Eren Badur - 31289
Kemal Yılmaz - 31097
Korhan Erdoğdu - 30838
Yusuf Sinan Özmen - 31067

Rate My Professor App

Rate My Professor App, is a basic designed app in order to help college students evaluate and review their professors. The platform allows students to share their opinions and experiences with specific instructors, providing valuable insights for other students who may be considering taking a course with those professors.

We started our backend by first scraping the data that we'll use in the project from https://bannerweb.sabanciuniv.edu/ we scraped the name of instructors, the classes they teach, the codes of the classes and the names of the classes. The scraping and parsing of the data is done with the help of a python script. The collected data is stored in a MongoDB Atlas database in the format used by our java entities

We have 4 main entities: classInfo, teacherInfo, reviews, students.

Our database consists of four collections: *class_info, teacher_info, reviews, and students.*

***classInfo* consists of the following elements: classCode,className and a list of teacherInfo object ids that store the information of which teachers are teaching this course.**

***teacherInfo* consist of the following elements: primaryInstructor which is the name of the teacher, a list of object ids of the review's the teacher got, a list of object ids of the classes they teach, number of ratings they got and their teacher rating given by students.**

***students* consist of the following elements: name and major of the student, the list of class codes of the classes they take and the object ids of the review's they made on teachers.**

***reviews* consist of the following elements: object id of the student that made the review, the object id of the teacher that the review is made for, the rating given by the student and the comment of the student about the teacher.**

Repository files for each entity are used to connect the entities to their respective collections in the database.

Service files adds the functionalities of each entity, while classInfo,teacherInfo and students include simple create,find,delete and update operations. The reviewsService is more complex as it manages student and teacherInfo objects as it updates the ratings of the professors after each added,deleted and updated review.

Controllers are used to handle the HTTP request, mapping and they also invoke the functionalities of the service classes.

1. Class Endpoints

1- Get list of Classes

Description: List of Classes inside in our database. Id, Class Code, Class Name and Instructors' Id's are returned.
Address: http://localhost:8080/api/classes
Method: GET
Returns JSON Array:


```
:[
   {
      "id": "657511e98f1e6a5a3451ff94",
      "classCode": "AL 102",
      "className": "Academic Literacies",
      "instructorIds": [
         "657511138f1e6a5a3451b9e5",
         "657511138f1e6a5a3451b9e6",
         "657511138f1e6a5a3451b9e8",
         "657511138f1e6a5a3451b9e9",
         "657511138f1e6a5a3451b9ea",
         "657511138f1e6a5a3451b9eb",
         "657511138f1e6a5a3451b9ec"
      ]
   },



   {
      "id": "657511e98f1e6a5a3451ff97",
      "classCode": "ACC 201",
      "className": "Introduction to Financial Accounting and Reporting",
      "instructorIds": [
         "657511138f1e6a5a3451b9ed",
         "657511138f1e6a5a3451b9ee"
      ]
   },
```

```
{
    "id": "657511e98f1e6a5a3451ff98",
    "classCode": "ACC 201R",
    "className": "Introduction to Financial Accounting and Reporting-Recitation",
    "instructorIds": [
        "657511138f1e6a5a3451b9ed",
        "657511138f1e6a5a3451b9ee"
    ]
},


{
    "id": "657511e98f1e6a5a3451ff99",
    "classCode": "ACC 301",
    "className": "Managerial Accounting",
    "instructorIds": [
        "657511138f1e6a5a3451b9ee"
    ]
},
```


(The output contains a large amount of JSON objects; we are displaying four of them here.)

