МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий

|  |
| --- |
| *(наименование института полностью)* |
| Прикладная математика и информатика |
| (*Наименование учебного структурного подразделения*) |
| 09.03.03 Прикладная информатика |
| *(код и наименование направления подготовки / специальности)* |
| Цифровая трансформация бизнеса |
| *(направленность (профиль) / специализация)* |

**Проверяемые задания**

по учебному курсу « Технологии и средства конструирования ПО »

*(наименование учебного курса)*

|  |  |  |
| --- | --- | --- |
| Обучающегося | Дениса Юрьевича Кайманова |  |
|  | *(И.О. Фамилия* |  |
| Группа | ПИбд-2205а |  |
|  |  |  |
| Преподаватель | Гущина Оксана Михайловна |  |
|  | *(И.О. Фамилия)* |  |

Тольятти 2024

Файлы проекта доступны в репозитории по адресу:

<https://github.com/DroidVedroid/TiSKPO_webapp_project/>

# Проверяемое задание № 1

Цель работы: сформировать навыки развертывания среды разработки веб-приложения.

Задание. Развертывание среды разработки и реализации.

В качестве инструментария для разработки веб-приложения выберем язык Python и его фреймворк Flask. Этот выбор обосновывается несколькими преимуществами:

1. Простота и гибкость: Flask предлагает простой и минималистичный подход к созданию веб-приложений. Он не навязывает жестких правил и позволяет разработчику создавать приложения, соответствующие конкретным требованиям, без лишнего объема кода.
2. Быстрая разработка: Flask позволяет быстро начать работу над проектом благодаря своей простоте и интуитивно понятной структуре. Это особенно полезно для создания MVP или прототипов, когда необходимо быстро протестировать идею.
3. Хорошая документация: Flask имеет отличную документацию, которая содержит множество примеров кода и руководств по различным аспектам разработки веб-приложений. Это делает процесс изучения и работы с фреймворком более простым и эффективным.
4. Масштабируемость: хотя Flask начинается с минималистичного набора функций, он легко масштабируется и расширяется благодаря большому количеству расширений (Flask extensions) и сторонних библиотек. Это позволяет адаптировать приложение под изменяющиеся требования и потребности.

В качестве средства ORM (Object-Relational Mapping, – позволяет работать с базой данных, используя объектно-ориентированный подход) будем использовать библиотеку SQLAlchemy – мощный инструмент для работы с базами данных в Python, который отлично интегрируется с Flask, а также поддерживает различные СУБД, что делает работу с данными гибкой и удобной.

Для удобной работы с HTML будем использовать библиотеку Jinja2 – это шаблонизатор для Python, который используется во многих веб-фреймворках, включая Flask. Он позволяет разработчикам создавать динамические HTML-страницы, в которых данные могут быть вставлены и обработаны с помощью специальных конструкций и синтаксиса.

В качестве СУБД будем использовать SQLite.

В целом, выбор Python с фреймворком Flask обеспечивает простоту, гибкость, быструю разработку и масштабируемость веб-приложений, что делает их привлекательными для широкого круга задач и проектов.

Для разработки веб-приложения на Flask необходимо:

* установить Python и требуемые зависимости;
* создать базовую структуру приложения;
* определить модель данных (если используется ORM);
* настроить маршрутизацию;
* установить связь с базой данных;
* разработать необходимые представления и шаблоны.

В качестве редактора кода будем использовать Visual Studio Code. Установим Python, с помощью команды в терминале создадим виртуальное окружение (изолированное окружение Python, которое позволяет управлять зависимостями и версиями пакетов для каждого проекта отдельно):

python -m venv venv

Активируем виртуальное окружение:

venv\Scripts\activate

Установим Flask, SQLAlchemy и Jinja2 в виртуальное окружение с помощью pip. Pip – инструмент управления пакетами для языка программирования Python. Он позволяет устанавливать, обновлять и удалять пакеты Python из репозиториев Python и других источников:

pip install Flask SQLAlchemy Jinja2 Flask-SQLAlchemy

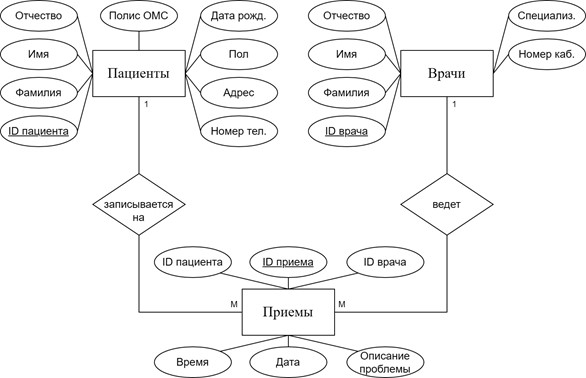
С помощью команды pip list проверим, что все пакеты успешно установлены.

# Проверяемое задание № 2

Цель работы: сформировать навыки создания простейшей веб-страницы.

Задание. Разработка веб-приложения.

Выберем вариант предметной области «Регистратура поликлиники» и разработаем для нее диаграмму «сущность-связь» в нотации Чена:



Создадим файл app.py и определим в нем наше Flask приложение.

from flask import Flask

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

@app.route('/')

@app.route('/index')

def index():

return 'Стартовая страница успешно загружена'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Flask включает в себя встроенный веб-сервер, который можно использовать для разработки и тестирования своих веб-приложений. Этот сервер обычно используется во время разработки и доступен по умолчанию по адресу http://127.0.0.1:5000/.

Для отображения динамического контента на веб-страницах в Flask используются шаблоны. Они позволяют разделять логику приложения и представление данных.

В корневом каталоге проекта создадим каталог с именем templates. Flask будет искать шаблоны в этом каталоге по умолчанию. Затем создадим файлы шаблонов с расширением \*.html:

* base.html – базовый шаблон с html кодом;
* patients.html – для вывода данных из таблицы Пациенты;
* doctors.html – для вывода данных из таблицы Врачи;
* appointments.html – для вывода данных из таблицы Приемы;
* patients-edit.html – для формы редактирования данных Пациенты;
* doctors-edit.html – для формы редактирования данных Врачи;
* appointments-edit.html – для формы редактирования данных Приемы.

В приложении Flask используем функцию render\_template для отображения шаблона. Эта функция принимает имя шаблона в качестве аргумента и, опционально, передает данные для использования в шаблоне.

С помощью конструкций {% extends 'base.html' %}, {% block title %} {% endblock %}, {% block body} {% endblock %} дополним базовый шаблон и изменим нужные веб-страницы. Ссылки записываются в виде: <a href="{{ url\_for (func) }}"></a>, где func – функция из файла app.py, отвечающая за обработку веб-страницы, которая должна быть доступной по этой ссылке.

Подключим стили: в папке проекта создадим файл static\css\main.css и между тегами <head> и </head> укажем ссылку на css файл:

<link rel="stylesheet" href="{{ url\_for('static', filename='css/main.css') }}">

Внесем правки в файл app.py, подключим к проекту подготовленные ранее веб-страницы:

from flask import Flask, render\_template

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

@app.route('/')

@app.route('/appointments')

def appointments():

return render\_template('appointments.html')

@app.route('/appointments-edit')

def appointments\_edit():

return render\_template('appointments-edit.html')

@app.route('/doctors')

def doctors():

return render\_template('doctors.html')

@app.route('/doctors-edit')

def doctors\_edit():

return render\_template('doctors-edit.html')

@app.route('/patients')

def patients():

return render\_template('patients.html')

@app.route('/patients-edit')

def patients\_edit():

return render\_template('patients-edit.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

# Проверяемое задание № 3

Цель работы: сформировать навыки разработки и реализации веб-приложения.

Задание. Разработка веб-приложения. Установка связи с базой данных.

Для определения структуры базы данных будем использовать файл models, а файл database – для настройки базы данных и создания экземпляра SQLAlchemy.

Используем рекомендации из официальной документации Flask для создания таблиц базы данных и установки связи между ними в соответствии с разработанной схемой ERD.

