GraphQL vs. REST API: Что и когда использовать

Описание

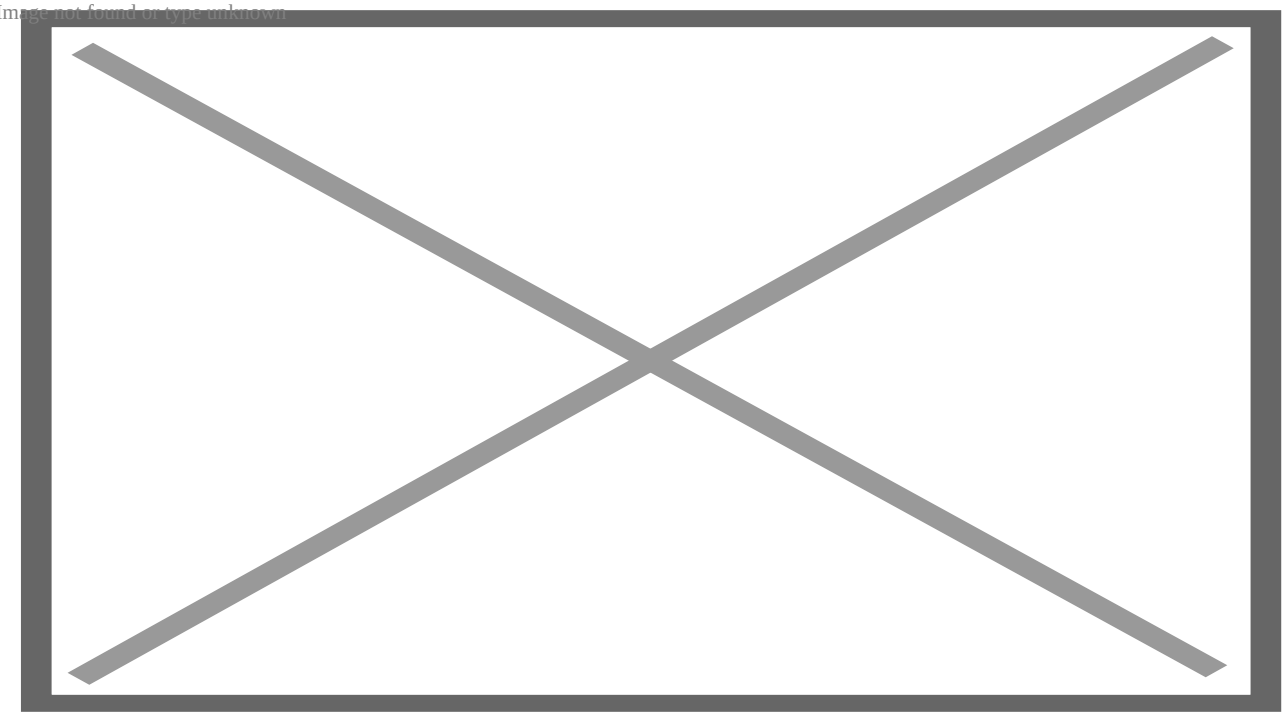
При работе над клиент-серверным приложением разработчикам часто приходится выбирать между GraphQL и REST API. Выбор одного из них может иметь далеко идущие последствия для скорости, масштабируемости и ремонтопригодности приложения. На протяжении многих лет Rest API был предпочтительным вариантом для многих разработчиков. REST API имеет четко определенную структуру, которая делает его простым в использовании и понимании. Таким образом, он позволяет разработчикам легко создавать сложные приложения.

Однако GraphQL стал сильным конкурентом с более эффективным и адаптируемым механизмом запросов. Используя его, разработчики могут просто получить ту информацию, которая им действительно необходима. Более того, модель данных GraphQL обеспечивает простую настройку. И GraphQL, и REST API предлагают различные преимущества и возможности. Поэтому выбор лучшего из них для проекта будет иметь большое влияние на его успех. В этой статье мы расскажем о GraphQL и REST API, их особенностях и преимуществах, случаях использования и ключевых различиях.

Что такое GraphQL?

GraphQL – это надежный язык запросов для API, разработанный компанией Meta. Он обеспечивает лучший способ создания APIS и улучшает вызовы RESTful API.

Image not found or type unknown



С помощью GraphQL разработчики могут использовать одну конечную точку для получения именно тех данных, которые им нужны. Это облегчает управление зависимостями данных и позволяет избежать избыточной выборки. В то время как REST API требует использования нескольких конечных точек для получения различных ресурсов. Однако до сих пор существуют некоторые заблуждения относительно того, чем GraphQL не является. Итак, давайте разберемся со следующими моментами:

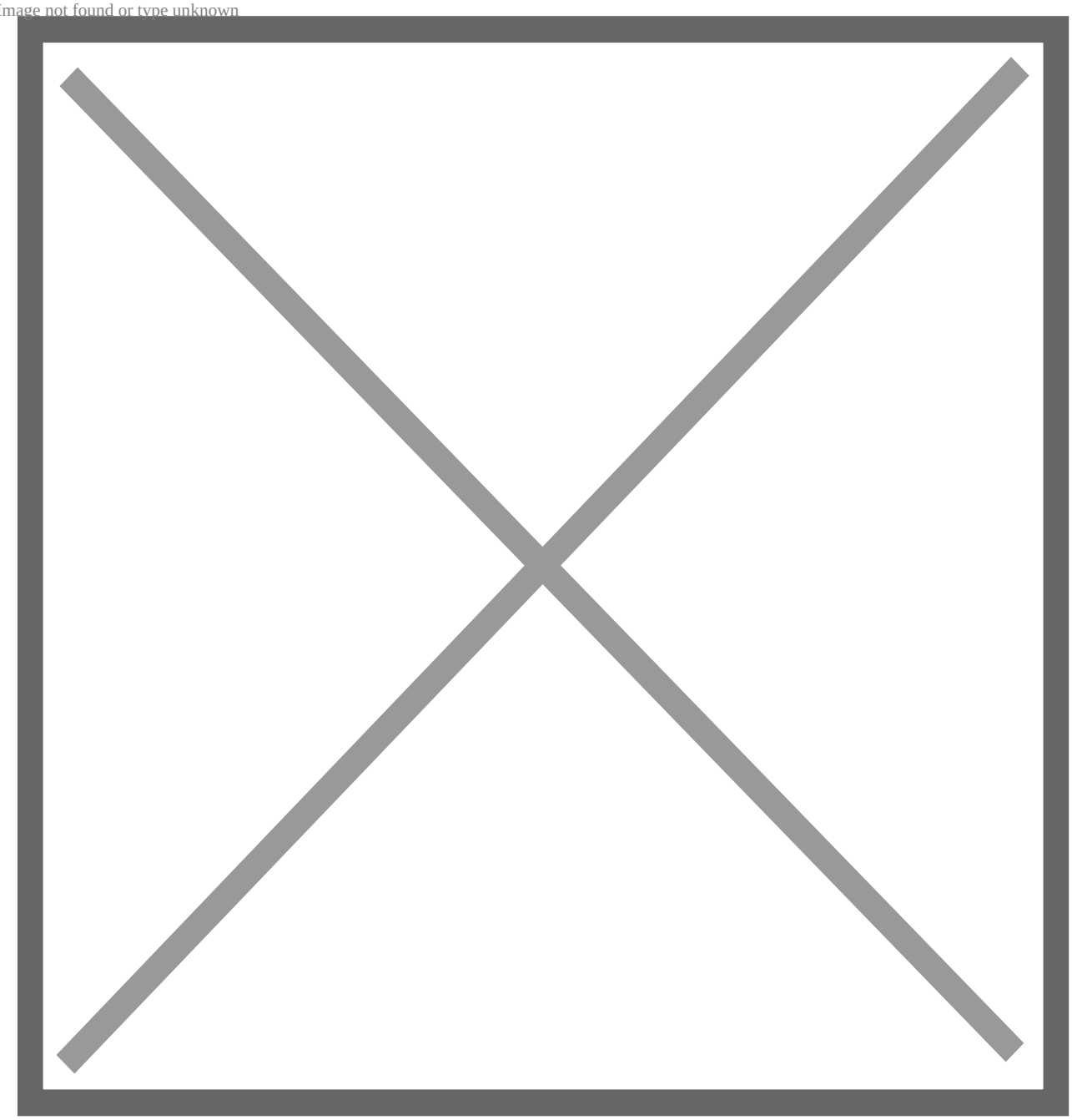
- Это не язык баз данных или ORM, а инструмент для запросов к API.
- Он не предназначен для замены REST API, а является альтернативой, которая может сосуществовать в одном проекте.
- Он не перегружен, не сложен, что делает его легким для изучения и внедрения.

GrapphQL был доступен в качестве проекта с открытым исходным кодом в 2015 году. С тех пор такие компании, как GitHub, Yelp и Shopify, приняли его на вооружение, поскольку его популярность растет. GraphQL превращается в важнейшую способность, которой должны владеть разработчики, в результате растущего спроса на более эффективные API.

Как работает GraphQL

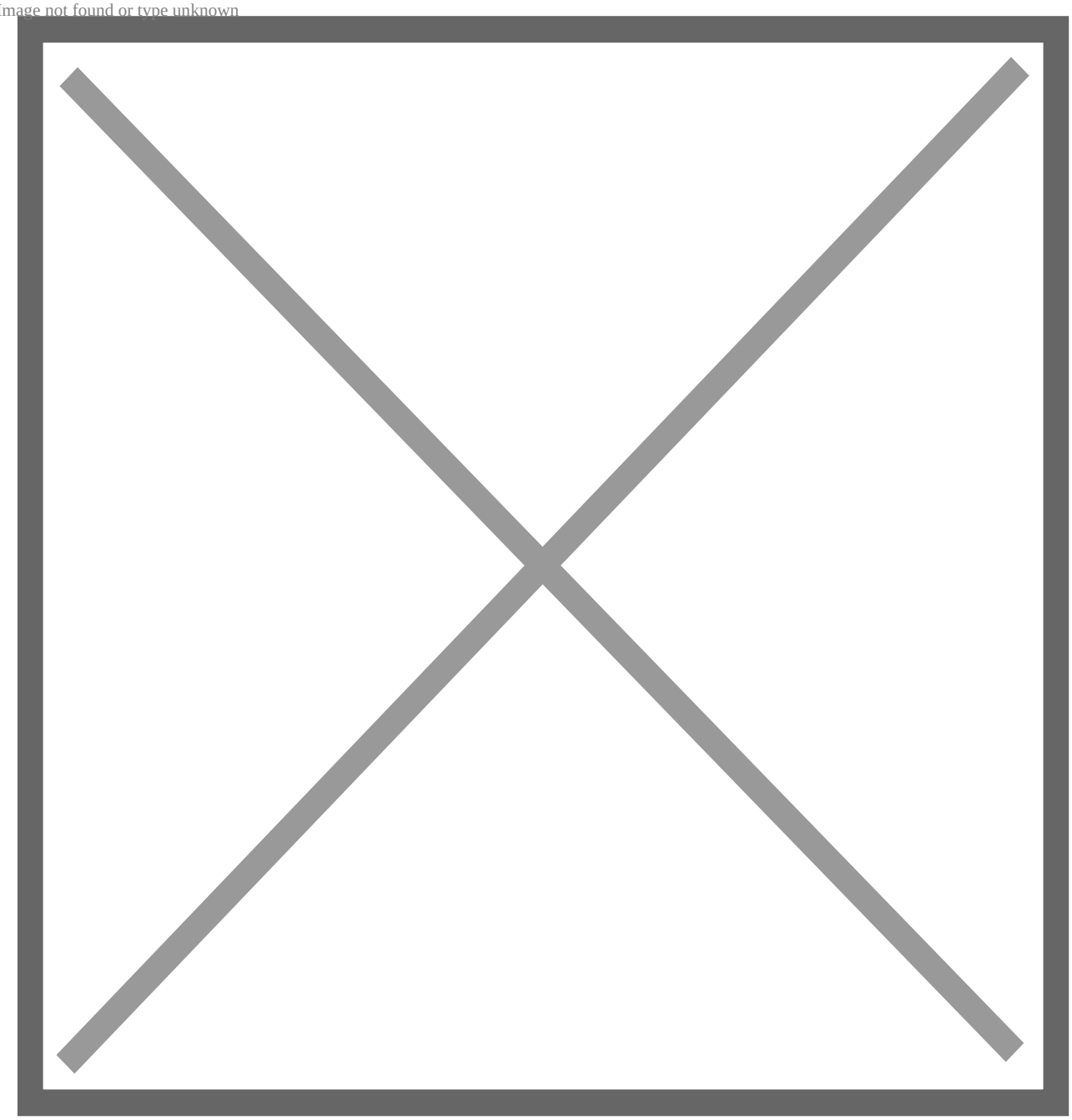
GraphQL построен на схеме, которая описывает типы данных, которые можно запрашивать, а также их взаимосвязи. Выступая в качестве связующего звена между клиентом и сервером, эта схема гарантирует, что обе стороны знают о данных, которые могут быть запрошены. Кроме того, она определяет, как эти данные будут представлены. Рассмотрим приложение для блога, использующее GraphQL API. Схема API может быть описана следующим образом:

Image not found or type unknown



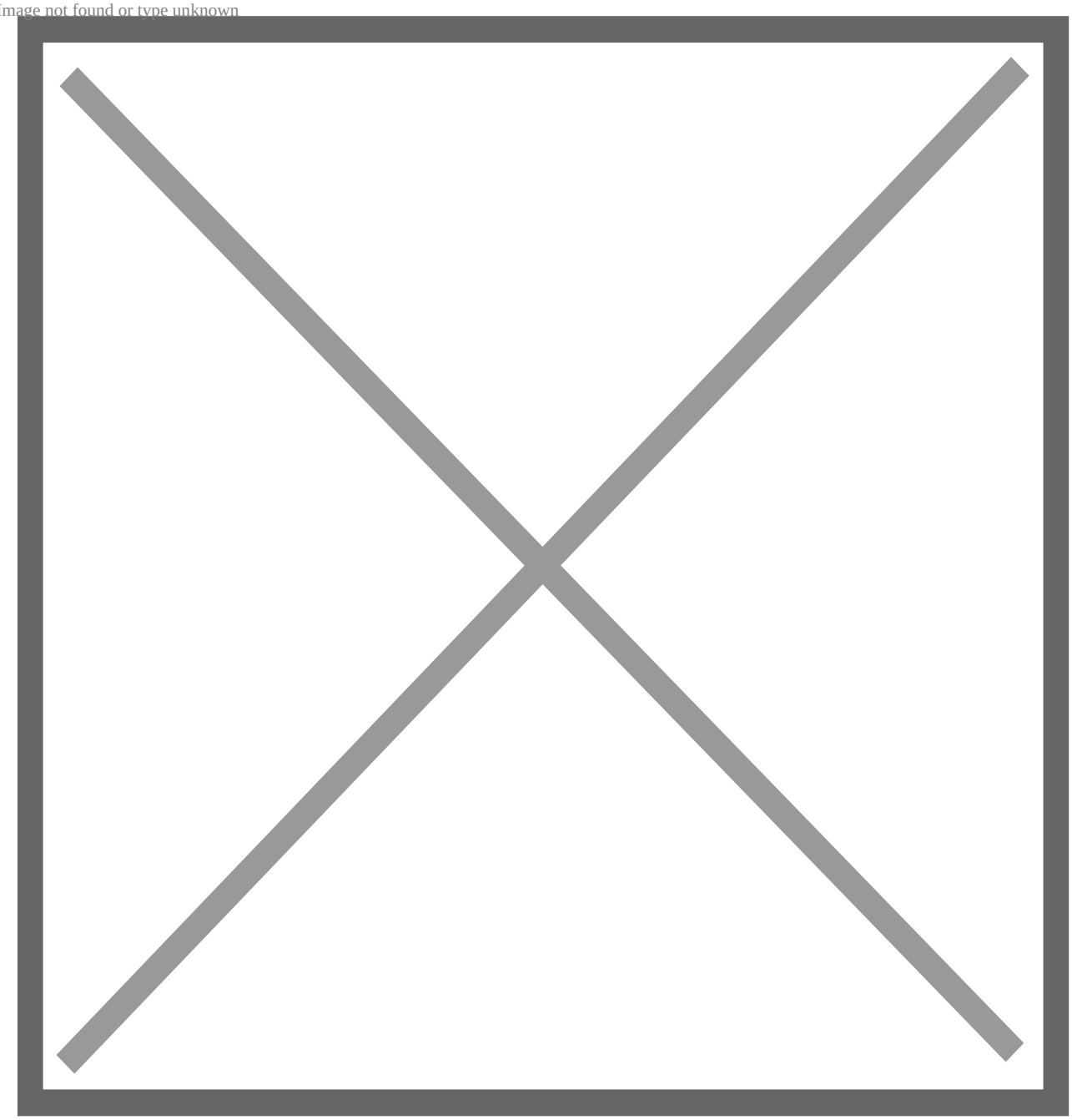
В этой схеме определены типы post и comment, а также тип Query. Он позволяет нам получать отдельные сообщения по ID. Поля, которыми обладает каждый тип, отражают данные, которые можно получить. Используя эту схему, мы можем с помощью приведенного ниже кода создать GraphQL-запрос для получения сообщения и комментариев к нему:

Image not found or type unknown



Результат этого запроса будет включать заголовок, тело, автора и ID сообщения. Он также вернет тело и автора любого комментария, связанного с этим постом. Вместо того чтобы отправлять несколько запросов к различным конечным точкам, мы можем получить все необходимые данные с помощью GraphQL с помощью всего одного вызова API. В результате накладные расходы снижаются, а функциональность API возрастает.

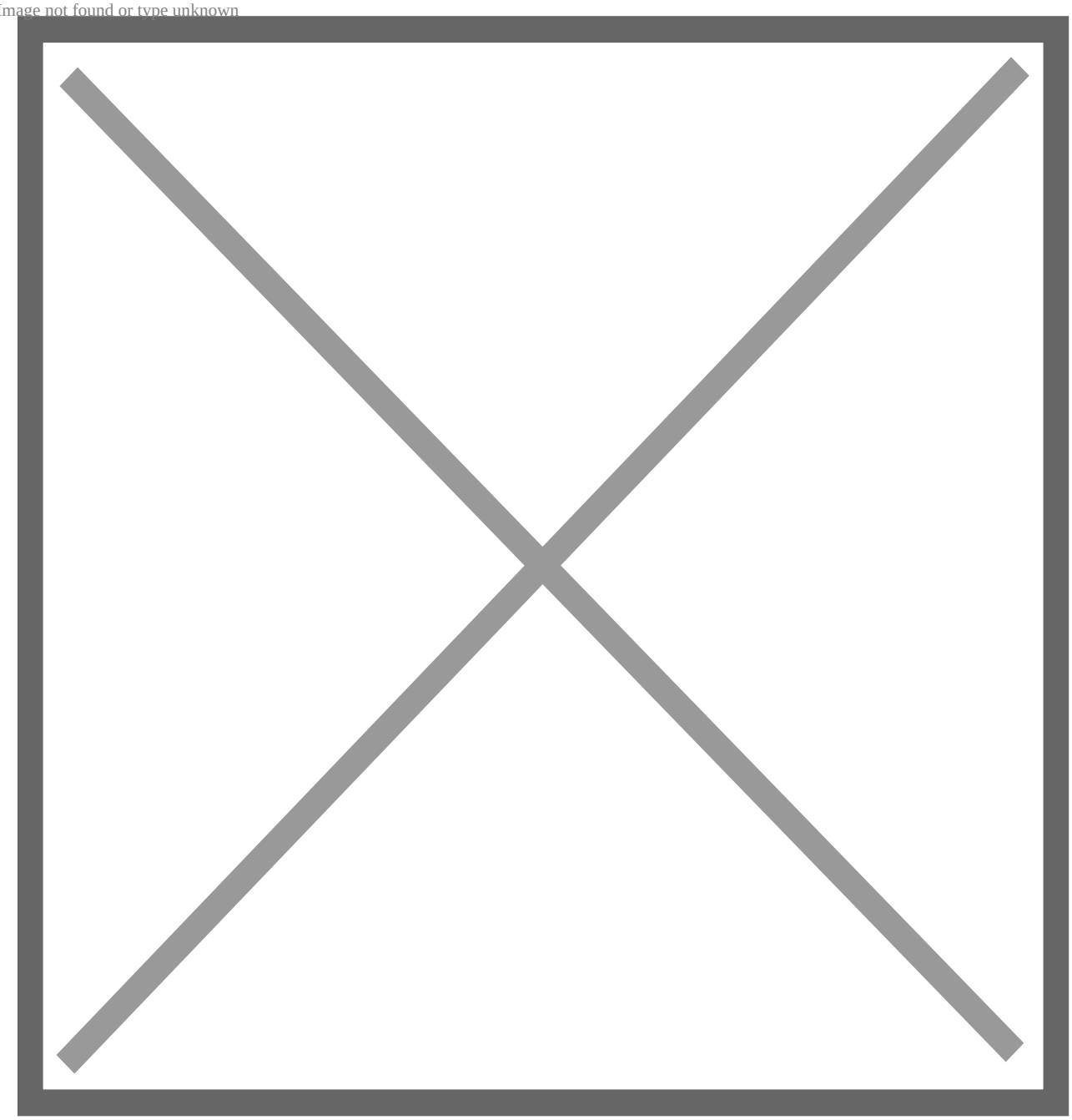
Image not found or type unknown



Что такое Rest API

REST API – это архитектура веб-сервиса. Она обеспечивает связь по протоколам HTTP между многими системами. Это набор архитектурных принципов, которые помогают в разработке масштабируемых, эффективных и универсальных веб-сервисов.

Image not found or type unknown



Тем не менее, это популярный вариант среди разработчиков, поскольку он использует такие распространенные методы HTTP, как:

- **GET:** Эта команда извлекает ресурсы
- **POST:** Для создания ресурса
- **PUT:** Изменяет состояние или обновляет ресурс, который может быть объектом, файлом или блоком.
- **DELETE:** удаление ресурса

В основе REST API лежит идея ресурсов, которые распознаются по характерным URL (Uniform Resource Locator). В зависимости от запроса клиента, каждый ресурс может иметь различное представление, например, следующее:

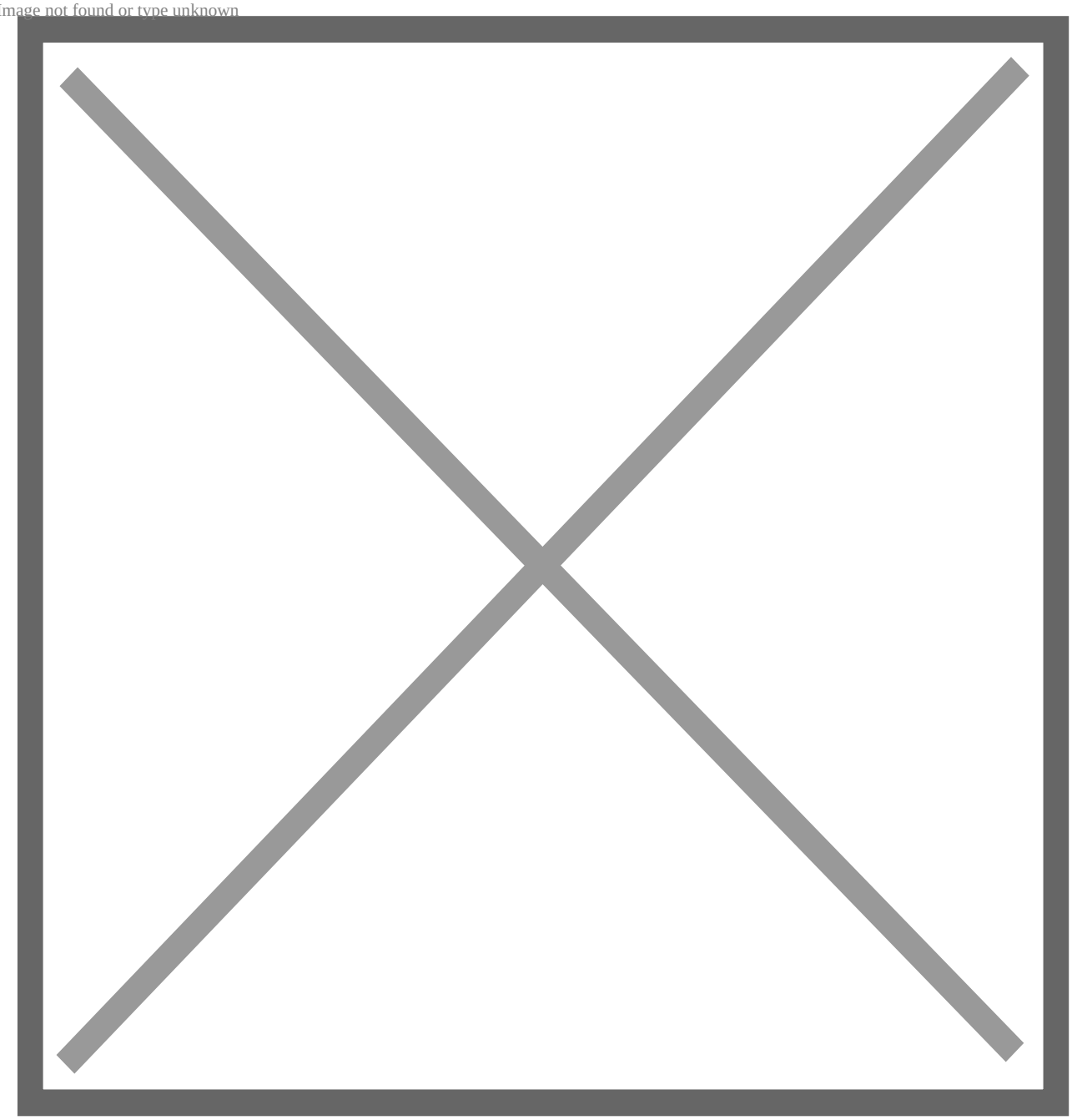
- JSON (JavaScript Object Notation),
- XML (расширяемый язык разметки),
- и HTML (язык разметки гипертекста).

