

## **VITHEA-Kids 3.0**

**Sílvia Maria Matos Timóteo**

Thesis to obtain the Master of Science Degree in

## **Information Systems and Computer Engineering**

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur  
Prof. José Alberto Rodrigues Pereira Sardinha

### **Examination Committee**

Chairperson: Prof. Francisco João Duarte Cordeiro Correia dos Santos

Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur

Member of the Committee: Prof. Alberto Abad Gareta

**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, caregiver's application and child's 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

---

## Resumo

---

As perturbações da aprendizagem pode provocar nas crianças uma maior dificuldade na compreensão ou até na expressão dos seus pensamentos e sentimentos. Estas dificuldades são notórias tanto na expressão oral como escrita. A dislexia e a perturbação específica da linguagem (PEL) são exemplos de perturbações na aprendizagem. No entanto, as capacidades afetadas por cada uma são diferentes. Enquanto a dislexia é caracterizada por dificuldades no reconhecimento, ortografia, escrita e decodificação de palavras com base genética, o PEL é caracterizado por dificuldades em diversos aspectos da linguagem, como recuperação lexical, fonologia, morfologia, sintaxe, semântica e pragmática. Indivíduos com esses transtornos enfrentam inúmeras adversidades no seu dia a dia, que podem ser minimizadas com a resolução de exercícios recomendados por terapeutas. Alguns esforços têm sido feitos para desenvolver aplicações com integração de exercícios adaptados às necessidades dos utilizadores. Um exemplo desse tipo de aplicação é o VITHEA-kids 2.0, que se trata de uma aplicação em português europeu cuja finalidade é ajudar crianças com perturbação do espectro do autismo bem como os seus cuidadores. Esta aplicação oferece uma forma de criar exercícios e permite a customização de alguns aspectos da plataforma para torná-la mais adaptada às necessidades das crianças (por exemplo, mostrar a letra sempre em maiúsculas), também possui uma personagem animada falante. Por se tratar de uma plataforma promissora, nesta tese pretende-se estender o VITHEA-kids 2.0 com novos exercícios para chegar a crianças com outras dificuldades de aprendizagem, nomeadamente dislexia ou PEL. Em relação à dislexia, foi feita uma pesquisa de exercícios para lidar com problemas de leitura e ortografia. Em relação ao SLI, os exercícios para lidar com a compreensão de orações relativas foram fornecidos por um especialista em aquisição de sintaxe. No entanto, a implementação de novos tipos de exercícios nesta plataforma era bastante complexo, uma vez que o código não era flexível e extensível. Desta forma, a presente tese foca-se também na refactorização de toda a plataforma, back-end, aplicação do cuidador e aplicação da criança, de forma a permitir que novos exercícios sejam adicionados ao VITHEA-kids 2.0. Assim, nesta tese criamos o VITHEA-kids 3.0 para suportar novos tipos de exercícios para crianças com dislexia ou SLI, e também tornamos a adição de outros tipos de exercícios mais fácil para os programadores.

**Palavras-chave:** Perturbações na Aprendizagem, PEL, Dislexia, refactorização, VITHEA Kids



---

## Acknowledgments

---

This dissertation represents the final phase of my academic life, being the culmination of years of many experiences and many learning not only at the academic and professional level, but also at the personal level.

First of all, I want to thank my family, especially my parents, Fernando Marques and Luísa Matos for all support and unconditional love, my dear brother, Ricardo Timóteo who always believed me and never let me give up. My brother was always there to me listening my problems and making them smaller. Also, I could not forget to thank my cousins, Samuel Machado and Bruno Machado who helped me keeping in the line. They are simply wonderful. Without no doubt my family played a fundamental role in making my dream of getting a master degree come true.

This work would not have been possible without my dissertation supervisors Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur and Prof. José Alberto Rodrigues Pereira Sardinha who actively provided me support, guidance, strength along this journey. Also, they have always believed me even when I did not believe in myself and have taught me more than I could ever imagine.

The prof. Ana Lúcia deserves a great deal of thanks for giving me all the insights and background related to Specific language impairment. Her help was crucial to find an exercise to help children with SLI work out some of their impaired skills.

I am grateful to Vânia Mendonça and Soraia Alarcão who taught me a great deal about both scientific research and life. They were absolutely tireless during the whole process. I am feeling so lucky for having such a great friends.

I also want to thank Beatriz Santos for making me believe in myself and for being there when I need. I could not have done this without her.

Last but not least, to my friends, my sincere and deepest thanks, for the presence, for the advice and for the strength they gave me in the most difficult moments, but also for the celebration of the victories.



*To my parents: Luísa Matos e Fernando Marques and to my dear brother:  
Ricardo Timóteo*



---

## Contents

---

<b>Abstract</b>	<b>iii</b>
<b>Resumo</b>	<b>iv</b>
<b>Acknowledgments</b>	<b>v</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Listings</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xv</b>
<b>List of Acronyms</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Goals/ Contributions . . . . .	2
1.3 Document Structure . . . . .	2
<b>2 Background</b>	<b>5</b>
2.1 Specific Language Impairment . . . . .	5
2.2 Dyslexia . . . . .	5
2.3 VITHEA-Kids – Virtual Therapist for Teaching Children . . . . .	7
2.4 Discussion . . . . .	10
<b>3 Related Work</b>	<b>11</b>
3.1 Exercises to assess relative clauses comprehension in children with SLI . . . . .	11
3.2 Dyslexia . . . . .	12
3.2.1 Orthon-Gillingham Approach . . . . .	13
3.2.2 Designing word exercises to children with dyslexia . . . . .	14
3.2.3 Software for children with dyslexia . . . . .	15
3.2.4 Exercises from a book . . . . .	17
3.3 Discussion . . . . .	18
<b>4 VITHEA-kids 3.0</b>	<b>19</b>
4.1 Implementation of features . . . . .	19
4.2 Refactoring VITHEA-kids . . . . .	22
4.2.1 Back-end . . . . .	23
4.2.2 Database Schema . . . . .	24
4.2.3 Caregiver's application . . . . .	26

4.2.4	AddExercise Component . . . . .	28
4.2.5	Exercise Component . . . . .	29
4.3	Child's application . . . . .	29
4.4	New type of exercise . . . . .	31
4.5	Discussion . . . . .	31
<b>5</b>	<b>Evaluation</b>	<b>33</b>
5.1	Selection image exercise . . . . .	33
5.1.1	Back-end . . . . .	33
5.1.2	Caregiver's application . . . . .	35
5.1.3	Child's application . . . . .	37
5.2	Selection in Image exercise . . . . .	39
5.2.1	Exercises for SLI . . . . .	39
5.2.2	Exercises for dyslexia . . . . .	39
5.3	Word Naming exercise . . . . .	39
5.4	Discussion . . . . .	40
<b>6</b>	<b>Conclusions and Future Work</b>	<b>41</b>
6.1	Conclusions . . . . .	41
6.2	Future work . . . . .	41
	<b>Bibliography</b>	<b>43</b>
	<b>Appendices</b>	<b>45</b>
	<b>Appendix A Current RM model</b>	<b>45</b>
	<b>Appendix B Proposal RM model</b>	<b>47</b>
	<b>Appendix C Register exercise controller</b>	<b>49</b>

---

## List of Figures

---

2.1	Caregiver's module - Creating a new exercise. . . . .	7
2.2	VITHEA-kids - Child's module. . . . .	8
2.3	Architectural changes from VITHEA-kids to VITHEA-kids 2.0 [20]. . . . .	9
2.4	Animated characters used . . . . .	9
3.1	Sentence-picture exercise [9]. . . . .	12
3.2	Sentence-scenario exercise [9]. . . . .	12
3.3	A possible image to use in exercises for children with SLI [19]. . . . .	13
3.4	Exercises of DysEggxia - (a) add a letter, (b) remove a letter (c) cut into words and (d) change a letter [28] . . . . .	16
3.5	Left: lesson, right: training exercise [29]. . . . .	17
3.6	Found words starting with the letter A (left) and found images that contains the sound "LH" (right) [30]. . . . .	17
3.7	Found the repeated drawing (left) and fill the blank spaces with the correct word (right) [30].	17
4.1	Layout for exercises of multiple choice. . . . .	20
4.2	Button with the word "Pêra" pressed . . . . .	20
4.3	Loading images . . . . .	21
4.4	Loading images . . . . .	21
4.5	Types of promptings . . . . .	22
4.6	Refactoring controller using Strategy design. . . . .	23
4.7	Differences between the current model and the proposed model concerning exercises . .	25
4.8	Front-end architecture . . . . .	27
4.9	Android activity layout . . . . .	30
5.1	Data structure for the new exercise using the previous architecture . . . . .	34
5.2	Form to create selection on an image(left) exercise to be solved by a child (right). . . . .	38
5.3	Form filled with information of an exercise for child with dyslexia image(left) exercise to be solved by a child (right). . . . .	39
5.4	Form to create a word naming exercise (left) exercise to be solved by a child (right). . . .	40
6.1	Tutor teaching SA syllable . . . . .	42
6.2	Syllable Exercise . . . . .	42
A.1	Class diagram of injection procedure in the service provider. . . . .	46
B.1	Class diagram of injection procedure in the service provider. . . . .	48





---

## Listings

---

4.1	Using factory pattern to instantiate the operations for multiple choice exercises. . . . .	24
4.2	Anotations that allow inheritance . . . . .	26
4.3	addExecise component . . . . .	27
4.4	Current addExercise Component . . . . .	28
4.5	Current Exercise Component . . . . .	29
4.6	Deserialization anotations . . . . .	31
5.1	Method to register an exercise. . . . .	34
5.2	Factory to generate the object of concrete class based on the exercise type name. . . . .	35
5.3	Current Exercise Component. . . . .	35
5.4	AddExercise component's HTML with the tag for selectionImage exercise . . . . .	36
5.5	Exercise class of the child's application with the new fields . . . . .	37
5.6	Class Exercise changed to allow mapping. . . . .	38
C.1	Register exercise controller . . . . .	49



---

## List of Tables

---



---

## List of Acronyms

---

<b>ASD</b>	Autism Spectrum Disorder
<b>INSIDE</b>	Intelligent Networked Robot Systems for Symbiotic
<b>JPA</b>	Java Persistence API
<b>RM</b>	Relational model
<b>OG</b>	Orthon-Gillingham
<b>ORM</b>	Object-relational mapping
<b>L2F</b>	Spoken Language Systems Laboratory
<b>MVC</b>	Model-view-controller
<b>SLI</b>	Specific Language Impairment
<b>TTS</b>	Text-To-Speech
<b>VITHEA</b>	Virtual Therapist for Aphasia Treatment
<b>VITHEA-kids</b>	Virtual Therapist for Teaching Children 2.0
<b>VITHEA-kids 3.0</b>	Virtual Therapist for Teaching Children 3.0
<b>JSON</b>	JavaScript Object Notation)
<b>GAIPS</b>	Intelligent Agents and Synthetic Characters Group
<b>XML</b>	eXtensible Markup Language
<b>GIF</b>	Graphics Interchange Format
<b>API</b>	Application Programming Interface
<b>HTTP</b>	Hypertext Transfer Protocol



# 1

---

## Introduction

---

This thesis consists of extending an existing application inspired by the needs of children with Autism Spectrum Disorder (ASD) and their caregivers: Virtual Therapist for Teaching Children 2.0 (VITHEA-kids) 2.0 in a way that could also help children with other learning disabilities, in particular Specific Language Impairment (SLI) and dyslexia. However, to make this possible a profound reformulation of VITHEA-kids 2.0 was needed.

In this section, we present the motivation behind this thesis, the goals that we intend to achieve and, lastly, the structure of this thesis.

### 1.1 Motivation

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<sup>1</sup>, 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 [12].

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<sup>1</sup>. In Portugal, according to an epidemiological study, 5.4% of the primary school-age children were diagnosed with dyslexia [31].

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 [4].

---

<sup>1</sup> <https://dyslexiaida.org/frequently-asked-questions-2/> last access 01/06/2020

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 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 Portuguese 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 [8], as described in Section 2.3.

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 issue is well described at chapter 1 and 4. In 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.

## 1.2 Goals/ Contributions

The main purpose of this thesis is to provide an application that comprises exercises in European Portuguese for children with dyslexia and SLI. These exercises should be free and customizable according to its user's needs/preferences.

VITHEA-kids 2.0 is a promising system to help us accomplish our main goal, since taking advantage of its infrastructure will allow us to save time, helping us to focus on the implementation of exercises. Among other aspects, VITHEA-kids is in European Portuguese, free, customizable and designed taking into account feedback from therapists of individuals with ASD.

Considering our intention of providing helpful exercises for children with dyslexia and SLI, this thesis's goals are the following:

- Search for exercises that are normally used by specialists to help children with dyslexia and SLI to cope with their difficulties. Regarding SLI, this work is being supervised by a specialist in syntax acquisition who provided some exercises with the intention of dealing with difficulties that children with SLI face. The main goal of these exercises consists of helping children in the comprehension of relative clauses, since it is one of the main problems.
- Implement a solution that can be applied to different exercises used in therapies that could be as useful for dyslexia as for SLI.
- Refactor VITHEA-kids 2.0 to turn this application in a more extensible one, and thus make the implementation of new exercises, with focus in other learning disabilities, a less complex task. We also intend to conclude incomplete features of VITHEA-kids 2.0.
- Make VITHEA-kids 2.0 more flexible and clean to allow other researchers understand the code with less difficulty and also contribute to VITHEA-kids 3.0.

## 1.3 Document Structure

Firstly, we presented the motivation and the main goals of our proposal. In Chapter 2, we present some background concepts in order to better understand the context of this work (Dyslexia and SLI). In Chapter 3, relevant works about dyslexia and SLI, such as approaches and applications that deals



with these learning disabilities, are described. The purpose in this chapter consists of identifying which kind of exercise could be created using the existing features of VITHEA-kids 2.0 and also, which ones would require the implementation of a new type of exercise. In Chapter 4, We describe with detail which features of VITHEA-kids 2.0 were not concluded prior to this thesis and what we did to conclude them, as well as, bugs found and how they were fixed. Additionally, the process of refactoring VITHEA-kids 2.0 to achieve a flexible platform is described. In Chapter 5, an evaluation is presented, setting side by side both versions of VITHEA-kids, when implementing a new type of exercise. In this chapter, we also present feedback of a researcher of INSIDE, about her experience on implementing a word naming exercise by taking advantage of the new architecture. Lastly, the document ends with the Chapter 6, presenting some conclusions, highlights and future work.



