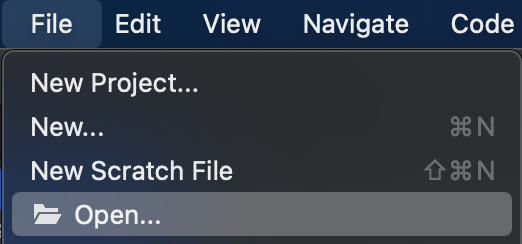
1. **Создание директории**

* Открываем терминал
* Вводим следующие команды:
* **mkdir MyWebApp** – создаем директорию с именем "MyWebApp"
* **cd MyWebApp** – переходим в директорию

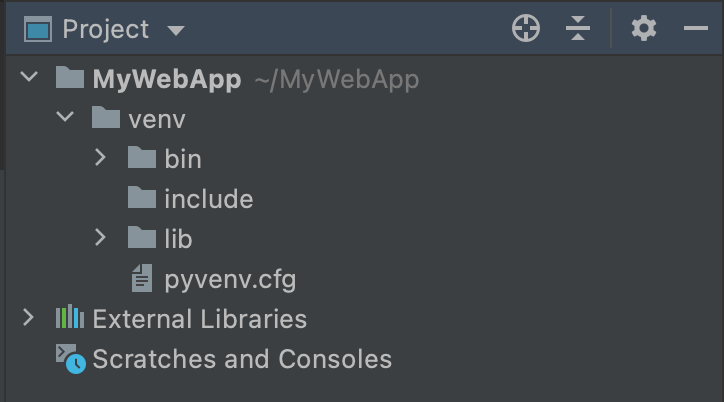
* После создания директории создаем и активируем виртуальную среду Python

1. **Открываем директорию в PyCharm:**

* Открываем нашу директорию:

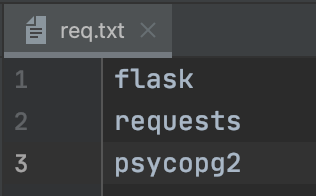


* Структура проекта на данном этапе выглядит следующим образом:



1. **Устанавливаем необходимые инструменты для дальнейшей работы:**

* Создаем файл req.txt
* Заполняем файл req.txt названиями необходимых нам инструментов



* Устанавливаем инструменты в терминале через менеджера пакетов Python pip3:
* **pip3 install -r req.txt**

1. **Создаем приложение:**
   * Создаем файл app.py

* Импортируем необходимые инструменты

import requests

from flask import Flask, render\_template, request

import psycopg2

* Создаем приложение

app = Flask(\_\_name\_\_)

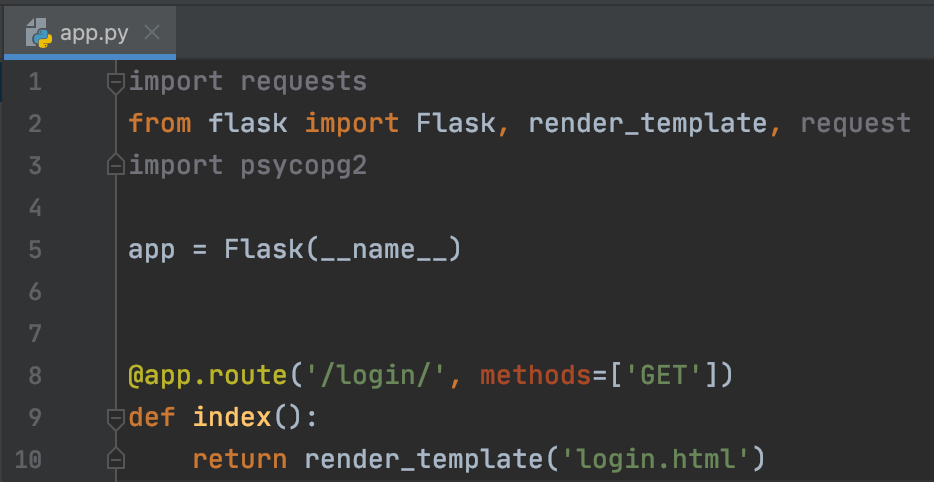
* Создаем первый декоратор

@app.route('/login/', methods=['GET'])

def index():

return render\_template('login.html')

На данном этапе файл app.py должен выглядеть следующим образом:



* Создаем директорию templates
* Внутри этой директории создаем файл login.html
* Удаляем содержимое файла login.html и вставляем текст разметки

