VITHEA-Kids 3.0

Slvia Maria Matos Timteo
silvia.timoteo@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

September 2020

Abstract

Language disorders can make it difficult for kids to understand what people are saying or to express their own thoughts and feelings through spoken and written language. Among others, Dyslexia and Specific Language Impairment (SLI) are such language disorders. However, the skills affected by each one are different. Whereas dyslexia is characterized by difficulties in word recognition, spelling, writing and decoding with a genetic basis, SLI is characterized by difficulties in different aspects of language, such as lexical retrieval, phonology, morphology, syntax, semantics and pragmatics. Individuals with these disorders face numerous adversities in their daily life, which can be minimized by solving exercises recommended by therapists. Some efforts have been made to develop applications that can provide these exercises and be suitable for its users. An example of such application is VITHEA-kids 2.0, which is an European Portuguese application for helping children with Autism Spectrum Disorder and their caregivers. This application provides a way of creating exercises and allows the customization of some aspects of the platform to make it more suitable for the children needs (e.g. show letter always in upper-case), also has a talking animated character. Since this platform is a promising one, in this thesis, we intend to extend VITHEA-kids 2.0 with new exercises in order to reach children with other learning disabilities, namely dyslexia or SLI. Regarding dyslexia, a research of exercises to deal with reading and spelling problems was made. Regarding SLI, the exercises to deal with relative clauses comprehension were provided for a specialist in syntax acquisition. However, implementing new types of exercises in this platform were not possible, since the code was not flexible to the addition of new types of exercises. This way, the present thesis also focus on refactoring the whole platform, back-end, caregivers application and childs application, in order to allow new exercises to be added to VITHEA-kids 2.0. Thus, in this thesis we create VITHEA-kids 3.0 to support new types of exercises for children with dyslexia or SLI, and also we make the addition of further types of exercise easier for developers.

Keywords: Learning disabilities, SLI, Dyslexia, refactorization, VITHEA Kids

1. Introduction

The adequate development of language is one of the fundamental factors to childhood development, since language is needed to understand others, to express our thoughts or make ourselves understood. Worldwide, around 15-20% of the population has a language-based learning disability [1], which consists of problems with age-appropriate reading, spelling, and/or writing. The symptoms may be visible since the early years of life, which often makes people with a learning disability frustrated and, in extreme levels, could lead to depression [4]. Many of the difficulties that have been identified in children with some impairment in learning could be strongly connected to a language disorder, such as dyslexia and SLI. Dyslexia affects around 70-80% of the population with a language-based learning disability, and consists of a specific learning difficulty typically characterized by difficulties in word recognition, spelling, and decoding, with a genetic basis[1]. In Portugal, according to an epidemiological study, 5.4% of the primary school-age children were diagnosed with dyslexia[5]. SLI is characterized by difficulties with learning and usage of language, thus a child with SLI does not develop speech and skills in the expected way. Grammar, vocabulary and, frequently, phonology are learned with difficulty. Also, when reading or listening, they may only focus on a few content words and then they deduce the meaning from them, thus contributing to a poor comprehension [1]. Both dyslexia and SLI are disorders that require early interventions to minimize the inherent problems, such as for poor spelling and problems with syntax. This way, with appropriate support and intervention, people with learning disabilities can achieve success in school, at work, in

[1] https://dyslexiaida.org/frequently-asked-questions-2/ last access 01/06/2020

relationships, and in the community. For many parents it is not always possible to get their children in contact with therapists, often because they can not afford it. There are already IT solutions with the purpose of helping children with their learning difficulties, however these solutions are paid or include paid features, mostly applications are not in European Portugues and have few customization options. In order to fulfill these gaps, VITHEA-kids, an educational application for helping children, inspired by the needs of children with ASD and their caregivers was developed. VITHEA-kids 2.0 was released where a deep reformulation of the used technologies was made and new modules were implemented such as prompting and reinforcement strategies [3]. Hence, the primary purpose of this work is to look for the main types of exercises that are used to improve skills that children with SLI and dyslexia struggle with, and enrich VITHEA-kids 2.0 with them. However, despite Vithea having an excellent infrastructure, the code developed to implement the exercise does not allow adding new types. This way, there will be a great focus on refactoring all modules that concern the exercises, to ease the implementation of new types by developers.

