UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

Sistema de reservas de laboratorios para el departamento de LCC
Laboratory reservation system for the LCC department

Realizado por
Tom van Greevenbroek

Tutorizado por
Gabriel Jesús Luque Polo
Francisco Luna Valero

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, septiembre de 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Sistema de reservas de laboratorio para el departamento de LCC

Laboratory reservation system for the LCC department

Realizado por
Tom van Greevenbroek

Tutorizado por
Gabriel Jesús Luque Polo
Francisco Luna Valero

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2023

Fecha defensa: septiembre de 2023

Abstract

This project presents the development and implementation of a laboratory reservation system tailored to the needs of the Language and Computer Science (LCC in Spanish) department at the University of Málaga. The system's primary objective is to streamline the allocation of laboratories to different academic groups while adhering to a set of predefined constraints.

The system allows administrators to import reservation lists efficiently, enabling a centralized approach to reduce scheduling conflicts and optimize resource allocation.

One of the key features of this system is its ability to handle complex constraints. Teachers can specify their requirements, such as preferred laboratories, location, equipment availability, and time slots. The system then intelligently assigns laboratories to groups, taking into account these constraints, ensuring a fair and efficient allocation process.

To accomplish this, the project utilizes task schedulers, genetic algorithms, and user-friendly interfaces. It also incorporates a Docker solution, along with the necessary manuals for deployment and usage.

In conclusion, this laboratory reservation system offers a comprehensive solution to the complex task of laboratory allocation, benefiting both teachers and administrators by optimizing resource utilization and minimizing conflicts.

Keywords: reservation system, resource allocation, genetic algorithms, task scheduler

Resumen

Este proyecto presenta el desarrollo e implementación de un sistema de reserva de laboratorios adaptado a las necesidades del departamento de Lenguajes y Ciencias de la Computación (LCC) de la Universidad de Málaga. El objetivo principal del sistema es optimizar la asignación de laboratorios a diferentes grupos académicos respetando al mismo tiempo un conjunto de restricciones predefinidas.

El sistema permite a los administradores importar listas de reservas de manera eficiente, lo que facilita un enfoque centralizado para reducir los conflictos de reservas y optimizar la asignación de recursos.

Una de las características clave de este sistema es su capacidad para manejar restricciones complejas. Los profesores pueden especificar sus requisitos, como laboratorios preferidos, ubicación, disponibilidad de equipos y franjas horarias. Luego, el sistema asigna de manera inteligente los laboratorios a los grupos, teniendo en cuenta estas limitaciones, garantizando así un proceso de asignación justo y eficiente.

Para lograr esto, el proyecto utiliza programadores de tareas, algoritmos genéticos e interfaces de usuario amigables. También incorpora una solución Docker, junto con los manuales necesarios para su despliegue y uso.

En conclusión, este sistema de reserva de laboratorios ofrece una solución integral a la compleja tarea de asignación de laboratorios, beneficiando tanto a los profesores como a los administradores al optimizar la utilización de recursos y minimizar los conflictos.

Palabras Clave: sistema de reservas, asignación de recursos, algoritmos genéticos, programador de tareas

Contents

1 Introduction

In an era driven by technology and innovation, the effective management of university resources has become paramount. The Language and Computer Science (LCC) department at our institution is no exception, facing the challenge of optimizing laboratory allocation while accommodating various constraints. This project endeavors to address this challenge through the development and implementation of a Laboratory Reservation System tailored to the unique needs of our department.

This introduction chapter provides a holistic view of the project, starting with the motivation behind its inception, followed by the clear objectives it seeks to achieve and the methodology to do so. Additionally, it outlines the structure of this document, offering readers a roadmap for navigating through the various components and insights that make up this endeavor. Through the collaborative efforts of teachers, administrators, and technology, this system aims to revolutionize the way laboratory resources are allocated within the LCC department, leading to increased efficiency and reduced conflicts.

1.1 Motivation

The Department of Languages and Computer Sciences (LCC) at the University of Malaga (UMA) stands as one of the institution's largest academic departments, boasting a substantial faculty and a diverse array of courses. With the responsibility of managing 12 laboratories within the Technology Complex, the LCC department caters to the instructional needs of nearly 100 different subjects spanning both the Technical School of Computer Engineering and the Technical School of Telecommunications Engineering. These subjects, further divided into numerous small groups, predominantly rely on laboratory facilities for effective teaching and learning experiences.

Presently, the department operates a reservation platform [32] that allows users to request

laboratory reservations. However, the crucial assignment of laboratory resources remains a manual endeavor, undertaken by department technicians. This manual assignment process poses significant challenges due to the substantial volume of reservation requests, the necessity to adhere to multiple constraints (such as scheduling and laboratory type), and the desire to incorporate specific assignment preferences (such as consistently assigning the same laboratory to a given group or situating small groups from the same larger group in adjacent laboratories).

The motivation behind this project is, therefore, clear and compelling. The current manual allocation process not only proves intricate and time-consuming but also presents opportunities for inefficiencies and human errors. Automating the laboratory reservation and allocation process stands to greatly enhance departmental management. By deploying an intelligent reservation system, we aim to alleviate the department staff from the burden of this complex task, allowing them to redirect their efforts toward more productive and strategic activities.

This project seeks to harness the power of technology to optimize resource utilization, minimize scheduling conflicts, and provide a seamless laboratory allocation experience for both faculty and students. In doing so, it aligns with the broader mission of the LCC department to deliver excellence in education while embracing efficiency-enhancing solutions.

1.2 Objectives

The overarching objective of this project is to address the pressing needs of the Department of Languages and Computer Sciences (LCC) at the University of Malaga (UMA). Specifically, we aim to develop a web-based laboratory reservation application designed to intelligently allocate laboratory classrooms to small groups. This allocation process, as previously noted, is intricate and challenging due to the substantial volume of reservation requests and the array of constraints that must be adhered to, some of which are mandatory, while others are desirable for optimizing resource allocation. The key objectives of this project include:

1. **Automated Allocation**: Develop a robust and efficient system capable of automating the allocation of laboratory classrooms to small groups based on a thorough analysis of incoming reservation requests. This automation will significantly reduce the burden of manual allocation tasks currently borne by department technicians.

2. **Integration and Data Exchange**: Ensure seamless interaction with existing systems by enabling the import and export of reservation data in various formats. The system should support the loading of files containing essential information and facilitate their export in the specific format required by external visualization systems.

3. **User-Friendly Interface**: Create an intuitive and user-friendly interface that allows users to view and interact with reservation information effortlessly. The system will highlight any detected problems, enabling users to make necessary modifications to resolve issues.

4. **Deployment Efficiency**: Implement container-based technologies to facilitate the deployment of the developed tool on the department's servers. This approach will ensure compatibility with existing services while maintaining operational efficiency.

By achieving these objectives, this project aspires to provide the LCC department with a cutting-edge reservation system that not only streamlines the allocation process but also enhances the overall management of laboratory resources. It aims to empower administrators, teachers, and students with a powerful tool that optimizes resource utilization and minimizes conflicts, ultimately contributing to the department's mission of delivering high-quality resource management.

1.3 Methodology

The methodology is crucial in ensuring the successful and organized execution of the project. We will be following the agile **Kanban methodology** [22], which emphasizes iterative development and employs a set of task status boards for efficient project management. We utilize multiple task status boards to provide a clear overview of project progress, as you can see in Figure 1.

These boards serve different purposes, including:

- **Backlog**: Used for defining all pending tasks and requirements.
- **To Do**: Tasks that are urgent or represent broader functionalities to be implemented.
- **In Progress**: Tasks that are actively being worked on.

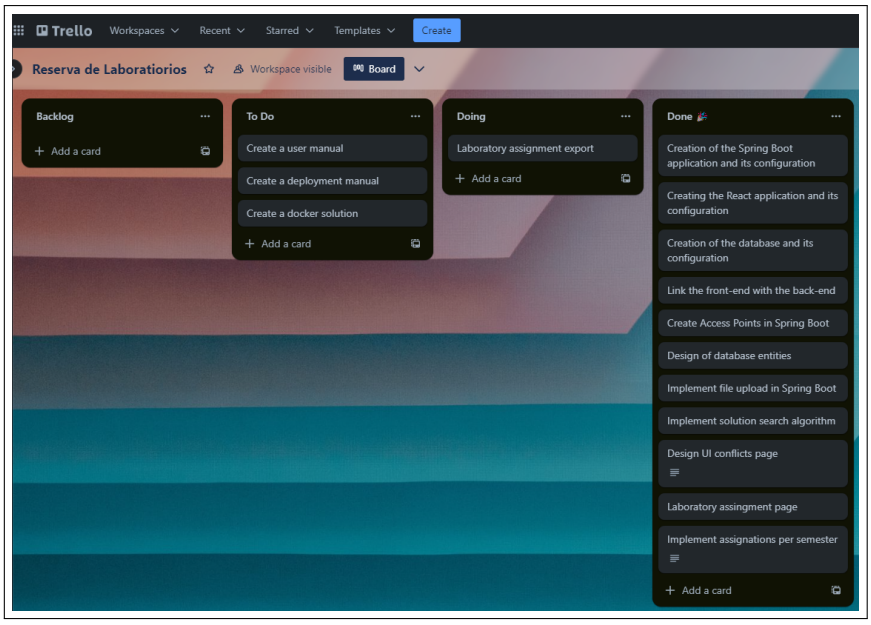


Figure 1: Trello Kanban board

- **Done**: Tasks that have been completed and meet the specified requirements.

For each of the tasks on our board, we follow a development process that adheres to the principles of the following steps:

1. Creation of tasks based on the defined and/or found requirements.
2. Prioritization of tasks by assigning them to the “to do” board.
3. Selecting a task to start with and moving it to “doing.”
4. Writing the necessary code for that task.
5. Creating tests for the selected task, when applicable.
6. Verifying that the requirements of the task have been met.
7. Repeating the process from the beginning.

1.4 Structure of the Document

This document is organized into several chapters, each contributing to a comprehensive exploration of the Laboratory Reservation System for the Department of Languages and Computer Sciences. The following chapters outline the content and focus of each section:

Chapter 1: Introduction sets the stage for the project, presenting the motivation, objectives, methodology, and an overview of the document's structure.

Chapter 2: Technologies and Tools delves into the technological landscape that underpins the project. It discusses the tools, frameworks, and technologies employed in the development of the reservation system.

Chapter 3: Analysis conducts a comprehensive examination of the problem, reservation workflows, CSV file examples, data types, CSV export formats, functional requirements, non-functional requirements, and use cases.

Chapter 4: Modeling and Design details the architectural and design aspects of the reservation system, presenting the system's database schema and user interface designs.

Chapter 5: Development offers an in-depth account of the system's development process, including final user interface design, coding, configuration, testing and deployment.

Chapter 6: Conclusions and Future Work summarizes the project's outcomes and achievements, reflecting on the impact of the reservation system, lessons learned, and potential avenues for future enhancements and research. This is also available in Spanish in **Chapter 7**.

Additionally, this document includes an appendix housing supplementary materials, including the **Deployment Manual** (Appendix A) and **User Manual** (Appendix B). These manuals provide detailed instructions for administrators and users, respectively, to effectively utilize and deploy the reservation system, and are written in the language of the users, which is Spanish.

The structured organization of this document guides readers through the project's various phases, from its introduction (Chapter 1) to its technical details (Chapters 2-5), concluding with insights and future prospects (Chapter 6).

2

Technologies and Tools

The successful development of the Laboratory Reservation System for the Department of Languages and Computer Sciences relies on a carefully chosen array of technologies and tools. This section provides an overview of the software stacks and development environments employed throughout the project. These technologies, meticulously selected to cater to specific project requirements, played pivotal roles in shaping the system's functionality and usability.

2.1 Front-end Technologies

The front-end of the Laboratory Reservation System is built upon a modern and responsive web application framework. The key technologies and tools used in this domain include:

2.1.1 ReactJS

ReactJS [24], developed and maintained by Facebook, is a powerful and widely adopted JavaScript library for building dynamic user interfaces. It has gained immense popularity due to its component-based architecture and the efficient Virtual DOM (Document Object Model) it employs for rendering. ReactJS allows developers to create reusable UI components, making it easier to manage complex user interfaces and promote code reusability.

React's ecosystem is rich and includes libraries such as React Router for routing, Redux for state management, and Material UI for pre-designed, customizable UI components, among others. Its extensive community support, coupled with a large number of third-party libraries and tools, makes ReactJS an excellent choice for building modern and responsive web applications.

One of React's distinctive features is its ability to efficiently update only the parts of the DOM that have changed, thanks to its Virtual DOM. This optimization results in improved performance, especially in applications with frequent user interactions and updates. React also provides a declarative approach to UI development, enabling developers to describe how the UI should look for any given state, and React takes care of updating the UI when the state changes.

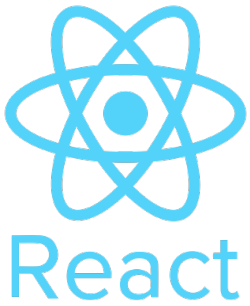


Figure 2: React logo

2.1.2 Vite

Vite [33] is a modern build tool designed for JavaScript and TypeScript projects. What sets Vite apart is its remarkable speed and efficiency during development.

It accomplishes this by using an innovative approach that leverages native ES modules to deliver faster builds and a smoother development experience. Vite's development server starts swiftly, resulting in quick page reloads and near-instantaneous updates when you make changes to your code.

Figure 3: Vite logo

2.1.3 Material UI

Material UI [30] is a renowned UI component library for React applications that follows the Material Design guidelines established by Google. This library provides a comprehensive set of pre-designed and customizable UI components, ranging from buttons and forms to navigation bars and data tables. Material UI's components are visually appealing and offer a consistent look and feel, making it easier for developers to create aesthetically pleasing user interfaces.

One of the standout features of Material UI is its flexibility. Developers can customize the appearance and behavior of Material UI components to align with the project's specific design requirements. This level of customization ensures that the UI remains cohesive with

the overall design language of the application.

2.1.4 TypeScript

TypeScript [29] is a statically typed superset of JavaScript that brings strong typing and enhanced tooling to the world of web development. Developed and maintained by Microsoft, TypeScript is designed to improve code quality, catch errors during development, and enhance the development experience.

One of TypeScript's primary advantages is its static type checking. Developers can declare variable types, function parameter types, and return types, which the TypeScript compiler enforces. This early error detection helps prevent runtime errors and ensures code behaves as intended. TypeScript also provides code editors with better context-aware auto-completion and code navigation, enhancing productivity.

2.1.5 Additional front-end libraries

In addition to the core front-end technologies mentioned above, the Laboratory Reservation System leverages several additional libraries to enhance its functionality and development process:

- **dayjs** [2] is a lightweight JavaScript library for handling date and time operations. It offers an alternative to the more extensive moment.js library and provides a simple API for manipulating and formatting dates and times.
- **dotenv** [15] is a Node.js module used for managing environment variables in web applications. It simplifies the process of configuring and accessing environment-specific settings, enhancing the system's flexibility and security.
- **react-csv** [16] and **react-papaparse** [17] are libraries designed to handle CSV (Comma-Separated Values) data in React applications. They enable efficient parsing and manipulation of CSV files, which can be particularly useful for importing and exporting data within the Laboratory Reservation System.
- **Prettier** [21] is an opinionated code formatter that enforces consistent code style. It automatically formats code to adhere to predefined rules, reducing formatting-related

inconsistencies and enhancing code readability.

- **ESLint** [6] is a versatile linting utility for JavaScript and TypeScript projects. It analyzes code for potential issues and enforces coding standards and best practices. Integrating ESLint into the development workflow ensures code quality and maintainability.

These additional libraries contribute to the overall robustness and functionality of the front-end components of the reservation system, improving both user experience and code quality.

2.2 Back-end Technologies

The back-end of the reservation system is powered by robust Java-based technologies and a relational database management system. The key components in this category are:

2.2.1 Java

Java [18] is a versatile and widely-used programming language known for its portability, robustness, and extensive community support. It is a particularly suitable choice for developing server-side applications and web services. Java's "write once, run anywhere" philosophy allows developers to build applications that can run on various platforms without modification. This portability is achieved through the use of the Java Virtual Machine (JVM), which executes Java bytecode.

The back-end of the Laboratory Reservation System relies on Java to provide the core functionality of the application. Java's strong typing, object-oriented nature, and comprehensive standard library make it well-suited for building complex systems. Additionally, Java boasts a vast ecosystem of libraries and frameworks that aid in various aspects of development, from database connectivity to web service creation.

Figure 4: Java logo

2.2.2 Spring Boot (Maven)

Spring Boot [25] is a powerful and widely adopted framework for building Java applications, and **Maven** [1] is the chosen build tool for managing dependencies and building the project. Spring Boot simplifies the development of Java-based web applications by providing a set of pre-configured templates and conventions. It reduces the need for extensive boilerplate code and streamlines the development process.

One of Spring Boot's standout features is its built-in support for creating RESTful web services. It simplifies the creation of APIs and allows for the seamless integration of various components, such as databases and security features. Spring Boot's reliance on convention over configuration and its extensive ecosystem of add-ons make it an excellent choice for rapidly developing robust back-end services.

Figure 5: Spring Boot logo

2.2.3 PostgreSQL

PostgreSQL [20] is a powerful open-source relational database management system (RDBMS) known for its reliability, extensibility, and advanced features. It is a top choice for web applications that require a robust and scalable database back-end. PostgreSQL offers support for complex data types, indexing, and querying capabilities, making it suitable for handling various types of data efficiently.

2.2.4 Additional Back-end Libraries

In addition to the core back-end technologies, the Laboratory Reservation System incorporates several additional libraries and tools to enhance its functionality and development process:

- **FlywayDB** [7] is a database migration tool that simplifies the management of database schema versions. It ensures that database changes are versioned and applied consistently, making it easier to maintain and evolve the database schema as the application evolves.

The Laboratory Reservation System relies on PostgreSQL as its database management system to store and manage data related to laboratory reservations. PostgreSQL's support for ACID (Atomicity, Consistency, Isolation, Durability) transactions ensures data integrity and consistency, even in high-concurrency environments.

Figure 6: PostgreSQL logo

- **Lombok** [13] is a Java library that reduces boilerplate code by automatically generating getters, setters, and other common code constructs. It simplifies code development and improves code readability.
- **springdoc-openapi** [26] is a library used to generate OpenAPI documentation for RESTful APIs implemented with Spring Boot. It provides a clear and interactive API documentation that aids developers in understanding and utilizing the available endpoints.
- **JavaFaker** [4] is a library for generating random data, such as names, addresses, and more. It is particularly useful for populating test databases with realistic but non-sensitive data.
- **Testcontainers** [27] and **JUnit Jupiter** [12] are tools used for containerized testing and unit testing in Java applications. Testcontainers simplifies the setup of testing environments by running services (e.g., databases) in Docker containers, while JUnit Jupiter is a powerful testing framework for writing unit tests in Java.
- **SLF4J** [23] serves as a logging facade for Java applications, providing a consistent and abstracted logging API.

These additional libraries and tools, including monitoring solutions, contribute to the robustness, maintainability, and functionality of the back-end components of the Laboratory Reservation System.

2.3 Development Tools

In order to help us with the development of the mobile application we have used different tools. Below we will see each one in detail:

2.3.1 Visual Studio Code

Visual Studio Code [14] is a versatile and widely adopted code editor developed by Microsoft. It has gained immense popularity among developers due to its extensibility, robustness, and a rich ecosystem of extensions. Visual Studio Code supports a wide range of programming languages and offers powerful features such as code autocompletion, integrated version control, and a vibrant developer community. It is an excellent choice for coding and debugging in various programming languages.

