MyBand – a software system for music performances assistance

Cosmin Beniamin Cristea
Ovidius University of Constanta
124 Mamaia Bd, 900527,
Constanta, Romania
cosmin0009@gmail.com

Dorin-Mircea Popovici
Ovidius University of Constanta
124 Mamaia Bd, 900527,
Constanta, Romania
dmpopovici@univ-ovidius.ro

ABSTRACT
In our paper we introduce MyBand, a software system dedicated to assisting a band during musical performances. After presenting our motivation related to current state of the art, we give short insights concerning software design process and technical aspects of our work. After a section dedicated to two research issues that lead us to the implementation of two algorithms for chords recognition and transposing, we conclude by discussing the MyBand's usability and some future directions we focus on.

Author Keywords
Software system; interaction; music performance.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Human Factors; Design; Measurement.

INTRODUCTION

In the purpose of improving musical rehearsals, a good organization and management of dossiers containing scores, tabs or song lyrics are needed, and there are few systems to do that. However, any of these systems are useless when the information needed to be saved increases or changes are needed to be made because the sheets are always lost or mixed, the editing cannot be done efficiently because each sheet has to be changed, and a lot of time is wasted in doing so.

As we know, a musical group which has a weekly activity, must play song covers, so every week they need to learn 6 new songs. Therefore, there are already lyrics, tabs or scores for these songs, or if they cannot find them, they make them. As in any other fields, time is very precious, and the disorganization of rehearsals reduces the quality of the music they sing.

After analyzing all the songs and stand file factors that disrupted a musical group rehearsal, as consequence productivity and quality, we created a list of features the app should have to eliminate the disorganization, but at the same time to offer other features that the classic stand could not provide, in order that group's productivity will increase.

CURRENT EFFORTS

One of the most popular application to assist in musical rehearsals is OnSong [1]. By storing thousands of songs, translating the songs' key or applying a capo, formatting the chords and lyrics for optimal viewing, adding notes and drawings, creating songs or editing existing ones, OnSong permits to quickly set lists and organizing events. Other essential facilities that this application offers are sharing easily with other members of the band, viewing chords diagrams for multiple instruments when holding on them, the auto scroll mode, or the use of a pedal for hands-free operation, playing audio files for rehearsals or just the instrumental for solo singers, setting the metronome tempo by tapping successively on the screen, integrating online chords libraries of quality and legal content, and all of these features are assumed to have a lower total price than the cost of dossiers with files.

SongBook is the most complex application for administration of songs collections with lyrics and chords, and it can be used on most mobile platforms, but also, they are available on Macs or Windows PC's, for more comfortable use [2]. The SongBook contains all the base features for this kind of application like transposing the key, creating song lists used for rehearsals or concerts, formatting the lyrics and chords, but it has also some unique features or some of them are better developed than any other application, on them is the access to an extended library of chords diagrams for different instruments like guitar, ukulele, mandolin, using of chordpro format [12], consisting in formatting the text and chords of a song with higher precision. One of the biggest disadvantages of this app is the lack of a common library with songs.

Planning Center Music Stand is a part of a larger suite of applications that are dedicated to churches to ease the organization various services, and the library of songs is also only with religious songs [3]. To be able to use this application, you need a monthly subscription to the full suite of applications, and for the various bonus features, it pays in addition. Being part of a larger suite, it has some unique features, such as syncing with all the applications in the package, which is necessary and useful for those who need the whole package, also offers a web platform to help with better management, this being an asset to other applications. Music Stand has more features that offer an edge, including a metronome that adjusts by itself when the song is changed, but also the ability to connect a pedal for changing the songs.

My Band's actual position towards these applications is below their level, but the goal is to combine what they have the best, and the result will be the most complete application of its kind. In the moment we are writing our paper, we already developed the main features of it.

The rest of this paper is organized as follows. The 3rd section presents some design issues of our application development and we highlight general architecture of our solution. Section 4 presents some research challenge we face with concerning chords' recognition and transposing. Then, we discuss on MyBand's system usability and we conclude with future work.

OUR SOLUTION – MYBAND SYSTEM

The proposed system, My Band offers a pack of features to fill the main functional and nonfunctional requirements, and has a modular architecture [9], making it very easy to maintain or add new features. My Band is designed for mobile devices, especially for tablets, and one of its objectives is to be independent of other applications, with that we mean the ability of managing everything about a band, song, plan or live session.

In the following we will focus on some software design and technical aspects.

Software design

From the beginning, we established that our software solution's main functional requirements are to create an account, login, add/edit/delete/view songs, add/edit/delete/view plans, download/upload songs, refresh plans list and session connection. We also consider some important non-functional requirements as follows:

1.Easy to use. The user does not need to be trained to use the system. In terms of usability, learnability and efficiency issues are addressed by using the constantly displayed main menu consisting of the main modules of the application, so the user is easy to find the action he/she wants to execute,

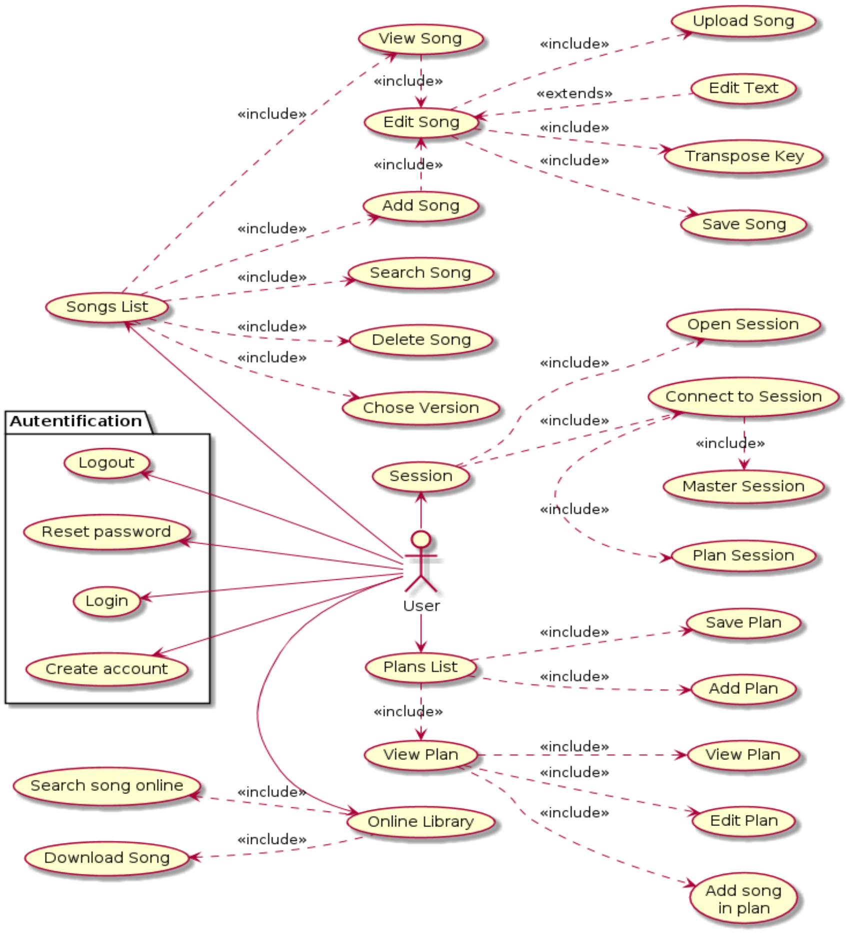


Figure 1: The Complete Use Cases Diagram (designed using LiveUML [9]).

and once they learned the menu they are on a click distance from any of functionalities. Also using this kind of menu, it helps with memorability issues, by seeing the menu it is easy to remember all the functionalities and steps to follow. There are not many errors which could be made and the existing consists in saving a song or plan with wrong data, but any of these can be edited later, so it can be fixed the errors. The satisfaction of using it is given by the short number of actions the user needs to do to accomplish its needs and by the quick response of the system [14].

2.Portability. The system can run on any operating systems

3.Security. Interruption of program execution does not compromise the accuracy and integrity of the data. The server must resist a large data traffic

4.Robustness. The system checks the date entered by the software actors and displays the error messages and requests the correct data entry.

All these requirements were used in visually designing a complete use-case diagram (Figure 1), which models the system functionality using actors and usage cases, according Unified Modeling Language (UML) standard [7].

The use cases are a set of actions, services, and functions the system needs to work. In this context, our system develops and operates, such as a website. Actors are individuals or entities that operate under defined roles within the system [4].

In order to create a supplement description of some use cases, as for "add song" one, we created the corresponding activity diagram (Figure 2), that may take the place of the activity flow section of a use case description. This diagram describes the interaction between the system and the actor [5].

In this specific Activity Diagram has described the interaction between system and user when a new song is added. The flow starts with the user's request in order to add a new song, followed by the system response by creating a form where the user can enter all the information needed to create a song. Finally, when the user has entered all the information and requested to save the song, the system applies the algorithms for chords recognition, transpose to the chosen key, and format the given text with the detected chords, and makes a version with just lyrics. All these formatted texts are saved in the song details.

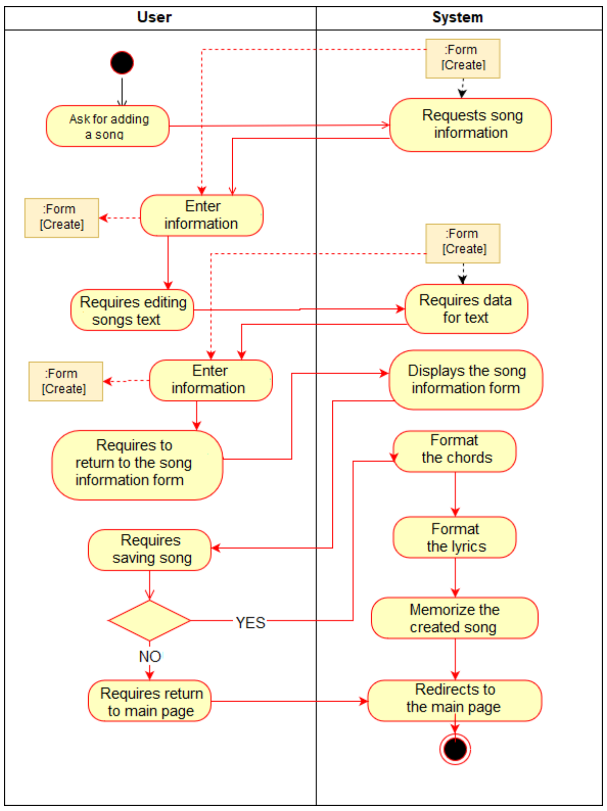

User
System
:Form [Create]
Ask for adding a song
Requests song information
Enter information
:Form [Create]
:Form [Create]
Requires editing songs text
Requires data for text
:Form [Create]
Enter information
Displays the song information form
Requires to return to the song information form
Format the chords
Requires saving song
Format the lyrics
YES
Memorize the created song
NO
Requires return to main page
Redirects to the main page

Figure 2: Activity Diagram for Use Case Add Song. (designed using draw.io [13]).

In order to structurally describe our system, we designed the class diagram depicted in Figure 3.

After analyzing the functionality requirements, we detected the real-life entities on which the system models should be developed. In this diagram, the attributes of each class are described, but also the type of relationship between them. The main entity is the *Song*, which has its attributes and methods. The *Song* is saved in *SongRegister*, and every *Song* contains a *Key*. Every *Key* contains more *Chords*. Every *Song* is created and used by a *User*. Every *User* is making part of a *Band*, can create and use *Plans* of a *Band*, and these *Plans* are saved in a *PlanRegister*. Also, every user can open a *Session*, and this *Session* can use a created *Plan*.

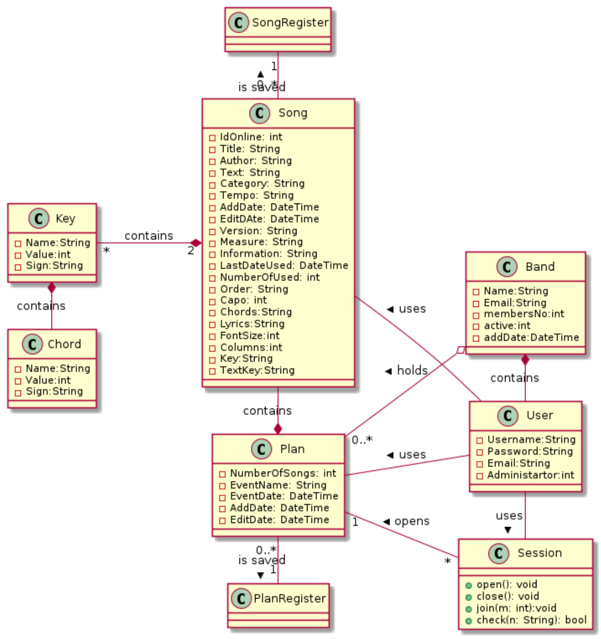


Figure 3: The complete class diagram (designed using LiveUML [8]).

Technical aspects

The whole ecosystem of this application is based on the .NET libraries (Figure 4). For developing the cross-platform mobile application we used the Xamarin framework, building a Xamarin.Forms project. Xamarin is a framework developed by Microsoft for developing cross-platform applications, whose performance, and feel are close to native.

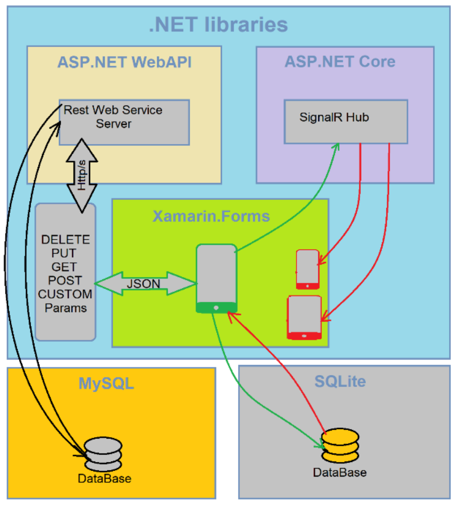


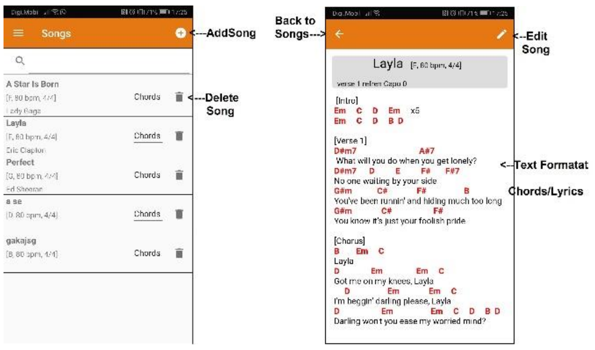
Figure 4: General MyBand's system architecture.

For the connection with an external database, we created a RESTful API with ASP.NET Web API to assure a secure connection. RESTful API is way architectural. RESTful API is an architectural style used for communication in web services, using HTTP requests like GET, POST, UPDATE, DELETE.

For the real-time communication between devices, we used SignalR with ASP.NET Core. The external database is a MySQL database, and the local one (on device) is an SQLite database. We found these technologies to be compatible with both operating systems, iOS and Android.

The songs page is filled with a list of songs, saved in SQLite database founded on the device. On this page, you can view songs, delete them, add new songs, or edit the existing ones.

Everything done on this page affects only what is saved on the device, later the user can upload the changes to everyone (Figure 5).



A Star Is Born
[F, 80 bpm, 4/4]
Lady Gaga
Chords

Layla
[F, 80 bpm, 4/4]
Eric Clapton
Chords

Perfect
[G, 80 bpm, 4/4]
Ed Sheeran
Chords

a se
[D, 80 bpm, 4/4]
Chords

gakajsg
[B, 80 bpm, 4/4]
Chords

```
Layla [F, 80 bpm, 4/4]
verse 1 refren Capo 0

[Intro]
Em  C  D  Em  x5
Em  C  D  B D

[Verse 1]
D#m7                A#7
What will you do when you get lonely?
D#m7   D     E     F#    F#7
No one waiting by your side
G#m       C#       F#          B
You've been runnin' and hiding much too long
G#m        C#         F#
You know it's just your foolish pride

[Chorus]
B    Em   C
Layla
D         Em          Em   C
Got me on my knees, Layla
  D          Em         Em   C
I'm beggin' darling please, Layla
D          Em           Em   C   D   B D
Darling won't you ease my worried mind?
```

Figure 5: Songs page and SongView page.

On the left is an image showing the song's view page. This page has a section of content that in this case is filled with the webpage created by the run of the algorithm for detection and formatting of the chords. This page contains a song edit button. At the click of the button will be displayed a form with all the current details of the song, these details can be modified and saved later.

RESEARCH CHALLENGES

The main reason for this application is to replace the classic dossiers with files and to stop printing thousands of files every time when we need a new song or we need to edit an old one, but also it has a second more challenging feature needed, and this is to detect from a given plain text which are the chords, to transpose them in the chosen key, and to format 2 different pages, the first one with chords and the second one just with lyrics. To make that possible we created two algorithms, for chords' recognition and transposing.

The Chords Recognition Algorithm

This algorithm has as an input a plain text, which should contain the lyrics and chords together.

The first step of this algorithm is to split the input text in words, so it creates an array of words.

The second step is to check every word in the array if it is a chord or not. To make that possible we created a regular expression for standard chords format. A regular expression (RegEx) [6], is a string which describes a model/pattern to search in another string, or even an entire file. To build this regular expression, we first created finite state automata (FSA) that contain different final states (Figure 6).

In order to reach one of these states, it is necessary that the text entered conforms to one of the standard formats of the chords.

We will briefly describe how this FSA diagram works. The chord can start with many white spaces. To reach the first final state (q1) it needs to have at least one basic note between A and G. Once this state is reached the input will be accepted as a chord.

From (q1) it can reach 4 more states. The first one will be (q2) when after the basic note, it has #, ##, b or bb, and the chord will look like A#. The second one(q3) is when the chord is not basic anymore, and it has after the basic note one of the following tags: sus, add, maj, min, aug, dim, m.

Each of these tags means, a different pattern of this chord, for example, m comes from minor, so the chord will have first a minor third, and second a major third. We will discuss this later, in the section with music theory needed to build these algorithms.

The third state is (q4) and it can be reached when after the basic note is a number from 1 to 9, for example, one of the most used is 7, so the chord will be A7.

The last one is not a final state, and basically from each state reached already it should be able to reach state (q5), state (q2) can only be reached from the state (q1) when the state (q3) can be also reached from (q2). State (q4) can be reached from both (q2) and (q3). The state (q5) is a delimiter between basic chord and bass note, so everything before reaching this state is a chord, and what is after (q5) is a single note, the bass note.

Based on this FSA, we created a RegEx, which can be translated and used in any programming language.

If the word checked is found to be a chord, the third step of the algorithm is to transpose the found chord from the input key to the current key. To do that we used the second algorithm for transposing, which we will explain it later.

After the chord is transposed, there will be HTML markers applied to the returned chord, to make it look like a chord. If the checked word is not a chord, it will create a string with all the words found before a chord, and this string will be marked with HTML markers as lyrics.