2. VITHEA-kids 3.0

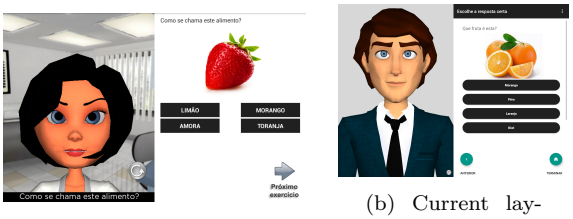
We identified an exercise to be implemented in VITHEA-kids. However, there were still bugs to be solved, as well as features to be finished to achieve an usable platform. Moreover, adding new exercises had become a big challenge by itself, given the way that the existing features were implemented. For this reasons, the present section describes all changes that had lead to the production of a new version, VITHEA-kids 3.0.

2.1. Implementation of features

Whenever a child finish a set of exercises we would like to present him with a GIF as a reinforcement. However every GIF we tried to insert was always static in our application. For this reason, we integrated Glide [2] to make possible the proper load of this image format into VITHEA-kids 3.0. The interface did not support long answers. We created new layouts that could accommodate long answers like sentences, as can be seen in Figure 1.

Also, there was no feedback when an answer button is pressed. To solve this, we opted for changing background color, whenever the button is pressed. In order to set different colors, it was necessary to create two different backgrounds, one for pressed state and another for unpressed state as we can see in Figure 2. Since the platform should be able to help users recover from errors, an error message was added whenever the user type their credentials incorrectly. Also, an encouraging message was added for the the users to create exercises and/or classes when there are no exercises.

[2]https://github.com/bumptech/glide last access 30/03/2020



(a) Previous layout.

(b) Current layout.

Figure 1: Layout for exercises of multiple choice.

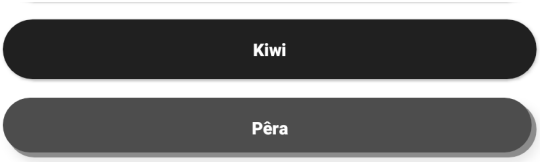


Figure 2: Button with the word "Pra" pressed

Before this current version of VITHEA-kids a therapist had requested a button to repeat the uttered sentences. So in VITHEA-kids 2.0 there was actually a button that when pressed, it did what was expected. However, duo to the new version of Unity, that button was not there. Therefore, it was necessary to add it again. Since the animated characters is an imported Unity project, it was necessary to modify this project in order to add the button and also its behaviour.

In an exercise, each image was loaded immediately before being displayed. As a result, there were network requests during the process of solving the exercises, making the user to wait in every exercise. The same therapist that had requested the button, had also pointed out this as a problem. In order to avoid this, all images are now loaded when selecting a new class of exercises, which often delayed it, as shown in Figure 3.

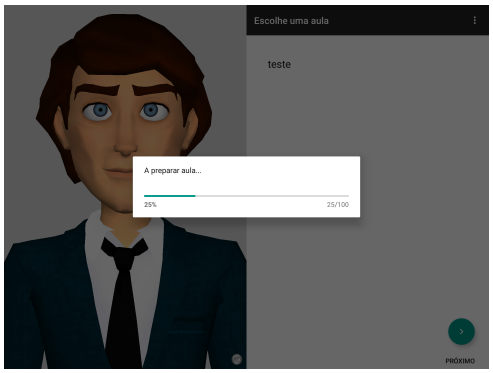


Figure 3: Loading images

A confirmation dialog is now shown whenever the

user tries to return to the main menu like in Figure 4 to assure if the user is sure about interrupting the class of exercises, since doing this way it will not be saved any progress of the current class.

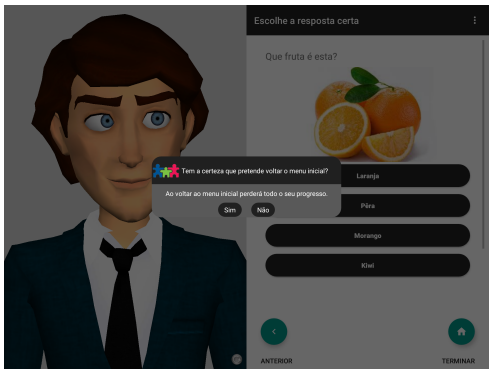


Figure 4: Loading images

VITHEA-kids 2.0 made available the five types of **prompting** for the multiple choice exercises: read the remaining answers, change color of the right answer, change size of the distractors, scratch the distractors and hide distractors.The caregiver could combine some types of prompting. However, these strategies was not implemented for the multiple choice exercise whose answers were images. Therefore, it was necessary to implement them in the child's application, as well as the prompting strategies (the prompting is always given or the prompting is given when the child selects a wrong answer). In the first version of VITHEA-kids, there was the request of having the exercises's text in upper case. This way, it makes sense provide freedom of choice for caregivers regarding the way in which the text is displayed in the child application. This type of customization has been implemented in the first version of VITHEA-kids, however it was no longer available in the last version. Therefore, we implemented it again. So now, the exercise's text can be displayed in the way it was typed or in upper case, regarding the caregiver's choice. Also, in the first version of VITHEA-kids, it was possible for the caregivers to sort the exercises of a class or choose to sort them in a random way. However, as that customization was also no longer available we implemented again. During visualizations of VITHEA-kids and also during the development of the features mentioned above, some bugs were identified, such as, the logout was not working properly, it was possible at times to find the exercises and other informations of the child logged before. Sometimes, after failing to login, the buttons did not answer to the user's interaction. Theses bugs are now fixed in the current version of VITHEA-kids.

2.2. Refactoring VITHEA-kids

The whole process of refactoring the different components of VITHEA-kids 2.0 architecture such as database model, back-end, caregiver's application and child's application to ease the implementation of new exercises will be presented.

2.2.1 Back-end

VITHEA-kids 2.0 uses Play Framework to develop an API in Java. Play's architecture is RESTful by default, which means it uses HTTP requests to GET, PUT, POST and Delete data. At its core, Play is based on the MVC pattern, that separates an application into three main logical components: the model, the view, and the controllers. Using this framework, to expose a REST API is simple as the developer just need to match an HTTP verb (GET, PUT, POST and Delete) with an associated action defined in a custom controller in a configuration file named 'routes'. In this subsection, the focus are the controllers, part of the system that handles the requests from both caregiver's application and child's application, such as register an exercise, delete an exercise, get all exercises and so on. There is one main controller that deals with the exercises, which has the three methods: register exercise (called whenever the caregiver creates a new a exercise), edit exercise (called whenever the caregiver wants to edit an existent exercise) and delete exercise (called whenever the caregiver decides to delete an existent exercise).

Each type of exercise has its own characteristics, which requires different implementations to register, edit and delete exercises. However, in VITHEA-kids 2.0, the different implementations are handled by a single method, where there is a conditional statement for each type of exercise, which is not a scalable solution. The same happens with the other methods mentioned before. This way of implementation is not desirable for the following reasons:

- For every new exercise, it will be necessary to include the code in each method to register, edit and delete, which makes the method increasingly larger, and consequently the code becomes harder to maintain.

- It is difficult to implement new types of exercises and vary existing ones since there was no independence in the implementation of the various types of exercises.

It is possible to avoid these problems by defining classes that encapsulate different operations algorithms. A design pattern that encapsulates in this way is called **Strategy** [2].

Therefore, following the **Strategy** design, the diagram shown in Figure 5 was obtained.

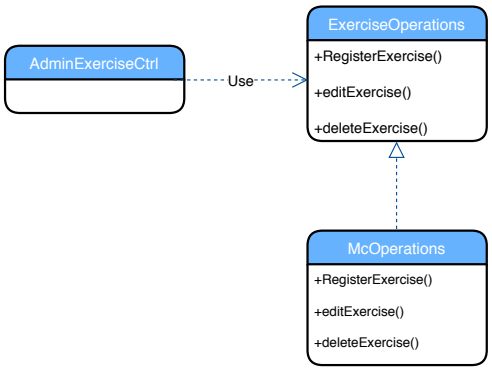


Figure 5: Refactoring controller using Strategy design.

The `AdminExerciseCtrl` class is the controller and it is responsible for register, delete or edit an exercise. These operations strategies are not implemented by the controller. Instead, they are implemented separately by a subclass that implement the interface `ExerciseOperations` class. `ExerciseOperations`'s subclasses may implement different strategies. For example, `mcOperations` implements a strategy for creating, editing and deleting a multiple choice exercise. Currently, to implement a new type of exercise, it is just necessary to create a class that implements `ExerciseOperations`.