# 2

---

## Background

---

In this chapter, to fully understand the context around this project, we provide several concepts related to SLI and dyslexia, which are the basis of this thesis. Therefore, in Section 2.1, we describe what SLI consists of, and which are the underlying problems; the Section 2.2 provides a historical context of dyslexia, the way it is currently seen and also the main difficulties identified in individuals with this learning disability; in Section 2.3, we give an overview of the platform VITHEA-kids 2.0 since we intend to integrate new exercises there.

### 2.1 Specific Language Impairment

According to Laurence B. Leonard [4] the term SLI “is applied to children who exhibit a significant deficit in language ability yet display normal hearing, age-appropriate scores on tests of nonverbal intelligence, and no obvious signs of neurological damage”.

The significant deficit in language showed by children diagnosed with SLI is linked to difficulties encountered in different aspects of language, such as lexical retrieval, phonology, morphology, syntax, pragmatics, and semantics [17]. Therefore, they tend to show a severe deficit not only in speech production, but also in sentence comprehension. Studies on the acquisition of relative clauses have shown that children struggles with comprehension and production of relative clauses [21]. Relative clauses are classified according to the syntactic movement. Subject relatives are derived by movement from the subject as in (e.g. “The girl [that draws the grandmother]”), whereas object relatives implicates movement from the object position (e.g. “The girl [that the grandmother draws]”) [21]. Comprehension and production of subject relatives is better when comparing to object relatives by children with SLI [10].

### 2.2 Dyslexia

Dyslexia is a specific learning disability that interferes with the acquisition and processing of language. It has been characterized by difficulties with accurate word recognition, poor spelling and decoding abilities [18]. This way, children with dyslexia present a significant set of alterations in reading and writing, which can lead to a negative impact in their school learning.

The first study of this problem was analyzed by an ophthalmologist, who believed that dyslexia was, in some way, associated with a disease of the visual system [11] in the late 19th century, in Great Britain. A British ophthalmologist, James Hinshelwood, also writing at the turn of the century, speculated that such difficulties with reading and writing were due to “congenital word blindness”, and for many years the dominant view was that dyslexia was caused by a cerebral disease or injury.

In 1887, a German ophthalmologist, Rudolf Berlin, was the first using the word “dyslexia” instead of the word blindness. “Dyslexia” has a Greek origin, meaning “difficulty with words” [1].

The first case of developmental dyslexia was stated by Pringle-Morgan in the British Medical Journal in 1896. Pringle-Morgan, a general practitioner, recognized as the father of developmental dyslexia, mentioned a child, apparently with a typical development, without any brain damage, who presented difficulties in reading. Based on this and other similar case, Morgan concluded that dyslexia has a congenital nature that is a condition present at birth, whether inherited or caused by environment [11]. The following excerpt describes the situation of the child mentioned before.

“Percy F.— a well-grown lad, aged 14—is the eldest son of intelligent parents. . . He has always been a bright and intelligent boy, quick at games, and in no way inferior to others of his age. His greatest difficulty has been—and is now—his inability to learn to read. This inability is so remarkable, and so pronounced, that I have no doubt it is due to some congenital defect. . . the greatest efforts have been made to teach him to read, but, in spite of this laborious and persistent training, he can only with difficulty spell out words of one syllable. . . The schoolmaster who has taught him for some years says that he would be the smartest lad in the school if the instruction were entirely oral. . . His father informs me that the greatest difficulty was found in teaching the boy letters, and they thought he never would learn them.”  
(Morgan, 1896, p. 1378)

In 1925, an American neurologist, Dr. Samuel T. Orton proposed the first theory about the causes that could explain specific reading difficulties. He argued that these difficulties were caused by a dominance of one side of the brain. Teaching strategies that he developed during his research are still in use today [11]. Since then, the number of studies focused on understanding dyslexia has increased.

Currently the most consensual definition comes from the International Dyslexia Association (2003), which states the following: <sup>1</sup>

“Dyslexia is a specific learning disability that is neurobiological in origin. It is characterized by difficulties with accurate and/or fluent word recognition and by poor spelling and decoding abilities. These difficulties typically result from a deficit in the phonological component of language that is often unexpected in relation to other cognitive abilities and the provision of effective classroom instruction. Secondary consequences may include problems in reading comprehension and reduced reading experience that can impede growth of vocabulary and background knowledge.”

In a study conducted in 2012, Rodrigues [22] points out the main difficulties that characterizes Dyslexia, as follows: inversion of letters in reading and writing, omission of words in reading and writing, difficulty converting letters into sounds and words, difficulty recovering sounds and letters from memory, difficulty learning meaning from letters and sounds.

---

<sup>1</sup> <https://dyslexiaida.org/definition-of-dyslexia/> last access 15/05/2018

## 2.3 VITHEA-Kids – Virtual Therapist for Teaching Children

VITHEA-kids [20], [8], developed at Spoken Language Systems Laboratory (L2F)<sup>2</sup> and also at Intelligent Agents and Synthetic Characters Group (GAIPS), is an application for helping children with ASD improving their language and skills. Furthermore, it is a tool that allows caregivers to create exercises and perform a set of customizations to meet the needs of each child. These exercises are aimed at improving vocabulary learning, word picture association or generalization, and emotions recognition. It was developed using the infrastructure of an in-house award-winning platform: VITHEA [24], which has the intention of helping people with aphasia through solving oral word naming exercises created by their therapists, presented by a talking animated character, called Catarina, using Text-To-Speech (TTS) [23]. VITHEA has two different modules: one where the caregiver manage the exercises, as well as his/her children, and another where people with aphasia can solve the exercises. So, the decision of using VITHEA to build VITHEA-kids was motivated by the fact that it has these modules, which allows the caregiver to create exercises and upload multimedia resources. Also, there are studies defending that children with ASD have a better performance doing the exercises along with an animated character [5], this way, choosing VITHEA was also motivated for having a talking animated character.

As VITHEA-kids is build on VITHEA, there are also two distinct modules, the caregiver's module and the child's module. On the caregiver's module, the caregiver can manage child users' accounts and exercises. There are also preference options, where the caregiver can customize some aspects of the interaction between the child user and the child module. As for the exercises, they consist of multiple choice characterized by a specific topic (e.g., "Animals"), a difficulty level (e.g., Introductory, Intermediate or Advanced), the question/instruction, the stimulus (optional), the correct answer and a set of incorrect answers (distractors), between zero and three, as shown in Figure 2.1. There are two variations of this multiple choice exercise: one in which the stimulus is an image and the answers are textual and another in which the stimulus is textual and the answers are images.

The screenshot shows a web-based form titled "Criar um novo exercício: Identificar imagem". The form includes several input fields and buttons. At the top, there is a "Tema:" dropdown menu with the text "Selecione um tema...". Below it is a "Nível de Dificuldade:" dropdown menu with the text "Selecione um nível...". A "Pergunta:" text input field follows. Then, there is a label "Imagem a identificar:" next to a small image icon. Below these are three text input fields for "Resposta correcta:", "Hipótese errada 1:", and "Hipótese errada 2:". A plus sign icon is located between the last two hypothesis fields. At the bottom of the form are three buttons: "Cancelar", "Apagar tudo", and "Criar".

Figure 2.1: Caregiver's module - Creating a new exercise.

On the child's module, the talking animated character called Catarina which presents the exercise and also utters the greetings and the reinforcement message whenever the child answer correctly to an exercise, was kept. Both greetings and congratulations can be set by the caregiver for each child. Another customizable option is the set of reinforcement images to be used when the child answers correctly.

The child can choose a sequence of exercises he/she wants to play (see Figure 6.2a) that was firstly created by the caregiver specifically for this child. The chosen exercise is displayed on his/her device screen. When a child is solving an exercise (see Figure 6.2b), if he/she picks the right answer on a first

<sup>2</sup> [https://www.l2f.inesc-id.pt/w/Welcome\\_to\\_the\\_Spoken\\_Language\\_Systems\\_Lab](https://www.l2f.inesc-id.pt/w/Welcome_to_the_Spoken_Language_Systems_Lab) last access 30/05/2018

try, the reinforcement image is displayed; otherwise the distractor disappears and the right answer is highlighted (prompting) to help the child choosing the right one.



Figure 2.2: VITHEA-kids - Child's module.

In spite of all promising features implemented in VITHEA-kids, in terms of architecture, some problems were identified. For example, errors were not easy to understand since search on a great dimension log file was necessary. Debugging was also a not trivial task given the large number of eXtensible Markup Language (XML) configurations files, where a certain error would be hard to detect. Another problem was the performance that was not the best since after some page requests, the system turned very slow until it crashed. Also, every change in the code required a redeploy of the project, and each redeploy takes a lot of time, what is not productive for the developer.

These problems pointed out led to a need for a software reengineering. As consequence, a new version of VITHEA-kids, with a technological reformulation, took place to improve developer's experience [20]. Also, new features were introduced.

VITHEA-kids 2.0 follows a client server model, where the child's application (Android) and the caregiver's application (Web App) work as clients. The server provides services to both of them, following a Model-view-controller (MVC) pattern.

As for the technologies used, they have undergone significant changes, as can be seen in Figure 2.3. The server side uses Play framework<sup>3</sup>, which is an open source web application framework that aims to optimize developer productivity, instead of Spring<sup>4</sup>. MySQL<sup>5</sup> database was kept to store all the information needed. Regarding front-end, Angular framework was chosen<sup>6</sup>, which makes it easier to create appealing interfaces with fewer lines of code. For the child's application, Android<sup>7</sup> is used, as in the previous version.

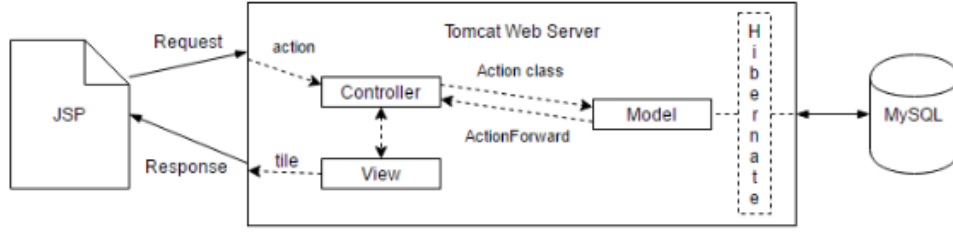
<sup>3</sup> <https://www.playframework.com/> last access 05/05/2018

<sup>4</sup> <https://spring.io/> last access 05/03/2019

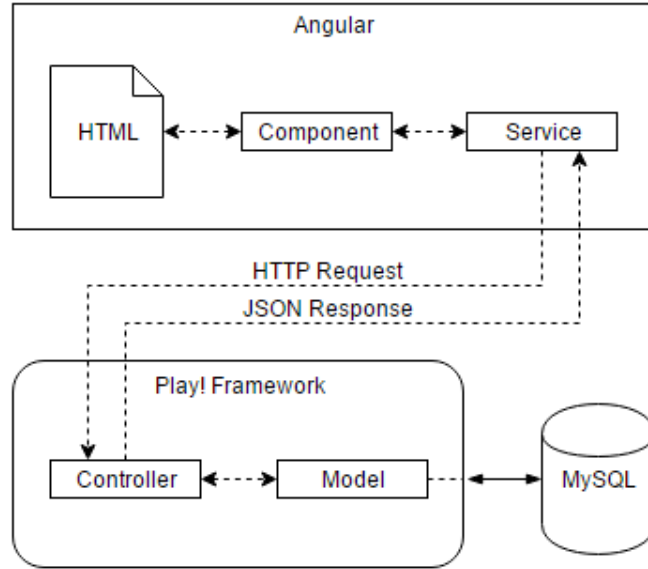
<sup>5</sup> <https://www.mysql.com> last access 03/03/2019

<sup>6</sup> <https://angular.io/docslast> access 04/05/2020

<sup>7</sup> <https://developer.android.com/ndk/> last access 04/05/2020



(a) Vithea-kids.



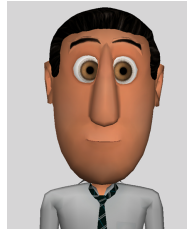
(b) Vithea-kids 2.0.

Figure 2.3: Architectural changes from VITHEA-kids to VITHEA-kids 2.0 [20].

Besides the reformulation of the technologies used, VITHEA-kids was also extended: prompting was extended, such as the possibility of changing the visual look of the set of answers, as well as prompting strategies (the prompting is always given, the prompting is given when the child selects a wrong answer or no prompting). Reinforcement strategies could also be customized (continuous reinforcement, the reinforcer is always given, first attempt reinforcement, the reinforcer is given only when the child chooses the correct answer at first attempt and no reinforcement). In the emotions's module, the caregiver may choose to enable or disable the animated character emotions when uttering its sentence. Also, it is possible to select an animated character (between Catarina, Filipe and Edgar [7]) to be presented in the child's app.



(a) Catarina



(b) Edgar



(c) Filipe

Figure 2.4: Animated characters used

After this overview, given its architecture, it is possible to conclude that VITHEA-kids 2.0 appears to be a promising system to achieve the main goal of this thesis. For this reason, our work will be focused

in extending this system to reach children with dyslexia and with SLI with studied exercises presented in the next chapter. However, VITHEA-kids 2.0 still has a few bugs that needs to be solved, as well features to be finished to achieve an usable platform, that does not affect directly the exercises, but could result in a lack of interest, by the caregiver and children, in using VITHEA-kids. Also, adding new exercises could become a big challenge by itself, given the way that the existing features are implemented, namely there are a lack of modularization, repetitive code segments, which make the code harder to understand and maintain. This way, a large part of this thesis focus on fix the bugs and on refactoring VITHEA-kids 2.0, thus contributing VITHEA-kids 3.0.

