



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

BACHELOR IN COMPUTING ENGINEERING

**Gamification-based Tool to Practice Basic
Exercises with Wind Instruments**

Antonio Pulido Hernández

July, 2020

**GAMIFICATION-BASED TOOL TO PRACTICE BASIC EXERCISES WITH
WIND INSTRUMENTS**



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

Department of Tecnologías y Sistemas de Información

**BACHELOR IN COMPUTING ENGINEERING
SOFTWARE ENGINEERING**

**Gamification-based Tool to Practice Basic
Exercises with Wind Instruments**

Author: Antonio Pulido Hernández

Advisor: David Vallejo Fernández

Co-advisor: Diego Molero Marín

July, 2020

Antonio Pulido Hernández

Ciudad Real – Spain

E-mail: Antonio.Pulido@alu.uclm.es

Telephone: 639 871 986

© 2020 Antonio Pulido Hernández

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Some of the names used by companies to differentiate their products and services are registered brands. The names that appear in this documents and when the author was informed about the registered brands, will be in caps or with a proper name.

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE

VOCAL

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Abstract

Music can be understood as the art of combining sounds to produce beauty following the rules of rhythm, melody, and harmony. Learning how to play a musical instrument is considered as an important part of education nowadays. The reason is that musical learning helps to develop the creative thinking of students.

Musicians need to practice countless hours aiming to learn how to correctly play an instrument. There are music schools where students learn and practice with different instruments. However, they also need to practice exercises at home. These exercises, which are performed as a warm-up prior to a practice session, are aimed at developing technical skills with the instrument. In this context, one of the challenges to be addressed is that these exercises tend to be repetitive and boring, which can result in a lack of motivation.

The project discussed in this document was conceived as a way of solving this issue when apprentices study at home. The main problem is that most of the time apprentices perform these exercises in an incorrect way and they do not have feedback until the next class. This situation may ultimately lead into a lack of motivation. Thus, the key idea behind this project is to design, develop, and validate a software application to address this issue.

The proposed application will be primarily oriented to tablets. The reason behind this approach is that the screen size of a tablet allows a correct visualization of music sheets. Another important aspect of the application is the integration of gamification techniques to give feedback and motivate students. In this context, the application will give support to wind instruments. The reason is that in order to measure and record the sounds produced by an instrument it is necessary that only one note is played at a time. The problem is that there are instruments, for example, the piano that can play two notes at the same time pressing two keys. As a result, the application will focus its attention to wind instruments that can only play one note at the same time.

Resumen

La música puede ser definida como el arte de combinar sonidos para producir belleza siguiendo las reglas del ritmo, melodía y armonía. El aprendizaje musical es considerado una parte esencial de la educación en este momento. Esto se debe a que el aprendizaje musical ayuda a desarrollar la creatividad de los estudiantes.

Los músicos necesitan practicar incontables horas tratando de aprender como tocar correctamente su instrumento. Hay escuelas musicales donde los estudiantes practican diferentes instrumentos. Sin embargo, es necesario que practiquen ejercicios en casa también. Estos ejercicios, que son practicados como un calentamiento previo a la realización de una práctica más extensa, son necesarios para el desarrollo de habilidades técnicas con el instrumento. En este contexto, uno de los retos abordados es que estos ejercicios tienden a ser repetitivos y aburridos, lo que puede resultar en una falta de motivación.

El proyecto detallado en este documento ha sido concebido como una manera de resolver el problema de los estudiantes cuando estudian por su cuenta. El principal problema es que la mayoría de los estudiantes realizan ejercicios de una manera incorrecta y no tienen una realimentación hasta la siguiente clase. Esta situación puede llegar a producir una falta de motivación. Por eso, la idea principal detrás de este proyecto es el diseño, desarrollo y validación de una aplicación software que aborde este problema.

La aplicación propuesta estará orientada principalmente a tablets. La razón detrás de este enfoque es el tamaño de las pantallas de las tablets que es apropiado para la visualización de partituras. Otro aspecto importante de la aplicación es la integración de técnicas de gamificación para dar retroalimentación y motivar a los estudiantes. In este contexto, la aplicación dará soporte a instrumentos musicales. La razón es que para medir sonidos producidos por instrumentos musicales es necesario que solo una nota se escuche a la vez. El problema es que hay instrumentos, por ejemplo, el piano, con el que se pueden tocar más de una nota a la vez. Por esta razón la aplicación estará orientada a los instrumentos de viento que solo pueden tocar una nota a la vez.

Thank-you note

Este proyecto surge de la fusión de mis dos grandes pasiones la música y la informática. Debo dar las gracias a muchas personas que han formado parte de mi vida y que me han ayudado a crecer como persona, en especial a mis padres por la educación que he recibido de ellos y por las oportunidades que me han brindado, sin su ayuda y sin su apoyo en momentos difíciles no hubiera llegado hasta aquí.

Me gustaría agradecer a mi profesor de música José Julián, ha sido quien me ha estado enseñando a tocar la trompeta desde hace más de 10 años y gracias a él descubrí mi amor por la música.

También a mis profesores en la carrera, en el instituto y en el colegio, ellos me enseñaron todo lo que sé y algún día me gustaría ser como ellos.

Gracias a todos los amigos que tengo y que siempre han estado allí para apoyarme, a los que conozco de toda la vida del pueblo, a los amigos que hice en el instituto, a los que he hecho en la carrera y en especial a Eduardo, Ismael y Ana por todas esas tardes juntos.

Gracias a mi hermano pequeño, por ser siempre alguien en el que confiar y apoyarme, espero que dentro de 3 años estés presentando tu TFG.

Gracias a toda mi familia en general, abuelos, tíos, y primos. En especial a mi abuelo Pedro, siempre me enseñaste la importancia de estudiar y querías que todos tus nietos tuviéramos la oportunidad de tener una carrera, hubiera sido una gran felicidad que estuvieras aquí para verme en este momento.

Por supuesto también gracias a mis compañeros de Alarcos, en especial a David por dejarme formar parte de un proyecto tan apasionante.

Y para concluir me gustaría dar las gracias a mis dos tutores en este proyecto, gracias Diego por ser un amigo y compañero que siempre ha estado para ayudarme. También a mi director y amigo David por todos sus consejos, sus horas invertidas en ayudarme cuando lo he necesitado y su confianza en mí cuando surgió la idea de este proyecto.

Antonio

A mis padres

Contents

| | |
|---|--------------|
| Abstract | v |
| Resumen | vii |
| Thank-you note | ix |
| Contents | xiii |
| List of Tables | xvii |
| List of Figures | xix |
| List of code listings | xxi |
| List of acronyms | xxiii |
| 1 Introduction | 1 |
| 2 Objectives | 5 |
| 2.1 General objective | 5 |
| 2.2 Specific objectives | 5 |
| 3 State of art | 7 |
| 3.1 Musical training | 7 |
| 3.1.1 Basic concepts | 7 |
| 3.1.2 General learning | 10 |
| 3.1.3 Trumpet practice | 12 |
| 3.1.4 Basic exercises to practice with wind instruments | 14 |
| 3.2 Gamification in education | 15 |
| 3.2.1 Basic concepts | 15 |
| 3.2.2 Applications of Gamification | 16 |
| 3.3 Technology and music | 19 |

0. CONTENTS

| | | |
|----------|--|-----------|
| 3.3.1 | Relationship between music and video games | 19 |
| 3.3.2 | Mobile applications for wind instruments | 20 |
| 4 | Methodology | 23 |
| 4.1 | Software development methodology | 23 |
| 4.1.1 | Why Extreme programming | 23 |
| 4.1.2 | The application of Extreme Programming | 24 |
| 4.2 | Hardware and Software Resources | 25 |
| 4.2.1 | Hardware resources | 25 |
| 4.2.2 | Software resources | 25 |
| 5 | Architecture | 27 |
| 5.1 | Architecture overview | 27 |
| 5.2 | Graphical user interface layer | 29 |
| 5.2.1 | Menu Module | 30 |
| 5.2.2 | Exercise Module | 31 |
| 5.3 | Control and tracking layer | 33 |
| 5.3.1 | Tracking Module | 33 |
| 5.3.2 | Manager Module | 35 |
| 5.4 | Persistence Layer | 37 |
| 5.4.1 | JSONParser | 38 |
| 5.4.2 | Web Server Module | 39 |
| 5.5 | Design patterns | 41 |
| 6 | Results | 45 |
| 6.1 | Work distribution | 45 |
| 6.1.1 | Iterations | 45 |
| 6.1.2 | User stories | 47 |
| 6.2 | Windy Melody: Final Result | 50 |
| 6.3 | Project cost and resources | 58 |
| 6.4 | Project statistics | 58 |
| 7 | Conclusions | 61 |
| 7.1 | Reached objectives | 61 |
| 7.2 | Problems faced | 62 |
| 7.3 | Addressed competences | 62 |
| 7.4 | Future work | 63 |

| | | |
|----------|--|-----------|
| A | Appendix A | 67 |
| A.1 | Image References | 67 |
| B | Appendix B | 71 |
| B.1 | Routine represented in a JSON file | 71 |
| | References | 75 |

List of Tables

| | | |
|------|---|----|
| 6.1 | User story US-LOGIN. | 47 |
| 6.2 | User story US-REGISTER. | 48 |
| 6.3 | User story US-PROFILE. | 48 |
| 6.4 | User story US-STATISTICS. | 48 |
| 6.5 | User story US-LOGOUT. | 48 |
| 6.6 | User story US-SCALE. | 48 |
| 6.7 | User story US-TUNER. | 49 |
| 6.8 | User story US-CREATIVE. | 49 |
| 6.9 | User story US-PRACTICE. | 49 |
| 6.10 | User story US-LONG. | 49 |
| 6.11 | User story US-EXERCISE. | 49 |
| 6.12 | Estimated budget for the developed project. | 58 |

List of Figures

| | | |
|------|---|----|
| 3.1 | Notes from Do4 to Do5 | 8 |
| 3.2 | Notes duration. | 9 |
| 3.3 | Rest notes | 9 |
| 3.4 | Types of clefs | 10 |
| 3.5 | Music sheet | 10 |
| 3.6 | Trumpet parts | 12 |
| 3.7 | Trumpet tessiture | 13 |
| 3.8 | Do Mayor Scale | 14 |
| 3.9 | Long notes exercise | 15 |
| 3.10 | Rapid notes exercises. | 15 |
| 3.11 | Starbucks reward app | 17 |
| 3.12 | mango health app | 18 |
| 3.13 | Duolingo web app | 18 |
| 3.14 | Tonestro app | 21 |
| 3.15 | How to play Trumpet app | 21 |
| 3.16 | Canciones de trompeta | 21 |
| | | |
| 5.1 | General architecture of the system. | 28 |
| 5.2 | GUI Layer Architecture | 30 |
| 5.3 | State diagram | 33 |
| 5.4 | Control and Tacking layer | 34 |
| 5.5 | Midi to note | 36 |
| 5.6 | Example in the Manager Module | 37 |
| 5.7 | Database diagram | 38 |
| 5.8 | Database diagram | 40 |
| 5.9 | Factory method | 42 |
| 5.10 | MVC design pattern | 43 |
| 5.11 | Builder design pattern | 44 |

0. LIST OF FIGURES

| | | |
|------|--|----|
| 6.1 | Working environment. | 50 |
| 6.2 | Login Menu. | 51 |
| 6.3 | Register Menu. | 52 |
| 6.4 | Profile Menu. | 52 |
| 6.5 | Statistics Menu. | 53 |
| 6.6 | Create scale menu. | 54 |
| 6.7 | Exercise Menu. | 55 |
| 6.8 | Creative Menu. | 56 |
| 6.9 | Tuning Menu. | 56 |
| 6.10 | Long Notes Menu. | 57 |
| 6.11 | Practice Mode. | 57 |
| 6.12 | Additions and Deletions per week. | 59 |
| | | |
| A.1 | Login reference. | 67 |
| A.2 | Register reference. | 68 |
| A.3 | Profile reference | 68 |
| A.4 | Statistics reference. | 69 |
| A.5 | Routines reference. | 69 |
| A.6 | Scale reference. | 70 |
| A.7 | Icons designed https://www.flaticon.es/autores/freepik | 70 |

List of code listings

| | | |
|-----|--|----|
| 5.1 | Script associated to Login Menu | 31 |
| 5.2 | Script associated to Login Menu | 37 |
| 5.3 | Example of Exercise in JSON File | 39 |
| 5.4 | Example of Exercise in JSON File | 41 |
| B.1 | Example of Routine in JSON File | 71 |
| B.2 | Example of Routine in JSON File | 72 |
| B.3 | Example of Routine in JSON File | 73 |

List of acronyms

| | |
|--------------|--------------------------------------|
| MVC | Model View Controller |
| JSON | JavaScript Object Notation |
| GUI | Graphical User Interface |
| HTTPS | Hypertext Transfer Protocol Secure |
| PHP | Hypertext Preprocessor |
| MIDI | Musical Instrument Digital Interface |
| SQL | Structured Query Language |
| NoSQL | not only SQL |

Chapter 1

Introduction

Music has been one of the most important form of arts in the History of Humanity. According to several studies, music its an essential part in the culture and psychology of people [Kra03], “music reflects culture“, it was present in prehistory when people used their voice or simple percussion instruments and it is present nowadays in a lot of different forms.

There is no an agreement in the scientific community of when music instruments were created by humans, it is roughly said that it started between 60,000 and 40.000 years ago. The possible oldest known wind music instrument is the *Divje Babe Flute*, it was created 43.000 years ago, there are experts that argued that maybe is not a musical instrument and it is part of the *Aurignacian culture*, however the *Divje Babe Flute* is still known as a Neanderthal flute. It is not until 1400 BC that the first form of musical notation appears, an ancient tablet created in Babylonia and it has simple instructions to perform a melody with a lyre [Wes94]. **Musical notation** was born as a system to visually represent, modern musical notation is used since the 13th century, it includes concepts as *tempo*, *pitch* and *duration*.

As vocal music was the first type of music created by humans and percussion is considered the second one, we could say that wind instruments were the third. Humans at that time used shells, bones or plants to create wind instruments. The first ones were able to produce only one pitch and it was common to get different instruments in order to create a melody playing them. It was not until the discovery of finger holes that basic melodies and harmonies could be played with just one wind instrument, since them wind instruments were evolving slowly until the 19Th century. It was in 1820 were the first modern trumpets were created and 1840 when the saxophone was invented by Adolphe Sax [How03]. It was also a gold time for composers and instrument makers, new techniques and mechanics where applied to create due to the advances in the technology. These days most of the wind instruments are based on the ones created in that century.

Learning to play an instrument is a tedious task that requires years of practice. It implies to practice with the instrument and be able to read different music sheets. Nowadays, musicians start learning **musical language** before practicing with any instrument and when they could read simple music sheets they start practicing with an instrument.

1. INTRODUCTION

Musical language is as important as learning how to play an instrument for musicians. Read properly a music sheet is not a trivial issue; musicians have studied for years in order to acquire the knowledge necessary to understand a music sheet. In addition, musical language includes another concepts such as *ear training* with melodic dictations and rhythm with rhythmic dictations.

The traditional model of learning how to play an instrument is simple. Students go once or twice a week to classes where they are taught how to play some notes and melodies. Then, at home, they have to practice with their instrument music sheets that will be played the next week in class. Music has a curious aspect: even if you do not know anything about music theory you could guess when something is played wrong in a melody. However, this is not enough for musicians that are aiming to master their skills. Only practicing at home and finding out what they are doing wrong until the next class could not be efficient.

Music cannot be understood without **technology**, and that is why listening to music has changed over the last centuries. In the past, people had to go to a concert or a festival in order to be able to listen to music. It was not until 1860 that the first song was recorded; its name was *Au clair de la lune*, a French Folk. Nowadays, music is unimaginable without recording devices. The revolution that technology has brought to music was not only the possibility of recording songs, it is also changing the way we learn music. In the past, there were no many ways of learning to play an instrument. The only option was going to an academy or being taught by someone that already knew how to play that instrument. However, that situation has changed thanks to the Internet and new technologies. The Internet is the primary source of information and even if it is extremely difficult to learn music only using the Internet. It is possible to complement the role of a teacher in learning music. For example, there are tutorial videos where musicians can learn some concepts about their instruments.

One of the major issues of music learning is the lack of motivation when musicians practice with their instruments at home, since they must spend hours performing repetitive exercises that are essential for them. There are studies that affirm that mastering a wind instrument requires more than 10.000 hours [SDHM96]. One aspect to take into consideration is that musicians that were in a music course spent less than half of time that the ones that study without any help. We could say that even if it is necessary to practice a lot of time is more important to use efficiently that time.

New technologies are present in our lives, due to the increasing accessibility to information through e-learning has changed; there are a lot of online platforms that are used as a help for students. Depending on the type of information a different approach could be taken, there are applications that support students in some tasks. There are platforms that are used as a way of exchange information between students and teachers while there are others that stimulate the self-learning on the students.

One of the more influential technology is mobile devices. They are way more versatile than desktop computers and they are more present in our lives. According to the *Global Consumer Survey 2017* made by Deloitte¹ 94% of people used Smartphones the last 24 hours. And the market of mobile applications is increasing every day. The use of mobile application as a source of information is common nowadays, knowledge of any field can be acquire with a mobile device.

Self-learning applications are getting more and more popular nowadays. However, a problem of this kind of applications is that it is not easy to make students motivated to learn and take advantage of this approach. One of the solutions that is arising is the use of gamification techniques in order to motivate students [CLSBÁRCG18]. A lot of researchers have been studying that try to explain what makes video games fun and how use it in education [Ma180], using the game mechanics in educational application could improve the results.

On the other hand, it is not an easy task to make an educational application fun by applying some games mechanics. Gamification techniques should come with a great idea and a design that adapts to them. There are plenty of examples of success of applications that apply gamification techniques. They can even be applied in other fields that are not education. It is possible to observe examples of this kind of applications in different domains like health, business and workplace.

Taking into account the concepts explained above, the idea of making an application that helps musicians to study at home keeping the motivation appeared, using the mobile technologies that provide ease to use and a connection to Internet its possible to create an application that helps students to deal with the lack of motivation. ***Windy Melody*** is an application that was born with the idea of allowing musicians to practice common exercises for wind instruments that are useful for beginners and experts, while they have fun by means of gamification techniques. Every musician even if it is an expert or a beginner practice some exercises to improve their skill, these exercises have all the same basic principles, the only different is the difficulty of them.

Windy Melody, the project discussed in this document, applies gamification techniques that give feedback to users and defines clear learning goals. The users will have the freedom of even creating their own exercises, which is a way of motivating them. What is more, creating their own exercises allows them to set a proper difficulty for them. *Windy Melody* applies the traditional learning techniques in the field of music, and it gives feedback about the exercise they are playing. Furthermore, the users will have different profiles assigned to each instrument they play and they would be able to consult the statistics of each profile. In addition, *Windy Melody* is an Android application designed especially for tablets. It will not be easy for musicians to read a music sheet in their mobile phones but the size of a

¹<https://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte-ES-TMT-Consumo-Movil-2017.pdf>

1. INTRODUCTION

common tablet fits well, since it takes the advantages of the mobile applications explained above. Even if this application is thought to use it at home with your instrument, a mobile application is more comfortable for users than a desktop application.

Chapter 2

Objectives

In this chapter, the general objective and the specific objectives proposed for this project are introduced and described.

2.1 General objective

The main objective of this project is **to design and develop a software tool that facilitates and motivates, through gamification techniques, to practice wind instruments**. It is intended to create a mobile application easy to use and that is appropriate for beginners with their instruments and for experts that want to use this application to improve their skills or to warm up. The application needs to record the sounds that an instrument produce and evaluate if its the right note every time the musician plays them.

2.2 Specific objectives

In order to carry out the development of the project, the general objective is structured into the following specific objectives:

- **Scalability.** In order to create a system able to adapt to changes it is necessary to create a scalable system. This type of system would allow the addition of new features and the inclusion of new types of exercises, routines, and instruments. Furthermore, the client could change the requirements in any moment and the system has to adapt easily to these changes.
- **User's motivation.** Gamification techniques will be applied in the design and development of the project in order to encourage musicians to practice exercises that are essential for them. The system will have a profile menu and a statistics menu where the users will be able to check their progress. For every exercise the system will use gamification techniques to show what notes they are playing.
- **Ease of use.** Having a pleasant user experience is as important as the provided functionality. The user will follow the quality guidelines for the development in Android-based devices¹ in order to achieve this objective.

