Самостоятельная работа РК2 - Создание приложения для заметок на Flask

1. Настройка окружения и установка Flask
Для начала убедитесь, что у вас установлен Python и pip. Создайте новую директорию для проекта и настройте виртуальное окружение:
python -m vev vev

Активируйте виртуальное окружение и установите Flash:
source venv/bin/activate # Для Windows используйте v env\Scripts\activate
pip install Flask

2. Структура проекта
В вашей директории проекта создайте следующую структуру:

- app.py: основной файл приложения.
- /templates: папка для HTML шаблонов.
- /static: папка для статических файлов (CSS, JS).
- /database: папка для файлов базы данных (можете использовать MySQL).

3. Основной код Flask
В app.py напишите базовый код приложения:

```
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

# Маршрут для главной страницы
@app.route('/')
def index():
    # Здесь будет код для получения заметок из базы данных
    return render_template('index.html')

# Маршрут для добавления заметки
@app.route('/add', methods=['POST'])
def add_note():
    # Здесь будет код для добавления новой заметки в базу данных
    return redirect(url_for('index'))

# Маршрут для редактирования заметки
@app.route('/edit/<int:id>', methods=['GET', 'POST'])
```

```
def edit_note(id):
    # Здесь будет код для редактирования заметки
    return render_template('edit.html')

if __name__ == '__main__':
    app.run(debug=True)
```

4. Работа с базой данных

Можно использовать MySQL для хранения заметок. Создайте модель данных и инициализируйте базу данных. Вы можете использовать SQLAlchemy как ORM.

4.1. Установка необходимых пакетов

Для начала вам нужно установить пакеты Flask-SQLAlchemy и PyMySQL (драйвер для MySQL):

```
pip install Flask-SQLAlchemy pymysql
```

4.2. Конфигурация Flask и SQLAlchemy

В файле app.py настройте Flask и SQLAlchemy для работы с вашей базой данных MySQL. Для этого добавьте следующий код:

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# Конфигурация базы данных
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://username:password@localhost/dbname'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Инициализация объекта SQLAlchemy
db = SQLAlchemy(app)
```

Замените username, password, localhost и dbname на соответствующие значения для вашей базы данных.

4.3. Создание модели данных

Определите модель для ваших заметок. В этом примере у нас будет простая модель с несколькими полями:

```
class Note(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100))
    content = db.Column(db.Text)

    def __init__(self, title, content):
        self.title = title
        self.content = content
```

4.4. Инициализация базы данных

Перед тем как запустить приложение, вам нужно создать таблицы в базе данных. Вы можете сделать это через Python shell:

```
from app import db
db.create_all()
```

Убедитесь, что вы запускаете эти команды в том же окружении, что и ваше приложение.

4.5. Работа с данными

Теперь вы можете добавлять, извлекать, обновлять и удалять заметки, используя методы SQLAlchemy. Например, для добавления новой заметки:

```
new_note = Note(title="Заголовок", content="Текст заметки")
db.session.add(new_note)
db.session.commit()
```

4.6. Интеграция в Flask-приложение

Используйте эти модели и методы в маршрутах вашего Flask-приложения для обработки запросов на добавление, редактирование и удаление заметок.

5. Создание шаблонов

Создайте HTML шаблоны для отображения заметок (index.html) и редактирования заметок (edit.html) в папке /templates.

Для создания HTML-шаблонов index.html и edit.html в вашем Flask-приложении для заметок, вы будете использовать систему шаблонов Jinja2, которая встроена во Flask. Эти шаблоны позволят вам динамически отображать содержимое из вашего Flask-приложения.

index.html: Этот шаблон будет использоваться для отображения списка всех заметок. Он может включать форму для добавления новых заметок и ссылки для редактирования или удаления каждой заметки.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Мои Заметки</title>
</head>
<body>
  <h1>Мои Заметки</h1>
  <form action="/add" method="post">
    <input type="text" name="title" placeholder="Заголовок">
    <textarea name="content" placeholder="Содержание заметки"></textarea>
    <button type="submit">Добавить Заметку</button>
  </form>
  <ul>
    {% for note in notes %}
      <li>
        <strong>{{ note.title }}</strong> - {{ note.content }}
        <a href="/edit/{{ note.id }}">Редактировать</a>
        <a href="/delete/{{ note.id }}">Удалить</a>
      </li>
    {% endfor %}
  </ul>
</body>
</html>
```

edit.html: Этот шаблон предназначен для редактирования существующей заметки. Он будет содержать форму с предзаполненными данными выбранной заметки.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Редактирование Заметки</title>
</head>
<body>
  <h1>Редактирование Заметки</h1>
```