2.3.2 DBeaver

DBeaver [3] is a highly regarded open-source database tool that simplifies database management tasks. It supports various databases, including PostgreSQL, MySQL, and Oracle, making it a valuable asset for developers and database administrators. DBeaver offers features such as schema exploration, SQL code generation, and data visualization, making it an essential tool for working with databases efficiently.

2.3.3 IntelliJ IDEA

IntelliJ IDEA [11] is a widely used integrated development environment (IDE) for Java and other programming languages. Developed by JetBrains, IntelliJ IDEA is known for its intelligent code assistance, code analysis, and a plethora of productivity features. It streamlines the development process by providing advanced debugging tools, intelligent refactoring, and seamless integration with build tools like Maven. IntelliJ IDEA is the preferred choice for Java development in the Laboratory Reservation System.

2.3.4 Docker Desktop

Docker Desktop [5] is a powerful platform for developing, shipping, and running applications in containers. It simplifies the deployment of applications by encapsulating them in

lightweight, isolated containers. Docker Desktop provides a user-friendly interface for managing containers and orchestrating multi-container applications. It is an essential tool for ensuring consistency and reliability in deploying the Laboratory Reservation System and its associated components.

2.3.5 GitHub Copilot

GitHub Copilot [10] is an innovative code completion and collaboration tool developed by GitHub in collaboration with OpenAI. It leverages the power of artificial intelligence to assist developers in writing code more efficiently and with greater accuracy. GitHub Copilot provides real-time suggestions and auto-completions for code, reducing coding errors and speeding up the development process. It supports a wide range of programming languages and frameworks, making it a valuable addition to the development toolkit for the Laboratory Reservation System.

2.4 Management Tools

The development process was facilitated by various tools, version control systems, and project management platforms. These include:

2.4.1 Git and GitHub

Git [8] is a widely used version control system that facilitates collaborative software development. **GitHub** [9], a web-based platform, enhances Git by providing features such as repository hosting, code review, issue tracking, and collaborative tools.

2.4.2 Trello

Trello [28] is a popular project management and collaboration tool that uses boards, lists, and cards to organize tasks and projects. It offers a visual and flexible approach to managing work, making it easy to plan, track progress, and assign tasks. Trello's user-friendly interface and integration capabilities help streamline project management activities for the Laboratory Reservation System.

3
Analysis

The success of the Laboratory Reservation System depends on a comprehensive understanding of the project requirements. In this section, we delve into the essential requirements that shape the system's design and functionality. These requirements serve as the foundation upon which the entire development process is built. A clear grasp of these requirements ensures that the system aligns with the department's needs and facilitates efficient laboratory reservation management.

3.1 Problem Analysis

In this section, we will provide an in-depth analysis of the reservation system, covering various aspects of its functionality and constraints.

3.1.1 Reservation Workflow

The reservation system operates as follows:

- Professors submit reservation requests through another system.
- Reservation request are emailed to the reservation manager (the user of our proposed system).
- The system user creates a CSV file containing all reservations requests and imports it into the system, with the header format as specified in the Table 1 that in our context will be mapped to our systems reservation model via the *ReservationDto* that we see in Table 2.

Table 1: CSV import file header format

código	tipo	titulación	asignatura	curso	grupo
subgrupo	profesor	email	departamento	inicio	fin
franja	localización	aula	num_alumnos	tipo_reserva	responsable
telefono	horario	sistema_operativo	características		

Table 2: ReservationDto format

publicId	teachingType	degreeName	subjectName	subjectCourse	subjectGroup
subjectSubgroup	professorName	professorEmail	departmentName	startDate	endDate
dayAndTimeSlot	location	laboratoryPreference	studentsNumber	type	responsibleName
responsiblePhone	schedule	operatingSystem	additionalEquipment		

3.1.2 CSV File Example

An example of both a creation and cancellation of reservation is to be seen from Table 3 to Table 7. The first row are the headers that were displayed in Table 1, the second row represent a new reservation, and the last row represents the cancellation of the previous row.

Table 3: CSV example file

código	tipo	titulación	asignatura	curso	grupo
49601	Docencia reglada	Graduado/a en Ingeniería de Computadores	Fundamentos de la Programación (306-5102-19-0104)	2	A
45000					

Table 4: CSV example file (continued)

subgrupo	profesor	email	departamento	inicio	fin
A3	Gabriel Luque	xxxxxxx@uma.es	LCC	16/09/2022	16/12/2022

Table 5: CSV example file (continued)

franja	localización	aula	num_alumnos
Viernes (08:45 - 10:30)	CTE - LCC Laboratorios	Lab-01	30

Table 6: CSV example file (continued)

tipo_reserva	responsable	telefono	horario
Reserva semanal	Francisco Luna	XXXXXXXXX	Preferente
Cancelación solicitud			49601

Table 7: CSV example file (continued)

sistema_operativo	características
Windows	Ninguno

3.1.3 Types

The reservation system supports various types of reservations for a given semester, with the flexibility to accommodate dynamic fields as well. Here, we will categorize the reservation types and provide insights into their characteristics.

Dynamic Fields: Some reservation fields can vary based on user requirements, making them dynamic in nature. These fields include **location**, **operating system**, and **additional equipment**.

Pre-Defined Fields:

1. **Reservation Type:**
    - **WEEKLY (*Reserva semanal*):** Long-term reservations.
    - **DAY (*Reserva de lista de días*):** Multiple rows in the CSV with matching *publicId*, each representing a short-term reservation for a day.

- **CANCELLATION (*Cancelación solicitud*):** Cancellation of a set of reservations where the *publicId* matches the *schedule* field in the CSV file. These reservations normally only contain the *publicId*, *type*, and *schedule* fields.

2. **Schedule Type:**

   - **PREFERRED (*Preferente*):** Reservations marked as *Preferente* are given priority and are considered primary choices for allocation.
   - **ALTERNATIVE (*Alternativo*):** Reservations with an *Alternativo* schedule type provide an alternative option in case of conflicts with preferred reservations.

3. **Teaching Type:**

   - **REGULATED (*Docencia reglada*):** This teaching type refers to regulated or formal teaching methods, such as traditional classroom lectures and structured educational programs.

4. **Days of the Week Type:**

   - **MONDAY (*Lunes*)**
   - **TUESDAY (*Martes*)**
   - **WEDNESDAY (*Miércoles*)**
   - **THURSDAY (*Jueves*)**
   - **FRIDAY (*Viernes*)**
   - **SATURDAY (*Sábado*)**
   - **SUNDAY (*Domingo*)**

5. **Anything Type:**

   - The "AnythingType" includes three values that may appear in the CSV file but can be translated to a null value in our system.
     - **NOTHING(*Ninguno*)**
     - **ANYTHING(*Cualquiera*)**
     - **ANY LABORATORY(*Cualquier laboratorio de LCC*)**

3.1.4 Export CSV Format

The format of the CSV to export is as shown in the Table 8, and includes the headers in the file too. The target audience for exported data is administrators who use the file for further processing.

Table 8: CSV export format

centro	titulación	asignatura	curso	grupo
fecha_inicio	fecha_fin	dia	hora_inicio	hora_fin
aula	descripción	tipo		

This format is of the exported file was specified by the administrators, each field being described as:

- **centro**: This field represents the center or location of the reservation.
- **titulación**: This field stands for degree of the reservation.
- **asignatura**: This field represents the subject of the reservation.
- **curso**: This field represents the academic course level (from 1 to 4) associated with a subject in the reservation.
- **grupo**: This field pertains to the group and subgroup classification of a subject within the reservation.
- **fecha_inicio**: This field represents the start date of the reservation.
- **fecha_fin**: This field denotes the end date of the reservation.
- **dia**: This field stands for the day of the week of the reservation.
- **hora_inicio**: This field represents the start time of the reservation for every specified day of the week, between the start date and end date.
- **hora_fin**: This field denotes the end time of the reservation for every specified day of the week, between the start date and end date.

- **aula**: This field represents the assigned laboratory. This will be the most important column since it will indicate what lab has been assigned to which reservation.
- **descripción**: This field stands for description. It is left out as empty, to be filled out by the administrators if needed.
- **tipo**: This field represents the reservation type, which will be either "WEEKLY" or "DAY".

3.2 Functional Requirements

We outline the functional requirements that define the core capabilities of the Laboratory Reservation System. These requirements specify the system's ability to handle key tasks such as loading reservation files, identifying conflicts, and more. These functional requirements serve as the foundation for building a system that effectively meets the needs of the department and streamlines laboratory reservation management.

- **FR-1**: The system must be able to load the reservations file.
- **FR-2**: The system must be able to delete canceled reservations (this will be done on file upload).
- **FR-3**: The system must be able to show reservations.
- **FR-4**: The system must be able to identify reservations that cause conflicts.
- **FR-5**: The system must be able to assign laboratories based on reservations.
- **FR-6**: The system must be able to show the assignment of laboratories.
- **FR-7**: The system must be able to export the mapping to a pre-established format.
- **FR-8**: The system must allow several differentiated reservation periods.

3.3 Non-Functional Requirements

We delve into the quality attributes and constraints that shape the user experience and system behavior. These non-functional requirements encompass aspects such as device compatibility and file format standards. Adhering to these requirements ensures that the Laboratory

Reservation System not only functions effectively but also delivers a user-friendly and reliable experience for its users.

- **NFR-1**: The user must be able to access the system from any web browser.
- **NFR-2**: The user will not be able to access the system from a mobile device.
- **NFR-3**: The format of the file to be uploaded will be CSV type.

3.4 Use Cases

The use cases for the reservation system provide a comprehensive framework for understanding how the system functions and interacts with its users. These use cases outline various scenarios and actions that users can perform within the system, encompassing tasks such as creating reservations, managing conflicts, and generating reports. By defining these use cases, we can gain a clear understanding of the system's functionality and ensure it meets the needs of its users while effectively managing resources and reservations.

Table 9: UC01 - Create a semester

Title	Create a Semester
Description	This use case allows a user to create a new semester.
Requisites	FR-8
Pre-condition	The user is on the landing page or the semester page.
Post-condition	A new semester is created in the system and marked as selected.
Main scenario	1. User clicks on the "CREAR CUATRIMESTRE" button. 2. A dialog box opens. 3. User enters a valid year in the "Año de inicio" field. 4. User selects a "Periodo" (either "Primero" or "Segundo"). 5. User presses the "CREAR" button to save the semester.

Alternative scenario	3a. User presses the “CANCELAR” button to abort the operation. 5.a. The system fails to save the new semester due to technical problems or data retrieval issues. 5.a.1. The systems shows an error message and a button to refresh the page.

Table 10: UC02 - Import a CSV file

Title	Import a CSV File
Description	A user must be able to import a CSV file into the system.
Requisites	FR-1, FR-2, FR-8, NFR-3
Pre-condition	The user is on the landing page and a semester has been created and selected.
Post-condition	The reservations are loaded in the system, a task to run the algorithm has been created and the user is redirected to the assignments page.
Main scenario	1. User drags-and-drop the CSV file into the designated import component that says “Suelte el archivo CSV aquí o haga clic para cargarlo”. 2. The system saves the reservations and deletes any reservation that is marked as deleted. 3. The system creates and executes a task to run the algorithm for the selected semester. 4. User is redirected to the assignments page and informed about the tasks status.

Alternative scenario	1.a. Instead of drag-and-drop, the user clicks on the import button. 1.a.1 User clicks on the the designated import component that says “Suelte el archivo CSV aquí o haga clic para cargarlo”. 1.a.2 User selects the file on its system. 1.a.3 Continue to step 2. 2.a The imported CSV file contains errors. 2.a.1 The system generates a list of errors for the user to review. 2.a.2 Continue to step 1. 2.b There is already an imported list of reservations for the selected semester. 2.b.1 The system will prompt a confirmation dialog to override the current reservations and assignments. 2.b.2.a.1 The user presses the “CONFIRMAR” button. 2.b.2.a.2 The dialog box is closed. 2.b.2.a.3 The system deletes any reservation and assignment related to the selected semester. 2.b.2.b.1 The user presses the “CANCEL” button. 2.b.2.b.2 The dialog box is closed.

Table 11: UC03 - See the assignation conflicts list

Title	See the assignation conflicts list
Description	Users would want to see a list of all conflicts of the assignations.
Requisites	FR-4
Pre-condition	At least one CSV file has been imported for the selected semester, and the algorithm has been successfully executed.

Post-condition	The user is presented with a list of assignment conflicts and has the option to view the details of individual reservations.
Main scenario	1. User navigates to the “Asignaciones” page from the left menu. 2. The system displays a list of assignment conflicts.
Alternative scenario	2.a. The system fails to load the list due to technical problems or data retrieval issues. 2.a.1. The systems shows an error message and a button to refresh the page. 2.b. The user wants to one of the reservations that is causing the conflict. 2.b.1. User clicks on the ID of one of the conflicting reservations in the list. 2.b.2. The system redirects the user to the reservations page. 2.b.3. The system will highlight the reservation that was selected previously.

Table 12: UC04 - See the assignation conflicts calendar

Title	See the assignation conflicts calendar
Description	Users can access a dedicated page from the system’s navigation menu to view an assignment calendar view provides a visual representation of assignment conflicts.
Requisites	FR-6
Pre-condition	At least one CSV file has been imported for the selected semester, and the algorithm has been successfully executed.
Post-condition	The user can view assignment per laboratory in a weekly calendar, also seeing where there are conflicts.

Main scenario	1. User navigates to the “Asignaciones” page from the left menu. 2. The system initially presents assignment conflicts in a list format. 3. The user clicks the dedicated toggle button “CALENDARIO” designed to switch to the calendar view. 4. The system transforms the display into a calendar view, visually representing the assignment conflicts.
Alternative scenario	2.a. The system fails to load the list due to technical problems or data retrieval issues. 2.a.1. The systems shows an error message and a button to refresh the page.

Table 13: UC05 - Export reservation assignments

Title	Export reservation assignments
Description	Users can initiate the export of reservation assignments from the “Asignaciones” (Assignments) page.
Requisites	FR-7
Pre-condition	At least one CSV file has been imported for the selected semester, and the algorithm has been successfully executed.
Post-condition	The exported data is downloaded on the users system.
Main scenario	1. User navigates to the “Asignaciones” page from the left menu. 2. The user clicks the “EXPORTAR” button. 3. The file is downloaded into the users system.
Alternative scenario	

Table 14: UC06 - Show the list of imported reservations

Title	Show the list of imported reservations
Description	Users can access the reservations page to view the list of imported reservations.

Requisites	FR-3
Pre-condition	At least one CSV file has been imported for the selected semester, and the algorithm has been successfully executed.
Post-condition	The system displays a table containing information about all imported reservations.
Main scenario	1. User navigates to the “Reservas” page from the left menu. 2. The page displays a table listing all imported reservations.
Alternative scenario	2.a. The system fails to load the list of imported reservations due to technical problems or data retrieval issues. 2.a.1. The systems shows an error message and a button to refresh the page.

Table 15: UC07 - Delete an existing semester

Title	Delete an existing semester
Description	Users have the capability to delete an existing semester from the system. This action triggers a confirmation dialog to ensure that the user is aware of the consequences, as deleting a semester will remove all related reservations, assignations, and conflicts.
Requisites	FR-8
Pre-condition	The user is on the “Cuatrimestre” page accessible from the left menu.
Post-condition	Upon confirmation, the system permanently deletes the selected semester along with all associated reservations, assignations, and conflicts.

Main scenario	1. The user locates the semester they want to delete. 2. Next to the selected semester, the user clicks the “BORRAR” button. 3. The system displays a confirmation dialog warning the user about the consequences of deleting the semester. 4. The user confirms their decision by clicking the “CONFIRMAR” button in the confirmation dialog. 5. The dialog box is closed. 6. The system permanently deletes the selected semester and all related reservations, assignations, and conflicts.
Alternative scenario	4.a. The user decides not to delete the semester and clicks the “CANCELAR” button in the confirmation dialog. 4.a.1 The dialog box is closed. 5.a. The system fails to delete the semester due to technical problems or data retrieval issues. 5.a.1. The systems shows an error message and a button to refresh the page.

Table 16: UC08 - Change the selected semester

Title	Change the selected semester
Description	Users have the capability to change the selected semester in the system. The selected semester determines the context for all actions taken in the pages above the divider in the left menu.
Requisites	FR-3
Pre-condition	The user is on the “Cuatrimestre” page accessible from the left menu.
Post-condition	The selected semester is changed, and the system updates the context for various actions and displays accordingly.

Main scenario	1. The user locates the semester they want to set as the selected semester. 2. Next to the selected semester, the user clicks the radio button to mark it as the selected semester. 3. The system updates the selected semester, and the user is now operating within the context of the newly selected semester.
Alternative scenario	3.a. The system fails to select the semester due to technical problems or data retrieval issues. 3.a.1. The systems shows an error message and a button to refresh the page.

Table 17: UC09 - See a laboratory list

Title	See a laboratory list
Description	Users can access and view a list of laboratories available within the system. The list provides essential information about each laboratory.
Requisites	FR-5
Pre-condition	The user is on the “Laboratorios” page accessible from the left menu.
Post-condition	The user can see a table displaying information about all the laboratories within the system.
Main scenario	1. The system displays a table listing all the laboratories.
Alternative scenario	1.a. The system fails to load the list due to technical problems or data retrieval issues. 1.a.1. The systems shows an error message and a button to refresh the page.

Table 18: UC10 - Create a laboratory

Title	Create a laboratory

Description	Users can create a new laboratory within the system. The laboratory creation process involves specifying various attributes and optional configurations.
Requisites	FR-5
Pre-condition	The user is on the “Laboratorios” page accessible from the left menu.
Post-condition	A new laboratory is created in the system with the specified attributes and configurations.
Main scenario	1. The user clicks on the “CREAR” button. 2. A dialog box opens with the following fields: - Nombre (Name) [Required] - Capacidad (Capacity) [Required] - Localización (Location) [Required] - Sistema operativo (Operating System) - Equipo adicional (Additional Equipment) - Laboratorios adyacentes (Adjacent Laboratories, multi-selector of existing laboratories) 3. The user fills in the required fields and any optional configurations. 4. The user clicks the “GUARDAR” button to confirm. 5. The new laboratory is created and added to the list of laboratories.
Alternative scenario	1.a. The user clicks on the “CREAR” button but decides to cancel the operation. 1.a.1. The dialog box is closed, and no new laboratory is created. 5.a. The system fails to save the laboratory due to technical problems or data retrieval issues. 5.a.1. The systems shows an error message and a button to refresh the page.

Table 19: UC11 - Edit a laboratory

Title	Edit a laboratory
Description	Users can edit the attributes and configurations of an existing laboratory within the system. This process allows for updates and modifications to the laboratory's details.
Requisites	FR-5
Pre-condition	The user is on the "Laboratorios" page accessible from the left menu, and there is at least one existing laboratory.
Post-condition	The selected laboratory's attributes and configurations are updated with the specified changes.
Main scenario	1. The user clicks on the "EDITAR" button next to a laboratory. 2. A dialog box opens with the following fields filled out with the selected laboratories details: - Nombre (Name) [Required] - Capacidad (Capacity) [Required] - Localización (Location) [Required] - Sistema operativo (Operating System) - Equipo adicional (Additional Equipment) - Laboratorios adyacentes (Adjacent Laboratories, multi-selector of existing laboratories) 3. The user updates the fields with the desired changes. 4. The user clicks the "GUARDAR" button to confirm the changes. 5. The selected laboratory's attributes and configurations are updated.

Alternative scenario	1.a. The user clicks on the “EDITAR” button but decides to cancel the operation. 1.a.1. The dialog box is closed, and no changes are made to the laboratory. 5.a. The system fails to save the laboratory due to technical problems or data retrieval issues. 5.a.1. The systems shows an error message and a button to refresh the page.

Table 20: UC12 - Delete a laboratory

Title	Delete a laboratory
Description	Users can delete an existing laboratory from the system. This action triggers the removal of all associated reservations, conflicts, and semesters related to the deleted laboratory.
Requisites	FR-5
Pre-condition	The user is on the “Laboratorios” page accessible from the left menu, and there is at least one existing laboratory.
Post-condition	The selected laboratory is permanently deleted, and all associated data, including reservations, conflicts, and semesters, is removed from the system.
Main scenario	1. The user clicks on the “BORRAR” button next to a laboratory. 2. A confirmation dialog box appears, warning the user about the consequences of the deletion. 3. The user confirms the deletion by clicking the “CONFIRMAR” button in the dialog. 4. The selected laboratory and all associated data are permanently removed from the system.

Alternative scenario	3.a. The user decides to cancel the operation. 3.a.1. The dialog box is closed, and no changes are made. 4.a. The system fails to delete the laboratory due to technical problems or data retrieval issues. 4.a.1. The systems shows an error message and a button to refresh the page.

4

Modeling and Design

In this section, we delve into the visual and structural representation of the Laboratory Reservation System. Modeling plays a crucial role in understanding, designing, and communicating various aspects of the system, from its database structure to the user interface. In the following subsections, we will explore the Database Model and UI mock-ups, providing a comprehensive view of the system's architecture, data relationships, and user interactions. Each modeling aspect contributes to the system's clarity, maintainability, and effective development.

4.1 Database Model

The entity-relationship diagram, captured in Figure 7, is the essential components of the Laboratory Reservation System designed for a university setting. It involves several interconnected tables, each serving a distinct purpose in managing academic resources and reservations.

The central table, "reservation," acts as the heart of the system, tracking reservations with specific needs. These reservations are associated with academic degrees, subjects, professors, and responsible individuals. Academic semesters play a crucial role in scheduling, and tasks related to running the algorithm are monitored through the "task" table.

To facilitate efficient management, the system also stores information about academic degrees, subjects, professors, departments, and laboratory facilities in dedicated tables. Additionally, "adjacent laboratory" relationships provide insights into which laboratories are adjacent to each other, offering valuable spatial information.

Furthermore, the system is equipped to store conflicts between reservations and assignments made between reservation and laboratory for a semester. It maintains a detailed history of database schema changes through the "flyway schema history" table.

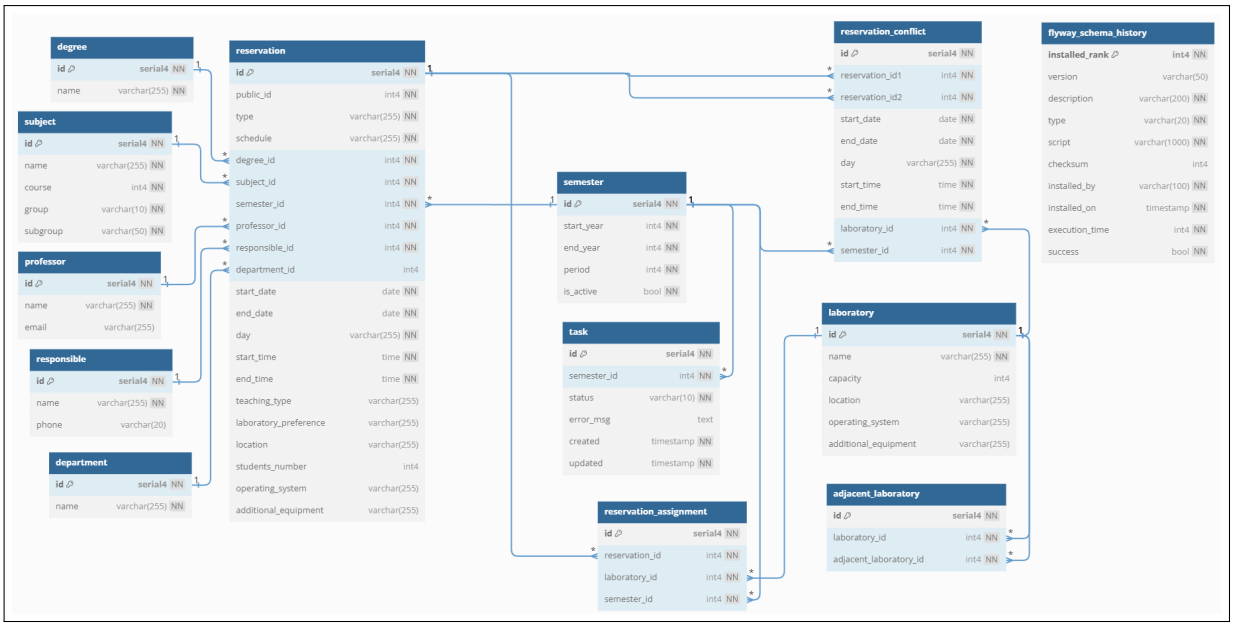


Figure 7: Entity-relationship diagram

4.1.1 Table: *degree*

The "degree" table stores information about academic degrees. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each academic degree.
- **name**: The name of the academic degree.

4.1.2 Table: *subject*

The "subject" table stores information related to academic subjects, within the four year courses, offered within the university. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each subject.
- **name**: The name of the subject or course.
- **course**: An integer representing the course number or level. This is a number between 1 and 4, both inclusive.
- **group**: A varchar field indicating the group to which the subject belongs.

- **subgroup**: A varchar field specifying the subgroup within the subject.

4.1.3 Table: *professor*

The "professor" table stores information about professors within the university. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each professor.
- **name**: The name of the professor.
- **email**: The email address of the professor. Can be null.

4.1.4 Table: *responsible*

The "responsible" table contains information about individuals responsible for the reservation, normally being the professor itself. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each responsible individual.
- **name**: The name of the responsible individual.
- **phone**: The phone number of the responsible individual. Can be null.

4.1.5 Table: *department*

The "department" table holds data related to academic departments within the university. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each department.
- **name**: The name of the department.

4.1.6 Table: *reservation*

The "reservation" table serves as the core of the Laboratory Reservation System, storing information about reservations made for laboratory facilities. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each reservation.

- **public_id**: An integer identifier used for public reference to the reservation.
- **type**: A varchar field specifying the reservation type.
- **schedule**: A varchar field providing details about the reservation schedule. In case of a "CANCELLATION" reservation type, this field would represent the "public_id" to be removed.
- **degree_id**: The identifier of the academic degree associated with the reservation.
- **subject_id**: The identifier of the academic subject for which the reservation is made.
- **semester_id**: The identifier of the academic semester during which the reservation occurs.
- **professor_id**: The identifier of the professor responsible for the reservation.
- **responsible_id**: The identifier of the responsible individual for the reservation.
- **department_id**: The identifier of the academic department. Can be null.
- **start_date**: The start date of the reservation.
- **end_date**: The end date of the reservation.
- **day**: The day of the week for the reservation.
- **start_time**: The start time of the reservation.
- **end_time**: The end time of the reservation.
- **teaching_type**: A varchar field specifying the teaching type. Can be null.
- **laboratory_preference**: Information about laboratory preference. Can be a list of laboratories separated by commas. Can be null.
- **location**: The location of the reservation. Can be null.
- **students_number**: An integer indicating the number of students. Can be null.
- **operating_system**: Information about the required operating system. Can be null.

- **additional_equipment**: Details about any additional equipment needed for the reservation. Can be null.

4.1.7 Table: *semester*

The "semester" table is responsible for storing information related to academic semesters. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each semester.
- **start_year**: An integer representing the starting year of the semester.
- **end_year**: An integer representing the ending year of the semester.
- **period**: An integer specifying the period or term within the academic year. This will be the number 1 or 2.
- **is active**: A boolean indicating whether the semester is currently active. Default is false.

4.1.8 Table: *task*

The "task" table is essential for tracking tasks and their statuses. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each task.
- **semester_id**: The identifier of the academic semester associated with the task.
- **status**: A varchar field indicating the status of the task (QUEUED, RUNNING, COMPLETED, ERROR, OUTDATED).
- **error_msg**: A text field for recording error messages associated with the task. Can be null.
- **created**: A timestamp representing the creation date and time of the task.
- **updated**: A timestamp representing the last update date and time of the task.

4.1.9 Table: *reservation_assignment*

The "reservation assignment" table tracks the assignment of reservations to specific laboratories for each semester. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each reservation assignment.
- **reservation_id**: The identifier of the reservation associated with the assignment.
- **laboratory_id**: The identifier of the laboratory assigned to the reservation.
- **semester_id**: The identifier of the academic semester during which the assignment occurs.

4.1.10 Table: *reservation_conflict*

The "reservation conflict" table is designed to capture conflicts that may arise between reservations, particularly when there is a scheduling overlap. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each reservation conflict.
- **reservation_id1**: The identifier of the first reservation involved in the conflict.
- **reservation_id2**: The identifier of the second reservation involved in the conflict.
- **start_date**: The start date of the conflict.
- **end_date**: The end date of the conflict.
- **day**: The day of the week for the conflict.
- **start_time**: The start time of the conflict.
- **end_time**: The end time of the conflict.
- **laboratory_id**: The identifier of the laboratory associated with the conflict.
- **semester_id**: The identifier of the academic semester during which the conflict occurs.

4.1.11 Table: *laboratory*

The "laboratory" table is responsible for storing information related to the university's laboratory facilities. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each laboratory.
- **name**: The name of the laboratory.
- **capacity**: An integer representing the maximum capacity of the laboratory. Can be null.
- **location**: The location of the laboratory. Can be null.
- **operating_system**: Information about the operating system installed in the laboratory. Can be null.
- **additional_equipment**: Details about any additional equipment available in the laboratory. Can be null.

4.1.12 Table: *adjacent_laboratory*

The "adjacent laboratory" table defines a many to many relationship between laboratories, specifically identifying which laboratories are adjacent to each other. It includes the following attributes:

- **id (Primary Key)**: A unique identifier for each adjacency relationship.
- **laboratory_id**: The identifier of the laboratory to which the adjacency information pertains.
- **adjacent_laboratory_id**: The identifier of the laboratory that is adjacent to the specified laboratory.

4.1.13 Table: *flyway_schema_history*

The "flyway schema history" table is auto-generated by the flywaydb library and serves as a log for tracking the version history of database schema changes. It includes the following attributes:

- **installed rank**: An integer representing the installed rank of a particular schema version.
- **version**: A varchar field that can store the version number of the schema. Can be null.
- **description**: A varchar field providing a description of the schema change.
- **type**: A varchar field specifying the type of schema change.
- **script**: A varchar field containing the SQL script associated with the schema change.
- **checksum**: An integer representing the checksum of the SQL script (used for verification). Can be null.
- **installed_by**: A varchar field indicating the user who applied the schema change.
- **installed_on**: A timestamp representing the date and time when the schema change was applied.
- **execution_time**: An integer specifying the time taken to execute the schema change.
- **success**: A boolean field indicating the success or failure of the schema change.

4.2 User Interface Mock-Ups

In this section, we will provide an overview of the User Interface (UI) mock-ups for the Laboratory Reservation System. These mock-ups are intended to visually represent the design and layout of the system's user interfaces, offering a preview of the user experience. In our case project we've made use of Wireframe [34] to create mock-ups for each page.

4.2.1 Landing Page

The landing page, or home page, is the first page the user access when entering the system. In this landing page we should be able to create or select a semester and then import a file, as you can see in Figure 8.

4.2.2 Semesters Page

In the semesters page in Figure 9 we will be able to see a list of semester and create a new one.

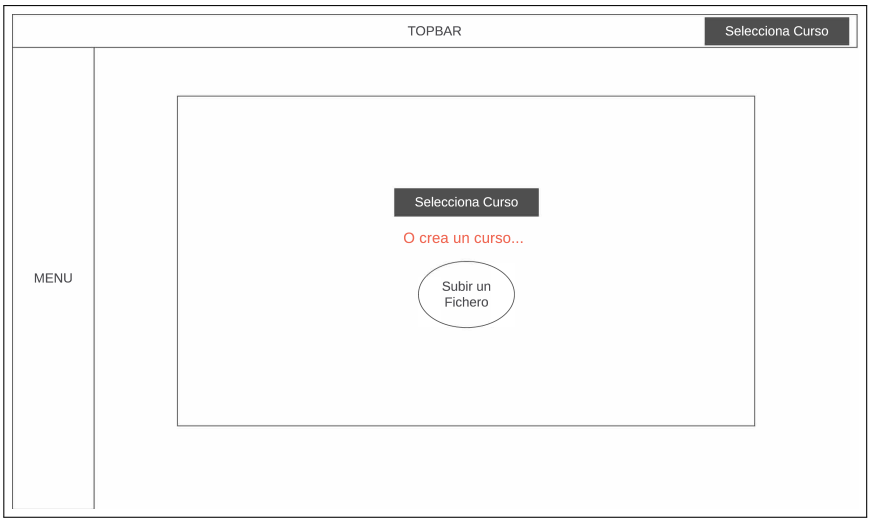


Figure 8: Landing page mock-up

Figure 9: Semesters page mock-up

4.2.3 Reservations Page

In the reservations page in Figure 10 we will be able to see a list of reservations and create a new one.

4.2.4 Conflicts Page

The conflicts page in Figure 11 is composed of two components, one showing a calendar with conflicting reservations and another component listing out all the conflicts to be resolved.

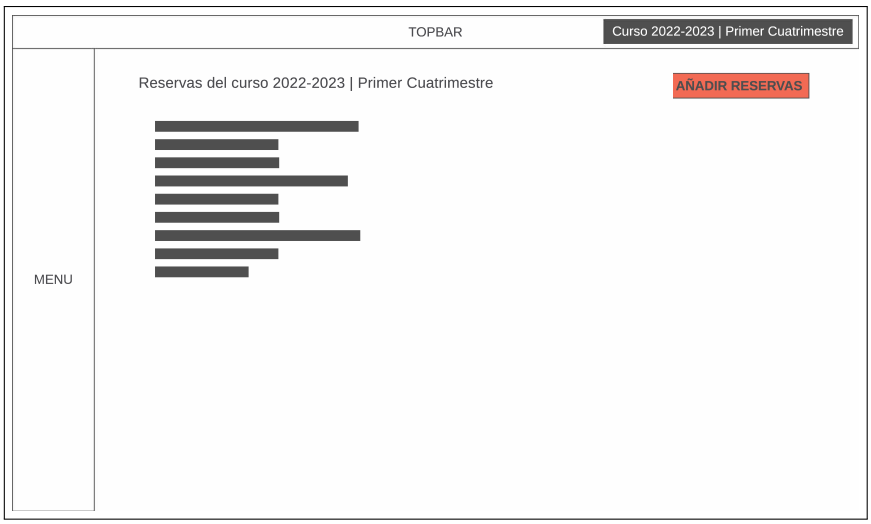


Figure 10: Reservations page mock-up

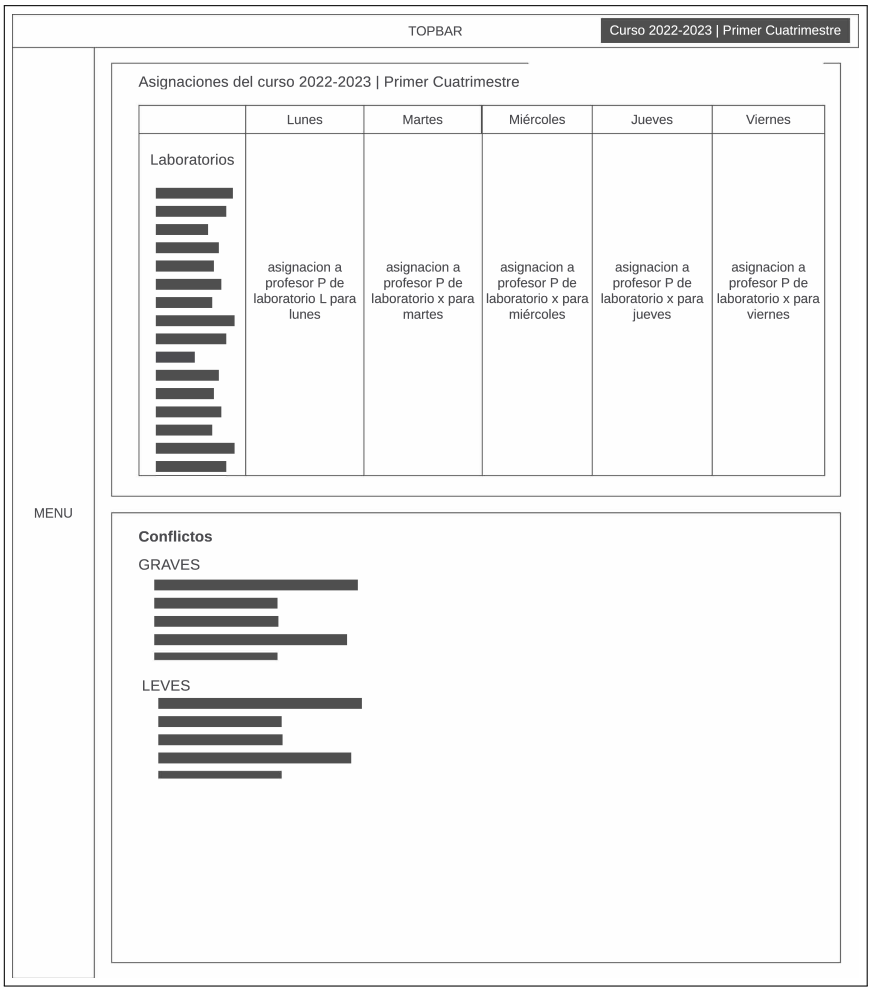


Figure 11: Conflicts page mock-up

5 Development

In this section, we delve into the heart of our project, exploring its development journey and the key components that bring it to life. From the user interface that provides a seamless interaction experience to the intricate file import processes, and the powerful genetic algorithm that underpins the system's decision-making, this section offers an in-depth view of the development intricacies.

We'll walk you through each facet of our development journey, from the elegant user interface components, including the Landing Page, Semesters Page, Reservations Page, Conflicts Page, Laboratories Page, and more, to the essential functionalities like file import and task scheduling. You'll gain insight into the inner workings of our genetic algorithm, examining its properties, constraints, procedures, and practical test cases.

Additionally, we'll explore the containerization of our application through Docker, presenting a detailed overview of Docker Container setups, including test containers, API DockerFiles, UI DockerFiles, and Docker Compose configurations. By the end of this section, you'll have a comprehensive understanding of how our project was brought to life through meticulous development.

5.1 User Interface

The final result of the user interface includes a Material UI Theme [31] that provides a visually appealing and cohesive design. This theme incorporates a carefully chosen color palette to enhance the overall user experience.

The primary color scheme, represented by the hexadecimals #053C5E (dark), #1F7A8C (main), and #BFDBF7 (light). The dark and light variations create a pleasing contrast that ensures readability and accessibility across the interface.

Complementing the primary colors, the secondary color palette adds depth and vibrancy

to the UI. It features #498469 (dark), #ABB58E (main), and #FB9039 (light). These colors are strategically applied to highlight key elements and interactions, providing visual cues and aiding in user navigation.

Before diving into the next subsections, it is worthy noting that there is a loading state that involves a loading spinner when fetching or mutation data, and in case of an error there is an error component that will be rendered with the error message and a button to recover.

