Дипломная работа по теме: Сравнение различных подходов к реализации REST API: DjangoRest Framework, FastAPI и Flask-RESTful

Автор: Урманов Шухрат Шавкотович

Содержание

[Введение 3](#_Toc177309717)

[Обоснование выбора темы 3](#_Toc177309718)

[Определение цели и задач исследования 4](#_Toc177309719)

[Основные понятия и определения 5](#_Toc177309720)

[Методы и подходы к разработке REST API 6](#_Toc177309721)

[Обзор инструментов для разработки REST API на Python 8](#_Toc177309722)

[Сравнительный анализ производительности и удобства использования 9](#_Toc177309723)

[Проектирование и реализация REST API для тестирования 11](#_Toc177309724)

[Тестирование и оценка производительности 13](#_Toc177309725)

[Заключение 14](#_Toc177309726)

# Введение

## **Обоснование выбора темы**

1. В современном мире разработка веб-приложений и взаимодействие с REST API (Representational State Transfer — передача состояния представлени — **это программная архитектура, которая определяет условия работы API (Application Programming Interface, программный интерфейс приложения)**. Простыми словами, REST API предусматривает предоставление доступа серверного ПО к собственным данным клиентским приложениям по определённым URL-адресам). REST API используются для создания взаимодействий между различными сервисами и приложениями, и выбор подходящего инструмента для их реализации является ключевым фактором успеха проекта.
2. Потребности рынка: растет спрос на специалистов, владеющих навыками разработки REST API. Компании стремятся к выбору наиболее эффективного инструмента для обеспечения высокой производительности, безопасности своих сервисов.
3. Практическая значимость: Работа над сравнением DjangoRest Framework, Flask-RESTful и FastAPI позволит применить полученные знания на практике и даст опыт создания API с использованием различных подходов. Это поможет разработчикам лучше понять преимущества и недостатки каждого из инструментов в реальных условиях.

## **Определение цели и задач исследования**

Цель исследования: Разработать REST API с использованием DjangoRest Framework, FastAPI и Flask-RESTful, провести их сравнение.

* Обзор фреймворков для создания REST API: Провести анализ DjangoRest Framework, Flask-RESTful и FastAPI, изучить их особенности, сильные и слабые стороны, а также выбрать критерии для их сравнительного анализа.
* Разработка сценариев сравнения: Определить параметры и критерии для сравнения фреймворков, такие как производительность, удобство разработки и поддержка документации. Подготовить сценарии использования для проведения тестирования.
* Реализация API и проведение тестирования: Разработать тестовые REST API на основе каждого фреймворка, провести тестирование с использованием различных методов и инструментов для оценки производительности и удобства разработки.

Цели и задачи исследования направлены на практическое сравнение различных фреймворков, что позволит определить их сильные и слабые стороны для использования в реальных проектах.

Основные понятия и определения  
  
Обзор основных понятий в области тестирования эффективности предобученных моделей распознавания объектов

1. **Фреймворк** (англ. framework — «каркас, структура») — **готовый набор инструментов, который помогает разработчику быстро создать продукт**: сайт, приложение, интернет-магазин, CMS-систему.

Простыми словами, это заготовки кода с заранее подобранными по совместимости частями и чёткой структурой, которые позволяют пользователю выполнить типовой проект, используя минимальные затраты собственных усилий.

1. **Веб-приложение (Web Application)**: это программное обеспечение, которое запускается в веб-браузере. Компании должны обмениваться информацией и оказывать услуги удаленно. Они используют веб-приложения для удобной и безопасной связи с клиентами. В контексте данного проекта, веб-приложение будет использоваться для сравнения и демонстрации эффективности различных предобученных моделей распознавания объектов.
2. **Сериализация:** Процесс преобразования данных из внутреннего формата в формат, пригодный для передачи через API (например, JSON).
3. **Маршрутизация:** Механизм сопоставления HTTP-запросов с соответствующими методами API, обеспечивающий правильную обработку запросов.
4. **Производительность API:** Оценка скорости обработки запросов, задержек и пропускной способности API в условиях нагрузки.
5. REST (Representational State Transfer): **архитектурный стиль взаимодействия компонентов распределённого приложения в сети**.

Это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать

1. API (Application Programming Interface): **это программный интерфейс приложений, набор инструкций, который позволяет разным приложениям общаться между собой**.