## **2.4 Discussion**

To summarize, children with dyslexia struggle with word recognition, by poor spelling and decoding abilities. Regarding SLI, children with this kind of disorder present troubles on comprehension and on production of relative clauses. This way, we will provide children with this kind of disorders a customizable type of exercise that allows them to practice skills that they struggles at, namely spelling and relative clauses comprehension. We will use VITHEA-kids 2.0 that has proved to be a promising platform to achieve this goal, especially because it is free and allow the caregivers to manage the exercises. However, the architecture of VITHEA-kids 2.0 is not prepared to be extend with new types of exercises. Therefore, the first iteration of this thesis will consist of solving the identified bugs, as well, refactoring the back-end, caregiver's module and child's module, contributing this way to the version 3.0 of VITHEA-kids.



# 3

---

## Related Work

---

As we have seen in Chapter 2, children with SLI have troubles, among other areas, in the comprehension and production of relative clauses. Regarding dyslexia, we saw that children with this learning disability struggle with reading, spelling, math, among others. Additionally, these affected areas could possibly be improved if children with these learning disabilities were given tools for its development [15], [3]. Having this in mind, several authors have searched for ways in which software solutions could be useful and also therapists have applied approaches and exercises to improve the affected abilities, as we will see in the next sections.

Hence, this chapter is focused on works and approaches used by therapists/researchers: in Section 3.1, we will present two exercises which are used to assess subject and object relatives involving children with SLI; in section 3.2 we will describe an approach commonly used to implement reading programs. The way of how Luz Rello, a Spanish researcher, creates exercises to improve spelling will be explained in detail. Some software solutions to address difficulties felt by children with dyslexia will be present. We also present a sample of paper-based exercises used in therapies. Lastly, we give a brief discussion about all approaches and exercises found to better understand which exercises are possible to be created with VITHEA-kids 2.0 and which ones would require a profound modification to the platform. With this discussion, we found a potential an exercise to be implemented in VITHEA-kids 2.0.

### 3.1 Exercises to assess relative clauses comprehension in children with SLI

As we mention at the introduction of this thesis, the main goal, regarding the SLI, consists of helping children with SLI in the comprehension of relative clauses. This way, we will present exercises used to assess children with SLI. Friedmanna et al. (2006) conducted an experiment to test the comprehension of relative clauses in Hebrew-speaking kinder-garden children where they have concluded, as already mentioned at the Chapter 2, that children better understand subject relative clauses than object relative clauses [21]. In that experiment, they used two types of exercises: sentence-picture matching and sentence-scenario matching. In both exercises, the relative clause is presented orally. For the sentence-

picture matching, two pictures are presented: one of the pictures matches the sentence, and the other picture has the two participants in the action reversed, as shown in Figure 3.1). For instance, if the sentence is “Show me the elephant that the lion is wetting.”, the child should choose to the image that matches this sentence. For the sentence-scenario matching exercise, two scenarios composed by physical objects are presented by the experimenter, one of which matches the sentence and the other involves a reversal of the two participants in the action (Figure 3.2). The children are asked to point at the scenario that matches the sentence, such as “Show me the cow that is kissing the chicken”. In these exercises, there is a special care with the choice of figures presented in the sentences. These should always be of the same gender and number, for example, an elephant and a lion. This is done to keep the noun phrases as similar as possible and to prevent an agreement cue on the verb.

An expert in the field hypothesized that a type of exercise similar to these, could be used for children to train relative clauses. In that exercise, an image would be presented to the child, with three main elements, two of which would be of the same type in order to induce some confusion for the child. That image would be preceded by a question related with the image. For example, the question “Que cavalo é que o boi mordeu” (“What horse did the ox bite?”) would be a good one (the Figure 3.3).

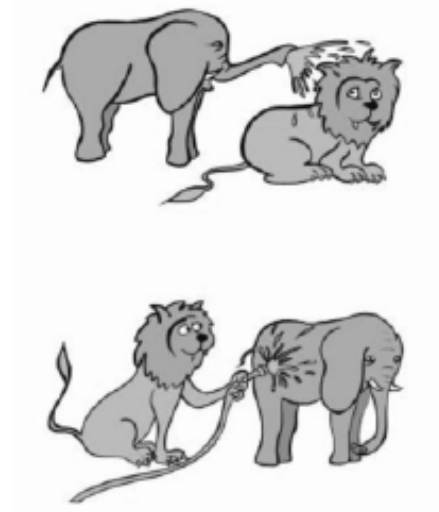


Figure 3.1: Sentence-picture exercise [9].



Figure 3.2: Sentence-scenario exercise [9].

## 3.2 Dyslexia

In this section we will describe some approaches to improve skills affected by dyslexia. We will also present some applications, which aim at improving abilities affected by this learning disability.



Figure 3.3: A possible image to use in exercises for children with SLI [19].

### 3.2.1 Orthon-Gillingham Approach

Orthon-Gillingham [16] is an instructional approach commonly used to implement reading programs in the United States for dyslexic students. A key characteristic of Orthon-Gillingham (OG) reading instruction is multisensory that involves the use of visual, auditory and kinesthetic-tactile pathways, often referred the Language Triangle [16]. Associations are consistently made between the visual (the language we see), auditory (the language we hear) and kinesthetic-tactile (the language we feel) pathways in learning to read and spell.

Additionally, OG approach is often characterized as being <sup>1</sup>:

- **Explicit:** The rules and patterns of decoding and encoding are explicitly taught. As opposed to the traditional learners that can understand these rules and patterns naturally, students with dyslexia need to be taught every rule and pattern directly (or explicitly). Students also need to be taught exceptions to the rules.
- **Systematic and structured:** Systematic instruction means that the new concepts should be taught in the exact same way every time. This way the brain is not spending more energy trying to figure out a new method, but rather expects the routine of learning and, therefore, it will only focus on the new concept to be learnt. Structured instruction means that the concepts taught follow an order that shows the relationship between what was previously learned and the new material to be taught.
- **Sequential and cumulative:** The instructions must begin with the easiest and most basic concepts and progress methodically to more difficult material. Concepts taught must be reviewed to strengthen memory.
- **Multisensory:** Multisensory involves the use of visual, auditory and kinesthetic-tactile pathways at the same time in order to help the information be stored in long term memory.
- **Individualized:** Teaching should follow an approach one on one in order to customize each lesson to the student's needs.
- **Diagnostic and prescriptive:** The lessons are diagnostic in the sense that the instructor continuously monitors the answers of the learner in order to identify and analyse both the student's problems and progress. That information is essential to plan the next lesson. Therefore, the lesson is prescriptive since it is always prepared according to the problems found in previous lessons and the progress obtained.

This approach is quite effective since students with dyslexia often reveal weaknesses in underly-

<sup>1</sup> <https://homeschoolingwithdyslexia.com/orton-gillingham-approach-teaching-reading/> last access 22/05/2018

ing language skills involving speech sound (phonological) and print (orthographic) processing, and in creating brain pathways that connect speech and print. So, the key characteristic of this approach, multisensory, is a quite effective way to build the brain pathways used for reading and spelling once connect many brain area and must transmit information with sufficient speed and accuracy.

Another problem present in Dyslexia consists in the lack of phonemic awareness<sup>2</sup>. That is individuals are aware of the sounds linked to the words. In consequence of this, learning to recognize words automatically (“by sight”) or fast enough to allow comprehension becomes a quite difficult task.

Terry Orton<sup>3</sup>, who pioneered the study of learning disabilities, suggested that teaching the “fundamentals of phonic association with letter forms, both visually presented and reproduced in writing until the correct associations were built up”, would help students of any age.

Usually, teachers who use this approach help students perceive the speech sounds in words (phonemes) by looking in the mirror when they speak or exaggerating the movements of their mouths.

Having said this, the approach OG combines multisensory techniques with teaching the phonemes (units of sound that distinguish one word from another), morphemes (smallest meaningful unit of a language, such as prefixes, suffixes, and roots) and spelling rules.

### 3.2.2 Designing word exercises to children with dyslexia

Children with dyslexia present a problem in automatic letter writing and naming, which is related to impaired inhibition and verbal fluency and may explain their spelling problems [2]. In light of this, technology can play an important role since conventional methods of teaching, such as paper-based exercises, are often not so effective [15], due to the fact that there are many limitations underlined, such as the following ones [26]:

- (a) Paper-based exercises present an unappealing format which means children may lose motivation as they are doing their exercises.
- (b) Children may face some difficulty with writing in paper.
- (c) It is complicated to adapt each exercise to the particular need that a child could have.

Luz Rello, a spanish researcher, has focused her work on finding a solution for dyslexia, in a way that could be integrated in an application. Rello and her team developed the app IDEAL ebook reader [14] in order to assist reading. They also developed Dyseggxia [27] to improve the spelling of children (these apps will be described in next section).

L. Rello et al published an article “Design of word exercises” [25], which presents a method of designing word exercises to support children with dyslexia. This method is based on patterns found in errors written by individuals with dyslexia. It comprises six stages which are: exercise type, word selection, word modification, selection of distractors, creation of difficulty levels and text layout.

The definition of **exercise type** hinges on dyslexic errors, specific difficulties and the current pedagogical exercises which leads to six types of exercises:

- **Insertion:** the screen displays a word with a missing letter and the user is asked to insert a letter from a set of possibilities displayed on the screen, e.g. \*timestre r, m, n, s, p, (correct: trimestre).
- **Omission:** the screens display a word with an extra letter and the user is asked to identify it and remove it, e.g. \*asccessible, (correct: accessible).
- **Substitution:** A word with a wrong letter is displayed and the user has to identify and substitute the wrong letter by another letter from a set of possibilities displayed on the screen, e.g. \*abter r,b,h,f, (correct: after).

<sup>2</sup> [https://en.wikipedia.org/wiki/Phonemic\\_awareness](https://en.wikipedia.org/wiki/Phonemic_awareness), last access 24/05/2020

<sup>3</sup> [https://en.wikipedia.org/wiki/Samuel\\_Orton](https://en.wikipedia.org/wiki/Samuel_Orton), last access 24/05/2019

- **Derivation:** the root of a word is displayed with a set of suffixes, where only one is correct. The user has to identify the correct suffix e.g. \*blue able, age, ish, ment, (correct: blueish).
- **Separation:** The display presents a set of words without spaces. The user must to separate the character chain into different words, e.g. \*polarbear,(correct: polar bear).

For **word selection**, only the words that appear in the Royal Spanish Academy Dictionary and in the New Oxford American Dictionary are taken into account. Also, it is used a “DysCorpus” (Spanish corpus of texts written by children with dyslexia), to select only the words whose frequency is equal or greater to a frequency threshold. Besides that, only words whose length is between 3 and 12 letters are taken into account, since children encounter a few problems with very short and very long words. All words are presented in lemmas, which mean they are presented in their canonical form. For example, the adjective “guapo”/ (“handsome”) would appear in its least-market form (singular masculine). So, the words are presented in lemmas since morphological processes as inflection and conjugation occurs with some regularity and therefore are processed differently than the rest of the lexicon acquisition. These exercises have the goal of correcting words, therefore **word modification** implies that errors were made. To determinate the errors two criteria are used, **error letter selection** which means select the letters that appears more in errors, and **error position** that checks which insertion and omission of letters happened more.

**Distractors** consists in wrong choices together with the right answer, in multiple choice item. So, groups of phonetically and orthographically similar graphemes are used when selecting distractors.

### 3.2.3 Software for children with dyslexia

#### Spelling Abilities

Dyseggxia (A Computer-Based Method to Improve the Spelling of Children with Dyslexia) is a software, where Luz Rello et al implemented the method explained above [28].

Besides its main goal, which consists of helping children to learn how to identify dyslexic errors (errors made by children with dyslexia) in a way they could develop strategies to write better, the effectiveness of the method previously mentioned was also tested. An experiment in a primary school was conducted. As a control condition, Word Search<sup>4</sup>, a word puzzle game, was used. Two groups were formed, one that played DysEggxia and another that played Word Search. Using a within-subject design, Rello et al compared the evolution of reading and writing performance, as well as the subjective perceptions of these two groups. It was found on the basis of this experiment that children improved their spelling significantly compared to playing Word Search.

#### Visual spatial and calculation abilities

Although mastering written language is one of the main symptoms of dyslexia, other related skills may be affected such as oral language, numeracy, notational and organizational skills. Games may play an important role in improving these skills [13]. It is believed that such an impairment in these skills can be overcome by learning chess since we need them to play well. With this motivation, Rello et al. have developed a platform (web based) to play chess with the goal of understanding if people with dyslexia play chess in a different way, because once the hypothesis is proved, they may conclude that chess could improve affected abilities by dyslexia, such as visual spatial and calculation [29].