However, there was still a problem: the controller `AdminExerciseCtrl` can not predict what subclass of `ExerciseOperations` should instantiate since it depends on the exercise type. The class only knows when use the operations, not what kind of operations. This creates a dilemma: The class must instantiate subclasses, but it only knows about interface, which it cannot instantiate. We solve this by using the **Factory Method** pattern [2] since it encapsulates the knowledge of which `ExerciseOperations`'s subclass to create and moves this knowledge out of the controller `AdminExerciseCtrl`.

2.3. Caregiver's application

As already mentioned, the caregiver's application was developed in Angular 2. Angular 2 follows a components-based approach to web development. Components are essentially reusable UI building blocks that are easy to test and reuse. They correspond a sets of screen elements that Angular can choose among and modify according our program logic and data. Angular uses also services that are Typescript classes, usually responsible for fetching data from the server, validating user input and so on. They can be developed for specific tasks needed in a given component. In the case of this application, they are mostly used to communicate with the server in order to fetch or send information.

Through the analysis of these concepts and the front-end's code the following architecture of VITHEA-kids 2.0 arises.

In short, there are several concerns that could be split out into small components to make the code more readable, less complex and extensible.

The `Exercise component` is responsible for listing all created exercises and preview them. In terms of code, to preview each exercise, the component checks if every possible element of an exercise is not null. If it is not, the component display it. So, to preview another type of exercise, it would be necessary to add new validations for the new fields. Once again, this would introduce more complexity and less flexibility, making the code harder to understand and maintain. Therefore, a solution would be to delegate functionality to smaller components.

Having said this, the following subsections describes how the `AddExecise` and `Exercise` components were broken into smaller ones. Furthermore, all changes have the goal of creating a different component for every type of exercise in order to make easy the new exercises implementation, since this way the developer could only focus on the exercises that he/she intends to implement, without the need of understanding how other types of exercises are implemented.

2.4. AddExercise Component

Since every kind of exercise has its creation form and its logic, a good solution would be to create one component for each form.

In light of this, we created a `AddComponent` for each exercise that encapsulates each specific form. Now the `AddComponent` is more generic since its main responsibility is to render the proper creation form according to the type of exercise selected by the caregiver. This is possible thanks to an Angular's feature that allows to inject components inside other components by adding a tag (a reference to a component).

Also, we created an `editExercise` component that follows the same logic of `AddExercise`, which means there is a different component to edit each exercise.

2.5. Exercise Component

We created two new components, one for each type of multiple choice. Each one has only the code (logic and HTML template) related to the corresponding exercise type. Now, the `Exercises Component` just iterates over all exercises and injects a component according to exercise type.

3. Child's application

The child's application consists of an Android application where the child can solve the exercises previously created by his/her caregiver.

When the child plays VITHEA-kids for the first time, a login screen is displayed, in which the child's credentials should be typed. After that, a menu is displayed to select a given class.

So far, each class may be composed of two kinds of multiple choices. In each exercise, the child can skip to the next exercise or return to the previous exercise, if it exists. Also, a reinforcement screen is shown between exercises displaying an image whenever the child answers correctly to an exercise.

At the architecture level, all interactions with the application are handled through two Activities classes. An activity class is usually associated to a screen with a graphical user interface and it dictates the UI and handles the user interaction with the smartphone screen [3]. Each activity has a XML Layout file configured, which contains all the UI elements.

Following the login, the main Activity is created. This activity is associated with all the remaining screens, such as the exercise screen, menu screen and so on. The layout associated with the main activity is divided in half, in which one of the sides is the container of the unity character and the other is a container of the following views[4]:

- List of classes associated with the child;
- Current exercise;
- Reinforcement.