API может быть:

* **У языка программирования**. Помогает функциям корректно общаться друг с другом.
* **У операционной системы**. Помогает программам получать данные из памяти или менять настройки ОС.
* **У веб-сервисов**. Позволяет другим веб-сервисам и программам подключаться и работать совместно.

Методы и подходы к разработке REST API

DjangoRest Framework (DRF):

DjangoRest Framework это **набор инструментов для создания веб-сервисов и API на основе фреймворка Django**.

Он предоставляет компоненты и инструменты, которые упрощают процесс разработки API и помогают сократить количество кода, необходимого для создания полнофункциональных и безопасных веб-сервисов.

**Некоторые возможности DRF:**

* **Быстрая разработка API**. Готовые компоненты: сериализаторы, представления, маршрутизация и система аутентификации, — значительно упрощают создание API.
* **Гибкая структура**. DRF предоставляет множество настраиваемых параметров и возможность увеличить функциональность API с помощью пользовательских компонентов и расширений. [1](https://www.nic.ru/help/dlya-chego-ispol6zuetsya-django-rest-framework_11684.html)
* **Использование различных методов аутентификации**. DRF предоставляет множество способов авторизации, таких как аутентификация на основе токена, OAuth, JWT и другие.
* **Сериализация данных**. Мощные инструменты сериализации и десериализации данных API позволяют преобразовывать объекты моделей Django и другие данные в различные форматы данных, такие как JSON или XML, и обратно.
* **Валидация и обработка данных**. С помощью инструментов для обработки входных данных можно определить правила валидации в соответствии с требованиями приложения.
* **Автоматическая генерация документации API на основе кода**. DRF поддерживает использование инструментов (например, Swagger) для создания более детальной и интерактивной документации.

Flask-RESTful:

Flask-RESTful — **Flask-RESTful** — это **расширение для Flask, которое упрощает создание RESTful API**.

**Основные возможности Flask-RESTful**:

* определение класса ресурсов, который содержит методы для каждого метода HTTP;
* разделение маршрутов и ресурсных методов, что упрощает обслуживание и масштабирование API;
* автоматический анализ запросов и форматирование ответов;
* встроенная поддержка проверки входных данных, обработки ошибок и ограничения скорости;
* интеграция с другими расширениями Flask, что позволяет создавать мощные и многофункциональные API.

FastAPI:

FastAPI — **веб-фреймворк для создания API, написанный на Python**.

Он предлагает декларативный подход к определению маршрутов, запросов и моделей данных, что делает процесс разработки API более простым и понятным.

**Некоторые особенности FastAPI:**

* **Автоматическая генерация документации**. Фреймворк создаёт документацию для API на основе аннотаций Python с использованием стандарта OpenAPI.
* **Поддержка асинхронного программирования**. Это позволяет создавать эффективные и быстрые веб-приложения, способные обрабатывать большое количество запросов одновременно без значительного снижения производительности.
* **Валидация данных**. Фреймворк предоставляет мощные инструменты для валидации данных, что помогает обеспечить правильную обработку и сохранность информации в веб-приложениях.

# Обзор инструментов для разработки REST API на Python

DjangoRest Framework (DRF):

DRF является мощным расширением для Django, которое добавляет все необходимые инструменты для создания REST API. Основные преимущества DRF включают тесную интеграцию с Django ORM, что облегчает работу с базами данных, наличие готовых решений для аутентификации и авторизации, а также продвинутая система сериализации данных. DRF поддерживает создание сложных API и является отличным выбором для проектов с уже существующими приложениями на Django или при необходимости работы с крупными системами.

Flask-RESTful:

Flask-RESTful — это легковесное расширение для Flask, которое добавляет возможности по созданию REST API. Его простота и минималистичный подход делают его отличным выбором для небольших и средних проектов, где требуется гибкость и контроль над каждым аспектом API. Flask-RESTful легко настраивается и интегрируется с различными сторонними библиотеками, предоставляя разработчику большую свободу в выборе инструментов. Он идеален для проектов, где требуется минимальная структура и высокая кастомизация.