When the algorithm ends, all the information processed is parsed into a string of words and tags and saved into the database. When the saved song is opened, it loads all the information it needs from this string, in this way the algorithm should run only once.

The Chords Transposing Algorithm

This algorithm has as an input the original key, the chord, and the current key. Its job is to find the equivalent chord of the given chord in the current key. To do so we have created a dictionary [10] with all the keys and their chords, also a dictionary with all the keys and their notes (Figure 7).

Each key has its order for chords, so the first step of the algorithm is to check if the given chord is making part of given key chords. If it is so, it will return the position of the chord in the original key, and the chord found in the same position for the current key. When a chord is transposed, it will be saved in a dictionary, so it will not be a need to translate that chord again.

The problem is when the input chord is not making part of original key chords, so the algorithm needs to transpose it as a simple note. Each note has a value, for example, the first one is A and has the value 1, but after is A# and Bb which have equal value, 2. So the algorithm must decide which one of them to use.

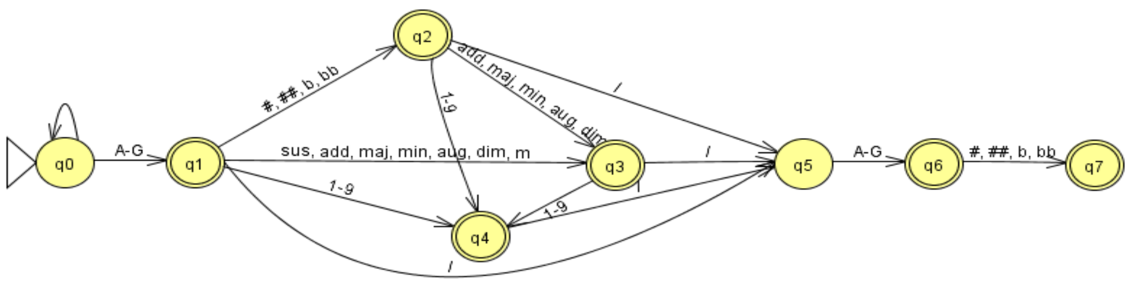


Figure 6: FSA for standard chords.

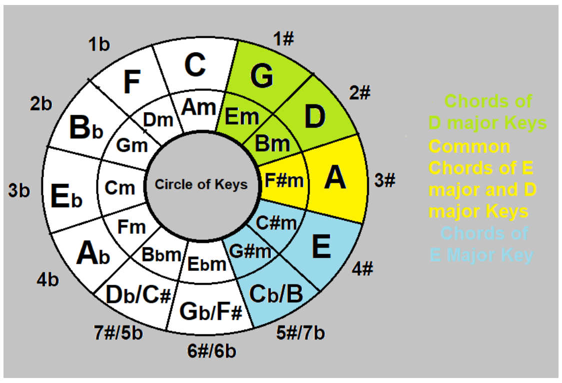
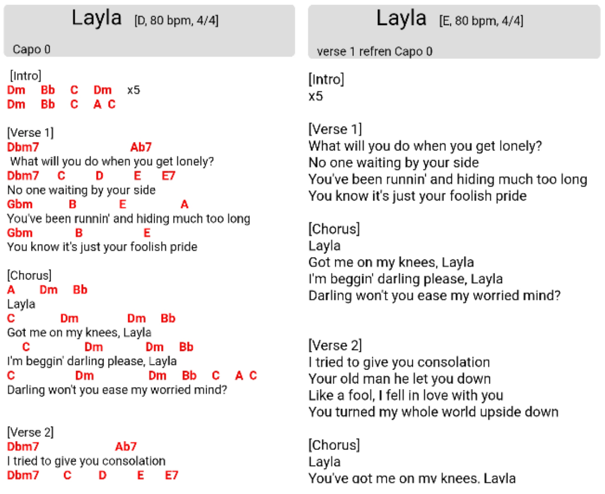


Figure 7: The circle of keys (adapted from [11]).

