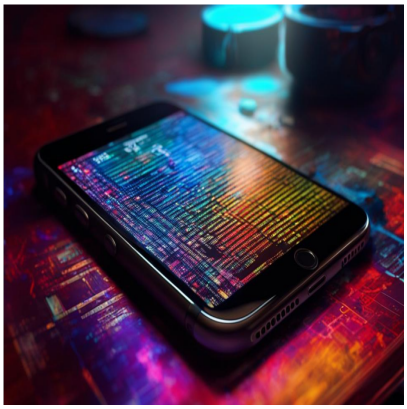


REST против gRPC: что выбрать для следующего проекта

Описание

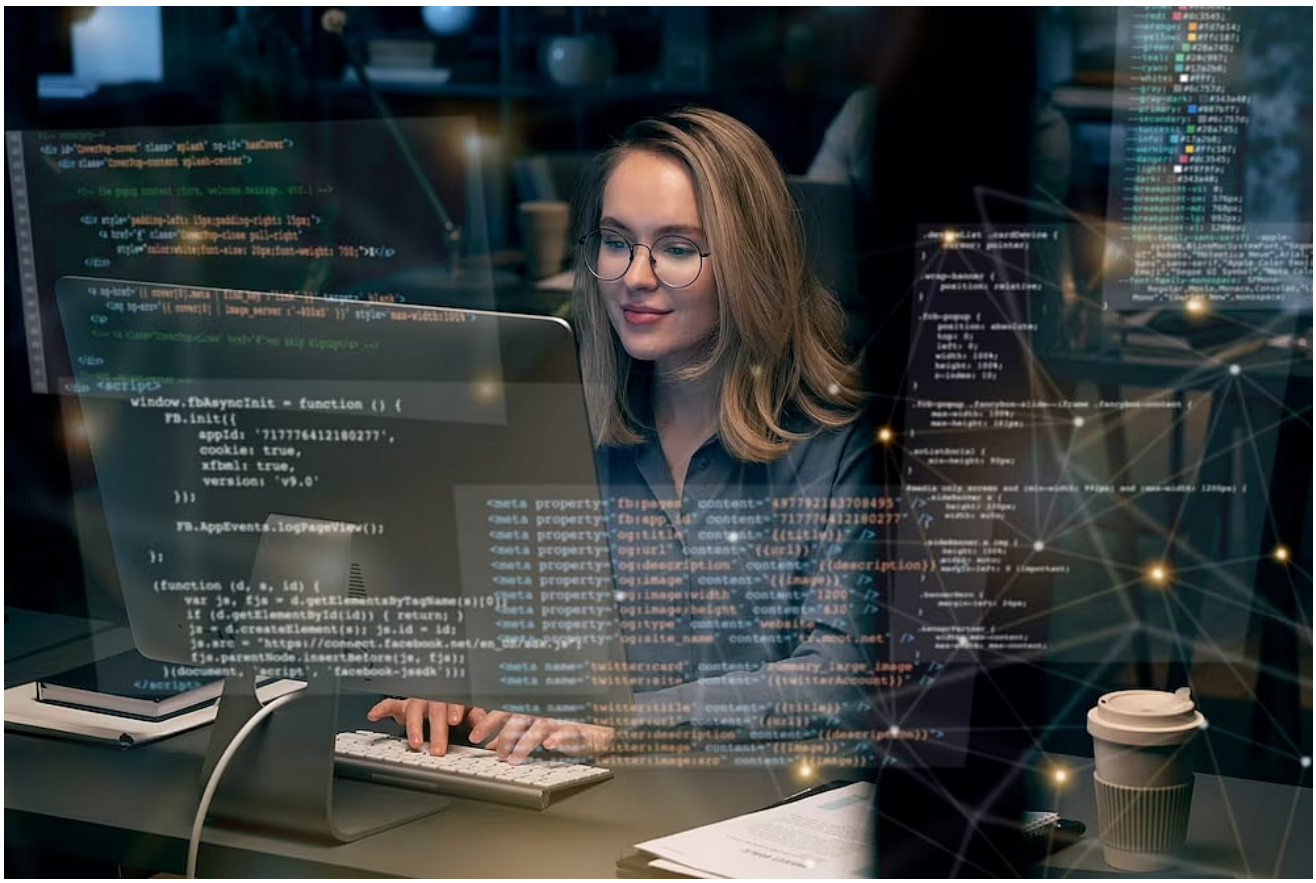
При работе с API вы часто сталкиваетесь с REST и gRPC. Несмотря на то, что Rest доминирует в этой области уже много лет, gRPC оказывается достойным конкурентом. Rest и gPRC – это два разных способа разработки API. API выступают в качестве коммуникационного моста между сервисами, которые могут представлять собой сложные системы, расположенные на разных компьютерах или написанные на разных языках. В этой статье мы познакомимся с Rest и gRPC, расскажем об их сходствах и различиях, а также о том, где можно использовать каждый из них.

