Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Sistema de Apoio à Criação da Distribuição do Serviço Docente

Pedro Monteiro 97484
Eduardo Fernandes 98512
José Trigo 98597
André Gomes 97541

Projeto em Informática
Licenciatura em Engenharia Informática

June 22, 2022

Abstract

The DSD is a platform that provides DSDers with a better way to plan and execute the distribution of Professors across classes and curricular units. This process is a very complex task and there is no standardized way to perform it, with many departments relying on excel spreadsheets to perform the job at hand.
We aim to offer a better way to solve this complex task by building a practical and powerful platform that lets DSDers assign Professors to classes whilst being able to easily view all necessary information required to elaborate the work distribution.
This document describes the entire project, its components, its architecture and how it should be used.

Acknowledgements

We would like to thank our supervisors, Professor José Vieira and Professor Tomás Oliveira e Silva, for all the help and constant feedback provided throughout the development of the project. We also want to thank the STIC engineers who helped us in the implementation of some features and for the feedback about the application.
Finally, we would like to thank the professors from the department of mechanical engineering who tested our application and were willing to listen to our proposal.

Contents

List of Figures

Abbreviations

DSD Distribuição do Serviço Docente
UC Curricular Unit
JSON JavaScript Object Notation
CSV Comma Separated Value
XLS Excel Binary File Format
DSDer Person who performs DSD

1 Introduction

1.1 Context

This is the informatics final project, that focus on assigning teachers to existing classes, which is a very complex and tiring task, due to the information required in the process being spread apart and hard to access.

It was developed throughout the semester and the main objective is to make this whole process faster, simpler and easier.

1.2 Motivation

When it came to choosing a theme for our final project from the list given to us, we thought that most of the themes given, while having very similar difficulties, lacked in impact towards our lives, we wanted this project to leave a mark, and what a better topic for that than a platform for our university, which we hope will help our teachers in making the fatiguing task of creating the DSD a simpler piece of work.

1.3 Goals

The main goal of this project was to create a decision support tool with the following features:

- Integrated access to each teacher's DSD history;
- Tool for surveying the wishlist of teachers in which they indicate their teaching preferences;
- Development of a graphical tool for the construction of the DSD;
- Formalization of the rules that a DSD must obey to allow a automatic validation before its passage to the elaboration of the schedules;

1.4 Document Structure

In addition to the introduction, this document has five more chapters. In chapter 2, it is described the state of the art on resource allocation and all the technologies involved within this project. Chapter 3 presents the elicitation requirements process and the system's requirements and architecture. On chapter 4, the implementation of the working prototype is described in its four main parts: the user interface, the API, the database and the docker containers. Then, chapter 5 presents the usability tests conducted during this project, including its sample, method and results evaluation. At last, chapter 6 makes an overview of the work done on this project, its main results, conclusions and future work.

2 State of The Art

2.1 Related Projects

The goal of this section is to present work whose domain intersects with this project. Several research was done on resource allocation softwares. An overview of those systems is presented below.

2.1.1 Float

In this project, Glenn Rogers says "Float provides the most accurate view of your capacity to plan work more efficiently".

Float is a resource management platform to plan your team's best work. With this software you are able to plan your projects based on your team's real capacity. Set project budgets using total hours or fees and accurately forecast the resources you will need. You can split, insert, or replace tasks with a single-click. All changes occur in real time and are tracked via your team's activity feed.

2.1.2 SoftExpert

SoftExpert is the market leader in solutions for integrated compliance management, innovation and digital transformation, with more than 2,000 customers and 300,000 users in more than 40 countries around the world.

Like Float, this tool is used for managing and allocating resources.

The problem is that DSD is a very specific task and none of these software can satisfy the resolution of it. This task contains several specific aspects such as sabbaticals or how to assign teachers to classes and so you need a tool built with this problem in mind.

Figure 1: Related work softwares.

2.2 Technology

This section will present the technologies used during this project, by making a brief description of their purpose and main features. We will describe their main advantages and disadvantages, as well as the main reasons leading to their choice.

2.2.1 ReactJS

ReactJS is a free, open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. There are many reasons to why we chose to use

React, the main being the flexibility, performance, being able to reuse components and for it being easy to learn.

2.2.2 FastAPI

FastAPI is an open source Python web development framework that makes building APIs easier.

We chose to use it due to its fast performance and intuitive use. Also, with FastApi and SwaggerUI is possible to have an interactive documentation interface for APIs.

Figure 1 represents the documentation interface for some of our endpoints.

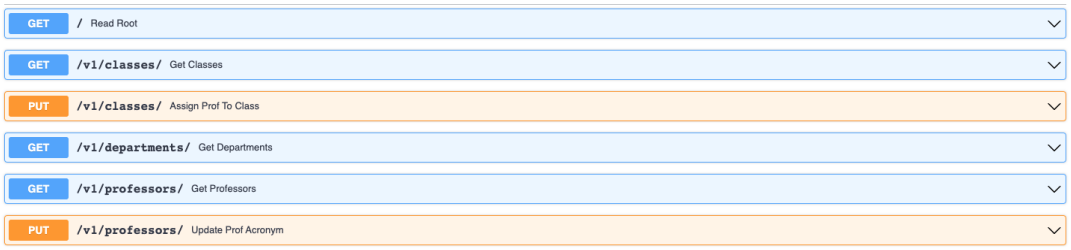

Figure 2: Project endpoints documentation.

2.2.3 MySQL

MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL). Is one of the most recognizable technologies in the modern big data ecosystem.