As a consequence of that, all functionalities inherent to these views are concentrated in a single activity class. This was implemented this way in order to avoid unity's character loading whenever there is a screen change. However, this way of implementation leads to many code lines in a single class, witch turns the code almost unreadable and more bug prone. Also, the time to add any feature is affected in negative way. Furthermore, changing a layout is almost impossible since there are big dependence between views. In other words, one change is some kind of view could involve unintended changes in other layouts. Keeping this implementation could make these problems worse when adding new kind of exercises. Therefore, it is not a flexible and scalable solution. So, refactor this activity was necessary in order to add exercises in a flexible way. To accomplish this, we followed an fragment-oriented architecture. A Fragment is a modular section of an Activity, that has its own life-cycle. It might be seen as a sub-activity since it has its own layout and its own behaviour, which enables more modular activity design[5]. Moreover, it is possible to combine multiple fragments in a single activity to build a multi-panel UI. Furthermore, with fragments, adding a new exercise is easier since it is just necessary to create a fragment and its respective layout. Also, the layouts for the new types of exercises are easier to create, since each layout is independent of the other layouts. Therefore, the main activity layout is now divided in four main areas, as we can see in the Figure 6:

- **Animated character**, which occupies half of the screen. As it is supposed to be always present, independently of the child interaction, it was declared in a static way;
- **Toolbar**, which provides the application settings;
- **Fragment place holder**, which defines an empty container layout to be set by the main activity. The activity replaces the current fragment by another that could be the "list of classes", "the multiple choice image", "the multiple choice text" or "the reinforcement";
- **Navigation view** which allows the child navigate between exercises. This view only appears once a class is selected.

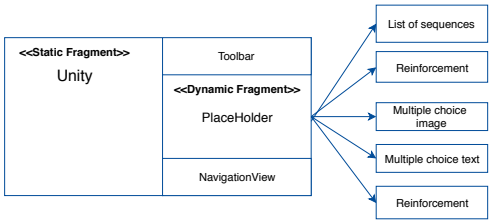


Figure 6: Android activity layout

Given this division, there is now a fragment for each type of exercise. The reinforcement and list of classes of exercises were also implemented through fragments.

In addition to the use of fragments, it was necessary to refactor the model classes. That should reflect the model classes of the server, considering the refactorization described in Subsection **??**. Once the model was changed to support inheritance, the child's application model should also support to guarantee the automated mapping of the data coming from the server. Firstly, inheritance was implemented using the `Exercise` class as the basis and having `MultipleChoiceExercise` as a subclass. Also, at the exercise class,the annotations presented in the were added to accomplish the au-

[3] https://developer.android.com/guide/components/activities/intro-activities.html

[4] https://developer.android.com/reference/android/view/View.html.

[5] https://developer.android.com/reference /android/app/Fragment.html

tomated mapping. This annotations allow deserialization. When deserializing, the actual code being executed will know the expected class, through the property `dtype` of JSON that comes from server.

Using fragments leads to a more modular code. To add a new exercise, it is just necessary to create a fragment and its respective layout. Layouts for the new types of exercises are easier to create, since each layout is independent of the other layouts.

4. New type of exercise

After refactoring, we focused on the last goal of this thesis, which consists in extending the VITHEA-kids 3.0 with another type of exercise. In the related work, a versatile type of exercise was identified that can be used by children with dyslexia or with SLI, depending on how the exercise is created by the caregiver. The identified exercise consists of a question and an image, where it is possible to select an area of the image. The selected area corresponds to the area where the child should touch to finish the exercise successfully on the mobile device. The implementation of this exercise was another contribution to this thesis.

5. Evaluation

In this section we are going to show the improvements in the three main components of VITHEA-kids, back-end, caregiver's application and child's application. This way, we are able to compare the scalability of both architecture. Also, in this section we are going to describe new exercises that are now possible to create with the new exercise type, which match our main goal of helping children with SLI or dyslexia. Furthermore, to reinforce the improvement inherent to the new architecture, we give a description of a new type of exercise implemented by a researcher, as well as her feedback when implementing the exercise, using the new architecture.

5.1. Selection image exercise Implementation

After all the research, a new type of exercise was identified that was not possible to achieve with the current exercises was identified, which we named as selection image. With this type of exercise, it is possible to create exercises for children with SLI as well as for children with dyslexia. This exercise should allow the caregiver to define a specific area to be taken as correct, inside an image. At the child's application, the child has to touch inside the area previously defined by the caregiver. As we intend to show how easy it has become to implement new types of exercises we we are going to illustrate the main differences between the two architecture, using the implementation of this new type of exercise as example.

