

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**



**Histological challenge**  
**A digital book that is also a game**

**Paulo Sérgio Silva Babo**

Dissertation carried out within the scope of :  
Master's in informatics and Computing Engineering

Supervisor: António Fernando Vasconcelos Cunha Castro Coelho  
Co-supervisor: Paula Cristina Paulo Videira da Silva

January 2020



# Resumo

Histologia é o estudo da estrutura microscópica do material biológico e as maneiras pelas quais os componentes individuais estão relacionados estrutural e funcionalmente. O ensino de histologia envolve uma abordagem prática, e o acesso ao material histológico pode às vezes ser problemático, o que causa desmotivação e perda de interesse por parte dos alunos.

Neste documento, o problema é analisado e é proposta uma solução para aumentar o interesse e a motivação dos alunos em relação à disciplina. O resultado é uma plataforma on-line, acessível em qualquer lugar, que permite que os alunos se envolvam em desafios histológicos, apoiados por uma narrativa convincente e um atlas com imagens histológicas. Além disso, o conteúdo da plataforma é proceduralmente gerado por ficheiros, o que significa que, ao editar esses ficheiros, a plataforma pode oferecer desafios e conteúdo de contextos educacionais diferentes da histologia.



# Abstract

A histologia (“ciência que estuda a microanatomia de células, tecidos e órgãos como vista através de um microscópio” [18]) hoje em dia é ensinada principalmente em sessões de microscopia de laboratório. Isso representa um problema para os alunos, pois eles normalmente não possuem um microscópio e o acesso a material de estudo de boa qualidade é restrito e escasso, pois os alunos têm de procurar atlas impressos ou conteúdo on-line de outras universidades. A disciplina de "Histologia Funcional" é oferecida pela Universidade Portuguesa do Porto, para os alunos do 1º ano do Curso de Bioquímica. Durante o curso, os alunos recebem semanalmente 5 horas de contato, onde são ensinados por uma mistura de exposições teóricas, observação de preparações microscópicas e jogos educativos. Nos últimos 2 anos, há uma discrepância entre a nota média de 15,4 valores e a média de aprovações de 61%. Pensa-se que esta dificuldade em obter aprovação esteja relacionada com a falta de disponibilidade de material específico de apoio e, portanto, uma maior adversidade na motivação e autonomia do aluno ao tentar estudar e aprender fora do ambiente de sala de aula.

Neste projeto, apresentamos uma ferramenta on-line que pode ser considerada uma mistura entre um livro e um jogo digital e visa ajudar os alunos a alcançar as competências exigidas pela disciplina. Essa ferramenta pode ser usada em qualquer lugar, inclusive na aula, e em qualquer dispositivo eletrônico com acesso à Internet, fornecendo aos alunos conteúdo histológico e uma maneira de testar e aprimorar seu conhecimento. Para aumentar a interatividade e a motivação, a ferramenta utiliza a narrativa para fornecer conteúdo teórico ao aluno. Esta história tem um personagem principal e um personagem lateral e, através de diálogos, exposições teóricas, imagens histológicas digitais, diagramas e desenhos, fornece aos alunos o conteúdo da disciplina. O conhecimento adquirido é então testado e aprimorado pela introdução de muitos desafios diferentes relacionados ao conteúdo da história, que devem ser respondidos corretamente pelos alunos para obter progresso. A ferramenta oferece também um atlas histológico feito de imagens microscópicas e suas respectivas informações.

Todo o conteúdo informativo da ferramenta, incluindo histórias, desafios e imagens de atlas, é gentilmente fornecido pela professora Paula Cristina Silva, da Histologia Funcional, do Instituto de Ciências Biomédicas Abel Salazar, no Porto.

|

# Table of Contents

<b>Introduction .....</b>	<b>19</b>
1.1 Context .....	19
1.2 Motivation .....	20
1.3 Objectives .....	21
1.4 Document Structure.....	21
<b>Problem Characterization .....</b>	<b>23</b>
2.1 An online histologic book .....	23
2.2 Approach .....	24
<b>State of the Art Review .....</b>	<b>26</b>
3.1 E-learning .....	26
3.1.1 E-learning technologies.....	27
3.1.1.1 CD-ROM	27
3.1.1.2 Learning Management Systems	27
3.1.1.3 Content Management Systems	27
3.1.1.4 Augmented / Virtual Reality	27
3.1.1.5 Games	28
3.1.2 Microlearning, a current strategy in E-learning. ....	28
3.2 Gamification.....	28
3.2.1 Gamification in E-learning .....	29
3.2.2 Related Work. ....	30
3.2.2.1 Kahoot! –	30
3.2.2.2 Duolingo –	30
3.2.2.3 Codecademy –	30
<b>Solution.....</b>	<b>31</b>
4.1 Overview .....	31
4.2 Folder/File Structure .....	32
4.2.1 Frontend .....	32

4.2.2	Backend.....	34
4.3	Database .....	39
4.4	Authentication/Authorization.....	42
4.4.1	JWT authentication/authorization. ....	43
4.4.2	Registration (signup).....	43
4.4.3	Login .....	44
4.5	Procedural generation.....	44
4.5.1	Challenges and story. ....	44
4.5.1.1	Challenges and story file structure (for each chapter).	44
4.5.1.2	Parsing the files into the database.	47
4.5.2	Atlas. ....	48
4.5.2.1	Atlas file structure (for each chapter).	48
4.5.2.2	Parsing into the database.	49
4.6	API .....	49
4.7	User Interface.....	51
4.7.1	Registration View.....	51
4.7.2	Login View.....	53
4.7.3	Main Menu View. ....	54
4.7.4	Leaderboard View. ....	55
4.7.5	Chapter Selection View (challenges). ....	56
4.7.6	Chapter Selection View (Atlas).....	58
4.7.7	Storyboard View. ....	59
4.7.8	Challenge Selection View. ....	60
4.7.9	Challenge View (CHOICE). ....	61
4.7.10	Challenge View (MCHOICE).....	63
4.7.11	Challenge View (CROSSWORD). ....	64
4.7.12	Challenge View (FILL).....	66
4.7.13	Challenge View (LETTERS).....	67
4.7.14	Challenge View (PAIR).....	69
4.7.15	Challenge View (GROUP).....	70
4.7.16	Challenge View (BOSS).....	72
4.7.17	Atlas Item Selection View.....	73
4.7.18	Amplified Atlas Item View.....	74
4.7.19	Admin Area (Users) View.....	76
4.7.20	Admin Area (Chapter/Challenges) View. ....	77
4.8	User Testing. ....	79
	<b>Conclusions and Future Work.....</b>	<b>81</b>
	<b>Attachments.....</b>	<b>83</b>
6.1	Attachment 1 – API description. ....	83



6.2	Attachment 2 – User test questionnaire.....	122
6.3	Attachment 3 – Example of .CSV files. ....	128
6.3.1	Chapter/Dialog file.....	128
6.3.2	Atlas file. ....	128



# Pictures List

Figure 1 - .....	34
Figure 2 - .....	41
Figure 3 - .....	55
Figure 4 - .....	56
Figure 5 - .....	57
Figure 6 - .....	58
Figure 7 - .....	60
Figure 8 - .....	61
Figure 9 - .....	62
Figure 10 - .....	63
Figure 11 - .....	65
Figure 12 - .....	66
Figure 13 - .....	68
Figure 14 - .....	69
Figure 15 - .....	71
Figure 16 - .....	72
Figure 17 - .....	74
Figure 18 - .....	75
Figure 19 - .....	76
Figure 20 - .....	77
Figure 21 - .....	79
Figure 22 - .....	81



# Tables List

Table 1 – .....	35
Table 2 – .....	37
Table 3 – .....	42
Table 4 – .....	42
Table 5 – .....	42
Table 6 – .....	43
Table 7 – .....	43
Table 8 – .....	43
Table 9 – .....	44
Table 10 – .....	44
Table 11 – .....	46
Table 12 – .....	46
Table 13 – .....	47
Table 14 – .....	47
Table 15 – .....	48
Table 16 – .....	48
Table 17 – .....	48
Table 18 – .....	49
Table 19 – .....	49
Table 20 – .....	49
Table 21 – .....	49
Table 22 – .....	49
Table 23 – .....	51
Table 24 – .....	51
Table 25 – .....	54
Table 26 – .....	55
Table 27 – .....	57
Table 28 – .....	58

Table 29 – .....	59
Table 30 – .....	60
Table 31 – .....	62
Table 32 – .....	62
Table 33 – .....	64
Table 34 – .....	65
Table 35 – .....	66
Table 36 – .....	67
Table 37 – .....	69
Table 38 – .....	70
Table 39 – .....	72
Table 40 – .....	73
Table 41 – .....	75
Table 42 – .....	76
Table 43 – .....	77
Table 43 – .....	79







# Abbreviations and Symbols

EG - Educational games.

APP - Application.

API - Application Program Interface.

JWT – JSON web token.



## Chapter 1

# Introduction

### 1.1 Context

Histology is a biomedical science that studies the microanatomy of cells, tissues, and organs. Histology is important as many diseases occur at cellular/tissue level and knowing how a tissue normally looks and works can help to identify certain diseases, what causes them as well as if an applied treatment is working [19].

It is usually divided into 2 big groups, general and systematic histology. While general histology studies the four fundamental types of tissues (epithelial, connective, muscular and nervous), systematic histology studies the microanatomy of the different organs and systems of the human body.

As a course, histology is usually taught in a practical self-directed way, where students are handed microscopes as well as slides and are motivated to make their own research. In the beginning of this century, many advances were made regarding the capture of digital images from microscopes and, therefore, big databases were created and shared online. This enables students to study not only in class but also at home as complimentary practice. The problem is that using these strategies the student studies, most of the time, alone, and this can affect his/her motivation and engagement in a negative way. It is known that motivation is a key factor in human behavior and that low motivation leads to a reduction of the time spent in an activity as well as the effort and persistence invested in it. [1]

Nowadays university students use modern technology devices since they were kids and dedicate big part of their time to technology activities. Since their contact with technology is so developed and integrated in their daily life, why not take advantage of it? Technological devices, such as a computer or a smart phone, can be powerful tools in the creation of unique and unprecedented experiences for academic and training organizations that can be complementary to

## Introduction

the classical methods. Therefore, it is advantageous that teachers adapt their methods of teaching and embrace these new technologies in their pedagogical practices in a way that everyone comes out winning.

Teachers have an important role in the learning efficiency of their students and one key competence is the ability of keeping student focus and engagement in their performing tasks [2]. To aid teachers in their quest towards education, Educational Games (EGs) can be introduced, these EGs have great effect and have been used as educational tools for long, and with really great results [3-7]. They are powerful, innovative learning tools that instead of having their focus in entertainment are more education driven [8]. EGs can be used by the teacher to raise student's motivation, enthusiasm and moral, helping them integrate in and contribute to the learning environments they constitute [9, 10].

Other powerful tool is storytelling, as old as humanity, it can be considered an art that greatly impacts humans and their behavior. Stories can be told in a variety of ways including pictures, messages, conversations, presentations, letters, audio and visual recordings. With the advance of technology, it is now possible to tell stories in a digital way, making use of such powerful tool in universities to teach students is only beneficial since they also can engage, motivate, inspire deep reflection and support student centered activities [11].

Aimed to increase biomedical student's internal motivation and self-regulation for learning, it was planned the creation of a digital of book for General Histology. We propose the integration of digital story with digital educational games. In the beginning of each book chapter the theoretical background is presented as a story and then students are confronted with some challenges related with the story. The main goal of this digital histology book is to establish an environment that optimize students' internal motivation, active learning, and ultimately their intellectual development and performance.

## 1.2 Motivation

Regarding that Histology is a practical science that can only be taught, given its subject, by making use of a microscope and theoretical slides, it is really hard for students to study at home, as they either don't own a microscope or can't get their hands to good, free and usable study material from their own university. They normally study by printed atlases and some digital tools freely available from other universities. This can be very hard and frustrating and reduce student motivation and efficiency in the leaning of Histology.

The course of "Functional Histology" given by the University of Porto to the 1<sup>st</sup> year's students of the Biochemistry Degree, is example of this problem, as during the course the students are only provided with 5 hours of contact per week and even though they are provided with

## Introduction

content during classes by either theoretical expositions, observation of microscope preparations or educational games it is not enough, and when students go home or are alone at the microscope they suffer from a lack of supportive quality content. In the last 2 years of the course even though the average grade was 15.4 values there was a significant 31% of students that didn't obtain course approval. The lack of supportive outside class environment material is seen as one of the main reasons why this happened, because it creates a lack of autonomy in the study and therefore the motivation of the student drops considerably. Motivation is a key feeling when aiming to achieve goals and when it is missing the way to something can become a real challenge.

In order to battle this lack of content as well as the lack of motivation it was envisioned a solution that not only provides theoretical content but also tries to increase student motivation and focus by making use of the powerful motivational tools that are EGs and storytelling, this in the form of an online application that can be used anywhere in any electronic device connected to the internet and can be easily updated by the teacher to satisfy his/her needs.

### 1.3 Objectives

The main objective is to create an online application that is able to give the students quality histological study material, available anywhere and in any smart electronic device, which provides a histological atlas, a storytelling and EG's challenges. Within the app there is also the implementation of motivational factors such as points and progress that can also help students give their best and be more focused and engaged while learning histology.

To allow that teachers must be able to update the app accordingly to their needs without having any actual code knowledge, the proposed solution implements a procedural generation module that can create/update the entire application (story, atlas and challenges) by making use of simple editable and understandable files, this way the teacher can modify the entire application without much effort.

### 1.4 Document Structure

The document is divided in 3 more chapters. In every chapter a different concept is presented and subdivided. In the 2<sup>nd</sup> chapter, "Problem Characterization", is where we analyze the described problem, propose a solution and make an overview of the chosen approach taken towards solving it. Next comes the 3<sup>rd</sup> chapter, "State of the Art Review", in this chapter it is made a literature review on the concepts related to the project, it is taken an approach on the Elearning concept, explaining it and showing some technologies and methods and also on the Gamification concept, where the concept and its application to e-learning is described, it is also shown some related work and projects. Following, the 4<sup>th</sup> chapter is presented, "Solution", in this

## Introduction

chapter the implemented platform , per se, is described. An overview of it is done and more technical concepts about the struture, database and the other platform modules , are shown. Also, information about the user testing is provided. Finally , in the 5<sup>th</sup> chapter, “Conclusion and Future Work”, conclutions about the project are given and possible future work is debated.

At the end of the document references are provided, followed by attatchemnts.

## Chapter 2

# Problem Characterization

### 2.1 An online histologic book

As stated previously, the need for complementary material to teach Histology lead to a solution that not only could provide the students with such material as well as try to help them psychologically in the matters of motivation and focus towards the course. The idea came from Prof. Paula Silva and it is an online “histologic book” that tells the story of the adventures of Karl Mayer and the Cell Wizard, together they must save the “Histokingdom” from the hands of the malefic villain. In their quest they must travel through different histologic places and overcome all the trials left by the villain, so they can progress and free the Histokingdom. This story is used to support the students learn the necessary complementary theoretical contents of the course and explain the elements found in the four fundamental tissues present in General Histology. As for the challenges they are used as a learning and testing tool and complement the story in the learning of said tissues. There are 8 different types of challenges that can be used by the teacher:

#### **CHOICE -**

A quiz with multiple-choice questions and with a given number of possible answers but only a correct one. Questions are similar to the ones of the theoretical final exam.

#### **MCHOICE -**

A quiz with multiple-choice questions and with a given number of possible answers but with multiple correct ones.

#### **CROSSWORD -**

A crossword game in which the answers to histological questions, called hints, are written in rows of squares that cross each other so that some letters are shared.

## Problem Characterization

### **FILL-**

In this challenge the player must choose the most appropriate words, amongst a list, to complete blank spaces in order to form a correct sentence.

### **GROUP-**

Students must choose the appropriate group, between 2, for a given question.

### **LETTERS-**

Students must find the answer to the question using some of the letters that are given.

### **PAIR-**

Students must match left items to the right items, creating correct connections.

### **BOSS-**

This challenge is a quiz competition where the goal is to correctly answer a series of consecutive multiple-choice questions about the chapter. Four possible answers are given, and the student who is playing must choose the correct one. Aimed to prepare students, the game questions have the structure and the difficulty level of the ones of theoretical final exam and they cannot fail, otherwise must start the challenge again.

Also, the teacher asked for the integration of a simple histologic atlas in the application where histologic pictures are presented and a subtitle accompanies, explaining and giving meaning to the image.

The thought solution that could put all this together was an online app with a simple interface that could not only offer all this content to the student but also be easily and totally updatable by the teacher without changing any code. This also enables teachers from other courses or areas to be able to create a support tool regarding their context, easily creating an entirely different experience. With this comes the necessity of a system implementation, which content can be entirely and easily updated without any coding. Therefore, comes the need to implement a procedural generation solution that aids teachers in the creation and update of the App content. This solution implements a procedural generation module that handles all App's content management (story, challenges, atlas) by making use of simple files that can be easily edited by the teachers, without any coding knowledge needed. This also enables the App to be flexible and therefore used in many different contexts without having to change any code.

## **2.2 Approach**

The approach used was the creation of an html/javascript/css/php gamified platform that provides users with an atlas, a narrative that helps them understand and compliment class contents and lets them engage in challenges to test and refine their knowledge, also there is the implementation of a point system and leaderboards in order to boost motivation and healthy competition between students. To access the app users must sign in into the system, this can only



## Problem Characterization

be done using “Validation Keys” distributed by the teacher, and after that, log in using the sign in credentials. This Validation Keys as well as the all the database contents can be easily managed by the teacher in an admin restricted area that allows them to manage and supervise the application.

The entire app is done via an html/javascript/css interface that communicates with an online server database by the means of a php API and AJAX requests. This database contents are flexible and can be updated any time in the admin area by making use of .CSV files created from a simple excel spreadsheet. These files have a fixed syntax that enables the above described procedural generation module to work and create the application contents.

This allows the solution to be accessed anywhere by any device with a browser and an internet connection, since the technologies used are multi-platform. Also, by making use of this technologies it makes it easy to update code and refine the app without having much trouble with version and deployment since the only thing needed is a webserver and database host.

By using the same code and altering the .CSV files opens horizons to many different educational apps that can make use of the same APP model to teach and support many students in many different areas of education.

## Chapter 3

# State of the Art Review

### 3.1 E-learning

E-learning is the use of writing and communication technologies, visualization, and storage in order to convey information with the principal objective of education. This concept was firstly focused in the improvement of task accomplishment and only after some time it turned its focus to student's education. The term "E-learning" was not the first term used to describe the use of computerized systems to aid in the learning and teaching process and was first used in 1983 by Mary Alice White in a journal article entitled "Synthesis of Research on Electronic Learning." and was defined as "learning via electronic sources, such as television, computer, videodisk, teletext, videotext." (White, 1983, p. 13), E-learning stands for electronic learning and is nowadays a very common and known term used to describe systems that make use of technology and the internet to overcome space and time issues in the learning process.

As society progresses so does its demands, causes an increasingly need for students to adapt and become even more efficient and autonomous in learning. Therefore, academics must also to adapt and understand the process of acquiring information while trying to increase its efficiency. Given the need in the fast search and acquirement of knowledge, E-learning development has been a priority for many years now and many comparison studies, pedagogical aspects, perception studies, and evaluation to monitoring studies were carried out. This has had great effect in the development of e-learning strategies and models throughout the years. [20]

Nowadays, the benefits of e-learning are being systematically integrated as a complement of traditional teaching by many universities around the world aiming to achieve better results in many aspects of student performance, such as: satisfaction, motivation, efficiency and effectiveness [14].