As MySQL was developed in C/C++, its transfer between systems is facilitated, having interface modules for multiple languages, such as Java, Python, PHP, etc.

It has high execution and storage power, and tables can store large volumes of data. Also, allows high-speed data access.

Among all the advantages we chose this technology due to the familiarity with it, as it has been used by us in previous projects and throughout the course.

2.2.4 Docker

Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

We decided on using Docker due to it leading to an improved developer productivity: Compared to VMs, containers are faster and easier to deploy, provision and restart. This makes them ideal for use in continuous integration and continuous delivery (CI/CD) pipelines and a better fit for development teams adopting Agile and DevOps practices.

Figure 3: Technologies used.

3 System Requirements and Architecture

3.1 System Requirements

This section presents the system requirements specification, as a result of the first phase of the prototype development. The following subsections begin with a description on the requirements elicitation process, followed by a context description, actors, use-case, non-functional requirements and the system's assumptions and dependencies.

3.1.1 Requirements Elicitation

Due to the complexity of the problem, at the start of the project we had a two week familiarization phase in order to fully understand what was being asked of us. After those two weeks, we concluded that these were the main requirements:

- List information about UCs and Professors
- Allocate teachers to UCs
- Give allocation recommendations
- View professor's hours history
- View professor's wishlist history
- Export DSD in CSV/JSON/Excel format
- Customizable acronyms
- Display useful contacts for the DSD process

3.1.2 Context Description

This subsection presents a description on how the system is expected to be used by our actors. After login in as the DSDer, you should be presented with 4 panels, two of them filled with information (the right one with teachers info and the left one with UCs info), and two of them empty. Upon clicking on a UC the empty panel to the left should fill up with more detailed information regarding that UC, in order to help the DSDer. The same happens regarding teachers and the empty panel to the right.

The DSDer has multiple options to show the information present in our system, due to the various filters and search bars. After having enough information to make a decision, the DSDer can then write a teachers acronym on an empty UC cell in order to assign him to

that class. When done, the DSDer can then go to the 'Validation' tab in order to receive some feedback about the DSD he as created. Then he can export it, either to '.json', '.csv' or '.xls'.

3.1.3 Actors

The actors which will interact with our system can be split into two different categories: the DSDer, a person who is in charge of using the platform to arrange the DSD each year; and the teachers, who will use the platform to fill out the data relative to each one of them.

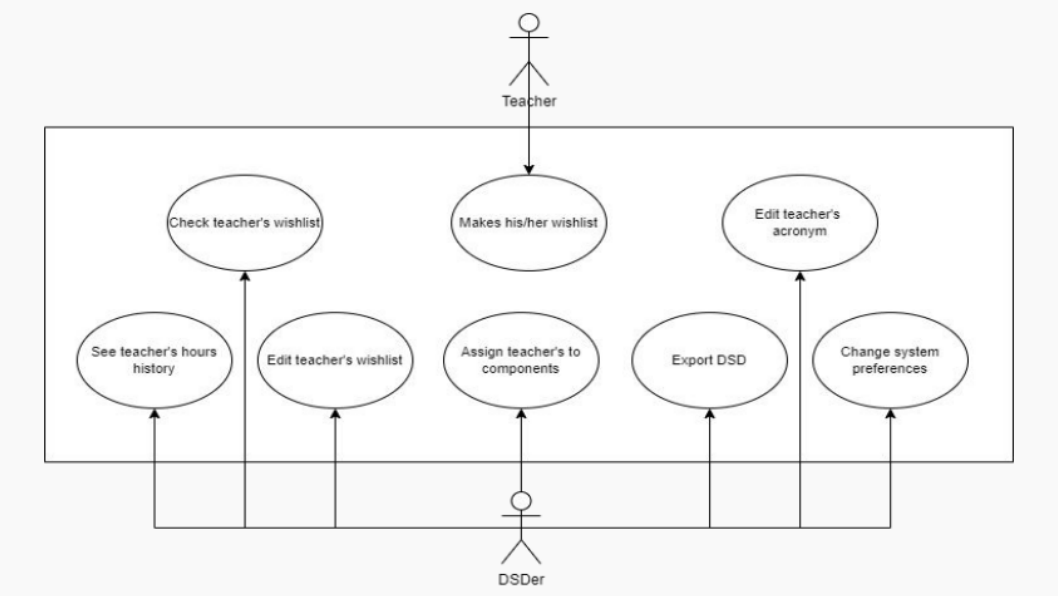

Figure 4: Actors and Use Cases.

3.1.4 Use Cases

- DSDer:
  - Check teacher's wishlist;
  - See teacher's hours history;
  - Edit teacher's wishlist;
  - Assign teachers to classes;
  - Export the DSD;
  - Edit teachers acronyms;
  - Change system preferences;
- Teacher:
  - Make their wishlist;

3.1.5 Non-functional Requirements

- Maintainability: The system must rapidly be able to be restored after a failure occurs;
- Availability: The system, when needed, should not have down-times;
- Performance: The data used in the system must be rapidly available;
- Usability: The interface must be design to correspond with the problem and with the needs of the Actors;
- Data Integrity: The data present in the database must always be accurate and consistent;
- Interoperability: The system (DSD module) is only a part of a bigger system;

3.1.6 Assumptions and Dependencies

In order for the system to work as expected the following assumptions are made:

- The data inserted into the database by the University social services is all correct and always up to date when creating the DSD;
- All the DSDers must be familiarized with working with teacher acronyms rather than with their full names;
- When creating the DSD, all the processes that may lead to information changes, as in teacher hiring, sabbaticals, students grades, etc, are all finished;

3.2 System Architecture

This section presents an overview of the system architecture, describing its domain and technological model. The architecture of our system can be split into 3 parts:

- Frontend
- Backend
- Storage/Database

When it comes to Frontend, we used ReactJS, as previously mentioned, to make our UI. In the Storage/Database section, we relied mainly on the use of MySQL, and, to retrieve all of the data from the database, in the Backend section, we used FastAPI.

