ADA: Mobile application for music dictation

Agustín Pardo
Licenciatura en Jazz y Música Creativa
Universidad Tecnológica del Uruguay
Uruguay
agustin.pardo@utec.edu.uy

Franco Wanseéle
Instituto de Computación
Universidad de la República
Uruguay
franco.wanseele@gmail.com

Martín Nogueira
Instituto de Computación
Universidad de la República
Uruguay
maitinn2@hotmail.com

Regina Motz
Instituto de Computación
Universidad de la República
Uruguay
rmotz@fing.edu.uy

***Abstract*—ADA is a mobile application for automatic generation of music dictations guided by the teacher's settings. The application is designed to meet the training needs of undergraduate students in music performance. It responds to the need for a tool developed in open source software that allows the teacher to guide the training of his students by minimizing the effort dedicated to the production of the dictations and maximizing the training possibilities of the students. The first version of the ADA application developed in open source JavaScript for the generation of melodic and rhythmic dictations is presented.**

***Index Terms*—Educational mobile application, generation of melodic and rhythmic dictations**

I. Introduction

Melodic, rhythmic and harmonic dictations are practices that confront students of music performance degrees with several challenges, one of the main ones being the identification of musical elements through audition. In order to facilitate these learning processes, the design of the courses has a high component of face-to-face exercises in dictation. For this, the teacher must first design a dictation that is aligned with the characteristics he/she wishes to practice with the students and then reproduce it with an instrument. This work methodology requires a constant effort on the part of the teacher to design different dictations and needs to be present to reproduce it with the students. On the other hand, there is a strong character of subjectivity, since dictations designed by different teachers, in spite of training the same aspect of a course, will not necessarily present the same level of difficulty. This presents a clear disadvantage at the moment of evaluating a student, since his performance will strongly depend on who develops the evaluation and by whom he was previously trained.

Considering also the modality of classes through videoconference platforms, the reproduction of the dictation through an instrument involves sound distortions, interferences or even unforeseen cuts or slowdowns in the reproduction due to failures in the internet connection. A possible alternative is for the teacher to record the dictations with his instrument and send it to the students, who will download it to their devices and then listen to the dictation and transcribe it to a staff, which they will send as a file to the teacher to be corrected and finally the teacher will return the document indicating the mistakes they have made. This way of working is extremely tedious both for the teacher, who must record their dictations in an acceptable quality in a highly routine way, and for the students, who must wait a considerable time until they receive their corrections.

In response to this issue, the proposal arises to incorporate technology to transform face-to-face dictations into asynchronous dictations, minimizing the production effort invested by teachers in the design and creation of the dictations, but at the same time valuing the individuality of each student.

This is done on the basis of the development of a melodic, rhythmic and harmonic dictation generator tool. This tool should allow the person in charge of a course to publish configurations focusing on the difficulties he/she wishes to be trained and the students enrolled in that course can access the generations of dictations with those guidelines as many times as they wish, obtaining new dictations each time but all with the same level of difficulty. This is also an advantage from a pedagogical point of view because it ensures that regardless of who the teacher is who puts together the dictation, all students will have access to the same level of difficulty. This is also an advantage from a pedagogical point of view because it ensures that regardless of who is the teacher who creates the dictation, all dictations will be equivalent in their complexity for each course module. When changing course modules, students find dictation options generated with the new difficulties.

Supported by the current wide access to mobile technologies, such as cellular phones, our proposal consists of a mobile application for automatic generation of musical dictations. This paper describes the first version of the application called ADA [1] that generates melodic and rhythmic dictations.

II. Related work

ADA is an interactive software mobile application that allows teachers to express the aspects they want to train with

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

dictations without the need to have any programming knowledge. Other existing applications, such as www.theoria.com, Tenuto, or Good-Ear, are limited in the specificity of the possible aspects to train, noticing after a while of use that no progress is made in the training with new challenges. On the other hand, proposals for the use of programming languages, such as the use of Haskell [2], which allow ways of composing music through algorithms, are a tool with great potential for generating melodic, rhythmic and harmonic dictations, being able to achieve different specificities and levels of difficulty. but they have the limitation of being oriented to be used by people with a strong knowledge of programming.

