GraphQL

Юлия МЕДЕЛЯН

Departamentul Ingineria Software și Automatică, grupa TI-202FR, Facultatea Calculatoare, Informatică și Microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Moldova

Автор-корреспондент: Меделян Юлия: iulia.medelean@isa.utm.md

Аннотация: *Статья посвящена анализу и исследованию GraphQL, который представляет собой язык запросов для API, а так же является средой выполнения этих запросов. GraphQL позволяет клиентам запросить только те данные, которые им необходимы, и обеспечивает таким образом более эффективное взаимодействие между клиентом и сервером. Показаны преимущества технологии GraphQL в контексте создания современных веб-приложений. Дано сравнение GraphQL с Rest - двух разных подходов к разработке API. Указаны их общие характеристики и преимущество использования GraphQL, что способствует увеличению независимости между front-end и back-end разработчиками, поскольку front-end может формировать запросы и комбинировать данные, не требуя изменений от back-end разработчиков после определения схемы данных.*

Введение

GraphQL - это язык запросов и манипулирования данными для API, а также среда выполнения этих запросов. Разработанный в 2012 году внутри Facebook, он стал открытым в 2015 году и с 2018 года управляется GraphQL Foundation [1]. Проект получил популярность благодаря использованию в продуктах, таких как Facebook, Airbnb, GitHub и других.

Использование GraphQL

Создание GraphQL в Facebook было мотивировано необходимостью эффективной работы с разнородными данными и преодоления ограничений REST-архитектуры. Например, при работе с объемными и разнородными данными, как в случае соцсетей, использование REST-эндпоинтов становится неудобным. GraphQL предлагает "умный" единственный эндпоинт, способный обрабатывать сложные запросы и возвращать данные в нужной форме.

Схемы GraphQL API

Основой GraphQL API является схема данных, описывающая типы данных и структуру ответов на запросы. Эта схема служит контрактом между сервером и клиентом, определяя доступные типы объектов, поля и их взаимосвязи [2]. При использовании GraphQL клиенту не важно, откуда и как получаются данные, поскольку он просто отправляет запрос, а сервер возвращает результат, соответствующий схеме.

Взаимосвязь клиента и сервера при работе с GraphQL. Тут я хотела бы остановиться на том, как, собственно, устроена работа клиента и сервера при использовании GraphQL. Так как я не back-end-разработчик, то расскажу только вкратце о работе с серверной частью, не вдаваясь в подробности.

GraphQL обеспечивает поддержку на различных платформах, таких как веб, Android, iOS и другие. Клиент GraphQL формирует запрос данных или запрос на их изменение в соответствии со схемой и отправляет его на GraphQL-сервер. Этот сервер, являясь HTTP-сервером, связан с определенной схемой GraphQL. Весь трафик запросов от клиента и ответов проходит через эту схему.