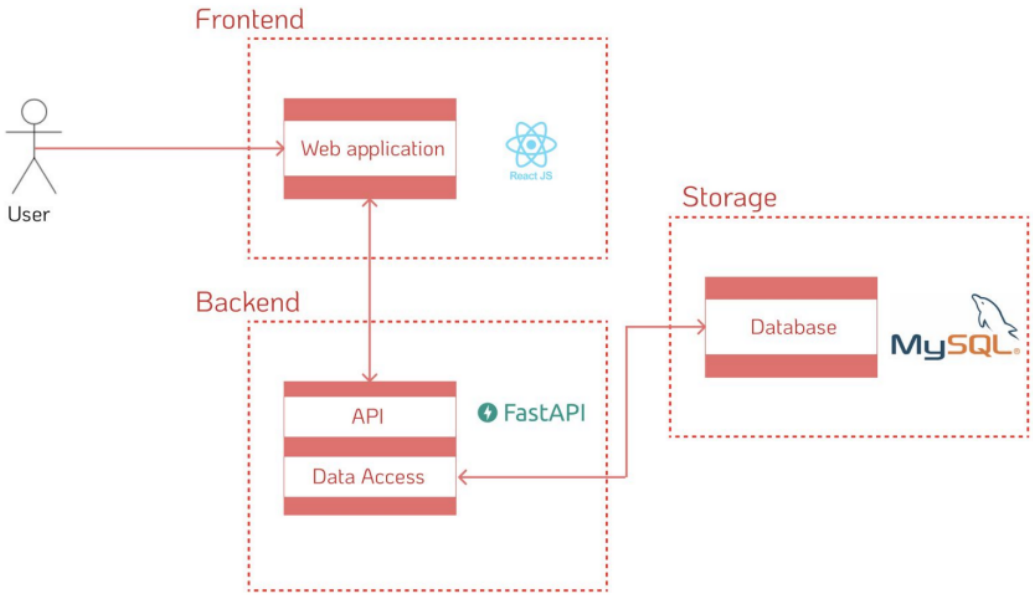

Figure 5: System Architecture.

4 Implementation

4.1 User interface

For the users that would do DSD process, we have created a web based application with ReactJS.

4.1.1 Mock up

After the requirements elicitation and before starting the implementation phase we have created a first set of mock ups with Drawio editor and then a more formal one with Balsamiq editor to establish a skeleton for the main interfaces that are the home page (6, 8 and 9) and the change acronym page (7).

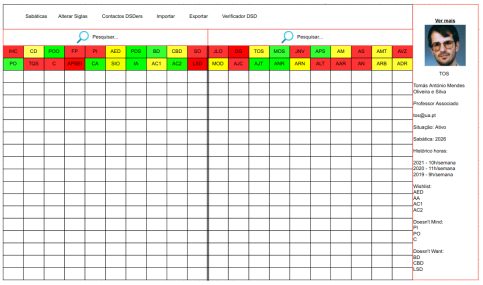

Figure 6: First mock up of the home page showing teacher information.

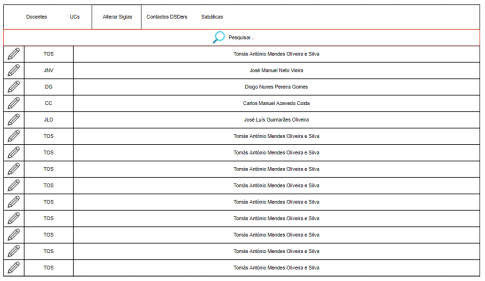

Figure 7: First mock up of the change acronym page.

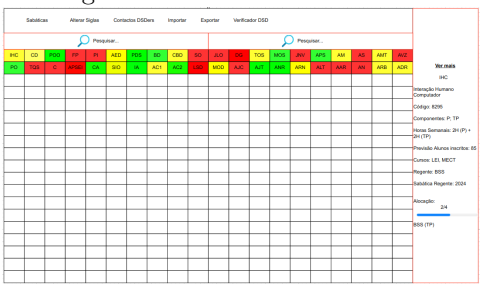

Figure 8: First mock up of the home page showing UC information

Figure 9: Home page mock up done with Balsamiq.

4.1.2 Visual colors

During the mock up construction we have also studied color combinations. As this is a project that is part of a university information system, we decided to adopt a color palette that already exists on other platforms already created for the same. These colors are much more user-friendly as they are pastel colors as we can see in the figure (10).