2- Create New Class

Description: Creates a new class object and adds it to the database.
Address: http://localhost:8080/api/classes
Method: POST
Accepts: Application/JSON

```
{
   "classCode": "TST 101",
   "className": "Database Testing",
   "instructorIds": [
      "657511138f1e6a5a3451b9ee"
   ]


}
```

Returns JSON Object

```
{
   "id": "6575bb9583903172bacf8e0b",
   "classCode": "TST 101",
   "className": "Database Testing",
   "instructorIds": [
      "657511138f1e6a5a3451b9ee"
   ]
}
```

3- Find Class by Id

Description: Finds Class by Id. Id, Class Code, Class Name, Instructor Id's returned.
Address: http://localhost:8080/api/classes/{id}
Method: GET
Returns JSON Object:

```
{
   "id": "6575bb9583903172bacf8e0b",
   "classCode": "TST 101",
```

```
    "className": "Database Testing",
    "instructorIds": [
        "657511138f1e6a5a3451b9ee"
    ]
}
```


4- Find class by ClassCode:

Address: http://localhost:8080/api/classes/findByCode/{ClassCode}
Test: http://localhost:8080/api/classes/findByCode/CS 310
Method: GET
Returns JSON Object:


```
{
    "id": "657511e98f1e6a5a3451ffe4",
    "classCode": "CS 310",
    "className": "Mobile Application Development",
    "instructorIds": [
        "657511138f1e6a5a3451ba1b"
    ]
}
```


5- Update Class Info

Description: Updates Class Information. Class Code, Class Name and Instructor Ids' are being put.
Address: http://localhost:8080/api/classes/6575bb9583903172bacf8e0b
Method: PUT
Accepts JSON:


```
{
    "classCode": "TST 102",
```

```
    "className": "Database Testing 2",
    "instructorIds": [
        "657511138f1e6a5a3451b9ee"
    ]
}
```

Returns JSON object:


```
{
    "id": "6575bb9583903172bacf8e0b",
    "classCode": "TST 102",
    "className": "Database Testing 2",
    "instructorIds": [
        "657511138f1e6a5a3451b9ee"
    ]
}
```


6- Delete Class Info

Description: Deletes Class Information. Adding only the Class Id.
Address: http://localhost:8080/api/classes/6575bb9583903172bacf8e0b
Method: DELETE

7- Adding Professor to Class

Description: Adds Professor Id to the Class.
Address: http://localhost:8080/api/classes/{classId}/addProfessor/{professorId}
Test:
http://localhost:8080/api/classes/657511e98f1e6a5a3451ff97/addProfessor/657511138f1e6a5a3451b9ec
Method: PUT

8- Removing Professor From Class

Description: Removes Professor Id from the Classes.
Address: http://localhost:8080/api/classes/{classId}/removeProfessor/{professorId}
Test:
http://localhost:8080/api/classes/657511e98f1e6a5a3451ff97/removeProfessor/657511138f1e6a5a3451b9ec
Method: PUT

2. Teacher Endpoints

1- Get list of Teachers

Description: List of Teachers inside in our database. Id, Class Code, Class Name and Instructors' Id's are returned.
Address: http://localhost:8080/api/teachers
Method: GET
Returns JSON Array:

2- Create New Teacher

Description: Creates new teacher objects and adds it to the database.
Address: http://localhost:8080/api/teachers
Method: POST
Accepts: Application/JSON


```
{
   "primaryInstructor": "Kemal Yılmaz",
   "reviewIds": [],
   "averageRating": 0.0,
   "numOfRatings": 0,
   "classCodes": [
      "TST 101"
```

```
    ]
}
```


Returns JSON Object


```
{
    "id": "6575c27c83903172bacf8e0c",
    "primaryInstructor": "Kemal Yılmaz",
    "reviewIds": [],
    "averageRating": 0.0,
    "numOfRatings": 0,
    "classCodes": [
        "TST 101"
    ]
}
```


3- Find Teacher by Id

Description: Finds Teacher by Id. Id, Teacher Name, Instructor Id's returned.
Address: http://localhost:8080/api/teachers/{id}
Test: http://localhost:8080/api/teachers/6575c27c83903172bacf8e0c
Method: GET
Returns JSON Object:


```
{
    "id": "6575c27c83903172bacf8e0c",
    "primaryInstructor": "Kemal Yılmaz",
    "reviewIds": [],
    "averageRating": 0.0,
    "numOfRatings": 0,
    "classCodes": [
        "TST 101"
    ]
}
```


4- Find teacher by Name

Description: Finds teacher object by name
Address: http://localhost:8080/api/teachers/findByInstructor/{instructorName}
Test: http://localhost:8080/api/teachers/findByInstructor/Altuğ Tanaltay