### **3.1.1 E-learning technologies.**

The attention given to E-learning as well as its use has been continuously increasing, therefore there is a constant need for both technologies evolution and upgrade used as a support mechanism to teach. There are many technologies that can be used to provide information that have been used since the early 90's. Many technologies persisted with the passing of time and continued to be used. However, with the appearance of new technologies, some of the older and less effective ones have been exchanged. Some of the technologies used in the history of E-learning are:

#### **3.1.1.1 CD-ROM**

CD-ROM media was mainly used in the early 90's as a starting e-learning technology, it was used to hand learning material in either text or multimedia formats to students in long distance programs. This technology encourages students in learning alone and by themselves increasing their autonomy in such activity. It was used special training programs that would not only provide the student with information but also exercises in which he could train the acquired knowledge. This computer-based training can be compared for example with the likes of foreign language learning tutorials. [20]

#### **3.1.1.2 Learning Management Systems**

Learning management systems are information systems that make use of software to deliver courses and training programs to students, teachers and administrators. These systems are highly customizable software that can be used to analyze strengths and weaknesses of the user and adapt accordingly, concentrating more in their weakness. They often provide a method of long-distance communication with educators that helps users clarify any doubt they have. [20, 21]

#### **3.1.1.3 Content Management Systems**

Content management systems are systems that aim to facilitate a collaborative creation and management of digital content. Users in this type of systems are usually given role-based access to the management of documents and digital assets inside the system, all this in a unique and centralized environment. Example of this type of system is the well-known Moodle. [20, 22]

#### **3.1.1.4 Augmented / Virtual Reality**

Whereas augmented reality can be described as a reality where real life objects are either substituted by or integrated with virtual 3d objects, virtual reality is a full virtual 3d world created with the purpose of appearing real creating a high state of immersion. Both virtual environments

## State of the Art Review

can be used to catch student's attention while increasing their focus and engagement in learning activities. Also, they help creating and showing abstract and nonexistent concepts that can be easier interpreted by the student by looking at them than by imagining them in their mind. They also allow the creation of "safe" environments that make possible the simulation and experience of certain activities without any real word consequences. [20, 21]

### **3.1.1.5 Games**

Gaming technologies can also be used to enhance and ease the process of learning, by using built-in simulations and interactions games can provide the player a rich learning environment without the pressure or boredom of classical methods, being fun and interactive are their biggest advantages since it promotes in the students a wellbeing and relaxed feeling that can break many learning barriers. Furthermore, some of these games can have an online component, which motivates and facilitates knowledge sharing and creation. These types of games have many types and concepts and range from text-based games to games with rich virtual environments and many players that can play simultaneously. [20]

### **3.1.2 Microlearning, a current strategy in E-learning.**

As stated previously, the world is evolving and with it so are learning processes, one of the trending learning strategies and methods is microlearning. Microlearning is a learning process of low duration that aims to only one small learning objective. [23] This type of learning is said to have in average 20% better results than the classical learning method where students are given high amounts of information at a time. [24] This is due to the fact that in classical learning methods there is less interaction between students and the information given, whereas in the microlearning process the student after acquiring a small amount of information is able to test it right away, providing a more dynamic learning experience. Furthermore, when small parcels of information are given it leads to the thought that said information is crucial and being able to the know right away what is expected from it also facilitates the whole process of learning. [24] This said, the use of microlearning as an e-learning strategy is advisable when possible since short bits of information are more manageable and digestible and that helps students absorb and retain information. [24]

## **3.2 Gamification**

Games always have been present in a human being's life. They are existent in our daily routine and, more than providing entertainment, also help us develop physical and intellectual capabilities that are useful in life. This happens because to play a game and meet its goals, the

player must overcome challenges and, while also having fun. This helps to break psychological barriers that are present in more classic methods of personal development. Having this principle in mind, it is advantageous to create applications with mechanics that have the objective of stimulating specific skills of the users. The use of applications that incorporate classic game mechanics in order to make the user improve is called “gamification”.

The potential of the use of games as a tool for teaching has been studied for more than two decades, and even though the term “gamification” has been introduced in the early 2000s, only in 2011, it was given the proper attention. From the broad area of the Ludification of Culture, Sebastian Deterding defines Gamification as the use of game elements and game design techniques in non-game contexts [17]. Since then, there has been an increasing number of applications developed for this purpose under many domains, such as productivity, finance, health, education, and sustainability [13].

The purpose of using gamification is to enhance the user experience in activities not related to games, by the contact of the users with applications that drive them to have the same level of engagement as if they were playing a classic game. This concept is said to work [12] because it creates in the user motivational sensations that are typically induced by traditional games. These sensations are generated by the implementation of fundamental game mechanics in the application, such as the existence of points, leaderboards, achievements/badges, levels, story, progress, challenge, and rewards. Most of these motivational elements present positive effects when implemented and tend to engage better and captivate user attention, generating better results when it comes to acquiring competencies or skills.

### **3.2.1 Gamification in E-learning.**

As previously said, gamification is being applied in different context domains to improve specific activities and processes. This method was highly used to improve corporation employee’s skills and in education. The correct use and combination of e-learning and gamification is a powerful tool and can raise, as referred previously, the satisfaction, engagement, effectiveness, and efficiency of the students. The right combination of e-learning, gamification, and well-designed tasks takes the student to a different level of engagement and concentration only typical of games, creating like this a more efficient and fun learning environment. The essence of this method does not reside in the technology itself but more in the creation of said productive learning environment. With a decision/reward system, designed proposedly to increase concentration levels in the learning experience, makes the student not only engage in learning activities with a more significant frequency but also makes them engage with a clearer mind. This provides the student with continuous self-improvement and feedback experiences that adapt to his actions and help the student progress fast.

### **3.2.2 Related Work.**

Due to the growth and use of this concept, more and more game based applications, whose purpose is to motivate and help the user are emerging, and being used in various contexts of application like, education, business and personal chores. These platforms are used to provide the user with different good sensations, such as motivation, engagement, enjoyment, sense of achievement and status in order to increase their productivity and efficiency in the desired tasks. [26]

#### **3.2.2.1 Kahoot! –**

Kahoot! is an online gamification platform for quiz generation and being designed to serve as a social learning tool as it is an application for group sessions. It lets a user create a series of quizzes and playable questions in which participants compete against each other. As a motivational context, at the end of each game session, the results and statistics are shown. The great advantage of this application is the existence of the functionality to export both the results and a descriptive analysis of them in each session, as this data can be used later by instructors or teachers. [15]

#### **3.2.2.2 Duolingo –**

As the previous platform, is a learning platform with gamification elements. It is one of the most widely used language learning platforms [16]. To achieve the primary goal of teaching a language, the application presents several challenges for the user to overcome, while points and a rewards system motivates and propels the player.

#### **3.2.2.3 Codecademy –**

Codecademy is an online platform that aims to teach its users software development and coding. In this platform it is asked the users to complete various lessons and exercises and awards them with badges, progress and points in order to give users a sense of fulfillment.

## Chapter 4

# Solution

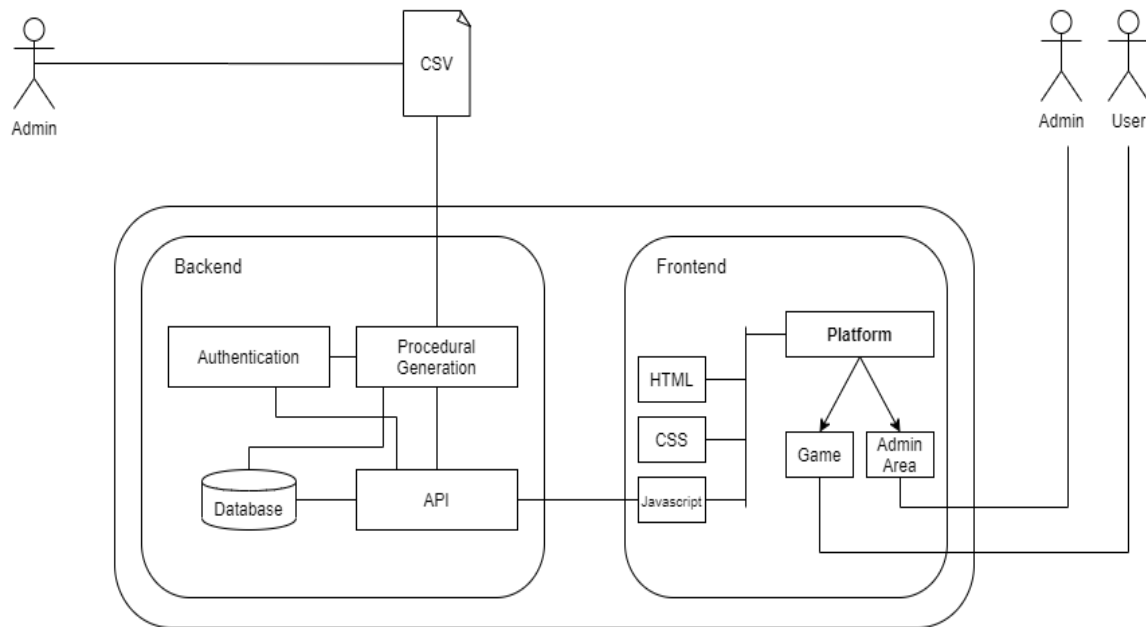
### 4.1 Overview

It is proposed a solution supported by a gamified platform that allows the user to engage in the histologic story and following challenges mentioned above, while also serving as an atlas that aids students in their studies. This platform is composed of two modules (backend and frontend). The backend is responsible for generating and storing all the data for the challenges, authenticating the user, and sending data to the frontend. The frontend is accountable for retrieving and processing that data and then providing it to the student as a fluid and game-like experience.

The architecture of this solution is depicted in figure 1. The teacher/admin specifies the story and the challenges on an XLS file that needs to be saved as a CSV file for an easier parsing. This simple solution allows for any teacher to create or adapt a specific solution to their class. This file is processed by the procedural generation module that stores in the database the required information for deploying the digital book to the students. There is an authentication module for the teacher/admin and a PHP API that create the HTML content for the frontend.

Built with the use of HTML, JavaScript, and CSS, the frontend is designed to have a game-like and straightforward interface that allows the user to navigate fluidly between screens and engage in a game scenario while presenting the challenges. It is also responsible for communicating with the API, via AJAX, either to retrieve or to send data. These technologies were chosen since any smart device with a browser and an internet connect can access and run them.

## Solution



**Figure 1** - Architecture of the platform.

## 4.2 Folder/File Structure

The folder/file structure of the proposed solution is also organized in two main modules, “Frontend” and “Backend”. These modules then open into more files and folders, their organization and meaning are described in the next sections.

### 4.2.1 Frontend

This module (folder) contains all the files that take direct part on the user interface and interaction with the platform. These files provide direct interaction between user and platform and contain the html, javascript, css and assets used to create and enhance the user interface and that allow the users to engage in the platform visually. In this module interfaces for platform navigation, as well as for interfaces that let the user engage in the 8 types of challenges, explore the atlas and experience the narrative are created and contact with the backend is established, this contact is needed to retrieve persistent database information, that is then shown to the user, making use of said interface Also, its where the files for the admin area interface are provided, they allow a visual management of the platform and its users.

Its file structure is described below:

#### Frontend/

**Table 1** – File/Folder description of the Frontend folder.



## Solution

Name	Description
assets/	Folder that holds the frontend module assets that are used in its .html files.
assets/bootstrap/	Folder that holds bootstrap library include files.
assets/css/styles.css	File that contains all personalized css of the user interface.
assets/datatables/	Folder that holds the datatables library include files.
assets/fonts/Prime-Regular.ttf	File that contains the text font used in the platform.
assets/ images/	Folder that holds the platform static images (ex: icons).
assets/jquery/	Folder that holds the jquery library files.
admin_chapter.html	File that contains the html and javascript of the admin area chapter section.
admin_dashboard.html	File that contains the html and javascript of the admin main section.
admin_home.html	File that contains the html and javascript of the admin area sidebar.
admin_users.html	File that contains the html and javascript of the admin area users' section.
amplified_atlas.html	File that contains the html and javascript of an atlas item page.
atlas.html	File that contains the html and javascript of an atlas chapter items page.
BOSS.html	File that contains the html and javascript of a “BOSS” type challenge.
CHOICE.html	File that contains the html and javascript of a “CHOICE” type challenge.
choose_atlas_chapter.html	File that contains the html and javascript of the of the chapter selection menu for the atlas.
choose_challenge.html	File that contains the html and javascript of the challenge selection menu.
choose_chapter.html	File that contains the html and javascript of the chapter selection menu for the challenges.

## Solution

CROSSWORD.html	File that contains the html and javascript of a “CROSSWORD” type challenge.
dialog.html	File that contains the html and javascript of the storytelling page.
FILL.html	File that contains the html and javascript of a “FILL” type challenge.
GROUP.html	File that contains the html and javascript of a “GROUP” type challenge.
index.html	File that contains the html and javascript of the index page (first page)
leaderboard.html	File that contains the html and javascript of the user’s leaderboard page.
LETTERS.html	File that contains the html and javascript of a “LETTERS” type challenge.
login.html	File that contains the html and javascript of the user’s login page.
Main_menu.html	File that contains the html and javascript of the platform’s main menu.
MCHOICE.html	File that contains the html and javascript of a “MCHOICE” type challenge.
PAIR.html	File that contains the html and javascript of a “PAIR” type challenge.
Signup.html	File that contains the html and javascript of the user’s signup page.

### 4.2.2 Backend

This module (folder) contains all the files that take part on the platform’s chapter, atlas and users info creation and persistence as well as API files that retrieve said info. The backend provides contact between the user interface (frontend) and all the persisted data (API files), a mode of access and update of the database, as well as the authentication module, where files responsible for authentication of the users, and generation module, where all the files responsible for the generation of the chapters, challenges and story are, as well as the dynamic assets used. It’s the destination of the csv files created by the admin in order to generate said content. Also, in here, configuration of the platform constants is made.

Its file structure is described below.

Solution

**Backend/**

**Table 2** – File/Folder description of the Backend folder.

<b>Name</b>	<b>Description</b>
API/	Folder that contains all API files.
API/clear_database.php	File that contains the php that accepts an admin associated token and empties the whole database.
API/generate_registration_key.php	File that contains the php that accepts an admin associated token and generates a registration key.
API/get_atlas_item.php	File that contains the php that accepts an atlas item id, retrieves its info and outputs it as JSON data.
API/get_BOSS_arguments.php	File that contains the php that accepts a chapter id, retrieves its BOSS challenges info and outputs it as JSON data.
API/get_challenge_arguments.php	File that contains the php that accepts a challenge id, retrieves its info and outputs it as JSON data.
API/get_chapter_atlas.php	File that contains the php that accepts a chapter id , retrieves its atlas items' info and outputs it as JSON data.
API/get_chapter_challenges.php	File that contains the php that accepts a user associated token as well as a chapter id, retrieves its chapters list and outputs it as JSON data.
API/get_chapter_dialog.php	File that contains the php that accepts a chapter id, retrieves its dialog (story) info and outputs it as JSON data.
API/get_chapter_info.php	File that contains the php that retrieves chapter info and outputs it as JSON data.
API/get_leaderboard.php	File that contains the php that retrieves the 10 users with more points info and outputs it as JSON data.
API/get_registration_keys.php	File that contains the php that accepts an admin associated token, retrieves all registration keys and outputs them as JSON data.
API/get_show_dialog.php	File that contains the php that accepts a user associated token as well as a chapter id, retrieves if said user

## Solution

	completed the first challenge of said chapter and outputs that information as JSON data.
API/get_user_info.php	File that contains the php that accepts a user associated token, retrieves said user's info and outputs it as JSON data.
API/get_users_info.php	File that contains the php that accepts an admin associated token, retrieves every user's info and outputs it as JSON data.
API/login.php	File that contains the php that accepts an email and password, verifies if said email and password are associated with a user, and if that's true, outputs an encrypted JSON web token as JSON data.
API/reset_users_progress.php	File that contains the php that accepts an admin associated token and resets all user's challenge progress.
API/signup.php	File that contains the php that accepts, a valid email, password, name and validation key and registers a user in the platform.
API/unlock_lock_chapter.php	File that contains the php that accepts an admin associated token as well as a chapter id and either locks or unlocks said chapter.
API/verify_BOSS.php	File that contains the php that accepts a chapter id, a BOSS challenge id, a user associated token and the user selected answers and verifies if they are correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.
API/verify_CHOICE.php	File that contains the php that accepts a CHOICE challenge id, a user associated token and the user selected answer and verifies if it is correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.
API/verify_CROSSWORD.php	File that contains the php that accepts a CROSSWORD challenge id, a user associated token and the user selected answers and verifies if they are correct. If so,

## Solution

	<p>completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_FILL.php	<p>File that contains the php that accepts a FILL challenge id, a user associated token and the user selected answer and verifies if they are correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_GROUP.php	<p>File that contains the php that accepts a GROUP challenge id, a user associated token and the user selected answers and verifies if they are correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_LETTERS.php	<p>File that contains the php that accepts a FILL challenge id, a user associated token and the user selected answer and verifies if it is correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_MCHOICE.php	<p>File that contains the php that accepts a GROUP challenge id, a user associated token and the user selected answers and verifies if they are correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_PAIR.php	<p>File that contains the php that accepts a PAIR challenge id, a user associated token and the user selected answer and verifies if is correct. If so, completes the challenge for said user, unlocks the next and outputs the number of points won as JSON data, otherwise outputs the correct answer.</p>
API/verify_token.php	<p>File that contains the php that receives a user associated token, verifies if it is valid and outputs the result as JSON data.</p>

## Solution

API/verify_token_admin.php	File that contains the php that receives a admin associated token, verifies if it is valid and outputs the result as JSON data.
authentication/	Folder that holds authentication related files.
authentication/libs/	Folder that holds JSON web token library include files.
authentication/TokenChecker.php	File that contains the php class that creates and validates JSON web tokens.
authentication/User.php	File that contains the php class that persists new users in the database and validates users' credentials
database/	Folder that hold files that communicate directly with the database.
database/config/db.php	File that contains the php that is responsible for connecting to the database.
database/DatabaseController.php	File that contains the php class that is responsible for directly executing queries on the database as well as treating the retrieved info.
generation/	Folder that holds the procedural generation files.
generation/atlas/	Folder that holds the .CSV files needed for the generation of the chapters atlas.
generation/chapters	Folder that holds the .CSV files needed for the generation of the chapters challenges and dialog.
generation/reload_atlas.php	File that contains the php that is responsible for reading the generation/atlas/ directory files and generating the atlas contents in the database.
generation/reload_challenges.php	File that contains the php that is responsible for reading the generation/chapters/ directory files and generating the challenges and dialog info in the database.
images/atlas/	Folder that holds the atlas items images.
images/challenges/	Folder that holds the challenges' images
core.php	File that contains the php used for the platform's core configuration.

## Solution

### 4.3 Database

The database for the solution's platform is a MySQL database hosted in the localhost and with the name "histology\_db". The platform makes use of the database module (db.php) to connect to this database using the MySQL private credentials, this returns a MySQL connection that is used to directly interact with the database via php. The database structure was created having in mind the need of a platform with registered users and its information ('users' table) that can only signup with the use of one-time use keys ('registrationKeys' table). Also, there is the need to persist data about the generated chapters ('chapter' table) and its contents, namely, challenges info and data ('challenge' and 'arguments' table, respectively), atlas items description and images ('atlas' table) and the storyboard moments associated with every chapter ('dialog' table). There is also a need for the persistence of user progress, otherwise users would see their challenges progress disappear after logout, this is made by persisting information that relations users and challenges ('userchallenges' table).

This database holds all the persistent info of the platform and can be controlled by the users and the admin via an API and the use of a database controller that interacts directly with it. Its tables relation is depicted in figure 2 and the entities configuration and description are presented after.

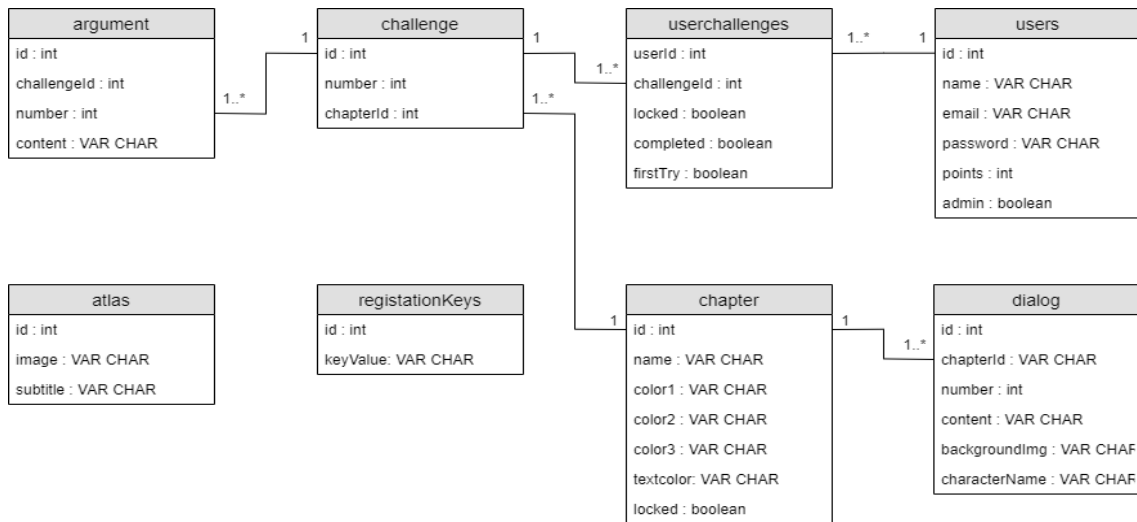


Figure 2 - Database's relational schema

- **users** – represents and holds the platform's user data.

Table 3 – 'user' table's columns description.

Column name	Type	Description
id	Integer / AUTO INCREMENT	User's id in the system.

## Solution

name	VAR CHAR	User's first and last name.
email	VAR CHAR	User's registration email.
password	VAR CHAR	User's password encrypted hash. (using bcrypt php algorithm)
points	Integer	User's current number of points.
admin	Boolean	Represents if the user is or is not a system admin.

- **registration Keys** – represents and holds data of the keys that can be used for user registration.

**Table 4** – ‘registrationKeys’ table’s columns description

Column name	Type	Description
Id	Integer / AUTO INCREMENT	Key id in the system.
keyValue	VAR CHAR	13-character, unique string that represents the key value used for user registration.

- **chapter** – represents and holds existent chapter’s info.

