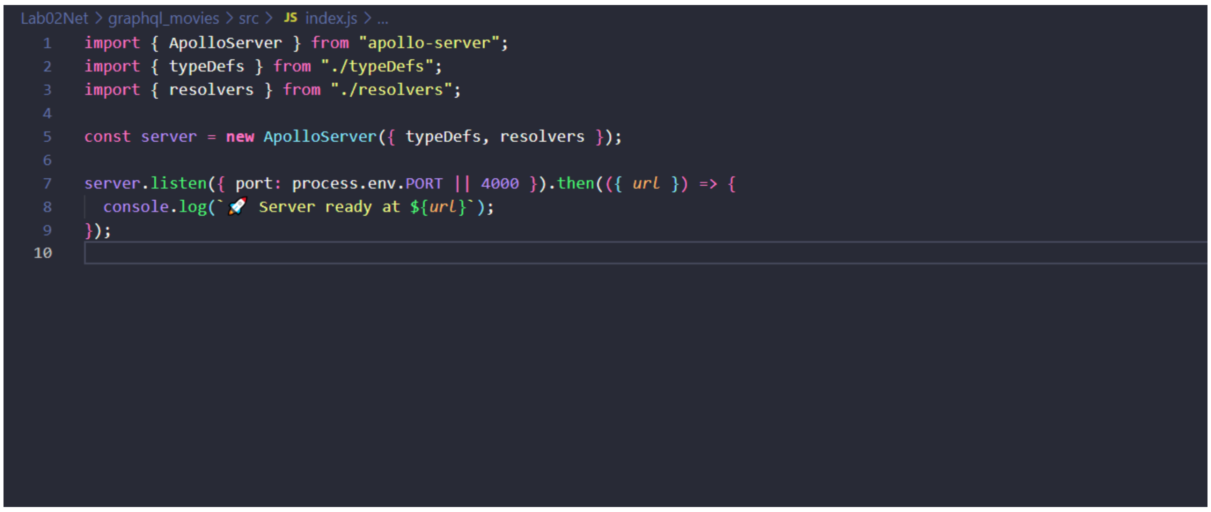
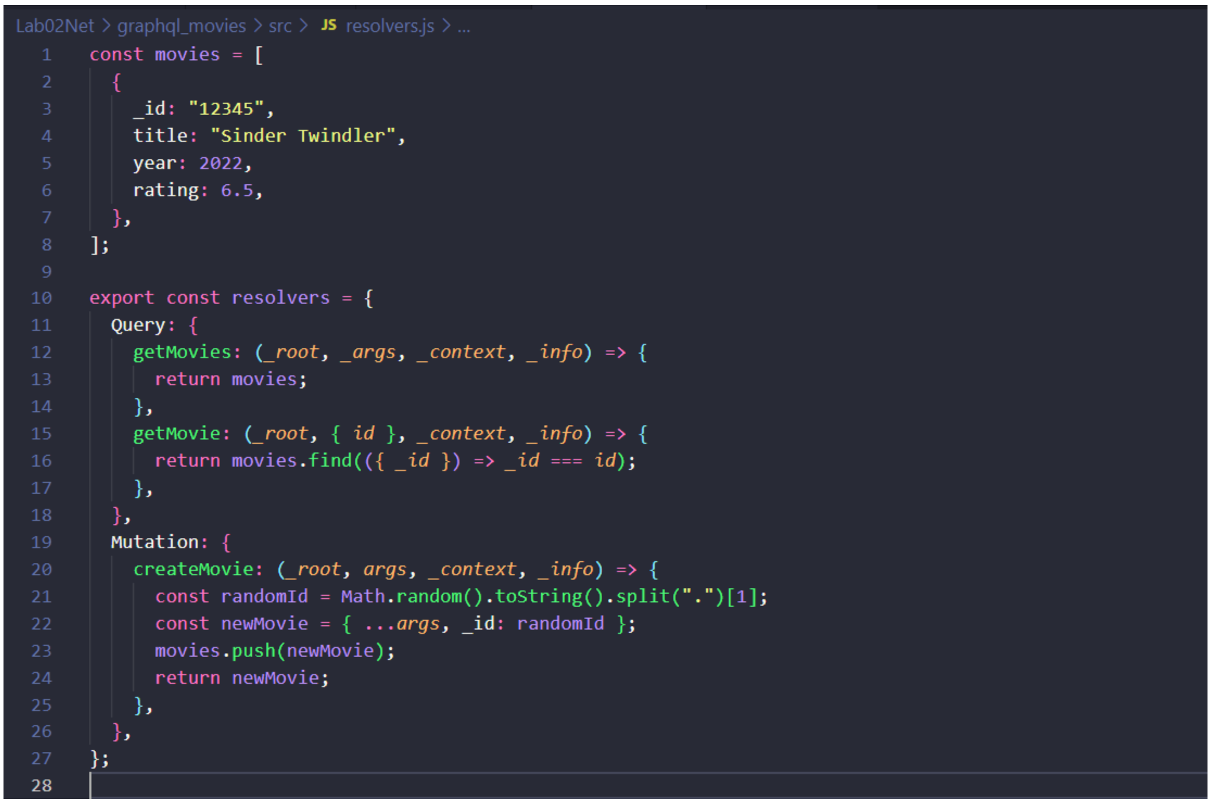
