1) принимающий задачу и складывающий в очередь, реализованную в сервисе

POST => http://localhost:8080/task/add

{

     "title": "Task",

     "description": "Description",

     "time": "2023-01-01T12:00:00",

     "status": "NEW"

}

Можно назначить исполнителя сразу: (Если он есть)  
  "performerId": "1"

2) считывающий 3 задачи из реализованной очереди и складывающий их в БД несколькими потоками(PostgreSQL).

GET => http://localhost:8080/task/record

3) выдающий все задачи из базы в списке с сокращенными данными (id, title, status).

GET => http://localhost:8080/task/all

4) выдающий задачу по id с полным описанием.

GET => http://localhost:8080/task/{id} -> http://localhost:8080/task/1

5) меняющий задачу по id (все кроме id и performer).

PUT => http://localhost:8080/task/{id}/edit -> http://localhost:8080/task/1/edit

{

     "title": "New Title",

     "description": "New Description",

     "status": "WORK"

}

6) назначить на задачу исполнителя.

PUT => http://localhost:8080/task/{id}/assign -> http://localhost:8080/task/1/assign

(Если уже создан работник с id = 1)

{

    "performerId": "1"

}

7) CRUD операции с сущностями Workers. Не забыть показывать краткую информацию по задачам, назначенным на исполнителя.

**Create**

POST => http://localhost:8080/workers/create

{

    "name": "Danil",

    "position": "Manager",

    "avatar": "https://sait.com/avatar.png"

}

**Read (ALL)**

GET => http://localhost:8080/workers

**Read {ID}**

GET => http://localhost:8080/workers/{id} -> http://localhost:8080/workers/1

**Update**

PUT => http://localhost:8080/workers/{id}/update -> http://localhost:8080/workers/1/update

{

    "name": "new Danil",

    "position": "Developer",

    "avatar": "https://sait.com/new\_avatar.png"

}

**Delete**

DELETE => http://localhost:8080/workers/{id}/delete -> http://localhost:8080/workers/1/delete

**краткая информация по задачам**

GET => http://localhost:8080/workers/{id}/tasks -> http://localhost:8080/workers/1/tasks