response stating that the exercises were sent successfully.

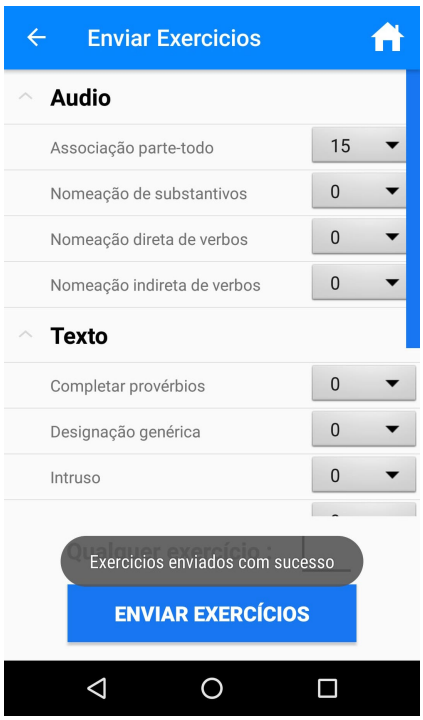

Figure 3.27: Prompt showing the exercises were sent to patients successfully

By clicking on the home button (on the right corner), the application will return the clinician to main menu.

3.3.3.B See Patients Statistics

To send the exercises to patients as discussed in Section 3.3.3.A, the clinician needs to analyse patient’s therapy performances. The clinician can access summary statistics of patient’s therapy from ”See Patients Statistics” activity for analysis. He can check the number of exercises completed, and how good or bad the performance is by checking how many exercises were right, wrong or had no responses. With the help of this summary statistics as depicted in Figure 3.29, the clinician can send exercises to the patient accordingly.

When ”See Patients Statistics” activity is initiated, the following activity will be shown as depicted in Figure 3.28. The phenomenon and three states of selecting the patients is the same as discussed in Section 3.3.3.A. Only the patients that accepted being monitored by the clinician will appear on this selection.

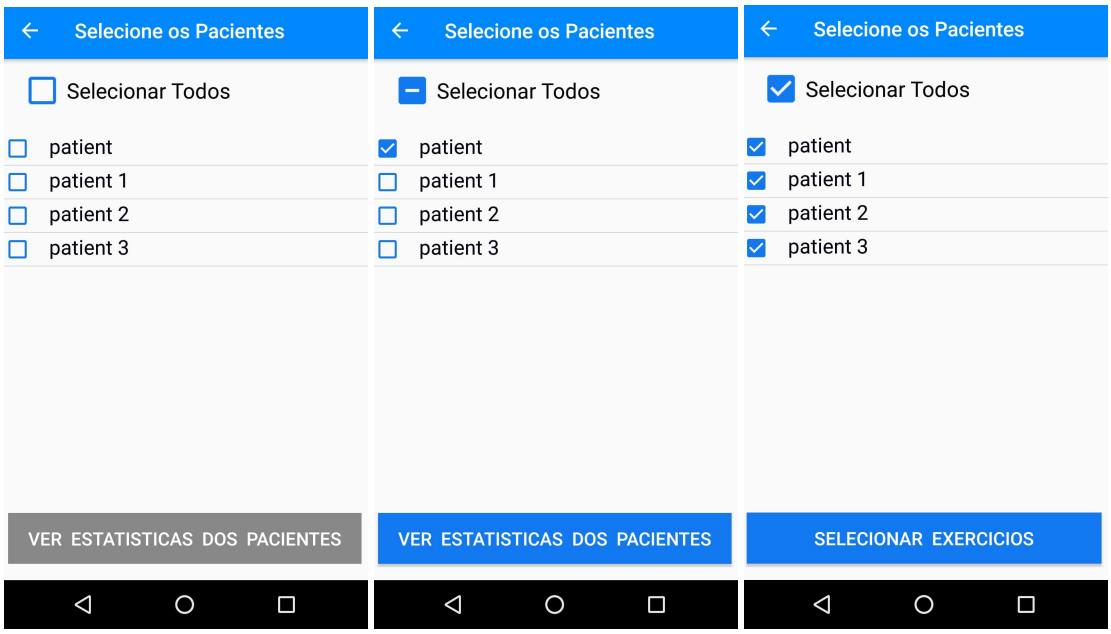

Figure 3.28: Three states of the selection process to see the patient's Statistics: first state (left), second (center) and third state (right)

After selecting the patients from Figure 3.28 activity, following activity will be presented as shown in Figure 3.29 below. This is the summary statistics of a patient's performance.

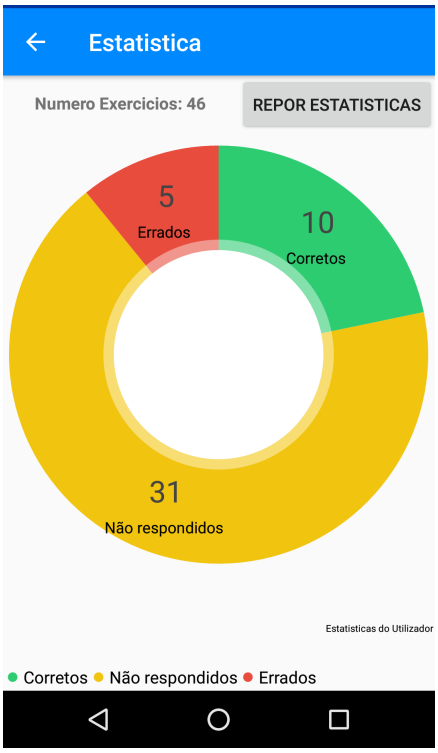

Figure 3.29: Statistics Summary of a patient on a Clinician module

The clinician will be able to see how many exercises the patients have been doing, what is the ratio between right and wrong answers, and how many were unanswered to have an idea about how your patients are performing.

3.3.3.C Manage Patients

If the option ”Patient Management” from the home activity shown in Figure 3.23, is selected, the following activity will be presented to the clinician as shown in Figure 3.30 below. Only the patients who have accepted to be monitored from Figure 3.33 (left) activity, would be shown in the following activity.

Figure 3.30: Patient Management: To remove patients (left) and to add patients (right)

From the above activity Figure 3.30 (left), a clinician can either add or remove his patients. If the clinician ever wants to remove any of his patients, there is a button next to each of the patient’s name that will allow him to do so.

In order to add new patients, the clinician must know the email address that the patient is registered with. Clicking on ”Adicionar Paciente” will take the clinician to the next activity as shown in Figure 3.30 (right), where the patient’s email address is entered. A request will be sent to the patient, mentioning that the clinician wants to add him for monitoring. The patients needs to accept the request in order to provide the clinician with the privilege to monitor patient’s activities as discussed in Section 3.3.4.

3.3.3.D Edit Profile

Clinicians can access ”Edit Profile” activity from the home activity shown in Figure 3.23, to manage their existing profile. Edit Profile activity for the clinician module, illustrated in Figure 3.31 is similar to the Edit Profile of patient’s module as shown in Figure 3.17.

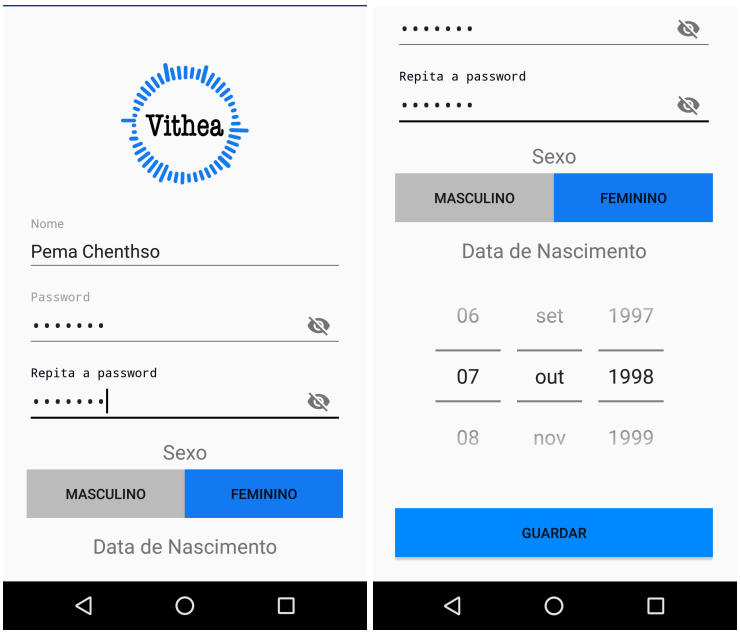

Figure 3.31: Edit Profile Therapist

Edit profile activity allows the clinician to change his password, full name, date of birth or sex. If any changes were made, the clinician will be prompted with with a message asking weather he really wants to do it or not as shown in following fig. 3.32. As explained in Section 3.3.1, if the user changes the password, a new token will be saved in the Account Manager.