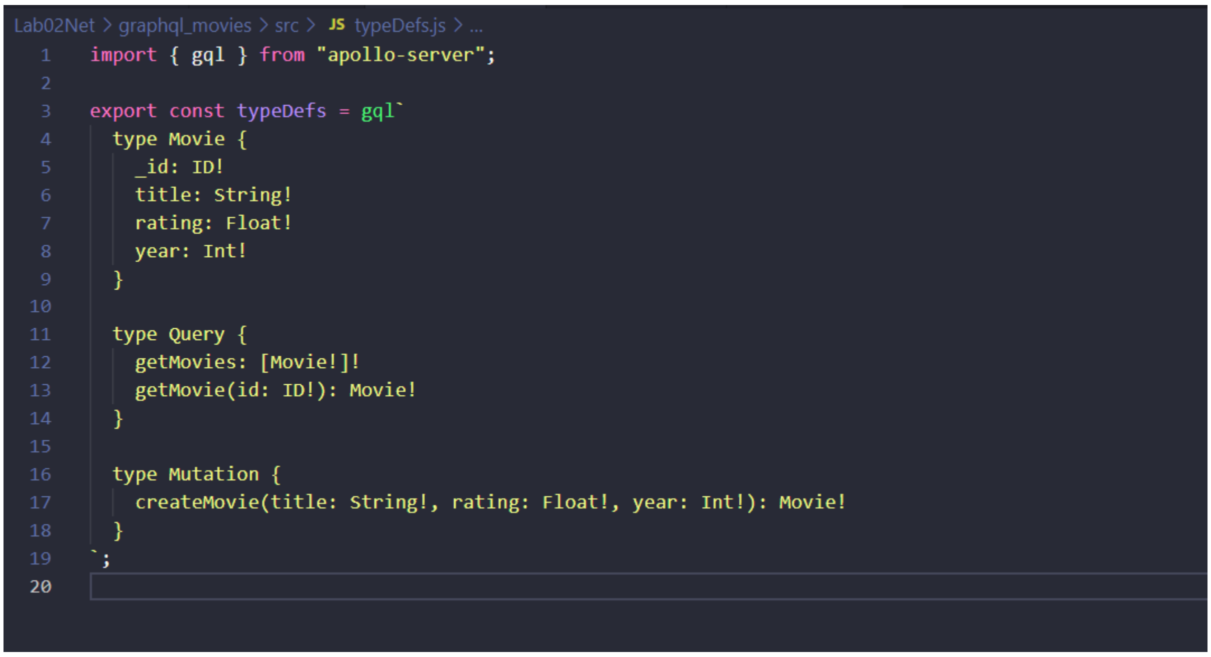

