Faculdade De Engenharia Da Universidade Do Porto

Alumni Around the World

Jénifer Graça Gouveia Constantino

U. PORTO
FEUP **FACULTY OF ENGINEERING**
UNIVERSITY OF PORTO

ASSOCIAÇÃO DE ANTIGOS ALUNOS DE ENGENHARIA INFORMÁTICA DA FEUP

Bolsa dos Colaboradores SASUP

Supervisor: João Carlos Pascoal Faria

June 31, 2024

Contents

List of Tables

List of Figures

1. Abbreviations and Symbols

SASUP   Serviços da Ação Social da Universidade do Porto

EI   Engenharia Informática

2. Introduction

The current project was developed in the context of the program "Bolsa dos Colaboradores" promoted by SASUP, Serviços da Ação Social da Universidade do Porto, from January 1st to June 31st of 2024.

Informatics Engineering in FEUP has existed since 1995, making 2024 the 30th year of its existence. To commemorate such an accomplishment, FEUP is organizing an event where previous students from Informatics Engineering can gather again and share experiences they've had throughout the years.

With this happening, a **need** to understand where students are and what they have been doing in the past years came up. For this reason, the current solution aims to visually show the alumni distribution worldwide, using LinkedIn as a source of information. The solution further presents features to ease the user experience in finding their colleagues by providing filtering and searching inputs. Because it's also important to keep data updated, administrator features were also implemented and will be described in future sections.

The development of this project has included meetings for two-to-two weeks with Professor João Carlos Pascoal Faria and monthly meetings with José Coutinho alongside Professor Rui Neves and Professor Raul Vidal. In these meetings, requirements, and solutions to the posed challenges were discussed. Unit tests were not performed because of the short development time and the current solution's introductory nature. Further, only functional requirements are stated in the current document for the same reasons.

This solution implied a lot of challenges, such as gathering the alumni LinkedIn links or dealing with the non-normalized fields in LinkedIn. This document aims to describe the development process and the obtained results. It first states the goals, needs, and use cases, going through the elicited requirements, solutions' limitations, technologies, and deployment process. It concludes by wrapping up the findings and stating future steps.

3. Goals and Requirements

This chapter thoroughly details the solution, addressing goals, requirements, and use cases. The Use Case diagram highlights only the features that have been implemented. Additionally, a list of elicited functional requirements is included, with notes indicating their implementation status.

3.1. Solution

The solution came to address **2 primary needs**:

1. Enable alumni to keep in contact with each other.
2. Get alumni information: work positions, location, and more.

The solution extracts information from Sigarra and alumni's LinkedIn profiles to address these needs, enabling cross-reference information and obtaining more accurate data. Sigarra extracts information on the alumni's courses and years of conclusions, while LinkedIn extracts information on the alumni's current location, job, and much more. More detail on the data the Proxycurl API provides is described in the section ***"Technologies"***.

The solution provides a 3D world map that illustrates the alumni distribution worldwide across countries or cities. To enable a better user experience, it's possible to search alumni by their name and to filter alumni based on their course and/or conclusion year. Administrator functionalities were also implemented since the information collected using Proxycurl API needs to be updated. Both these functionalities can be seen in the Use Case Diagram in ***Annex 9.3***.

Lastly, throughout the development, User Stories were elicited. ***Annex 9.2*** illustrates them and indicates if they were or were not achieved. The "Monthly Meeting" value corresponds to meetings where Professor João Carlos Pascoal Faria, Professor Raul Moreira Vidar, Rui Neves, and José Carlos Coutinho were present.

3.2. Requirements

Because of the limited time to develop the solution, only the functional requirements are listed in ***Annex 9.1***. These requirements were elicited from meetings held throughout the semester and associated with a requirement number, description, source from where they were elicited, priority according to the MoSCoW method, and an indication of whether they were implemented.

The source attribute assumes values such as "Monthly Meeting" and "2-week meeting". The first possible value corresponds to meetings where Professor João Carlos Pascoal Faria, Professor Raul Moreira Vidar, Rui Neves, and José Carlos Coutinho were present. The second possible value corresponds to meetings where only Professor João Carlos Pascoal Faria was present.

Lastly, as was previously mentioned, requirements were prioritized based on the MoSCow method where:

- **M: Must Have.** For requirements that are needed for the project to be completed.
- **S: Should Have.** For requirements that are important but not critical for the system to work.

- **C: Could Have.** For requirements corresponding to desirable features that don't affect the projects' success.
- **W: Would Have.** For requirements with the lowest priority or not necessary for the timeline.

3.3. Limitations

The solution includes information only from alumni whose **LinkedIn** profiles could be **accessed**. Not all fields from these profiles are **normalized** or filled in, as LinkedIn does not require them to be mandatory. For example, while the country field is mandatory, the city field is not, resulting in a **discrepancy** between the **number of alumni** shown in the country and city views. Additionally, the **city** field lacks normalization, leading to some unrecognizable values by the **GeoNames API**. Similar challenges were encountered with the course field. Details on how these issues were addressed can be found in the "*Implementation*" section.

In some cases, the Proxycurl API could not retrieve the alumni's **profile pictures**, so those alumni are displayed without images in the solution. Further, this API only returns information from public LinkedIn profiles. Lastly, the Alumni EI group on LinkedIn includes professors as participants, which is why they also appear in the current solution, even though they might not be alumnis.

4. Technologies

This section gives an in-depth description and justification of the chosen technologies. A technology diagram is presented in *Annex 9.4* to illustrate better the technologies used.

For the frontend, the technologies were React, JavaScript, and node.js; for the backend, the chosen ones were SpringBoot and Java.

To get the information from the users' LinkedIn profile, the Proxycurl API was used, and to get the coordinates of a given place, be it a city or country, GeoNames API was used. Both these APIs posed some challenges, which are described in the next subsections.

4.1.1.   Proxycurl API

The **LinkedIn official API** was initially considered to get the alumni's LinkedIn profile information. However, it was possible to conclude that, besides being paid, the usage of this API is only allowed by LinkedIn Partners, which requires a long and difficult process to get approved. More information on this API can be analyzed in **[1] [2] [3] [7]**.

After some research, **Proxycurl API** was found. This API is like the official LinkedIn API but only works with Public LinkedIn Profiles and differentiates in latency and data completeness. This option scrapes data from the alumni's LinkedIn account.

Proxycurl provides several APIs that enable one to get information from LinkedIn. These APIs include the School API, Company API, People API, Customer API, Jobs API, Contact API, Search API, and Meta API. Only the **People API** was used for the current solution being developed. More information can be found in **[4] [5]**. These references also illustrate how to make a request and its results. Further, this Proxycurl API also has **costs** represented in reference **[6]**.

Getting information from LinkedIn profiles posed several challenges **concerning data management**. Several fields were not mandatory and not normalized. Such fields include city, job position, university name, etc. Further, profile pictures were only available for 30 minutes, and for so, these had to be stored in the current solution, more information in the "*Implementation*" section.

4.1.2.   GeoNames API

This API retrieves user **coordinates** based on their country or city. The country information is included to improve city coordinates' accuracy.

While coordinates were successfully obtained for all countries, the API could not locate some cities. The solution to this issue is discussed in the "*Implementation*" section. Additionally, it's important to note that the free version of this API has a limit of **1000 requests per hour**. More details about this API can be found in references **[8]** and **[9]**.

5. Implementation

This section aims to describe how the solution was developed.

As previously mentioned, several **challenges** were encountered, primarily related to **data normalization**. Different alumni provided various names for the courses they completed at FEUP and reported different conclusion years from the actual ones. This discrepancy complicated the clustering process for presentation and filtering alumnis distributed worldwide. To address this issue, **LinkedIn** data was cross-referenced with **Sigarra's** data to obtain more **accurate information**.

To understand the architecture of the provided solution, a **Component Diagram** is presented in **Annex 9.5**. This diagram is divided into three levels of abstraction. The third level offers a detailed view of the solution structure, which includes the following folders:

- Controller. Should handle HTTP requests and return responses.
- Model. Defines the data structure and business logic.
- Repository. Handles data access and persistence.
- Service. Contains business logic and service layer code.

Regarding backend development, the **Strategy Pattern** design was implemented. This pattern allows the definition of a family of algorithms, encapsulating each one and making them interchangeable, enabling the algorithm to vary independently from the clients that use it. This implementation is illustrated in the class diagram presented in **Annex 9.22**, where only the relevant methods were listed.

The **application.properties** file stores configurations such as database access credentials, API keys, symmetric keys, and paths for storing files and images. An illustration of this file is found in **Annex 9.17**.

For the database structure, an **Entity-Relationship (ER)** diagram is provided in **Annex 9.6**. This diagram shows an *"Alumni"* table containing information obtained from the Proxycurl API. Other tables are populated based on this table. The city field might have **empty values** since it is not mandatory on LinkedIn. Although it is a bad practice, finding a way to obtain this value is important since every alumnus should have a city listed. The issue lies not with the ER diagram's structure but with alumni not filling out this field on LinkedIn. The "Course" table may eventually include attributes such as description or name, which would need to be filled using other sources.

Further, when the Proxycurl API is called, besides storing its result on the Alumni table, alumni **profile pictures** need to be downloaded into the system so that it's possible to get them whenever needed. In the Proxycurl API response, a link to the alumni profile's picture is provided, and available for only 30 minutes. For this reason, these images are stored in the path "**public/Images**" of the frontend. This resulted in a challenge while developing since it provoked a reload of the page when the admin inserted a new image into this folder. This is due to React's development server watching for changes in the project's files to enable hot reloading, helping developers refresh the app to reflect the latest changes. Nonetheless, this behavior doesn't happen during deployment since it's not a development environment.

Lastly, to **enhance security**, authentication is performed in such a way that if the administrator tries to refresh the page in the settings page, he gets automatically **redirected to the login page**, requiring him to insert the password again.

The next subsections explain the developed endpoints and features, as well as an explanation of how to deploy the current solution to the virtual machine being used.

5.1. Endpoints

This section aims to document the used endpoints. The next subsection will further explain the features resulting from these endpoints using flowchart diagrams.

5.1.1.   AdminController.java

Table 1. Admin Controller. Verify Pass.

Requirement	Description
Method/Endpoint	**POST**/admin/verifyPass
Functionality	Verifies if the admin password is correct.

Body Example	{
	    "password": "admin99"
	}

Parameters	The request doesn't have parameters.
Response Codes	200: returns the password is valid 500: Error while password verification
Response Example	{"validPassword":true}

Table 2. Admin Controller. Change Admin Pass.

Requirement	Description
Method/Endpoint	**POST**/admin/changeAdminPass
Functionality	Updates the admin password.

Body Example	{
	    "newPass": "admin88",
	    "oldPass": "admin99"
	}

Parameters	The request doesn't have parameters.
Response Codes	200: returns success on updating admin's password 500: error while password update
Response Example	{"success":true}

Table 3. Updates Proxycurl API Key.

Requirement	Description
Method/Endpoint	**POST**/admin/updateApiKey
Functionality	Updates the API key in use.
Body Example	{

	"apiKey": "{YOUR_Proxycurl_API_KEY}" }
Parameters	The request doesn't have parameters.
Response Codes	200: returns success on updating API Key 500: error while updating API Key
Response Example	{"success":true}

Table 4. Admin Controller. Replace alumni data.

Requirement	Description
Method/Endpoint	**POST**/admin/replaceAlumnus
Functionality	Deletes the Alumni table and calls the Proxcurl API for every alumnus in the provided Excel file. These are then added to the DB.

Body Example	{
	    "file": "Binary file data (Blob)"
	}

Parameters	The request doesn't have parameters.
Response Codes	200: alumni replaced successfully 500: error while replacing the alumnus data.
Response Example	None

Table 5. Admin Controller. Add alumni data.