В модуле database определяем путь к создаваемой БД и создаем функцию для ее инициации:

engine = create\_engine('sqlite:///app.db')

db\_session = scoped\_session(sessionmaker(autocommit=False,

autoflush=False,

bind=engine))

Base = declarative\_base()

Base.query = db\_session.query\_property()

def init\_db():

import models

Base.metadata.create\_all(bind=engine)

В модуле models определяем типы данных атрибутов таблиц создаваемой БД в виде класса Python, например:

class Patient(Base):

\_\_tablename\_\_ = 'Пациенты'

id = Column(Integer, primary\_key=True)

firstname = Column(String(50), nullable=False)

birthday = Column(String(8), nullable=False)

Затем в консоли Python инициируем создание БД последовательными командами:

>>> from database import init\_db

>>> init\_db()

>>> exit()

В результате получим файл базы данных app.db с заданной структурой и связями между таблицами.

Реализуем функционал отображения данных из БД на веб-странице, для этого нам нужно сначала получить эти данные из базы данных, а затем передать их в HTML-шаблон для отображения.

Сначала определим представление Flask для обработки запросов на отображение этой таблицы (пример для таблицы Врачи):

@app.route('/doctors')

def doctors():

doctors = Doctor.query.all()

return render\_template('doctors.html', doctors=doctors)

Затем используем ранее созданный HTML-шаблон doctors.html для отображения данных, с помощью цикла for переберем данные из таблицы Врачи, а далее выведем их на веб-страницу:

{% for doctor in doctors %}

<tr>

<td>{{ doctor.id }}</td>

<td>{{ doctor.lastname }}</td>

<td>{{ doctor.firstname }}</td>

<td>{{ doctor.patrname }}</td>

<td>{{ doctor.spec }}</td>

<td>{{ doctor.cab }}</td>

</tr>

{% endfor %}

Аналогичным образом организуем вывод из базы данных в HTML-шаблоны таблицы Приемы и таблицы Пациенты.

# Проверяемое задание № 4

Цель работы: сформировать навыки разработки и реализации веб-приложения.

Задание. Разработка веб-приложения. Функционал добавления и удаления записей в базе данных.

Чтобы добавить запись из HTML-формы в базу данных с использованием Flask, нам нужно настроить маршрут, который будет обрабатывать отправленные данные из формы, а затем выполнить сохранение в базе данных. Пример кода для записи данных в таблицу Врачи:

@app.route('/doctors-edit', methods=['POST', 'GET'])

def doctors\_edit():

if request.method == 'POST':

lastname = request.form['lastname']

firstname = request.form['firstname']

patrname = request.form['patrname']

spec = request.form['spec']

cab = int(request.form['cab'])

new\_doctor = Doctor(lastname=lastname, firstname=firstname,

patrname=patrname, spec=spec, cab=cab)

db\_session.add(new\_doctor)

db\_session.commit()

return redirect('/doctors')

else:

return render\_template('doctors-edit.html')

Здесь мы добавили новый маршрут /doctors-edit, который будет обрабатывать POST-запросы, отправленные из HTML-формы. Функция doctors\_edit() получает данные из формы с помощью request.form, создает новую запись в таблице Врачи на основе полученных данных, добавляет ее в сессию базы данных и сохраняет изменения.

После успешного добавления мы перенаправляем пользователя обратно на страницу просмотра записей из таблицы БД Врачи, где он может увидеть обновленный список записей, включая только что добавленную.

Аналогичным образом организуем внесение данных в БД для таблицы Приемы и таблицы Пациенты.

Для реализации функционала удаления записи из базы данных понадобится следующее:

* в HTML-шаблоне создать форму для отправки запроса на удаление записи;
* определить маршрут в Flask приложении, который будет обрабатывать запросы из формы на удаление записи, далее получить ID записи из запроса и использовать ORM для удаления записи с этим ID из БД.

@app.route('/doctors/delete\_record', methods=['POST', 'GET'])

def doctors\_delete\_record():

record\_id = int(request.form['record\_id'])

record = Doctor.query.filter\_by(id=record\_id).first()

if record is None:

abort(404)

db\_session.delete(record)

db\_session.commit()

statement = f'Запись с id = {record\_id} удалена успешно'

return render\_template('status.html', statement=statement)

Если операция удаления завершена успешно, то возвращаем сообщение о статусе операции в HTML-шаблон.