Lab02Net > graphql_movies > src > JS index.js > ...

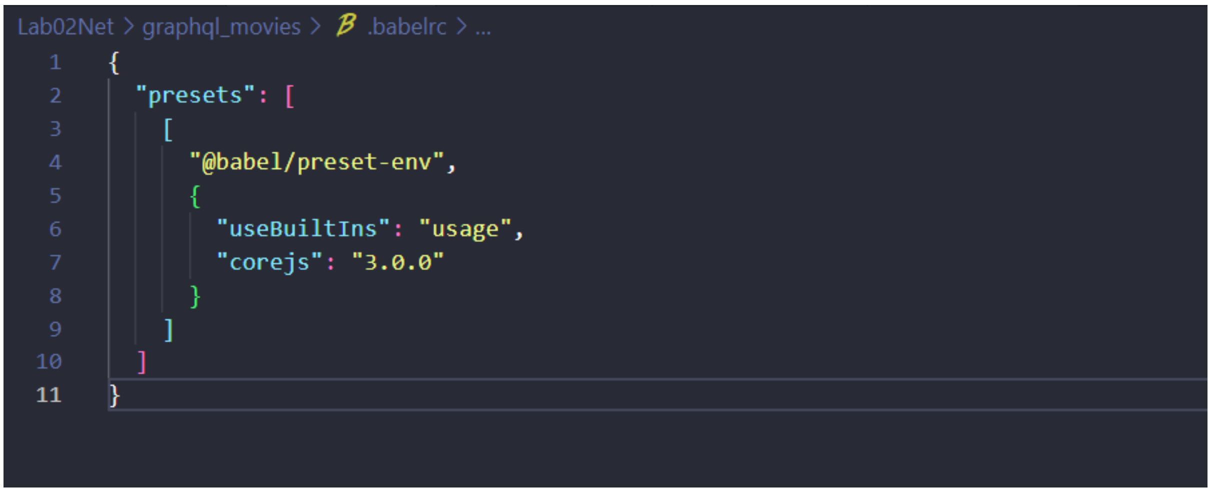
```javascript
import { ApolloServer } from "apollo-server";
import { typeDefs } from "./typeDefs";
import { resolvers } from "./resolvers";

const server = new ApolloServer({ typeDefs, resolvers });

server.listen({ port: process.env.PORT || 4000 }).then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});

```


Lab02Net > graphql_movies > src > JS resolvers.js > ...

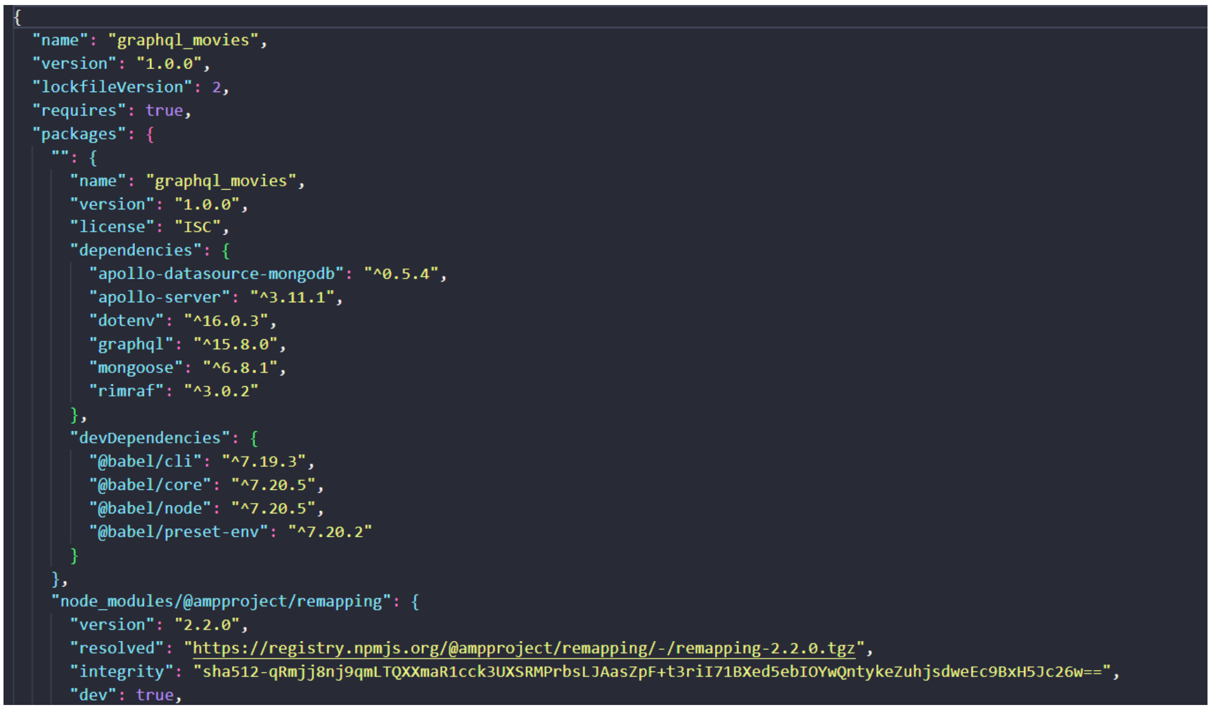
```javascript
const movies = [
  {
    _id: "12345",
    title: "Sinder Twindler",
    year: 2022,
    rating: 6.5,
  },
];

export const resolvers = {
  Query: {
    getMovies: (_root, _args, _context, _info) => {
      return movies;
    },
    getMovie: (_root, { id }, _context, _info) => {
      return movies.find(({ _id }) => _id === id);
    },
  },
  Mutation: {
    createMovie: (_root, args, _context, _info) => {
      const randomId = Math.random().toString().split(".")[1];
      const newMovie = { ...args, _id: randomId };
      movies.push(newMovie);
      return newMovie;
    },
  },
};