**Table 5** – ‘chapter’ table’s columns description.

Column name	Type	Description
id	Integer	Chapter’s order number.
name	VAR CHAR	Chapter’s name.
color1	VAR CHAR	Chapter’s primary color code in hexadecimal.
color2	VAR CHAR	Chapter’s secondary color code in hexadecimal.
color3	VAR CHAR	Chapter’s ternary color code in hexadecimal.
textcolor	VAR CHAR	Chapter’s text color code in hexadecimal.
locked	Boolean	Represents if the chapter is or is not locked.

- **challenge** – represents and holds all the platform’s challenges info, this table represents challenges and does not store the actual challenge data. Every challenge is related to one, one only chapter.

**Table 6** – ‘challenge’ table’s columns description.



## Solution

Column name	Type	Description
id	Integer	Challenge's id in the system.
number	Integer	Challenge's order number in its chapter.
chapterId	Integer	Challenge's chapter id. Id that identifies a chapter table row.

- **argument** – represents and holds every challenge actual data (arguments), every argument is related to one, and one only challenge.

**Table 7** – ‘argument’ table’s columns description.

Column name	Type	Description
id	Integer/ AUTO INCREMENT	Argument's id in the system.
challengeId	Integer	Argument's challenge Id. Id that identifies a challenge table row.
number	Integer	Argument identifier number. This gives a meaning to the argument in a challenge context.
content	VAR CHAR	Argument value (content) string.

- **users challenges** – represents every user's progress in every challenge and holds a relation between a user and a challenge.

**Table 8** – ‘userchallenges’ table’s columns description.

Column name	Type	Description
userId	Integer	User's id associated with the relation.
challengeId	Integer	Challenge's id associated with the relation.
locked	Boolean	Represents if the specified challenge is locked or not for the specified user.
completed	Boolean	Represents if the specified challenge was or not successfully completed by the specified user.
firstTry	Boolean	Represents if the specified user already tried to complete the specified challenge.

## Solution

- **dialog** – represents and holds all the dialog data, every row represents a dialog moment and is related to one, and one only chapter.

**Table 9** – ‘dialog’ table’s columns description.

Column Name	Type	Description
Id	Integer / AUTO INCREMENT	Dialog moment id
chapterId	Integer	Dialog moment chapter Id. Id that identifies a chapter table row.
number	Integer	Dialog moment order number in the specified chapter.
content	VAR CHAR	Dialog moment string. (Quote)
backgroundImg	VAR CHAR	Name of the background media of the dialog moment.
characterName	VAR CHAR	Name of the character that speaks the content (quote).

- **atlas** – represents and holds every atlas item data.

**Table 10** – ‘atlas’ table’s columns description.

Column name	Type	Description
id	Integer	Atlas item id in the system.
image	VAR CHAR	Name of the atlas item image.
subtitle	VAR CHAR	String that explains the atlas item image.

## 4.4 Authentication/Authorization

The platform is protected by an authentication module that controls the access to its information as well as its pages. To access its contents a user must register in the system and after log into it, only then he is able to play the challenges, view the story and access the atlas. There is a restricted access area and functionalities only available to admins. The authorization and authentication control are made using JSON Web Tokens (JWT) and is explained below.

#### 4.4.1 JWT authentication/authorization.

In order to communicate with the backend API a user must first be authenticated by the authentication module, this is done via JWT mechanism, that as said in [25] “is a compact, URL-safe means of representing claims to be transferred between two parties”. This allows the authentication of an user in the API via the exchange of an encrypted token that contains user information (in this case, the user id, user email and user role) , the token is created at user login using its information and is encrypted using the JWT library and with the use of a private key (stored in Backend/core.php configuration file), it is then sent to the frontend via HTTP response and stored in browser cookies. Every time the user needs to permission from the API it appends the token to the AJAX request’s JSON used to communicate with the API, the token is then decrypted by the JWT library, also making use of the same private key, and if it checks, it lets the requested action take place, otherwise the access is denied. Also, the token contains information about the user’s role and by decrypting it also lets the API perform authorization operations by accessing that information. This method of authentication and authorization was chosen because it is mostly safe, not safe only if someone gets access to the token or the private key (highly unlikely) and allows for the creation of a stateless API.

#### 4.4.2 Registration (signup)

To access the solution’s platform a user must first registry in the platform system, this can be done by accessing the sign in page and providing the asked information by the registration form (described below). This information is related to the user and holds its data, the provided credentials (email and password) are then used to authenticate into the system. To register successfully the user must provide a registration key handed by the admin, otherwise it is impossible for it to register. The key is verified by the backend authentication module and if accompanied of valid registration info the user is persisted in the database and the key discarded so it cannot be used again.

- **Required registration data.**

**Table 11** – Registration Form description.

Field name	Description
Registration key	13-character key handed out by the admin.
First name	User’s first name.
Last name	User’s last name.
Email address	User’s email address (any provider can be used).
Password	User’s chosen password, needs to be bigger than 8 characters.

## Solution

Password confirmation	Needs to contain the same string as the Password field.
-----------------------	---

### 4.4.3 Login

After registration, for the user to access the actual platform contents it must login into the system. After inputting the login credentials and submitting them in the login page, the platform's backend authentication module verifies if they are valid by accessing the database and comparing the values with the persistent ones in the users table. If the verification succeeds the user is given a JWT that is then stored in browser cookies and enables the access of the user to the platform. To access some information, it is then required the sending of this token, that is always verified by the authentication module, for the user to be authenticated.

- **Required login data**

**Table 12** – Login Form description.

Field name	Description
Email address	Registered user email address
Password	Password associated with the email address.

## 4.5 Procedural generation

### 4.5.1 Challenges and story.

To create the challenges the admin must create both the story and the challenges. For that, the only tool required is a spreadsheet to create an XLS file (then saved as a CSV) for every chapter. These files are uploaded by the teacher to the platform and inserted in the *Backend/generation/chapters/* folder. On admin order, the folder with the .csv files for the chapters is scanned and every chapter file is parsed on the backend of the application. This module must account for the structure of the chapter file, and for every row, parse the content into the appropriate database table. To parse the chapter files into the database, we rely on PHP to read the content and, using the file structure described below, insert the parsed information into the correct database table with the first chapter and challenge unlocked to all users.

#### 4.5.1.1 Challenges and story file structure (for each chapter).

The challenges/story xls file structure is depicted in following table. Note that BOSS type challenges **must** come at the end of the Challenges section since they are the final challenges of a chapter and even though they are multiple rows, they are combined into only one challenge.

Solution

**Table 13** – Challenge/Story xls file structure.

<b>Chapter</b>	<i>CHAPTER NAME, COLOR1, COLOR2, COLOR3, TEXTCOLOR</i>
	<i>&lt;indication row&gt;</i>
<b>Challenges</b>	<i>CHALLENGE NUMBER, TYPE, ARGUMENT 1, ARGUMENT 2, ARGUMENT N</i>
	<i>&lt;empty row&gt;</i>
	<i>&lt;indication row&gt;</i>
<b>Dialogs</b>	<i>NUMBER OF QUOTE, TYPE, QUOTE, CHARACTER, BACKGROUND MEDIA</i>

- **Meaning of each element:**

**Chapter –**

**Table 14** – Chapter elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHAPTER NAME</i>	Name given to the challenge.
<i>COLOR1</i>	Primary color of the chapter color scheme.
<i>COLOR2</i>	Secondary color of the chapter color scheme.
<i>COLOR3</i>	Tertiary color of the chapter color scheme.
<i>TEXTCOLOR</i>	Text color of the chapter color scheme.

**Challenges –**

The meaning of each element differs on the *TYPE* of the challenge and is described in the next tables.

Note: It is possible to use images in the arguments of the challenges by inserting a variation of the string “Figure\_name:imagenam.png” where *imagenam.png* is the name of the image file in the *Backend/images/challenges/* folder.

**CHOICE / BOSS –**

**Table 15** – BOSS type challenge elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“CHOICE” or “BOSS”)
<i>ARGUMENT 1</i>	Question of the challenge.
<i>ARGUMENT 2</i>	Index of the correct answer.
<i>ARGUMENT [3-6]</i>	Possible answers to the question

Solution

### MCHOICE –

**Table 16** – MCHOICE type challenge elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“MCHOICE”)
<i>ARGUMENT 1</i>	Question of the challenge.
<i>ARGUMENT 2</i>	Indexes of the correct answer. Syntax: “index1/index2/indexN”
<i>ARGUMENT [3-...]</i>	Possible answers to the question

### FILL –

**Table 17** – FILL type challenge elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“FILL”)
<i>ARGUMENT 1</i>	Question of the challenge. Syntax: “( )” means it’s one of the filling spots.
<i>ARGUMENT 2</i>	Indexes of the correct words in order. Syntax: “index1/index2/indexN”.
<i>ARGUMENT [3-...]</i>	Possible word for the filling spots.

### CROSSWORD –

**Table 18** – CROSSWORD type challenge elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“CROSSWORD”)
<i>ARGUMENT [1-15]</i>	Hint and respective word of crossword. Syntax: “hint:word”

### LETTERS –

**Table 19** – LETTERS type challenge elements meaning.

<b>Element</b>	<b>Meaning</b>
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“LETTERS”)
<i>ARGUMENT 1</i>	Question of the challenge.
<i>ARGUMENT 2</i>	Correct answer.

Solution

### PAIR –

**Table 20** – PAIR type challenge elements meaning.

Element	Meaning
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“PAIR”)
<i>ARGUMENT [1-...]</i>	Correct pairs. Syntax: “leftN // rightN”

### GROUP –

**Table 21** – GROUP type challenge elements meaning.

Element	Meaning
<i>CHALLENGE NUMBER</i>	Number of the challenge in the chapter.
<i>TYPE</i>	Type of the challenge. (“GROUP”)
<i>ARGUMENT 1</i>	Left and right groups name. Syntax: “left // right”
<i>ARGUMENT [2-...]</i>	Questions of the challenge.

### Dialogs –

**Table 22** – Dialog lines element meaning.

Element	Meaning
NUMBER OF QUOTE	Number of the quote in the chapter.
TYPE	Type of the dialog. (“DIALOG”)
QUOTE	Line of dialog.
CHARACTER	Name of the character that speaks the quote.
BACKGROUND MEDIA	Name of the media file used for background in the dialog scene.

#### 4.5.1.2 Parsing the files into the database.

To parse the challenge/story files into the database it is used the *Backend/generation/reload\_challenges.php* that firstly cleans the database’s tables ‘chapter’, ‘challenge’, ‘argument’ and ‘dialog’. After it scans the *Backend/generation/chapters/* directory, an challenge id counter is initiated, as well as a chapter id counter and for every chapter file that is found it is started a parsing instruction that analyses every line of the file separating the line string into the elements, and then acting accordingly.

## Solution

If it is the 1<sup>st</sup> line of the file, it will analyze it as a chapter information row and insert every element into the respective ‘chapter’ table column by using the element meaning described above and the chapter id counter as the id.

The second line of the file is skipped since it is only a table header used to help the admin fill the file correctly.

The following rows until it encounters an empty line are challenge rows. The first 2 elements are challenge info (not arguments), so they are inserted into the respective ‘challenge’ table columns using the meaning described above, in this case, challenge number and challenge type and using the value of the challenge id counter as the id. The remaining elements are treated as arguments of the respective challenge and inserted into the ‘argument’ table, being their number the ‘number’ column and their content the ‘content’ column, also they are associated with the current challenge by using the value of the challenge id counter. For every row the challenge id counter value is incremented.

The empty line is skipped and the one after is also skipped since it is also used to help the admin fill the file correctly.

The following rows until the end of the file are dialog rows. The elements are inserted into the ‘dialog’ table respective columns by using the meaning described above and are associated with the respective chapter using the chapter id counter value.

The chapter id counter value is incremented, and it proceeds to analyze the next file in the directory if there’s any.

### 4.5.2 Atlas.

To create the atlas the admin must create the atlas. And like the challenges and story, the only tool required is a spreadsheet to create an XLS file (then saved as a CSV) for every chapter. These files are uploaded by the teacher to the platform and inserted in the *Backend/generation/atlas/* folder. On admin order, the folder with the .csv files for the chapters is scanned and every atlas chapter file is parsed on the backend of the application. The insertion mode is the same used in the challenges / story, but instead the PHP must account for the following file structure.

#### 4.5.2.1 Atlas file structure (for each chapter).

The atlas xls file structure is depicted in following table.

**Table 23** – Atlas xls file structure.

	<i>&lt;indication row&gt;</i>
<b>Atlas items</b>	<i>IMAGE NAME, TEXT</i>



Solution

- **Meaning of each element.**

**Table 24** – Atlas items element meaning.

<b>Element</b>	<b>Meaning</b>
<i>IMAGE NAME</i>	Name of the image file of the atlas item.
<i>TEXT</i>	Text explaining the image of the atlas item.

#### **4.5.2.2 Parsing into the database.**

To parse the atlas files into the database it is used the *Backend/generation/reload\_atlas.php* that firstly cleans the ‘atlas’ database table. After it scans the *Backend/generation/atlas/* directory, a chapter id counter is initialized and for every chapter file that is found it is started a parsing instruction that analyses every line of the file separating the line string into the elements, and then acting accordingly.

The first line is skipped since it is used to help the admin fill the file correctly.

The following lines until the end of the file are analyzed as atlas items rows and the elements are inserted into the ‘atlas’ table columns by making use of the element meaning described above, also every atlas item is associated with the respective chapter using the chapter id counter value.

The chapter id counter value is incremented, and it proceeds to analyze the next file in the directory if there’s any.

## **4.6 API**

To stablish communication between the frontend and backend modules it was developed a simple PHP stateless API. The API is accessed via AJAX requests present in the frontend files and its responses analyzed via JavaScript. The requests as well as the responses are JSON encoded strings. The following information describes the API, a more technical description is made in Attachment 1.

- **Authenticating in the API.**

As described previously in the authentication/authorization section, the authentication in the API is made via the sending of a valid JWT in the request made.

- **Available API actions.**

## Solution

The API provides access to the retrieve of user, chapter, story and atlas information, as well as the updating of said data. Also allows the verification of user answers to the challenges and the execution of admin actions like generating the platform content and managing it. A list with the provided API operations and a small description is supplied next:

- Clear database – clears the whole database content, only used by an admin user.
- Generate registration key – generates a new registration key, only used by an admin user.
- Retrieve atlas item – retrieves a given atlas item data.
- Retrieve BOSS challenge arguments – retrieves all BOSS type challenges data related to a given chapter.
- Retrieve challenge arguments (not BOSS type) – retrieves a given challenge's data, the challenge can't be BOSS type.
- Retrieve Atlas items – retrieves every atlas item of a given chapter.
- Retrieve user challenges information – retrieves the data regarding the progress of a given user in the challenges.
- Retrieve story dialogs – retrieves the data of every dialog moment of a given chapter.
- Retrieve chapters – retrieves the info of every chapter.
- Retrieve leaderboard users – retrieves info about the 10 users with the most points.
- Retrieve every registration key – retrieves every registration key that can still be used, only used by an admin user.
- Retrieve if first time playing the chapter challenges – retrieves if it is the first time a user plays a given chapter.
- Retrieve user info – retrieves the info of a given user.
- Retrieve every user info – retrieves the info of every registered user.
- User authentication – authenticates a registered user, via JWT.
- Reset every user's challenge progress – resets every user's progress in the challenges (locks them all but the first).
- User registration – registers a user if the provided registration key is correct.
- Lock/unlock chapter – locks or unlocks a given chapter.

## Solution

- Verify BOSS challenge answer – verifies the answer of a given BOSS type challenge and retrieves its correct answer.
- Verify CHOICE challenge answer - verifies the answer of a given CHOICE type challenge and retrieves its correct answer.
- Verify CROSSWORD challenge answer - verifies the answer of a given CROSSWORD type challenge and retrieves its correct answer.
- Verify FILL challenge answer - verifies the answer of a given FILL type challenge and retrieves its correct answer.
- Verify GROUP challenge answer - verifies the answer of a given GROUP type challenge and retrieves its correct answer.
- Verify LETTERS challenge answer - verifies the answer of a given LETTERS type challenge and retrieves its correct answer.
- Verify MCHOICE challenge answer - verifies the answer of a given MCHOICE type challenge and retrieves its correct answer.
- Verify PAIR challenge answers - verifies the answer of a given PAIR type challenge and retrieves its correct answer.
- Verify JWT – verifies if a JWT is valid.
- Verify admin JWT – verifies if a JWT is valid and of an admin.

## 4.7 User Interface.

The user interface was entirely made with the use of html elements, css and javascript and can be used in any type of screen (desktop or mobile). The chosen design was applied in order to provide a simple, direct and easy navigation. The navigation is made mainly using buttons that let the user change views or select elements. The user must always enter via the index page and navigate by using the buttons otherwise the views cannot be accessed, this was made so the platform would feel more like a game or an APP.

### 4.7.1 Registration View.

*Frontend/signup.html*

In this view it is shown to the user a registration form, as well as an image that represents the involved entities in the project. In order to register the user must fill the registration form and click the “Signup” button. After, one or more messages appear in the screen reporting the status of the registration.

## Solution

To change to the “Login” view the user must click the “Login here!” anchor that is shown on the screen.

### Javascript-

**Table 25** – Registration View javascript description.

Event	Action
<b>Signup button click</b>	<p>Whenever the “Signup” button is clicked a verification to the registration form fields is performed and it is made sure that every field is correctly filled, i.e:</p> <ul style="list-style-type: none"><li>• <i>Registration Key</i>: has 13 characters.</li><li>• <i>First Name</i>: is filled.</li><li>• <i>Last Name</i>: is filled.</li><li>• <i>Email Address</i>: has a correct syntax.</li><li>• <i>Password</i>: has 8 characters.</li><li>• <i>Confirm Password</i>: is the same as the password field.</li></ul> <p>If a field is not correctly filled the user is shown a message reporting the error.</p> <p>If everything checks these conditions an AJAX post request is made to the <i>Backend/API/signup.php</i>. If the returned response is successful, the user is now registered, and it is shown a “You can now login!” message. Otherwise, the registration fails, and an error message is shown.</p>
<b>“Login here!” anchor click</b>	Login View is loaded.

## Solution

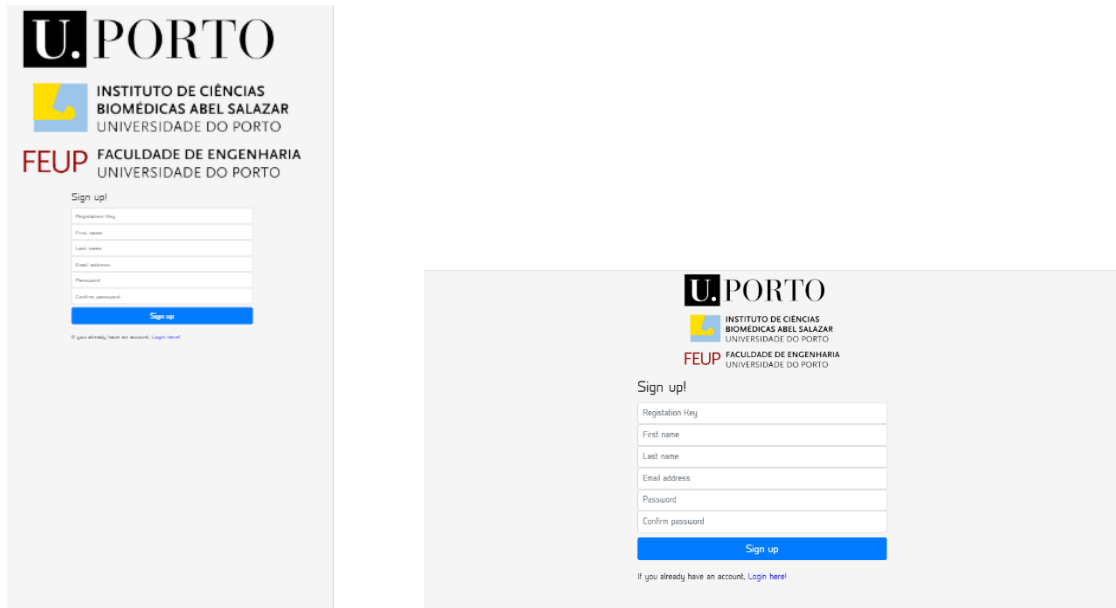


Figure 3 - Mobile / Desktop Registration View Interface.

### 4.7.2 Login View.

*Frontend/login.html*

In this view it is shown to the user a login form, as well as an image that represents the involved entities in the project. In order to login the user must fill the login form correctly and click the “Sign in” button. If the user successfully logs in, it gains access to the platform’s contents and the Main Menu view is shown.

To change to the “Registration” view the user must click the “Sign up here!” anchor that is shown on the screen.

