Тестируемое приложение

- Приложение представляет собой простую страничку-визитку с формой авторизации и возможностью зарегистрировать нового пользователя. Так же есть поддержка управления пользователями через API.
- Приложение поставляется в docker-контейнере и может быть запущено только в нем.
- Приложение для своей работы обязательно требует базу данных MySQL.
- В приложении предусмотрены баги для демонстрации решения по автоматизации тестирования.

Подробное описание приложения

Страница авторизации

Содержит 2 поля для ввода (логин и пароль), а так же кнопку авторизации. Содержит кнопку для регистрации.

У всех полей должна присутствовать валидация. MAX_LENGTH_USERNAME = 16
MIN_LENGTH_USERNAME = 6
MAX_LENGTH_PASSWORD = 255
MIN_LENGTH_PASSWORD = 1

По нажатию на кнопку входа - проверяет авторизационные данные, и, если они верны - пропускает на главную страницу, иначе выдает ошибку.

Авторизация должна работать корректно, а именно:

- валидный пользователь успешно авторизовывается;
- невалидный пользователь не авторизовывается и отображается ошибка.

Страница регистрации

Содержит 7 полей для ввода (имя, фамилия, отчество (**опционально**), логин, почта и пароль с подтверждением), чекбокс принятия условий, а так же кнопка регистрации. Почта используется только для информационных целей, отправки писем или подтверждение регистрации через письмо на почту не предусмотрено.

У всех полей должна присутствовать валидация. MAX_LENGTH_NAME = 45
MIN_LENGTH_NAME = 1
MAX_LENGTH_SURNAME = 255
MIN_LENGTH_SURNAME = 1 MAX_LENGTH_MIDDLE_NAME = 255 MIN_LENGTH_MIDDLE_NAME = 1
MAX_LENGTH_USERNAME = 16
MIN_LENGTH_USERNAME = 6
MAX_LENGTH_PASSWORD = 255 # Для поля с подтверждением пароля идентично
MIN_LENGTH_PASSWORD = 1 # Для поля с подтверждением пароля идентично MAX_LENGTH_EMAIL = 64
MIN_LENGTH_EMAIL = 6

По нажатию на кнопку регистрации - проверяет введенные данные, и, если они верны и не содержат ошибок - пропускает на главную страничку, иначе выдает ошибку.

Регистрация должна работать корректно, а именно:

- все поля должны валидироваться;

- в случае успеха новый пользователь добавляется в БД;
- в случае неуспеха пользователь не добавляется в БД и выдается ошибка.

Главная страница

Содержит меню, информацию о текущем пользователе, различные ссылки на сторонние ресурсы и кнопку Logout.

Главная страница должна корректно работать, а именно:

- все ссылки должны вести на правильные ресурсы и работать одинаково;
- информация о текущем пользователе корректно отображается в правом верхем углу на любом разрешении;
- если у пользователя удалось получить VK ID, то данный идентификатор отображается аналогично в правом верхнем углу;
- в нижней части страницы отображается случайный мотивационный факт о python;
- кнопка Logout должна успешно разлогинивать пользователя.

Конфигурация приложения

Приложение требует указание конфиг файла через опцию `--config=<path/to/config>`. Конфиг файл имеет обязательные и необязательные параметры. Все опции в конфиг файле задаются в формате `KEY = VALUE`. Количество пробелов не имеет значение. Поддерживаемые опции:

- Опционально:
  - `APP_HOST = '<app_host>'` # хост, на котором будет запущено приложение ('0.0.0.0' default)
  - `APP_PORT = '<app_port>'` # порт, на котором будет запущено приложение ('8080' default)
  - `MYSQL_HOST = '<mysql_host>'` # хост mysql, к которому будет подключаться приложение ('localhost' default)
  - `MYSQL_PORT = '<mysql_port>'` # порт mysql, к которому будет подключаться приложение ('3306' default)
  - `MYSQL_DB = '<mysql_db_name>'` # имя базы, к которой будет подключаться приложение ('vkeducation' default)
- Обязательно
  - `VK_URL = '<vk_url_host>:<vk_url_port>'` # хост и порт VK API, у которого приложение будет запрашивать VK ID пользователя.