Fig. 1 shows the comparison of the ADA Auditory Training application, with respect to other existing proposals that were familiar to the teachers of the working group. Comparisons are with respect to "Live music programming in Haskell" [7], Trubadur [8], and "Music Theory Web" [9]

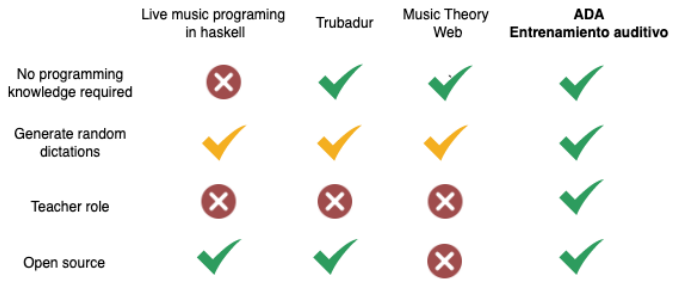

Fig. 1. Comparison of ADA with other tools.

III. Methodology

For the development of the ADA Entrenamiento Auditivo App (named below ADA) [2],[3] a co-production methodology was followed.

The participation of teachers and students occurred from the initial phase of the conception process and also during the development stage. Professors and advanced students of the Jazz and Creative Music career of the Universidad Tecnológica de Uruguay acted not only as qualified informants, domain experts, but also as users actively involved in proposing the application's functionalities, its frontend design and the evaluation of the information flows supported by ADA.

It is important to note that neither the computer team handled the musical concepts nor the language necessary to develop ADA, nor did the music team handle the computational concepts or the language necessary to initially achieve a fluid exchange of the application's functionalities. This was solved with basic training of the computer team in music terminology complemented by sessions of detailed presentations of the design decisions of the selected algorithms to the music team in a scenario adapted from methodologies of *living labs* [4]. This added to the use of broad interview methods, focus groups, and also user stories, all carried out from the first stages, allowed to achieve a successful dynamic of exchanging requirements and application functionalities.

On the other hand, for the development stage, the use of agile methodologies was of great benefit [5]. Incremental delivery is of great value due to delivering functionalities in each increment allowing validate what has been developed with people who know the subject. In addition, agile methods allow you to easily adapt to changes, something essential when the development team does not have full knowledge of the main issues of the software to be developed. As a principle, it is also highlighted that the development methodology is oriented to people and not to processes. This means that people on the team should be allowed to work without rigorous processes in place. Finally, it is important to maintain simplicity, both in the software and in the development process, which allowed not adding complexity to the project.

IV. Design

The main focus in the design was to address aspects of the flow of use, always prioritizing that the application is intuitive and easy to understand for users. The flow of use was thought from both the music and the educational side at the same time. From the music side, the objective was focused on identifying which were the parameters that needed to be configured from the application, in order to set up dictation configurations. From the educational side, emphasis was placed on the flow of use in which students access the generated dictations since this involves a series of steps that imply: dictation reproduction, access to the solution and grading.

A. Creation of dictations

Figure 1 shows the two screens for course creation that contain all the musical concepts necessary for the configuration of a dictation (intervals in major/minor, melodic turns, start/end notes, different melodic elements with their respective priority, number of musical beats and rhythmic cells) in addition to containing more general aspects, such as the name or description of the dictation. Such concepts were addressed in the low fidelity prototype, the purpose of presenting these screens is to be able to address some design decisions made, detailed information on the design stage can be found at [6].

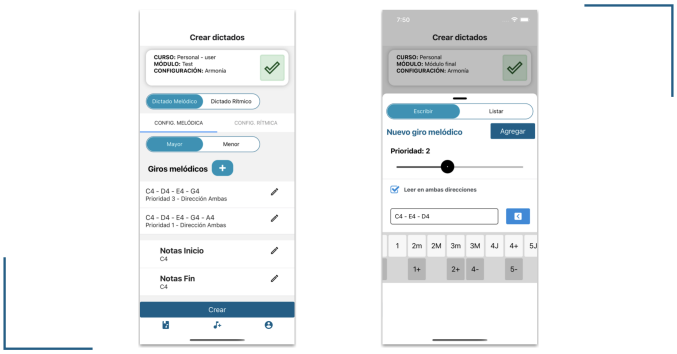

Fig. 2. Example of course creation.