Figure 3.32: Prompt asking if the clinician wants to save the changes made

3.3.4 Patient Clinician Association

As discussed in the subsections above, when a clinician adds a patient to monitor, that particular patient must consent. Therefore, a prompt will appear in the Home Menu as shown in Figure 3.33 (left) in the patient's module. The prompt also has a checkbox, in case the patient is sure he does not want to be asked again. This means, when he presses "No", without pressing the checkbox this question will be asked again. This might be useful if the patient is not sure who the clinician is the first time the prompt appears.

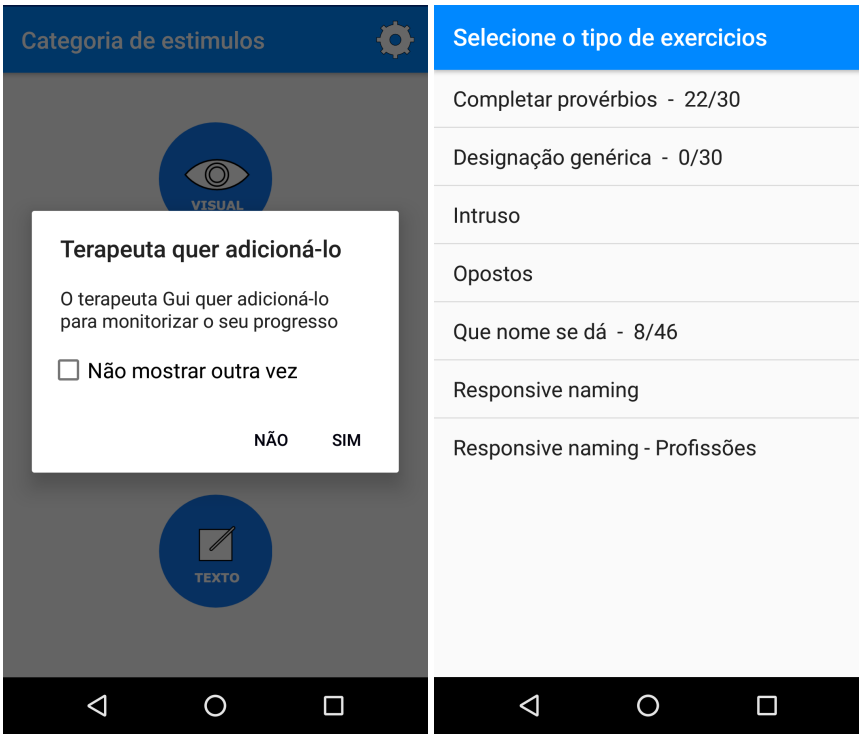

Figure 3.33: Prompt asking patient to accept his clinician (left) and Activity showing exercises sent by the clinician in the patient's module (right)

Once the patient accepts to be added, his clinician can track and monitor the speech therapy progresses. If a clinician thinks that his patient might need an extra effort in a certain category, he can send exercises fine tuned for the patient. Figure 3.33 (right) shows how the exercises received from clinician are presented to the patient. It shows the number of exercises he should practice in different subcategories that his clinician chose for him. Every time he concludes an exercise, these will be added and displayed as "number of exercises completed / total number of exercises to perform". For example in Figure 3.33 (right), the patient has completed 6 out of 46 exercises that were sent by his clinician in "Que nome se da" subcategory. Once he achieves the goal, the number of exercises next to the name will disappear.

4

VITHEA Web Services

Contents

In this chapter, the server application VITHEAWS will be discussed thoroughly. First, we explain why a RESTful architecture and JSON as data representation were chosen, then the architecture of the server is described. Finally we report the re-engineering performed.

4.1 Service Oriented Architectures And Data Representation

This section will discuss why the chosen Service-Oriented Architecture (SOA) was REST and why JSON is the preferred data representation.

The three most commonly used web services standards, for server applications, that implement SOAs are: Simple Object Access Protocol (SOAP), Remote Procedure Call (RPC), and REST. SOA is an architectural style in which services are provided to other components of the application. A service is a logical representation of a repeatable business activity with a given outcome. Web Services are platform independent exposed APIs which can be used from remote servers.

REST is an architectural style, its architecture is based on resources and interfaces. Any information can be a resource and they share a uniform interface to transfer the state between the client and server. In comparison with SOAP, REST is lightweight, this is due to the fact that SOAP uses XML and therefore has some overhead. The results from REST is easier to be read by humans, and it is also flexible and easier to build. In terms of security, since the requests go through HTTP or HTTPS, the firewall can verify each message by analysing the HTTP command used. The fundamental problem with RPC is coupling. RPC clients become interconnected to the service implementation in several ways and it becomes hard to change service implementation without having to do the same for the clients. Clients become coupled with the service because they are required to know the procedure names, as well as, the parameters order, and types. In case the procedure signature needs to be changed, for example, a change in the number of arguments, the order or the type, the clients procedure implementation would also have to be changed accordingly. Furthermore, considering that there are already some web services deployed, written with a RESTful architecture, REST was chosen as the architectural style.

Regarding **data representation** with REST, two standards were evaluated, JSON and XML. Both the data representations can describe and facilitate the transport and consumption of hierarchical data structures. However, JSON has a simpler syntax than XML, because XML is a markup language and it requires more data for the same amount of information, therefore creating an unnecessary overhead. Another advantage of JSON is that, it is significantly faster than XML [19]. Taking into account that the previous mobile application from VITHEA as explained in Section 2.1 also uses JSON, it might help as some code could be reused. Therefore, JSON will be the preferred data representation for all the aforementioned reasons.

4.2 Server Architecture

The implemented architecture remained the same as the previous VITHEAWS as shown in Figure 4.1, and it will be explained in detail in this section.

When the previous version of VITHEAWS was built, the main components (Apache Tomcat, Spring, Hibernate, and MariaDB) were standard in the industry, providing the best possible approach, while being open source and flexible. These main components are still one of the standard for web servers, hence the architecture remained the same for this project, only the version of the components were updated. In this section, each component will be analysed and discussed.