Что такое rest?

Rest (Representational State Transfer) – это архитектурный программный подход, который диктует правила обмена данными между программными компонентами. Rest основан на стандартном протоколе обмена данными в Интернете – HTTP. Все API, основанные на архитектурном стиле REST, называются RESTful API. С другой стороны, веб-сервисы, основанные на архитектурном дизайне REST, называются RESTful веб-сервисами. Архитектурный стиль REST руководствуется этими принципами;

- **Унифицированный интерфейс:** Сервер должен передавать данные в стандартном формате. Однако передаваемые данные могут иметь формат, отличный от внутреннего представления ресурса серверного приложения.
- **Нестационарность:** Сервер должен выполнять каждый запрос клиента независимо от предыдущих запросов. Клиентские запросы на ресурсы могут быть в любом порядке, и каждый запрос изолирован от остальных.
- **Многоуровневая система:** Представляет собой слой авторизованных посредников между сервером и клиентом. Клиент может соединяться с этими авторизованными посредниками и при этом получать ответы от сервера.
- **Возможность кэширования:** Некоторые ответы хранятся у посредника или клиента, чтобы увеличить время отклика.
- **Код по требованию:** Серверы временно настраивают или расширяют функциональность клиента, передавая ему программный код.

Преимущества REST



- **Масштабируемость:** API REST известны своей масштабируемостью, поскольку они оптимизируют взаимодействие клиента и сервера. Кэширование и отсутствие статичности – основные особенности, снижающие нагрузку на сервер.
- **Гибкость:** RESTful API обеспечивают полное разделение клиента и сервера. Такие сервисы позволяют разделить и упростить различные серверные компоненты, которые могут развиваться независимо друг от друга.
- **Независимость:** Вы можете писать серверные и клиентские приложения на разных языках программирования без ущерба для дизайна API.

Варианты использования Rest

- Веб-интерфейсы
- Веб-сервисы
- Архитектура микросервисов

Что такое gRPC?

gRPC – это фреймворк для удаленного вызова процедур (RPC), который может работать в любой среде. Этот фреймворк с открытым исходным кодом разработан как высокопроизводительный протокол, который может эффективно соединять сервисы между центрами обработки данных и в них самих.

Клиентское приложение может вызвать метод серверного приложения на другой машине, как если бы это был локальный объект. В gRPC вы определяете службу и указываете методы, которые можно вызывать удаленно, с их параметрами и типами возврата. gRPC имеет подключаемую проверку работоспособности, аутентификацию, балансировку нагрузки и поддержку трассировки. Этот фреймворк использует HTTP 2 и буферы протоколов для передачи данных. При обмене данными вместо URL-адреса ресурса вызывается процедура.

Преимущества gRPC

- **Масштабируемость:** gRPC позволяет установить среду выполнения одной командой и начать масштабировать миллионы RPC в секунду.
- **Простое определение сервисов:** Используйте Protocol Buffers, чтобы определить свои службы и запустить их.

