

Express y Middlewares: Repaso

ExpressJS es la librería por excelencia para crear APIs. Es minimalista, y muy extensible. Se extiende por medio de los middlewares.

Los middlewares son funciones de ExpressJS que tienen acceso a los objetos Request y Response.

Bases de datos: Lo básico

Las bases de datos son un método por excelencia para persistir datos a lo largo de nuestra aplicación, permitiendo operar sobre ellos en diferentes momentos del tiempo. Existen dos tipos de bases de datos: Relacionales y no relacionales.

Relacional: Es una base de datos estructurada, los datos son organizados dentro de tablas. La mayoría de las veces, hay datos dentro de ellas que están directamente vinculados a otras tablas, por lo que deben unirse dichas tablas para poder devolver el resultado completo.

No relacional: Las bases de datos no relacionales, en cambio, no mantienen relación entre documentos. En realidad, **no deberían** tener relación. Sino que en un único documento, debería estar toda la información correspondiente a él.

MongoDB

MongoDB es uno de los motores más conocidos y más utilizados de base de datos no relacional. Al igual que en el resto de DB no relacionales, la información se guarda en forma de documentos estilo JSON.

Vamos a utilizar la librería por excelencia para operar con MongoDB, también ampliamente conocida, mongoose.

```
// newly added document
{
  "name" : "Product",
  "description" : "This is a product",
  "price" : 1.45,
  "_id" : ObjectId("63754802232eff6d49d99ccf")
}
```

Mongoose

La librería consiste en crear Modelos de datos. Como MongoDB no tiene estructura, siendo una base de datos no relacional, mongoose nos agrega una funcionalidad extra que nos permite mantener cierto orden en nuestros datos. Una "estructura" básica, si se quiere. Esto es opcional, no es necesario realizarlo, pero haciéndolo nos va a facilitar mucho nuestras tareas diarias.

Una vez teniendo el modelo, podemos realizar operaciones sobre él.

```
1 const mongoose = require('mongoose');
2
3 const ProductSchema = new mongoose.Schema({
4   name: {
5     type: String,
6     required: true
7   },
8   description: String,
9   price: Number
10 });
11
12 module.exports = mongoose.model('Product', ProductSchema);
```

API: Repaso

Qué es una API? Son siglas de Application Programming Interface. Es un método de comunicación entre dos o más aplicaciones

- **Rest**

- Existen endpoints que representan una determinada entidad
- No se usan endpoints con verbos ni acciones, sino que la acción a realizar se determina según el tipo de método HTTP con el que se solicita la url
- Generalmente se utiliza GET para obtener un listado o un recurso determinado, POST para crearlos, PUT o PATCH para actualizar y DELETE para borrarlo
- La api es stateless. Esto significa, que no se guardan estados entre las diferentes requests, y cada una se trata por separado
- It's all about JSON.
- Se utilizan códigos de respuesta HTTP para informar estados, como por ejemplo
404 si no se encuentra un elemento, 201 si un elemento fue creado exitosamente, etc

- **GraphQL**

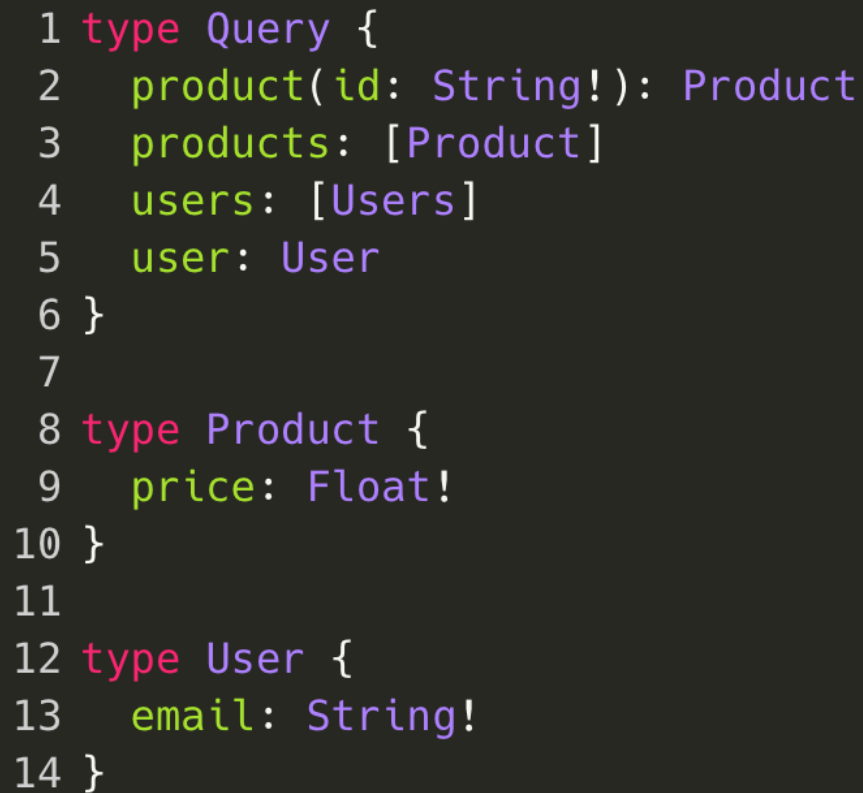
Es una forma de consultar las APIs. Es un lenguaje estilo "Query". Donde cada request decide qué datos recibir, en vez del servidor. Se pueden consultar múltiples entidades en la misma request a diferencia de REST donde cada entidad tiene su propio endpoint, y se deben hacer llamadas independientes.

A diferencia de una API REST, GraphQL si tiene ciertas normas que deben ser respetadas para que las requests funcionen. Entre ellas, todas las llamadas son mediante POST. La API debe devolver JSONs, y en el cuerpo de cada request se deben especificar los campos a devolver.

Se implementa utilizando resolvers. Se le llama resolver a la función correspondiente a cada "endpoint". Hay que tener en cuenta que en GraphQL, se le dice vulgarmente endpoint a cada entidad. El endpoint tal como lo veníamos conociendo, es decir la url, siempre es la misma.

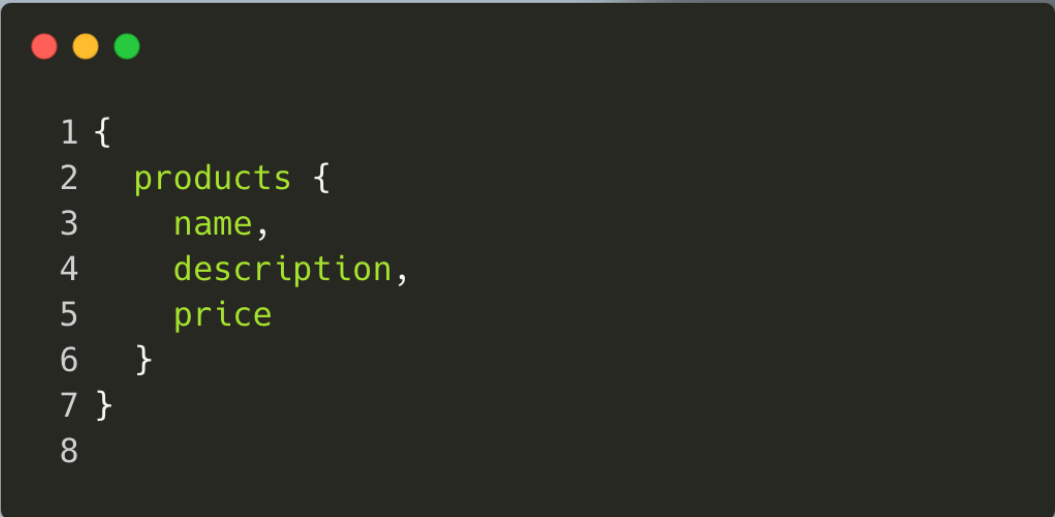
A su vez, es auto-documentada, ya que toda la información que se puede consultar junto con los parámetros necesarios para eso, se puede consultar dentro del esquema de graphql. Mientras que en una API REST, hay que hacerlo manualmente, o con herramientas como Swagger.

Ejemplo de Schema GraphQL



```
1 type Query {  
2   product(id: String!): Product  
3   products: [Product]  
4   users: [Users]  
5   user: User  
6 }  
7  
8 type Product {  
9   price: Float!  
10 }  
11  
12 type User {  
13   email: String!  
14 }
```

Ejemplo Request



```
1 {  
2   products {  
3     name,  
4     description,  
5     price  
6   }  
7 }  
8
```

Actividad

- Implementar MongoDB y Mongoose a la API rest de la actividad anterior