TODO List API Backend Documentation

This documentation covers the endpoints and functionalities of the TODO List API backend project. The project allows users to create, read, update, and delete tasks from their todo list. It also provides authentication facilities to ensure user data privacy.

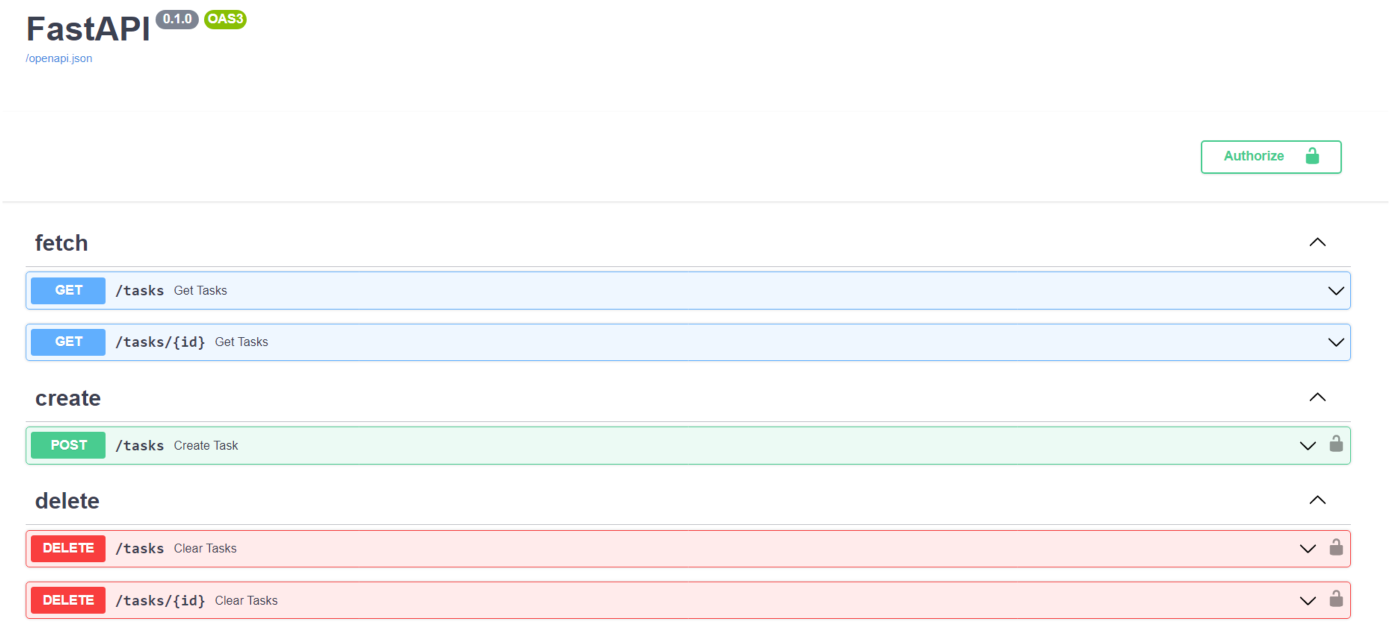
Technologies Used:

The following technologies were used to develop this project:

- Python3
- MySQL
- FASTAPI
- JSON Web Tokens (JWT) for authentication (PyJWT)

API Endpoints:

The API provides the following endpoints for managing tasks:



FastAPI 0.1.0 OAS3

/openapi.json

Authorize

fetch

GET /tasks Get Tasks

GET /tasks/{id} Get Tasks

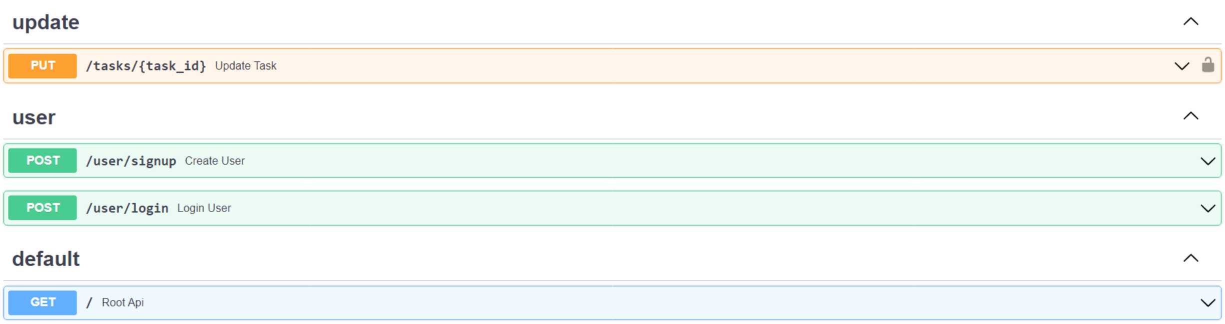
create

POST /tasks Create Task

delete

DELETE /tasks Clear Tasks

DELETE /tasks/{id} Clear Tasks



update

PUT /tasks/{task_id} Update Task

user

POST /user/signup Create User

POST /user/login Login User

default

GET / Root Api

Authentication Endpoints:

#For user SignUp

```
@app.post("/user/signup", tags=["user"])
```

```
@app.post("/user/signup", tags=["user"])
def create_user(user: UserSchema):
   # Create new user into the database
   with cnx.cursor() as cursor:
       sql = "INSERT INTO User (fullname, email, password)
VALUES (%s, %s, %s)"
       cursor.execute(sql, (user.fullname, user.email,
user.password))
       cnx.commit()

   # Return a success message
   return {"Message": f"New user created {user.fullname}"}
```

#For user LogIn

```
@app.post("/user/login", tags=["user"])
```

```
@app.post("/user/login", tags=["user"])
def login_user(credential: UserLoginSchema):
   # Login user
   with cnx.cursor() as cursor:
       sql = "SELECT fullname FROM User WHERE email=%s and
password=%s"
       cursor.execute(sql, (credential.email,
credential.password))
       username = cursor.fetchone()
       if username is None:
           raise invalid_user

   # Return a success message
   return signJWT(credential.email)
```

Reading Endpoints:

To get all tasks and a specific task by {id}:

```
#All Tasks

@app.get("/tasks", tags=["fetch"])
async def get_tasks():
   # To get all tasks
   with cnx.cursor() as cursor:
       cursor.execute("SELECT * FROM Task")
       results = cursor.fetchall()
   return results
```

```
# To get a specific task with id: tak_is
@app.get("/tasks/{id}", tags=["fetch"])
async def get_tasks(id: int):
   # To get a specific task
   with cnx.cursor() as cursor:
       cursor.execute("SELECT * FROM Task WHERE id=%s", id)
       results = cursor.fetchone()
   return results
```

For creating a task:

```
# Define the POST endpoint to create a new task
@app.post("/tasks", tags=["create"],
dependencies=[Depends(authenticate)])
async def create_task(task: TaskCreate):
   # Insert the new task into the database
   with cnx.cursor() as cursor:
       sql = "INSERT INTO Task (name, description, status)
VALUES (%s, %s, %s)"
       cursor.execute(sql, (task.name, task.description,
task.status))
       cnx.commit()

   # Return a success message
   return {"message": "Task created successfully"}
```

For Updating a Task:

```
# To update a specific task with id : task_id
@app.put("/tasks/{task_id}", tags=["update"],
dependencies=[Depends(authenticate)])
async def update_task(task_id: int, task_update: TaskUpdate):
   # Update the task in the database
   with cnx.cursor() as cursor:
       sql = "UPDATE Task SET name=%s, description=%s, status=%s
WHERE id=%s"
       cursor.execute(sql, (task_update.name,
task_update.description, task_update.status, task_id))
       cnx.commit()

   # Return a success message
   return {"message": "Task updated successfully"}
```

For Deleting all tasks and a specific task by {id}

```
# To clear all tasks
@app.delete("/tasks", tags=["delete"],
dependencies=[Depends(authenticate)])
async def clear_tasks():
    # Delete all tasks from the database
    with cnx.cursor() as cursor:
        sql = "DELETE FROM Task"
        cursor.execute(sql)
        cnx.commit()

    # Return a success message
    return {"message": "All tasks deleted successfully"}


# To clear a tasks
@app.delete("/tasks/{id}", tags=["delete"],
dependencies=[Depends(authenticate)])
async def clear_tasks(id: int):
    # Delete all tasks from the database
    with cnx.cursor() as cursor:
        sql = "DELETE FROM Task WHERE id=%s"
        cursor.execute(sql, id)
        cnx.commit()

    # Return a success message
    return {"message": f"A task with id {id} deleted
successfully"}
```

