MasterVoicing - A whispers to voiced speech assistant

Maria João Araújo Soutelo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Dr. Eng. Aníbal João de Sousa Ferreira

September 21, 2017
MasterVoicing - A whispers to voiced speech assistant

Maria João Araújo Soutelo

Mestrado Integrado em Engenharia Informática e Computação

September 21, 2017
Abstract

Aphonia, also known as loss of voice, is a condition that affects the human phonetic system and is characterized by the inability of a speaker to produce normal speech. It can range from partial loss, known as hoarseness, to an almost complete loss of voice, where the voice is nothing but a whisper. Its causes can vary, from physical disease related with injuries, medical procedures or bad habits, such as voice misuse, to mental disorders. Whispering is a natural form of speech for people in some social situations where privacy is desired or silence is recommended. However, for patients with aphonia, whispering is generally their primary way of communication. This can become a problem, because of the difficulty to communicate with other people, and can even cause problems in daily lives or even work related activities. There are some solutions for this problem regarding laryngectomized patients, like the use of an electrolarynx, that recreates an artificial voice, the use of esophageal speech and the tracheo-esophageal puncture with prosthesis, but all of them have some disadvantages and require some degree of practice to master speaking. In terms of technologies, there also exist silent speech interfaces, that are not yet convenient solutions. Some mobile applications also try to help with this problem, that are generally based in text-to-speech conversion. They require a text input by the user that is followed by its reproduction in speech, resulting in a slow and unnatural usage. Some of these applications operate in real-time by means of a simple click on predefined buttons with text, which also have limitations. With that in mind, the goal in this dissertation is to develop MasterVoicing, a mobile application, for the iOS platform, whose purpose is to give aphonics another alternative to communicate, using their natural way of communicating - whispering. Its validation is verified by the performance of usability tests and its aim is to work in real-time, integrating a whisper-to-speech conversion algorithm that reconstructs natural, voiced, speech from whispers, giving aphonics an easy tool to regain some of their communication freedom, without the drawbacks of other methods that are available.
Resumo

A afonia, também conhecida como perda de voz, é uma condição que afecta o sistema fonético humano e que se caracteriza pela inabilidade de uma pessoa produzir sons normais de fala. Esta incapacidade pode variar de grau entre a perda parcial de voz, conhecida como rouquidão, até à perda quase total de voz, sendo que neste caso a voz consiste apenas em sussuro. As suas causas podem ser físicas, relacionadas com ferimentos, procedimentos cirúrgicos ou maus hábitos, como mau uso da voz, ou causas psicológicas, relacionadas com problemas mentais ou traumas experimentados. Sussurrar é uma forma natural de comunicação para as pessoas em certas situações sociais em que a privacidade édesired ou o silêncio é recomendado. No entanto, para os afónicos, sussurrar é geralmente o seu principal meio de comunicação. Isto pode revelar-se um problema, por causa da dificuldade de comunicar com outras pessoas, e pode até causar problemas no seu dia-a-dia ou trabalho. Existem algumas soluções para este problema relativamente a pacientes laringectomizados, como o uso de uma eletrolaringe, que recria uma voz artificial, o uso da voz esofágica e a prótese traqueoesofágica, mas todas elas têm as suas desvantagens e requerem alguma prática e aprendizagem para conseguir algo semelhante à voz normal. Em termos de tecnologias, existem também interfaces de fala silenciosa que, contudo, não são ainda soluções convenientes de utilizar. Existem também aplicações móveis que tentam ajudar com este problema, que são geralmente baseadas na conversão texto-para-fala. Elas requerem a inserção de texto por parte do utilizador, à qual se segue a sua reprodução em fala, o que resulta numa utilização lenta e artificial. Algumas destas aplicações funcionam em tempo real, através de um simples clique em botões com texto predefinido, mas têm também limitações do ponto de vista prático. Tendo isto em consideração, o objectivo desta dissertação é desenvolver uma aplicação móvel, denominada *MasterVoicing*, para a plataforma iOS, que pretende fornecer aos afónicos outra alternativa de comunicar, utilizando o seu meio natural de comunicar - sussurrar. A sua validação é verificada pela realização de testes de usabilidade e o seu objectivo é funcionar em tempo real, integrando um algoritmo de sussurro-para-fala, que reconstrói fala natural e audível a partir de sussurros, de forma a fornecer aos afónicos uma ferramenta fácil para recuperarem alguma da sua liberdade de comunicação, sem os aborrecimentos dos outros métodos que se encontram disponíveis.
Acknowledgements

I would like to express my very great appreciation to my supervisor Prof. Aníbal Ferreira for all his help, guidance and useful critiques. I would also like to thank Professor Sérgio Ivan Lopes for his initial help regarding iOS development and Professor Nuno Fonseca for providing some advice regarding specific choices about the implementation of the application.

Besides that, I would like to thank my family and Vítor for all the encouragement and patience.

Finally, I would like to give a special thanks to all the other people that I won’t be naming here who contributed in some way for the concretization of this work.

Maria João Araújo Soutelo
"Every path is the right path. Everything could’ve been anything else. And it would have just as much meaning."

Mr. Nobody (2009)
Contents

1 Introduction ......................................................... 1
  1.1 Context .......................................................... 1
  1.2 Motivation ....................................................... 2
  1.3 Objectives ....................................................... 2
  1.4 Dissertation Structure ......................................... 3

2 State of the Art ..................................................... 5
  2.1 Overview ......................................................... 5
  2.2 Aphonia .......................................................... 5
    2.2.1 Causes ...................................................... 5
    2.2.2 Current Solutions .......................................... 7
      2.2.2.1 Esophageal speech ..................................... 8
      2.2.2.2 Electrolarynx .......................................... 9
      2.2.2.3 Tracheo-esophageal puncture with prosthesis ........... 10
    2.2.4 Silent speech interfaces ................................... 11
    2.2.5 Text-to-speech applications ................................ 13
  2.3 Related Mobile Applications ................................... 17
    2.3.1 Voice Translation Applications .............................. 17
    2.3.2 Voice Changing Applications ................................ 18
    2.3.3 Sound Amplifying Applications .............................. 18
  2.4 Signal Processing Methods for Converting Whispered Speech into Voiced Speech 19
    2.4.1 MELP and CELP Approaches ................................ 20
    2.4.2 Statistical Voice Conversion ................................ 20
    2.4.3 Other Approaches .......................................... 21
  2.5 Conclusions ..................................................... 21
## 3 iOS Overview

3.1 Introduction ........................................... 23  
3.2 iOS History ........................................... 23  
3.3 iOS Development ...................................... 24  
   3.3.1 Technologies .................................... 24  
   3.3.2 Xcode ........................................... 25  
   3.3.3 Programming Languages ......................... 27  
3.4 iOS Distribution ...................................... 28  
3.5 Conclusions .......................................... 28

## 4 MasterVoicing Application

4.1 Introduction and Overview ............................. 29  
4.2 Implementation ....................................... 29  
   4.2.1 Software Development Process ..................... 30  
   4.2.2 Software Development Methodologies and Framework ........................................... 31  
      4.2.2.1 Software Prototyping ......................... 31  
      4.2.2.2 Scrum ........................................ 32  
      4.2.2.3 Adopted Methodology ......................... 34  
   4.2.3 Initial Study ..................................... 36  
   4.2.4 Initial Requirements Analysis ................. 38  
   4.2.5 Initial Tests .................................... 38  
   4.2.6 Throwaway Prototype ............................ 42  
   4.2.7 Application Development ......................... 44  
      4.2.7.1 Development Preparation ...................... 45  
      4.2.7.2 Sprint 1 .................................... 47  
      4.2.7.3 Sprint 2 .................................... 49  
      4.2.7.4 Sprint 3 .................................... 54  
4.3 Algorithm ............................................. 56  
   4.3.1 Implementation .................................. 56  
4.4 Final Product ......................................... 59  
   4.4.1 Features ......................................... 59  
   4.4.2 User Interface .................................... 60  
   4.4.3 Code Structure ................................... 65
## CONTENTS

A.7 Participants’ Opinions/Suggestions - Collected Data ....................................... 107
A.8 Post-Test Questionnaire - Collected Data ....................................................... 108
A.9 Facilitator Guide .......................................................... 109
A.10 Participant Guide ........................................................ 111
A.11 Global Rating - Calculated Values ......................................................... 113
# List of Figures

2.1 Anatomy before and after a total laryngectomy (Adapted from [29]) ... 8
2.2 Esophageal speech (Adapted from [25]) .......................... 8
2.3 Types of electrolarynges (Adapted from [25]) ......................... 9
2.4 Tracheo-esophageal puncture with prosthesis (Adapted from [25]) ... 10
2.5 Overview of the speech production process (Adapted from [34]) .... 11
2.6 Screenshot of a TTS iOS mobile application - "Speak - Text To Speech" (Adapted from [39]) .......................... 14
2.7 TTS devices .................................................. 15

3.1 iOS architecture layers (Adapted from [92]) .......................... 25
3.2 Xcode Interface Builder (Adapted from [112]) ....................... 26
3.3 Xcode Debugger default layout (Adapted from [113]) ............... 26
3.4 Simulator tool (Adapted from [110]) ................................ 26

4.1 iOS audio stack (Adapted from [164]) .............................. 39
4.2 Sample projects - User interface [165] [166] [169] [170] ............ 40
4.3 Test application adapted from SO-32541268 [172] - User interface ... 41
4.4 Throwaway Prototype - Audio engine architecture (Created with [175]) ... 43
4.5 Throwaway Prototype - User interface ................................ 44
4.6 MasterVoicing user interface mockups (Created with [178]) ....... 46
4.7 Sprint 1 - Burndown chart (Adapted from ZenHub [176]) ........ 47
4.8 Sprint 1 - User interface ........................................ 49
4.9 Sprint 2 - Burndown chart (Adapted from ZenHub [176]) ........ 51
4.10 Sprint 2 - Audio engine architecture - After implementing the audio visualization (Created with [175]) ..................... 51
LIST OF FIGURES

4.11 *Sprint 2* - Audio engine architecture - After refactoring (Created with [175]) . . . 52
4.12 *Sprint 2* - Abandoned feature UI components - Edit name and share audio file . . 52
4.13 *Sprint 2* - User interface - Implemented features . . . . . . . . . . . . . . . . . . 53
4.14 *Sprint 2* - User interface - Visual feedback . . . . . . . . . . . . . . . . . . . . . 53
4.15 *Sprint 2* - Audio engine architecture - Final version (Created with [175]) . . . . 54
4.16 *Sprint 3* - Burndown chart (Adapted from ZenHub [176]) . . . . . . . . . . . 54
4.17 *MasterVoicing*’s user interface - Main features . . . . . . . . . . . . . . . . . . 60
4.18 *MasterVoicing*’s user interface - Instructions . . . . . . . . . . . . . . . . . . . . 61
4.19 *MasterVoicing*’s user interface - Settings . . . . . . . . . . . . . . . . . . . . . 62
4.20 *MasterVoicing*’s user interface - Color themes (Created with [184]) . . . . . . . 62
4.21 *MasterVoicing*’s icon (Created with [184]) . . . . . . . . . . . . . . . . . . . . 64
4.22 Evaluation - Effectiveness formula (Adapted from [189]) . . . . . . . . . . . . . 69
4.23 Evaluation - Effectiveness results . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
4.24 Evaluation - Time based efficiency formula (Adapted from [189]) . . . . . . . . . 70
4.25 Evaluation - Overall relative efficiency formula (Adapted from [189]) . . . . . . . 70
4.26 Evaluation - Overall relative efficiency results . . . . . . . . . . . . . . . . . . . . 71
4.27 Evaluation - Task level satisfaction - Tasks difficulty . . . . . . . . . . . . . . . . 71
4.28 Evaluation - Post-test questionnaire - Mean values and 95% confidence intervals . 72

A.1 Usability tests - *Facilitator Guide* - Page 1 . . . . . . . . . . . . . . . . . . . . . 109
A.2 Usability tests - *Facilitator Guide* - Page 2 . . . . . . . . . . . . . . . . . . . . . 110
A.3 Usability tests - *Participant Guide* - Page 1 . . . . . . . . . . . . . . . . . . . . . 111
A.4 Usability tests - *Participant Guide* - Page 2 . . . . . . . . . . . . . . . . . . . . . 112
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison between current solutions for aphonia [24] [25]</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Similarities and differences of the related mobile applications</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Relationship between sections and software development process activities</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Scrum components [135]</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Adapted methodology components</td>
<td>34</td>
</tr>
<tr>
<td>4.4</td>
<td>Sample projects - Capabilities and limitations</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Development tasks - Timeline</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Sprint 1 - User stories</td>
<td>48</td>
</tr>
<tr>
<td>4.7</td>
<td>Sprint 2 - User stories</td>
<td>50</td>
</tr>
<tr>
<td>4.8</td>
<td>Sprint 3 - User stories</td>
<td>55</td>
</tr>
<tr>
<td>4.9</td>
<td>MasterVoicing’s features</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>MasterVoicing’s user interface - Custom animations</td>
<td>63</td>
</tr>
<tr>
<td>4.11</td>
<td>MasterVoicing’s user interface - Alerts</td>
<td>64</td>
</tr>
<tr>
<td>4.12</td>
<td>MasterVoicing’s developed code - Classes (Swift)</td>
<td>65</td>
</tr>
<tr>
<td>4.13</td>
<td>MasterVoicing’s developed code - Views</td>
<td>66</td>
</tr>
<tr>
<td>4.14</td>
<td>Audio performance tests - Results</td>
<td>67</td>
</tr>
<tr>
<td>4.15</td>
<td>Usability tests - Tasks</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>Dissertation objectives - Timeline</td>
<td>73</td>
</tr>
<tr>
<td>A.1</td>
<td>Usability tests - Pre-test interview - Collected data</td>
<td>101</td>
</tr>
<tr>
<td>A.2</td>
<td>Usability tests - Tests tasks - Collected data - Task 1</td>
<td>102</td>
</tr>
<tr>
<td>A.3</td>
<td>Usability tests - Tests tasks - Collected data - Task 2</td>
<td>103</td>
</tr>
<tr>
<td>A.4</td>
<td>Usability tests - Tests tasks - Collected data - Task 3</td>
<td>104</td>
</tr>
<tr>
<td>A.5</td>
<td>Usability tests - Tests tasks - Collected data - Task 4</td>
<td>105</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>A.6 Usability tests - Tests tasks - Collected data - Task 5</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>A.7 Usability tests - Participants’ opinions/suggestions - Collected data</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>A.8 Usability tests - Post-test questionnaire - Collected data</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>A.9 Usability tests - Global rating - Calculated values</td>
<td>113</td>
<td></td>
</tr>
</tbody>
</table>
Abbreviations

AAC  Alternative and Augmentative Communication
ASR  Automatic Speech Recognition
CELP Code-Excited Linear Prediction
DS   Direct Synthesis
EMA  ElectroMagnetic Articulography
EMG  ElectroMyoGraphy
FFT  Fast Fourier Transform
GMM  Gaussian Mixture Models
iOS  iPhone Operating System
MELP Mixed-Excited Linear Prediction
MRI  Magnetic Resonance Imaging
NAM  Non-Audible Murmur
PMA  Permanent Magnet Articulography
SDK  Software Development Kit
SSI  Silent Speech Interfaces
SVC  Statistical Voice Conversion
TEP  Tracheo-Esophageal Puncture
TTS  Text-To-Speech
UI   User Interface
Chapter 1

Introduction

For some voice disorders, as aphonia, whispering is a natural and primary way of communication. Being only able to whisper is a somewhat distressing condition that currently lacks a practical technological solution. The current mobile applications available to solve this problem are limited in terms of functionality or do not work in real time, which means for the user a slow and unnatural usage.

The work of this dissertation consists in the creation of a mobile application for iOS - MasterVoicing - which purpose was to include an algorithm converting whispered speech into voiced speech provided by Prof. Aníbal Ferreira, the supervisor of this work, and its posterior distribution through the App Store.

1.1 Context

According to a study conducted in 2005, claiming to be the largest epidemiologic study concerning the prevalence of voice disorders in the general population, "almost 30% of the adult population has experienced a voice disorder during their lifetime, and nearly 7% reported a current voice problem" [1]. Voice disorders can be a real problem to those affected, depending on their occupation. For instance, voice problems in teachers have a significant occupational impact, that result in some of them missing work and causing restrictions in work-related activities [2]. Furthermore, this condition can prevail in every kind of occupation that relates to an extensive use of the voice, such as with professional singers, actors, attorneys or salespeople [3]. They can also have significant psychological side effects, like depression, anxiety and somatic concerns [4]. One of this disorders is aphonia, also known as loss of voice. With that in mind, the aim of this dissertation is to develop a mobile application for the iOS platform to help patients with aphonia.
Introduction

1.2 Motivation

Being able to only whisper can be embarrassing and even frustrating, both in personal and occupational situations. Therefore, the motivation for the work of this dissertation is, in the first place, to create a mobile application to help people with aphonia or that can only whisper, for some physical or psychological reason, with their impairment. The creation of this application is also motivated by the previous creation of another application to help to reduce the effects of a speech disorder - stuttering - also developed for the iOS platform, called MasterPitch [5]. Moreover, it is driven by the concern with the lack of practical solutions available to solve this problem. Lastly, there is also the desire to create an innovative application, with features that are not currently available in the applications on the Apple’s App Store or Google Play store.