RESTPful API использует эти ресурсы для получения данных, создания записи, обновления записи или ее удаления.

Принцип работы REST API

REST API работает, позволяя пользователям отправлять HTTP-запросы на серверы, которые представляют ресурсы через URL-адреса. После обработки запроса сервер отправляет обратно информацию в заданном формате (JSON или XML). Например, представьте себе веб-приложение, которое позволяет пользователям получить доступ к информации, связанной с книгами. Используя RESTful API, клиенты могут получить подробную информацию об одной книге или о нескольких книгах. Чтобы получить информацию о конкретной книге, клиент отправляет запрос HTTP GET, используя URL ресурсов. Ссылка может быть следующей:https://example.com/api/books/123. После обработки запроса и нахождения книги с идентификатором “123” сервер выдает ответ в выбранном формате (JSON).

Image not found or type unknown



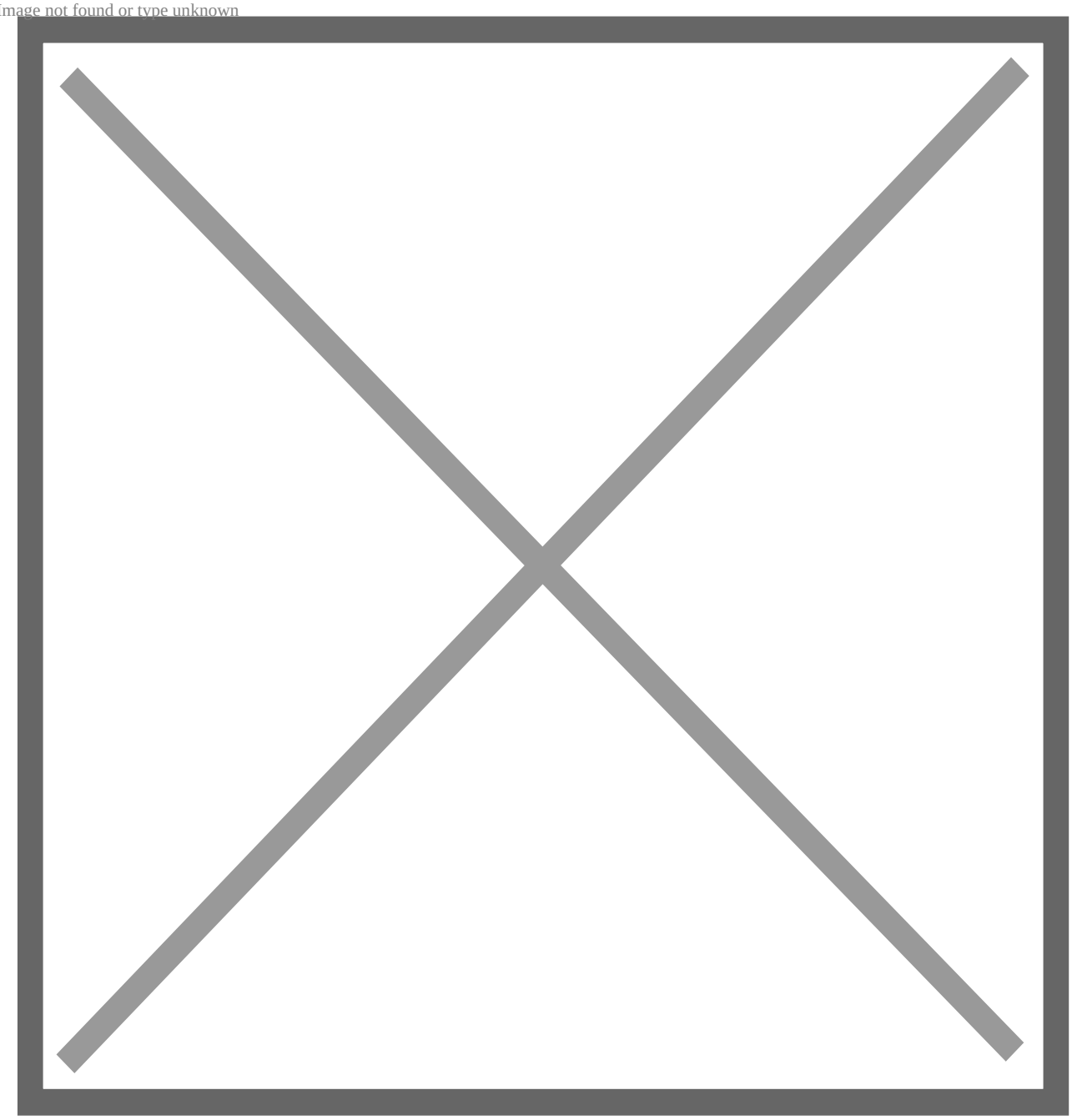
Аналогичным образом, чтобы получить список книг, клиент отправляет HTTP GET запрос на сервер с URL, например, “https://example.com/api/books”. Таким образом, сервер отвечает в требуемом формате, например, JSON.

