Документация серверной части приложения

Введение

Сервервая часть приложения производит работу с базой данных приложений (использованная БД - MongoDB (https://en.wikipedia.org/wiki/MongoDB)). Она была реализована на языке Python, который является удобным языком для создания небольших приложений, на которых нет высокой нагрузки и спроса к молниеносной производительности. База данных MongoDB также проста в использовании, соответствует стандартам ACID, ко всему прочему имеет модуль для нативной работы из языка Python, поэтому решено было использовать именно её в связке и Python. Коммуникация между серверной и клиентской частями приложения производится по принципу микросервисной архитектуры, а именно через порт на localhost клиент из фронта в бекенд отправляется запрос, в бекенде происходит взаимодействие с базой данных, после чего возвращается ответ (успешно ли выполнен запрос)

Структура Базы Данных

Общие вещи

- База данных - MongoDB
- Взаимодействие с базой производится при помощи библиотеки Pymongo (https://pypi.org/project/pymongo/)
- В БД хранится 2 коллекции (коллекция пользователей и коллекция весов)
- Пароли в базе данных хранятся в хешированном виде, что является де-факто стандартом безопасности в вопросе хранения паролей
- Для хеширования использована библиотека bcrypt (https://en.wikipedia.org/wiki/Bcrypt)

Хранимые значения

1. Коллекция пользователей

```
* login
* password hash
```

2. Коллекция весов

```
* Date
* Value
* user's login
```

API

Возвращаемое значение "result with description", обозначает, что

- При удачном выполнении метод возвращает:

```
{
result: OK,
description: operation competed successfully
}
```

- При неудачном выполнении метод возвращается

```
{
result: error,
description: <описание ошибки>
}
```

Users

- [x] /users/authorize/
    - Parameters:
        1. login
        2. password
    - return: result with description
- [x] /users/delete/
    - Parameters:
        1. login
        2. password
    - return: result with description
- [x] /users/find/
    - Parameters:
        1. login
    - description: check if user with such login exists
    - return: result with description
- [x] /users/change_login/
    - Parameters:
        1. login
        2. new_login
    - description: if user with specified $login exists, his login now is gonna be changed to $new_login
    - return: result with description
- [x] /users/change_login/

    - Parameters:
        1. login
        2. new_password
    - description: if user with specified $login exists, his password now is gonna be changed to $new_password
    - return: result with description

Weights:

- [x] /weight/add/
    - Parameters:
        1. user's login
        2. date
        3. value
    - description: add a weight instance to the Database
    - return: result with description
- [ ] /weights/find/
    - Parameters:
        1. user's login
        2. start
        3. end
    - description: finds such weights, that correspond to specified $user and is associated with a date in segment [$start, $end]
    - return: JSON array with found weights
- [ ] /weights/delete
- Parameters:
    1. user's login
    2. date
    - description: deletes weight instance which corresponds to specified $user and has been taken on specified $date
    - return: result with description

Покрытие тестами

- [ ] Создать тесты