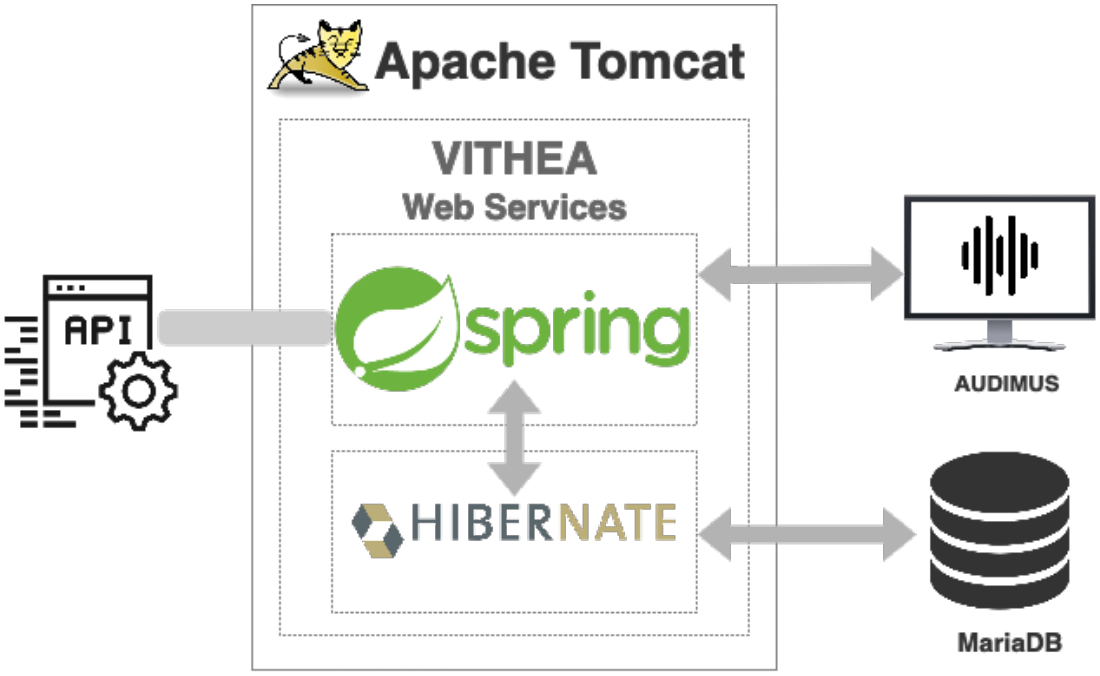

Figure 4.1: Server Architecture

Apache Tomcat is used to host Java web applications, in this case VITHEAWS. Maven helps to deploy the web server to Tomcat. Maven is a project management tool that builds the server application, and afterwards produces a *.war* file, which is then deployed to Tomcat. VITHEAWS uses **Spring** to develop REST APIs, to add security, and is also responsible for the communication with **AUDIMUS**. Spring can also integrate with **Hibernate**. Hibernate is an ORM used to map an object-oriented domain model to a relational database, it is responsible for the communication with the database, in this case MariaDB, this is where all the data is stored. All the main components will be discussed in detail in this section.

4.2.1 Apache Tomcat

Apache Tomcat, is an open source web server and servlet container developed by the Apache Software Foundation. Apache Tomcat provides tools for configuration and management, it can also be configured with XML configuration files [20]. The web server is responsible for listening to all incoming requests, loading the respective servlet to handle incoming client requests, process them and reply back with a response. The main reasons to use Apache Tomcat are :

- **Lightweight**: Tomcat is a lightweight application, providing relatively quick load and redeploy times.
- **Open Source**: Tomcat is free and has all the advantages of being open source.
- **Flexible**: It has extensive and customization options that can be tweaked to fit the unique needs of the application.
- **Stability**: Tomcat is a mature and stable platform, having a stable server is essential in order to provide a good service.

In this project Tomcat runs as a service in the operating system, meaning that even if it crashes for any reason, the operating system would try to restart the service, thus allowing for a greater availability.

4.2.2 Spring

Spring is a powerful, lightweight, open source framework and IoC container used for application development. IoC is a programming principle used for inverting different controls (any class responsibility) to achieve loose coupling, allowing classes to be more testable, maintainable and extensible, which is done with dependency injection [21]. The main advantages of using Spring are [22]:

- **Maturity**: Spring has been around for many years, and it has become mature enough to become a default solution for most common problems.
- **Active community**: Spring is an open source project. Having an active community means faster bug reporting and fixing, plus there are answers to the most common questions, which makes development faster.
- **Usability**: Spring has multiple configuration options which makes it easy for developers to start and configure what is necessary. It also has good documentation and tutorials.
- **Modularity**: Spring is highly modular, the developer only need to import the modules necessary not the whole framework. Due to this, it is easy to integrate with other frameworks like Hibernate.

- **Testing**: Spring supports Test Driven Development (TDD), the application is written with Plain old Java object (POJO)s and this allows unit testing. It also provides Mock Objects for MVC.

Spring Security and Spring Web MVC, have been used to implement the server side services. Spring Security is a non-intrusive framework providing a set of authentication and access-control services, HTTP digest authentication, and HTTP basic authentication. In this project, HTTP authentication is used to verify if the user is whom he claims to be and all the services are protected with this authentication. Spring MVC is a framework that allows building flexible and loosely coupled web applications and REST services. MVC stands for Model, View, and Controller respectively, this design pattern separates the business logic, presentation logic and navigation logic.

- **Model** - A model contains the data of the application.
- **Controller** - A controller contains the business logic of an application.
- **View** - A view represents the provided information in a particular format.

Spring MVC can fulfill HTTP requests, delegate data processing responsibilities to other components, and prepare the response that needs to be given. All three fundamental attributes when building REST services.

Listing 4.1 is an example to illustrate how easy it is to create a REST service. The Controller annotation is used so that this class can be *autodetected* through the *classpath* scanning. The *RequestMapping* annotation is used to map HTTP requests to handler methods of MVC and controllers, in this case the URL would be `http://serverURL/services/getInfoAndExercises`. With *RequestMethod.POST*, we can see that this URL only accepts the HTTP POST method. Then it receives a parameter, it must be named username.

Listing 4.1: Example of Rest Service in Spring

```
1 @Controller
2 @RequestMapping("/*")
3 public class LoginService {
4     @RequestMapping(value = "/getInfoAndExercises"
5     , method = RequestMethod.POST, produces = "application/json")
6     public @ResponseBody UserInfo getInfoAndExercises
7     (@RequestParam("username") String username) {
8     }
9 }
```

4.2.3 Hibernate

Hibernate is an ORM for the Java programming language, it is free and distributed under the GNU Public License. An ORM provides a framework for mapping an object-oriented domain model to a relational database, meaning its primary feature is mapping Java classes to database tables, and vice versa [23].

Hibernate's main features are as follows:

- It uses Hibernate Query Language (HQL), similar to Structured Query Language (SQL) but provides full support in polymorphic queries. This means that the query will return all the persistent classes instance which extends the queried class. This also provides database independence, if the developer ever wants to change the database, HQL would remain the same.
- It provides lazy loading of data, meaning data would only be loaded when needed.
- Annotations in order to create an association between tables. In the example below, it can be seen that User has a 1 to 1 relationship to AccountRole.

Listing 4.2: Example of annotations in Hibernate

```
1    public class User implements Serializable
2    {
3        @Id
4        @GeneratedValue(strategy = GenerationType.IDENTITY)
5        private Long userId;
6
7        @OneToOne(fetch = FetchType.EAGER, mappedBy = "user")
8        private AccountRole accountRole;
9    }
```

- Support for SQL queries, this is useful in case the query becomes complex for HQL
- Caching helps increase the performance by reducing the need for application access the disk.

Spring supports integration with Hibernate, for resource management, Data Access Object (DAO) implementations, and transaction strategies. The advantage of using Spring with Hibernate is that it makes the application layering clearer with any data access and transaction technology, and it allows loose coupling of application objects.

4.2.4 Database

MariaDB is a community-developed, commercially supported fork of the MySQL Relational Database Management System (RDBMS). It is a free and open-source software under the GNU General Public License [24].

A relational database is based on the relational model of data, it organizes data into one or more tables (or "relations") of columns and rows. Each table has its own unique key, that is how a row is identified. In order to have backwards compatibility with the already existenting tables in the database the engine selected for the database remained as MyISAM [25], it is light and non-transactional engine with great performance.

MyISAM has the following characteristics:

1. Data is stored with the low byte first, making the data machine and operating system independent. There is no significant speed penalty for storing data low byte first.

2. Supports concurrent inserts, meaning that new rows can be inserted into it at the same time that other threads are reading from the table.

3. All numeric key values are stored with the high byte first to allow better index compression.

4.3 Re-engineering

As discussed previously, the implemented architecture remained mostly the same. In fact, this architecture was developed around 6 years ago, making it a big span of time in which all the technologies used are now outdated due to the rate at which all the technology is progressing today. However, Apache Tomcat, Spring and Hibernate kept getting updates throughout this span of time, making them a relevant and viable solution.

The project dependencies needed to be updated, typically an application uses a dependency manager in order to manage dependencies better, in this project the dependency manager is Maven. Maven is not only a dependency manager, but a project management tool based on Project Object Model (POM). POMs are XML files that contain information related to the project and configuration information. Besides the dependencies, it also controls the project life cycle and the logic for executing plugin goals at defined phases in a life cycle. A useful example on how Maven is a powerful tool is when deploying the application to a Tomcat server, the server requires the project to be in *.war* format, all that needs to be done is run one command *mvn clean tomcat7:redeploy*. Maven will take care of all the steps: download dependencies, compile, etc, in the end it will generate a .war file and deploy it to Tomcat. This tool is

extremely useful when trying to update the libraries, because the developer only has to modify the XML, and maven will fetch it and compile the code with that version of the library.