Описание базы данных

- БД должна работать и быть доступна для приложения по `mysql://MYSQL_HOST:MYSQL_PORT/MYSQL_DB`
- `MYSQL_DB` должна быть создана до старта приложения
- В БД присутствует единственная таблица `test_users` следующего формата

```
CREATE TABLE `test_users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `surname` varchar(255) NOT NULL,
  `middle_name` varchar(255) DEFAULT NULL,
  `username` varchar(16) DEFAULT NULL,
  `password` varchar(255) NOT NULL,
```

```
  `email` varchar(64) NOT NULL,
  `access` smallint DEFAULT NULL,
  `active` smallint DEFAULT NULL,
  `start_active_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`),
  UNIQUE KEY `ix_test_users_username` (`username`)
);
```

- Описание полей (**обращайте внимание на размер поля**):
  - `id` - идентификатор пользователя;
  - `name` - имя пользователя;
  - `surname` - фамилия пользователя
  - `middle_name` - отчество пользователя (опционально);
  - `username` - имя пользователя для авторизации, **не может повторяться**;
  - `password` - пароль пользователя;
  - `email` - почтовый адрес пользователя, не может повторяться;
  - `access` - флаг доступа для пользователя (0|1);
  - `active` - флаг присутствия авторизационного пользователя на странице (0|1);
  - `start_active_time` - время последнего успешного входа пользователя.
- Приложение может быть запущено с опцией `--setup`, которая выполняет инициализацию БД, самостоятельно создает схему таблицу `test_users`. С данной опцией само приложение не запускается. При необходимости заполнить эту таблицу пользователями вы можете воспользоваться --init-file в mysql с предварительно подготовленными вами файлами.
- Приложение работает с базой от пользователя `test_qa` с паролем `qa_test`.
- У данного пользователя должен быть доступ в базу `MYSQL_DB` и таблицу `test_users`.

Логика работы приложения

- При успешной регистрации пользователя все регистрационные данные добавляются в таблицу `test_users`. Флаг `access` по умолчанию выставляется в `1`.
- При неуспешной авторизации на странице отображаются корректные всплывающие сообщения с ошибками. Приложение возвращает корректные HTTP коды ответа в случае ошибок.
- Доступ на главную страницу разрешен пользователю, у которого поле `access` выставлено в `1`.
- Поле `access` проверяется на каждый запрос, если доступ запрещен - пользователь деавторизовывается.
- Любой вход пользователя на главную страницу проставляет поле `active` в `1`, а `start_active_time` во время входа.
- Любой выход пользователя проставляет поле `active` в `0`.

Всю отладочную информацию приложение выводит в stdout.

Получение VK ID пользователя

- При запросе главной страницы приложение делает запрос в `VK_URL` для получения VK ID текущего пользователя и отображает его в правом верхнем углу. Если по каким-то причинам получить идентификатор пользователя не удалось - VK ID не отображается.

- Формат запроса:
  - `GET http://<VK_URL>/vk_id/<username>`, где
    - `VK_URL` - опция из конфига;
    - `username` - текущий авторизованный пользователь.
- Формат ожидаемого ответа:
  - Если пользователь найден

    ```
    Status: 200 OK
    Content-Type: application/json
    Response: {'vk_id': <vk_id>}
    ```

  - Если пользователь не найден

    ```
    Status: 404 Not Found
    Content-Type: application/json
    Response: {}
    ```

- Опция `VK_URL` обязательна в конфиге.
- Сервис должен работать **до старта приложения**.

Описание API

На все запросы должно быть корректное сообщение/статус код, обращайте на это внимание.

- Сущность создана - код 201;
- Сущность существует/не изменилась - код 304;
- Сущность удалена - код 204;
- Плохой запрос - код 400;
- Сущности не существует - код 404;
- Пользователь не авторизован - код 401;
- Действие выполнено - код 200.

Статус 500 недопустим. **Ошибка 'Internal Server Error' недопустима и является багом.**

Добавление пользователя

Формат запроса

```
POST http://<APP_HOST>:<APP_PORT>/api/user
Content-Type: application/json
Body:
{
    "name": "<name>",
    "surname": "<surname>",
    "middle_name": "<middle_name>",
    "username": "<username>",
    "password": "<password>",
    "email": "<email>"
}
```

- Поле 'middle_name' - опциональное;
- **Поля должны валидироваться аналогично форме регистрации;**
- В случае успеха пользователь добавляется в БД.

Удаление пользователя

Формат запроса

```
DELETE http://<APP_HOST>:<APP_PORT>/api/user/<username>
```

- В случае успеха пользователь удаляется из БД.

