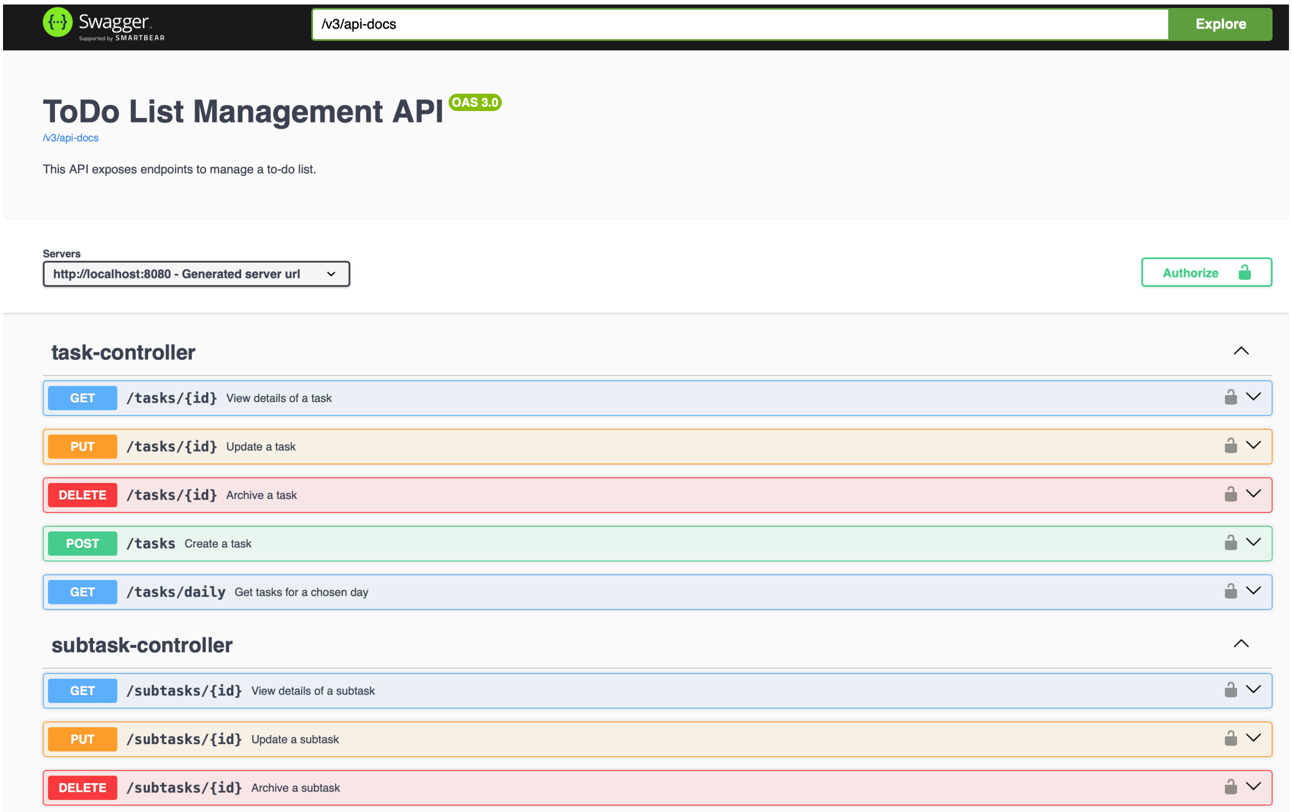
Testing the Application

I recommend using Swagger to test the endpoints that this REST API exposes. You may access it at http://localhost:8080/swagger-ui/index.html



ToDo List Management API

This API exposes endpoints to manage a to-do list.

