GraphQL, PWA

and the world of Software Architectures – from a front-end point of view

Jamie Maria Schouren

IN THIS ARTICLE YOU'LL FIND OUT:

- What benefits GraphQL brings to the table
- Differences between REST and GraphQL
- What Service Oriented Architecture is

I have seen the future, and it looks a lot like GraphQL. Mark my words: in 5 years, newly minted full-stack app developers won't be debating RESTfulness anymore, because REST API design will be obsolete.

Alan Johnson
Is GraphQL The Future?, May 2018

If you ask a random developer what she or he knows about web development, you will probably hear the word REST in the answer. In a time where API driven architectures are quickly becoming the standard for fast growing e-commerce companies, RESTful web services are taking a dominant position in this climate.

While REST is still the standard for most companies; in early 2018, something new appeared on the horizon. Something that would forever change the way we develop APIs. A framework developed by Facebook, a query language for APIs and a runtime for fulfilling those queries with your existing data: GraphQL.

GraphQL was developed for internal use by Facebook in 2012 and was publicly released in 2015. In early 2018, however, many big companies started to introduce GraphQL in their stack – and boy, did it deliver.

REST VS GRAPHQL

Any developer looking to connect a system or service with APIs, in most cases blindly chooses RESTful to connect web services. This is a very logical choice, as the basic principles of REST allow both front-end and back-end developers to build web applications in a fast and intuitive way.

RESTful webservices, however, have one major disadvantage: the API is deciding exactly which data, in which combination and in which format, the front-end application can retrieve. The front-end developer in this case is heavily depending on the structure of the API, a responsibility of the back-end developer.

Running just one front-end application won't get you into much trouble, as long as the front-end developer and the back-end developer work closely together and document how the API needs to be used. But as soon as your API needs to serve data to more front-end applications or even other back-end services, you will run into problems very quickly.

Every front-end application or service has its own need for certain data – which might even change frequently – something the API developer needs to take into consideration when developing and maintaining the API. In practice, this means that each application calling the API will need to make multiple calls to get the data it needs. Within the process, a lot of data will be retrieved that a specific application does not even need, causing performance issues very easily.

To prevent performance issues by retrieving data we don't need, a back-end developer can build API end-points which exactly match the data requirements of each (front-end) application. However, this will cause very tight coupling between the API and the front-end applications: making it more difficult to build, manage or maintain, and at the same time turning it into an unnecessarily complex system.

As you can imagine, Facebook ran into these issues in 2011 when they decided to refactor their whole application landscape into an API driven architecture. With their websites, a bunch of apps serving nearly all mobile platforms, and God knows what other services they are sending their (your?) data to, they reached the limitations of RESTful web services within no time.

Facebook then decided to build a new framework. A framework that would make the exchange of data between APIs and front-end applications easier, more efficient and more flexible. One with all the advantages of RESTful web services, but without its weaknesses: and that is how GraphQL was born.

BUT WAIT, WHAT IS GRAPHQL?

In short: GraphQL is a query language and a runtime environment (an external service such as Apollo) which allows you, similar to REST, to connect an application with any another application out there.

GraphQL gets you exactly the data you need, from just one single API, in real time. No more overload of data you don't need, no more back-end developers getting frustrated to build complex services for each application, no more front-end developers waiting for the back-end developer to deliver before they can continue, no more depreciation of performance, and if that wasn't enough: GraphQL will automatically write documentation for you.

With GraphQL, a user, for example a front-end application, can send request to a back-end application. The request is a query specifically explaining what it needs from the API. The runtime of GraphQL, which is running in the back-end application, then translates these queries into functions that can retrieve and even modify data.

GraphQL is really a game changer. Front-end developers can now build their front-ends, add their queries and will know exactly how the data will return – even if the API is not ready yet. They can truly work simultaneously with, and independent from, the back-end developer as they don't need to care about where the data is coming from, and they can develop with just mock-up data. As long as the right data will come in the end.

One major difference between RESTful webservices and GraphQL is that RESTful webservices provide more than one endpoint to retrieve or modify data, while GraphQL only needs one. This allows you to, with just one call, get the exact data you need: nothing more and nothing less. Not only will it get you the data you need, but it will exactly give you the data in the same structure as you put in the request. This way you, as a front-end developer, will know exactly what you get and how you get it – making the use of GraphQL a whole lot easier, more efficient and way more flexible than when using RESTful webservices.

ZINE MAGE

Want more?

Get your free copy of Magazine & enjoy reading!

Download Magezine