Requirement	Description
Method/Endpoint	**POST**/admin/addAlumnus
Functionality	Calls the Proxycurl API only over the alumnus of the provided Excel file that are not already on the DB. These are then added to the DB.

Body Example	{
	    "file": "Binary file data (Blob)"
	}

Parameters	The request doesn't have parameters.
Response Codes	200: Alumni added successfully 500: Error while replacing the alumnus data.
Response Example	None

Table 6. Admin Controller. Update alumni data.

Requirement	Description
Method/Endpoint	**POST**/admin/updateAlumnus
Functionality	Calls the Proxycurl API over every Alumni in the provided Excel file. If the alumni already exists on the DB, his register is updated. If the alumni don't exist, the register is added.

Body Example	{
	     "file": "Binary file data (Blob)"
	 }

Parameters	The request doesn't have parameters.
Response Codes	200: Alumni updated successfully 500: Error while updating the alumnus data.
Response Example	None

Table 7. Admin Controller. Backup alumni data to Excel.

Requirement	Description
Method/Endpoint	**POST**/admin/readAPIResultToExcel
Functionality	Put the result of the Proxycurl API in an Excel file.
Body Example	The request doesn't need a body.
Parameters	The request doesn't have parameters.
Response Codes	200: binary data representing the Excel file 400: Bad request
Response Example	binary data representing the Excel file

5.1.2. AlumniController.java

Table 8. AlumniController. Returns the created GeoJson.

Requirement	Description
Method/Endpoint	**GET**/setupLocation/getGeoJson
Functionality	Returns the GeoJSON file created based on the user's filters: course, year, and geoJsonType (country or city).
Body Example	The request doesn't need a body.
Parameters	courseFilter – can assume values of: LEIC, L.EIC, M.EIC, MIEIC, MEI, yearConclusionFilter – array of length 2. It corresponds to the conclusion year of an alumni in a given course. Ex: [2017, 2018] geoJsonType – can assume one of two values: 'countries' or 'cities'
Response Codes	200: binary data representing the GeoJSON file 500: null
Response Example	binary data representing the GeoJSON file

5.2. Features

This section aims to illustrate how user and admin features were implemented. These are followed with **flowchart diagrams** to define better what is being explained.

5.2.1. User Features

These features can be accessed **without a login**, including visualizing alumni worldwide and filtering and searching for alumni.

A **GeoJson file** is generated to display alumni worldwide based on the user's filters: course and/or conclusion year, and the GeoJson type: country or city. An example of this file is provided in **Annex 9.15**. The content of this file is performed by receiving an Excel file containing Sigarra information, such as the year of the conclusion and course, and containing the alumni LinkedIn links. If the alumni are not associated with a LinkedIn link, or if the LinkedIn link is not associated with an alumni, the solution won't consider this alumni or LinkedIn link.

To avoid **concurrency** issues and increase efficiency, GeoJson files are created and stored in a "LocationGeoJson" folder, which acts as a caching mechanism. This folder is located outside the main folder to prevent constant reloading of the program. Each file is **uniquely named** based on the filters applied. If a file with the corresponding filters already exists, it is automatically returned. Otherwise, a new file is created, stored in the folder, and returned to the user. The flowchart for this feature is presented in **Annex 9.7**. This flowchart only presents a high-level view of this mechanism.

This GeoJson file is returned by the endpoint described in the subsection "AlumniController.java." It is important to notice the content of this file has some limitations:

- The GeoJson file only has information of alumni to which it was possible to get the LinkedIn link.
- Country is a mandatory field on LinkedIn, so all alumni show up on this view. On the other side, the city is not a mandatory field. For this reason, there are fewer alumni showing up in the city view.
- Some cities were not recognized by the GeoNames API, even when using the country to give more precision. To solve this challenge, the non-recognizable cities had to be hardcoded into recognizable ones. This can be seen in **Annex 9.16**.

5.2.2. Admin Features

These features can only be accessed by an authorized user, the **administrator**. As such, login is needed, and once performed, the user can replace, update, add alumni data, get Proxycurl API result in a readable Excel file, change the administrator password, and update the key used to perform requests to the Proxycurl API.

The **login** functionality is performed by validating the user's inserted password. If the hash of this password matches the one stored in the database, the user is considered valid and can log in. The flowchart for this mechanism can be found in **Annex 9.8**.

The **change password** functionality requires the user to insert the current password. If the hash matches the one in the database, the system hashes the new password and updates it on the database. The flowchart for this mechanism can be found in **Annex 9.9**.

The **Proxycurl API Key update** functionality encrypts the newly inserted API key using **symmetric encryption**. The symmetric key in this process is stored in the "*application.properties*" file. After encryption, the API Key in the database is updated with the new encrypted key. The flowchart for this mechanism is illustrated in **Annex 9.10**.

The administrator can **download the result of the Proxycurl API to an Excel** file to enable easier interpretation of the alumni data. The flowchart for this mechanism can be found in **Annex 9.11**.

Lastly, there are three buttons the admin can iterate to choose how to **manipulate the Alumni table**:

- **Replace Alumni data**. Deletes the content of all tables, including the Alumni table, and calls the Proxycurl API for each alumni in the provided Excel file. It further deletes all GeoJson files previously created, as explained in the previous section, *"User Features"*. All tables are then repopulated based on the new alumni data. The flowchart for this mechanism can be found in **Annex 9.12**.
- **Add Alumni data**. Deletes the content of all tables, except the Alumni table. Reads the provided Excel file, and if the alumni are not already in the DB, the Proxycurl is called, and the information is **added** to the Alumni table. This functionality **doesn't update** the alumni data already in the database. The flowchart for this mechanism can be found in **Annex 9.13**.
- **Update Alumni data:** Deletes the content of all tables**, except** the Alumni table. Reads the provided Excel file and calls the Proxycurl API for each alumni. If the alumni already exist in the database, his data gets **updated,** otherwise his data gets **added** to the alumni table. The flowchart for this mechanism can be found in **Annex 9.14**.

As mentioned at the beginning of this section, the Alumni table is the table that stores the result obtained from the call to Proxycurl API. When calling this API, a symmetric key is needed to decrypt the API Key stored on the database. This symmetric key is stored in the "*application.properties*" file.

These buttons require the administrator to insert an Excel File. This Excel file is validated before any functionality is executed, ensuring the correct functionality of the solution and avoiding inconsistent values on the database. If the file is incorrect, an **error file is downloaded** with the information on the alterations the alumni need to perform so the system can accept the file and execute the needed functionality.

The structure of this file can be seen in **Annex 9.23**. It's possible to see that the file needs to have a number, name, LinkedIn link, and Course column. The solution validates the Excel headers and the content of each column:

- The "Número" column should be a numeric value of 9 digits.
- The "Nome" should be a type string or formula.
- The "Link Após Inquérito e Pag SIGARRA" should start with: "https://www.linkedin.com/in/" and end with "\".
- The "cursos" should assume a structure of {courseName}{conclusionYear}.

It's important to follow these structures. Otherwise, the program won't work as expected. The column names can be changed by accessing the "application.properties" file.

5.3. Deployment

The deployment of the current solution was performed on a virtual machine with the following **characteristics**:

1. Domain: eic30anos.fe.up.pt
2. IP: 10.227.242.123

3. Hostname: eic30anos.fe.up.pt
4. Login: eic30anos
5. Pass: zaX9Reinaip?i7Eew7so

A set of keys was provided for the website to be accessed outside of FEUP. The project is currently in the "*alumniProject*" folder in the Linux Virtual Machine.

Before deploying the solution, it should be ensured that:

1. **Frontend:** package.json should have the parameter "homepage" pointing to "/alumnieiworld". The request should be performed as illustrated in **Annex 9.18**, and the routing should be as illustrated in **Annex 9.18**.
   App.js PrivateRoute.js AdminSettings.js AdminLogin.js
2. **Backend:** ensure the locationGeoJson folder is in path: src/locationGeoJson, not inside the main folder. Ensure the application.properties file has the configuration presented in **Annex 9.17**. Notice the location of the folder "locationGeoJson" is not the same when running the solution in the VM and in the local computer, on the VM a path such as: "/home/eic30anos/alumniProject/backend/src/locationGeoJson" should be used. Lastly, adding the current DB username and password to the application.properties is important.

To **transfer** MySQL DB to the VM:

1. Export local DB from PhpMyAdmin.
2. Transfer the SQL file to the VM. This file is now inside 'eic30anos' with the name 'alumnifeup.sql'.
3. Create a new DB on the VM.
   SSH into your VM
   Login into MySQL: sudo mysql -u root -p with the password: new_password
   Create a new database: CREATE DATABASE alumnifeup;
   Create a new user and granted privileges:
   > Still in the MySQL prompt create a new user: CREATE USER 'alumniroot'@'localhost' IDENTIFIED BY 'new_password';
   >
   > Grant all privileges on the new database to the new user: CREATE ALL PRIVILEGES ON you_dabase_name.* TO 'your_username'@'localhost';
   >
   > Flush privileges to apply changes: FLUSH PRIVILEGES;
4. Exit MySQL by executing: "EXIT"
5. Impot the SQL file into the new database, my executing this command: sudo mysql -u alumniroot -p alumnifeup < ./alumnifeup.sql. It will ask for a password, you should put the DB password, in this case: new_password.
6. Verify the import.

To be able to **deploy the website**:

1. **Log** into the virtual machine:
   ssh eic30anos@10.227.242.123 < zaX9Reinaip?i7Eew7so
2. **FileZilla** was used to transfer local projects inside the VM. The used parameter of FileZilla were:

Host: eic30anos.fe.up.pt
Username: eic30anos
Password: zaX9Reinaip?i7Eew7so
Port: 22 (the default one)

3. **Deploy code** on the VM:
   cd alumniProject < cd frontend < npm install < npm run build
4. **Compile** backend code:
   cd alumniProject < cd backend < chmod +x mvnw < ./mvnw clean package < mvn clean install -DskipTests
5. Set **Nginx** configuration. The used configuration can be seen in **Annex 9.19**. To edit this file:
   sudo nano /etc/nginx/sites-available/default < sudo nginx -s reload.
6. **Create a system service** for the backend by executing:
   'sudo nano /etc/systemd/system/backend.service' - the code for the created service can be seen in **Annex 9.20**.
   Enable and start the service by executing:
   'sudo systemctl enable backend.service' < 'sudo systemctl start backend.service'.
   Update the services by executing:
   'sudo systemctl status backend.service'.
   If this service needs to be manually restarted:
   'sudo systemctl restart backend.service'

If it is intended to **run the solution on the VM**, then:

1. cd alumniProject < cd frontend < npm install < npm start => it will run on port 3000
2. cd alumniProject < cd backend < chmod +x mvnw < ./mvnw clean package < mvn clean install -DskipTests < java -jar target/alumniFEUP-0.0.1-SNAPSHOT.jar

If any of those ports are in use, execute the following commands to stop the process: "sudo lsof -i :<portNumber> < sudo kill -9 <PID>"

6. Results and Analysis

This section aims to explain and illustrate the result of the current developed solution.

Accessing the main page with the alumni distribution worldwide is possible by accessing the URL: "**eic30anos.fe.up.pt/alumnieiworld**". It can also be accessed using a year filter such as "**eic30anos.fe.up.pt/alumnieiworld/2016**". This interface allows users to search for alumni or filter alumni by course and/or conclusion year by choosing a country or city view. These interfaces can be seen in **Annex 9.21**.