Особенности GraphQL

GraphQL – это универсальный и эффективный язык запросов, поскольку он обеспечивает надежную типизацию и иерархический поиск данных. Вот некоторые

ключевые особенности GraphQL, которые сделали его популярным среди пользователей:

Image not found or type unknown



- **Сильно типизированный:** GraphQL предлагает схему, включающую несколько типов данных, доступных для API, таких как поля, объекты и ссылки. Для обеспечения легитимной доставки данных эта схема используется для проверки запросов и ответов.
- **Иерархическая структура:** GraphQL позволяет клиентам указывать точные данные, которые им требуются. Таким образом, возвращается меньше

избыточных данных, что повышает скорость работы API.

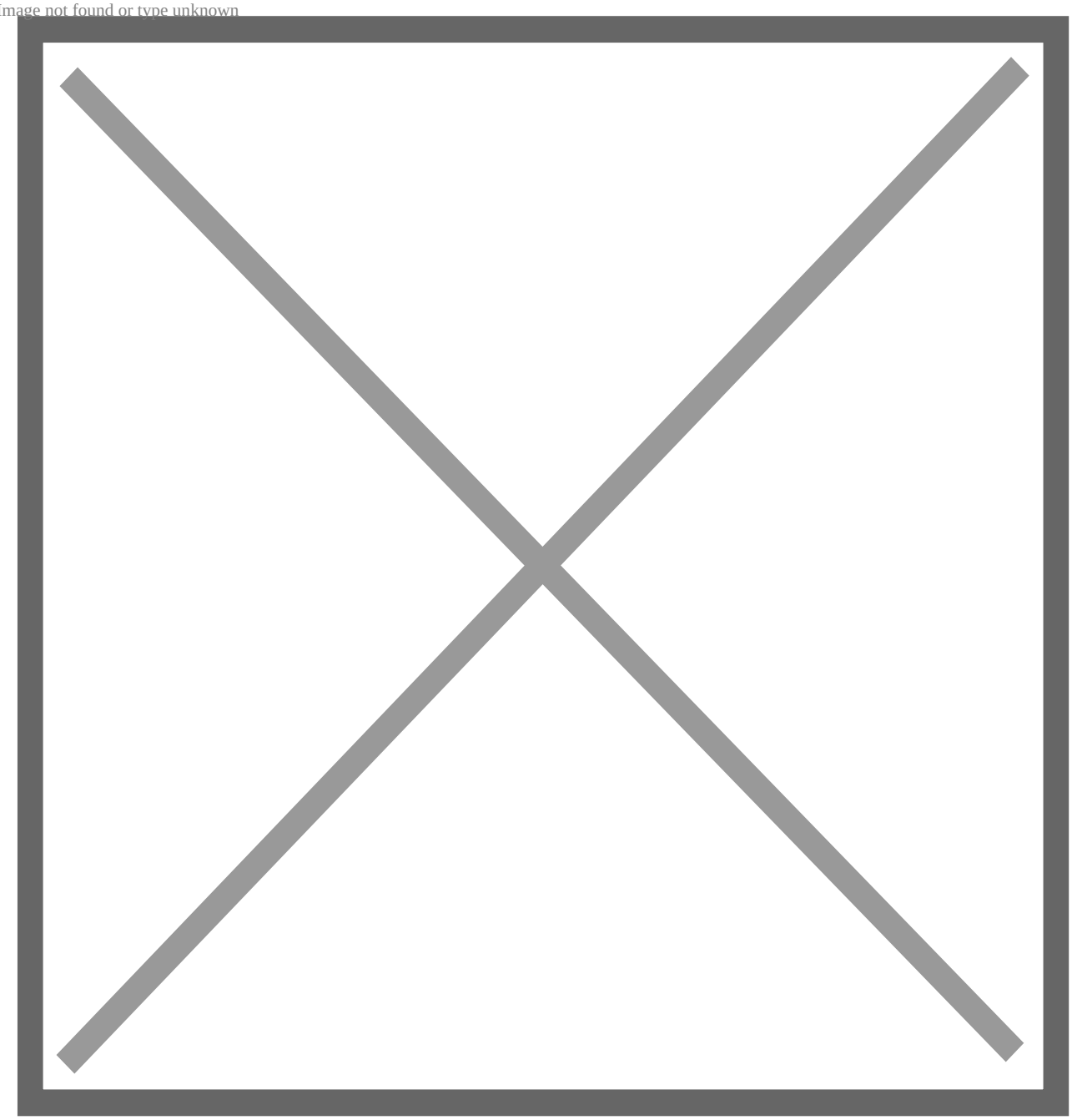
- **Более эффективно:** Благодаря GraphQL клиенты могут получить множество ресурсов с помощью одного запроса. Он использует только одну конечную точку для всех запросов, что позволяет использовать кэширование и пакетные запросы.
- **Ориентированность на клиента:** GraphQL позволяет клиенту контролировать получаемые данные, что уменьшает количество обращений к серверу.
- **Агностичность:** поскольку GraphQL не зависит от базовой базы данных и технологического стека, он может быть интегрирован с любой технологией бэкенда.
- **Интроспективный:** Поставляется с системой интроспекции, которая позволяет клиентам узнать о своих доступных данных, типах данных и ссылках.
- **Модель подписки:** Модель подписки позволяет пользователям получать обновления данных в режиме реального времени. Клиенты могут подписаться на изменения определенных данных и получать обновления при их изменении.

Теперь, когда вы знаете особенности GraphQL, давайте рассмотрим, что REST API может предложить своим клиентам.

Особенности REST API

В REST API большое внимание уделяется следованию набору стандартов, что делает API очень доступным, адаптируемым и легко масштабируемым:

Image not found or type unknown



- **Нестационарный:** RESTful API включает все необходимые данные в каждый запрос. Таким образом, он не имеет статических данных, масштабируем и прост в управлении.
- **Архитектура клиент-сервер:** Архитектура клиент-сервер, при которой клиенты запрашивают данные у сервера, а он их возвращает. Таким образом, внешняя и внутренняя системы могут быть построены и поддерживаться отдельно.
- **Ресурсно-ориентированная:** Доступные данные представлены ресурсами.

Для извлечения или изменения каждый ресурс имеет определенный URL.

- **Операции CRUD:** REST API управляет ресурсами посредством CRUD (Create, Retrieve, Update и Delete) действий.
- **Согласованный интерфейс:** Предлагает единый интерфейс для взаимодействия с ресурсами, упрощая архитектуру и обслуживание API.
- **Возможность кэширования:** Поддерживает кэширование, что уменьшает количество запросов к серверу и повышает эффективность.
- **Многослойная структура:** Поддерживает многоуровневую структуру, включающую прокси-сервер, что повышает гибкость и масштабируемость.

Преимущества GraphQL

После изучения особенностей GraphQL давайте рассмотрим преимущества, которые выделяют его на фоне других.

- **Улучшенная производительность:** GraphQL повышает производительность за счет минимизации количества данных, передаваемых по сети.
- **Упрощенная разработка API:** Разработка API становится проще благодаря единой охватывающей схеме. Следовательно, это упрощает процесс разработки и снижает вероятность ошибок.
- **Повышенная гибкость:** Разработчики могут точно описать необходимые им данные и то, как они должны быть организованы. Поэтому они могут использовать различные типы клиентов, например, мобильные и веб-приложения.
- **Улучшенный опыт разработчиков:** Предлагает фреймворки и инструменты, которые упрощают создание, тестирование и отладку API.
- **Улучшенная документация:** Благодаря самодокументирующимся схемам API становится проще понять и использовать.
- **Более быстрая итерация:** Обновления схемы могут быть сделаны без влияния на текущих клиентов. Таким образом, можно легко совершенствовать API и добавлять в него новые функции.
- **Более простая агрегация данных:** Пользователи могут объединять информацию из нескольких API и источников в один запрос. Таким образом, агрегация данных может быть упрощена за счет менее сложного кода бэкенда.

Приложения и сценарии использования: GraphQL

Когда речь заходит о создании и использовании API, GraphQL предлагает уникальное решение. При использовании по назначению он может стать идеальным инструментом для указанных ниже случаев применения:

Создание API

GraphQL обычно используется для разработки API, которые обеспечивают более быстрый метод доступа и получения данных. Он помогает разработчикам указать точные поля и структуру данных, которые они хотят запросить, что делает API более легким и быстрым.

Безголовая CMS

Когда речь идет о безголовой CMS, GraphQL может быть использован в качестве слоя данных. Он позволяет отделить контент от слоя отображения. Кроме того, безголовые CMS позволяют разработчикам эффективно и гибко извлекать контент и управлять им.

Разработка мобильных приложений

Поскольку мобильные приложения часто имеют ограниченную пропускную способность, быстрое получение данных становится необходимым. Именно здесь GraphQL становится идеальным инструментом для разработки мобильных приложений. Кроме того, он упрощает для разработчиков реализацию таких функций, как поддержка офлайн и кэширование.

Совместные приложения

Функция подписки в GraphQL очень важна для приложений, которые требуют участия пользователей и изменения данных в режиме реального времени. Таким образом, клиенты могут подписаться на обновления и получать немедленную передачу данных от сервера.

Микросервисы

В архитектуре микросервисов сервисы, как правило, требуют связи друг с другом и имеют различные требования к данным. GraphQL снижает эту сложность,предлагая единый интерфейс для получения данных из различных сервисов.

Электронная коммерция

Универсальность и эффективность GraphQL в получении и управлении данными о товарах может улучшить веб-сайты и приложения электронной коммерции. Это позволяет реализовать такие функции, как динамическое обновление информации о наличии товара, руководства по покупке для конкретного пользователя и специальные предложения.

Наука о данных

Гибкие и мощные возможности GraphQL по получению и анализу данных делают его жизнеспособной технологией для приложений науки о данных. Это облегчает разработчикам проведение расширенного анализа и моделирования данных из самых разных источников.

Социальные сети

С помощью GraphQL программисты могут запрашивать и манипулировать информацией о пользователях, статьями в блогах и другим контентом. Это позволяет динамично обновлять пользовательские ленты и создавать более персонализированный опыт для конечного пользователя.

Приложения и примеры использования: REST API

Вот некоторые ключевые приложения и случаи использования REST API:

Мобильное приложение

REST API – это отличный вариант для разработки бэкэнд-сервисов для мобильных приложений. Он просто извлекает данные из различных источников. Например, из баз данных, облачных хранилищ, онлайновых мобильных сервисов и т.д.

Веб-приложения

REST API оптимальны для создания веб-приложений, требующих доступа к данным из многочисленных источников. Он предлагает единый метод доступа и манипулирования данными, минимизируя сложность веб-приложений.

Интернет вещей (IoT)

Вы можете использовать RESTful API для связи устройств Интернета вещей (IoT) с облачным программным обеспечением. Например, умный термостат может взаимодействовать с облачной службой, регулирующей температуру в доме, используя REST API.