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Login</title>

</head>

<body>

<form action="" method="post">

<p>

<label for="username">Username</label>

<input type="text" name="username">

</p>

<p>

<label for="password">Password</label>

<input type="password" name="password">

</p>

<p>

<input type="submit">

</p>

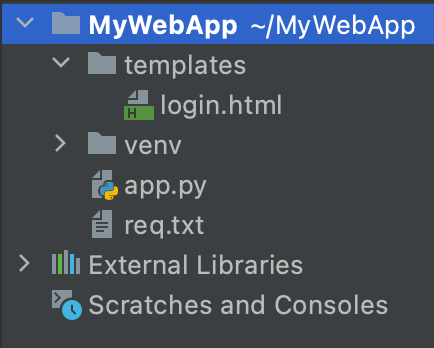
</form>

</body>

</html>

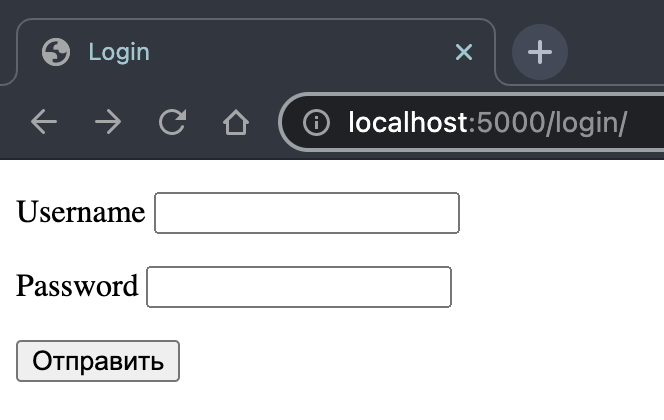
* Запускаем приложение

На данном этапе структура проекта выглядит следующим образом:



В терминале в папке MyWebApp запускаем команду

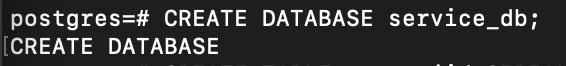
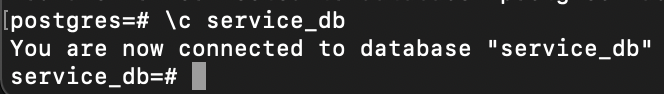
* **flask run**
* Переходим по ссылке <http://localhost:5000/login/>

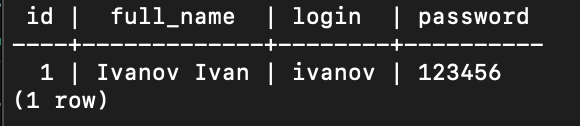


1. **Устанавливаем PostgreSQL**
   * Обновляем список пакетов

* **sudo apt update**
* Устанавливаем пакеты Postgres и contrib
* **sudo apt install postgresql postgresql-contrib**
* Запускаем сервер PostgreSQL
* **sudo -u postgres psql**

1. **Создаем базу данных**
   * Создаем базу данных

* **CREATE DATABASE service\_db;**
  + Подключаемся к базе данных
* **\c service\_db**
  + Создаем схему
* **CREATE SCHEMA service;**
  + Создаем таблицу пользователей
* **CREATE TABLE service.users (id SERIAL NOT NULL, full\_name VARCHAR NOT NULL, login VARCHAR NOT NULL, password VARCHAR NOT NULL);**
  + Заполняем таблицу пользователей
* **INSERT INTO service.users (full\_name, login, password) VALUES ('<Полное имя пользователя>','<логин>', '<пароль>');**
* Проверяем заполнение таблицы
* **SELECT \* FROM service.users;**

****

1. **Модернизируем приложение:**
   * В файл app.py добавляем подключение к базе данных сразу после строки "app = Flask(\_\_name\_\_)"

conn = psycopg2.connect(database="service\_db",

user="<логин создателя БД>",

password="",

host="localhost",

port="5432")

* Добавляем курсор для обращения к базе данных

cursor = conn.cursor()

* Создаем еще один декоратор

@app.route('/login/', methods=['POST'])

def login():

username = request.form.get('username')

password = request.form.get('password')

cursor.execute("SELECT \* FROM service.users WHERE login=%s AND password=%s", (str(username), str(password)))

records = list(cursor.fetchall())

return render\_template('account.html', full\_name=records[0][1])