Authentication:

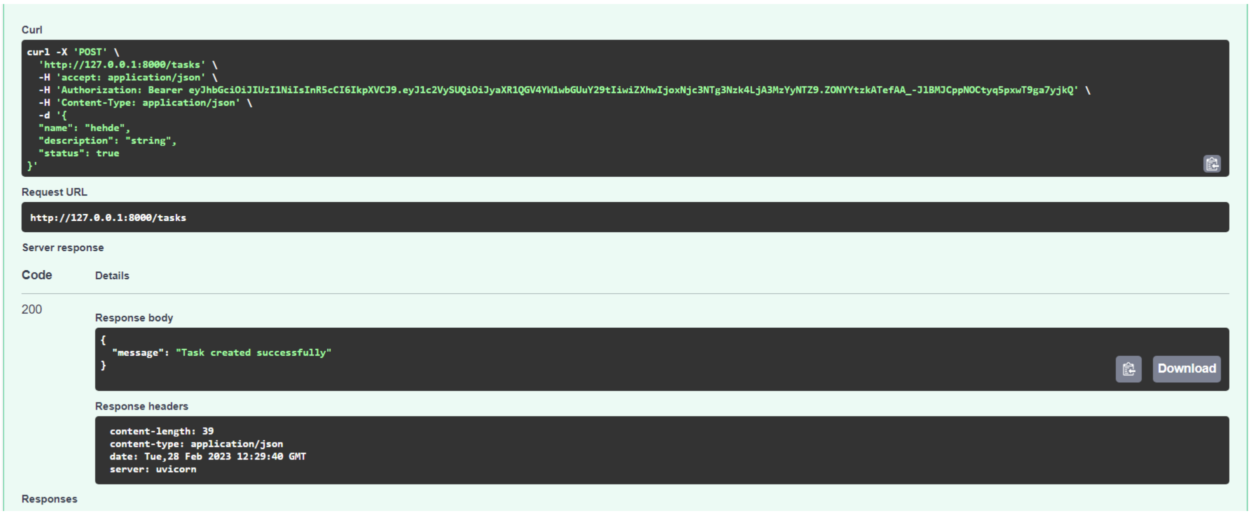
The **API uses JSON Web Tokens (JWT) to authenticate users**. Upon successful registration or login, the API returns an access token. The access token must be included in the Authorization header of all subsequent requests to authenticated endpoints.

Database:

The application stores user data and tasks in a MySQL database. The following tables are used:

- users: Stores user data including name, email, and password.
- tasks: Stores task data including id, name, description, status, and userId.

SCREENSHOTS:



Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/tasks' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiJyaXR1QGV4YW1wbGUuY29tIiwiZXhwIjoxNjc3NTg3Nzk4LjA3MzYyNTZ9.ZONYYtzkATefAA_-J1BMJCppNOCtyq5pxwT9ga7yjkQ' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "hehde",
  "description": "string",
  "status": true
}'
```

Request URL