Servers: http://localhost:8080 - Generated server url

task-controller

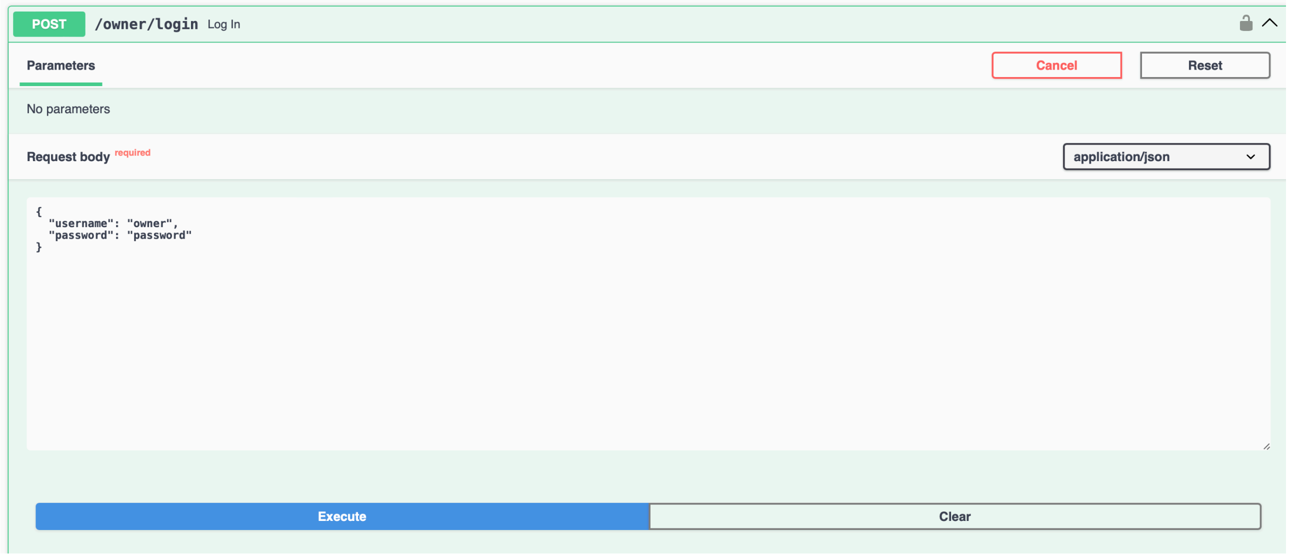
- GET `/tasks/{id}` View details of a task
- PUT `/tasks/{id}` Update a task
- DELETE `/tasks/{id}` Archive a task
- POST `/tasks` Create a task
- GET `/tasks/daily` Get tasks for a chosen day

subtask-controller

- GET `/subtasks/{id}` View details of a subtask
- PUT `/subtasks/{id}` Update a subtask
- DELETE `/subtasks/{id}` Archive a subtask

This application implements the Spring Security based on JWT. Please, follow these steps to authenticate and obtain a token:

Login to Get JWT Token



POST `/owner/login` Log In

No parameters

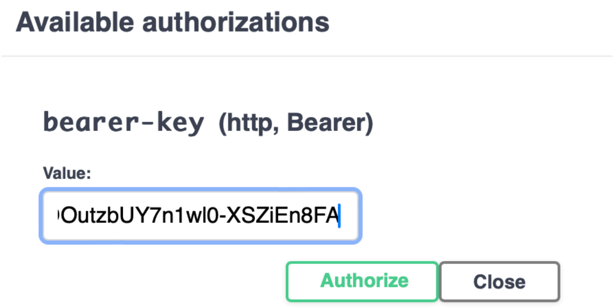
Request body required: application/json

```
{
  "username": "owner",
  "password": "password"
}
```

Try to log in with the credentials used in this request body. In the response body, you will get the generated token. It will be valid for one hour or until the session ends.

Using the JWT Token

Include the JWT in the Authorization header for subsequent requests. In Swagger you can do this by clicking in “Authorize” at the top of the page, or at the lock icons next to each endpoint.