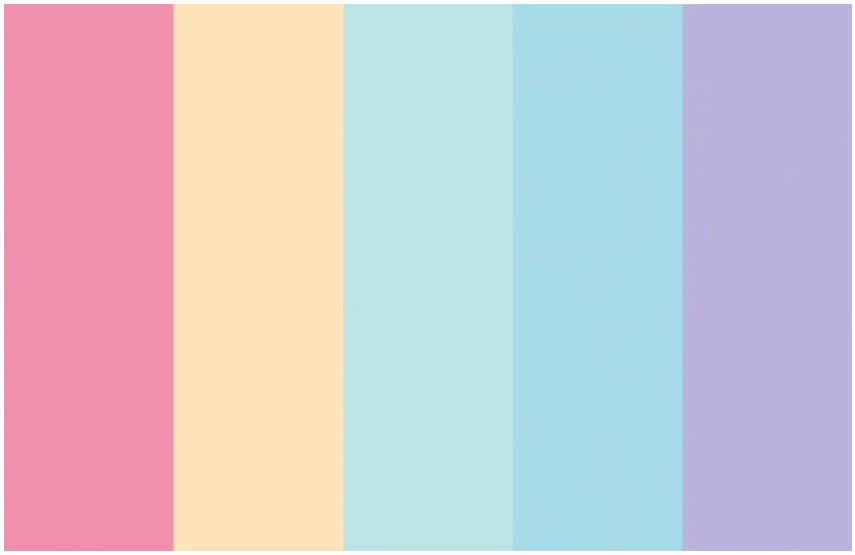

Figure 10: Some colors used in the system.

4.1.3 General application

Sets of web pages (served by ReactJS)

Navbar The user can navigate access different functionalities form here.

Figure 11: General application: Navbar.

As this system is supposed to integrate with PACO, the navbar was inspired by this system. Here exists five main buttons that the user can click. He can:

- Access the home page by clicking the logo of the university;
- Access the page where he can change acronyms;
- Validate the DSD that he already has;
- Export the DSD in different formats;
- Refresh the information from the database.

Home page All important information is here.

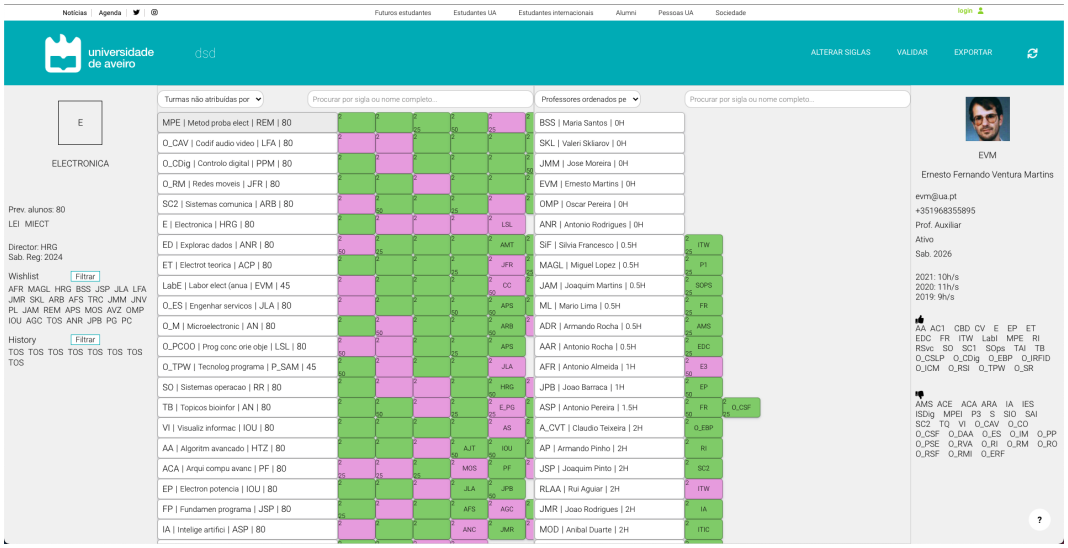

Figure 12: General application: Home page.

Here is where the most important information exists. All the information that is necessary to perform the DSD is mostly loaded on this screen. It is from this page that we can access all the other features that the system has.

Main panel Some teacher's and UC's information is shown here.

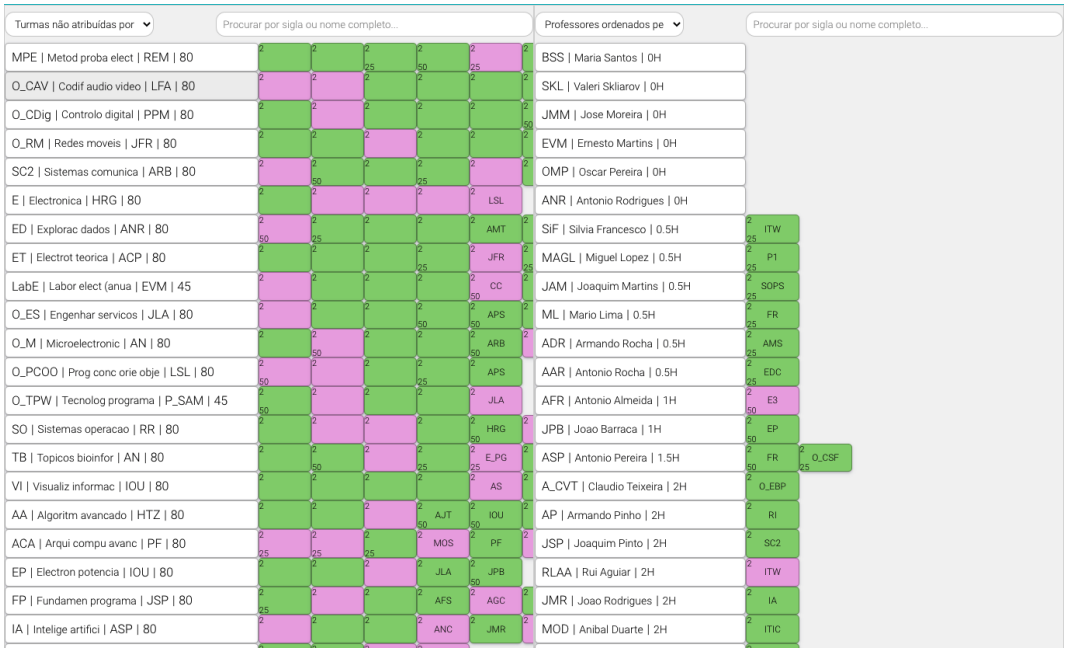

Figure 13: General application: Main panel.