5.1.1 Landing Page

The landing page in Figure 12 serves as the entry point to the system providing the functionality to create or, if it exists, select a semester and, afterwards, upload a CSV file to create a task that will execute the algorithm.

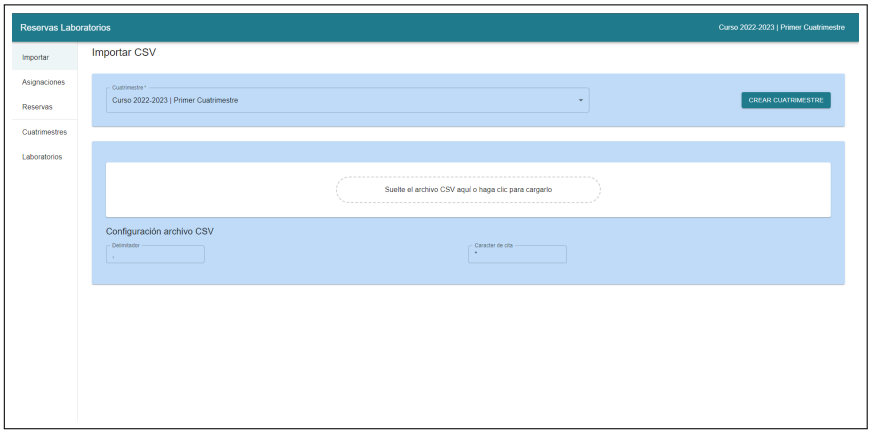


Figure 12: Landing page

5.1.2 Semesters Page

The semesters page in Figure 13 presents a clear overview of existing academic semesters and offers the ability to create new ones as seen in Figure 14. There is also a radio button next to each semester so you can mark it as the selected semester to work with.

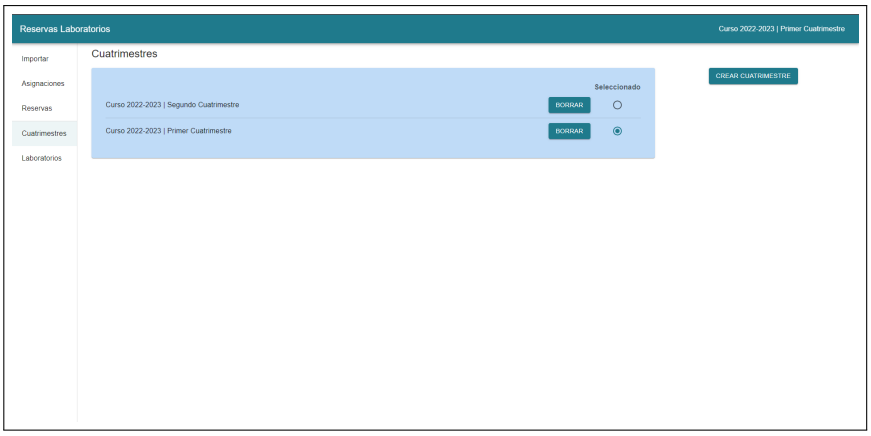


Figure 13: Semesters page

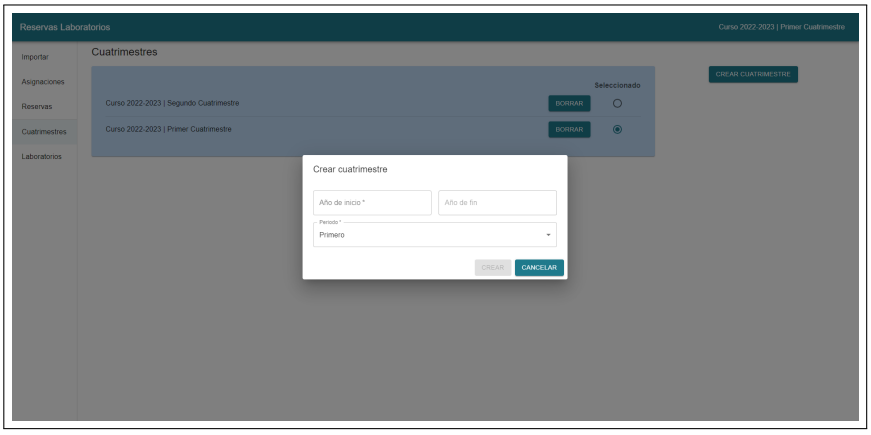


Figure 14: Create semester dialog

5.1.3 Reservations Page

This page is only available after the user has imported a CSV file. The reservations page in Figure 15 shows a table of existing reservations for the selected semester that is visible in the top bar.

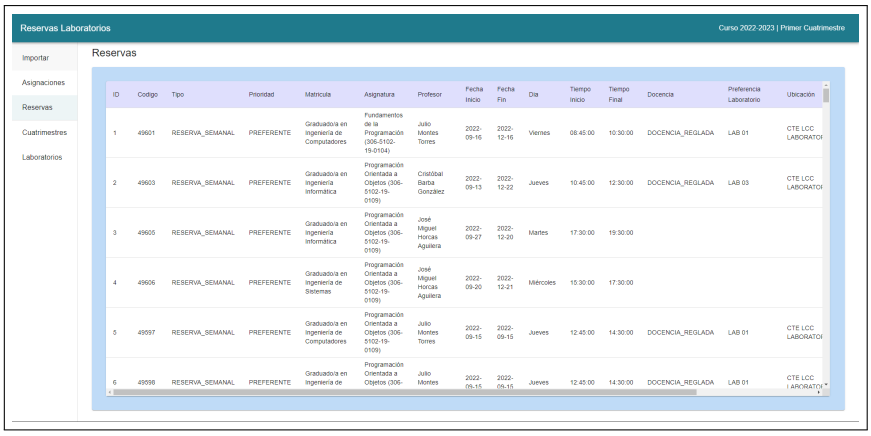


Figure 15: Reservations page

5.1.4 Conflicts Page

This page is only available after the user has imported a CSV file. Once imported the user will see that the algorithm is running like in Figure 16, and it would also show an error if an exception happens during the execution of the algorithm as shown in Figure 22.

Figure 16: Conflicts page - Algorithm running

The conflicts page consists of two toggleable components. The first component is a comprehensive list of conflicts (shown in Figure 17) that includes reservations with the 'Alternative' schedule type highlighted in green. This highlighting helps users easily spot potential modifications that could resolve the conflicts.

The second component is a calendar display (depicted in Figure 18) that highlights reservations with conflicts in red and assignments without conflicts in blue, while showing the public_id of the reservation. Additionally, it initializes the calendar with reservations spanning

the first week and restricts the user from navigating beyond the date range of all reservations combined.

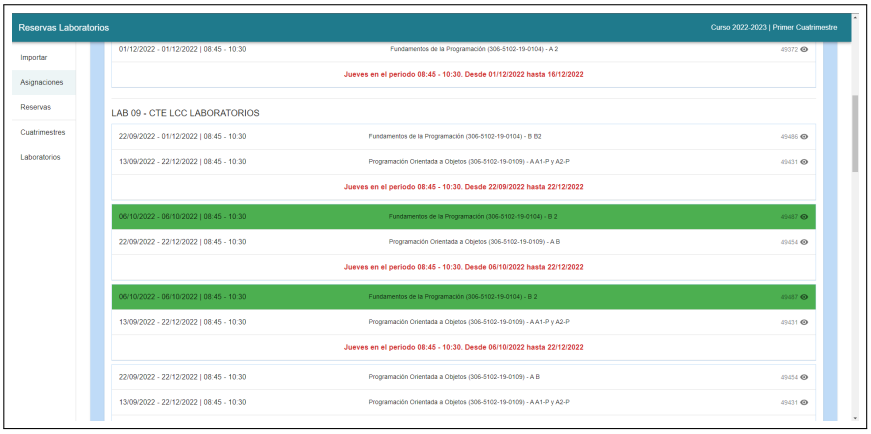


Figure 17: Conflicts page - List option

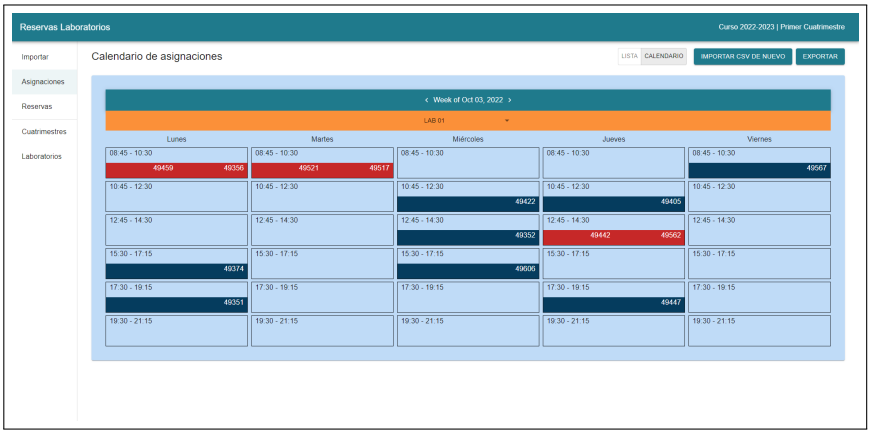


Figure 18: Conflicts page - Calendar option.

5.1.5 Laboratories Page

The laboratories page features an interface for managing laboratories. It includes a visual representation of laboratories as shown in Figure 19, and allows the creation of new laboratory profiles (Figure 20), while also being able to edit or delete existing laboratories.

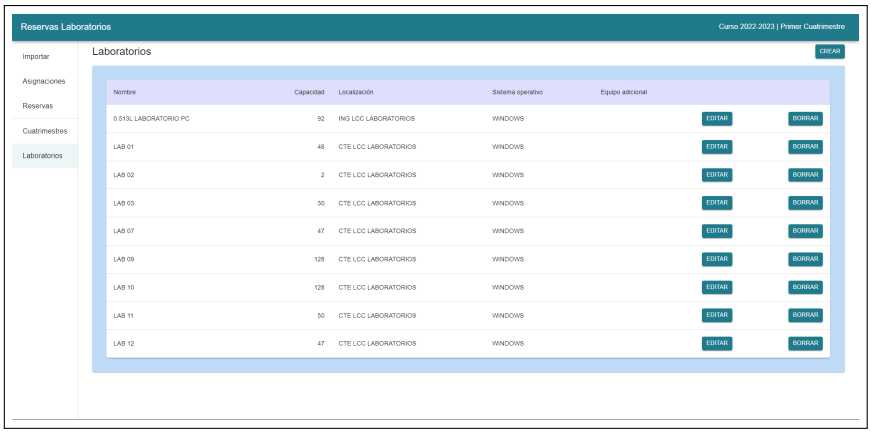


Figure 19: Laboratories page

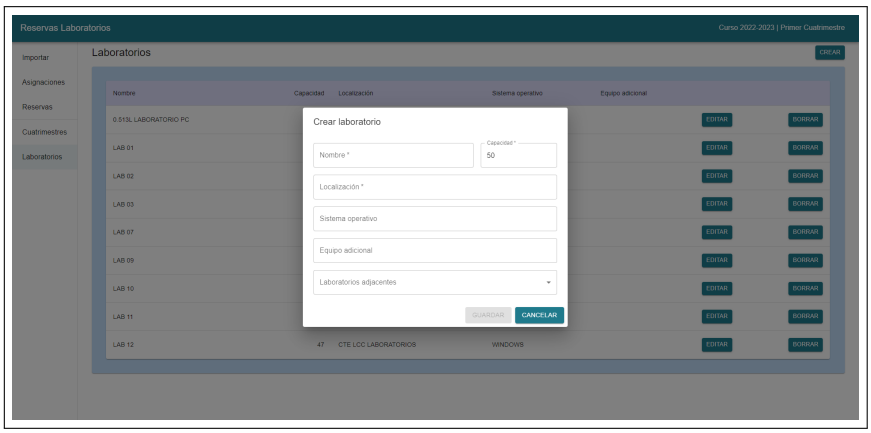


Figure 20: Create Laboratory dialog

5.1.6 Confirmation Dialog

This is a shared confirmation dialog component used in the laboratories, semesters and landing page to confirm with the user an action that could have consequences as showed in Figure 21.

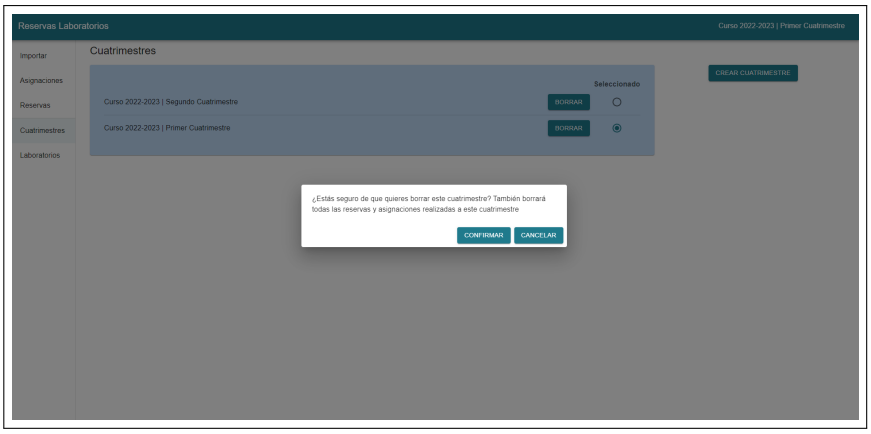


Figure 21: Confirmation dialog with custom messages - Deleting a semesters

5.1.7 Error Component

The "Error component" is a crucial part of the user interface designed to inform users about unexpected errors or issues that have occurred within the system. This component is employed across various sections of the application to effectively communicate error messages to the user. It plays a vital role in enhancing the user experience by providing clear and concise feedback when something goes wrong. Figure 22 illustrates an example of the Error component in action, displaying an error message related to an issue encountered while running the algorithm and also providing a button to refresh the page.

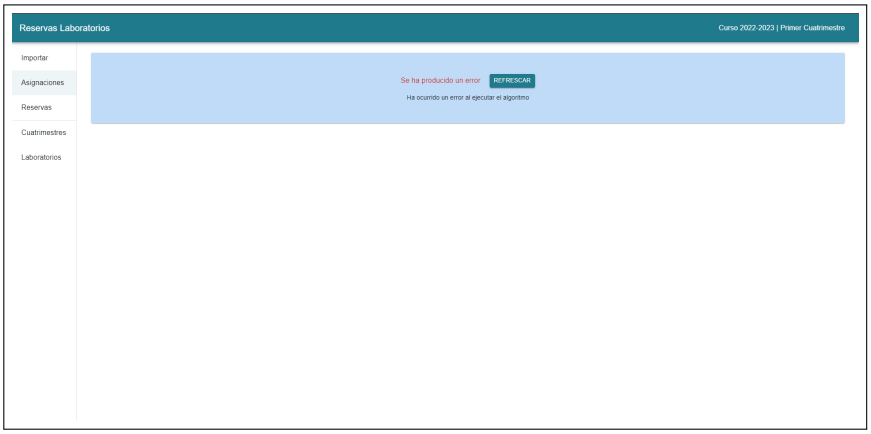


Figure 22: Error component with error message - Error running algorithm

Having crafted the user interface components that form the face of our system, it's time to delve deeper into the back-end intricacies that power its functionality. In the sections that follow, we'll explore the critical back-end elements that enable seamless data processing and

decision-making. Our journey begins with the file import feature, a crucial bridge between user-generated data and the system's core functionality.

5.2 File Import

In this section, we will delve into the development details of the file import feature. This feature allows users to upload CSV files containing reservation data to the system. The uploaded data is processed and used to create an assignment for a laboratory later. This process includes validation checks to ensure data integrity and a task scheduling mechanism to efficiently handle and process the imported data.

5.2.1 Validation

The first step in the file import process is validation. When a user uploads a CSV file, the system performs various checks and validations to ensure the data is accurate and can be processed correctly. These validations include checking for required fields, verifying data formats, and identifying any potential errors in the file. If the file contains errors, the system stores the error messages for that Data Transfer Object (DTO) and proceeds to the next one, until transforming all DTOs. Only a valid and error-free list of reservation is stored in the system, in case of any error the system will return a list of *ReservationDtoWithError* that includes the DTO, row number in the file and error message. An example of how this looks is to be seen in Figure 23, where each of the errors can be expanded to see what the DTO looks like, helping the user to identify any issues with the imported file.

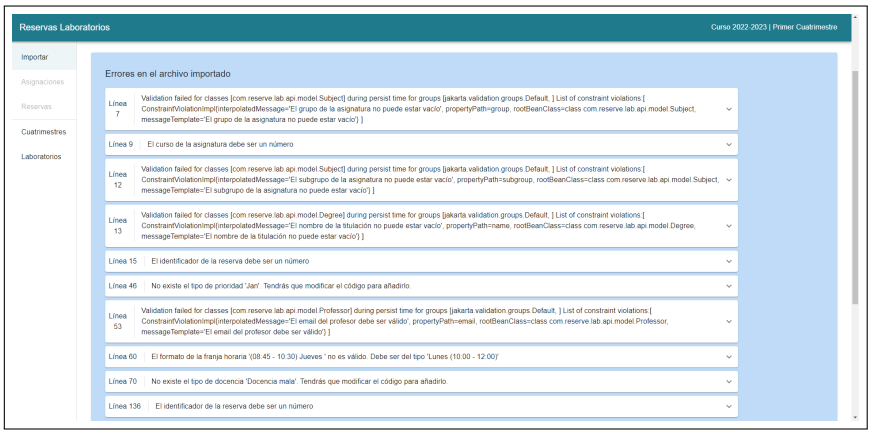


Figure 23: Errors in imported CSV file

If there was a previous imported file for the selected semester, then these reservations and assignments would be over-written.

5.2.2 Task Scheduling

After completing the import process, the system will automatically create a tasks related to currently selected semester. The scheduler runs every 30 seconds to see if there are any tasks with the type "'QUEUED", if so, it will run the algorithm once over the semester in the task and save the task as "RUNNING". The frequency of which the scheduler is executed can be changed from the *application.yml* file, in the format of a *cron* expression [19].

If the execution of the task fails, it will be marked as "ERROR", and the associated error message will be stored in the database. Furthermore, if any tasks remain in the "RUNNING" status for more than 30 minutes since the last update, the system will mark them as "OUTDATED". Finally, if the algorithm has run successfully, the task will be marked as "COMPLETED". All of these states are handled accordingly in the user interface.

Finally, in addition to the core genetic algorithm utilized for laboratory assignment, our system is designed to accommodate the implementation of other scheduling algorithms, by modifying the algorithm to run from the scheduler service.

5.3 Genetic Algorithm

The genetic algorithm offers a robust solution to the intricate laboratory assignment problem. This problem, characterized by its NP-hard nature and the multitude of constraints it entails, demanded a technically sound and adaptable approach. The laboratory assignment problem, given its complexity, necessitates an algorithmic approach capable of efficiently minimizing conflicts, optimizing resource allocation, and meeting a plethora of constraints. The genetic algorithm presented itself as a compelling choice due to its versatility and adaptability.

Notably, our system's design allows for seamless integration and interchangeability of algorithms, enabling us to explore and implement different techniques as needed. In this section, we unveil the inner workings of our genetic algorithm, shedding light on its properties, constraints, procedures, and rigorous testing, all of which play pivotal roles in delivering an efficient and effective reservation management system.

