University of Crete
Department of Computer Science

Development of a REST API and a web-based client application for the student population data available from the University of Crete Open Data Catalog

Emmanouil Filippakis

Supervisor: Yiannis Tzitzikas

Co-Supervisors: Dimitris Plexousakis, Marios Pitikakis

Time period: 20/10/2020 – 01/04/2021

Contents

Timetable

Description	From	Until	Condition
Phase programming	20/10/2020	27/10/2020	COMPLETED
Record goals, types of users, requirements	28/10/2020	03/11/2020	COMPLETED
Architecture and Tools	04/11/2020	15/11/2020	COMPLETED
Version A	16/11/2020	30/12/2020	COMPLETED
Screen Design	03/01/2021	15/01/2021	COMPLETED
Version B	16/01/2021	30/01/2021	COMPLETED
Tests	01/02/2021	08/02/2021	COMPLETED
Complete application, Report	10/02/2021	01/04/2021	COMPLETED

1 Introduction

The purpose of the document is to present the design and development of a REST API for the open availability of mainly historical data about the student population of UoC. This data is available from the Open Data Catalog of the University of Crete, which uses the CKAN platform .

Upon completion of the document there will be a complete picture of what the developing system is and does. The development takes place within the framework of the diploma thesis at the Department of Computer Science of the University of Crete during the period from October 2020 to April 2021.

1.1 General description

With the implementation of this system,

1. The design and development of a REST API for the open availability of mainly historical data (in different forms and formats) regarding the student population of the University of Crete is achieved.
2. A client application is created to visualize the available statistical data, which makes use of the above REST API.
3. A visual documentation is implemented (Swagger) of the REST API that mainly helps developers and simplifies the implementation of the back - end and the consumption/use of the API by the applications (client side).
4. A client application is implemented, which automates the management and import of an XLSX file into a database table.

1.2 System purpose

The software of the systems that will be developed will serve the open data service of the University of Crete and will offer a much more efficient utilization of the data compared to the current situation.

Via Swagger UI , each developer will be able to use specific elements in the form of JSON , CSV , HTML , XML as well as for some functions, representation graphs.

Through REST client application , search/filtering and data visualization services (graphs per year) will be provided to the user .

Via Upload client application , the Admin is provided with effective tools for managing and entering data into the Database.

This application, by simplifying and facilitating the utilization of data that is only in XLSX format, succeeds in making the data more accessible to the public of developers.

2 Usage Scenarios and Types of Users

The system utilizes data from the University of Crete regarding:

1. Undergraduate students, by gender and year of study.
2. Undergraduate students, by gender and age.
3. Postgraduate students by year of graduation and gender.
4. Graduated undergraduates by year of graduation.
5. Postgraduate students by year of study and gender.
6. PhD candidates by year of study and gender.
7. PhDs by year of graduation and gender.

All the above statistics are available per Academic Department of the University of Crete and specifically for each of the following 16 Departments:

Department of Literature
Department of History and Archeology
Department of Philosophical and Social studies
Educational Department of Primary Education
Pedagogical Department of Preschool education
Sociocoly Department
Department of Economics
Psychology Department
Department of Political Science
Department of Mathematics and Applied Mathematics
Physics Department
Biology Department
Department of Chemistry
Department of Computer Science
Department of Materials Science and Technology
Medical School

The Swagger UI based on the above data has users:

- The Developers , who will extract specific categories of data, based on the parameters they enter, from the database in JSON , HTML , CSV , XML format as well as image / png in graph format.

The REST client application has users :

- Any visitor who wishes to browse in a simple and easy-to-use way the available data, which can be made available either in the form of tables or in the form of graphs.

The Upload client application has users :

- The Admin , who has the ability to upload an XLSX file to the application 's directory and to automatically import it into the NW.

3 Requirements

The above REST design API was made with practicality in mind, as well as minimalism. The main requirement, therefore, was the implementation of lean and robust code both in the Front - End and in the Back - End part.

Each function is independently implemented in terms of code, thus offering the application significant performance, but above all security. Also, the application offers portability for any kind of system, since there are no hard - coded parts in the code.

It should also be noted that the system is largely extensible, since each new function is an independent method in the main code.

4 Application Screens

4.1 Swagger UI

The implementation of Swagger UI aims for simplicity, without making its use complex and confusing both for the developer and for the Admins .

Swagger is a free and open source source framework , which by reading the structure of the API, can automatically create beautiful and interactive documentations . Finally it can also automatically generate client libraries for the API in multiple languages and explore other features such as automated testing. This is achieved by asking the API to return a YAML or JSON containing a detailed description of the entire API. (https://swagger.io/).

Visually the basic roles and types of Swagger users APIs are presented below:W

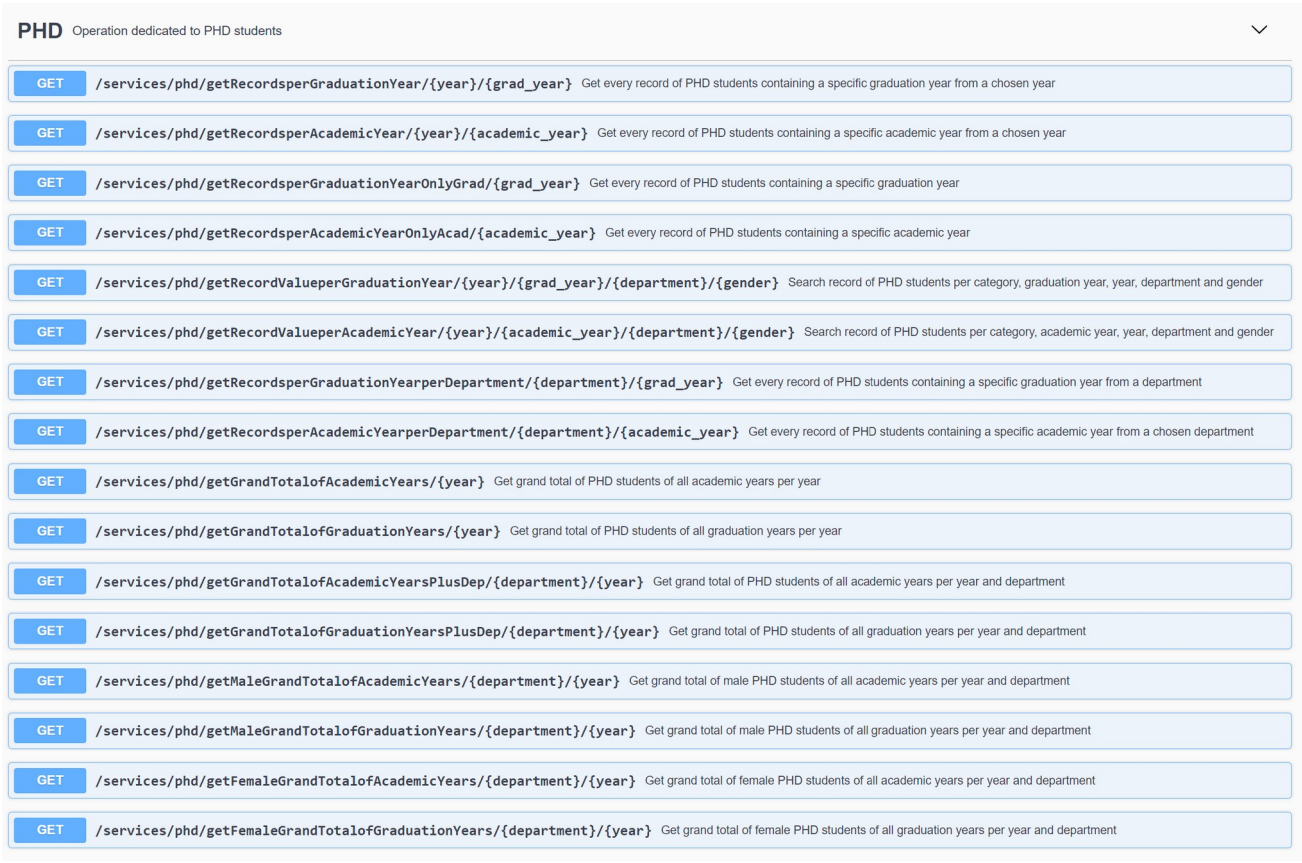


Figure 4.1.1 Functions regarding the data for the doctoral students of the University of Crete

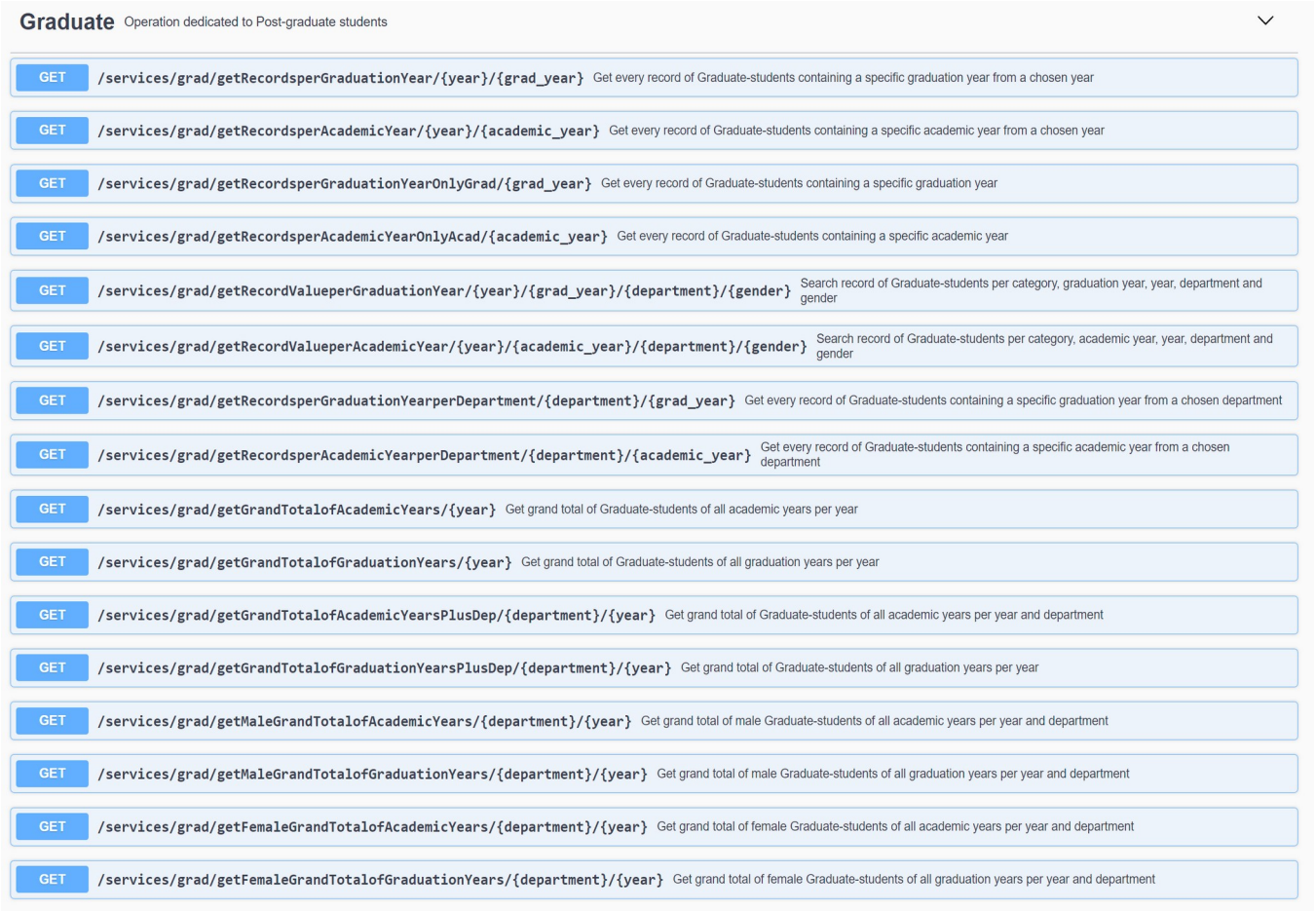


Figure 4.1.2 Operations regarding data for postgraduate students of the University of Crete

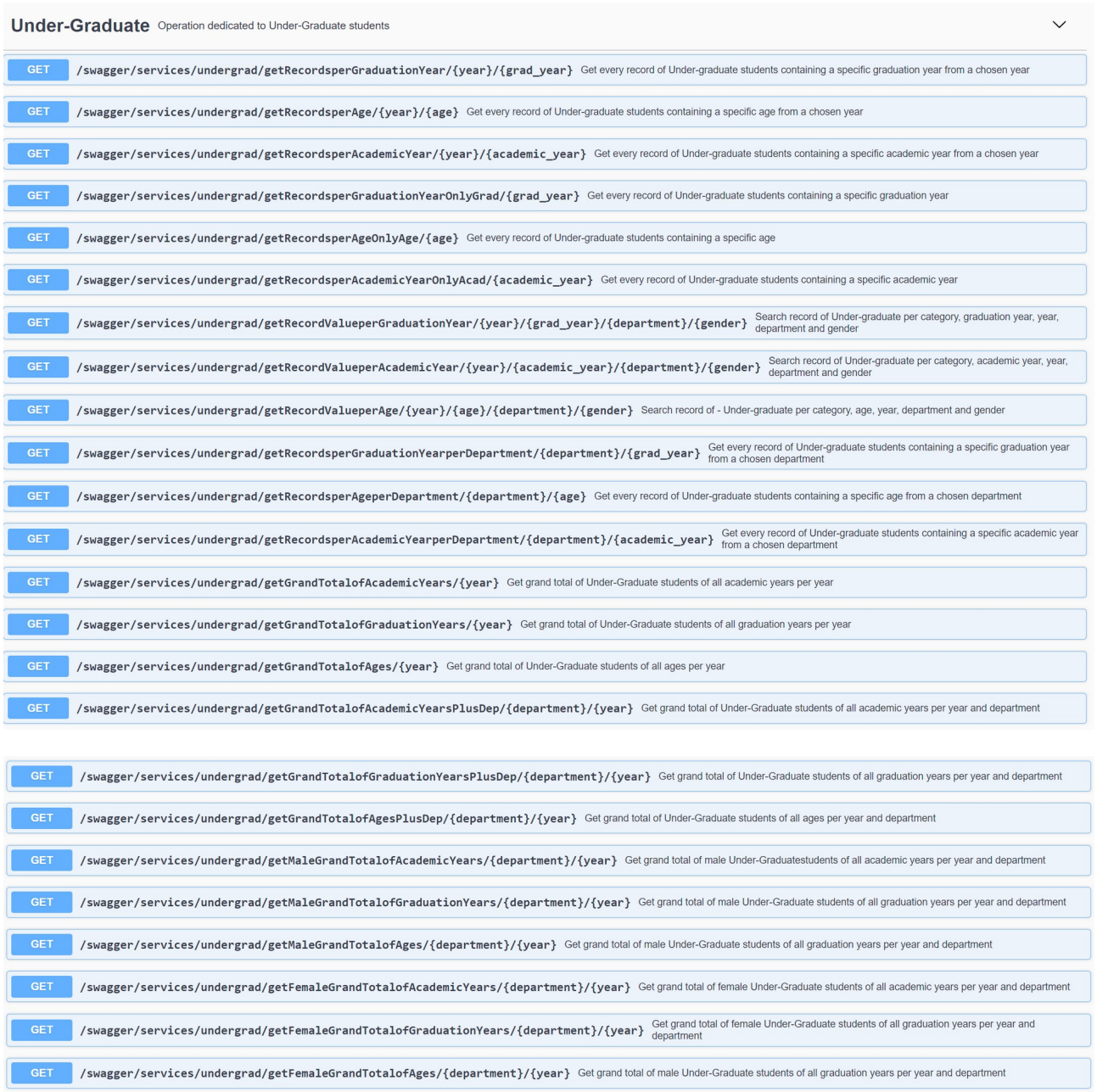


Figure 4.1.3 Functions regarding the data for the undergraduate students of the University of Crete

Specifically for developers, efforts have been made to make the UI as simplified and usable as possible. The variables in each function are enumerated variables , and the formats of the result differ depending on the operation.

Below is an example of the total PhD students, of all available academic years, by specific year in each format,

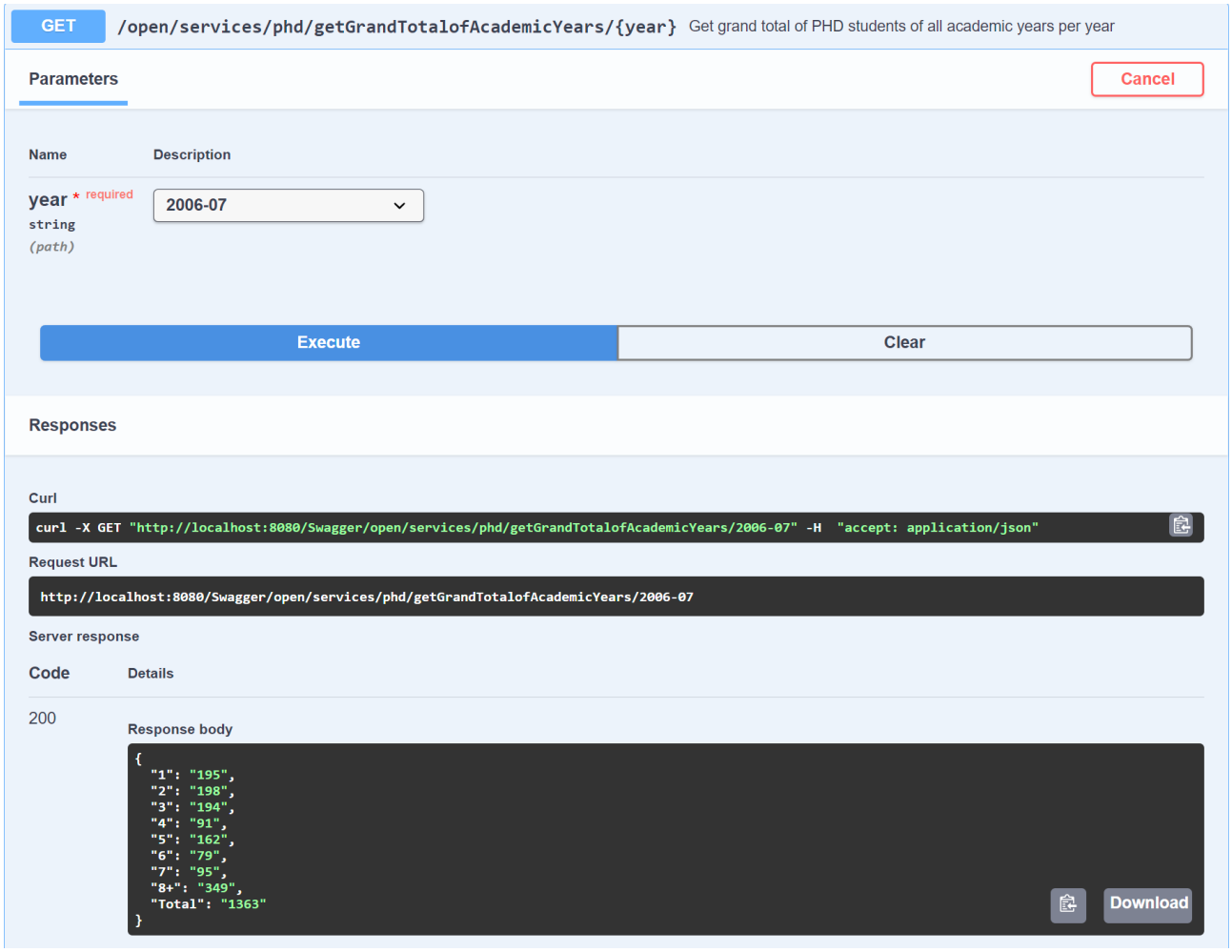


Figure 4.1.4 In JSON format

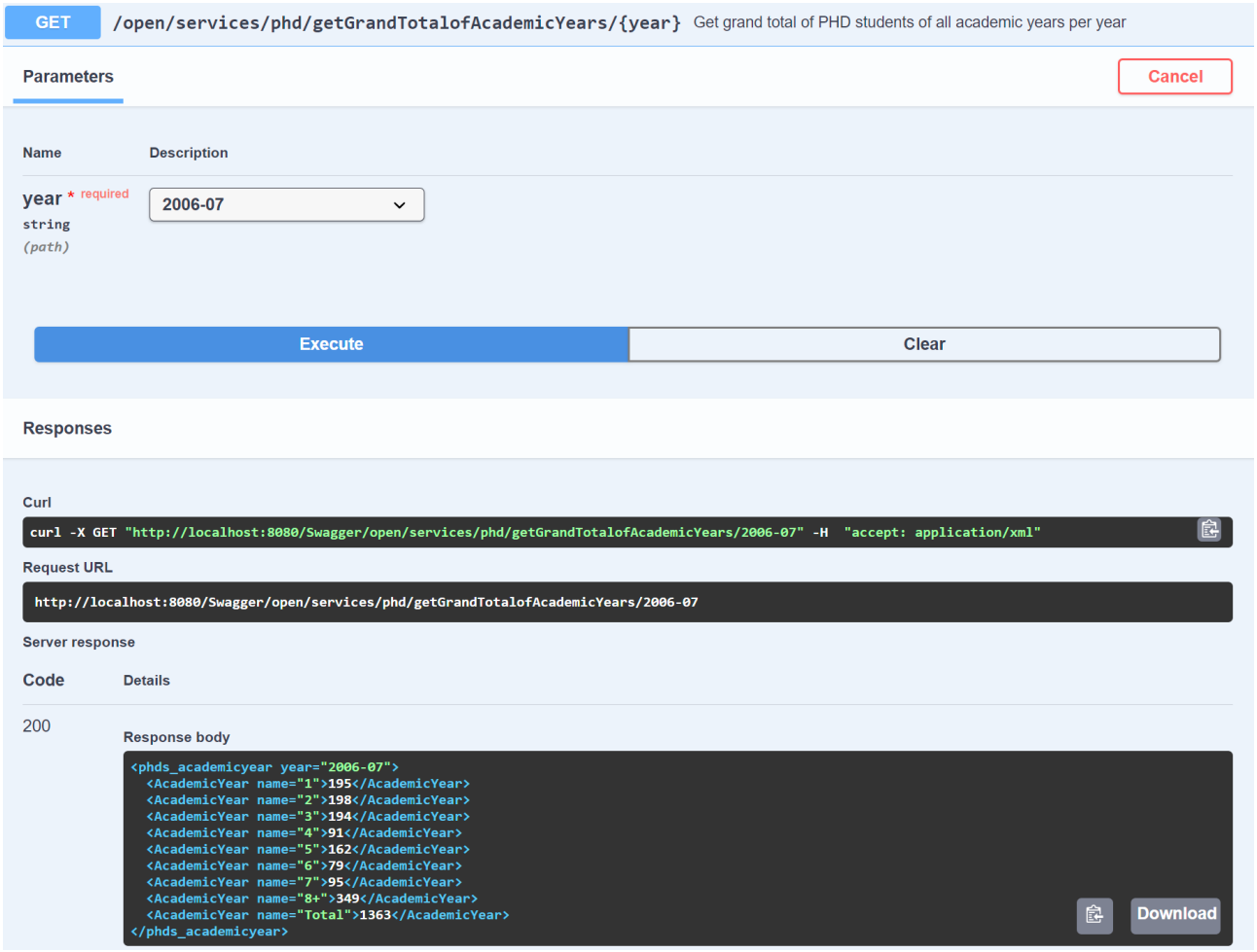


Figure 4.1.5 In XML format

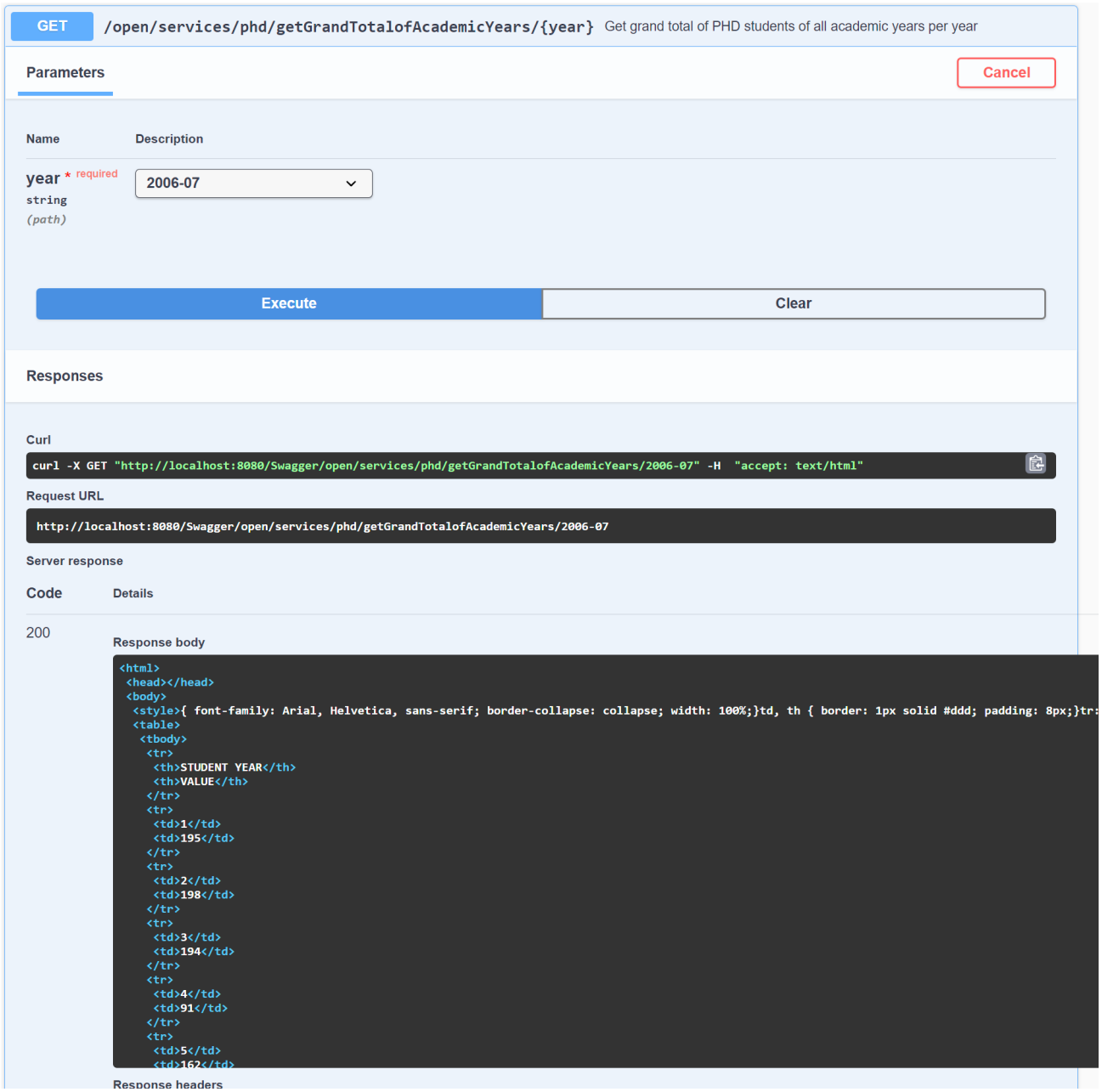


Figure 4. 1 . 6 In HTML format

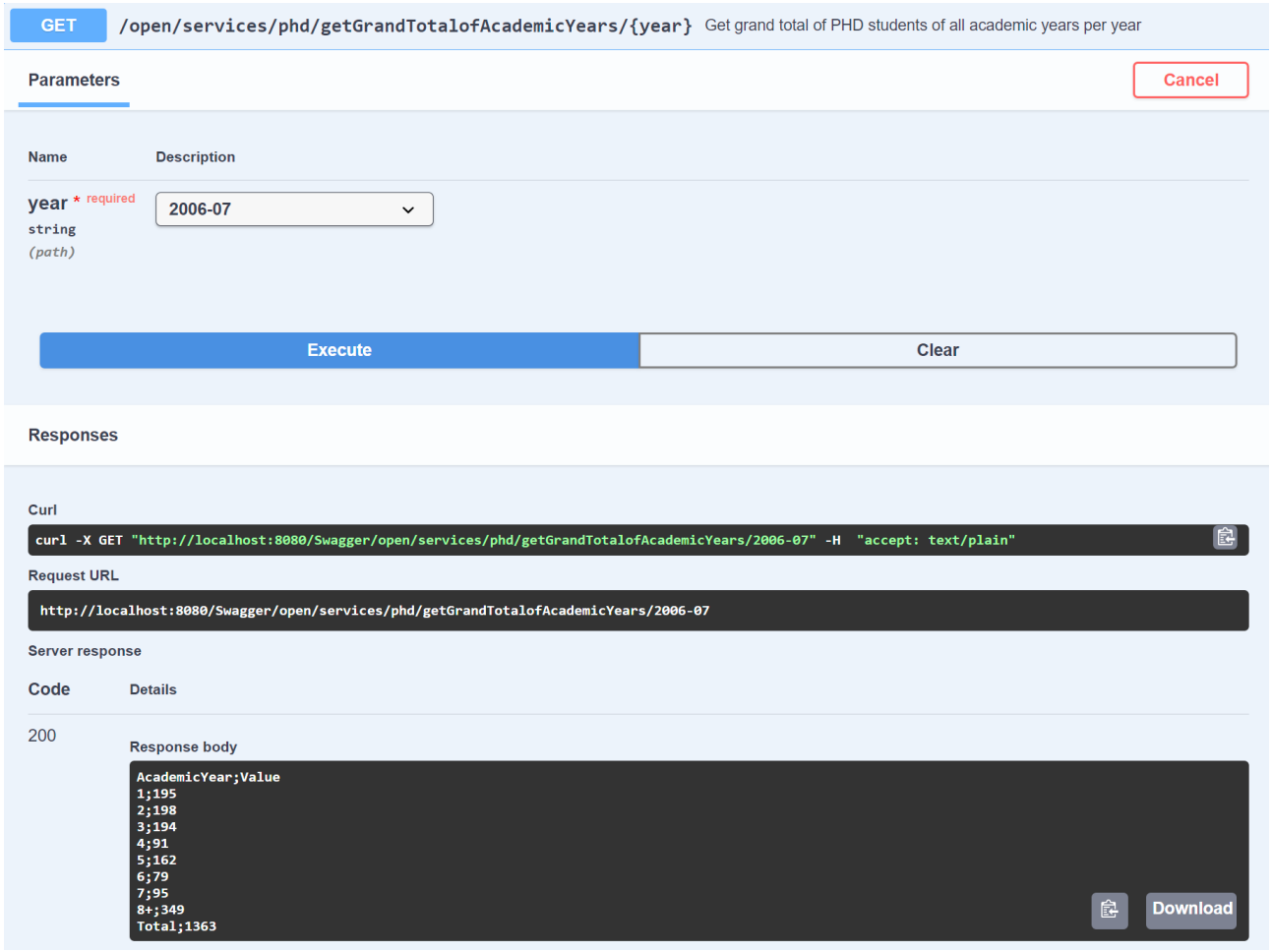


Figure 4.1.7 In CSV format

GET /swagger/services/phd/getGrandTotalofAcademicYears/{year} Get grand total of PHD students of all academic years per year

Parameters Cancel

Name Description

year * required
string
(path)

2006-07

Execute Clear

Responses

Curl

```
curl -X GET "http://147.52.17.140:8080/Swagger/swagger/services/phd/getGrandTotalofAcademicYears/2006-07" -H  "accept: image/png"
```

Request URL

```
http://147.52.17.140:8080/Swagger/swagger/services/phd/getGrandTotalofAcademicYears/2006-07
```

Server response

Code Details

200

Response body

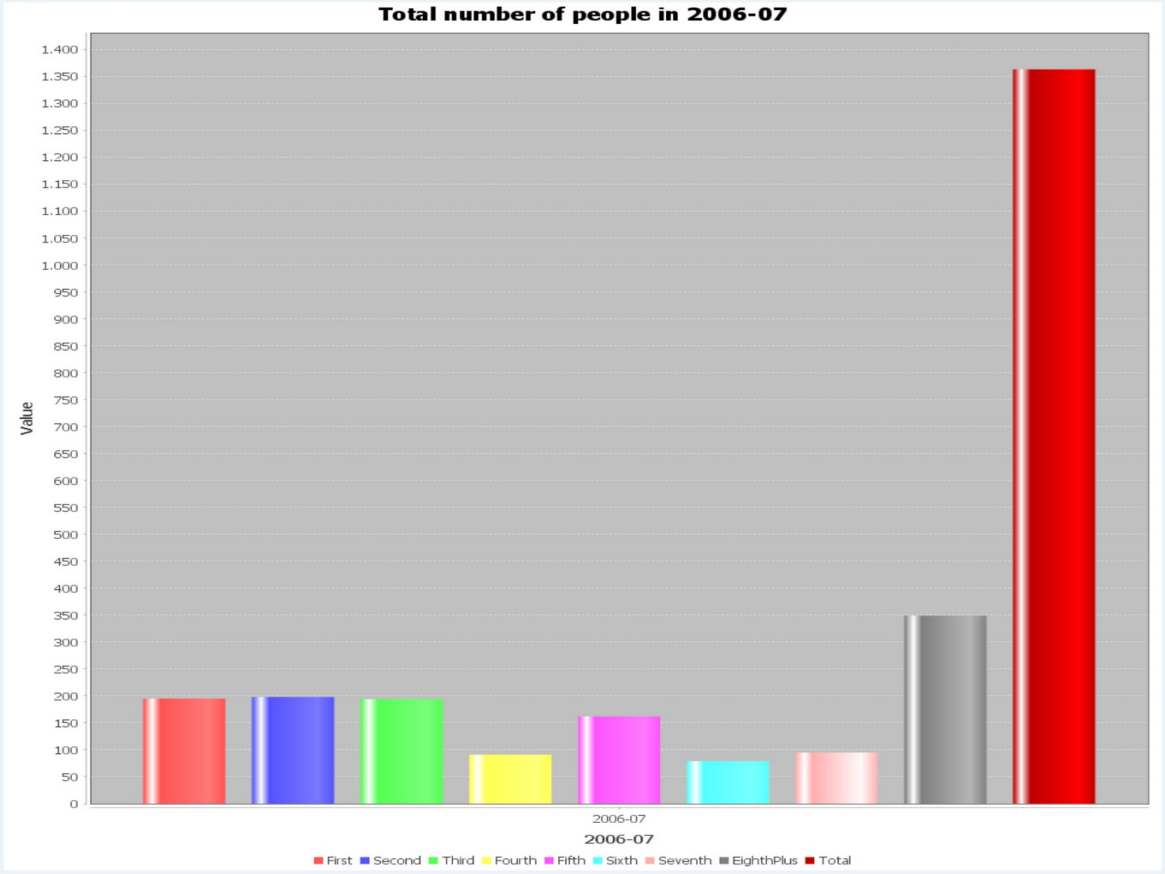


Figure 4.1.8 In graph form

4.2 REST client application

The client application is an example of using REST API. Depending on the user's choices, the application returns the requested statistical data, visualized. The web pages are created dynamically so that the respective table or graph is displayed each time.

The implementation of REST client application aims for elegance, without making its use complex, since it refers to simple users. There are two main functionalities in the application: a) data visualization with tables (**DATA** button) and b) data visualization with graphs (**CHARTS** button). It is implemented with a dropdown menu that leads the user to the desired option.

Contents of the dropdown menu DATA :

1. Graduate students per
    a. Graduation year
    b. Academic year
2. PhD students per
    a. Graduation year
    b. Academic year
3. Undergraduate students per
    a. Graduation year
    b. Academic year
    c. Age

Contents of the dropdown menu CHARTS :

1. Graduate students per
    a. Graduation year
    b. Academic year
2. PhD students per
    a. Graduation year
    b. Academic year
3. Undergraduate students per
    a. Graduation year
    b. Academic year
    c. Age

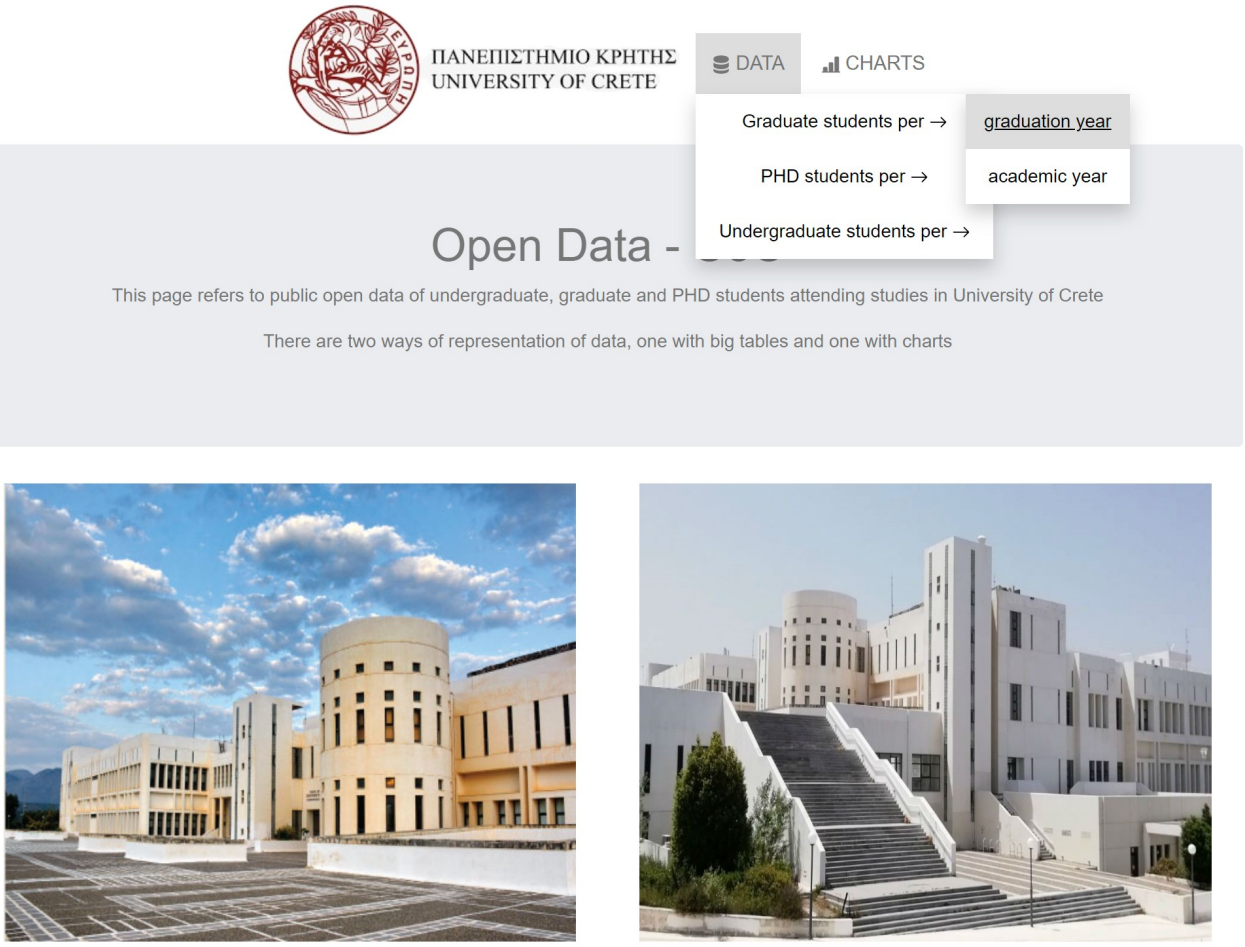


Figure 4.2.1 Welcome page

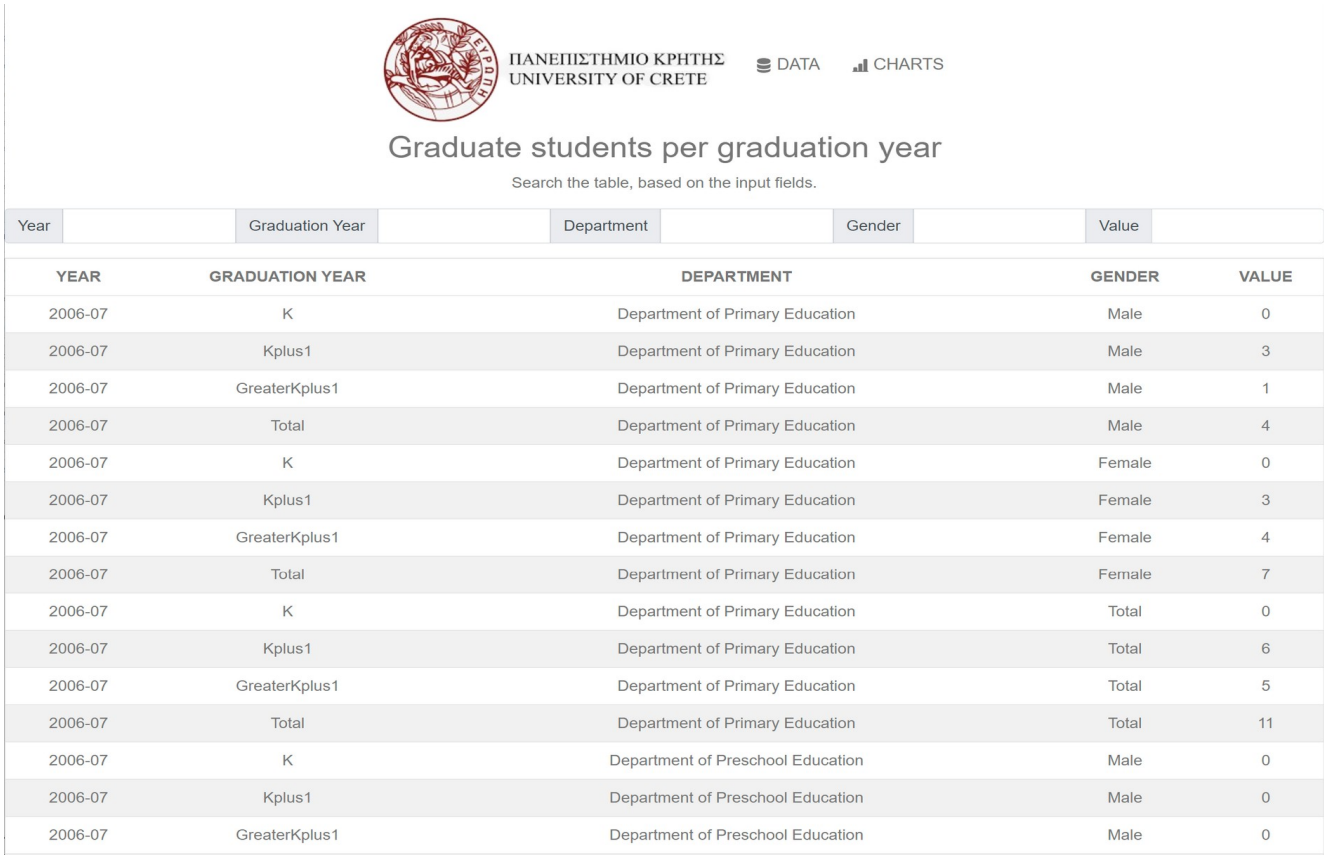


Figure 4.2.2 Example of a table in the DATA category with all the data of postgraduate students by year of graduation

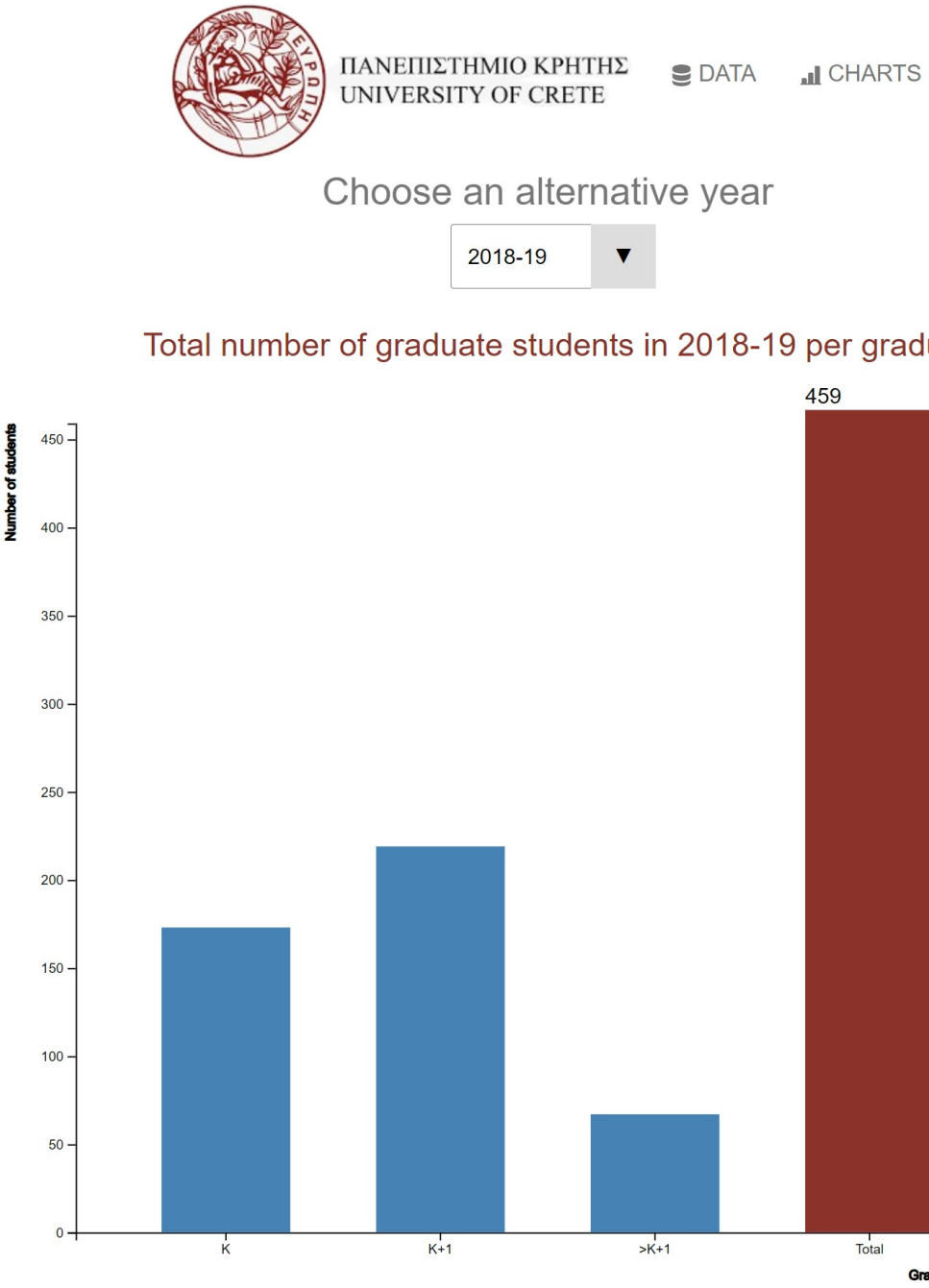


Figure 4.2.3 Example of a graph in the CHARTS category of postgraduate students in the period 2018-19

4.3 Upload client application

That Upload client application , is an auxiliary application for the Admin , which undertakes the entire process of uploading a statistical data file (in xlsx format) by the Admin to the application directory, mapping Greek words to English and entering the modified data into a table in the Database.

Figure 4.3.1 Page reception of Upload client application

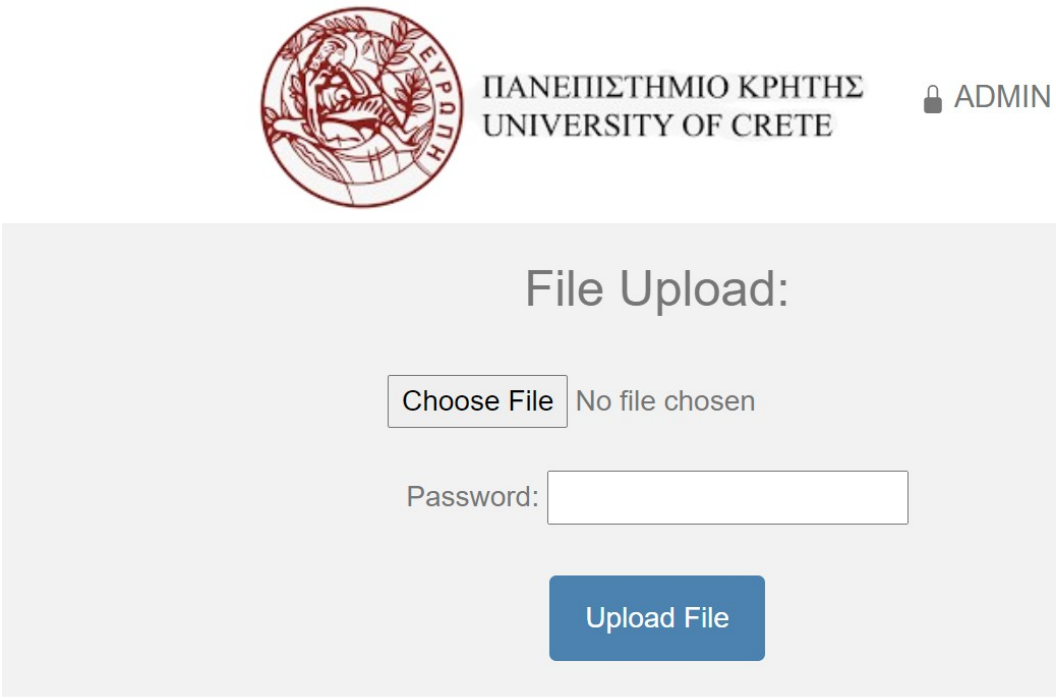


Figure 4.3.2 Upload function of Upload client application

5 Design and Modeling

5.1 REST API design/implementation

The design of REST API, implemented through Jersey, is an open source Java framework and aims to develop RESTful Web services. These support seamless data availability and remove low-level details from client-server communication.

REST is not a protocol but a set of architectural constraints and best practices. Developers of an API can implement REST in a number of ways. When a client request is made via a RESTful API, it transfers a representation of the resource to the requestor or endpoint. This status information may be provided in different formats via HTTP, such as JSON, HTML, XML or plain text. JSON is the most popular way of representation and transmission because it allows reading by both humans and machines.

Advantages of developing a REST API :

- Client - Server : Client and server are separate from each other and allows to evolve separately.
- Stateless : The RESTs APIs are stateless , meaning that calls can be made independently of each other and each call contains all the necessary data to complete successfully.
- Cache : Because a stateless API can increase generic requests, handling large volume of incoming and outgoing calls, a REST API should be designed to encourage cacheable data storage.
- Uniform interface: It is important for the customer to have a uniform interface that allows the application to evolve independently without the application's services, models, and actions being tightly coupled to the API .
- System level: REST APIs have different layers of architecture that work together to create a hierarchy that helps create a more scalable application.
- Code on Demand : Code on demand allows code or applets to be streamed through the API for use within the application.

5.2 Using Swagger

The use of Swagger framework is very important, since it is a visual documentation of REST API. It mainly concerns developers, simplifying the implementation of the Back End part and the consumption of the API from the side of the applications (Client Side).

Swagger.yaml contains all the information related to the connection of Swagger UI with REST API application, which includes:

- The name of the server where the application has been deployed.
- All Paths, of REST API application with the appropriate variables, the return format if the response has a code of 200 (successful).

For every new addition and improvement to the system, swagger.yaml must be renewed with e.g. new years in the corresponding variables to show the new options to the user.

5.3 Datasets and their import into the DB

The datasets used were taken from the Open Data Catalog of the UoC in XLSX file format from the page http://opendata.uoc.gr/el/dataset/ooiththikoe-ttahoyemoe-tt-k-2006-16 . Some sections only had data for WOMEN and TOTAL, so the MALE column was set to zero. Also some cells even though they had numerical values, represented Strings. So, they were converted from String to Number to be recognized by the DB input algorithm.

Each dataset is imported through the Upload client application. Admin uploads an XLSX file to the application directory and the algorithm reads this file and converts it into a suitable table in the DB. In the input algorithm some Greek terms are mapped to English, so the DB contains English characters so that the data can be provided in English. The identification code (password) is in the config application file . The name of each dataset is automatically converted into a table with the same name (in lower case) in the NW. To update the DB, the above function is used again to delete the table and re-do the insert operation (with the new/updated data) to the DB.

For example, if the Admin wishes to import statistics of the year 2019-2020, a new workbook named 2019-2020 should be created inside the XLXS file with the same structure as the workbooks in the past. The input algorithm checks, up to the TOTAL column (which concerns the total and is located at the end of the first table of each workbook). No further information needs to be provided.

If the Admin wants to import a new XLSX file with a different category of data, then the file needs to follow the structure of the previous (existing) files.

If the form and name of the datasets is not respected, the operation of insertion **will fail.**

Σ2. Σύνολα Προπτυχιακών Φοιτητών, ανά φύλο και έτος σπουδών

Παρουσιάζει τα σύνολα του πληθυσμού Προπτυχιακών Φοιτητών, ανά φύλο και έτος σπουδών

Έτος αναφοράς: 2018-19 (Βάσει υπολογισμού Ακαδημαϊκού έτους)

		ΤΜΗΜΑ→	Π.Τ.Δ.Ε.			Π.Τ.Π.Ε.			ΙΣΤ. Α.			ΦΙΛ			ΦΚΣ			ΦΥΣΙΚΗΣ		
	ΕΤΟΣ ΣΠΟΥΔΩΝ ↓		Α	Θ	Σύνολο	Α	Θ	Σύνολο	Α	Θ	Σύνολο	Α	Θ	Σύνολο	Α	Θ	Σύνολο	Α	Θ	Σύνολο
ΣΥΝΟΛΑ ΦΟΙΤΗΤΩΝ ΑΝΑ ΕΤΟΣ	1		26	157	183	7	178	185	71	124	195	27	133	160	26	142	168	93	67	160
	2		29	162	191	9	180	189	55	116	171	23	132	155	27	125	152	87	50	137
	3		39	147	186	14	160	174	44	104	148	22	125	147	27	105	132	85	48	133
	4		39	145	184	5	195	200	49	96	145	29	123	152	34	113	147	72	53	125
	5		16	73	89	4	134	138	45	78	123	30	102	132	25	83	108	82	55	137
	6		4	32	36	3	55	58	39	57	96	10	43	53	26	71	97	76	41	117
	7		7	13	20	6	41	47	30	50	80	29	39	68	23	61	84	53	29	82
	8		3	11	14	3	26	29	16	33	49	9	21	30	19	30	49	39	23	62
	9		2	9	11	5	20	25	18	31	49	8	35	43	5	17	22	38	16	54
	10		3	17	20	5	19	24	15	37	52	9	36	45	7	27	34	31	12	43
	11+		122	178	300	19	150	169	96	203	299	49	178	227	115	240	355	93	32	125
	Σύνολο		**290**	**944**	**1234**	**80**	**1158**	**1238**	**478**	**929**	**1407**	**245**	**967**	**1212**	**334**	**1014**	**1348**	**749**	**426**	**1175**

Figure 8.1 Indicative form of the UnderGraduateStudents_AcademicYear dataset

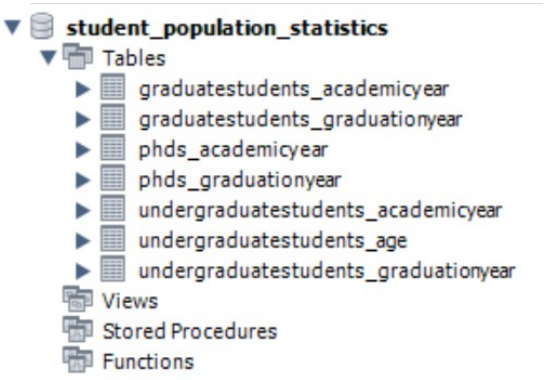
5.4 Database Design

Files from the UoC Open Data Catalog are only offered in XLSX format . The insertion function developed, loops through the entire file and places the data in the DB, in 5 columns per table. Each dataset contains, for each date, a workbook (a total of 13 years: from 2016 to 2019).