Here is where all the information of a teacher or a UC exists in a compact form. On the left side is where all the UC's are, and the teacher's are on the right side. We can assign a teacher to a class by just writing his/her acronym on the cell pretended. When a teacher is

assigned to a class, automatically the information is updated on the right side of the main panel, therefore this teacher's hours will increase and a cell with the name of the UC to which it was assigned will be shown. Also, if the teacher's hours exceed the limit, then his cell will be colored red, indicating that this teacher already has assigned hours above the expected, as we can see in the figure below (14).

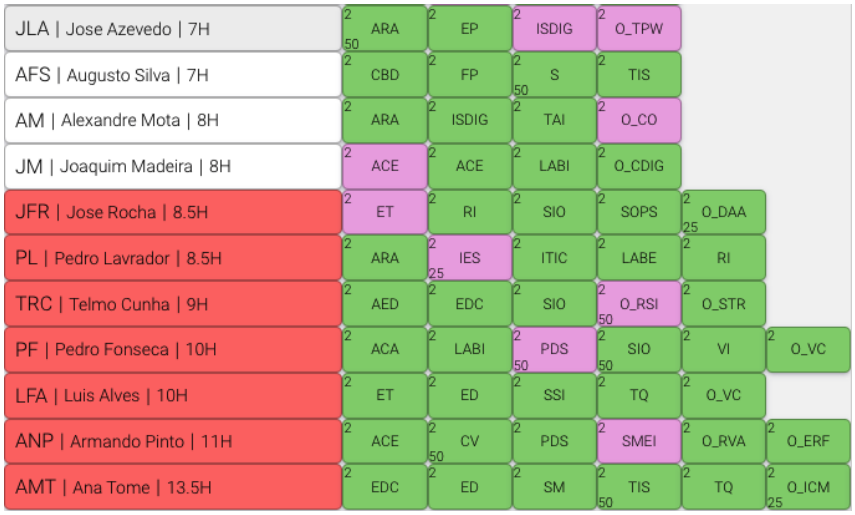

Figure 14: General application: Overloaded teacher's.

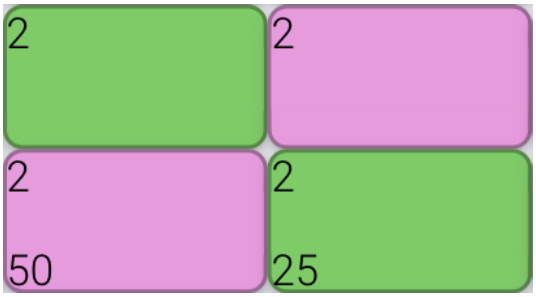

Figure 15: General application: Cells.

These cells are classified by color, with each color referring to a different component. The number in the upper left corner indicates the number of hours in that class and the lower left number indicates the percentage of assigned teacher participation in the class. Also this information can be accessed by the user so that he can perform the DSD in the best way possible, as presented in the figure below (16).

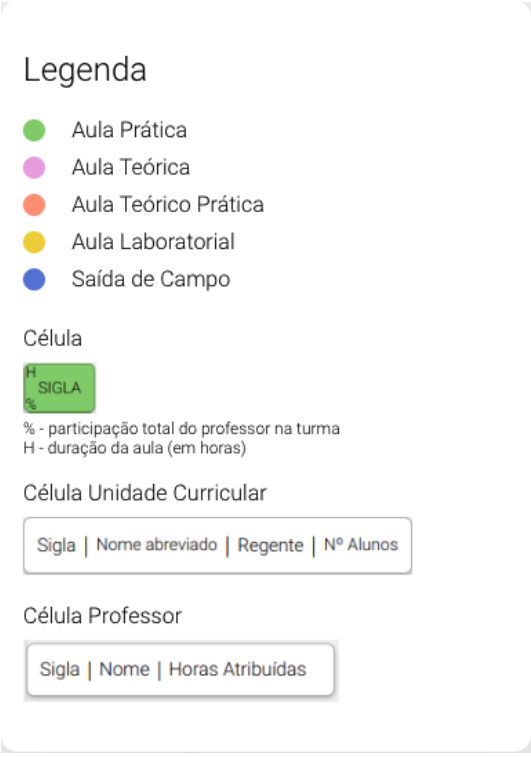

Figure 16: General application: Help panel with some useful information for the user.

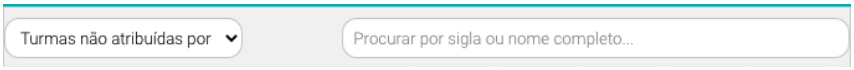

Figure 17: General application: Filter and search bar.

As shown in the figure above (17), cells can be sorted by filters and searched name/acronym. This is good because it allows the user greater freedom in interacting with the information.

Side panels All teacher's and UC's information is shown here.

Figure 18: General application: Side panel containing all UC information.

Figure 19: General application: Side panel containing all teacher information.

Here is where all the information of a UC or a professor can be found. Several essential information are shown to the user because without them the realization of the DSD would be much more complicated. The wishlist of each teacher indicating what each one prefers or does not prefer to teach is an example of this kind of important information.

Also it is possible to only get the teachers that prefere a certain UC by just clicking in the button ”Filtrar” presented in the UC side panel wich is good because we can get immediately the teachers that want to perform that class.