5.3.1 Procedure

The genetic algorithm follows a well-defined procedure to assign reservations to laboratories effectively. The population will be a list of solutions, for which each will have a list of assignments that includes a laboratory and reservation, with their penalty score. The procedure involves several steps:

1. **Setup**: Delete any previous assignments and conflicts related to the selected semester.
2. **Initial population**: An initial population of potential solutions (assignments) is generated.
3. **Fitness evaluation**: The fitness of each solution in the population is evaluated, considering the constraints.
4. **Genetic algorithm Loop**: The core genetic algorithm loop iterates through generations, including:
   (a) **Selection**: The selection of promising solutions from the current population.
   (b) **Crossover**: Combining selected solutions to create offspring.
   (c) **Mutation**: Introducing variations in offspring to maintain diversity.
   (d) **Fitness evaluation**: Re-evaluating the fitness of offspring.
   (e) **Replacement**: Replacing the current population with offspring.
   (f) **Termination condition**: Checking for a termination condition, such as penalty score of 0, to stop the algorithm early.
5. **Best solution selection**: The best solution (assignments) is selected from the final population.
6. **Solution storage**: The best solution, along with any potential conflicts, is stored in the system.

The "Solution" class represents a potential solution of assigning reservations to laboratories while minimizing conflicts and optimizing resource utilization. It consists of two main components:

- **Assignments**: This component is represented as a list of "ReservationAssignment" objects. Each "ReservationAssignment" represents the assignment of a reservation to a laboratory for a specific semester. When the best solution is selected, this list would be saved to the database.

- **Penalties**: This component is an "EnumMap" that tracks the different types of constraints that are broken, and is calculated when executing the algorithm's fitness evaluation. The "PenaltyType" enumeration categorizes these constraints, and the associated integer value counts the occurrences of each penalty type. Penalties serve as indicators of how well a particular solution adheres to the defined constraints.

Additionally, it logs various performance metrics, such as execution times for each algorithm step and the occurrence counts of each penalty type at the end of the algorithm. While these metrics are crucial for algorithm refinement and internal analysis, they are not stored in the database and are intended for internal use only.

5.3.2 Constraints

In the context of scheduling and managing reservations within a laboratory or educational environment, various constraint categories play a crucial role in ensuring efficient and effective operations. These categories, denoted as A, B, C, and D, encompass a wide range of rules and conditions that govern reservation-related activities. Category A focuses on reservation conflicts, both standard and those involving alternative schedules, along with additional equipment availability. Category B addresses issues related to laboratory assignments, including preferred laboratory unavailability and location mismatches. Category C deals with subject-specific constraints, ensuring that reservations are distributed appropriately across laboratories and that assignments for the same subject remain consistent. Finally, Category D introduces soft constraints concerning group reservations. Together, these constraint categories provide a comprehensive framework for optimizing the reservation and scheduling processes while maintaining quality and fairness. The penalty score for each type of constrain is calculated by multiplying the score of that penalty times the occurrence.

- **A1-1 RESERVATION CONFLICT** (Score: 800): A conflict occurs when two reservations for the same laboratory have overlapping time slots for the same day and date, at

any time of the semester.

- **A1-2 RESERVATION CONFLICT WITH ALTERNATIVE SCHEDULE** (Score: RESERVATION CONFLICT / 2): Similar to the RESERVATION CONFLICT, but one of the conflicting reservations has an alternative schedule type.

- **A1-3 LABORATORY WITH DIFFERENT OPERATING SYSTEM** (Score: 600): If the operating system is specified in a reservation but is not available in the assigned laboratory then this constraint is violated.

- **A1-4 RESERVATION ADDITIONAL EQUIPMENT NOT AVAILABLE** (Score: 500): If additional equipment is specified in a reservation but is not available, this constraint is violated.

- **A2-1 RESERVATION CONFLICT PER WEEK** (Score: RESERVATION CONFLICT / 10): This constraint counts the number of weeks when two reservations overlap, with both being "preferred" schedule types.

- **A2-2 RESERVATION CONFLICT WITH ALTERNATIVE SCHEDULE PER WEEK** (Score: RESERVATION CONFLICT WITH ALTERNATIVE SCHEDULE / 10): This constraint calculates the number of weeks when two schedules collide, with one being an alternative schedule.

- **A2-3 RESERVATION CONFLICTS NOT DISTRIBUTED EQUALLY ACROSS SUBJECTS** (Score: 200): Soft constraint that ensures conflicts are balanced among subjects, preventing one subject from having significantly more conflicts than others.

- **B-1 RESERVATION PREFERRED LABORATORY NOT ASSIGNED** (Score: 25): This constraint is violated when the laboratory specified for a reservation in the import file is not available or assigned.

- **B-2 LABORATORY IN DIFFERENT LOCALIZATION** (Score: 40): This constraint is violated if the specified location for a reservation in the import file does not match the assigned laboratory's location.

- **B-3 RESERVATIONS WITH SAME PUBLIC ID NOT IN SAME LABORATORY** (Score: 30): This constraint checks that reservations with the same publicId (e.g., "Reserva de lista de días") are assigned to the same laboratory.

- **C-1 RESERVATIONS OF SAME SUBJECT NOT IN ADJACENT LABORATORIES** (Score: 50): This constraint checks if assignments of the same subject are not in adjacent laboratories. Each laboratory may have a set of adjacent laboratories.

- **C-2 RESERVATIONS OF SAME SUBJECT NOT IN SAME LABORATORY** (Score: 50): Ensures that assignments for the same subject are consistently in the same laboratory throughout the semester.

- **D-1 MORE THAN ONE RESERVATION PER GROUP** (Score: 10): When a group has more than one reservation, this soft constraint is violated.

These constraints play a crucial role in optimizing reservations, minimizing conflicts, and ensuring efficient resource allocation in the system. When a constraint is violated, it contributes to the penalty score of a solution. The penalty score reflects how well a particular solution complies with the defined constraints. The process of calculating the penalty score is as follows:

- We keep track of the total amount of occurrences of each penalty type, but also separately the total amount of penalty type A1, since we rather have a solution with no conflicts and a large a mount of minor penalties, than one where the penalty types are distributed equally.

- We iterate through the penalties "EnumMap", tracking the occurrence (for both total and A1 constraints) and summing up the total penalty score, which is the occurrence of that penalty type times the score it has.

  Total Penalty Score $= \sum_i (\text{Occurrences of Penalty Type}_i \times \text{Score of Penalty Type}_i)$

- If the A1 constraint penalties are less than 10% of the total penalties, the overall score is reduced by 75%.

Overall, the penalty-based approach is designed to enforce adherence to specified constraints. Its primary goal is to enhance reservation management and optimize resource utilization, aiming to minimize conflicts and attain a penalty score of 0, signifying full compliance with all constraints.

5.3.3 Operators

The **initial population** operator is responsible for creating the initial population of solutions for the genetic algorithm. Given a specific semester, it retrieves a list of reservations matching the given semester. It then proceeds to create a population of solutions, each representing a potential assignment of reservations to laboratories. The population size is determined by the algorithm's properties. To create each random solution, it takes the list of reservations and all laboratories as inputs and follows these steps:

1. Create an empty solution.
2. Shuffle the list of reservations randomly, ensuring a random order for assignment.
3. Iterate through the shuffled reservations and, for each reservation, identify suitable laboratories based on compatibility.
4. Assign the reservation to a randomly selected suitable laboratory.
5. Calculate and set the penalty score for the random solution based on penalty occurrences.
6. Add the solution to the population, and repeat until we've reached the specified population size.

The **selection** operator determines which solutions from the current population will be chosen to form the selected subset for crossover and potentially serve as parents for the next generation. This operator consists of two key steps:

1. **Elitism selection** - Aims to preserve a portion of the best solutions from the current population. The number of solutions retained through elitism is determined by the *Elitism Replacement Rate*, specified in the algorithm's properties.

(a) Sort the solutions in the population in ascending order based on their penalty scores.

(b) Retain a portion of the solutions with the lowest penalty scores, representing the best-performing individuals.

(c) Add the elitism solutions to the selected solutions.

2. **Rank-Based selection** - Focuses on selecting the remaining solutions for the next generation based on their ranking and fitness. This step ensures diversity in the selected subset.

   (a) Calculate the number of remaining solutions needed to reach the desired population size.

   (b) Calculate the total penalty score of the entire population.

   (c) For each remaining solution required, generate a random value between 0 and 1.

   (d) Iterate through the population solutions, calculating the cumulative probability of selection based on the solutions' penalty scores.

   (e) When the cumulative probability surpasses the random value, select the corresponding solution and add it to the selected solutions.

This ensures that solutions with better fitness, as well as diverse solutions, have a chance to contribute to the next generation. It strikes a balance between preserving the best-performing individuals and exploring other potential solutions.

The **crossover** operator is a pivotal step in the genetic algorithm, responsible for creating new offspring by combining attributes from selected parent solutions. In our implementation, we perform crossover between pairs of selected solutions to generate a new population of potential solutions. Here's an overview of the crossover process:

1. Initialize an empty list, *offspring*, to store the newly created solutions.

2. Iterate through the selected solutions, pairing them off consecutively.

3. For each pair of parents, *parent1* and *parent2*, decide whether to perform crossover based on the specified crossover rate.

4. If crossover is to occur, create a new solution, *newOffspring*, as follows:

    (a) Choose a random point, known as the crossover point, to perform the crossover operation.

    (b) Sort reservations within both parents by reservation ID to ensure consistency.

    (c) Combine assignations from *parent1* up to the crossover point.

    (d) Append assignations from *parent2* after the crossover point.

    (e) Calculate the penalty score for the newly created offspring based on penalty occurrences.

    (f) Add the new offspring to the *offspring* list.

5. If crossover is not performed for a pair, directly pass *parent1* and *parent2* as offspring.

6. Continue this process as many times as solutions exist after the selections operator.

The **mutation** operator plays a crucial role in introducing random changes to the offspring solutions. It involves a double mutation process: one for the entire solution and another for reservations with conflicts.

For the whole solution, the mutation operator iterates through the list of offspring solutions and, for each solution, iterates over the list of assignments. Here's the process:

1. Iterate through the list of offspring solutions.

2. For each solution, iterate over the list of assignments.

    (a) For each assignment, determine whether to apply a mutation based on the specified mutation rate.

    (b) If mutation is to occur, identify laboratories that are available for the reservation, except for the one currently assigned.

    (c) Randomly select an available laboratory from the list of available laboratories. If the list of laboratories is empty, it will skip the assignment and go to the next.

    (d) Update the reservation assignment to the newly selected laboratory.

3. Continue this process for all assignments in the solution.

For reservations with conflicts, the repair mutation operator identifies assignments that are involved in conflicts and iterates over them. Here's the process:

1. Select assignments with conflicts from the solution.
2. For each assignment involved in a conflict, determine whether to apply a mutation based on the specified mutation-repair rate.
3. If mutation is to occur, identify laboratories that are available for the reservation, except for the one currently assigned.
4. Randomly select an available laboratory from the list of available laboratories. If the list of laboratories is empty, it will skip the assignment and go to the next.
5. Update the reservation assignment to the newly selected laboratory.
6. Continue this process for all assignments with conflicts in the solution.

The mutation operator is instrumental in diversifying the population and allowing the genetic algorithm to explore different assignment possibilities. While mutations are random and small in scope, they can lead to the discovery of improved solutions as the algorithm progresses through generations.

The **replacement** operator is a pivotal step in forming the next generation of solutions within the genetic algorithm. It determines how the new population will be assembled based on the fitness scores of solutions from the parent population and the offspring. Here's an overview of the replacement process:

1. Initialize a new population, referred to as *newPopulation*.
2. Implement a replacement strategy, in this case, the *elitism* method, which prioritizes the best solutions.
3. Add the solutions from the parent population and the offspring population to *newPopulation*.
4. Sort *newPopulation* in ascending order based on penalty scores.

5. Remove solutions from the end of *newPopulation* until the population size matches the specified limit.

6. Randomize the order of solutions in *newPopulation*.

7. Return *newPopulation* as the updated population for the next generation.

The *Replacement* operator ensures that the genetic algorithm retains the best-performing solutions while introducing diversity through offspring solutions. By maintaining a balanced population size, it contributes to the algorithm's exploration and exploitation of potential solutions. The *elitism* strategy, in this case, prioritizes the best solutions, helping the algorithm converge toward optimal solutions over time.

5.3.4 Properties

The genetic algorithm is configurable through the *application.yml* file or the swagger endpoint, allowing fine-tuning of its behavior. The **AlgorithmProperties** class defines these properties, including:

- **populationSize**: The size of the population used in the genetic algorithm.
- **maxGeneration**: The maximum number of generations the algorithm will iterate through.
- **crossoverRate**: The crossover rate, a value between 0 and 1, determining the probability of crossover during reproduction.
- **mutationRate**: The mutation rate, a value between 0 and 1, specifying the probability of mutation in the genetic algorithm. In our case it would be assigning another suitable laboratory to an existing assignation.
- **mutationRepairRate**: The mutation-repair rate, a value between 0 and 1, specifying the probability of mutation in the genetic algorithm. In our case it would be assigning another suitable laboratory to an existing assignation when the reservation of this assignation is in a conflict.
- **elitismReplacementRate**: A value between 0 and 1, specifying the portion of best solutions to preserve in our population during the selection operator.

These properties influence the algorithm's performance and can be adjusted to meet specific requirements.

5.3.5 Test Cases

To validate the genetic algorithm's functionality, a set of test cases is employed. This tests are tested under the following algorithm configuration:

- **populationSize:** 50
- **maxGeneration:** 20
- **mutation-rate:** 0.5
- **mutation-repair-rate:** 0.8
- **crossover-rate:** 0.5
- **elitism-replacement-rate:** 0.5

Moreover, the reservations used in these tests have a start date at most 50 days into the future of the date when tested, while the end date has at most 50 days into the the future of the created start date. This represents real life scenarios where all reservations are in the same date slot, and the time slots are one of the following randomly selected options: (08:45 - 10:30), (10:45 - 12:30), (12:45 - 14:30), (15:30 - 17:15), (17:30 - 19:15), (19:30 - 21:15).

These test cases assess various aspects of the algorithm's performance, with random generated data from the *data generator service*, including:

- Ensuring the algorithm runs successfully and assigns reservations to laboratories.
  - **Parameters:** 100 reservations, 1 semester, 10 laboratories.
  - **Results** in Figure 24.
  - **Asserts:**
    - The number of assignments must equal to the number of reservations.
    - For each assignment, it is checked that it has a laboratory assigned, and each reservation is associated with a laboratory exactly once.

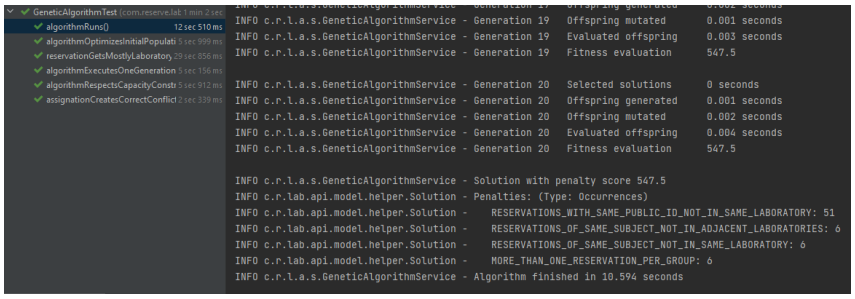


Figure 24: Algorithm runs

  * For each reservation provided it is ensured that it is present in the assignations.
  * In the case of conflicts, it is verified that the reservations involved in conflicts are different from each other.

- Verifying that reservations predominantly respect laboratory preferences.
  - **Result 1** in Figure 25 with **parameters:** 100 reservations with a preferred laboratory, 1 semester, 8 laboratories.

```
Solution with penalty score 670.0
Penalties: (Type: Occurrences)
   RESERVATIONS_WITH_SAME_PUBLIC_ID_NOT_IN_SAME_LABORATORY: 63
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_ADJACENT_LABORATORIES: 9
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_SAME_LABORATORY: 5
   MORE_THAN_ONE_RESERVATION_PER_GROUP: 9
Algorithm finished in 4.644 seconds
```

Figure 25: Algorithm respects laboratory preference - Test 1

  - **Result 2** in Figure 26 with **parameters:** 100 reservations with a preferred laboratory, 1 semester, 4 laboratories.
  - **Asserts:**
    * Ensure that the count of the number of assignments where a reservation's laboratory preference matches the assigned is greater than 80% of the total amount of reservations.

- Confirming that the algorithm correctly identifies and reports conflicts.

```
Solution with penalty score 1757.5
Penalties: (Type: Occurrences)
   RESERVATION_CONFLICT: 4
   RESERVATION_CONFLICT_PER_WEEK: 4
   RESERVATION_CONFLICTS_NOT_DISTRIBUTED_EQUALLY_ACROSS_SUBJECTS: 8
   RESERVATIONS_WITH_SAME_PUBLIC_ID_NOT_IN_SAME_LABORATORY: 47
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_ADJACENT_LABORATORIES: 5
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_SAME_LABORATORY: 4
   MORE_THAN_ONE_RESERVATION_PER_GROUP: 5
Algorithm finished in 18.725 seconds
```

Figure 26: Algorithm respects laboratory preference - Test 2

- **Parameters:** 2 identical reservations with preferred laboratory, 1 semester, 1 laboratory.
- **Results** in Figure 27.

```
Solution with penalty score 405.0
Penalties: (Type: Occurrences)
   RESERVATION_CONFLICT: 1
   RESERVATION_CONFLICT_PER_WEEK: 7
   RESERVATION_CONFLICTS_NOT_DISTRIBUTED_EQUALLY_ACROSS_SUBJECTS: 1
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_ADJACENT_LABORATORIES: 1
   MORE_THAN_ONE_RESERVATION_PER_GROUP: 1
Algorithm finished in 3.079 seconds
```

Figure 27: Algorithm identifies conflicts correctly

- **Asserts:**
  - Ensure that there is exactly one conflict.
  - Verify that the days of both reservations in the conflict are equal.
  - Verify that the day of the first reservation in the conflict matches the conflicts day.

- Ensuring that capacity constraints are respected in assignments.
  - **Parameters:** 100 reservations, 1 semester, 100 laboratories made of the reservations.
  - **Results** in Figure 28.
  - **Asserts:**

```
Solution with penalty score 487.5
Penalties: (Type: Occurrences)
   RESERVATIONS_WITH_SAME_PUBLIC_ID_NOT_IN_SAME_LABORATORY: 65
Algorithm finished in 6.812 seconds
```

Figure 28: Algorithm respects capacity constraint

  * For each assignment, ensure that the laboratory's capacity is greater than or equal to the number of students in the reservation.
  * Confirm that the size of conflicts is zero.

- Assessing the algorithm's ability to optimize the initial population.
  - **Parameters:** 100 reservations, 1 semester, 10 laboratories, 1 initial random solution.
  - **Results** in Figure 29.

```
Solution with penalty score 655.0
Penalties: (Type: Occurrences)
   RESERVATIONS_WITH_SAME_PUBLIC_ID_NOT_IN_SAME_LABORATORY: 69
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_ADJACENT_LABORATORIES: 5
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_SAME_LABORATORY: 5
   MORE_THAN_ONE_RESERVATION_PER_GROUP: 5
Algorithm finished in 7.993 seconds
```

Figure 29: Algorithm optimizes first solution

  - **Asserts:** Verifies that the new solution by the algorithm has a better penalty score that the initial solution.