1.3 Objectives

The main objective of this work is to design and develop an iOS application to help people with aphonia - MasterVoicing - and submit it to the Apple’s App Store, which can be divided in the following steps:

- Initial study of the problem and its current solutions;
- Further study of the state of the art about related mobile applications and methods for conversion of whispered speech to voiced speech;
- Familiarization and knowledge acquisition for the understanding of the iOS development environment and frameworks, with a special emphasis on the ones related to audio manipulation;
- Familiarization and knowledge acquisition regarding the programming languages involved in the development of the application and its algorithm (Objective-C, Swift and C++);
- Development of a throwaway prototype, that is a working proof-of-concept, which confirms the viability of the development of the application and provides a better understanding of its possible features - this also includes the performance of preliminary tests with simple real-time audio processing algorithms;
- Development of the application using a methodology adapted from an Agile software development methodology - Scrum - that includes the programming of all its features and the design of the user interface (UI) following the iOS Human Interface Guidelines;
Introduction

• Adaptation and encapsulation in the application of the algorithm for the conversion of whispered to voiced speech;

• Verification and validation of the algorithm;

• Evaluation of the application by the performance of usability tests;

• Distribution of the application through the App Store.

1.4 Dissertation Structure

This dissertation is divided in five chapters. The current and first chapter (Chapter 1) contains the context, motivation, objectives and structure of this document.

The remaining chapters of this dissertation, including a brief overview of their contents, are the following:

• Chapter 2 - Provides an explanation of the problem and current solutions of aphonia, a comparison of existing applications with similarities with MasterVoicing and a brief statement and explanation of some of the methods used for the conversion of whispers into voiced speech;

• Chapter 3 - Contains overall information about the history, development and distribution of mobile applications for the iOS platform;

• Chapter 4 - Documents all the process of implementation of the MasterVoicing application, briefly explains the voice processing algorithm used, and presents the resulting final product and its evaluation;

• Chapter 5 - Includes the achievements, suggestions for future work improvements of MasterVoicing and conclusions of this dissertation.
Introduction
Chapter 2

State of the Art

2.1 Overview

This chapter provides the definition of aphonia - the medical condition that MasterVoicing intends to provide an alternative solution to - and a definition, review and comparison of its current solutions (Section 2.2). Furthermore, it includes an overview about the current mobile applications available, in the two largest digital stores according to its number of applications [6] (App Store\(^1\) [8] and Google Play\(^2\) [10]), that are similar in terms of functionality with MasterVoicing (Section 2.3). Lastly, the final section of this chapter mentions some of the methods developed for the conversion of whispers into voiced speech and briefly characterizes its performance (Section 2.4).

2.2 Aphonia

Aphonia is a voice disorder\(^3\) - a type of speech disorder\(^4\) - that consists in the absence of a laryngeal tone, sounding like whispered speech [12]. It can have many causes, related with physical injuries or mental disorders and it can be a temporary or permanent condition.

2.2.1 Causes

Aphonia can be caused by several physical or psychological conditions [13]. This latter type of conditions are different from the former because an examination of the throat reveals normal movement of the vocal folds [14] and usually those affected cannot speak, but in some cases the patient is capable of vocalize in a whisperlike voice [15] [16]. Known examples of these

---

\(^1\)Store of applications owned by Apple that sells applications for their operating system, iOS (iPhone Operating System), exclusive to their devices [7].

\(^2\)Store owned by Google that sells several digital content, including books, movies, music and applications that runs in devices with the operating system Android [9].

\(^3\)Condition that comprises atypical or absent production of voice (usually related to alterations associated with its characteristics such as quality, pitch, volume, resonance or duration) [11].

\(^4\)A disability related to the articulation of speech sounds [11].
psychological conditions are hysterical aphonia or selective mutism [13], where patients cannot talk because of emotional or mental causes.

Relatively to physical conditions, there are several reasons that can cause aphonia, and may involve injuries or diseases, such as ([13] [17]):

- Benign or carcinogenic tumors on the larynx or thyroid;
- Removal of some part of even the complete larynx structure due to disease\(^5\);
- Injury after neck or chest surgery;
- Damage of the nerves that affect the larynx;
- Vocal folds paralysis\(^6\);
- Severe laryngitis - viral, fungal or bacterial infection [19];
- Thickening of the vocal folds;
- Nodules or polyps on the vocal folds\(^7\);
- Misuse of the voice (excessive and/or incorrect use of voice, smoking, abusive alcohol or caffeine consumption or the exposure to air pollution);
- Excessive coughing due from allergies;
- Breathing problems that affect the speech;
- Gastroesophageal reflux disease (known as acid reflux), where stomach contents flow upwards into the esophagus irritating the vocal folds;
- Primary progressive aphasia - a neurological syndrome that impairs the speech;
- Other disorders of the nervous system like spasmodic dysphonia [20], myasthenia gravis [21], multiple sclerosis [22] and parkinson’s disease [23].

\(^5\)Known as partial or total laryngectomy [18].

\(^6\)It is usually only in one of the vocal folds, but it makes the junction of both hard and ends up affecting the voice and speech [17].

\(^7\)Frequent among people who make a great use of the voice, like singers or teachers.
2.2.2 Current Solutions

Even though aphonia has many causes, aphonics are frequently patients who have been submitted to a laryngectomy [24]. For these patients, it is usually considered by the doctors, that their best option to regain their voice is by means of speech therapy or surgical methods [25] [24]. These methods consist in learning to speak using esophageal speech or the surgical option of inserting a tracheo-esophageal puncture (TEP) with prosthesis [25]. Besides that, there are technological solutions that can be used by all patients with aphonia, such as the electrolarynges, silent speech interfaces [26] and text-to-speech (TTS) applications. There are other basic methods that aphonics can use to communicate such as writing in paper or using sign language, but the solutions presented here allow a user to reproduce speech in some way, that it is not the normal laryngeal speech\(^8\). Therefore, it is considered that there are currently five main solutions for aphonia, which are compared in Table 2.1:

- Esophageal speech;
- Electrolarynx;
- Tracheo-esophageal puncture with prosthesis;
- Silent speech interfaces;
- Text-to-speech applications.

For a better understanding of the first three solutions, it is important to understand the differences in terms of anatomy that a laryngectomy causes. A total laryngectomy is a complex surgery that consists in the removal of several components of the neck. According to [27] it consists in the removal of different types of body parts such as bone, cartilage, membranes like vocal folds, some rings of the trachea and even some muscles, like all of the muscles of the larynx, which possess a significant impact on the respiratory, swallowing and speaking capabilities. As this is a complicated process, we can simplify its understanding by focusing on the main components of the neck. As illustrated in Figure 2.1a, the human throat contains different components, including the esophagus, the trachea and the larynx. The esophagus is the organ through which food passes, the trachea is the tube through which the air passes to and from the lungs and the larynx contains the vocal folds, that are essential for phonation [28]. After a laryngectomy, the larynx is removed, and all the connection between lungs and mouth is cut off, that is, the esophagus and trachea are

\(^8\)Traditional speech which involves the oscillation of the vocal folds.
State of the Art

no longer connected at any point (Figure 2.1b) and the laryngectomized patient breathes through a hole in the neck called stoma [28].

![Figure 2.1: Anatomy before and after a total laryngectomy (Adapted from [29])](image)

2.2.2.1 Esophageal speech

Esophageal speech is a form of speech where the speaker uses the esophagus as a pseudo-lung where air is stored and released in a timely fashion to produce voice [24] and it is the oldest method of voice restoration, that has been used for more than 100 years [30]. This means that it involves oscillation of the esophagus, in contrast with the tradition laryngeal speech that involves the oscillation of the vocal folds. It is a cost effective solution [30], because it only requires speech therapy sessions to learn and does not need extra devices, maintenance or procedures [25]. Even thought it gives a resemblance of the same freedom of speech as laryngeal speech, because it does not require the use of the hands, it has some disadvantages, the most important being that there can be difficulties with loudness and phrasing (the voice is often relatively low in volume and the length of sentences that can be spoken continuously are short [25]). Besides that, it is also difficult to learn for most of the patients and requires some time to master, usually in the span of months (a period of nine to twelve months),

![Figure 2.2: Esophageal speech (Adapted from [25])](image)
which means that some patients prefer other solutions or just give up this method after not seeing fast results (the success rates of the development of useful speech are about 70%) [24].

2.2.2.2 Electrolarynx

An electrolarynx, alternatively known as artificial larynx, is a simple and straightforward device that allows a means of producing speech after a total laryngectomy, and whose primary advantage is that it can be used right after leaving the hospital after surgery [24]. The first electrolarynx was developed in 1859, at a time when the primary speech recovery method was esophageal speech [30]. It creates a vibration that simulates the vibration of the vocal cords and can be transformed as the patient articulates words in order to produce speech [25]. The voice produced can be of great volume, the device usage is uncomplicated and does not require much learning effort [25]. The only disadvantages are that the voice created is quite mechanical, requires the use of the hands, battery replacement or recharging and can be quite expensive [25] [31]. There are two types of electrolarynges, as shown in Figure 2.3 [25].

![Electrolarynx](image)

(a) Electrolarynx of contact  
(b) Electrolarynx with oral adapter

Figure 2.3: Types of electrolarynges (Adapted from [25])

The most common one is the contact electrolarynx (Figure 2.3a - also called extraoral electrolarynx [30]). This device is used by the contact with the skin of the neck, where the vibration is transmitted within the mouth through the tissues of the neck into the resonating cavities of the vocal tract [24]. The other one is the electrolarynx with oral adapter (Figure 2.3b - also called intraoral electrolarynx), that has an supplementary part that resembles a straw (also termed an oral tube) [25], where the vibration is produced directly inside the mouth.
2.2.2.3 Tracheo-esophageal puncture with prosthesis

Until 1979, when the introduction of the first approach to a surgery of tracheo-esophageal voice restoration with a prosthesis was made, the principal methods of communication after a total laryngectomy were the esophageal speech and the electrolarynx [27]. A tracheo-esophageal puncture is a surgical procedure that can be done to patients who were submitted to a laryngectomy during the same surgery (primary TEP) or in a later date in a separate procedure (secondary TEP), but the basic technique of the procedure is the same [25]. During the surgery a small hole, or puncture, that goes from the back wall of the trachea into the front wall of the esophagus, exactly behind of the trachea, is made where a prosthesis is inserted [25] [27]. This prosthesis has a unidirectional valve that lets air to flow, through the prosthesis, from the trachea to the esophagus and prevents the passage through the prosthesis into the trachea of the esophagus contents, such as food and liquids, during swallowing [25]. When the prosthesis is in place, it is possible to create tracheo-esophageal speech by covering, after breathing and while exhaling, usually with the thumb (Figure 2.4), the opening in the neck (stoma). This happens because the air passes through the prosthesis and is expelled inside of the esophagus, going up the throat and out of the mouth, creating sound, while the mucosa of the esophagus and pharynx vibrate with the air passage [25] [27].

![Figure 2.4: Tracheo-esophageal puncture with prosthesis (Adapted from [25])](image)

The main disadvantage of this solution is that the prosthesis requires maintenance, such as cleaning regularly to avoid infections, so that failure in proper care often leads to health problems.
State of the Art

[24]. Other disadvantages are that the surgery and maintenance can be expensive, since the prosthesis requires changing every few months⁹ [25]. Besides that, it can also have problems related with leakage from the esophagus into the trachea and fungal infections [25]. Nevertheless, the speech created by a TEP prosthesis can be very clear and improves with practice, and there are also extra devices that allow for hands-free usage, like the adjustable tracheostoma valve (Figure 2.4) [25]. Moreover, the success rates of development of useful speech are between 80% to 90% for tracheo-esophageal speech, that is different but not necessarily superior or inferior to esophageal or electrolaryngeal speech [24].

2.2.2.4 Silent speech interfaces

Traditional speech recognition methods are based on voice elements identification that cannot be used by people with speech-related disorders, because of the lack of an audible acoustic signal [33]. By contrast, a silent speech interface (SSI) is a technology, still in development stages, that enables speech communication, by using different sensors to acquire information from the inaudible elements of the human speech production process, like the articulators¹⁰ or the brain [35]. This produces a digital representation of speech that can be synthesized and interpreted [35] to produce artificial speech. The speech production process is thought to be the most complex motor task performed by human beings and implies the coordination of a complex set of events [34]. For a better understanding of this process, it is better to think about it considering its division into several stages, as the model proposed in Figure 2.5 suggests.

Figure 2.5: Overview of the speech production process (Adapted from [34])

It starts with the conceptualization and formulation of speech in the brain, following with the activation of the muscles related to the articulatory control, that create the movement of the

⁹There are two types of prosthesis: indwelling and non-indwelling, whose difference is that the first needs to be removed and retit by a professional and the latter can be removed and changed by the patient [32].

¹⁰The most important elements of the human body for speech production [34].
articulations, that result in observable and acoustic articulation effects, like the movement of the vocal tract, facial changes, non-audible murmur\textsuperscript{11} (NAM) and speech.

Silent speech interfaces can be used for communication by people with speech disorders, or people without any problem of speech in noisy environments or when security or privacy is required \cite{33}. Different kinds of sensors exist to capture biosignals related to speech, that are used not only to perform speech recognition without sound, but are also the core elements of several previously proposed types of SSIs \cite{33} \cite{36}. Regarding the decoding of the brain activity related to the speech process, significant work has been done, but with limited success. In terms of articulatory control, there are some SSIs that use the sensing of the electrical activity of the muscles related to the speech process, such as the surface electromyography (EMG)\textsuperscript{12} \cite{36}. Others use the sensing of the movements of the speech articulators, such as the use of non-audible murmur NAM microphones \cite{33}, electromagnetic articulography (EMA)\textsuperscript{13}, permanent magnet articulography (PMA)\textsuperscript{14}, magnetic resonance imaging (MRI), radar, video and ultrasound or a mix of both \cite{36}.

All of this speech-related information captured by these approaches is then utilized to find out the digital acoustic signal related with the desired speech information \cite{36}. This is usually done by resorting to automatic speech recognition (ASR) to decode the information collected by the sensors and then using a TTS synthesizer for the generation of the final recreated artificial voice from the recognized text. However, this method has some disadvantages, that brought up the creation of an alternative approach, called direct synthesis (DS). As the name suggests, this approach consists in the direct speech synthesis from the information gathered by the sensors to the corresponding acoustic signal without an intermediate recognition step \cite{36}. Comparing with the "recognize first and synthesize later" approach, DS has some advantages like not being limited to a specific vocabulary, being language-independent and allowing real-time speech synthesis \cite{36}. Particularly for the laryngectomized, this can involve simultaneously recording both data using different SSIs’ sensors and the patient’s voice before laryngectomy\textsuperscript{15}, since the sensors’ data alone do not offer sufficient information about the speech characteristics. Then, after the laryngectomy, all this information is used to generate artificial speech \cite{36}. If it is only possible to record the

\textsuperscript{11}NAM sounds are sounds with low amplitude created in the vocal tract, that result of the resonance that air produces while it passes through the larynx \cite{33}.

\textsuperscript{12}It is called "surface" because it uses non-invasive electrodes, unlike conventional EMG \cite{35}.

\textsuperscript{13}Monitorization of the motion of a set of stationary points inside of the vocal tract \cite{35}.

\textsuperscript{14}Capturing the magnetic field generated when a user "speaks" by using several magnets attached to the articulators \cite{36}.

\textsuperscript{15}The recording of the voice prior to surgery is also done for the use in TTS applications, using services like "my-own-voice", to create a synthetic voice that captures characteristics of the original voice \cite{37}.}
voice before surgery, the patient is asked, after the surgery, to mime speaking along to their own voice to gather the necessary sensorial information [36]. In other cases, where no information is gathered before the surgery, patients can mime along to a "donor voice"\footnote{Patients can request a family member or friend to have their voice synthesized and then use it for themselves [37].} that ends up being the artificial voice they speak with [36].

According to some sources [35], some of these solutions work accurately in silence and in noisy environments, and can be used by patients with aphonia and are low cost in terms of production. However, there are still problems regarding the fact that they are not quite ready for the market and are still very obtrusive for the user in terms of the devices that they need to wear for them to work (most of these systems require the use of contact sensors that are unpleasant and uncomfortable or use optical systems susceptible to external and ambient interferences [33]). However, that may change in the future, with some new research and development about contactless silent speech recognition systems\footnote{Like the proposed application to implement a contactless SSI using as a sensor an impulse radio ultra-wide band (IR-UWB) radar [33].} [33] [35].

\hspace{1cm} 2.2.2.5 Text-to-speech applications

Text-to-speech applications are applications that use a device, usually a smartphone, a tablet or a computer, but can also have their own specific hardware, known as TTS devices, that allow a user to read text out loud. These applications reproduce artificial speech from text, usually text inputted by the user, but can have more features. They are the most common solution for common people to work around the problem of aphonia.

In these applications, the most common and basic feature operates as follows: the user inputs what he/she wants to communicate in the form of text and the application speaks for him/her at the push of a button (features in color red in Figure 2.6 - the two features on the top right of the figure).

For smartphones and tablets, there are other extra features (as an example, the features in color blue in Figure 2.6 - the two features on the bottom right of the figure and left) that can be found in this type of applications like:

- Favorite phrases: the ability to add phrases to a list of favorites, with the main purpose of using them frequently\footnote{Some can be organized in categories based on location, audience, and situation [38].} [39] [40];

- Phrases history: a record of the previously "spoken" phrases [38] [39];
State of the Art

- Settings for the characteristics of the artificial voice, like the volume, pitch, rate, gender, language and speaker\(^{19}\) [39] [40] [42] [43] [44];
- Save text as audio file for later reproduction (for example in a presentation) [43] [45] [46];
- Buttons with text, sorted by category (customizable [47] [48] or predefined [49]);
- Prediction of the most likely next word the user is going to write [50] [51];
- Speaking each word, sentence or new line as it is typed [40] [50];
- Clear text after speaking [39] [40];
- Reading out loud the text of files in several formats like ".pdf", ".epub" or ".docx" [42] [52];
- Reading out loud Internet’s webpages by fetching its text content given their URL [53].

