# Что такое GraphQL?

GraphQL — это язык запросов, используемый клиентскими приложениями для работы с данными. C GraphQL связано такое понятие, как «схема» — это то, что позволяет организовывать создание, чтение, обновление и удаление данных в вашем приложении (то есть — перед нами четыре базовые функции, используемые при работе с хранилищами данных, которые обычно обозначают акронимом CRUD — create, read, update, delete).

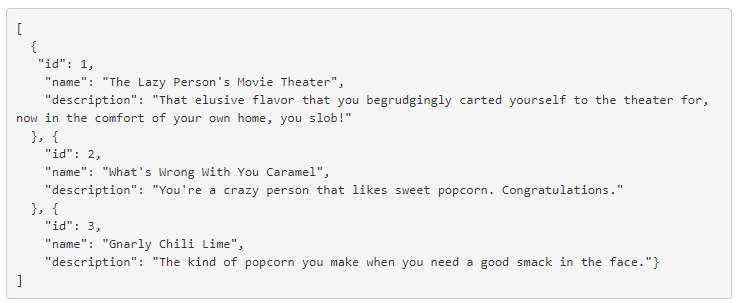
Выше было сказано, что GraphQL используется для работы с данными в «вашем приложении», а не «в вашей базе данных». Дело в том, что GraphQL — это система, независимая от источников, данных, то есть, для организации её работы неважно — где именно хранятся данные.

Данная технология представляет собой приукрашенные GET или POST запросы. В то время как GraphQL, в целом, вводит некоторые новые концепции, касающиеся организации данных и взаимодействия с ними, внутренние механизмы этой технологии полагаются на HTTP-запросы.

# Переосмысление технологии REST

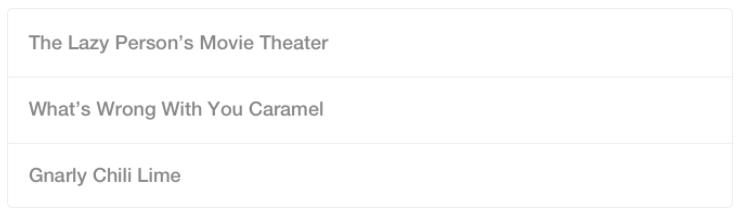
Гибкость — это то, что отличает технологию GraphQL от широко известной технологии REST. При использовании REST, если всё сделано правильно, конечные точки обычно создают с учётом особенностей некоего ресурса или типа данных приложения.

Например, при выполнении GET-запроса к конечной точке /api/v1/flavors ожидается, что она отправит ответ, выглядящий примерно так:



В таком ответе ничего катастрофически неправильного нет, но подумаем о пользовательском интерфейсе, или скорее о том, как мы намереваемся потреблять эти данные.

Если мы хотим вывести в интерфейсе простой список, который содержит лишь названия имеющихся видов попкорна (и ничего другого), то этот список может выглядеть так, как показано ниже.



На самом деле, то, что сервер отправляет в ответ на запрос клиента ненужные данные, это не полностью наша вина. REST — это механизм получения данных, который можно сравнить с рестораном, в котором официант спрашивает посетителя: «Чего вы хотите?», и, не особенно обращая внимание на его пожелания, говорит ему: «Я принесу вам то, что у нас есть».

Если же отбросить в сторону шутки, то в реальных приложениях подобное может вести к проблемным ситуациям. Например, мы можем выводить различные дополнительные сведения о каждом виде попкорна, наподобие сведений о цене, информации о производителе или диетологических сведений. При этом негибкие конечные точки REST сильно усложняют получение специфических данных о конкретных видах попкорна, что ведёт к неоправданно высокой нагрузке на системы и к тому, что получающиеся решения оказываются далеко не такими, которыми разработчики могли бы гордиться.

# Как технология GraphQL улучшает то, для чего использовалась технология REST

При поверхностном анализе вышеописанной ситуации может показаться, что перед нами всего лишь незначительная проблема. «Что плохого в том, что мы отправляем клиенту ненужные данные?». Для того чтобы понять масштабы, в которых «ненужные данные» могут стать большой проблемой, вспомним о том, что технология GraphQL была разработана компанией Facebook. Этой компании приходится обслуживать миллионы запросов в секунду.

Что это значит? А то, что при таких объёмах значение имеет каждая мелочь.

GraphQL, если продолжить аналогию с рестораном, вместо того, чтобы «нести» посетителю «то, что есть», приносит именно то, что посетитель заказывает.

Мы можем получить от GraphQL ответ, ориентированный на тот контекст, в котором используются данные. При этом нам не нужно добавлять в систему «одноразовые» точки доступа, выполнять множество запросов или писать многоэтажные условные конструкции.

# Как работает GraphQL?

Как мы уже говорили, GraphQL, для передачи данных клиенту и получения их от него, полагается на простые GET или POST-запросы. Если подробнее рассмотреть эту мысль, то оказывается, что в GraphQL есть два вида запросов. К первому виду относятся запросы на чтение данных, которые в терминологии GraphQL называются просто запросами (query) и относятся к букве R (reading, чтение) акронима CRUD. Запросы второго вида — это запросы на изменение данных, которые в GraphQL называют мутациями (mutation). Они относятся к буксам C, U и D акронима CRUD, то есть — с их помощью выполняют операции создания (create), обновления (update) и удаления (delete) записей.