- Measuring the execution time for a specific scenario to ensure timely processing.
  - **Parameters:** 100 reservations, 1 semester, 10 laboratories, population size of 100, max 100 generation.
  - **Results** in Figure 30.
  - **Asserts:** Verifies that the algorithm finds a solution under 1 minute, for a population of 100 solutions, where each solution has 100 reservations to assign across 10 laboratories, for a total of 100 generations.

```
Solution with penalty score 460.0
Penalties: (Type: Occurrences)
   RESERVATIONS_WITH_SAME_PUBLIC_ID_NOT_IN_SAME_LABORATORY: 50
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_ADJACENT_LABORATORIES: 4
   RESERVATIONS_OF_SAME_SUBJECT_NOT_IN_SAME_LABORATORY: 2
   MORE_THAN_ONE_RESERVATION_PER_GROUP: 4
Algorithm finished in 20.901 seconds
```

Figure 30: Algorithm execution time measured

In conclusion, these test cases rigorously assess the functionality and performance of our reservation assignment algorithm. Through a series of comprehensive tests, we have verified that the algorithm successfully assigns reservations to laboratories while adhering to constraints and preferences. It also handles conflicts, respects laboratory capacities, and optimizes the initial population effectively. However, the development process was not without its challenges.

Some of the performance issues encountered during the development of this system included slow response rates when reading data from the database, particularly from the 'laboratories' table, while attempting to create an initial population. In response to this issue, the algorithm underwent modifications to accept a pre-fetched list of available laboratories. This adjustment allowed the algorithm to perform these verifications in memory, eliminating the need for unnecessary database queries. These improvements significantly enhanced the algorithm's efficiency and overall performance.

5.4 Docker Container

Docker containers provide an efficient and consistent way to package and run different components of the system, ensuring seamless integration and portability. This section delves into the utilization of Docker containers for various purposes within the system. To see a full manual on how to deploy this solution, see Appendix A.

5.4.1 Test Containers

TestContainers is a powerful framework that facilitates the use of Docker containers for integration testing. In the context of the reservation management system, test containers are leveraged to set up and manage a PostgreSQL database container during integration tests.

This allows for a controlled and isolated database environment, ensuring the reliability and consistency of tests.

The **TestContainerInitializer** class initializes the PostgreSQL container using the **PostgreSQLContainer** provided by **TestContainers**. It configures the database connection properties for Spring Boot applications, ensuring that tests run against the PostgreSQL container seamlessly.

Neadless to say, that to execute these tests on a machine, there must be a docker environment ready to be used.

5.4.2 API DockerFile

The API Dockerfile defines the containerization process for the reservation management system's back-end. It is based on the **openjdk:17-jdk-slim** image and performs the following tasks:

- Sets the working directory within the container.
- Copies the compiled JAR file of the Spring Boot application into the container.
- Exposes port 8080 to allow external access to the Spring Boot application.
- Specifies the command to run the Spring Boot application within the container.

5.4.3 UI DockerFile

The UI Dockerfile defines the containerization process for the reservation management system's front-end. It utilizes two stages: the builder stage and the final stage. In the builder stage, it uses the **node:alpine** image to build the front-end application. It performs the following tasks:

- Sets the working directory within the builder stage.
- Copies package.json and package-lock.json to install dependencies.
- Installs dependencies and builds the front-end application.

In the final stage, it uses the **nginx:1.16.0-alpine** image to serve the built front-end application. It performs the following tasks:

- Copies the built front-end files from the builder stage to the nginx image.
- Removes the default nginx configuration.
- Copies a custom nginx configuration for serving the front-end.
- Exposes port 80 to allow access to the front-end application.
- Specifies the command to run nginx in the foreground.

5.4.4 Docker Compose

Docker Compose is used to orchestrate the deployment of the reservation management system as a set of interconnected containers. The *docker-compose.yml* file defines the configuration for multiple services, including the back-end, front-end, and PostgreSQL database. It also specifies network settings for container communication.

The services configured in the *docker-compose.yml* file include:

- **back-end**: Defines the configuration for the back-end container, specifying the build context, dependencies on the PostgreSQL container, port mapping, and network settings.
- **front-end**: Defines the configuration for the front-end container, specifying similar build and dependency settings as the back-end, along with port mapping.
- **postgrel**: Configures the PostgreSQL container, setting environment variables for database configuration, port mapping, and network settings.

6

Conclusions and Futures Lines of Research

6.1 Conclusions

The completion of the reservation management system marks a significant milestone, not only in addressing the specific needs of the LCC department but also in providing a robust infrastructure that can be further expanded and customized to meet evolving requirements. This project stands as a testament to the versatility and adaptability of software solutions in streamlining complex processes within educational institutions.

One of the primary achievements of this project is its ability to empower the LCC department with a comprehensive toolset for managing laboratory reservations efficiently. The system's user-friendly interface, backed by a reliable back-end, simplifies the reservation process, minimizes conflicts, and enhances overall productivity. The dynamic allocation of resources, including laboratories and equipment, ensures that resources are utilized optimally, aligning with the department's goals of efficiency and resource optimization.

The ability to create, edit, and delete semesters, import reservations from CSV files, and visualize assignments and conflicts provides the LCC department with unprecedented control and visibility over their laboratory management process. Furthermore, the system's extensibility allows for the integration of additional features and functionalities tailored to the department's unique requirements, ensuring long-term relevance and adaptability.

Throughout the development of this project, the exploration of genetic algorithms as an optimization technique adds a valuable dimension to our skill set as a software engineer. The application of genetic algorithms, although not always the most straightforward approach,

offers insights into solving complex optimization problems efficiently. The choice to employ genetic algorithms showcases a commitment to mastering advanced techniques in algorithm design and optimization.

In retrospect, while an exhaustive method might have sufficed for the scale of this project, the pursuit of genetic algorithms demonstrates a forward-looking mindset and an interest in tackling more substantial challenges in the future. Genetic algorithms have the potential to shine in scenarios involving larger datasets and intricate optimization objectives, making them a valuable addition to our arsenal of problem-solving tools.

As the reservation management system continues to evolve and cater to the unique needs of the LCC department, it also serves as a testament to our dedication to leveraging cutting-edge technology and methodologies to deliver innovative solutions. This commitment to growth and learning positions us as a valuable asset in the field of software engineering, poised to address diverse challenges and drive progress in the academic and technological landscape.

6.2 Future lines of Research

One promising avenue for future research lies in exploring alternative optimization algorithms. While the genetic algorithm implemented in this system has proven effective, there is room to investigate other approaches, such as exhaustive search methods. An exhaustive approach could provide a benchmark to evaluate the efficiency of the genetic algorithm and determine whether it is indeed the optimal solution for this specific problem. Comparative studies between different optimization techniques can shed light on their respective strengths and weaknesses, ultimately leading to more informed algorithm choices in reservation management systems.

Furthermore, exploring the integration of actuators and implementing liveliness checks within container orchestration systems like Docker and Kubernetes opens up intriguing avenues of research. Ensuring the robustness and real-time health monitoring of these systems can enhance their reliability and resilience. Investigating ways to automate actions based on the liveliness status can contribute to more efficient and fault-tolerant containerized environments.

Furthermore, the scalability of the system should be a focus of future research. As the user base and dataset sizes grow, the system's performance and efficiency become paramount. Ex-

ploring techniques for distributed computing, parallelization, and cloud-based solutions can ensure that the system remains responsive and capable of handling large-scale academic operations. Research in this direction can lead to a more robust infrastructure that accommodates the evolving needs of educational institutions.

In conclusion, the journey of research and development in reservation management systems continues to unfold. The system presented here serves as a foundation upon which future innovations can be built. By investigating alternative algorithms, embracing AI technologies, and addressing scalability challenges, researchers can unlock new horizons in academic reservation management, making educational resources more accessible and optimizing their utilization.

7

Conclusiones y Líneas Futuras

7.1 Conclusiones

La finalización del sistema de gestión de reservas marca un hito significativo, no solo en la satisfacción de las necesidades específicas del departamento de Lenguajes y Ciencias de la Computación (LCC), sino también en la provisión de una infraestructura robusta que puede expandirse y personalizarse aún más para satisfacer las necesidades cambiantes. Este proyecto representa un testimonio de la versatilidad y adaptabilidad de las soluciones de software para simplificar procesos complejos en instituciones educativas.

Uno de los principales logros de este proyecto es su capacidad para empoderar al departamento de LCC con un conjunto completo de herramientas para gestionar eficientemente las reservas de laboratorios. La interfaz de usuario amigable, respaldada por una sólida infraestructura, simplifica el proceso de reserva, minimiza conflictos y mejora la productividad general. La asignación dinámica de recursos, incluidos laboratorios y equipos, garantiza una utilización óptima de los recursos, en línea con los objetivos del departamento de eficiencia y optimización de recursos.

La capacidad para crear, editar y eliminar semestres, importar reservas desde archivos CSV y visualizar asignaciones y conflictos proporciona al departamento de LCC un control y visibilidad sin precedentes sobre su proceso de gestión de laboratorios. Además, la capacidad de extensión del sistema permite la integración de funciones y características adicionales adaptadas a las necesidades únicas del departamento, garantizando su relevancia y adaptabilidad a largo plazo.

A lo largo del desarrollo de este proyecto, la exploración de algoritmos genéticos como

técnica de optimización agrega una dimensión valiosa a nuestra habilidad como ingenieros de software. La aplicación de algoritmos genéticos, aunque no siempre sea el enfoque más directo, ofrece perspectivas para resolver eficientemente problemas complejos de optimización. La elección de emplear algoritmos genéticos demuestra un compromiso en dominar técnicas avanzadas en diseño de algoritmos y optimización.

En retrospectiva, aunque un método exhaustivo podría haber sido suficiente para la escala de este proyecto, la búsqueda de algoritmos genéticos muestra una mentalidad orientada hacia el futuro y un interés en abordar desafíos más sustanciales en el futuro. Los algoritmos genéticos tienen el potencial de destacar en escenarios que involucran conjuntos de datos más grandes y objetivos de optimización intrincados, convirtiéndolos en una adición valiosa a nuestro conjunto de herramientas para resolver problemas.

A medida que el sistema de gestión de reservas continúa evolucionando y atendiendo a las necesidades únicas del departamento de LCC, también sirve como un testimonio de nuestra dedicación para aprovechar tecnología de vanguardia y metodologías para ofrecer soluciones innovadoras. Este compromiso con el crecimiento y el aprendizaje nos posiciona como un activo valioso en el campo de la ingeniería de software, listos para abordar desafíos diversos y avanzar en el panorama académico y tecnológico.

7.2 Líneas Futuras

Una prometedora vía para futuras investigaciones radica en explorar algoritmos de optimización alternativos. Si bien el algoritmo genético implementado en este sistema ha demostrado ser efectivo, existe espacio para investigar otros enfoques, como los métodos de búsqueda exhaustiva. Un enfoque exhaustivo podría proporcionar un punto de referencia para evaluar la eficiencia del algoritmo genético y determinar si es realmente la solución óptima para este problema específico. Estudios comparativos entre diferentes técnicas de optimización pueden arrojar luz sobre sus respectivas fortalezas y debilidades, lo que finalmente conducirá a una elección más informada de algoritmos en sistemas de gestión de reservas.

Además, explorar la integración de actuadores y la implementación de controles de actividad dentro de sistemas de orquestación de contenedores como Docker y Kubernetes abre interesantes vías de investigación. Garantizar la solidez y el monitoreo del estado en tiempo real de estos sistemas puede mejorar su confiabilidad y resiliencia. Investigar formas de au-

tomatizar acciones basadas en el estado de actividad puede contribuir a crear entornos en contenedores más eficientes y tolerantes a fallos.

Además, la escalabilidad del sistema debe ser un foco de investigación futura. A medida que crece la base de usuarios y el tamaño de los conjuntos de datos, el rendimiento y la eficiencia del sistema se vuelven fundamentales. La exploración de técnicas de cómputo distribuido, paralelización y soluciones basadas en la nube puede garantizar que el sistema siga siendo receptivo y capaz de manejar operaciones académicas a gran escala. La investigación en esta dirección puede llevar a una infraestructura más sólida que se adapte a las necesidades cambiantes de las instituciones educativas.

En conclusión, el viaje de investigación y desarrollo en sistemas de gestión de reservas continúa desarrollándose. El sistema presentado aquí sirve como base sobre la cual se pueden construir futuras innovaciones. Mediante la investigación de algoritmos alternativos, la adopción de tecnologías de IA y el abordaje de desafíos de escalabilidad, los investigadores pueden abrir nuevas perspectivas en la gestión académica de reservas, facilitando el acceso a recursos educativos y optimizando su utilización.

References

[1] Apache. *Maven*. URL: https://maven.apache.org/ (visited on 03/15/2021).

[2] day.js. *day.js*. URL: https://day.js.org/ (visited on 06/23/2023).

[3] DBeaver. *DBeaver*. URL: https://dbeaver.io/ (visited on 06/23/2023).

[4] DiUS. *JavaFaker*. URL: https://github.com/DiUS/java-faker (visited on 06/23/2023).

[5] Docker. *Docker Desktop*. URL: https://www.docker.com/products/docker-desktop/ (visited on 03/15/2021).

[6] ESLint. *ESLint*. URL: https://eslint.org/ (visited on 06/23/2023).

[7] FlywayDB. *FlywayDB*. URL: https://flywaydb.org/ (visited on 06/23/2023).

[8] Git. *Git*. URL: https://git-scm.com/ (visited on 10/03/2021).

[9] GitHub. *GitHub*. URL: https://github.com/ (visited on 10/03/2021).

[10] GitHub. *GitHub Copilot*. URL: https://copilot.github.com/ (visited on 06/23/2023).

[11] JetBrains. *IntelliJ IDEA*. URL: https://www.jetbrains.com/idea/ (visited on 03/15/2021).

[12] JUnit. *JUnit Jupiter*. URL: https://junit.org/junit5/ (visited on 03/15/2021).

[13] Project Lombok. *Project Lombok*. URL: https://projectlombok.org/ (visited on 06/23/2023).

[14] Microsoft. *Visual Studio Code*. URL: https://code.visualstudio.com/ (visited on 10/03/2021).

[15] npm. *dotenv*. URL: https://www.npmjs.com/package/dotenv (visited on 06/23/2023).

[16] npm. *react-csv*. URL: https://www.npmjs.com/package/react-csv (visited on 06/23/2023).

[17] npm. *react-papaparse*. URL: https://www.npmjs.com/package/react-papaparse (visited on 06/23/2023).

[18] Oracle. *Java*. URL: https://www.java.com/es/ (visited on 06/23/2023).

[19] Eugen Paraschiv. *Schedule a Task Using Cron Expressions*. URL: https://www.baeldung.com/spring-scheduled-tasks#schedule-a-task-using-cron-expressions (visited on 08/11/2023).

[20] PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (visited on 06/23/2023).

[21] Prettier. *Prettier*. URL: https://prettier.io/ (visited on 06/23/2023).

[22] Scrum. *Scrum with Kanban*. URL: https://www.scrum.org/scrum-kanban (visited on 06/23/2023).

[23] SLF4J. *SLF4J - Simple Logging Facade for Java*. URL: http://www.slf4j.org/ (visited on 06/23/2023).

[24] Meta Open Source. *React*. URL: https://react.dev/ (visited on 06/23/2023).

[25] Spring. *Spring Boot*. URL: https://spring.io/ (visited on 03/15/2021).

[26] springdoc-openapi. *springdoc-openapi*. URL: https://springdoc.org/ (visited on 06/23/2023).

[27] Testcontainers. *Testcontainers*. URL: https://www.testcontainers.org/ (visited on 06/23/2023).

[28] Trello. *Trello*. URL: https://trello.com/ (visited on 03/15/2021).

[29] TypeScript. *TypeScript*. URL: https://www.typescriptlang.org/ (visited on 06/23/2023).

[30] Material UI. *Material UI*. URL: https://mui.com/ (visited on 06/23/2023).

[31] Material UI. *Material UI Theme*. URL: https://mui.com/material-ui/customization/theming/ (visited on 06/23/2023).

[32] UMA - Department of Languages and Computer Sciences. *Reservations Platform*. URL: https://lcc.reservas.aulas.uma.es (visited on 07/04/2023).

[33] Vite. *Vite*. URL: https://vitejs.dev/ (visited on 06/23/2023).

[34] *Wireframe*. URL: https://wireframe.cc/ (visited on 06/23/2023).

Appendix A
Manual de Despliegue

Manual de Despliegue

Paso 1: Localizar los archivos de la aplicación

Para comenzar con el despliegue es necesario ubicarse en la carpeta donde se encuentra las siguientes carpetas (descomprimidas) y archivos:

- reservas-laboratorios-api
- reservas-laboratorios-ui
- docker-compose.yml

En caso de no disponer de los archivos, se puede descargar desde el siguiente enlace: https://github.com/tomgreef/reservas-laboratorios

Paso 2: Generar el Archivo JAR de Spring Boot

1. **Usando la Terminal:**
    - Abre una terminal.
    - Navega a la carpeta del proyecto `reservas-laboratorios-api`.
    - Ejecuta el comando:

      ```
      ./mvnw clean package
      ```

    - El archivo .jar se generará en el directorio `target`.
2. **Usando Eclipse:**
    - Abre Eclipse.
    - Importa el proyecto `reservas-laboratorios-api`.
    - Haz clic derecho en el proyecto en el Explorador de Proyectos.
    - Selecciona "Run As" > "Maven build...".
    - En el campo "Goals", ingresa: `clean package`.
    - Haz clic en "Run".
    - El archivo .jar se generará en el directorio `target`.
3. **Usando IntelliJ IDEA:**
    - Abre IntelliJ IDEA.
    - Importa el proyecto `reservas-laboratorios-api`.
    - En la ventana de Herramientas de Maven, busca tu proyecto.
    - Expande el árbol de carpetas de tu proyecto.
    - Selecciona "Lifecycle" > "clean".
    - Luego, selecciona "Lifecycle" > "package".
    - El archivo .jar se generará en el directorio `target`.

Paso 3: Ejecutar la Solución con Docker

1. Asegúrate de tener Docker Desktop instalado en tu máquina y arrancado.
2. Abre una terminal y navega al directorio que contiene tu archivo `docker-compose.yml`.
3. Ejecuta el siguiente comando para construir y arrancar los contenedores Docker:

   ```
   docker-compose up -d --build
   ```

La configuración de Docker Compose creará y arrancará los un conjunto de contenedores llamado "reservas-laboratorios", que contendrá los siguientes contenedores: "api", "ui", "postgre_db".

Acceso a la Aplicación:

- La interfaz de usuario (UI) estará accesible en http://localhost. El contenedor "ui" mapea su puerto 80 al host.
- La API del back-end estará accesible en http://localhost:8080. El contenedor "api" mapea su puerto 8080 al host. También dispone de una interfaz de swagger accessible en http://localhost:8080/swagger-ui/index.html.

Detener los Contenedores:

- Para detener los contenedores Docker, acceda a Docker Desktop y pare el contenedor `reservas-laboratorios`

Actualizar la Aplicación:

- Para actualizar la aplicación, modifica el código fuente en los directorios de proyectos respectivos (reservas-laboratorios-api o reservas-laboratorios-ui). En el caso de modificar el código fuente de la API, es necesario generar nuevamente el archivo .jar de Spring Boot.
- Reconstruye el contenedor Docker correspondiente ejecutando nuevamente `docker-compose up -d --build`.

Limpieza:

- Para eliminar los contenedores y recursos asociados, ejecuta desde el directorio que contiene tu archivo `docker-compose.yml`:

  ```
  docker-compose down
  ```

Nota: Este manual proporciona instrucciones para ejecutar la solución Docker y generar el archivo .jar necesario para el proyecto Java usando varias herramientas. Asegúrate de ajustar cualquier detalle específico a tu entorno y flujo de trabajo.