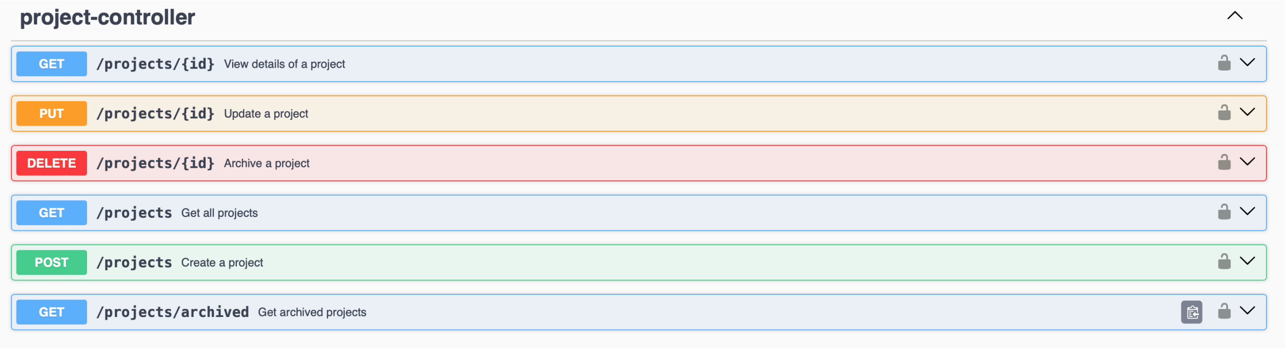
Available authorizations

bearer-key (http, Bearer)

Value: OutzbUY7n1wl0-XSZiEn8FA

Testing the other endpoints:

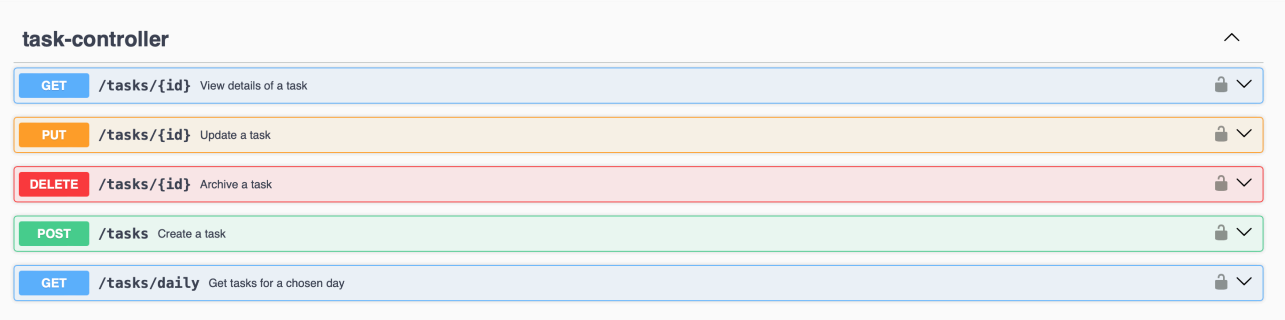
Projects endpoints:



project-controller

```
GET     /projects/{id}     View details of a project
PUT     /projects/{id}     Update a project
DELETE  /projects/{id}     Archive a project
GET     /projects          Get all projects
POST    /projects          Create a project
GET     /projects/archived Get archived projects
```

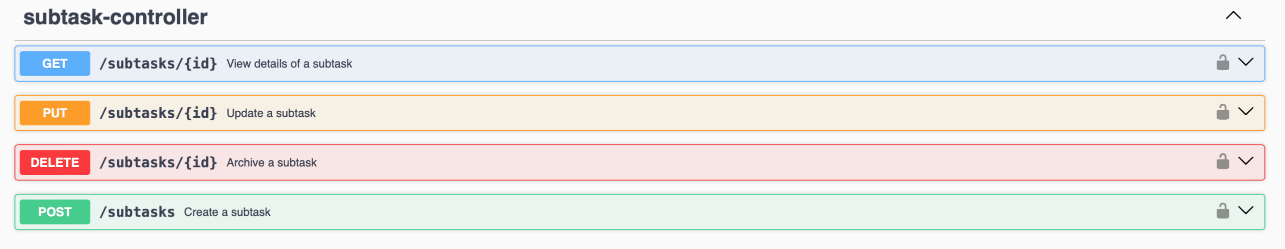
Tasks endpoints:



task-controller

```
GET     /tasks/{id}  View details of a task
PUT     /tasks/{id}  Update a task
DELETE  /tasks/{id}  Archive a task
POST    /tasks       Create a task
GET     /tasks/daily Get tasks for a chosen day
```

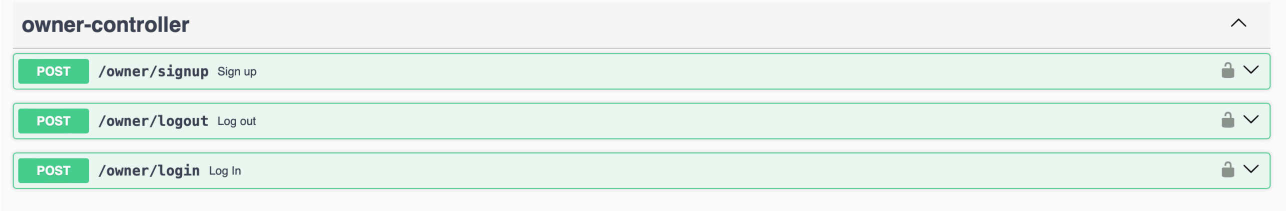
Subtasks endpoints:



subtask-controller

```
GET     /subtasks/{id} View details of a subtask
PUT     /subtasks/{id} Update a subtask
DELETE  /subtasks/{id} Archive a subtask
POST    /subtasks      Create a subtask
```

Owner endpoints:



owner-controller

POST /owner/signup Sign up

POST /owner/logout Log out

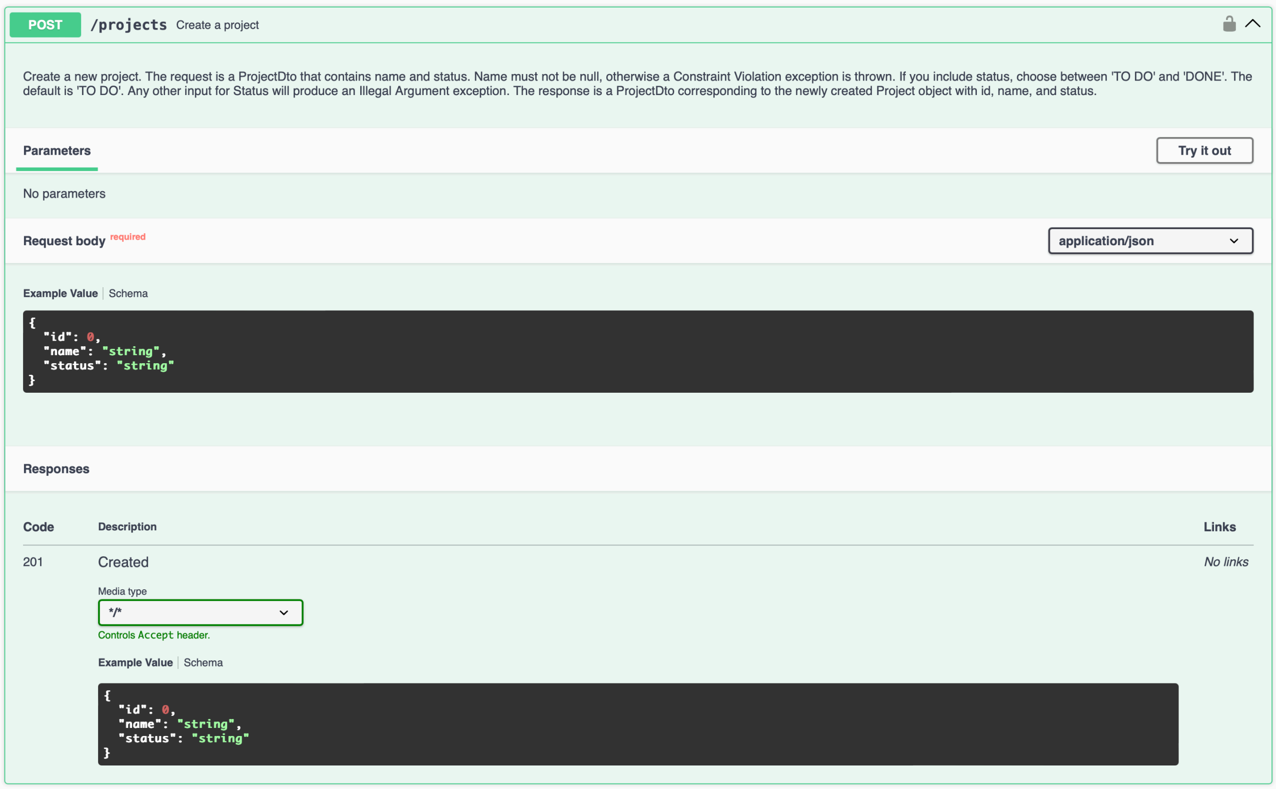
POST /owner/login Log In

Logout:

You have to insert as input your token. When you execute the call, the token will be revoked. If you try to use the token after this process, you will get a 401 - Unauthorized status.

TESTING SCENARIO

Let’s try to explore the endpoints with some concrete data. In order to test a specific endpoint, click on it and the information below will be displayed. You have to click on “Try it out” and fill in the Parameters (Request Body in the following case). Click “Execute” and you will get the response.



POST /projects Create a project

Create a new project. The request is a ProjectDto that contains name and status. Name must not be null, otherwise a Constraint Violation exception is thrown. If you include status, choose between 'TO DO' and 'DONE'. The default is 'TO DO'. Any other input for Status will produce an Illegal Argument exception. The response is a ProjectDto corresponding to the newly created Project object with id, name, and status.

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "status": "string"
}
```

Responses

Code	Description	Links
201	Created	No links

Media type: */*

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "status": "string"
}
```

Keep in mind:

The valid arguments for Status are: “TO DO”, “DONE” or you can leave it empty/null -> it defaults to “TO DO’.

The valid arguments for Priority are: “LOW”, “MEDIUM”, “HIGH” or empty/null -> it defaults to “LOW”.

Inserting the field “name” as NULL will produce an exception.

In the database I have put some initial data.

Considering that you have completed the log in steps with the credentials mentioned at the beginning of this document (please log in if you already logged out), you may try a flow like this:

- Get all projects

- Get details for a project (Use the id of one of the projects you got in the response of the first call)

- Get task details (Use the id of one of the tasks you got in the previous response)

Here you also get in the response a **computed property** “daysUntilDue” which is transient and represents the calculation of days until “dueDate”.

- Get subtask details (Use the id of one of the subtasks you got in the previous response)

- Get daily tasks (Put the date in the format yyyy-mm-dd)

- Archive a task (Its subtasks will also be archived)

- Archive a project (Its tasks will also be archived)

- Get archived projects

You may then try to:

- Update a project

You can rename a project or change its status.

- Update a task

You can rename a task, change its description / status / priority / dueDate. You can move it to another project by inserting its id.

- Update a subtask

You can rename a subtask, change its description / status. You can move it to another task by inserting its id.

- Create another project

- Add a task to the project

- Add a subtask to the task

- Create another user (sign up) and log in with the new credentials. Try to access some of the data above. You will not be able to receive any of them, since they are only accessed by their owner,

Conclusion

This documentation should help you run and test the **Todo List Management** application effectively. For any further questions or issues, please reach out to me.