The platform has lessons on how to play chess and about chess theory, exercises to practice each lesson learnt as shown in Figure 3.5, and also allows the child to play against an elementary opponent. It comprises the following exercises:

<sup>4</sup> <http://thewordsearch.com/> last acces 29/05/2018

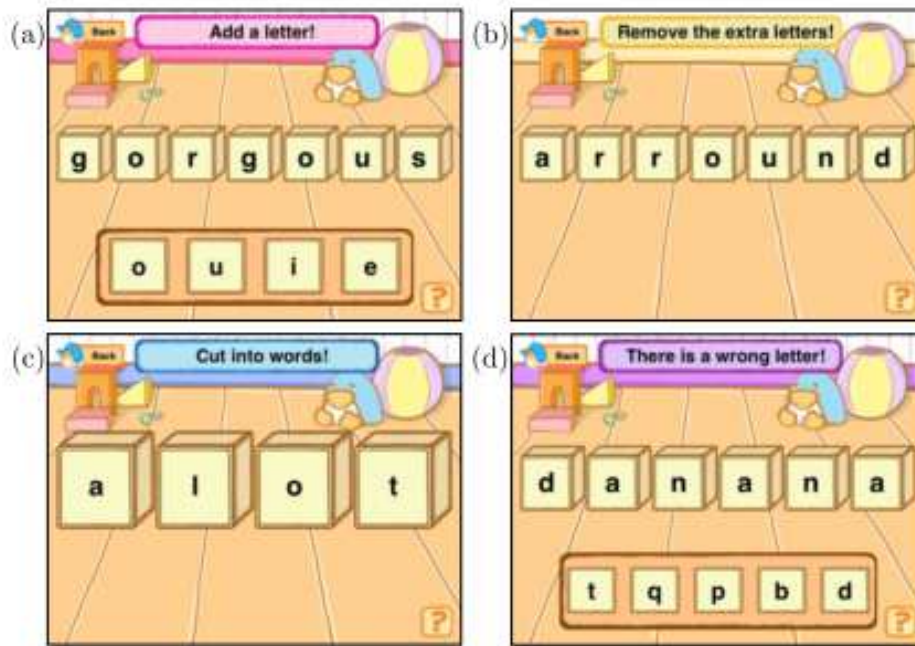


Figure 3.4: Exercises of DysEggxia - (a) add a letter, (b) remove a letter (c) cut into words and (d) change a letter [28]

- (a) Square exercise – In this exercise, the player has to click on a certain square of the chess board, designated by their coordinates.
- (b) Color exercise – the player is asked to identify the color of a square where a certain piece is located or a move takes place.
- (c) Piece exercise – the player is asked to identify if in a certain square there is or will be a piece after a move.
- (d) Moving pieces exercise - the player has to move a piece according to the chess rules.

There is also an algorithm that plays elementary chess that allows the child to play a game against the machine in order to practice the piece movements, as we can observe in Figure 3.5. This part the lesson demands more skills, such planning and predicting at least two piece movements, which are related to the executive functions, highly correlated with dyslexia.

In order to investigate if people with dyslexia play in a different way, a study was made with two groups, one with and other without dyslexia, where they completed a chess lesson on-line. To compare the evaluation of the groups, mouse tracking measures were used. The obtained results showed that people with dyslexia spend more time reading the chess theory, doing training exercises and playing chess than others which is normal considering to the fact that the theory is basically composed by text and figures and people with dyslexia read significantly slower. The results also suggest that dyslexia may have some impact on chess performance which could indicate that some skills needed to play chess are highly related with dyslexia, such as visuospatial attention disorder. However there is still not an evaluation in an educational environment, so it is not possible to conclude with accuracy that chess may help in some skills highly related with dyslexia.



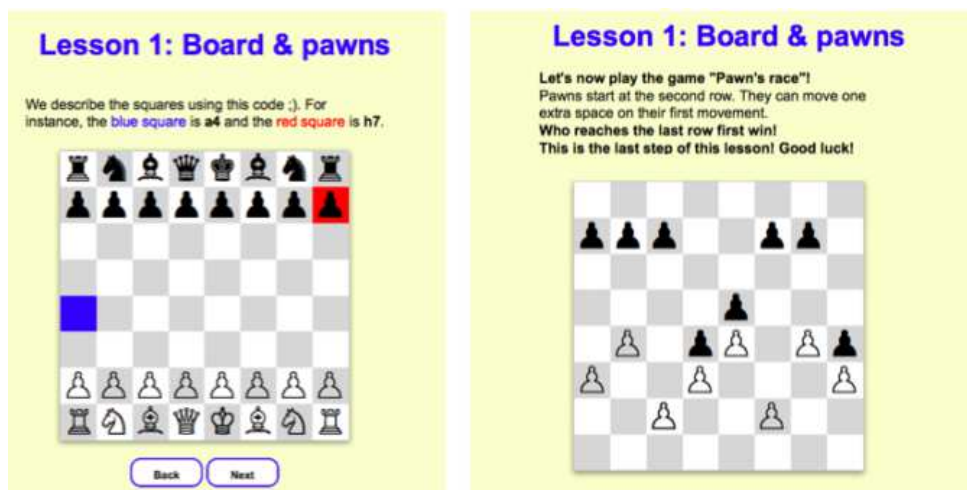


Figure 3.5: Left: lesson, right: training exercise [29].

### 3.2.4 Exercises from a book

Although there are some applications and approaches to develop domains affected by dyslexia, to the best of our knowledge there are still no free applications in Portuguese. Currently, using exercise manuals is the main method to address the problems identified in dyslexia. As such, in this subsection will be exposed some of the exercises used in therapy [30].

In the book “Dislexia – Caderno de Reeducação Pedagógica” [30], there are several exercises that follow this structure, varying only in terms of images, phrases and/or words used.

These exercises aim at training basic skills related to reading-writing and also to develop phonological and psychomotor skills (laterality, spatiotemporal orientation), among others.

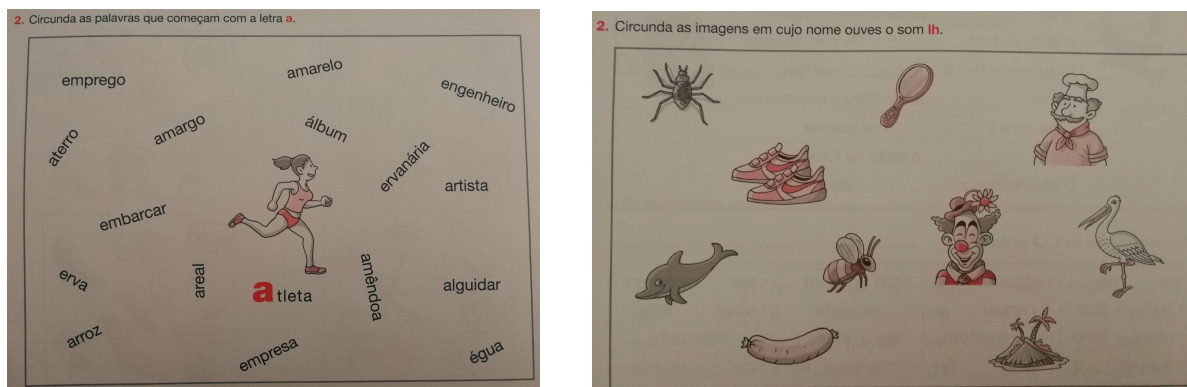


Figure 3.6: Found words starting with the letter A (left) and found images that contains the sound “LH” (right) [30].

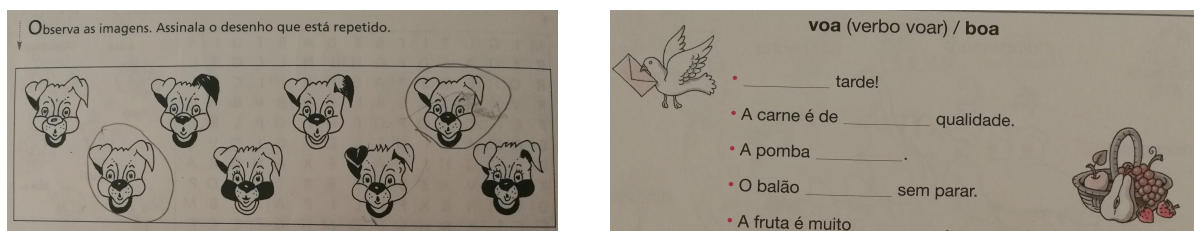


Figure 3.7: Found the repeated drawing (left) and fill the blank spaces with the correct word (right) [30].

Figure 3.6 presents two exercises: in the first, the child has to surround the words that starts with the

specified letter. In the second, in which the child has to surround the figure where the respective name has the appointed sound.

With respect to Figure 5.4, in the exercise of the left side, the child has to mark the repeated image. Regarding the exercise of the right side the child has to fill the blank space with the right words.

As we will in the next sections, the exercise that consists of filling the blank spaces is the only one that could be created using VITHEA-kids 2.0.

### 3.3 Discussion

In Section 3.1, two exercises are presented to evaluate the comprehension of relative clauses. It was hypothesized that a type of exercise similar could be used as training. This hypothesis should be verified in a future work. In that exercise, a phrase would be presented orally such as “Que cavalo é que o boi mordeu”(“What horse did the ox bite?”), as well as, an illustrative image of that sentence. To solve this type of exercise, following the example, the child would only have to point to the horse that bites the ox. Currently, there is a Portuguese research project in progress using this type of exercise to assess the comprehension relative clauses

However, as described in the Background (Chapter 2), VITHEA-kids 2.0 only allows the creation of multiple-choice exercises. Therefore, there is no support for this type of exercise, since this one requires the identification of specific areas, the correct answers.

In Section 3.2.1, it was described the OG approach. As was mentioned, this approach combines multisensory techniques with teaching the phonemes. Despite that any type exercise will not be implemented in VITHEA-kids 3.0 related to this approach, it was worth to mentioned it, given the success that it has achieved. After talking with an expert in the field, it was possible to find a few exercises, based in this approach that it will be described at future work section.

As we could see at subsection 3.2.2, Rello developed a method of designing word exercises to improve spelling. In this method, she presented four types (insertion, omission, substitution, separation). Three of them could be accomplished using VITHEA-kids 2.0. For **insertion**, it would just be necessary use as a question one word with a missing letter and the answers should be the possible letters. For **omission**, a word with an extra letter could be used as a question and a set of letter could be used as answers, containing the extra letter. And lastly, for the **separation**, it could be displayed as a question a set of words without spaces, and the answers could comprise the various ways of separation. Regarding the **Substitution** exercise, this is not possible to accomplish.

We present a xadrez game in subsection 3.2.3, also developed by Luz Rello et al.. This game was implemented taking into account the needs of children with dyslexia and it aims to improve some abilities, such as visual spacial and calculation. Although, it seems an attractive application, it would require too much graphic computation, that so far is not supported by VITHEA-kids 2.0.

In subsection 3.2.3, we present four types of exercises, taken from a book used in therapies. The one that consists of filling the blank space could be easily created using VITHEA-kids 2.0, since the sentence with the blank space would be the question and the possible words the answers. The other three exercises are not possible to create, since, like the exercises used in SLI, require the identification of specific areas inside an image as correct.

In conclusion, an exercise where it could be possible to specify areas on an image it would be useful for dyslexia, as well as for SLI. Therefore, our goal is to implement this type of exercise in VITHEA-kids 3.0.



# 4

---

## VITHEA-kids 3.0

---

In Chapter 3, we identified an exercise to be implemented in VITHEA-kids. However, there were still bugs to be solved, as well as features to finish to achieve an usable platform. Although these issues does not affect directly the implementation of exercises, they could result in a lack of interest, by the caregiver and the children, in using VITHEA-kids. Therefore, it was important to resolve the identified problems, in a first iteration of this project. Moreover, adding new exercises had become a big challenge by itself, given the way that the existing features were implemented. Generally, there was a lack of modularization, repetitive code segments, the classes had too many code lines, not being easy to implement new exercises. Additionally, there were few comments in code and also the technical documentation was practically non-existent. Consequently, the code follows a steep learning curve, and as a result, refactoring is needed. For these reasons, the present chapter describes all changes that had lead to the production of a new version, VITHEA-kids 3.0.

### 4.1 Implementation of features

In this section, what we have done regarding the implementation of required features that were not totally implemented or were not properly implemented will be described.

The solved issues mentioned below are related to the child application.

**Support for GIF animations** Whenever a child finish a set of exercises we would like to present him with a Graphics Interchange Format (GIF) as a reinforcement. However every GIF we tried to insert was always static in our application. For this reason, we integrated Glide <sup>1</sup> to make possible the proper load of this format image into VITHEA-kids 3.0. Regarding Glide, this consist of a fast and efficient open source media management and image loading framework for Android.

**User Interface problems** As mentioned before, a multiple choice exercise with textual answers may include an image as an optional stimulus. Therefore, if no stimulus image is defined for a given exercise,

---

<sup>1</sup> <https://github.com/bumptech/glide> last access 30/03/2020

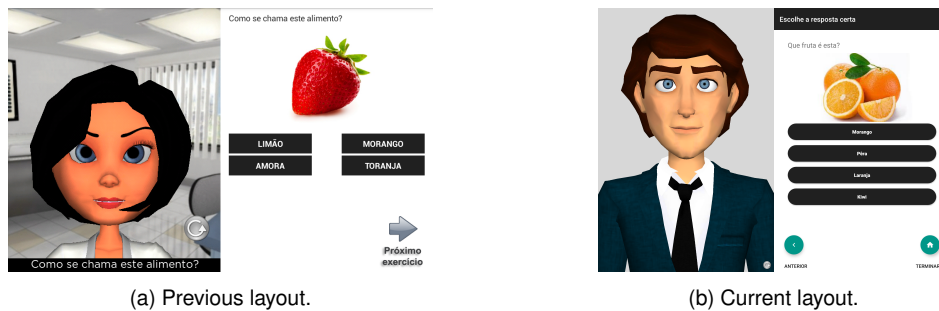


Figure 4.1: Layout for exercises of multiple choice.

the space for the image should not exist. It was necessary to add a condition to verify whenever the stimulus is empty and draw the view according with that.

The interface did not support long answers. We created new layouts that could accommodate long answers like sentences, as can be seen in Figure 4.1. 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 4.2.

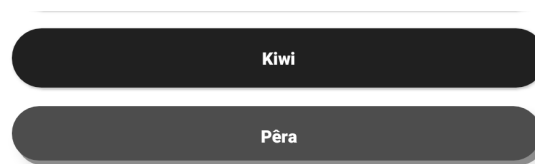


Figure 4.2: Button with the word "Pêra" pressed

**Feedback** 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 users to create exercises and/or classes when there are no exercises.

**Requested features** 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 4.3.

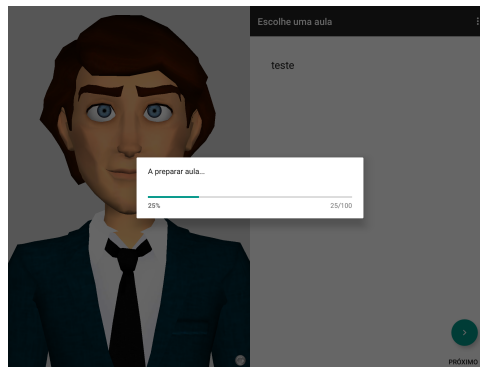


Figure 4.3: Loading images

A confirmation dialog is now shown whenever the user tries to return to the main menu like in Figure 4.4 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.