```

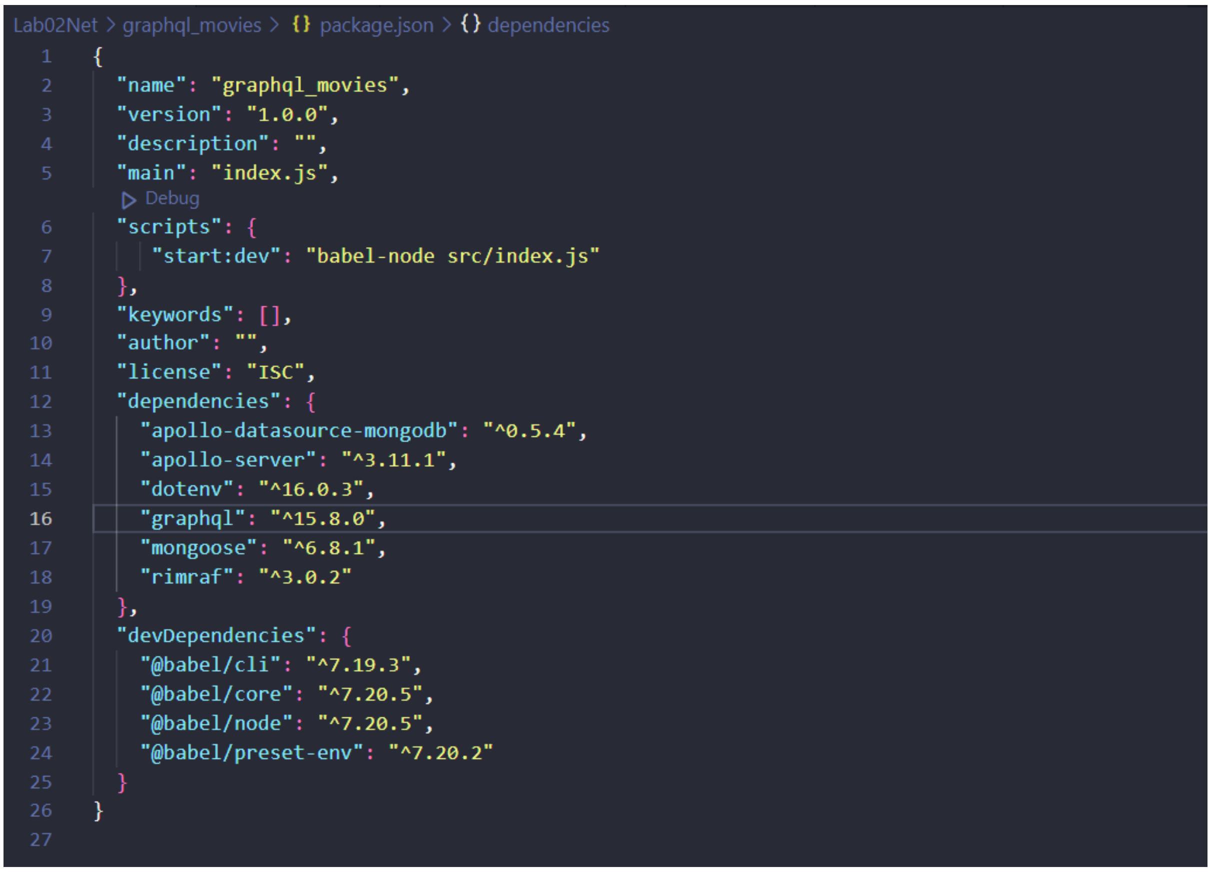

Lab02Net > graphql_movies > src > JS typeDefs.js > ...

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

export const typeDefs = gql`
  type Movie {
    _id: ID!
    title: String!
    rating: Float!
    year: Int!
  }

  type Query {
    getMovies: [Movie!]!
    getMovie(id: ID!): Movie!
  }

  type Mutation {
    createMovie(title: String!, rating: Float!, year: Int!): Movie!
  }
`;

```

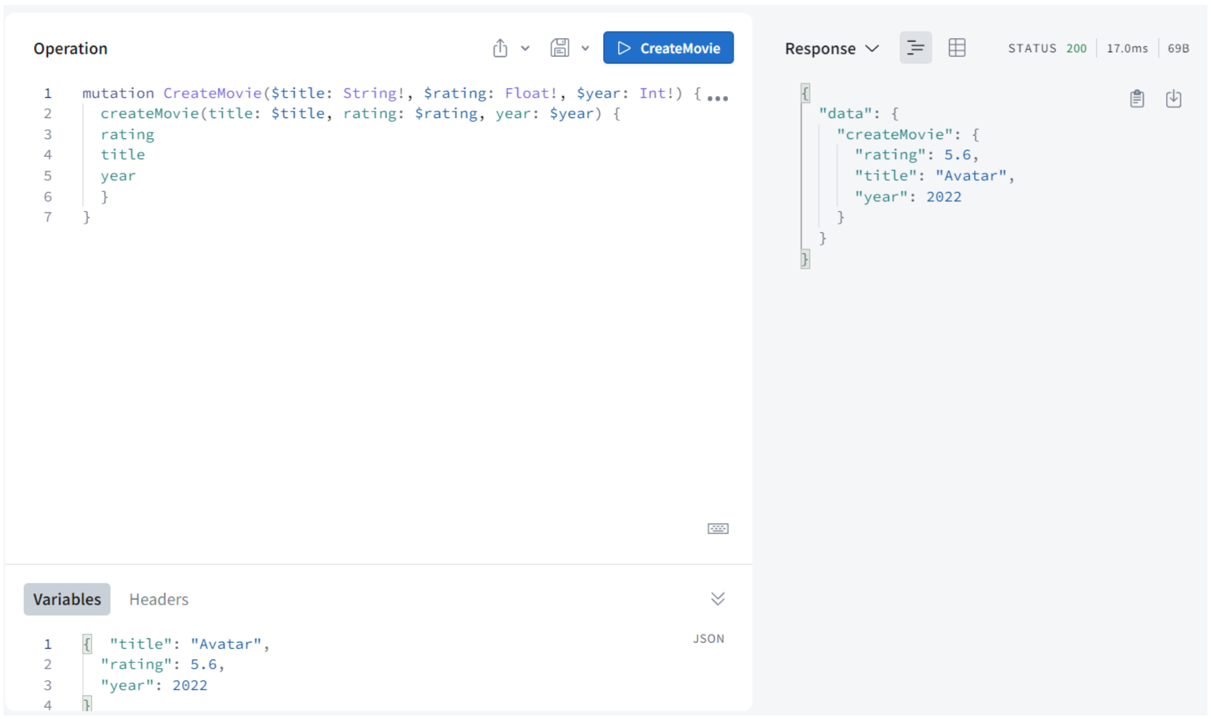

Lab02Net > graphql_movies > .babelrc > ...

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "useBuiltIns": "usage",
        "corejs": "3.0.0"
      }
    ]
  ]
}
```


```json
{
  "name": "graphql_movies",
  "version": "1.0.0",
  "lockfileVersion": 2,
  "requires": true,
  "packages": {
    "": {
      "name": "graphql_movies",
      "version": "1.0.0",
      "license": "ISC",
      "dependencies": {
        "apollo-datasource-mongodb": "^0.5.4",
        "apollo-server": "^3.11.1",
        "dotenv": "^16.0.3",
        "graphql": "^15.8.0",
        "mongoose": "^6.8.1",
        "rimraf": "^3.0.2"
      },
      "devDependencies": {
        "@babel/cli": "^7.19.3",
        "@babel/core": "^7.20.5",
        "@babel/node": "^7.20.5",
        "@babel/preset-env": "^7.20.2"
      }
    },
    "node_modules/@ampproject/remapping": {
      "version": "2.2.0",
      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-2.2.0.tgz",
      "integrity": "sha512-qRmjj8nj9qmLTQXXmaR1cck3UXSRMPrbsLJAasZpF+t3riI71BXed5ebIOYwQntykeZuhjsdweEc9BxH5Jc26w==",
      "dev": true,
```