On the other side, to access the administrator functionality, the URL "**eic30anos.fe.up.pt/alumnieiworld/admin**" is provided. In this interface, it's possible to manipulate the Alumni table in the database, change the admin password, and update the Proxycurl API Key. The interfaces can be seen in **Annex 9.21**.

7. Conclusion and Future Work

This document describes the solution to be used on the event that commemorates EI 30th anniversary.

Throughout the development, **several challenges** were posed. The first challenge was to find an API that enabled the extraction of the alumni's information from their LinkedIn profile; later, the challenge to get all alumni's LinkedIn links proceeded, and a last challenge related to data normalization had to be addressed.

Concerning the data normalization problem, this was mitigated with the "course" and "year of conclusion" fields by crossing information with Sigarra. Nonetheless, this problem persists in other fields, such as job positions and city names.

Further, to try to solve the problem related to the absence of the alumni LinkedIn links, several e-mails have been sent requesting this information. Nonetheless, not all were possible to obtain.

Considering these limitations, it was possible to develop a solution that **enables** users to see alumni distribution worldwide, access their LinkedIn links, search for them by name, and filter by course and conclusion year. It was also possible to develop administrative features that allow alumni data to be updated in the database.

A **test-driven approach** was **not** followed while developing due to the limited time and the introductory nature of the current solution. For the same reasons, this current document could also be further improved by documenting non-functional requirements, presenting a goal tree diagram, and providing backend and frontend class diagrams. These are all considered future work.

Overall, it's possible to conclude the needed objective was achieved, all mandatory requirements were implemented, and the previously mentioned challenges were tackled as much as possible.

7.1. Future Work

As previously mentioned, the **document** could be further improved by providing a **goal tree diagram**, documenting **non-functional requirements**, and providing a backend and frontend **class diagram**.

Furthermore, there are still implementation improvements. Concerning the **Entity-Relationship diagram**, cities could be related to countries instead of being related to the user, as represented in **Annex 9.6**. A **timeout mechanism** could also be implemented on the administrator's side to ensure greater security.

User interaction would be improved if a loading mechanism was added while the alumni table is being populated by one of the three buttons presented in the administrator interface: "replace," "add," and "update". This would provide **feedback** to the user. Nonetheless, meanwhile a popup was added informing the user the process might take some time, **Annex 9.20**. The AlumniEI and FEUP logos are being shown in the development environment, but in production, the images are not being displayed, being a future refactor. Lastly, when hoovering the clusters in the interface of the

worldwide alumni distribution, the conclusion year should show as 2009/2010 and not only 2010; this way, it's equal to the format indicated in the conclusion year filter, enhancing **consistency**.

Non-implemented requirements listed in **Annex 9.1** could also be implemented. It would be good to implement requirements Rq22 and Rq23 since they allow the admin to populate the solution without calling the Proxycurl API and spending credits. This would be good in case of any malfunctioning when executing the "replace," "add," and "update" alumni buttons.

Lastly, **unit tests** could also be added to ensure the system works as expected, allowing better system scalability and easier refactoring.

8. References

[1] https://learn.microsoft.com/en-us/linkedin/shared/references/v2/profile/basic-profile

[2] https://learn.microsoft.com/en-us/linkedin/shared/references/v2/profile

[3] https://learn.microsoft.com/en-us/linkedin/shared/integrations/people/profile-api

[4] https://nubela.co/proxycurl/docs?ref=nubela.co#people-api

[5] https://nubela.co/proxycurl/docs?ref=nubela.co#people-api

[6] Pricing: https://nubela.co/proxycurl/pricing

[7] https://nubela.co/blog/ultimate-guide-to-linkedin-api_people-profile-api_with-python-examples/

[8] https://www.geonames.org/export/JSON-webservices.html

[9] https://www.geonames.org/export/web-services.html

9. Annex

9.1. Requirements

9.1.1. Functional Requirements

Table 9. Functional Requirements.

Requirement	Description	Source	Priority	Implemented
Rq1	The **solution** should include alumni's LinkedIn and Sigarra as a source of **information**.	Monthly Meeting	M	Yes
Rq2	The **solution** should consider LEIC, L.EIC, MEI, M.EIC, and MIEIC **courses**.	Sigarra	M	Yes
Rq3	The **system** should only be **accessible** by members of AlumniEI FEUP on LinkedIn	2 Week Meeting	C	No
Rq4	The **user** should be able to **see** the alumni distribution through a **world** map.	Monthly Meeting	M	Yes
Rq5	The **user** should be able to see the **distribution** of alumni across countries.	Monthly Meeting	M	Yes
Rq6	The **user** should be able to see the **distribution** of alumni across cities.	Monthly Meeting	M	Yes
Rq7	The **user** should be able to see the **distribution** of alumni based on the company.	2 Week Meeting	S	No
Rq8	By giving an alumni name, the **user** should be able to **search** for the alumni.	Monthly Meeting	S	Yes
Rq9	The **user** should be able to **filter** alumni by course.	Monthly Meeting	S	Yes
Rq10	The **user** should be able to **filter** alumni by the conclusion year.	Monthly Meeting	S	Yes
Rq11	The **user** should be able to **filter** alumni by company.	2 Week Meeting	S	No
Rq12	The **user** should be able to **filter** alumni based on their professional position.	2 Week Meeting	S	No
Rq13	The **user** should see the total number of alumni being **displayed**.	Monthly Meeting	S	Yes
Rq14	The **user** should be able to **access** an alumni LinkedIn page.	Monthly Meeting	S	Yes

Rq15	The **user** should be able to provide **feedback**.	Monthly Meeting	M	Yes
Rq16	The **user** should be able to **join** the AlumniEI LinkedIn group.	Monthly Meeting	M	Yes
Rq17	The **user** should be able to see **statistics** on alumni's professional position	Monthly Meeting	C	No
Rq18	The **admin** should be able to **access** the admin panel using a **link** different from the one users use to view the alumni distribution worldwide.	2 Week Meeting	M	Yes
Rq19	The **admin** should be able to **log in**.	2 Week Meeting	M	Yes
Rq20	The **admin** should be able to **log out**.	2 Week Meeting	M	Yes
Rq21	The **admin** should be able to get the **results of the Proxycurl API in an Excel file**.	2 Week Meeting	S	Yes
Rq22	The **admin** should be able to get a **backup file** with the alumni LinkedIn link, the respective response of the Proxycurl API, courses, and respective conclusion years.	2 Week Meeting	C	No
Rq23	The **admin** should be able to **populate** the solution using the **backup file**.	2 Week Meeting	C	No
Rq24	The **admin** should be able to replace alumni data.	2 Week Meeting	M	Yes
Rq25	The **admin** should be able to add alumni data.	2 Week Meeting	M	Yes
Rq26	The **admin** should be able to update the alumni data.	2 Week Meeting	S	Yes
Rq27	The **admin** should be able to change his password.	2 Week Meeting	M	Yes
Rq28	The **admin** should be able to change the Proxycurl API Key.	2 Week Meeting	M	Yes

9.2. User Stories

Table 10. User Stories.

User Story	Description	Source	Achieved
US1	As an **alumni student**, I want to see where my classmates are so that I can keep in touch.	Monthly Meeting	Yes
Rq2	As an **alumni student**, I want to know which alumni students are in a specific location so I can interact with them.	Monthly Meeting	Yes
Rq3	As an **alumni student**, I want to know which alumni students work at a certain company to help me or advise me on finding a job.	Monthly Meeting	No
Rq4	As an **alumni student**, I want to know which other alumni students work in a specific field/industry so that I can approach them and find out more about that field/industry.	Monthly Meeting	No
Rq5	As an **alumni student**, I want an easy way to connect with other alumni using the tool to avoid going to LinkedIn to do so.	Monthly Meeting	Yes
Rq6	As an **alumni student**, I want an easy way to join AlumniEI's LinkedIn group to avoid going to LinkedIn.	Monthly Meeting	Yes
	As an **alumni student**, I want to know which companies have job positions available and which alumni are working in the company so that I can get in contact and understand if I'm suitable for the position.	Monthly Meeting	No
Rq7	As a **member of AlumniEI**, I want to know which other alumni work in a specific field/industry so that I can invite them to related events.	Monthly Meeting	No
Rq8	As a **former student group member**, I want to know which alumni went through that student group, and I want to know statistics about them.	Monthly Meeting	No
Rq9	As **course director or marketing manager**, I want to see who the biggest employers are.	Monthly Meeting	No
Rq10	As the **course director or marketing manager**, I want to see the distribution by professional functions/technologies...	Monthly Meeting	No
Rq11	As a **course director or marketing manager**, I want to see the geographical distribution around the world.	Monthly Meeting	Yes
Rq12	As an **administrator**, I want to be able to replace the alumni data on the system with new, updated ones so that I can have a solution with updated information.	2 Week Meeting	Yes
Rq13	As an **administrator**, I want to be able to keep the alumni information the system already has and add new alumni's information to it so that I can add new alumni without changing the information of the existing ones.	2 Week Meeting	Yes

Rq14	As an **administrator**, I want to be able to update the alumni data on the system and add new alumni information to it so that I can have a solution with updated information.	2 Week Meeting	Yes
Rq15	As an **administrator**, I want to be able to backup alumni's data so that I don't have to call the Proxycurl API again in case anything goes wrong.	2 Week Meeting	Yes
Rq16	As an **administrator**, I want to be able to change my administrator password so that I can have more security over my account.	2 Week Meeting	Yes
Rq17	As an **administrator** I want to be able to change my Proxycurl API key so that I can keep on calling the Proxycurl API.	2 Week Meeting	Yes