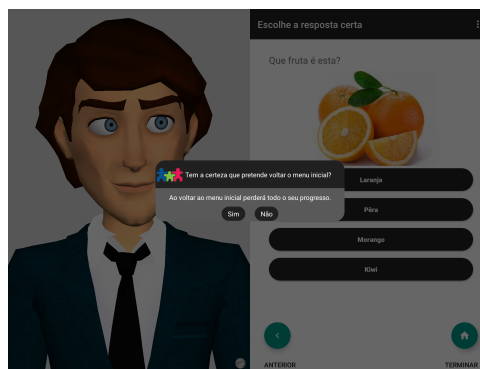


Figure 4.4: Loading images

**Features conclusion** VITHEA-kids 2.0 made available the following 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
- 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), Figure 4.5.

**Additional customization options** 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.

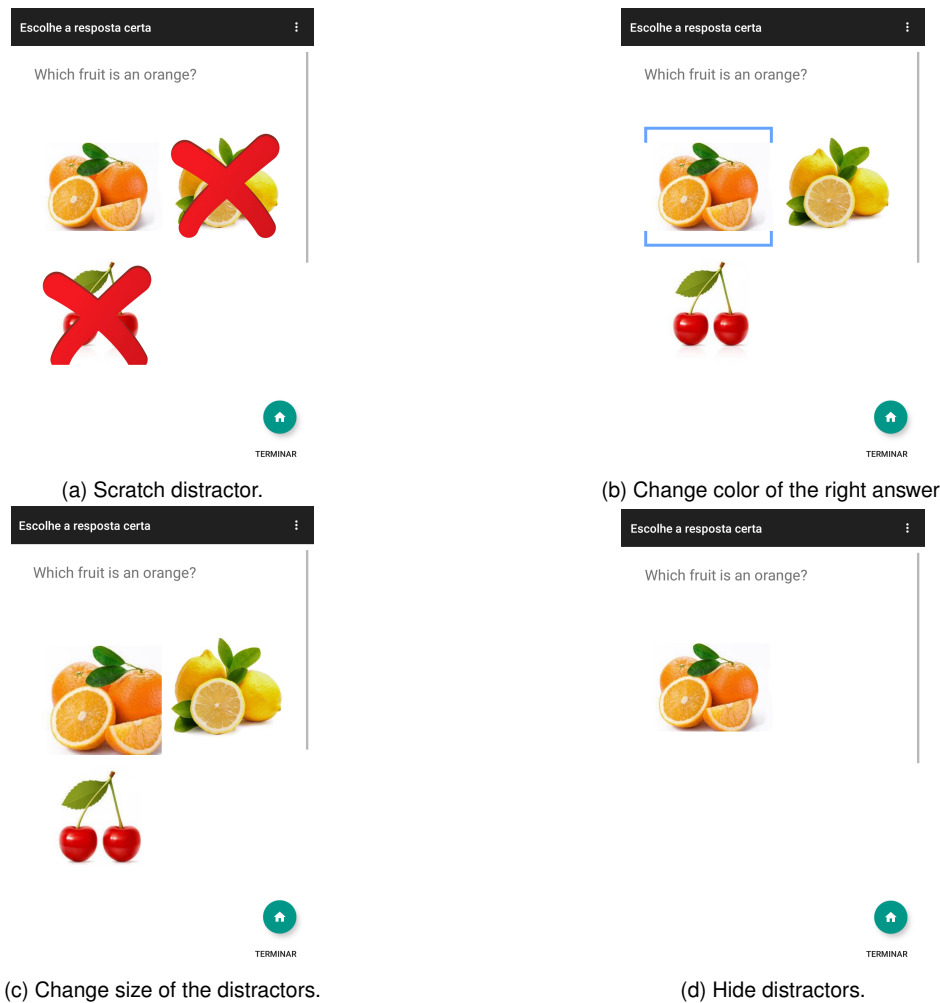


Figure 4.5: Types of promptings

However, as that customization was also no longer available we implemented again.

**Other bugs** 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.

**Layout files** The layout files in child's application were organized, removing unused files and grouping the used ones according to their function.

## 4.2 Refactoring VITHEA-kids

The present section describes 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.

### 4.2.1 Back-end

As mentioned in Chapter 2, VITHEA-kids 2.0 uses Play Framework to develop an Application Programming Interface (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 following methods:

- Register exercise – called whenever the caregiver creates a new a exercise.
- Edit exercise – called whenever the caregiver wants to edit an existent exercise.
- 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. In the Appendix C, we find the method to register the two types of exercises supported by VITHEA-kids 2.0 (multiple choices with image answers and multiple choices with textual answers). As we can verify, 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 is encapsulated this way is called **Strategy** [6].

Therefore, following the **Strategy** design, the diagram shown in Figure 4.6 was obtained.

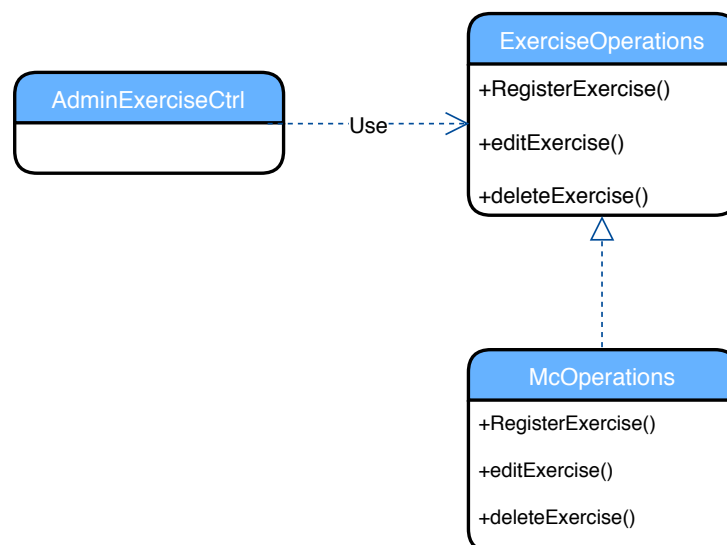


Figure 4.6: Refactoring controller using Strategy design.

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` cannot 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 [6] since it encapsulates the knowledge of which `ExerciseOperations`'s subclass to create and moves this knowledge out of the controller `AdminExerciseCtrl`.

---

```
public ExerciseOperations selectExerciseOperations(String exerciseType){  
  
    if(exerciseType.equals("text") || exerciseType.equals("image")){  
        return new McOperations();  
    }  
}
```

---

Listing 4.1: Using factory pattern to instantiate the operations for multiple choice exercises.

## 4.2.2 Database Schema

Play framework comes with Ebean Object-relational mapping (ORM) <sup>2</sup> which allows to generate the database schema based on model classes using Java Persistence API (JPA). Play also comes with Evolutions, which is a tool for tracking database changes and generate change scripts. This means that VITHEA-kids's database schema is generated through model classes, along with JPA annotations.

In order to validate whether the VITHEA-kids 2.0 database supports the implementation of new types of exercises, the corresponding relational model was created Appendix A.1.

### New proposal for database scheme

By analysing the current relational model (Appendix A.1), the following problems were detected:

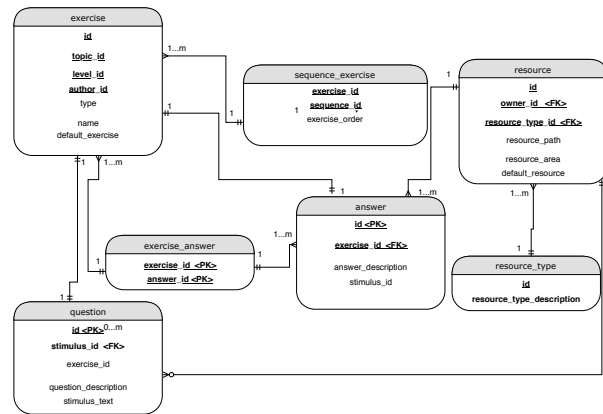
- The `Exercise` entity store all possibles fields of an exercise regardless if they are common to all types of exercises or just specific to one. This leads to the existence of null values and have therefore contributed to a non-normalized table.
- There is no flexibility for a new requirement, which consists in allowing more than one right answer. This is not possible because there is only one-to-one relationship between the `Exercise` and the `Answer` entities.
- There are some predefined levels and topics for all caregivers. However, they are replicated whenever a caregiver is created, as if they have been created by him, leading to redundant data.

In order to deal with the first problem pointed out, a solution decision would be to create specializations of the `Exercise` entity. Taking into account that VITHEA-kids allows the creation of two types of multiple choice, as already mentioned, the solution would be to create a table for each type. These tables

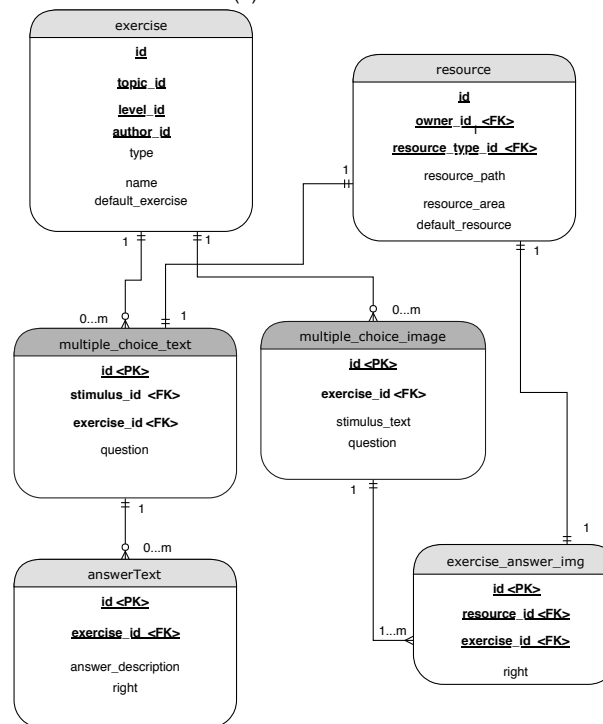
---

<sup>2</sup> <http://ebean-orm.github.io/> last access 14/06/2020

would then be specializations of the exercise entity, just like any other exercise to be implemented. With this solution, there is no need to have a relation between `Question` and `Resource` entities as shown in Figure 4.7a. That should be only between the `Resource` and `multiple_choice_text`, remembering that `multiple_choice_img` does not have image as stimulus as shown in Figure 4.7b. Also, the type of answers are different for the two kind of multiple choice. For the `multiple_choice_text` the answers are text and for the `multiple_choice_image` the answers are images. That said, instead of having an answer table that includes both possibilities, there would be two tables, one containing the textual response and another that is related to the `Resource` entity as shown in Figure 4.7b.



(a) Current model



(b) Proposed model

Figure 4.7: Differences between the current model and the proposed model concerning exercises

Moreover, since the question of a multiple choice is just a string, it is not necessary to have a table to save it, as there was before. The string could be a field in the multiple choice tables.

Regarding the second problem related to the inability of having more than one right answer, the solution would be to introduce a new Boolean field. This would take a true value if the answer was correct, and false otherwise. Also, there would be just one relation between the `exercise` and `Answer`

entity, one-to-many, as shown in Figure 4.7b.

As for the third problem raised, we need to keep to replicate the predefined levels and topics for every caregiver solution since some caregivers may decide to delete any predefined topic and/or level and others may keep them. Therefore, we need to be redundant regarding to predefined levels and topics.

The full model, comprising the solutions proposed above, can be found in Appendix B.1.

## Implementation of the database schema

After the main problems and their solutions were identified, the implementation phase was started. It started by researching the possible ways of introducing a specialization. Among the valid options for implementing a specialization such as One table implementation, Two-table implementation and Three-table implementation, it was opted for single table. Although it was not the most flexible form, given the various attributes that each exercise can have, it was the only one supported by Ebeans. Nevertheless, in the class model it was possible to create inheritance, that is, the different types of exercises inherit from the exercise class, which contain the common attributes. This possibility is important since this way we can separate the fields by exercise instead having all possible fields in only one class.

Not forgetting that the database is generated at the expense of the class model, what happens in practice is that this inheritance is mapped into a single table. This mapping generates a type attribute, which corresponds to each subclass of the exercise. Thus, when database operations (select, delete, update, insert) are performed on a type of exercise, the system always knows the type of exercise on which it is operating. For example, when a select query at multiple choice exercises is performed, respective fields are obtained. Contrary to what happened in the previous implementation, in which would obtain all the fields, even the fields that were empty. To achieve this mapping, two annotations were used which are shown in listing 4.2.

---

```
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "dtype")
```

---

Listing 4.2: Annotations that allow inheritance

In order to allow more than one right answer, we added in the `Answer` class a boolean attribute designated by `right`.

As for the problem of information redundancy, since it is outside of the scope of this project, it has not been implemented.

### 4.2.3 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.



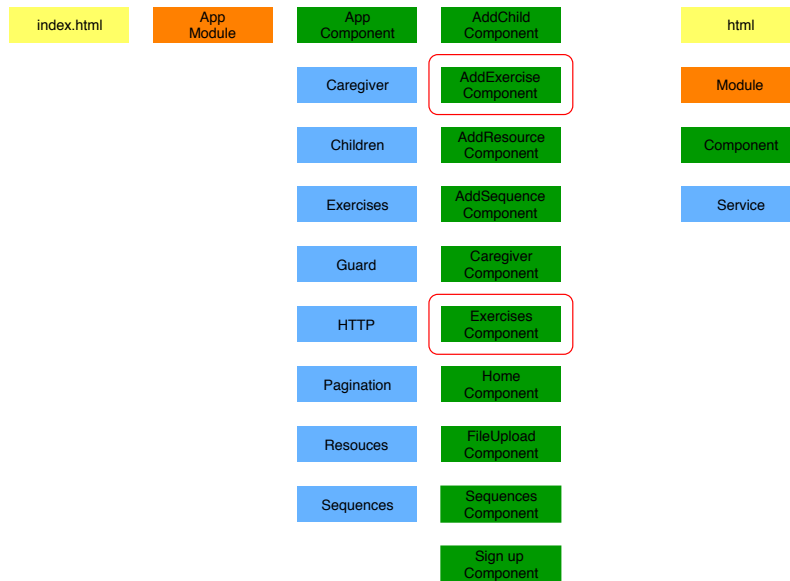


Figure 4.8: Front-end architecture

Since our main goal is to ease the implementation of new types of exercises, the components that are going to be discussed are related with the exercises (marked with a red border).