The following Table 4.1 shows the list of updated libraries for this project. Jackson is mentioned in the table, this is not a main component, but it is an important library for processing JSON in Java. Apache Tomcat was also updated to version 7.0.96, this version increases the stability of Tomcat and adds security and other minor libraries were also updated.

Table 4.1: List of updated libraries

Library	Previous Version	Updated Version
Java	6	8
Spring MVC and ORM	3.2.0	5.1.9
Spring Security	3.1.3	5.1.6
Hibernate	3.3.2	5.3.10
Jackson	1.9.11	2.9.9

In the original VITHEA platform, there were two different web applications, one for the patients (Vithea) and other for the clinicians (VitheaAdmin). The decision to divide the project into two web applications made sense in the older version, because VITHEA mobile application would only be used by the patients. In order to not have all the code duplicated and have to manage two code bases, VITHEAWS would use a library with code from VitheAdmin, this way if the project changed, the developer would only have to create a new library and import it into VITHEAWS.

However, with VITHEA 2.0 the patients and the clinicians both share the same mobile application, while also sharing more business logic, for example, with the clinicians being able to add patients. Having two separate projects would be harder to maintain, and it would lead to duplicated code in both the projects. This approach would have been very time consuming as well, especially while building the new APIs, due to the substantial development in both the clinicians and the patients business logic. Due to all the aforementioned advantages, both project were merged into VITHEAWS.

Updating the above mentioned libraries and merging both projects required considerable refactoring of the code. While updating the libraries there were also deprecated methods, these are methods annotated with @Deprecated. Methods that are deprecated usually mean that the feature will be removed or discontinued in the future, or that there is a better approach, for example the new approach could be more efficient, or that the method had a bug and is now replaced by another improved method. Therefore the developer is advised to change these methods. Some of the best practices with the main components also changed, especially the integration between Spring and Hibernate.

The APIs that were added or modified, are documented in Appendix A, each of them is explained briefly. It can be seen in Listing A.1, all the APIs will send an HTTP header with the credentials of the user, this credential is the authentication token mentioned above in Section 3.3.1.

4.4 Database Modifications

The database was modified to support clinician-patient association as well as the new features illustrated in the following Figure 4.2, which shows the new tables were added (tables with blue header). The previous model of the database was not deleted, this should be done because it is no longer used. A brief explanation will be given about each new table:

- **TherapistPatientAssociation**: This table was added to enable the clinicians to add patients.
- **UserAccess**: This table is used to record when a user enters the application.
- **ExerciseSentToPatient**: This table is used to keep track of the stimuli sent from the clinician to the patient.
- **UserStatistics**: This table is used to record the answers to an exercise given by a patient. This table also enables the clinician to keep track of each of his patients.

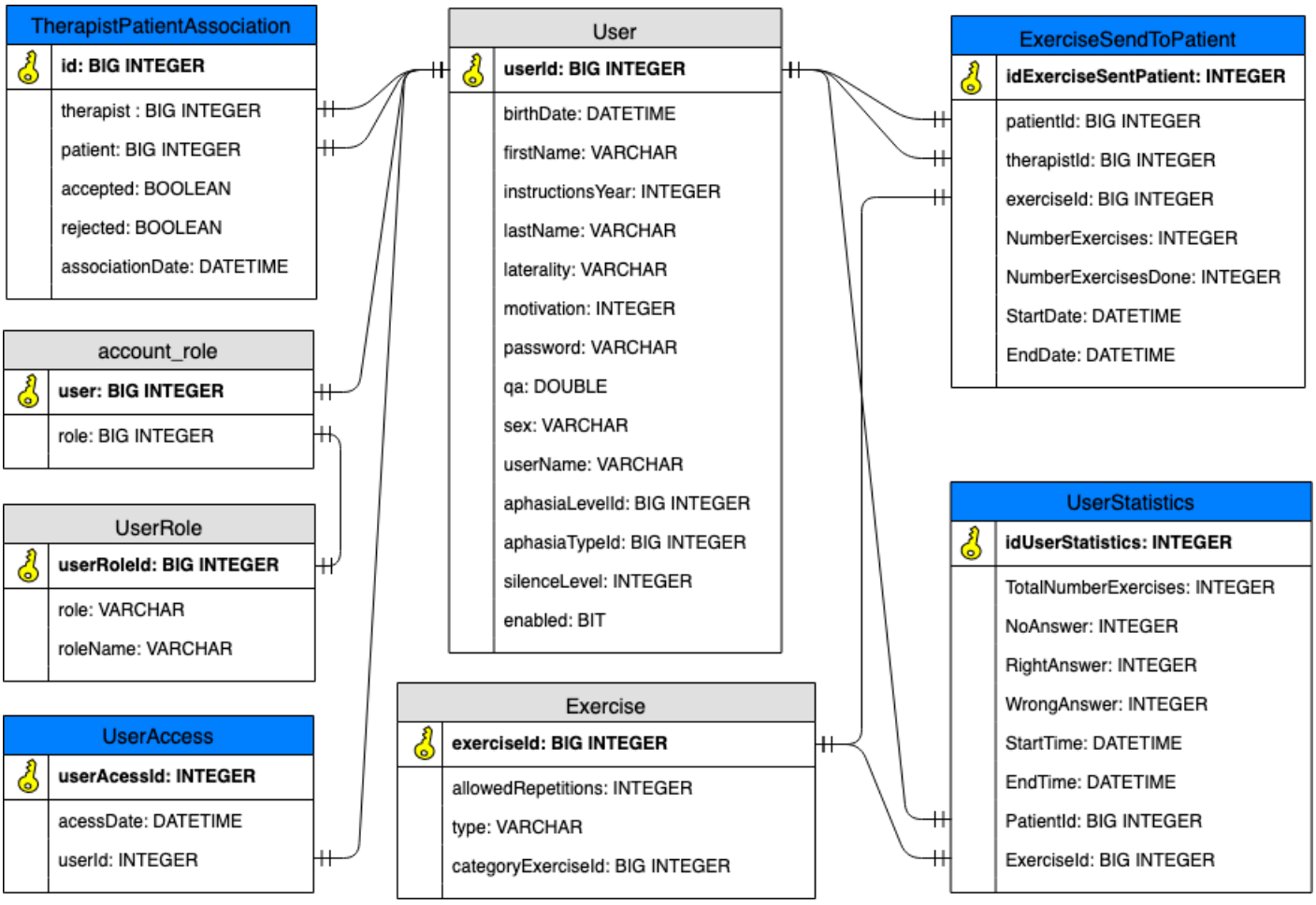

Figure 4.2: Database Structure

5

Evaluation

Contents

This chapter discusses the evaluation of VITHEA 2.0. This project evaluation will determine if the new VITHEA 2.0 mobile application developed has achieved its intended goals and outcomes to ensure that it performs as it is suppose to. To evaluate the performance of the application, user testing were performed to provide evidence on whether the UI and UX of the mobile application was improved as intended or not.

The goal of this thesis is to develop a native mobile application, for the Android platform, including the functionalities offered by VITHEA web application and the older version of VITHEA mobile application. In particular, the specific objectives in terms of application development discussed in Section 1.3 were fully achieved. This chapter describes the evaluation conducted aimed to dilucidate weather the new VITHEA 2.0 mobile application performs as it is supposed to. To this end, besides the general objectives, other features and functionalities were also added in order to improve the UI and UX to make VITHEA 2.0 more reliable, easy to navigate, user-friendly and intuitive. These new features are summarized in Section 5.1. Furthermore, to evaluate the performance of the application, user testings were performed to provide evidence on whether the UX of the mobile application was satisfactory. This evaluation is reported in Section 5.2.

Finally, it is worth noting that the validation of the adequacy of the VITHEA platform as a method for speech therapy is out of the scope of this thesis. The previous assessments performed to evaluate the VITHEA platform [14] were done through a survey of speech therapists from different institutions and online questionnaires of 30 speech therapy professionals. The results achieved were remarkably good with an average score of 4.14 in a likert scale of 5 points (1 to 5).