Appendix B
Manual de Usuario

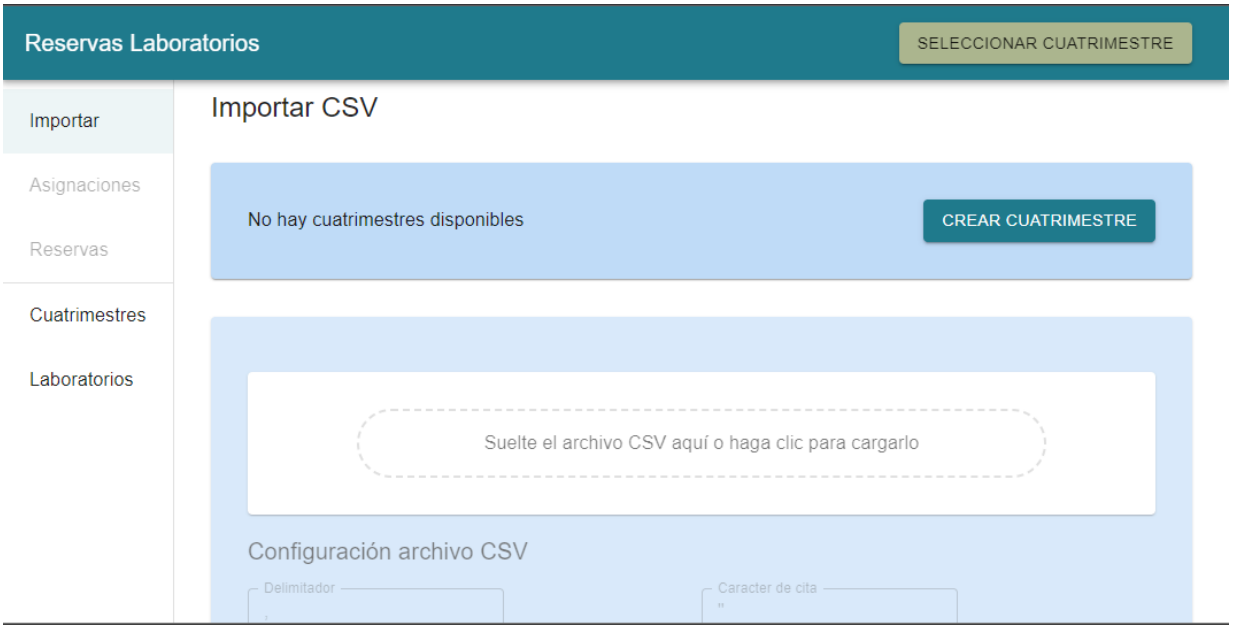
Manual de Usuario

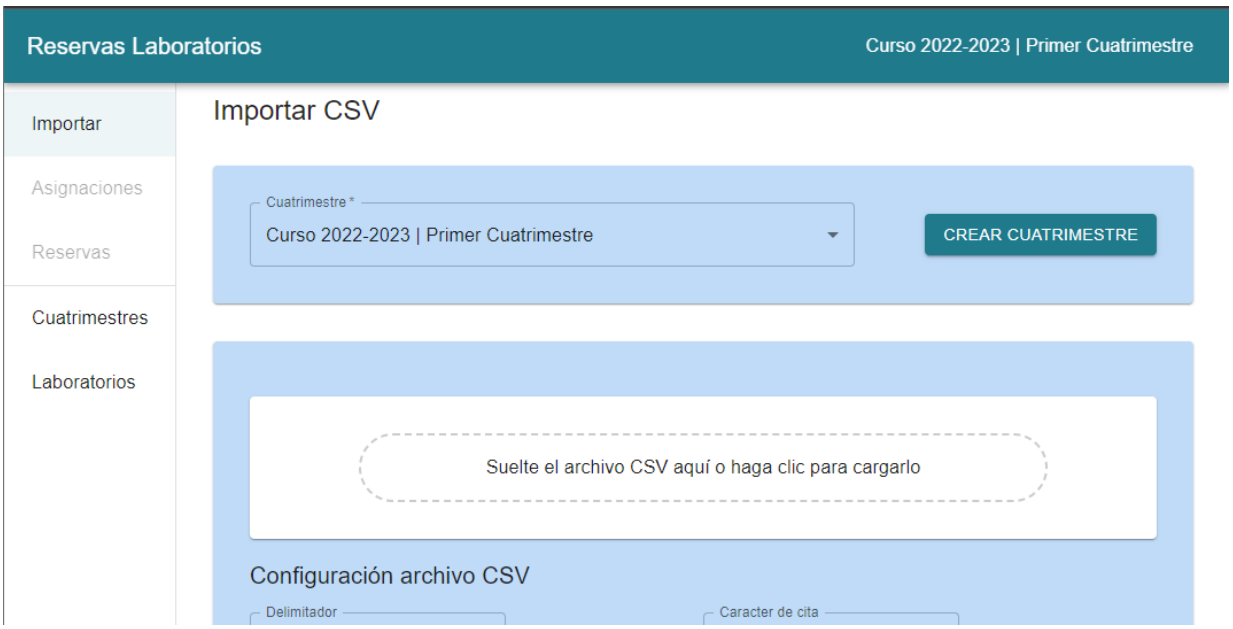
Página de Importar

Crear un Cuatrimestre

Para crear un cuatrimestre, tienes dos opciones:

1. Haz clic en el botón "CREAR CUATRIMESTRE" en la página de importar.
2. Si ya tienes un cuatrimestre seleccionado, selecciona el nombre del cuatrimestre en la barra superior y luego haz clic en el botón "SELECCIONAR CUATRIMESTRE" en la misma barra superior.

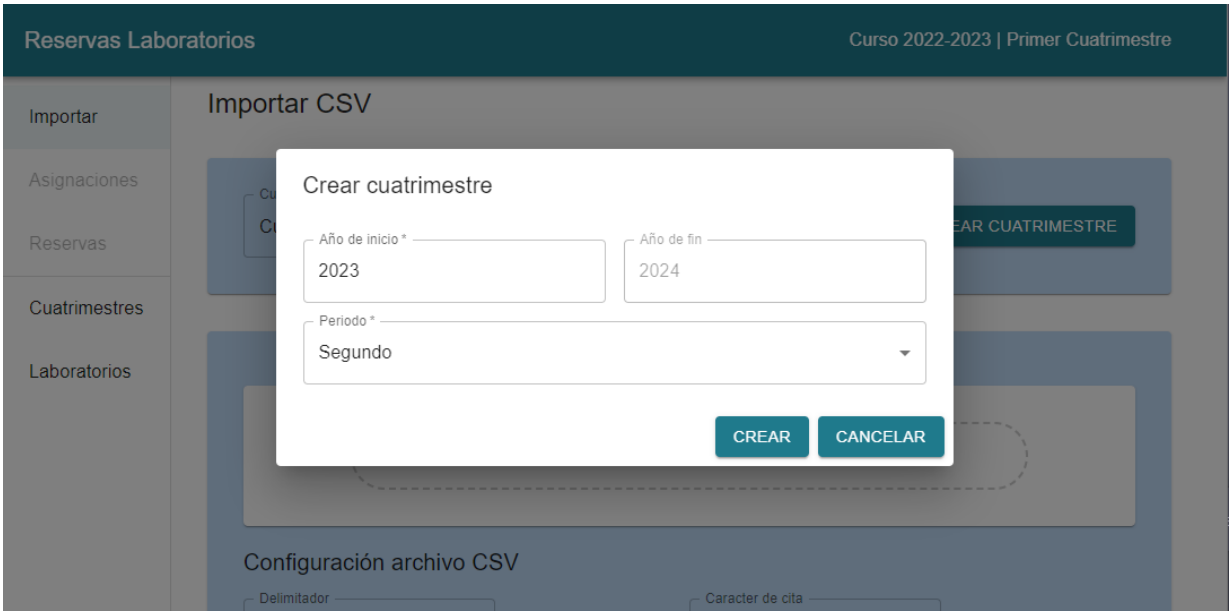




En cualquiera de los casos anteriores, se abrirá un diálogo que te permite ingresar la información del nuevo cuatrimestre:

- Año (4 dígitos).
- Selecciona entre dos opciones de periodos: primero o segundo.

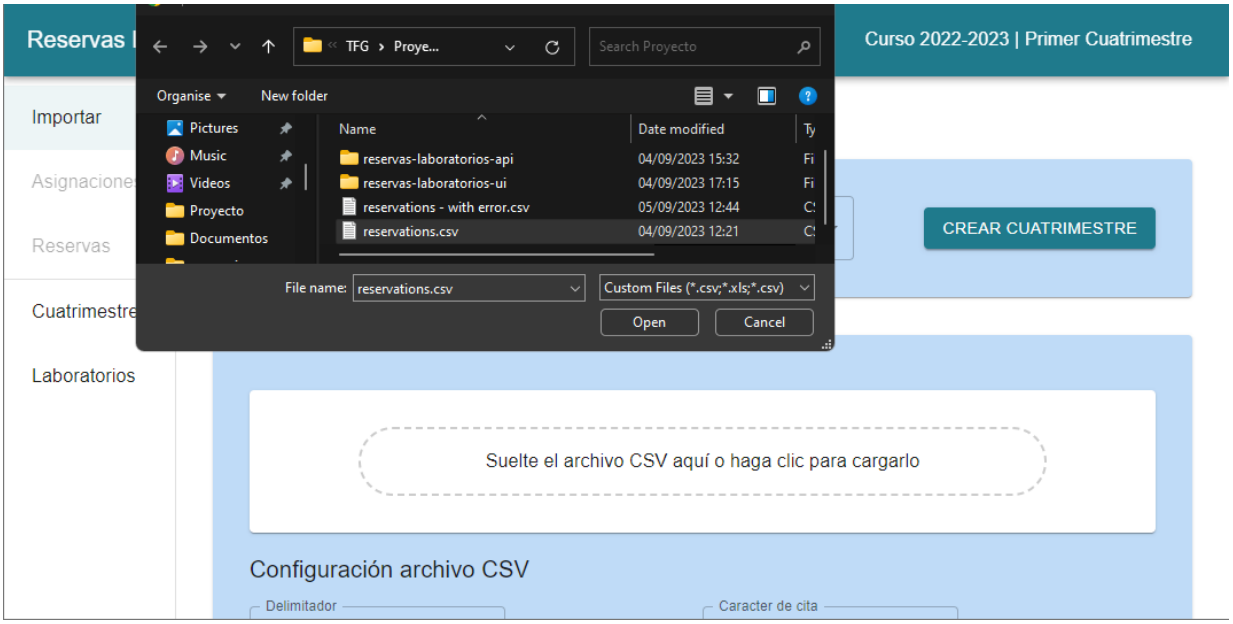
Haz clic en "CREAR" para crear el nuevo cuatrimestre y seleccionarlo como cuatrimestre activo. Si deseas cancelar la creación del cuatrimestre, haz clic en "CANCELAR". Si los datos ingresados son incorrectos o ya existen, se mostrarán los errores correspondientes.



Importar Reservas desde un Archivo CSV

Una vez que tengas un cuatrimestre seleccionado, podrás importar reservas desde un archivo CSV.

- **Subir Archivo CSV**: Para cargar reservas, haz clic en la sección de "Importar archivo CSV" en la página de importar o arrastra el archivo CSV y suéltalo en la caja de carga.

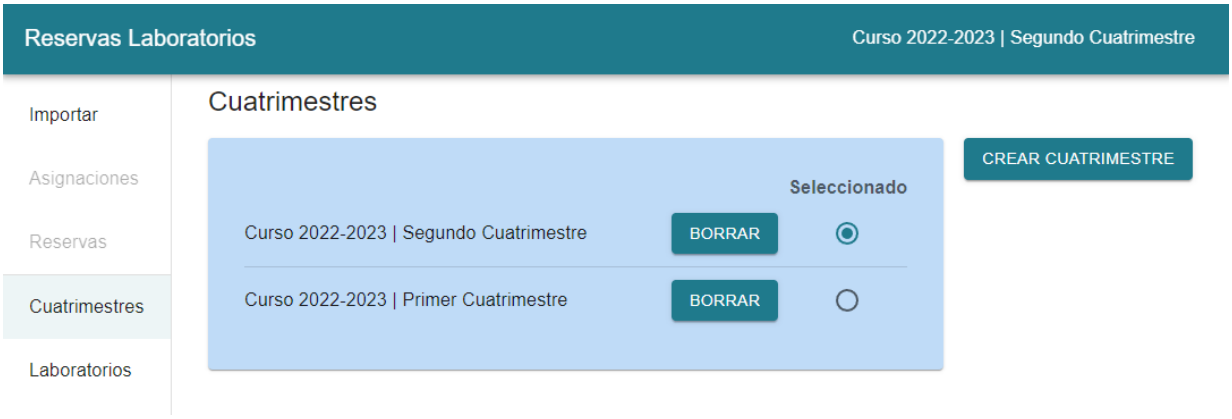


- **Configura el Delimitador y el Carácter de Cita**: Puedes cambiar la configuración del delimitador y el carácter de cita en los dos campos ubicados debajo del bloque de subida de archivo. Asegúrate de que coincidan con el formato de tu archivo CSV.
- **Carga de Reservas**: Una vez que hayas cargado el archivo CSV y configurado los parámetros, el sistema comenzará a cargar las reservas de laboratorios. Al finalizar, recopilará la lista de laboratorios preferentes y verificará que existen en el sistema, de lo contrario, crearán automáticamente los laboratorios faltantes con los valores de la reserva correspondiente.
- **Validación de Datos**:
    - Si hubiera algún error en el formato del CSV o los datos fueran incorrectos, la aplicación mostrará los errores por fila en la lista de reservas. Debes corregir los errores antes de continuar.
    - En caso de no haber errores, la aplicación te redirigirá automáticamente a la página de asignaciones, donde comenzará la ejecución del algoritmo y mostrará el estado de la tarea.

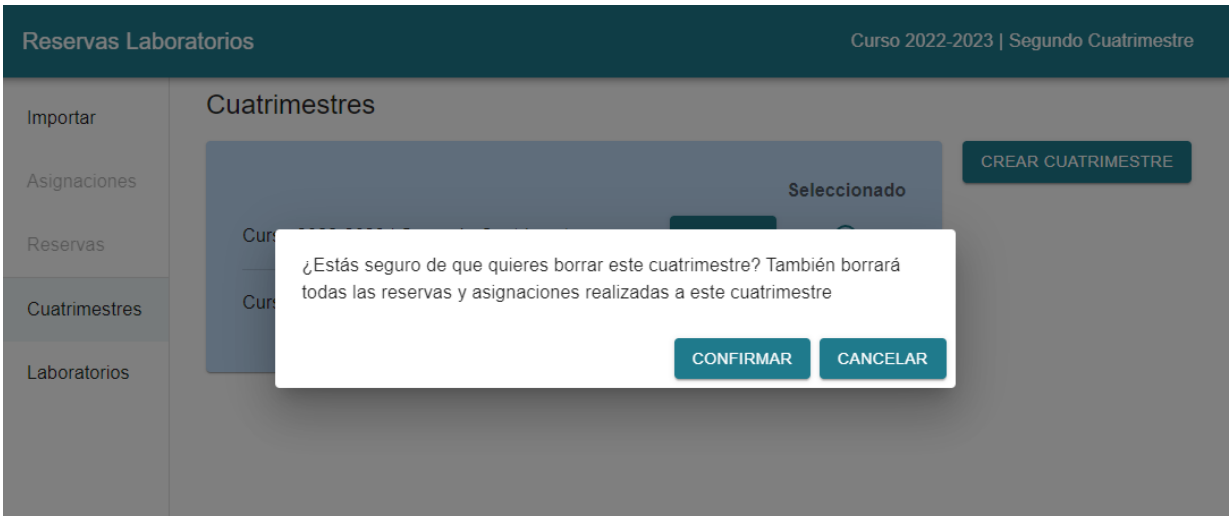
Página de Cuatrimestres

En la página de cuatrimestres, puedes realizar las siguientes acciones:

- **Crear un Cuatrimestre**: Para crear un nuevo cuatrimestre, sigue las instrucciones mencionadas en la sección correspondiente.
- **Cambiar la Selección de Cuatrimestre**: Puedes cambiar la selección de cuatrimestre utilizando el "radio button". Simplemente selecciona el cuatrimestre que desees en la lista y este se activará como cuatrimestre actual.



- **Borrar un Cuatrimestre**: Si deseas eliminar un cuatrimestre, ten en cuenta que esto también eliminará todas las reservas y asignaciones relacionadas con ese cuatrimestre. Para hacerlo, sigue estos pasos:
    1. Haz clic en el cuatrimestre que deseas borrar en la lista.
    2. Se abrirá un diálogo de confirmación.
    3. Puedes elegir entre "Cancelar" para abortar la operación o "Confirmar" para borrar el cuatrimestre y todas las entidades relacionadas.



Recuerda que estas acciones son importantes y pueden afectar la información de tu aplicación, así que úsalas con cuidado.

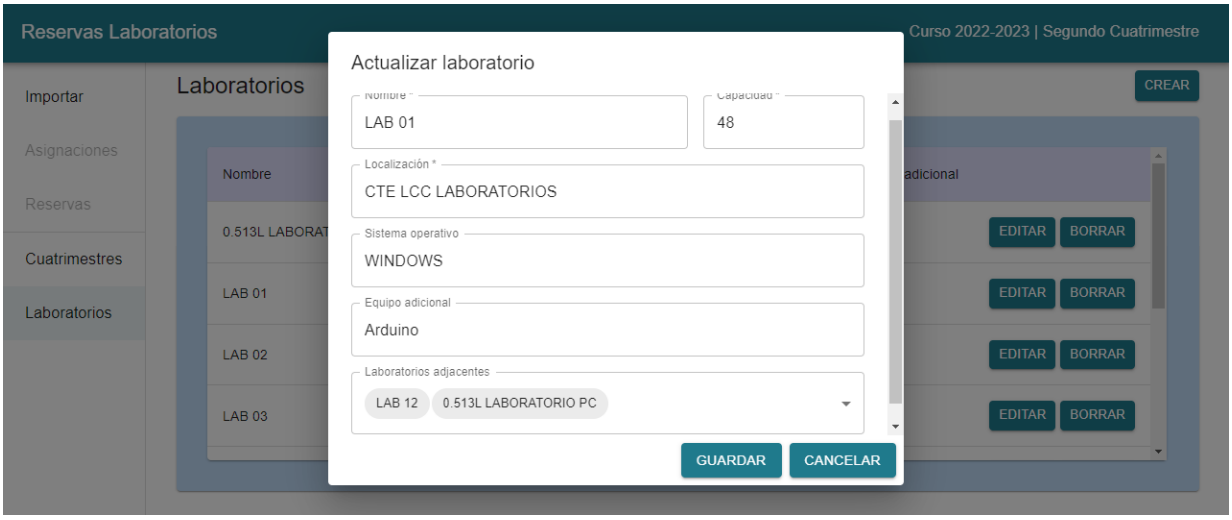
Página de Laboratorios

En la página de laboratorios, puedes realizar las siguientes acciones:

Crear Laboratorios Nuevos

Para crear un nuevo laboratorio, sigue estos pasos:

1. Haz clic en el botón "Crear Laboratorio Nuevo". Se abrirá un diálogo con los siguientes campos obligatorios:
    - Nombre*.
    - Capacidad* (número).
    - Localización* (puedes seleccionar una existente o crear una nueva).
    - Sistema Operativo (puedes seleccionar uno existente o crear uno nuevo).
    - Equipo Adicional (puedes seleccionar uno existente o crear uno nuevo).
    - Laboratorios Adyacentes (puedes seleccionar uno o varios laboratorios existentes). Ten en cuenta que este campo creará una relación bidireccional entre los laboratorios seleccionados.