9.3. Use Case Diagram
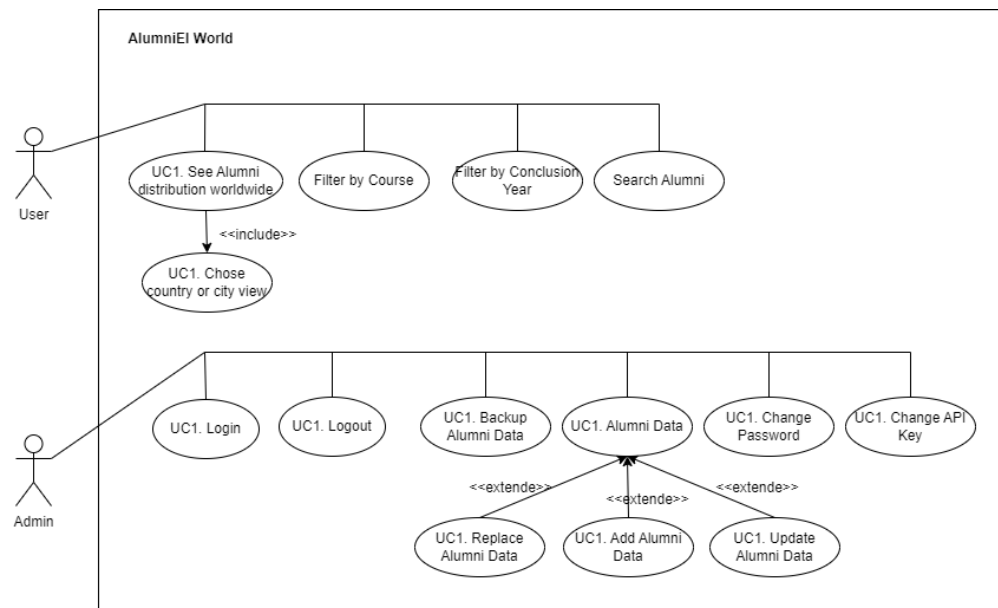
Figure 1. Use Case Diagram.

9.4. Technology Diagram
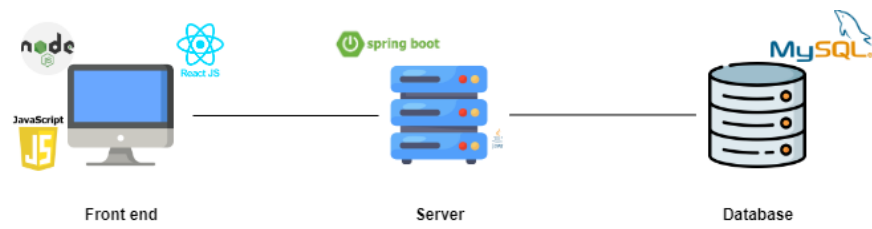
Figure 2. Technology Diagram.

9.5. Component Diagram
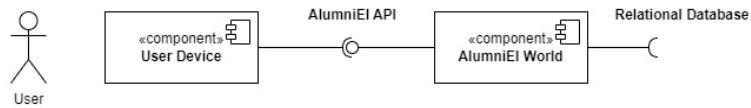
Figure 3. Component Diagram Level 1.
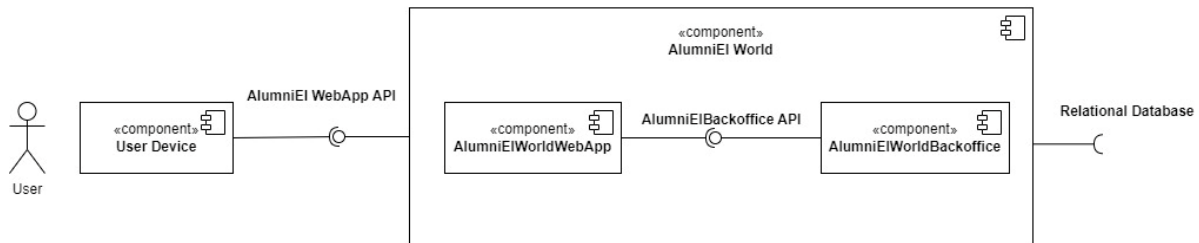
Figure 4. Component Diagram Level 2.
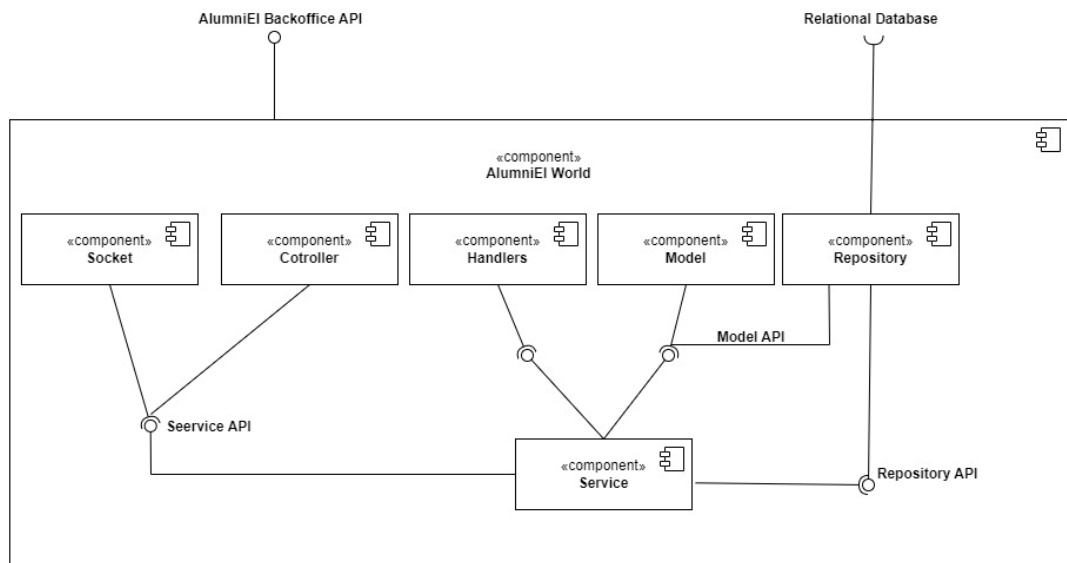
Figure 5. Component Diagram Level 3.

9.6. Entity-Relationship Diagram

alumni

id
linkedin_info
linkedin_link

This table stores the information gotten from the API. It is then used to populate the tables below

alumni_eic

id
alumni_name
linkedin_link
city_id
country_id

alumni_eic_has_course

id
year_of_conclusion
alumni_eic_id
course_id

course

id
abbreviation

Admin

id
username
password_hash
created_at

city

id
city
city_coordinates
n_alumni_in_city

country

id
country
country_coordinates
n_alumni_in_country

Figure 6. Entity-Relationship Diagram.

9.7. Flowchart: Get GeoJson File
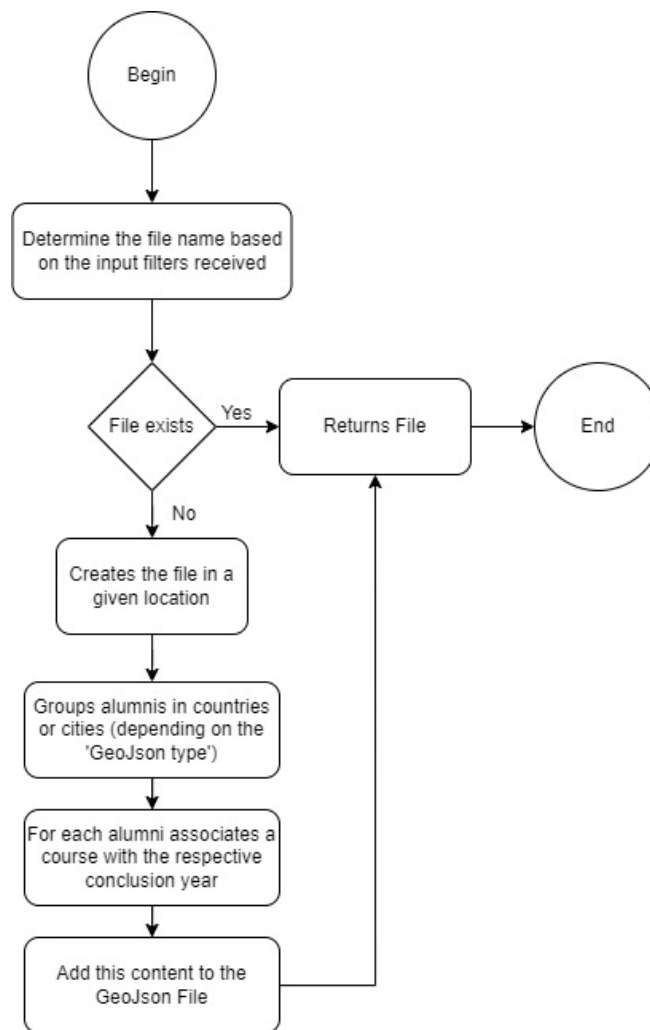
Figure 7. FlowChart: Get GeoJson File.

9.8. Flowchart: Login
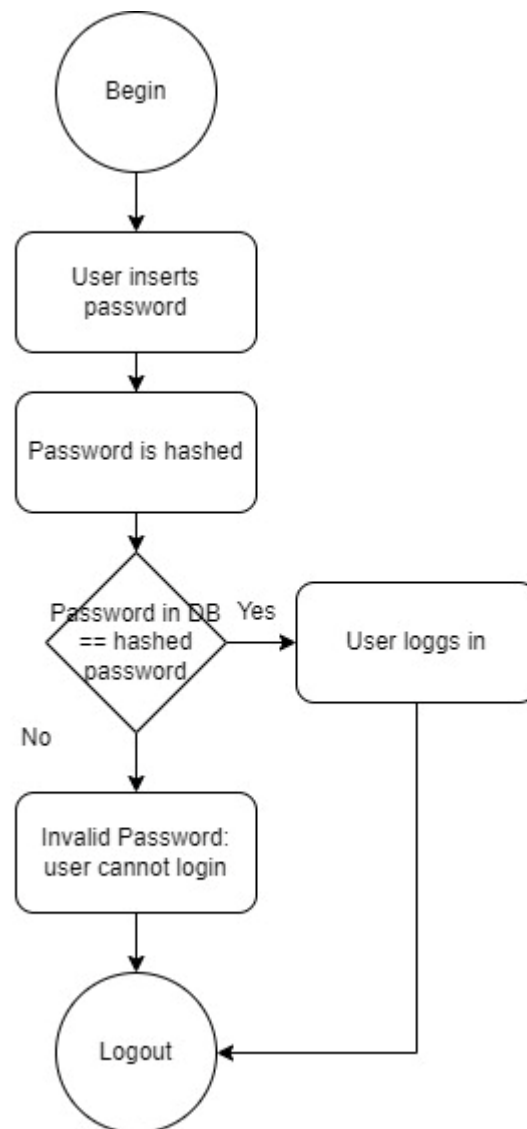
Figure 8. Flowchart: Login.

9.9. Flowchart: Change Administrator Password
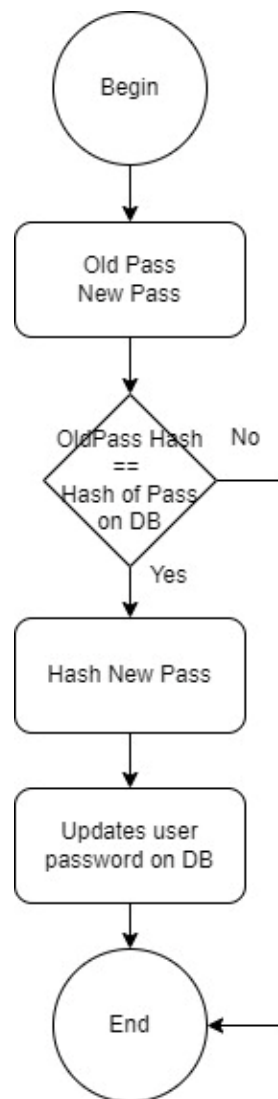
Figure 9. Flowchart: Change Administrator Password.