5.1 New Features in the Client Application

This section discusses the improvements made in the patient module, and in the new clinician module.

5.1.1 New Features in the Patient Module

The following list shows a brief summary of the new features and functionalities added to the patient module in order to meet the current trend of UX.

- The module is now robust enough to adapt to screen rotation and different screen sizes including tablets.
- It is now possible to reproduce all the different types of stimuli: audio, video, images and text.
- Authentication is done through the Android Account Manager. With this feature the user does not have to input the email and password in the application every time they launch the application, the

Account Manager saves a token and this token is used to authenticate the user. This feature is explained in detail in Section 3.3.1

- A setting menu was implemented to provide costum functionalities that enables the patient to customise the application to his/her preference. It is discussed thoroughly in Section 3.3.2.A
- Patient's can track their overall performance and progress in order to verify if they are improving or not, as depicted in Figure 3.18. VITHEA 2.0 also allows user to check how they performed after the end of each round of exercises, the application will prompt a summary statistics as shown in Figure 3.13.
- Adaptation to three virtual therapists, the patients can now select a virtual therapist of their choice from one of the three virtual therapists available.
- Feedback prompts with changing background colour, were implemented to give the user a visual feedback, this is provided after an answer to the exercise is given. This gives user a immediate feedback to know if the response recorded was correct or incorrect as shown in Figure 3.14 and Figure 3.15 respectively.
- Users can challenge themselves by choosing a number of attempts they want per exercise. The range of the attempts goes from one to five, but the user can also deactivate this functionality and have unlimited attempts to finish the exercises.
- While recording the responses, the user can cancel it by sliding the record button to the left. With a limited number of attempts per exercises this feature might be useful, if the user realizes that he has made a mistake.
- The functionality of playing back the audio response after each exercise was added, to let the users know what they said and how it was recorded.
- A logo was created and added as shown in Figure 3.5, to give the application an authentic and clean aesthetic look.

5.1.2 New Features in the Clinician Module

The following list shows a brief summary of the new features and functionalities added to the new clinician module in order to meet the current trend of UX.

- The module is robust enough to adapt to screen rotation and different screen sizes including tablets.

- Authentication is also done through the Android Account Manager. This feature is explained in detail in Section 3.3.1

- An association between clinicians and patients was added, this enable a clinician to add or remove patients that he wants to monitor, as seen in Figure 3.30

- The clinician can now send stimuli to the patient. In order to do so, the clinician needs to add the patients, and they will have to accept the request from their clinician to be monitored. This feature is discussed in Section 3.3.3.A

- The clinician can review their patients progress on their dashboard and accordingly he/she can refine the number of exercises to stimulate improvements in the area where patients might need an extra effort. This feature is discussed in Section 3.3.3.B

- The clinician is able to change his/her profile, this includes, changing his/her name, password, sex, birth of date. This feature is discussed in Section 3.3.3.D

- The logo used in the patient module is also used in this module, as it can be seen in Figure 3.5.

5.2 User Testing

Tests with system users is a crucial part in the development of this system. It is expected that the majority of the users will have an age over 65, so the majority of the participants in the user testing should be over this age limit. Their feedback is essential in order to modify and improve the different aspects of the application.

Due to time constraints, it was not possible to test the application with aphasic patients and real clinicians. 20 people participated in the evaluation, they were divided into pairs, each one tested a module (patient or clinician) and interacted with the application. The following Figure 5.1, depicts the age of the participants. There was an effort in trying to find participants who were above 65 years old, since this is the target age for this application.

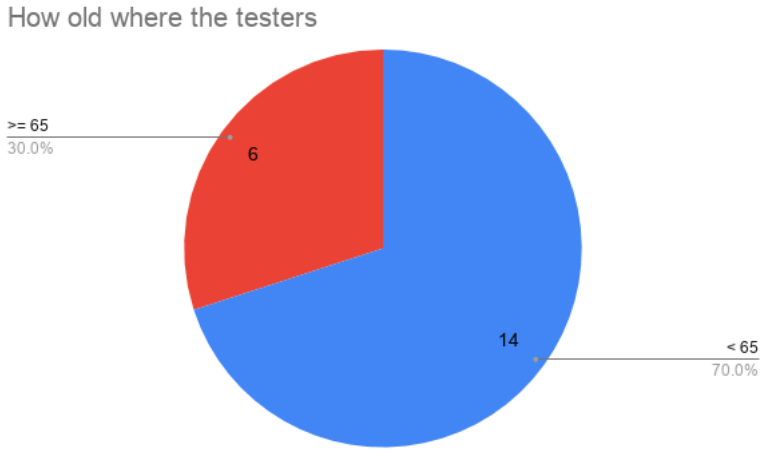

Figure 5.1: Age of Participants

5.2.1 Quality Assurance Testing

Quality assurance testing was performed to verify if all the functionalities of VITHEA 2.0 are working properly or not. This test was done by a reduced number of users that are proficient with mobile applications, to guarantee that the system is functioning as it should and does what it is excepted to. These participants also suggested how to improve some functionalities.

This test was done throughout the development of both modules of the application, in order to check if there were any bugs in the main developed features. It was done with five participants, two proficient with technology and three with limited proficiency. This group was fundamental for the development as they helped to find bugs that were promptly fixed, while also providing suggestions and feedback on how to improve UX. With their feedback, new features were developed. An example of a feature that was developed from feedback, was the ability for patients to be able to change the text size as per their liking, by sliding a seek-bar which is discussed in Section 3.3.2.A.

5.2.2 Subjective Usability Testing

Subjective usability testing gives feedback on how real users interact with the system. It was divided into two parts, first the users were able to explore the application without restraints, in order to get familiar with the UI. In the second part of the test, the users performed certain tasks, that were subjected to usability metrics. The metrics will be discussed in the next section and will be used to evaluate the system. The following list show the different tasks performed by participants as clinicians and as patients respectively.

- **Task for the Clinician Module**

1. Register in VITHEA 2.0 as a clinician;
2. Register a patient to monitor;
3. Send exercises, including 10 random exercises, to the patient;
4. Verify the statistics of the patient after he finished the exercises;
5. Change the name, and another attribute in the user profile;

- **Task for the Patient Module**

1. Register in VITHEA 2.0 as a patient;
2. Select a category of exercises;
3. Select a subcategory of exercises, and complete 15 stimuli;
4. Change the virtual therapist;
5. Change the number of attempts;
6. Select a subcategory of exercises that the clinician sent and complete 15 stimuli;
7. Check the statistics after finishing the exercises;
8. Change the password, and another attribute in the profile;

5.2.3 Objective Testing using Usability Metrics

For objective testing, the following metrics were used to verify if the application was effective, efficient and satisfying.

- **Effectiveness**: Measure the precision and completeness with which the users finishes the tasks, using the following formula.
  - $\frac{\text{Number of tasks completed}}{\text{Number of task undertaken}}\text{x}100$
  - The number of errors will be another measure regarding effectiveness. The formula used to calculate the number of errors is: $\frac{\text{Number of errors}}{\text{Number of attempts}}$
- **Efficiency**: It measures the time required to complete each task.
- **Satisfaction** will be measured through a questionnaire to determine how the application ranks in terms of UI, UX and robustness. A System Usability Scale (SUS) questionnaire will be given to each user at the end of the test, this is a usability measurement tool, consisting of a number of questions with five answer options.

5.2.3.A Effectiveness

Effectiveness tests were performed to check if the participants could produce the desired output and complete the tasks. Of the 20 participants who took part in the test were able to complete all the tasks successfully. A task is considered completed if the participant was able to finish it, regardless of the number of attempts. Using the formula mentioned above the effectiveness was 100%.

The tasks for the clinician module were simple are there is no relevant information to report in terms of errors.

Figure 5.2 shows the percentage of errors reported for the tasks carried out in the patient module, calculated base on the formula mentioned. The majority of the errors occurred while performing the exercises. This is the most complex part of the module so some errors were expected, but as participants got familiar with the application and its UI the error rate also reduces as illustrated in Figure 5.2. It can be seen by comparing two categories, ”Select a subcategory of exercises and complete 15 exercises” and ”Select a subcategory of exercises that the clinician sent and complete 15 stimuli” in Figure 5.2, the second time the exercises were performed there were fewer errors.