```
http://127.0.0.1:8000/tasks
```

Server response

Code Details

200

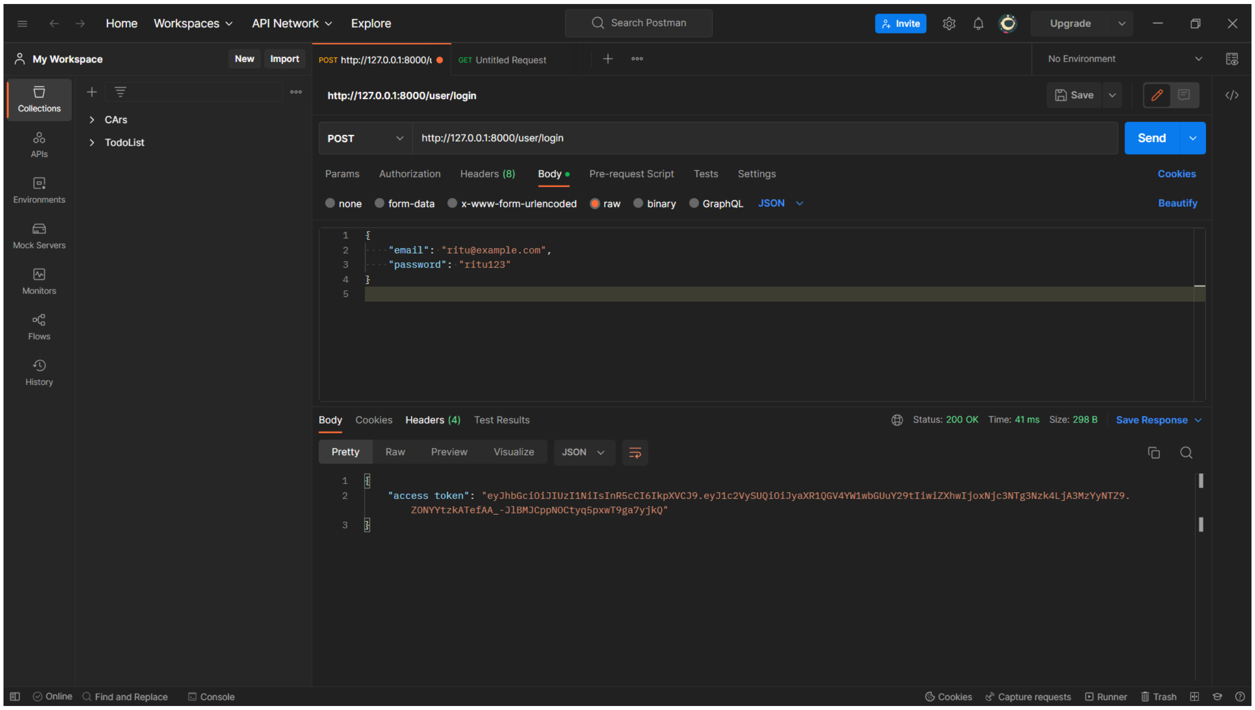
Response body

```
{
  "message": "Task created successfully"
}
```

Response headers

```
 content-length: 39
 content-type: application/json
 date: Tue,28 Feb 2023 12:29:40 GMT
 server: uvicorn
```

Responses



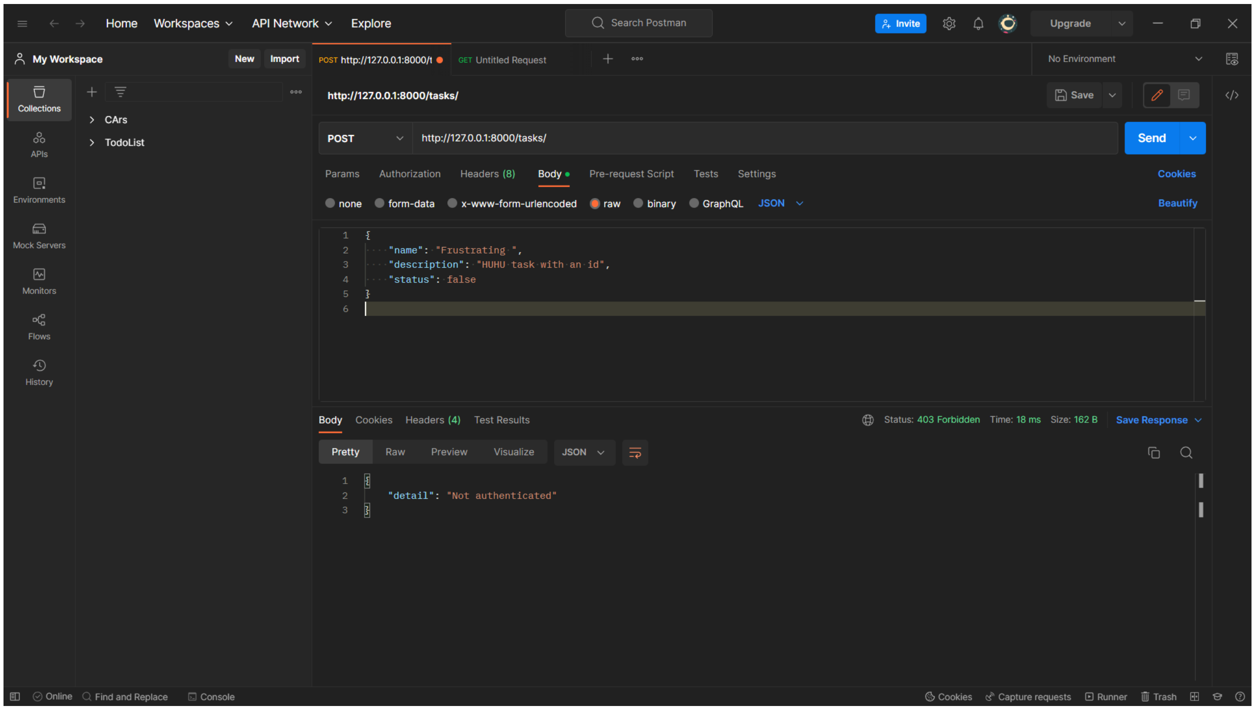
http://127.0.0.1:8000/user/login

POST http://127.0.0.1:8000/user/login