¹<https://material.io/design/usability/accessibility.html#writing>

2. OBJECTIVES

- **Intuitiveness.** The application will integrate application views that the user will understand. Even if the application is useful for expert musicians that know a lot of technical concerns about music, beginners with their instrument have to be comfortable to create exercises without knowing music theory, only the basics for reading a simple music sheet.
- **Validation regarding a specific wind instrument.** An application of these characteristics is intended to work with a high variety of instruments, in this case wind instruments. In order to validate the system, the trumpet in SiB is the chosen instrument. There are other wind instruments, like the clarinet and the flugelhorn, which are in SiB too. This means that validating the system with the trumpet will allow us to ensure that it will work with most wind instruments.

Chapter 3

State of art

3.1 Musical training

Sounds are present in the world everywhere, *sound* can be perceived from birds, other people, when two objects collide, with instruments. So what kind of sounds are considered as music? Music can be understood as the art of combining sounds following the rules of rhythm, melody and harmony. One of the most common and traditional ways of creating music is using instruments, learning how to play an instrument it is not a trivial issue and requires a lot of years of study and practice. Musicians take extremely serious their practice with their instruments because any of them even if they are amateurs or professionals have to follow specific routines if they want to improve or even to keep their skills.

3.1.1 Basic concepts

Sounds are an essential part of music, a sound is created by the vibration of an object and this causes the medium around it to vibrate too. Vibrations which travel by air are called longitudinal waves, these waves are what humans can hear [dIF13].

longitudinal waves have two magnitudes which determine how the wave is and how the sound will be perceived by humans, these two magnitudes are wave length and frequency. These are the physical aspects of sound. However, in music we are more interested in the concepts of *pitch* and *intensity* [Ler04]:

- **Pitch:** it is a concept that is closely related with the concept of frequency and almost equivalent. While frequency is a physical concept and absolute (440 Hz, 446 Hz), pitch is referred to the ear's perception of that frequencies, depending on the pitch we have different note names.
- **Intensity:** it depends on the position of the listener with respect to the source of sound, it will decrease as the listener moves away from the source and will increase if the listener gets closer. This property is what makes a sound louder or softer.

These concepts are necessary to explain what is a *musical note*. A note or tone is the pitch of a particular sound and the time it lasts, this means that if we want to refer to a particular frequency the term pitch can be used. However, if this sound has a particular duration the

3. STATE OF ART

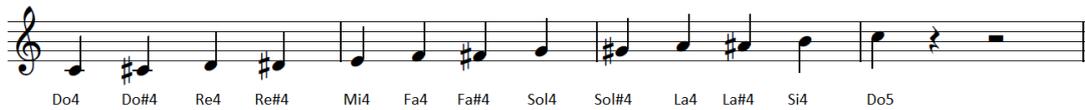


Figure 3.1: Notes from Do4 to Do5

term note is preferred. In music there are 7 names of notes which are: *DO, RE, MI, FA, SOL, LA, SI*. Even so, it is common knowledge that there are more than 7 possible notes, this is handled by octaves and semitones. An octave has 12 different notes that are separated by a semitone but in an octave are used the 7 names previously commented, to represent the other 5 notes sharps and flats are used.

This notation is not unique, there are different notations used to name the notes. The most popular ones are the Latin notation which is explained above and the English notation that has 7 names of notes as well but the names are different, these are the equivalences LA(A), SI(B), DO(C), RE(D), MI(E), FA(F), SOL (G) (see Figure 3.1).

Notes have a pitch assigned, which means that there is a way to calculate the frequency that a particular note has. The frequencies assigned to the different pitches has changed through history. The standard way nowadays is to have the pitch *LA 4* with a frequency of *440Hz*. The rest of frequencies are calculated using the next formula where *n* is the number of semitones that are between *LA 4* and the note that is calculated [TC06]:

$$Frequency = 440 * 2^{n/12} \quad (3.1)$$

As notes have different names depending on the pitch they have, the duration of a note it has a particular name. The most common types are: *Semibreve, Minim, Crotchet, Quaver or Semiquaver*.

Music is commonly represented graphically in a music sheet where it is possible to observe the following elements:

- **Staff:** it is made by 5 horizontal lines, notes can be placed between the lines or in the lines and depending where they are placed the pitch of the note changes. It is possible to place notes below and above the staff, in that case it is mandatory to add complementary lines in order to differentiate the notes.
- **Notes:** depending on the duration of the note they have completely different shapes, as we can observe in Figure 3.2.
- **Beats:** the speed in a music sheet is defined by the beats per minute. There are music



Figure 3.2: Notes duration.

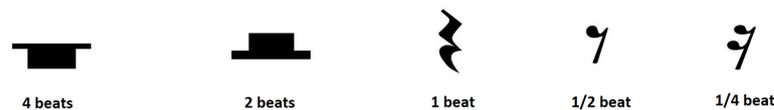


Figure 3.3: Rest notes

sheets that specifies a concrete number for the bpm. However, the most common way is to specify a range of bpm, in music this ranges have names, for example Adagio(66 – 76 bpm), Andante (76 – 108 bpm) or Allegro(110 – 168 bpm).

- **Rest notes:** it is a special type of note, a rest note specifies the absence of any sound, there are different symbols for rest notes depending on the duration of the absence of sound (see Figure 3.3).
- **Clef:** it is used to assign a particular pitch to a line of the staff. There are 4 commonly used clefs which are treble clef, bass clef, tenor clef (see Figure 3.2). The most common one is the treble clef which assign sol4 to the third line of the staff. This means that when a note is placed in the third line its corresponding frequency is 391.995 Hz.
- **Time signature:** it is a notation to specify how many beats are contained in each bar and what duration has each beat. A 2/4 music sheet has 2 beats and a duration of a crotchet (4) for each beat.

3. STATE OF ART

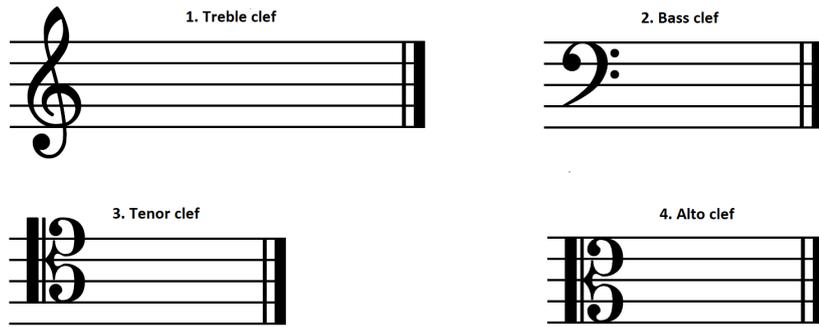


Figure 3.4: Types of clefs



Figure 3.5: Music sheet

3.1.2 General learning

Music is part of our lives. It is studied in elementary schools and in secondary schools. There are also music academies where people learn how to play instruments like the trumpet, the piano or the clarinet. There are also superior music studies in which people study a bachelor of music to become a professional musician.

Music learning covers a wide range of knowledge, it goes from music history to technical skills for learning how to play an instrument. In our case we will focus the attention on how musicians learn to play an instrument.

As music is taught at schools, kids start to learn some concepts like rhythm or to differentiate two sounds with different frequencies. In addition, students at secondary schools learn how to play the flute, they learn to play each note and to read a music sheet.

Although music learning is part of the education nowadays, the knowledge learned in basic education is limited and should be complemented if the aim of the student is to be a professional. Musicians that want to learn how to play an instrument usually go to an academy in which they are taught how to play one or more instruments.

Music learning is not just about learning to play an instrument, every musician in academies learn musical language. *Musical language* covers the part of learning how to read music sheets, creating compositions and be capable of distinguish between different notes. Musical language allows musicians to be able to play more complex music sheets and not just simple ones.

There are a lot of different instruments and each one has specific characteristics, but they can be divided in three distinct mayor types that are called families which are as follows [Car02]:

- **Wind instruments:** the sound is produced by the vibration of the air. Depending on the type of the mouthpiece and the length of the tube the sound is different. Some examples are the trumpet, clarinet, and saxophone.
- **String instruments or chordophones:** the sound is produced by the vibration of the strings, some of them are played by plucking them with the fingers like the guitar, others are hit with a hammer and the rest are rubbed with a bow like the violin.
- **Percussion instruments:** this instruments are a bit special and can be identified because they are played by striking them. Percussion instruments usually play rhythm. However, they also can play melody or harmony, for example the xylophone play melody in some music sheets.

Even inside a family of instruments there are differences between them. The bigger family would be wind instruments. These instruments of this family are formed by one or several tubes and the sound it is produced depends on the length of the tubes, this kind of instruments do not allow to play different pitches at the same time like the guitar or the piano, this property is called *monophony* [APL10]. Wind instruments are also divided in two types, *brass instruments* and *woodwind instruments*.

- **Woodwind instruments:** these instruments used to be made of wood, nowadays they are not strictly made of wood. They are played by blowing air through their mouthpiece and can changes the frequency by opening or closing the holes they have. These family includes instruments such as the flute, the clarinet, and the oboe.
- **Brass instruments:** they got their name because they were made of brass and they are still made by it. The sound is usually made by the vibration of the musician's lips and they tend to end in a bell-like form. These family includes instruments such as the trumpet, the cornet and, the bugle.

A concept that is important in order to understand the differences between instruments even in the same family is *tuning*, tuning can be referred to the act of adjust an instrument so the frequency of a note corresponds to the theoretical frequency. For example, the trumpet has tubes that can be adjusted to make the total length of all the trumpet shorter or larger,

3. STATE OF ART

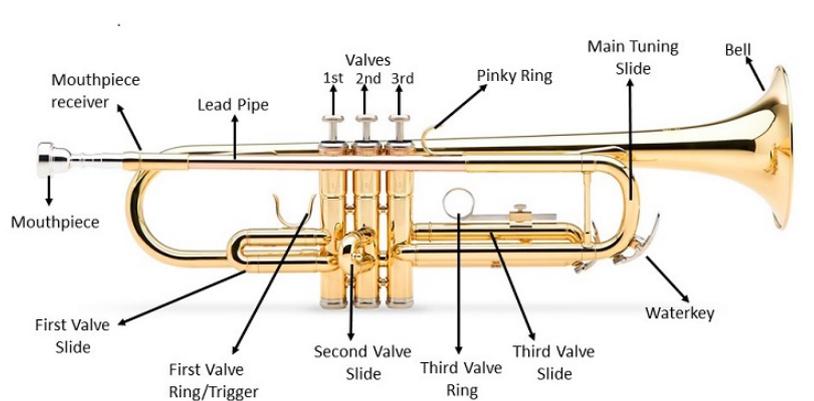


Figure 3.6: Trumpet parts

changing the length of the trumpet changes the frequency. The act of tuning is made by trying to play the standard pitch LA 4 (440Hz) and adjusting the tubes until the frequency played is 440 Hz or similar.

3.1.3 Trumpet practice

To understand how an instrument is played and how musicians practice with them it is useful to focus the attention in only one instrument and then extrapolate it to the others. The case of study will be the trumpet.

The trumpet is a wind instrument, in particular a brass one that has 3 piston valves that are used to open or close some tubes of the trumpet in order to modify the sound (see figure 3.6¹).

The trumpet has a large history, it is considered that the first trumpets were made 3.500 years ago. However, they were very different from the modern trumpets, they didn't have valve pistons and had only one tube. We could say that modern trumpets started around the year 1820, they had valve pistons and were very similar with the ones nowadays.

There are different types of trumpets, for example *bass trumpet* which is the largest type of trumpet or the *piccolo trumpet* that is the smallest one. However, the most common ones are the *SIB trumpet*. They are called SIB trumpets because their tuning is SIB, this tuning is 1 tone below the standard tuning which is DO.

SIB Trumpets are common in bands and some orchestras, while others use DO trumpets. However, they are usually the first ones that a musician learns to play.

¹<https://allmusicalinstruments.net/trumpet-for-beginners/trumpet-parts/>

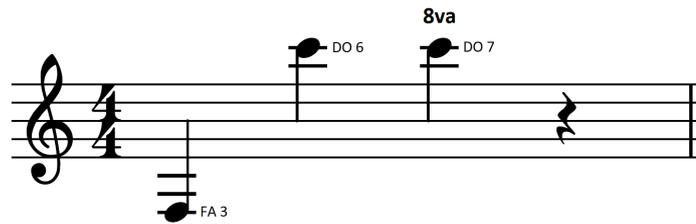


Figure 3.7: Trumpet tessitura

Any wind instrument has a range of notes that are comfortable to play, this means that even if it is possible to play notes below and above this range the sound will tend to be more difficult to control and composers prefer to use other instruments to play this notes. This range is called *tessitura* and for the trumpet the standard one is between FA 3 and DO 7 as we can observe in Figure3.7.

Exercises that are used to practice with the trumpet usually have notes inside its tessitura, so in order to improve with the trumpet and any other instrument musicians practice to increase the range of notes they can play inside the tessitura of their instrument.

As any other brass instrument, the trumpet put a considerable workload in the lips of a musician, this particularity makes necessary to train the resistance and the strength of the lips, for trumpet players is common to use exercises with long notes for that purpose. It is also common to play scale exercises to improve the quality of the sound, trying to play the exact frequency for that note and the right strength and duration in order to make a clean sound and there are other exercises that improve the skills with the piston valves that make possible to play more different notes in a shorter period of time.

Despite having only 3 piston valves which means that there are only 9 possible combinations the trumpet is able to produce every note inside its tessitura, therefore it is possible to produce more than 40 notes. Depending on the position of the lips, the sound of a trumpet will vary, when the lips are closer the pitch will increase. For example, DO 4 and SOL 4 are 2 different notes that in the trumpet are played with no valves pressed. One of the main drawbacks of this characteristic is that students that are beginners with the trumpet find difficult to learn all the positions of the notes.

Brass instruments have a lot of tubes that are used to modify the total length of the instrument, this is a relevant difference with woodwinds instruments. For instance, despite the trumpet is relatively small it has a total length of 2 meters taking into account every tube.

For beginners, there are a lot of manuals that explain how to play the trumpet. One of the most popular ones is the Louis Maggio system for brass [MM85], this book has been taking



Figure 3.8: Do Mayor Scale

as a reference in academies for decades and it is still used.

3.1.4 Basic exercises to practice with wind instruments

Learning to play an instrument is not just practicing famous music sheets or the ones that are going to be played in a concert. Musicians expend a huge amount of time on practicing exercises to improve skills like velocity or the quality of the sound they are able to produce.

There are specific exercises for brass instruments and for woodwind instrument. however, brass instruments can be divided in 3 different types: exercises to improve the quality of the sound produced (intensity, proximity to the perfect frequency), exercises to improve the lung's capacity and exercises to improve velocity (rapid notes).

- **Scale exercises:** a scale is a sequence of ordered notes, they are ordered by ascendant pitch and there are different types of scales depending on the distance between the notes, the most known types are mayor and minor scales. For example, mayor scales have this separation between notes: tone, tone, semitone, tone, tone, tone, semitone for example in Figure 3.8
- **Long notes exercises:** this is a type of exercise that is designed to improve the resistance of the lungs. It is recommended to practice between 5-15 minutes with this type of exercises but more would be unfavorable for the musician. The notes are usually at least semibreve for example in Figure 3.9.
- **Rapid exercises:** this type of exercise is designed to improve the skills with the fingers or the hands in order to be able to play more shorter notes like semiquavers or even shorter notes. Depending on the instrument the importance of these exercises varies. In the case of the trumpet, practicing with this exercises makes easier to change the position of the piston valves improving the sound of each short note, for example in Figure 3.10.

Depending on the instrument the importance of each type is a bit different, there are some instruments that require more velocity like the trumpet or the clarinet, while others require a higher resistance, for example the tuba. Even if the exercise type is the same, they could look totally different because of each tessitura, for some instruments is common to have



Figure 3.9: Long notes exercise



Figure 3.10: Rapid notes exercises.

notes with high frequencies, however, others have notes with low frequencies.

3.2 Gamification in education

The concept of gamification can be defined as *the use of game design elements in non-game contexts* [Gro12]. Gamification is not a modern concern; it has been used along the human history in order to teach children how to do some daily task or even for adults. The purpose of using gamification in repetitive tasks is to make people more motivated while they are doing those tasks.

3.2.1 Basic concepts

Although the idea of making non-game activities look like games is not a recent concern, the term *gamification* is a recent one. The term was adopted in the scientific community, in a general way in 2010. Since then, gamification has become a new field of research, there are important articles about the benefits of gamification [KC16] and depending on the author it's possible to observe different components that gamification should have. However, it is accepted that these are the essential components that have to be taken into account to apply gamification:

- **Accomplishment:** setting objectives for the players to give a reason to play the game. This will make people feel accomplishment when they achieve that goal.
- **Rules:** rules are an important part of any game, they specify what the player can not do and what actions the user is able to do in order to complete a goal.
- **Feedback:** the players should have feedback about their progress of reaching a goal. This could be made by sending messages and informing the players of their actual progress.
- **Rewards:** give rewards when the user puts effort and achieves a goal, the main purpose of giving rewards is to increase the motivation of the users.

3. STATE OF ART

- **Motivation:** there are two types of motivation, one is the extrinsic motivation which refer to the motivation given by external sources. For example, rewards and intrinsic motivation that is related with the people desires of getting new things and achieving difficult goals.

Gamification has a psychological aspect; it tries to motive users and in order to achieve that its necessary to understand how motivation works. There are multiple articles or books that are taken as a reference in order to understand motivation, one of the most famous one is *A Dynamic Theory of Human Motivation* [Mas58], according to this article, the physical, psychological and affective necessities are the motivations of people. Taking into account these theories, gamification techniques are created. Even if there are countless techniques and possible ways to apply gamification, these are some of the most popular ones:

- **Leaderboards:** it allows users to compare each other in some way and increase motivation by appealing to the competitive feelings of people.
- **User statistics:** allows the users to check their progress in the application. It is a good idea to incorporate graphical elements to make it easier for the user
- **Virtual economy:** giving ways for the users to get coins and create an economy based on them inside the game.
- **Build a community:** it offers to the users the opportunity of share their progress, discuss with other users and improve themselves together in the application.
- **Narrative:** a relevant part in some games is the story behind them, this element helps the user to get involve in the game.

It is important to understand that gamification is not only to add some features like goals or rewards to an existing application or activity. Gamification techniques are applied when a non-game activity is thought as a game, this means that gamification is applied in the design state of the activity and not as an extra feature.

3.2.2 Applications of Gamification

Gamification can be applied in different domains, it is not something that is only applied in learning, there are a lot of companies that are applying gamification for their products or even in their business processes in order to benefit from the advantages of gamification.

- **Product gamification:** gamification can be used to make a product more attractive for the end user, companies could fail on making an attractive product for the public even if the functionality of the product is quite good. For example, *The Nike + Run Club*² is an application of Nike that offers to runners the possibility to join a community of runners and share their achievements between them.

²<https://www.nike.com/es/nrc-app>



Figure 3.11: Starbucks reward app

- **Work gamification:** using gamification techniques in a work environment could improve the motivation of the worker, the aim is to improve the productivity by giving positive feedback to the worker and making their daily tasks less repetitive. One good example could be the US Army, they create games as a way of training their pilots and other soldiers.
- **Marketing gamification:** original marketing campaigns is what a lot of companies are searching for. There are some companies for example Starbucks that used gamification techniques in the application *My Starbucks Reward* (see Figure 3.11)³
- **Health:** treatments in healthcare can be tedious for patients, using gamification could help patients to have more motivation. It is also used for young and old people, it is easier for them remember the rules of a game than the steps of a medical treatment. For example, mango Health⁴ (see Figure 3.12) developed a mobile application that provide information about the medication they are taking like side effects in order to avoid dangerous situation and reminds them when to take their medication.
- **Gamification of learning:** one of the main concerns of education is how to increase the motivation and the interest of the students for a subject. Gamification is used in education to get these things. An example of gamification of learning is the application called *Duolingo*⁵, it is an application to learn foreign languages that uses the principles of gamification.

