MagicTask

Проект

Необходимое ПО для запуска

1. *MySQL*
2. *Java 8+*

3. Intellij IDEA
4. PostmanCanary (использовалось для тестирования запросов)

Инструкция по запуска

1. Запустите проект в среде
2. Создайте базу данных MySQL
3. Ее настройки введите в application.properties
4. Запустите приложение

Также для вашего удобства сервер расположен на ресурсе https://magictasker.herokuapp.com/ , и вы можете иметь доступ для тестирования 24/7.

API-TASKS

Все запросы(/api/task/**) должны быть отправлены с basic auth.

[POST] /api/tasks/

Метод добавляет задачу в список задач авторизованного пользователя.

```
Method: POST
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
nameTask	Имя задачи	form-data	String
description	Доп. информация	form-data	String

[PUT] /api/tasks/{idTask}

Метод обновляет задачу.

```
Method: PUT
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
idTask	Номер задачи	URL шаблон	Long
nameTask	Имя задачи (с уч обновления)	form-data	String
description	Обновленное описание	form-data	String
date	yyyy-MM-dd (дата создания)	form-data	Date

[PUT] /api/tasks/completed/{idTask}

Метод переводит запрос в положение выполнено. В истории задачи ставится отметка – выполнено.

```
Method: PUT
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
idTask	Номер задачи	URL шаблон	Long

[DELETE] /api/tasks/{idTask}

Метод удаляет из БД задачу с заданным id

```
Method: DELETE
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
idTask	Номер задачи	URL шаблон	Long

[GET] /api/tasks/

Метод возвращает все задачи пользователя

```
Method: GET
```

Пример запроса

```
localhost:8080/api/tasks/
```

Пример ответа


```
[
    {
        "id": 2,
        "nameTask": "hellosome",
        "description": "wow",
        "changes": [
            {
                "id": 3,
                "dateUpdate": "2020-04-04",
                "description": "Created"
            }
        ]
    },
    {
        "id": 4,
```

```
        "nameTask": "newTask",
        "description": "Test",
        "changes": [
            {
                "id": 5,
                "dateUpdate": "2020-04-04",
                "description": "Created"
            }
        ]
    }
]
```

Выходные параметры

Имя параметра	Тип	Комментарий
id	Long	Уникальный id задачи
nameTask	String	Имя задачи
description	String	Описание задачи
(changes) id	Long	id изменения
(changes) dateUpdate	dateUpdate	Дата изменения статуса
(changes) description	description	Описание изменения статуса

Changes.description имеет 3 возможных статуса: "Created", "Updated", "Completed"

Created – задача создана

Updated – задача обновлена

Completed – задача закрыта

[GET] /api/tasks/{id}

Метод возвращает задачу по id

```
Method: GET
```

Пример ответа

```
{
    "id": 2,
    "nameTask": "hellosome",
    "description": "wow",
    "changes": [
        {
            "id": 3,
            "dateUpdate": "2020-04-04",
            "description": "Created"
        }
    ]
}
```

Посмотреть таблицу с расшифровкой параметров

[GET] /api/tasks/filter/uncompleted

```
Method: GET
```

Пример ответа


```
[
    {
        "id": 2,
        "nameTask": "hellosome",
        "description": "wow",
        "changes": [
            {
                "id": 3,
                "dateUpdate": "2020-04-04",
                "description": "Created"
            }
        ]
    },
    {
        "id": 4,
        "nameTask": "newTask",
        "description": "Test",
        "changes": [
            {
                "id": 5,
                "dateUpdate": "2020-04-04",
                "description": "Created"
            }
        ]
    }
]
```


Посмотреть таблицу с расшифровкой параметров

[GET] /api/tasks/filter/completed

Метод показывает список завершенных задач

```
Method: GET
```

Пример ответа


```
[
    {
        "id": 2,
        "nameTask": "hellosome",
        "description": "wow",
        "changes": [
            {
                "id": 3,
```

```
                "dateUpdate": "2020-04-04",
                "description": "Created"
            },
            {
                "id": 6,
                "dateUpdate": "2020-04-04",
                "description": "Completed"
            }
        ]
    }
]
```


Посмотреть таблицу с расшифровкой параметров

[GET] /api/tasks/filter/byperiod

Метод возвращает задачи с заданного периода от dateFrom до dateTo

```
Method: GET
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
dateFrom	yyyy-MM-dd	form-data	Date
dateTo	yyyy-MM-dd	form-data	Date

Пример ответа


```
[
    {
        "id": 2,
        "nameTask": "hellosome",
        "description": "wow",
        "changes": [
            {
                "id": 3,
                "dateUpdate": "2020-04-04",
                "description": "Created"
            },
            {
                "id": 6,
                "dateUpdate": "2020-04-04",
                "description": "Completed"
            }
        ]
    }
]
```


Посмотреть таблицу с расшифровкой параметров

API-LOGIN

Данные методы созданы для регистрации пользователя.

[POST] /api/login/reg

Метод создает пользователя в системе

```
Method: POST
```

Параметры запроса

Имя параметра	Комментарий	Тип параметра	Тип
userName	Имя пользователя	form-data	String
password	Пароль	form-data	String
firstName	Имя	form-data	String
secondName	Фамилия	form-data	String

Типы исключений

UserNotFoundException

Исключение выпадает в случае отсутствия пользователя с данным именем в системе.

UserWithThisNameExistsException

Если пользователь с данным userName уже зарегистрирован в системе, выпадает данное исключение.

TaskExistsException

Выпадает при создании в один день 2 задач с одинаковым именем.

TaskNotFoundException

Выпадает, если задача с данным id не найдена в системе.

TaskCantBeUpdatedException

При попытке обновить завершенное задание выпадает данная ошибка.

База данных

В качестве БД я буду использовать MySQL. БД будет состоять из 3 сущностей: user, task и changes. Всё это будет сделано через JPA/HIBERNATE. Спринг берет на себя создание пула соединений, таблиц, что сильно упрощает нам жизнь.

Состав user

1. id
2. username
3. password (hashed b2crypt)
4. firstName
5. secondName
6. Tasks (onetomany)

Состав task

1. id
2. nameTask
3. User
4. Changes (onetomany)

Состав changes

1. id
2. dateUpdate
3. description
4. task