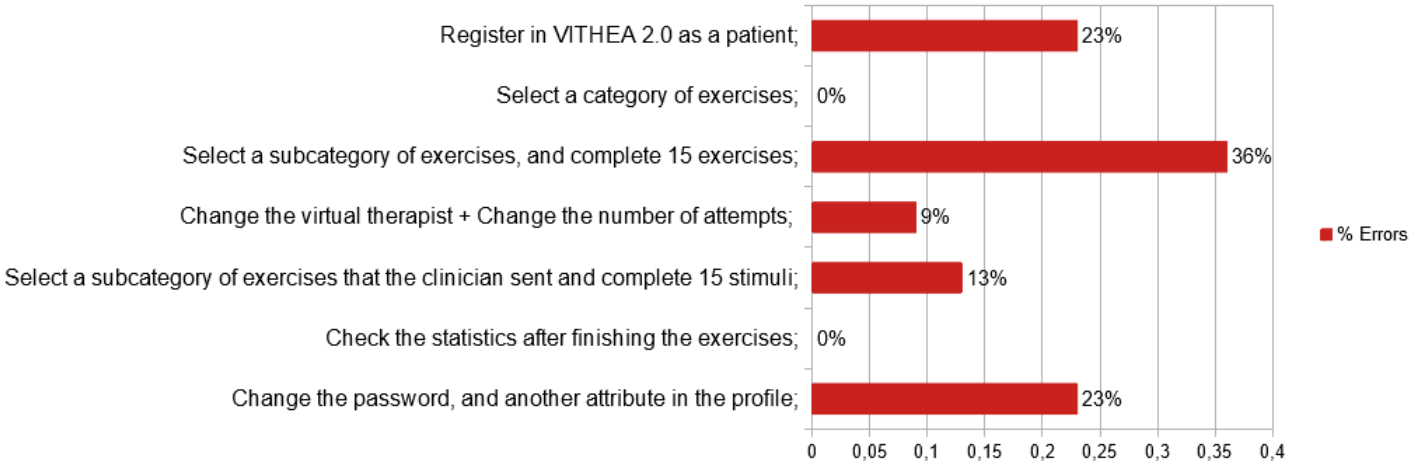

Figure 5.2: Number of errors for the patient module tasks

5.2.3.B Efficiency

The average time that each user took to complete each task was recorded, in order to calculate the effectiveness. A user has as many attempts as he would like in order to complete the task.

Figure 5.3 shows the efficiency measured on the patient module, of each task mentioned in Section 5.2.2. The main focus of these tasks were usability and robustness of the patient module, overall the results are good, with participants being able to complete the tasks within a reasonable time frame.

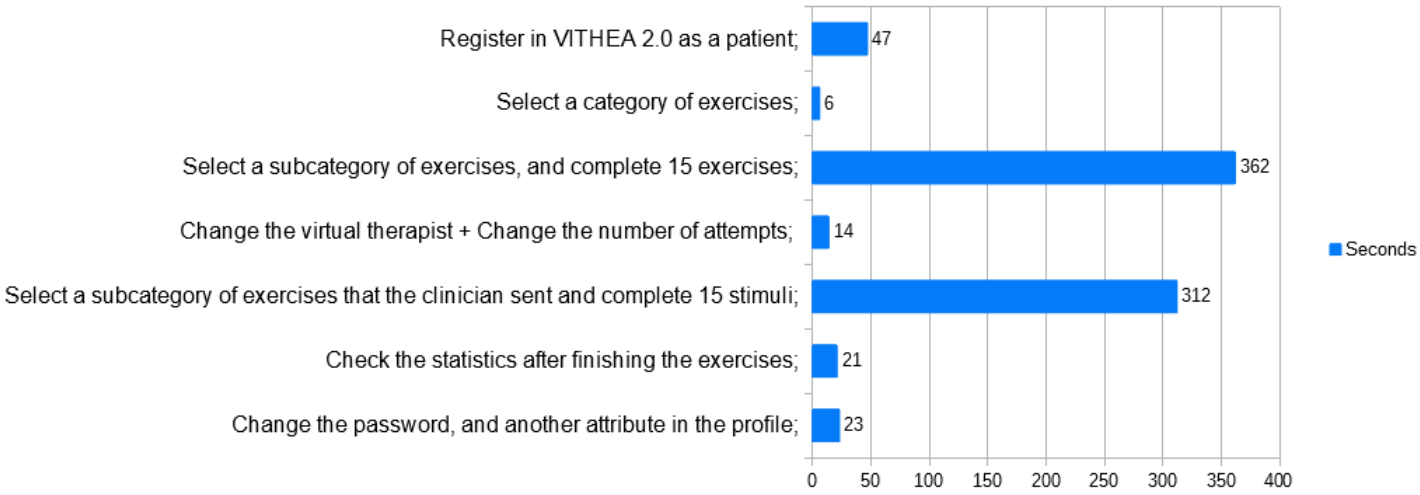

Figure 5.3: Efficiency for the patient module tasks

The second time that the participants tried to complete 15 stimuli, it was faster since they were already familiarised with the UI as illustrated in Section 5.2.2 (the time taken to complete ”select a subcategory of exercises that the clinician sent and complete 15 stimuli” was less than the task ”select a subcategory of exercises and complete 15 exercises”).

Figure 5.4 shows the efficiency on the clinician module, of each task mentioned in Section 5.2.2. The main focus of these tasks were usability and robustness of the clinician module, overall the results are good, with participants being able to complete the task within a reasonable time frame.

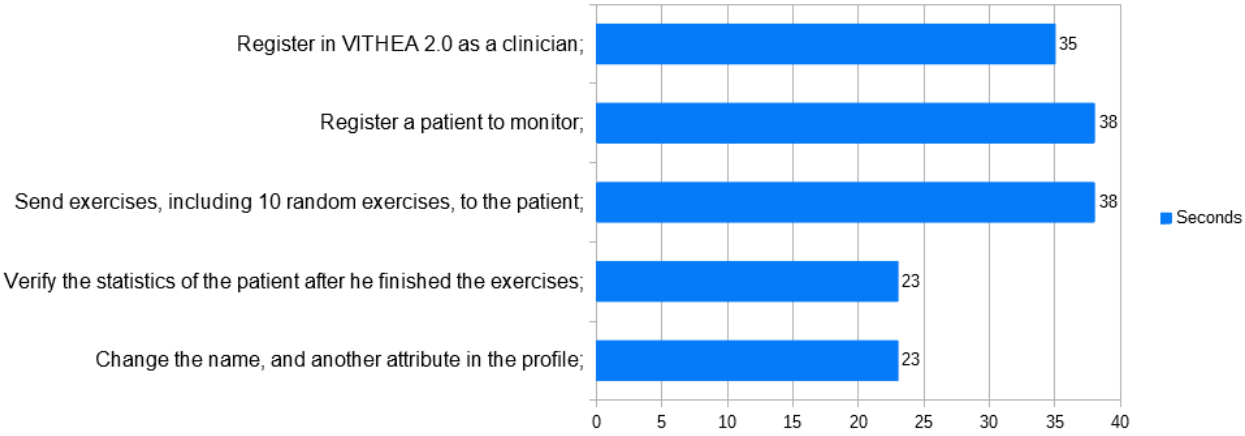

Figure 5.4: Efficiency for the clinician module tasks

5.2.3.C Satisfaction

The following chart illustrated in Figure 5.5, depicts the answers given to the questionnaire. The results from the first four questionnaires are overwhelmingly ”excellent” and ”very good” suggesting that the

participants were satisfied with the UX of VITHEA 2.0. However, the main area of discontent was the virtual therapist.