³<https://www.starbucks.com/rewards/>

⁴<https://www.mangohealth.com/>

⁵<https://www.duolingo.com/>

3. STATE OF ART

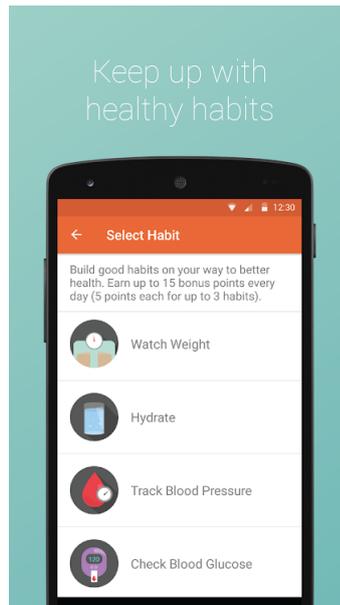


Figure 3.12: mango health app

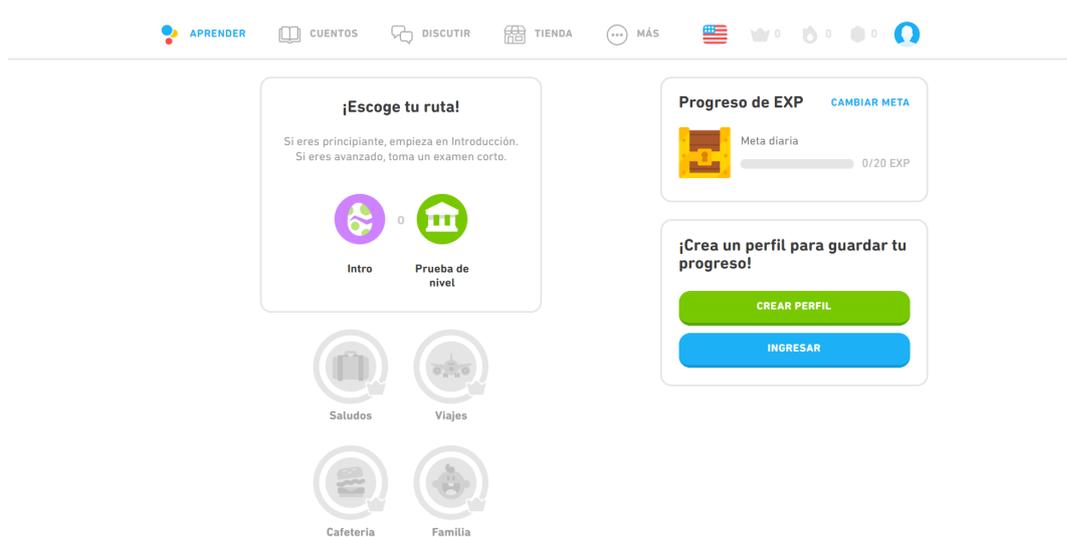


Figure 3.13: Duolingo web app

It is argued that the high percentage of students that fail to graduate could be decreased applying gamification in schools and universities, even if gamification of learning was tested with great results in the past, it is not until the appearance of computers and Internet that gamification has become one of the most popular ways of learning.

Gamification of learning is having a huge impact on the way of learning nowadays, this impact can not be understood without taking into account E-Learning. E-learning was a great innovation in education but had an important drawback, it was lacking emotional interactions on the users. As it is mentioned before, one of the essential parts of gamification is motivation and have a psychological effect on the user, using gamification in E-learning is a possible solution to the lack of emotions in E-learning.

3.3 Technology and music

Technology has changed the world and due to this change music is so different now, new types of music genres have been created but talking about classical music it has not changed too much at least on the surface. If someone thinks about classical music it will come to their mind orchestras. Is there any difference that people without musical knowledge can guess between orchestras 50 years ago and now? Nowadays what is changing a bit is the way we learn music, it is just not go to class and practice at home, technology can be used to help to practice or search for some theoretical information.

3.3.1 Relationship between music and video games

The relationship between music and video games is a very close one, a good soundtrack can improve the overall quality of the game. Music can be used to transmit feelings to the user and improve the immersive experience of the user. Imagine a horror game without a soundtrack or a soundtrack that can not help you to feel as closer as possible to the game.

This type of music is called background music, it refers to all the sounds that appear in a video game. There are several studies that explain the relationship between background music and the success of video games [Fu15]. From the perspective of this studies it can be concluded that video games that take the most from background music can cause a more immersive experience in players. Every sound in the game can be used as a complement of the action inside the game. The challenge in this aspect is to link in a consistent way the music and the action of the game, that is the reason why there are soundtracks designed for a concrete game.

The importance of music in video games is huge, there are several examples of video games that were independently developed and became very famous for giving a great user experience and one of the reasons was the soundtrack. For example, the video game called *Bastion*⁶

Even if the relationship explained above is the principle one, there are other ways of using music in video games. There are some video games that has music as the core of the game and players have to interact with the music. At this moment, music stops to be an added feature to improve the overall of the game and becomes part of the game-mechanics of the video game.

It can be observed that there are known games that actually use music as the principle part of the game, for example *Patapon 2* were the user has to command a civilization using the principles of rhythm that are the same like in every music sheet.

Even tho, these games have music as their core they are not made aiming to teach music or practice music, games that are made intending to teach music or to practice it are less known,

⁶<https://store.steampowered.com/app/107100/Bastion/>

the main reason could be that not everybody is so interested in music, the majority of this games are intended to learn how to play piano. The piano is usually the second instrument that every musician learns and there are many applications that teach how to play piano.

3.3.2 Mobile applications for wind instruments

It is possible to find apps in the Play Store which are used to learn how to play instruments, most of them focus the attention on teaching the correct position of the fingers in order to play a particular note, for example *How to play Trumpet*⁷ in 3.15. Which is an app that allow you to learn how to play several notes by using the piano as a reference. However, the position of the piston valves in a trumpet is not enough to determine a musical note. There are some notes that are played using the same valves but changing the position of the lips.

Another application which tries to solve this problem is called *canciones de trompeta - aprender a tocar*⁸. This teaches how to play the trumpet by letting you use 3 buttons as pistons and to simulate the position of the lips while playing a music sheet. It has 5 options ordered by intensity so for example the app will ask you to reproduce a bass sound you will have to choose which pistons are needed and, in this case, a low intensity. Even if this solves the problem, it stills looks unnatural even to ones that have played the trumpet for years. However, apps that check the sound of the trumpet in order to determine if it is played correctly are so limited. One example of this kind of app can be *tonestro*⁹ (see Figure 3.14) which let you select the instrument you play and while you play one of the several music sheets it has the app, it will check if the sound is the correct one. Even if this is quite good to learn when you have a trumpet it has some drawbacks, because it is difficult to determine some variations like linking a note.

In addition, there are applications that focus their attention in ear training, musicians needs to train their ears in order to identify the different pitches produced by an instrument just listening it. The traditional way of practicing this skill was going to a music class and the teacher would play notes and the students would try to guess what pitch was, this type of exercise is called melodic dictation. For example the *Troubadour: A Gamified e-Learning Platform for Ear Training* [PVŠ⁺20] use gamification techniques in order to motivate the students to practice these exercises.

These applications have something in common, they use gamification principles. We can observe that these applications give feedback of the actions that the user takes and there is a clear goal which is playing the correct notes. In the case of *tonestro*, there are also rewards when the user play well enough a music sheet and it unlocks other music sheets.

⁷https://play.google.com/store/apps/details?id=com.spintec.android.HowToPlay.Trumpet&hl=es_419

⁸https://play.google.com/store/apps/details?id=com.learnmaster.trumpet.songs&hl=es_419

⁹<https://www.tonestro.com/>



Figure 3.14: Tonestro app

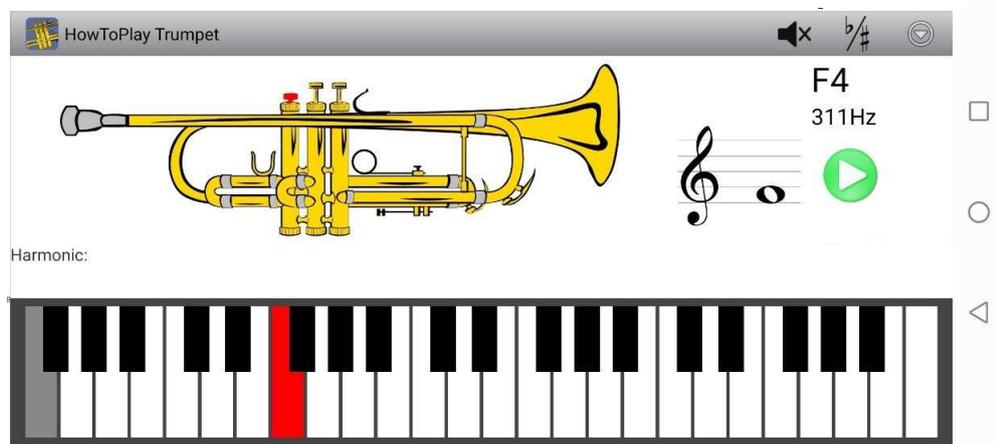


Figure 3.15: How to play Trumpet app



Figure 3.16: Canciones de trompeta

Chapter 4

Methodology

This chapter will explain which methodology is used and the variations needed for this particular project. In addition, the resources used during the development of this project will be detailed.

4.1 Software development methodology

In order to develop this project, the main ideas of *Extreme Programming* [Jos08] have been followed. It may be difficult to apply pure extreme programming due to special characteristics of an undergraduate project. For example, pair programming is not an option with just one member in the development team. However, the general characteristics where extreme programming is appropriate fits perfectly with the characteristics of the project. One of this characteristics would be dynamically changing software requirements which was described as an important characteristic of this project as well. Another would be having a small, co-located extended development team which is the case of this project.

4.1.1 Why Extreme programming

Choosing a methodology is a critical part of any software project. The first step is to decide if it needs a traditional methodology or an agile methodology.

For this project, some traditional methodologies were considered. For example, OPenUP¹ (Open Unified Process). However, traditional methodologies are not appropriate for this project. One of the problems would be that traditional methodologies are rigid and not appropriated for projects with changing software requirements, another difficulty in this context would be the less user involvement in the project process that is a risk in this project.

Agile methodologies as extreme programming follows the 4 principles of the **agile manifesto**². Agile methodologies are more flexible and put more attention in the interaction with the user than tradition ones. This project has some characteristics like uncertainty which are handled better with agile methodologies. In addition, changing requirements will be common in this protect so it is appropriate to base the project in short iterations with agile

¹<http://www.utm.mx/~caff/doc/OpenUPWeb/index.htm>

²<https://agilemanifesto.org/iso/es/manifesto.html>

4. METHODOLOGY

methodologies. Another relevant point is that agile methodologies promote a fluent communication with the client and this will allow us to refine requirements and make changes every iteration.

Between agile methodologies as adaptive software development, feature-driven development or extreme programming. The last one was chosen due to its high adaptability and its management of changes. Extreme programming is though for many experts as one of the best agile methodologies for adapting to changes as we can observe in the article of Kent Beck *Embracing change with extreme programming* [Bec99].

4.1.2 The application of Extreme Programming

Extreme programming is based on values. It is possible to add more values but are going to focus our attention on the principle ones.

- **Simplicity.** Do what is needed and asked for, but no more. This principle is unquestionable in almost every agile development and of course this case is not an exception, the time is important and it can not be lost in modules that are not necessary and the client doesn't need it.
- **Communication.** Everyone is part of the team and we communicate face to face. This principle is applied not only inside the development team, that has only one member, this principle is applied to any stakeholder.
- **Respect.** Everyone gives and feels the respect they deserve as a valued team member. In our case is not only inside the development team, the customer has to be like a valued member.
- **Feedback.** Take every iteration commitment seriously. The customer must know which is the state of the software at the end of every iteration.
- **Courage.** We will tell the truth about progress and estimates. Trying to lie about the progress will make it even worse. It is not critical fail one user story in a certain iteration we should just adapt to changes and keeping a good relationship with the client.

Taking into account the principles above a set of rules are defined [Bec99]:

- **Fix Extreme programming when it breaks.** It is a good idea to follow the basic rules, however, it is possible to change whatever does not work. For example, in this project was needed to adapt or ignore some rules like pair programming.
- **Iterations and user stories.** Iterations in extreme programming should be between 1 week and 3 weeks. It is supposed that 1 week is the best choice even though it seems short. The iterations in this project have a duration of 2 weeks, this is understandable because of having just 1 member in the development team. At the end of each

iteration a review meeting takes places with the client, in this meeting the customer have to decide if the user stories are completed or not and choose with the help of the development team what user stories will be done in the next iteration.

- **The customer is always available.** The customer's work is not only to help the development team, he should be integrated. In this case the customer will be an important part in the project.
- **Pair programming.** Even though pair programming [HDAS09] is a rule in extreme programming, it will be impossible because of having just 1 person in the development team. This rule and the other rules that implies more than 1 member in the development team will be omitted but without forgoing the principles of Extreme programming that are related with that rules.

4.2 Hardware and Software Resources

In this section the different resources used during the development of the project are listed and detailed.

4.2.1 Hardware resources

- **Graphic card:** it is a *NVIDIA GeForce GTX 1060*³, this graphic card provides a high performance in graphical processing and it is commonly used in gaming computers
- **Tablet Galaxy Tab A(2016)**⁴: since the development of the system was oriented to tablet devices, it was necessary to use a tablet to test the system and this was the selected one.
- **Samsom USB Microphone:** it is a microphone provided by Furious Koalas Interactive⁵ used in the development of the recording part of the application.
- **MSI- Gs63 stealth 8re computer:** it is one of the best gaming laptops in the market. It has a 16 GB Ram memory that was essential during the development.
- **Trumpet:** even if it is not a real hardware, it is an element that it is important to be considered as a resource of this project. The trumpet was essential to test the correct functioning of the application.

4.2.2 Software resources

- **Textstudio**⁶: it is an open source environment which work as an editor for /Latex documents. This software was used during the documentation of the project.

³<https://www.nvidia.com/en-sg/geforce/products/10series/geforce-gtx-1060/>

⁴<https://www.samsung.com/es/tablets/galaxy-tab-a-10-1-2026-t580/SM-T580NZKEPHE/>

⁵<https://www.furiouskoalas.com/>

⁶<https://www.texstudio.org/>

4. METHODOLOGY

- **Overleaf**: it is an cloud-based /Latex editor,this software was used to save the work done in the Textstudio editor.
- **Unity**⁷: Unity is a game engine developed by Unity Technologies, it gives to users the opportunity of creating 2D and 3D games. One of the main advantages of this game engine is that it supports about 30 platforms such as Android, IOS or Windows.
- **Github**⁸: it is an software that provides online hosting for software development version control, it operates with Git a version control system that takes track of changes in a software project. It was used to save the software of the project.
- **Visual Studio Code**⁹: it is a free source code editor made by Microsoft, it may be used with different programming language like C, PhP or JavaScript. The main use in this project was for programming in C and it was configured as the main code editor for Unity scripts.
- **draw.io**¹⁰: it is a diagramming application, it allow users to create and share diagrams, it has support form UML and for useful types of diagrams that were used to designed the application. In addition, some of the diagrams created with this tool are showed in this document.
- **JabRef**¹¹: it is an open source reference management software, its main purpose in this project was to handle the different bibliographic references using BibLaTeX.

⁷<https://unity.com/es>

⁸<https://github.com/>

⁹<https://code.visualstudio.com/>

¹⁰<https://app.diagrams.net/>

¹¹<https://www.jabref.org/>

Chapter 5

Architecture

This chapter presents a detailed analysis of the different parts of the system. A top-down approach will be followed in order to discuss the architectural decisions that has been made. Initially, an overview of the system will be introduced and explained in order to better understand the system as a whole. Afterwards, each part of the architecture will be explained in detail.

5.1 Architecture overview

The architecture of the system follows a layered-based approach (see Figure 5.1). This type of model allows the creation of a scalable system, which is divided into several independent layers. As a result, each layer offers different functionality, so that if a change is necessary in one layer that change would only affect the components of that layer, given that the interface remains the same. In addition, this type of architecture increases the system adaptability.

The structure of the system will be divided into three layers: *graphical user interface layer*, *tracking and control layer*, and *persistence layer*. Each layer will be divided into multiple functional modules, and each module will perform well-defined tasks.

- **Graphical user interface layer.** It is in charge of the graphical part of the application and the interaction of the user with the application. In addition, it controls the gamification-related functionality of the system, which represents a key feature in the developed application. The GUI layer is composed of two different modules: i) the *menu module*, which is in charge of controlling the different menus of the application that allow the user to register, login and select a profile, and ii) the *exercise module*, which manages the information about the exercise that the user is playing every moment.
- **Tracking and control layer.** Its responsibility is to control the information that the users produce in the GUI layer. If the user interacts with the application through the GUI layer, then the *tracking and control layer* will be in charge of transforming that information into structures that are understandable in the domain of the application. The information handled in this layer could vary between information about the user

5. ARCHITECTURE

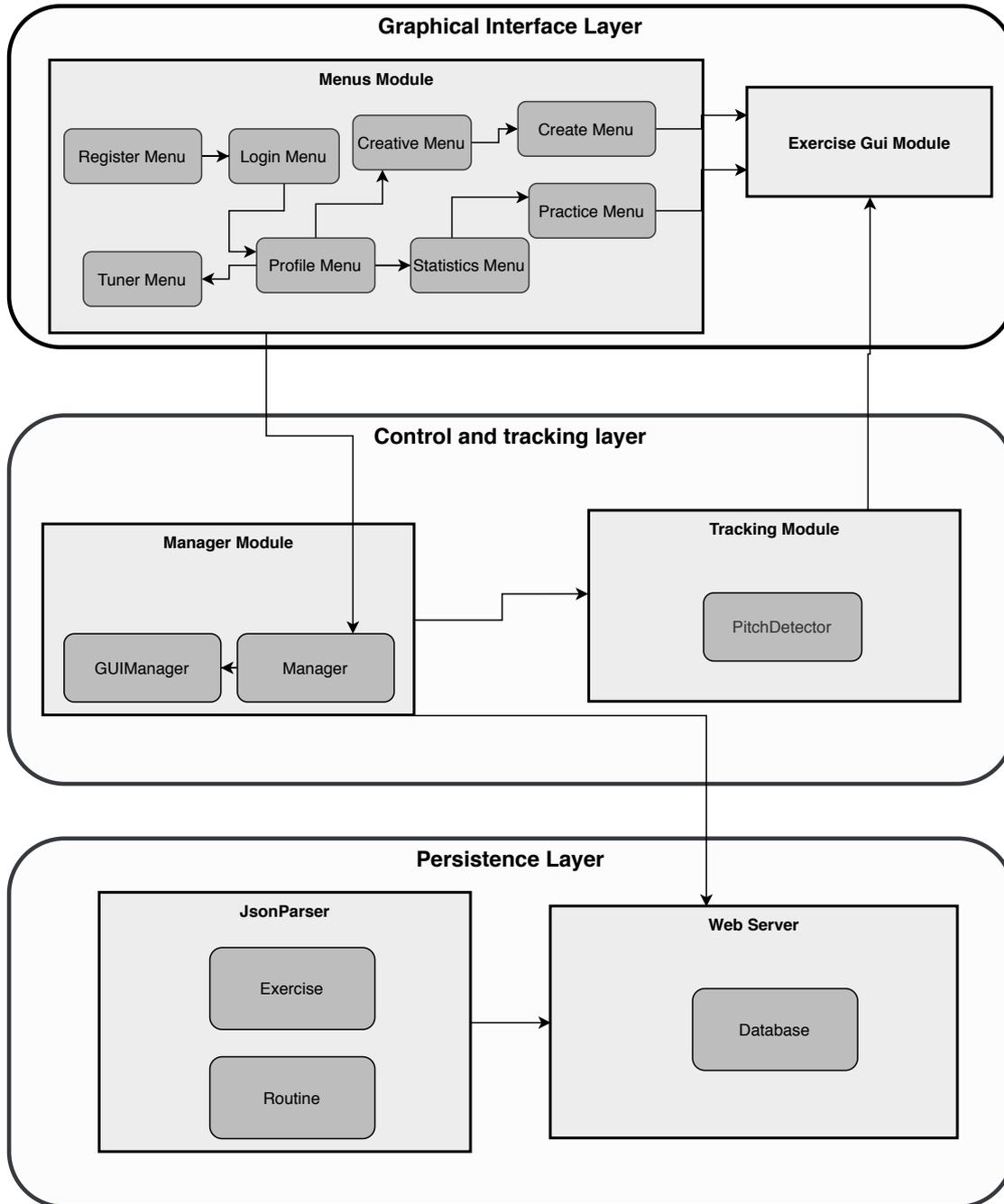


Figure 5.1: General architecture of the system.

and frequencies produced by an instrument. This layer is divided into two modules: i) the *manager module*, which is in charge of the information related with the menus, and ii) the *tracking module*, which is responsible for the information related with sounds and frequencies produced when the user is practicing with an instrument.