Lab02Net > graphql_movies > {} package.json > {} dependencies

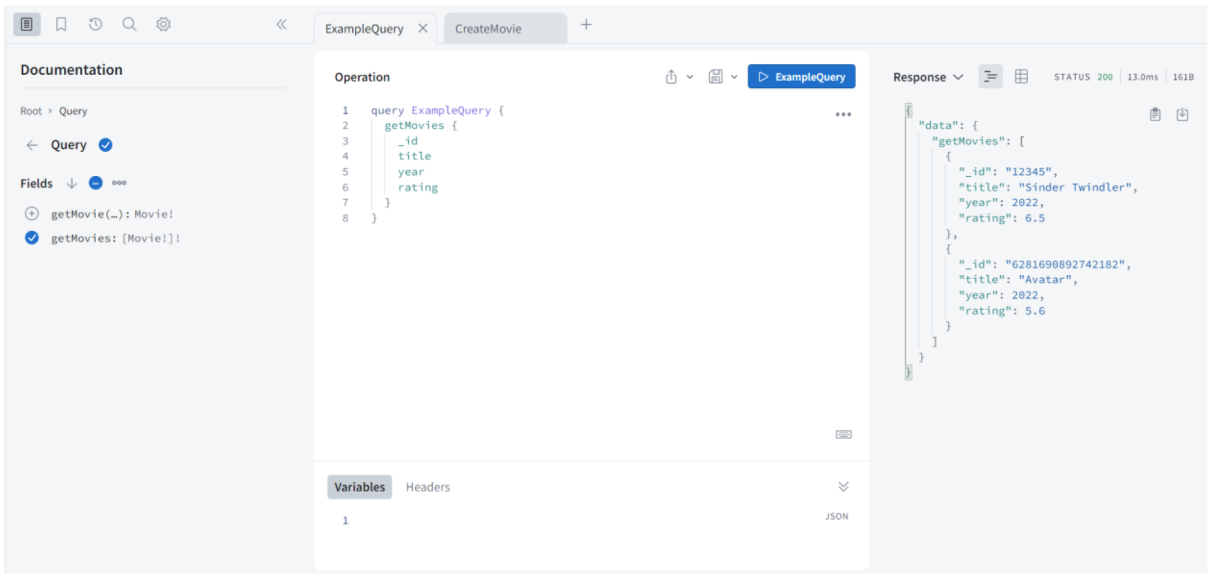
```json
{
  "name": "graphql_movies",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
     "start:dev": "babel-node src/index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "apollo-datasource-mongodb": "^0.5.4",
    "apollo-server": "^3.11.1",
    "dotenv": "^16.0.3",
    "graphql": "^15.8.0",
    "mongoose": "^6.8.1",
    "rimraf": "^3.0.2"
  },
  "devDependencies": {
    "@babel/cli": "^7.19.3",
    "@babel/core": "^7.20.5",
    "@babel/node": "^7.20.5",
    "@babel/preset-env": "^7.20.2"
  }
}
```



```graphql
mutation CreateMovie($title: String!, $rating: Float!, $year: Int!) {
  createMovie(title: $title, rating: $rating, year: $year) {
    rating
    title
    year
  }
}
```