![Figure 2.6: Screenshot of a TTS iOS mobile application - "Speak - Text To Speech" (Adapted from [39])](image)

Besides that, there are TTS devices, that usually have even more features, such as the ability to send emails and text messages [54] [55]. Some of them also have features typically found on computers and smartphones such as the access to social networks, the ability of taking and sharing photos, browsing the Internet or playing games. In most cases, they are considered to be

\(^{19}\)Some applications have different speakers for the same language [39] [40] [41].
alternative and augmentative communication (AAC\textsuperscript{20}) devices, and can be a good communication tool for aphonics, thanks to their features. The main advantage between using these devices and a general TTS application in a smartphone is that the hardware of these devices is usually designed with the intent of facilitate communication. This means that the hardware of TTS devices can have features that, like smartphones, unusually have. For example, good speakers (Figure 2.7a) can help making the artificial speech audible in places with noise \cite{57} and a dedicated real keyboard (Figure 2.7b) can help those who prefer to type in real keys instead of a touchscreen \cite{58}.

![TTS devices](image)

\textbf{Figure 2.7: TTS devices}

In terms of usability, there are what seem to be very good applications, according to their reviews, that use the predefined buttons category mixed with button personalization \cite{47} and possibility of eye control \cite{59}, which are probably the fastest way available, in terms of technological solutions, for a person that cannot speak to communicate. Another advantage of these applications is that most of them are free and are available in different platforms and devices. However, they can also be expensive \cite{47} \cite{54} \cite{57} and do not match the same freedom that \textit{MasterVoicing} intends to provide its users, by allowing them to speak in real-time without adding any input besides their own "voice". Another disadvantage of these applications is that they are only as fast as its user is at inputting text. For that reason, they are not a very convenient solution for real-time conversations. Besides that, it is also important to note that, according to UNESCO\textsuperscript{21}, almost 17\% of the world’s adult population is not able to read nor write \cite{60}. Therefore, this is not a solution for everyone. Moreover, the main important features found in these applications can be scattered around, that is, to get all the functionalities cited above, the user may have to download several applications and use different devices\textsuperscript{22}, which is also an inconvenience.

\textsuperscript{20}Consists in all means of communication, besides oral speech, used to express thoughts \cite{56}.

\textsuperscript{21}United Nations Educational, Scientific and Cultural Organization \cite{60}.

\textsuperscript{22}Because there exist for example applications for tablets only \cite{50}.
Table 2.1: Comparison between current solutions for aphonia [24] [25]

<table>
<thead>
<tr>
<th>Solution</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Esophageal speech</td>
<td>– Does not require devices or procedures</td>
<td>– Difficult to learn for most patients</td>
</tr>
<tr>
<td></td>
<td>– It is cost effective</td>
<td>– Requires some time to master optimal voice quality</td>
</tr>
<tr>
<td></td>
<td>– It is hands-free</td>
<td>– The voice is often low in volume and the length of sentences that can be spoken continuously are short</td>
</tr>
<tr>
<td>Electrolarynx</td>
<td>– Can be used right after surgery thanks to its simplicity of usage</td>
<td>– Produces a mechanical artificial voice</td>
</tr>
<tr>
<td></td>
<td>– The voice produced can be of great volume</td>
<td>– Requires maintenance and it is not hands-free</td>
</tr>
<tr>
<td>TEP with prosthesis</td>
<td>– Speech can be very clear and improves with practice</td>
<td>– The prosthesis requires regular cleaning and changing, can fall out and it is not always hands-free</td>
</tr>
<tr>
<td></td>
<td>– Do not require actual speech, only speech movements</td>
<td>– It is usually expensive</td>
</tr>
<tr>
<td>Silent speech interfaces</td>
<td>– Work in places with a lot of noise</td>
<td>– Not yet ready for commercial uses</td>
</tr>
<tr>
<td></td>
<td>– Not yet ready for commercial uses</td>
<td>– Need more research and development to work consistently and accurately</td>
</tr>
<tr>
<td>Text-to-speech apps</td>
<td>– Many are available for free</td>
<td>– Can be expensive if the user wants a dedicated device or more functionalities</td>
</tr>
<tr>
<td></td>
<td>– A better solution than other simple writing communication methods</td>
<td>– Not very convenient solution for someone to have a conversation in real-time (are only as fast as its user is at inputting text)</td>
</tr>
<tr>
<td></td>
<td>– A better solution than other simple writing communication methods</td>
<td>– Require that its user knows how to write</td>
</tr>
</tbody>
</table>
2.3 Related Mobile Applications

In terms of mobile applications, a thorough search of the Apple App Store and the Google Play Store\(^{23}\) show that there is not any mobile application to assist people with aphonia, with their sporadic or permanent disability, that uses whisper-to-speech conversion. However, there are applications that can be related to the application developed for two reasons: the first is that they try to solve the same problem and the other is their similarities in terms of functionality, even though being used for other purposes. Related to the first reason are the TTS applications, mentioned in the previous section (Section 2.2.2.5). The other type of applications found on the applications’ stores that are related to *MasterVoicing* for the former reason can be sorted in the following categories:

- Voice translation;
- Voice changing;
- Sound amplifying.

All of these applications record and process speech. Some of them use it to retrieve information and others only to transform its audio signals in some way. The similarities and differences of these three types of applications with respect to the *MasterVoicing* application are summarized in Table 2.2 for a quick comparison, after the three following sections that contain more detailed information about each type.

2.3.1 Voice Translation Applications

Voice translation applications \[^{[61]}\] \[^{[62]}\] \[^{[63]}\] are applications that, as the name suggest, translate spoken words between different languages. They work by recording what a user says in a given language and give the corresponding translation of that same recording in another language. They do not actually work in real-time, that is, they are not always recording what the user says. This is one of the main differences between these applications and the *MasterVoicing* application. The process of functioning of these applications is recording the voice of the user, processing the sound to identify the spoken words, translate them and then use a TTS synthesizer with the correct audible translation. Another difference is that this process is language dependent, and needs to process the actual words the user say, while *MasterVoicing* is language independent. Finally, the

\[^{23}\)This search was narrowed to these two stores because they are the ones which have the higher number of applications available for download \[^{[6]}\].
last main difference is that they do not actually change the recorded speech, they only use it to process its meaning, so that the final result is not a changed version of the original recorded voice, being instead an artificial voice produced by a TTS synthesizer.

2.3.2 Voice Changing Applications

Voice changing applications work in a similar way as voice translation applications, in the way that they record what the user say, but to a different end - to allow it to change the recorded audio with different sound effects. However, they are different from the previous applications in the sense that they do not require the processing and understanding of the speech that is embedded in the recordings. Also, the sole purpose of these applications is recreative, not having an actual useful end as the previous applications, hence the sound effects are, for example:

- Based in changes of the basic characteristics of the voice, like pitch or speed [64];
- Robotic, "ghostly", helium voice, alien and monster alike voice effects [65] [66] [67];
- Changing the voice to sound like the voice of a celebrity or fictional character [65] [66].

They are marketed as applications for voice changing that allow to save the recorded voices just for fun or for example to make prank calls24 [68] [69] [70] [71].

Most of these applications also do not work in real-time [64] [72] [73] or, if they do, usually require headphones [66] [74]. Nevertheless, there are cases of these applications that work in real-time and without headphones, but may not do it very successfully, according to certain reviews [75].

2.3.3 Sound Amplifying Applications

Lastly, sound amplifying applications have another main purpose - to help people with hearing problems [76] [77] [78] [79] - even thought they are sometimes marketed by its programmers for another purposes, like "auditory espionage" [80]. They typically record the surrounding sounds using the microphone of the mobile device and then process and enhance them to provide an easier and clearer audition to the user [81]. Just like MasterVoicing, they work in real-time, and are always recording and producing sound, but require headphones to work (the reason for this may be avoiding problems related with feedback or simply because the usability of these applications makes more sense with them).

24Even thought this requires extra charges to the user [68] [69] [70] [71].
State of the Art

Table 2.2: Similarities and differences of the related mobile applications

<table>
<thead>
<tr>
<th>Application Type</th>
<th>Similarities</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice translation</td>
<td>– Work without headphones</td>
<td>– Do not work in real-time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– The spoken voice has no resemblance to the original recorded voice</td>
</tr>
<tr>
<td>Voice changing</td>
<td>– Some work in real-time</td>
<td>– Have only recreative purposes</td>
</tr>
<tr>
<td></td>
<td>– Some work without headphones</td>
<td></td>
</tr>
<tr>
<td>Sound amplifying</td>
<td>– Work in real-time</td>
<td>– Need headphones to work</td>
</tr>
<tr>
<td></td>
<td>– Some exist with the intention to help people with disabilities</td>
<td></td>
</tr>
</tbody>
</table>

2.4 Signal Processing Methods for Converting Whispered Speech into Voiced Speech

Currently, the majority of technological systems related with speech communication, such as ASR systems, operate exclusively with audible and vocalized speech. This means they are incapable of operating with whispered speech or do it rather poorly [82] [83]. The reason for this to happen is that, according to its nature and mechanism of production, whispering is considerably different from voiced speech [84]. Whispered speech is characterized by "a noisy structure" and the lack of sound production that involves moving the vocal folds [84] [85]. For this reason, whispering does not have the same fundamental frequency of voice, pitch contours and other important features [84] [85]. Also, the power of its sound spectrum is weaker compared to voiced speech, which makes it more susceptible to the interference of ambient background noise [82] [84] [85]. Due to these differences, processing and reconstructing whispered speech is more challenging than voiced speech [85], which include two possible approaches [82] [83]:

- Work directly with the whispered voice input;
- Work with a "speech-like" form of signal created from the whispered voice input.

The latter approach may be preferred and it is usually termed "speech reconstruction" or "conversion of whispers into natural speech" [82] [83].
The conversion of whispered speech to voiced speech can be done resorting to several methods such as:

- Mixed-Excited Linear Prediction (MELP);
- Code-Excited Linear Prediction (CELP);
- Statistical Voice Conversion (SVC);
- Other new approaches or hybrid or improved versions of MELP and CELP.

The first two methods consist in voice coders (also known as vocoders) that were originally designed for the transmission of compressed speech, but are now used for the purpose of relating whispered and voiced speech signals.

2.4.1 MELP and CELP Approaches

MELP is the oldest of the vocoders and reportedly works considerably well. However, it requires training using the original voice of the speaker before performing whispers to voiced speech conversion. This means that it cannot be used when it is not possible to record the original voice of the speaker. Also, it is not appropriate for real-time operation.

CELP introduces the use of an "excitation codebook" with the purpose to remove the requirement of a priori speaker's original voice information, and provides a more natural synthetic speech.

Both MELP and CELP work well for phonemes and single words, but results are poor and lack evaluation for continuous speech. Moreover, they have a considerable computational complexity. These two approaches work by decomposing whispers into components and then reconstructing speech with transformed (or improved) pitch information. The idea behind this is that whispered speech is similar to speech with no pitch and therefore it can also be decomposed just like fully voiced speech.

2.4.2 Statistical Voice Conversion

Another approach, that was created more recently, is the Statistical Voice Conversion (SVC). As the name suggest, this method uses statistical methods - making use, for example, of Gaussian Mixture Models (GMMs) - to model pitch and parameters of the speech from parallel whispered and voiced speech training information. Some of the implementations of these methods
use NAM microphones, that capture body-conducted unvoiced speech and have some advantages like being immune to external noise disturbances and providing an effective conversion [86]. Even so, they have some disadvantages, such as not taking into consideration some information that is suppressed in the process of capturing information, not being suited for real-time operation [86] and involving a high computational complexity [83].

### 2.4.3 Other Approaches

Other approaches, new, or based on the previous methods [91], have also been introduced, with the main difference that some of them do not require a priori speaker information (do not require parallel original speech and whisper input information) [87]. Also, they have other advantages such as a lower computational complexity, compared to the previous ones, are designed for real-time operation and improve the reconstruction quality of the whispered voice [82] [87].

### 2.5 Conclusions

In order to create an application that can help aphonics, it is necessary to understand the problem of aphonia, as well as its current solutions, their problems and their limitations. This chapter gives some insight into this matter as well as the current state of what the current mobile applications’ technologies can offer in terms of features. According to the research fulfilled, two conclusions are that even thought there are currently some different solutions for aphonia, that have been improved and created over the years, they all have different disadvantages and some need further development. It is proposed that a new solution lies in the development of a mobile application that converts whispers to speech, combining, in a certain way, the technologies of mobile applications and the methods for reconstruction of natural speech from whispered speech, also referred in the last section of the chapter.
State of the Art
Chapter 3

iOS Overview

3.1 Introduction

iOS is the operating system that runs on iPhone, iPad and iPod Touch devices, managing its hardware and providing the required technologies for the implementation of native apps [92].

This chapter gives a brief overview about iOS history, development and application distribution. Apple provides a more thorough and detailed insight about the content addressed in Sections 3.3 and 3.4 in the Apple Developer website [93]. This chapter will focus only on information related with iOS, even though it shares some aspects with other operating systems used by different Apple devices, such as macOS, tvOS and watchOS.

3.2 iOS History

Before actually getting into the theme of iOS development, it is important to understand a bit of this operating system’s history and evolution.

The creation of iOS was marked with the introduction of iPhone, in January of 2007 [94]. The current Apple devices’ operating system didn’t have a specific name until March of the same year, when the beta version of the first Software Development Kit\(^1\) (SDK) was launched for the iPhone [96]. After that, it was known as iPhone OS [96].

In 2008, Apple introduced the App Store, a place where anyone could sell an app for a price of their choosing [94]. This happened the year after Apple first told developers that the development of applications in iPhone would be made by creating web applications for Safari [97], its native web browser. Since that, iPhone OS changed and evolved over the years, and its name was changed in June of 2010, with the introduction of iOS 4 and the iPhone 4 [96]. Over the years, the App Store

\(^1\)A collection of frameworks (libraries, headers, and resources) that represent the API for a specific operating system version [95].
has been proving to be the right decision, by currently having 2.2 million applications available
[98] and generating billions in profits for both Apple and developers [94].

In the following years, with each new version of iOS, many features were introduced, with
much or less success, such as iMessage (texting over data), Siri (Apple’s voice-activated virtual
assistant [94]), Apple Maps and Passbook (a virtual wallet, later renamed to Wallet). With iOS
7, and the release of the iPhone 5S and iPhone 5C, Apple unveiled a totally new UI design, that
promoted a new flatter aesthetic, in detriment of an skeuomorphic one [94].

Since iOS 9, Apple started to devote its developing resources towards improving the over-
all stability of the operating system and adding smaller, but useful, features, to improve overall
usability [94].

With the newly announced iOS 11, scheduled to be released in the Fall 2017, Apple will
possibly make some of the biggest enhancements to iOS, both related to significant new features
(including all new augmented reality and machine learning frameworks) and overall minor refine-
ments [99].

3.3 iOS Development

The most important prerequisite needed to develop a native iOS application is Xcode, presented
in Section 3.3.2, that, together with a basic understanding about the related technologies (Section
3.3.1) and the tools it contains, can help make better choices about how to design and implement
iOS applications [92].

3.3.1 Technologies

Apple delivers most of its system’s interfaces in special packages called frameworks. A frame-
work "is a directory that contains a dynamic shared library and the resources (such as header files,
images, and helper apps) needed to support that library" [92]. To use frameworks, they have to be
added to the application’s project on Xcode. The architecture of iOS contains those frameworks,
which can be viewed as a set of layers, as shown in Figure 3.1 [92]:

- **Cocoa Touch** - contains fundamental frameworks, most importantly the ones that define the
  appearance of the application [100];

- **Media** - contains the graphics, audio and video technologies used in the application [101];

- **Core Services** - contains fundamental system services of the application [102];
iOS Overview

- **Core OS** - contains the low-level features that most higher-level technologies are built upon [103].

![iOS Architecture Layers](image)

Figure 3.1: iOS architecture layers (Adapted from [92])

The lower layers contain fundamental services and technologies; higher-level layers provide more refined technologies and services [92].

The initial versions of iOS were designed to support binary files on devices using a 32-bit architecture. In iOS 7, support was introduced for 64-bit architecture and the release of iOS 11 will set that the 32-bit architecture will no longer be supported. This was pushed by Apple because applications may run faster when compiled for the 64-bit runtime, because of the availability of extra processor resources in 64-bit mode [103].

### 3.3.2 Xcode

**Xcode** is a complete developer toolset used to create applications for iOS [95], and includes the SDK that enables the creation of applications that run on specific versions of iOS [104].

**Xcode** packages the iOS applications as bundles [105]. A typical application bundle has different types of files: an `Info.plist` file, an executable file, resource and other support files [106].

The basic utilities of Xcode are the **Interface Builder**, the **Debugger** and the **Documentation**. The Interface Builder (Figure 3.2) offers a graphical environment for building the UI of the application [107], the Debugger (Figure 3.3) aids the process of finding and eliminating problems in the applications’ code [108] and the Documentation provides most of the information needed for the development [109].

As part of the Xcode toolset, there are another two main tools: **Simulator** and **Instruments** [110][111].

---

2 A bundle is a directory in the file system that groups related resources together in one place [105].

3 “A structured file that contains configuration information for the application” [106].
The Simulator tool (Figure 3.4) provides a simulation of iOS, from the most recent version to the legacy versions, for testing a build of an application without needing an actual physical device, but not as reliable [110].
The *Instruments* tool is designed to help developers, by providing testing and performance-analysis of an application [111].

For testing purposes, there are also a test navigator designed to facilitate the process of creation, management, running and reviewing of code tests [114].

### 3.3.3 Programming Languages

There are two choices of native programming languages for iOS development: Objective-C and Swift [115]. Objective-C is an older language, designed in the early 1980s, while the creation of Swift only began in 2010. Objective-C became the foundation of Apple products prior to 1996, while the first version of Swift was released in 2014 [115].