The `AddExercise` component allows the caregiver to create an exercise of the two kinds of multiple choice (one in which the stimulus is image and answers are textual and another in which the stimulus is text and the answers are images) and, also, to edit an existing one. Through code analysis, it was possible to point out the following problems:

- The way to distinguish if it will be a creation or and editing of an existent exercise it is through a variable (a holder of an id). If that variable is null it is a creation, otherwise it is and editing.
- The way to display the correct form for an exercise type, in terms of code, is through conditional statements, that verify the type of the form element to be created or edited. In this component's HTML file, for each form element, there is a conditional statement in order to know which elements should be displayed as shown in listing 4.3. This assumes that all exercises more or less follow the same structure, so there is no flexibility to accommodate new types of exercises.

```
<div class="row">
  <div class="form-group">
    <label for="stimulus"> {{ '_Stimulus_' | translate }} </label>
    <div *ngIf="newExercise.type == 'image'">
      <input class="form-control" id="stimulusText" maxlength="75" [(ngModel)]="newExercise.stimulusText" name="stimulusText" #stimulusText="ngModel">
    </div>
    <p *ngIf="newExercise.type == 'text'"> {{ '_SelectImagesStimulus_' | translate }} </p>
    <div *ngIf="newExercise.type == 'text'" style="max-height: 100px; overflow-y: auto;">
      <em *ngIf="stimulusImgs?.length === 0">{{ '_NoResources_' | translate }} </em>
      <image-picker *ngIf="!loading" [(ngModel)]="stimulusImgs" name="stimulusImgs" [selected]="newExercise.stimulus" name="newExercise.stimulusImgs" [multiSelect]="false"> </image-picker>
    </div>
  </div>
</div>
```

```

<div class="row">
  <label for="answer"> {{ '_Answer_' | translate }} <span class="text-danger"> *
  </span></label>

  <div *ngIf="newExercise.type == 'text'">
    <input type="text" class="form-control" id="rightAnswer" maxlength="75" [(
      ngModel)]="newExercise.rightAnswer" (ngModelChange)="
      validateRightAnswerText()" name="rightAnswer" #rightAnswer="ngModel">
    <div *ngIf="rightAnswerTextError" class="alert alert-danger"> {{ '
      _RequiredRightAnswer_' | translate }} </div>
  </div>
</div>

```

Listing 4.3: addExercise component

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 describe how the `AddExercise` 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 the new exercises implementation easy, 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.

#### 4.2.4 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). For example, in Listing 4.4, we found part of the `Exercise` component's HTML file, and the tags `app-add-exercise-mc-image` is a reference for the component that implements the creation form of that type of exercise. With this changes, as it can be seen in Listing 4.4, the `AddExercise` Component is much cleaner and extensible.

```

<div class="row">
  <div *ngIf="type == 'text'">
    <app-add-exercise-mc-text></app-add-exercise-mc-text>
  </div>

  <div *ngIf="type == 'image'">
    <app-add-exercise-mc-image></app-add-exercise-mc-image>
  </div>

  <div *ngIf="type == 'imageSelection'">
    <app-add-exercise-selectionImage></app-add-exercise-selectionImage>
  </div>

```

```

        <div *ngIf="type == 'speech'">
            <app-add-exercise-speech></app-add-exercise-speech>
        </div>
    </div>

```

Listing 4.4: Current addExercise 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.

## 4.2.5 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 as shown in Listing 5.3.

```

<div class="panel panel-default" *ngFor="let exercise of pagedItems |
    exerciseFilter:searchBy | exerciseTypeFilter: imageFilter:textFilter">
    <div *ngIf="!loading" class="panel-body">
        <div *ngIf="exercise.dtype.toLowerCase() == 'multiplechoice'">
            <app-show-exercise-mc-image [exercise]="exercise" x *ngIf="
                exercise.type.toLowerCase() == 'image'" ></app-show-exercise-mc-image>
            <app-show-exercise-mc-text [exercise]="exercise" *ngIf="
                exercise.type.toLowerCase() == 'text'" ></app-show-exercise-mc-text>
        </div>
        <div *ngIf="exercise.dtype.toLowerCase() == 'speechexercise'">
        </div>
    </div>
</div>

```

Listing 4.5: Current Exercise Component

## 4.3 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<sup>3</sup>. 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

<sup>3</sup> <https://developer.android.com/guide/components/activities/intro-activities.html>

divided in half, in which one of the sides is the container of the unity character and the other is the container of the following views<sup>4</sup>:

- 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 a 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 implement exercises in a flexible way. To accomplish this, we followed a 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 behavior, which enables more modular activity design<sup>5</sup>. Moreover, it is possible to combine multiple fragments in a single activity to build a multi-panel UI. Futhermore, 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 4.9:

- **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.

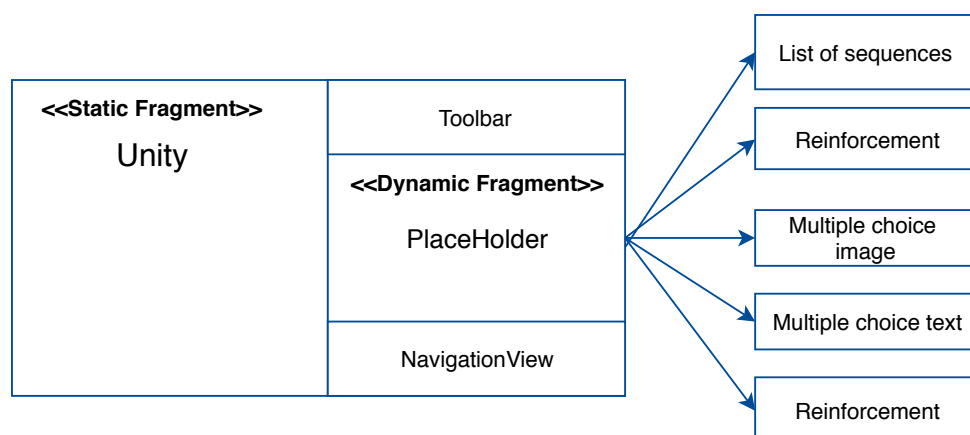


Figure 4.9: Android activity layout

Given this division, there is now a fragment for each type of exercise. The reinforcement and list of

<sup>4</sup> <https://developer.android.com/reference/android/view/View.html>.

<sup>5</sup> <https://developer.android.com/reference/android/app/Fragment.html>

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 4.2.2. 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 listing 4.6 were added to accomplish the automated mapping. These annotations allow deserialization. When deserializing, the actual code being executed will know the expected class, through the property `dtype` of JavaScript Object Notation (JSON) that comes from server.

---

```
@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="
dtype")
@JsonSubTypes({
    @JsonSubTypes.Type(value = MultipleChoice.class, name = "MultipleChoice"),
```

---

Listing 4.6: Deserialization annotations

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.

## 4.4 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. In the next chapter, we will use the implementation of this exercise to illustrate how flexible the code has become after a profound restructuring of all the components that involve the exercise. For this illustration, we will present the implementation using the two architecture, clarifying the advantages of this new architecture over the previous one.

## 4.5 Discussion

In short, the work of this thesis began with the implementation of some functionalities in the child's application that were once identified by specialists in the field. In addition, we have solved some problems that could compromise the child's motivation when using this application, namely UI problems and application bugs. After this first iteration of this work, we focused on restructuring the code that involves the exercise, on the various components existing in VITHEA-kids 2.0 (backend, caregiver's application, children's application) in order to promote a cleaner and extensible code. Regarding the backend, the strategy and factory design pattern were introduced at the level of the controllers that concern the MVC pattern. Given the limitations of play framework, and since the database was created from the class model it was not possible to restructure the database tables. This way we just left a proposal for restructuring the database for future work. Despite this difficulty, we managed to improve the ORM, creating as many specializations in the exercise class as there are types of exercises in VITHEA-kids 2.0. As

for Caregiver's application, taking into account that the framework architecture used (Angular) is based on components, the solution started to separate the details of each exercise into different components, and to create a generic component that aggregates all types of exercises. Regarding the application of the child, since there was only one activity to manage all the functionalities, it was decided to follow a fragment oriented architecture to promote a greater separation of responsibilities, instead of being all logic consolidated in just one class. Finally, we focused on one of the main objectives of this thesis: the implementation of an exercise that could tackle some gaps felt by children with dyslexia or SLI.

# 5

---

## Evaluation

---

In Chapter 4 we described the refactorization process of each component of VITHEA-kids 2.0, which had the goal of turning VITHEA-kids 2.0 in a more scalable version, in a way we could enrich it with new exercises. Therefore, in the present chapter we illustrate the differences between the previous and the new architecture describing the implementation of a new type exercise, selection image exercise, in both architectures. For this, we are going to cover 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

In Chapter 3, a new type of exercise that was not possible to achieve with the current exercises was identified, which we named as selection image. As it was mentioned in 3, 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 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 mentioned in Section 4.2.2. Following this, we will describe how it would be implemented in the previous architecture and how is it implemented in the new architecture.

### In the previous architecture

Regarding the classes model that generates the database via Ebean Object-relational mapping, as we said at section 4.2.2, it has the class `Exercise` with all fields of all exercises. This way, following this approach, since there is no inheritance of `Exercise`, the required fields for the new exercise should be added to this class<sup>5.1</sup>. In respect to the controller responsible for register, edit and delete exercise, it would be necessary to add a conditional statement and the logic associated to the new type of exercises as shown in Listing 5.1 (for reasons of simplification the code for each type of exercise is omitted). As we observe in figures, this kind of solution is not scalable for new types of exercises.

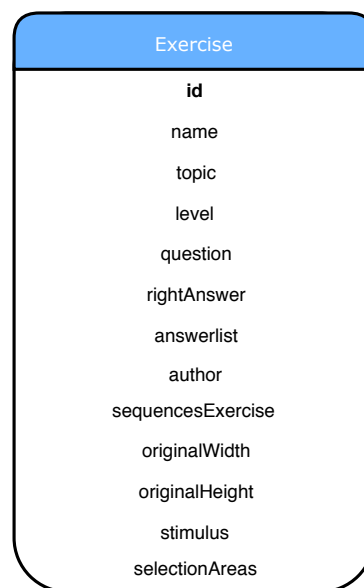


Figure 5.1: Data structure for the new exercise using the previous architecture

```
public Result registerExercise() {  
    if(registerExerciseForm.get("type").equals("text")) {  
  
    } else if(registerExerciseForm.get("type").equals("image")) {  
  
    } else if(registerExerciseForm.get("type").equals("selectionImage")) {  
  
    }  
}
```

Listing 5.1: Method to register an exercise.

### In the new architecture

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) as it is shown at Listing 5.2. 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.

---

```
public ExerciseOperations selectExerciseOperations(String exerciseType){

    if(exerciseType.equals("text") || exerciseType.equals("image")){
        return new McOperations();
    }

    if(exerciseType.equals("selectionImage")){
        return new SelectionImageOperations();
    }
}
```

---

Listing 5.2: Factory to generate the object of concrete class based on the exercise type name.

## 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. Following this, we are going to describe how these screens would be implemented in the previous architecture and how was implemented after the refactorization process.

### Previous architecture

In respect to creating and editing an exercise, as mentioned at Section 4.2.3, all logic concerning with the user interface and the respective behaviour are implemented through a single angular component. The way to differentiate whether it is a creation or an edition of an exercise is through the arguments, passed in the URL. In the case the argument is an id, the component loads the chosen exercise by the user that was once created by him, otherwise the component shows only the fields needed to be filled to create the type of exercise chosen. Furthermore, the two types of multiple choices are also implemented through a single component, `Exercise` component. Regarding the HTML file present in the `Exercise` component, to allow the creation and edition of a new type exercise, it would be necessary intercalate the code already existent with the one related with the new exercise, such as the snippet in the Listing 5.3. This means, as the web page structure follows the order that the code was written, it is necessary to put a conditional statement for every possibility that could be displayed first in the page, accordingly with the type of exercise.

Also, it would be necessary to add the typescript code that handles the behaviour related to the new exercise. Yet again, that is not extensible and not flexible, since there is no independence between exercises.

---

```
<div class="row">
  <br>
```

```

<label for="answers"> {{ '_Distractors_' | translate }}</label>
<div class="form-group" *ngIf="newExercise.type === 'text' ">
  <p>{{ '_NoDistractors_' | translate }}</p>
  <input type="text" class="form-control" id="distractor1" maxlength="75" [(ngModel)]="newExercise.distractor1" name="distractor1" #distractor1="ngModel">
  <input type="text" class="form-control" id="distractor2" maxlength="75" [(ngModel)]="newExercise.distractor2" name="distractor2" #distractor2="ngModel">
  <input type="text" class="form-control" id="distractor3" maxlength="75" [(ngModel)]="newExercise.distractor3" name="distractor3" #distractor3="ngModel">
</div>
</div>

<div class="row" *ngIf="newExercise.type === 'selectionImage' ">
  <br>
  <p> {{ '_chooseImage_' | translate }}</p>
  <em *ngIf="images?.length === 0">{{ '_NoResources_' | translate }}</em>
  <div style="max-height: 100px; overflow-y: auto;">
    <image-picker *ngIf="loading" id="images" [(ngModel)]="images" [multiSelect]="false" [selected]="images" (onSelected)="this.getImage($event)" name="images"> </image-picker>
  </div>
  <div *ngIf="this.imageError" class="alert alert-danger">{{ '_RequiredImage_' | translate }}</div>
  <div *ngIf="this.selectionError" class="alert alert-danger">{{ '_RequiredSelection_' | translate }}</div>
</div>

```

Listing 5.3: Current Exercise Component.

