Development of a Platform for the Quimera Animal Physiology Project

Authors: Fernanda Bonfim Santos[1], Jean-Jacques De Groote[2]

Collaborators: Maria Eduarda do Prado Carvalho, Gelson Genaro

Centro Universitário Barão de Mauá

[1]fernandacomputerscience@gmail.com, [2]jean.georges@baraodemaus.br

Abstract

In recent years, there has been growing interest in developing alternative educational methods that promote effective learning, especially in sensitive areas such as animal physiology. This paper introduces an innovative proposal for the creation of an educational platform aimed at students seeking to understand the systems present in animal bodies. The primary goal of this platform is to provide an engaging and motivating educational approach through an educational portal incorporating gamified study rooms. By utilizing this approach, the aim is not only to promote active learning but also to stimulate student interest in the addressed topics, particularly animal physiology. Additionally, the platform seeks to highlight interactive and effective methods that can replace the need for live animals in educational contexts, reflecting ethical concerns and promoting more sustainable practices. In this context, this paper explores the design, development, and potential benefits of this innovative educational platform, providing a significant contribution to advancing education in biological sciences (RHARDER et al., 2010).

Introduction

Education in animal physiology often requires the use of live animals in practical experiments, raising ethical and logistical concerns. The Quimera project emerges as a response to these challenges, proposing an innovative solution based on computational simulations and interactive learning. This paper explores the development of a web platform for the Quimera project, detailing its technical architecture, functionalities, and educational applications (DE NORONHA, 2019).

The platform for the Quimera project was developed following a structured and methodological approach. Initially, literature reviews were conducted to identify the requirements and mechanisms necessary for creating an effective educational platform. Based on these analyses, the methods and technologies to be employed were established (RHARDER et al., 2010).

Materials and Methods

The calculations and data leading to the results used in this work were processed on the backend, saving memory and processing resources on the devices interacting with the platform. The development process involved flowcharts, UML diagrams, and data spreadsheets (e.g., Excel), progressing to the development of the platform's REST API. JavaScript with Node.js was used, alongside the Express framework and Mongoose library to connect to the MongoDB Atlas database. The REST API followed the MVC (Model-View-Controller) architecture pattern. In this context, user and experiment schemas were defined and managed in the model layer, while application logic functions were implemented in the controller layer. This setup enabled API endpoint creation. The entire backend was developed using functional programming principles.

For platform development, the team created a front-end, a REST API, and Python scripts for numerical simulation operations. The back-end of the application utilized JavaScript in Node.js and the MongoDB Atlas database to record information about each activity on the platform. React.js was used to build the user interface.

Results and Discussion

In constructing the platform, a base experiment was used to develop activities applicable to classroom use. Below are descriptions of the experiment and the developed platform features.

Experiment

The base experiment involves correcting water loss in an animal's body through different types of organism control. Students choose from 20 options, only two of which—ADH (Antidiuretic Hormone) and hypothalamus control—are effective. For simulation purposes, ADH has 20% effectiveness, while the hypothalamus has 80%.

This choice reflects the hypothalamus's greater contribution. Other options have no effect on water loss correction in this experiment.

Students receive an introductory study, including the case of the animal experiencing water loss, and must use their knowledge to select the correct item to prevent abnormal water loss. The program interface's selectors are divided into two categories, with only ADH and hypothalamus options being effective. Each option has an associated weight representing its efficacy in correcting water loss.

A graph helps interpret the results, displaying the expected experiment outcomes with the two correct options (ADH and hypothalamus) versus the student's choice.

Main Features

Quimera offers various functionalities designed to enhance user learning experiences, including:

1. Registration and authentication for teachers and students.

2. Creation and management of virtual experiments.

3. Simulations of abnormal water loss in animals.

4. Generation of graphs and analysis of experimental data.

Three data models were defined to represent system entities: Student, Teacher, and Experiment Room. Each model has attributes describing their characteristics and relationships with other entities. Mongoose Schema was used to define fields, data types, and validations.

In the MongoDB database used for the project, there are three main collections: experiment, teacher, and student. These collections store documents in BSON format, MongoDB's binary representation for data (Redmond & Wilson).

- Experiment Collection: Stores information related to experiments conducted on the platform, including parameters, results, and key dates.
- Teacher Collection: Stores teacher data such as name, email, institution, and subjects taught.
- Student Collection: Maintains student records, including name, email, course, and academic period.
- Controllers implemented API logic, manipulating received data and interacting with the database as needed. Three main controllers were created: one for each data model (Student, Teacher, and Experiment Room).