Method:GET
Returns JSON object:


```
{
   "id": "657511138f1e6a5a3451ba1b",
   "primaryInstructor": "Altuğ Tanaltay",
   "reviewIds": [],
   "averageRating": 0.0,
   "numOfRatings": 0,
   "classCodes": [
      "CS 310",
      "ENS 491",
      "IT 541"
   ]
}
```


5- Update Teacher Info

Description: Updates Teacher Information. Class Code, Class Name and Instructor Ids' are being put.
Address: http://localhost:8080/api/teachers/{id}
Test: http://localhost:8080/api/teachers/6575c27c83903172bacf8e0c
Method: PUT
Accepts JSON:


```
{
   "primaryInstructor": "Yusuf Özmen",
   "reviewIds": [],
   "averageRating": 4.8,
   "numOfRatings": 5,
   "classCodes": [
      "TST 102"
   ]
}
```


Returns JSON object:


```
{
   "id": "6575c27c83903172bacf8e0c",
   "primaryInstructor": "Yusuf Özmen",
```

```
    "reviewIds": [],
    "averageRating": 4.8,
    "numOfRatings": 5,
    "classCodes": [
        "TST 102"
    ]
}
```


6- Delete Teacher Info

Description: Deletes Teacher Information. Adding only the Teacher Id.
Address: http://localhost:8080/api/teachers/{id}
Test: http://localhost:8080/api/teachers/6575c27c83903172bacf8e0c
Method: DELETE

7- Adding Class to Teacher

Description: Adds Class code to the Teacher.
Address: http://localhost:8080/api/teachers/{teacherId}/addClass/{classCode}
Test: http://localhost:8080/api/teachers/657511138f1e6a5a3451ba1b/addClass/ECON 201
Method: PUT

8- Removing Professor From Class

Description: Removes Professor Id from the Class.
Address: http://localhost:8080/api/teachers/{teacherId}/removeClass/{classCode}
Test: http://localhost:8080/api/teachers/657511138f1e6a5a3451ba1b/addClass/ECON 201
Method: PUT

3. Student Endpoints

1- Create a Student

Description: Creates a new student object and adds it to the database.
Address:http://localhost:8080/api/students
Test:http://localhost:8080/api/students
Method: POST
Accepts: Application/JSON

```
{
   "name": "Eren Badur",
   "major": "Computer Science",
   "reviewIds": [],
   "classes" : ["CS 310"]
}
```

Returns JSON object

```
{
   "id": "6575d1582980a559358b1189",
   "name": "Eren Badur",
   "major": "Computer Science",
   "classes": [
      "CS 310"
   ],
   "reviewIds": []
}
```

2- Find Student by id

Description: Finds Student by id. Id, Student Name, Student major and classes and the ids of the reviews are returned.
Address:http://localhost:8080/api/students/{id}
Test:http://localhost:8080/api/students/6575d1582980a559358b1189
Method: GET
Returns JSON Object

```
{
   "id": "6575d1582980a559358b1189",
   "name": "Eren Badur",
   "major": "Computer Science",
```

```
    "classes": [
        "CS 310"
    ],
    "reviewIds": []
}
```


3- Find Student by name

Description: Finds Student by name. Id, Student Name, major and classes and the ids of the reviews are returned.
Address: :http://localhost:8080/api/students/findByName/{name}
Test:http://localhost:8080/api/students/findByName/Eren Badur
Method: GET

Returns: JSON object


```
{
        "id": "6575d1582980a559358b1189",
        "name": "Eren Badur",
        "major": "Computer Science",
        "classes": [
            "CS 310"
        ],
        "reviewIds": []
    }
```


4- Getting all student names

Description: Gets Students inside the database. Id, Student Code, Student Name, Student Major, List of Classes and Review id's are returned.
Address: http://localhost:8080/api/students
Test: http://localhost:8080/api/students
Method: GET
Returns JSON Array:


```
[
    {
        "id": "6575d1582980a559358b1189",
        "name": "Eren Badur",
        "major": "Computer Science",
        "classes": [
```

```
            "CS 310"
        ],
        "reviewIds": []
    }
]
```