5.1.1 Back-end

For the new exercise to be implemented it is necessary to store in database an image, a question, original width and height of the chosen image and also selection's coordinates. The selection is a rectangle performed inside the image. The original width and height have the goal of keeping track the ratio of selection's coordinates. In terms of implementation, it requires alterations in the entity classes and in the controllers. Regarding the database, that will not be covered, since it is a reflection of the entity classes.

As it was already mentioned, Play framework offers a way of inheritance (Single table). With this feature, we have now the common fields to all exercises in the class `Exercise` and we created another one, extending from the `Exercise` with the specific properties of this exercise (image, question, width, height, selection's coordinates). In respect to the controller responsible for the exercise's operations (create, edit, delete), to implement the operations for the new exercise we need first to add to our factory the name of the new type in order to instantiate the appropriate object (the one that has the right operations). After that, we implement the class `SelectionImageOperations` with the logic associated to the behaviour of each operation, not forgetting that the class should extend from `ExerciseOperations` (an interface that defines the behaviour of the exercise controller). This way of implementation assure us more flexibility and also more independence between different types of exercises.

5.1.2 Caregiver's application

For the caregiver application, in respect to the exercise, we have three screens, one to create an exercise, one to edit and another to preview the exercise.

In the new architecture, we need to create a folder to contain three new angular components. Those components are `add-exercise-selectionImage`, `edit-exercise-selectionImage` and `show-exercise-selectionImage`. The `add-exercise-selectionImage` is a component that implements all necessary logic to create an exercise that allows us to draw a selection inside an image. So in this component, we have an HTML file where we implemented the user interface that allows the caregiver create the exercise and store it in the server, we have a CSS file where we develop the style for the exercise form and a typescript file where we implement the logic that deals with the caregiver's input and send it to the server.

The `edit-exercise-selectionImage`, as well as the `show-exercise-selectionImage` follow the same logic, which means every one has three

files Html file, CSS file and typescript file. However, the `edit-exercise-selectionImage` was implemented to allow the caregiver edit an exercise that was once created by hi. The `show-exercise-selectionImage` was implemented to allow the preview the exercise already created.

After creating these three components it is necessary to edit HTML files of four more general components which are `add-exercise`, `edit-exercise`, `exercise` and `add-sequence` to make possible that the new components mentioned before could be instantiated when needed. In `add-exercise` HTML file we introduce an conditional statement that verifies if the exercise type is a selection image and inside that statement we reference a component `add-exercise-selectionImage`. In the `edit-exercise` we do exactly the same we did in `add-exercise`.

The exercise can be previewed when the caregiver lists all exercises or when checking a given class. Therefore, to make possible `show-exercise-selectionImage` being created and instantiated it is necessary to add the exercise name into `exercise` and `add-sequence`, in order know which component is going to be created. We can do this way, since every exercise corresponds to a component, and each component knows how should be rendered and which information needs. With this solution we have more independence between exercises.

5.1.3 Child's application

For the child's application we need to make possible the information retrieving about the new exercise from server and also create an user interface where the child can solve the type of exercise. So, we are going to describe how it would be made by using the previous architecture and how it is done through the new architecture.

Regarding the exercise implementation itself, a block code is necessary to be added that contains all logic behind the possible interactions when solving the exercise into `VitheaKidsActivity` class (mentioned in Section 4). Also, it is necessary to add a XML file to implement the user interface regarding to new type of exercise. This kind of implementation is not scalable for new exercises since this way we just increase the complexity of the `VitheaKidsActivity`class, making it more difficult to maintain it and extend it.

Since so far we have all types of exercises extending from a class `Exercise` that contains all common properties, in the back-end, we need to reflect this into child's application. Therefore, it is necessary to create a class that extends from `Exercise`, in the child application, in the same way it was made in the back-end. Also, in the class `Exercise` of `Child application` it is required the line highlighted in Listing **??**, in order to make possible the mapping between the JSON that comes from the back-end and the respective class.

Regarding the exercise implementation itself, it is necessary to create a Fragment with the logic associated with selection image exercise. Also, it is necessary to create a XML layout, to implement the layout of this exercise. This way we assure more independence between different types of exercises and consequently this solution becomes more scalable, in terms of adding new types of exercises.