Change acronym's page Change acronym of a certain teacher.

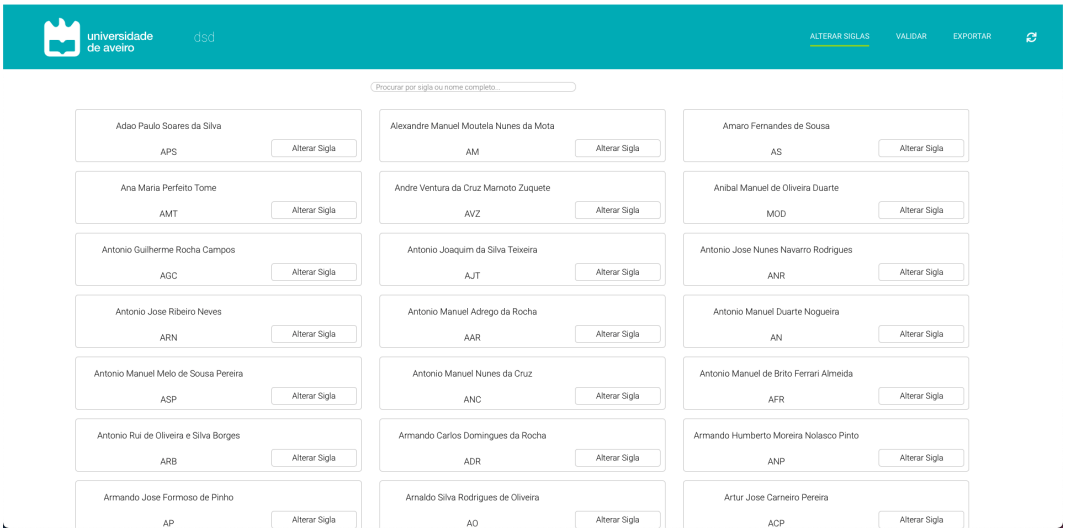

Figure 20: General application: Change acronym page.

Here is where the user will be able to change the acronyms of each teacher if he wants to. He can search for each teacher by their current acronym or by their full name.

This tool is very important because if the user has some problems with the acronyms of some of the teachers, he can simply change them and thus maximize the DSD elaboration process.

Validate tool This tool can verify if there are some problems in the current DSD.

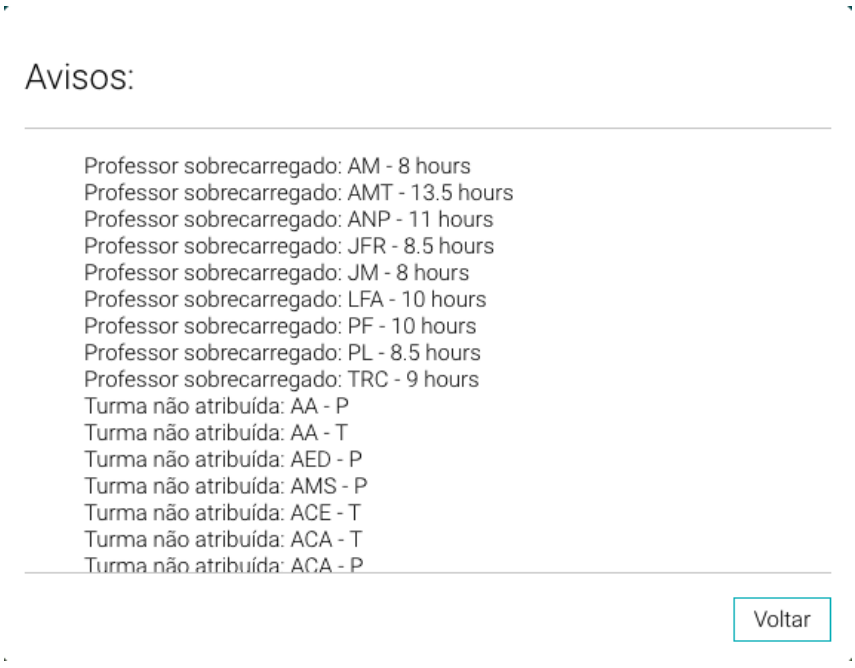

Figure 21: General application: Validate modal box.

This tool allows you to check the current DSD for errors and show them to the user.

Specifically, it checks if there are teachers with a workload above the expected and if there are classes that do not have an assigned teacher.

This is a very important feature as it allows the user to check if there are any problems with their DSD before ending it.

Export tool This tool can export the DSD in different formats.

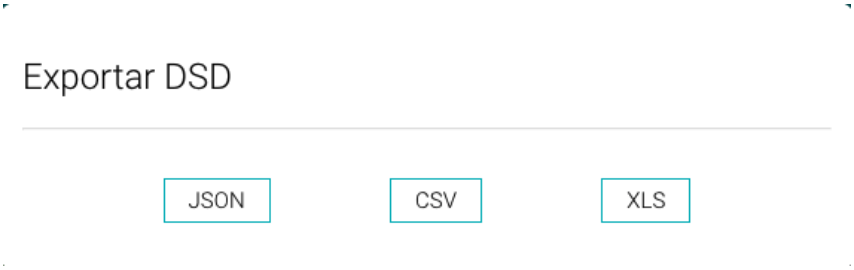

Figure 22: General application: Export modal box.

This tool was created in order to allow the user to have the DSD in different formats. Therefore, the DSD can be exported in 3 formats:

- JSON;
- CSV;
- XLS.

When choosing one format, the DSD is automatically downloaded in that format.

