GRAPHQL – ЯЗЫК ЗАПРОСОВ И МАНИПУЛИРОВАНИЯ ДАННЫМИ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

Роман ПОГОНЕЦ

Департамент Программная Инженерия и Автоматика, группа TI-196, Факультет Вычислительной Техники, Информатики и Микроэлектроники, Технический Университет Молдовы, Кишинев, Молдова

Автор Корреспондент: Роман ПОГОНЕЦ, e-mail: roman.pogonet@isa.utm.md

Научный руководитель: Дориан САРАНЧУК, DISA, FCIM, UTM

Аннотация: *Данная работа посвящена GraphQL. Она освещает историю этого популярного языка запросов, рассказывает о его преимуществах, недостатках и принципе работы. В статье будет рассмотрен пример, показывающий принцип работы GraphQL, а так же будет сделан вывод на основе рассмотренного материала.*
Ключевые слова: *GraphQL, язык запросов, данные, запрос, таблицы.*

Введение

GraphQL – это язык запросов и обработки данных, открытый для API и среда исполнения, предназначенный для выполнение запросов к существующим данным. GraphQL был разработан внутри компании Facebook в 2012 году, а затем выпущен в открытый доступ в 2015 году [1].

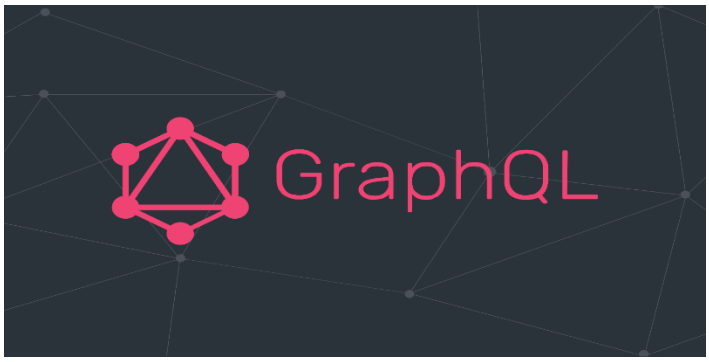

Рисунок 1. GraphQL

GraphQL был разработан с учётом гибкости, скорости и пожеланий разработчиков. GraphQL позволяет разработчикам создавать API по своему усмотрению и предоставлять функциональность API клиентам, используя спецификацию GraphQL.

Одна из ключевых концепций запросов GraphQL заключается в том, что возвращаемые данные предсказуемые. Вместо дополнительных строк кода возвращается, что было запрошено. Такая декларативная выборка данных особенно полезная на мобильных устройствах с ограниченной пропускной способностью: приложения, использующие GraphQL, также стабильны и быстры, поскольку они контролируют запрашиваемые и получаемые данные, а не сервер. Они также быстро работают на медленных сетевых соединениях, поскольку один запрос может одновременно возвращать несколько запрашиваемых данных [2].

Основываясь на упрощении данных GraphQL, API организован по типам и полям, а не по конечным точкам. Это означает, что все данные доступны из одной конечной точки, что позволяет приложениями запрашивать только то, что им нужно. Снижая эксплуатационную сложность, GraphQL присоединяется к рабочему процессу оптимизированных решений для подключения API.

Как работает GraphQL?

GraphQL поддерживает подробные запросы данных, предоставляя клиентам больше контроля над отправляемой информацией. Клиенты отправляют запросы GraphQL в виде запросов, а сервер возвращает ответы в формате JSON. GraphQL не требует специальной архитектуры приложения, может использоваться в различных средах (включая интегрированные среды разработки [IDE]) и может применяться с существующими инструментами управление API или поверх существующих RESP API.

Важные термины в GraphQL включает:

Схема – Созданная разработчиком API для указания того, какие данные могут быть запрошены клиентом, схема состоит из типов объектов, которые определяют, что может быть запрошено клиентом, и полей, которые представляют атрибуты объекта.

Запрос – запросы проверяются и зачем выполняется на основе схемы: GraphQL не позволяет выполнять запросы без определения типов объектов.

Преобразователь – это преобразователя, связанная с каждым полем в схеме, вызывается для генерации значения при выполнении API. Преобразователи являются важным архитектурным элементом GraphQL.

Одной из уникальных особенностей GraphQL являются то, что ответ отражает структуру запроса. Это упрощает анализ клиента, поскольку формат ответа сервера полностью предсказуем [3].