9.10. Flowchart: Update API Key
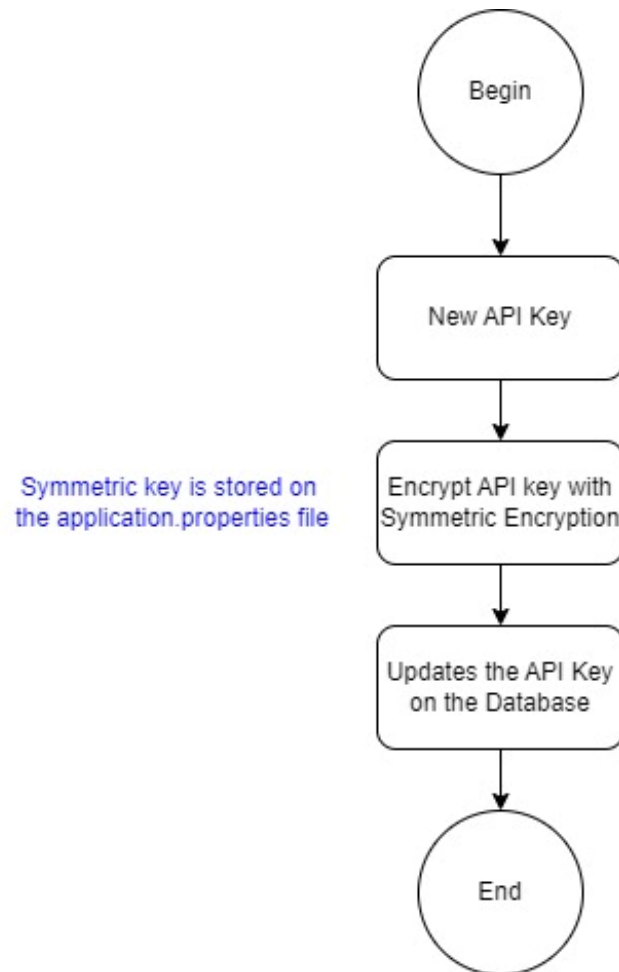
Figure 10. Flowchart: Update API Key.

9.11. Flowchart: Proxycurl Result to Excel
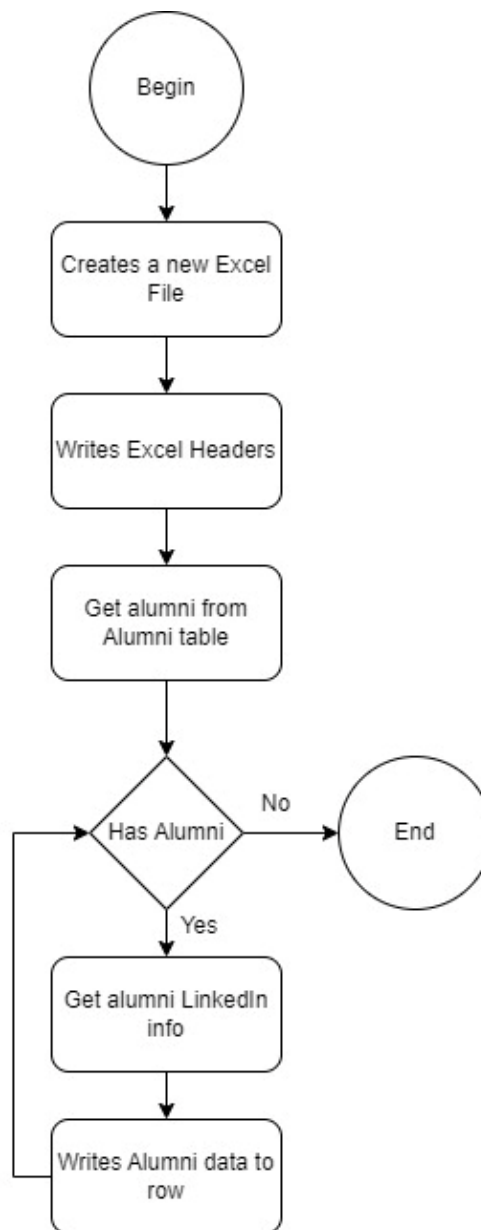
Figure 11. Flowchart: Proxycurl Result to Excel.

