Создание RESTful API

№ урока: 2 **Курс:** Flask

Средства обучения: Python3 и любимая среда разработки, например, PyCharm.

Обзор, цель и назначение урока

Узнать, что такое архитектура REST и как она работает, рассмотреть альтернативы REST: gRPC и GraphQL. Научиться создавать RESTful API с помощью Flask-RESTful. Начнем знакомство с ORM SQLAlchemy и познакомимся со SwaggerUI.

Изучив материал данного занятия, учащийся сможет:

- Правильно подбирать архитектурный стиль под специфику проекта.
- Проектировать и разрабатывать RESTful API на основе Flask.
- Моделировать схемы баз данных с помощью SQLAlchemy.

Содержание урока

1. Обзор архитектуры REST
2. Построение RESTful API: CRUD запросы
3. Моделирование схемы базы данных с помощью SQLAlchemy
4. SwaggerUI
5. Альтернативы REST: gRPC и GraphQL

Резюме

- **REST** — это акроним, сокращение от английского Representational State Transfer, передача состояния представления. Это стиль архитектуры программного обеспечения для построения распределенных приложений.
- Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола.
- Системы, поддерживающие REST, называются RESTful-системами.
- Ключевое понятие в REST — это ресурс. Ресурс имеет состояние, и мы можем его получать или изменять при помощи представлений. Под представлением можно понимать JSON/HTML/XML/текст в определенном формате или что угодно, что позволяет нам понимать состояние ресурса или его модифицировать. Наше приложение отвечает за некоторое множество таких ресурсов. Кстати, совокупное состояние ресурсов — это и есть состояние приложения.
- У ресурса есть идентификатор (универсальный код ресурса (URI)), который уникально идентифицирует этот ресурс. Например, URI для определенного клиентского заказа может быть таким: https://myapi/orders/1
- Интерфейсы REST API используют единый интерфейс, который позволяет отделить реализации клиента и службы. Для REST API, созданных на основе протокола HTTP, единый интерфейс будет использовать стандартные HTTP-команды для выполнения операций с ресурсами. Наиболее часто выполняемые операции: GET, POST, PUT, PATCH и DELETE.
- **GET**. Возвращает представление ресурса по указанному универсальному коду ресурса (URI). Текст ответного сообщения содержит сведения о запрашиваемом ресурсе.
- **POST**. Создает новый ресурс по указанному URI. Текст запроса содержит сведения о новом ресурсе.

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- GET vs POST: основная разница состоит в том, что POST-запросы не кешируются и изменяют данные на сервере, в то время как GET - кешируются и не изменяют данные на сервере.
- POST vs PUT: В модели REST запросы POST постоянно применяются к коллекциям. POST /customers/1 - Ошибка, только POST /customers/ - создание кастомера. Запрос PUT создает ресурс или обновляет имеющийся ресурс. PUT /companies/3/employees/john запросит сервер обновить данные или создать, если не существует, john в коллекции сотрудников компании 3.
- **PUT**. Создает или заменяет ресурсы по указанному URI. В тексте сообщения запроса указан создаваемый или обновляемый ресурс.
- **PATCH**. Выполняет частичное обновление ресурса. Текст запроса определяет набор изменений, применяемых к ресурсу.
- **DELETE**. Удаляет ресурс по указанному URI.
- Одна из наиболее распространенных проблем с REST — это чрезмерная и недостаточная загрузка данных (over and under fetching). Это происходит потому, что ресурсы возвращают фиксированные структуры данных. API очень сложно спроектировать так, чтобы он мог предоставить клиентам точные данные.
- **API** - аббревиатура расшифровывается как Application Programming Interface, или интерфейс для программирования приложений. Интерфейс, который позволяет разработчикам использовать готовые блоки для построения приложения. В случае с разработкой мобильных приложений в роли API может выступать библиотека для работы с "умным домом" – все нюансы реализованы в библиотеке, и вы лишь обращаетесь к этому API в своём коде.
- **SQLAlchemy** — это библиотека, которая облегчает взаимодействие между программами Python и базами данных. В большинстве случаев эта библиотека используется как инструмент Object Relational Mapper (ORM), который переводит классы Python в таблицы реляционных баз данных и автоматически преобразует вызовы функций в операторы SQL. SQLAlchemy предоставляет стандартный интерфейс, который позволяет разработчикам создавать независимый от базы данных код для взаимодействия с широким спектром механизмов баз данных.
- **Swagger** — это фреймворк для спецификации RESTful API. Его прелесть заключается в том, что он дает возможность не только интерактивно просматривать спецификацию, но и отправлять запросы – так называемый Swagger UI.
- **GraphQL** - язык запросов с открытым исходным кодом, разработанный Facebook. Он создавался как более эффективная альтернатива REST для разработки и использования программных интерфейсов приложений.
- GraphQL обладает множеством достоинств, например:
    - Вы получаете информацию именно в том объёме, в котором запрашиваете. В отличие от REST, ответ на запрос не будет содержать ненужных данных.
    - Вам будет необходима всего одна конечная точка, никаких дополнительных версий для единственного API.
    - GraphQL — сильно типизированный язык, что позволяет предварительно оценить корректность запроса в рамках системы типов этого синтаксиса, до исполнения. Это позволяет разрабатывать более мощные API.
- **gRPC** — опенсорсный фреймворк для удаленного вызова процедур (RPC), разработанный компанией Google, работает поверх HTTP/2. gRPC используется как более удобная альтернатива REST.

Закрепление материала

- Что такое REST?
- Что такое API?

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Опишите разницу между POST и PUT запросами.
- Опишите разницу между GET и POST запросами.
- Опишите разницу между PATCH и PUT запросами.
- Назовите минусы REST.
- Что такое ORM?
- Зачем использовать ORM?

Дополнительное задание

Задание
Придумайте более эффективную реализацию метода PATCH, а также попробуйте передать данные с помощью метода GET.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Изучите материалы из рекомендуемых ресурсов.

Задание 3
Самостоятельно изучите коды состояния HTTP.

Задание 4
Создайте модель Актера, у которой будут следующие параметры: id - integer pk - true, name - str, birthday - date, is_active - bool. Попробуйте создать ее по примеру модели Фильма, а, также, попробуйте почитать какие бывают отношения между таблицами и определите, какое отношение между таблицами Фильм и Актер.

Рекомендуемые ресурсы

Общие сведения о REST:
https://docs.microsoft.com/ru-ru/azure/architecture/best-practices/api-design
https://ru.wikipedia.org/wiki/REST

Что такое API?
https://habr.com/ru/post/464261/

Mike Bayer: Introduction to SQLAlchemy - PyCon 2014:
https://www.youtube.com/watch?v=P141KRbxVKc
Что такое ORM, как она работает и как ее использовать?
https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one

Обзор HTTP протокола:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Swagger:
https://habr.com/ru/post/322460/

RPC vs REST vs GraphQL:

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

https://www.youtube.com/watch?v=IvsANO0qZEg&t=2257s

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com