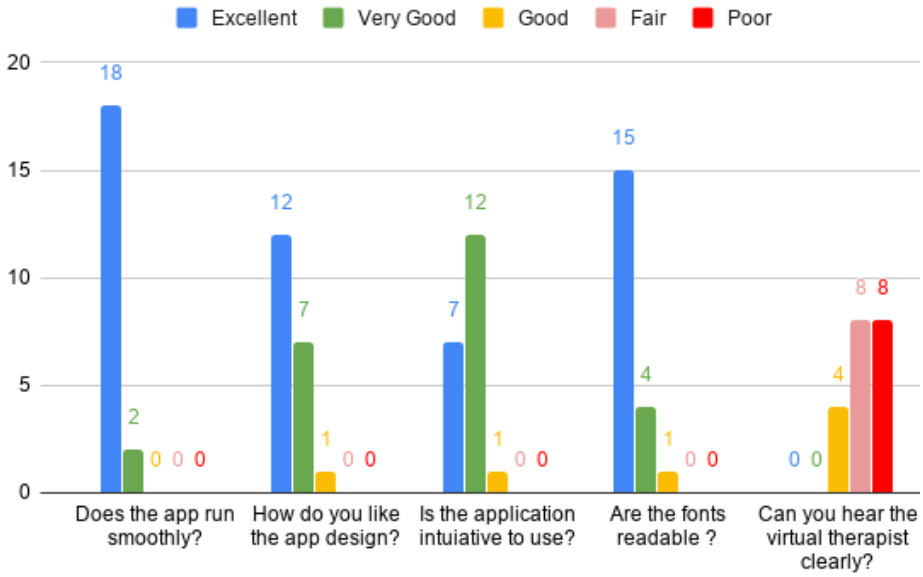

Figure 5.5: Results from the questionnaire

The main complains of the participants were that in some exercises it was not clear what was the intended answer, while others complains were related with the virtual therapist, which was hard to understand. In terms of usability and robustness all the scores are high, as well as intuitiveness, these are key factors while evaluating an application and all scored high.

Virtual Therapist Dissatisfaction

A new version of the virtual therapists was included in this project, this version has three virtual therapist to chose from. Figure 5.6 shows the available virtual therapists.

Figure 5.6: Three virtual therapist: Filipe (left), Catarina (center) and Edgar (right)

In regard to the dissatisfaction shown in Figure 5.5, to the virtual therapist is mainly due to:

- In this version it is not possible to control the speech rate, the main complain when testing the application was that the virtual therapist is hard to understand, being able to control the speech rate would improve this, it could also be added as a setting in the settings menu.
- There is no API to know when the virtual therapist has stopped talking. This is a problem because, for example in the case of audio exercises, it is hard to coordinate when to start reproducing audio.

5.3 Summary

All the objectives that this thesis aimed at, were successfully achieved. Additionally other features and functionalities were added.

In order to evaluate both the modules, testing was done, it was focused in the usability, robustness and UI of the application. In terms of testing, quality assurance testing was extremely helpful, it help to detected bugs early and the participants also gave excellent feedback, that was used to improve or add new features in both the modules. All the metrics show that the system functions as intended, for example all 20 participants were able to perform and finish the tasks given successfully. The number of errors while doing the tests was low, except in the most complex part of the patient module. This could be addressed by having a introductory video showing how to use the system, this video could be played after the patient registers, and would be available in the settings menu, in case the patient has doubts on how to use the system. The results of this survey were remarkably good as shown in Figure 5.5. The application was rated as excellent by more than 50% of the participants concerning the VITHEA 2.0 application in terms of robustness, usability and UI.

The main area of discontent was the virtual therapist because of the difficulty in understanding what is being said. Another complain is that it is not clear what is being asked to do in some exercises. Both the complains are out of scope in terms of this thesis: the exercises were previously done and stored in the database that VITHEA 2.0 uses; the virtual therapist was only integrated and it lacks some features that could be added in the future.

Due to time limitation, the tests and evaluations for VITHEA 2.0 were not performed by real users (Portuguese speaking aphasia patients) and real clinicians. This should be done in the future to fine tune the application, since these are the users that will practice with the application modules.

6

Conclusion

Contents

This chapter reports on the conclusions of this work. First, the aim of the project is restated, then an overall summary of the main achievements and results is presented. Finally, this chapter includes a description of further work that could be carried out in the future in order to make this mobile application a state of the art mobile application.

6.1 Conclusions

A person's speech and language could be significantly impaired due to damage in specific regions of the brain, as in the case of stroke or brain injuries causing aphasia. VITHEA 2.0 is aimed to be a speech rehabilitation Android mobile application, for phones and tablets, that features exercises designed to help people who lost the ability to correctly formulate spoken Portuguese language. VITHEA 2.0, developed in this work, successfully provides a native re-engineered Android mobile application to support the current functionalities of VITHEA platform, as well as extend the features of the existing VITHEA Android prototype mobile application. It also re-engineered all the outdated software and technologies, and added new functionalities to provide users with a more personalized UX, thus making remote virtual speech therapy more immersive for patients.

VITHEA 2.0 mobile application kept the same client-server architecture like its predecessor. The client application was developed in Android Studio and includes two modules, one for patients and one for their clinicians. The server application is based on the previous VITHEA web application which was written in Java. The main components of the server application were kept, only the version were updated, as those components are still relevant and one of today's standard.

The client application presents therapy exercises as well as render to the patients an immediate feedback after each exercise and a summary statistics after each session. This is done so that the patient can keep track of his/her therapy progresses. These statistics can be monitored and managed by the patient's clinician on his/her module so that the clinician can review his/her patients work and progresses. The clinician is also capable of adjusting the patient's therapy by sending a more personalised number and type of stimuli in the area where they might need an extra effort. At the server side, Apache Tomcat was used to host the VITHEAWS using Spring Security and Spring MVC, this module was then integrated with Hibernate in order to communicate with Maria DB (a relational database).

In order to assess both modules, an evaluation was performed focused on usability, robustness and UI of the application. All the metrics have show that the system functions as intended. The number of errors while doing the tests was low except in the most complex part of the patient module. This could be addressed by having an introductory video as mentioned in the previous chapter. The results of the subjective survey were remarkably good as shown in Figure 5.5. The application was rated as excellent by more than 50% of the participants on items related with its robustness, usability and UI.

All the objectives that this thesis aimed at, were successfully pursued and achieve. Additionally other features and functionalities were added.

6.2 Future Work

VITHEA 2.0 system can be extended and enhanced in the following aspects to improve this mobile application:

- Apart from speech therapy, it could also feature new types of exercises targeting other language and cognitive functionalities such as memory, writing and problem-solving skills therapy.
- With other types of exercises that do not depend on the virtual therapist and the ASR, the application could have an offline module that would allow patients to do their therapy without the help of a virtual therapist. For example multiple question exercises like VITHEA Kids.
- Explore and implement Artificial Intelligence (AI) technology to make the current mobile application smarter enough to adapt and optimize exercises as per the patient's improvement. For example, the application will present harder or easier exercises based on patient's performance in real-time, making the application customized for the patient's unique need. Constant Therapy mentioned in Section 2.4.1 does this to improve the patient engagement.
- Incorporate more languages to diversify the user base. This would imply that the exercises should be in multiple language, the virtual therapist should be able to talk in multiple languages and the ASR should also be able to handle multiple languages.
- In terms of database, it would be beneficial to change from MyISAM engine to InnoBD engine. It is a modern transnational engine, it will increase reliability, and recover better from crashes.
- Apache Tomcat was upgraded to the most recent version of Tomcat 7, VITHEAWS was able to deploy correctly so it was considered enough. This version of Tomcat is in long term support, eventually this support will end. Before the support ends, Apache Tomcat should be upgraded to version 9, it requires Java 8 or later in order to function properly. In this project Java was updated to version 8, so there should not be any restrictions while upgrading to Apache Tomcat version 9.
- The clinician module can be provide with additional statistical information in order to improve the monitoring of the patients. For example, it could show in each sub category of exercises, how the patient is doing. Another interesting feature would be to control the time taken to complete each exercise.