- **Persistence layer.** It is in charge of processing and storing the data that the application needs about users, profiles, exercises, routines, and statistics. The proposed application

needs a permanent storage in order to save this information even when the user closes the application. It has 2 important parts: i) the *web server* that controls the permanent storage in the cloud and manages a database and ii) the *jsonparser* which is in charge of parsing JSON files into structures that the *persistence layer* can process.

5.2 Graphical user interface layer

The GUI layer will be responsible for managing the interaction of the user with the system, this layer will be in charge of the correct **visualization** of the different screens inside the application. In addition, it will be responsible for the representation of the music sheet that the user will use to practice an exercise.

Since this layer is in charge of visualization, it is necessary to take into account the **gamification** aspects of the application while designing the layer. Create an application which is not a game but looks like one is not an easy task. Every screen needs to be designed following several rules and without forgetting to provide a good user experience.

There were different possibilities in order to apply gamification techniques, it is possible to create a system using gamification techniques without any specific technology. However, another possibility is to work with technologies that are specific for creating games. In this case the decision was to use a **game engine** which is a software-development environment designed to build video games. The use of a game engine will facilitate the integration of gamification techniques in the application and it will make possible to control the execution of an exercise like if it was a level of a game. On the other hand, it is possible that the use of a game engine could make another aspects of the development more difficult due to the specific features that a game engine has. However, the benefits that come from using a game engine specially in the part of practicing an exercise overcome the possible drawbacks.

There are several game engines in the market that were suitable for the development of a project of these characteristics. For example, the *Unreal Engine*¹ and *Unity*. After studying the advantages and disadvantages of the two technologies the decision was to use Unity. This decision can be explained if we take into account that the development is oriented to Android devices in particular to tablet devices and Unity has the possibility of creating multi-platform applications.

The GUI layer is the one that interact the most with Unity features, in order to take the most from Unity, it was necessary to study and understand how it works and what are the main features used in the creation of video games to use them in the development of our application. According to the Unity Manual² **GameObject** is the most important concept in the Unity Editor. Every possible entity in a video game is a *GameObject* in Unity. For example, characters, lights, enemies, special effects and every object you can imagine. Considering

¹<https://www.unrealengine.com/en-US/>

²<https://docs.unity3d.com/2018.4/Documentation/Manual/GameObjects.html>

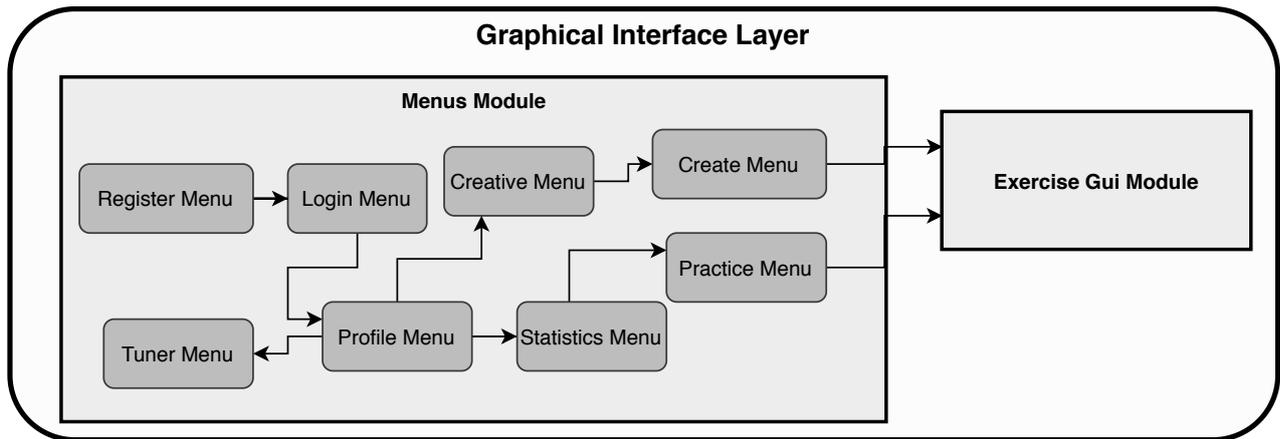


Figure 5.2: GUI Layer Architecture

the importance of *GameObjects* in Unity, the GUI of the application has to be designed using them, what's more, it has to be based on them, every screen in the GUI will be based on *GameObjects*, and every input, text or image will correspond to a *GameObject* in this system in order to follow the rules of interactions in Unity.

In addition, *GameObjects* can have scripts associated to them, these scripts are used to control everything about *GameObjects*, from its position in the screen to the several functionalities a *GameObject* has. In Unity scripts inherit from the class *Monobehaviour*, this class has special methods as *Start()* and *Update()*. The *Update()* method is called every frame in the application, it can be understood as a loop that is managing the events that happen to the *GameObject* associated. These methods will be vital in the implementation of the most important part of this layer which is the visualization of the music sheet.

As this layer will mainly result into several screens inside the application, it is interesting to analyze the flow of the application, in Figure 5.3 it is possible to observe the relationships between the different screens.

This layer will be divided in two modules that will take the responsibility of different tasks: the *Menu Module* and the *Exercise Module*

5.2.1 Menu Module

In order to create an application that will have different menus or screens that a user will use to interact with it, it is necessary to design and implement a solution that would allow the management and representation of the different screens involved in the application.

As it was exposed above, in this module the intention is to take the most from the *Unity Editor* in order to apply gamification techniques in the design of the different screens. The *Menu Module* is defined by the different scripts and *GameObjects* involved in the correct visualization of the different screens in the application. The idea was to create a separated

part of the application that would only manage the graphical part with the aim of improving the **scalability** of the system.

There were different options to create screens using the Unity features and applying gamification. One of the options was creating a **Unity Scene** for each screen in the application, according to the Unity Manual in each Unity Scene you place your environments, obstacles, and decorations, essentially designing and building your game in pieces. As a result, each Menu of the system would be defined by a Scene. However, the decision was to create a Unity Scene for the complete *menu module*.

This decision can be understood due to the specific features of a Unity Scene. Each Unity Scene can be understood as a level in a video game, for this reason Unity destroys each GameObject that were in the previous Scene. In this case, there are elements that have to be used in different menus. While there are some ways of preventing the destruction of GameObjects when changing a Scene, it was considered that it was a better option to use the same Scene for the different menus in order to facilitate the reuse of elements in different menus.

Even if the menus are in the same Scene, they are handled in a separated way. Each menu will be defined by a parent GameObject, a set of child GameObjects that will represent the different elements in a screen and a script that will handle the interaction of the user with that menu.

Scripts that handle a menu look similar, as we can observe in listing 5.1 each script will have some variables that correspond to each input, button or text in the screen and a set of GameObjects that correspond to the menus that can be reached from the one that handles script. In addition, the scripts will have methods to send and receive information from other layers, in this example the script have a Login() method to send the data from an user a receive a response.

```

1 public class LoginMenu : MonoBehaviour{
2     public TMP_InputField usernameField;
3     public TMP_InputField passwordField;
4     public TMP_Text errorText;
5     public Button loginButton;
6     public GameObject loginMenu;
7     public GameObject profileMenu;
8     public GameObject successfulPanel;
9     public GameObject registerMenu;

```

Code listing 5.1: Script associated to Login Menu

5.2.2 Exercise Module

The GUI layer does not only handle the visualization of menus where the users can check their information, it also manages the **representation of a music sheet** when users practice with their instrument. It was convenient to separate this module from the Menu module because the data and the way it is managed this screen was totally different.

5. ARCHITECTURE

As it has a totally different function from the other Menus it was necessary to create a specific module to handle the exercise practice of the application. In addition, as it was explained in the *menu module* Unity Scenes are used when working with different levels in a video game. As we are considering the *menu module* as a way of configuring the exercises that will be practiced by the user we could consider the exercise practice as a level of a game. As a consequence, the decision was to create a new Unity Scene that was a completely different environment in order to look like a level of a game. Even if this approach will make more difficult the relationship between the *menu module* and the *exercise module*, the advantage of this approach is that the system is taking the most from the technology we are using in order to integrate gamification techniques and create an application that looks like a game.

The *Exercise Module* is in charge of the correct visualization of the music sheet and the interaction with it. As we know, depending on the pitch of a note the position in the staff will change, what's more, depending on the duration of that note the shape of it also changes. As a result, the module will deal with the visualization of notes dynamically.

Another aspect to take into account in this module is the necessity of measuring the **time** in a music sheet, as we know, in order to play correctly a note it is necessary to start playing it in the correct moment. The *Exercise Module* needs to measure the time and calculate when the user has to play each note considering the velocity of the exercise.

In order to solve these problems, we decided to use another time the features that Unity provides. This module will have a set of GameObjects to control the execution of an exercise. The most important GameObjects in this module are the *ExerciseMenu*, which will handle everything related with the correct representation of the music sheet and the *MicrophoneDector* which will be in charge of recording any sound that is produced when this Module is active. The *ExerciseMenu* will represent the information received from the Manager Module, this information will be composed by the information about notes, exercises and time. In addition, it will interact with the tracking and control layer in order to check if the notes played are the correct ones. It will also highlight the incorrect notes that the user plays.

In addition, it was essential to measure the time accurately in order to create a good environment to practice music. The simplest option was to use the Update() method that Unity provides and check whether the note has to be played at that time or not. The Update() method is executed every frame in the application, the problem was that if the method executes every frame, it was possible to have an error measuring when a note had to be played. Tests were performed in order to check if the error was acceptable and the decision was to assume that error because it was negligible.

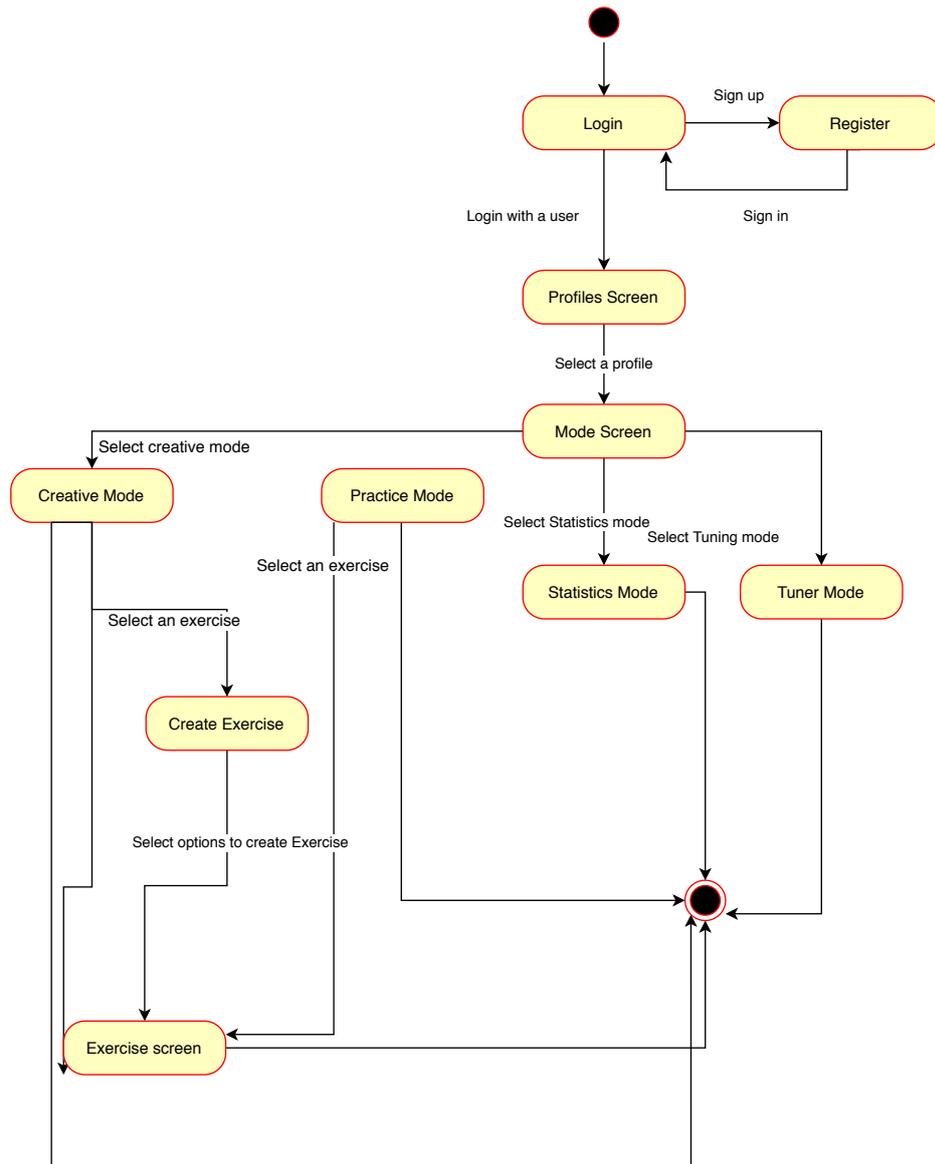


Figure 5.3: State diagram

5.3 Control and tracking layer

This layer (See Figure 5.4) is in the middle of the system and is in charge of **transforming the information** produced in the GUI layer into information that can be handled by the domain of the application. In addition, it will send to the Persistence layer what is the information that has to be permanently stored. It has 2 modules, *the Manager Module* which deals with the information from the Menu Module, and the *Tracking Module* which will be in charge of tracking the pitches of an instrument while the user is practicing.

5.3.1 Tracking Module

The tracking module will be responsible for the management of the audio input in the system. While users are playing with their instruments is necessary to handle that inform-

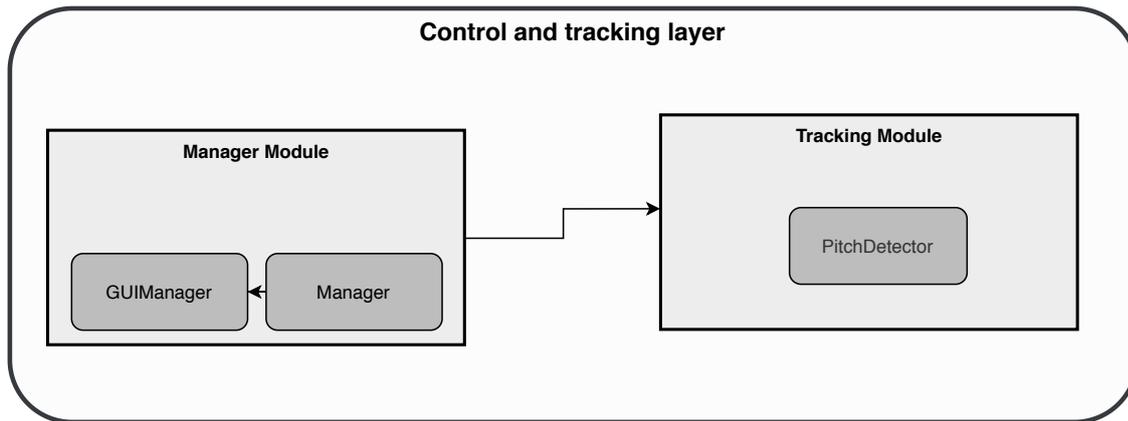


Figure 5.4: Control and Tacking layer

ation and transform it into numerical values in order to check if the notes that are played correspond with the ones in the music sheet displayed.

In this context, it was necessary to study the different options to record the sounds and extract their corresponding frequencies, one possibility was to create our own script that could perform this specific task. However, it was rejected due to the complexity of this kind of functions. Finally, the decision was to use an external library that could be integrated the architecture and could perform this task. The library selected was the *Human Voice Pitch Detector*, a library in the Unity store that uses the **RAPT Pitch Tracking algorithm** described in the paper *A Robust Algorithm for Pitch Tracking* [TK95]. It consists of a Cross-correlation function that gets the frequency that is currently recorded.

Tracking the different frequencies was another challenge. The problem was that if frequencies were measured over time and registered the result, the amount of data would be enormous and some of it would be just noise that would disturb the correct performance of the application. The solution was to take samples over a little amount of time and register the value that was repeated several times ignoring the noise, this approach was tested with music instruments and worked perfectly as it was intended. In addition, the implementation of this approach used another relevant feature of Unity that are **coroutines**. Coroutines are like a function that has the ability to pause execution and return control to Unity but then to continue where it left off on the following frame, this was the way of solving the problem of recording accurately the frequencies.

Moreover, it was also necessary to convert the frequencies into pitches, in order to make this possible the idea was to transform frequencies first to the **MIDI Note System**. MIDI was born as a standard which describes a protocol that allows different electronic instruments, computers and devices to connect and interact which each other. In this case, the relevant part is the *note system* inside the protocol that associates each possible pitch to a natural number. This notation was used in the application instead of pitches that would have made

the Module more complex.

Once the frequencies produced by an instrument are converted into the MIDI *note system*³(see Figure 5.5). It is possible to compare the Midi value with the values that the player wants to play. The Tracking Module will be in charge of this comparison as well as the conversion from MIDI *note system* into pitches that could be handled by the Exercise Module in the GUI layer.

5.3.2 Manager Module

As we explained in the GUI layer, scripts associated to menus will send and receive information in order to perform actions that the user has requested. The information sent and received needs some processing and calculations and this module will be in charge of **managing information**.

The necessity of this module could be explained considering that the architecture model that the system follows is a layered model. As it was explained before it is necessary that the layers perform different tasks and if something in any layer changes the others would stay unchangeable. It was possible to have this processing in the GUI layer. However, this situation would go against the philosophy of the layered model. In this situation, if a change in the GUI layer was needed it would be necessary to change the processing of the information. In contrast, creating a module of this characteristics in *the tracking and control layer* will make possible to change the GUI for a totally new one and still using the same processing of the information what leads into a more scalable system.

This module is divided into two components, the *Manager* and the *GUIManager*. They perform similar tasks, as they receive and transform the information of the *GUI* layer and send it to the *Persistence Layer*. Since they perform a similar task, it was reasonable to create just one component in order to control this information. At first, the application had only one component. However, the volume of information was growing, and it could be divided in two types. On the one hand, one type was the domain application information, which is the information related with user, exercises, routines and statistics. On the other hand, it was information that had to be calculated for the *Exercise menu*. These calculations are related with the position of notes, the tuning of the current instrument and graphical configurations related with the music sheet displayed. Thus, as a way of separating concepts and increasing the scalability, a division between these two managers was executed.

As it can be observed, in listing 5.2 the methods in the *manager* have the same structure, it receives information in this case about a user, then it consults the *Persistence layer* and returns a response to the *GUI layer*. In Figure 5.6 is possible to observe the flow of information that has to be controlled by the *manager module* in order to check the users credentials.