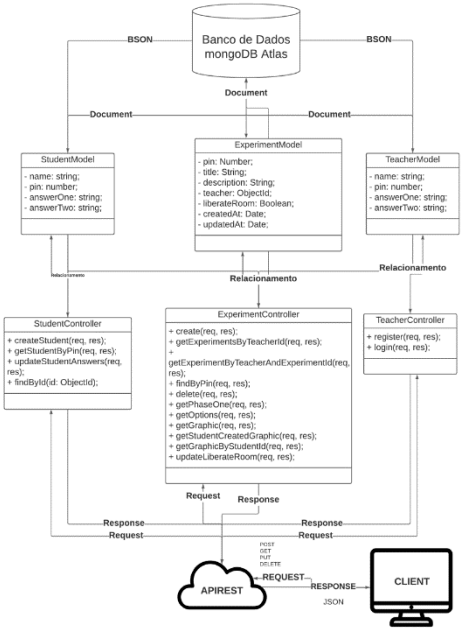
Controllers were responsible for implementing the API's logical functionalities, manipulating data received from requests and interacting with the database as necessary. Three main controllers were created: one for each data model (Student, Teacher and Experiment Room). Each controller has several functions responsible for performing operations such as creating, querying, updating and deleting records.

This diagram demonstrates how a REST API was developed with the MVC software architecture model, including its templates, schemas, and parameters. It shows how JSON constraints and responses are made to the client and how documents in BSON (Binary Json) format are stored in database collections.

Folder Structure & MVC

The project's folder architecture was organized according to the MVC (Model-View-Controller) pattern, ensuring a clear and modular structure. The folders were divided into models, controllers, routes, services, contexts, components, pages and utilities, each with its specific responsibility. This facilitated the maintenance, scalability and extensibility of the code throughout the project's development. Figure 1 presents a diagram that outlines how this architecture was designed, (Gama, N., & Campos, M. L. M.).

Figure 1 - MVC - Rest Api Diagram.



Fonte: Próprios autores.

The backend structure was divided into three main components:

- Models: This folder contains the schemas and attributes of the objects used in the application. Models define the structure of data and its relationships in the database.
- Controllers: This is where the logical part responsible for manipulating data models resides. Two subfolders were created in this directory:
- Auth: Contains the controllers responsible for authenticating system users, including functions such as registration, login and data recovery.
- Experiment: Encompasses controllers related to experiments, such as creation, data manipulation and graph generation.
- Database: This directory handles the connection to the database. The conn.js file is responsible for connection attempts to MongoDB from a string stored in the .env file.

Project architecture

- The project's folder architecture was organized according to the MVC (Model-View-Controller) pattern, ensuring a clear and modular structure. The folders were divided into models, controllers, routes, services, contexts, components, pages and utilities, each with its specific responsibility. This facilitated the maintenance, scalability and extensibility of the code throughout the development of the project, (Richardson, L., & Amundsen, M.).
- The backend structure was divided into three main components:
- Models: This folder contains the schemas and attributes of the objects used in the application. Models define the structure of data and its relationships in the database.
- Controllers: This is where the logical part responsible for manipulating data models resides. Two subfolders were created in this directory:
- Auth: Contains the controllers responsible for authenticating system users, including functions such as registration, login and data recovery.
- Experiment: Encompasses controllers related to experiments, such as creation, data manipulation and graph generation.
- Database: This directory handles the connection to the database. The conn.js file is responsible for connection attempts with MongoDB from a string stored in the .env file.

Organization of the Front-end Structure

The front-end folder structure was meticulously organized to optimize the development and maintenance of the project. The src directory represents the root file that houses all the raw content of the front-end project, while the files outside it are the default ones, including index.js.

1. Reusable Components

The components folder contains all the reusable components of the code. For example, BaseAuth is a component that contains the base style of the user authentication page, such as the background and card settings, as well as an area for inserting content. Each component is developed with an index.jsx file, which contains the React JavaScript code, and its own style file, styles.css.

2. Contexts and Providers

The context folder houses all the contexts and providers used in the application. Among them, the Users context stands out, which includes information relevant to the teacher, facilitating its reuse throughout the platform. In addition, there is the Authorization context, which manages the Boolean state of the button responsible for releasing the experiment results to the students.

3. Navigation Routes

The navigation routes are organized in the routes folder, created from the pages intended for user navigation. For better organization, two new subfolders were created: StudentPages and TeacherPages, which contain the JSX and CSS files for each page, clearly separating those accessed by the regular user and the teacher.

4. User Pages

The pages folder includes all the files created to build the user navigation pages. In order to properly separate the files for the pages intended for regular users and teachers, the StudentPages and TeacherPages folders were created. Within them, there are the JSX and CSS files for each page individually, ensuring a clear and intuitive organization of the structure.