- Make two videos to explain how each module works, it would be played upon user registration, it should also be available in a menu or as a help button, in case the user had doubts on how to use the system.
- The virtual therapist needs to communicate with the TTS server, this communication is slow, making the virtual therapist a bottleneck in this system. This could be improved by upgrading the server.
- In the future, if the application is made available to the general public, General Data Protection Regulation (GDPR) must be considered. This is a regulation in European Union (EU) law on data protection and privacy for all individual citizens of the EU and the European Economic Area (EEA). Being a controller of personal data, means that there must be a disclosure stating what data is being collected, what is the purpose, and the statement must also include if the data is being shared with any third parties. The highest-possible privacy settings must be set by default.

Bibliography

[1] National Aphasia Association, "Aphasia definitions - national aphasia association," www.aphasia.org/aphasia-definitions, [Online; accessed 29-December-2018].

[2] A. Pompili, "Virtual therapist for aphasia treatment," Bachelor's Thesis, Rome University of Tor Vergata, 2011.

[3] C. P. B. Filipe, "Vithea kids 2.0," Master's thesis, Instituto Superior Técnico, 2015.

[4] L. Thorburn, M. Newhoff, and S. S. Rubin, "Ability of subjects with aphasia to visually analyze written language, pantomime,and iconographic symbols," *Am. J. Speech Lang. Pathol*, p. 174–179, 1995.

[5] S. K. Bhogal, R. Teasell, and M. Speechley, "Intensity of aphasia therapy, impact on recovery. stroke," *Journal of the American Heart Association*, vol. 34, 2003.

[6] H. Goodglass, "Understanding aphasia: technical report," *In Academy Press, University of California, San Diego*, 1993.

[7] Instituto Nacional de Estatística, "Anuário estatístico de portugal 2017," 2018.

[8] Direção-Geral da Saúde, *A Saúde dos Portugueses 2016*. ISSN: 2183-5888.

[9] V. Bohra, G. A. Vikram, S. Jain, A. Duggal, V. V. Ghuge, and A. Srivastava, "Clinicoanatomical correlation in stroke related aphasia," *Annals of Indian Academy of Neurology*, vol. 18, pp. 424–9, 2015.

[10] Statista, "Mobile phone website traffic worldwide," https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/, [Online; accessed 3-January-2019].

[11] Pedotto, Kelly and Chen, Vivey, "The 2017 U.S. mobile app report," July 2017.

[12] S. Adepu and R. F. Adler, "A comparison of performance and preference on mobile devices vs. desktop computers," in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, Oct 2016, pp. 1–7.

[13] Google Developers - Android, “Android version distribution,” https://developer.android.com/about/dashboards/, [Online; accessed 3-January-2019].

[14] A. Pompili, “New features for on-line aphasia therapy,” Master’s thesis, Instituto Superior Técnico, 2013.

[15] M. Gajewski and W. Zabierowski, “Analysis and comparison of the spring framework and play framework performance, used to create web applications in java,” in *2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, May 2019, pp. 170–173.

[16] Xamarin, “Visual studio tools for xamarin,” https://visualstudio.microsoft.com/xamarin/, [Online; accessed 7-January-2019].

[17] Flutter, “Flutter made by google,” https://flutter.io/, [Online; accessed 7-January-2019].

[18] Gradle, “What is gradle ?” https://docs.gradle.org/current/userguide/what_is_gradle.html, [Online; accessed 27-October-2019].

[19] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, “Comparison of JSON and XML data interchange formats: a case study,” *Caine*, vol. 9, pp. 157–162, 2009.

[20] Apache, “Apache Tomcat,” https://tomcat.apache.org/, [Online; accessed 10-October-2019].

[21] Spring, “Spring framework,” https://spring.io/projects/spring-framework/, [Online; accessed 10-October-2019].

[22] ——, “Spring framework documentation,” https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html, [Online; accessed 11-October-2019].

[23] Hibernate, “Hibernate ORM,” https://hibernate.org/orm/, [Online; accessed 12-October-2019].

[24] MariaDB, “About MariaDB,” https://mariadb.org/about/, [Online; accessed 12-October-2019].

[25] ——, “MyISAM in MariaDB,” https://mariadb.com/kb/en/library/myisam-storage-engine/, [Online; accessed 10-October-2019].

A

Project Documentation

Listing A.1: APIs

```
/*This API will be the first request from the mobile application, it
    will send the username and return the type of account, the type
    of exercises available, and if the user is a patient, it will
    return list of clinicians that would like to monitor his
    progress*/
@POST("getInfoAndExercises")
Call<InfoAndExercisesDTO> getUserInfoAndExercises(
    @Header("Authorization") String credentials, @Part("username")
    String username);

/*This API will concatenate the String exerciseId in the path to the
    server, thus getting the question with that exerciseId, a list of
```

```
    questions with that exerciseId will be returned.*/
@GET("services/getQuestions/{exerciseId}/")
Call<ListQuestions> getQuestions(@Header("Authorization") String
    credentials, @Path("exerciseId") String exerciseId);

/*This API will send the audio from the exercise, the audio should
    be in .wav format, it will also send the questionID, in order to
    get the accepted answers from the database, a string will be
    returned containing "Muito Bem" in case of success or "Tente
    Outra Vez" in case of failure*/
@POST("services/sendResp/")
Call<String> getResponseFromAudimus(@Header("Authorization") String
    credentials, @Part("username") RequestBody username, @Part("qId")
    RequestBody questionID, @Part MultipartBody.Part file);

/*This API will register the user, the class UserDTO has all the
    information required to insert the user in the database, a
    boolean will also be sent in order to know if the user is a
    patient or not.*/
@POST("services/registerUser/")
Call<RegistrationReply> registerUser(@Part("user") UserDTO user,
    @Part("isPatient") boolean isPatient);

/*This API will update the user information, the class UserDTO has
    all the information required to update the user.*/
@POST("services/updateUser/")
Call<SimpleReply> updateUserInfo(@Header("Authorization") String
    credentials, @Part("user") UserDTO user);

/*This API is used by the clinician to add new patients, the class
    SimpleReply will be returned, this class has a boolean that
    specifies if the request was completed successfully and a message
    with the reason why it failed in case of failure.*/
@POST("services/addPatient/")
Call<SimpleReply> addPatient(@Header("Authorization") String
```

```
    credentials, @Part("patient") String patientUsername,
    @Part("therapist") String therapistUsername);

/*This API is used by the clinician to remove patients, the class
    SimpleReply will be returned, it will produce the same result as
    mentioned above.*/
@POST("services/removePatient/")
Call<SimpleReply> removePatient(@Header("Authorization") String
    credentials, @Part("patient") String patientUsername,
    @Part("therapist") String therapistUsername);

/*This API is used by the clinician to get all the patients that
    accepted,being monitored. It will return a list of patients.*/
@POST("services/getPatients/")
Call<List<UserDTO>> getAllPatients(@Header("Authorization") String
    credentials, @Part("therapist") String therapistUsername);

/*This API is used to record the response from the patient, to being
    monitored by a clinician. It will return a SimpleReply object.*/
@POST("services/patientDecision/")
Call<SimpleReply> patientDecision(@Header("Authorization") String
    credentials, @Part("therapistPatientAssociation") long id,
    @Part("reject") boolean reject, @Part("accept") boolean accept);

/*This API is used to save in the database the list of exercises
    that one clinician would like to send to a list of patients. It
    returns a SimpleReply object.*/
@POST("services/sendExercisesToPatient/")
Call<SimpleReply> sendExercisesToPatient( @Header("Authorization")
    String credentials, @Part("therapist") String therapistUsername,
    @Part("patient") String patientUsernames, @Part("exercises")
    String exerciseIds);

/*This API is used to save in the database the statistics from a
    patient after each exercise, these statistics will be used by the
```

```
    clinician to monitor the patients progress. It returns a
    SimpleReply object.*/
@POST("services/addStatistics/")
Call<SimpleReply> addStatistics(@Header("Authorization") String
    credentials, @Part("user") String userName, @Part("statistics")
    UserStatistics statistic, @Part("exerciseId") long exerciseId);

/*This API is used by the clinician to get from the database the
    statistics from a list of patients in order to monitor their
    progress. It returns a object UserStatisticsDTO, this is the
    representation of the fields from the database.*/
@POST("services/getStatisticsFromPatients/")
Call<UserStatisticsDTO> getStatisticsFromPatients(
    @Header("Authorization") String credentials, @Part("patient")
    String patientUsernames);
```