- **Кроссплатформенность:** Этот фреймворк генерирует идиоматические клиентские и серверные заглушки для различных платформ и языков.
- Двунаправленная потоковая передача и интегрированная авторизация.

Варианты использования gRPC

- Веб-интерфейсы
- Веб-сервисы
- Потоковые приложения
- Коммуникация микросервисов

REST и gRPC: сходства

- **Механизм обмена данными:** Оба архитектурных решения позволяют серверам и клиентам обмениваться данными. Однако обмен этими данными осуществляется на основе определенных правил.
- **Подходят для масштабируемых и распределенных систем:** Асинхронная связь и дизайн без статических данных как REST, так и gRPC позволяют легко масштабировать их API.
- **Используйте коммуникацию на основе HTTP:** Оба используют HTTP, наиболее предпочтительный протокол связи в Интернете.
- **Гибкость:** Вы можете использовать REST и gRPC с различными языками программирования и технологиями.

REST против gRPC

Службы REST и gRPC различаются следующим образом;

Обмен данными

В **REST** API данные, передаваемые от одного программного компонента к другому, должны быть выражены в формате JSON. JSON должен быть сериализован и переведен на язык программирования для обмена данными. Однако Rest API также могут обмениваться данными в таких форматах, как HTML и XML. По умолчанию **gRPC** использует формат Protocol Buffers. Однако он также поддерживает JSON. Буферы протокола не являются человекочитаемыми. Сервер использует язык описания интерфейса Protocol Buffer для определения структуры данных. Затем gPRC сериализует структуру данных в двоичный формат. Затем он десериализует данные в любой указанный язык программирования.

Модель коммуникации

В **REST** клиент посылает серверу один запрос, а сервер в ответ отправляет ответ. Клиент должен дождаться ответа сервера, прежде чем продолжить работу. Это модель “запрос-ответ”. В **gRPC** клиент может отправлять один или несколько запросов серверу, получая соответственно один или несколько ответов.Соединения данных могут быть многие-ко-многим, многие-один, один-ко-многимили один-один. gRPC использует модель взаимодействия клиент-ответ.

Генерация кода

В **gRPC** встроены встроенные функции генерации кода на стороне сервера и клиента. Вы можете найти эти возможности в различных языках благодаря компилятору Protocol Buffers. gRPC генерирует код на стороне сервера и клиента после определения структуры в файле proto. В REST отсутствуют встроенные функции генерации кода. Если вам нужна эта функция, вы можете использовать сторонние инструменты.

Протокол HTTP

API **REST** используют протокол HTTP 1.1. Чтобы отправить запрос на REST-сервис, вам нужен URL-адрес ресурса. По протоколу HTTP 1 информация передается между компьютером и веб-сервером. URL ресурса в REST-сервисе виден клиенту. Разработчики API контролируют структуру URL-адресов ресурсов. gRPC использует

HTTP 2. Эта версия HTTP была представлена в 2015 году и используется в таких браузерах, как Internet Explorer, Safari и Chrome. В отличие от HTTP 1, в котором все хранится в виде обычного текста, этот новый формат использует инкапсуляцию двоичного формата, что дает больше возможностей для доставки данных и ускоряет весь процесс.

Структура данных полезной нагрузки

REST использует XML или JSON для отправки и получения данных. JSON – наиболее часто используемый формат для отправки динамических данных в REST, поскольку он гибок и не требует какой-либо структуры. Данные JSON также читаемы человеком. Единственная проблема с JSON – это не очень высокая скорость, так как его необходимо сериализовать и транслировать во время передачи данных.

gRPC использует буферы протокола для сериализации данных полезной нагрузки. Это сильно сжатый формат, который уменьшает объем данных в сообщениях. Этот фреймворк использует Protobuf для автоматического преобразования сильно типизированных сообщений в языки программирования клиента и сервера.