Основные различие между REST API и GraphQL

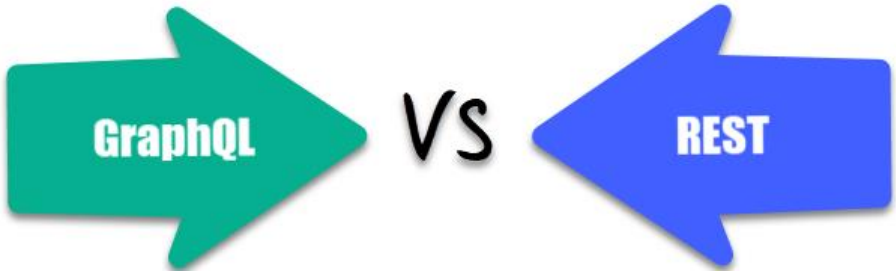

Рисунок 2. GraphQL vs Rest

GraphQL

GraphQL – это серверная технология прикладного уровня, разработанная компанией Facebook которая выполняет запросы к существующим данным он следует архитектуре, управляемой клиентом, может быть организовать в виде схемы, имеет растущее сообщество, высокая скорость разработки, кривое обучение сложное для понимания, строго типизирован, имеет единственную конечную точку, использует метаданные для проверки запросов [4].

REST

Rest – это стиль архитектурный программного обеспечения, определяющий набор ограничений для построение веб-сервисов. Используя серверно-управляемую архитектуру, Rest может быть организован с точки зрения конечных точек, большое сообщество, медленная скорость разработки, умеренная для понимание, слабо типизирован, несколько конечных точке, нет возможности кэширование машиночитаемых метаданных [5].

Преимущества GraphQL

- Предоставляет язык запросов, который является декларативным, а не императивным.
- Он иерархичен и ориентирован на продукт.
- GraphQL строго типизирован. Это означает, что запросы выполняются в определенной системе контексте.
- Запросы в GraphQL кодируются на клиенте, а не на сервере.
- Он обладает всеми возможностями прикладного уровня модели OSI.

- GraphQL обеспечивает читабельные запросы.
- В GraphQL легко интегрирует со многими базами данных.
- Данные могут быть получены с помощью одного вызова API.
- Помогает в пакетной обработке запросов и кэширование.
- Настраивает запросы в соответствии с вашими потребностями.
- Помогает находить формы в нужном формате
- GraphQL автоматически синхронизирует документацию с изменениями API.
- Эволюция API возможна без управления версиями.
- Его можно использовать для быстрого прототипирования приложений.
- Поля GraphQL могут быть переданы на более высокий уровень компонента для повторного использования.
- Это позволяет вам выбирать, какие функции выставлять и как они работают [6].

Недостатки GraphQL

- Недостаток ресурсов на бэкэнд части.
- Отсутствует шаблонов проектирования для сложного приложения.
- Проблемы с производительностью при сложных запросах.
- Слишком много для небольших приложений.
- Не полагается на метод кэширования HTTP, позволяющих сохранять содержимое запроса.
- Вам необходимо изучить язык определения схемы GraphQL, прежде чем внедрять стратегии GraphQL.
- GraphQL использует одну конечную точку вместо кэширования HTTP.
- Не лучшее решение для простых приложений, поскольку оно может добавить сложности [7].

Заключение

Реализация GraphQL может быть сложной, но это впечатляющий шаг вперед в обработке дынных. GraphQL не решает все проблемы, но это определенно технология, которую стоит попробовать. Например можно начать с размышлений о наиболее запутанных и неудобных подсистем обработки данных в вашем проекте и попытаться реализовать эти подсистемы на GraphQL.

Библиография

1. Подробности о GraphQL [online]. [дата обращения 10.03.2023], Доступно: https://habr.com/ru/company/ruvds/blog/445268/
2. Введение в GraphQL [online]. [дата обращения 11.02.2023], Доступно: https://graphql.org/learn/
3. Что такое GraphQL [online]. [дата обращения 11.02.2023], Доступно: https://www.redhat.com/en/topics/api/what-is-graphql
4. GraphQL vs Rest [online]. [дата обращения 12.02.2023], Доступно: https://www.guru99.com/graphql-vs-rest-apis.html#:~:text
5. GraphQL против Rest [online]. [дата обращения 12.02.2023], Доступно: https://www.rubrik.com/blog/technology/19/11/graphql-vs-rest-apis
6. GraphQL [online]. [дата обращения 12.02.2023], Доступно: https://hygraph.com/academy/what-is-graphql
7. Что такое GraphQL [online]. [дата обращения 12.02.2023], Доступно: https://www.nginx.com/resources/glossary/graphql/