As for the exercise preview, it is necessary to add if statements in the HTML file as well as in the typescript file, to implement the interface of the new exercise and to load exercise's data such the question, the stimulus, the region and so on. Once again there is no independence between the exercises.

## New architecture

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 him. 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 a conditional statement that verifies if the exercise type is a selection image and inside that statement we reference a component `add-exercise-selectionImage` as shown in Listing 5.4. In the `edit-exercise` we do exactly the same we did in `add-exercise`.

```

<div class="row">
  <div *ngIf="type == 'text'">

```

```

        <app-add-exercise-mc-text></app-add-exercise-mc-text>
    </div>

    <div *ngIf="type == 'image'">
        <app-add-exercise-mc-image></app-add-exercise-mc-image>
    </div>

    <div *ngIf="type == 'imageSelection'">
        <app-add-exercise-selectionImage></app-add-exercise-selectionImage>
    </div>

    <div *ngIf="type == 'speech'">
        <app-add-exercise-speech></app-add-exercise-speech>
    </div>
</div>

```

---

Listing 5.4: AddExercise component's HTML with the tag for selectionImage 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.

#### Previous architecture

In respect to retrieving new exercise's information from server, the `Exercise` class in android application should reflect the `Exercise` class, located on the back-end, in order to assure the mapping between the both classes. This means that we need to add the image, the question, the original width, the original height and the coordinates of the selection as presented in the Listing 5.5.

---

```

@JsonIgnoreProperties(ignoreUnknown = true)
public class Exercise {

    @JsonProperty private Long exerciseId;
    @JsonProperty private Topic topic;
    @JsonProperty private Level level;
    @JsonProperty private Question question;
    @JsonProperty private Answer rightAnswer;
    @JsonProperty private List<Answer> answers;
    @JsonProperty private String type;
    @JsonProperty private Long originalWidth;
    @JsonProperty private Long originalHeight;
    @JsonProperty private CoordinatesSelection coordinates;
}

```

---

Listing 5.5: Exercise class of the child's application with the new fields

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 4.4). 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.

## New architecture

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 5.6, in order to make possible the mapping between the JSON that comes from the back-end and the respective class.

```
@JsonTypeInfo(use=JsonTypeInfo.Id.NAME, include=JsonTypeInfo.As.PROPERTY, property="
dtype")
@JsonSubTypes({
    @JsonSubTypes.Type(value = MultipleChoice.class, name = "MultipleChoice"),
    @JsonSubTypes.Type(value = SelectionImageExercise.class, name =
"SelectionImageExercise") })
```

Listing 5.6: Class `Exercise` changed to allow mapping.

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.

The figure consists of two side-by-side screenshots from a mobile application. The left screenshot, titled 'Criar Exercício', shows a form for creating a new exercise. It includes a 'Tipo de exercício' section with radio buttons for 'Resposta em fala', 'Resposta em imagem', 'Resposta em texto', and 'Seleção em imagens'. Below this is a 'Tema' section with a dropdown menu set to 'Animais'. The 'Nivel de dificuldade' section has a dropdown menu set to 'Fácil'. The 'Pergunta' section contains the text 'What horse did the ex bite'. The 'Estímulo' section shows three small image thumbnails and a larger image of three horses. At the bottom, there are buttons for 'Apagar seleção', 'Criar Exercício', and 'Cancelar'. The right screenshot shows the exercise interface for a child. It features a large character of a man in a suit, a question 'What horse did the ex bite', and an image of three horses. At the bottom, there are buttons for 'ANTERIOR' and 'TERMINAR'.

Figure 5.2: 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 sections 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 arises 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 é 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 5.2.

### 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 replicated 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 5.3 .



Figure 5.3: Form filled with information of an exercise for child with dyslexia image(left) exercise to be solved by a child (right).

## 5.3 Word Naming exercise

Beside the exercise we had implemented, another researcher from Intelligent Networked Robot Systems for Symbiotic (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.

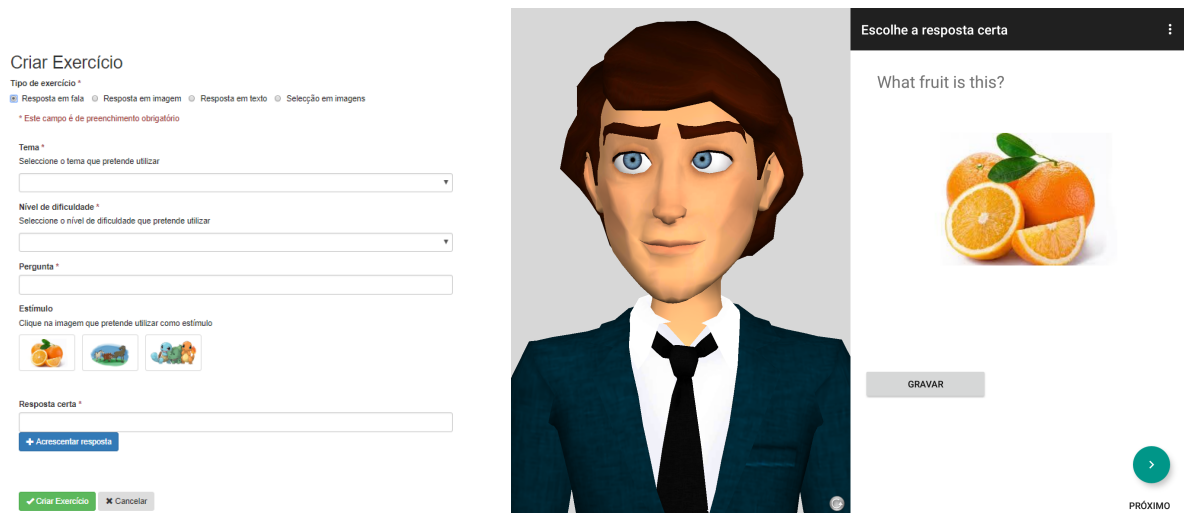


Figure 5.4: Form to create a word naming exercise (left) exercise to be solved by a child (right).

## 5.4 Discussion

In this chapter, one of the goals consists of demonstrating the improvements made in all components that make up the Vithea kids (back-end, caregiver's application and child's application). Throughout what has been described in this chapter, a certain pattern is observed in all components of VITHEA-kids 2.0, namely the lack of independence between exercises. This lack of independence leads to the existence of classes with many lines of code, and a greater probability of generating application bugs. Thus, the strategy was to create this independence, in all components taking into account the languages and frameworks used. This way, the implementation of the new selection image exercise became an easier task and thus we managed to achieve one of the goals of this thesis: "Refactor VITHEA-kids 2.0 to turn this application in a more extensible one, and thus make the implementation of new exercises, with focus on other learning disabilities, a less complex task". In addition, in the light of some examples of exercises shown in the present chapter, we managed to prove the versatility of the new type of exercise implemented "Selection Image", since it allows caregivers to create exercises for children with dyslexia or SLI. Thus we demonstrate that the goal "Implement a solution that can be applied to different exercises used in therapies that could be as useful for dyslexia as for SLI" has been achieved.

# 6

---

## Conclusions and Future Work

---

### 6.1 Conclusions

Worldwide, there are children with some learning disability, such as dyslexia and SLI. It is not always possible to provide these children with therapies, many times due 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 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 the difficulties felt by 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.

### 6.2 Future work

After searching for therapies and exercises used by specialists to fight difficulties felt by children with dyslexia, we found out some useful ideas, described with more detail in Section 3.3, that could

be implemented in VITHEA-kids. These ideas are based on Orthon-Gillingham approach 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, described in subsection 3.2.1, 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 6.1. 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 6.1: Tutor teaching SA syllable

Also, another proposed exercise, by the therapist, following the OG, with special emphasis on multi-sensory 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

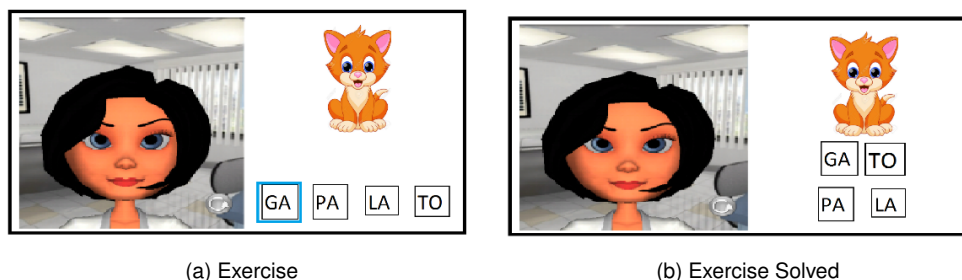


Figure 6.2: Syllable Exercise



---

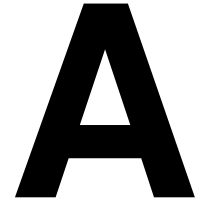
## Bibliography

---

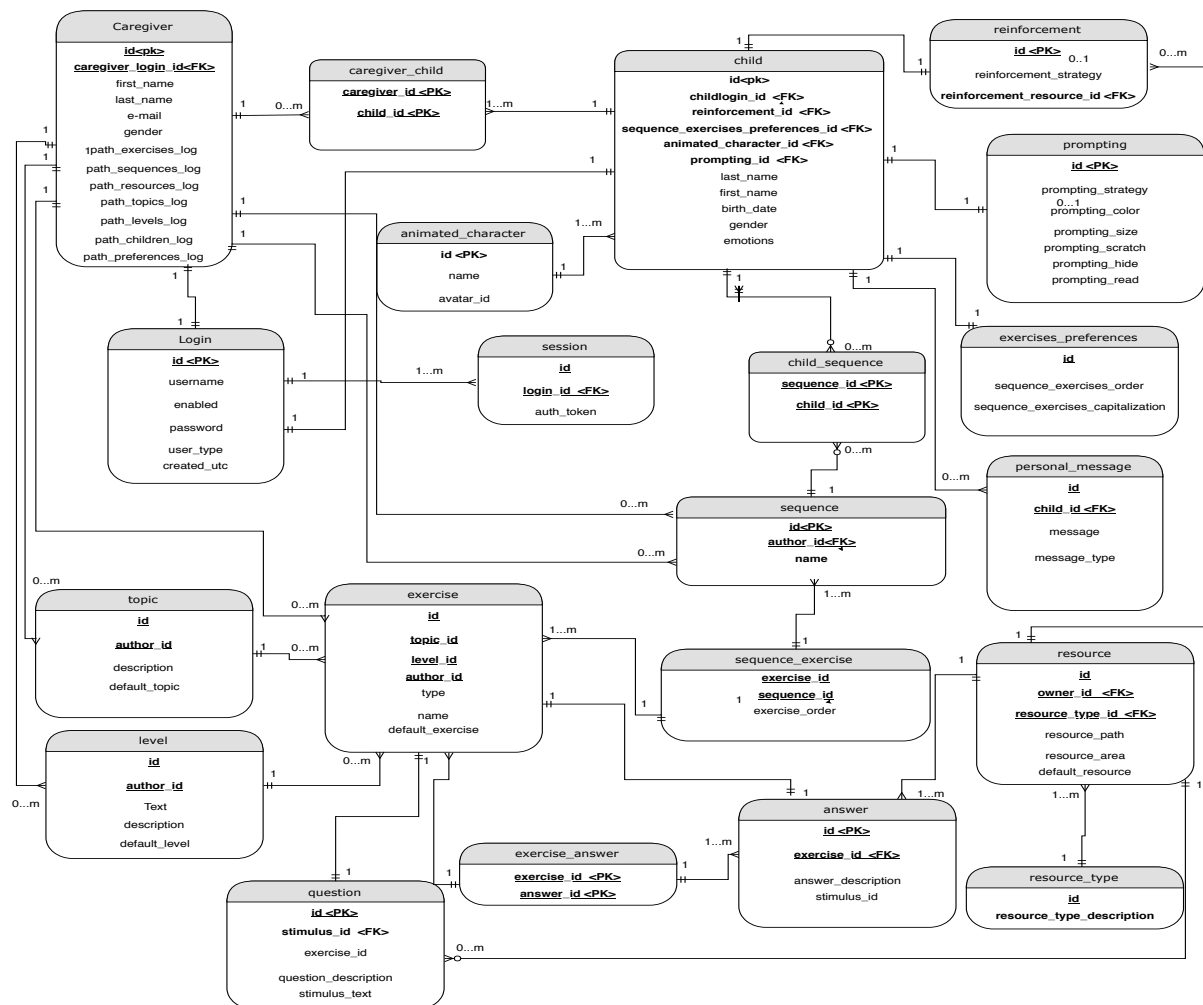
- [1] Alan Beaton. *Dyslexia, reading and the brain: A sourcebook of psychological and biological research*. Psychology press, 2004.
- [2] Virginia W. Berninger, Kathleen H. Nielsen, Robert D. Abbott, Ellen Wijsman, and Wendy Raskind. Writing problems in developmental dyslexia: Under-recognized and under-treated. *Journal of School Psychology*, 46(1):1–21, 2008.
- [3] Dorothy V M Bishop. What Causes Specific Language Impairment in Children? *Current directions in psychological science : a journal of the American Psychological Society*, 15(5):217–221, 2006.
- [4] Dorothy V M Bishop and Leonard Laurence B. *Speech and Language Impairments In Children*. Psychology Press, Purdue University, Indiana, USA, 2014.
- [5] Alexis Bosseler and Dominic W Massaro. Development and Evaluation of a Computer-Animated Tutor for Vocabulary and Language Learning in Children with Autism. *Journal of Autism and Developmental Disorders*, 33(6):653–672, 2003.
- [6] Gama. Erich, Helm Richard, Johnson Ralph, and Vlissides John. *Design Patterns: Elements of Reusable Object-Oriented Softwares*. Addison-Wesley Professional, 1994.
- [7] Pedro Fialho and Luísa Coheur. ChatWoz: Chatting Through a Wizard of Oz. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, ASSETS '15*, pages 423–424, New York, NY, USA, 2015. ACM.
- [8] Cláudia Patricia Balixa Filipe, Maria Luísa Torres Ribeiro Marques da Silva Coheur, and José Alberto Rodrigues Pereira Sardinha. An application to help children with communication disorders. Master's thesis, Instituto Superior Técnico, 2017.
- [9] Naama Friedmann, Adriana Belletti, and Luigi Rizzi. Relativized relatives : Types of intervention in the acquisition of A-bar dependencies. (2), 1998.
- [10] NAAMA FRIEDMANN and RAMA NOVOGRODSKY. The acquisition of relative clause comprehension in Hebrew: a study of SLI and normal development. *Journal of Child Language*, 31(3):661–681, 2004.
- [11] Javier Gay. The evolution of research on dyslexia History of reading disability. 32(1):3–30, 2001.
- [12] Sheryl M Handler. Dyslexia: What you need to know. *Contemporary Pediatrics*, 33(8):18, 2016.
- [13] Christopher B. Hayes. *Dyslexia in Children: New research*. Nova Science Publishers, 2006.
- [14] Gaurang Kanvinde, Luz Rello, and Ricardo Baeza-Yates. IDEAL: a Dyslexic-Friendly eBook Reader. *14th international ACM SIGACCESS conference*, pages 205–206, 2012.