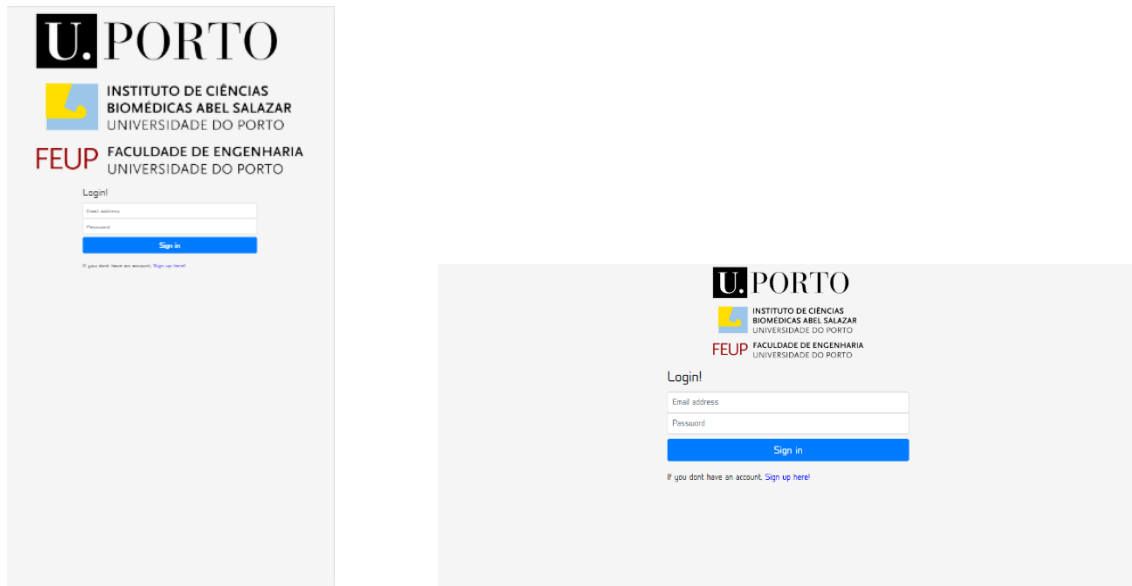
**Javascript-**

Table 26 – Login View javascript description.

Event	Action
<b>Sign in button click</b>	Whenever the “Sign in” button is clicked an AJAX post request is made to the <i>Backend/API/login.php</i> . If the request is successful, the user is redirected to the Main Menu view, and it is shown a “You can now login!” message. Otherwise, the login fails, and an error message is shown.

## Solution

<b>“Sign up here!” anchor click</b>	Registration view is loaded.
-------------------------------------	------------------------------



**Figure 4 - Mobile / Desktop Login View Interface.**

### 4.7.3 Main Menu View.

*Frontend/main\_menu.html*

In this view it is shown, at the top, an header that holds the authenticated user information (name and points), as well as a “logout” button (if the user is an admin it also holds an Admin button, that takes the user to an admin restricted area). The “logout” button is used whenever the user wants to logout.

Also, there are two big main buttons, “Adventure” and “Atlas”, the first one takes the user to a chapter selection screen related to the challenges, whereas the second one takes it to a chapter selection screen related to the atlas.

In left down corner of the screen exists a “trophy” button, this button, if clicked, takes the user to the leaderboard view.

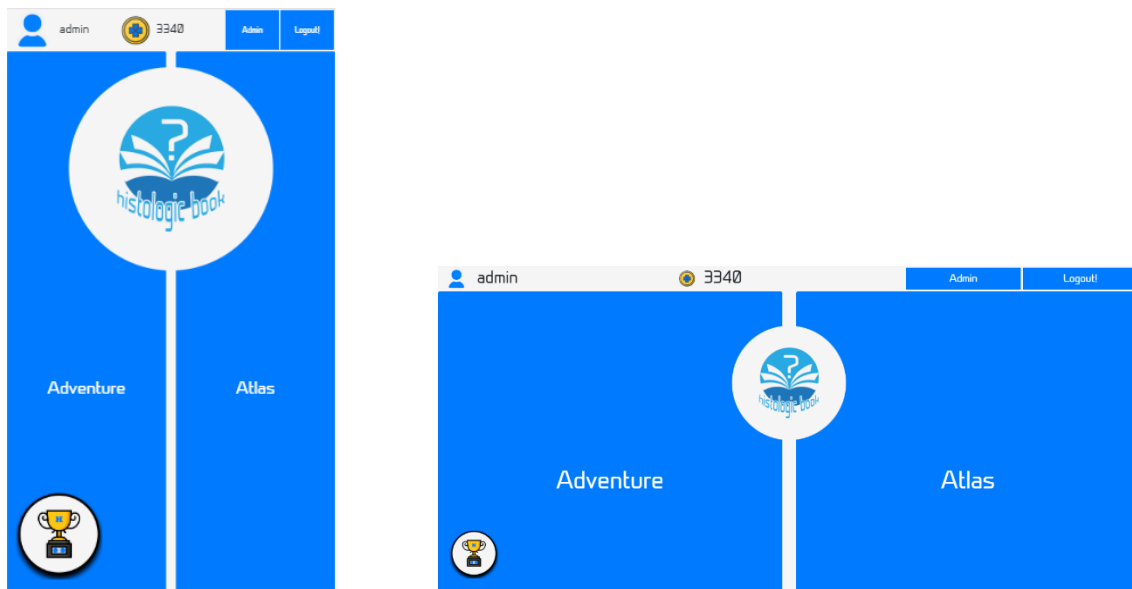
#### Javascript-

**Table 27 – Main Menu View javascript description.**

Event	Action
-------	--------

## Solution

<b>On document load</b>	Whenever the document loads, an AJAX post request is made to the <i>Backend/API/verify_token_admin.php</i> , if the request is successful the previously mentioned “admin” button is added to the header.  After, an AJAX post request is made to the <i>Backend/API/get_user_info.php</i> , if the request is successful the user information (name and points) is shown in the header.
<b>Adventure Button Click</b>	Chapter Selection (challenges related) view is loaded.
<b>Atlas Button Click</b>	Chapter Selection (atlas related) view is loaded.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Trophy Button Click</b>	Leaderboard view is loaded.
<b>Admin Button Click</b>	Admin Dashboard view is loaded.



**Figure 5** - Mobile / Desktop Main Menu View Interface.

### 4.7.4 Leaderboard View.

*Frontend/leaderboard.html*

In this view it is shown, at the top, a header that holds the authenticated user information (name and points), a “back” button and a “logout” button. The “back” button is used to return to the Main Menu view and the “logout” button to logout the user.

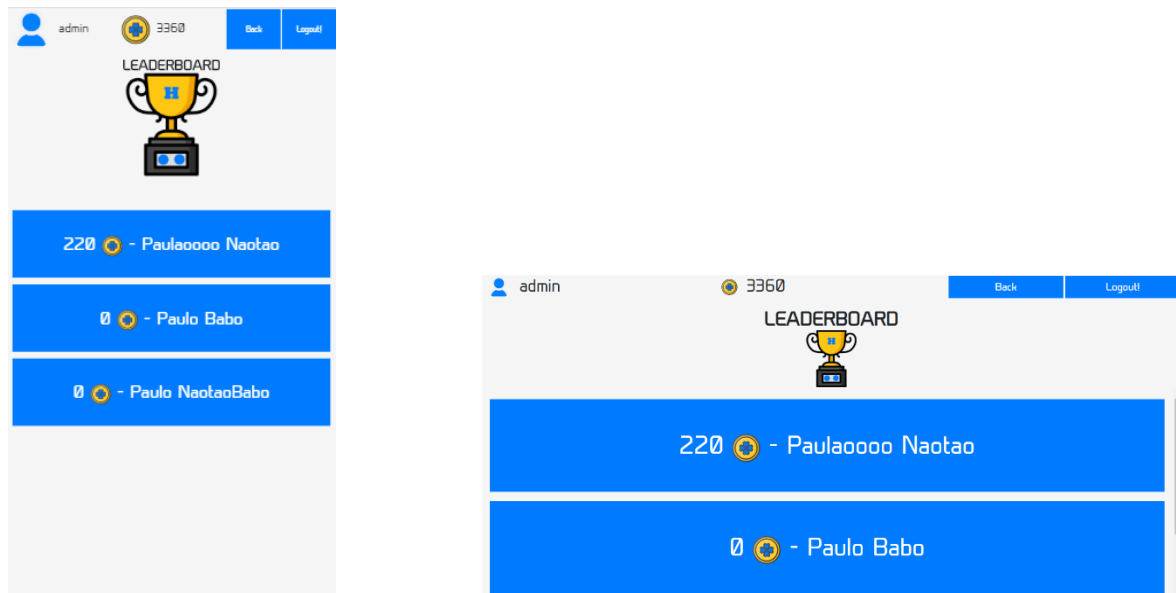
## Solution

The rest of the view is composed with a leaderboard with the user's name of the 10 users with more points

### Javascript –

**Table 28** – Leaderboard View javascript description.

Event	Action
<b>On Document Load</b>	Whenever the document loads, an AJAX post request is made to the <i>Backend/API/get_user_info.php</i> , if the request is successful the user information (name and points) is shown in the header.  After, an AJAX post request is made to the <i>Backend/API/get_leaderboard.php</i> , if the request is successful the name and points of the 10 registered users with more points is shown.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Back Button Click</b>	Main Menu view is loaded.



**Figure 6** - Mobile / Desktop Leaderboard View Interface.

## 4.7.5 Chapter Selection View (challenges).

*Frontend/choose\_chapter.html*



## Solution

In this view it is shown, at the top, a header that holds the authenticated user information (name and points), a “back” button and a “logout” button. The “back” button is used to return to the Main Menu view and the “logout” button to logout the user.

The rest of the screen is filled with a button for each chapter, with the respective chapter’s name in it. If clicked, these buttons take the user to the Challenge Selection view.

### Javascript –

Table 29 – Chapter Selection (Challenges) View javascript description.

Event	Action
<b>On Document Load</b>	Whenever the document loads, an AJAX post request is made to the <i>Backend/API/get_user_info.php</i> , if the request is successful the user information (name and points) is shown in the header.  After, an AJAX post request is made to the <i>Backend/API/get_chapters_info.php</i> , if the request is successful the buttons for each chapter are created and decorated with the chapter’s color scheme, if the chapter is locked a lock icon is shown next to name. If the request fails an error message is shown.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Back Button Click</b>	Main Menu view is loaded.
<b>Chapter Button Click</b>	Challenge Selection view of the clicked chapter is loaded, and the selected chapter id and color scheme colors saved in variables.

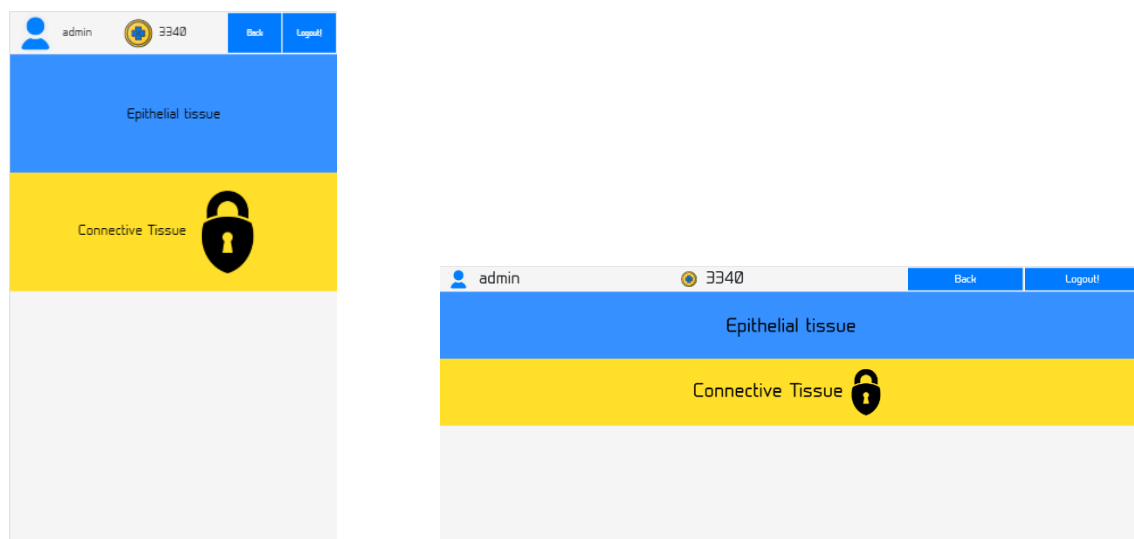


Figure 7 - Mobile / Desktop Chapter Selection (Challenges) View Interface.

Solution

### 4.7.6 Chapter Selection View (Atlas).

*Frontend/choose\_atlas\_chapter.html*

In this view it is shown, at the top, a header that holds the authenticated user information (name and points), a “back” button and a “logout” button. The “back” button is used to return to the Main Menu view and the “logout” button to logout the user.

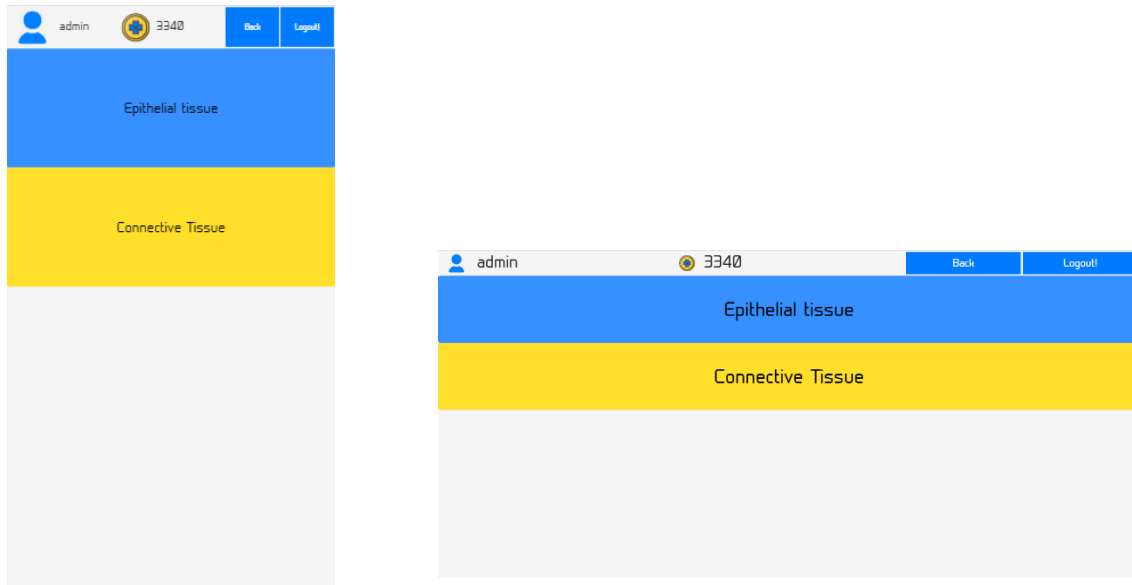
The rest of the screen is filled with a button for each chapter, with the respective chapter’s name in it. If clicked, these buttons take the user to the Atlas Item Selection view.

#### Javascript –

**Table 30** – Chapter Selection (Atlas) View javascript description.

Event	Action
<b>On Document Load</b>	Whenever the document loads, an AJAX post request is made to the <i>Backend/API/get_user_info.php</i> , if the request is successful the user information (name and points) is shown in the header.  After, an AJAX post request is made to the <i>Backend/API/get_chapters_info.php</i> , if the request is successful the buttons for each chapter are created and decorated with the chapter’s color scheme. If the request fails an error message is shown.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Back Button Click</b>	Main Menu view is loaded.
<b>Chapter Button Click</b>	Atlas Item Selection view of the clicked chapter is loaded, and the selected chapter id saved in a variable.

## Solution



**Figure 8** - Mobile / Desktop Chapter Selection (Atlas) View Interface.

### 4.7.7 Storyboard View.

*Frontend/dialog.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Chapter Selection view.

The rest of the screen is dialog area, with the media chosen to the background of the dialog moment, in the back and the actual dialog on top of it. Whenever a character “speaks” it is show is line of dialog, a portrait of the character and a “Next” button that advances for the next dialog moment, if there’s any.

At the top right corner, there is a “skip” button that skips the whole story moment and loads the Challenge Selection View.

**Javascript –**

**Table 31** – Storyboard View javascript description.

Event	Action
<b>On Document Load</b>	On document load, an AJAX post request is made to Backend/API/get_chapter_dialog.php, if the request is successful the dialog info for the given chapter is saved in an array and the

## Solution

	first dialog moment is shown, otherwise an error message is displayed.
<b>Back Button Click</b>	Chapter Selection View is loaded
<b>Skip Button Click</b>	Challenge Selection View is loaded.
<b>Next Button Click</b>	The next dialog moment is shown, if it is the last one the of the story moment happens.
<b>End of the story moment</b>	Challenge Selection View is loaded.

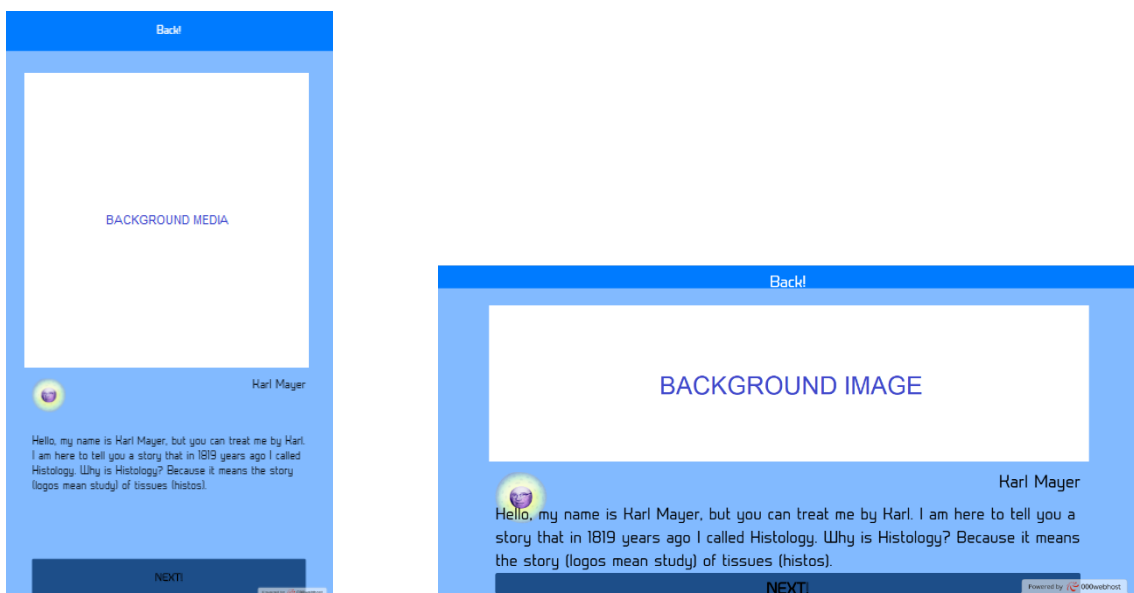


Figure 9 - Mobile/Desktop Storyboard View user interface.

### 4.7.8 Challenge Selection View.

*Frontend/choose\_challenge.html*

In this view it is shown, at the top, a header that holds the authenticated user information (name and points), a “back” button and a “logout” button. The “back” button is used to return to the Chapter Selection view and the “logout” button to logout the user.

The rest of the screen is filled with a button for each challenge, with the respective challenge’s number in it. If clicked, these buttons take the user to the actual challenge view.

**Javascript –**

**Table 32 –** Challenge Selection View javascript description.

## Solution

Event	Action
<b>On Document Load</b>	<p>Whenever the document loads, an AJAX post request is made to the <i>Backend/API/get_user_info.php</i>, if the request is successful the user information (name and points) is shown in the header.</p> <p>After, an AJAX post request is made to the <i>Backend/API/get_chapter_challenges_info.php</i>, if the request is successful the buttons for each challenge are created and decorated with the chapter's color scheme, if a challenge is locked for the authenticated user, a lock icon is shown next to its number, and if it is complete a right icon is shown instead. If the request fails an error message is shown.</p> <p>Also, the chapter's color scheme is applied to the view.</p>
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Back Button Click</b>	Chapter Selection view is loaded.
<b>Challenge Button Click</b>	Challenge view is loaded.



Figure 10 - Mobile / Desktop Challenge Selection View Interface.

### 4.7.9 Challenge View (CHOICE).

*Frontend/CHOICE.html*

## Solution

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that comes the challenge question, the available answers buttons and a “Check” button.

In order to answer the question, the user must click the desired answer button and click the “check” button. When the “check” button is clicked a modal will appear with the result of the challenge, if the user fails, the correct answer is displayed.

### Javascript –

**Table 33** – CHOICE type Challenge View javascript description.

<b>Event</b>	<b>Action</b>
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_Arguments.php</i> , if the request is successful, the challenge’s question is displayed, the answers buttons created, and the chapter’s color scheme applied. If the request fails an error message is shown.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Answer Button Click</b>	Saves the index of the clicked answers as the selected answer.
<b>Check Button Click</b>	If an answer is selected, a modal is open and an AJAX post request is made to the <i>Backend/API/verify_CHOICE.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer.