Все эти запросы и мутации отправляют на URL GraphQL-сервера, который, например, может выглядеть как https://myapp.com/graphql, в виде GET или POST-запросов.

# Зачем может понадобиться использовать GraphQL?

Как и в случае с любой новой технологией, после первого знакомства с GraphQL вы можете задаться вопросом о том, зачем вам может пригодиться нечто подобное. Честно говоря, на этот вопрос нельзя дать однозначного и простого ответа. Очень уж много всего нужно учесть для того, чтобы такой ответ найти. И можно, кстати, подумать о том, чтобы вместо GraphQL просто выбрать проверенную временем технологию REST или напрямую обращаться к базе данных. Собственно говоря, вот несколько идей, над которыми стоит поразмыслить в поисках ответа на вопрос о том, нужна ли вам технология GraphQL.

# -Вы стремитесь уменьшить количество запросов, выполняемых с клиента

Многие приложения страдают от того, что им приходится выполнять слишком много HTTP-запросов, от того, что делать это приходится слишком часто, и от того, что это — сложные запросы. В то время как использование технологии GraphQL не позволяет полностью отказаться от выполнения запросов, эта технология, если ей правильно пользоваться, способна значительно уменьшить количество запросов, выполняемых со стороны клиента (во многих случаях для получения некоего набора связанных данных достаточно лишь одного запроса).

Является ли ваш проект приложением с множеством пользователей, или приложением, обрабатывающим огромные объёмы данных (например — это нечто вроде системы для работы с медицинскими данными), использование GraphQL определённо улучшит производительность его клиентской части.

# -Вы хотите избежать денормализации данных, проводимой лишь ради того, чтобы оптимизировать работу механизмов построения пользовательского интерфейса

В приложениях, в которых используются большие объёмы реляционных данных, часто может возникать «ловушка денормализации». Хотя такой подход и оказывается рабочим, он, вне всякого сомнения, далёк от идеала. Его применение может плохо влиять на производительность систем. Благодаря использованию GraphQL и вложенных запросов необходимость в денормализации данных значительно уменьшается.

# -У вас есть множество источников информации, к которым вы обращаетесь из разных приложений

Эта проблема может быть частично решена с помощью традиционных REST API, но даже при таком подходе одна проблема всё ещё остаётся: единообразие запросов, выполняемых с клиентской стороны. Предположим, что в ваш проект входят веб-приложение, приложения для iOS и Android, а также API для разработчиков. В подобных условиях вам, вероятнее всего, придётся, на каждой платформе, «мастерить из подручных материалов» средства для выполнения запросов.

Это ведёт к тому, что приходится поддерживать, на разных платформах, несколько реализаций HTTP, это означает отсутствие единообразия в средствах выполнения запросов и наличие запутанных конечных точек API (вы, наверняка, такое уже видели).

# - Может быть технология GraphQL — это верх совершенства? Стоит ли мне прямо сейчас выбросить мой REST API и перейти на GraphQL?

Нет, конечно. Ничто не совершенно. И, надо отметить, работать с GraphQL не так уж и просто. Для того чтобы создать работающую схему GraphQL, нужно выполнить множество обязательных шагов. Так как вы только изучаете данную технологию, это может вывести вас из равновесия, так как нелегко бывает понять то, чего именно не хватает в вашей схеме для правильной работы системы. При этом сообщения об ошибках, возникающих на клиенте и на сервере, могут оказаться не особенно полезными.

Далее, использование GraphQL на клиенте, в том, что выходит за рамки языка запросов, не стандартизовано. Хотя работу с GraphQL могут облегчить различные библиотеки, самыми популярными из которых являются Apollo и Relay, каждая из них отличается собственными специфическими особенностями.

GraphQL — это, кроме того, всего лишь спецификация. Пакеты вроде graphql (этот пакет используется внутри пакета express-graphql, применённого в нашем примере) — это всего лишь реализации данной спецификации. Другими словами, разные реализации GraphQL для разных языков программирования могут по-разному интерпретировать спецификацию. Это может привести к возникновению проблем, идёт ли речь о разработчике-одиночке, или о команде, в которой, при работе над разными проектами, используются разные языки программирования.

# Итоги

Несмотря на то, что внедрение GraphQL может оказаться непростой задачей, эта технология представляет собой впечатляющий шаг вперёд в сфере обработки данных. GraphQL нельзя назвать лекарством от всех болезней, но с этой технологией, определённо, стоит поэкспериментировать. Начать можно, например, поразмыслив о самой запутанной и неопрятной подсистеме, используемой в вашем проекте при работе с данными, и попытавшись реализовать эту подсистему средствами GraphQL.

GraphQL можно реализовывать инкрементно. Для того чтобы извлечь выгоды из применения этой технологии нет нужды переводить на GraphQL абсолютно всё. Так, постепенно вводя в проект GraphQL, можно разобраться с этой технологией самому, заинтересовать команду, и, если то, что получится, всех устроит, двигаться дальше.

GraphQL — это, в конечном счёте, всего лишь инструмент. Применение GraphQL не означает необходимости в полной переработке всего, что было раньше. При этом надо отметить, что GraphQL — это технология, с которой, определённо, стоит познакомиться. Многим стоит подумать и о применении этой технологии в своих проектах. В частности, если ваши проекты кажутся не особенно производительными, если вы занимаетесь разработкой сложных интерфейсов, наподобие панелей управления, лент новостей или профилей пользователей, то вы уже знаете о том, где именно вы можете опробовать GraphQL.