The main design objective of this functionality was to be able to group the functionalities of adding, editing and deleting each of the concepts present in a single screen. This was

done to avoid redirecting the teaching user to a new screen, in which he/she has to perform actions and in case of error go backwards in the navigation flujo, thus preventing the user from losing focus on what he/she really wants to configure. This goal was achieved by using a design element called *botton sheet* which consists of a sliding panel that appears from the bottom of the screen. This will appear every time you want to add, edit or delete any element of the configuration, giving the user the feeling that it always remains on the same screen and in case of any mistake simply close the element by sliding downwards. down.

The second screen in Figure 1 shows the same sliding panel, which is used when you want to add new melodic turns. Three sections can be distinguished at this point. The first one consists of a *switch* which you choose whether you want to write a new melodic turn or select one from the list provided by the system. In both cases, a priority is assigned on a scale from 1 to 5 using a *slider*, a graphic element that allows the user to slide the indicator circle over the line, setting a value within a certain range. Finally, if it was decided to write the melodic turn, there is a keyboard and a box where the user's typing will appear.

The keyboard was designed in conjunction with advanced music students, where each key represents an interval which will depend on whether minor or major intervals were chosen in the melodic setting. For more details on the details of configuration see [6].

Given a large number of elements to configure and how complex it can be for the user to have to think about priorities or even write melodic turns, is that the system implements an inconsistency check. This is implemented with the principle of error prevention and handling in mobile systems in mind, which has to be informed to the user in some graphical form. By way of addressing how inconsistencies are handled, Figure 2 shows an inconsistency alert in the dictation setup.

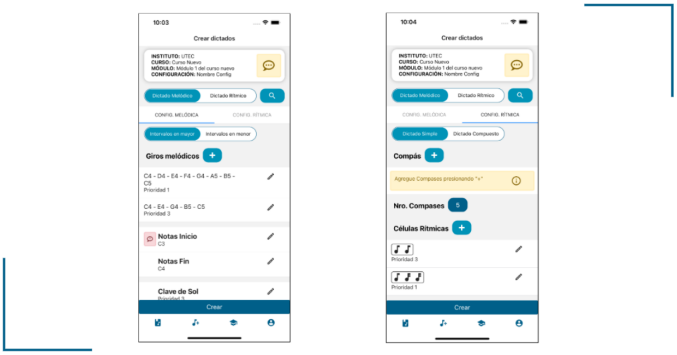

Fig. 3. Example of inconsistency in the creation of a course.

Contrasting the screens in Figure 2, you can see two warnings (shown in yellow) corresponding to the general setting and the rhythmic setting. These warnings are visible when there is missing information and it must be completed by the user. On the other hand, within the melodic configuration an error can be identified in the starting notes (shown in red color). Those are shown when there is a wrong data entered that is not consistent with the rest of the configuration. In these cases, by clicking on this indicator the system will display a window containing an informative text about the particular inconsistency and possible solutions to correct it. The incorrectly entered data that are controlled are the following:

- There can be no starting notes or end notes which do not belong to a melodic turn, due to it would be impossible to begin or end on such a note.
- At least one clef (G or F) with priority greater than zero must exist.
- There must be at least one tonality with a priority greater than zero.

B. Play dictation

When the student enters the application to play a dictation, they first see the application logo, which consists of *sticks* (drum sticks) as a musical element, giving the impression that it is hitting each other. This sound is present at the beginning of each dictation generated by the application, which works to indicate the musical beats, which can be seen as the previous preparation to listen to a musical dictation. This is the reason why this musical element was chosen as the logo, due to we want to generate the same feeling of preparation when starting the application and seeing the logo. In addition, musically, the sound played by the sticks at the beginning of each dictation is the reference given to the listener of the speed at which the dictation will be (Figure 4).

Fig. 4. ADA application logo.

The screens presented in Fig. 5 are those corresponding to those presented, in the flow of use that a user with the role of student who wishes to reproduce a dictation has. The first screen shows the modules and within these the dictation configurations, based on the course that is selected. For each of the dictation configurations, the description that was established is added. This will be useful for the student to know what musical elements he will be training.

