

Wimuuv

Documentação REST

IADÉ - Faculdade de Design, Tecnologia e Comunicação Engenharia Informática — 3.º semestre
Luís Silva (20200606); Ricardo Abrantes (20201033)

Programação Orientada por Objetos (Miguel Bugalho);

URL da API geral: <http://wimuuv.herokuapp.com>

Recurso Student

Ir buscar todos os alunos

Devolve todos os alunos na base dados cada um com informação do curso

Url: `/api/student (get)`

Resultado:

```
[ {  
  
  "id": 1,  
  "stu_email": "example@iol.xyz",  
  "stu_crse_id": 1,  
  "stu_photo_id": 3,  
  "name": "Diogo Panamera",  
  "bdate": "2001-06-12",  
  "currentAge": 20,  
  "gender": "M"  
},  
{  
  
  "id": 2,  
  "stu_email": "example2@iol.xyz",  
  "stu_crse_id": 1,  
  "stu_photo_id": 4,  
  "name": "Laura Macho",  
  "bdate": "2000-05-07",  
  "currentAge": 21,  
  "gender": "F"  
}  
... ]
```

Ir buscar um aluno dado um id

Devolve o alunocorrespondente ao id com informação do curso

Url: `/api/student/{id} (get)`

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{  
  
  "id": 1,  
  "stu_email": "example@iol.xyz",  
  "stu_crse_id": 1,  
  "stu_photo_id": 3,  
  "name": "Diogo Panamera",
```

```
"bdate": "2001-06-12",
"currentAge": 20,
"gender": "M"
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): O aluno não foi encontrado

```
{
  "timestamp": "2021-11-18T09:26:16.089+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "Student with id 3 not found.",
  "path": "/api/student/3"
}
```

Adicionar um aluno

Recebe a informação do aluno e qual o curso e cria um novo aluno
O curso tem de já existir

Url: /api/student/new (**post**)

Dados:

```
{
  "name": "Ricardo coelho",
  "email": "example3@iol.xyz",
  "photoId": 3,
  "password": "admin",
  "bdate": "2000-06-12",
  "gender": "M",
  "crseId": 2
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 3,
    "name": "Ricardo coelho",
    "email": "example3@iol.xyz",
    "password": "admin",
    "bdate": "2000-06-12",
    "gender": "M",
    "crseId": 2,
    "photoId": 3,
    "currentAge": 21
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/student"
}
```

Recurso Type

Ir buscar todos os alunos

Devolve todos os alunos na base dados cada um com informação do curso

Url: /api/type (get)

Resultado:

```
[
  {
    "id": 1,
    "name": "Conhecimento",
    "event": "Palestra"
  },
  {
    "id": 2,
    "name": "Música",
    "event": "Concerto"
  },
  {
    "id": 3,
    "name": "Conhecimento",
    "event": "Workshop"
  }
  ... ]
```

Ir buscar um tipo dado um id

Devolve o tipo correspondente ao id com informação do tipo

Url: /api/type/{id} (get)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{
```

```
    "id": 1,
    "name": "Conhecimento",
    "event": "Palestra"
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): O tipo não foi encontrado

```
{
  "timestamp": "2021-11-18T09:26:16.089+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "Student with id 10 not found.",
  "path": "/api/student/10"
}
```

Adicionar um tipo

Recebe a informação do tipo

Url: /api/type (post)

Dados:

```
{
  "name": "Conhecimento",
  "event": "Palestra"
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 1,
    "name": "Conhecimento",
    "event": "Palestra"
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/type"
}
```

Recurso Student_Rate

Ir buscar todos os alunos

Devolve todos os alunos na base dados cada um com informação do curso

Url: /api/student_rate (**get**)

Resultado:

```
[ {  
  "id": 1,  
  "stuRateEv": 7,  
  "comment": "bom evento e spot acolhedor",  
  "stuRid": 1,  
  "evRid": 1  
},  
{  
  "id": 2,  
  "stuRateEv": 8,  
  "comment": "boa organizacao",  
  "stuRid": 2,  
  "evRid": 1  
},  
{  
  "id": 3,  
  "stuRateEv": 8,  
  "comment": "Adorei este evento",  
  "stuRid": 1,  
  "evRid": 2  
}  
... ]
```

Ir buscar um student_rate dado um id

Devolve o student_rate correspondente ao id

Url: /api/student_rate/{id} (**get**)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{  
  "id": 1,  
  "stuRateEv": 7,  
  "comment": "bom evento e spot acolhedor",  
  "stuRid": 1,  
  "evRid": 1  
}
```

```
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): O student_rate não foi encontrado

```
{
  "timestamp": "2021-11-18T09:26:16.089+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "Student with id 15 not found.",
  "path": "/api/student/15"
}
```