4.2 FastAPI

An API is an interface through which one program or web site talks to another. They are used to share data and services, and they come in many different formats and types.

As stated earlier, we used FastAPI to establish communication between the web application and the database due to familiarity with the Python programming language and it's intuitive use.

GET endpoints:

- /: Root endpoint. Redirects to documentation;
- **/v1/classes**: Returns all classes in database. Allows filtering by: id, year, uc_id, component, hours, prof_id;
- **/v1/departments**: Returns all departments in database. Allows filtering by: id, acronym, name, address, phone;
- **/v1/professors**: Returns all Professors in the database. Allows filtering by: prof_id, nmec, email, phone, acronym, name, rank, situation, department;

- **/v1/dsders**: Returns all DSDers in the database. Allows filtering by: id;
- **/v1/courses**: Returns all courses in the database. Allows filtering by: id, acronym, name, department, director;
- **/v1/ucs**: Returns all UCs in the database. Allows filtering by: id, acronym, name, director;
- **/v1/wishlists**: Returns all wishlists entries in the database. Each entry associates a UC with the Professor's preference. Allows filtering by: id, year, prof_id, class_id;
- **/v1/classes_main_panel_info**: Returns all the information necessary to populate the UCs main panel. Allows sorting by column in ascending and descending order;
- **/v1/professors_main_panel_info**: Returns all the information necessary to populate the Professors main panel. Allows sorting by column in ascending and descending order and filtering by prof_id;
- **/v1/prof_total_hours**: Returns the number of total hours assigned to each Professor;
- **/v1/validate_dsd**: Returns all unassigned classes (where value is null) and overallocated Professors (where the number of total assigned hours is greater than the max_hours parameter);
- **/v1/export_dsd**: Triggers a file download in either JSON, CSV or XLS format of the DSD's current state.

POST/DELETE endpoints:

- **/v1/classes**: Post/Delete classes to/from the Database;
- **/v1/departments**: Post/Delete departments to/from the Database;
- **/v1/professors**: Post/Delete Professors to/from the Database;
- **/v1/dsders**: Post/Delete DSDers to/from the Database;
- **/v1/courses**: Post/Delete courses to/from the Database;
- **/v1/ucs**: Post/Delete UCs to/from the Database;
- **/v1/wishlists**: Post/Delete wishlist entries to/from the Database.

PUT endpoints:

- **/v1/classes**: Assigns/Removes a Professor to a class. Required parameters: class_id, prof_id;
- **/v1/professors**: Updates a Professor's acronym. Required parameters: prof_id, acronym.

4.3 Data persistence

The whole platform is fed by a relational MySQL database, running inside a docker container. Here is the original domain model for the database, which suffered light changes, described below in the brief description of each table and its fields section.

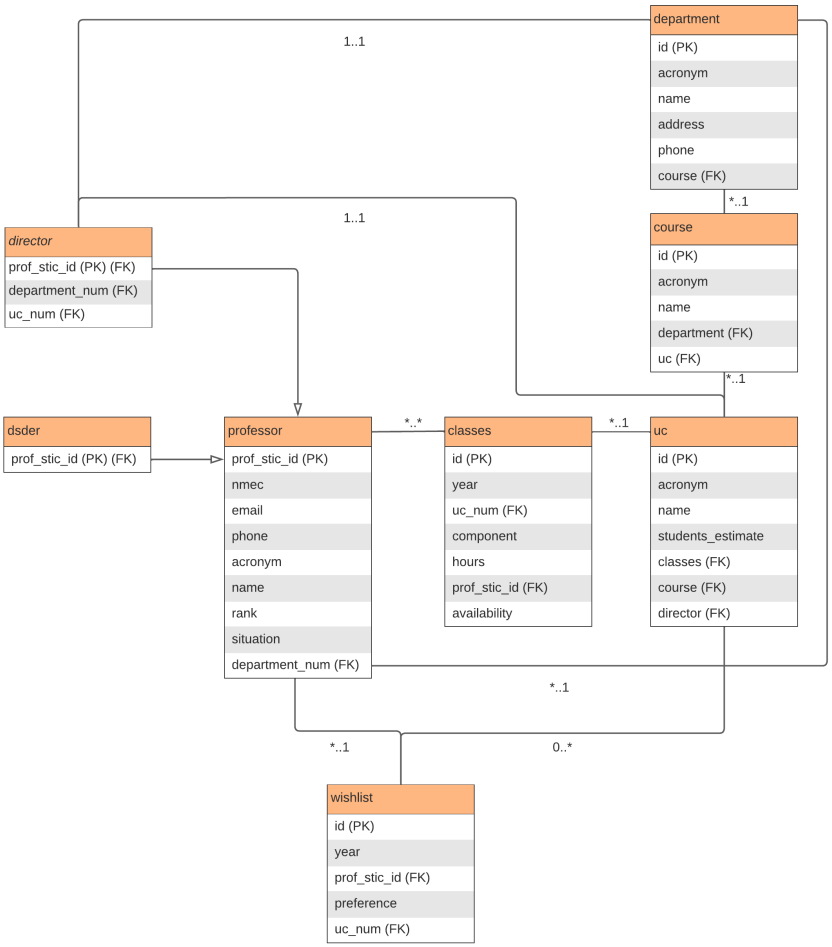

Figure 23: Domain Model.