Once a dictation configuration is selected, the user is redirected to the list of dictations. These are identified with a number corresponding to the order in which they were generated. Additionally, useful information is presented, such as the clef in which the dictation is, the tonality and, in case

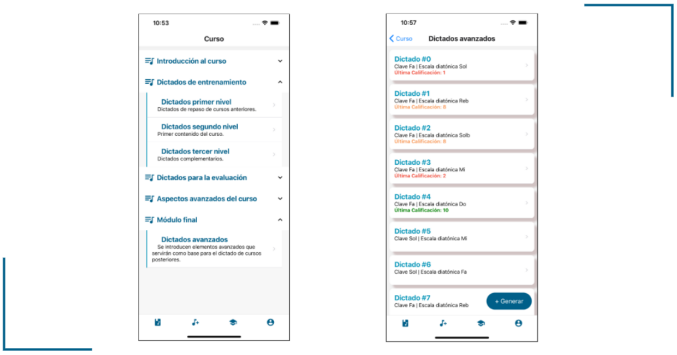

Fig. 5. Play dictation in ADA application.

the student has performed the dictation, the calification, from 1 to 10, obtained in the last attempt will be shown. At this point, in addition, a color scale composed of green, orange and red is used, corresponding from the highest to the lowest grade and thus allowing the user to quickly identificate his performance. These colors were chosen taking into account to generate a good contrast with the background of the screen, for this reason the use of yellow was discarded.

Finally, by accessing a dictation, the screen where the student will carry out his training with the dictation playback is obtained. Figure 5 shows two screens corresponding to the reproduction of the dictation and the access to its solution respectively, (the range of errors in the solution is calculated according to the length of the dictation, see details in [6]). Each of the buttons of these screens (except the first one) opens a modal in which the student can indicate if his errors were mostly errors related to the rhythmic aspect, melodic aspect or both equally. This information will be useful, both for students and teachers, to know which aspect to focus on when continuing with the auditory training.

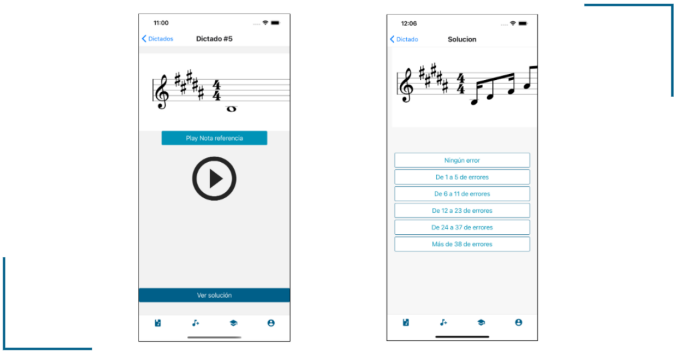

Fig. 6. Play dictation.

V. Implementation

A. Technology

The system architecture was developed as a service-oriented: SOA platform. This allows reusing elements due to the interfaces offered by the services, which communicate through the network; and allows having distributed systems in which each of the components are independent services. A multiplatform development was chosen in order to make an application that works on both iOS and Android with a single code.

For the selection of the technology to be used in the Front-end and taking into account the requirement of being open source with the objective that it can be maintained and grow adapting to the needs of its users. For this reason, the most popular frameworks in the community were investigated. This with the objective that in the future, the technology in which the application is developed will not be an impediment when incorporating new functionalities by interested people. This led to consider two frameworks developed by technological giants: on the one hand Flutter developed by Google and on the other React-native, developed by Facebook. The main and most important advantage of Flutter is that it develops a single project for all operating systems, which means a considerable cost and production time reduction. However, considering the advantage that the development team already had experience in React-native, we chose to use it. While for the Backend we were in the dilemma between FLASK technology of the python language and NODEJS technology of the javascript language, one of the most popular languages available.

ToneJS, a JavaScript library that provides an API for generating MIDI files, is used to generate multimedia files that contain the dictations. This acronym stands for Musical Instrument Digital Interface (MIDI) and files with this extension store data containing a series of instructions that can be translated into sound. These instructions are messages that tell a certain emulated instrument what are the musical notes, their duration, the playing strength and the modulations of the sound parameters, information necessary for the generation of dictation sounds. This library has a very wide configuration in order to control the configuration of the generated rhythmic and melodic dictations. On the other hand, the main parameters to configure are the musical figures and the notes associated to each one of them.

Finally, the last parameter to be configured is the instrument on which the melody will be played. Together with teachers of the music institute we decided to take as default instrument the acoustic grand piano (which is included as an instrument in the ToneJS library) due to it does not add complexity to the dictation. For future versions of the application it is contemplated to be able to select different musical instruments in case it is of value for auditory training. In addition to the parameters mentioned above, there are others such as playback speed, sound volume or even the ability to add a delay at the beginning of each note, which were left fixed.

Once the dictation is listened to, the next step will be for the student to access the solution plotted on a pentagram, for which a new set of tools is introduced. VexFlow3 is an open-source API which allows, by entering certain musical parameters, the rendering of a pentagram. This API contains a ”little language” called EasyScore, which is a simple way to write musical elements that are then translated into the elements that VexFlow needs to write in a pentagram. It should

be noted that this tool generates the pentagram to be rendered in a web browser, so WebView4 is used to display it in the mobile application.

B. Algorithms

The dictation generation is done on the server side due to it could count with more resources and thus obtain better performance results, as well as it facilitates the access to the database to store the results. First, just like setting up a configuration of dictations in the application, the random generation is divided into two stages: the generation of rhythmic elements and the generation of melodic elements.

The generation of rhythmic elements is the first step in the generation of dictations since it is based on the results of this stage that the melodic elements are generated. As input of this first stage we have the rhythmic cells, the beats, the number of beats and if the dictation will be simple or compound, having, the first two elements, an associated probability. Knowing these data, the first step is to obtain a set of rhythmic cells corresponding to a beat, and for these to be valid the values of the figures must be taken into account and contrasted with the value of each pulse. At this point, we must keep in mind that from the beat we have that the numerator indicates the number of pulses within a beat, and the denominator is the value of each pulse. Knowing this, rhythmic cells will be taken randomly (taking into account the probability associated with each one) and from this, the following is calculated: for the set of rhythmic cells obtained so far, the value of each figure belonging to each rhythmic cell is added and multiplied by the denominator. This will result in a number that has the following interpretation: if it is less than the value of the numerator is that another rhythmic cell must be randomly added to the set and the function is recursively called again to redo that calculation; If it is equal to the value of the numerator, it means that the set of rhythmic cells corresponds to a beat, which makes the set generated so far part of the final dictation; if the number is greater than the numerator, it means that the values of the set are larger than the value that has to have a beat, so the generated set is invalid. In this last case, we proceed to remove the last rhythmic cell obtained and call the function again recursively and thus randomly obtain another different rhythmic cell, repeating this step until a valid set is obtained. It should be noted that this procedure is possible since it was stipulated that the system will have rhythmic cells that will always complete exactly one or more pulses and never a fraction of them. The procedure explained above will result in a set of rhythmic cells corresponding to a beat, so these steps are repeated iteratively until the indicated number of beats is completed. In this way, a set of rhythmic cells is obtained, which is translated into a set of musical figures from which the melodic dictation generation stage begins.

The melodic dictation generation step consists of assigning a note to each figure obtained in the previous step, following the rules of the dictation configuration so that the result is coherent. The data that are used are the following: melodic turns, the range (which is not set by the user but is a parameter set in the system), starting notes, end notes, clefs, keys, tonalities, reference note and number of figures, the latter being a data obtained from the generation of rhythmic elements. As initial steps are obtained, in a completely random way, the starting and fin notes. Obtaining the key in which the dictation will be generated is done randomly but taking into account the probability associated with each of the keys, and with this last result is that the range of the dictation is obtained. With the previous data, the tonality is obtained, again randomly and taking into account the priority of each one, and based on this last result, two actions are executed: on the one hand, transform the melodic turns to the selected tonality and on the other hand, this last result is adapted to the range, modifying the heights of the melodic turns accordingly. In case the established melodic turns are not able to adapt to the range, the steps will be repeated, with another different key, until obtaining melodic turns that are within the range of notes established by the range. From what is obtained, we proceed to transform the reference note, the starting note and the fin note to that tonality, and 'these last two elements adapt them to the range. With these results, and having the starting note as the first note of the dictation, all the elements used by the dictation are defined. The next three steps consist of selecting the rest of the musical notes in an appropriate way and thus assigning each of them to the figures obtained in the previous step. Due to space problems, the reader is directed to find the details of the dictation generation algorithms in the publication of [6].

