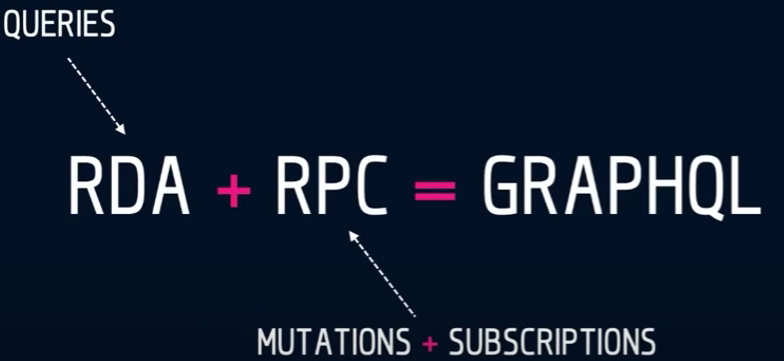
**GraphQL**

GraphQL — это язык запросов для API, который в последнее время приобрел значительную популярность. Он был разработан внутри Facebook в 2012 году до публичного выпуска в 2015 году. GraphQL позволяет клиентам определять структуру требуемых данных. Сервер возвращает именно эту структуру

GraphQL это синтаксис, который описывает как запрашивать данные, и, в основном, используется клиентом для загрузки данных с сервера. GraphQL имеет три основные характеристики:

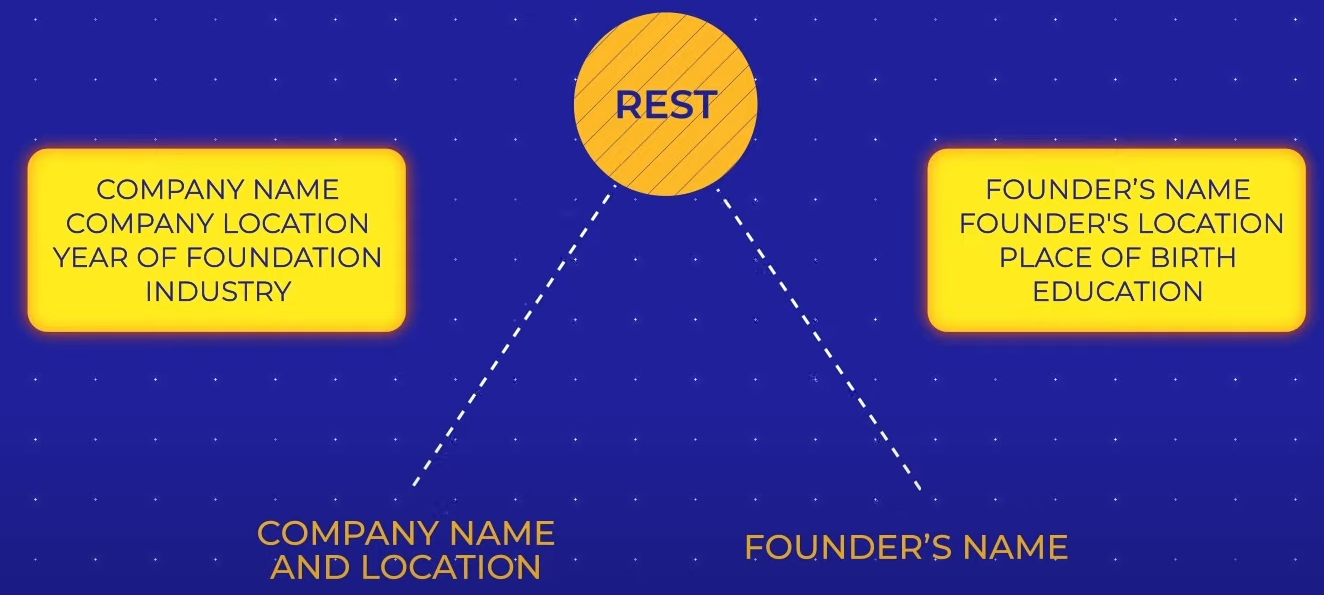
* Позволяет клиенту точно указать, какие данные ему нужны.
* Облегчает агрегацию данных из нескольких источников.
* Использует систему типов для описания данных.



С помощью этой технологии можно запрашивать только те значения, которые нам нужны. В отличие от других API

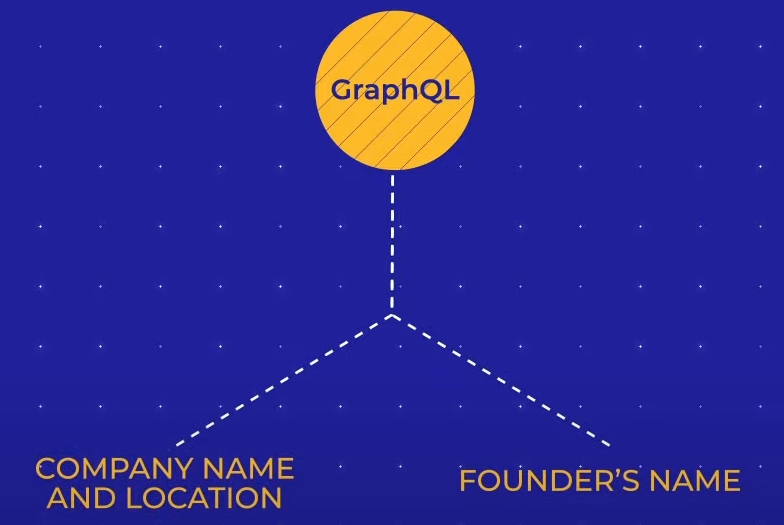
REST APIs have shortcomings. RESTful APIs don't know what specific information you might be looking for, and so when you call a REST API endpoint, it gives back all the information it has. This can mean that we are sending extra information that you don't need, or it can mean that we aren't sending all the information you need and that you must call multiple endpoints to get what you want. Either of these cases can slow things down, and for big applications with many users, that can become problematic. GraphQL was designed to deal with these issues.

Например если нам нужно узнать только название компании, ее расположение и имя основателя. То REST API сделает 2 запроса. В первом запросе вернется вся информация о компании, а во втором все основатели. И после нам нужно будет вручную это все отфильтровывать

