

¿Qué es GraphQL? Creando un servidor desde cero con queries

https://www.youtube.com/watch?v=QG-qbmW-wes&t=135s&ab_channel=midudev

Primer Servidor GraphQL

```
npm init -y
```

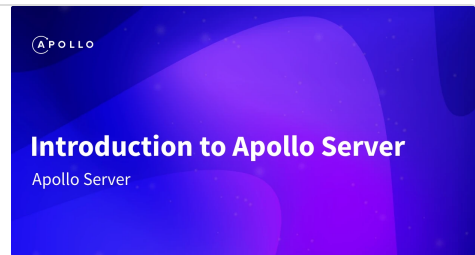
```
star41@DESKTOP-486LDHS:~/personalProjects/graphql/graphql-server$ npm init -y  
Wrote to /home/star41/personalProjects/graphql/graphql-server/package.json:  
  
{  
  "name": "graphql-server",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "description": ""  
}
```

Instalación de dependencias: Iniciar un Proyecto

Introduction to Apollo Server

Apollo Server is an open-source, spec-compliant GraphQL server that's compatible with any GraphQL client, including Apollo Client. It's the best way to build a production-ready, self-documenting

 <https://www.apollographql.com/docs/apollo-server/>



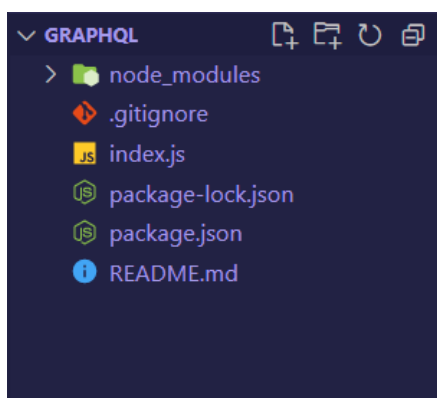
Apollo Server es un servidor GraphQL de código abierto que cumple con las especificaciones y es compatible con cualquier cliente GraphQL, incluido Apollo Client . Es la mejor manera de crear una API de GraphQL autodocumentada y lista para la producción que puede usar datos de cualquier fuente.

```
npm i apollo-server graphql
```

Modificar el package.json agregar la siguiente Linea:

```
"type":"module" en el package para que se utilice el emma scrip module
```

Crear una Archivo index.js. Donde pondremos los datos a los que queremos hacer las consultas:



```

1
2 //Es indiferente de donde salgan los datos:
3 const persons = [
4   {
5     name: "Juan", //name: 'Juan',
6     age: 20, //age: 20,
7     country: "MX", //country: 'MX'
8     id: "defrgrovkr",
9   },
10  {
11    name: "Pedro", //name: 'Pedro',
12    age: 30, //age: 30,
13    country: "MX", //country: 'MX'
14    id: "defrgrovks",
15  },
16  {
17    name: "Maria", //name: 'Maria',
18    age: 25, //age: 25,
19    country: "MX", //country: 'MX'
20    id: "defrgrovkp",
21  },
22 ];

```

Una vez que ya tenemos los datos, vamos a describir los datos para ello necesitaremos GraphQL. Por lo cual es necesario importarlo:

```
import { gql } from "apollo-server";
```

Definiciones de Datos

```

//Debemos de describir los datos con graphql:
//gql lo que hace es ejecutar un string
// Si queremos hacer peticiones a graphql, debemos de describir las peticiones
//El signo de exclamacion es para declarar que algo es obligatorio

const typeDefinitions = gql`
  type Person {
    name: String!

```

```

    age: Int
    country: String!
    info: Info!
    id: ID!
  }

```

Cuando queremos hacer peticiones tenemos un typeQuery

```

type Query {
  personCount: Int!
  allPersons: [Person]!
  findPerson(name: String!): Person
}
;

```

En GraphQL tenemos : Las definiciones de los datos, y de donde sacamos estos datos

En el Código anterior cuando a la Query le pedimos el `personCount` de donde saca el numero `Int` ? Para ello vamos a crear una Constante que se llamara resolvers:

Resolvers

```

const resolvers = {
  Query: {
    personCount: () => persons.length,
    allPersons: () => persons
  },
}

```

Creación Del Servidor

En la definición de datos, la constante siempre debe de llamarse `typeDefs` , en caso que le hayamos puesto otro nombre debemos pasarlo en la definición del servidor. De igual forma para los `resolvers`

```

//Creacion del Servidor
const server = new ApolloServer({
  typeDefs: typeDefinitions,
  resolvers,
});