VI. Conclusions and Future Work

As a result of the work developed, a generator of rhythmic and melodic dictations was created that differs from the existing ones because of its versatility to be configured, allowing to achieve dictations with different levels of difficulty and specially designed to train specific aspects defined by the teacher. Additionally, the methodology used for the design and development of the application, carried out within the framework of co-production methodologies [3].

Functional tests verify that the functionalities behave as expected. This is associated with the business logic, i.e. that the results obtained are correct and consistent from a musical point of view. Within this type, the ones that were most present were unit tests. These were executed by the developers once a specific functionality was closed, with the objective of ensuring that the code unit, within a given component, provides the appropriate results. On the other hand, once a flow of use was closed, integration tests were executed, where different functionalities of the software are tested as a group.

The user acceptance tests were done in two stages: most of the project consisted of virtual meetings where the application and the functioning of what had been developed so far were shown. In the final step of the project, along with the release of the application, users were given test perfiles so that they could test the main fluices of use of the application. Through such activity, valuable feedback was obtained with which the following updates of the system were left raised.

Regarding the results provided by the application, it is observed that both the dictations and the solutions provided are coherent and respond to the rules established in the dictation settings. On the other hand, comparing different results, a good management of randomness is observed, which makes the dictations generated by the application varied. This point solves the problem expressed by some music students, who expressed that there are platforms, oriented to auditory training, that after a brief use are repetitive and not very varied.

The ADA application generates quality dictations, making it a good resource to be used in the teaching of music courses. This allows auditory training to be carried out independently by the students (based on the course previously set by the teacher) and thus dedicate the classes to address issues that necessarily require interaction between student and teacher. This has the objective of working as a complement and not with the goal of substituting the auditory trainings in the classes. On the other hand, the application was used by advanced music students with the objective of setting up an initial course that would be useful for students who wish to enter in careers related to music and must face an admission test. In this way, the application will work as an extra complement for these students to acquire the basic musical training they must have for admission. To these configurations can be added a greater dificulty, including other types of melodic elements, which would be useful for students with greater knowledge to continue perfecting their musical hear. The released application has the necessary features so that the main flow for creating and generating dictations can be used by both teachers and students. Although ADA is at the first version of the application, it has a solid development as a base, on which to add and build new functionalities. This makes us consider the application as a contribution to music education, both formal and informal.

The next steps for future work is to extend the generation to harmonic dictations.

REFERENCES

[1] ADA repository: $https://github.com/francowanseele/Entrenamiento\_Auditivo$

[2] E.B. Sanders and P.J. Stappers,“ Co-creation and the new landscapes of design,” Codesign, 4(1): 5-18, 2008.

[3] C. Sjodin, P. Kristensson, “Customers’ experiences of co-creation during service innovation,” International Journal of Quality and Service Sciences, 4(2): 189-204, 2012.

[4] A. Folstad, “Guide to online applications for user involvement in living lab innovation,” Information Systems and Technology for Organizations in a Networked Society: 34: 96-108, 2013.

[5] I. Sommerville, “Software Engineering (9.a ed.),” Addison-Wesley, 2011.

[6] F. Wanseéle, L. Nogueira, “Entrenamiento auditivo para músicos ADA - Aplicación de Dictados para la Armonía.” Tesis de grado. Universidad de la República (Uruguay). Facultad de Ingeniería, 2021.

[7] H. Thielemann, “Live music programming in Haskell”. Halle, Germany, 2013.

[8] Link, https://trubadur.si, TRUBADUR. Last visit:19/9/2022.

[9] Link, https://www.teoria.com, Music Theory Web. Last visit:19/9/2022.