Brief description of each table and its fields:

- **Departments**:
  - **dept_id**: department's ID;
  - **acronym**: department's acronym;
  - **dept_name**: department's name;
  - **dept_address**: department's address;
  - **phone**: department's phone number;

- **Professors**:
  - **prof_id**: Professor's ID;
  - **nmec**: Professor's mechanical number;
  - **email**: Professor's email;
  - **phone**: Professor's phone number;
  - **acronym**: Professor's acronym;
  - **prof_name**: Professor's full name;
  - **prof_rank**: Professor's rank;
  - **situtation**: Professor's situation (Active, Retired, ...)
- **DSDers**:
  - **dsder**$_id : DSDer'sID$;
- **Courses**:
  - **course_id**: Course's ID;
  - **acronym**: Course's acronym;
  - **course_name**: Course's full name;
  - **department**: Course's department;
  - **director**: Course's director;
- **UCs**:
  - **uc_id**: UC's ID;
  - **acronym**: UC's acronym;
  - **uc_name**: UC's full name;
  - **students_estimate**: UC's students estimate;
  - **director**: UC's director;
- **Classes**:
  - **class_id**: Classes's ID;
  - **year_int**: Classes's year;
  - **uc_num**: Classes's UC code;
  - **component**: Classes's component (Pratical, Theoretical,...);
  - **class_hours**: Classes's duration in hours;
  - **prof_id**: Classes's assigned Professor;
  - **availability_percent**: Professor's participation in the classe;
- **Wishlists**:

- **wishlist_id**: Wishlist's ID;
- **year_int**: Wishlist's year;
- **professor**: Identifies the Professor associated with this entry;
- **preference**: Wishlist's stated preference; (Likes/Dislikes/Neutral);
- **uc_id**: Identifies the UC associated with this entry;

4.4 Docker Containers

Docker was the chosen strategy to guarantee the portability and isolation of our deployed platform.

For reasons of ease of development, debugging and for being the recommended method in the official docker documentation, we decided to create a dedicated container for each service:

- **react container**: Container running the web application;
- **db container**: Container running mysql database;
- **adminer container**: Container running a tool called adminer used for database management;
- **api container**: Container running FastAPI;

Regarding performance losses in this approach, docker is extremely efficient because, unlike traditional VMs, it only creates an isolated filesystem and uses the host OS kernel.

5 Tests and Results

After having a meeting with teachers responsible for making the DSD from multiple departments from our university in order to present, test and get feedback on our project, we came to the conclusion that the some people would like, in addition to writing an acronym on a cell in order to assign a teacher to a class, to work with a 'drag and drop' function, with the objective of having a more versatile and personalized experience.

Apart from that, all the feedback was positive regarding our approach and our final product, with the exception that a few people found it peculiar for us to work with acronyms, since it is a feature that is only natural to our department (DETI).

6 Conclusions and Future Work

6.1 Project Summary

This project started with a period of familiarization with the topic of DSD, understanding what DSD really is, who makes it and what is necessary to do it, since it is not a very well known topic by the university students.

After the topic was well known by the entire team, and after successive meetings with the supervisors, the technologies to be used were decided, as described previously throughout the report.

During the construction phase, we followed an Agile methodology, always maintaining communication between all team members and supervisors. Through the feedback received, some of the initial requirements were changed, which allowed us now to have a project that pleases the customer.

6.2 Main Results

Looking at the expected goals of this project and the main features provided by the working prototype, it is safe to say the main goals were accomplished. The web application allows the user to assign a teacher to a class and also check teacher and UC information.

As planned, due to lack of time and other projects we were not able to implement all the features we wanted, like the teachers wish list forms page.

However, it is possible to use the project and effectively create the DSD, validate it and then export it to different formats.

6.3 Conclusions

Being involved in this project has opened up horizons for us in terms of what goes behind the process of assigning teachers to UCs, as well as leaving us more familiar with technologies never previously used, and further solidifying our knowledge on those we already had some experience with.

This was also the first time we ever had such and important job to do with multiple tasks, so it really showed us how important it is to plan in advance, distribute chores, get to know your workmates strengths and weaknesses, and much more.

With this done, we hope that our platform will bring an end to the outdated system of doing the DSD.

6.4 Future Work

Due to the lack of time, resources, and the importance of other functionalities, we couldn't implement as much as we wanted, so in terms of future work, two things come straight to mind:

- Students per UC estimate: One of the more important functionalities we would like to implement would be a way to estimate how many students were going to take part in a certain UC, in order to better know how many classes and teachers it would take to make the DSD for that UC. Right now, due to the overall great complexity of gathering the information needed to perform such a estimate, we concluded that it wasn't possible to implement it without it affecting negatively the outcome of our project.
- Drag and Drop: When showing the final product we came up with to people who weren't directly involved with the project, one point most of them made was that a

'Drag and Drop' feature to assign teachers to classes would make the whole process faster and more intuitive. Although we had already though of such feature, due to complexity reasons, we decided to go with a different approach in order to not jeopardize the outcome of our work. Given more time, we would be glad to implement it.

7 References

[1] ”Drawio Editor” https://app.diagrams.net/

[2] “Balsamiq Editor” https://balsamiq.com/

[3] “What is MySQL” https://www.talend.com/resources/what-is-mysql/

[4] “MySQL” https://www.devmedia.com.br/introducao-ao-mysql/27799

[5] ”ReactJS” https://reactjs.org/

[6] ”FastAPI” https://en.wikipedia.org/wiki/FastAPI

[7] ”Float” https://www.float.com/

[8] ”SoftExpert” https://www.softexpert.com/

[9] ”Docker” https://www.ibm.com/in-en/cloud/learn/docker

[10] ”Docker Compose” https://docs.docker.com/compose/

[11] ”Stic” https://www.ua.pt/pt/stic/