## Solution

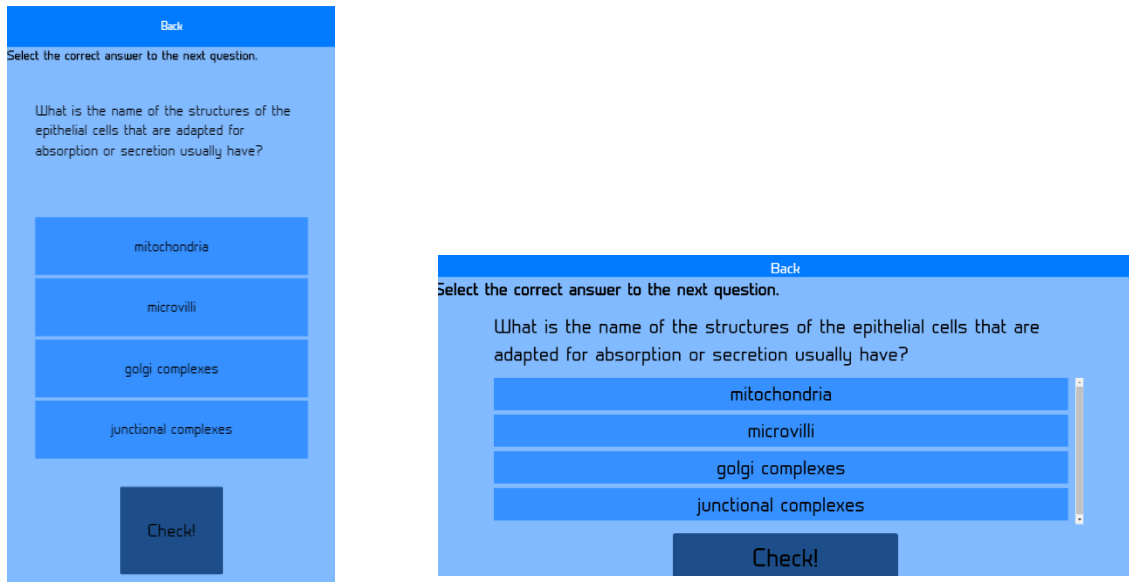


Figure 11 - Mobile / Desktop CHOICE type Challenge View Interface.

### 4.7.10 Challenge View (MCHOICE)

*Frontend/MCHOICE.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that comes the challenge question, the available answers buttons and a “Check” button.

In order to answer the question, the user must click the desired answers button and click the “check” button. When the “check” button is clicked a modal will appear with the result of the challenge, if the user fails, the correct answer is displayed.

**Javascript –**

**Table 34 –** MCHOICE type Challenge View javascript description.

Event	Action
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_Arguments.php</i> , if the request is successful, the challenge’s question is displayed, the answers

## Solution

	buttons created, and the chapter’s color scheme applied. If the request fails an error message is shown.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Answer Button Click</b>	Adds the answer index to an array of selected answers. If that answer is already selected it deselects it and removes the index from the array.
<b>Check Button Click</b>	If an answer is selected, a modal is open and an AJAX post request is made to the <i>Backend/API/verify_MCHOICE.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer.



Figure 12 – Mobile/Desktop MCHOICE type Challenge View user interface.

### 4.7.11 Challenge View (CROSSWORD).

*Frontend/CROSSWORD.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that an empty crossword is shown, as well as a “check” button.



## Solution

In order to complete the challenge, the user must complete the crossword and then click the “check button”. When the “check” button is clicked a modal will appear with the result of the challenge, if the user fails, the correct answer is available for the user to see.

If a word’s number button is clicked, the hint for the given word is shown at the top of the challenge area.

### Javascript –

**Table 35** – CROSSWORD type Challenge View javascript description.

<b>Event</b>	<b>Action</b>
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_Arguments.php</i> , if the request is successful, the challenge arguments in the response are treated and a matrix for the crossword is generated using a developed crossword generation algorithm, and the words mapped to the respective hints. The matrix is then represented visually in the challenge area and the chapter’s color scheme applied.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Word Number Button clicked</b>	Hint for the respective word is shown.
<b>Check Button Click</b>	A modal is open, and an AJAX post request is made to the <i>Backend/API/verify_CROSSWORD.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, the crossword is filled with the correct answer, depending on the correctness of the words in the user’s answer, the Word Number Buttons are colored green (correct) or red (incorrect), and the model is filled with a incomplection message. It is then possible for the user to close the modal in order to check the correct answer.

## Solution

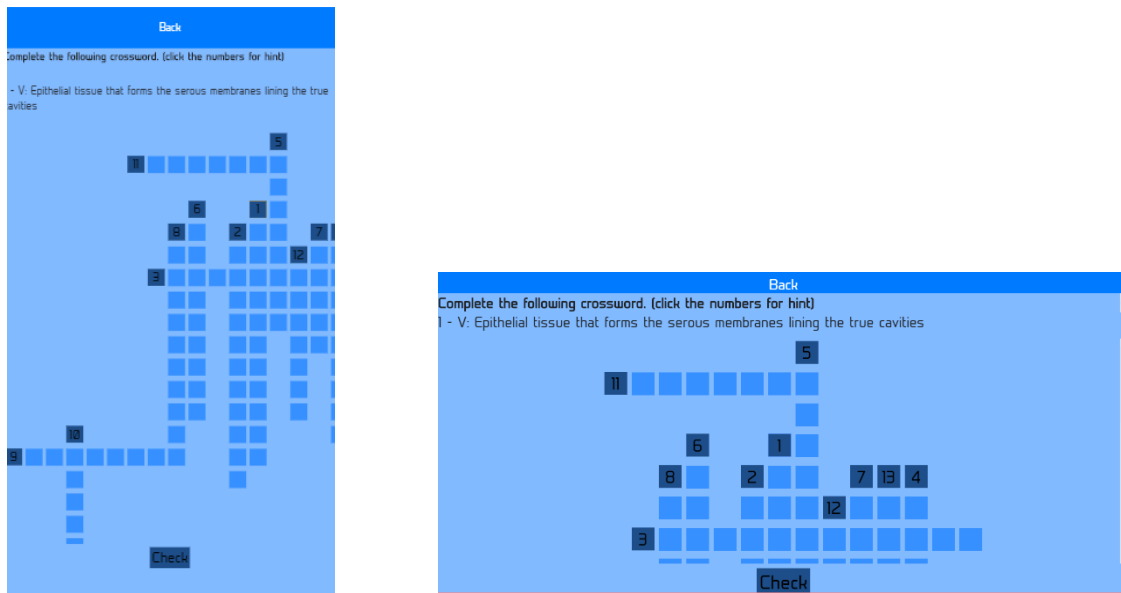


Figure 13 - Mobile / Desktop CROSSWORD type Challenge View Interface.

### 4.7.12 Challenge View (FILL).

*Frontend/FILL.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that the text with the space buttons, possible answer buttons and the “check” button.

In order to answer the challenge, the user must fill the space buttons by clicking the possible answer button and then pressing the “check” button. When the “check” button is clicked a modal will appear with the result of the challenge, if the user fails, the correct answer is displayed.

**Javascript –**

Table 36 – FILL type Challenge View javascript description.

Event	Action
On Document Load	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_arguments.php</i> , if the request is successful, the question is created by swapping every occurrence

## Solution

	of a “ ( ) ” with a space button, and the options buttons are created. The chapter’s color scheme is applied.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Space Button Click</b>	If filled with an option string, deselects the string’s correspondent option button. Also removes the string from the button and the option index from the array of selected options.
<b>Option Button Click</b>	Fills the first empty Space Button with the option string, and the option button is marked as selected. Also adds the option index to an array of selected options.
<b>Check Button Click</b>	A modal is open, and an AJAX post request is made to the <i>Backend/API/verify_FILL.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer in order.

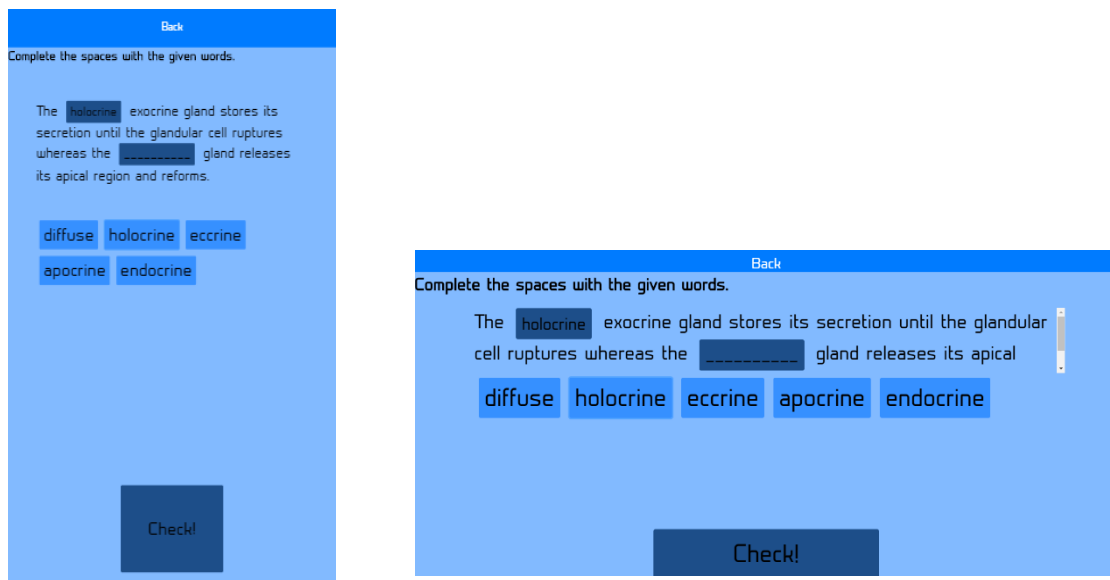


Figure 14 - Mobile / Desktop FILL type Challenge View Interface.

### 4.7.13 Challenge View (LETTERS)

*Frontend/FILL.html*

## Solution

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that, the question, buttons representing the answer’s letters (answer buttons), and the possible letters buttons.

In order to answer the challenge, the user must fill the answer buttons by clicking the possible characters buttons and then pressing the “check” button. When the “check” button is clicked a modal will appear with the result of the challenge, if the user fails, the correct answer is displayed.

### Javascript –

**Table 37** – LETTERS type Challenge View javascript description.

<b>Event</b>	<b>Action</b>
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_arguments.php</i> , if the request is successful, the question is displayed, the answer buttons created, and the possible letters buttons too. These last buttons are a combination of the answer’s letters and random letters. Then, the chapter’s color scheme is applied.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Answer Button Click</b>	If filled with a letter, is filled with a blank and deselects the Letter Button with the id associated with that position. Also removes the letter from the answer array and button id from the association array.
<b>Letter Button Click</b>	Fills the first empty Answer Button with the correspondent letter, and the button is marked as selected. Also adds the letter to an array in the corresponding position and does the same with the button id.
<b>Check Button Click</b>	A modal is open, and an AJAX post request is made to the <i>Backend/API/verify_LETTERS.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer.

## Solution



Figure 15 - Mobile / Desktop LETTERS type Challenge View Interface.

### 4.7.14 Challenge View (PAIR)

*Frontend/PAIR.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge, after that, an area that holds a left and a right group of options, an area that holds the pairs already made and the “check” button.

In order to answer the challenge, the user must make connections between items of the left and right group and press the “check” button.

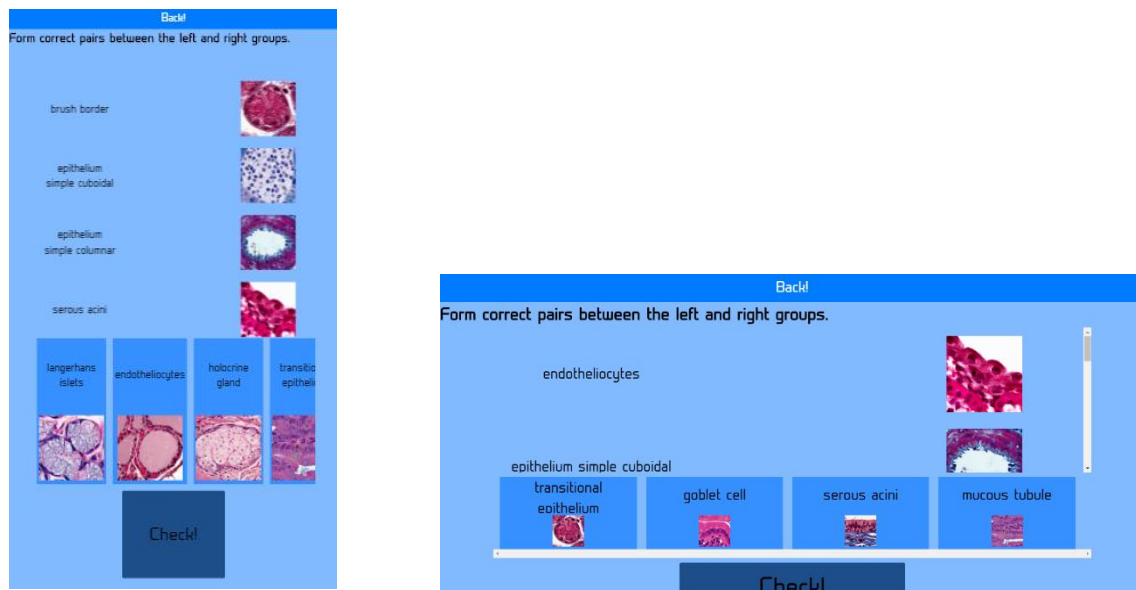
**Javascript –**

Table 38 – PAIR type Challenge View javascript description.

Event	Action
On Document Load	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_arguments.php</i> , if the request is successful, the left and right group items are created. Then, the chapter’s color scheme is applied.

## Solution

<b>Item Button Click</b>	If no other item or an item from the same group is selected, selects the item. If an item from a different group is selected, removes the items from the respective groups, and creates a pair in the pair's holder.
<b>Pair Button Click</b>	The pair is destroyed, and the composing items are returned to the respective groups.
<b>Check Button Click</b>	A modal is open, and an AJAX post request is made to the <i>Backend/API/verify_LETTERS.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer.



**Figure 16** - Mobile / Desktop PAIR type Challenge View Interface.

### 4.7.15 Challenge View (GROUP)

*Frontend/GROUP.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

## Solution

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge, after that, in a first instance a “start” button and then 2 vertical buttons identifying each group.

When the start button is pressed a phrase is shown and a timer next to it starts counting.

The challenge advances for the next phrase (if there are more) if the user chooses a group or the timer reaches 0.

If there are no more phrases left, the challenge ends and the results are displayed.

### Javascript –

**Table 39** – GROUP type Challenge View javascript description.

<b>Event</b>	<b>Action</b>
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_challenge_arguments.php</i> , if the request is successful, the left and right groups are filled with their correspondent name and the challenge phrases saved in an array.
<b>Start Button Click</b>	Phrase is displayed and timer is shown.
<b>Group Button Click</b>	The correct group of the phrase is shown and if there are more, advances for the next phrase. Saves the selected phrase into the correspondent clicked group array.
<b>Timer reaches 0</b>	The correct group of the phrase is shown and if there are more, advances for the next phrase. Saves the selected phrase into the wrong group array.
<b>Game End</b>	A modal is open, and an AJAX post request is made to the <i>Backend/API/verify_GROUP.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the user’s correct answers and wrong answers (properly indicated).

## Solution

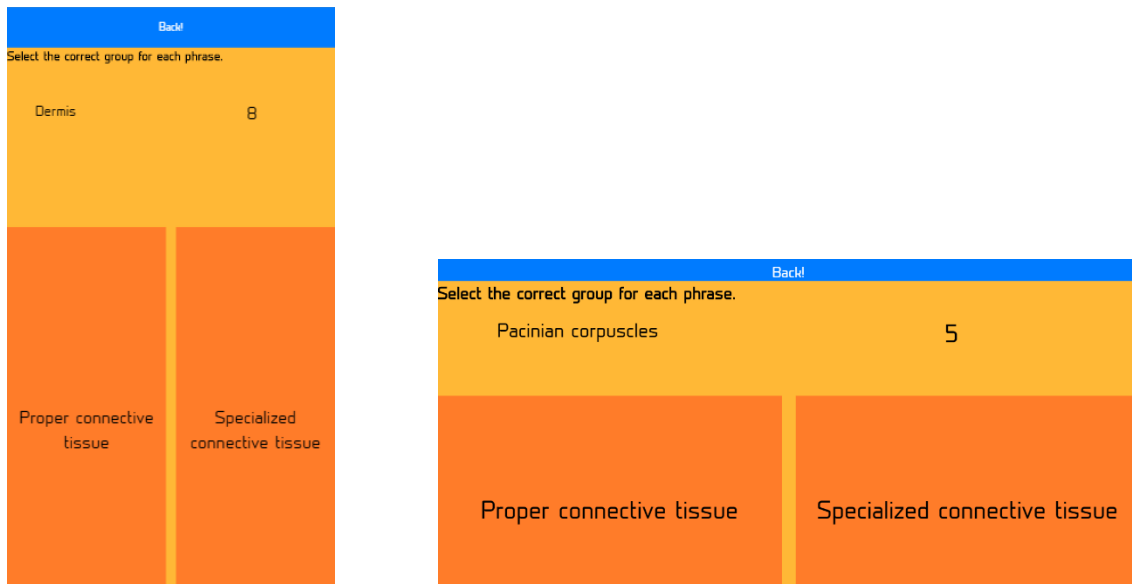


Figure 17 - Mobile / Desktop GROUP type Challenge View Interface.

### 4.7.16 Challenge View (BOSS)

*Frontend/BOSS.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Challenge Selection view.

The rest of the screen is challenge area, and at the top there is an indication of what to do in order to complete challenge. After that comes the challenge question, the available answers buttons and a “Check” button.

In order to answer the question, the user must click the desired answer button, and if it is a correct answer, the next question is shown as well as its possible answers. If the user fails, the correct answer to the given question is shown and a modal appears with an incompleteness message.

When the last question is answered correctly the user completes the challenge, unlocking the first one of the next chapter.

**Javascript –**

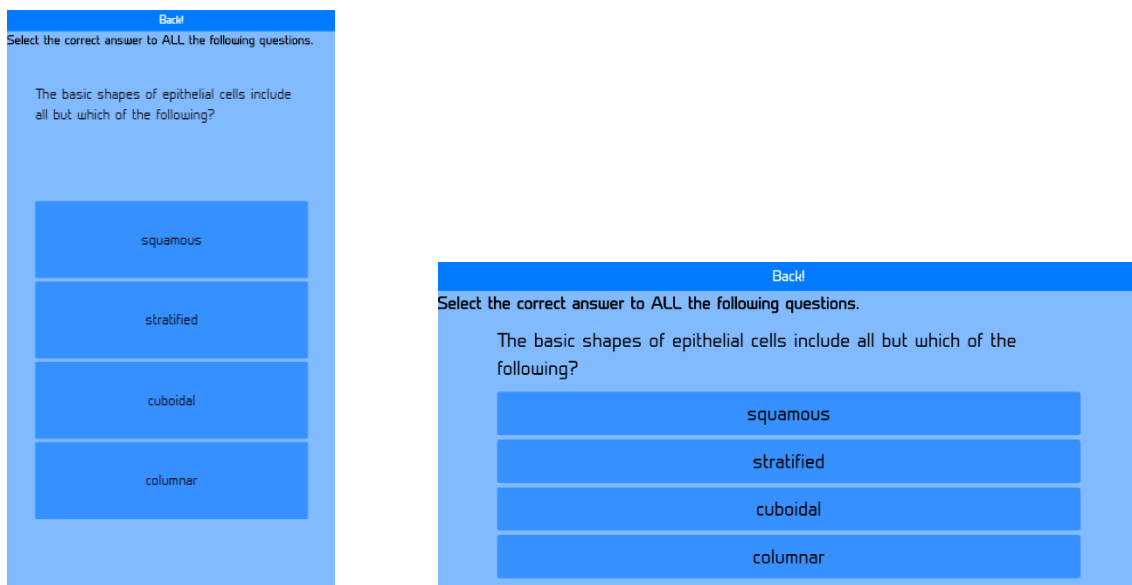
Table 40 – BOSS type Challenge View javascript description.

Event	Action
On Document Load	When the document loads, an AJAX post request is made to the <i>Backend/API/get_BOSS_arguments.php</i> , if the request is



## Solution

	successful, the response is parsed and the arrays containing every question info are saved, and the first question and its respective possible answers are shown.
<b>Back Button Click</b>	Challenge Selection view of the chapter is loaded.
<b>Answer Button Click</b>	If the clicked answer was the correct one it advances for the next question until there are no more left, and the challenge is complete, else, it shows the correct answer and a modal is open with a incomplection message.
<b>On challenge completion</b>	A modal is open and an AJAX post request is made to the <i>Backend/API/verify_BOSS.php</i> , if the request is successful, it verifies the response contents to check if the answer was the correct one. If so, the modal is filled with a congratulations message and the won points, otherwise, it is filled with the correct answer.



**Figure 18** - Mobile / Desktop BOSS type Challenge View Interface.

### 4.7.17 Atlas Item Selection View.

*Frontend/atlas.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Chapter Selection View (Atlas).