FastAPI:

FastAPI — это современный фреймворк, ориентированный на высокую производительность и работу с асинхронными операциями. Он позволяет легко создавать быстрые и масштабируемые API, поддерживая автоматическую валидацию данных и генерацию документации на основе аннотаций типов Python. FastAPI отличается высокой скоростью обработки запросов благодаря использованию асинхронных функций и простотой в использовании при создании как синхронных, так и асинхронных маршрутов. Этот инструмент идеально подходит для проектов, требующих высокой производительности и обработки большого количества запросов.

# Сравнительный анализ производительности и удобства использования

Производительность:

FastAPI: Высокая производительность благодаря поддержке асинхронных запросов ( это запрос без ожидания ответа от сервера. Т.е. скрипт создал объект, а объект послал запрос на сервер. При этом скрипт продолжает выполняться дальше и не ждет ответа от сервера. ) и использованию современных технологий, таких как Starlette (**лёгкий ASGI-фреймворк, который используется при веб-разработке на Python**. Он подходит для создания высокопроизводительных asyncio-сервисов) и Pydantic (это **библиотека для Python, предназначенная для валидации и трансформации данных**. Она помогает разработчикам гарантировать, что входные данные соответствуют установленным правилам и типам, а также обеспечивает их автоматическое преобразование в нужные форматы). FastAPI показывает значительно лучшие результаты при высоких нагрузках, обрабатывая большое количество запросов с минимальной задержкой.

Flask-RESTful: Flask сам по себе не поддерживает асинхронные операции, что ограничивает его производительность при больших нагрузках. Однако, для небольших и средних приложений производительности Flask достаточно.

DjangoRest Framework: DRF работает синхронно, и его производительность ниже по сравнению с FastAPI. Однако его интеграция с Django делает его более медленным при больших нагрузках из-за дополнительных абстракций.

Простота кода:

FastAPI: Код в FastAPI максимально упрощен за счет аннотаций типов и встроенной валидации данных. Асинхронные функции реализуются интуитивно, что делает код компактным и читаемым.

Flask-RESTful: Flask-RESTful предлагает простую структуру и минимум конфигураций, что позволяет разработчикам иметь полный контроль над кодом. Однако это может привести к необходимости написания большего объема кода для базовых функций.

DjangoRest Framework: DRF требует больше шаблонного кода, особенно для сериализации и обработки запросов, что может усложнить структуру кода. Однако это компенсируется высокой гибкостью и функциональностью.

Масштабируемость:

FastAPI: Поддержка асинхронности и отличная работа с большим количеством соединений делают FastAPI лучшим выбором для масштабируемых приложений, особенно в распределенных системах и микросервисах.

Flask-RESTful: Flask можно масштабировать, но для этого потребуется дополнительная конфигурация и использование сторонних инструментов для улучшения производительности.

DjangoRest Framework: DRF имеет встроенные механизмы для работы с большими проектами, но его масштабируемость ограничена из-за отсутствия асинхронной поддержки и общей производительности Django.

Документация:

FastAPI: Автоматически генерирует документацию OpenAPI (это **формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API)** и JSON Schema (это **спецификация для описания структуры JSON-документов**.  Она определяет содержание, структуру, типы данных и ожидаемые ограничения внутри JSON-документа, что помогает обеспечить согласованность и целостность JSON-данных в разных приложениях. ), что является большим преимуществом для разработчиков, особенно при создании сложных API.

Flask-RESTful: Flask-RESTful имеет хорошие примеры документации, но требуется ручная генерация документации API или использование дополнительных библиотек, таких как Swagger.

DjangoRest Framework: DRF имеет отличную документацию, как для фреймворка, так и для самого Django, что делает его удобным для начинающих и опытных разработчиков.

# Проектирование и реализация REST API для тестирования

Планирование структуры и логики API:

1. Определение основных маршрутов и ресурсов, таких как пользователи, задачи и категории.
2. Разработка логики обработки CRUD операций для каждого ресурса: создание, чтение, обновление и удаление пользователей, задач и категорий.
3. Планирование авторизации и аутентификации для управления доступом к задачам и пользователям.
4. Проектирование структуры данных для сериализации и десериализации, включая использование моделей данных и схем.

Реализация базового функционала на DjangoRest Framework, Flask-RESTful и FastAPI:

1. DjangoRest Framework:

* Разработка моделей данных для пользователей, задач и категорий с использованием Django ORM и миграций.
* Создание сериализаторов для преобразования данных моделей в JSON и обратно.
* Настройка маршрутов и обработчиков с использованием ViewSet и маршрутизаторов DRF.
* Реализация аутентификации и авторизации с использованием встроенных механизмов DRF.