Смена пароля пользователя

Формат запроса

```
PUT http://<APP_HOST>:<APP_PORT>/api/user/<username>/change-password
Content-Type: application/json
Body:
{
    "password": "<new password>"
}
```

- В случае успеха у пользователя меняется пароль;
- Новый пароль не может совпадать с паролем из БД.

Блокировка пользователя

Формат запроса

```
POST http://<APP_HOST>:<APP_PORT>/api/user/<username>/block
```

- В случае успеха пользователю проставляется `access = 0` в БД.

Разблокировка пользователя

Формат запроса

```
POST http://<APP_HOST>:<APP_PORT>/api/user/<username>/accept
```

- В случае успеха пользователю проставляется `access = 1` в БД.

Статус приложения

Формат запроса

```
GET http://<APP_HOST>:<APP_PORT>/status
```

Формат ответа

```
Status: 200 OK
Content-Type: application/json
Body:
{
    "status": "ok"
}
```

- Сигнализирует о том, что приложение запустилось и готово к работе.

Пример запуска приложения

Добавить созданный образ в docker images можно командой `docker load -i myapp`.

Последовательность действий:

- Запустить БД в отдельном контейнере:
  - `MYSQL_CONTAINER_NAME` - произвольное имя контейнера с базой;
  - `ROOT_PASSWORD` - произвольный пароль для пользователя root
  - docker run --name `<MYSQL_CONTAINER_NAME>` -e MYSQL_ROOT_PASSWORD=`ROOT_PASSWORD` -p 3306:3306 -d mysql:latest
- Зайти в поднятую БД под пользователем root и паролем `ROOT_PASSWORD`
  - Создать произвольную базу данных `<DB_NAME>`;
  - Добавить пользователя `test_qa` с паролем `qa_test` и дать ему необходимые права на БД и таблицу.
- Написать mock-сервис для получения VK_API, собрать с ним docker-образ `vk_api` и запустить
  - `VK_API_CONTAINER_NAME` - произвольное имя контейнера с VK API
  - `VK_API_PORT` - произвольный порт, на котором поднят mock-сервис VK API
  - docker run --name `<VK_API_CONTAINER_NAME>` -p `<VK_API_PORT>`:`<VK_API_PORT>` -d vk_api:latest
  - Если у вас проблемы на этом этапе - переходите дальше.
- Создать произвольный конфиг файл `<CONFIG_FILE>` в произвольной директории `<PATH_TO_CONFIG>`, заполнить все поля:
  - APP_HOST = '0.0.0.0'
  - APP_PORT = `<APP_PORT>` - произвольный порт, на котором будет поднято приложение
  - MYSQL_HOST = `<MYSQL_DOCKER_HOSTNAME>` - имя контейнера с базой
  - MYSQL_PORT = 3306
  - MYSQL_DB = `<DB_NAME>` - имя созданной БД
  - VK_URL = `<VK_API_HOSTNAME>:<VK_API_PORT>`
- Запустить приложение из приложенного к заданию образа myapp:
  - `<DOCKER_PATH_TO_CONFIG>` - директория внутри контейнера, куда будет маунтиться `<PATH_TO_CONFIG>`
  - `<APP_PORT>` - порт приложения, указанный в конфиге
  - `<MYSQL_CONTAINER_NAME>` - имя контейнера с базой
  - `<MYSQL_DOCKER_HOSTNAME>` - hostname базы, который будет доступен внутри докера при линковке
  - `<VK_API_CONTAINER_NAME>` - имя контейнера с VK API
  - `<VK_API_HOSTNAME>` - hostname VK API, который будет доступен внутри докера при линковке
  - docker run -v `<PATH_TO_CONFIG>`:`<DOCKER_PATH_TO_CONFIG>` -p `<APP_PORT>`:`<APP_PORT>`
  - -link `<MYSQL_CONTAINER_NAME>`:`<MYSQL_DOCKER_HOSTNAME>`
  - -link `<VK_API_CONTAINER_NAME>`:`<VK_API_HOSTNAME>` myapp /app/myapp --config=`<DOCKER_PATH_TO_CONFIG>/<CONFIG_FILE>`
  - Пример конфига и команды запуска приведен в репозитории. Указанный способ запуска выше - стандартный, но подразумевается его переработка под docker-compose.

После всех выполненных действий появится возможность зайти на веб-приложение из браузера по `http:<APP_HOST>:<APP_PORT>` и проводить тестирование.