* Создаем файл account.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

<form action="" method="post">

{% if full\_name %}

<p>Hello, {{full\_name}}! </p>

{% endif %}

</p>

</form>

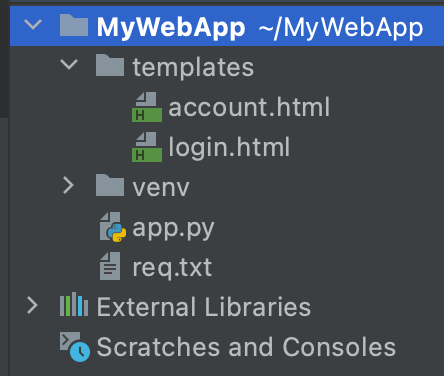
</head>

<body>

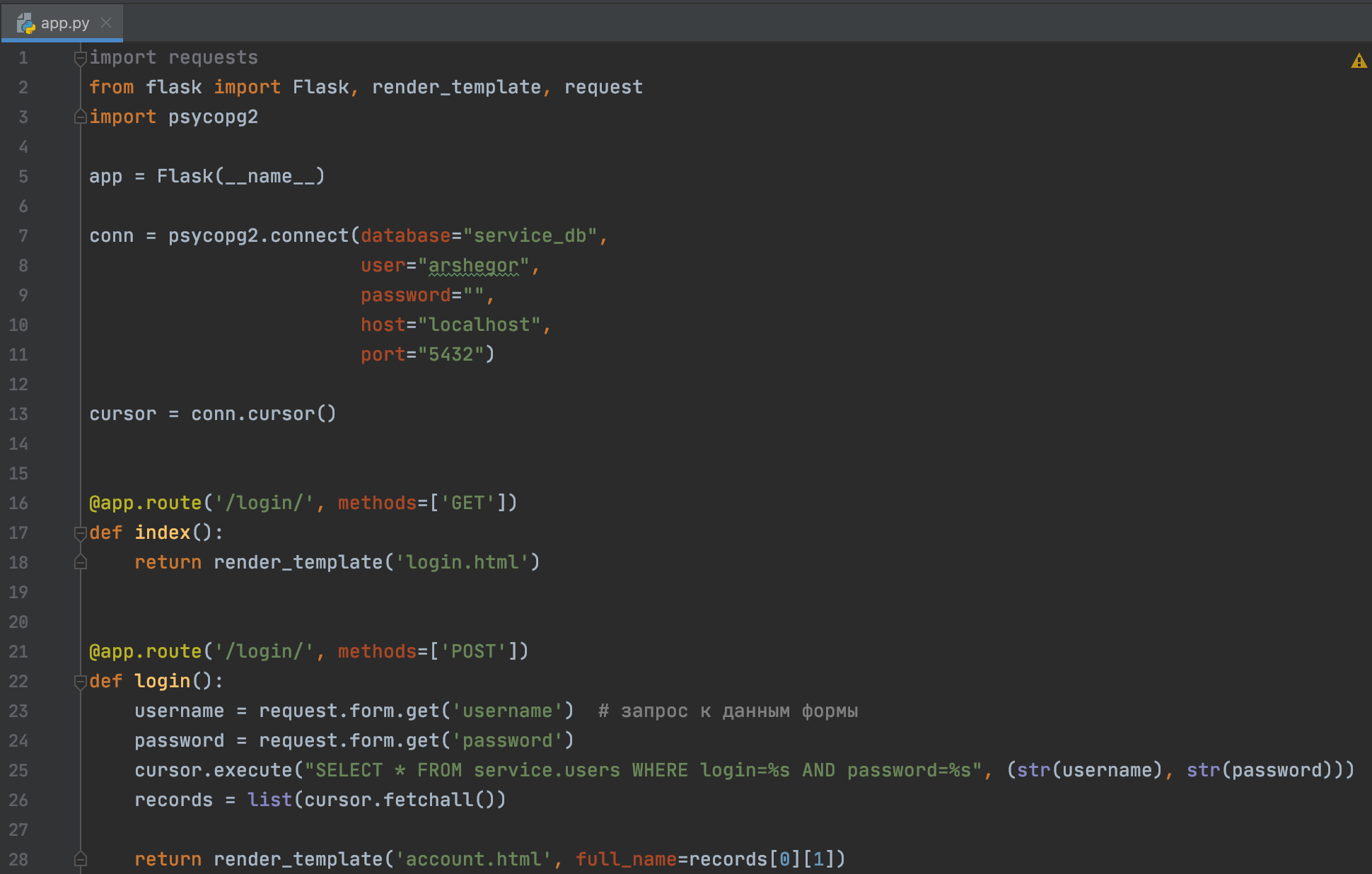
</body>

</html>

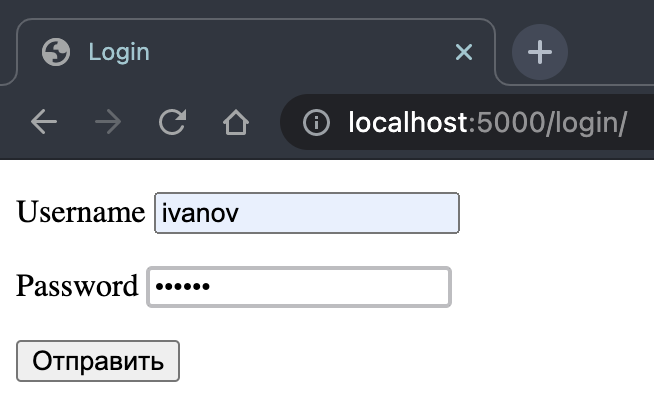
Структура проекта выглядит следующим образом:

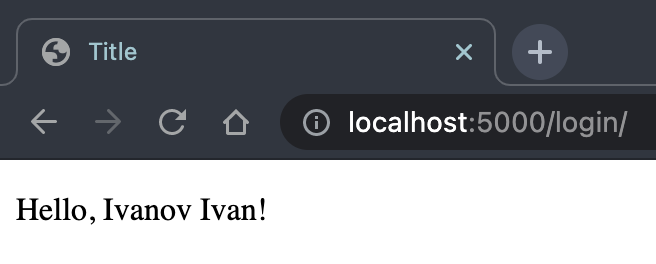


Файл app.py выглядит следующим образом:



Посмотрим что получилось





1. **Домашнее задание**
   * Дополнить таблицу users в базе данных до 10 пользователей
   * Сделать обработку исключения на ввод пустого логина и пароля
   * Сделать обработку исключения на отсутствие пользователя в базе данных
   * Вывести на странице аккаунта помимо имени пользователя его логин и пароль