```
{
    "email": "ritu@example.com",
    "password": "ritu123"
}
```

Status: 200 OK Time: 41 ms Size: 298 B

```
{
    "access token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiJyaXR1QGV4YW1wbGUuY29tIiwiZXhwIjoxNjc3NTg3Nzk4LjA3MzYyNTZ9.
        ZONYYtzkATefAA_-JlBMJCppNOCtyq5pxwT9ga7yjkQ"
}
```



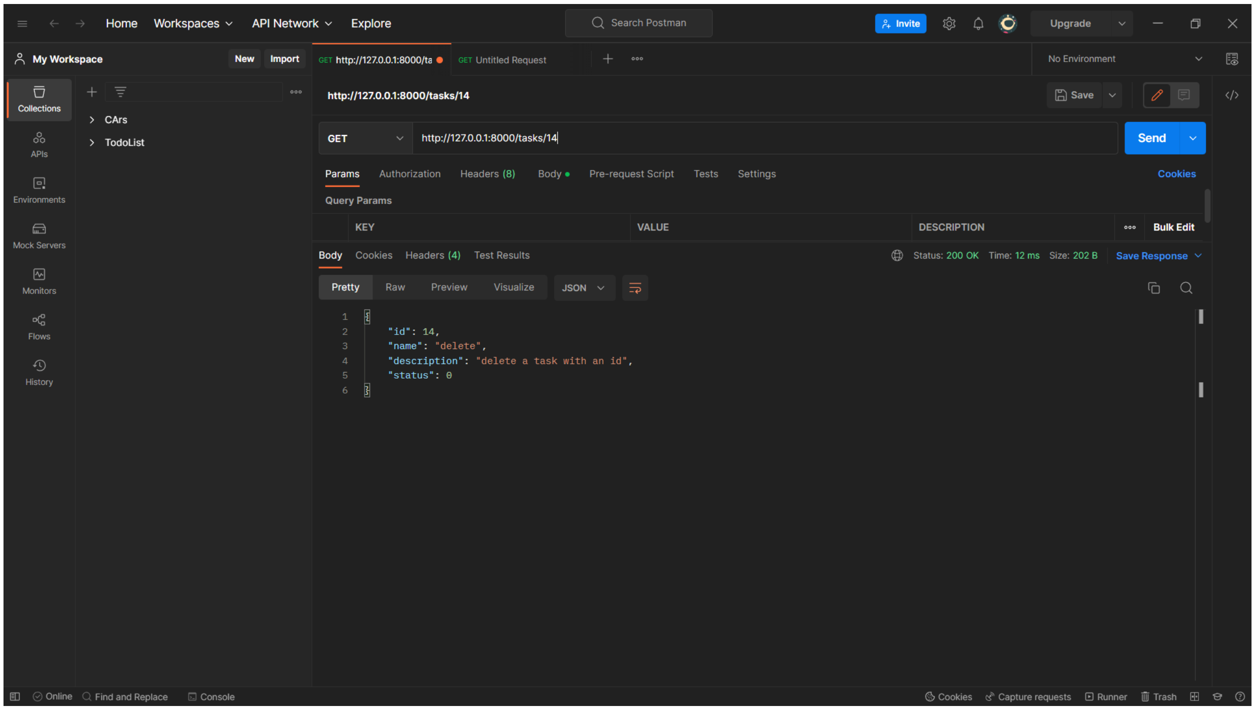
http://127.0.0.1:8000/tasks/

POST http://127.0.0.1:8000/tasks/