³<https://newt.phys.unsw.edu.au/jw/notes.html>

5. ARCHITECTURE

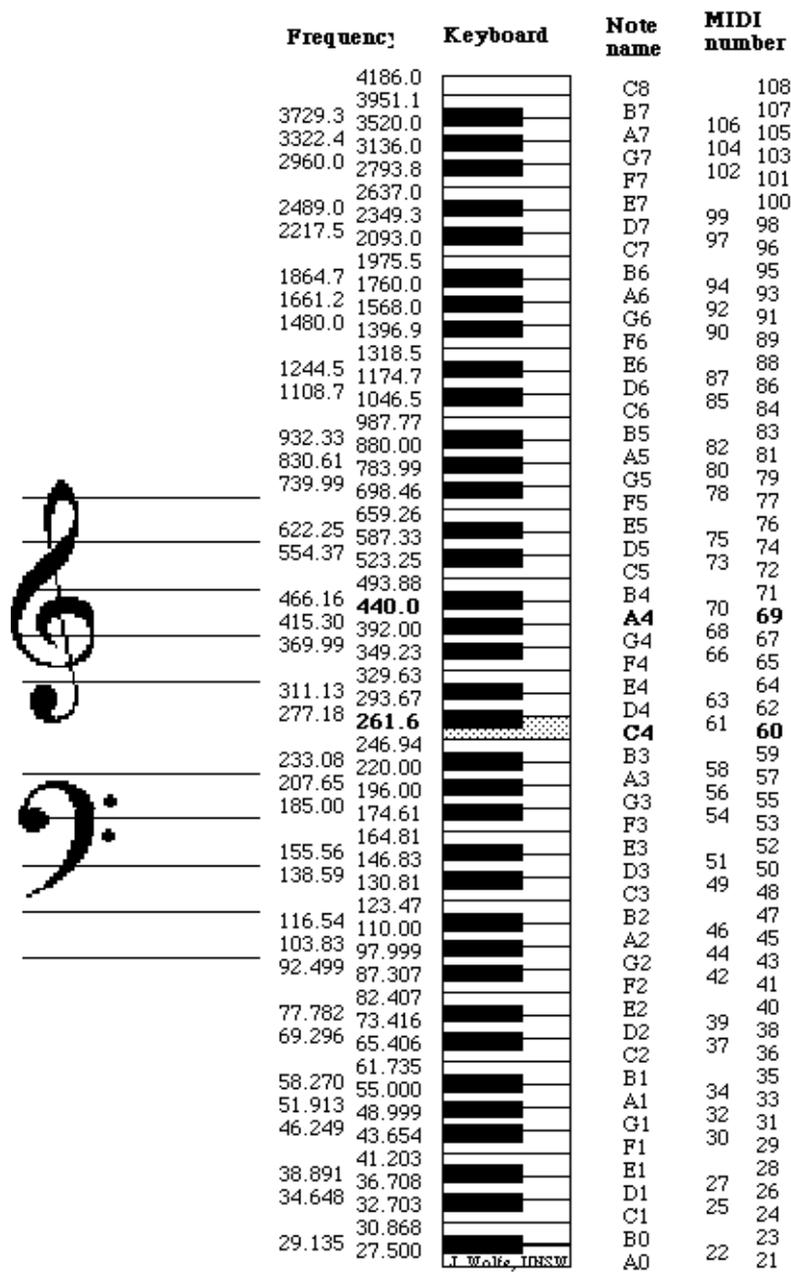


Figure 5.5: Midi to note

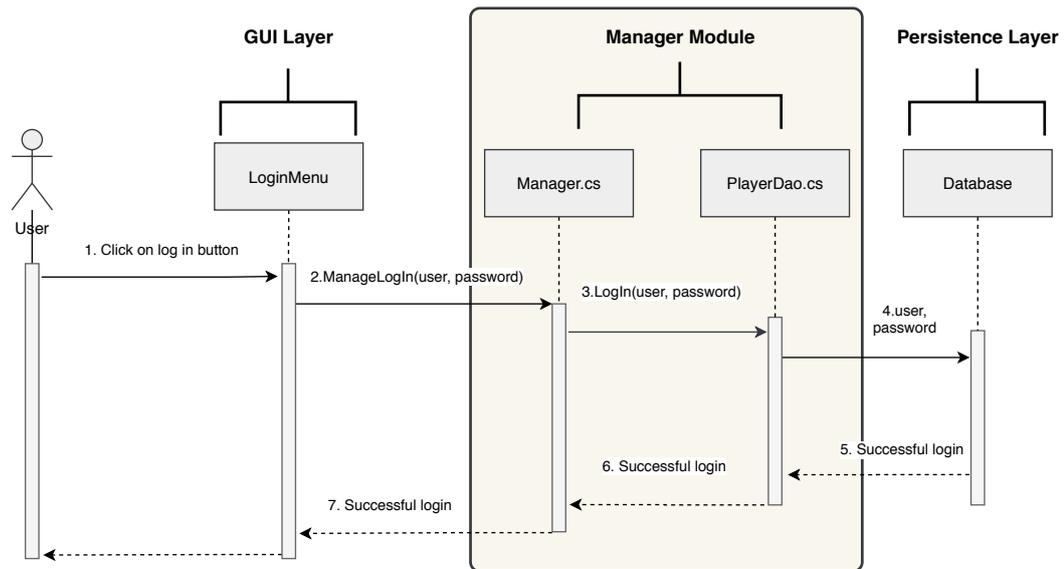


Figure 5.6: Example in the Manager Module

```

1  public string Login(string username, string password){
2      string response = "error";
3      Player loggedPlayer = null;
4      try{
5          loggedPlayer = this.daoPlayer.getPlayer_User_PasswordM(username, password);
6      }catch (Exception e ){
7          UnityEngine.Debug.Log("[LOGIN] Exception"+e.Message);
8          response = "Wrong credentials";
9      }
10     if(loggedPlayer == null){
11         response ="User not registered";
12     }else{
13         response = "User logged";
14         this.currentPlayer = loggedPlayer;
15     }
16     return response;
17 }

```

Code listing 5.2: Script associated to Login Menu

5.4 Persistence Layer

Any application in the market needs to **store data**, there are applications that only need to store data while it is working, in that case it is not necessary to use a permanent storage. However, our system needs to store data even when the user close the application, the reason is that the system stores the data about the progress of the users, credentials, exercises and this kind of information has to be **permanently available**. It would be possible to create a local permanent storage for the application. Nevertheless, it was proposed that having an online permanent storage to manage the information of every user could be more interesting for future work on analysis.

In addition, this layer will have another responsibility that is giving support to the creation of exercise using data in a file instead of using the screens of the application.

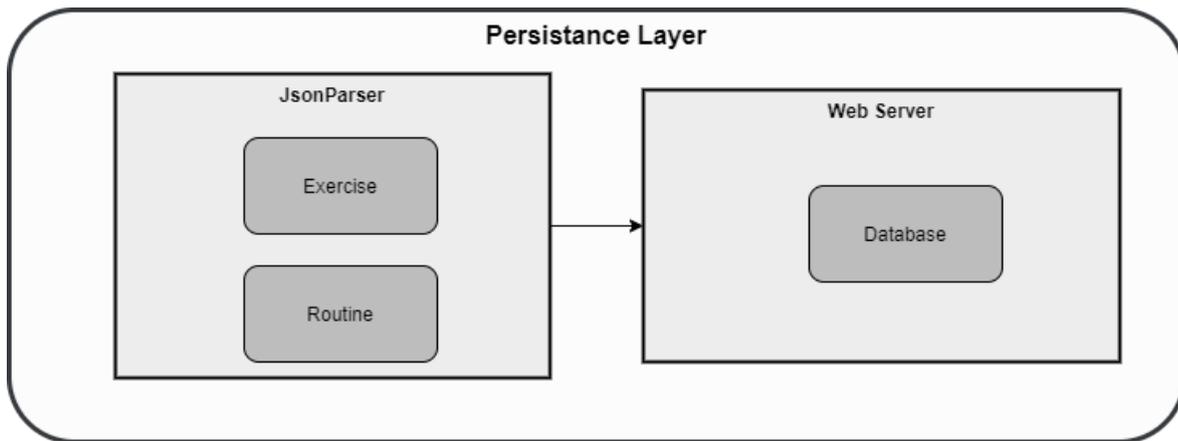


Figure 5.7: Database diagram

The Persistence Layer will have two components the *Web Server* that manages the permanent store in the cloud and the *JSONParser* that process JSON files.

5.4.1 JSONParser

Users will be able to create new exercises using the different menus of the application. However, the idea of **exchanging exercises** between users was thought; and there were different possibilities to accomplish this objective. Finally, we decided to add a module able to read files that contains the information about an exercise or a routine and add them into the system. Even if this functionality is not available for users at this moment it still have several parts fully implemented.

There were different options to represent an exercise in a file, one option would be using standard formats for exchange information about music sheets. The most spread format is MusicXML⁴ which was designed for sharing music sheet files between applications. Even though, this format is complex, and it has several features that would not be useful for our system. At the end, we decided to use JSON files that would have the same information that the user fills to create an exercise inside the application. Following this approach allows the users to understand the information inside these files and modify it.

JSON is a standard file format and a data-interchange format that was based on a subset of JavaScript. However, it is a language-independent format and nowadays is one of the most popular data-interchange formats in any application[HSN⁺14].

The *JsonParser* Module has a script associated which is called *JsonParser.cs* that is in charge of reading data about exercises (see Code 5.3) and transform it into information that the system can **store permanently**. In addition, The *JsonParser.cs* script is able to process routines as well, this feature was tough as a way to interchange routines between users. For example, a music teacher could create a routine and give the JSON file to the students in order

⁴<https://www.musicxml.com/>

to practice it.

```

1  {
2    "type": "Exercise",
3    "Exercise": {
4      "ExerciseType": "Scale",
5      "Scale": {
6        "beats": 60,
7        "tune": "Sib",
8        "timeSignature": "4/4",
9        "firstNote": "LA# 3",
10       "scaleType": "Mayor Scale",
11       "duration": 4
12     }
13   }
14 }

```

Code listing 5.3: Example of Exercise in JSON File

5.4.2 Web Server Module

As it was exposed above, the system needs a permanent storage, this module was created in order to handle the information that was going to be saved permanently in the system. The decision to solve this problem was to use a **database** where the data would be saved in a permanent way.

A database is a collection of structured information stored in a computer system [Dar09]. There are mainly two types of databases, *relational databases* and **NoSQL** databases. NoSQL databases store unstructured or semi-structured data while Relational databases store structured data and are table-based. In order to decide what type of database was the best option for the system it was necessary to take into account different aspects as performance, scalability and what type of data is going to be stored. The system will store exercises, information about players information and statistics, this type of information is already structured. As a result, the decision was to use a relational database.

Any relational database needs a schema, which is the organization of the data in the database. In this case the designed schema for the database handles the information about players, profiles, exercises and the relationships between them (see Figure 5.8).

Any database needs a Database Management System to work. There are different technologies in the market like Oracle Database⁵, MySQL⁶ or Sql Server⁷. The selected one for our system was MySQL that is an open-source relational database management system which is owned by Oracle.

Connecting the system to a database is easily performed when the connection is with a computer, the device can connect directly to the database with a driver. However, a mobile application like our system can not connect directly to a database.

One way to solve this problem is to use a web server, the mobile device will make HTTP

⁵<https://www.oracle.com/es/database/>

⁶<https://www.mysql.com/>

⁷<https://www.microsoft.com/es-es/sql-server/sql-server-downloads>

5. ARCHITECTURE

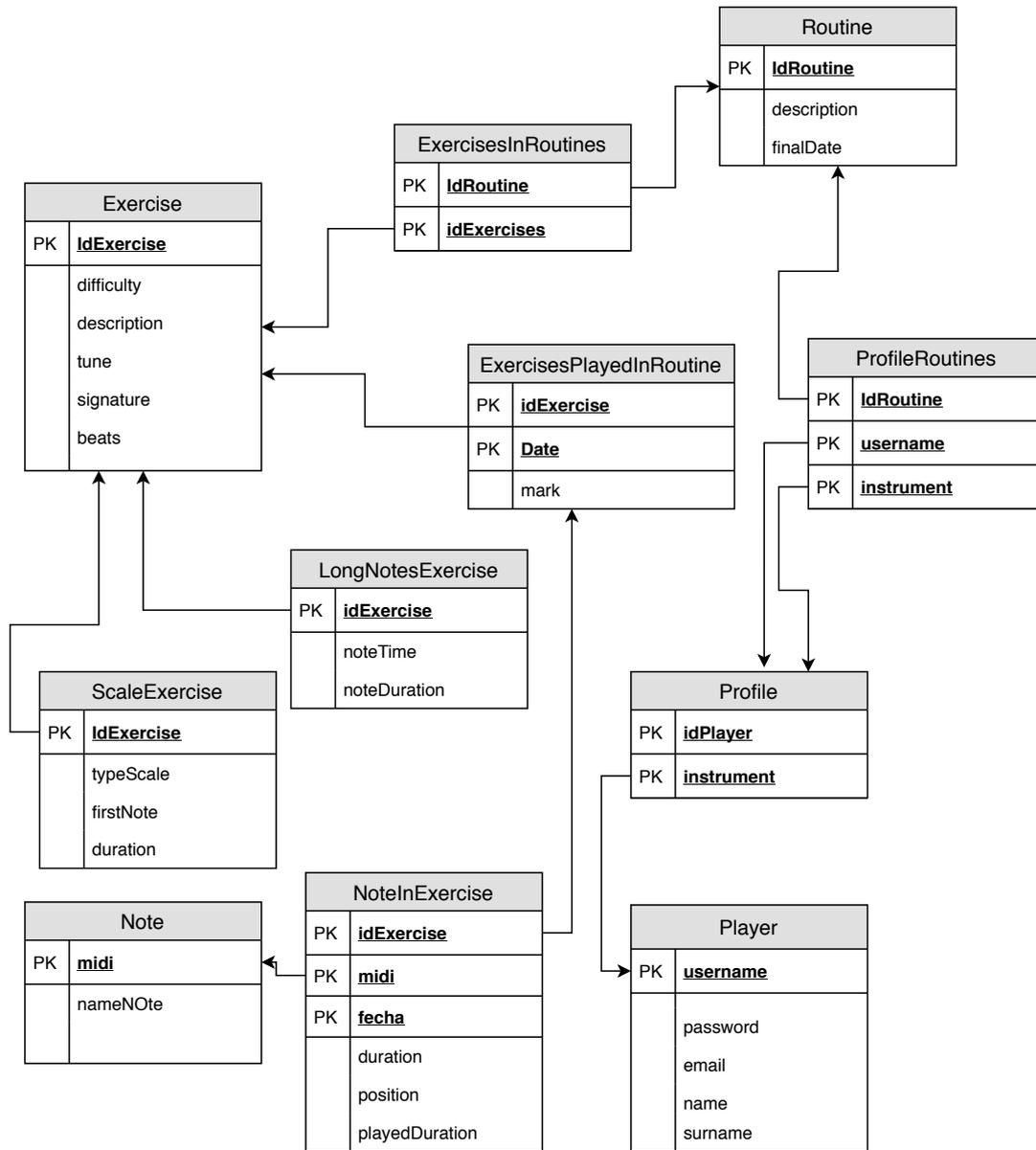


Figure 5.8: Database diagram

requests to the web server, then the server will connect to the database and execute the SQL Queries and finally it will return to the mobile device the requested data.

The technology used to handle the web server was *XAMPP*⁸, XAMPP is an open source software developed by Apache friends⁹ which works as a local server, its main functionality is to use it while developing the software in order to test the system without having to create an online server. Our web server will receive HTTP request to get the resources that are stored in the Database and will execute PHP code to execute the SQL Queries.

In order to understand better how the web server is able to execute different SQL sentences and retrieve information, this listing is an example (see Listing 5.4).

```

1  $sql = "Select * from Player Where username = '" . $loginUser . "'";
2  $result = $conn->query($sql);
3  if ($result->num_rows > 0) {
4      // output data of each row
5      while($row = $result->fetch_assoc()) {
6          if($row["password"] == $loginPass){
7              $sarr = array('username' => $row['username'],
8                  'name' => $row['name'], 'surname' => $row['surname'],
9                  'email' => $row['email'],
10                 'password' => $row['password']);
11             echo json_encode($sarr);
12         }else{
13             echo "Not correct password";
14         }
15     }
16 } else {
17     echo "No user with that name";
18 }
19 $conn->close();

```

Code listing 5.4: Example of Exercise in JSON File

5.5 Design patterns

A design pattrer is a general repeatable and proved solution of design problems that are common in the process of development software [FA12]. The idea behind a design pattern is to use solutions that are accepted and checked in other projects and adapt that solution to the system you are implementing. In the development of this project the convenience of using several design patters to increase the scalability of the system was discussed. In this context,the decision was to implement the pattern *factory method*,the pattern *model-view-controller* and the pattern *builder* to solve several problems that are detailed in the next subsections.

Factory method

This design pattern is a **creational pattern** and a simplification of the design pattern *abstract factory*. It provides an interface for creating different objects. This pattern is commonly used when it is necessary the creation of different objects that are similar to each other. The main advantage of this pattern is that it promotes the loose-coupling. In other

⁸<https://www.apachefriends.org/es/index.html>

⁹<https://www.apachefriends.org/download.html>

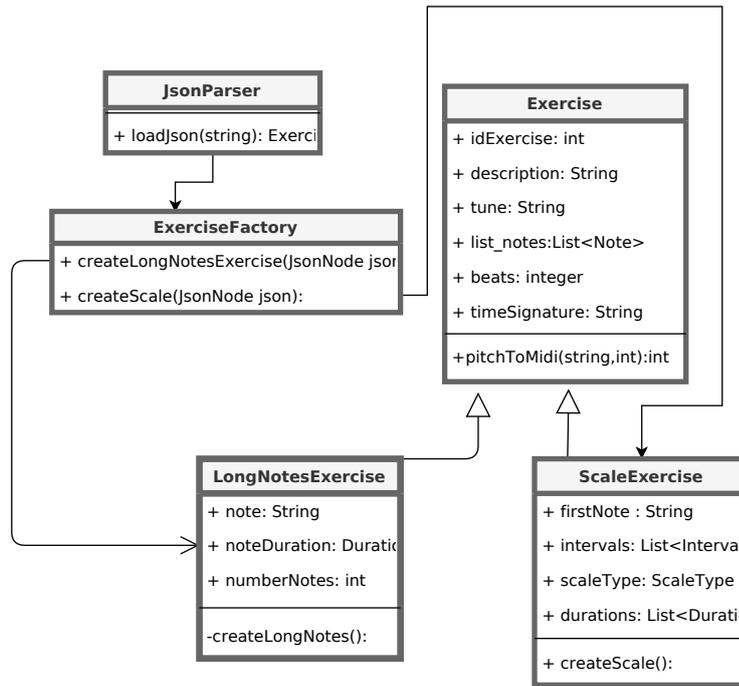


Figure 5.9: Factory method

words, this approach tries to have elements that depended on each other to the least extent. The idea is to interact only with the interface and not knowing anything more about the different objects that could be created.

As we explained before, the *JSONParser Module* will read files with information of different types of exercise and Routines. This information will be transformed into different types of exercises and routines. In order to prevent coupling, the factory method was used in this module. As a result, it will be easier to add new types of exercises into the system without changing other parts.

As we can observe in the Figure 5.9 the Class *JsonParser.cs* sends a JSON string with the information of different types of exercises and the class *ExerciseFactory.cs* is in charge of creating the different types of exercises.

Model view controller (MVC)

The design pattern *Model view controller* (MVC) (see Figure 5.10) is an **architectural pattern** that its main purpose is to isolate the domain of the application from the Graphical User Interface. This pattern is composed of three different parts which are the view, the controller and the model. The main benefits for this project are: i) Modifications only affect one of the parts of the pattern and ii) It is appropriate for video games that can have multiple GUIs.

As it can be noticed, the approach of the first benefit is similar to the benefits of using a layered architecture. This pattern fits perfectly in a layered architecture with 3 layers.

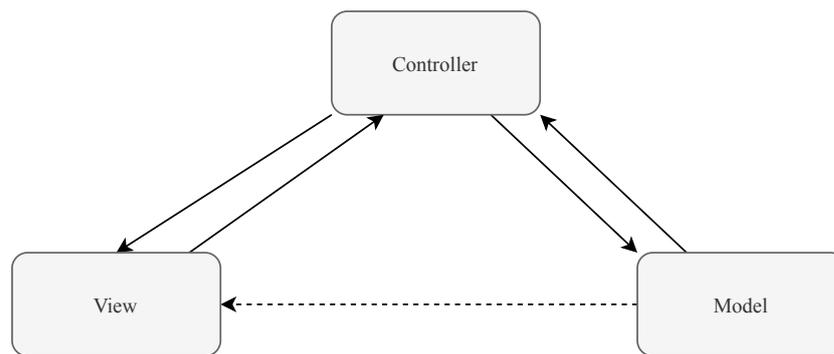


Figure 5.10: MVC design pattern

The idea was to relate each part of the design pattern with one of the layers (GUI layer -> View, control and tracking layer -> controller, and persistence layer -> model). Even if this relationship is not totally exact, it will give to the system a more defined separation between different layers.

Another consequence of using this design pattern is that it is possible to change the GUI depending on the client demands. It can be possible that the client would want to change almost everything in the graphical part of the application. In this case, using this pattern it would only be necessary to change the *View*, since the other 2 parts could stay unchangeable.

- **View:** it is the graphical user interface, the user interacts with this part of the system through buttons and inputs. In this concrete project the View will be closely related with the GUI layer. However, it is actually not the same thing.
- **Controller:** it receives the information from the view and translate it to the domain of the application. Most of the functions related with the controller will be implemented in the *Manager Module*.
- **Model:** the actions received in the model are independent from the view, it only works with structures of the domain of the application and it usually receives operation of type creation, update, consult or delete. In this application the model corresponds to the classes related with exercises and users.

Builder

This design pattern is a **creational pattern**. It provides a way of resolving the creation of different complex objects. It is important to notice that this pattern is very similar with the *factory method* pattern. However, even if they are both creational pattern, they are used in different contexts. The *factory method* pattern is used when it is necessary to create a family

5. ARCHITECTURE

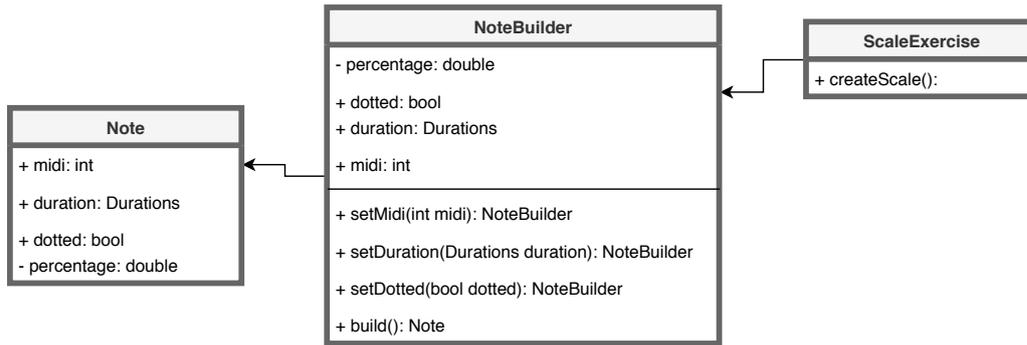


Figure 5.11: Builder design pattern

of objects with a hierarchy. For example, in this system there are different types of exercises, that is the reason why it was more accurate to apply the *factory method* in the example explained above. In contrast, when the objects to create do not have an hierarchy, the typical solution is to use the *Builder* pattern in order to simplify the creation of that objects.

The builder pattern encapsulates the creation and assembling of the different parts of an object. Then if a class needs to create an object it has to call the builder that will create that object instead of creating it directly. At first. Following this approach will make simpler the creation of complex object.

In this system, the builder pattern was used in the creation of *note objects*. In this Figure 5.11 it is possible to observe a diagram that explains how it is implemented this pattern in the system.

Chapter 6

Results

In this chapter the results obtained in the development of **Windy Melody** will be discussed. First, the work distribution during the development of the project will be detailed. Then, the cost and resources related with the development of the project will be explained. The source code of the project is available online on GitHub:

<https://github.com/xl0lux1>

6.1 Work distribution

As mentioned in the chapter 4, the chosen development methodology is extreme programming. It is a methodology based on short iterations. In this section, it will be discussed the different tasks developed in each iteration, at the end of each iteration the work has been validated with the tutor.

6.1.1 Iterations

As it is explained in chapter 4 the distribution of the project will be divided in short iterations. Each iteration will last between 2 or 3 weeks and they will have some user stories associated.

| 1# Initial Planning and study of technologies | |
|--|---------|
| User story | - |
| Duration | 2 weeks |
| Once the idea of creating a system that allow users to practice with their instruments was defined, a research about the different technologies that allow the creation of this kind of system was executed. Between the different technologies in the market as Unity and Unreal Engine, Unity was chosen as the motor engine for the application due to its facility to work with mobile technologies. It was also necessary to choose a technology capable of recording the sound and process it into frequencies, in this case the decision was to use the <i>Human Voice Pitch Detector</i> library of Unity. In addition, current systems that have similar systems that were related with music learning and specially with wind instruments were analyzed. | |

6. RESULTS

| 1# Testing microphone Detector | |
|---|---------|
| User story | - |
| Duration | 2 weeks |
| <p>During these weeks the technologies selected in the previous iteration were prepared to be used in the development of the project. In addition, a simple prototype able to process the sounds recorded from a microphone and detect the corresponding pitches was developed. Tests were performed using a trumpet, the idea was to play different notes with the trumpet in order to check if the frequencies displayed in the screen were the ones played by the trumpet. Graphically, the idea was to display the frequencies in an ascendant order like it would be in a music sheet.</p> | |

| 3# Simple music sheet | |
|---|---------|
| User story | - |
| Duration | 3 weeks |
| <p>Once the system was able to detect pitches from instruments in a consistent way the next step was to create a simple exercise, in this a scale and test how the different notes could be measured and displayed in the screen. It was necessary the creation of a new screen based on the previous one. However, the new screen was designed in order to display the notes of the current exercise in a music sheet.</p> | |

| 4# Identification of users | |
|---|---------------------------------|
| User story | US-REGISTER US-LOGIN US-PROFILE |
| Duration | 3 weeks |
| <p>In this iteration the logic of the system and screens related with the identification of the users were developed. As a result, it was necessary to develop screens to register into the application and to login, this screens were created taking into account the principles of design in Android applications aiming to improve the user experience. The login screen will ask for a username and a password, this information is necessary to access profile of an user. The register screen asks for the username, name, surname, email, and password in order to create an account. In addition, another feature developed in this iteration was the possibility of choosing an instrument.</p> | |

| 5# Creation of Exercises and creative mode | |
|--|--------------------------------|
| User story | - US-SCALE US-LONG US-CREATIVE |
| Duration | 2 weeks |
| <p>Once the user was able to create an account and access to it the next step was to allow users to create their own exercises, the idea was to modify the current structure that allow the creation of a simple scale and permit the user create different types of exercises, in this case scales and long notes exercises. It was necessary to create a specific screen to list the exercises that a user created before. In addition, two new screens one for each type of exercise were created. Each screen has general inputs that would have every type of exercise and specific inputs for the concrete type.</p> | |

| 6# Practice Mode creation and logout | |
|--|-------------------------|
| User story | - US-LOGOUT US-PRACTICE |
| Duration | 2 weeks |
| <p>During these weeks the practice mode menu was developed. This menu will show the routines and the corresponding exercises for the routines that an user have. In addition, it was added a new feature in the others screen that would allow the user to log out from the application in a safety way.</p> | |

| 7# Change Exercise module and join it with menus | |
|---|-------------|
| User story | US-EXERCISE |
| Duration | 2 weeks |
| <p>In this iteration, the simple screen that was created in the iteration 3 was changed to fit with the style of the rest of the application. The visualization of the notes, buttons and background was changed to look like a music sheet. In addition, it was necessary to create the logic to send an exercise from the <i>creative menu</i> or the practice menu to the exercise menu.</p> | |

| 8# Tuning menu and Statistics Menu | |
|--|------------------------|
| User story | US-TUNER US-STATISTICS |
| Duration | 2 weeks |
| <p>Once the main functionality of the system was working. The next step was to create a tuning menu that would allow users to tune their instruments so the results of the exercises will be more accurate. In addition, during this sprint some of the graphical aspects of the application were changed or redesigned.</p> | |

6.1.2 User stories

User stories are an important part in every Agile project. They reason is that they represent a feature of the system but in the perspective of the client, user stories are the best way to understand what the system needs to do. In this subsection the user stories that were defined during the development of the project are detailed.

| | |
|---------------------|---|
| Identifier | US-LOGIN |
| Role | As a user... |
| Functionality | I want to be able to access to the system |
| Acceptance criteria | The user must be able to access the system using their credentials, that credentials are an username and a password. In case the credentials are wrong the system needs to give feedback to the user about the impossibility of accessing the system. |

Table 6.1: User story US-LOGIN.

6. RESULTS

| | |
|---------------------|---|
| Identifier | US-REGISTER |
| Role | As a user... |
| Functionality | I want to be able to create an account in the system |
| Acceptance criteria | The user must be able to create an account in the system inserting their information, this information consist in his name, surname, username, email and password. In case an account has already been created with the same username the system will not create the account and needs to give feedback of the possible errors to the user. |

Table 6.2: User story US-REGISTER.

| | |
|---------------------|--|
| Identifier | US-PROFILE |
| Role | As a user... |
| Functionality | I want to be able to access to a profile related with a particular instrument |
| Acceptance criteria | The user must be able to select a profile related with an instrument after accessing the system. The system must show the different instruments that are supported and the user needs to be able to select the one that want to use at that moment |

Table 6.3: User story US-PROFILE.

| | |
|---------------------|--|
| Identifier | US-STATISTICS |
| Role | As a user... |
| Functionality | I want to get feedback about statistics of my profile |
| Acceptance criteria | The user must be able to consult the statistics of a concrete profile. The information showed in the statistics screen must include the number of exercise, the number of routines, the number of exercises played and the number of notes played. |

Table 6.4: User story US-STATISTICS.

| | |
|---------------------|---|
| Identifier | US-LOGOUT |
| Role | As a user... |
| Functionality | I want to close the application safely |
| Acceptance criteria | The user must be able to log out from the application using a button that needs to be placed in each screen except login screen, register screen and exercise screen. |

Table 6.5: User story US-LOGOUT.

| | |
|---------------------|---|
| Identifier | US-SCALE |
| Role | As a user... |
| Functionality | I want to create Scale exercises |
| Acceptance criteria | The user must be able to create a scale exercise using a menu in the application. The menu needs to contain the following inputs: beats per minute, first note, time signature, type of scale and duration of notes |

Table 6.6: User story US-SCALE.

| | |
|---------------------|---|
| Identifier | US-TUNER |
| Role | As a user... |
| Functionality | I want to be able to tune my instrument. |
| Acceptance criteria | The user must be able to check the tuning of his instrument in a separated screen. For this purpose, it is necessary to show the current pitch played by an instrument and the current frequency of recorded. |

Table 6.7: User story US-TUNER.

| | |
|---------------------|---|
| Identifier | US-CREATIVE |
| Role | As a user... |
| Functionality | I want to be able to see all the exercises that i have created and see what kind of exercise i can create. |
| Acceptance criteria | The user must be able to visualize to visualize the beats and time signature of all the exercises that he created. In addition, the different types of exercises need to be displayed |

Table 6.8: User story US-CREATIVE.

| | |
|---------------------|--|
| Identifier | US-LOGOUT |
| Role | As a user... |
| Functionality | I want to see all the routines that i have and each of the exercises of that routine |
| Acceptance criteria | The user must be able to visualize a Routine and the different exercises that form that routine. |

Table 6.9: User story US-PRACTICE.

| | |
|---------------------|---|
| Identifier | US-LONG |
| Role | As a user... |
| Functionality | I want to create Long Notes exercises |
| Acceptance criteria | The user must be able to create a long note exercise using a screen in the application. The menu needs to contain the following inputs: beats per minute, first note, time signature, number of notes and duration of notes |

Table 6.10: User story US-LONG.

| | |
|---------------------|---|
| Identifier | US-EXERCISE |
| Role | As a user... |
| Functionality | I want to practice with my instrument a music sheet and get feedback about the results |
| Acceptance criteria | The user must be able practice a music sheet that has to be displayed in a screen of the application. The notes have to change the colour between green, orange and red depending on how good was performed that particular note. |

Table 6.11: User story US-EXERCISE.

6.2 Windy Melody: Final Result

Windy melody is the name of the application resulting of the development of this project. It follows the architecture explained in the chapter 5. In this section the flow of the application and the different menus that compose it are explained. Additionally the working environment used during the development of the project is showed in Figure 6.1.

Login Menu

As soon as the application is initialized, this menu will be displayed (see Figure 6.2). As with any other application that handles the data of different users, it is necessary to control who is the user that is **accessing to the system**. Users will use this menu to access to their account, any student will have to write their username and password in order to enter to their respective accounts. If an user do not have an account created in the system yet, they would have to click on the *sing it up* link and access the *register menu* where they can create an account. This menu was intended to be a simple log in, that is the reason why gamification techniques were not applied in this screen. However, other concerns as *Android Guidelines* were applied during the development of this menu.

Register Menu

If an user do not have an account in the system they could access to this menu to **create a new account** in *Windy melody* (see Figure 6.3). In order to create a new account, it is necessary to fill all the information that is requested, the information includes the username,

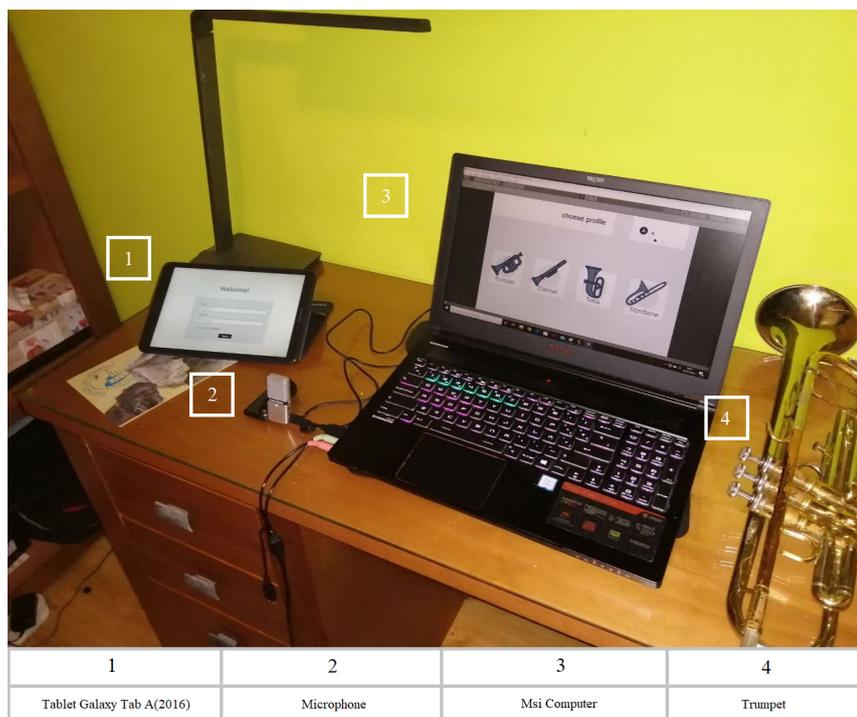


Figure 6.1: Working environment.

name, surname, email and password. Once the user press the *Register* button, they system will validate the information inserted and if every field was filled. If some information is missing the system will prompt a message to the user informing him that all the fields are required to create an account. Then, when it is checked that all the information was valid the system will try to create a new account. If the process was successfully completed the user will be redirected to the login menu so he can access to the system. However, if there was any problem the system will prompt a message informing the user that was impossible to create that account and the reason. This menu followed the same approach as the *login menu* , a simple menu where it was not necessary to apply gamification techniques.

Profile menu

Once a user has accessed to its account through the login, this screen will be displayed (see Figure 6.4). In this screen the user will be able to select a profile. Each profile will be related with a **different wind instrument**. The idea behind choosing a profile is that there are musicians that practice more than one instrument and it is reasonable to have separated profiles so save exercises for each instrument. Another important consideration designing this screen was the inclusion of icons to represent the different instruments. Icons for instruments will help to musicians that are not language-speakers to select the instrument they want and improve the user experience. For now, the different instruments that can be selected are the trumpet, the clarinet, the tuba, and the trombone.

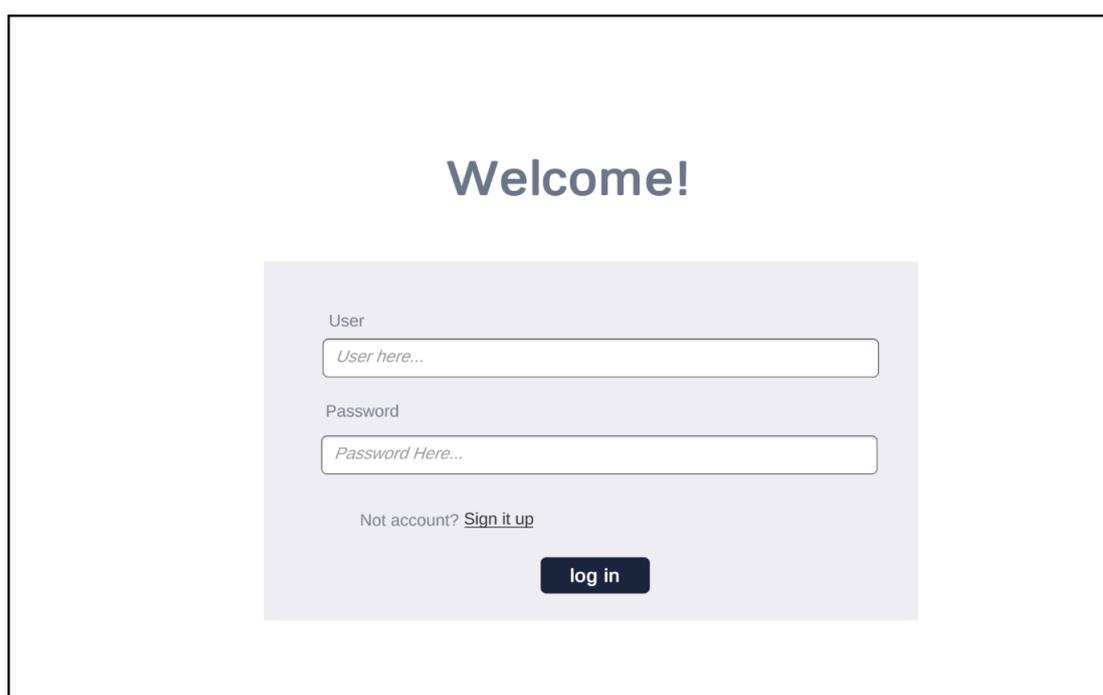


Figure 6.2 shows a login menu interface. At the top, the word "Welcome!" is displayed in a large, bold, blue font. Below this, there is a light gray rectangular area containing the login form. The form has two input fields: "User" with a placeholder "User here..." and "Password" with a placeholder "Password Here...". Below the password field, there is a link "Not account? Sign it up". At the bottom of the form, there is a dark blue button labeled "log in".

Figure 6.2: Login Menu.

6. RESULTS

Welcome, create your account

User

Email

Name

Surname

Password

Already have an account? [Sign in](#)

Register

Figure 6.3: Register Menu.

choose profile

U user



Trumpet



Clarinet



Tuba



Trombone

Figure 6.4: Profile Menu.

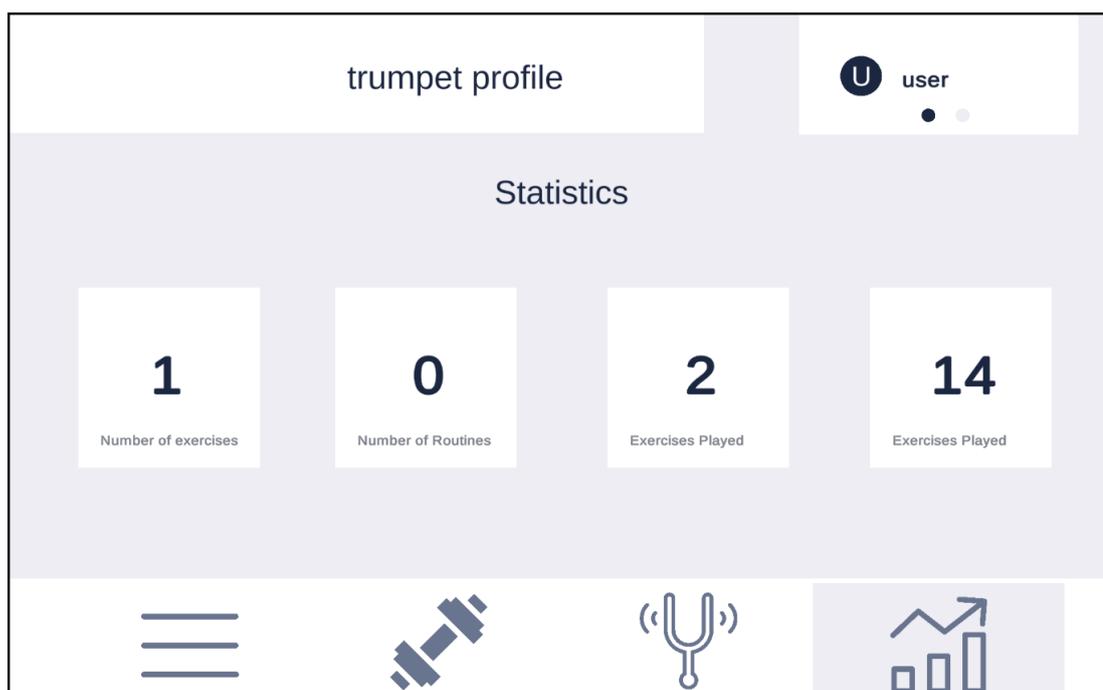


Figure 6.5: Statistics Menu.

Statistics menu

This is one of the menus that a user that have selected a profile can access (see Figure 6.5). This menu will allow the user to consult some statistics that are related to their profile. In other words, these statistics are specific for **one instrument**. In this menu the user will show four different squares where a number and the meaning of that number are displayed. For now, the information displayed is the number of exercises, the number of routines, the exercises played, and the notes played.

Tuning menu

Users could navigate to this menu any time they want. Users will be able to **tune their instruments** using this menu. As it is known, in order to measure the frequencies produced by an instrument in a proper way, it is necessary that the instrument has its appropriate tuning. The *Tuning menu* will record the sounds produced by the instrument and will show the pitch, frequency and midi recorded. The idea is that the musician will try to play the note called LA 4 that has a frequency of 440 Hz. If the frequency showed is not near 440 Hz then the musician has to tune its instrument until it is near enough.

Create Scale menu

the purpose of this menu is to allow users to create their own exercises, in particular exercises of type scale (see Figure 6.6). The idea behind this menu is to make it as easier as possible for a beginner musician. As a result, the different options for creating a scale are simple and only ask for information that every musician knows. In order to create a scale, the

Figure 6.6: Create scale menu.

user needs to fill in every field and then press the saving button. Once that button is pressed, the system will save that exercise and the user will be redirected to the creative menu.

Long Notes menu

the purpose of this menu is to allow users to create long notes exercises (see Figure 3.9). This menu is very similar to the *create scale menu*, it also has several inputs that are necessary to create an exercise of this type. The user has to fill every field and then press the save button to create the exercise. Then, the user will be redirected to the *creative menu* where he could visualize all the exercises associated with that profile.

Log Out and Profile

This feature can not be considered as a screen by its own. However, it is integrated in almost every menu inside the application. It has two parts: i) a log out button that when is pressed the user can close the application and ii) a text with the username of the current user.

Creative menu

In this menu the users will have access to their exercises and to create new ones (see Figure 6.8). As it can be observed, this menu has a list of exercises. This **list of exercises** are the ones that a user has previously created with that profile. The information displayed for each exercise will be the time signature, the beats per second and the number of that exercise. In order to practice one of these exercises the user only has to press in the white square that



Figure 6.7: Exercise Menu.

contains the exercise and the user will be redirected to the *exercise menu*. In addition, this screen has two additional buttons: i) create scale exercise and ii) create long notes exercise. Each button will redirect the user to another screen where it is possible to create each type of exercise. As it is possible to notice, the number of exercises that a user can have could vary. In order to handle this issue, the buttons that contains the exercise are created dynamically depending on the number of exercises and they are inside a scroll panel so users only have to scroll if they want to see all the exercises that they have.

Practice menu

in this menu the users will have access to their routines (see Figure 6.11). The idea is that the user could have a **set of routines** and each routine would have a set of exercises associated. In addition, it also has a scroll panel when the number of exercises make impossible to display all in the screen. The user will have to press an exercises and the system will redirect him to the *exercise menu* where the exercise could be performed.

Exercise menu

The exercise menu will be the screen where user will practice with their instruments. As it can be observed in the Figure 6.7, this menu displays a **music sheet** that contains an exercise, in this example it is a scale. This screen will be displayed when a user select an exercise to practice in the *creative menu* or in the *practice menu*. The notes of that exercise are calculated and placed in their corresponding places according to the rules of tuning and depending on the type of clef. This screen contains two buttons: i) the *rec button* that will

6. RESULTS

enable the microphone in order to record pitches produced by an instrument and ii) the *go back* button that when is pressed it consider the exercises finished and redirect the user to the profile screen. Once the *rec button* is pressed users can start practicing with their instruments. While they are playing the notes will change its colour to green, orange, or red depending on how well the note was played.

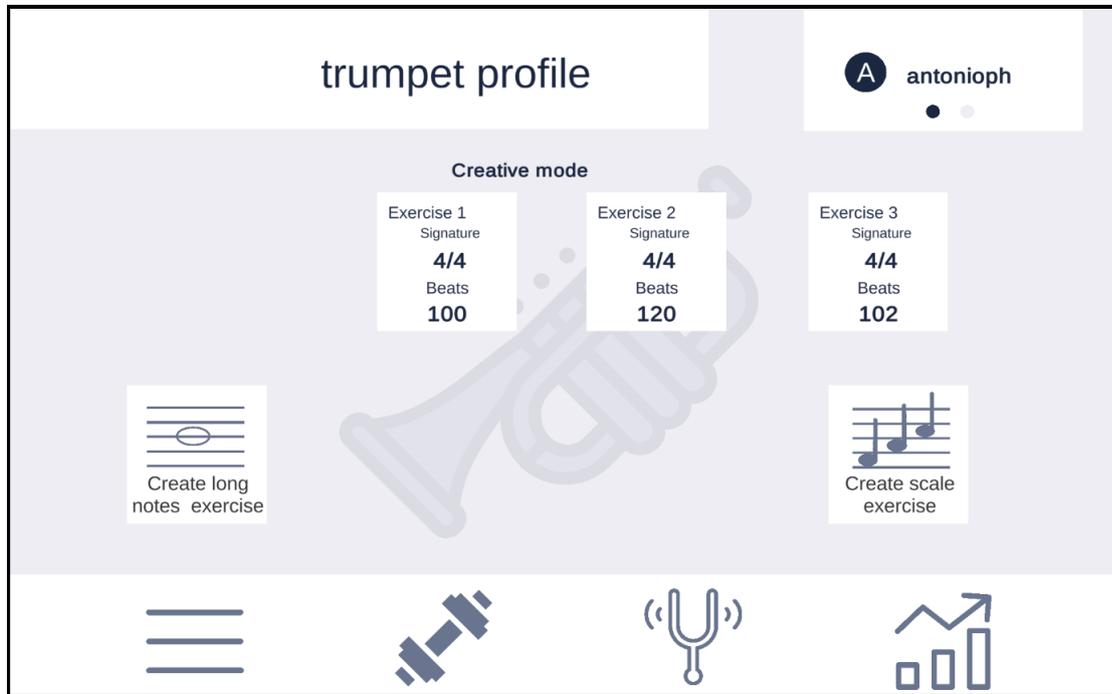


Figure 6.8: Creative Menu.

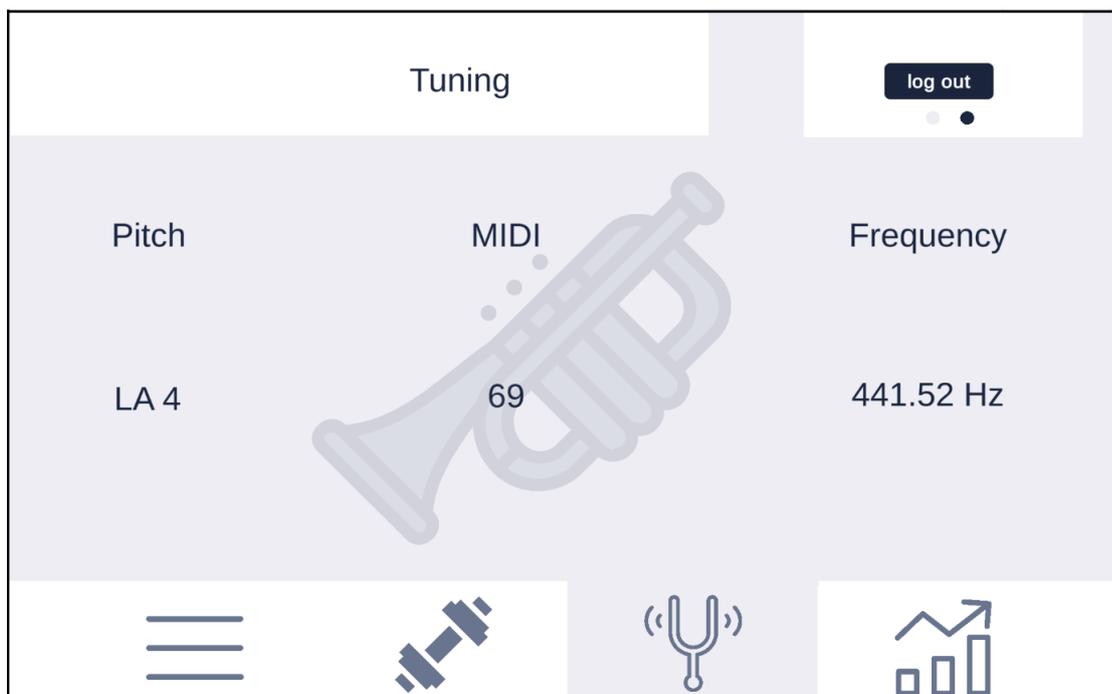


Figure 6.9: Tuning Menu.

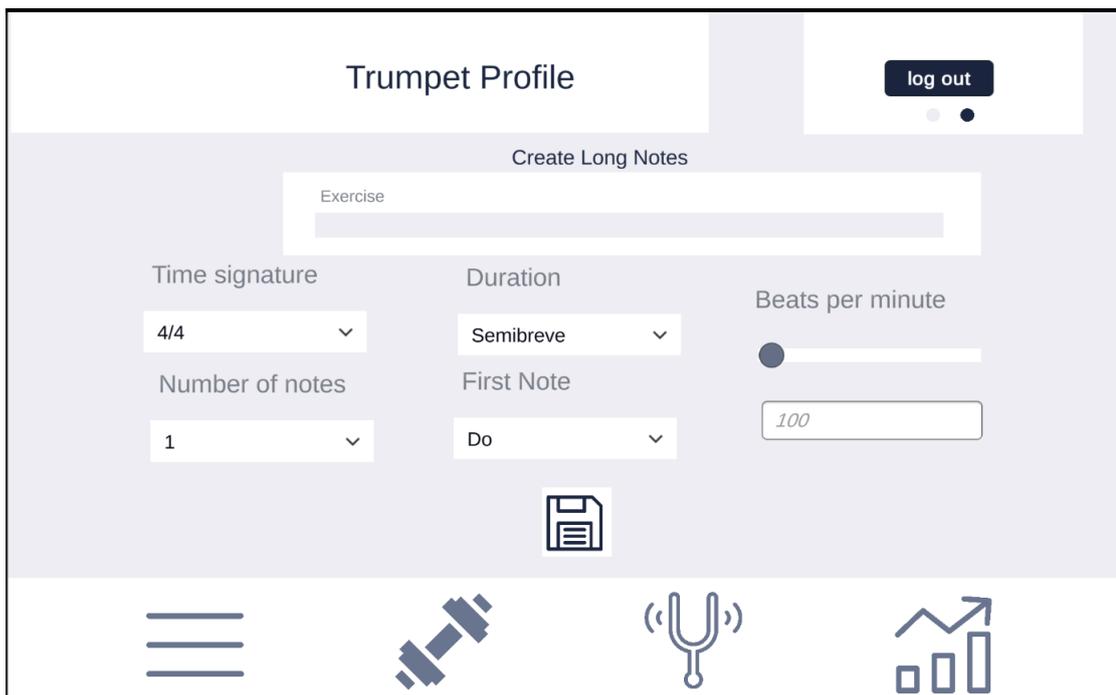


Figure 6.10: Long Notes Menu.

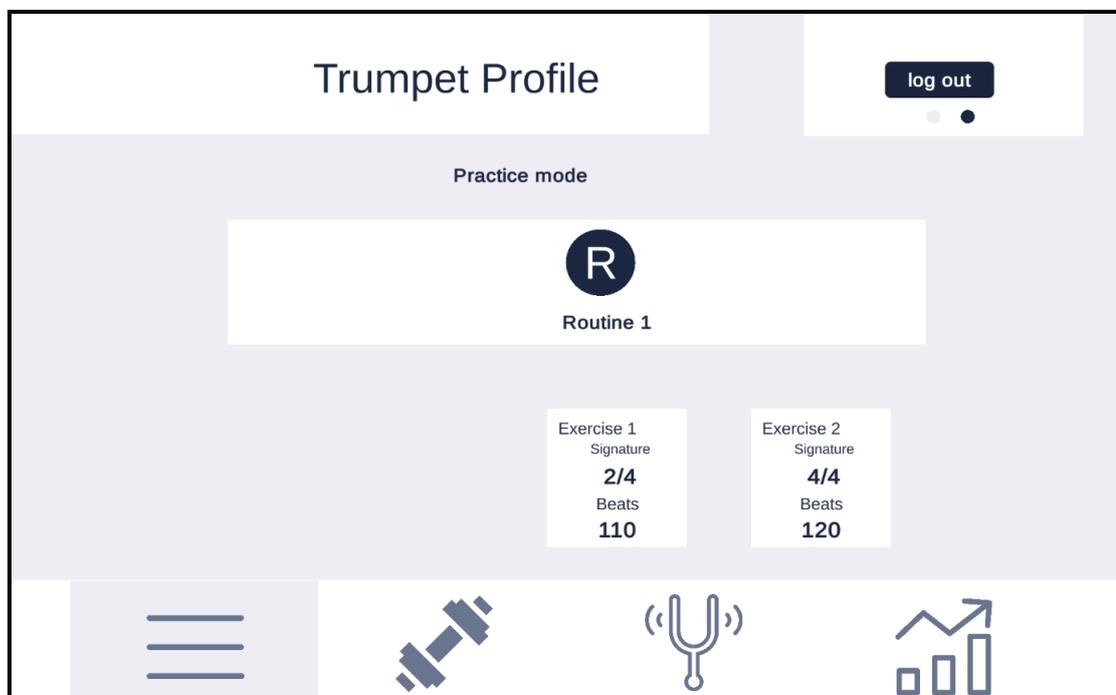


Figure 6.11: Practice Mode.

6.3 Project cost and resources

In this section the projected cost and resources used during the development are detailed and explained (see table 6.3). The type of resources this project needed where both software and hardware resources. In addition, other resources as a trumpet were needed.

This project required a high quality microphone in order to test how the different sounds were recorded and to make sure that there was not any problem with the input device. The microphone selected was the Go mic usb Microphone of Samsom and costed 55.25 €. Other Hardware resources as the Graphic card, the Msi computer and the trumpet were not took into account because they were not acquire specially for the development of this project.

| Resource | Cost | Amount | Total |
|----------------------------|------------------|---------|-----------------|
| USB Microphone | 55.25€ | 1 | 55.25€ |
| Human Voice Pitch Detector | 13€ | 1 | 13€ |
| Personnel salary | 1894.375 €/month | 4 month | 7577.5€ |
| Total budget | | | 7645.75€ |

Table 6.12: Estimated budget for the developed project.

Software resources can have a cost in terms of licensed, in this case the library Human Voice Pitch Detector¹ in the Unity Store had a cost of 13€. On the other hand, software resources as *GitHub* were used with a student account so it did not have any additional cost.

In addition, in order to be as close as possible to the reality a salary for the developer has been estimated. According to a popular web site called payscale², the estimated salary for a junior developer is about 21.65€/hour. This project needed a total of 350 hours which means a total salary of 7577.5€.

6.4 Project statistics

In this section, some statistics of the project will be detailed. During the development of the project the technology used to save the software was GitHub. It is possible to use this technology in order to get statistics related with the development of the project.

The repository in GitHub was created the 12th of February and it has commits until the 28 of July. It is possible to observe different programming language that are present in the repository. For example PHP and JSON . However, the 97% of the lines of code are written in C#.

¹<https://assetstore.unity.com/packages/tools/audio/human-voice-pitch-detector-109019>

²https://www.payscale.com/research/ES/Job=Junior_Software_Engineer/Salary

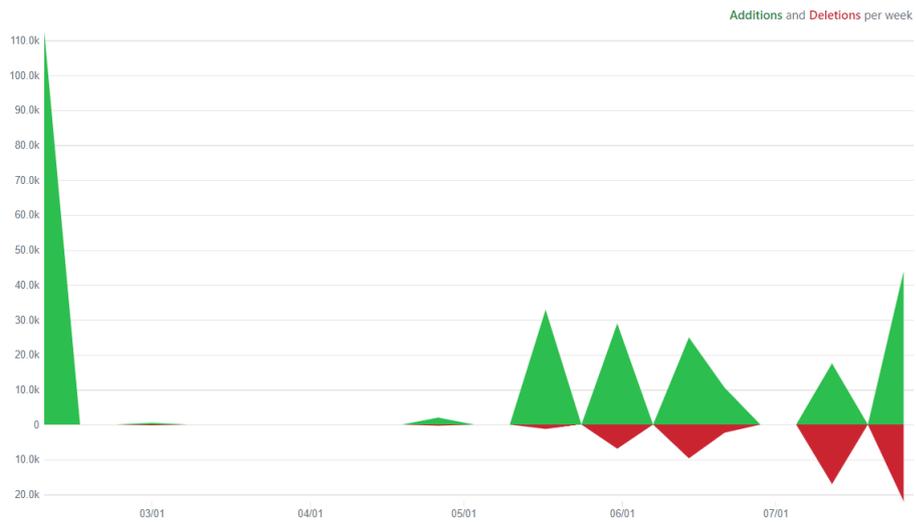


Figure 6.12: Additions and Deletions per week.

The only contributor of the repository was the author of the document, in this Figure 6.12 the additions and deletions per week in lines of codes can be appreciated.

Chapter 7

Conclusions

This chapter discusses the obtained results, the reached objectives, and the problems faced while developing the software.

7.1 Reached objectives

In this section, I will explain in detail how the objectives proposed in chapter 2 have been accomplished.

- **Scalability.** This objective can be considered completed thanks to several factors. One of these factors is the type of architecture employed in the development of the system, the layered model divides the system into three layers that are independent from each other. This independence implies that if we change something in one layer it would not affect the other two layers. In addition, the application was designed taking into account that new types of exercises could be added to the application. Furthermore, the addition of new wind instruments is also possible too. However, new instruments will impact on the application due to the different tuning process between instruments.
- **User's motivation.** This objective is considered accomplished due to the use of gamification techniques used in order to keep users motivated while they study with their instruments. The menus of the application are created with the intention of evoking a feeling of familiarity with video games. Furthermore, each exercise in the application is considered as a level of a video game.
- **Ease of use.** In order to develop a system that provides a good user experience, it is necessary to create a GUI with several menus that are intuitive and give feedback to the user about their actions. In this particular case, the system counts with different menus that are intuitive. In addition, the application gives visual feedback when the user takes any action. Another feature that improve the usability of the application is the possibility of creating your own exercises, this feature is designed to allow any type of musician to understand what type of exercise they are creating, it is not necessary to be a professional in order to fill properly the fields necessary to create an exercise. This feature will allow beginners to feel comfortable while it still provides full control to the user to create exercises of the difficulty they want.

7. CONCLUSIONS

- **Intuitive system.** Making an intuitive application is a good way to improve the usability of the application. In order to create an intuitive system, it is important to develop a GUI that is easy to understand for every user and does not require any extra explanation. In the proposed system, most of the menus were designed so every user could understand what to do in each one. However, there are some menus that need a bit of previous knowledge about music. For example, the *Create Exercise* menu but as this system is created for musicians and it is considered that is necessary a minimum music knowledge to use the application we can consider that the objective is accomplished.
- **Validation regarding a specific wind instrument.** An application that is intended to work with different wind instruments as *Windy Melody* needs to configure some features considering aspects like the tuning and tessitura. In order to validate the system, we decided to test it with a concrete wind instrument, the instrument selected was the trumpet in SIB. Once the system was checked with the trumpet, we ensured that the system would work for another SIB wind instruments. Furthermore, it should work with any problem any other wind instruments if the tune and the tessitura selected is the correct for it. In addition, it was also tested with a clarinet and the same results were obtained.

7.2 Problems faced

During the development of the project there was some difficulties. In this section the more relevant ones will be explained.

One of the difficulties faced was how to handle notes and their correct **visualization**. A note can be understood as a pitch that sounds during a concrete amount of time. However, the problem comes when it is necessary to display that note to different wind instruments. Each instrument has a particular tune, a concept that is explained in chapter 3. As a result, the same note needs to be displayed in a different position for each wind instrument, that is the reason why it is necessary to select an instrument in the *profile menu*. In addition, there are other considerations as the clef of the music sheet that also changes the position of the notes.

Another important aspect that has influenced the project was the current world situation due to the global pandemic and the resulting **lockdown**. In this context, it was required to change the way of working and the planification. For example, the face to face meetings with the client that were planned had to be cancelled. The solution to this concrete problem was to have online meetings instead of face to face.

7.3 Addressed competences

In this section the competences related with the intensification of software engineering that were applied in the development of this project are listed and explained.

- **[IS1] Ability to develop system applying software engineering techniques.** In order to create this system, some of the most known techniques related with software engineering were applied. For example, the application of design patterns during the development when it was possible. In addition, standards for the development of Android applications, in concrete for tablets were followed in the creation of the GUI.
- **[IS3] Ability to create a system integrating available resources.** In order to accomplish the objectives of this project different technologies were used. The technologies were selected as a result of a study of the different possible solutions for a concrete problem, then these technologies were integrated to create this system.
- **[IS4] Ability to identify problems and design software solutions.** During the creation of this system it is possible to observe different phases. First of all, the problem was analyzed during the first iterations of the development of the project, then a solution was designed taking into account the different technologies selected. Finally, the software project was implemented, tested and documented.

