Employee REST API – Testing Manual

Index

1. Downloading the API.

This Api is versioned using a Github repository, for downloading go to https://github.com/aldoobed/employee-REST.git, this link also could be use for clone the respository using any a git client.

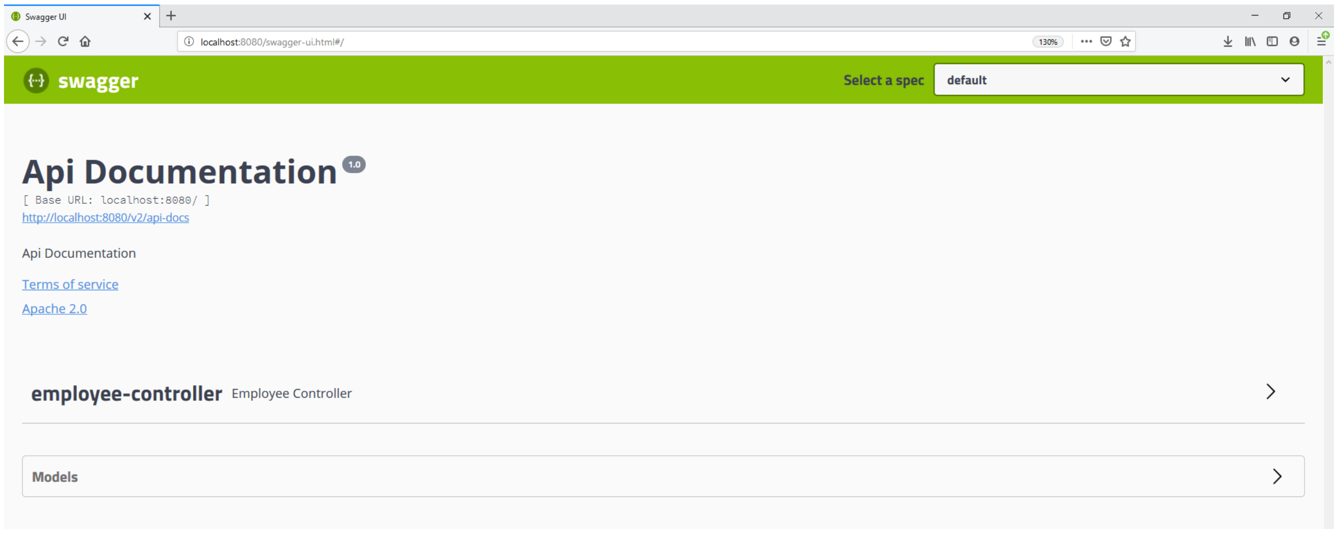
2. Building and Running.

This project was developed using spring boot and it has been provided with maven wrapper, so for running you just have to:

1. Open a command line window or terminal
2. Go to [download path]/employee-REST
3. Execute:
   unix base: `./mvnw spring-boot:run`
   windows: `mvnw.cmd spring-boot:run`
4. Open a web browser and enter url: http://localhost:8080/swagger-ui.html
5. REST API methods will be shown and could be run from there.

3. Interacting with the API

This api contains live documentation provided by using Swagger2, this helps to create a user interface to interact with the api urls. Once opened the URL in step below, it will show api description, click on "employee-controller" to show methods.



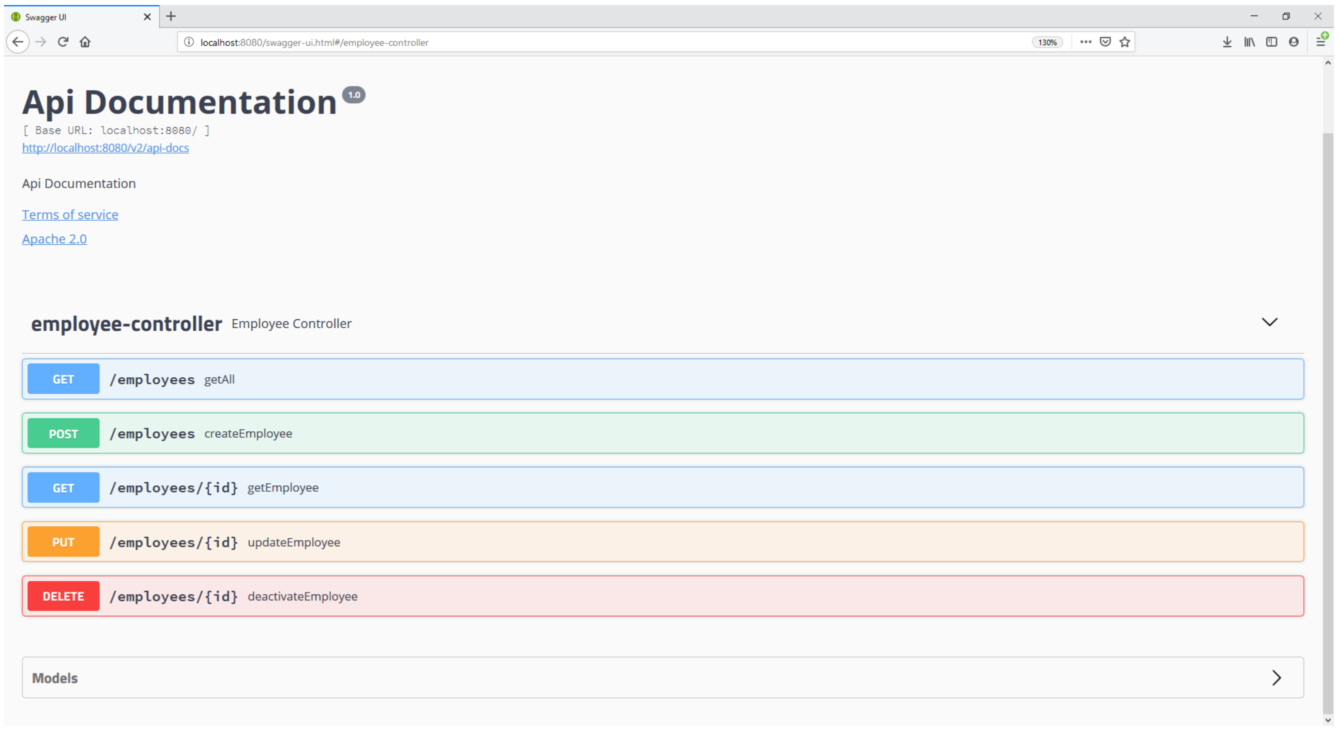
Api Documentation 1.0
[Base URL: localhost:8080/]
http://localhost:8080/v2/api-docs

Api Documentation

Terms of service
Apache 2.0

employee-controller Employee Controller

Models


Swagger UI

localhost:8080/swagger-ui.html#/employee-controller

Api Documentation 1.0

[Base URL: localhost:8080/]

http://localhost:8080/v2/api-docs

Api Documentation

Terms of service

Apache 2.0

employee-controller Employee Controller

GET /employees getAll

POST /employees createEmployee

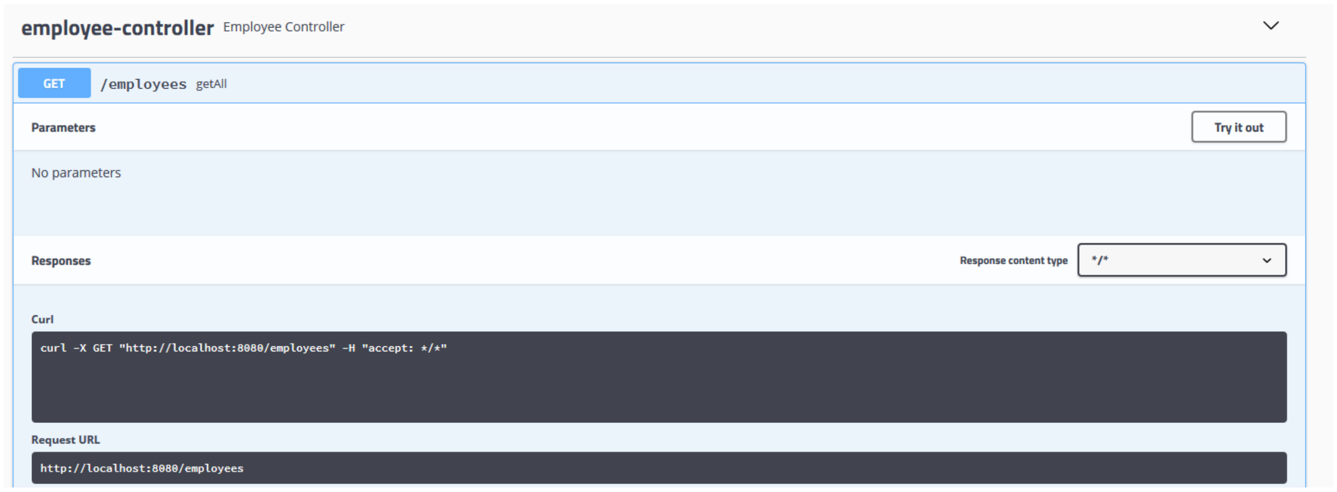
GET /employees/{id} getEmployee

PUT /employees/{id} updateEmployee

DELETE /employees/{id} deactivateEmployee

Models

When clicking in any method, description and examples for each one will be shown and also a “try it out” button that will be used for our request testing.


employee-controller Employee Controller

GET /employees getAll

Parameters

Try it out

No parameters

Responses

Response content type */*

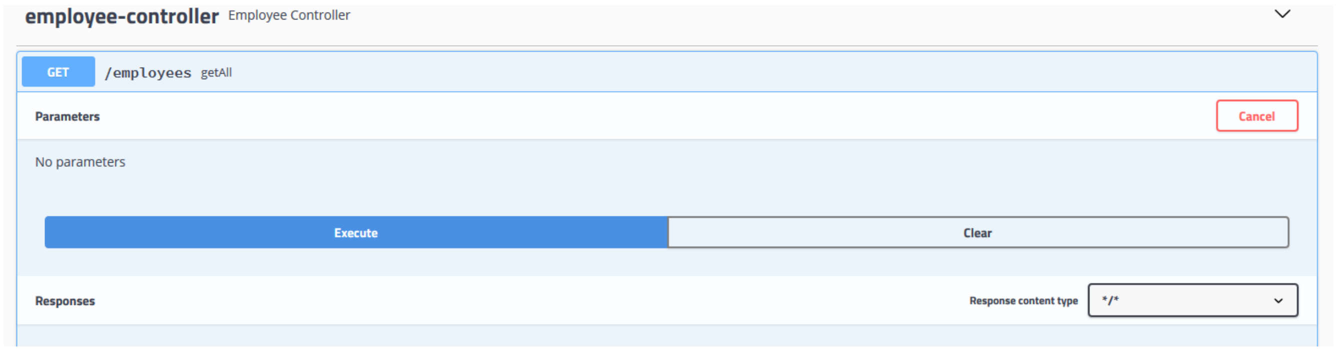
Curl

```
curl -X GET "http://localhost:8080/employees" -H "accept: */*"
```

Request URL

```
http://localhost:8080/employees
```

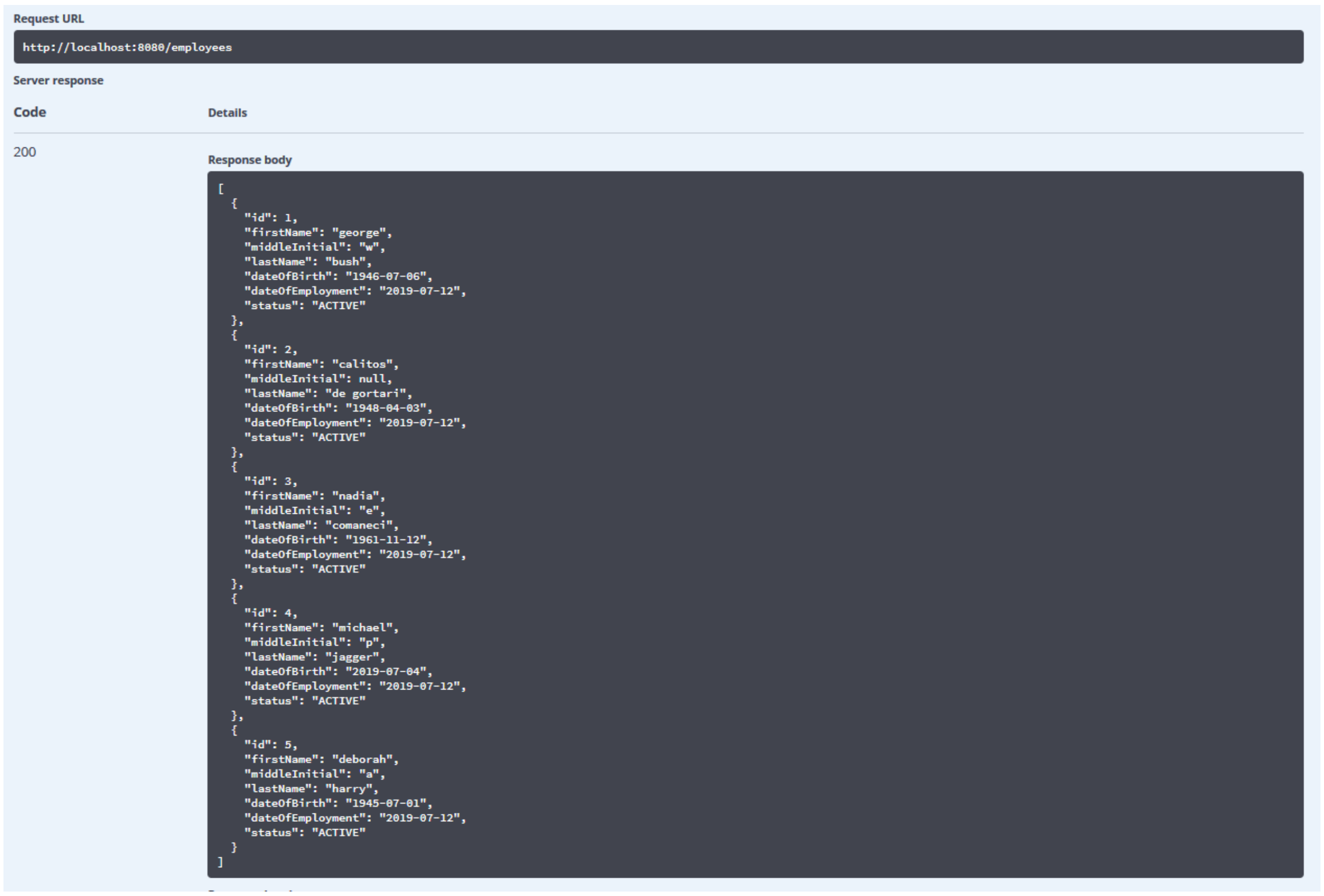
3.1. Get all employees


employee-controller Employee Controller

GET `/employees` getAll

The api preloads 5 employees that can be fetch with getAll method, just click on the “try it out” button and then “execute”.

Preloaded data will be shown below:


Request URL

```
http://localhost:8080/employees
```

Server response

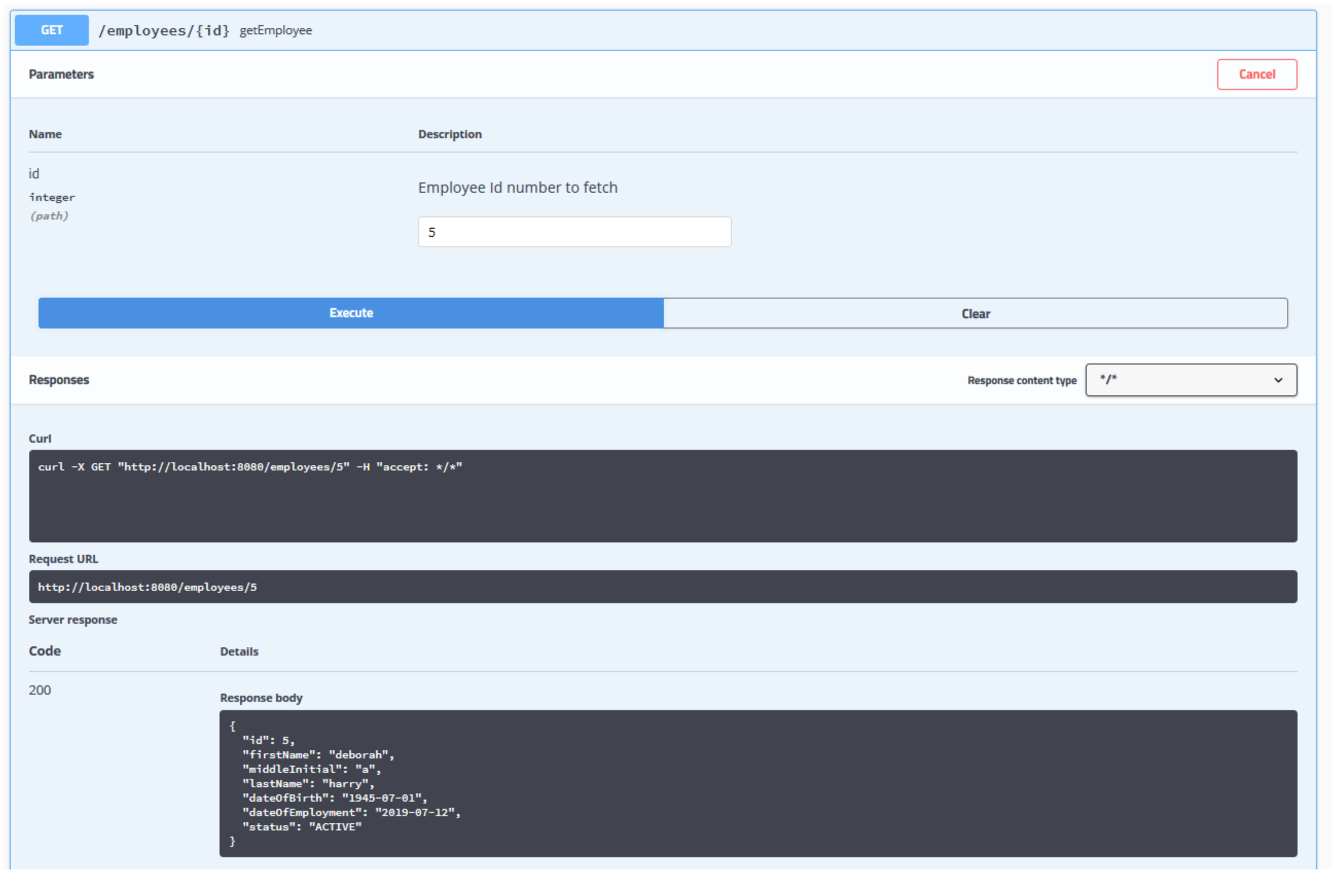
Code: 200

Response body

```
[
  {
    "id": 1,
    "firstName": "george",
    "middleInitial": "w",
    "lastName": "bush",
    "dateOfBirth": "1946-07-06",
    "dateOfEmployment": "2019-07-12",
    "status": "ACTIVE"
  },
  {
    "id": 2,
    "firstName": "calitos",
    "middleInitial": null,
    "lastName": "de gortari",
    "dateOfBirth": "1948-04-03",
    "dateOfEmployment": "2019-07-12",
    "status": "ACTIVE"
  },
  {
    "id": 3,
    "firstName": "nadia",
    "middleInitial": "e",
    "lastName": "comaneci",
    "dateOfBirth": "1961-11-12",
    "dateOfEmployment": "2019-07-12",
    "status": "ACTIVE"
  },
  {
    "id": 4,
    "firstName": "michael",
    "middleInitial": "p",
    "lastName": "jagger",
    "dateOfBirth": "2019-07-04",
    "dateOfEmployment": "2019-07-12",
    "status": "ACTIVE"
  },
  {
    "id": 5,
    "firstName": "deborah",
    "middleInitial": "a",
    "lastName": "harry",
    "dateOfBirth": "1945-07-01",
    "dateOfEmployment": "2019-07-12",
    "status": "ACTIVE"
  }
]
```

3.2. Get employee by id

For fetch and employee using it's id we will use the getEmployee method by clicking on "try it out", it will show an input field where employee id should be entered, then click on "execute".



```
curl -X GET "http://localhost:8080/employees/5" -H "accept: */*"
```

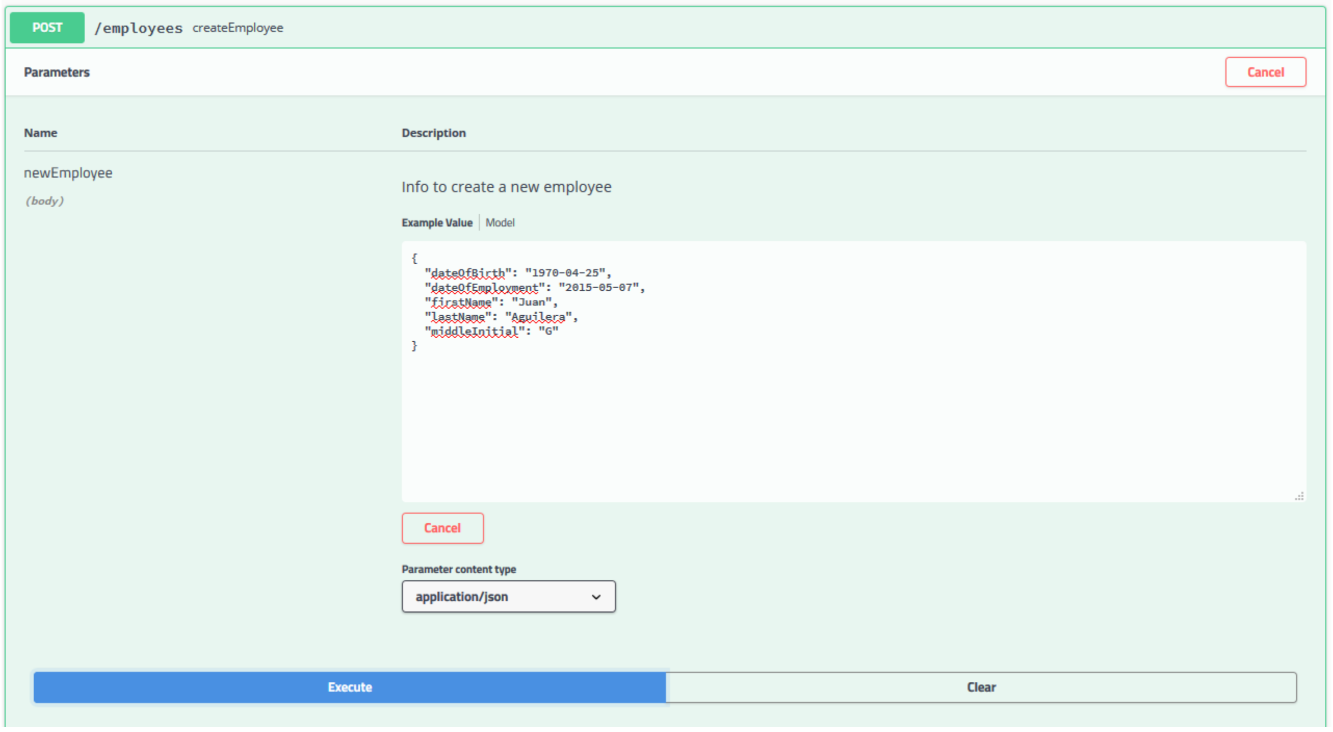
```
http://localhost:8080/employees/5
```

```
{
  "id": 5,
  "firstName": "deborah",
  "middleInitial": "a",
  "lastName": "harry",
  "dateOfBirth": "1945-07-01",
  "dateOfEmployment": "2019-07-12",
  "status": "ACTIVE"
}
```

3.3. Create an employee

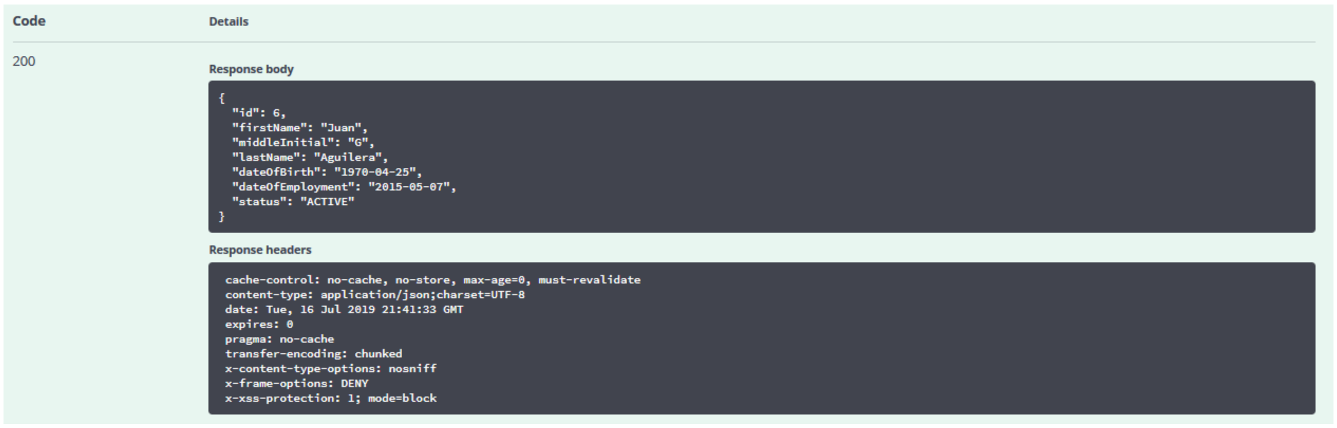
For creating an employee the api use POST method, by clicking on the method followed by clicking on "try it out" we will see the input parameters, for the employee info, an example JSON is preloaded, fields are validated, if any of data entered does not match with this validations an error response will be thrown.

NOTE: The date format acepted for dateOfBirth and dateOfEmployement is "YYYY-MM-DD"



```
{
  "dateOfBirth": "1970-04-25",
  "dateOfEmployment": "2015-05-07",
  "firstName": "Juan",
  "lastName": "Aguilera",
  "middleInitial": "G"
}
```

As response, the api send back the new employee info.



```
{
  "id": 6,
  "firstName": "Juan",
  "middleInitial": "G",
  "lastName": "Aguilera",
  "dateOfBirth": "1970-04-25",
  "dateOfEmployment": "2015-05-07",
  "status": "ACTIVE"
}
```

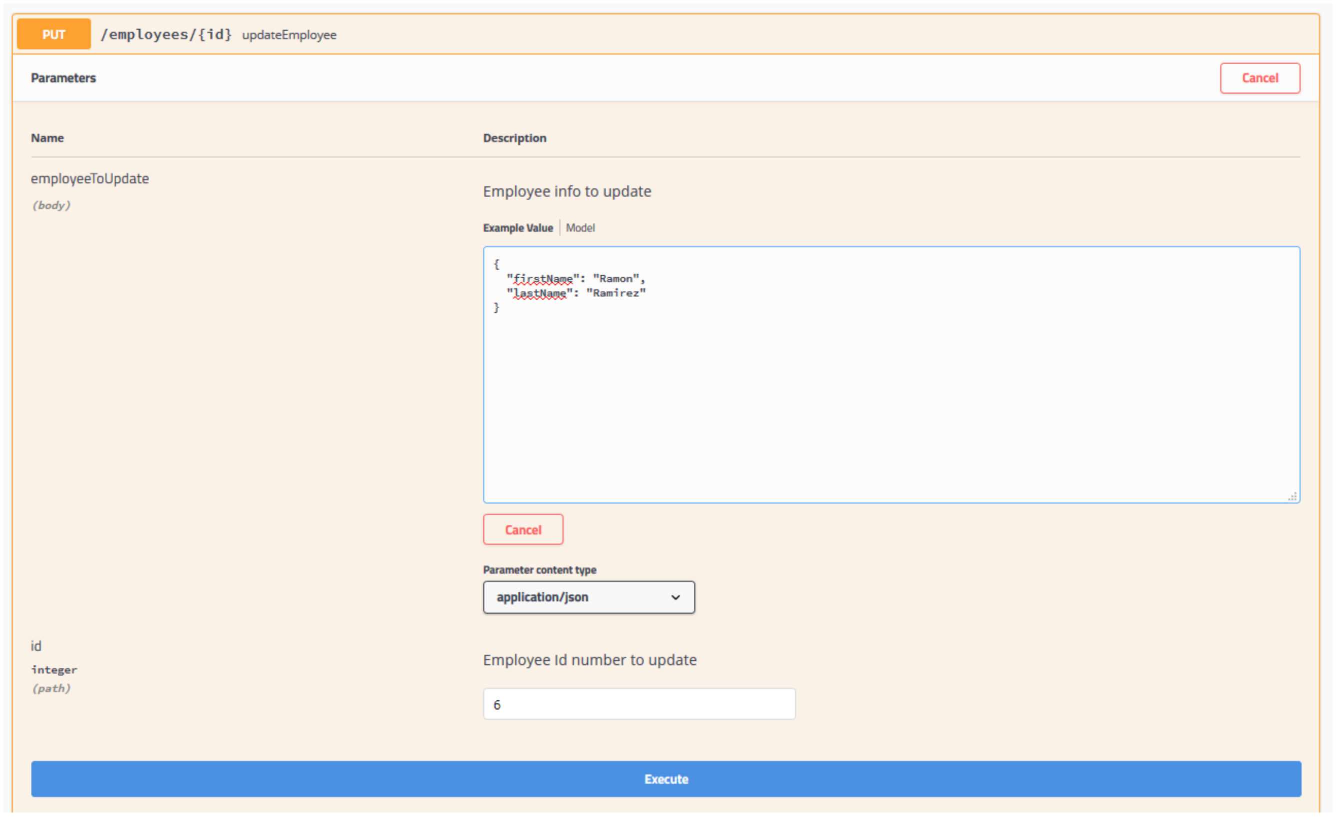
```
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json;charset=UTF-8
date: Tue, 16 Jul 2019 21:41:33 GMT
expires: 0
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```

3.4. Updating employees

The next method is for update employees, this method needs an employee id as param and a JSON object with the info to update.

Click on “try it out” for updateEmployee method will show an example employee JSON, the information to be update should be entered, attributes that will not be used could be removed, fields are validated.

NOTE: The date format acepted for dateOfBirth and dateOfEmployement is “YYYY-MM-DD”



employeeToUpdate
(body)

Employee info to update

```
{
  "firstName": "Ramon",
  "lastName": "Ramirez"
}
```

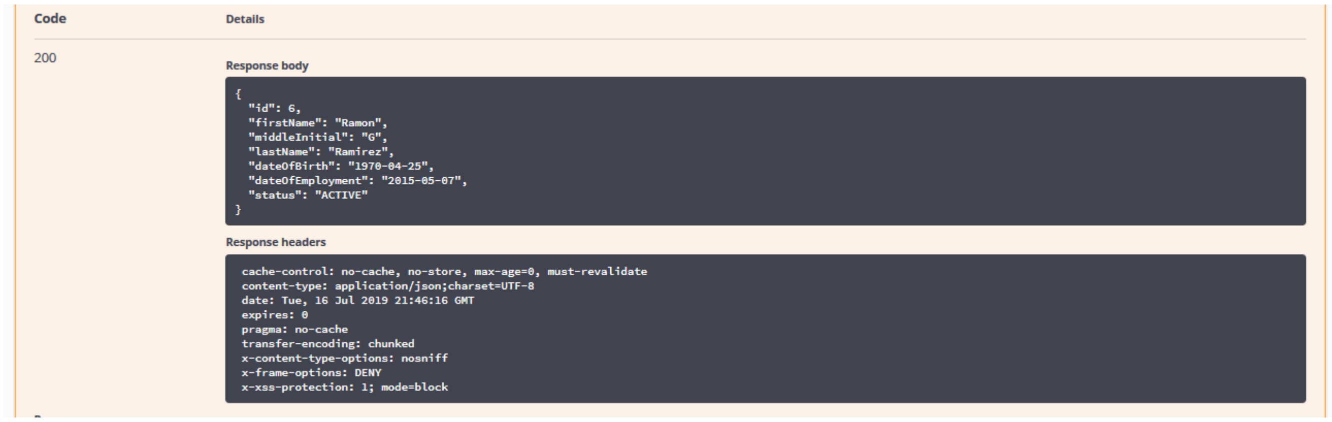
Parameter content type: application/json

id
integer
(path)

Employee Id number to update

6

As response, the api send back the employee with updated info.



200

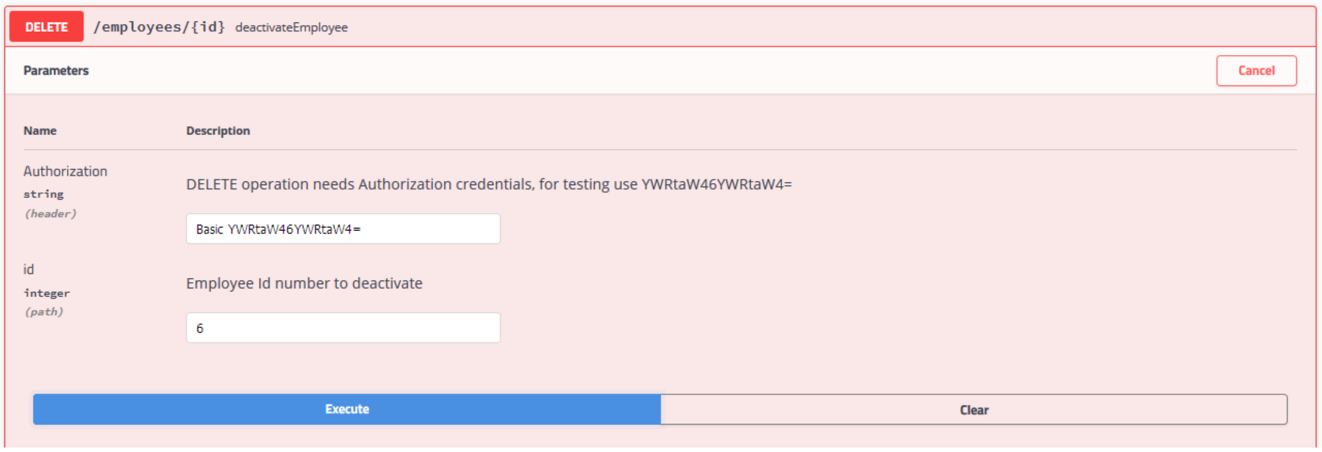
```
{
  "id": 6,
  "firstName": "Ramon",
  "middleInitial": "G",
  "lastName": "Ramirez",
  "dateOfBirth": "1970-04-25",
  "dateOfEmployment": "2015-05-07",
  "status": "ACTIVE"
}
```

```
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-type: application/json;charset=UTF-8
date: Tue, 16 Jul 2019 21:46:16 GMT
expires: 0
pragma: no-cache
transfer-encoding: chunked
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 1; mode=block
```

3.5. Deleting an employee

This method needs to be provided with Authorization header. The api implements basic authorization, the user and password are “admin”, this user and password are base64 encoded, alsa, the api should know the employee id to delete, click on “try it out” to enter the values.

NOTE: The encoded user:password for admin:admin is YWRtaW46YWRtaW4=, the string to be entered as value for Authorization header should be “Basic YWRtaW46YWRtaW4=”.



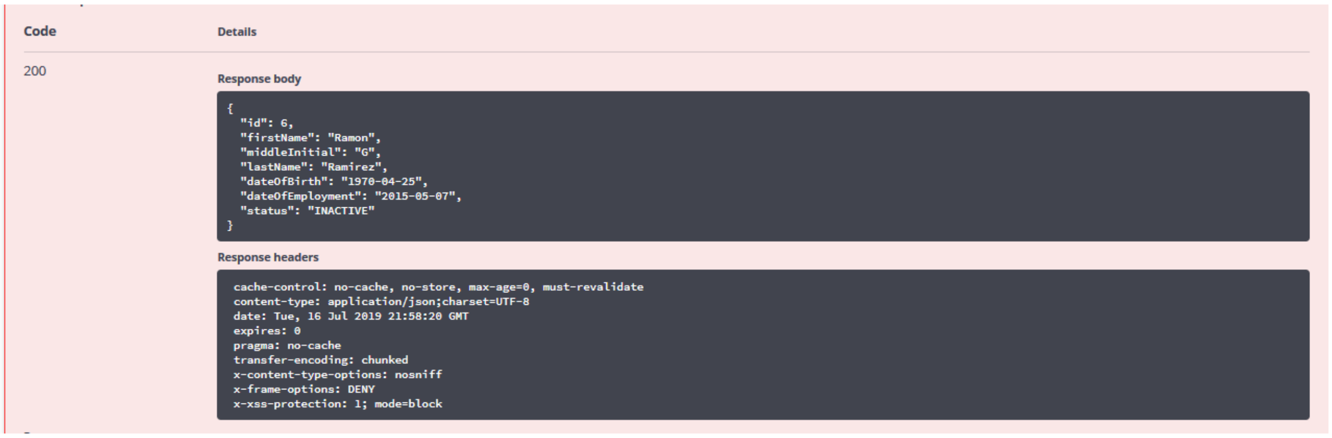
DELETE operation needs Authorization credentials, for testing use YWRtaW46YWRtaW4=

Basic YWRtaW46YWRtaW4=

Employee Id number to deactivate

6

As response, the api send back the employee that has been deactivated.



```
{
  "id": 6,
  "firstName": "Ramon",
  "middleInitial": "G",
  "lastName": "Ramirez",
  "dateOfBirth": "1970-04-25",
  "dateOfEmployment": "2015-05-07",
  "status": "INACTIVE"
}
```

```
 cache-control: no-cache, no-store, max-age=0, must-revalidate
 content-type: application/json;charset=UTF-8
 date: Tue, 16 Jul 2019 21:58:20 GMT
 expires: 0
 pragma: no-cache
 transfer-encoding: chunked
 x-content-type-options: nosniff
 x-frame-options: DENY
 x-xss-protection: 1; mode=block
```

3. Shutting down the API

For Shutting down the api just go to the command line window or terminal opened to run and hit Ctrl+C, this will stop server and api.