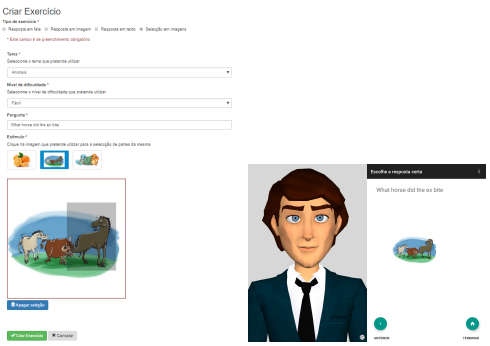


Figure 7: Form to create selection on an image(left) exercise to be solved by a child (right).

5.2. Selection in Image exercise

In the previous section we focused in describing the main differences when implementing the new exercise using the previous architecture and the current one, implemented by us in order to show our contribution in this thesis, regarding the goal of refactoring VITHEA-kids and making the implementation of new exercises an easier task. After refactoring VITHEA-kids 2.0, a new version of it arise with a new type of exercise. This way, in the present section, the potential of this new type of exercise will be shown as well as how we can create exercises identified in Related work chapter that could be user in therapies for children with dyslexia or SLI.

5.2.1 Exercises for SLI

As mentioned at section 3.1, an expert in the field hypothesized an exercise to be used by children to train relative clauses, since comprehension and production presents itself as a problem for them. In short, this exercise consist of a sentence illustrating the idea of that sentence. This exercise leads the child to do what is in the sentence. For example, in the sentence "Que cavalo que o boi mordeu?" ("What hore did the ox bite?), the child has to touch in the part of the screen that corresponds

to the horse that suffered the action. With the new type of exercise, several similar exercises can be created to practice and improve relative clauses comprehension, as shown in the Figure 7.

5.2.2 Exercises for dyslexia

Regarding dyslexia some exercises of an exercise manual for children with dyslexia can be replicated with the new type of exercise. For example, the exercises, mention in section 3.2.4, can be produces in VITHEA-kids 3.0, as we can observe in the image. This way, we might conclude that with this new type of exercise we can provide children with dyslexia a list of different exercises to practice skills where they feel more difficulties, such as the exercise presented in Figure 8 .

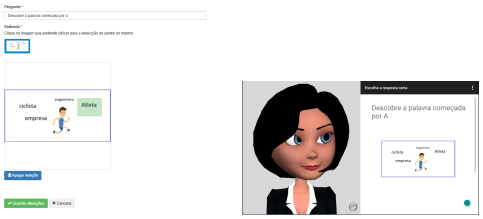


Figure 8: Form filled with information of an exercise for childs with dislexia image(left) exercise to be solved by a child (right).

5.3. Word Naming exercise

Beside the exercise we had implemented, another researcher from INSIDE have provided VITHEA-kids 3.0 with a new type of exercise, more specifically a word naming exercise. In this exercise the child has to say orally the name that appears in the screen. Also, this exercise has to be created previously and added to a class by the caregiver. Therefore, the researcher had to pass for every component of the VITHEA-kids, back-end, caregiver's application and child's application. Based on the feedback received and having into account that the researcher had already knowledge about de the previous architecture, the exercise was simple to implement and not so confused how it would be in the previous architecture. Also, the implementation was not a very time-consuming process.

6. Conclusions and Future Work

Worldwide, there are children with some learning disability, such dyslexia and SLI. It is not always possible to provide these children with therapies, many times duo to financial problems. Therefore there is a need to develop a solution that addresses this problem. Taking advantage of the technology and the enjoyment felt by the children when playing with mobile devices, to create an application that seems like a great solution. With this view in

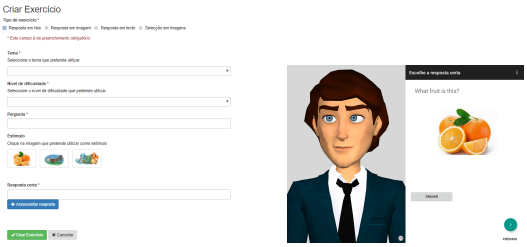


Figure 9: Form to create a word naming exercise (left) exercise to be solved by a child (right).

mind, we found VITHEA-kids 2.0 a promising platform to achieve our main goal of using technology to create exercises, to help children with learning disabilities, namely dyslexia and SLI, that could be solved through a mobile device. VITHEA-kids 2.0 was an application inspired by the needs of children with ASD, allowing the caregiver create exercises to be solved by their children in order to fight some kind of impairment associated to them. Also, this application is free, easy to use and it is European Portuguese.