Regarding the DB, MySQL was used, with 7 tables, as many as the available datasets . Below is described the assignment of each dataset to a table and the number of records per table/dataset that were finally entered into the DB.

Dataset name	Table name to the DB	Number of records
Graduatestudents_academicyear	Graduatestudents_academicyear	3,275
Graduatestudents_graduationyear	Graduatestudents_graduationyear	2,608
Phds_academicyear	Phds_academicyear	6,003
Phds_graduationyear	Phds_graduationyear	4,845
Undergraduatestudents_academicyear	Undergraduatestudents_academicyear	8,030
Undergraduatestudents_age	Undergraduatestudents_age	12,043
Undergraduatestudents_graduationyear	Undergraduatestudents_graduationyear	6746

Example of the graduatestudents_academicyear table



YEAR	STUDENT_YEAR	FACULTY	GENDER	VALUE
2006-07	First	Department of Primary Education	Male	6
2006-07	Second	Department of Primary Education	Male	4
2006-07	Third	Department of Primary Education	Male	5
2006-07	FourthPlus	Department of Primary Education	Male	6
2006-07	Total	Department of Primary Education	Male	21
2006-07	First	Department of Primary Education	Female	11
2006-07	Second	Department of Primary Education	Female	11
2006-07	Third	Department of Primary Education	Female	8
2006-07	FourthPlus	Department of Primary Education	Female	12
2006-07	Total	Department of Primary Education	Female	42
2006-07	First	Department of Primary Education	Total	17
2006-07	Second	Department of Primary Education	Total	15
2006-07	Third	Department of Primary Education	Total	13
2006-07	FourthPlus	Department of Primary Education	Total	18
2006-07	Total	Department of Primary Education	Total	63
2006-07	First	Department of Preschool Educat...	Male	1
2006-07	Second	Department of Preschool Educat...	Male	0
2006-07	Third	Department of Preschool Educat...	Male	0
2006-07	FourthPlus	Department of Preschool Educat...	Male	2

5.5 REST Client Application

The REST Client application was implemented using JAVA programming language with the help of Jersey Framework. Each function has as its main attribute path, which corresponds to REST API .

The application includes the following core files:

- Graduate.java , for graduate student functions
- Undergraduate.java , for undergraduate functions
- PhD.java , for PHD student functions
- DB.java , to connect to the DB
- ApplicationConfig.java , for parameterization (e.g. defining the path where the application will be deployed)

Graphs are used through D3.js, a JavaScript library for data-driven document management. D3 helps bring data to life using HTML, SVG and CSS. D3's emphasis offers the full capabilities of modern browsers, combining powerful visualization components and a data-driven approach to DOM manipulation.

In this particular application, the Back-End part draws the data from the DB (for the function the user has called), creates a CSV file based on them and stores it in the directory /Data/ of the application. The CSV is then managed via JavaScript. X and Y axes are created with the appropriate data, thus presenting the final interactive graph to the user.

5.6 Libraries

The following libraries were used ,

- For the management of Microsoft Documents (Upload client application),

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-examples</artifactId>
    <version>4.1.2</version>
</dependency>
```

- To create a connection to the DB (all 3),

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.22</version>
</dependency>
```

- To provide JAVA functions (all 3),

```
<dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>javax.ws.rs-api</artifactId>
    <version>2.1.1</version>
</dependency>
```

- For the use of the Jersey Framework (Swagger & ClientREST),

```
<dependency>
    <groupId>org.glassfish.jersey.core</groupId>
    <artifactId>jersey-client</artifactId>
    <version>2.32</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.containers</groupId>
    <artifactId>jersey-container-servlet</artifactId>
    <version>2.32</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.core</groupId>
    <artifactId>jersey-common</artifactId>
    <version>2.32</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.inject</groupId>
    <artifactId>jersey-hk2</artifactId>
    <version>2.32</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.core</groupId>
    <artifactId>jersey-server</artifactId>
    <version>2.32</version>
</dependency>
```

6 System Architecture and Tools

6.1 Environment setup and deployment

The deployment of the three applications is done through a .war file placed in Tomcat's webapps folder application server.

There is a folder named Config, which contains the db.txt regarding connection parameters and NW information, as well as the password.txt (only in Upload client application) for the Admin ID.

In Swagger API, to re-deploy to another server, there must be provided a **new IP** to the swagger.yaml.

db contents.txt :

- The first line refers to the URL of the database
- The second line refers to the name of the database
- The third line refers to the PORT of the database
- The fourth line is for USERNAME
- The fifth line concerns the PASSWORD of the specific user

In ClientREST API for re-deployment to another server, the const variable server in index.html, must be altered with the **new IP** of the server.

Regarding the DB, the Admin must connect to MySQL with a user account that has write & read rights to the specific database schema , e.g. through MySQL Workbench .

In summary, modifications to the above can easily be made by changing only these files, without the need to recompile the code.

6.2 Technologies/ frameworks used

The following technologies/ frameworks were used to implement the systems

- As Application Server version of Tomcat was used Server 9.0
- The MySQL version was used as Database Workbench 8.0
- Jersey Framework
- The Swagger Framework was used which allows describing the structure of an API, and each API has the ability to describe its own structure.

7 System Test

The Swagger system is quite responsive and fast, as far as developer functions are concerned. For Admins , the conversion and data entry of a . xlsx to table in DB, timed from 45 seconds (3000+ records) to 2 minutes (12,000+ records).

As for REST Client application, to retrieve the data visualized in a table, times vary depending on the dataset. The smallest dataset with about 3000+ records is calculated in 1 second, while the most complex one with about 12,000+ records is calculated in 4 seconds. The creation and display of graphs has no noticeable delay for the user.

The application was tested and its functionality checked on a server with the following specifications:

- Fujitsu ESPRIME 920 with Windows Server 2019
- Processor: Intel i5-4590 3.30GHz
- RAM: 16GB
- SSD: 256GB

Other information:

- IP:PORT -> http://147.52.17.140:8080/
- Directory where the applications were deployed -> C:\Tomcat9.0\webapps
- Directory Swagger UI -> C:\Tomcat9.0\webapps\Swagger
- Directory ClientREST UI -> C:\Tomcat9.0\webapps\ClientREST
- Directory Upload UI -> C:\Tomcat9.0\webapps\Upload

The applications are available for use at the following URLs :

- http://147.52.17.140:8080/Swagger/
- http://147.52.17.140:8080/ClientREST/
- http://147.52.17.140:8080/Upload/

8 Epilogue

Web applications were implemented based on the independence of functions, as well as fast response. As new data is introduced over time, the three applications will be automatically updated.

In REST API/Swagger, with the existence of new Datasets (beyond the current 7) new functions can be created that will return an even greater variety and different type of information for the students of the University of Crete.

Regarding Datasets, great efforts have been made to maintain the current format to facilitate their creators now and in the future. The only requirement is to maintain the correct spacing between the columns.

With the data available so far, a wide range of information is covered, since the developer can exploit to a large extent the anonymous statistical data of the students of the University of Crete.

As for REST Client application, new types of charts and tables could be implemented, offering the visitor a richer user experience. For example, provide graphs by department or by gender and year or by gender and department. The combinations are endless.

Sources

1. http://opendata.uoc.gr
2. https://eclipse-ee4j.github.io/jersey/
3. https://swagger.io/
4. https://ckan.org/
5. http://tomcat.apache.org/tomcat-9.0-doc/
6. https://www.mysql.com/products/workbench/
7. https://poi.apache.org/
8. https://d3js.org/