Additional Details

Result Tables Folders: This folder contains files with arrays extracted from Excel spreadsheets that represent graphs of body water drop. The data was divided into different files for better organization.

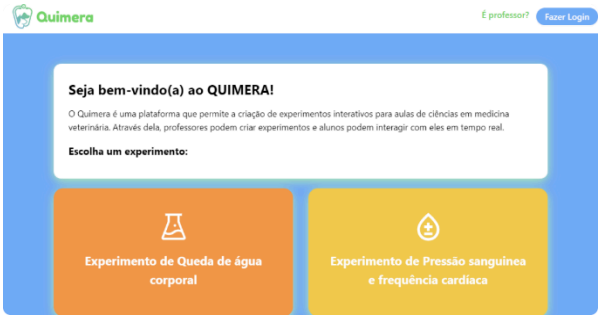
.env file: This is where sensitive application settings are stored, such as the database connection string. For security reasons, this file is not included in the Git repository.

.gitignore file: Responsible for excluding files that should not be shared in the Git repository from version control, such as node_modules, yarn_lock and .env.

Platform and Operation Images

This section presents the platform's operating steps. The platform's home screen is shown in Figure 2.

Figure 2. Platform Home Screen



Fonte: Próprios autores.

The system operation is then divided into eleven steps.

Step 1
Teacher access allows them to create a registration with their data on the platform (Fig.3).

Figure 3. Teacher registration.

Source: Authors themselves.

Step 2

The teacher logs into the platform with the data provided previously (Fig.4).

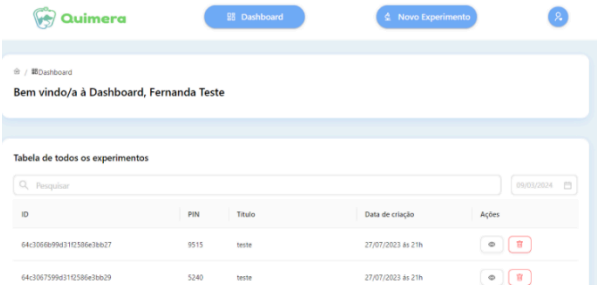
Figure 4. Teacher login.

Source: Authors themselves.

Etapa 3

O professor é redirecionado após o login para a Dashboard (Página principal), onde ele pode visualizar experimentos anteriores na tabela ou ir para a página de criar novos experimentos (Fig. 5).

Figura 5. Dashboard, onde o professor visualiza o histórico de experimentos.

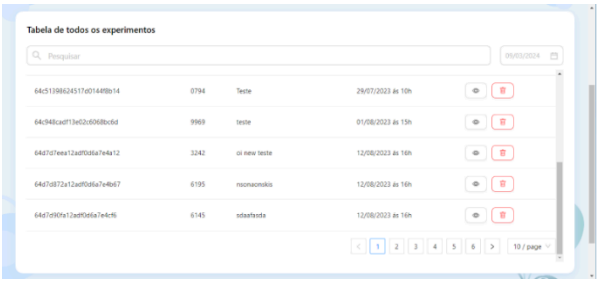


Source: Authors themselves.

Step 4

The all experiments view table allows you to filter by date, name, delete experiment and view experiment (Fig. 6).

Figure 6. Table of experiments with search filters.

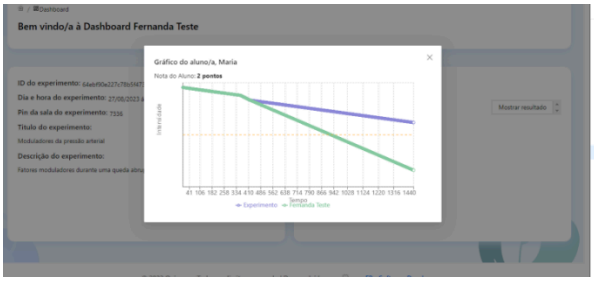


Source: Authors themselves.

Step 5
The teacher can view information about the experiment room and a list with the names of all the students who participated and performed the experiment. By clicking on 'Show result' it is

possible to see the student's result in a graph and grade (Fig. 7).

Figure 7. Student result in graph and grade

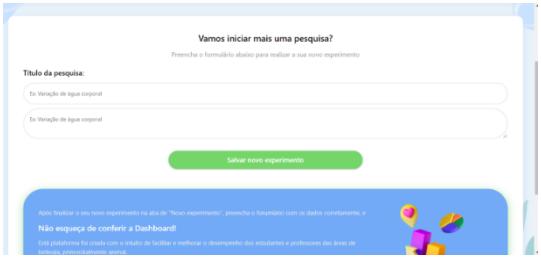


Source: Authors themselves..