Swift and Objective-C projects are nearly identical. The main difference between them is that Swift has no header files. In terms of interoperability, it is possible to use Objective-C APIs in Swift and vice-versa, which is also an advantage. However, unlike Objective-C or C, it is not possible to import C++ code directly into Swift; it is necessary to create an Objective-C or C wrapper for the C++ code [116].

The fact that Swift is a "still evolving" language, contrary to Objective-C, means that every language update requires updating the application’s code to the new version [115]. However, the frequent updates add to the belief that Apple and the developer community are committed to Swift [115]. Another difference between Swift and Objective-C is that the former only supports operating systems prior to iOS 7 [115], but that is likely not a worrisome problem, since only 3% of devices currently use a lower version than iOS 9 [117].

Swift is stated as a powerful and modern programming language that is also easy to learn, and it is designed for safety, performance and software design patterns [118] [119]. Its purpose is to combine the best aspects of modern languages with the wisdom from its open-source community and Apple expertise [120]. At the end of 2015, the Swift language and all its supporting components were published as Open Source[^4] [119].

Apple has made many efforts to promote the learning of the new language by creating helpful guides, documentation and learning tools like the *playgrounds*[^5], therefore, Swift seems to be the best bet for the future of iOS development.

[^4]: For reference, there is a complete explanation of the definition of Open Source in [121].

[^5]: Small programs that can instantly show the results of its written code [122].
3.4 iOS Distribution

The distribution of applications in the App Store requires a membership on the Apple Developer Program. Besides the main benefit to distribute applications through the App Store, the Apple Developer Program gives access to exclusive software and tools, such as operating systems’ beta releases, advanced app capabilities, testing and support [123]. Enrolling in the Apple Developer Program requires an Apple ID and an annual fee. Developers can enroll as an individual or as an organization [124].

The App Store makes the process of distributing an application easier, since it provides a focused channel for users to discover, purchase and download applications. Moreover, it handles worldwide payment processing and does not charge hosting fees, giving developers at least 70% of sales revenue [123]. With the membership, it is even possible to beta test the applications with a limited number of users before rolling it out to the complete App Store’s users [123]. For the process of submission and management of the applications for sale, the Apple Developer Program provides access to a suite of web-based tools, called iTunes Connect [125].

The applications submitted in iTunes Connect are all reviewed, to make sure that they comply with a set of rules and guidelines, described in App Review [126], and require approval before being put on sale on the App Store [125].

3.5 Conclusions

iOS and its development process have evolved over the years, with new operating system features and developer tools improvements. Before iOS, there was no defined mindset of "Everyone Can Code" [127], but today this philosophy seems to be increasingly more popular and accepted, which means that developing tools and programming languages are becoming simpler, but without compromising power. Moreover, the easy access to application’s distribution its seemingly proving to be a good means of creating new jobs for single developers and small mobile development companies.
Chapter 4

MasterVoicing Application

4.1 Introduction and Overview

This chapter documents all of the steps and information related with the creation of MasterVoicing. It contains four main sections, that follow the current one, which provide a better understanding of the process: Implementation (Section 4.2), Algorithm (Section 4.3), Final Product (Section 4.4) and Evaluation (Section 4.5).

The first section includes all the information related to the learning process, initial tests, throw-away prototype and actual development made in the implementation of MasterVoicing. It takes into account all of the chosen paths for the development and design of the application, including its reasons, and also the experienced problems and how they were tackled and solved (or unresolved).

The next two sections include an explanation and presentation of all the features of the final product of this dissertation, the iOS application developed - MasterVoicing - and a brief explanation of the encapsulated algorithm.

The last section includes all the tests performed for the evaluation of the application.

4.2 Implementation

The current chapter, related to the implementation of MasterVoicing, can be divided in three main parts, not explicitly shown in the organization of the following sections. The first part includes Section 4.2.1 and Section 4.2.2, which provide a better understanding of the software development process. The second part includes four sections, where all of the initial learning process that resulted in a working prototype is documented: Section 4.2.3, Section 4.2.4, Section 4.2.5 and Section 4.2.6. Finally, the last part includes the last section, Section 4.2.7, that covers all the steps of development of the final product (portrayed in Section 4.4).
4.2.1 Software Development Process

Software development, especially at the professional level, can be complex and have many challenges [128], which is one of the reasons why software development processes were created and developed.

A software development process is an outline for the activities necessary for software production [129] [130]. The adoption of a software development process can have many advantages, related with improvements in productivity, and the reduction of costs related with production, future maintenance and improvements of the resulting product [129] [130].

There are four fundamental activities related with the process of developing software [130]:

- Specification: definition of the wanted capabilities and functionalities of the software;
- Design and Implementation: developing of the software based on the specification;
- Validation: testing to confirm that everything works as the specification;
- Evolution: addition of new features and maintenance of current ones.

These steps were surely necessary for the creation of MasterVoicing, and are documented in the following sections of this chapter, whose implicit relationships are suggested in Table 4.1.

Table 4.1: Relationship between sections and software development process activities

<table>
<thead>
<tr>
<th>Activities</th>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>– 4.2.2 Software Development Methodologies</td>
</tr>
<tr>
<td></td>
<td>– 4.2.3 Initial Study</td>
</tr>
<tr>
<td></td>
<td>– 4.2.4 Initial Requirements Analysis</td>
</tr>
<tr>
<td>Design and Implementation</td>
<td>– 4.2.5 Initial Tests</td>
</tr>
<tr>
<td></td>
<td>– 4.2.6 Throwaway Prototype</td>
</tr>
<tr>
<td></td>
<td>– 4.2.7 Application Development</td>
</tr>
<tr>
<td>Validation</td>
<td>– 4.5 Evaluation</td>
</tr>
<tr>
<td>Evolution</td>
<td>– 5.2 Future Work</td>
</tr>
</tbody>
</table>

Every software development process includes these four activities in some way, even if they use different approaches to each one, and is usually labeled as one of two types: "plan-driven" (traditional approach) or "agile" (modern approach) [130].

30
4.2.2 Software Development Methodologies and Framework

A software development methodology is a particular approach to a given software development process, that comprises a specific and well defined set of tasks and techniques for the management and completion of a software product. There are several methodologies with different characteristics, according to how they handle the different stages of the software development process.

Agile based methodologies and frameworks were developed due to dissatisfaction with the heavyweight approach of the plan-driven ones [130]. This traditional methods resulted in more time being spent in the specification activity (Table 4.1) than all of the following ones [130]. This was considered by many a sluggish approach, and a waste of precious time for development, hence the name given to the modern approaches, which intended to fasten the whole software development process, was "agile" [130].

According to a recent global survey [131], Agile development is currently very popular, and several companies, from different industries, are actively using it or trying to adopt it. For this reason, its type of methodologies seemed to be the most recommended one, and was the chosen for the development of the application. However, as a way to better understand the possible capabilities desired in the application, it was thought it was a better approach to beginning with a software prototype, before the actual final development.

Sections 4.2.2.1 and 4.2.2.2 contain a brief explanation of the software development methodology and framework related with the implementation of MasterVoicing, the former for being used as a first approach and the latter for serving as a basis for the final product’s development.

4.2.2.1 Software Prototyping

Software prototyping consists in the creation of an incomplete demonstration of the desired final product.

Prototypes are often used in development, and can be useful, both low-fidelity and high-fidelity\(^1\) ones. They are usually used for two main reasons and in different phases of the development process [130] [133]:

- In the specification, to better understand the requirements before the actual design and programming of the application and validate its implementation;

- In the design, to explore and define the UI design.

\(^1\)High-fidelity prototypes provide more interactivity, have more realistic visuals and include all the content that would appear in the final product, being as close as possible to its final design [132].
There are two main types of software prototypes [133] [134]:

- **Throwaway** - used to validate or define the requirements and discarded after that for the development of the final product;

- **Evolutionary** - based on building an actual functional product, with minimal functionality, so that can be later improved upon to result in the final product.

Prototypes can also be developed using two different approaches: vertical or horizontal. A horizontal prototype is used mainly for the understanding of the UI design, since it gives a broader view of the entire final product, without going in depth or detail about internal functionalities [133]. On the contrary, a vertical prototype is used to demonstrate a detailed elaboration of a specific functionality or set of functionalities [133]. Therefore, horizontal prototypes are better for getting an overview of the final product as a whole, whereas vertical prototypes are useful for getting in exact detail of the technical aspects of a given functionality [133].

The initial phase of the implementation of MasterVoicing was the development of a prototype of the application, that include its core features. The characteristics of this prototype are better explained in Section 4.2.6.

### 4.2.2.2 Scrum

*Scrum* is a framework for dealing and solving complex tasks, created with the purpose of delivering software products productively, in an adaptable manner [135]. It consists of a set of values, roles, events, artifacts and rules, better explained in Table 4.2 [135]. Each one of these components serves a particular purpose and they are necessary for the success of its usage and results [135]. While sometimes called a methodology, it does not have the detailed and formal structure of one, and has the purpose to ensure that all of the process is accounted for transparency, inspection and adaptation, with a set of values everyone needs to follow [135]. Based on a recent global survey [131], it can be said, with a high degree of confidence, that Scrum is currently the most widely used Agile implementation.

This framework is supposed to be lightweight, simple to understand, but is not necessarily easy to master, which is one of the reasons the role of Scrum Master exists [135]. The Scrum Master is the one responsible for making sure that the framework process is understood and put in practice properly by the whole team, with the application of all its components [135]. The Development Team, as the name implies, are the individuals responsible for creating the actual accountable product, which creation is managed by the vision of the Product Owner [135].
Table 4.2: Scrum components [135]

<table>
<thead>
<tr>
<th>Type of Components</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Values</strong></td>
<td>– Commitment</td>
</tr>
<tr>
<td></td>
<td>– Courage</td>
</tr>
<tr>
<td></td>
<td>– Focus</td>
</tr>
<tr>
<td></td>
<td>– Openness</td>
</tr>
<tr>
<td></td>
<td>– Respect</td>
</tr>
<tr>
<td><strong>Roles</strong></td>
<td>– Product Owner</td>
</tr>
<tr>
<td></td>
<td>– Scrum Master</td>
</tr>
<tr>
<td></td>
<td>– Development Team</td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td>– Sprint</td>
</tr>
<tr>
<td></td>
<td>– Sprint Planning</td>
</tr>
<tr>
<td></td>
<td>– Daily Scrum</td>
</tr>
<tr>
<td></td>
<td>– Sprint Review</td>
</tr>
<tr>
<td></td>
<td>– Sprint Retrospective</td>
</tr>
<tr>
<td><strong>Artifacts</strong></td>
<td>– Product Backlog</td>
</tr>
<tr>
<td></td>
<td>– Sprint Backlog</td>
</tr>
</tbody>
</table>

Finally, we have the events and artifacts. They are used with a set of rules and tactics that are usually adapted according to different constraints, such as the type of project or size of the team. The first steps in the process are filling the Product Backlog and planning a Sprint, in the initial Sprint Planning meeting. The Product Backlog holds everything that the product requires to be considered "done" and the Sprint Backlogs have the specific tasks chosen to be completed for the closing of the correspondent Sprint. A Sprint is a planned set of days, preferably no more than a month, in which the Development Team works in a potentially usable product, meant to be incremented with each consecutive Sprint. During a Sprint, there are daily fast meetings for the Development Team, called Daily Scrum, to coordinate activities and create a plan for the next day. After the conclusion of a Sprint, there are two meetings with different purposes: Sprint Review and Sprint Retrospective. In the Sprint Review, all the stakeholders of the project inspect the completed work on the Sprint and adapt the Product Backlog if necessary. In the Sprint Retrospective they focus the inspection on other aspects, like the team relationships, the process and tools used, and identify possible improvements and adaptations to make for the next Sprint. [135] [136]
4.2.2.3 Adopted Methodology

Since the creation of software development methodologies stem from the goal of better managing teams of professionals, that develop software on a professional level, they usually are not inherently design to be used by a single person. This is, for example, noticed in Scrum, with the existence of different roles in a team and daily meetings, explained in the previous section. For this reason, it was necessary to adopt a methodology for personal development, to be used for the development of the MasterVoicing application.

The software methodology used was created using an adapted version of Scrum, also loosely inspired by two other proposed methods of a software development method for single developers, which are based in Agile development methodologies: Cowboy [136] and Scrum Solo [137].

The general idea was to take some of the advised steps of these methodologies, choosing the ones that make the most sense for the development of the application, and design a personal adapted methodology, with all the advantages of the Agile philosophy, but only one team member. The resulting components of the methodology are shown on Table 4.3.

Table 4.3: Adapted methodology components

<table>
<thead>
<tr>
<th>Type of Components</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td>– Product Owner</td>
</tr>
<tr>
<td></td>
<td>– Single Developer</td>
</tr>
<tr>
<td>Events</td>
<td>– Initial Requirements Meeting</td>
</tr>
<tr>
<td></td>
<td>– Sprint</td>
</tr>
<tr>
<td></td>
<td>– Sprint Planning</td>
</tr>
<tr>
<td></td>
<td>– Sprint Review</td>
</tr>
<tr>
<td></td>
<td>– Sprint Retrospective</td>
</tr>
<tr>
<td>Artifacts</td>
<td>– Product Backlog</td>
</tr>
<tr>
<td></td>
<td>– Sprint Backlog</td>
</tr>
<tr>
<td></td>
<td>– Code</td>
</tr>
</tbody>
</table>

The main differences from this adapted methodology with the official Scrum methodology is that there was no Scrum Master or Daily Scrum meetings, since it was intended for personal development. However, there was a Product Owner – This works’ supervisor. The duration of the
Sprints were also thought to be less than the span of two weeks, to adapt to the development time constraints and distribute the work more evenly.

The Product Backlog was populated with User Stories\(^2\) and Epics\(^3\), based on the information gathered in the Initial Requirements Meeting (Section 4.2.4) and during the creation of the Throwaway Prototype (Section 4.2.6), that took place before the final application development (Section 4.2.7).

The Sprint Planning was made one day before the beginning of each Sprint, and focused on choosing Epics for each one by tackling the most important User Stories first, so that it would be possible to have a working product as soon as possible, and choosing the following Sprint duration. In order to clarify the importance of each User Story, they were attributed estimations, based on their apparent complexity, using a numeric value attribution with a relative valuation, where they were estimated relative to each other, after deciding the most and least value to the most and least complex ones, respectively.

The Sprint Review and Sprint Retrospective meetings took place one day after each Sprint, with the supervisor, and the following Sprints were planned to begin as soon as possible.

In the end of each Sprint, a Sprint Backlog and a Burndown Chart\(^4\) were available for the evaluation of the completed product (the tools and methods used for the gathering of this information are indicated and detailed in Section 4.2.7).

The Code artifact corresponds to the code of the initial Throwaway Prototype and the code of the MasterVoicing application. Their programming followed certain rules to facilitate the whole process and to produce code with the best quality possible, that were the following:

- All the files were organized and kept in a code repository (in a private project in GitHub);
- During programming, conventions for formatting, naming and commenting were followed, for better legibility and comprehension;
- Refactoring was done while developing, as an ongoing effort (some of this effort is noted in Section 4.2.7).

---

\(^2\)Well defined unit of work, also known as a task, usually described with the template "As a \(<\text{type of user}>\), I want to \(<\text{goal, objective}>\), so that I can \(<\text{benefit, value}>\)" [138].

\(^3\)Set of user stories with a similar theme [138].

\(^4\)In the context of agile software development, it is a chart that shows how quickly user stories are closed/finished during a Sprint [139].
4.2.3 Initial Study

This section contains a description of the actions conducted to begin the achievement of the third and fourth objective steps described in Section 1.3, reiterated and labeled here to facilitate the train of thought:

- **Step 1** - Familiarization and knowledge acquisition for the understanding of the iOS development environment and frameworks, with a special emphasis on the ones related to audio manipulation;

- **Step 2** - Familiarization and knowledge acquisition regarding the programming languages involved in the development of the application and its algorithm (Objective-C, Swift and C++).

For the initiation of **Step 1**, an initial research was conducted, for a better understanding of the architecture of iOS, the available audio frameworks and the whole process of development of mobile applications for iOS (all briefly described in Chapter 3). The *Apple Developer* website [93] is a great resource for this matter, since it provides detailed informative guides, about all the tools and steps necessary for the development and distribution of a mobile application for iOS, and several sample code projects [140].

Simultaneously, to start **Step 2**, it was also made a research about the available learning resources for Objective-C and Swift. The conclusions of this research were:

- A simple search in Google’s search website [141] for "objective-c programming language" and "swift programming language" shows that there are many more results for Objective-C (as it would be expected from an older language) and that the Swift’s top hits are much more recent than the ones about Objective-C (as it would be expected from a new language, that has become one of the most appreciated by developers [142]);

- It is easier to find updated tutorials and explained documentation for Swift - as opposed to tutorials about Objective-C that tend to be older and no longer supported for future changes and recent iOS versions [143] [144];

- The *Apple Developer* website promotes Swift and leaves Objective-C "in the background" - even though the front pages promote Swift, many of the available guides and most of the sample code provided are still in Objective-C;
MasterVoicing Application

• The Apple Developer website has a lot of resources to learn Swift, that include documentation, sample code, videos and even iTunes U\(^5\) courses \[147\];

• There are several online courses focused on iOS development \[148\] \[149\] \[150\] \[151\] \[152\] \[153\];

• There are dedicated blogs curated by developers about iOS programming in Swift \[154\] \[155\] \[156\];

• Despite being a relatively new language, there are already available various tools \[157\] and open source code (currently, a search for "language:Swift" in GitHub\(^6\) results in over 35,000 repository results \[159\]) written by the developer community.

Instead of choosing the approach of "reading all the documentation thoroughly first", it was preferred a quick read of the most important guides and frameworks for the scope of the development of the application and to start "learning by doing". Therefore, taking two specific online courses, from Udemy\(^7\) \[161\], seemed the most appropriate method to learn the fundamentals of iOS development. The courses were the following:

• "iOS 10 and Xcode 8 - Complete Swift 3 & Objective-C Course" \[162\];

• "The Complete iOS 10 & Swift 3 Developer Course" \[163\].

This two courses contributed to the continuation of Step 2, by providing a learning source about the basic syntax of the Swift and Objective-C languages, the process of designing the UI using the tools provided by Xcode and an insight about more advanced iOS features.

The final decision to make was choosing which IDE to use to initiate the learning process of Swift and Objective-C. Previous experience with Xcode, even thought only with C++ projects, from other course units, made easier the decision to use it. It is also the one recommended by Apple for the development to all of its devices, includes a set of useful development tools (summarized in Chapter 3) and facilitates the distribution process. Besides that, the chosen Udemy courses encourage its use and have sections explaining its basic functionalities.

After this initial phase, it was made a simple requirements analysis, explained in the following section, Section 4.2.4, and tests that would lead to the development of a prototype of the application (Section 4.2.5).

\(^5\)iTunes U is a dedicated section of Apple’s iTunes with several educational content from different sources, including well known learning institutions \[145\] \[146\].
\(^6\)An online development platform for code sharing and publishing, based on Git, with many added features \[158\].
\(^7\)An online learning marketplace that provides courses taught by expert instructors \[160\].
It is important to note that all of the previous steps were just a means to start the learning process related to Step 1 and Step 2, even thought this process was continuous, and continued until the final objectives’ steps, with the development and distribution of MasterVoicing.

4.2.4 Initial Requirements Analysis

In every project of software development, it is important to make some kind of requirements analysis [130], that is usually done in the initial stages of the conception of the problem. This process, depending on the chosen development methodology, can be somewhat redone and the requirements may change or evolve over time.

After the initial study of the problem and some insight in the capabilities of iOS devices and its possible features, there was a general image of the fundamental behavior that the application could have. So, after that first step, an Initial Requirements Meeting took place, where the following basic functionalities were determined:

- The user should be able to use the application by whispering into the devices’s microphone; after this, the device must output voiced speech processed from the inputted whispered voice;
- The user should be able to control whether or not the application is capturing his voice;
- The ideal situation is the one where there is no need for the use of headphones;
- The application should work in a "real-time" like manner; that is, the time interval between the point where the user speaks and the device outputs the processed result should be as small as possible.

These functionalities were studied in the initial tests (Section 4.2.5) and with the creation of the Throwaway Prototype (Section 4.2.6), and were later detailed, improved and reframed as User Stories, shown in Section 4.2.7, for the development of the application.

4.2.5 Initial Tests

The initial tests were based on different sample code projects, some sourced directly from the Apple Developer website and others from open source projects found on GitHub.

One of the projects was used to understand how to integrate C++ code into the application (for the encapsulation of the algorithm); the other ones were used to understand the audio capabilities and frameworks that could be used to deal with audio manipulation in real-time.
MasterVoicing Application

There are currently two main audio frameworks that can be used for audio programming in iOS, as seen in Figure 4.1: AVFoundation and AudioToolbox, that contain AVAudioEngine and AUAudioUnit (they provide different approaches to audio manipulation).

![iOS audio stack](image)

Figure 4.1: iOS audio stack (Adapted from [164])

The first focus was on the study of the audio related projects, which can be divided in terms of programming language (Swift or Objective-C) and by the audio framework they use for audio input and output manipulation, as seen in Table 4.4.

<table>
<thead>
<tr>
<th>Project</th>
<th>Framework</th>
<th>Language</th>
<th>Capabilities</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>aurioTouch</td>
<td>AudioToolbox</td>
<td>Objective-C</td>
<td>- Processes the audio input and output in real-time</td>
<td>- It is not intended to work with the built-in speaker</td>
</tr>
<tr>
<td>aurioTouch2.0-Swift</td>
<td></td>
<td>Swift</td>
<td>- Allows simultaneous recording and playback</td>
<td>- Only performs the modification of audio visually</td>
</tr>
<tr>
<td>AVAEMixerSample</td>
<td>AVFoundation</td>
<td>Objective-C</td>
<td>- Processes the audio input and output in real-time</td>
<td>- Does not allow simultaneous recording and playback</td>
</tr>
<tr>
<td>AVAEMixerSample-Swift</td>
<td></td>
<td>Swift</td>
<td>- The processed audio is routed to the output</td>
<td>- Does not perform audio visualization</td>
</tr>
</tbody>
</table>
These first two projects use the AudioToolbox framework, and are called aurioTouch [165] and aurioTouch2.0-Swift [166]. As the name suggests, the second project is an open source project that is a translation into Swift of Apple’s aurioTouch sample code project (written in Objective-C). They are a demonstration of handling audio input and output using an Audio Unit\(^8\) (particularly an AURemoteIO\(^9\)) for the visualization of a regular time domain waveform (as shown in Figure 4.2a) and its correspondent frequency domain waveform (by computing a Fast Fourier Transform (FFT) of the input).

The other two projects, AVEAMixerSample [169] and AVEAMixerSample-Swift [170] (also a translated version into Swift of the original Apple’s sample code project), offer a higher-level approach by using the AVFoundation framework. They demonstrate playback, recording and mixing using AVAudioEngine\(^{10}\) (composed by different types of nodes with different capabilities, as shown in Figure 4.2b).

The study of the capabilities and limitations of these sample applications, stated in Table 4.4, were decisive for the choices made for the development of the Throwaway Prototype, explained in Section 4.2.6. Furthermore, to see if there were any advantages in terms of audio performance

---

\(^8\) Audio manipulation and processing modules that can be used both inside an application or created as an extension to be used in a host app [167].

\(^9\) An AudioUnit designed to handle simultaneous hardware input and output [168].

\(^{10}\) Class that defines a set of connected audio nodes, that can be used to generate, process or mix audio [171].
MasterVoicing Application

between the usage of Swift or Objective-C, measures were made to the latencies of the audio session in each of the projects, and it was concluded that they were the same in both languages.

With this first sample projects, it was possible to test multiple features that allowed some room to understand which programming language and audio framework to use in the prototype, and also served as a basis for its development.

Finally, the sample project used for learn how to send information between Swift and Objective-C was: "SO-32541268 - Mix Swift, Objective-C, C and C++ files in the same Xcode project" [172]. This project provides simple "Hello World" methods that give an example of how to communicate between Swift, Objective-C, C and C++ files. With the basis of its code, it was possible to make a test application with two functionalities (Figure 4.3):

- A basic sum and multiplication calculator, that sends two inputted numbers from Swift to a C++ function, where it processes the calculations and returns them to Swift, to being showed in the UI;

- A more complex "Change Numbers" option, that involves the processing, in a C++ function, of an array of floats sent from a Swift file, returning to it with the processed numbers, and printing them into the console.

![Figure 4.3: Test application adapted from SO-32541268 [172] - User interface](image)

The creation of this adapted test application involved the understanding of Objective-C wrapper files (files needed to connect Swift and Objective-C methods) and bridging header files (files needed to connect Objective-C and C++ methods).
MasterVoicing Application

After the confirmation that both of the wanted functionalities - audio related and algorithm encapsulation - were feasible by themselves, it was possible to move to the next step: the creation of a prototype that would confirm the possible capabilities of the application, by joining those two functionalities (Section 4.2.6).

In the end, the chosen applications to help with the creation of the prototype were the adapted version of SO-32541268 and AVAEMixerSample-Swift; all of the other sample code was not necessary and was, therefore, discarded.

As a final note, it is also worth to mention that it was initially considered as a possibility the use of an external open source framework, such as AudioKit [173], but due to the lack of solutions for the real-time processing required and deprecation concerns, this idea was early abandoned.

4.2.6 Throwaway Prototype

With the experience provided by the initial tests, a set of decisions were made to initiate the development of a prototype of the application. The first main decisions were choosing which programming language and audio framework to work with. Since there were no actual differences between Objective-C and Swift noticed in the initial tests, and the initial research showed a more favorable recent documentation support for Swift, the most recent language was given priority for the development. The choice of the audio framework was the most challenging one, because of the known limitations of AVFoundation regarding audio processing. Nonetheless, it was the chosen one because it was thought it could fulfill the programming of the required functionalities (as recommended by Apple, higher level frameworks should be preferred over lower-level frameworks whenever possible [92]). It was also decided that, for testing the audio processing, an open source pitch shifting algorithm would be used [174], for the recreation of the complexity of the final voice processing algorithm. The final decision was the development approach: to choose whether to focus the prototype in the audio functionalities or the UI and usability workflows. Developing the main audio functionalities without worrying about the UI seemed the most appropriate approach (vertical prototyping approach). Summarizing, the characteristics of the developed prototype in terms of decisions are:

- **Programming languages:** always Swift when it is possible; used Objective-C for the bridging header between Swift and the algorithm in C++;

- **Audio framework:** AVFoundation;
**MasterVoicing Application**

- **Audio processing**: pitch shifting algorithm, that shifts the voice pitch higher or lower according to a hard-coded given factor;

- **Development approach**: creating a proof of concept with all the audio related problems solved, from the control of the audio session to solving the problem of audio processing with a C++ algorithm.

The architecture logic of the audio engine, implemented in the prototype, is depicted in Figure 4.4. The audio is captured from the devices’ microphone by the **Input Node**, where it is installed a tap, to process the audio and to schedule it to play in the **Real Time Player**. The **Real Time Player** routes the audio through its connection with the **Main Mixer Node**, where a tap is also installed for recording the audio in a **File**. The **Recording Player** routes the recorded audio, contained by the **File**, when required, back to the **Main Mixer Node**. The **Main Mixer Node** is connected to the **Output Node**, that provides that the audio it receives is sent to the chosen output (built-in speaker, internal hear speaker, external output).

![Audio Engine Architecture Diagram](image)

**Figure 4.4: Throwaway Prototype - Audio engine architecture (Created with [175])**

The understanding of this architecture was time demanding and rather challenging, because **AVAudioEngine** is not usually used or firstly designed for the mix of real-time audio manipulation and processing.

The final features of this prototype, whose UI can be seen in Figure 4.5, were the following:

- Capturing audio in real-time from the device’s internal microphones;
- Processing and playback of the captured audio in real-time;
MasterVoicing Application

- Recording the processed audio in a temporary file for later playback;
- Playback of the recorded audio file;
- Changing recording and playback volumes and playback panning.

![MasterVoicing Application](image)

Figure 4.5: Throwaway Prototype - User interface

The main purpose of the creation of this prototype was to confirm the possibility of development of the desired features of MasterVoicing and to better understand the audio engine and audio session management, by using the sample code related with those functionalities as a template. However, at its development, several problems were encountered anyway, too specific and numerous to be accounted for in this document, mostly related with the correct audio manipulation, that were successfully overcomed.

4.2.7 Application Development

The application was developed using the previous described (Section 4.2.2.3) software development methodology. The development was divided in three Sprints, described in Sections 4.2.7.2, 4.2.7.3 and 4.2.7.4, whose themes were:

- **Sprint 1** - Main Features;
- **Sprint 2** - Advanced Features;
- **Sprint 3** - Final Improvements.
MasterVoicing Application

Before the development process started, from the 28th of June to the 14th of July, a set of steps were made for its preparation; these are described in Section 4.2.7.1.

A visual timeline of all the involved steps of the development can be seen in Table 4.5; these events were previously explained in Section 4.2.2.3.

Table 4.5: Development tasks - Timeline

<table>
<thead>
<tr>
<th>Months</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>Days</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preparation</td>
<td>28</td>
<td>15</td>
<td>16</td>
<td>26</td>
</tr>
<tr>
<td>Sprint 1 Planning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 1 Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 1 Retrospective</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 2 Planning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 2 Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 2 Retrospective</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 3 Planning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 3 Meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 3 Review</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2.7.1 Development Preparation

Before starting the development of the application, some steps needed to be addressed:

- Outline the adapted software development methodology to use;
- Choose the appropriate tools to put the methodology in practice;
- Prepare the necessary environment and install the necessary tools;
- Delineate the desired features and general appearance of the application.
MasterVoicing Application

This means that, firstly, it was necessary to make a thorough research about current development methodologies, in order to choose one and adapt it for the current work (Section 4.2.2).

Following this decision, it was made a research about the possible tools to use to help to put the methodology in practice; the chosen ones were GitHub and ZenHub [176]. ZenHub is directly connected with GitHub, it is free for single developer use and it is very easy to install, which made it the most appealing choice.

After creating a code repository in GitHub and installing ZenHub, the next step was to think about the desired features of the application and create User Stories to express them (enumerated in Tables 4.6, 4.7 and 4.8). At the same time, it was necessary to think about the appearance of the application, that is, the UI. For this purpose, some research was made, with a main focus in real applications’ UIs and the official Apple recommendations (principally the Apple "Human Interface Guidelines" for iOS [177]). With this information, some mockups of an UI were made, which final version can be seen in Figure 4.6. The mockups were made using Figma, a "collaborative interface design tool" [178], using UI elements from "iOS Design Kit" [179].

![Figure 4.6: MasterVoicing user interface mockups (Created with [178])](image)

The final steps of the preparation were to learn how to use ZenHub\(^1\), add the previously created User Stories to its interface, estimate the effort of each one with Story Points\(^2\), and plan and create Epics and Sprints.

---

\(^1\)For clarification purposes, in GitHub and ZenHub, Sprints, User Stories and Story Points have different designations, respectively: Milestones, Issues and Estimates [180].

\(^2\)The Story Points were estimated using the default ZenHub’s estimation method - a set of fibonacci numbers, with an extra maximum value of 40: 1, 2, 3, 5, 8, 13, 21 and 40.
4.2.7.2  Sprint 1

For the first Sprint, Epics were chosen with the purpose to create an initially basic, but working, version of the application. This Epics and its User Stories are detailed in Table 4.6, which also contains the completion date of each one, which evaluation, regarding the correspondent story points, result in a burndown chart, depicted in Figure 4.7.

![Burndown report](image)

Figure 4.7: Sprint 1 - Burndown chart (Adapted from ZenHub [176])

It was also decided that 10 days would be an ideal duration for the first Sprint (the total estimation of its Story Points were 104 points, which resulted in a reasonable average of 10 points per day), and the next ones would be adapted from its completion velocity.

The most challenging features, noted at the Sprint Review, were, as expected, the ones of the "Basic User Interface" Epic. There was also a bug at the end of the Sprint, while recording audio, where sometimes the application would crash after retrying to record, only with the voice processing turned on, that was resolved in Sprint 2, by rearranging the audio engine architecture and logic. After this first Sprint, the audio engine architecture was equivalent at the Throwaway Prototype one (Figure 4.4), since it was the basis for many of the features of this Sprint. Another thing noted in the Sprint Review was that, by looking at the burndown chart, it was clear that the first two big user stories should have been more specific to begin with, which was why it was decided, in the Sprint Retrospective, that the next Sprint’s ones should be reviewed and made smaller or more detailed, which is noticeable in its burndown chart, in Figure 4.9.
# MasterVoicing Application

Table 4.6: Sprint 1 - User stories

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>SP</th>
<th>Description</th>
<th>CD</th>
<th>Epic</th>
</tr>
</thead>
<tbody>
<tr>
<td>US01</td>
<td>Basic UI Design</td>
<td>40</td>
<td>As a user, I want to have visual and easy to use controls, so that I can interact with the application and understand what I can do in terms of features.</td>
<td>Jul 23</td>
<td>Basic User Interface</td>
</tr>
<tr>
<td>US02</td>
<td>Basic UI Structure</td>
<td>40</td>
<td>As a user, I want that the application works logically in the expected way and responds to the visual commands, so that I can use it without problems.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US03</td>
<td>Record Voice</td>
<td>5</td>
<td>As a user, I want to record my voice, so that I can use it with other features of the application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US04</td>
<td>Playback Voice</td>
<td>5</td>
<td>As a user, I want to playback a processed version of my recorded voice, so that I can be heard and understood.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US05</td>
<td>Change Volume</td>
<td>1</td>
<td>As a user, I want to change the volume of the recording, so that I can adapt it to my needs.</td>
<td></td>
<td>Basic Audio Processing</td>
</tr>
<tr>
<td>US06</td>
<td>Playback Output: built-in speakers</td>
<td>2</td>
<td>As a user, I want the playback to work with the built-in speakers of my device, so that I can use it everywhere without extra accessories.</td>
<td>Jul 25</td>
<td></td>
</tr>
<tr>
<td>US07</td>
<td>Audio Processing: algorithm adaptation and encapsulation</td>
<td>3</td>
<td>As a user, I want that the application uses an algorithm for audio processing, so that I can hear my voice being altered in real time or after recording it.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US08</td>
<td>Playback Output: headphones</td>
<td>5</td>
<td>As a user, I want the playback to work with headphones, so that I have more privacy while using the application.</td>
<td></td>
<td>More Playback Options</td>
</tr>
<tr>
<td>US09</td>
<td>Playback Output: external speakers</td>
<td>3</td>
<td>As a user, I want the playback to work with external speakers, so that I can use better quality accessories for audio output.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SP** - Story Points  **CD** - Completion Date
**MasterVoicing Application**

The first UI implementation was inspired by the design of the previously shown mockups, and the final version of the UI after *Sprint 1* can be seen in Figure 4.8. The reason why its basic implementation took so long, even though there was a simple way of using the native UI elements of iOS, was that it was decided to custom-draw or change all the UI elements, and create custom-made animations to make them more appealing. This is more detailed in Section 4.4.

Figure 4.8: *Sprint 1* - User interface

### 4.2.7.3 *Sprint 2*

The second *Sprint* focused on the improvement of the current features and on the addition of advanced features, described in Table 4.7.

By evaluating the burndown report of *Sprint 1*, in the *Sprint 1 Review*, it was possible to plan the duration of *Sprint 2*. According to its estimated 129 *Story Points*, at the same velocity as *Sprint 1*, *Sprint 2* was planned to have the duration of 12 days.