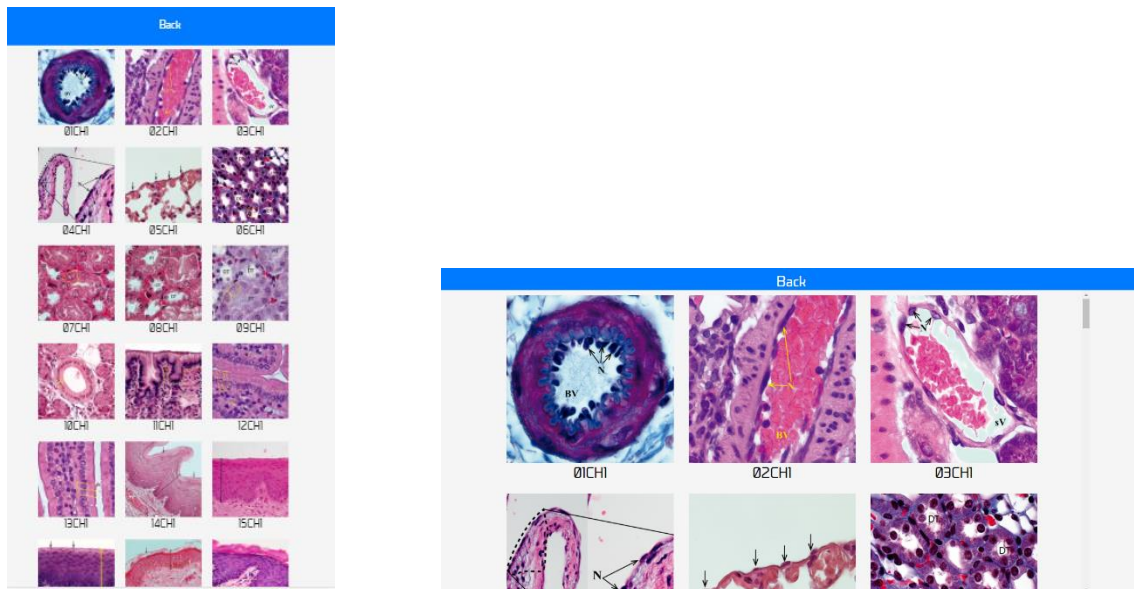
## Solution

The rest of the screen is filled with the atlas items (image and name), to amplify an item and view its subtitle the user must click the item button.

### Javascript –

**Table 41** – Atlas Item Selection View javascript description.

Event	Action
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_chapter_atlas.php</i> , if the request is successful, the atlas items are shown. They are displayed one after the previous one loads, in order to increase performance.
<b>Back Button Click</b>	Chapter Selection View (Atlas) is loaded.
<b>Atlas Button Click</b>	The Amplified Atlas View of the clicked item is loaded, also the actual amplified item id is saved in a variable.



**Figure 19** - Mobile / Desktop Atlas Item Selection View Interface.

### 4.7.18 Amplified Atlas Item View.

*Frontend/amplified\_atlas.html*

In this view it is shown, at the top, a “Back” button that is used to return to the Chapter Selection View (Atlas).

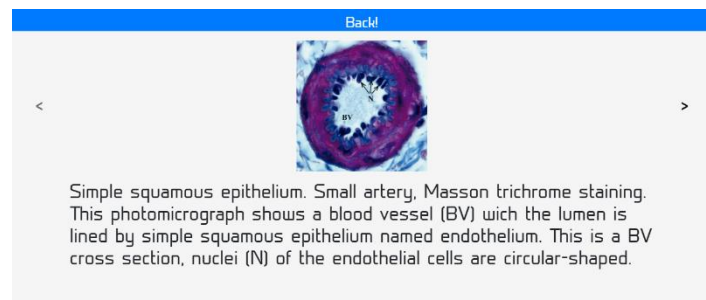
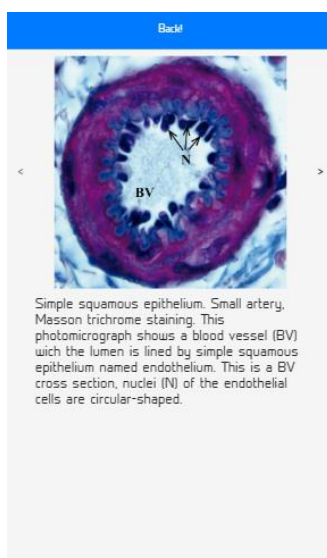
## Solution

The rest of the screen is filled with the amplified item info, this is, image and subtitle. The user can navigate to the next and the previous item (if they exist) by clicking the arrow buttons in the sides of the image.

### Javascript –

**Table 42** – Amplified Atlas Item View javascript description.

Event	Action
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_atlas_item.php</i> , if the request is successful, the atlas item image and subtitle are displayed.
<b>Back Button Click</b>	Atlas Item Selection View is loaded.
<b>“next” Button Click</b>	The actual item id is incremented, and an AJAX post request is made to the <i>Backend/API/get_atlas_item.php</i> , if the request is successful, the atlas item image and subtitle are displayed.  This button is disabled if there are no next Atlas item in the given chapter.
<b>“previous” Button Click</b>	The actual item id is decremented, and an AJAX post request is made to the <i>Backend/API/get_atlas_item.php</i> , if the request is successful, the atlas item image and subtitle are displayed.  This button is disabled if there are no previous Atlas item in the given chapter.



**Figure 20** - Mobile / Desktop Amplified Atlas Item View Interface.

### 4.7.19 Admin Area (Users) View

*Frontend/admin\_users.html*

In this area the admin can manage the user individually as well as in group. Also, it is in here that the registration keys are shown and generated.

There is at the top, a header that holds the authenticated admin information (name), a “game” button and a “logout” button. The “game” button is used to return to the Main Menu view and the “logout” button to logout the admin.

The rest of this view is composed by a user’s data table, showing their information followed by a registration key area, where it is possible to view and generate the keys.

Also, there is an action button that resets every user’s progress in the challenges.

To navigate between admin area views there is a side bar to the left, with the possible sections.

#### Javascript –

**Table 43** – Admin Area (Users) View javascript description.

Event	Action
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_users_info.php</i> , if the request is successful, the users info contained in the response is displayed in the user’s data table.  There is also made an AJAX post request to the <i>Backend/API/get_registration_keys</i> , if the request is successful, the registration keys info contained in the response is displayed in the registration key’s table.
<b>Game Button Click</b>	Main Menu View is loaded.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Sidebar Chapter/Challenges section Click</b>	Admin Area (Chapter/Challenges) View is loaded.

## Solution

<p><b>Generate Keys Button</b></p>	<p>Generates the number of desired keys by making that number of AJAX post requests to the <i>Backend/API/generate_registration_key.php</i>, if a request is successful and not every key was yet generated it is sent another request. Whenever the desired number of keys was generated or there is an unsuccessful request, the generation process stops and it is displayed a status message.</p>
<p><b>Reset Users Progress Button</b></p>	<p>Makes an AJAX post request to the <i>Backend/API/reset_users_progress.php</i>, a message is then shown with the status of the request.</p>

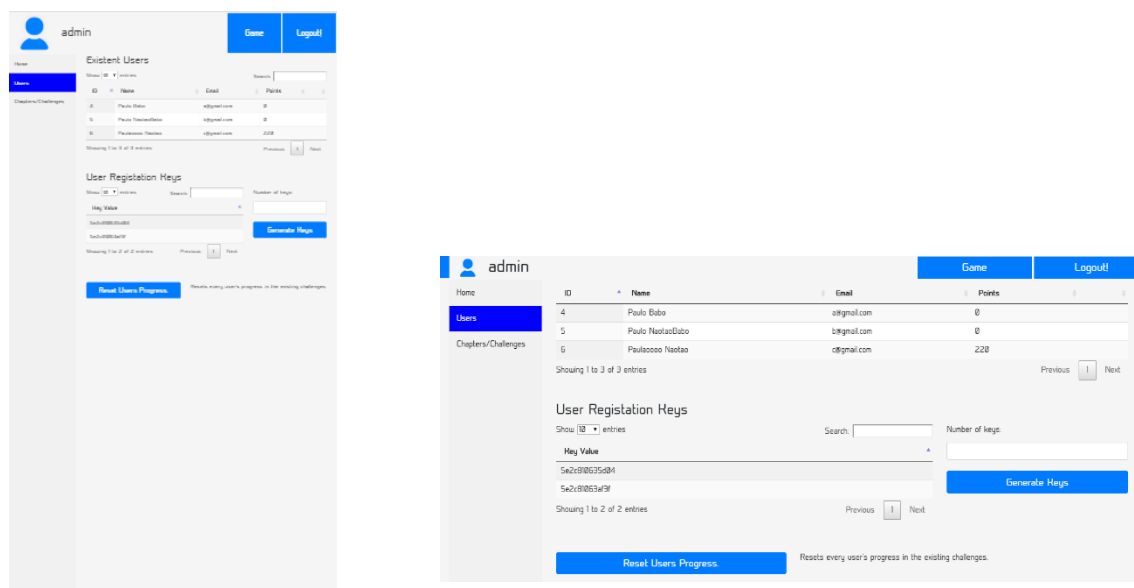


Figure 21 - Mobile / Desktop Admin Area (Users) View Interface.

### 4.7.20 Admin Area (Chapter/Challenges) View.

*Frontend/admin\_chapters.html*

In this area the admin can lock/unlock a chapter and view its info, reload the challenges and the atlas and clear the database, by clicking the respective buttons.

There is at the top, a header that holds the authenticated admin information (name), a “game” button and a “logout” button. The “game” button is used to return to the Main Menu view and the “logout” button to logout the admin.

To navigate between admin area views there is a side bar to the left, with the possible sections.

Solution

**Javascript –**

Table 44 – Admin Area (Chapter/Challenges) View javascript description.

<b>Event</b>	<b>Action</b>
<b>On Document Load</b>	When the document loads, an AJAX post request is made to the <i>Backend/API/get_chapters_info.php</i> , if the request is successful, the chapters info is displayed in the chapters data-table, otherwise an error message is shown.
<b>Lock/Unlock Chapter Button Click</b>	An AJAX post request is made to the <i>Backend/API/unlock_lock_chapter.php</i> , if the request is successful the chapter info is updated in the table, and a success message shown. Otherwise an error message is shown.
<b>Reload Challenges Button</b>	An AJAX post request is made to the <i>Backend/API/reload_challenges.php</i> , if the request is successful the chapters table is reloaded with the new chapter's info, and a success message shown. Otherwise an error message is shown.
<b>Reload Atlas Button</b>	An AJAX post request is made to the <i>Backend/API/reload_atlas.php</i> , if the request is successful a success message is shown. Otherwise an error message is shown.
<b>Clear Database Button</b>	An AJAX post request is made to the <i>Backend/API/clear_database.php</i> , if the request is successful a success message is shown. Otherwise an error message is shown.
<b>Logout Button Click</b>	User is logged out (jwt cookies cleared) and login view is loaded.
<b>Sidebar Users section Click</b>	Admin Area (Users) View is loaded.

## Solution

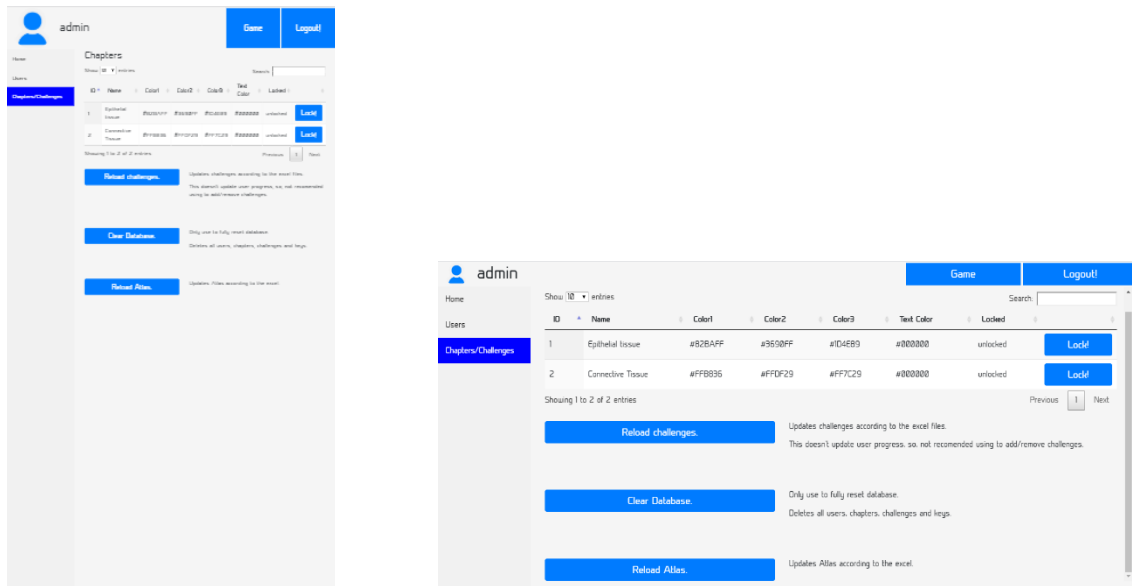


Figure 22 - Mobile / Desktop Admin Area (Chapters/Challenges) View Interface.

## 4.8 User Testing.

In the final stages of development, a user test was performed with 10 students that were attending the course of “Functional Histology” in the Institute of Biomedical Sciences Abel Salazar. In the test the users were asked to do simple tasks and then provide feedback about them. The tasks were:

- Register and login into the system.
- Navigate to the challenge selection screen.
- Conclude a challenge of each type.
- Amplify an Atlas Item.

While the students were performing the tasks various problems were encountered, most of them had to do with the code and not the interface navigation. Also, the students while performing said tasks were giving verbal feedback on what they would like to be added or changed.

At the end it was asked them to answer a questionnaire (Attachment 2.) in order to evaluate the usability and efficacy of the platform. Even though, there were a small number of answers, they were positive and showed good forecasts.

## Solution

After the test some of given feedback was implemented and the problems encountered were solved. Also, the user interface was enhanced.



## Chapter 5

# Conclusions and Future Work

This work has been motivated by the difficulty of engaging the students in learning histology beyond using microscopy sessions in the lab and the lack of material for them to do so.

So, in order to try solving this problem it was developed an online platform, that can be used anywhere, with the use of a PC or a smartphone, and that incorporates not only information but also engaging elements in order to help the students learning progress. The platform makes use of game design elements such as an engaging narrative and challenges in order to offer students an attractive and interactive way of learning.

To help the teacher create/manage the platform's content it was added a restricted area where the teacher can oversee the users and update the narrative and challenges with ease by uploading a csv file to the backend of the platform.

This allows the students to not only access histologic content anywhere and anytime, but also lets them test and improve their knowledge while participating in engaging and fun challenges that are accompanied by a compelling narrative.

A major limitation of this solution is the fact that the professor still needs to create files and fill them appropriately, otherwise the platform contents will not be correctly displayed. Future work should focus on providing the teacher with a tool that lets them create and update the challenges visually, so it is easier for them to create a personalized platform.

Also, in order to increase engagement and motivation from the students the interface should be enhanced, and music and animations added.

Even though the expected platform was developed and the key mechanics implemented I would have liked to have more time to work on it, as I would improve the admin area as well as

## Conclusions and Future Work

add more gamification elements so the students would be able to use it more and have more fun with it while also learning.

# Attachments

## 6.1 Attachment 1 – API description.

- **Clear database.**

**API:**

Backend/API/clear\_database.php

**Method:**

POST

**Description:**

Empties all database tables.

**Permission:**

Admin.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.

**Success:**

- message {string} – Message describing the result of the call.

**Example response:**

HTTP /1.1 200 OK

{

    “message”: “database successfully empty”

}

## Attachments

### **Error:**

- message {string} – Message describing the result of the call.

### **Example response:**

```
HTTP /1.1 401 Unauthorized
{
    "message": "access denied"
}
```

## • **Generate registration key.**

### **API:**

Backend/API/generate\_registration\_key.php.

### **Method:**

POST

### **Description:**

Creates a new registration key and inserts it into the 'registrationKeys' database table.

### **Permission:**

Admin.

### **API parameters:**

- jwt {String} – JSON web token of the authenticated user.

### **Success:**

- message {string} – Message describing the result of the call.

### **Example response:**

```
HTTP /1.1 200 OK
{
    "message": "Success"
}
```

### **Error:**

- message {string} – Message describing the result of the call.

## Attachments

### Example responses:

HTTP /1.1 400 Bad Request

```
{  
    "message": "Something went wrong"  
},
```

HTTP /1.1 401 Unauthorized

```
{  
    "message": "Access denied!"  
}
```

## • Retrieve atlas item.

### API:

Backend/API/get\_atlas\_item.php

### Method:

POST

### Description:

Retrieves a given atlas item information from the database.

### Permission:

None.

### API parameters:

- itemId {number} – Id of the atlas item.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – Array containing the database information of the item.
  - id {number} – id of the atlas item.
  - image {string} – name of the atlas image.
  - Subtitle {string} – text of the atlas item.
  - chapterId {number} – id of the chapter associated to the atlas item.

**Example response:**

```
HTTP /1.1 200 OK
{
  "message": "success!",
  "data": {
    "id": "382",
    "image": "image.png",
    "subtitle": "subtitle text",
    "chapterId": "1"
  }
}
```

- **Retrieve BOSS challenge arguments.**

**API:**

Backend/API/get\_BOSS\_arguments.php

**Method:**

POST

**Description:**

Retrieves the arguments of all the BOSS challenge rows of a given chapter.

**Permission:**

None.

**API parameters:**

- chapterId {number} – id of the given chapter.

**Success:**

- message {string} – Message describing the result of the call.
- data {array} – Array of strings that represent JSON encoded BOSS challenge rows information.

**Example response:**

## Attachments

HTTP /1.1 200 OK

```
{
  "message": "Success!",
  "data": [{"id": "1", "challengeId": "42" ...}]
}
```

### • Retrieve challenge arguments (not BOSS type).

#### API:

Backend/API/get\_challenge\_arguments.php

#### Method:

POST

#### Description:

Retrieves an array of the given challenge arguments independently of the challenge type.  
Can't be used for BOSS challenge types.

#### Permission:

None.

#### API parameters:

- challengeId {number} – id of the given challenge.

#### Success:

- message {string} – Message describing the result of the call.
- Data {array} – array with the argument information of the given challenge.
  - id {number} – id of the argument.
  - challengeId {number} – given challenge Id.
  - number {number} – number identifier of the argument meaning.
  - content {string} – argument text content.

#### Example response:

HTTP /1.1 200 OK

```
{
  "message": "Success",
```

## Attachments

```
    "data": [  
        {  
            "id = 1,  
            "challengeId" = 3,  
            "number" = 1,  
            "content" = "question text"  
        },  
        ...  
    ]  
}
```

### • Retrieve Atlas items.

#### API:

Backend/API/get\_chapter\_atlas.php

#### Method:

POST

#### Description:

Retrives the image name of every atlas item of a given chapter.

#### Permission:

None.

#### API parameters:

- chapterId {number} – id of the given chapter.

#### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing every atlas item image name of the given chapter
  - id {number} – id of the atlas item.
  - image {string} – name of the item's image file.

#### Example response:



## Attachments

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": [
    {
      "id": 1,
      "image": "image_name.png"
    },
    ...
  ]
}
```

- **Retrieve user challenges information.**

**API:**

Backend/API/get\_chapter\_challenges\_info.php

**Method:**

POST

**Description:**

Retrieves the challenges info of a given chapter as well as the authenticated user progress on them.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- chapterId {number} – id of the given chapter.

**Success:**

- message {string} – Message describing the result of the call.
- data {array} – array containing the given chapter challenges information as well as the user progress on them.

## Attachments

- id {number} – id of the challenge.
- number {number} – number of the challenge in the given chapter.
- Type {string} – type of the challenge
- userId {number} – authenticated user id.
- Locked {Boolean} – 0 or 1, represents if the challenge is or isn't locked for the authenticated user.
- Completed {Boolean} - 0 or 1, represents if the challenged is or isn't completed by the authenticated user.
- firstTry {Boolean} – 0 or 1, represents if the authenticated user already tried to complete the challenge.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": [
    {
      "id": 1,
      "number": 1,
      "type": "CHOICE",
      "userId": 1,
      "locked": 0,
      "completed": 0,
      "firstTry": 1
    },
    ...
  ]
}
```

- **Retrieve story dialogs.**

## Attachments

### API:

Backend/API/get\_chapter\_dialog.php

### Method:

POST

### Description:

Retrieves all the dialog items of a given chapter.

### Permission:

None.

