Modern Web APIs with GraphQL

Hélder Vasconcelos
Luis Gonçalves

Outline

- Who are we
- What is a Web API
- Web API Technologies
- What is GraphQL
- GraphQL advantages
- GraphQL vs REST
- Languages that implement GraphQL
- Demo - build GraphQL Api
- Resources
- Q & A

What is a Web API?

An **Application Programming Interface** – is software programmed **contract** written to function as a communication bridge between the web applications running on the browser and the backend or cloud services.

Web APIS Technologies

- **XML Based RPC APIs** (SOAP)**-** a XML based protocol Introduced in Late 90's.

- **AJAX** - Introduced in 2005 soon became a popular approach for making web sites dynamic, giving birth to the "Web 2.0" concept.

- **REST** (Representational State Transfer) - Introduced on 2000 , uses a passing of resource representations, as opposed to messages or function calls.

- **GraphQL** - based on schema types. It was developed internally by Facebook and released as open source on 2015.

What is GraphQL?

- GraphQL is just a web **protocol** that specifies the way we build and query remote APIs using a Tree/JSON like syntax.

```
reactorHubNews {
    title
    description
}
```

- Based on a strongly **typed** language - GraphQL Schema Definition Language (SDL).

```
type Post {
  title: String!
  description: String!
}
```

What is GraphQL?

- A protocol that allows the client to specify exactly **what data it needs** from a model.

```
posts {
   title
}
```

- It allows to aggregate data from **multiple relations** in a single query.

```
posts {
   title
   description
   user {
      name
   }
}
```

GraphQL Type System - Built in scalar types

- **Int** - An Integer type, example 10
- **Float** - Floating point number , example 3.43
- **String** - A sequence of characters , example “Hello World”
- **Boolean** -
- **ID** - Object identifier

GraphQL Type System - User Defined Types

- GraphQL uses types to ensure the clients know the fields supported by a resource. The types are defined by the user following the GraphQL SDL specification

```
type Post {
  title: String!
  description: String!
  author: User!
}

type User {
 fullName: String!
 email: String!
}
```

GraphQL Query

- Used to fetch data from the server using the GraphQL SDL syntax.
- Describe what data the requester wishes to fetch from whoever is fulfilling the GraphQL query.

```
query reactorNews {
   posts(sortedBy: createdDesc) {
      title
      description
       author {
        fullName
      }
    }
}
```

GraphQL Mutation

- Used to change resources data or execute actions on the server
- Client specifies the arguments and the action to be executed and at the end receives a response or a resource updated

```
mutation createPost {
   createPost({
     title: “Reactor is celebrating a party”
      Description: “Beer, sparkling water and much more….:)” })
  {
     id
     title
  }
}
```

GraphQL Subscription

- Used to get realtime resource or data updates
- Users get a notification every time the resource subscribed gets updated or changed.

```
subscription newSubscriber {
   newsLetterSubscriberCreated{
      id
      subscriber {
         fullName
         email
      }
      createAt
   }
}
```

GraphQL Advantages

- You get the data you request and need for a particular scope
- Excellent developer tooling and experiences since the specification defines API introspection
- Only one endpoint to connect,
    - `Example POST api.reactorhub.com /graphql/api`

GraphQL vs REST

- GraphQL has only one url/endpoint
- In REST, the layout and size of data returned is determined by the server . The server has a predetermined amount of properties and relations that sends on a call.
- In REST, to retrieve relational data you need to make multiples API calls making the data data rendering more

GraphQL implementations

Server:

- C# / .NET
- Clojure
- Elixir
- Erlang
- Go
- Groovy
- Java
- JavaScript
- PHP
- Python
- Scala
- Ruby

Client:

- C# / .NET
- Clojurescript
- Go
- Java / Android
- JavaScript
- Swift / Objective-C iOS
- Python

Demo

- Let’s create a GraphQL API in node.js
- Create GraphQL server with ApolloServer
- Use gql to define schema (typedefs)
    - types
    - Queries
    - Mutations
- Define resolvers.
- Show playground to explore and test API

Resources

- Sites
    - GraphQL (https://graphql.org/)
    - GraphQL Weekly newsletter (https://www.graphqlweekly.com/)
    - GraphQL Cheat Sheet (https://github.com/sogko/graphql-schema-language-cheat-sheet)
    - Apollo (https://www.apollographql.com/)
    - Prisma (https://www.prisma.io/)
    - Hasura (https://hasura.io/)
    - GraphQL Yoga (https://github.com/prisma/graphql-yoga)
- Podcasts
    - https://graphqlpatterns.simplecast.fm/
    - https://graphqlradio.com/ (last episode > 1 year ago)
- Books
    - The Road to GraphQL (free ebook)

TAIKAI

Thanks! Any Questions ?

www.taikai.network

✉ luis@taikai.network   @luisfigoncalves

✉ helder@taikai.network   @heldervasc