```

Iniciar el Servidor

```
//iniciando el servidor
//listen inicializa y luego devuelve una promesa y recuperamos la url del servidor

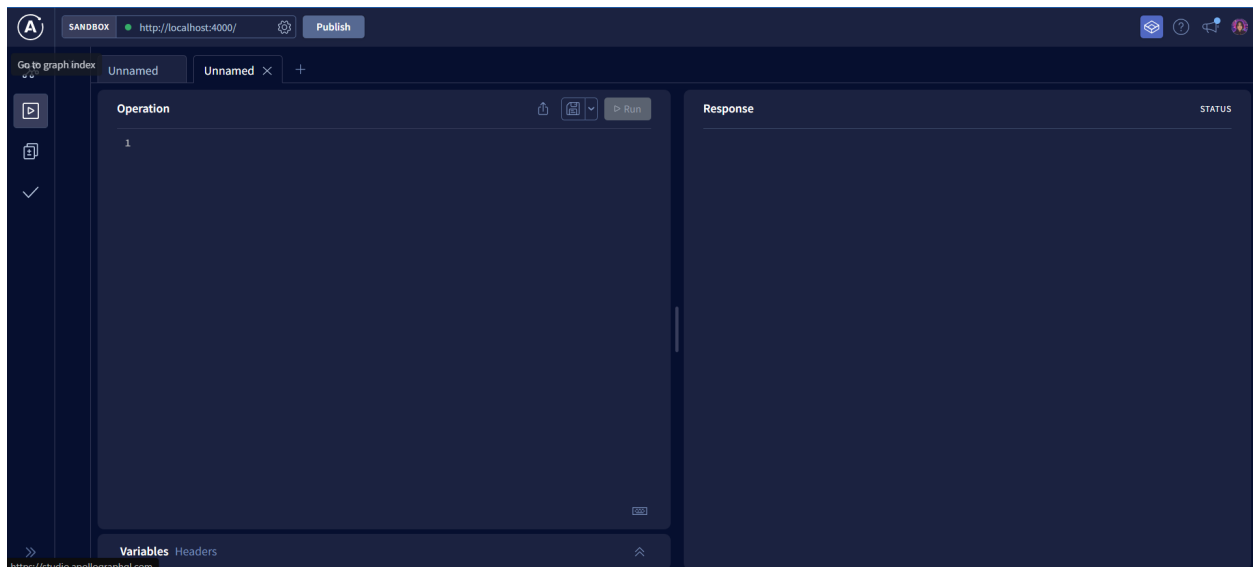
server.listen().then(({ url }) => {
  console.log(`Server ready at ${url}`);
});
```

Correr el Servidor

```
node index.js
```

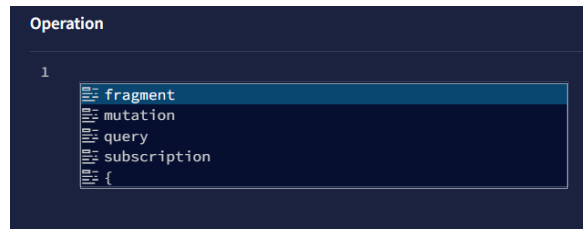
```
PS C:\Users\Usuario\OneDrive\Desktop\GraphQL> node index.js
Server ready at http://localhost:4000/
```

Colocar la ruta <http://localhost:4000/> en nuestro navegador



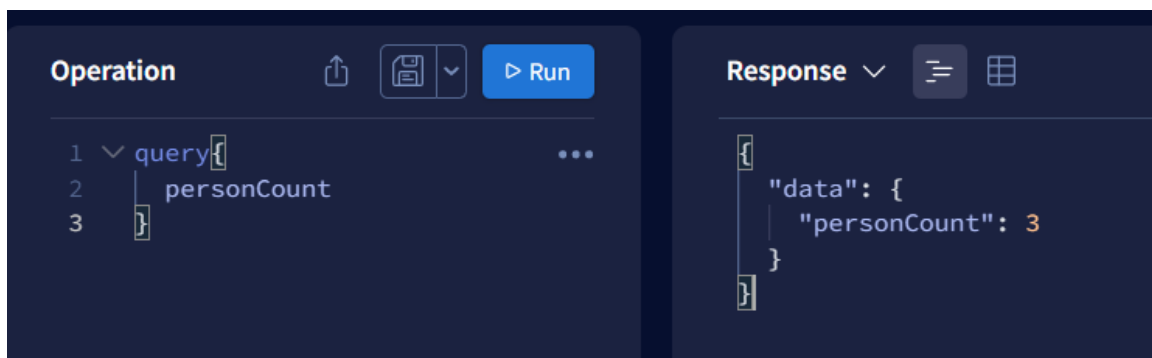
Comandos útiles en nuestro GraphQL playGround