### API parameters:

- chapterId {number} – id of the given chapter.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing the given story items.
  - id {number} – id of the story item.
  - number {number} – number of the item in the given chapter.
  - characterName {string} – name of the character speaking.
  - content {string} – quote of dialog.
  - backgroundImg {string} – name of the background media file.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": [
    {
      "id": 1,
      "number": 1,
      "characterName": "character",
      "backgroundImg": "image_name.png",
```

## Attachments

```
        },  
        ...  
    ]  
}
```

### • Retrieve chapters.

**API:**

Backend/API/get\_chapters\_info.php

**Method:**

POST

**Description:**

Retrieves every chapter's info.

**Permission:**

None.

**API parameters:**

None.

**Success:**

- message {string} – Message describing the result of the call.
- data {array} – array containing every chapter's info.
  - id {number} – id of the chapter.
  - name {string} – number of the item in the given chapter.
  - color1 {string} – hexadecimal code of the primary color of the chapter.
  - color2 {string} – hexadecimal code of the secondary color of the chapter.
  - color3 {string} – hexadecimal code of the tertiary color of the chapter.
  - textcolor {string} – hexadecimal code of the text color of the chapter.
  - locked {boolean} – 0 or 1, identifies if chapter is or isn't locked.

## Attachments

### Example response:

```
HTTP /1.1 200 OK
{
  "message": "success",
  "data": [
    {
      "id": 1,
      "name": "chapter name",
      "color1": "#000000",
      "color2": "#000000",
      "color3": "#000000",
      "textcolor": "#000000",
      "locked": 0
    },
    ...
  ]
}
```

- **Retrieve leaderboard users.**

**API:**

Backend/API/get\_leaderboard.php

**Method:**

POST

**Description:**

Retrieves the 10 users with most points, in descendent order.

**Permission:**

None.

**API parameters:**

## Attachments

None.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing the user's necessary info.
  - name {string} – name of the user.
  - points {number} – number of points of the user.

### Example response:

```
HTTP /1.1 200 OK
{
    "message": "success",
    "data": [
        {
            "name": "user name",
            "points": 3000,
        },
        ...
    ]
}
```

## • Retrieve every registration key.

### API:

Backend/API/get\_registration\_keys.php

### Method:

POST

### Description:

Retrieves every registration key available for use.

### Permission:

Admin.

## Attachments

### API parameters:

- jwt {String} – JSON web token of the authenticated admin.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing the keys.
  - keyvalue {string} – key string.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": [
    {
      "keyvalue": "1b2b3j4m12314",
    },
    ...
  ]
}
```

### Error:

- message {string} – Message describing the result of the call.

### Example response:

HTTP /1.1 401 Unauthorized

```
{
  "message": "access denied"
}
```

- **Retrieve if first time playing the chapter challenges.**

### API:

Backend/API/get\_show\_dialog.php

## Attachments

**Method:**

POST

**Description:**

Retrieves if the user tried to complete the first challenge of a chapter.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- chapterId {number} – id of the given chapter.

**Success:**

- message {string} – Message describing the result of the call.
- data {boolean} – Boolean that indicates if user tried or not to complete the first challenge of the given chapter.

**Example response:**

```
HTTP /1.1 200 OK
{
    "message": "success",
    "data": false
}
```

### • Retrieve user info.

**API:**

Backend/API/get\_user\_info.php

**Method:**

POST

**Description:**

Retrieves the authenticated user's necessary info.

**Permission:**



## Attachments

User.

### API parameters:

- jwt {String} – JSON web token of the authenticated user.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – Array with the necessary user info.
  - name {string} - name of the authenticated user.
  - points {number} – number of points of the authenticated user.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": {
    "name": "user name",
    "points": 1000
  }
}
```

## • Retrieve every user info.

### API:

Backend/API/get\_users\_info.php

### Method:

POST

### Description:

Retrieves the info of every registered user.

### Permission:

Admin.

### API parameters:

## Attachments

- jwt {String} – JSON web token of the authenticated admin.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – Array containing the info of every user.
  - id {number} – id of the user.
  - name {string} – name of the user.
  - email {string} – email of the user.
  - points {number} – number of points of the user.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": [
    {
      "id": 1,
      "name": "user name",
      "email": "email@email.com",
      "points": 2000
    },
    ...
  ]
}
```

### Error:

- message {string} – Message describing the result of the call.

### Example responses:

HTTP /1.1 401 Unauthorized

```
{
  "message": "Access denied!"
}
```

- **User authentication.**

**API:**

Backend/API/login.php

**Method:**

POST

**Description:**

Authenticates the user if it is registered in the system and retrieves its corresponding JWT.

**Permission:**

None.

**API parameters:**

- email {String} – email of the user to be authenticated.
- Password {String} – Corresponding password.

**Success:**

- message {string} – Message describing the result of the call.
- jwt {string} – Corresponding authentication JWT token.
- Admin {boolean} – 0 or 1, indication if the user is or isn't an admin.

**Example response:**

HTTP /1.1 200 OK

```
{  
    "message": "Successful login.",  
    "jwt": "tokenValue",  
    "admin": 1  
}
```

**Error:**

- message {string} – Message describing the result of the call.

**Example responses:**

## Attachments

```
HTTP /1.1 401 Unauthorized
{
  "message": "Login failed."
}
```

- **Reset every user's challenge progress.**

**API:**

Backend/API/reset\_users\_progress.php

**Method:**

POST

**Description:**

Resets the challenge progress of every user, by emptying the 'userchallenges' database table.

**Permission:**

Admin.

**API parameters:**

- jwt {String} – JSON web token of the authenticated admin.

**Success:**

- message {string} – Message describing the result of the call.

**Example response:**

```
HTTP /1.1 200 OK
{
  "message": "success"
}
```

**Error:**

- message {string} – Message describing the result of the call.

**Example responses:**

```
HTTP /1.1 401 Unauthorized
{
```

## Attachments

```
        "message": "Access denied."
    }
}
```

HTTP /1.1 401 Bad Request

```
{
    "message": "Something went wrong."
}
```

### • **User registration.**

#### **API:**

Backend/API/signup.php

#### **Method:**

POST

#### **Description:**

Registers the user in the system if the Registration Key is valid.

#### **Permission:**

None.

#### **API parameters:**

- name {string} – user’s name.
- email {string} – user’s email.
- password {string} – user’s password.
- registrationKey {string} – user’s registration key.

#### **Success:**

- message {string} – Message describing the result of the call.

#### **Example response:**

HTTP /1.1 200 OK

```
{
    "message": "successful signup"
}
```

Attachments

```
}
```

**Error:**

- message {string} – Message describing the result of the call.

**Example responses:**

HTTP /1.1 401 Unauthorized

```
{
```

```
  "message": "Access denied."
```

```
}
```

HTTP /1.1 401 Bad Request

```
{
```

```
  "message": "Something went wrong."
```

```
}
```

- **Lock/unlock chapter.**

**API:**

Backend/API/unlock\_lock\_chapter.php

**Method:**

POST

**Description:**

Locks or unlocks a chapter.

**Permission:**

Admin.

**API parameters:**

- jwt {string} – JSON web token of the authenticated admin.
- chapterId {number} – id of the given chapter.
- action {string} – action to perform (lock or unlock).

**Success:**

## Attachments

- message {string} – Message describing the result of the call.

### Example response:

```
HTTP /1.1 200 OK
{
  "message": "challenge updated"
}
```

### Error:

- message {string} – Message describing the result of the call.

### Example responses:

```
HTTP /1.1 401 Unauthorized
{
  "message": "Access denied."
}
```

```
HTTP /1.1 401 Bad Request
{
  "message": "Something went wrong."
}
```

## • Verify BOSS challenge answer.

### API:

Backend/API/verify\_BOSS.php

### Method:

POST

### Description:

Verifies if the answers selected by the user, in an unlocked BOSS type challenge, are correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

### Permission:

## Attachments

User.

### API parameters:

- jwt {String} – JSON web token of the authenticated user.
- chapterId {number} – id of the chapter related to the challenge.
- challengeId {number} – id of the first BOSS type challenge of the given chapter.
- selectedAnswers {array} – array containing the indexes of the chosen answers for the challenge.

### Success:

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.

### Example responses:

HTTP /1.1 200 OK

```
{  
    "message": "success",  
    "result": "correct",  
    "points": 20  
}
```

HTTP /1.1 200 OK

```
{  
    "message": "success",  
    "result": "incorrect",  
}
```

### Error:

- message {string} – Message describing the result of the call.

### Example responses:

HTTP /1.1 401 Unauthorized



## Attachments

```
{  
  "message": "Access denied."  
}
```

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Challenge locked."  
}
```

HTTP /1.1 401 Bad Request

```
{  
  "message": "Something went wrong."  
}
```

- **Verify CHOICE challenge answer.**

**API:**

Backend/API/verify\_CHOICE.php

**Method:**

POST

**Description:**

Verifies if the answer selected by the user, in an unlocked CHOICE type challenge, is correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- selectedIndex {number} – Index of the chosen answer for the challenge.

## Attachments

### Success:

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.

### Example responses:

HTTP /1.1 200 OK

```
{  
  "message": "success",  
  "result": "correct",  
  "points": 20  
}
```

HTTP /1.1 200 OK

```
{  
  "message": "success",  
  "result": "incorrect",  
}
```

### Error:

- message {string} – Message describing the result of the call.

### Example responses:

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Access denied."  
}
```

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Challenge locked."  
}
```

## Attachments

```
}
```

```
HTTP /1.1 401 Bad Request
```

```
{
```

```
    "message": "Something went wrong."
```

```
}
```

- **Verify CROSSWORD challenge answer.**

**API:**

Backend/API/verify\_CROSSWORD.php

**Method:**

POST

**Description:**

Verifies if the answers selected by the user, in an unlocked CROSSWORD type challenge, are correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- userAnswers {array} – Array with the chosen user answers.

**Success:**

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.

**Example responses:**

```
HTTP /1.1 200 OK
```

```
{
```

## Attachments

```
    "message": "success",
    "result": "correct",
    "points": 20
  }
```

HTTP /1.1 200 OK

```
{
  "message": "success",
  "result": "incorrect",
}
```

### **Error:**

- message {string} – Message describing the result of the call.

#### **Example responses:**

HTTP /1.1 401 Unauthorized

```
{
  "message": "Access denied."
}
```

HTTP /1.1 401 Unauthorized

```
{
  "message": "Challenge locked."
}
```

HTTP /1.1 401 Bad Request

```
{
  "message": "Something went wrong."
}
```

## Attachments

- **Verify FILL challenge answer.**

**API:**

Backend/API/verify\_FILL.php

**Method:**

POST

**Description:**

Verifies if the answer selected by the user, in an unlocked FILL type challenge, is correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- selectedIndexes {string} – string with the indexes of the selected answers ordered and separated by a '/'.

**Success:**

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.
- answer {array} – string with the correct answers indexes, ordered and separated by a '/'.

**Example responses:**

```
HTTP /1.1 200 OK
{
    "message": "success",
    "result": "correct",
    "points": 20
}
```

## Attachments

HTTP /1.1 200 OK

```
{  
  "message": "success",  
  "result": "incorrect",  
  "answer" : "1/3"  
}
```

### **Error:**

- message {string} – Message describing the result of the call.

### **Example responses:**

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Access denied."  
}
```

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Challenge locked."  
}
```

HTTP /1.1 401 Bad Request

```
{  
  "message": "Something went wrong."  
}
```

- **Verify GROUP challenge answer.**

### **API:**

## Attachments

Backend/API/verify\_GROUP.php

### Method:

POST

### Description:

Verifies if the answers selected by the user, in an unlocked GROUP type challenge, are correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

### Permission:

User.

### API parameters:

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- leftChoices {array} – Array with the left group chosen answers.
- rightChoices {array} – Array with the right group chosen answers.

### Success:

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.
- answer {array} – string with the correct user answers.
  - right {array} – array containing the correct right group string answers.
  - left {array} – array containing the correct left group string answers.

### Example responses:

HTTP /1.1 200 OK

```
{  
    "message": "success",  
    "result": "correct",  
    "points": 20  
}
```

## Attachments

HTTP /1.1 200 OK

```
{
  "message": "success",
  "result": "incorrect",
  "answer":
    {
      "right": [
        "right1",
        ...
      ]
      "left": [
        "left1",
        ...
      ]
    }
}
```

### **Error:**

- message {string} – Message describing the result of the call.

### **Example responses:**

HTTP /1.1 401 Unauthorized

```
{
  "message": "Access denied."
}
```

HTTP /1.1 401 Unauthorized

```
{
```



## Attachments

```
        "message": "Challenge locked."
    }
}
```

HTTP /1.1 401 Bad Request

```
{
    "message": "Something went wrong."
}
```

### • **Verify LETTERS challenge answer.**

#### **API:**

Backend/API/verify\_LETTERS.php

#### **Method:**

POST

#### **Description:**

Verifies if the answer selected by the user, in an unlocked LETTERS type challenge, is correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

#### **Permission:**

User.

#### **API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- selectedString {string} – String of the selected answer.

#### **Success:**

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.

#### **Example responses:**

## Attachments

HTTP /1.1 200 OK

```
{  
  "message": "success",  
  "result": "correct",  
  "points" : 20  
}
```

HTTP /1.1 200 OK

```
{  
  "message": "success",  
  "result": "incorrect",  
}
```

### **Error:**

- message {string} – Message describing the result of the call.

#### **Example responses:**

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Access denied."  
}
```

HTTP /1.1 401 Unauthorized

```
{  
  "message": "Challenge locked."  
}
```

HTTP /1.1 401 Bad Request

```
{  
  "message": "Something went wrong."  
}
```

```
}
```

- **Verify MCHOICE challenge answer.**

**API:**

Backend/API/verify\_MCHOICE.php

**Method:**

POST

**Description:**

Verifies if the answer selected by the user, in an unlocked MCHOICE type challenge, is correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- selectedIndexes {string} – string of the selected answer indexes separated by a '/'.

**Success:**

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.
- Answer {array} – Array containing the indexes of the correct answers.

**Example responses:**

```
HTTP /1.1 200 OK
```

```
{
```

```
    "message": "success",
```

```
    "result": "correct",
```

```
    "points": 20
```

## Attachments

```
}
```

```
HTTP /1.1 200 OK
```

```
{  
    "message": "success",  
    "result": "incorrect",  
    "answer" : [  
        "1",  
        ...  
    ]  
}
```

### **Error:**

- message {string} – Message describing the result of the call.

#### **Example responses:**

```
HTTP /1.1 401 Unauthorized
```

```
{  
    "message": "Access denied."  
}
```

```
HTTP /1.1 401 Unauthorized
```

```
{  
    "message": "Challenge locked."  
}
```

```
HTTP /1.1 401 Bad Request
```

```
{  
    "message": "Something went wrong."  
}
```

- **Verify PAIR challenge answers.**

**API:**

Backend/API/verify\_PAIR.php

**Method:**

POST

**Description:**

Verifies if the answer selected by the user, in an unlocked PAIR type challenge, is correct. And if so, unlocks the next challenge and increments the authenticated user's points accordingly.

**Permission:**

User.

**API parameters:**

- jwt {String} – JSON web token of the authenticated user.
- challengeId {number} – id of the challenge.
- selectedConnections {array} – array containing the selected pairs strings, pair strings have the following syntax: “left // right”.

**Success:**

- message {string} – Message describing the result of the call.
- result {string} – Message describing the result of the verification.
- points {number} – Number of points won by the authenticated user.
- Answer {array} – Array containing the pair strings.

**Example responses:**

HTTP /1.1 200 OK

```
{  
    "message": "success",  
    "result": "correct",  
    "points": 20  
}
```

## Attachments

HTTP /1.1 200 OK

```
{
  "message": "success",
  "result": "incorrect",
  "answer" : [
    "left1 // right1",
    ...
  ]
}
```

### **Error:**

- message {string} – Message describing the result of the call.

### **Example responses:**

HTTP /1.1 401 Unauthorized

```
{
  "message": "Access denied."
}
```

HTTP /1.1 401 Unauthorized

```
{
  "message": "Challenge locked."
}
```

HTTP /1.1 401 Bad Request

```
{
  "message": "Something went wrong."
}
```

- **Verify JWT.**

## Attachments

### API:

Backend/API/verify\_token.php

### Method:

POST

### Description:

Verifies if a JWT is valid.

### Permission:

None.

### API parameters:

- jwt {String} – JSON web token of the authenticated user.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing the data associated to the JWT.
  - id – id of the user associated with the JWT.
  - email – email of the user associated with the JWT.
  - admin – 0 or 1, identifies if the user is or is not an admin.

### Example response:

HTTP /1.1 200 OK

```
{
  "message": "success",
  "data": {
    "id": 1,
    "email": "email@email.com"
    "admin": 0
  }
}
```

### Error:

## Attachments

- message {string} – Message describing the result of the call.

### Example response:

```
HTTP /1.1 401 Unauthorized
{
    "message": "Access denied."
}
```

## • Verify admin JWT.

### API:

Backend/API/verify\_token\_admin.php

### Method:

POST

### Description:

Verifies if a JWT is valid and is associated with an admin.

### Permission:

None.

### API parameters:

- jwt {String} – JSON web token of the authenticated admin.

### Success:

- message {string} – Message describing the result of the call.
- data {array} – array containing the data associated to the JWT.
  - id – id of the user associated with the JWT.
  - email – email of the user associated with the JWT.
  - admin – 0 or 1, identifies if the user is or is not an admin.

### Example response:

```
HTTP /1.1 200 OK
{
    "message": "success",
```



## Attachments

```
    "data": {  
        "id": 1,  
        "email": "email@email.com"  
        "admin": 1  
    }  
}
```

### **Error:**


- message {string} – Message describing the result of the call.

#### **Example response:**

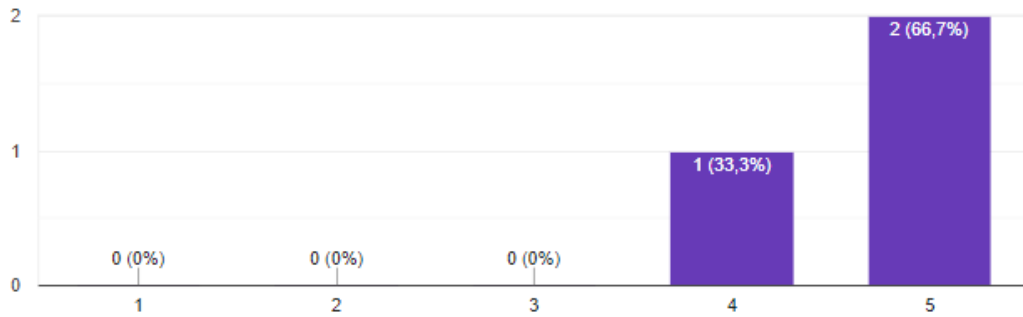
```
HTTP /1.1 401 Unauthorized  
{  
    "message": "Access denied."  
}
```


End of attachment 1.

## 6.2 Attachment 2 – User test questionnaire.

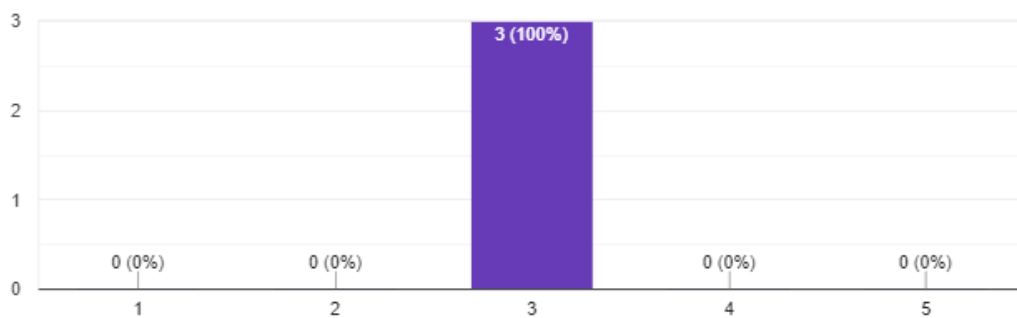
From 1 to 5, how do you measure the importance of the app being available anywhere with an internet connection? 

3 respostas



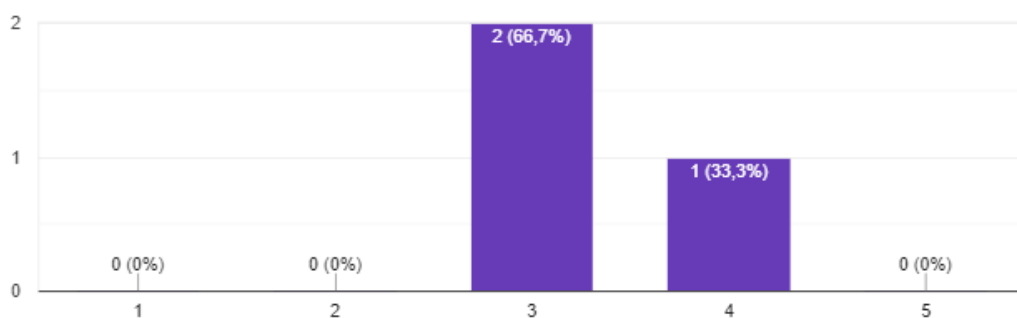
From 1 to 5. While doing the challenges, how easy was it to understand the game mechanics and know what to do in each situation 

3 respostas



From 1 to 5. How would you rate the overall attractiveness and aspect of the game?

