**Лабораторная работа №4**

Тема: «Создание веб-приложения для отображения реляционной базы данных с использованием python-фреймворка Flask»

**Цель работы**

Целью данной лабораторной работы является создание веб-приложения с использованием фреймворка Flask для отображения данных из реляционной базы данных в виде HTML таблицы.

**Задачи**

1. Создание базы данных SQLite с данными о сотрудниках офиса.
2. Написание кода на Python для подключения к базе данных и получения данных.
3. Реализация веб-страницы с использованием HTML и Jinja2 для отображения данных в виде HTML таблицы.
4. Запуск Flask приложения и проверка работоспособности.

**Используемые технологии**

Python

Flask

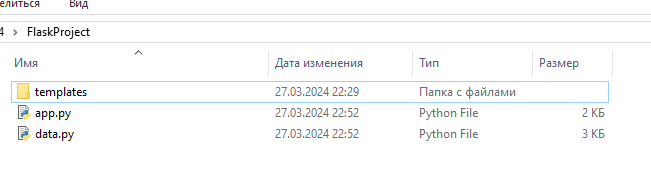
HTML

SQLite

**Подготовительный этап**

Предварительно, необходимо скачать все необходимые компоненты и библиотеки, в том числе, сам интерпретатор Python:

1. Переходим на <https://www.python.org/downloads/> и выбираем подходящую версию
2. Выполняем установку Python(не забываем поставить галочку на добавлении в PATH)
3. Устанавливаем Flask. Сделать это можно путем выполнения команды **pip install Flask** в windows-консоли(cmd.exe) или PowerShell
4. Далее необходимо выбрать директорию, где будет лежать сам проект, а затем создать внутри файл с названием **app.py**, который будет выступать в роли приложения
5. Помимо этого, для хранения шаблона веб представления, необходимо создать папку **templates** в директории проекта, и в ней файл **index.html**.
6. Также, выделим отдельный файл с кодом для создания и заполнения базы данных информацией. Для этого создадим файл в корневой директории и назовем его **data.py**

Рисунок 1 – Содержимое директории(папки) проекта

**Ход работы**

Создание базы данных

Для этой лабораторной требуется, чтобы была создана база данных SQLite. В нашем случае, содержащая информацию о студентах группы университета, которая представлена в виде следующей структуры:

|  |  |  |
| --- | --- | --- |
| stud\_group | | |
| Поле | Тип данных | Описание |
| id | INTEGER | Уникальный идентификатор студента |
| name | TEXT | Имя студента |
| age | INTEGER | Возраст студента |
| gender | TEXT | Пол студента |
| subject | TEXT | Основной предмет студента |
| course | INTEGER | Курс студента |

Начинаем работу с файла **data.py,** в котором реализуем подключение к базе и добавления данных.

Для начала подключим встроенный модуль питона sqlite3:



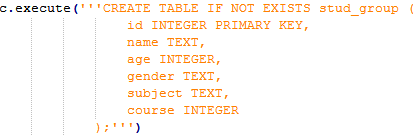
Создадим дескриптор подключения к базе. В данном примере база будет называется **database.db**:



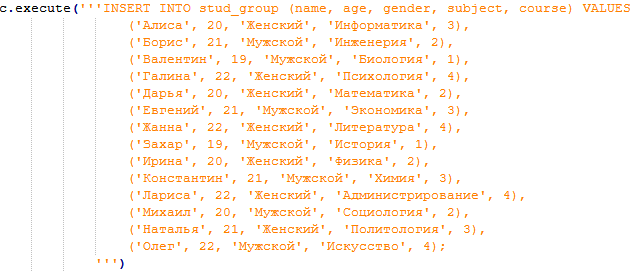
Создаем переменную, для хранения курсора, необходимого для выполнения SQL-запросов к базе:



После того, как мы установили соединение, используя курсор - выполняем запрос на создание таблицы с нашей структурой, если она еще не создана, следующим кодом:



Следом, выполняем запрос на добавление в таблицу нашей информации:

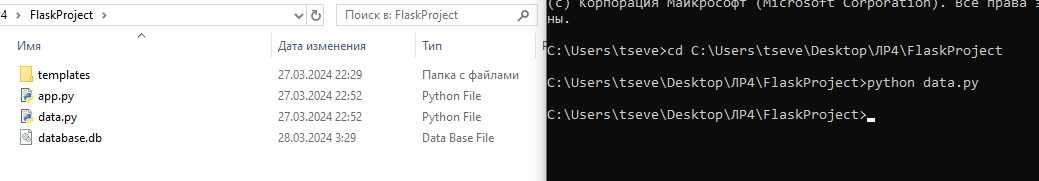


Используя ранее созданный дескриптор для хранения нашего соединения – выполняем операцию внесения изменений(**commit()**) и закрываем соединение:



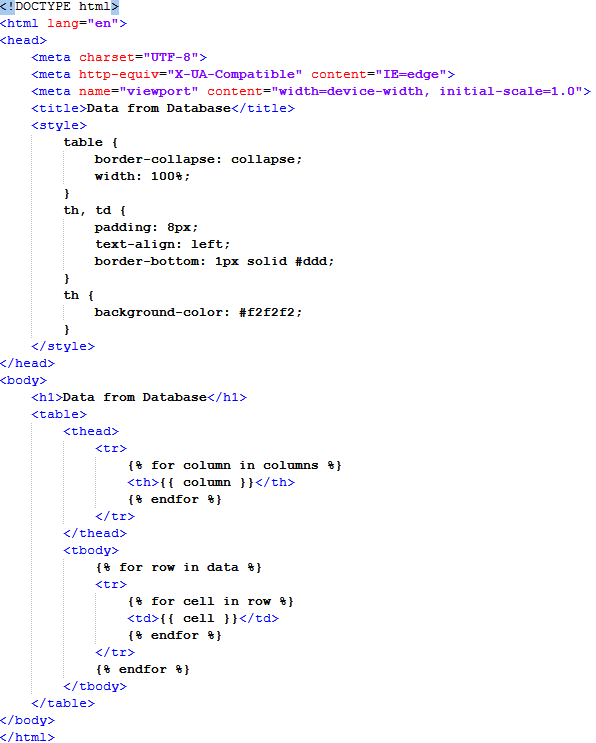
Скрипт с созданием базы и добавлением в нее данных – готов. Теперь, мы можем выполнить его, в результате чего увидим вновь появившийся файл с нашей базой в корневой папке проекта.

Выполнение скрипта можно инициировать командой **python data.py** в консоли:



Создание приложения

В файле **app.py**, который является основным в данном проекте – необходимо добавить функционал для выполнения задачи. Но перед этим определим простейшую структуру HTML-файла проекта, используя технологию шаблонизации Jinja2: добавим стиль, базовые элементы HTML-страницы и таблицу, содержащую шаблоны:



После определения вида страницы, можно преступить к заключительному этапу написания кода, заполнив файл **app.py**.

В первую очередь, подключим ранее используемую библиотеку sqlite3 для подключения к уже существующей базе и работой с ней:



Добавим модули библиотеки Flask и инициализируем приложение:



Далее создадим ключевую функцию, которая будет подключаться к базе, и извлекать из нее информацию, в том числе и информацию о самой таблице, а затем закрывать подключение с возвратом результирующих данных.

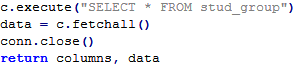
Объявим функцию, назвав ее, например, **get\_data()** и сразу добавим подключение с курсором для извлечения данных:



Так как необходимо автоматически генерировать необходимое кол-во столбцов таблицы – получим информацию о заголовке таблицы (**PRAGMA table\_info()**) в переменную **columns**, используя цикл по результирующему списку и выполним запрос на выбор всех строк таблицы:



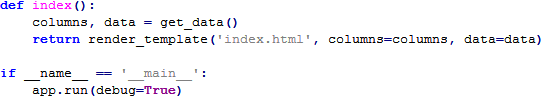
Запишем результирующий список строк в переменную **data** и закроем соединение с передачей двух переменных **columns** и **data**, содержащих: результирующие строки и названия колонок:



Создадим маршрут для отображения данных на веб-странице. Во Flask это можно сделать при помощи декоратора **@app.route()**. Укажем корневой каталог:

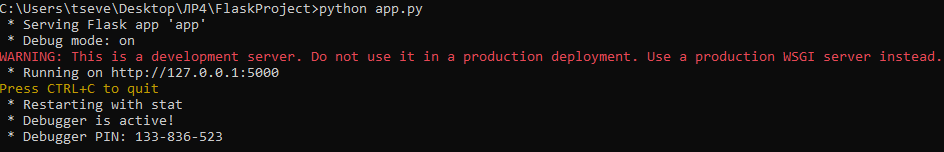


Завершающим действием будет создание функции отображения представления(веб-страницы), с уже полученными данными для передачи их в конечную веб-страницу **index.html** и добавлением запуска приложения:



Запуск приложения

Для того, чтобы убедиться, в работоспособности приложения - необходимо запустить **app.py** командой **python app.py**, используя консоль. В результате чего выведется сообщение с тем, что сервер запущен и с какими параметрами запуска, также будет указана ссылка на непосредственно сам сайт, который получился, в данном случае [**http://127.0.0.1:5000**](http://127.0.0.1:5000):



Результат работы:

