Table of Contents

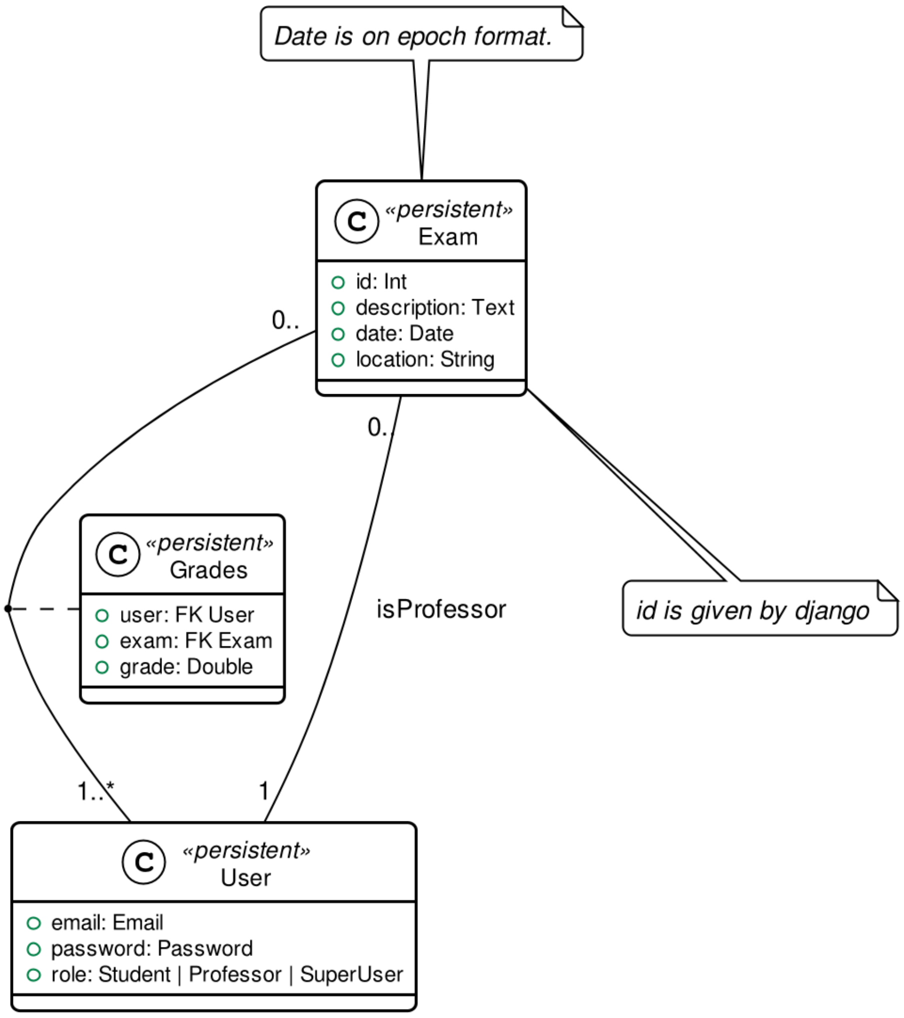


Figure 1: img

Coses a preguntar

- Sobre time i location (Exam)
- Datasource serveix per extraure el driver de connexió del servidor, i aixi poder canviar de base de dades. En el nostre cas ja ho fa SQLAlchemy (ORM), que extrau aquest tipus de connexio de SQLs. L'abstracció de quina BD utilitzar ho farem desde l'`.env`.
- Session es necessari? RMI qualsevol estudiant es pot connectar. Llavors, el Professor ha de dir els alumnes que es poden connectar? O mentre sigui un alumne es pot connectar al examen?

Notes de desenvolupament

- Canviar delete dels tests d'exam, només s'ha de poder esborrar si no té grades.

Arquitectura

Bàsics

- Donar un identificador a l'exam
- Guardar la descripció, la date/time i location —
- Poder borrar l'examen si no te cap nota
- Modificar la descripció de l'examen
- S'ha de poder buscar el contingut de l'examen amb l'id o la descripció parcial o sencera de l'examen.
- s'ha de poder descargar la informació de l'examen per identificador o per llistant tots els examens

Advanced

- S'ha de poder posar notes a un examen.
- S'ha de poder descarregar les notes d'un estudiant.
- S'ha de poder guardar i extreure tota la informació dels examens / teus examens.
- – de la merda que utilitza
- S'ha de poder gestionar l' accés de l'estudiant per id. (?)