9.12. Flowchart: Replace Alumni
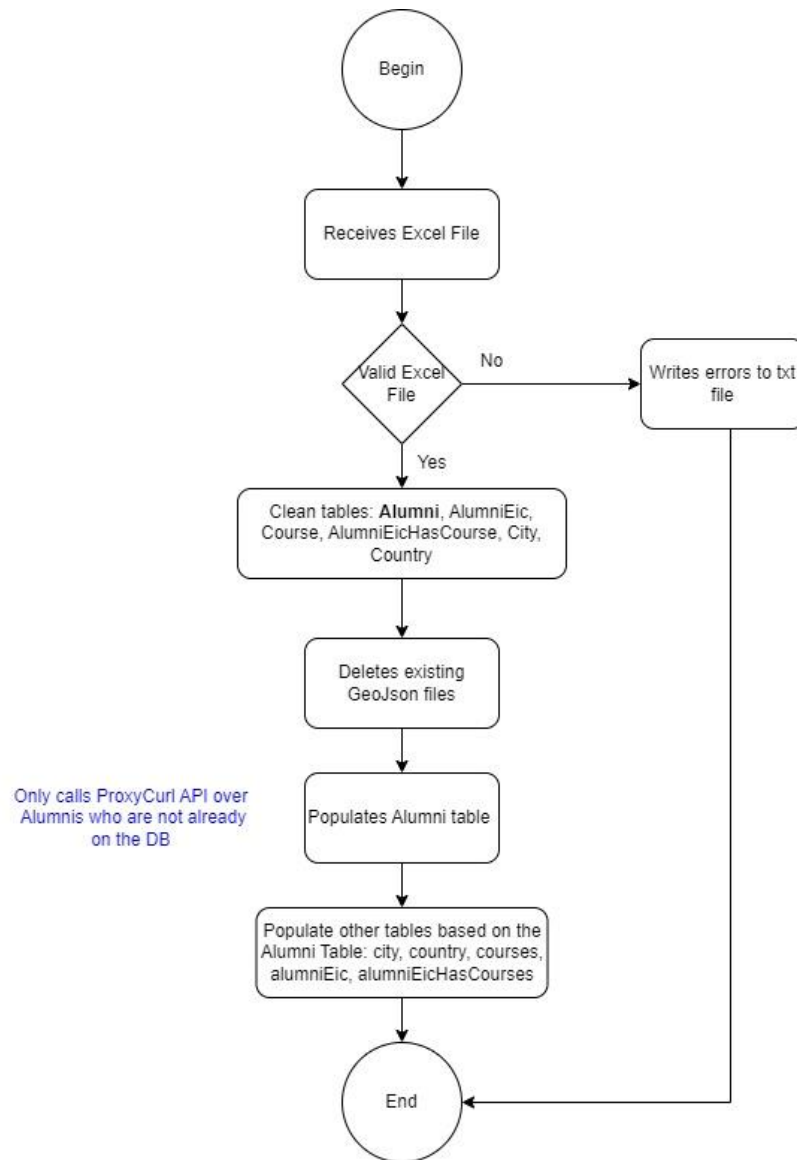
Figure 12. Flowchart: Replace Alumni.

9.13. Flowchart: Add Alumni
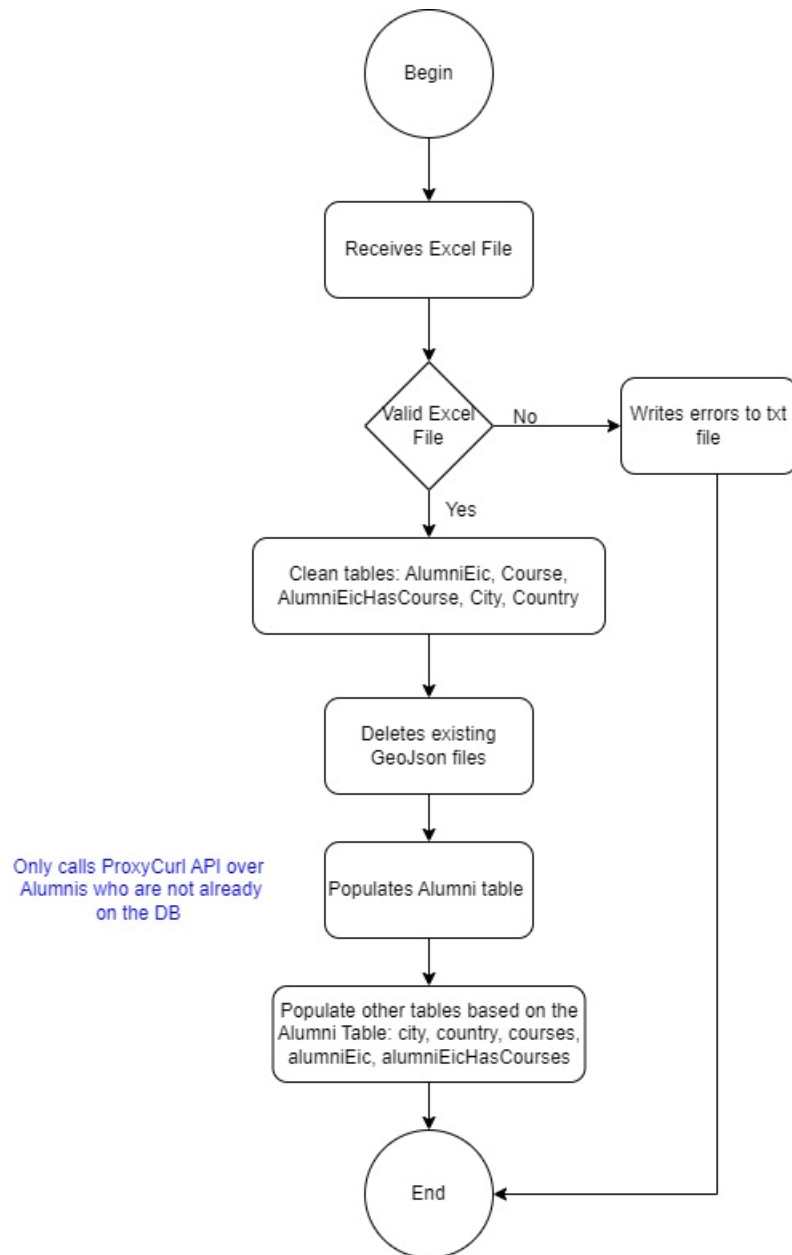
Figure 13. Flowchart: Add Alumni.

9.14. Flowchart: Update Alumni
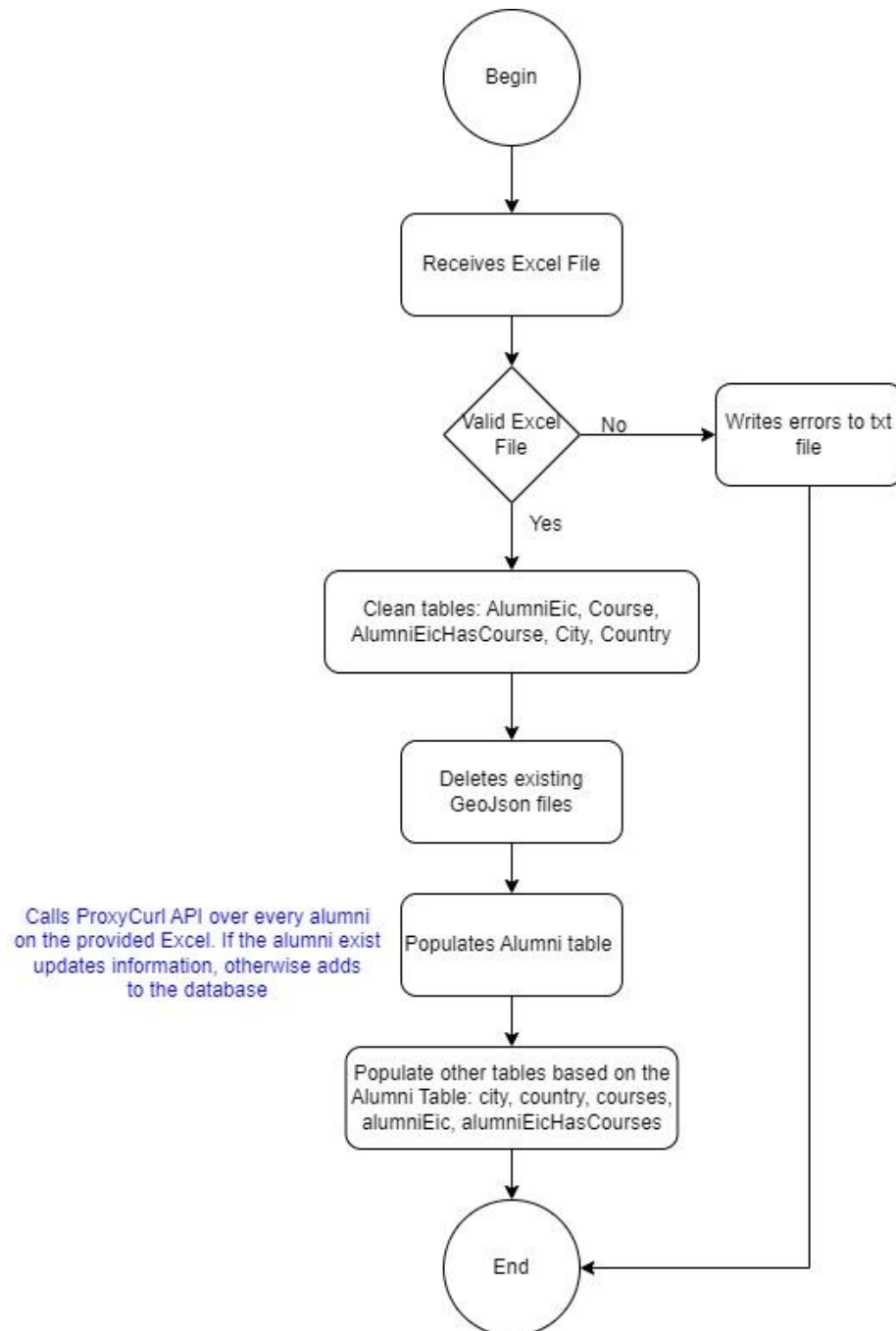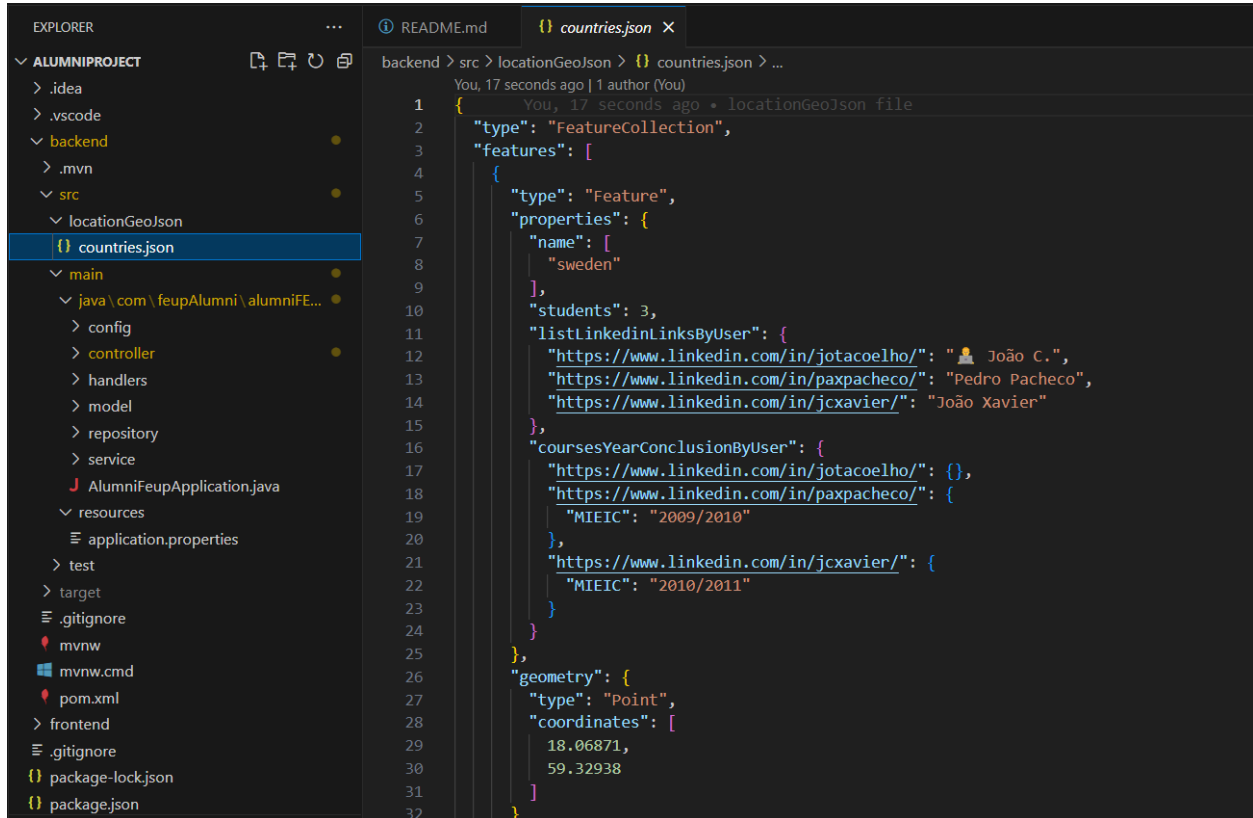
Figure 14. Flowchart: Update Alumni.

9.15. GeoJson File

Figure 15. GeoJson File Example.

9.16. Hardcoded Cities

```java
public class Location {

    // Converts city names that the GeoCoordinates API doesn't recognize to recognizable ones
    public static Map<String, String> convertUnrecognizableCities(){
        Map<String, String> convertCityNames = new HashMap<>(); // Converts unacceptable city names to acceptable ones
        convertCityNames.put("porto metropolitan area", "porto");
        convertCityNames.put("brussels metropolitan area", "brussels");
        convertCityNames.put("porto e região", "porto");
        convertCityNames.put("kraków i okolice", "kraków");
        convertCityNames.put("greater guimaraes area", "guimarães");
        convertCityNames.put("hamburg und umgebung", "hamburg");
        convertCityNames.put("braga e região", "braga");
        convertCityNames.put("greater viana do castelo area", "viana do castelo");
        convertCityNames.put("antwerp metropolitan area", "antwerp");
        convertCityNames.put("metropolregion berlin/brandenburg", "berlin");
        convertCityNames.put("greater ipswich area", "ipswich");
        convertCityNames.put("greater madrid metropolitan area", "madrid");
        convertCityNames.put("pontevedra y alrededores", "pontevedra");
        convertCityNames.put("greater cambridge area", "cambridge");
        convertCityNames.put("oslo og omegn", "oslo");
        convertCityNames.put("greater tokyo area", "tokyo");
        convertCityNames.put("greater barcelona metropolitan area", "barcelona");
        convertCityNames.put("greater cardiff area", "cardiff");
        convertCityNames.put("greater oslo region", "oslo");
        convertCityNames.put("greater aveiro area", "aveiro");
        convertCityNames.put("the randstad", "randstad");
        convertCityNames.put("geneva metropolitan area", "geneva");
        return convertCityNames;
    }

}
```

Figure 16. Hardcoded Cities.

9.17. Application.properties

```
#WARNING: Make sure you don't any uneccsary leading and trailing spaces
# configuration
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/alumnifeup
spring.datasource.username=alumniroot
spring.datasource.password=new_password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

geonames.username=jenifer12345

spring.servlet.multipart.max-file-size=20MB
spring.servlet.multipart.max-request-size=20MB

# Encryption constants
encryption.key=MiSQDMu6DeELxML90MjfQiyOCvcjnXglKYUKoGg5xMQ=
encryption.algorithm=AES

# GeoJson creation constants
json.defaultFileName=emptyFileLocation.json
json.fileLocation=/home/eic30anos/alumniProject/backend/src/locationGeoJson

# API Linkedin constants
# The problem of storing in: C:/alumniProject/frontend/public/Images is that the front end relaods because react app is constantly whatching
apiLinkedin.endpoint=https://nubela.co/proxycurl/api/v2/linkedin
apiLinkedin.savePicFolder=C:/alumniProject/frontend/public/Images

# API GeoCoordinates constants
apiGeoCoordinates.url=http://api.geonames.org/searchJSON
apiGeoCoordinates.username=jenifer12345

# Admin username
admin.username=admin
# Excel constants
    # Note: If more columns are added it's necessary to refactor method validateExcelFile from ExcelFilesHandler.java
    #     If the type of a certain column changes, for example from a number to a string, code needs refactor in the same place
excel.firstColumnName=Número
excel.secondColumnName=Nome
excel.thirdColumnName=Link Após Inquérito e Pag. SIGARRA
excel.forthColumnName=Cursos
excel.columnNumber=4
excel.sheet=0
excel.rowForNumero=0
excel.rowForNome=1
excel.rowForLinkedInLink=2
excel.rowForCursos=3
excel.linkedinPerfix=https://www.linkedin.com/in/
```

Figure 17. Application.properties.

9.18. Deployment: Frontend configuration

```
// Context for authentication
export const AuthContext = React.createContext();

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const login = () => {
    setIsLoggedIn(true);
  };

  const logout = () => {
    setIsLoggedIn(false);
  };

  return (
    <>
      <AuthContext.Provider value={{ isLoggedIn, login, logout }}>
        <Router>
          <Routes>
            <Route path="/alumnieiworld/:yearUrl?" element={<MapCmp/>} />
            <Route path="/alumnieiworld/admin" element={<AdminLogin/>} />
            <Route
              path="/alumnieiworld/adminDefinitions"
              element={
                <PrivateRoute><AdminSettings /></PrivateRoute>
              }
            />
            <Route path="/alumnieiworld/" element={<MapCmp/>} />
          </Routes>
        </Router>
      </AuthContext.Provider>
    </>
  );
}

export default App;
```

Figure 18. . Frontend Configuration: routing.

```
// Has functions related with the setup of the project when a new Alumni da
class setUp {

    /**
    * Fetches the geoJson
    */
    static async fetchGeoJson(courseFilter, yearsConclusionFilter, geoJsonT
        try {
            const params = new URLSearchParams({
                courseFilter: courseFilter,
                yearsConclusionFilter: JSON.stringify(yearsConclusionFilter
                geoJsonType: geoJsonType,
            });

            const response = await fetch(`/setupLocation/getGeoJson?${param
            if (response.ok) {
                const blob = await response.blob();
                console.log('GeoJson blob successfully fetched.');
                return blob;
            } else {
                console.error('GeoJson fetching failed.');
            }
        } catch (e) {
            console.log('Error while fetching the geoJson', e);
        }
    }

    // Verifies if the password is correct
    static async verifyCorrectPassword(password){
        try{
            const data = JSON.stringify({ password });
            const response = await fetch('/admin/verifyPass', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',  // Set the content
```

Figure 19. Frontend configuration: requests.

9.19. Nginx configuration

```
server {
        listen 80;
        server_name eic30anos.fe.up.pt;
        root /home/eic30anos/public_html/;
        index index.html index.htm;

        return 301 https://eic30anos.fe.up.pt$request_uri;
}

server {
        listen 443 ssl http2;
        root /home/eic30anos/public_html/;
        index index.html index.htm;
        server_name eic30anos.fe.up.pt;
        include ssl.conf;
        ssl_certificate /etc/ssl/eic30anos.fe.up.pt.pem;
        ssl_certificate_key /etc/ssl/private/eic30anos.fe.up.pt.key;

        location = / {
                return 301 https://eic30anos.fe.up.pt/pt/event;
        }
        location /alumnieiworld {
                alias /home/eic30anos/alumniProject/frontend/build/;
                try_files $uri $uri/ /alumnieiworld/index.html;
        }
        location /ws {
                proxy_set_header X-Real-IP  $remote_addr;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_pass http://localhost:8080;
        }
        location /setupLocation {
                proxy_set_header X-Real-IP  $remote_addr;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_pass http://localhost:8080;
        }
        location /admin {
                proxy_set_header X-Real-IP  $remote_addr;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_pass http://localhost:8080;
        }
        client_max_body_size 150M;
}
```

Figure 20. Nginx Configuration.