The burndown chart of this *Sprint* (Figure 4.9) shows that the choice, made in the previous *Sprint Retrospective*, of making the *User Stories* more specific and detailed, resulted in a more smooth development.

At the end of *Sprint 2*, a new bug remained to be solved, related with the handling and changing the audio output. The bug produced an unexpected behavior in the application when the output changed while processing the audio, that is, in real-time mode, while recording or while playing the previous recorded audio. This bug was handled in *Sprint 3*, with *User Story US27*. 
Table 4.7: Sprint 2 - User stories

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>SP</th>
<th>Description</th>
<th>CD</th>
<th>Epic</th>
</tr>
</thead>
<tbody>
<tr>
<td>US10</td>
<td>Pause Playing</td>
<td>13</td>
<td>As a user, I want to pause the playback of a recording, so that I can control the playback while it is not finished.</td>
<td>Jul 31</td>
<td></td>
</tr>
<tr>
<td>US11</td>
<td>Basic Settings Structure and UI</td>
<td>13</td>
<td>As a user, I want to have a list of settings, so that I have access to more options.</td>
<td>Aug 6</td>
<td>Basic Settings and Audio Processing</td>
</tr>
<tr>
<td>US12</td>
<td>Settings: max. recording duration</td>
<td>13</td>
<td>As a user, I want to change the maximum recording duration, so that I have better control on the internal audio files size.</td>
<td>Aug 7</td>
<td></td>
</tr>
<tr>
<td>US13</td>
<td>Settings: activate/deactivate voice processing</td>
<td>5</td>
<td>As a user, I want to be able to activate/deactivate the processing of my voice, so that I can choose how to record it.</td>
<td>Jul 30</td>
<td></td>
</tr>
<tr>
<td>US14</td>
<td>Show Time: recording</td>
<td>5</td>
<td>As a user, I want to see the audio recording time, so that I know the remaining and elapsed time.</td>
<td>Jul 31</td>
<td></td>
</tr>
<tr>
<td>US15</td>
<td>Show Time: recorded file playback</td>
<td>5</td>
<td>As a user, I want to see the playback time of the audio file that is playing, so that I have a visual feedback of the playback.</td>
<td>Aug 6</td>
<td>Extra UI Features</td>
</tr>
<tr>
<td>US16</td>
<td>Audio Visualization</td>
<td>40</td>
<td>As a user, I want to have some visual feedback of the audio, so that I know if it is being recorded or played.</td>
<td>Aug 2</td>
<td></td>
</tr>
<tr>
<td>US17</td>
<td>UI: visual improvements and feedback</td>
<td>13</td>
<td>As a user, I want that the visual interface of the application is pleasant to the eye and provide visual feedback of its state, so that its usage is more enjoyable and useful.</td>
<td>Aug 9</td>
<td></td>
</tr>
<tr>
<td>US18</td>
<td>Settings: change processing mode</td>
<td>1</td>
<td>As a user, I want to switch the feature of changing the processing mode with motion, so that I don’t use it unintentionally.</td>
<td>Jul 30</td>
<td>Extra Settings</td>
</tr>
<tr>
<td>US19</td>
<td>Settings: choose color theme</td>
<td>8</td>
<td>As a user, I want to change the color theme, so that I can personalize it and adapt it to current light conditions.</td>
<td>Aug 6</td>
<td></td>
</tr>
<tr>
<td>US20</td>
<td>Settings: change language</td>
<td>13</td>
<td>As a user, I want to change to a language more familiar to me, so that I better understand the features.</td>
<td>Aug 9</td>
<td></td>
</tr>
</tbody>
</table>

SP - Story Points  
CD - Completion Date
The main challenge of Sprint 2 was the implementation of the audio visualization. After a general research about pre-made available solutions for audio visualization, it was concluded that none of them would be adaptable to an audio engine with the same configuration of the one implemented in Sprint 1 and the particular aspect of using the scheduling of audio buffers to a player node (better explained in Section 4.4). Therefore, it was decided to create a custom audio visualization, which meant that the configuration of the audio engine needed to be altered. This new configuration, that adds two mixer nodes for the visualization is shown in Figure 4.10.

Figure 4.9: Sprint 2 - Burndown chart (Adapted from ZenHub [176])

Figure 4.10: Sprint 2 - Audio engine architecture - After implementing the audio visualization (Created with [175])
However, to solve the previous bug in recording, it was decided to reconfigure the audio engine architecture, which resulted in what can be seen in Figure 4.11.

The implementation of the audio visualization requires the installation of a "tap" in a node of the audio engine (typically a mixer node) that receives the audio buffers from the connected nodes. The first architecture, used in the prototype, wasn’t thought to include audio visualization, so, it only included the Main Mixer Node, whose tap was already being used for audio recording purposes (it is not possible to install two taps simultaneously in the same node, and it is not viable to capture, process, record and use the same buffer for visualization, as it may cause audible breaks in the audio playback). The new configuration allows to install a tap in each one of the added mixers (Real Time Mixer Node and Recording Mixer Node), and to use them when necessary.

One suggested and thought out feature which development was started but abandoned, because of time constraints, as decided in the Sprint 2 Retrospective, was the option of sharing a recorded audio file (with the added possibility of changing the name of the file before sharing), which UI components can be seen in Figure 4.12.
MasterVoicing Application

Also in the Sprint 2 Retrospective, it was suggested by the Product Owner that a new UI element was added, as a new feature for the final Sprint 3. This element would be a control for a given changeable characteristic of the algorithm; as for the pitch shifting algorithm, this means controlling the pitch factor (higher or lower than the real voice pitch).

Many of the implemented features in Sprint 2 translated in many improvements in UI design and visual feedback, being some of them depicted in Figures 4.13 and 4.14.

![Figure 4.13: Sprint 2 - User interface - Implemented features](image)

(a) Recording mode UI  
(b) Real time mode UI  
(c) Settings UI

![Figure 4.14: Sprint 2 - User interface - Visual feedback](image)

(a) Processing mode changes alert  
(b) Confirmation alert  
(c) Unsupported output alert
MasterVoicing Application

After a last refactoring of the code, it was possible to simplify even more the audio engine architecture and eliminate the two specifically created audio mixer nodes for visualization, as shown in Figure 4.15. This was possible by using a single tap in the Main Mixer Node, that was no longer tapped, thanks to the new audio configuration for audio recording to a File.

![Figure 4.15: Sprint 2 - Audio engine architecture - Final version (Created with [175])](image)

4.2.7.4 Sprint 3

The third and final Sprint consisted in the development of final improvements, described by the User Stories in Table 4.8, whose burndown chart is shown in Figure 4.16.

![Figure 4.16: Sprint 3 - Burndown chart (Adapted from ZenHub [176])](image)
Table 4.8: Sprint 3 - User stories

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>SP</th>
<th>Description</th>
<th>CD</th>
<th>Epic</th>
</tr>
</thead>
<tbody>
<tr>
<td>US21</td>
<td>Algorithm’s Property Alteration Slider</td>
<td>13</td>
<td>As a user, I want to have an UI element, such as a control slider, so that I can easily alter, in real time, a property of the algorithm.</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>US22</td>
<td>More Color Themes</td>
<td>3</td>
<td>As a user, I want to have a set of different color themes to chose, so that I can have a more personalized user experience.</td>
<td>Aug 29</td>
<td></td>
</tr>
<tr>
<td>US23</td>
<td>UI Element: &quot;Done&quot; Button</td>
<td>3</td>
<td>As a user, I want to have a clearly identifiable &quot;Done&quot; button, so that I can better understand its meaning.</td>
<td>Final UI Features</td>
<td></td>
</tr>
<tr>
<td>US24</td>
<td>UI Element: &quot;Settings&quot; Button</td>
<td>3</td>
<td>As a user, I want to have a clearly identifiable &quot;Settings&quot; button, so that I can better understand its meaning.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US25</td>
<td>Help: view application’s instructions</td>
<td>13</td>
<td>As a user, I want the application to have a visual aid that explains its instructions, so that I can better learn how to use its different functionalities.</td>
<td>Aug 31</td>
<td></td>
</tr>
<tr>
<td>US26</td>
<td>Application Icon Design</td>
<td>5</td>
<td>As a user, I want the application to have a distinct and identifiable icon, so that I can clearly distinguish it from the other applications on my device.</td>
<td>Sep 2</td>
<td></td>
</tr>
<tr>
<td>US27</td>
<td>Support Changing Audio Routes</td>
<td>21</td>
<td>As a user, I want that the application supports changing to a different audio output, so that I can use the output of my choice.</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

SP - Story Points        CD - Completion Date

This Sprint had the duration of 6 days, calculated according to the effort estimation (61 Story Points) that was made in the Sprint 3 Planning, the day after the final definition and estimation of the remaining User Stories, on August 27th. During the Sprint, in a meeting with the Product Owner, on August 30th, it was decided that there would be no possibility of having a well enough developed whisper-to-speech algorithm, due to time constraints. Therefore, the final version of the application uses the previous pitch shifting algorithm used in the Throwaway Prototype, described in Section 4.2.6. Further information about this algorithm and its integration with the application is given in the next section, Section 4.3.
In Sprint 2, the User Story US25 was thought to be presented differently than what was implemented in Sprint 3. As previously shown in Figure 4.13c, the instructions of the application were meant to be seen at the launch of the application and later accessible to the user in the settings’ menu. In the final version of the application, this feature is presented in the main screen as a helping feature, by using an overlay with its instructions.

The most challenging parts of this Sprint were the implementation of the instructions and the support for handling audio route changes. The first part required creating a more complex management of the UI views and correspondent view controllers and the second part required a better understanding of the audio engine and session management.

The additions and improvements made to the UI in Sprint 3 resulted in the final version of the MasterVoicing UI, shown in Section 4.4.

In the Sprint 3 Review it was concluded that all the desired functionalities for having a good enough final product were achieved, since it would likely be suited for submission through the App Store, if the initially thought whisper-to-speech algorithm had been the one used.

Since this last Sprint concluded the implementation of the final version of the application for this work’s scope, there was not a Sprint 3 Retrospective after the Sprint 3 Review.

4.3 Algorithm

This chapter contains a simple explanation about the algorithm included in the MasterVoicing application and its implementation. The algorithm is the pitch shifting algorithm mentioned in Section 4.2.6, that was used as a proof-of-concept in the creation of the Throwaway Prototype and for test purposes during the development of the final product. This algorithm uses a STFT (Short Time Fourier Transform) "for changing the perceived pitch of an audio signal by representing it as a sum of sinusoids and scaling the frequency of these sinusoids" [181].

4.3.1 Implementation

The algorithm is implemented using a C++ class and an Objective-C wrapper class, that connects the C++ class to Swift. The whole process involves five different files, correspondent to three different classes: Algorithm.hpp, Algorithm.cpp, Algorithm-Wrapper.h, Algorithm-Wrapper.mm and AudioEngine.swift. In AudioEngine.swift, the installInputNodeTap method (Listing 4.1) calls the wrapping function of the Objective-C wrapper class (Listing 4.2), which is directly connected
with the C++ class that contains the algorithm. The C++ class (Listing 4.3) contains a set of functions that are responsible for the audio processing of an incoming buffer.

```swift
private func installInputNodeTap()
{
  if (!_isInputNodeTapped == false)
  {
    _engine.inputNode?.installTap(onBus: 0, bufferSize: self._numOfSamples,
                               format: self._playerFormat) {buffer, when
                                buffer.frameLength = AVAudioFrameCount(AudioProperties.sampleSize)

      if (self._algorithmOn == true)
      {
        // call the algorithm’s wrapping function
        Algorithm_Wrapper().pitchShiftAlgorithm_wrapped(buffer.
                             floatChannelData?.pointee, andPitch: self._algorithmProperty)
      }

      if (self._isRecording == true)
      {
        do
        {
          try self._mixerOutputFile.write(from: buffer)
        }
        catch let error {
          fatalError("error writing buffer data to file, \(error.localizedDescription)"")
        }
      }

      if (self._scheduleInputBuffers == true)
      {
        // schedule player to play the buffers in sequence
        self._player.scheduleBuffer(buffer, completionHandler: nil)
      }
  }
  _isInputNodeTapped = true
}
```

Listing 4.1: Swift function: `installInputNodeTap()`

```
-(void)pitchShiftAlgorithm_wrapped:(float *)buffer andPitch:(float)value
{
  Algorithm().pitchShiftAlgorithm(buffer, value);
}
```

Listing 4.2: Objective-C wrapper function to call the algorithm in C++

```cpp
class Algorithm {
public:
  Algorithm();
  ~Algorithm();
  void pitchShiftAlgorithm(float * buffer, float pitchValue);
  void smbPitchShift(float pitchShift, long numSampsToProcess, long fftFrameSize, 
                     long osamp, float sampleRate, float *inData, float *outData);
  void smbFft(float *fftBuffer, long fftFrameSize, long sign);
  double smbAtan2(double x, double y);
};
```

Listing 4.3: Algorithm class header code of `Algorithm.hpp` [174]
MasterVoicing Application

Finally, the function `pitchShiftAlgorithm` calls the `smbPitchShift` function (Listing 4.4), which is the basis function of the algorithm. They both receive a `pitchShift` factor value between 0.5 (one octave down) and 2 (one octave up), where 1 does not alter the pitch in any way [181]. This factor is controlled by the Pitch Slider in the UI of the application.

```
void Algorithm::pitchShiftAlgorithm(float * buffer, float pitchValue)
{
    smbPitchShift(pitchValue, BUFFER_SIZE, MAX_FRAME_LENGTH, 4, SAMPLE_RATE, buffer, buffer);
}
```

Listing 4.4: C++ function that calls the `smbPitchShift` function

Besides the pitch changing factor, there are other required parameters that have to be filled to process the audio [181]:

- `numSampsToProcess` - number of buffer samples;
- `fftFrameSize` - FFT frame size used for the audio processing;
- `osamp` - oversampling value;
- `sampleRate` - audio’s sample rate;
- `indata` - input buffer before processing;
- `outdata` - output buffer after processing.

In the final product, the values of `BUFFER_SIZE` and `MAX_FRAME_LENGTH` correspond to 512 and `SAMPLE_RATE` equals 22050.0 Hz, as these were the required values for the intended whisper-to-speech algorithm. The oversampling value was tested to be used at its maximum value (32), which would work without any problems, but it was left at the minimum possible value (4), because it provided an apparent slightly decrease in audio processing latency.

One of the main advantages of this algorithm is that the input audio buffer can be the same as the output buffer, which means the data can be processed in-place, without memory allocations, that usually provoke audio playback problems, such as silent breaks between different buffer’s playback. Further details about this algorithm are better explained in a related article [181] and can be better understood with its source code [174].

After this whole process, that is repeated for each audio buffer, the altered audio buffer is scheduled to play sequentially in a player (Listing 4.1), that redirects it to the main mixer node, so that the audio can be heard in real time through the output node.
4.4 Final Product

This chapter documents some details about the features, UI and the developed code structure of the MasterVoicing application - the final product of this work.

4.4.1 Features

The application’s features can be divided in two categories: Main Features and Extra/Advanced Features (Table 4.9). The former are the most important, because the application wouldn’t make sense without them, whereas the latter are just enhancements to the original desired functionalities.

Table 4.9: MasterVoicing’s features

<table>
<thead>
<tr>
<th>Category</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Features</td>
<td>• Two modes for voice recording:</td>
</tr>
<tr>
<td></td>
<td>– <em>Real Time</em> and <em>Recording</em>.</td>
</tr>
<tr>
<td></td>
<td>• Two audio processing modes:</td>
</tr>
<tr>
<td></td>
<td>– <em>Voice Processing</em> and <em>Real Voice</em>.</td>
</tr>
<tr>
<td></td>
<td>• The capability of choosing the desired pitch of the audio processing and the general volume of the audio output.</td>
</tr>
<tr>
<td>Extra/Advanced Features</td>
<td>• Choose the application’s language, between English or Portuguese.</td>
</tr>
<tr>
<td></td>
<td>• Choose the maximum recording duration, while in recording mode.</td>
</tr>
<tr>
<td></td>
<td>• Choose a color theme from a set of predefined color themes.</td>
</tr>
<tr>
<td></td>
<td>• Change the voice processing mode, by shaking the device, and having the capability of switching this functionality on or off.</td>
</tr>
<tr>
<td></td>
<td>• Instructions that aid the understanding of the application’s features.</td>
</tr>
</tbody>
</table>

The UI of these features are shown in the next section, Section 4.4.2, and the structure of its developed code are explained in Section 4.4.3.
4.4.2 User Interface

Throughout the design of the UI of the application, the main focus was on the simplicity of the UI elements and the ease of user interaction.

When the application is first launched, the user is presented with a launch screen, as shown in Figure 4.17a, before the initial screen of the application appears. At its initial state, the application is presented in the Recording mode (playback mode) and in the Voice Processing mode (audio processing mode), as seen in Figure 4.17b. The language used in the UI elements is the device’s current language and the used colors are the ones of the Default color theme.

![Launch screen UI](image)
![Recording mode UI](image)
![Real time mode UI](image)

Figure 4.17: MasterVoicing’s user interface - Main features

The user can change the playback mode to Real Time mode (Figure 4.17c), in the respective button. In Real Time mode, the application records, processes and plays back the audio without the need of any user assistance. On the contrary, in Recording mode (Figure 4.17b) the user has to go through the process of interacting with the UI for the recording and playback of the audio. In both modes, the user needs to interact with the UI for choosing the desired pitch and volume values or the audio processing mode. The user can switch the audio processing on or off by choosing, respectively, the Voice Processing or Real Voice mode buttons.