2. Completa los campos con la información relevante.
3. Haz clic en "Guardar" para crear el nuevo laboratorio. El laboratorio se añadirá a la lista de laboratorios disponibles.

Editar Laboratorios

Para editar un laboratorio existente, sigue estos pasos:

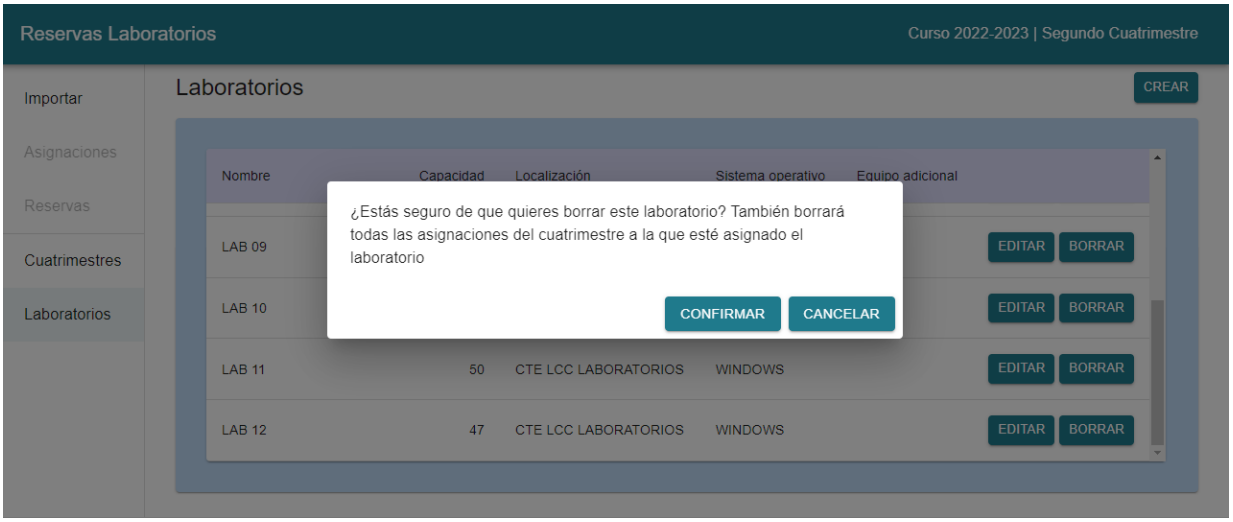
1. Haz clic en el laboratorio que deseas editar en la lista de laboratorios.
2. Se abrirá un diálogo con los campos rellenados con los datos del laboratorio existente.
3. Realiza las modificaciones necesarias en los campos.
4. Haz clic en "Guardar" para actualizar la información del laboratorio.

Borrar Laboratorio

Si deseas eliminar un laboratorio, sigue estos pasos:

1. Haz clic en el laboratorio que deseas borrar en la lista de laboratorios.

2. Se abrirá un diálogo de confirmación.



3. Puedes elegir entre "Cancelar" para abortar la operación o "Confirmar" para borrar el laboratorio. Ten en cuenta que esta acción no se puede deshacer y eliminará permanentemente el laboratorio y sus relaciones.

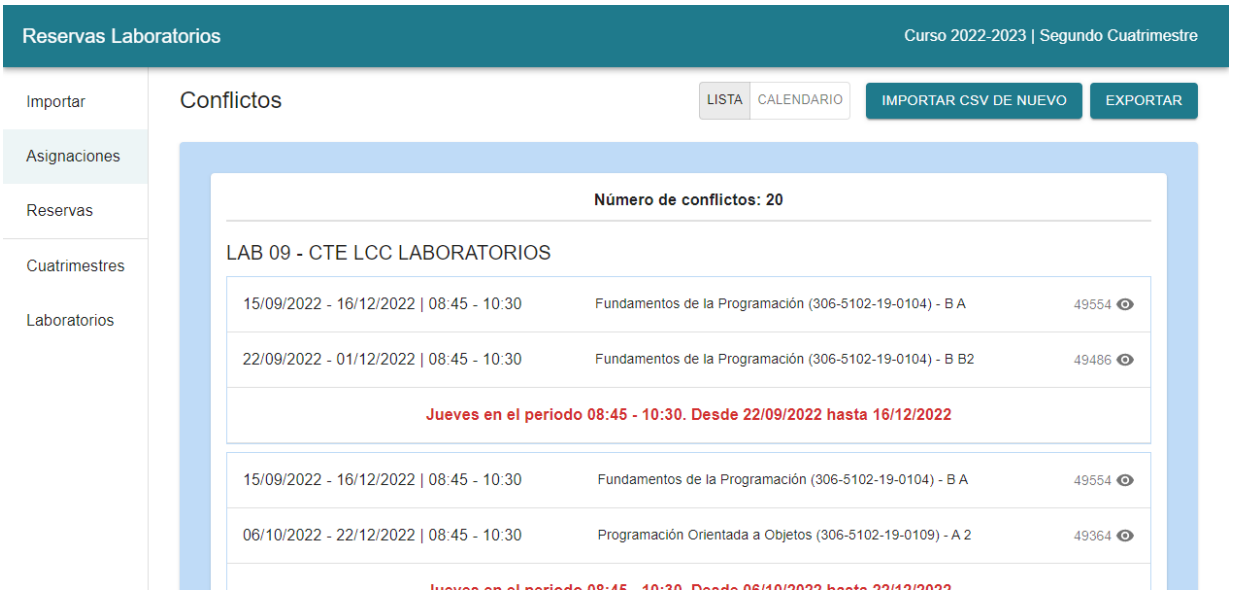
Recuerda que estas acciones son importantes y pueden afectar la configuración de laboratorios en tu aplicación, así que úsalas con cuidado.

Página de Conflictos

En la página de Conflictos podrás ver una lista de conflictos relacioandos al semestre seleccionado, luego puedes realizar las siguientes acciones:

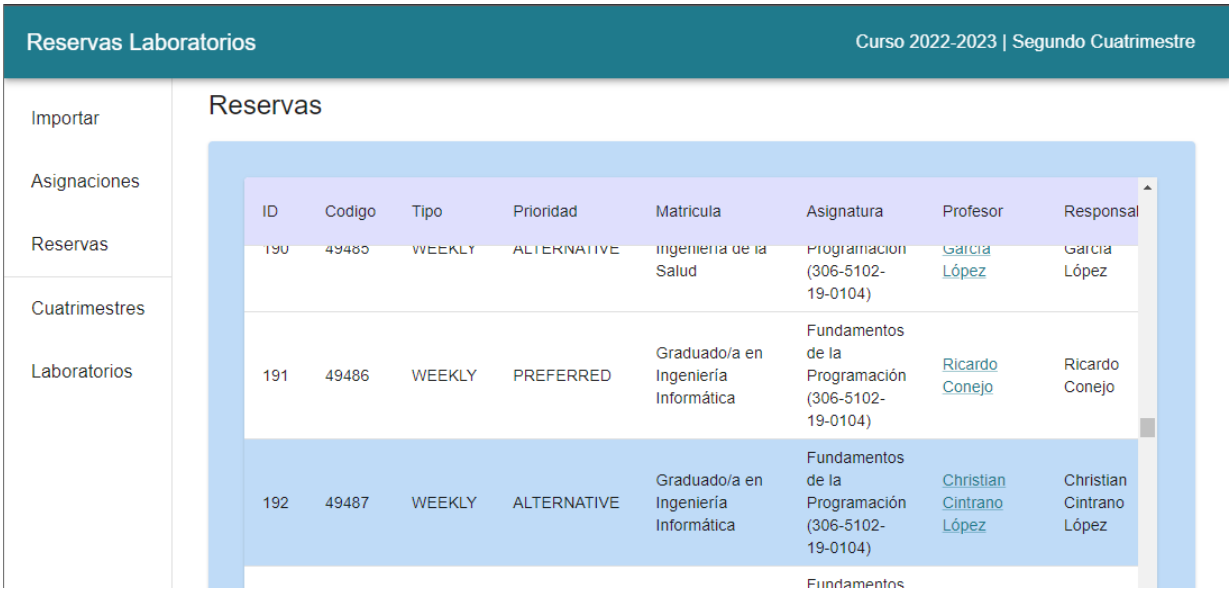
Exportar la Lista de Asignaciones

Para exportar la lista de asignaciones, haga clic en el botón "EXPORTAR". Se descargará un archivo con la lista de asignaciones en un formato específico.



Ver la Reserva en Detalles

Para ver más detalles sobre una de las reservas de un conflicto específico, haz clic en el código con el icono lateral derecho correspondiente en la lista de conflictos. Se te redirigirá a la página de reservas, donde se resaltará esa reserva.



Además, habrán conflictos donde una o ambas reservas tienen un color de fondo verde. Estas son reservas que tienen el tipo de reserva "Alternativo", y por lo tanto son que se pueden solventar con más facilad.

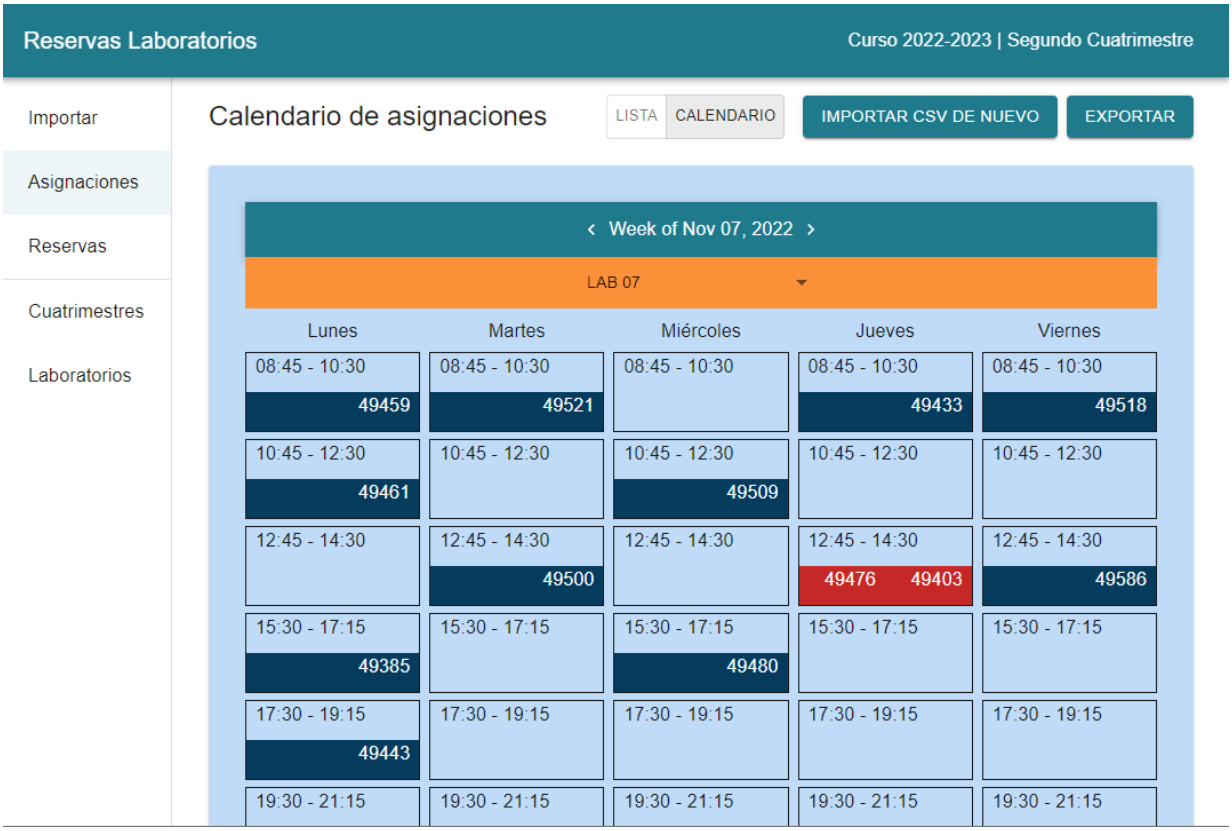
Volver a Importar

Si deseas volver importar un nuevo archivo CSV, sigue estos pasos, haz clic en "IMPORTAR CSV DE NUEVO". Se te redirigirá a la página de importar, donde puedes importar el archivo de nuevo.

Cambiar a la Vista de Calendario

Para cambiar a la vista de calendario y ver las asignaciones en formato de tabla, sigue estos pasos:

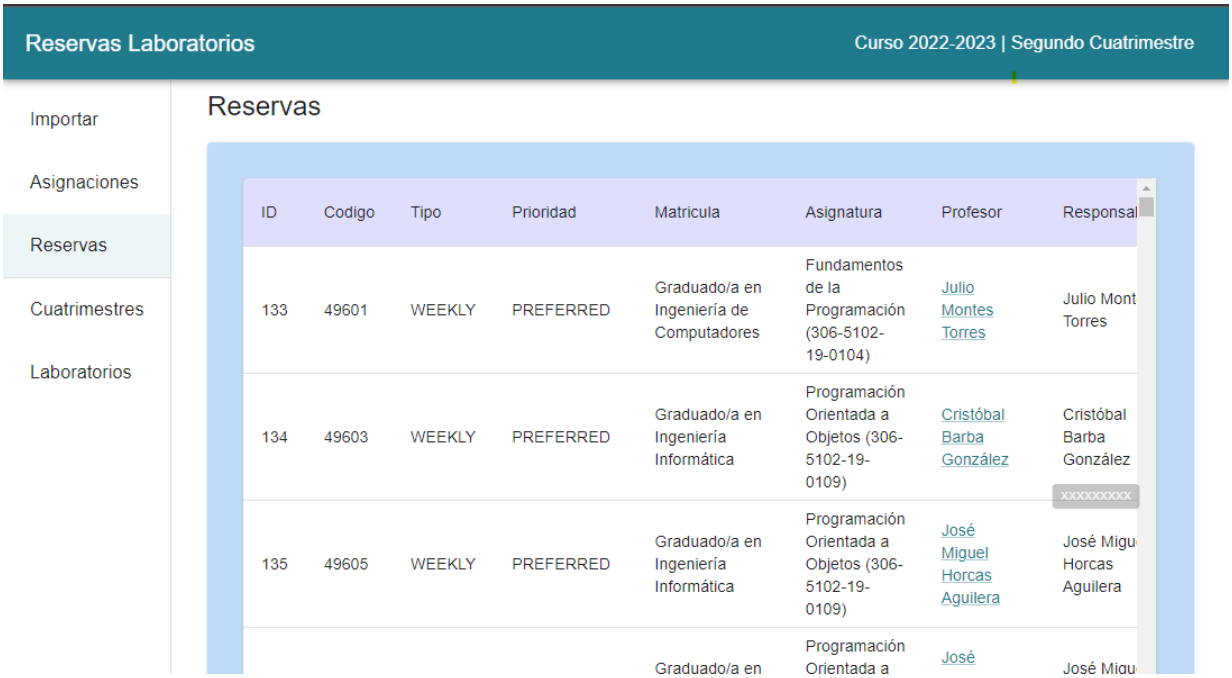
1. Pulsa el interruptor correspondiente para cambiar a la vista de calendario.
2. En la tabla, verás los días de lunes a viernes de una semana y el laboratorio seleccionado. Las franjas horarias en azul representan una asignación, mientras que las franjas en rojo indican que hay dos o más asignaciones para esa misma franja (conflicto).



Página de Reservas

En la página de Reservas, puedes ver la tabla de reservas correspondiente al cuatrimestre seleccionado. La tabla mostrará información relevante sobre cada reserva.

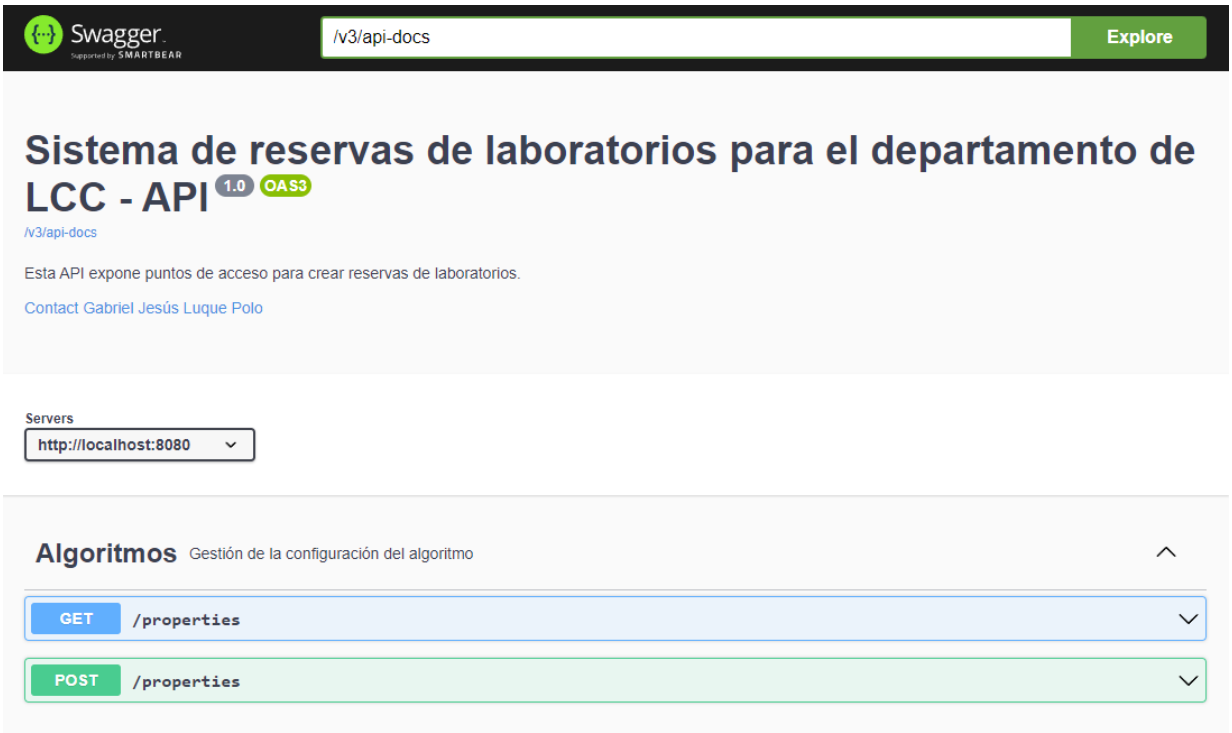
Se puede hacer click sobre el campo de profesor asociado a una reserva. Esto abrirá el cliente de correo predeterminado de tu sistema con la dirección de correo. Además, si dejas el cursor sobre el nombre del responsable, se mostrará un "tooltip" con el telefono de dicho responsable.



Endpoint Swagger del Back-End

En este punto de acceso puedes cambiar las propiedades de configuración del algoritmo genético para ajustar el comportamiento del sistema. Esto puede incluir configuraciones como la tasa de mutación, la población inicial, el número de generaciones, y más. Consulta la documentación del sistema para obtener detalles sobre cómo configurar estas propiedades y su impacto en el algoritmo genético.

Además, puedes realizar operaciones CRUD en diferentes entidades de la aplicación, dependiendo de las capacidades expuestas por el punto de acceso.



UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA