Chapter 2: TypeGraphQL

Strongly Typed Next.js

TypeGraphQL

By the end of this lesson, developers will be able to:

- Create a GraphQL API schema with TypeGraphQL
- Create a UserResolver for fetching user data

Introduction

Authentication is one of the most challenging tasks for developers just starting with GraphQL. There are a lot of technical considerations, including what ORM would be easy to set up, how to generate secure tokens and hash passwords, and even what HTTP library to use and how to use it.

In this section, we will start building a GraphQL API server. With MongoDB as our database, we learn how to incorporate TypeGraphQL and Typegoose. Let's begin with an overview of these libraries and their dependencies:

- TypeGraphQL: We will use TypeGraphQL to declare our models and their properties.
- Typegoose: Once the models are declared, we will use Typegoose to manage how MongoDB works in the database.
- jsonwebtoken: Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. `jsonwebtoken` will be used to generate a JWT which will be used to authenticate users.

TypeGraphQL

Starting to incorporate Typescript and GraphQL, we can utilize existing libraries for creating strongly typed schemas. For a brief introduction to TypeGraphQL, be sure to visit the TypeGraphQL documentation.

TypeGraphQL solves many problems for us, like schema validation, authorization and dependency injection, which helps develop GraphQL APIs quickly and easily. TypeGraphQL also integrates with several third party libraries like Typegoose.