Adicionar um student_rate

Recebe a avaliação de um evento e o comentário e cria um student_rate novo

Url: /api/student_rate/add (**post**)

Dados:

```
{
  "stuRateEv": 7,
  "comment": "bom evento e spot acolhedor",
  "stuRid": 1,
  "evRid": 1
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 1,
    "stuRateEv": 7,
    "comment": "bom evento e spot acolhedor",
    "stuRid": 1,
    "evRid": 1
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/student_rate"
```

```
}
```

Recurso Student_event

Ir buscar todos os dados da student_event

Devolve todos os alunos que participaram em eventos, em quais e quando entraram no evento

Url: /api/student_event (get)

Resultado:

```
[ {  
  
    "id": 1,  
    "entryId": 1,  
    "evEntrytime": "15:00:00",  
    "evId": 1  
  },  
  {  
  
    "id": 2,  
    "entryId": 2,  
    "evEntrytime": "16:00:00",  
    "evId": 2  
  }  
  ... ]
```

Ir buscar um aluno dado um id

Devolve o aluno que entrou no evento, qual o evento e a hora a que entrou.

Url: /api/student_event/{id} (get)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{  
  
  "id": 1,  
  "entryId": 1,  
  "evEntrytime": "15:00:00",  
  "evId": 1  
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): O student_event não foi encontrado

```
{  
  
  "timestamp": "2021-11-18T09:26:16.089+00:00",  
  "status": 404,  
}
```



```
"error": "Not Found",
"message": "Student_event with id 10 not found.",
"path": "/api/student/10"
}
```

Adicionar um student_event

Recebe a informação do id do aluno, hora de entrada no evento e qual o evento que participou.

Url: /api/student_event/add (**post**)

Dados:

```
{
  "entryId": 2,
  "evEntrytime": "16:00:00",
  "evId": 6
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 8,
    "entryId": 2,
    "evEntrytime": "16:00:00",
    "evId": 6
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc...",
  "path": "/api/student_event"
}
```

Recurso Student_course

Ir buscar todos os cursos

Devolve todos os cursos

Url: /api/student_course (get)

Resultado:

```
[ {  
  "id": 1,  
  "name": "Engenharia Informática"  
},  
{  
  "id": 2,  
  "name": "Marketing e Publicidade"  
},  
{  
  "id": 3,  
  "name": "Design"  
}  
... ]
```

Ir buscar um curso dado um id

Devolve o curso correspondente ao id

Url: /api/student_course/{id} (get)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{  
  "id": 2,  
  "name": "Marketing e Publicidade"  
}
```

Erros:

404 (HttpStatus.NOT_FOUND): O curso não foi encontrado

```
{  
  "timestamp": "2021-11-18T09:26:16.089+00:00",  
  "status": 404,  
  "error": "Not Found",  
  "message": "Student with id 6 not found.",  
  "path": "/api/student/6"  
}
```

Erros:

500: Erro do servidor

```
{  
  "timestamp": "2021-11-18T09:30:08.676+00:00",
```

```
    "status": 500,  
    "error": "Internal Server Error",  
    "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc  
    "path": "/api/student_course"  
}
```

Recurso State

Ir buscar todos os estados

Devolve todos os estados de eventos

Url: /api/state (**get**)

Resultado:

```
[ {  
  "id": 1,  
  "event": "A decorrer"  
},  
{  
  "id": 2,  
  "event": "Terminado"  
}  
... ]
```

Ir buscar um estado dado um id

Devolve o estado correspondente ao id

Url: /api/state/{id} (**get**)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{  
  "id": 1,  
  "event": "A decorrer"  
}
```

Erros:

404 (**HttpStatus.NOT_FOUND**): O estado não foi encontrado

```
{  
  "timestamp": "2021-11-18T09:26:16.089+00:00",  
  "status": 404,  
}
```

```
{
  "error": "Not Found",
  "message": "Student with id 10 not found.",
  "path": "/api/student/10"
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/student"
}
```

Recurso Spot

Ir buscar todos os spots

Devolve todos os spots.

Url: /api/spots (get)

Resultado:

```
[ {
  "id": 1,
  "name": "Cantinho de Santos",
  "longitude": -9.153784536860059,
  "latitude": 38.70787407385133,
  "description": "Lugar de esplanada",
  "photoId": 1
},
{
  "id": 2,
  "name": "IADE",
  "longitude": -9.152457364578318,
  "latitude": 38.707341487727426,
  "description": "Faculdade de Design e Tecnologias",
  "photoId": 2
}
... ]
```

Ir buscar um spot dado um id

Devolve o spot correspondente ao id.