Integració

- RMI ha de crear l'examen al ws.
- Els estudiants han de validar l'id abans de començar l'examen. Els hi donarà detalls de la connexió amb el servidor.
- S'ha de poder guardar les notes desde el WS.

`exam=exam_id`

Method

URL

What

get

exam/

List d’exams

get

exam/{exam}/

Detall de Exam (tot)

get

exam/search?description={text}/

Buscar descripció parcial.

post

exam/

Crea exam. pk no s’ha de donar.

put

exam/{exam}/

Modificar camps d’Exam (tots)

patch

exam/{exam}/

Partial update.

delete

exam/{exam}/

Deletes if professor and no grades

post

grades/

Penjar nota d’un examen.

get

grades/{user}/user/

List totes les notes d’un estudiant.

get

grades/

List all grades.

get

grades/{gradeid}

Detail a grade.

put

grades/{gradeid}

Updates a grade.

patch

grades/{gradeid}

Partially updates a grade.

delete

grades/{gradeid}

Deletes a grade.

post

auth/login/

Logins

get

auth/logout/

Logouts

post

auth/logout/

Logout

post

auth/password/change/

Password change.

post

auth/password/reset/

Password reset by email confirmation. Needs Email configuration

post

auth/password/reset/confirm/

Password Confirmation

post

auth/registration/

Register a new user.

post

auth/registration/verify-email

Verifies email. Needs Email configuration

get

auth/user/

Reads User. Needs authentication

put

auth/user/

Updates User

patch

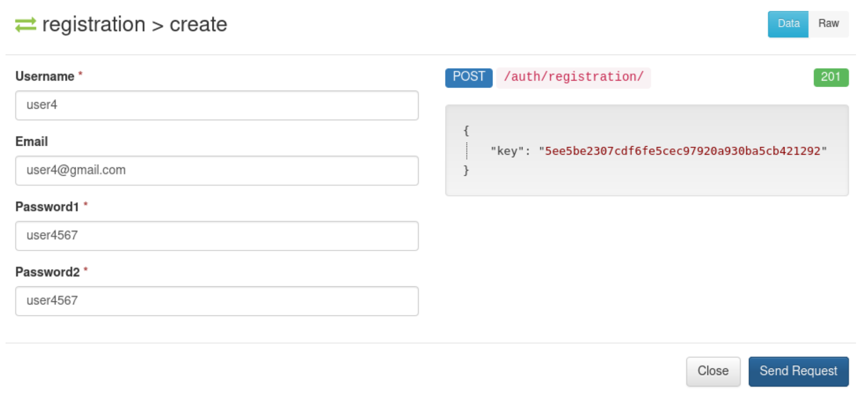
auth/user/

Partial update.

Screenshots

The screenshots are for the most important cases, there are endpoints that has been omitted, like user password change.

Authentication

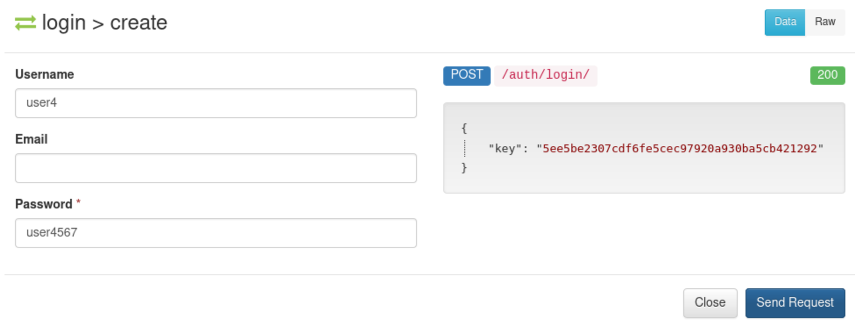


Username *	user4
Email	user4@gmail.com
Password1 *	user4567
Password2 *	user4567

POST /auth/registration/ 201

```
{
    "key": "5ee5be2307cdf6fe5cec97920a930ba5cb421292"
}
```

- Register


login > create

Data Raw

Username
user4

Email

Password *
user4567

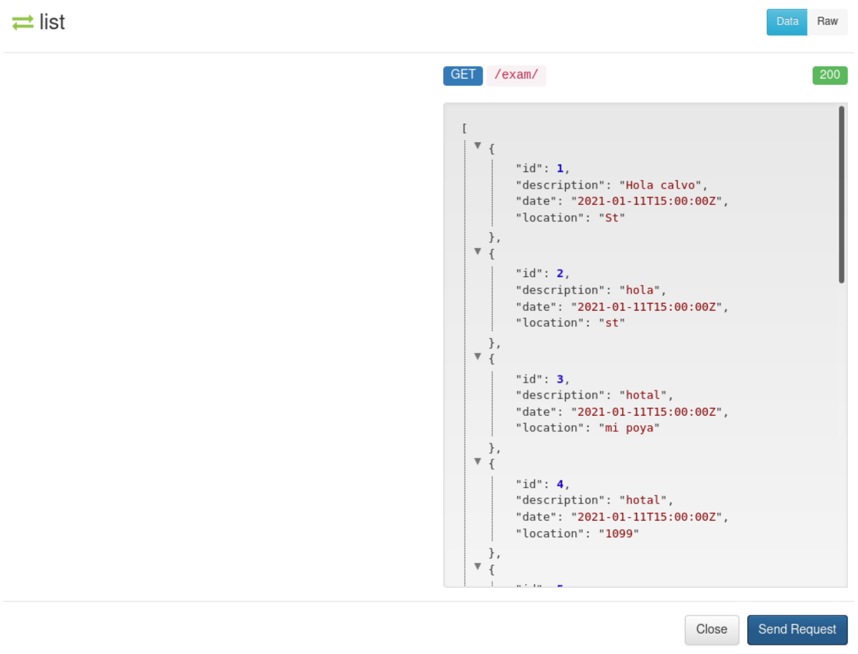
POST `/auth/login/` 200

```
{
    "key": "5ee5be2307cdf6fe5cec97920a930ba5cb421292"
}
```

Close Send Request

- Login

Exam


list

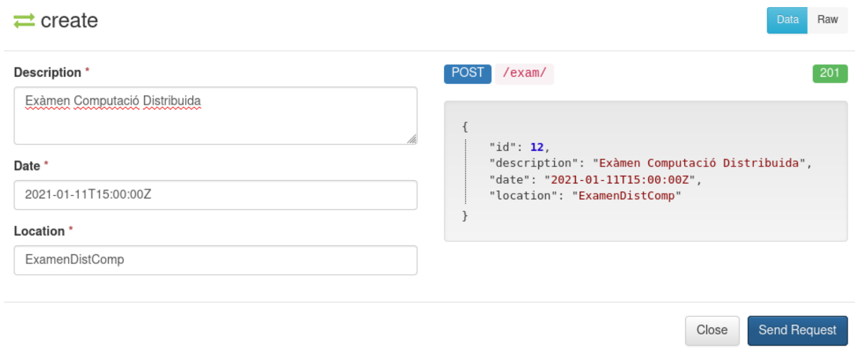
Data Raw

GET `/exam/` 200

```
[
  ▼ {
        "id": 1,
        "description": "Hola calvo",
        "date": "2021-01-11T15:00:00Z",
        "location": "St"
    },
  ▼ {
        "id": 2,
        "description": "hola",
        "date": "2021-01-11T15:00:00Z",
        "location": "st"
    },
  ▼ {
        "id": 3,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "mi poya"
    },
  ▼ {
        "id": 4,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "1099"
    },
  ▼ {
```

Close Send Request

- List exams


create

Data Raw

Description *
Exàmen Computació Distribuida

Date *
2021-01-11T15:00:00Z

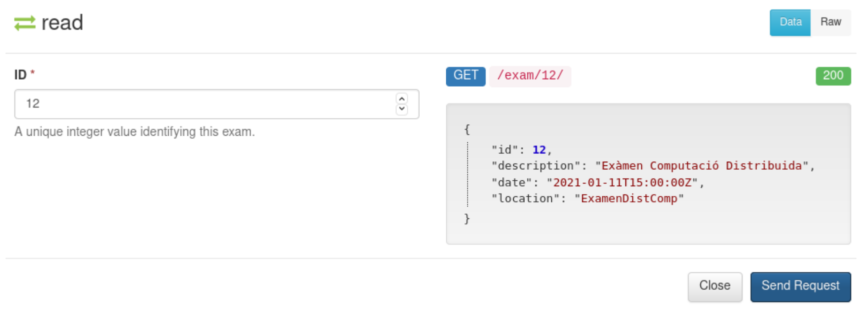
Location *
ExamenDistComp

POST `/exam/` 201

```
{
    "id": 12,
    "description": "Exàmen Computació Distribuida",
    "date": "2021-01-11T15:00:00Z",
    "location": "ExamenDistComp"
}
```

Close Send Request

- Create exam


⇄ read

Data Raw

ID *

12

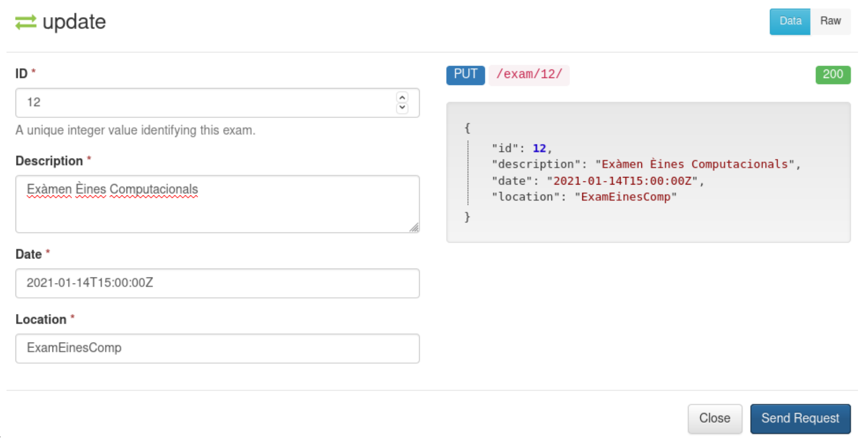
A unique integer value identifying this exam.

GET /exam/12/ 200

```
{
    "id": 12,
    "description": "Exàmen Computació Distribuida",
    "date": "2021-01-11T15:00:00Z",
    "location": "ExamenDistComp"
}
```

Close Send Request

- Read exam


⇄ update

Data Raw

ID *

12

A unique integer value identifying this exam.

Description *

Exàmen Èines Computacionals

Date *

2021-01-14T15:00:00Z

Location *

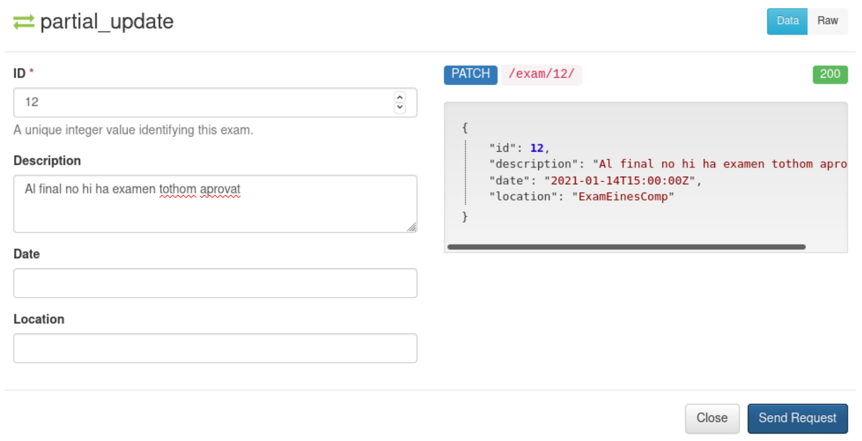
ExamEinesComp

PUT /exam/12/ 200

```
{
    "id": 12,
    "description": "Exàmen Èines Computacionals",
    "date": "2021-01-14T15:00:00Z",
    "location": "ExamEinesComp"
}
```

Close Send Request

- Update exam


⇄ partial_update

Data Raw

ID *

12

A unique integer value identifying this exam.

Description

Al final no hi ha examen tothom aprovat

Date

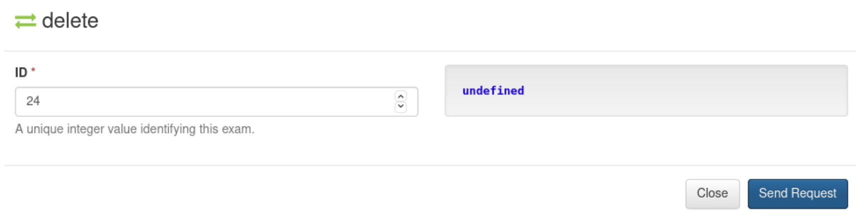
Location

PATCH /exam/12/ 200

```
{
    "id": 12,
    "description": "Al final no hi ha examen tothom apro
    "date": "2021-01-14T15:00:00Z",
    "location": "ExamEinesComp"
}
```

Close Send Request

- Patch exam


⇄ delete

ID *

24

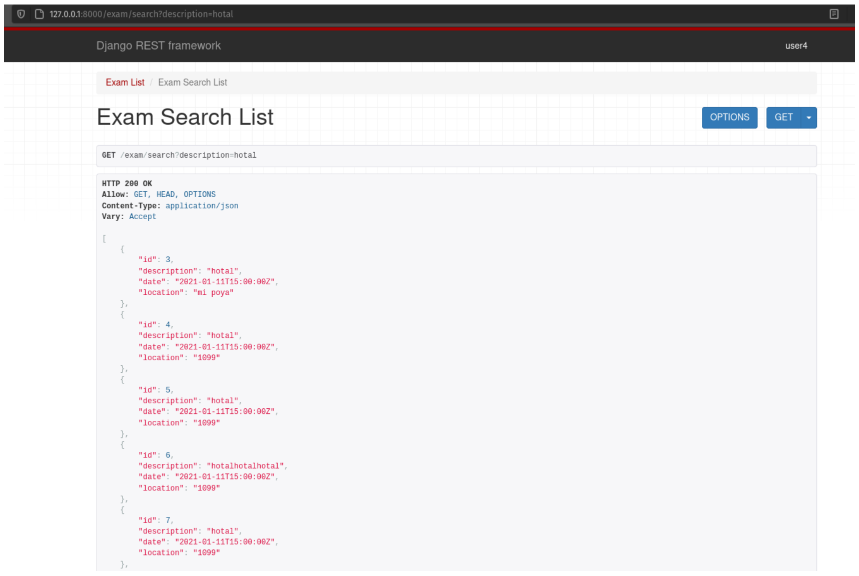
A unique integer value identifying this exam.

```
undefined
```

Close Send Request

- Delete exam

- Search exam



127.0.0.1:8000/exam/search?description=hotal

Django REST framework

user4

Exam List / Exam Search List

Exam Search List

OPTIONS GET

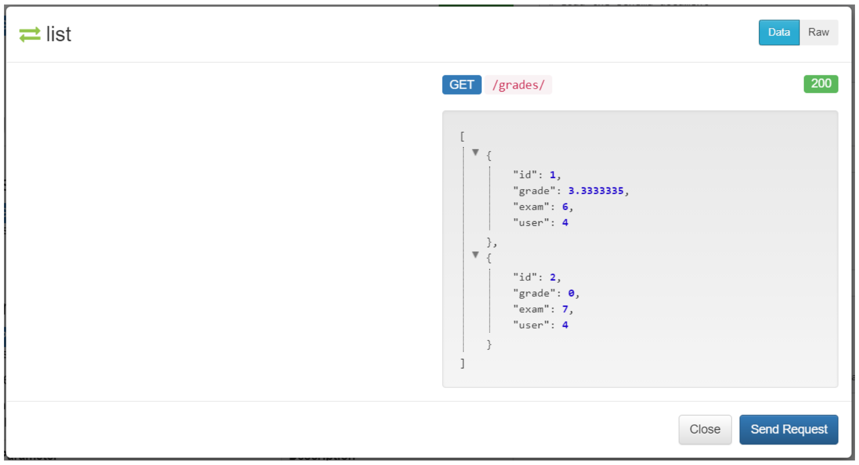
```
GET /exam/search?description=hotal
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 3,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "mi poya"
    },
    {
        "id": 4,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "1099"
    },
    {
        "id": 5,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "1099"
    },
    {
        "id": 6,
        "description": "hotalhotalhotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "1099"
    },
    {
        "id": 7,
        "description": "hotal",
        "date": "2021-01-11T15:00:00Z",
        "location": "1099"
    },
```

Grades

- List grades



list

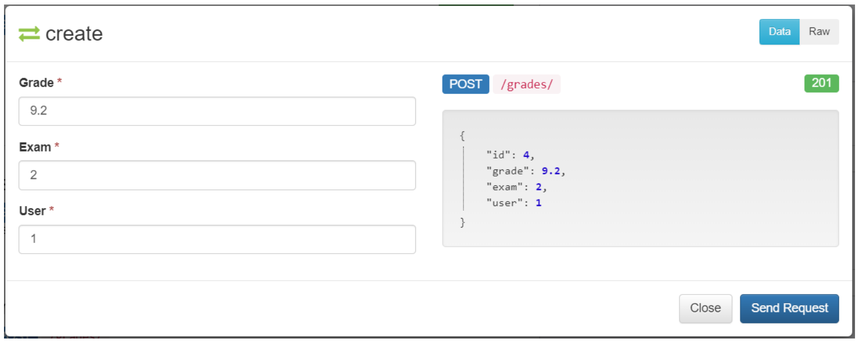
Data Raw

GET /grades/ 200

```
[
  {
      "id": 1,
      "grade": 3.3333335,
      "exam": 6,
      "user": 4
  },
  {
      "id": 2,
      "grade": 0,
      "exam": 7,
      "user": 4
  }
]
```

Close Send Request

- Create grade



create

Data Raw

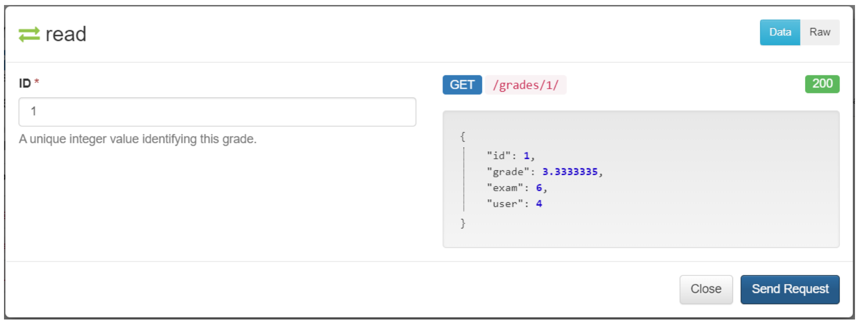
Grade *	9.2
Exam *	2
User *	1

POST /grades/ 201

```
{
    "id": 4,
    "grade": 9.2,
    "exam": 2,
    "user": 1
}
```

Close Send Request

- Read grade



ID *

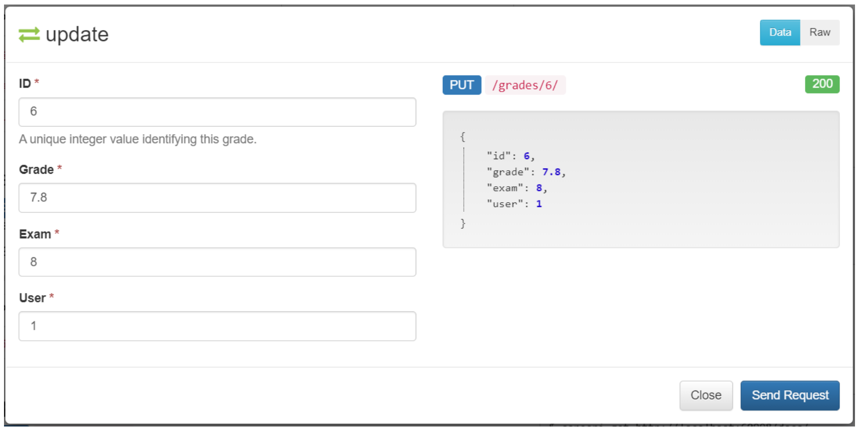
1

A unique integer value identifying this grade.

GET /grades/1/ 200

```
{
    "id": 1,
    "grade": 3.3333335,
    "exam": 6,
    "user": 4
}
```

- Update grade



ID *

6

A unique integer value identifying this grade.

Grade *

7.8

Exam *

8

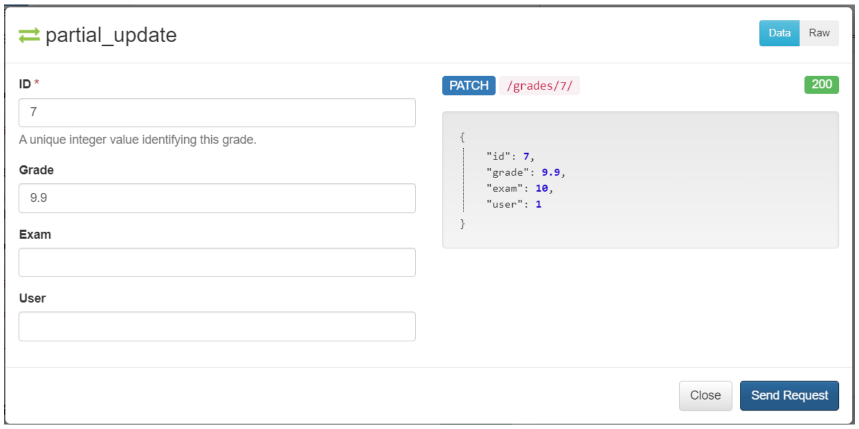
User *

1

PUT /grades/6/ 200

```
{
    "id": 6,
    "grade": 7.8,
    "exam": 8,
    "user": 1
}
```

- Patch grade



ID *

7

A unique integer value identifying this grade.

Grade

9.9

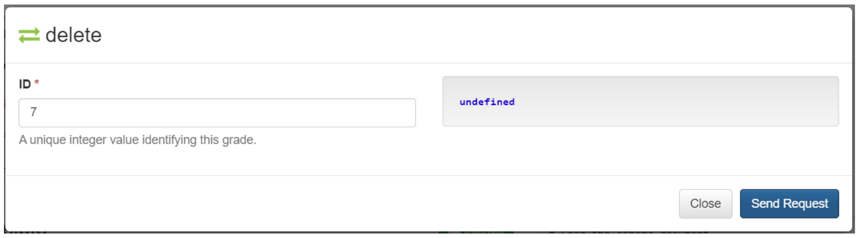
Exam

User

PATCH /grades/7/ 200

```
{
    "id": 7,
    "grade": 9.9,
    "exam": 10,
    "user": 1
}
```

- Delete grade

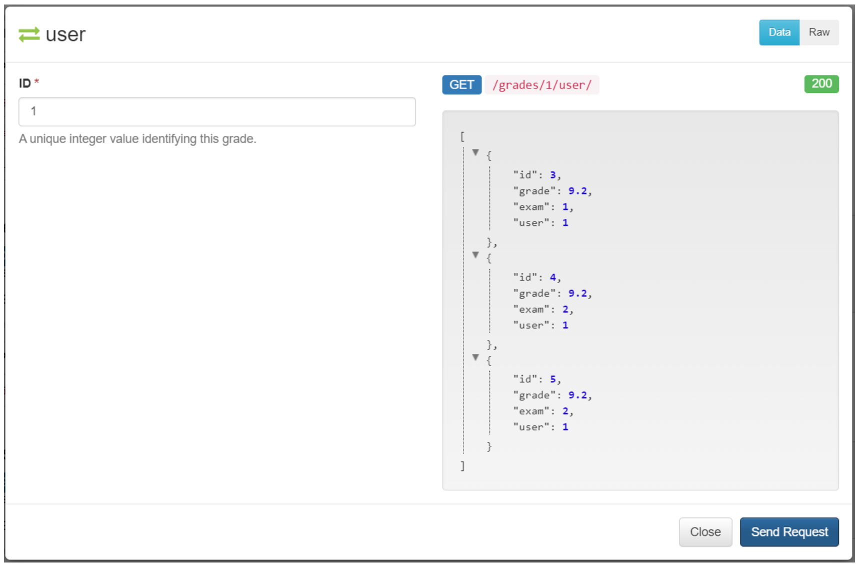


ID *

7

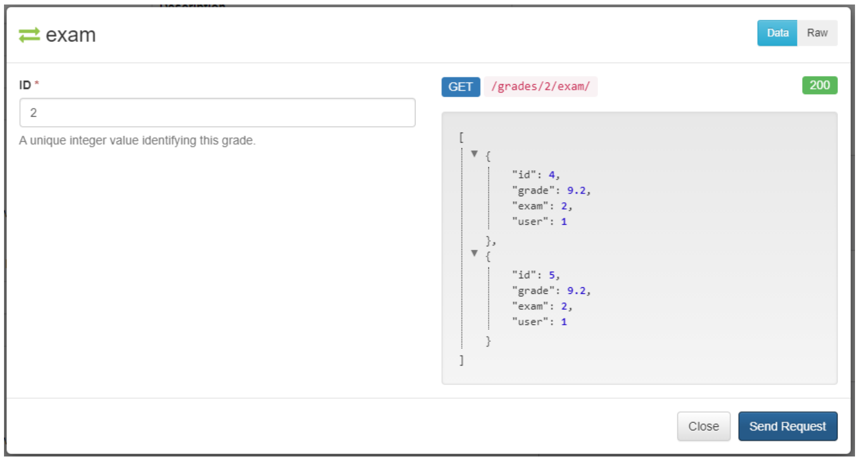
A unique integer value identifying this grade.

```
undefined
```

- Search user grades 

ID: 1

```
[
    {
        "id": 3,
        "grade": 9.2,
        "exam": 1,
        "user": 1
    },
    {
        "id": 4,
        "grade": 9.2,
        "exam": 2,
        "user": 1
    },
    {
        "id": 5,
        "grade": 9.2,
        "exam": 2,
        "user": 1
    }
]
```

- Search exam grades 

ID: 2

```
[
    {
        "id": 4,
        "grade": 9.2,
        "exam": 2,
        "user": 1
    },
    {
        "id": 5,
        "grade": 9.2,
        "exam": 2,
        "user": 1
    }
]
```

How To

Solution justification

Web Service

Technologies

- Django: We have chosen this technology because our familiarity with it and its ease to work with data models and ORM.
- Django rest framework: this framework is a powerfull and easy-to-use tool for building web REST API's, it includes mechanisms for searialization

and authentication, which we found necessary.
- SQLite: it is the Django default database (a postgres database is also configured using docker)
- Docker: It facilitates the configuration and portabiltiy of the project.

Decisions

- **Authentication**: we developed a simple autenticathon in which users once registered and logged are provided with a token that they will need to make specific api calls. There are custom permisions to prevent forbidden actions, like a student deleting an exam, or modifiying a grade. We used django_rest_auth, which provides endpoints for registration, authentication, password resset, retrieve and update user details, etc.
- **Get user**:

RMI modifications

- **HTTP**: We have made two adapter classes in order to encapsulate the http requests made to the web service by the client and the server. To make the request we have used OkHttp3, we were restricted to use a library from before java 8 because of RMI deprecation. We were unable to mock and test the api calls because OkHttp3 Request and Response object does not implement equals, and are final.
- **Client flow changes**: Now the client has to be identified in order to enter the exam session, so the first step is to ask for a correct user and password. Once authenticated correctly the user is given 3 options:
  - **search <keywords>** : searches exams by its description and outputs the information of the matched exams.
  - **list** : lists and outputs all the exams and its information.
  - **choose <id_exam>** : chose the desired exam in order to connect to its session. Once an exam is chosen, the flow works as before.
- **Server flow changes**: As happens with the client, the professor has to be identified in order to create an exam session, so the first step is to ask for a correct user and password. Once authenticated correctly it will be asked to introduce the following parametters in order to create the exam:
  - **description**: the desription of the exam.
  - **date**: the date of the exam, it needs a specific date format, as 2021-01-11T14:00:00Z.
  - **location**: the location of the exam (string). We decided that the location will be the bind key of the remote object that references that exact exam session. Once the last parammeter is filled, the exam will be created in the web service, as well as the session in which the students can connect to perform the exam. When the professor finnishes the exam all the grades are updated to the web service.

Hours dedicated

It is difficult to say, but we estimate an approximate of 90 hours. We are a group of three students, and we worked in this project for 6 days, 5 hours each day.