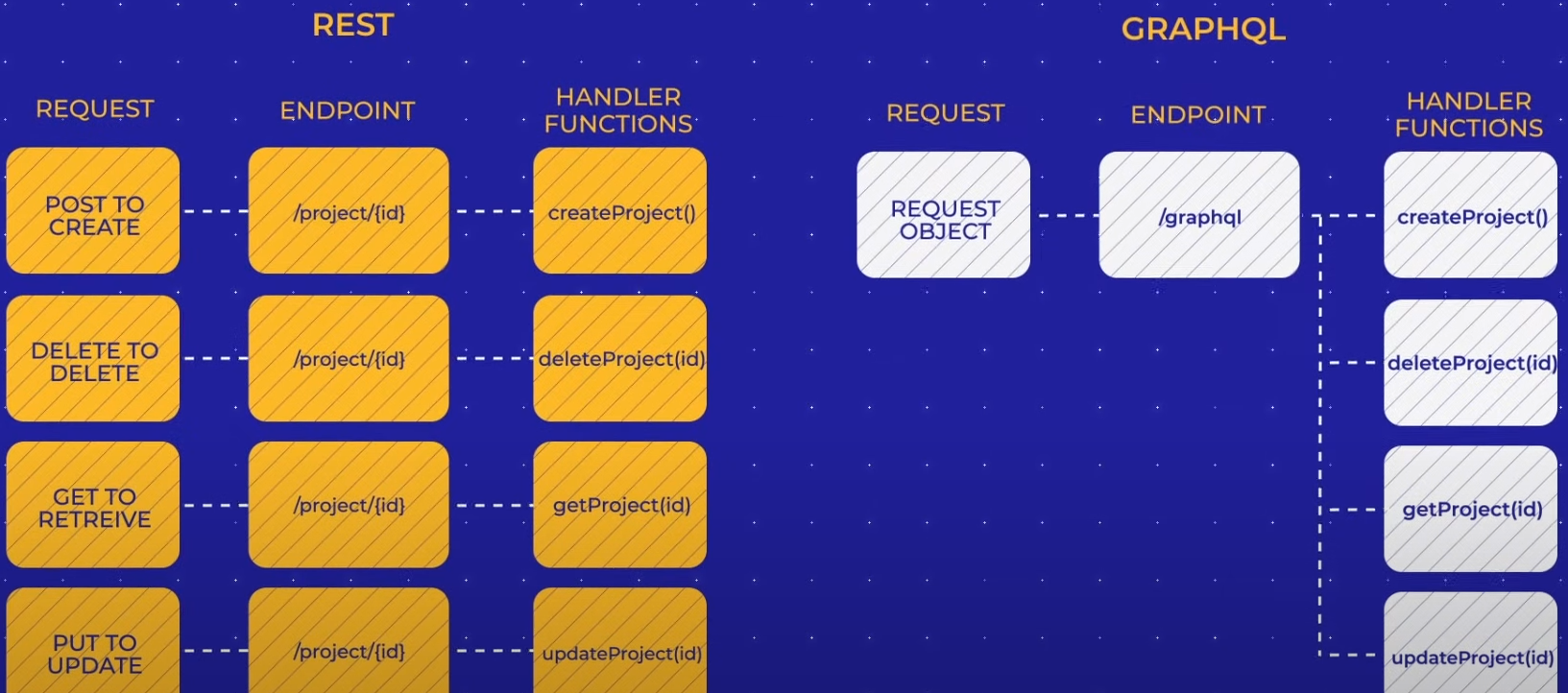


GraphQL is a query language for APIs, and so it requires you to specify in a query what you are looking for. With REST APIs, you will usually only need to know what the different endpoints are in order to find the information you are looking for, but with a GraphQL API, a single endpoint will contain most or all of the information you need and you will use queries to filter down that information to only the bits that you are interested in. This means that with GraphQL APIs, you will need to know the schema or structure of the data so that you know how to properly query it, instead of needing to know what all the endpoints are.

В GraphQL нужно составить всего 1 запрос, указав значения для фильтрации

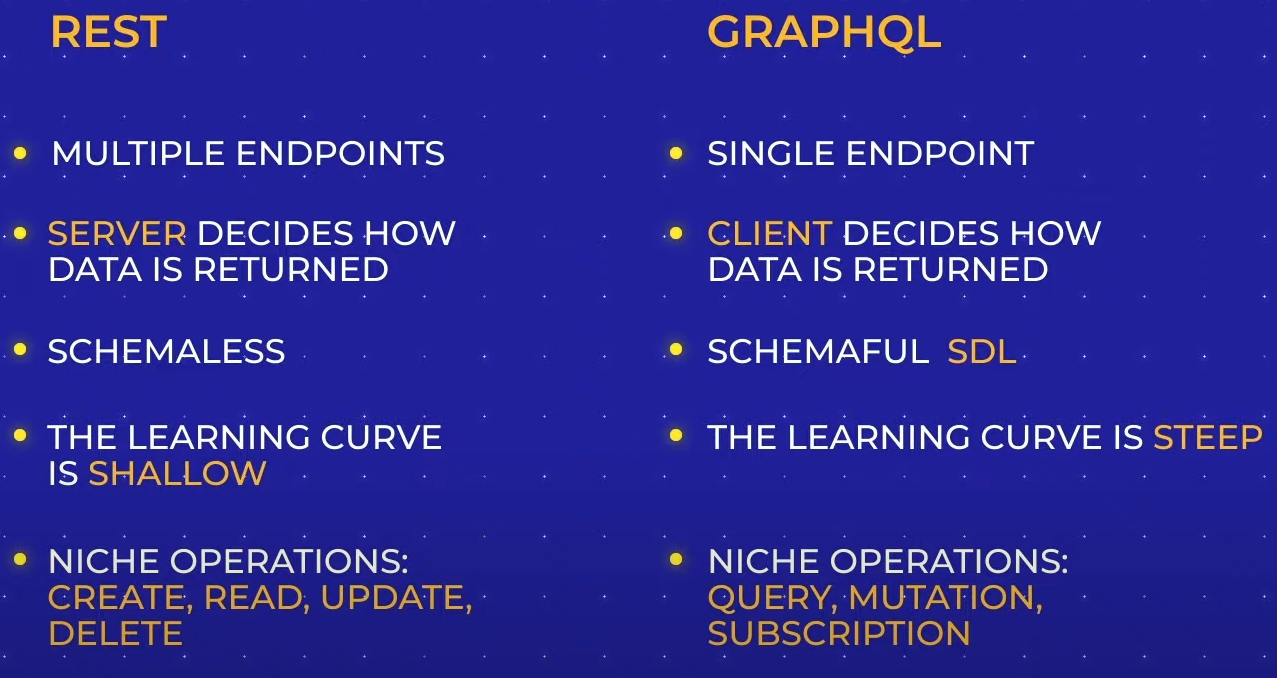


**Endpoints in REST vs GraphQL**



В REST для определенных значений, действий существует свой Endpoint

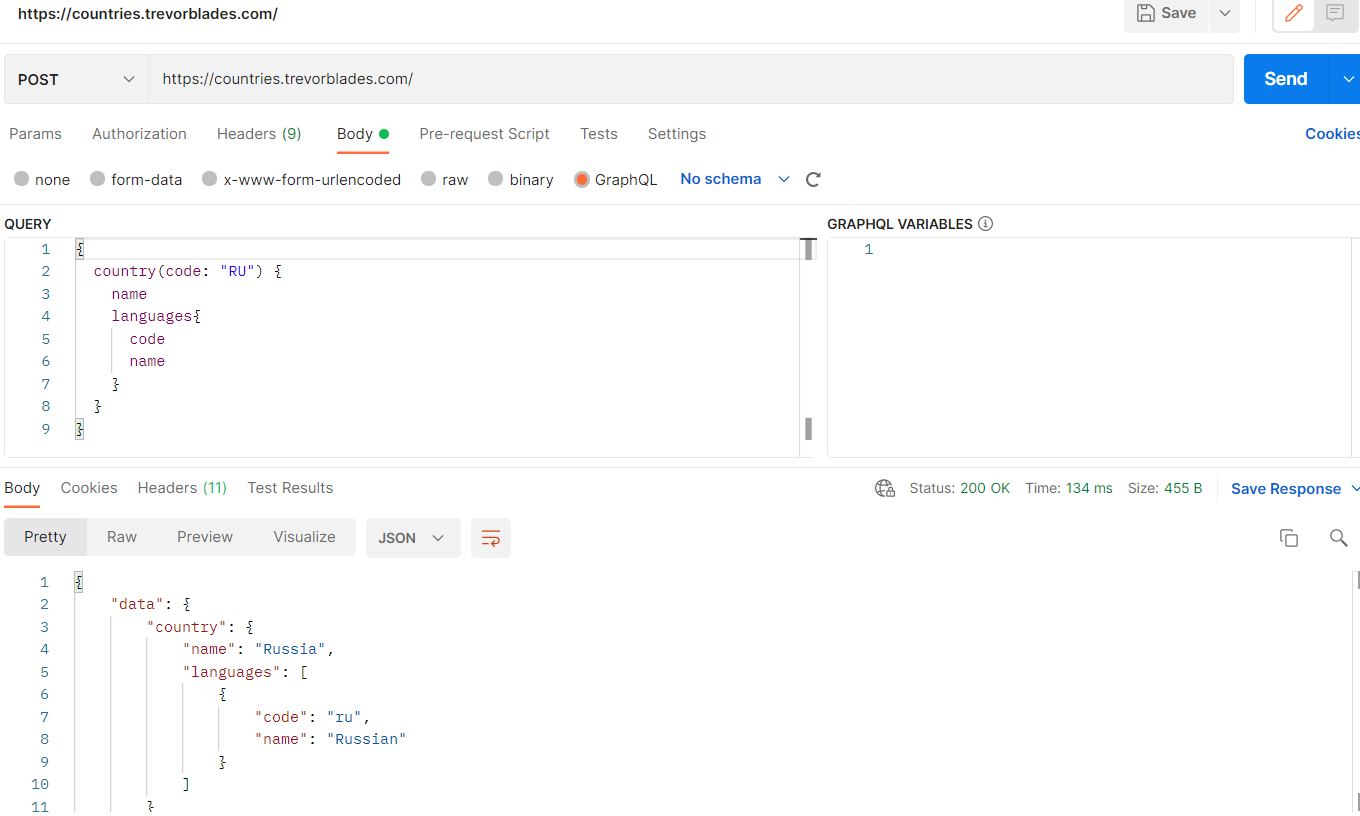
В GraphQL есть всего 1 Endpoint, и клиент сам решает какие значение ему вернуть, используя схему (Schema)



**GraphQL structure (Schema)**



Similar to calling a SOAP API, we will need to specify the service we want and do a POST request rather than a GET request



**GraphQL pros**

**Typed schema.**GraphQL publishes in advance what it can do, which improves its discoverability. By pointing a client at the GraphQL API, we can find out what queries are available.

**Fits graph-like data very well.**Data that goes far into linked relations but not good for flat data.

**No versioning.**The best practice with versioning is not to version the API at all.

While REST offers multiple API versions, GraphQL uses a single, evolving version that gives continuous access to new features and contributes to cleaner, more maintainable server code.

**Detailed error messages.**In a similar fashion to SOAP, GraphQL provides details to errors that occurred. Its error message includes all the resolvers and refers to the exact query part at fault.

**Flexible permissions.**GraphQL allows for selectively exposing certain functions while preserving private information. Meanwhile, REST architecture doesn’t reveal data in portions. It’s either all or nothing.

**GraphQL cons**

**Performance issues.**GraphQL trades off complexity for its power. Having too many nested fields in one request can lead to system overload. So, REST remains a better option for complex queries.

**Caching complexity.** As GraphQL isn’t reusing HTTP caching semantics, it requires a custom caching effort.

**A lot of pre-development education.**Not having enough time to figure out GraphQL niche operations and SDL, many projects decide to follow the well-known path of REST.

**GraphQL use cases**

**Mobile API.**In this case, network performance and single message payload optimization is important. So, GraphQL offers a more efficient data loading for mobile devices.

**Complex systems and microservices.**GraphQL is able to hide the complexity of multiple systems integration behind its API. Aggregating data from multiple places, it merges them into one global schema. This is particularly relevant for [legacy infrastructures](https://www.altexsoft.com/whitepapers/legacy-system-modernization-how-to-transform-the-enterprise-for-digital-future/) or third-party APIs that have expanded over time.