However, when exploring this application more closely, we found features incomplete, as well as bugs in VITHEA-kids 2.0 that had to be addressed. Also, after analysing the components that compose VITHEA-kids 2.0 we have also realised that a profound reformulation was needed, since there was no flexibility for the implementation of new types of exercises. Hence, VITHEA-kids 2.0 was submitted to a process of refactorization in every part of the application (back-end, caregiver's application and child's application). With this refactorization, VITHEA-kids 3.0 is now more extensible to new types of exercises.

In addition to the refactorization, we also developed a new type of exercise that allows caregivers define an area on a image to be taken as correct. In consideration of what was mention at the related work, this type of exercise is useful to create exercises with focus on dyslexia as well as on SLI.

Also to reinforce the benefits achieved with the refactorization, we got a very positive feedback by a researcher of INSIDE about the implementation of a word naming exercise.

However there was space for improvement, regarding new functionalities adding. After searching for therapies and exercises used by specialists to fight difficulties felt by children with dyslexia, we found out some useful ideas, that could be implemented in VITHEA-kids. These ideas are based on Orthon-Gillingham approach (OG) and were discussed with a specialized on the field. This ideas implies to implement:

- A Tutor, whose the main goal would be teach the sounds of syllables by providing children

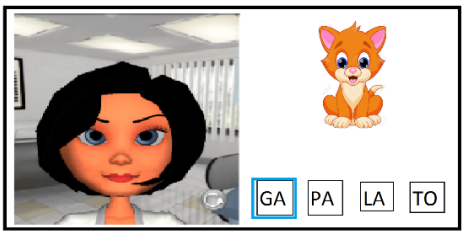
with dyslexia a set of syllables and their respective sounds.

- An exercise where could be possible to join syllables to form the word that matches the image presented on the display.

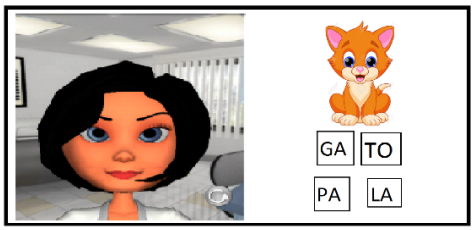
Regarding the OG approach, since it is as being **explicit**, and after having discussed with a therapist, it has raised the idea of introducing a tutor in VITHEA-kids. The main goal of this tutor would consists of providing children with dyslexia, a set of syllables and their respective sounds as in Figure 10. However, synthesize the syllables sounds could be a challenge since in Portuguese the syllables can vary according to the word. For example, the words cama (bed) and casa (home) have the same syllable ca, however, the sound in each word is different.

Figure 10: Tutor teaching SA syllable

Also, another proposed exercise, by the therapist, following the OG, with special emphasis on multisensory teaching, requires a few senses, such as, vision, audition and touch, all at same time. The main focus consists of turning children more aware of basic sound units of language. Regarding the details of the exercise, it consists of joining syllables to form the word that matches the image presented. In other words, an image is presented, as well as, a set of syllables (syllables that belongs to the word and others that not, to distract the child). To solve the exercise, the child has to drag each syllable close to the image in order to build the word that match the image. When a syllable is being moved, the corresponding sound is uttered. So far, this exercise is also not supported by VITHEA-kids.



(a) Exercise



(b) Exercise Solved

Figure 11: Syllable Exercise

References

[1] D. V. M. Bishop and L. Laurence B. *Speech and Language Impairments In Children.* Psychology Press, Purdue University, Indiana, USA, 2014.

[2] G. Erich, H. Richard, J. Ralph, and V. John. *Design Patterns: Elements of Reusable Object-Oriented Softwares.* Addison-Wesley Professional, 1994.

[3] C. P. B. Filipe, M. L. T. R. M. d. S. Coheur, and J. A. R. P. Sardinha. An application to help children with communication disorders. Master's thesis, Instituto Superior Técnico, 2017.

[4] S. M. Handler. Dyslexia: What you need to know. *Contemporary Pediatrics*, 33(8):18, 2016.

[5] A. P. Vale, A. Sucena, and F. Viana. Prevalência da Dislexia entre Crianças do 1 . º Ciclo do Ensino Básico falantes do Português Europeu. pages 45–56, 2011.