9.20. Backend service

```
[Unit]
Description=Backend Service
After=network.target

[Service]
User=eic30anos
WorkingDirectory=/home/eic30anos/alumniProject/backend
ExecStart=/usr/bin/java -jar /home/eic30anos/alumniProject/backend/target/alumniFEUP-0.0.1-SNAPSHOT.jar
Restart=always

[Install]
WantedBy=multi-user.target
```

Figure 21. Backend service.
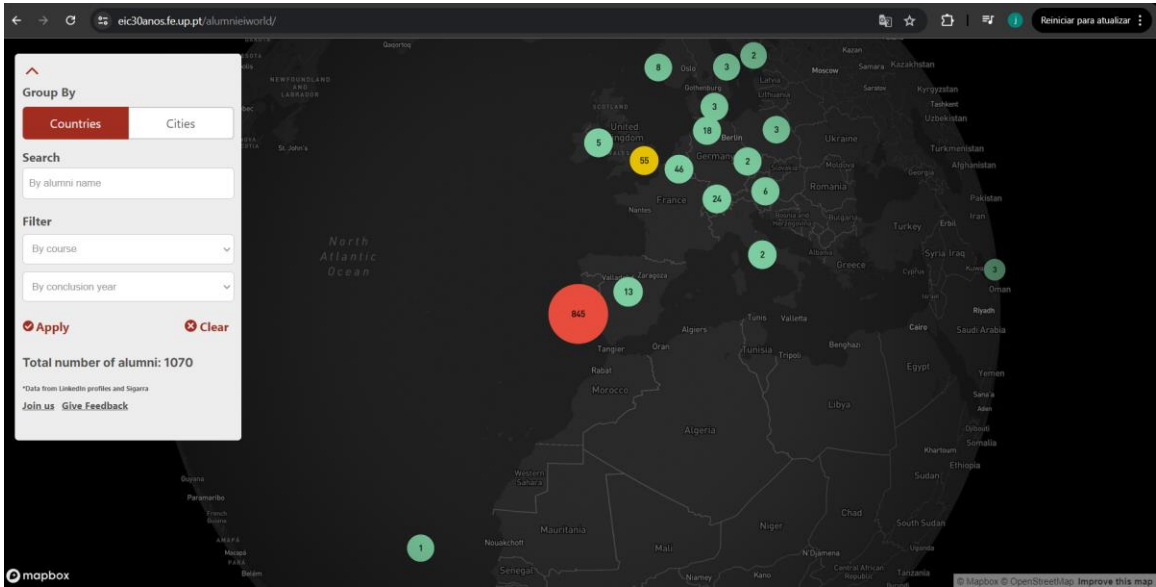
9.21. Solution's interfaces

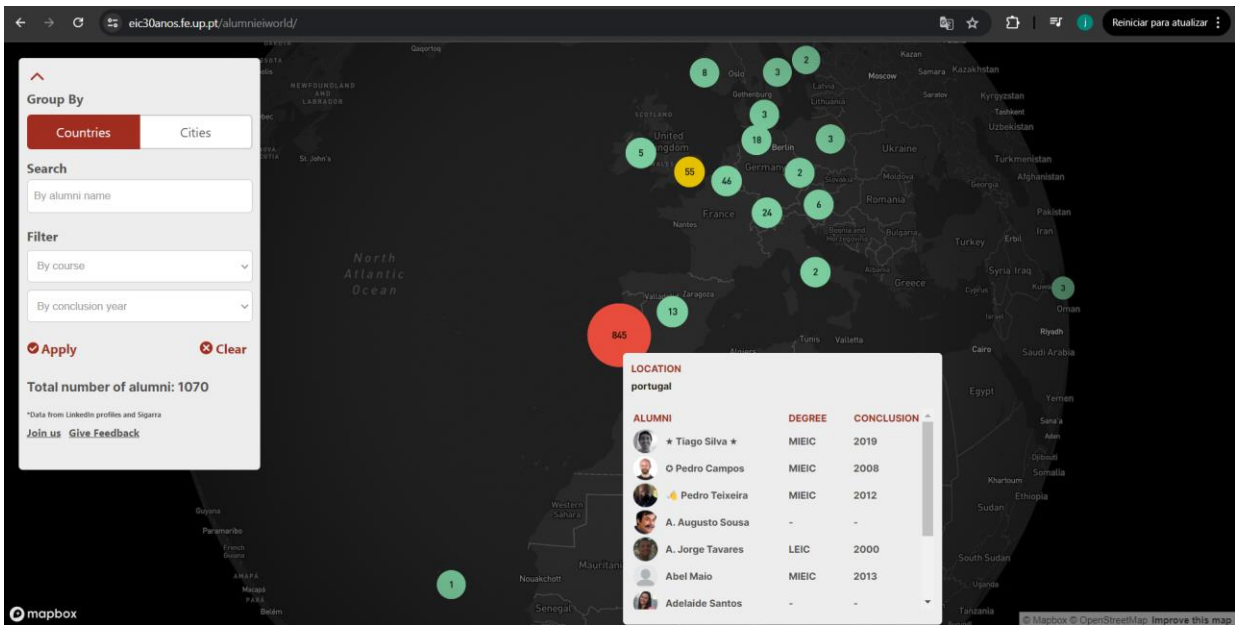*Figure 22. Accessing the 3D map with alumni distribution without URL year filtering.*

Figure 23. Accessing the 3D map with alumni distribution without URL year filtering.
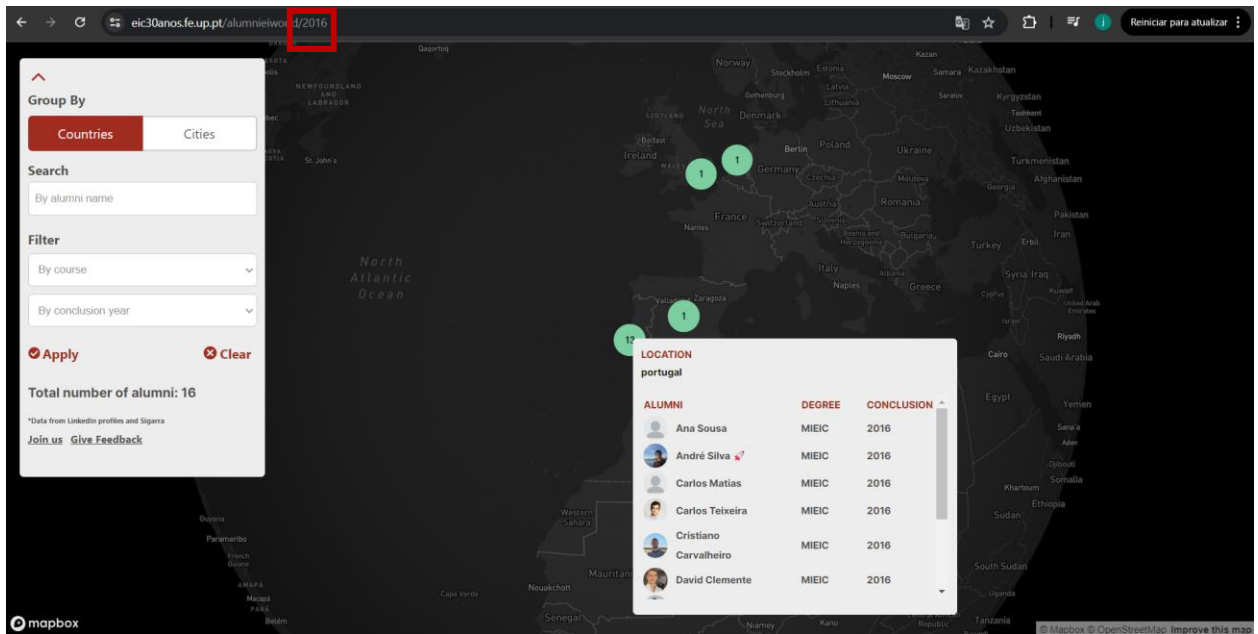
Figure 24. Accessing the 3D map with alumni distribution with URL year filtering.

Figure 25. Admin Login Page.
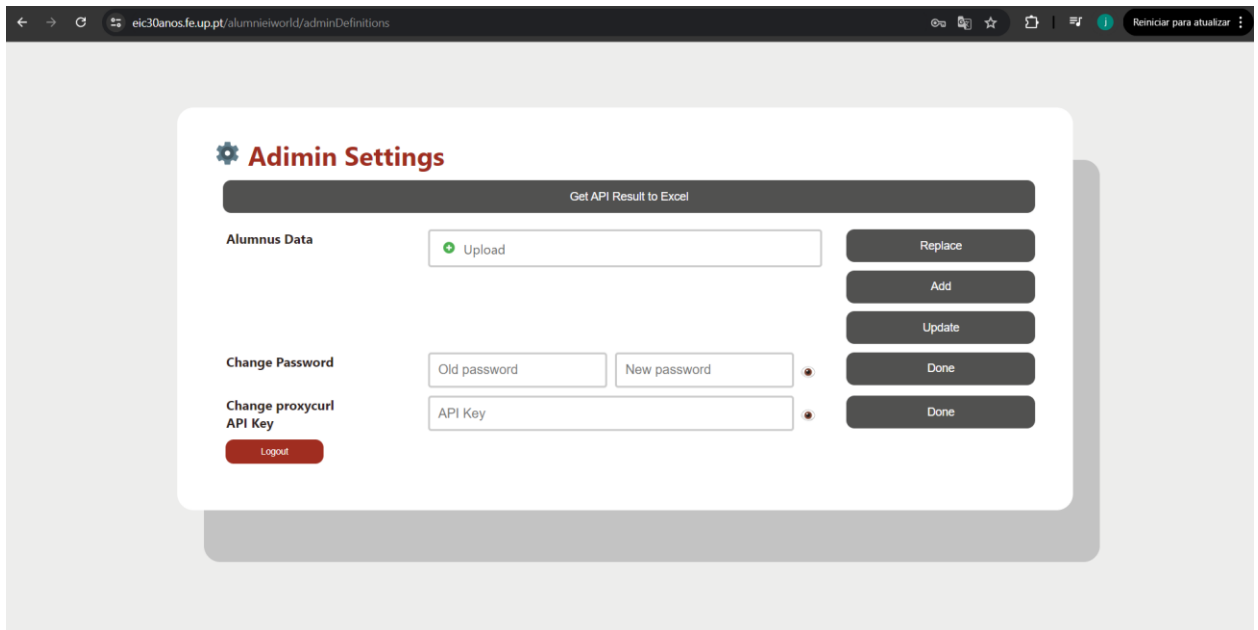
Figure 26. Admin Definitions Page.
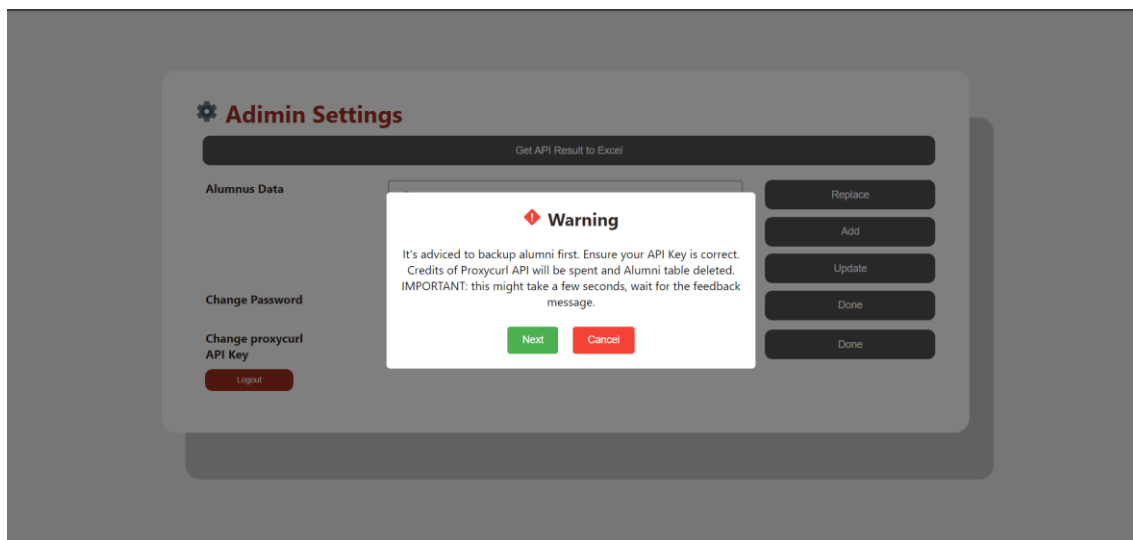
Figure 27. Warning for Replace Button.
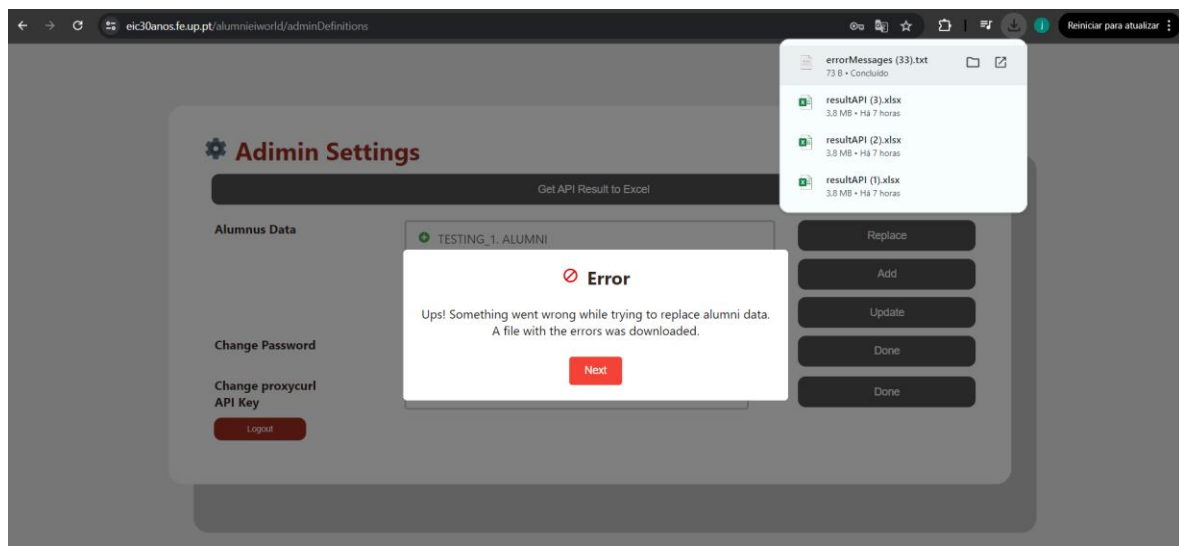
Figure 28. Error: invalid Excel file.