Веб-сайт электронной коммерции

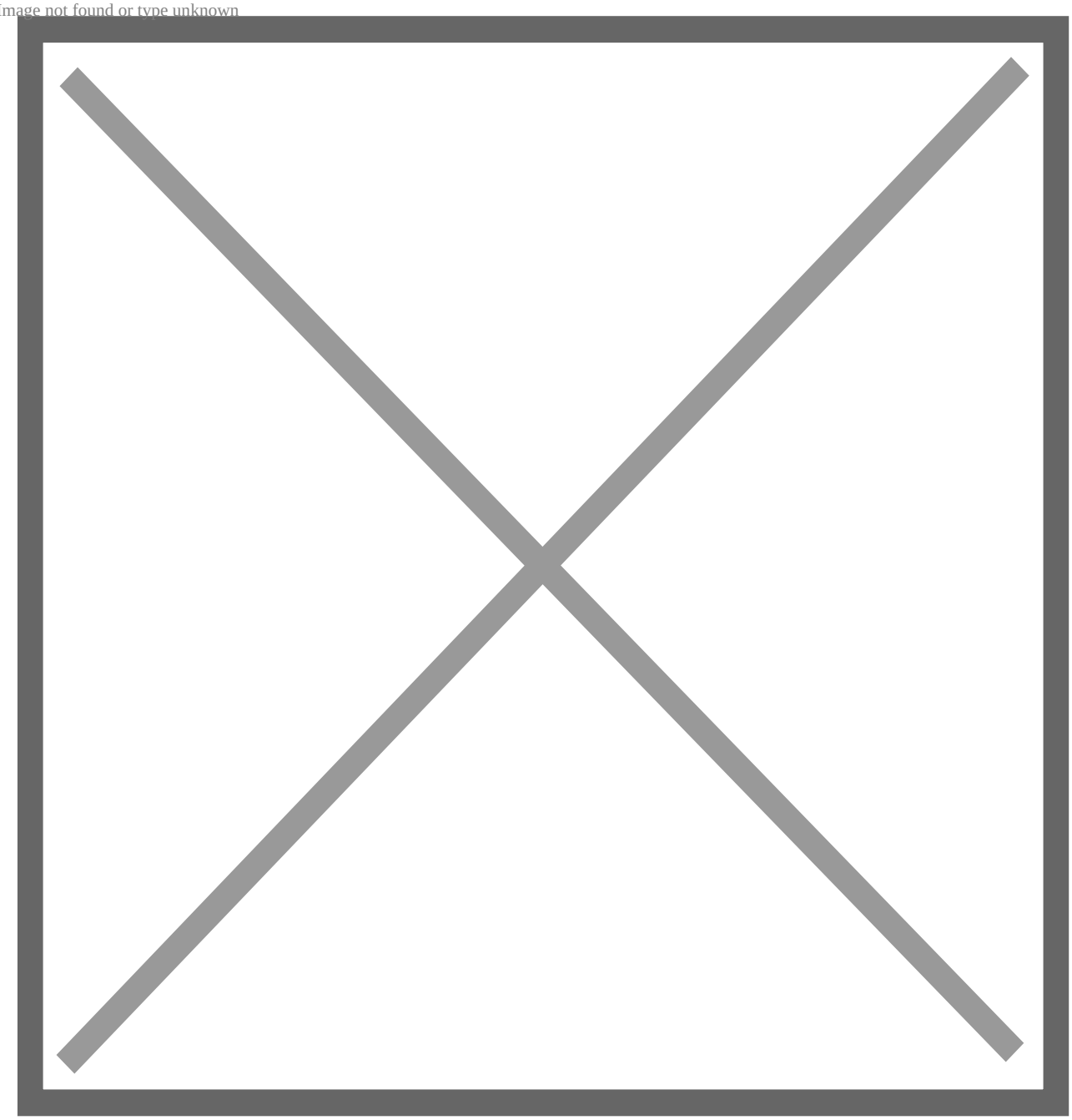
Для проведения транзакций и соединения со сторонними сервисами веб-сайты электронной коммерции часто используют RESTful API. Например, интернет-магазин может использовать RESTful API для получения данных о доставке от логистического провайдера или для приема платежей через платежный шлюз.

Платформы социальных сетей

RESTful API играют решающую роль в обеспечении организованного доступа к данным социальных сетей. Используя его, программисты могут получить доступ к данным пользователей с таких сайтов, как Twitter, Facebook и LinkedIn, для создания специализированных систем управления социальными сетями или собственных приложений.

GraphQL против REST API

Image not found or type unknown



Давайте сделаем краткий обзор различий между GraphQL и REST API:

Характеристика	**GraphQL**	**REST API**
Поиск данных	Клиенты могут запрашивать и получать только необходимые данные в различных форматах.	Клиенты могут запрашивать и получать все данные в заранее определенном формате.

Запрос и ответ на запрос данных	Может обрабатывать сложные данные и возвращать их из нескольких источников с помощью одного запроса.	Требуется несколько запросов к многочисленным ресурсам.
Гибкость запросов данных	Позволяет выполнять адаптируемые запросы, которые могут быть подобраны в соответствии с конкретными требованиями клиента.	Возможности модификации запросов относительно ограничены.
Функция кэширования	Поддерживается кэширование, что повышает производительность. Следовательно, нет необходимости в повторной обработке запросов.	Из-за предопределенного стиля возврата кэширование может быть более сложным.
Общая производительность	Эффективен для поиска высокочастотных данных.	Менее эффективен для высокочастотных данных.
Условие версионирования	Поскольку обновления схемы являются кумулятивными, версионность не нужна.	Это может привести к нерациональному использованию полосы пропускания и задержке времени отклика.
Кривая обучения	Более крутая кривая обучения, поэтому пользователь должен понять схему и метод запросов.	Упрощенный стиль запросов и ответов облегчает обучение и использование.

Документация	Достойное количество инструментов, документации и интеграции с IDE.	Ограниченные инструменты, документация и поддержка IDE.
Инструментарий	Расширяются вспомогательные программы, инструментарий и библиотеки.	Имеется хорошо отлаженный и надежный набор инструментов и ресурсов.

Примечание автора

GraphQL позволяет клиентам получать именно те данные, которые им нужны, за один запрос. Это отличный выбор для приложений со сложными требованиями к данным или высокочастотным получением данных. С другой стороны, REST API предлагает более надежную экосистему вспомогательного программного обеспечения и более прост в использовании. Он подходит для более простых приложений, требующих простоты использования. Кроме того, он включает в себя хорошо отлаженную экосистему инструментов и библиотек.

Заключительные размышления

Как вы можете видеть, API GraphQL и REST четко отличаются друг от друга своими преимуществами и недостатками. В целом, выбор между GraphQL и REST API – это вопрос предпочтений разработчика и требований приложения. Вы также можете изучить некоторые часто задаваемые вопросы и ответы на интервью по REST API.

Дата Создания
29.06.2023