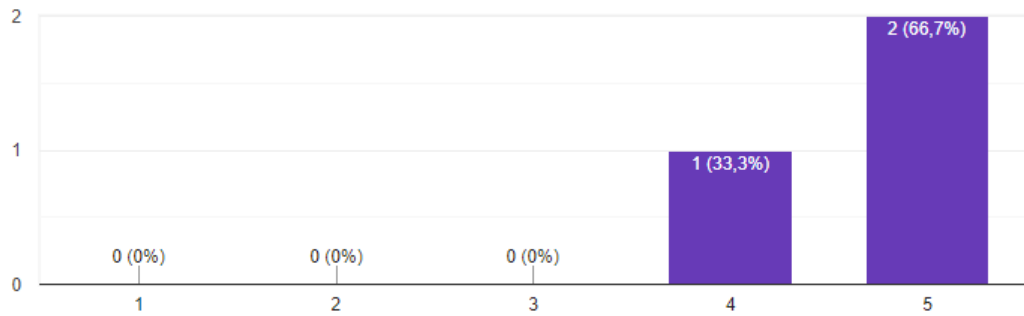
3 respostas



## Attachments

From 1 to 5, what do you think it's the overall usefulness of the app?

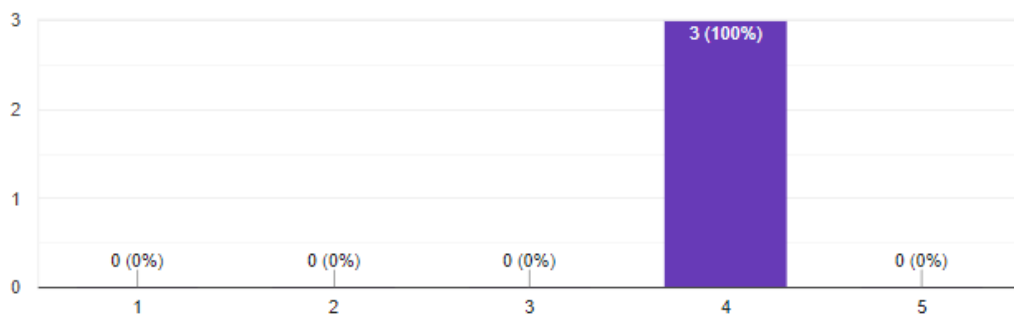
3 respostas



From 1 to 5, how would you evaluate the navigation in the app?



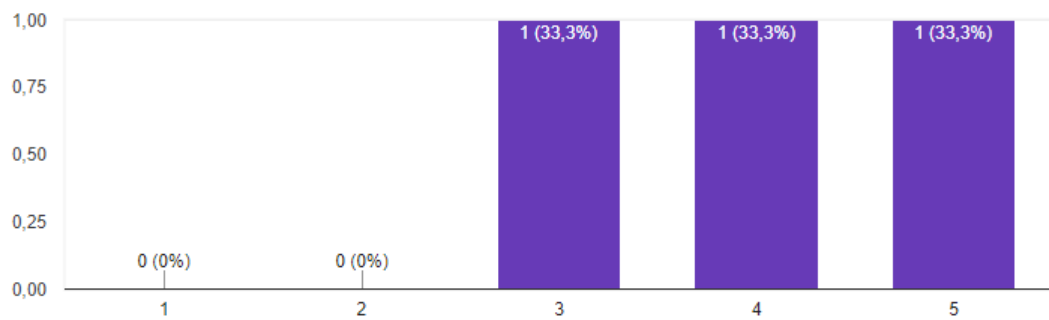
3 respostas



From 1 to 5, how do you measure the importance of the existence of points in your motivation to play the challenges



3 respostas

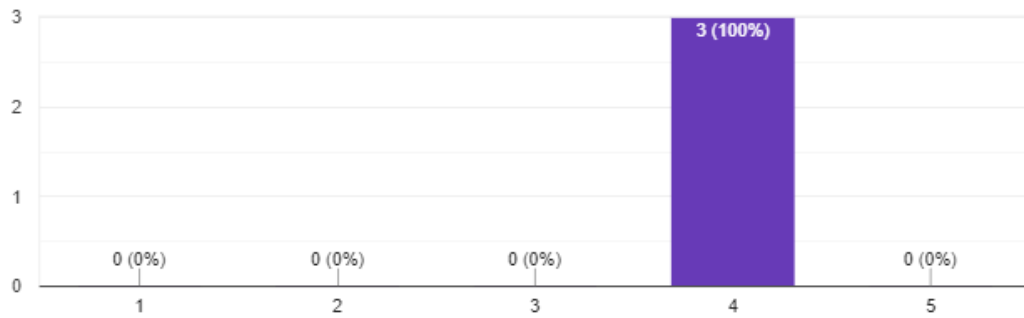


## Attachments

From 1 to 5. While doing the challenges, how easy was it to know where to click to perform the desired task



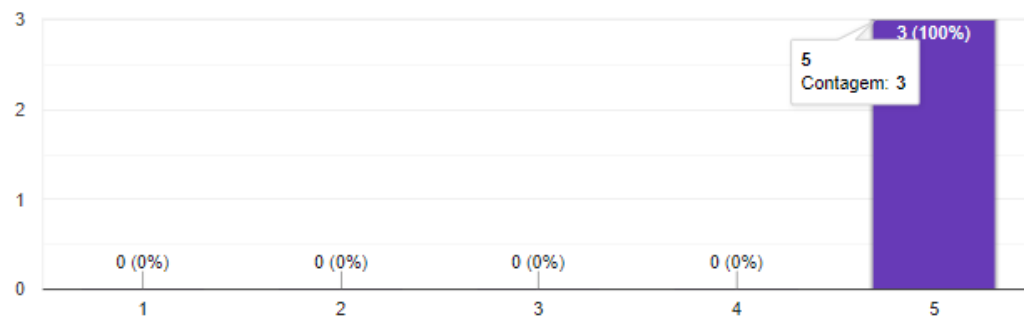
3 respostas



From 1 to 5, how do you measure the usefulness of the existence of the challenges in helping you learn?



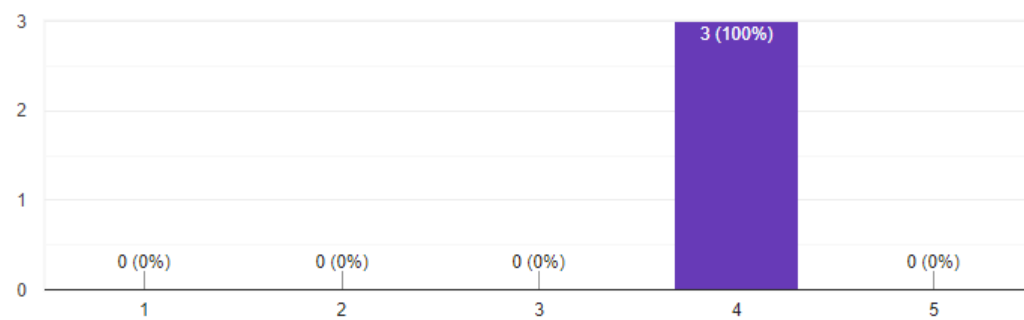
3 respostas



From 1 to 5, how do you measure the usefulness of the existence of a story to give you information instead of a classic approach



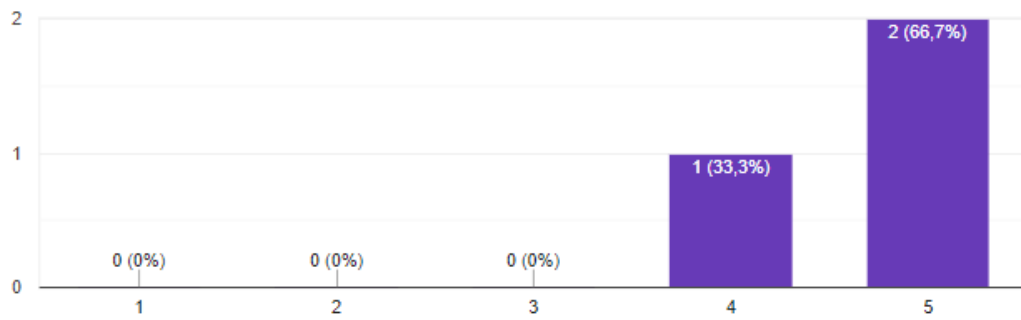
3 respostas



## Attachments

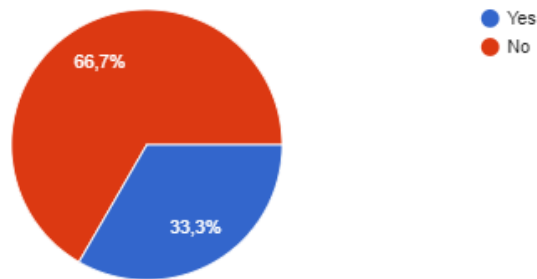
From 1 to 5, how do you measure the importance of the app being available anywhere with an internet connection?

3 respostas



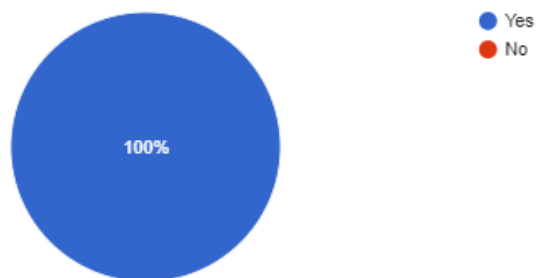
With enough content, do you think this app can substitute classic study methods?

3 respostas



Would you play this app in your free time? (for example: while on the bus)

3 respostas



## Attachments

Would you use this app to help you with your studies?



3 respostas



● Yes  
● No

Do you think this app can help save time in your studies?

3 respostas

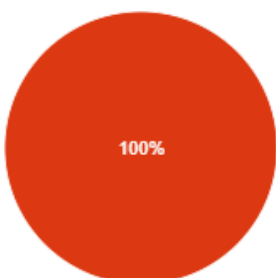


● Yes  
● No

Were you more engaged using the app than you would be studying?



3 respostas

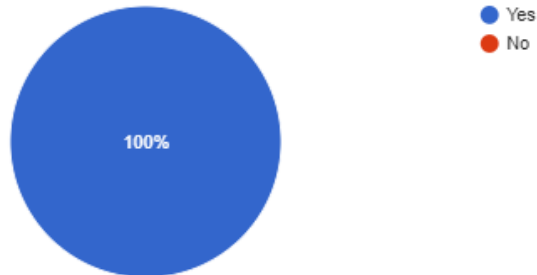


● Yes  
● No

## Attachments

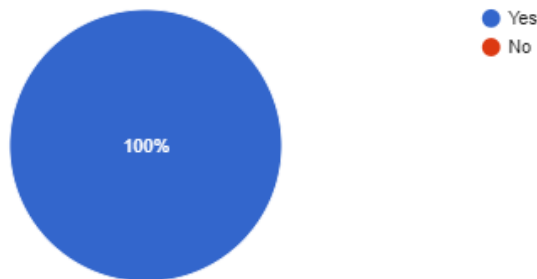
Do you think an addition of a leaderboard could be important in the motivation to use the app and play more?

3 respostas



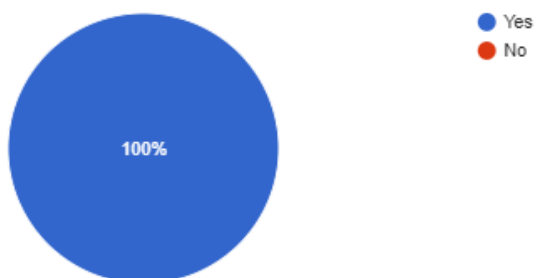
Do you think the implementation of this concept could be important in helping students learn and grow their motivation towards classes?

3 respostas



The challenges, were they fun? ^^

3 respostas



## 6.3 Attachment 3 – Example of .CSV files.

### 6.3.1 Chapter/Dialog file.

Epithelial tissue	#82BAFF	#3690FF	#1D4E89	#000000					
ORDER NUMBER	TYPE	ARGUMENT 1	ARGUMENT 2	ARGUMENT 3	ARGUMENT 4	ARGUMENT 5	ARGUMENT 6	ARGUMENT 7	ARGUMENT 8
1	CHOICE	What is the name of the structure?		2 mitochondria	microvilli	golgi complexes	junctional complexes		
2	CROSSWORD	Epithelial tissue that forms the lining of the gut	Epithelial tissue that lines the lining of the gut	Epithelium found only in the lining of the gut	Ductless gland that releases hormones	Gland in which secretions are released into the bloodstream	In this type of secretion, the secretory cells are not attached to each other	Their movement is directed towards the apical surface of the cell	Epithelium
3	FILL	The ( ) exocrine gland stores secretions in the cytoplasm		02-Apr diffuse	holocrine	eccrine	apocrine	endocrine	
4	GROUP	Simple squamous epithelium	Cava vein // Lung alveoli	Skin // Esophagus // Anus // Cornea					
5	LETTERS	Epithelial tissue that lines the endothelium							
6	PAIR	endotheliocytes // Figure_nagoblet cell // Figure_nar	epithelium simple cuboidal	brush border // Figure_nar	transitional epithelium	epithelium simple cuboidal	epithelium simple cuboidal	epithelium stratified cuboidal	serous cavity
7	BOSS	The basic shapes of epithelium		2 squamous	stratified cuboidal	simple cuboidal	columnar		
8	BOSS	The type of epithelium that lines the lining of the gut		4 stratified squamous	stratified cuboidal	simple cuboidal	simple squamous		
9	BOSS	The type of epithelium that lines the lining of the gut		1 simple squamous	simple cuboidal	stratified squamous	stratified cuboidal		
10	BOSS	The type of epithelium found in the lining of the gut		2 simple squamous	simple cuboidal	transitional	pseudostratified columnar		
1	DIALOG	Hello, my name is Karl Mayer	Karl Mayer		Karl Mayer				
2	DIALOG	Before the story begun let me introduce myself	Karl Mayer		Karl Mayer				
3	DIALOG	For different reasons, sometimes I can be very helpful	Karl Mayer		Karl Mayer				
4	DIALOG	I don't want to give you the virus	Karl Mayer		Karl Mayer				
5	DIALOG	In some cases, I must be very helpful	Karl Mayer		Karl Mayer				
6	DIALOG	This is a story of a little boy named Narrator							
7	DIALOG	Here in the mouth the wall of the mouth is made of	Cell Wizard	Mouth	Karl Mayer\Cell Wizard				
8	DIALOG	Yes, it is a non-keratinized stratified squamous epithelium	Karl Mayer	Mouth	Karl Mayer\Cell Wizard				
9	DIALOG	Look here is an entrance for the salivary gland	Cell Wizard	Mouth	Karl Mayer\Cell Wizard				
10	DIALOG	It is a salivary gland. Those that are found in the mouth	Karl Mayer	Salivary gland	Karl Mayer\Cell Wizard				

### 6.3.2 Atlas file.

IMAGE	TEXT
01CH1	Simple squamous epithelium. Small artery, Masson trichrome staining. This photomicrograph shows a blood vessel (BV) which the lumen is lined by simple squamous epithelium named endothelium.
02CH1	Simple squamous epithelium. Small artery, H&E staining. This photomicrograph shows a blood vessel (BV) which the lumen is lined by simple squamous epithelium named endothelium. Nuclei are visible in the endothelial cells.
03CH1	Simple squamous epithelium. Small vein (SV), H&E staining. The only part of the endothelial cells that can be seen are their flattened nuclei (N).
04CH1	Simple squamous epithelium. Cardiac serosa (pericardium), H&E staining. This photomicrograph shows a serosa composed by a very thin connective tissue limited by a single layer (mesothelium).
05CH1	Simple squamous epithelium. Lung serosa (pleura), H&E staining. Lung serosa is covered by a simple squamous epithelium known as the mesothelium.
06CH1	Simple cuboidal epithelium. Kidney, Masson trichrome staining. This kidney section contains the profiles of many distal tubules (DT), which have a simple cuboidal epithelium.
07CH1	Simple cuboidal epithelium. Kidney, H&E staining. These simple cuboidal epithelial cells have round nuclei that are in the center of the cells. The apical surface of the cell exhibits abundant microvilli.
08CH1	Simple cuboidal epithelium. Kidney, H&E staining. Proximal convoluted tubules (PT) are densely stained pink. Distal convoluted (DT) tubules are paler and lack the brush border.



## References

1. Ryan, R.M.J.P.o.m., “A motivational approach to self: Integration in personality edward I., deci and”. 1991. 38(237): p. 237-288.
2. Duncan, G.J. and K.J.W.o. Magnuson, “The nature and impact of early achievement skills, attention skills, and behavior problems”. 2011: p. 47-70.
3. Coulter, R., et al., “The effect of degree of immersion upon learning performance in virtual reality simulations for medical education”. 2007. 15: p. 155.
4. Vogel, J.J., et al., “Computer gaming and interactive simulations for learning: A meta-analysis”. 2006. 34(3): p. 229-243.
5. Ke, F., “A qualitative meta-analysis of computer games as learning tools”, in *Gaming and Simulations: Concepts, Methodologies, Tools and Applications*. 2011, IGI Global. p. 1619-1665.
6. DeSmet, A., et al., *A meta-analysis of serious digital games for healthy lifestyle promotion*. 2014. 69: p. 95-107.
7. Kordaki, M., A.J.C. Gousiou, and Education, “Digital card games in education: A ten year systematic review”. 2017. 109: p. 122-161.
8. Michael, D.R. and S.L. Chen, “Serious games: Games that educate, train, and inform”. 2005: Muska & Lipman/Premier-Trade.
9. Burguillo, J.C.J.C. and education, “Using game theory and competition-based learning to stimulate student motivation and performance”. 2010. 55(2): p. 566-575.
10. Girard, C., J. Ecalle, and A.J.J.o.C.A.L. Magnan, “Serious games as new educational tools: how effective are they? A meta-analysis of recent studies”. 2013. 29(3): p. 207-219.
11. Barrett, H. “Researching and evaluating digital storytelling as a deep learning tool”. in *Society for information technology & teacher education international conference*. 2006. Association for the Advancement of Computing in Education (AACE).
12. Hamari, J., Koivisto, J., and Sarsa, H. “Does gamification work? - A literature review of empirical studies on gamification”. In *Proceedings of the Annual Hawaii International Conference on System Sciences, (IEEE Computer Society)*. 2014. p. 3025–3034.
13. Groh, F. “Gamification: State of the Art Definition and Utilization”. In *Proceedings of the 4th Seminar on Research Trends in Media Informatics (RTMI'12)*, N. Asaj, B. Konings, M. Poguntke, F. Schaub, B. Wiedersheim, and M. Weber, eds. (Institute of Media Informatics Ulm University). 2012. p. 39–46.
14. Urh, M., Vukovic, G., Jereb, E., and Pintar, R.. “The Model for Introduction of Gamification into E-learning in Higher Education”. *Procedia - Social and Behavioral Sciences* 197. 2015. p. 388–397.
15. Ismail, M.A.-A., and Mohammad, J.A.-M. “Kahoot: A Promising Tool for Formative Assessment in Medical Education”. *Education in Medicine Journal* 9. 2017. p. 19–26.
16. Huynh, D., Zuo, L., and Iida, H. *Analyzing gamification of “Duolingo” with focus on its course structure*. In *Lecture Notes in Computer Science*. 2016. p. 268–277.
17. Deterding, S., Dixon, D., Khaled, R., and Nacke, L.. “From game design elements to gamefulness: defining “gamification””. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11)*. ACM, New York, NY, USA, 2011, p. 9-15.
18. Histology Guide, virtual histology laboratorie. Available at: <http://www.histologyguide.com/index.html> . Access in 20/January/2020.
19. Why Is the Study of Histology Important in Your Overall Understanding of Anatomy & Physiology?. Available at: <https://sciencing.com/study-histology-important-overall-understanding-anatomy-physiology-23515.html> . Access in 20/January/2020.
20. Kahiigi, Evelyn & Ekenberg, Love & Hansson, Henrik & Tusubira, Francis & Danielson, Mats. 2008. “Exploring the e-Learning State of art”. *The Electronic Journal of e-Learning*. 6.
21. 10 Emerging technologies in E-learning – e-leaning. Available at: <https://elearning.adobe.com/2019/03/10-emerging-technologies-e-learning/>. Access in 22/January/2020.

## Attachments

22. What is a Content Management System (CMS)? Available at: <https://searchcontentmanagement.techtarget.com/definition/content-management-system-CMS>. Access in 22/January/2020.
23. Microlearning: learn what it is, its benefits and when to use this tool. Available at <https://mobiliza.com.br/conheca-o-microlearning/>. Access in 22/January/2020.
24. Giurgiu, Luminița. (2017). “Microlearning an Evolving Elearning Trend”. *Scientific Bulletin*. 22. 10.1515/bsaft-2017-0003.
25. JSON Web Token (JWT). Available at <https://tools.ietf.org/html/rfc7519>. Access in 22/January/2020
26. Nah F.FH., Zeng Q., Telaprolu V.R., Ayyappa A.P., Eschenbrenner B. (2014) “Gamification of Education: A Review of Literature”. In: Nah F.FH. (eds) *HCI in Business*. HCIB 2014. *Lecture Notes in Computer Science*, vol 8527. Springer, Cham.