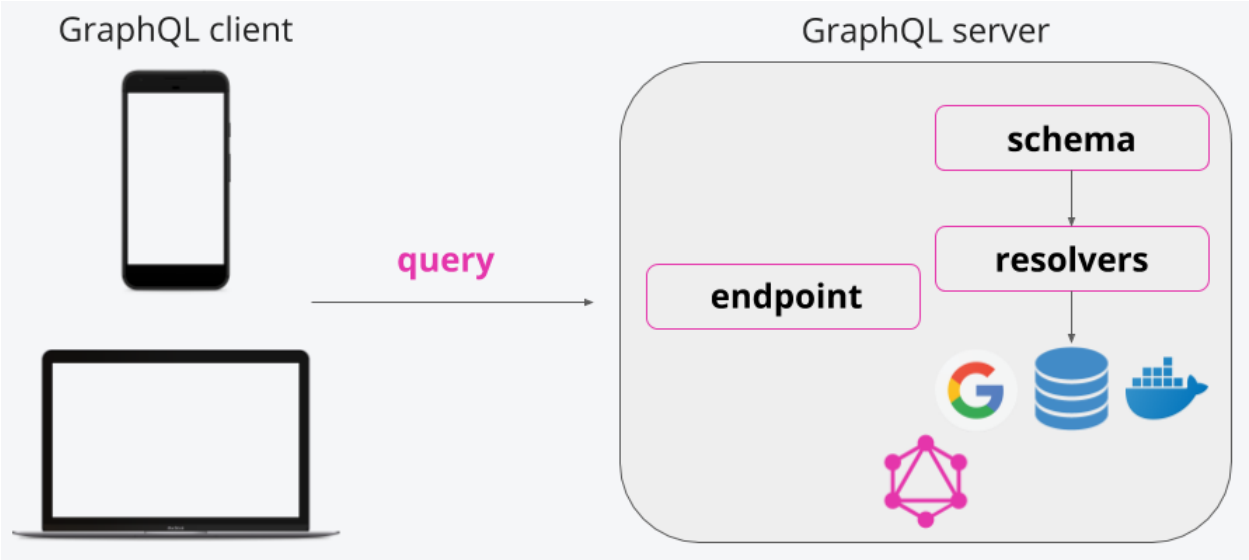

Рис. 1. Представление GraphQL

GraphQL-сервер не знает, как обработать запрос, пока не получит инструкции от специальных функций. Эти функции, называемые резолверами или распознавателями, связаны с соответствующими полями в схеме. Они определяют, как получить данные для запрошенных полей. В результате сервер возвращает клиенту ответ, структура которого соответствует запросу, обычно в формате JSON. Важно отметить, что GraphQL позволяет работать с различными источниками данных, включая базы данных (реляционные/NoSQL), результаты веб-поиска, Docker и другие.

Общие характеристики с Rest и преимущества

GraphQL и REST являются двумя разными подходами к разработке API, но они также имеют некоторые общие характеристики: [3]

1. Единый интерфейс:** Оба предоставляют единый интерфейс для работы с удаленными данными.
2. Формат ответа:** Оба чаще всего возвращают ответы в формате JSON.
3. Дифференциация запросов:** И GraphQL, и REST позволяют различать запросы на чтение и запись данных.

Однако у GraphQL есть несколько преимуществ по сравнению с REST:

1. Клиентоориентированный:** GraphQL позволяет клиенту запросить только ту информацию, которая ему нужна, определяя объем данных.
2. Единственный endpoint:** В отличие от REST, GraphQL требует только одного единственного endpoint, что упрощает конфигурацию и обслуживание.
3. Сильная типизация:** GraphQL является сильно типизированным языком, что позволяет проверить правильность запроса на этапе разработки.
4. Комбинирование запросов:** GraphQL предоставляет возможность комбинировать несколько запросов в один, что уменьшает количество сетевых запросов.
5. Гарантированный результат запроса:** Запросы к GraphQL API всегда возвращают ожидаемый результат, соответствующий схеме данных.

Использование GraphQL также позволяет уменьшить объем передаваемых данных на клиент, поскольку клиент запрашивает только необходимую информацию. Это также способствует увеличению независимости между front-end и back-end разработчиками, поскольку front-end может формировать запросы и комбинировать данные, не требуя изменений от back-end разработчиков после определения схемы данных.

Заключение

В заключение, GraphQL представляет собой концепцию построения API, которая способствует более слабой связности между клиентом и сервером. Важно отметить, что внедрение GraphQL не обязательно означает полный отказ от REST-архитектуры.

Мой взгляд заключается в том, что если структура данных в проекте проста, переход к GraphQL может быть необязательным и зависит от предпочтений разработчиков. Однако в случае работы с разнообразными и объемными данными GraphQL действительно упрощает задачи. Важно помнить, что GraphQL не требует переработки всего существующего кода, и выбор между GraphQL и REST зависит от индивидуальных предпочтений. В целом, я считаю, что ознакомление с GraphQL является ценным опытом, который может пригодиться в различных проектах.

Литература

[1] GraphQL: язык запросов для современных веб-приложений, Алекс Бэнкс, Ева Порселло, 2019г, 240стр
[2] "Разработка веб-приложений GraphQL с React, Node.js и Neo4j" [Online]. Available: https://www.labirint.ru/books/929724/
[3] GraphQL Eine Einführung in APIs mit GraphQL, Dominik Kress, 2020г, 202стр