Since the Main Features of the application are related with the recording, processing and playback of voice, its UI was thought out to be designed around that purpose. For that reason, the Extra Features are not the first thing the user sees on the screen; each one of them has a button to access them, more specifically, the Help (Figure 4.18) and Settings (Figure 4.19) buttons.
The design of the *Instructions* was inspired by one of the usual methods for helping users to understand UI features: instructional overlays [182]. They are available in both of the supported languages and the user can navigate them by swiping left or right, or clicking the *Next* button. After the user is ready to go back to using the application, he/she has to click on the *Done* button.

In *Settings*, the user has access to the rest of the *Extra Features*, which are described in the previous section (Section 4.4.1). One of these features involves choosing the application’s color theme, from 13 different predefined color themes, whose colors and names were inspired by a
set of images gathered by searching the name of the themes in Google (with the exception of the Default theme, which was inspired by the resulting images of searching the terms "Pink and Blue Candy"). The colors were picked from the images using an online tool that helps generate color schemes - Coolors [183]. An overall view of those themes can be seen in Figure 4.20, ordered by its respective name, from left to right: Dark, Light, Purple Flower, Watermelon, Purple Skin, Beach, B&G Cocktail, Rain, Talking Parrot, Orange, Sunny Sky, Night Sky and Default.

![Figure 4.19: MasterVoicing’s user interface - Settings](image)

![Figure 4.20: MasterVoicing’s user interface - Color themes (Created with [184])](image)
MasterVoicing Application

Besides these features, the application contains a set of custom animations and alerts (implemented in Sprint 2 and previously shown in Figure 4.14) whose purpose is to give the user some feedback about changes and different states of the application (Tables 4.10 and 4.11).

Table 4.10: MasterVoicing’s user interface - Custom animations

<table>
<thead>
<tr>
<th>Animation</th>
<th>Trigger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recording button</td>
<td>– When the recording starts</td>
<td>– The recording button is animated with size and opacity changes, that last until the recording stops</td>
</tr>
<tr>
<td>Play to Stop button</td>
<td>– When the recording starts</td>
<td>– The play button is animated to be transformed into the stop button</td>
</tr>
<tr>
<td>Stop to Play button</td>
<td>– When the recording is stopped</td>
<td>– The stop button is animated to be transformed into the play button</td>
</tr>
<tr>
<td>Play to Pause button</td>
<td>– When the playback starts or is resumed</td>
<td>– The play button is animated to fade away and the pause button is animated to fade in on its place</td>
</tr>
<tr>
<td>Pause to Play button</td>
<td>– When the playback is paused or finished</td>
<td>– The pause button is animated to fade away and the play button is animated to fade in on its place</td>
</tr>
<tr>
<td>Play button emphasis</td>
<td>– When there is no audio recording to play</td>
<td>– The play button is animated with a &quot;shaking&quot; effect</td>
</tr>
<tr>
<td>Stop button emphasis</td>
<td>– When the record button is pressed after the recording is already in progress</td>
<td>– The size of the stop button is momentarily enlarged to remind the user that it is already recording</td>
</tr>
<tr>
<td>Audio modes buttons</td>
<td>– When the audio playback or audio processing changing modes buttons are pressed</td>
<td>– The buttons are animated with a reversible position displacement</td>
</tr>
<tr>
<td>Audio visualization</td>
<td>– When the audio is being recorded or played, both in recording mode and real time mode</td>
<td>– A representation of the audio being recorded or played is presented</td>
</tr>
</tbody>
</table>
### MasterVoicing Application

Table 4.11: *MasterVoicing*’s user interface - Alerts

<table>
<thead>
<tr>
<th>Alert</th>
<th>Trigger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Playback Mode Changes</strong></td>
<td>– When the user presses the audio playback mode changing buttons</td>
<td>– The alert appears momentarily in the bottom of the screen with the name of the currently chosen audio playback mode</td>
</tr>
<tr>
<td><strong>Processing Mode Changes</strong></td>
<td>– When the user presses the audio processing mode changing buttons</td>
<td>– The alert appears momentarily in the bottom of the screen with the name of the currently chosen audio processing mode</td>
</tr>
<tr>
<td><strong>New Recording</strong></td>
<td>– When the user attempts to start a new recording and a previous recording was already made</td>
<td>– The alert informs the user that a new recording will overwrite the current one, and asks to confirm the start of the new recording</td>
</tr>
<tr>
<td><strong>Unsupported Output</strong></td>
<td>– When the audio output is the built-in speaker of the device and the user changes to Real Time mode</td>
<td>– The alert informs the user that the current output is unsupported and informs which are the supported ones</td>
</tr>
</tbody>
</table>

To avoid problems regarding low image quality with the resizing of the images, none of the UI elements uses images for its representation. Instead, they are all custom made or adapted from the native UI elements available. The following UI elements of the application are all custom designed and drawn in the views by using a drawing method that uses bezier curves: Record button, Play button, Stop button, Pause button, Settings button, Done button. The other UI elements are all adapted to match the custom appearance of the UI. Other important aspect of the UI is the custom developed audio visualization, that is also a custom made drawing of the audio buffer values, adapted to the devices’ screen size, using the same drawing method as the custom-made UI elements. The final design aspect of the application is its Icon, seen in Figure 4.21.

![Figure 4.21: MasterVoicing’s icon (Created with [184])](image)
4.4.3 Code Structure

The following Tables 4.12 and 4.13 contain a description of all the Swift classes of the developed code of the application, and their associated views.

Table 4.12: MasterVoicing’s developed code - Classes (Swift)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Associated View</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppDelegate</td>
<td>– The application’s delegate</td>
<td>—</td>
</tr>
<tr>
<td>ViewController</td>
<td>– The application’s main view controller</td>
<td>Main View</td>
</tr>
<tr>
<td>SettingsViewController</td>
<td>– View controller class</td>
<td>Settings View</td>
</tr>
<tr>
<td>InstructionsViewController</td>
<td>– View controller class</td>
<td>Instructions View</td>
</tr>
<tr>
<td>InstructionsGeneralViewController</td>
<td>– View controller class</td>
<td>—</td>
</tr>
<tr>
<td>FirstInstructionsPageViewController</td>
<td>– View controller class</td>
<td>First Instructions Page View</td>
</tr>
<tr>
<td>SecondInstructionsPageViewController</td>
<td>– View controller class</td>
<td>Second Instructions Page View</td>
</tr>
<tr>
<td>ThirdInstructionsPageViewController</td>
<td>– View controller class</td>
<td>Third Instructions Page View</td>
</tr>
<tr>
<td>FourthInstructionsPageViewController</td>
<td>– View controller class</td>
<td>Fourth Instructions Page View</td>
</tr>
<tr>
<td>FifthInstructionsPageViewController</td>
<td>– View controller class</td>
<td>Fifth Instructions Page View</td>
</tr>
<tr>
<td>SixthInstructionsPageViewController</td>
<td>– View controller class</td>
<td>Sixth Instructions Page View</td>
</tr>
<tr>
<td>AudioEngine</td>
<td>– Class that manages all the audio features, audio session, logic and components</td>
<td>—</td>
</tr>
<tr>
<td>GeneralLocalizer</td>
<td>– Class that manages the language of the application based on the localization settings</td>
<td>—</td>
</tr>
<tr>
<td>CurrentLanguageLocalizer</td>
<td>– Class that manages the current language on the scope of the application</td>
<td>—</td>
</tr>
</tbody>
</table>
### Table 4.13: MasterVoicing’s developed code - Views

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main View</strong></td>
<td>– Initial and primary view, that presents the contents related with the main features</td>
</tr>
<tr>
<td><strong>Settings View</strong></td>
<td>– Modal view that presents the contents of the settings’ menu</td>
</tr>
<tr>
<td><strong>Instructions View</strong></td>
<td>– View that serves as a container for the other view pages of the instructions</td>
</tr>
<tr>
<td><strong>First Instructions Page View</strong></td>
<td>– View that presents the content of the first page of the instructions</td>
</tr>
<tr>
<td><strong>Second Instructions Page View</strong></td>
<td>– View that presents the content of the second page of the instructions</td>
</tr>
<tr>
<td><strong>Third Instructions Page View</strong></td>
<td>– View that presents the content of the third page of the instructions</td>
</tr>
<tr>
<td><strong>Fourth Instructions Page View</strong></td>
<td>– View that presents the content of the fourth page of the instructions</td>
</tr>
<tr>
<td><strong>Fifth Instructions Page View</strong></td>
<td>– View that presents the content of the fifth page of the instructions</td>
</tr>
<tr>
<td><strong>Sixth Instructions Page View</strong></td>
<td>– View that presents the content of the sixth page of the instructions</td>
</tr>
</tbody>
</table>

The multiple view controller classes used are responsible for the changes performed in their respective associated views. The *AudioEngine* class handles all the audio related manipulation and connects it to the algorithm’s audio processing, through the classes in Objective-C and C++ (described previously in Section 4.3): *Algorithm-Wrapper.h, Algorithm-Wrapper.mm, Algorithm.hpp* and *Algorithm.cpp*. Adding to these files, there are also other accessory files, such as the *MasterVoicing-Bridging-Header.h*, the mandatory *Info.plist* and the application’s storyboard files: *Main.storyboard* and *LaunchScreen.storyboard*.

Besides this, the application uses localization, to ensure that the UI elements change if the language changes both inside the application or on the device, which use the *GeneralLocalizer* and *CurrentLanguageLocalizer* classes and require a set of *Localizable.strings* files. The chosen language inside the application overrides the original device’s language.

66
4.5 Evaluation

This section describes some of the tests made, after the conclusion of Sprint 3, to the final product of this work - the MasterVoicing application - and its results.

4.5.1 Audio Performance Tests

In the initial tests that were the basis of the implementation process of MasterVoicing, a few measures of the audio latencies were made, while the application was running, in order to compare the audio performance between Objective-C and Swift and the two different considered audio frameworks (AudioToolbox and AVFoundation), as previously stated in Section 4.2.5.

After finishing the implementation of MasterVoicing, the same simple audio performance tests were made to the application, in order to better understand the application’s quality in terms of audio processing speed. The measured values, the audio input and output latencies of the audio session, and its results are depicted in Table 4.14. The tests were made using different values for different characteristics, such as buffer duration, audio sampling frequency, number of samples of the buffer or testing with or without the algorithm processing the audio. However, the only characteristics that made an alteration in the resulting values were the sample size and sampling frequency of the audio, which is the reason they are the only variables considered in Table 4.14. It was concluded that if the audio is processed at a higher sampling frequency and corresponding buffer size, its latency values decrease slightly.

Table 4.14: Audio performance tests - Results

<table>
<thead>
<tr>
<th>Sampling Frequency (Hz)</th>
<th>Sample Size</th>
<th>Input Latency</th>
<th>Output Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>22050.0</td>
<td>512</td>
<td>0.01256 seconds</td>
<td>0.00794 seconds</td>
</tr>
<tr>
<td>44100.0</td>
<td>1024</td>
<td>0.01249 seconds</td>
<td>0.00630 seconds</td>
</tr>
</tbody>
</table>

4.5.2 Usability Testing

To complement the audio performance testing, usability tests were conducted with the purpose of evaluating the application in terms of its visual appearance and interaction. Besides the obvious goal of getting an impression of the application’s usability from the possible future users of the application, another goal of these tests was to find out possible problems and/or improvements that could be made to the application.

13“Latency is the time it takes for a signal to travel through a system” [185].
MasterVoicing Application

The tests involved 15 participants\textsuperscript{14}, with an average age of 47 years, and were designed to have three different phases: pre-test interview, test tasks and post-test questionnaire.

Even though it is recommended that the participants of these tests have a moderate experience with the testing environment \cite{187} \cite{188}, which, in the mobile application’s case, is its operating system, it was difficult to find people with the desired characteristics. However, instead of seeing that as a drawback, it was used as an advantage, by comparing the tests results of participants with different degrees of familiarization and experience regarding iOS (7 were beginners and 8 were intermediate users).

The pre-test interview, whose collected data is organized in Section A.1, consisted in a set of questions to find out the characteristics of the participants and their familiarization with iOS. This phase was conducted by the facilitator of these tests (the writer of this document, with the aid of a Facilitator Guide document, shown in Section A.9 - written in Portuguese), in the beginning of the usability testing process. For the performance of the test tasks, the participant had one Participant Guide document (Section A.10 - also in Portuguese), which included a test orientation script, the tasks to be performed, stated in Table 4.15 and the questions which consisted in the post-test questionnaire. The participants were asked to classify each one of the tasks, after its completion or abandonment, in terms of difficulty, between Easy, Medium and Hard (to measure satisfaction, as shown in Section 4.5.2.3). The post-test questionnaire, that served as a way to understand the general final global perception the participant had of the application, had a rating criteria to be rated with a scale from 1 to 6, correspondent to the following meanings, in the given order: Bad, Mediocre, Medium, Good, Very Good and Excellent.

Table 4.15: Usability tests - Tasks

<table>
<thead>
<tr>
<th>TN</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Change pitch and volume</td>
<td>Alter the audio’s pitch and volume.</td>
</tr>
<tr>
<td>2</td>
<td>Using different playback modes</td>
<td>Use the application in real-time mode.</td>
</tr>
<tr>
<td>3</td>
<td>Application’s appearance</td>
<td>Change the color theme of the application.</td>
</tr>
<tr>
<td>4</td>
<td>Instructions</td>
<td>Find and visualize the application’s instructions.</td>
</tr>
<tr>
<td>5</td>
<td>Recording time limitation</td>
<td>Change the maximum recording time to 20 seconds.</td>
</tr>
</tbody>
</table>

\textbf{TN - Task Number}

\textsuperscript{14}According to a research made by the Nielsen Norman Group \cite{186}, as little as five participants are enough, in usability testing, to find almost the same usability problems that would be found with more participants.
The **post-test questionnaire** rating criteria were:

- **RC 1** - Appearance and visual aspects;
- **RC 2** - Ease of use, navigation and interaction;
- **RC 3** - Features;
- **RC 4** - Performance and speed;
- **RC 5** - Global evaluation.

All of the tests phases were performed with the participants sitting in a quiet room, around the same table as the facilitator, who observed the participant, and interaction was established only when necessary and without compromising the credibility of the tests; each participant used a testing device, specifically an *iPhone 5S*, with the application, and were given a set of in-ear headphones for using while testing, if they thought so necessary.

The facilitator collected all the necessary data in each one of the tests, through the interview, questionnaire and observations, in order to evaluate the usability tests by using the usually considered usability metrics, which are [189]:

- **Effectiveness** - the degree to which a user can accurately accomplished goals or tasks;
- **Efficiency** - the resources spent relatively to the accurately accomplished goals or tasks;
- **Satisfaction** - the comfort and fulfillment the user feels while using the application.

The next four sections provide all the results about the evaluation of the application, extracted from all the information collected in the usability tests, which is included in Appendix A.

### 4.5.2.1 Effectiveness Results

The effectiveness was calculated for each task, according to the formula in Figure 4.22, whose results are shown in Figure 4.23 and show that the second and fourth tasks were successfully completed by all the participants and that the fifth task was the one that more participants abandoned.

\[
Effectiveness = \frac{\text{Number of tasks completed successfully}}{\text{Total number of tasks undertaken}} \times 100\%
\]

Figure 4.22: Evaluation - Effectiveness formula (Adapted from [189])
In terms of efficiency, two types of efficiency were calculated, according to the formulas in Figures 4.24 and 4.26, that correspond to the Time Based Efficiency and Overall Relative Efficiency (values depicted in Figure 4.26). The Time Based Efficiency has the calculated value of 0.1463 tasks/sec for all participants, and respectively 0.0655 tasks/sec and 0.2171 tasks/sec for beginners and intermediate users. The Overall Relative Efficiency results confirm that familiarity with the platform significantly facilitates the successful completion of the tasks.

\[
\text{Time Based Efficiency} = \frac{\sum_{j=1}^{R} \sum_{i=1}^{N} n_{ij}}{NR}
\]

Figure 4.24: Evaluation - Time based efficiency formula (Adapted from [189])

\[
\text{Overall Relative Efficiency} = \frac{\sum_{j=1}^{R} \sum_{i=1}^{N} n_{ij}t_{ij}}{\sum_{j=1}^{R} \sum_{i=1}^{N} t_{ij}} \times 100\%
\]

Figure 4.25: Evaluation - Overall relative efficiency formula (Adapted from [189])

\[15\] In these formulas, \(N\) is the total number of tasks, \(R\) is the number of participants, \(n_{ij}\) is the result of a task \(i\) by participant \(j\) (if the participant completes the task, this value is equal to 1; if not, then is equal to 0) and \(t_{ij}\) is the time spent by participant \(j\) to complete the task or until its abandonment (the task duration) [189].
4.5.2.3 Satisfaction Results

The Task Level Satisfaction was studied using the collected data from the test tasks (included on the tests tasks collected data, in Appendix A). These results are shown in Figure 4.27, with the opinions of the participants about each task difficulty. According to them, the first and fourth tasks were the easiest tasks and the fifth task was the hardest one to complete or understand.

Figure 4.27: Evaluation - Task level satisfaction - Tasks difficulty
4.5.2.4 Post-Test Questionnaire Results

The mean values of the collected data in Section A.8, correspondent to the global rating of the application, are shown in Figure 4.28, along with their corresponding 95% confidence intervals (detailed in Section A.11). This results show that the participants’ mean classification for the application was Very Good for all the rating criteria, with variations between Good and Excellent.

![Global Rating Chart](image)

Figure 4.28: Evaluation - Post-test questionnaire - Mean values and 95% confidence intervals

The opinions and suggestions from the participants, which are described in Section A.7, unveiled some flaws and missing elements in the application, which recommend future improvements, as described in the next chapter, in Section 5.2.