Поддержка браузеров

REST поддерживается всеми браузерами, поскольку использует HTTP 1.1. Это делает его идеальным выбором для веб-сервисов и API.

gRPC имеет ограниченную поддержку браузеров, так как основан на HTTP 2. Для поддержки всех браузеров необходимо добавить gRPC-web в качестве прокси-уровня. По этой причине gRPC в основном используется для внутренних систем.

Сопряжение клиент-сервер

REST – это архитектурный дизайн со свободной связью. Это означает, что клиент и сервер не должны знать о реализации друг друга. Эта особенность облегчает развитие RESTful API с течением времени, поскольку вам не нужно менять код клиента при изменении определений сервера.

gRPC – это тесно связанный фреймворк, в котором сервер и клиент должны иметь доступ к одному и тому же файлу proto. Если вам нужно внести какие-либо изменения в этот файл, вы должны также обновить сервер и клиент.

Rest против gRPC: краткое сравнение

Характеристика	REST	gRPC
Протокол HTTP	HTTP 1.1	HTTP 2
Поддержка браузеров	Поддерживает все браузеры, так как использует HTTP 1.1	Меньшая поддержка браузерами, поскольку используется HTTP 2
Генерация кода	Использование инструментов сторонних производителей	Встроенные функции генерации кода
Подход к проектированию	Сущностно-ориентированный дизайн	Сервис-ориентированный подход
Доступ к данным	URL-адреса ресурсов	Сервисные вызовы
Двунаправленная передача данных	Недоступно	Доступно
Реализация	Для реализации REST на стороне клиента или сервера не требуется никакого общего программного обеспечения	Программное обеспечение gRPC необходимо как на стороне клиента, так и на стороне сервера.
Модель коммуникации	Один клиент взаимодействует с одним сервером	Несколько моделей взаимодействия: один клиент посылает запросы нескольким серверам, один сервер взаимодействует с несколькими клиентами или один сервер взаимодействует с одним клиентом.

Когда использовать REST

RESTful API и веб-сервисы очень популярны. RESTful-сервисы легко реализовать, структурировать данные, они гибкие и читаемые. Вы можете использовать REST в следующих случаях;

- **Веб-архитектуры:** Вы можете создавать веб-, мобильные и мультиплатформенные API, используя архитектурный дизайн REST.
- **Простой обмен данными:** REST использует JSON – легко читаемый формат данных.
- **API, ориентированные на общественность:** Если вы планируете, что публика будет потреблять данные и использовать ваш API, REST будет хорошим выбором благодаря своей читабельности.

Когда использовать gRPC

gRPC не так популярен, как RESTful-сервисы. Однако у него есть уникальные особенности, которые позволят ему выделиться в следующих приложениях;

- **Мультиязычные системы:** gRPC подходит для архитектур микросервисов, написанных на разных языках программирования, где API вряд ли будет меняться.

- **Подключения к микросервисам:** Такие возможности, как двунаправленная потоковая передача и низкая поддержка браузеров, делают gRPC хорошим выбором для внутренних API.
- **Сети с потоковой передачей данных в реальном времени:** Вы можете использовать gRPC с внутренними службами, которые работают с большими объемами данных и требуют потоковой передачи в реальном времени.

Авторское мнение

Несмотря на то, что gRPC имеет некоторые специфические особенности, которые могут превзойти REST в таких приложениях, как Интернет вещей, последний выигрывает за счет своей читабельности, гибкости и широкого распространения. Более слабая поддержка gRPC браузерами делает его не самым удачным выбором для разработчиков, которые хотят создавать веб-сервисы. Универсальная поддержка RESTful-сервисов делает REST идеальным архитектурным стилем API для интеграции веб- и микросервисов.

Заключение

REST и gRPC – одни из многих архитектурных стилей API, которые вы можете выбрать при создании своего следующего API. Окончательный выбор будет зависеть от продукта, который вы хотите создать. RESTful-сервисы отлично подойдут для создания публичных API, в то время как gRPC – хороший выбор для таких сервисов, как мобильные приложения, не требующие поддержки браузера.

Дата Создания
24.03.2024