|  |
| --- |
| Никита Поглазов & Павел Савин: |
| Проект WebServer API. Мессенджер Webby |
| Пояснительная записка |

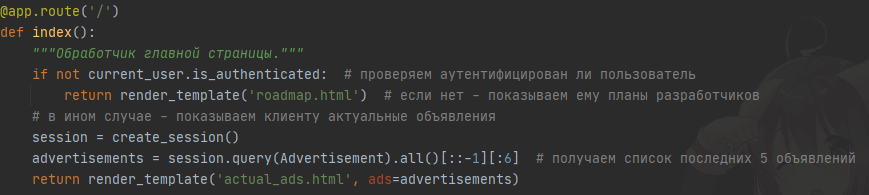
|  |
| --- |
| 20.04.2021 |

## Цель проекта.

Цель нашего проекта – практика в использовании веб-фреймворка *Flask*, включающим в себя набор инструментов для работы с *WSGI* (стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером) - *werkzeug* и шаблонизатор *Jinja2*. Для удобного взаимодействия с СУБД через код используется *sqlalchemy*, который позволяет пользоваться технологией *ORM* (технология программирования, которая связывает базы данных с концепциями объектно-ориентированного программирования).

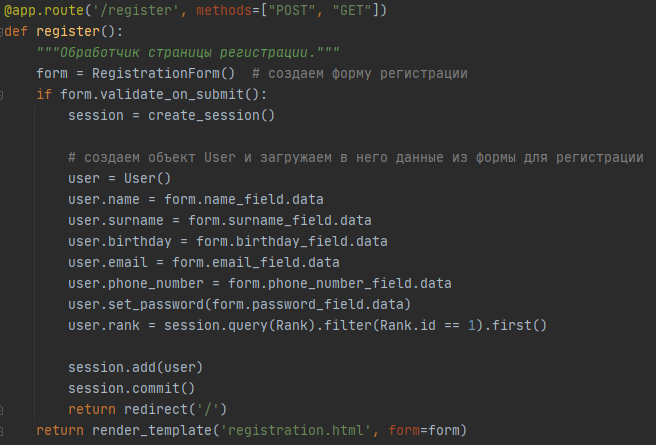
### Структура программы

Основную часть кода занимают *обработчики* – функции, которые активируются при переходе клиента на определенный адрес веб-сервера.

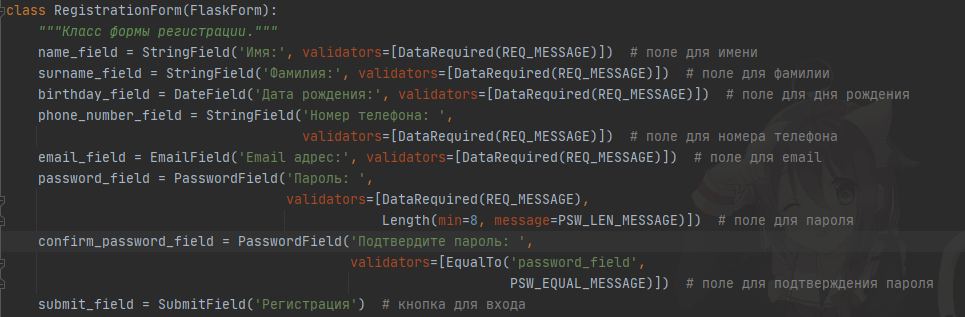
Пример простого обработчика 

Все что делают обработчики – обрабатывают запрос пользователя и отправляют обработанные данные в шаблон. Шаблон – html-файл, который помимо самой разметки использует синтаксис Jinja2 для выполнения простых операций, например циклов и условий, что позволяет в кратчайшие сроки сверстать гибкую страницу, которая будет отображать все данные как нам угодно.

Есть и более сложные обработчики, которые используют еще одну возможность *Flask* (модуля *flask-wtf*) – *формы*. Формы это класс, который содержит в себе поля для данных, которые будет заполнять пользователь на странице. Особенность таких обработчиков в том, что помимо получения данных с сервера (метод “*GET*”), они реализуют отправку данных на сервер (метод “*POST*”). Подобные обработчики используются для регистрации и входа на сайт, редактирования профиля, создания и изменения объявлений и даже отправки сообщений.