```
    <form action="/edit/{{ note.id }}" method="post">
        <input type="text" name="title" value="{{ note.title }}">
        <textarea name="content">{{ note.content }}</textarea>
        <button type="submit">Сохранить Изменения</button>
    </form>
    <a href="/">Назад к заметкам</a>
</body>
</html>
```

В обоих шаблонах используется синтаксис Jinja2 для вставки переменных ({{ }}) и управления потоком ({% %}). В index.html мы перебираем notes, что является списком всех заметок, переданных из Flask-приложения. В edit.html мы используем данные одной конкретной заметки (note) для отображения в форме.
Поместите эти файлы в папку /templates вашего Flask-проекта. Flask автоматически будет искать шаблоны в этой директории.

6. Стилизация
Используйте CSS для стилизации вашего интерфейса. CSS-файлы должны находиться в папке /static.
Создание файла CSS
Создайте файл, например style.css, в папке /static вашего Flask-проекта. В этом файле вы можете определить свои собственные стили. Вот пример содержимого:

```
body {
    background-color: #f8f9fa;
    font-family: 'Arial', sans-serif;
}

h1 {
    color: #007bff;
}

.form-control {
    border-radius: 0;
}

.btn {
    border-radius: 0;
```

```
}
```

Этот CSS-файл задает фоновый цвет страницы, стиль шрифта, цвет заголовков и убирает закругление углов у элементов форм и кнопок. Вы можете изменить дизайн под себя и улучшить.

Подключение CSS файла в HTML шаблонах
Чтобы использовать этот CSS-файл в ваших шаблонах, добавьте ссылку на него в <head> ваших HTML-файлов (index.html и edit.html). Пример для index.html:
Подключение CSS файла в HTML шаблонах
Чтобы использовать этот CSS-файл в ваших шаблонах, добавьте ссылку на него в <head> ваших HTML-файлов (index.html и edit.html). Пример для index.html:

```
<head>
  <!-- ... (Bootstrap CSS и другие теги) -->
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
```

Кастомизация дизайна
Теперь вы можете дальше кастомизировать свой CSS-файл, чтобы изменить дизайн элементов интерфейса в соответствии с вашими предпочтениями. Вы можете добавить стили для изменения внешнего вида кнопок, форм, шрифтов, цветов и многого другого.

Применение стилей
После добавления стилей и подключения CSS-файла ваши шаблоны будут использовать как стили Bootstrap, так и ваши собственные кастомные стили, что даст вам больше контроля над внешним видом вашего приложения.

7. Добавление и редактирование заметок
Реализуйте логику добавления и редактирования заметок в соответствующих маршрутах Flask.

1. В вашем файле app.py создайте маршрут для добавления заметки. Маршрут должен отслеживать POST-запросы и обрабатывать данные из формы отправки.
2. В вашем файле app.py создайте маршрут для редактирования заметки. Маршрут должен отслеживать GET и POST запросы и работать с конкретной заметкой.

8. Тестирование
Тестируйте ваше приложение, проверяя, корректно ли работают функции добавления, отображения и редактирования заметок.

1. Добавление Заметок

- Откройте ваше приложение в браузере.
- Введите заголовок и текст заметки в соответствующие поля формы на странице главной страницы.
- Нажмите кнопку "Добавить Заметку".
- Убедитесь, что заметка была добавлена и отображается на главной странице с указанным заголовком и текстом.

2. Редактирование Заметок

- На главной странице, найдите заметку, которую вы хотите отредактировать.
- Нажмите на кнопку "Редактировать" рядом с заметкой.
- Внесите необходимые изменения в форму редактирования заметки.
- Нажмите кнопку "Сохранить Изменения".
- Убедитесь, что изменения были сохранены и заметка теперь отображается с обновленным заголовком и текстом.

3. Просмотр Заметок

- На главной странице вы должны видеть список всех добавленных заметок.
- Просмотрите список, чтобы убедиться, что все заметки корректно отображаются.

4. Обработка ошибок

- Попробуйте сценарии, которые могут вызвать ошибки, такие как попытка редактирования несуществующей заметки или доступ к несуществующему URL.
- Убедитесь, что ваше приложение корректно обрабатывает такие ситуации и возвращает понятные сообщения об ошибках.

5. Исключительные ситуации

- Проведите тесты для проверки работы вашего приложения в нестандартных ситуациях, таких как попытка добавить пустую заметку или редактировать заметку без заголовка.
- Убедитесь, что ваше приложение обрабатывает такие ситуации и предоставляет информативные сообщения или инструкции.