7.4 Future work

In this section, the improvements or additions that could be added to this project in the future will be described.

- **Exchange of exercises routines between users:** with the aim of improving the user experience and increasing the motivation of students that use the application, it is proposed the integration of a new feature that allows students to exchange their routines and exercises with others. The idea is to use the current *JsonParser Module* and modify it to allow the creation of JSON files from exercises inside the application. As a result, students would use a screen to charge a JSON with the information about an exercise or a routine and the system would include it into their routines or exercises.
- **Add new types of exercises and instruments:** the current system has support for two types of exercises. These two types are the two more relevant for musicians that practice at home. However, the inclusion of new types would improve the quality of the learning process. Additionally, other instruments like saxophone or flute could be included in order to increase the number of musicians that would benefit from the system.
- **Graphical creation of exercises:** in order to give to users more freedom to create their exercises and modify them, specially for expert musicians, it is interesting to include a way of creating exercises dragging notes to a music sheet. For this purpose, it would not be necessary to include new technologies, the features that Unity gives support for this type of screens.
- **Teacher support tool:** in order to improve the quality in the educational aspect, it

is proposed to develop a feature that would allow teachers to monitor the progress of students in the application. In this context, the routines that a student practice would be sent by their teacher. In addition, it is proposed to add a feature for teachers to get feedback about the progress of their students in a routine. To achieve this objective, it would not be necessary to create a new system, the solution could be the integration of different types of users in the current system that would have different roles.

- **Testing with different types of instruments:** with the aim of adding new instruments to the application and validate the system for all the possible wind instruments that are suitable for this purpose, it is proposed to test different types of exercises with different wind instruments. The idea would be that once the system is validated for that instrument, it could be included in the possible instruments that a user can choose.
- **Publish the application:** this application was created with the intention of helping musicians to study at home. In order to increase the public for the application it would be interesting to publish it in application markets as *Google play store*¹. To accomplish this objective it would be necessary to pay a fee of 21,45€ to register a development account, then, after configuring privacy and other Google requirements it would be possible to publish the app in the *Google play store*.

¹<https://play.google.com/store/>

APPENDICES

Appendix A

Appendix A

A.1 Image References

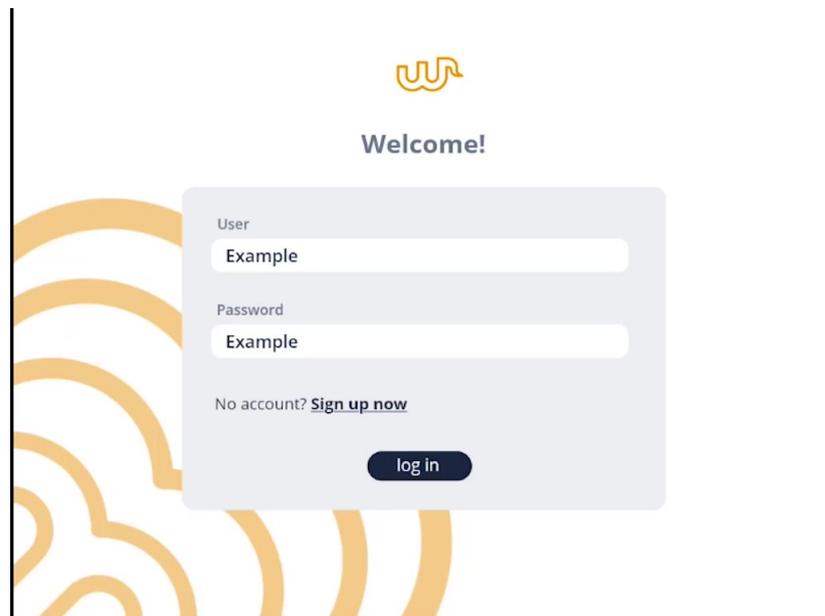


Figure A.1: Login reference.

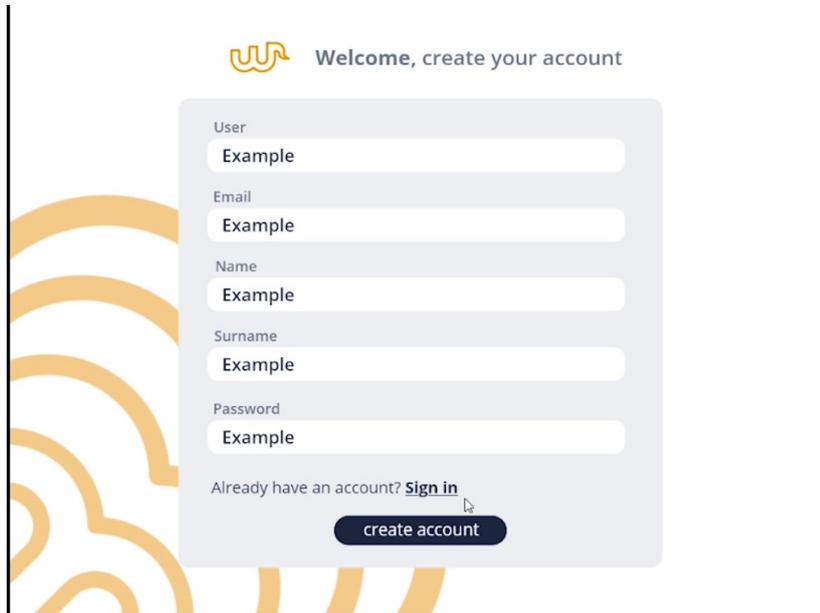


Figure A.2: Register reference.

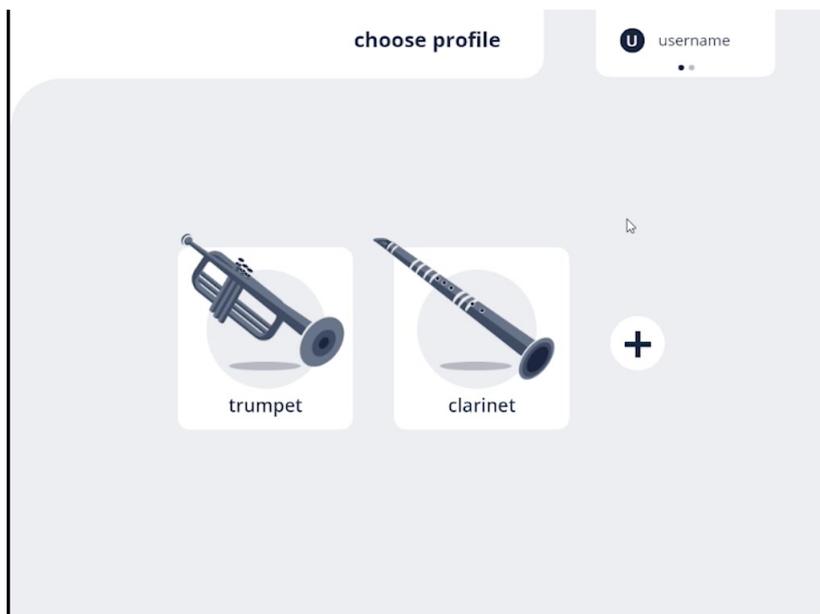


Figure A.3: Profile reference

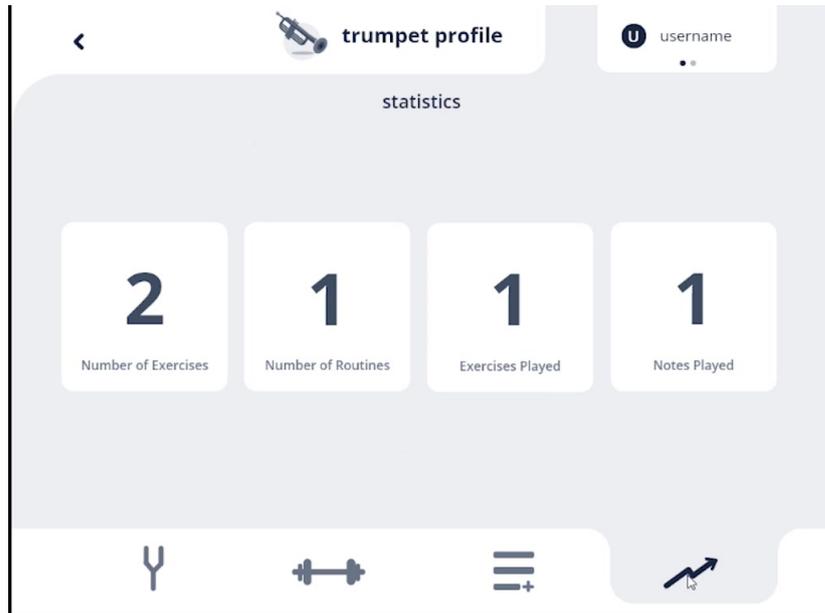


Figure A.4: Statistics reference.

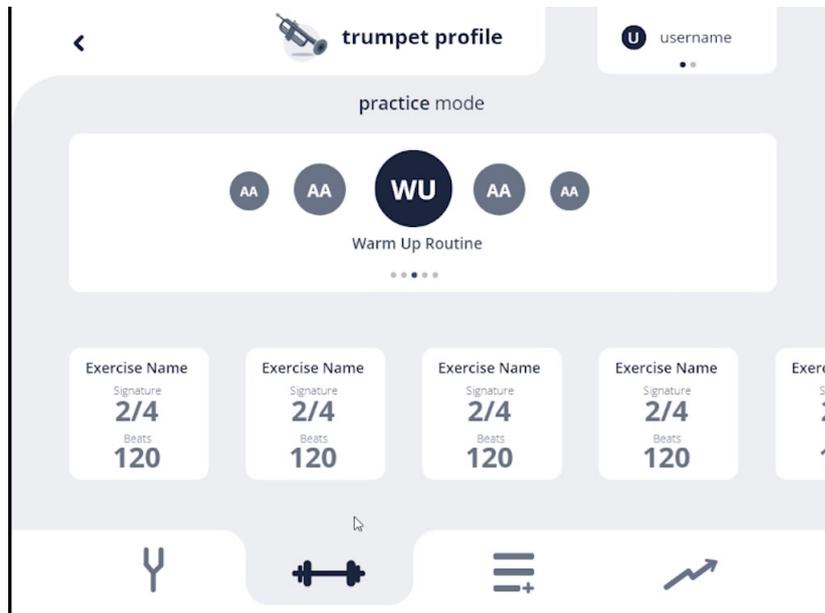


Figure A.5: Routines reference.

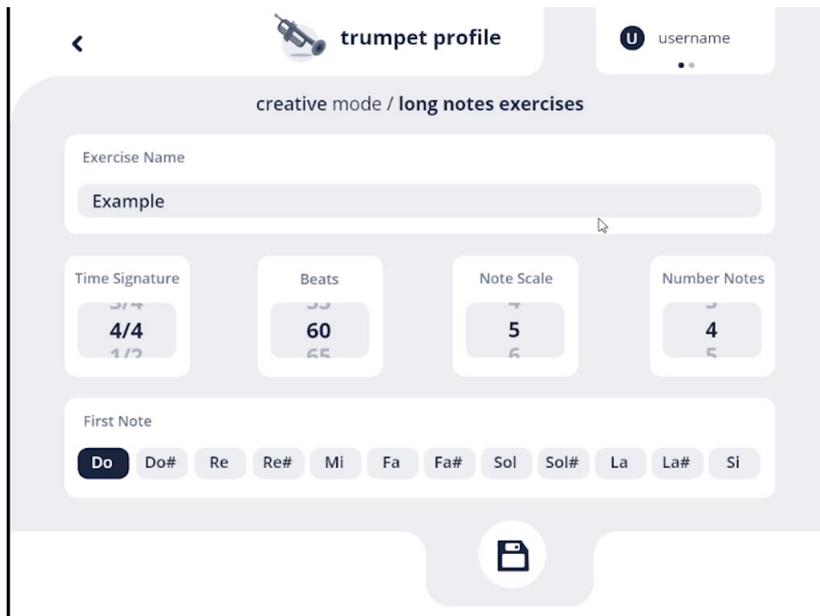


Figure A.6: Scale reference.

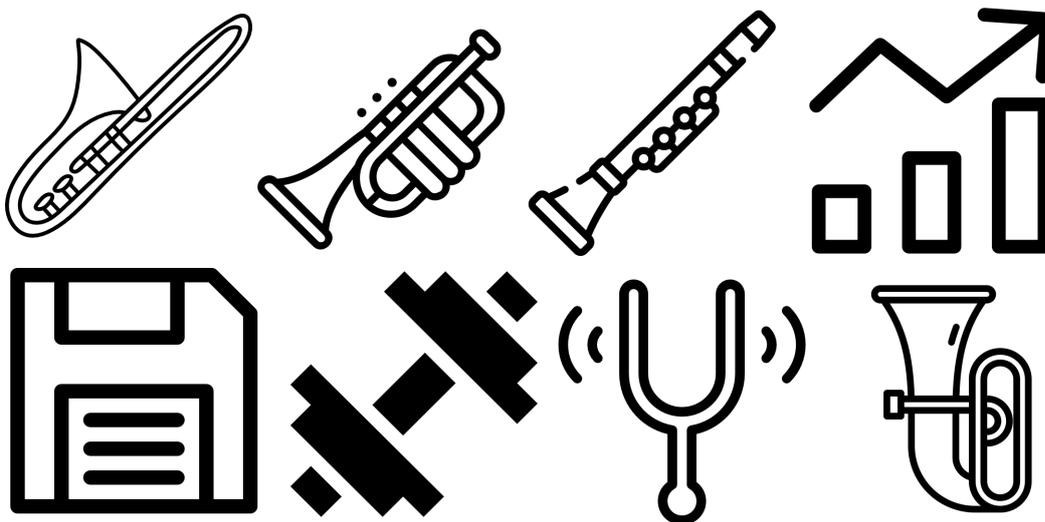


Figure A.7: Icons designed <https://www.flaticon.es/autores/freepik>.
and <https://www.flaticon.com/authors/good-ware>

Appendix B

Appendix B

B.1 Routine represented in a JSON file

```
1 {
2   "type": "Routine",
3   "description": "Routine for warn up",
4   "finalDate": "20/10/2020",
5   "1": {
6     "ExerciseType": "Scale",
7     "Scale": {
8       "beats": 100,
9       "tune": "Sib",
10      "timeSignature": "4/4",
11      "firstNote": "DO 4",
12      "scaleType": "Mayor Scale",
13      "duration": 8
14    }
15  },
16  "2": {
17    "ExerciseType": "Scale",
18    "Scale": {
19      "beats": 120,
20      "tune": "Sib",
21      "timeSignature": "2/4",
22      "firstNote": "RE 4",
23      "scaleType": "Mayor Scale",
24      "duration": 4
25    }
26  }
27 }
```

Code listing B.1: Example of Routine in JSON File

```

1  {
2    "type": "Routine",
3    "description": "Routine to improve sound D0 4",
4    "finalDate": "20/10/2020",
5    "1": {
6      "ExerciseType": "LongNotes",
7      "Scale": {
8        "beats": 60,
9        "tune": "SIb",
10       "timeSignature": "4/4",
11       "firstNote": "D0 4",
12       "noteTimes": 4,
13       "duration": 2
14     }
15   },
16   "2": {
17     "ExerciseType": "LongNotes",
18     "Scale": {
19       "beats": 60,
20       "tune": "SIb",
21       "timeSignature": "2/4",
22       "firstNote": "D0 4",
23       "noteTimes": 4,
24       "duration": 2
25     }
26   }
27 }

```

Code listing B.2: Example of Routine in JSON File

```

1  {
2    "type": "Routine",
3    "description": "Routine with rapid exercises",
4    "finalDate": "20/10/2020",
5    "1": {
6      "ExerciseType": "Scale",
7      "Scale": {
8        "beats": 160,
9        "tune": "SIb",
10       "timeSignature": "2/4",
11       "firstNote": "LA 4",
12       "scaleType": "Minor Scale",
13       "duration": 4
14     }
15   },
16   "2": {
17     "ExerciseType": "Scale",
18     "Scale": {
19       "beats": 160,
20       "tune": "SIb",
21       "timeSignature": "2/4",
22       "firstNote": "SI 4",
23       "scaleType": "Minor Scale",
24       "duration": 4
25     }
26   }
27 }

```

Code listing B.3: Example of Routine in JSON File

References

- [APL10] KA Akant, Rajesh Pande, y SS Limaye. Monophony/polyphony classification system using Fourier of Fourier transform. *International Journal of Electronics Engineering*, 2(2):299–303, 2010.
- [Bec99] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.
- [Car02] Adam Carse. *Musical wind instruments*. Courier Corporation, 2002.
- [CLSBÁRCG18] Mónica Carreño-León, Andrés Sandoval-Bringas, Francisco Álvarez-Rodríguez, y Yolanda Camacho-González. Gamification technique for teaching programming. En *2018 IEEE Global Engineering Education Conference (EDUCON)*, páginas 2009–2014. IEEE, 2018.
- [Dar09] Hugh Darwen. *An introduction to relational database theory*. Bookboon, 2009.
- [dIF13] Jesús Mariano Merino de la Fuente. *Las vibraciones de la música*. Editorial Club Universitario, 2013.
- [FA12] David Vallejo Fernández y Cleto Martin Angelina. *Desarrollo de videojuegos: Arquitectura del motor de videojuegos*. Escuela Superior de Informática, UCLM, 2012.
- [Fu15] Jiulin Zhang Xiaoqing Fu. The influence of background music of video games on immersion. *Journal of Psychology & Psychotherapy*, 5(4), 2015.
- [Gro12] Fabian Groh. Gamification: State of the art definition and utilization. *Institute of Media Informatics Ulm University*, 39:31, 2012.
- [HDAS09] Jo E Hannay, Tore Dybå, Erik Arisholm, y Dag IK Sjøberg. The effectiveness of pair programming: A meta-analysis. *Information and software technology*, 51(7):1110–1122, 2009.

- [How03] Robert S Howe. The Invention and early development of the saxophone, 1840-55. *Journal of the American Musical Instrument Society*, 29:97, 2003.
- [HSN⁺14] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, y Juan Pablo Bello. JAMS: A JSON Annotated Music Specification for Reproducible MIR Research. En *ISMIR*, páginas 591–596, 2014.
- [Jos08] José Joskowicz. Reglas y prácticas en eXtreme Programming. *Universidad de Vigo*, 22, 2008.
- [KC16] Ming-Shiou Kuo y Tsung-Yen Chuang. How gamification motivates visits and engagement for online academic dissemination—An empirical study. *Computers in Human Behavior*, 55:16–27, 2016.
- [Kra03] Lawrence Kramer. Musicology and meaning. *The Musical Times*, 144(1883):6–12, 2003.
- [Ler04] Fred Lerdahl. *Tonal pitch space*. Oxford University Press, 2004.
- [Mal80] Thomas W Malone. What makes things fun to learn? Heuristics for designing instructional computer games. En *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, páginas 162–169, 1980.
- [Mas58] Abraham Harold Maslow. *A Dynamic Theory of Human Motivation*. 1958.
- [MM85] Carlton MacBeth y Louis Maggio. *The original Louis Maggio system for brass*. Aven Corporation, 1985.
- [PVŠ⁺20] Matevž Pesek, Žiga Vučko, Peter Šavli, Alenka Kavčič, y Matija Marolt. Troubadour: A Gamified E-learning Platform for Ear Training. *IEEE Access*, 2020.
- [SDHM96] John A Sloboda, Jane W Davidson, Michael JA Howe, y Derek G Moore. The role of practice in the development of performing musicians. *British journal of psychology*, 87(2):287–309, 1996.
- [TC06] Kuo Cheang Tan y Boon Liang Chua. The sound of music and its link with mathematics. *Teaching Mathematics and Its Applications: International Journal of the IMA*, 25(4):181–188, 2006.
- [TK95] David Talkin y W Bastiaan Kleijn. A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, 495:518, 1995.

[Wes94] Martin Litchfield West. The Babylonian musical notation and the Hurrian melodic texts. *Music & letters*, 75(2):161–179, 1994.

Este documento fue editado y tipografiado con L^AT_EX empleando la clase **esi-tfg** (versión 0.20181017) que se puede encontrar en:
https://bitbucket.org/esi_atc/esi-tfg