For that, we created a dictionary with all the keys and their signs # or b. so if the current key has the sign # it will choose A#.



```
Layla [D, 80 bpm, 4/4]
Capo 0

[Intro]
Dm   Bb   C   Dm   x5
Dm   Bb   C   A C

[Verse 1]
Dbm7                     Ab7
 What will you do when you get lonely?
Dbm7   C      D      E    E7
No one waiting by your side
Gbm         B         E           A
You've been runnin' and hiding much too long
Gbm          B             E
You know it's just your foolish pride

[Chorus]
A    Dm   Bb
Layla
C         Dm          Dm   Bb
Got me on my knees, Layla
  C            Dm         Dm   Bb
I'm beggin' darling please, Layla
C           Dm          Dm   Bb   C   A C
Darling won't you ease my worried mind?

[Verse 2]
Dbm7                  Ab7
I tried to give you consolation
Dbm7    C     D      E    E7
```

```
Layla [E, 80 bpm, 4/4]
verse 1 refren Capo 0

[Intro]
x5

[Verse 1]
What will you do when you get lonely?
No one waiting by your side
You've been runnin' and hiding much too long
You know it's just your foolish pride

[Chorus]
Layla
Got me on my knees, Layla
I'm beggin' darling please, Layla
Darling won't you ease my worried mind?

[Verse 2]
I tried to give you consolation
Your old man he let you down
Like a fool, I fell in love with you
You turned my whole world upside down

[Chorus]
Layla
You've got me on my knees, Layla
```

Figure 8: Text formatted with and without chords.

Finally, based on the same RegEx, we can decide if a word is a chord or not. When a chord is found it will delete everything, including white spaces, and newline markers until it finds a word which is not a chord.

Music Theory

Part of the theory of music used in the development of the application consists of detecting and manipulating a song's chords. Any song is made up of a sequence of notes with different duration and height. These notes are made up of several tones and sub-tones, which may differ depending on the sound source. In turn, several notes form chords, which can take different forms, depending on the style of the song.

A simple example is the difference between a minor chord and a major one, a minor chord has on the first position the base note, on the second position a note with 3 semitones higher than the base note, and on third position a note with 7 semitones higher than the base note, whereas a major chord has on the second position a note with 4 semitones higher than the base note.

A group of chords forms a key, following simple rules that can be seen from the Circle of Key (Figure 7). Each Key is based on a basic chord, which also determines the height of this key.

By detecting chords, the position of the chord is also determined in its key, or whether or not the chord is part of this key, which is very useful when we want to change the height of a key, the chord taking the value of the chord that is on the same position in the new key.

SYSTEM TESTING AND EVALUATION

Usability testing of our application was performed by means of an experiment that involved 26 users, aged between 19 and 63, with 73% of the users under 30 years old. The sample of users was selected among members of a Pentecostal church, after a Sunday service. Questions were designed with answers on a Likert scale from 1 to 7.

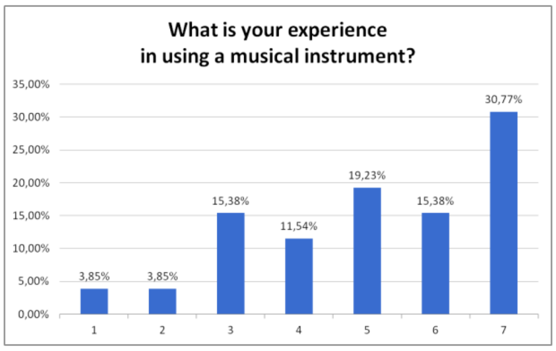


Figure 9: The extent of users' experience with musical instruments.

From the mobile devices use point of view, all users stated they used the mobile phone very frequently (score 7, meaning very frequent use, obtained from 80% of the users).

Given the specific of the app, we were also interested in the degree of experience in working with a musical instrument (Figure 9), as well as the existence of previous experience with similar application to the one proposed. The results are presented in Figure 10.

It can be noted that all users with experience in using musical instruments (except for 1), also had experience with similar apps.

The answers to item "I consider that using the app allowed me to focus better during the artistic representation." of the questionnaire were scored 6 and 7 by all participants, and only 2 users considered the information displayed to be confusing.

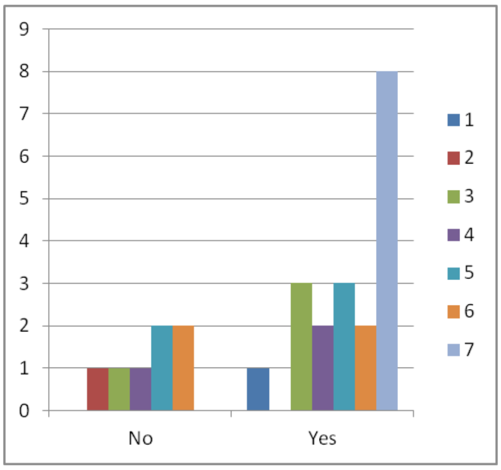


Figure 10: Distribution of users according to experience in using a musical instrument and experience in using a similar app.

The users' feedback was positive, among the comments we received, we mention: "I liked introducing new songs straight from the phone, without using a computer.", "Automatic switch of scales is great", "Easy to use, I like the ability to change colors to musical accords".

Also, it helped us gather some ideas regarding new features to be included in the application, such as: "It should offer a way to connect more devices together synchronously", "Export words together with musical accords in pdf format".

CONCLUSIONS

MyBand can easily ensure the minimum necessary for any band, filling its goal of replacing the dossiers, and providing extra features to help the musicians in their daily activities. It offers a community database of songs so every user can share their songs.

Our users are satisfied with the fact that the system requires no devices other than the mobile device for all the functionality, and that the chords transposition and detection function does not use a web service, this makes it much faster than other applications.

It is very helpful for bands to create their list of songs for concerts and rehearsals, and it is user-friendly with a simple and intuitive user interface.

It offers also the possibility to connect multiple devices for real-time sessions, being a very helpful feature for instrumentalists.

The goals for the near future are to implement all the missing feature and make it the most complete mobile application for musicians.

A goal that has not been met at the moment is that the live session does not work independently from the internet, which sometimes makes it work slower than we want because of the bad connection.

We want for this functionality to find a solution that does not use the Internet. We also want access to a database of songs with massive and legal content in the future, so users do not have to add too many songs.

REFERENCES

1. Jason Kichline. Onsong for android. https://www.indiegogo.com/projects/onsong-for-android#/ Accessed:2019-06-10.

2. Descriere songbook. https://www.linkesoft.com/songbook/index.html Accessed:2019-06-10.

3. Planning center music stand. https://planning.center/music-stand/ Accessed:2019-06-10.

4. Use case diagram. https://www.smartdraw.com/use-case-diagram/ Accessed:2019-06-12.

5. Activity diagram for use case. https://sis.binus.ac.id/2016/12/13/activity-diagram-for-use-case/ Accessed:2019-06-12.

6. Shaili Dashora. Regular expression in c#. https://www.c-sharpcorner.com/UploadFile/955025/regular-expression-in-C-Sharp/ Accessed:2019-06-13.

7. OMG: Unified Modelling Language Superstructure, version 2.0, ptc/03-0802, 2003.

8. LiveUML: https://liveuml.com
9. Modular Architecture: https://www.webopedia.com/TERM/M/modular_architecture.html
10. Musical chords and keys: http://www.guitaristsource.com/lessons/chords/keys/
11. Circle of Fifths: https://www.libertyparkmusic.com/the-circle-of-fifths/
12. ChordPro: https://www.chordpro.org/
13. draw.io: https://www.draw.io/
14. HCI and Design: http://www.nixdell.com/classes/HCI-and-Design-Spring-2017/Lecture14.pdf