9.22. Backend: Strategy Patterns
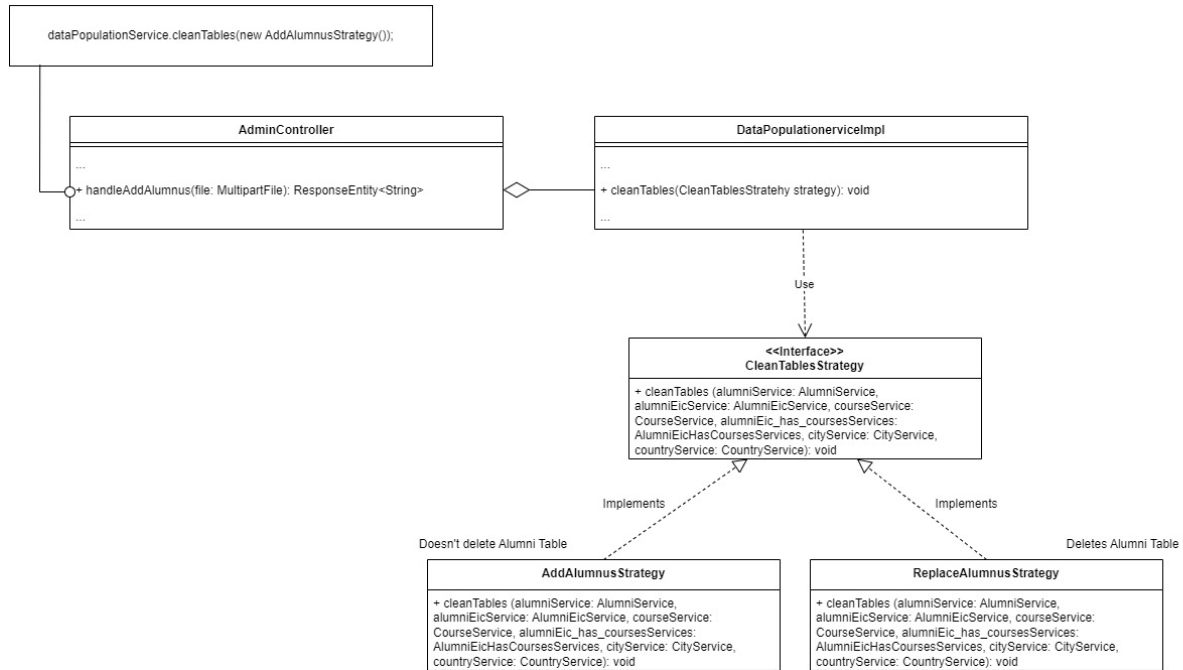
Figure 29. Strategy Pattern on Clean Tables.
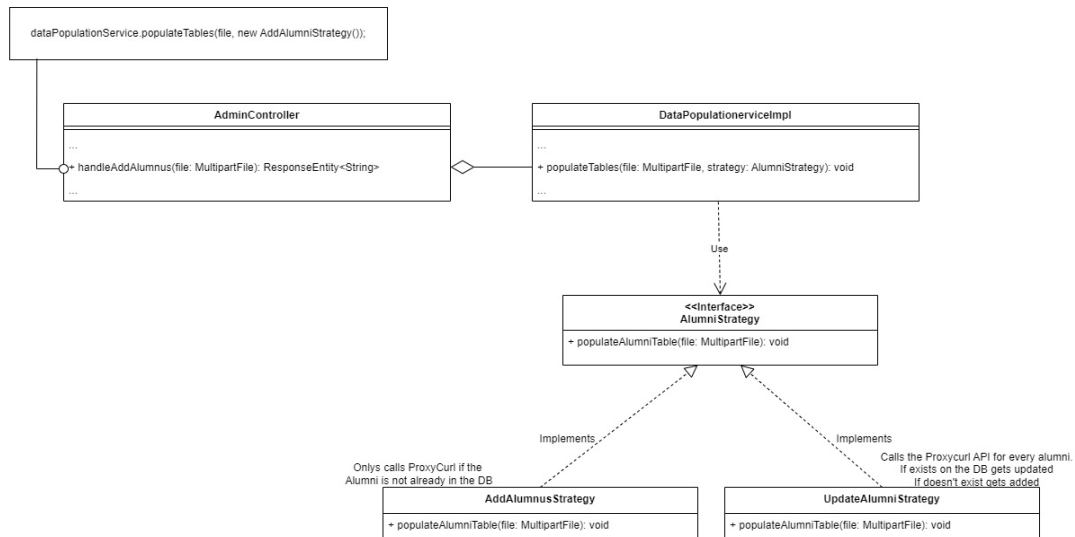
Figure 30. Strategy Pattern on Populate Alumni Table.

9.23. Excel Structure: Add alumni information

Número	Nome	Link Após Inquérito e Pag. SIGARRA	Cursos
200706596	Abel Almeida Maio	https://www.linkedin.com/in/abel-maio-53772a224/	MIEIC 2012/2013
201907487	Adelaide Isabel Miranda dos Santos	https://www.linkedin.com/in/adelaide-santos-5957a6220/	L.EIC 2021/2022
200604150	Admilo Élvio Mendes Ribeiro	https://www.linkedin.com/in/ribadmilo/	MIEIC 2011/2012
200005241	Adriano Filipe Pinheiro Teixeira	https://www.linkedin.com/in/adrianoteixeira/	LEIC 2004/2005
200003752	Adriano José da Fonte Meira	https://www.linkedin.com/in/adriano-meira-1b660917/	LEIC 2004/2005
200702653	Afonso da Rocha Graça	https://www.linkedin.com/in/afonsograca/	MIEIC 2011/2012
201506239	Afonso Jorge Moreira Maia Ramos	https://www.linkedin.com/in/afonsojramos/	MIEIC 2019/2020
200901960	Afonso Manuel Duarte de Melo Rosa	https://www.linkedin.com/in/afonso-rosa-35325376/	MIEIC 2013/2014
201009023	Afonso Neves Caldas	https://www.linkedin.com/in/afonso-caldas/	MIEIC 2014/2015
200005183	Alberto José Alves de Lemos	https://www.linkedin.com/in/alberto-lemos-995a971/	LEIC 2005/2006
199503451	Alberto José Ferreira Soares da Mota	https://www.linkedin.com/in/albertomota/	LEIC 1999/2000
200701621	Alexandre Campos Perez	https://www.linkedin.com/in/alexandrecperez/	MIEIC 2011/2012
199400152	Alexandre Jorge Teixeira Miranda Pinto	https://www.linkedin.com/in/alexandre-miranda-pinto/	LEIC 1998/1999
200601370	Alexandre José Monteiro Rodrigues	https://www.linkedin.com/in/ajrodrigues/	MIEIC 2010/2011
200002575	Alexandre Miguel Fragueiro Gonçalves	https://www.linkedin.com/in/alexandre-gonçalves-774a88a0/	LEIC 2005/2006
201005461	Alexey Seliverstov	https://www.linkedin.com/in/alexeyseliverstov/	MIEIC 2014/2015
200500425	Alfredo Miguel da Cunha Silvestre	https://www.linkedin.com/in/alfredo-silvestre-43b52b1/	MIEIC 2009/2010
200808117	Alice Clemente Perpétua	https://www.linkedin.com/in/aliceperpetua/	MIEIC 2013/2014
201603694	Álvaro Francisco Barbosa Miranda	https://www.linkedin.com/in/a-francisco-miranda/	L.EIC 2022/2023 M.EIC 2022/2023
200104999	Álvaro Gabriel Machado Caldas	https://www.linkedin.com/in/%C3%A1lvaro-caldas-pmp%C2%AE-7a31b711/	MIEIC 2008/2009
200202461	Álvaro José Valente Vasconcelos	https://www.linkedin.com/in/alvarovasconcelos/	MIEIC 2007/2008
200505486	Álvaro Manuel da Silva Monteiro	https://www.linkedin.com/in/alvarosilvamonteiro/	MIEIC 2008/2009
199703849	Amândio José Carvalho Alves da Silva	https://www.linkedin.com/in/amandiosilva/	LEIC 2001/2002
200803809	Amaro Gonzaga Martins da Silva	https://www.linkedin.com/in/zagasilva/	MIEIC 2012/2013
199504017	Ana Beatriz Furtado Stone	https://www.linkedin.com/in/ana-beatriz-stone/	LEIC 1999/2000
201906230	Ana Beatriz Melo Aguiar	https://www.linkedin.com/in/beatrizmaguiar/	L.EIC 2021/2022
201201786	Ana Carolina Ribeiro Moura	https://www.linkedin.com/in/ana-moura/	MIEIC 2016/2017
201002992	Ana Catarina Gonçalves Gomes	https://www.linkedin.com/in/anaggomes/	MIEIC 2014/2015

Figure 31. Excel File structure expected for buttons: Update, Add, Replace Alumni data.