Url: /api/spots/{id} (get)
Parâmetros: id - inteiro positivo que corresponde ao id do aluno a obter
Resultado: <pre>{ "id": 1, "name": "Cantinho de Santos", "longitude": -9.153784536860059, "latitude": 38.70787407385133, "description": "Lugar de esplanada", "photoId": 1 }</pre>
Erros: 404 (HttpStatus.NOT_FOUND): O spot não foi encontrado <pre>{ "timestamp": "2021-11-18T09:26:16.089+00:00", "status": 404, "error": "Not Found", "message": "Student with id 3 not found.", "path": "/api/student/3" }</pre>

Adicionar um spot Recebe a informação do spot.
Url: /api/spots (post)
Dados: <pre>{ "name": "Cantinho de Santos", "longitude": -9.153784536860059, "latitude": 38.70787407385133, "description": "Lugar de esplanada", "photoId": 1 }</pre>
Resultado: <pre>{ "message": "1 registration created", "object": { "id": 1, "name": "Cantinho de Santos", "longitude": -9.153784536860059, "latitude": 38.70787407385133, "description": "Lugar de esplanada",</pre>

```
    "photoId": 1
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/spots"
}
```

Recurso Orgs

Ir buscar todas as organizações

Devolve todas as organizadoras de eventos.

Url: /api/orgs (get)

Resultado:

```
[ {
  "id": 1,
  "name": "AE IADE",
  "initials": "AEI",
  "email": "aeiade@gmail.com",
  "password": "aeiade"
},
{
  "id": 2,
  "name": "Tuna Academica IADE",
  "initials": "TAI",
  "email": "taiade@gmail.com",
  "password": "taiade"
}
... ]
```

Ir buscar uma organização dado um id

Devolve a org correspondente ao id

Url: /api/orgs/{id} (get)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{
  "id": 2,
  "name": "Tuna Academica IADE",
  "initials": "TAI",
  "email": "taiade@gmail.com",
  "password": "taiade"
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): A org não foi encontrado

```
{
  "timestamp": "2021-11-18T09:26:16.089+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "Student with id 30 not found.",
  "path": "/api/student/30"
}
```

Adicionar uma organização

Recebe a informação da organização

Url: /api/orgs (post)

Dados:

```
{
  "name": "Tuna Academica IADE",
  "initials": "TAI",
  "email": "taiade@gmail.com",
  "password": "taiade"
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 2,
    "name": "Tuna Academica IADE",
    "initials": "TAI",
    "email": "taiade@gmail.com",
    "password": "taiade"
  }
}
```

Erros:

500: Erro do servidor

```
{
```

```
    "timestamp": "2021-11-18T09:30:08.676+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc
    "path": "/api/orgs"
}
```

Recurso Evento

Ir buscar todos os eventos

Devolve todos os eventos na base de dados.

Url: /api/events (get)

Resultado:

```
[ {
  "id": 2,
  "name": "Workshop AI",
  "description": "Um workshop para todos os interessados em inteligencia artificial",
  "typeId": 3,
  "date": "2021-11-22",
  "starttime": "15:00:00",
  "endtime": "17:00:00",
  "duration": "02:00:00",
  "orgId": 1,
  "spotId": 2,
  "capacity": 50,
  "photosId": 2,
  "stateId": 3,
  "rateId": 2
},
{
  "id": 1,
  "name": "5º Concerto TAI",
  "description": "Um concerto que todos vão gostar",
  "typeId": 2,
  "date": "2021-11-22",
  "starttime": "13:00:00",
  "endtime": "15:00:00",
  "duration": "02:00:00",
  "orgId": 2,
  "spotId": 1,
  "capacity": 200,
  "photosId": 1,
  "stateId": 3,
```



```
        "rateId": 1
    }
    ... ]
```

Ir buscar um evento dado um id

Devolve o evento correspondente ao id

Url: /api/events/{id} (get)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
{
  "id": 1,
  "name": "5° Concerto TAI",
  "description": "Um concerto que todos vão gostar",
  "typeId": 2,
  "date": "2021-11-22",
  "starttime": "13:00:00",
  "endtime": "15:00:00",
  "duration": "02:00:00",
  "orgId": 2,
  "spotId": 1,
  "capacity": 200,
  "photosId": 1,
  "stateId": 3,
  "rateId": 1
}
```

Erros:

404 ([HttpStatus.NOT_FOUND](#)): O aluno não foi encontrado

```
{
  "timestamp": "2021-11-18T09:26:16.089+00:00",
  "status": 404,
  "error": "Not Found",
  "message": "Student with id 20 not found.",
  "path": "/api/student/20"
}
```

Adicionar um evento

Recebe a informação do evento e publica na base de dados.

Url: /api/events/add (post)

Dados:

```
{
  "name": "5° Concerto TAI",
  "description": "Um concerto que todos vão gostar",
  "typeId": 2,
  "date": "2021-11-22",
  "starttime": "13:00:00",
  "endtime": "15:00:00",
  "duration": "02:00:00",
  "orgId": 2,
  "spotId": 1,
  "capacity": 200,
  "photosId": 1,
  "stateId": 3,
  "rateId": 1
}
```

Resultado:

```
{
  "message": "1 registration created",
  "object": {
    "id": 1,
    "name": "5° Concerto TAI",
    "description": "Um concerto que todos vão gostar",
    "typeId": 2,
    "date": "2021-11-22",
    "starttime": "13:00:00",
    "endtime": "15:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 1
  }
}
```

Erros:

500: Erro do servidor

```
{
  "timestamp": "2021-11-18T09:30:08.676+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "message": "could not execute statement; SQL [n/a]; nested exception is org.hibernate.exc",
  "path": "/api/events"
}
```

Ir buscar todos os eventos num determinado spot.

Devolve os eventos no spot correspondente ao id

Url: /api/events/spot/{id} (**get**)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
[
  {
    "id": 1,
    "name": "5º Concerto TAI",
    "description": "Um concerto que todos vão gostar",
    "typeId": 2,
    "date": "2021-11-22",
    "starttime": "13:00:00",
    "endtime": "15:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 1
  },
  {
    "id": 7,
    "name": "Convivio mazbrabos",
    "description": "convivio hardcore",
    "typeId": 4,
    "date": "2022-03-01",
    "starttime": "12:00:00",
    "endtime": "00:00:00",
    "duration": "12:00:00",
    "orgId": 1,
    "spotId": 1,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 2
  }
  ...]
```

Ir buscar todos os eventos de uma org.
Devolve os eventos da org correspondente ao id

Url: /api/events/org/{id} (**get**)

Parâmetros:
id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
[
  {
    "id": 1,
    "name": "5º Concerto TAI",
    "description": "Um concerto que todos vão gostar",
    "typeId": 2,
    "date": "2021-11-22",
    "starttime": "13:00:00",
    "endtime": "15:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 1
  },
  {
    "id": 15,
    "name": "Concerto N5 TAI",
    "description": "Concerto da Tuna",
    "typeId": 2,
    "date": "2022-07-14",
    "starttime": "15:00:00",
    "endtime": "17:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 240,
    "photosId": 1,
    "stateId": 3,
    "rateId": 2
  }
  ...]
```

Ir buscar todos os eventos de um determinado tipo.
Devolve os eventos com um determinado tipo através do id.

Url: /api/events/type/{id} (get)

Parâmetros:
id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
[
  {
    "id": 1,
    "name": "5º Concerto TAI",
    "description": "Um concerto que todos vão gostar",
    "typeId": 2,
    "date": "2021-11-22",
    "starttime": "13:00:00",
    "endtime": "15:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 1
  },
  {
    "id": 15,
    "name": "Concerto N5 TAI",
    "description": "Concerto da Tuna",
    "typeId": 2,
    "date": "2022-07-14",
    "starttime": "15:00:00",
    "endtime": "17:00:00",
    "duration": "02:00:00",
    "orgId": 2,
    "spotId": 1,
    "capacity": 240,
    "photosId": 1,
    "stateId": 3,
    "rateId": 2
  }
  ...]
```

Ir buscar todos os eventos de um determinado tipo.

Devolve os eventos com um determinado tipo num determinado spot através do id.

Url: /api/events/spot/{spot_id}/type/{type_id} (**get**)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
[
  {
    "id": 16,
    "name": "Palestra de Marketing",
    "description": "Estratégias de Marketing Inovadoras",
    "typeId": 1,
    "date": "2022-05-02",
    "starttime": "15:00:00",
    "endtime": "17:00:00",
    "duration": "02:00:00",
    "orgId": 1,
    "spotId": 2,
    "capacity": 200,
    "photosId": 1,
    "stateId": 3,
    "rateId": 2
  }
]
```

Ir buscar todos os eventos que um aluno participou.

Devolve os eventos que aluno participou através do id.

Url: /api/events/historico/{id} (**get**)

Parâmetros:

id - inteiro positivo que corresponde ao id do aluno a obter

Resultado:

```
[
  [
    {
      "id": 1,
      "name": "5º Concerto TAI",
      "description": "Um concerto que todos vão gostar",
      "typeId": 2,
      "date": "2021-11-22",
      "starttime": "13:00:00",
      "endtime": "15:00:00",
      "duration": "02:00:00",
      "orgId": 2,
      "spotId": 1,
      "capacity": 200,
      "photosId": 1,
      "stateId": 3,
      "rateId": 1
    }
  ]
]
```