4.6 Conclusions

The creation process of MasterVoicing was challenging, but also a great path to learn new technologies and concepts. This includes not only the novelty of learning how to develop mobile applications for iOS, but also the acquisition of knowledge about other subjects, more clearly addressed in other cycles of studies, such as audio processing. Besides that, it required the application and expansion of knowledge about previous learnt subjects, related with software engineering and human-computer interaction. Moreover, it is hoped that the creation of this application is only the beginning of a successful path for the creation of many more iOS applications to aid individuals with the particular type of speech considered in this dissertation, and that future updates will introduce new features (such as the ones suggested in 5.2) that improve its usefulness.
Chapter 5

Conclusions and Future Work

This chapter ends the work of this dissertation, which consisted in seeking the conclusion of its initial objectives, declared in Section 1.3, along with its writing, which documents the process of their completion.

Table 5.1: Dissertation objectives - Timeline

<table>
<thead>
<tr>
<th>Tasks / Month</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissertation writing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study of the problem and its solutions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Further study for the state of the art</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS development knowledge acquisition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming languages learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development of a throwaway prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development of the application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm adaptation and encapsulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm verification and validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evaluation of the application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribution of the application</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Completed

* Dissertation Planning: course unit that prepared and started this work
Conclusions and Future Work

5.1 Achievements

The objectives to achieve were planned monthly, between June and September of 2017, whose timeline is shown in the previous table: Table 5.1 (the achieved objectives are marked in it as Completed, in a green/darker color).

The timeline plan for the completion of the objectives was changed and delayed from its previous initial design, scheduled in the Dissertation Planning course unit, mainly because of two delaying situations. The first one was a delay in the initial development of the prototype, created by problems related to the complexity of the implementation of the desired audio features for the application, that revealed themselves in the initial tests and were successfully solved - this is detailed in Section 4.2. The second situation that also postponed the completion of MasterVoicing was a delay in the development of the algorithm, related to its complexity, which was meant to be used by the application.

As shown on Table 5.1, since the whisper-to-speech conversion algorithm intended to be used couldn’t be developed on the available time constraint, as previously explained, two of the objectives were abandoned:

- Algorithm verification and validation;
- Distribution of the application.

This also means that the adaptation and encapsulation of the algorithm was made throughout June, July and August, with the creation of the prototype, in Sprint 1 and later in Sprint 3, with the addition of the pitch changing slider.

Besides those two objectives, all the initial objectives were achieved.

5.2 Future Work

For future work, it is suggested the development of the abandoned features described in 4.2.7, to allow the user to share and save its recordings for later use.

From the opinions and suggestions from the participants, gathered in the usability testing, detailed in Section A.7, a set of improvements were thought for the application as possible future work:

- Accessibility improvements, such as the possibility to use a bigger font size or bigger UI elements;
Conclusions and Future Work

- Improvements on the instructions’ menu, related with content, design and navigation;
- Present the instructions’ menu right after the launch of the application;
- Redesign of the settings’ icon for better identification and recognition;
- Implement real-time changes in settings that produce immediate results, without requiring exiting its menu;
- Hiding the "Pitch" slider in the Real Voice audio processing mode;
- Improvements in alerts’s animations.

Regarding other improvements and new features, it is also suggested the implementation of the following:

- Full functionality of the application using the built-in-speaker of the device (currently, real-time mode requires an external audio output, due to the difficult problem of eliminating the audio feedback resulting from having recording and playback working at the same time);
- Improvements provided by the new announced features and updates of iOS 11 for AVFoundation, such as better support for real-time and offline audio processing (this would allow, for example, the offline processing of an imported audio file to the application) [164].

It is hoped that the initially desired whisper-to-speech algorithm is developed in the future. In that case, the two abandoned objectives should be completed, along with the adaptation and encapsulation of the algorithm in the application, in order to give an actual real-world usefulness to the application.

5.3 Conclusions

Even though the application does not use a voice processing algorithm related with whisper-to-speech conversion, which means it cannot be used for its intended main purpose, it consists in a complete application with several functionalities. Therefore, the application could also be used in the future with other audio processing algorithms, provided that they are adapted to the configuration of the audio buffer processing explained in Section 4.3. Even so, it has also room for improvements, regarding both its functionality and appearance.

In conclusion, it is thought that the continuation of the development of this application, including a functional whisper-to-speech algorithm, would be of great assistance for those with aphonia,
Conclusions and Future Work

since the current solutions all have disadvantages and do not seem to be as convenient as desired by those who may benefit from them.
References

Books


References


Articles


References


Conference Papers

References


Master’s Thesis


References

PhD Thesis


Web Pages


References


References


References


References


References


References


References


References


App Store Applications


References


References


Google Play Applications


References


Apple Developer Website

References


References


References


References


References

Stack Exchange Website


References


References


Other References

Appendix A

Usability Tests

A.1 Pre-Test Interview - Collected Data

Table A.1: Usability tests - Pre-test interview - Collected data

<table>
<thead>
<tr>
<th>PN</th>
<th>Age</th>
<th>Gender</th>
<th>Occupation</th>
<th>UT</th>
<th>Test Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84</td>
<td>Male</td>
<td>Retired College Professor</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>2</td>
<td>83</td>
<td>Female</td>
<td>Retired Basic School Professor</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>3</td>
<td>52</td>
<td>Female</td>
<td>Secretary</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>4</td>
<td>58</td>
<td>Male</td>
<td>Factory Worker</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>Female</td>
<td>Lawyer</td>
<td>Intermediate</td>
<td>September 7</td>
</tr>
<tr>
<td>6</td>
<td>55</td>
<td>Male</td>
<td>Lawyer</td>
<td>Intermediate</td>
<td>September 7</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>Male</td>
<td>College Student</td>
<td>Intermediate</td>
<td>September 7</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>Male</td>
<td>Unemployed</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>9</td>
<td>52</td>
<td>Female</td>
<td>House Maid</td>
<td>Beginner</td>
<td>September 7</td>
</tr>
<tr>
<td>10</td>
<td>54</td>
<td>Female</td>
<td>House Maid</td>
<td>Beginner</td>
<td>September 8</td>
</tr>
<tr>
<td>11</td>
<td>27</td>
<td>Female</td>
<td>Cashier</td>
<td>Intermediate</td>
<td>September 8</td>
</tr>
<tr>
<td>12</td>
<td>33</td>
<td>Male</td>
<td>Sales Worker</td>
<td>Intermediate</td>
<td>September 8</td>
</tr>
<tr>
<td>13</td>
<td>26</td>
<td>Female</td>
<td>Software Engineer</td>
<td>Intermediate</td>
<td>September 8</td>
</tr>
<tr>
<td>14</td>
<td>30</td>
<td>Male</td>
<td>Catering Trainer</td>
<td>Intermediate</td>
<td>September 8</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>Female</td>
<td>Social Worker</td>
<td>Intermediate</td>
<td>September 8</td>
</tr>
</tbody>
</table>

**PN** - Participant Number

**UT** - User Type (Familiarization with iOS)
Usability Tests

A.2 Tests Tasks - Collected Data - Task 1

<table>
<thead>
<tr>
<th>PN</th>
<th>CS</th>
<th>Duration</th>
<th>Difficulty</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completed</td>
<td>120 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Completed</td>
<td>180 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Failed</td>
<td>300 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Completed</td>
<td>10 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
</tbody>
</table>

PN - Participant Number
CS - Completion State
TN - Task Number
## Usability Tests

### A.3 Tests Tasks - Collected Data - Task 2

Table A.3: Usability tests - Tests tasks - Collected data - Task 2

<table>
<thead>
<tr>
<th>PN</th>
<th>CS</th>
<th>Duration</th>
<th>Difficulty</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completed</td>
<td>180 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Completed</td>
<td>70 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Completed</td>
<td>60 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Completed</td>
<td>48 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Completed</td>
<td>4 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Completed</td>
<td>4 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Completed</td>
<td>10 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Completed</td>
<td>6 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Completed</td>
<td>5 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Completed</td>
<td>4 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Completed</td>
<td>4 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
</tbody>
</table>

**PN** - Participant Number  
**CS** - Completion State  
**TN** - Task Number
## Usability Tests

### A.4 Tests Tasks - Collected Data - Task 3

Table A.4: Usability tests - Tests tasks - Collected data - Task 3

<table>
<thead>
<tr>
<th>PN</th>
<th>CS</th>
<th>Duration</th>
<th>Difficulty</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completed</td>
<td>120 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Completed</td>
<td>180 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Completed</td>
<td>10 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Failed</td>
<td>210 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Completed</td>
<td>7 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Completed</td>
<td>15 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Failed</td>
<td>423 seconds</td>
<td>Hard</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Completed</td>
<td>360 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Completed</td>
<td>300 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Completed</td>
<td>30 seconds</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Completed</td>
<td>26 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Completed</td>
<td>17 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Completed</td>
<td>10 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Completed</td>
<td>14 seconds</td>
<td>Easy</td>
<td></td>
</tr>
</tbody>
</table>

**PN** - Participant Number  
**CS** - Completion State  
**TN** - Task Number
# Usability Tests

## A.5 Tests Tasks - Collected Data - Task 4

Table A.5: Usability tests - Tests tasks - Collected data - Task 4

<table>
<thead>
<tr>
<th>PN</th>
<th>CS</th>
<th>Duration</th>
<th>Difficulty</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Completed</td>
<td>22 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Completed</td>
<td>85 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Completed</td>
<td>100 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Completed</td>
<td>28 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Completed</td>
<td>4 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Completed</td>
<td>11 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Completed</td>
<td>2 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Completed</td>
<td>60 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Completed</td>
<td>40 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Completed</td>
<td>5 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Completed</td>
<td>5 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Completed</td>
<td>3 seconds</td>
<td>Easy</td>
<td></td>
</tr>
</tbody>
</table>

**PN** - Participant Number  
**CS** - Completion State  
**TN** - Task Number
## Usability Tests

### A.6 Tests Tasks - Collected Data - Task 5

Table A.6: Usability tests - Tests tasks - Collected data - Task 5

<table>
<thead>
<tr>
<th>PN</th>
<th>CS</th>
<th>Duration</th>
<th>Difficulty</th>
<th>TN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Failed</td>
<td>300 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Failed</td>
<td>280 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Completed</td>
<td>40 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Failed</td>
<td>100 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Completed</td>
<td>15 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Completed</td>
<td>15 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Completed</td>
<td>6 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Failed</td>
<td>420 seconds</td>
<td>Hard</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Completed</td>
<td>60 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Completed</td>
<td>75 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Completed</td>
<td>60 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Completed</td>
<td>35 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Completed</td>
<td>11 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Completed</td>
<td>12 seconds</td>
<td>Easy</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Completed</td>
<td>13 seconds</td>
<td>Easy</td>
<td></td>
</tr>
</tbody>
</table>

**PN** - Participant Number  
**CS** - Completion State  
**TN** - Task Number
### A.7 Participants’ Opinions/Suggestions - Collected Data

Table A.7: Usability tests - Participants’ opinions/suggestions - Collected data

<table>
<thead>
<tr>
<th>PN</th>
<th>Opinions/Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Suggests bigger UI elements and font size, to improve readability</td>
</tr>
<tr>
<td>2</td>
<td>The instructions of the application should be presented at its initial launch</td>
</tr>
<tr>
<td>3</td>
<td>The instructions should have more specific help regarding the settings’ options</td>
</tr>
<tr>
<td>4</td>
<td>It is not easy to understand how to exit the instructions’ menu</td>
</tr>
<tr>
<td>5</td>
<td>The settings’ icon does not provide the intended meaning; suggests a new design</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>The &quot;Pitch&quot; slider should not appear in the Real Voice processing mode</td>
</tr>
<tr>
<td>8</td>
<td>The instructions’ menu should have an easier way to exit</td>
</tr>
<tr>
<td>9</td>
<td>Thought the application’s settings were hard to find</td>
</tr>
<tr>
<td>10</td>
<td>Suggested the font size should have the option of a bigger size</td>
</tr>
<tr>
<td>11</td>
<td>Changes in the settings should produce immediate changes</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The alerts informing about audio changes should be faster or not block UI interaction</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>The instructions’ menu should have a title similar to the settings’ menu</td>
</tr>
</tbody>
</table>

**PN** - Participant Number
Usability Tests

A.8 Post-Test Questionnaire - Collected Data

Table A.8: Usability tests - Post-test questionnaire - Collected data

<table>
<thead>
<tr>
<th>PN</th>
<th>RC 1</th>
<th>RC 2</th>
<th>RC 3</th>
<th>RC 4</th>
<th>RC 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

PN - Participant Number
RC - Rating Criterion
1 - Bad
2 - Mediocre
3 - Medium
4 - Good
5 - Very Good
6 - Excelent
A.9  *Facilitator Guide*

**Teste de Usabilidade - Documento do Examinador**

**Dados Pessoais** - Participante Nº ______
- Idade:
- Sexo: [ ] Masculino  [ ] Feminino
- Profissão:

**Entrevista Inicial** - Guião
- Qual o seu grau de familiarização com o sistema operativo iOS?
  [ ] Principiante  [ ] Intermédio  [ ] Especialista

**Avaliação de Tarefas** - Guião

**Tarefa 1** - Utilize a aplicação para alterar o seu tom e volume de voz.
- [ ] Tarefa concluída
- Duração da tarefa:
- Observações:

**Tarefa 2** - Utilize a aplicação no modo de reprodução em tempo real.
- [ ] Tarefa concluída
- Duração da tarefa:
- Observações:
### Tarefa 3 - Altere o esquema de cores da aplicação.
- Tarefa concluída
- Duração da tarefa:
- Observações:

### Tarefa 4 - Encontre e visualize as instruções da aplicação.
- Tarefa concluída
- Duração da tarefa:
- Observações:

### Tarefa 5 - Altere o tempo máximo de gravação para 20 segundos.
- Tarefa concluída
- Duração da tarefa:
- Observações:
Teste de Usabilidade - Tarefas - Participante Nº_____

Este conjunto de tarefas pretende servir como base ao estudo de usabilidade da aplicação MasterVoicing, uma aplicação, para o sistema operativo iOS, com o objectivo de ser um assistente para pessoas com afonia. A sua funcionalidade principal está simulada com a utilização de um algoritmo de mudança de tom de voz, e pretenderá utilizar futuramente um algoritmo de alteração de voz que permita a conversão de sussurros em fala audível.

No fim de cada tarefa, classifique, por favor, o grau de dificuldade que sentiu na realização da tarefa, entre as opções "Fácil", "Médio" e "Difícil". Para a realização de cada tarefa, ser-lhe-hão fornecidos, para além de um dispositivo com a aplicação, um conjunto de auscultadores, com o quais poderá testar o funcionamento da mesma.

---

**Tarefa 1 - Alterar o tom e volume da voz**
- Utilize a aplicação para alterar o seu tom e volume de voz.

✔️ Grau de dificuldade:  ☐ Fácil  ☐ Médio  ☐ Difícil

---

**Tarefa 2 - Utilizar diferentes modos de reprodução**
- Utilize a aplicação no modo de reprodução em tempo real.

✔️ Grau de dificuldade:  ☐ Fácil  ☐ Médio  ☐ Difícil

---

**Tarefa 3 - Aspecto da aplicação**
- Altere o esquema de cores da aplicação.

✔️ Grau de dificuldade:  ☐ Fácil  ☐ Médio  ☐ Difícil

---

**Tarefa 4 - Instruções**
- Encontre e visualize as instruções da aplicação.

✔️ Grau de dificuldade:  ☐ Fácil  ☐ Médio  ☐ Difícil

---

**Tarefa 5 - Limitação do tempo de gravação**
- Altere o tempo máximo de gravação para 20 segundos.

✔️ Grau de dificuldade:  ☐ Fácil  ☐ Médio  ☐ Difícil

---

Figure A.3: Usability tests - Participant Guide - Page 1
Questionário Final - Avaliação Global da Aplicação

A tabela seguinte contem um conjunto de critérios relativos à aplicação. Por favor classifique cada um deles, de acordo com um valor entre 1 e 6, que têm o seguinte significado:

◆ 1 - Mau  
◆ 2 - Mediocre  
◆ 3 - Médio  
◆ 4 - Bom  
◆ 5 - Muito Bom  
◆ 6 - Excelente

<table>
<thead>
<tr>
<th>Critérios de Avaliação</th>
<th>Classificação</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Aparência e aspectos visuais</td>
<td></td>
</tr>
<tr>
<td>- Facilidade de utilização, navegação e interacção</td>
<td></td>
</tr>
<tr>
<td>- Funcionalidades da aplicação</td>
<td></td>
</tr>
<tr>
<td>- Desempenho e velocidade</td>
<td></td>
</tr>
<tr>
<td>- Avaliação global</td>
<td></td>
</tr>
</tbody>
</table>

Se desejar fazer alguma observação ou sugestão relativamente à avaliação da aplicação, utilize a próxima secção.

Observações/Sugestões

Muito obrigado/(a) pela sua participação!
## A.11 Global Rating - Calculated Values

Table A.9: Usability tests - Global rating - Calculated values

<table>
<thead>
<tr>
<th>RC</th>
<th>Mean Value</th>
<th>CV</th>
<th>Min. Value</th>
<th>Max. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.733</td>
<td>0.232</td>
<td>5.502</td>
<td>5.965</td>
</tr>
<tr>
<td>2</td>
<td>5.267</td>
<td>0.356</td>
<td>4.911</td>
<td>5.623</td>
</tr>
<tr>
<td>3</td>
<td>5.133</td>
<td>0.261</td>
<td>4.872</td>
<td>5.395</td>
</tr>
<tr>
<td>4</td>
<td>5.667</td>
<td>0.312</td>
<td>5.354</td>
<td>5.979</td>
</tr>
<tr>
<td>5</td>
<td>5.733</td>
<td>0.300</td>
<td>5.433</td>
<td>6.034</td>
</tr>
</tbody>
</table>

**RC** - Rating Criterion  
**CV** - Confidence Value (95%)