Step 6

In 'New experiment', the Professor is redirected to the page that creates new experiments. Simply fill in the Title and description (Fig. 8).

Figure 8. New experiment creation section.



Source: Authors themselves.

Step 7

After creating a new experiment, the teacher is taken to the room where he is given a random pin so that other students can access the experiment (Fig.9).

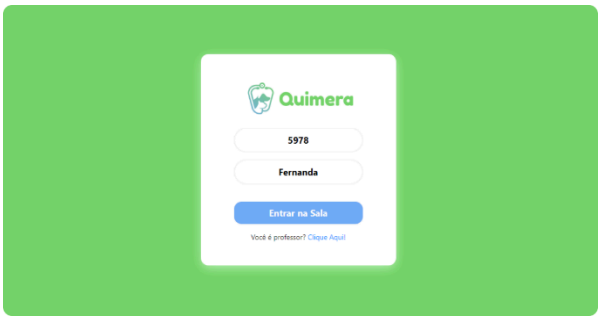
Figure 9. Generating a code for students to log in to a new activity.

Source: Authors themselves.

Step 8

Aluno entra na sala com o pin fornecido pelo professor e seu nome (Fig. 10).

Figure 10. Student access to the activity generated by the teacher.

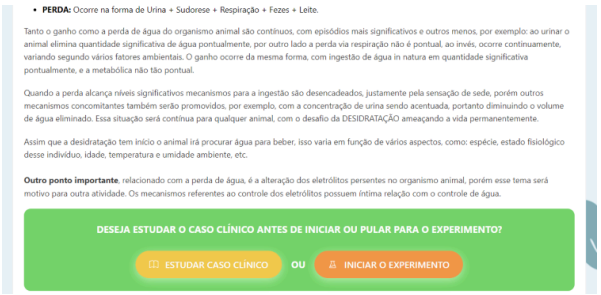


Source: Authors themselves.

Step 9

The student accesses the introductory room in which he performs the experiment and chooses between studying the clinical case or starting the experiment itself (Fig. 11).

Figure 11. Student access to the activity introduction.



Source: Authors themselves.

Step 10

Student's the clinical case of the animal before starting the experiment (Fig. 12).

Figure 12. Clinical case of the activity.

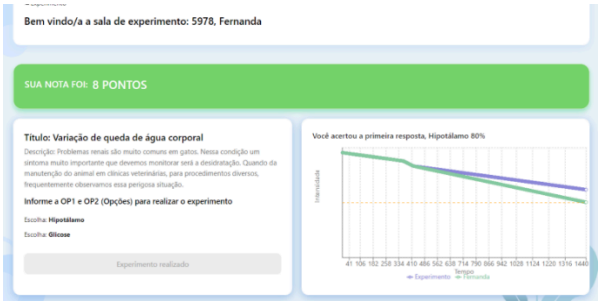
Source: Authors themselves.

Step 11

After the student has completed the study, he or she is redirected to the experiment page, where he or she must choose the two correct hormones that will solve the animal's problem. If the choice

is incorrect, the graph will show the consequences for the animal's health. The final result is only presented after the teacher grants access, from which point the student will be able to view the result, in addition to his or her final grade for the experiment (Fig. 13).

Figure 13. Graphical indication of the correct result compared to the student's choice.



Source: Authors themselves.

Conclusion

The Quimera platform demonstrates the potential of computational simulations and gamified learning in addressing ethical and logistical concerns in animal physiology education. This innovative approach supports active learning, ethical practices, and sustainable alternatives for biological sciences education.

References

DE NORONHA, D.X.. EscapeLab: a virtual learning object for teaching chemistry using Unity 3D. Computer Engineering, 2019. Accessed: March. 2023.

RHARDER, B. Nicole et al. Use of Simulation in Teaching and Learning in Health Sciences: a systematic review. Journal Of Nursing Education, [S.L.], v. 49, n. 1, p. 23-28, Jan. 2010.
Accessed: March. 2023.

Gama, N., & Campos, M. L. M. (2018). Understanding Model-View-Controller (MVC) Architecture in ASP.NET Core. In Advanced Applications of MVC Framework Using ASP.NET Core (pp. 1-16). IGI Global.
Accessed: March. 2023.

Richardson, L., & Amundsen, M. (2013). RESTful Web APIs: Services for a Changing World. "O'Reilly Media, Inc.". Accessed on: March. 2023. Available at: https://www.oreilly.com/library/view/restful-web-apis/9781449359728/

Redmond, E., & Wilson, J. R. (2012). Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. Pragmatic Bookshelf. Accessed on: January 1, 2021. Available at: https://pragprog.com/titles/rwdata/seven-databases-in-seven-weeks/