

## README.md

## Задание

### Scoring API

**Задание:** реализовать декларативный язык описания и систему валидации запросов к HTTP API сервиса скоринга. Шаблон уже есть в `api.py`, тесты в `test.py`, функционал подсчета сгора в `scoring.py`. API необычно тем, что пользователи дергают методы POST запросами. Чтобы получить результат пользователь отправляет в POST запросе валидный JSON определенного формата на локейшн `/method`.

**Disclaimer:** данное API ни в коей мере не являет собой best practice реализации подобных вещей и намеренно сделано "странно" в некоторых местах.

**Цель задания:** применить знания по ООП на практике, получить навык разработки нетривиальных объектно-ориентированных программ. Это даст возможность быстрее и лучше понимать сторонний код (библиотеки или сервисы часто бывают написаны с применением ООП парадигмы или ее элементов), а также допускать меньше ошибок при проектировании сложных систем.

**Критерии успеха:** задание **обязательно**, критерием успеха является работающий согласно заданию код, для которого написаны тесты, проверено соответствие пер8, написана минимальная документация с примерами запуска (боевого и тестов), в README, например. Далее успешность определяется code review.

### Структура запроса

```
{"account": "<имя компании партнера>", "login": "<имя пользователя>", "method": "<имя метода>", "token": "<аутентификационный токен>", "arguments": {<словарь с аргументами вызываемого метода>}}
```

- account - строка, опционально, может быть пустым
- login - строка, обязательно, может быть пустым
- method - строка, обязательно, может быть пустым
- token - строка, обязательно, может быть пустым
- arguments - словарь (объект в терминах json), обязательно, может быть пустым

### Валидация

запрос валиден, если валидны все поля по отдельности

### Структура ответа

ОК:

```
{"code": <числовой код>, "response": {<ответ вызываемого метода>}}
```

Ошибка:

```
{"code": <числовой код>, "error": {<сообщение об ошибке>}}
```

### Аутентификация:

смотри `check_auth` в шаблоне. В случае если не пройдена, нужно возвращать `{"code": 403, "error": "Forbidden"}`

## Методы

`online_score.`

### Аргументы

- `phone` - строка или число, длиной 11, начинается с 7, опционально, может быть пустым
- `email` - строка, в которой есть @, опционально, может быть пустым
- `first_name` - строка, опционально, может быть пустым
- `last_name` - строка, опционально, может быть пустым
- `birthday` - дата в формате DD.MM.YYYY, с которой прошло не больше 70 лет, опционально, может быть пустым
- `gender` - число 0, 1 или 2, опционально, может быть пустым

**Валидация аргументов** аргументы валидны, если валидны все поля по отдельности и если присутствует хотя бы одна пара `phone-email`, `first name-last name`, `gender-birthday` с непустыми значениями.

**Контекст** в словарь контекста должна прописываться запись `"has"` - список полей, которые были не пустые для данного запроса

**Ответ** в ответ выдается число, полученное вызовом функции `get_score` (см. `scoring.py`). Но если пользователь админ (см. `check_auth`), то нужно всегда отдавать 42.

```
{"score": <число>}
```

или если запрос пришел от валидного пользователя `admin`

```
{"score": 42}
```

или если произошла ошибка валидации

```
{"code": 422, "error": "<сообщение о том какое поле(я) невалидно(ы) и как именно>"}
```

### Пример

```
$ curl -X POST -H "Content-Type: application/json" -d '{"account": "horns&hoofs", "login": "h&f", "method": "online_score", "token": "55cc9ce545bcd144300fe9efc28e65d415b923ebb6be1e19d2750a2c03e80dd209a27954dca045e5bb12418e7d89b6d718a9e35af34e14e1d5bcd", "arguments": {"phone": "79175002040", "email": "stupnikov@otus.ru", "first_name": "Станислав", "last_name": "Ступников", "birthday": "01.01.1990", "gender": 1}}' http://127.0.0.1:8080/method/
```

```
{"code": 200, "response": {"score": 5.0}}
```

`clients_interests.`

### Аргументы

- `client_ids` - массив чисел, обязательно, не пустое
- `date` - дата в формате DD.MM.YYYY, опционально, может быть пустым

**Валидация аргументов** аргументы валидны, если валидны все поля по отдельности.

**Контекст** в словарь контекста должна прописываться запись `"nclients"` - количество id'шников, переданных в запрос

**Ответ** в ответ выдается словарь `<id клиента>:<список интересов>`. Список генерировать вызовом функции `get_interests` (см. `scoring.py`).

```
{"client_id1": ["interest1", "interest2" ...], "client2": [...] ...}
```

или если произошла ошибка валидации

```
{"code": 422, "error": "<сообщение о том какое поле(я) невалидно(ы) и как именно>"}
```

### Пример

```
$ curl -X POST -H "Content-Type: application/json" -d '{"account": "horns&hoofs", "login": "admin", "method": "clients_interests", "token": "d3573aff1555cd67dccf21b95fe8c4dc8732f33fd4e32461b7fe6a71d83c947688515e36774c00fb630b039fe2223c991f045f13f240913860502", "arguments": {"client_ids": [1,2,3,4], "date": "20.07.2017"}}' http://127.0.0.1:8080/method/
```

```
{"code": 200, "response": {"1": ["books", "hi-tech"], "2": ["pets", "tv"], "3": ["travel", "music"], "4": ["cinema", "geek"]}}
```

### Логирование

- скрипт должен писать логи через библиотеку `logging` в формате `'[% (asctime)s] %(levelname).1s %(message)s'` с датой в виде `'%Y.%m.%d %H:%M:%S'`. Допускается только использование уровней `info`, `error` и `exception`. Путь до логфайла указывается в конфиге, если не указан, лог должен писаться в `stdout`

### Тестирование

- Тестировать приложение мы будем после следующего занятия. Но уже сейчас предлагается писать код, что называется, `with tests in mind`.

### Распространенные проблемы:

- суть ДЗ в том, чтобы объявить поля один раз и не дублировать больше в `init` или еще где либо. Это, например (но не обязательно именно так), можно сделать с помощью метакласса, который при создании класса все `Field` соберет в массив, по которому можно будет итерироваться и валидировать

## Deadline

Задание нужно сдать через неделю. То есть ДЗ, выданное в понедельник, нужно сдать до следующего занятия в понедельник. Код, отправленный на ревью в это время, рассматривается в первом приоритете. Нарушение дедлайна (пока) не карается, пытаться сдать ДЗ можно до конца курсы. Но код, отправленный с опозданием, когда по плану предполагается работа над более актуальным ДЗ, будет рассматриваться в более низком приоритете без гарантий по высокой скорости проверки

## Обратная связь

Студент коммитит все необходимое в свой `github/gitlab` репозиторий. Далее необходимо зайти в ЛК, найти занятие, ДЗ по которому выполнялось, нажать "Чат с преподавателем" и отправить ссылку. После этого ревью и общение на тему ДЗ будет происходить в рамках этого чата.