Comando	Función
ctrl+space	saldrá un Autocomplete



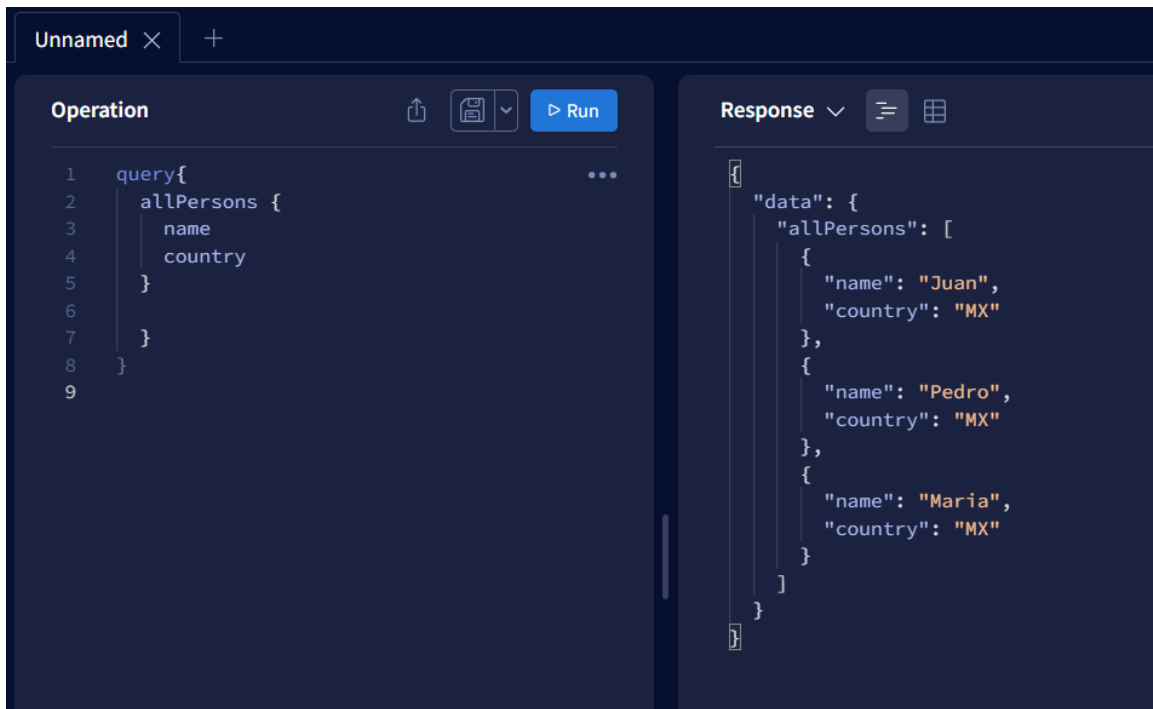
Haciendo una Query en GraphQL

El lenguaje de consultas de GraphQL esta basado en brackets {} asi que para hacer una debemos utilizarlos



Selección de Subcampos:

Cuando queremos recuperar allPersons que es una array de personas tenemos que decirle que campos queremos extraer.



Practicar con el **Ejemplo: API SPACEX**

<https://api.spacex.land/graphql/>

Agregando un Find Person a nuestra Query

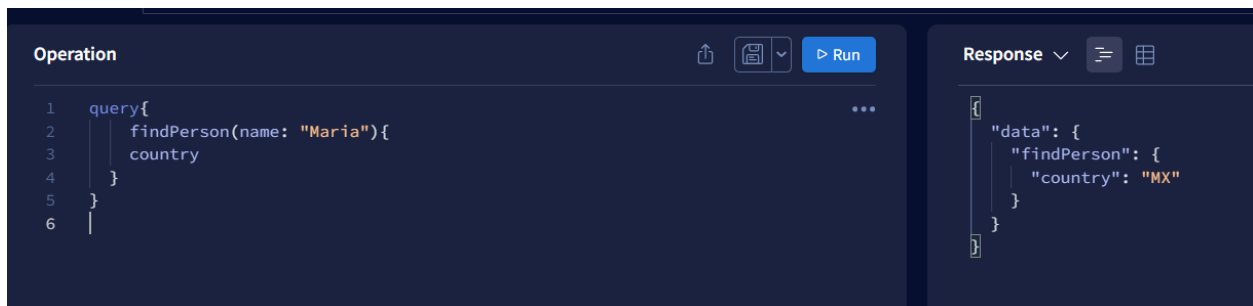
```
type Query {
  personCount: Int!
  allPersons: [Person]!
  findPerson(name: String!): Person
}
```

