Markdown Generation from a GraphQL Schema

Alessandro Buonerba
University of Greenwich

Abstract

Since GraphQL was publicly released in 2015, many developers adopted it to create new public faced APIs, but these are often poorly documented. Writing well-structured documentation requires time and manual work, and the tooling currently available at the time of this paper would require technologies and frameworks lock-in. This project aims to generate markdown files that can then be parsed in any framework of choice. Since the output will be in Markdown, the user can then use his parser to manipulate the documents and produce static files for their frontend.

Introduction

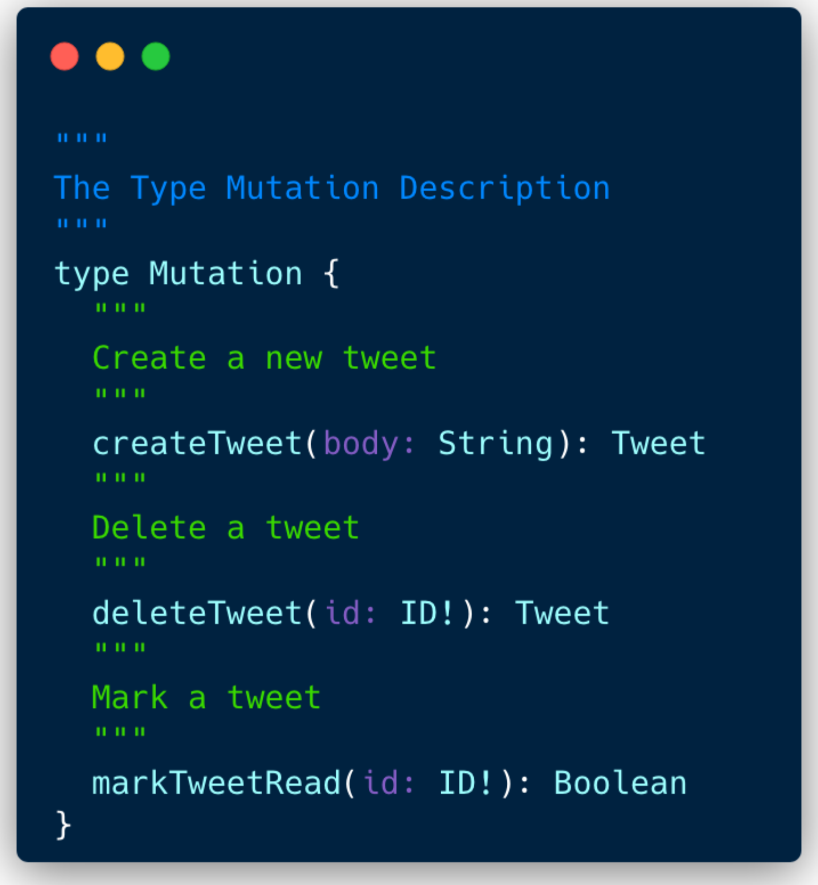
Documentation is a crucial aspect of the development of software. It is so important that every developer building a GraphQL API should be aware of it an keeping their Schemas documented to enable and facilitate consumers to understand how to use the API. A very a good way to document such APIs is not to be locked in with any vendor-specific that generates documentation on the fly given a specific schema but building static documentation that automatically updates on the fly when a change is made to it. The following literature review will point the differences between different APIs, analyse and critically evaluate the options developers have to reach the same goal and why the software implementation of the tool used in the project gives the producers better results, freedom and save them money. Security implications are also another important aspect of the lifecycle and this paper will dive into analysis and confrontation to understand this.

Keep your GraphQL API **secure,** while still have a strong **documentation,** **without** Introspection.

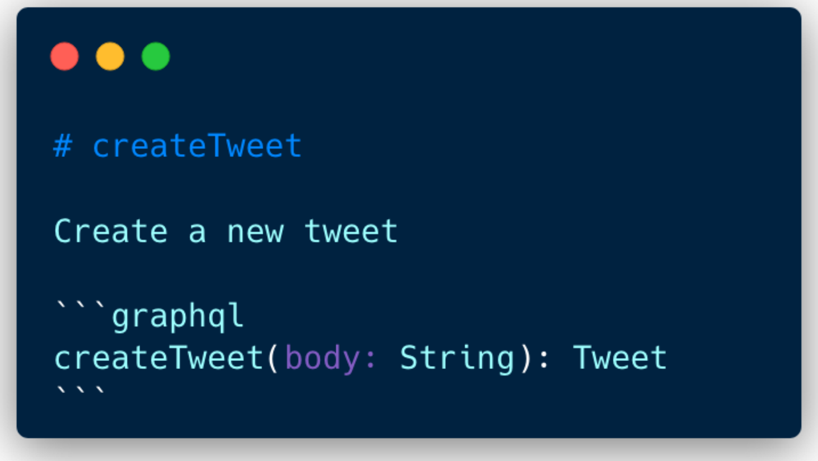
Take a picture to download the full paper

Methods

- Construct a GraphQL Schema
- Generate an AST object
- Manipulate the object
- Generate a Markdown file
- Recursively for each field and type



```
"""
The Type Mutation Description
"""
type Mutation {
  """
  Create a new tweet
  """
  createTweet(body: String): Tweet
  """
  Delete a tweet
  """
  deleteTweet(id: ID!): Tweet
  """
  Mark a tweet
  """
  markTweetRead(id: ID!): Boolean
}
```

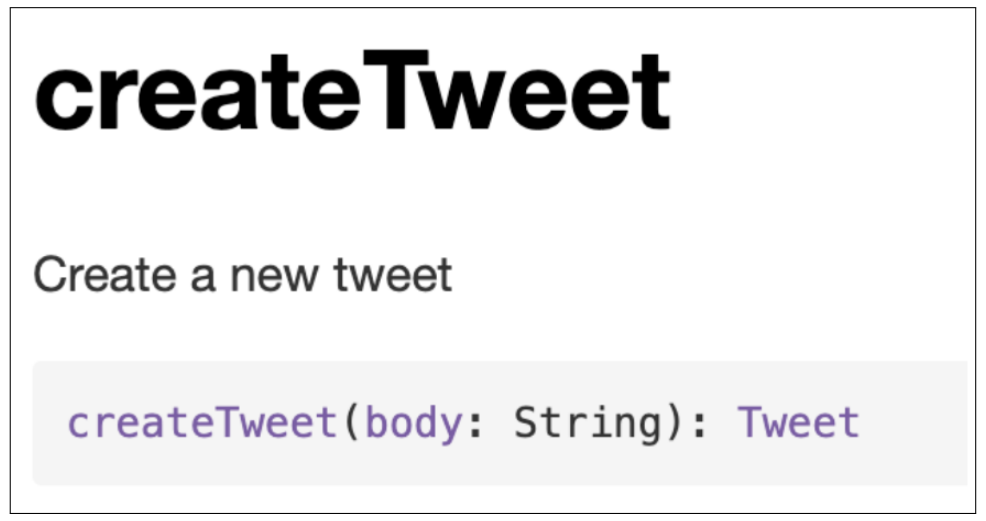


~~~
# createTweet

Create a new tweet

```graphql
createTweet(body: String): Tweet
```
~~~

Output



createTweet

Create a new tweet

```
createTweet(body: String): Tweet
```