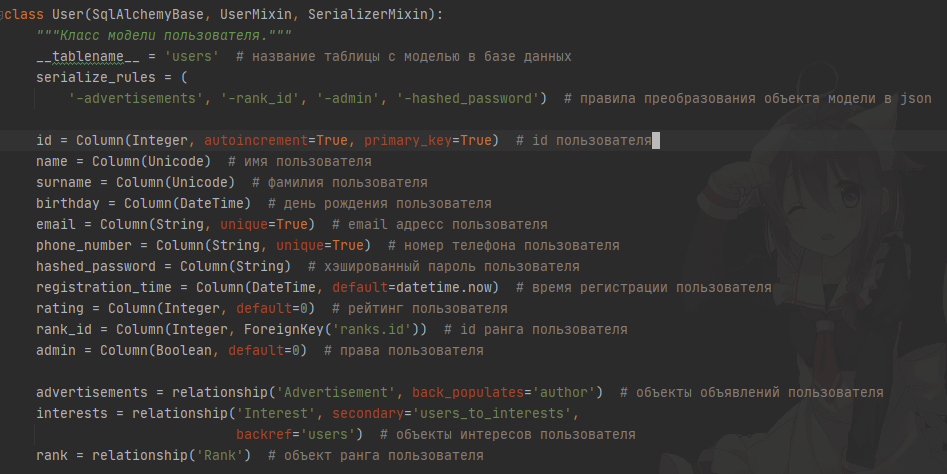
Пример сложного обработчика

Поля форм разделены на различные типы вводимых данных и содержат в себе название поля и валидаторы – набор функций/классов, которые проверяют введенные данные (например, DataRequired проверяет наличие данных в поле).

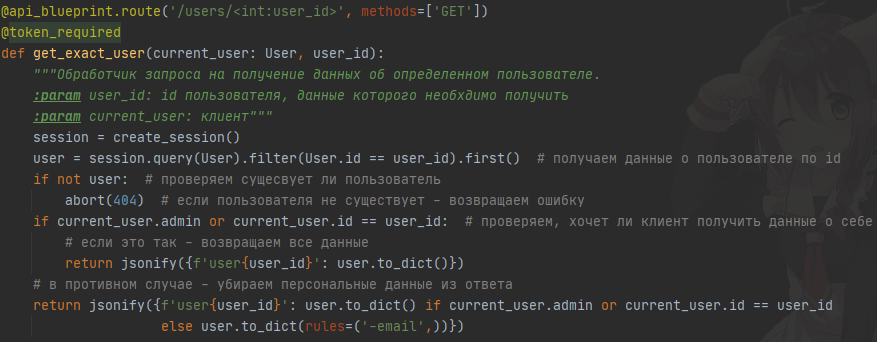
Пример формы 

Для аутентификации пользователя используется модуль *flask-login*, который сохраняет данные о клиенте в переменной *current\_user* с помощью класса UserMixin, от которого наследуется класс пользователя*.* Также, с помощью декоратора *login\_required* и поля *current\_user.is\_authenticated* мы можем отделить данные, которые может получать аутентифицированный и не аутентифицированный пользователи.

Для удобного взаимодействия с базой данных через код *sqlalchemy* позволяет создавать *модели*. *Модели* – классы, которые напрямую связаны с таблицей в СУБД. Столбцы в модели идентичны тем, что находятся в связанной базе данных, что позволяет, имея объект модели, получить доступ ко всем данным о ней. В модели также присутствуют отношения (relationship), которые позволяют связать одну модель с другой по внешнему ключу.

Пример модели

Для более продвинутых пользователей было создано API (описание методов, которыми одна компьютерная программа может взаимодействовать с веб-сервером). Все методы API сгруппированы в Blueprint, что позволяет добавить префикс к адресу API. Методы API по структуре идентичны обработчикам. Единственное их отличие заключается в том, что API не возвращают клиенту какую либо страницу, они возвращают json-объект – набор данных, который по структуре идентичен словарю. Для преобразования моделей базы данных в json объект используется класс SerializerMixin из модуля sqlalchemy-serializer, от которого наследуется класс модели. Этот класс добавляет модели метод .to\_dict() который преобразует все данные модели в словарь.

Пример метода API 

Для защиты данных пользователя была введена система токенов из модуля *jwt*. Перед тем как начать работу с API пользователю необходимо отправить свои аутентификационные данные на адрес /api/login, в котором будет создан объект его профиля и закодирован, вместе со “сроком годности” токена в 1 час, в токен.

В свою очередь, для проверки токена был написан декоратор token\_required, который проверяет токен и декодирует его, чтобы получить данные о клиенте.