Resolviendo la Query

findPerson recibe root, args y retorna la persona con el nombre que le demos

```
const resolvers = {
  Query: {
    personCount: () => persons.length,
    allPersons: () => persons,
    findPerson: (root, args) => {
      const { name } = args;
      return persons.find((person) => person.name === name);
    },
  },
}
```

Para Recuperar le damos el nombre de la persona y el campo a recuperar de dicha persona. En esta caso country



The screenshot shows the GraphQL Playground interface. On the left, under the 'Operation' tab, the query is: `query{ findPerson(name: "Maria"){ country }}`. On the right, under the 'Response' tab, the JSON response is: `{ "data": { "findPerson": { "country": "MX" } } }`. The 'Run' button is visible above the query input.

Si no encuentra la persona devuelve null



The screenshot shows the GraphQL Playground interface. On the left, under the 'Operation' tab, the query is: `query{ findPerson(name: "maria"){ country }}`. On the right, under the 'Response' tab, the JSON response is: `{ "data": { "findPerson": null } }`. The 'Run' button is visible above the query input.

Apollo Server tiene un resolver por defecto que nos busca la propiedad que le mandamos y retorna el valor.

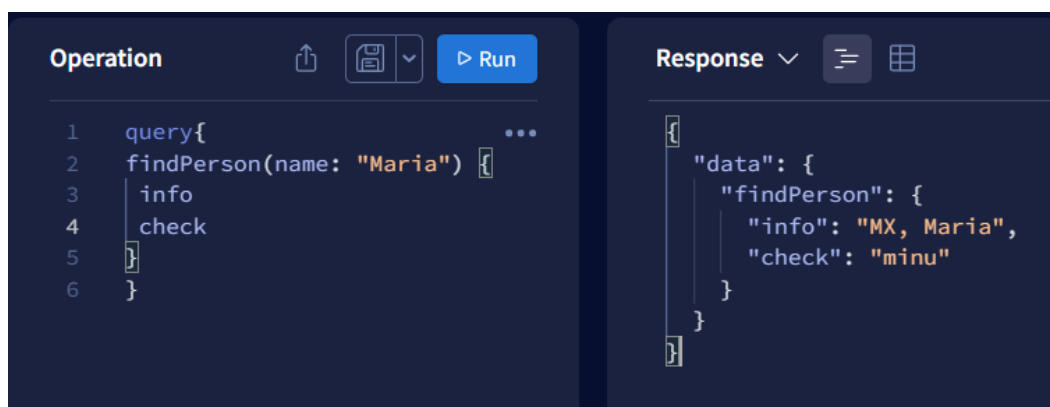
Resolvers Complejos

Quiero resolver un campo que retorne el name + el country. Para poder acceder a ellos tenemos que describirlos

```
Person: {  
  
  info: (root) => `${root.country}, ${root.name}`,  
  check: () => "minu",  
},
```

En la definición debemos de agregar los campos

```
type Person {  
  name: String!  
  age: Int  
  country: String!  
  info: String!  
  check: String!  
  id: ID!  
}
```



En GraphQL es mejor trabajar con objetos , así que el código anterior de la info de la persona lo vamos a transformar en un objeto

Definición:

```
type Info {  
  county: String!  
  name: String!  
}  
type Person {  
  name: String!  
  age: Int  
  country: String!  
  info: Info!  
  id: ID!  
}
```

Resolver

```
Person: {  
  info: (root) => {  
    return {  
      county: root.country,  
      name: root.name,  
    };  
  }  
},
```

Operation

Run

```
1 query{
2   findPerson(name: "Maria") {
3     info{
4       county
5       name
6     }
7   }
8 }
9 }
```

Response

```
{
  "data": {
    "findPerson": {
      "info": {
        "county": "MX",
        "name": "Maria"
      }
    }
  }
}
```