Московский государственный технический университет
имени Н. Э. Баумана

В. И. Кузнецов

REST приложение на фреймворке Django

Методические указания к выполнению
лабораторной работы №11 по дисциплине
«Языки интернет программирования»

Редакция от 1.09.2025

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Кузнецов В.И.

REST приложение на фреймворке Django: методические указания к выполнению практикума № 11 и лабораторной работы №11 по дисциплине «Языки интернет-программирования» / В. И. Кузнецов.

В методическом пособии приведены правила и методы реализации REST приложения и организации взаимодействия с браузерной частью приложения

Для студентов МГТУ им. Н.Э. Баумана, обучающихся по направлению «Информатика и вычислительная техника».

Теоретическая часть

Современные WEB приложения организованы на основе разделения обязанностей. Приложения состоят из 3 основных частей:

- Хранилище данных (СУБД и другие виды хранилищ)
- BackEnd (приложение на сервере отвечающее за логику приложения)
- FrontEnd (приложение в браузере/мобильное отвечающее за логику на стороне пользователя

В современных методологиях разработки, проектирование приложения на себя берет архитектор, а аналитик проектирует контракт взаимодействия бэк-фронт и структура базы данных. Но чаще эти роли в себе несет разработчик.

Backend (серверная часть)

Backend включает в себя бизнес-логику приложения, взаимодействие с базой данных, обработку пользовательских запросов и отправку соответствующих ответов фронтенду. Обычно реализуется на языках программирования, таких как Python, JavaScript (Node.js), PHP, Ruby, Go и др., и фреймворках вроде Django, Express.js, Laravel, Rails и других.

Frontend (клиентская часть)

Frontend представляет собой интерфейс пользователя, отображаемый браузером. Здесь сосредоточены HTML-разметка, CSS-стили и JavaScript-код, обеспечивающие интерактивность и визуализацию интерфейса. Современные инструменты включают React, Vue.js, Angular и другие библиотеки и фреймворки.

Как устроено взаимодействие?

Современные веб-приложения чаще всего используют **архитектуру RESTful API**, основанную на HTTP-запросах и протоколах передачи данных JSON или XML. Это означает, что фронтенд отправляет запросы на сервер, получает от него ответы и обрабатывает их для обновления состояния интерфейса.

Вот основные шаги взаимодействия:

1. **Инициация запроса**: Пользователь совершает какое-то действие (например, вводит данные в форму).

2. **Отправка AJAX-запроса**: Фронтенд формирует HTTP-запрос (GET, POST, PUT, DELETE и т.п.) и отправляет его на соответствующий endpoint сервера.

3. **Обработка на стороне сервера**: Сервер принимает запрос, проверяет права доступа, извлекает необходимые данные из базы данных и формирует ответ.

4. **Возврат результата**: Ответ возвращается клиенту в удобочитаемом формате (обычно JSON). Если возникла ошибка, сервер также возвращает соответствующее сообщение.

5. **Обновление интерфейса**: Получив ответ, фронтенд обновляет представление страницы (HTML/CSS/JS), отражая новые данные или изменения статуса операции.

Этот цикл повторяется каждый раз, когда пользователь взаимодействует с приложением.

Важные технологии и подходы

Для эффективной организации взаимодействия применяются различные методы и инструменты:

- **REST API**: Наиболее распространённый способ взаимодействия клиента и сервера посредством стандартных HTTP-методов.
- **GraphQL**: Альтернатива REST, позволяющая клиентам запрашивать именно те данные, которые нужны, уменьшая количество ненужных передач данных.
- **WebSockets**: Протокол для двустороннего взаимодействия в режиме реального времени, используемый для приложений с постоянной связью (чат, онлайн-игры и т.д.).
- **Fetch / Axios**: Библиотеки JavaScript для отправки асинхронных запросов с фронта.
- **JWT (JSON Web Tokens)**: Метод аутентификации и авторизации, широко применяемый в современном вебе.

Таким образом, взаимодействие между frontend и backend строится вокруг простых принципов обмена данными через стандартизированные форматы и

протоколы. Этот подход позволяет создавать масштабируемые, производительные и удобные для пользователя веб-приложения.

Важным этапом является документирование API для организации взаимодействия и последующего расширения и переиспользования. Swagger со спецификацией полностью покрывает эту задачу, а так же частично покрывает задачи тестирования через Web интерфейс.

Правильное проектирование системы является важным фактором для скорости работы системы и обеспечения безопасности.

Практическая часть

В процессе

Задания

1. Создать базовое приложение Django
2. Реализовать таблицу “comment” с 2 полями {Время типа timestamp, строка длинной 20 символов
3. Реализовать endpoint по следующим путям
   3.1. GET ‘/’ возвращает стартовую страницу со следующим содержимым
      - Время + кнопка «Обновить»
      - Таблица из 10 строчек и 2х столбцов время и текст
      - Форма из строчки и кнопки «Отправить»
   3.2. GET ‘/time/’ возвращает текущее время сервера в виде json {“time”: “значение”}
   3.3. GET ‘/comment/’ возвращает 10 последних значений из таблицы comment
   3.4. POST ‘/comment /save/’ принимает значение в виде json {“time”: “значение”, “text”:”значение” }. Значение времени получается с сервера перед отправкой данных. Данные должны сохраняться в ранее созданную таблицу.
4. Реализовать автоматическое документирование через swagger по пути ‘/swagger/’

Рекомендации

Все описанное в пункте 3 должно работать без перезагрузки страницы. Страница GET ‘/’ возвращает шаблон с базовым html и простым javascript приложением, оборачивающим кнопки и делающим запросы к основному Django приложению.

Для реализации приложения рекомендуются использовать библиотеку 'rest_framework' устанавливаемую командой pip install djangorestframework.

Для документации необходимо 'drf_yasg' устанавливаемая командой pip install drf-yasg.

В settings.py в списке MIDDLEWARE удалить строчки отвечающие за безопастность 'django.middleware.csrf.CsrfViewMiddleware' и 'django.middleware.clickjacking.XFrameOptionsMiddleware'

Полезные ссылки

https://www.django-rest-framework.org/

https://drf-yasg.readthedocs.io/en/stable/#

Требования к отчету

1. Приложить скриншоты страницы
2. Приложить скриншоты логов
3. Приложить скрин сгенерированной документации swagger
4. Приложить листинг кода

Контрольные вопросы

1. Что такое REST?
2. Формат данных JSON
3. Какие методы не используются в REST?
4. Что такое swagger и его назначение?
5. За что отвечает протокол GraphQL?