5- Updating student

Description: Updates Student Information. Student Code, Class Name and Instructor Ids' are being put.
Address:http://localhost:8080/api/students/{id}
Test: http://localhost:8080/api/students/6575d1582980a559358b1189
Method: PUT
Accepts: Application/JSON


```
{
    "name": "Eren Badur",
    "major": "Industrial Engineering",
    "reviewIds": [],
    "classes" : ["ENS 208"]
}
```


Returns: JSON object


```
{
    "id": "6575d1582980a559358b1189",
    "name": "Eren Badur",
    "major": "Industrial Engineering",
    "classes": [
        "ENS 208"
    ],
    "reviewIds": []
}
```


6- Deleting student
Description: Deletes Teacher object.
Address: http://localhost:8080/api/students/{id}
Test:http://localhost:8080/api/students/6575d1582980a559358b1189
Method: DELETE

4. Review Endpoints

1- Create a Review

Description: Creates a new review object and adds it to the database.
Address: http://localhost:8080/api/reviews
Test: http://localhost:8080/api/reviews
Method: POST

input:


```
{
   "studentId": "6575d1582980a559358b1189",
   "teacherId": "657511138f1e6a5a3451ba1b",
   "rating": 5.0,
   "comment": "What an amazing teacher!!"
}
```


2- Find Review by Id

Description: Finds review by id. Id, Student id, teacher id, comment, and rating are returned.
Address: http://localhost:8080/api/reviews/{id}
Test: http://localhost:8080/api/reviews/6575d64f2980a559358b118d
Method: GET


```
{
   "id": "6575d64f2980a559358b118d",
   "studentId": "6575d1582980a559358b1189",
   "teacherId": "657511138f1e6a5a3451ba1b",
   "comment": "What an amazing teacher!!",
   "rating": 5.0
}
```


3- Find Reviews for a Teacher

Description: Finds reviews for teachers. Id, Student Id, teacher Id, Comment and Rating is included.
Address:http://localhost:8080/api/reviews/teacher/{teacherId}
Test: http://localhost:8080/api/reviews/teacher/657511138f1e6a5a3451ba1b
Method: GET
Returns JSON array:


```
[
   {
      "id": "6575d64f2980a559358b118d",
      "studentId": "6575d1582980a559358b1189",
      "teacherId": "657511138f1e6a5a3451ba1b",
      "comment": "What an amazing teacher!!",
      "rating": 5.0
   }
]
```


4- Find Reviews Done by a Student

Description: Finds Reviews done by a Student. Id, Student Id, Teacher Id, Comment and Rating is being returned.
Address: http://localhost:8080/api/reviews/student/{studentId}
Test: http://localhost:8080/api/reviews/student/6575d1582980a559358b1189
Method: GET

Returns JSON array:


```
[
   {
      "id": "6575d64f2980a559358b118d",
      "studentId": "6575d1582980a559358b1189",
      "teacherId": "657511138f1e6a5a3451ba1b",
      "comment": "What an amazing teacher!!",
      "rating": 5.0
   }
]
```

5- Get All Reviews

Description: Gets all the Reviews. Id, Student Id, Teacher Id, Comment and Rating is gotten.
Address: http://localhost:8080/api/reviews
Test: http://localhost:8080/api/reviews
Method: GET


```
[
{
        "id": "6575d64f2980a559358b118d",
        "studentId": "6575d1582980a559358b1189",
        "teacherId": "657511138f1e6a5a3451ba1b",
        "comment": "What an amazing teacher!!",
        "rating": 5.0
    }
]
```


6- Update Review

Description: Updates the rating and the comment of the review with the given id. The teacher's rating is also updated in accordance with the changed rating.
Method: PUT
Address:
http://localhost:8080/api/reviews/{id}?newRating={newRating}&newComment={newComment}

Test:http://localhost:8080/api/reviews/6575eea2ab59a04fc7f73cd6?newRating=4.5&newComment="What a OK teacher!!"

7- Delete Review

Description: Deletes the review with the given id. Also deletes the review ID from student's and teacher's review list and updates the teacher's rating.

Method: DELETE
Address: http://localhost:8080/api/reviews/{id}
Test: http://localhost:8080/api/reviews/6575eea2ab59a04fc7f73cd6