1. Flask-RESTful:

* Определение моделей данных и создание таблиц в SQLite с использованием SQLAlchemy.
* Реализация маршрутов и обработчиков запросов для CRUD операций на базе данных.
* Настройка сериализации данных с помощью marshmallow или другой библиотеки для преобразования данных в JSON.
* Реализация базовой аутентификации и управления доступом.

1. FastAPI:

* Определение моделей данных с использованием Pydantic и создание таблиц в SQLite через SQLAlchemy.
* Реализация асинхронных маршрутов для обработки CRUD операций, поддерживающих высокую производительность.
* Автоматическая генерация документации API и валидация данных с помощью аннотаций типов.
* Настройка аутентификации и авторизации для управления доступом к задачам и пользователям.

Разработка Task Management System на каждой из этих систем позволила реализовать и протестировать базовый функционал для управления пользователями, задачами и категориями, а также оценить производительность и удобство использования различных подходов и инструментов.

# Тестирование и оценка производительности

## Методы тестирования

Нагрузочное тестирование: Это метод тестирования, который помогает определить, как система справляется с высокими объемами запросов и пользователей. Включает в себя моделирование нагрузки на API для оценки его устойчивости и производительности при различных уровнях нагрузки.

Профилирование запросов: Этот метод позволяет анализировать время выполнения запросов и выявлять узкие места в системе. Профилирование помогает понять, как различные фреймворки и их компоненты обрабатывают запросы и где могут возникать задержки.

## Результаты и интерпретация

На основе результатов тестирования, проведенного с использованием методов, описанных на сайте TechEmpowerBenchmarks, были сделаны следующие выводы:

- FastAPI: Показал наилучшие результаты по производительности, обеспечивая высокую скорость обработки запросов и низкую задержку. Его поддержка асинхронных операций значительно улучшает производительность при больших объемах трафика и параллельных запросах.

- Flask-RESTful:Показал удовлетворительные результаты, но уступал FastAPI в производительности при высоких нагрузках. Flask-RESTful справляется с меньшими объемами запросов хорошо, но при увеличении нагрузки может наблюдаться замедление.

- DjangoRest Framework: Продемонстрировал хорошую производительность, но не дотягивает до FastAPI в условиях высокой нагрузки. DRF хорошо справляется с обработкой запросов, однако его производительность может снизиться при увеличении количества одновременных пользователей.

Выводы из тестирования показывают, что для проектов с высокими требованиями к производительности и обработке большого количества параллельных запросов FastAPI является наиболее подходящим инструментом. Flask-RESTful и DjangoRest Framework также являются хорошими выборами, но их производительность может быть ограничена при значительных нагрузках.

# Заключение

Выводы по сравнительному анализу:

* FastAPI продемонстрировал наилучшую производительность благодаря поддержке асинхронных операций и эффективной обработке запросов. Его скорость и масштабируемость делают его идеальным выбором для высоконагруженных приложений, требующих быстрой обработки большого количества параллельных запросов.
* Flask-RESTful обеспечивает гибкость и простоту разработки, но его производительность может снижаться при увеличении нагрузки. Он хорошо подходит для небольших и средних проектов, где важна простота и минимализм.
* DjangoRest Framework обеспечивает богатую функциональность и интеграцию с Django, но может иметь ограничения по производительности при высоких нагрузках. Он лучше всего подходит для проектов с уже существующими приложениями на Django или для тех, где необходимы сложные функции и администрирование.

Рекомендации по выбору подхода в зависимости от проекта:

* FastAPI рекомендуется для проектов, требующих высокой производительности и обработки большого объема запросов, таких как крупные веб-приложения, микросервисы или высоконагруженные API.
* Flask-RESTful лучше использовать для небольших и средних проектов, где требуется высокая гибкость и минималистичный подход, а также для прототипирования или разработки MVP (минимально жизнеспособного продукта).
* DjangoRest Framework рекомендуется для проектов, которые уже используют Django или требуют богатой функциональности для работы с данными и управления пользователями. Это хороший выбор для полнофункциональных веб-приложений с комплексными требованиями.