```
{
    "name": "Frustrating ",
    "description": "HUHU task with an id",
    "status": false
}
```

Status: 403 Forbidden Time: 18 ms Size: 162 B

```
{
    "detail": "Not authenticated"
}
```

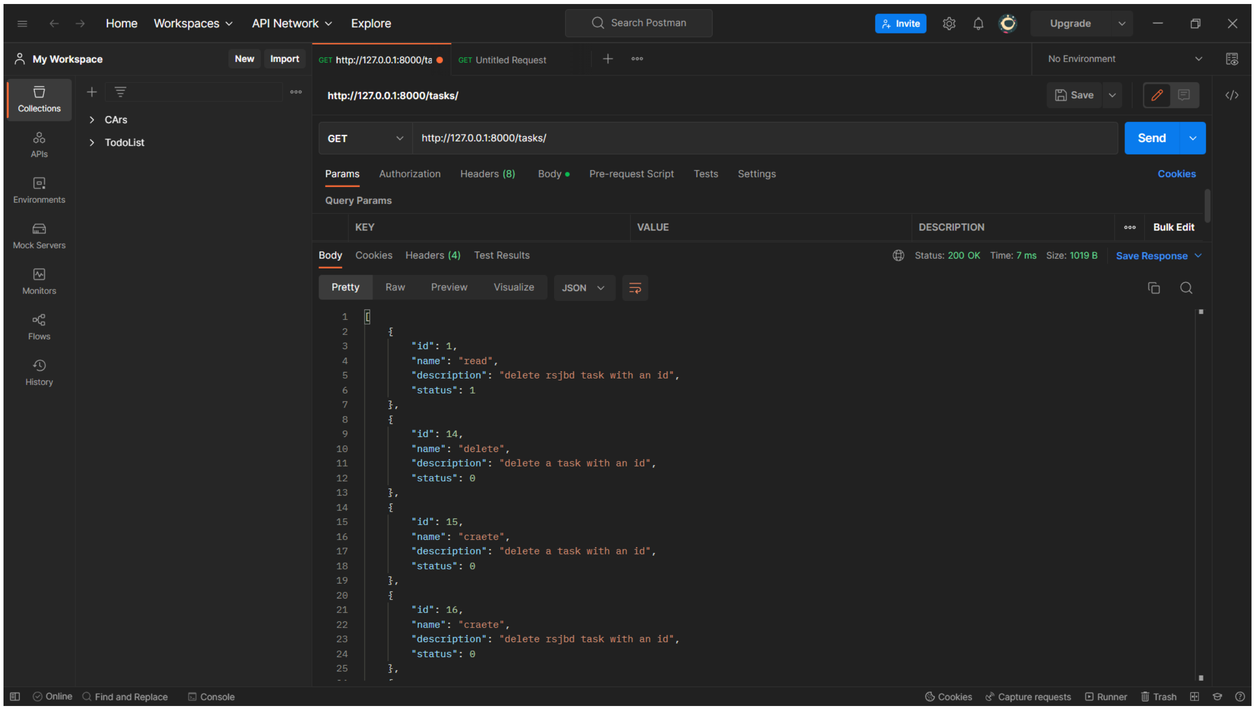


http://127.0.0.1:8000/tasks/14

GET http://127.0.0.1:8000/tasks/14

Status: 200 OK Time: 12 ms Size: 202 B

```
{
    "id": 14,
    "name": "delete",
    "description": "delete a task with an id",
    "status": 0
}
```



http://127.0.0.1:8000/tasks/

GET http://127.0.0.1:8000/tasks/

Status: 200 OK Time: 7 ms Size: 1019 B

```
[
    {
        "id": 1,
        "name": "read",
        "description": "delete rsjbd task with an id",
        "status": 1
    },
    {
        "id": 14,
        "name": "delete",
        "description": "delete a task with an id",
        "status": 0
    },
    {
        "id": 15,
        "name": "craete",
        "description": "delete a task with an id",
        "status": 0
    },
    {
        "id": 16,
        "name": "craete",
        "description": "delete rsjbd task with an id",
        "status": 0
    },
```