**Цель:** Изучить взаимодействие языка Python и БД Postgres. Создать приложение по регистрации в Базе Данных.

**Ход Работы:**

Самым первым делом занимаемся импортом всех нужных нам библиотек.

|  |
| --- |
| import requests |
|  | from flask import Flask, render\_template, request, redirect |
|  | import psycopg2 |

C Flask мы уже знакомы по предыдущим работам, а вот с psycopg2 работаем впервые. В сам проект можно скачать psycopg2- binary, если обычная не хочет работать, а обращаться как к обычной. Эта библиотека используется для взаимодействия с библиотекой PostgresSQL.

Создаём приложение

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

Теперь поработаем с самой базой данных.

Устанавливаем пакеты Postgres и contrib

sudo apt install postgresql postgresql-contrib

Запускаем сервер PostgreSQL

sudo -u postgres psql

После подключения создаём базу данных

**CREATE DATABASE service\_db;**

Подключаемся к базе данных

\c service\_db

Создаем схему

CREATE SCHEMA service;

Создаем таблицу пользователей с параметрами, которые будут в этой таблице.

CREATE TABLE service.users (id SERIAL NOT NULL, full\_name VARCHAR NOT NULL, login VARCHAR NOT NULL, password VARCHAR NOT NULL);

Заполняем таблицу пользователей (все параметры пользователей вне ковычек).

INSERT INTO service.users (full\_name, login, password) VALUES ('<Полное имя пользователя>','<логин>', '<пароль>');

Проверяем заполнение таблицы

SELECT \* FROM service.users;

Затем добавляем подключение к базе данных в приложение.

|  |
| --- |
| conn = psycopg2.connect(database="service\_db", |
|  | user="postgres", |
|  | password="123456789", |
|  | host="localhost", |
|  | port="5432") |

Для database задаём название нашей БД. В user задаём имя админа (по-умолчанию создаётся postgres) и в password – пароль (изначально пароль может быть не задан, необходимо задать его, подключившись к БД и введя команду \password).

Добавляем курсор для обращения к БД

cursor = conn.cursor()

Создаём декоратор

|  |
| --- |
| @app.route('/login/', methods=['GET']) |
|  | def index(): |
|  | return render\_template('login1.html') |

Создаём html-документ login1 и заполняем его заранее добавив кнопку для регистрации, к которой при подключим её функции позже.

|  |
| --- |
| <!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" value="login" name="login"> |
|  | <input type="submit" value="registration" name="registration"> |
|  | </p> |
|  |  |
|  | </form> |
|  |  |
|  | </body> |
|  | </html> |

Далее, в питоновском файле создаём ещё один декоратор.

|  |
| --- |
| @app.route('/login/', methods=['POST', 'GET']) |
|  | def login(): |
|  | if request.method == 'POST': |
|  | if request.form.get("login"): |
|  | warning = '' |
|  | username = request.form.get('username') |
|  | password = request.form.get('password') |
|  | if password == '' and username == '': |
|  | warning = 'You have not entered username and password' |
|  | elif password == '': |
|  | warning = 'You have not entered password' |
|  | elif username == '': |
|  | warning = 'You have not entered username' |
|  | else: |
|  | cursor.execute("SELECT \* FROM service.users WHERE login=%s AND password=%s", |
|  | (str(username), str(password))) |
|  | records = list(cursor.fetchall()) |
|  |  |
|  | if not records: |
|  | warning = 'Sorry, there is no such user in DataBase' |
|  | return render\_template('account.html', warning=warning) |
|  |  |
|  | else: |
|  | return render\_template('account.html', full\_name=records[0][1], warning=warning) |
|  |  |
|  | return render\_template('account.html', warning=warning) |
|  |  |
|  | elif request.form.get("registration"): |
|  | return redirect("/registration/") |
|  |  |
|  | return render\_template('login1.html') |

Создаём множественное условие if-else, чтобы избежать ошибок, когда пользователь не ввёл одно из полей или когда такого пользователя в БД нет. Для хранения сообщения об ошибке (чтобы потом передать её пользователю) создаём переменную warning, которую обнуляем как только получаем запрос на сервере (requsest.method=post). Также добавляем перенаправление на registration в случае нажатия соответствующей кнопки.

Создаём ещё один html-файл account, куда будем перенаправлять пользователя после ввода данных для входа.

|  |
| --- |
| <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <title>Title</title> |
|  | <form action="" method="post"> |
|  | {% if warning %} |
|  | <p>{{warning}}</p> |
|  | {% else %} |
|  | {% if full\_name %} |
|  | <p>Hello, {{full\_name}}! </p> |
|  | {% endif %} |
|  | {% endif %} |
|  |  |
|  | </form> |
|  | </head> |
|  | <body> |
|  |  |
|  | </body> |
|  | </html> |

Используем условные операторы для вывода одного из нескольких сообщений в зависимости от наличия или отсутствия ошибки.

Теперь создаём html-файл registration.

|  |
| --- |
| <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <title>Registration</title> |
|  | </head> |
|  | <body> |
|  | <form action="" method="post"> |
|  | <p> |
|  | <label for="name">Your name:</label> |
|  | <input type="text" name="name"> |
|  | </p> |
|  | <p> |
|  | <label for="login">Login:</label> |
|  | <input type="text" name="login"> |
|  | </p> |
|  | <p> |
|  | <label for="password">Password:</label> |
|  | <input type="password" name="password"> |
|  | </p> |
|  | <p> |
|  | <input type="submit" value="Sign In"> |
|  | </p> |
|  | </form> |
|  | {{warning}} |
|  | </body> |
|  | </html> |

Не забываем добавить сообщения о пустых полях, которые необходимо обязательно заполнить.

Добавляем в питоновский файл новый декоратор

|  |
| --- |
|  |
| @app.route('/registration/', methods=['POST', 'GET']) |
|  | def registration(): |
|  | if request.method == 'POST': |
|  | name = request.form.get('name') |
|  | login = request.form.get('login') |
|  | password = request.form.get('password') |
|  | warning = '' |
|  |  |
|  | if name == '' or login == '' or password == '': |
|  | warning = 'Please, fill every feild' |
|  | else: |
|  | cursor.execute('INSERT INTO service.users (full\_name, login, password) VALUES (%s, %s, %s);', |
|  | (str(name), str(login), str(password))) |
|  | conn.commit() |
|  |  |
|  | return redirect('/login/') |
|  |  |
|  | return render\_template('registration.html', warning=warning) |
|  |  |
|  |  |

Не забываем обнулять сообщение об ошибке в случае пустых полей, а также перенаправление на login после завершения регистрации, чтобы пользователь мог сразу войти.

В конце питоновского файла добавляем запуск приложения.

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_": |
|  | app.run() |

Вывод: В ходе проделанной работы мы изучили взаимодействие языка Python и БД Postgres, а также создали простой сайт с регистрацией в БД.