- [15] Jignesh Khakhar and Sriganesh Madhvanath. JollyMate : Assistive Technology for Young Children with Dyslexia. pages 576–580, 2010.
- [16] D Ritchey Kristen and Jennifer L Goeke. Orton-Gillingham and Orton-Gillingham-Based Reading Instruction: A Review of the Literature. *Journal of Special Education*, 40(3):171–183, 2006.
- [17] Hagar Levy and Naama Friedmann. Treatment of syntactic movement in syntactic sli: A case study. *First language*, 29(1):15–49, 2009.
- [18] G Reid Lyon, Sally E Shaywitz, and Bennett A Shaywitz. A definition of dyslexia. *Annals of dyslexia*, 53(1):1–14, 2003.
- [19] Alexandrina Martins. *Complexidade sintática em PEL e PEA*. PhD thesis, in prep.
- [20] Vânia Mendonça, Luísa Coheur, and Alberto Farinha. Extending VITHEA in order to improve children's linguistic skills. Master's thesis, Instituto Superior Técnico, 2015.
- [21] Rama Novogrodsky and Naama Friedmann. The production of relative clauses in syntactic SLI: A window to the nature of the impairment. *Advances in Speech Language Pathology*, 8(4):364–375, 2006.
- [22] Rodrigues Paula and Braz Maria. Intervenção Educativa em alunos com Dislexia na Aprendizagem das Ciências naturais. Master's thesis, 2012.
- [23] Sérgio Paulo, Luís Caldas De Oliveira, Carlos Mendes, Luís Figueira, Renato Cassaca, Céu Viana, and Helena Moniz. DIXI – A Generic Text-to-Speech System for European Portuguese. In *PROPOR 2008*, pages 91–100, 2008.
- [24] Anna Maria Pompili. New features for on-line aphasia therapy. Master's thesis, 2013.
- [25] Luz Rello. Design of word exercises for children with dyslexia. *Procedia Computer Science*, 27(Dsai 2013):74–83, 2013.
- [26] Luz Rello, Clara Bayarri, and Azuki Gòrriz. Dyslexia exercises on my tablet are more fun. *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility - W4A '13*, page 1, 2013.
- [27] Luz Rello, Yolanda Ota, and Martin Pielot. A Method to Improve the Spelling of Children with Dyslexia. *ASSETS 2014*, pages 6–13, 2014.
- [28] Luz Rello and Martin Pielot. A Computer-Based Method to Improve the Spelling of Children with Dyslexia. pages 153–160, 2014.
- [29] Luz Rello, Sergi Subirats, and Jeffrey P Bigham. An Online Chess Game Designed for People with Dyslexia. *13th Web for All Conference*, pages 1–8, 2016.
- [30] Helena Serra and Teresa Oliveira Alves. *DISLEXIA Caderno de Reeducação Pedagógica*. 2015.
- [31] Ana Paula Vale, Ana Sucena, and Fernanda Viana. Prevalência da Dislexia entre Crianças do 1 .  
º Ciclo do Ensino Básico falantes do Português Europeu. pages 45–56, 2011.





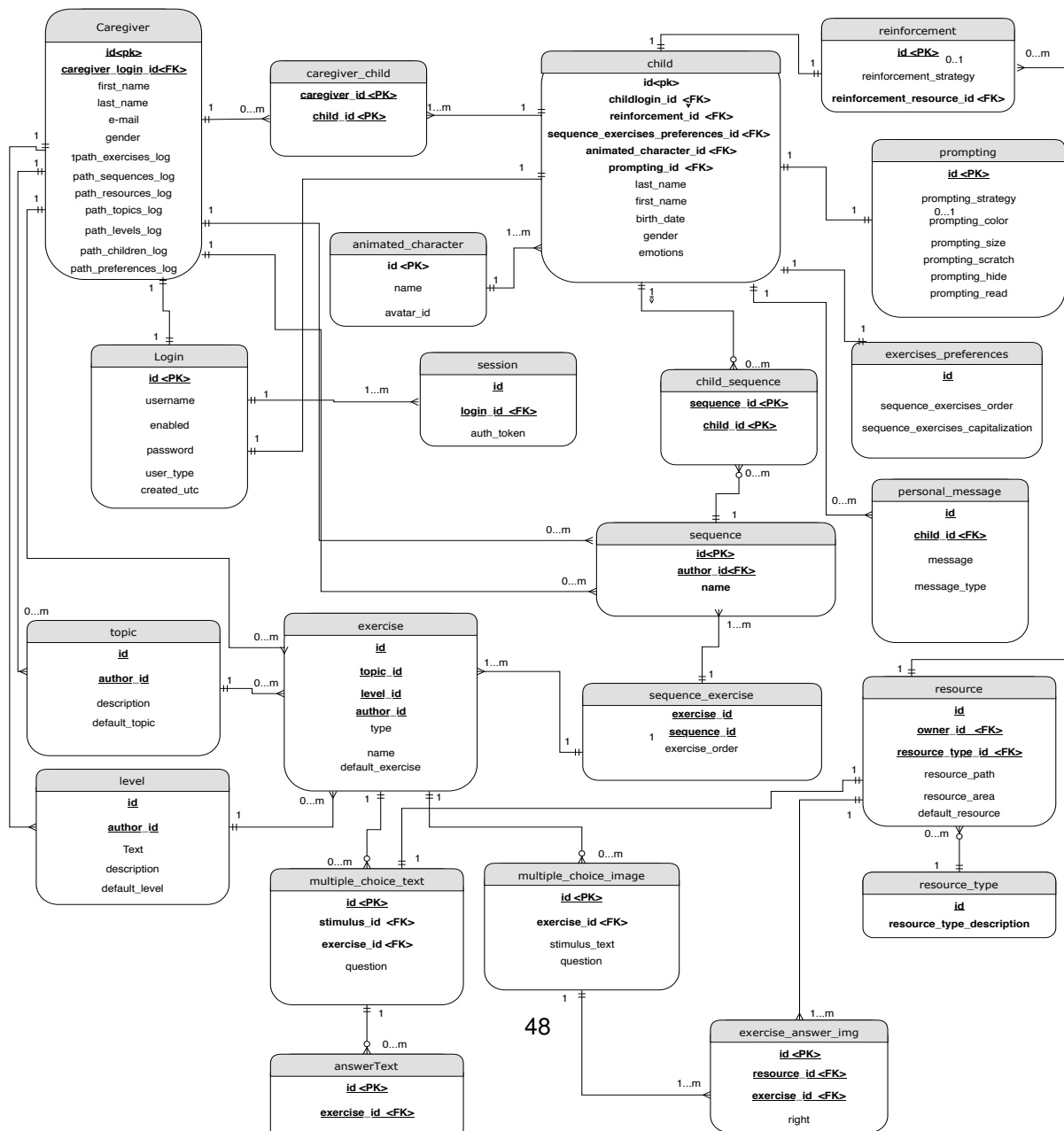
## Current RM model

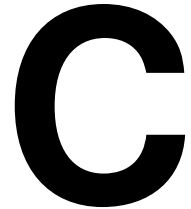




# B

## Proposal RM model





---

## Register exercise controller

---

```
@Inject
FormFactory formFactory;
public Result registerExercise() {
    DynamicForm registerExerciseForm = formFactory.form().bindFromRequest();

    Timestamp timestamp = new Timestamp(System.currentTimeMillis());

    int answers = 0;
    boolean stimulus = false;

    if (registerExerciseForm.hasErrors()) {
        return badRequest(registerExerciseForm.errorsAsJson());
    }

    Caregiver loggedCaregiver = Caregiver.findByUsername(SecurityController.getUser().
        getUsername());
    if (loggedCaregiver == null) {
        return badRequest(buildJsonResponse("error", "Caregiver does not exist.));
    }

    Exercise exercise = null;

    int topic;
    try {
        topic = parseInt(registerExerciseForm.get("topic"));
    } catch (NumberFormatException e) {
        topic = -1;
    }

    int level;
    try {
        level = parseInt(registerExerciseForm.get("level"));
    } catch (NumberFormatException e) {
        level = -1;
    }

    int stimulusId;
    try {
        stimulusId = parseInt(registerExerciseForm.get("stimulus"));
        stimulus = true;
    } catch (NumberFormatException e) {
        stimulusId = -1;
    }
}
```

```

        stimulus = false;
    }

String question = registerExerciseForm.get("question");

if(registerExerciseForm.get("type").equals("text")) {

    String sresourcesid = "";

    String answer = registerExerciseForm.get("rightAnswer");
    List<String> distractors = new ArrayList();
    answers++;

    registerExerciseForm.data().keySet().stream().filter((key) -> (key.startsWith("
        answers"))).forEachOrdered((key) -> {
        distractors.add(registerExerciseForm.data().get(key));
    });
    answers += distractors.size();

    exercise = new Exercise(loggedCaregiver, topic, level, question, stimulusId,
        answer, distractors, false);
    exercise.save();

    String content = stimulusId + "," + loggedCaregiver.getCaregiverId() + "," +
        exercise.getExerciseId() + "," +
        timestamp.toLocalDateTime() + "," + "Stimuli" + "," + "addToExercise" + "," + "
        false" + "\n";
    String pathResources = loggedCaregiver.getPathResourcesLog();
    adminLogs.writeToFile(pathResources, content);
} else if(registerExerciseForm.get("type").equals("image")) {
    int answerResourceId;
    String sresourcesid = "";

    try {
        answerResourceId = parseInt(registerExerciseForm.get("rightAnswerImg"));
        sresourcesid += answerResourceId + " ";
        answers++;
    } catch (NumberFormatException e) {
        answerResourceId = -1;
    }

    String stimulusText = registerExerciseForm.get("stimulusText");
    if(stimulusText != null) stimulus = true;
    else stimulus = false;

    List<Long> distractorsResourcesIds = new ArrayList<>();
    Map<String, String> data = registerExerciseForm.data();
    int numberDistractors = data.size();
    for(int i = 0; i < numberDistractors; i++){
        String key = "answersImg[" + i + "]";
        if(data.containsKey(key)){
            int answerId;
            try {
                answerId = parseInt(data.get(key));
                sresourcesid += answerId + " ";
            } catch (NumberFormatException e) {
                answerId = -1;
            }
            distractorsResourcesIds.add((long) answerId);
        }
    }
    answers += distractorsResourcesIds.size();

    exercise = new Exercise(loggedCaregiver, topic, level, question, stimulusText,
        answerResourceId, distractorsResourcesIds, false);
    exercise.save();

    String content = answerResourceId + "," + loggedCaregiver.getCaregiverId() + ",
        " + exercise.getExerciseId() + "," +
        timestamp.toLocalDateTime() + "," + "Answers" + "," + "addToExercise" + "," +
        " + "," + "false" + "\n";
    String pathResources = loggedCaregiver.getPathResourcesLog();

```



```

        adminLogs.writeToFile(pathResources, content);

        Object[] toArray = distractorsResourcesIds.toArray();
        for(int i = 0; i < toArray.length; i++){
            content = toArray[i] + "," + loggedCaregiver.getCaregiverId() + "," +
                exercise.getExerciseId() + "," +
                timestamp.toLocalDateTime() + "," + "Answers" + "," + "addToExercise" + "," +
                + "," + "false" + "\n";
            pathResources = loggedCaregiver.getPathResourcesLog();
            adminLogs.writeToFile(pathResources, content);
        }
    }

    exercise.save();

    String content = exercise.getExerciseId() + "," + loggedCaregiver.getCaregiverId() +
        "," + timestamp.toLocalDateTime() + "," +
        registerExerciseForm.get("type") + "," + "create" + "," + answers + "," +
        stimulus + "," + "false" + "\n";
    String pathExercise = loggedCaregiver.getPathExercisesLog();
    adminLogs.writeToFile(pathExercise, content);

    String content2 = level + "," + loggedCaregiver.getCaregiverId() + "," + exercise.
        getExerciseId() + "," +
        timestamp.toLocalDateTime() + "," + "addToExercise" + "," + "false" + "\n"
        ;
    String pathLevel = loggedCaregiver.getPathLevelsLog();
    adminLogs.writeToFile(pathLevel, content2);

    String content3 = topic + "," + loggedCaregiver.getCaregiverId() + "," + exercise.
        getExerciseId() + "," +
        timestamp.toLocalDateTime() + "," + "addToExercise" + "," + "false" + "\n";
    String pathTopics = loggedCaregiver.getPathTopicsLog();
    adminLogs.writeToFile(pathTopics, content3);

    return ok(Json.toJson(exercise));
}

```

---

Listing C.1: Register exercise controller