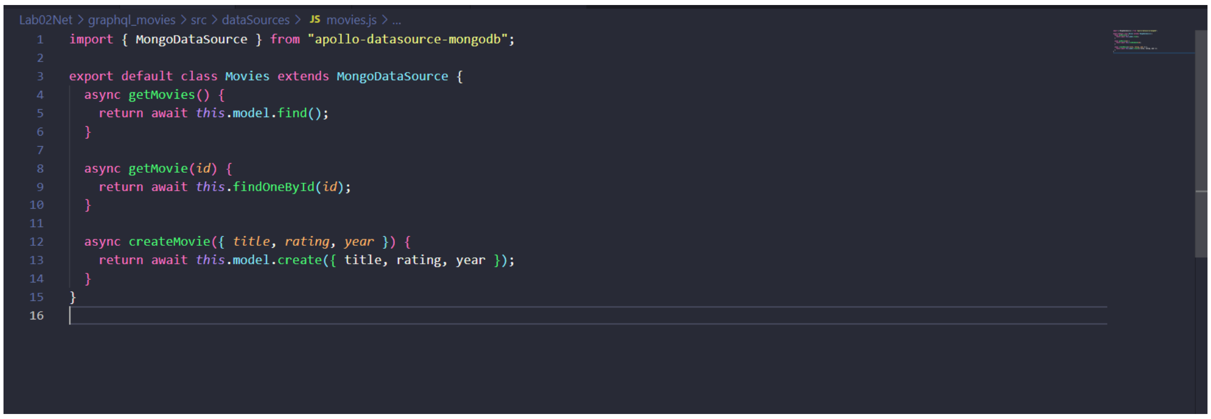
```json
{
  "data": {
    "createMovie": {
      "rating": 5.6,
      "title": "Avatar",
      "year": 2022
    }
  }
}
```

```json
{  "title": "Avatar",
  "rating": 5.6,
  "year": 2022
}
```

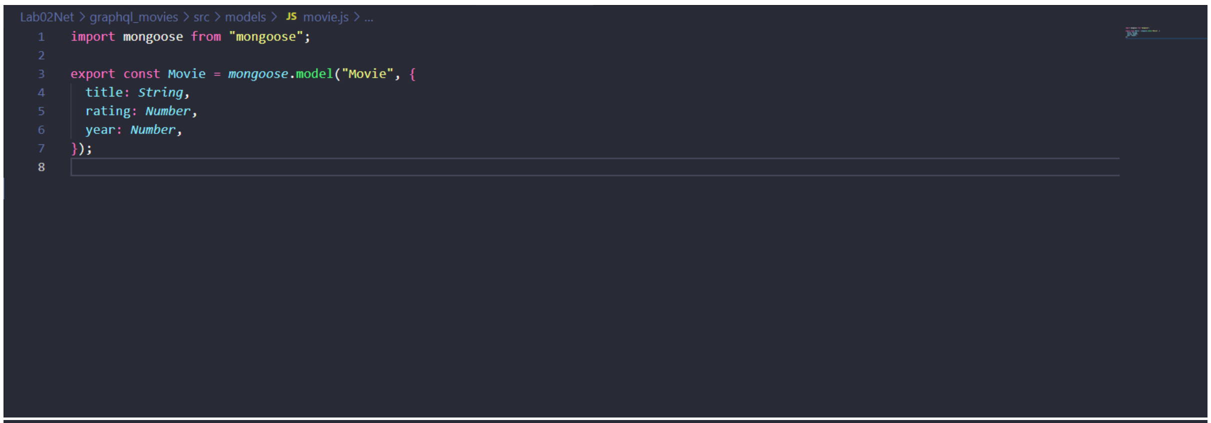


```graphql
query ExampleQuery {
  getMovies {
    _id
    title
    year
    rating
  }
}
```

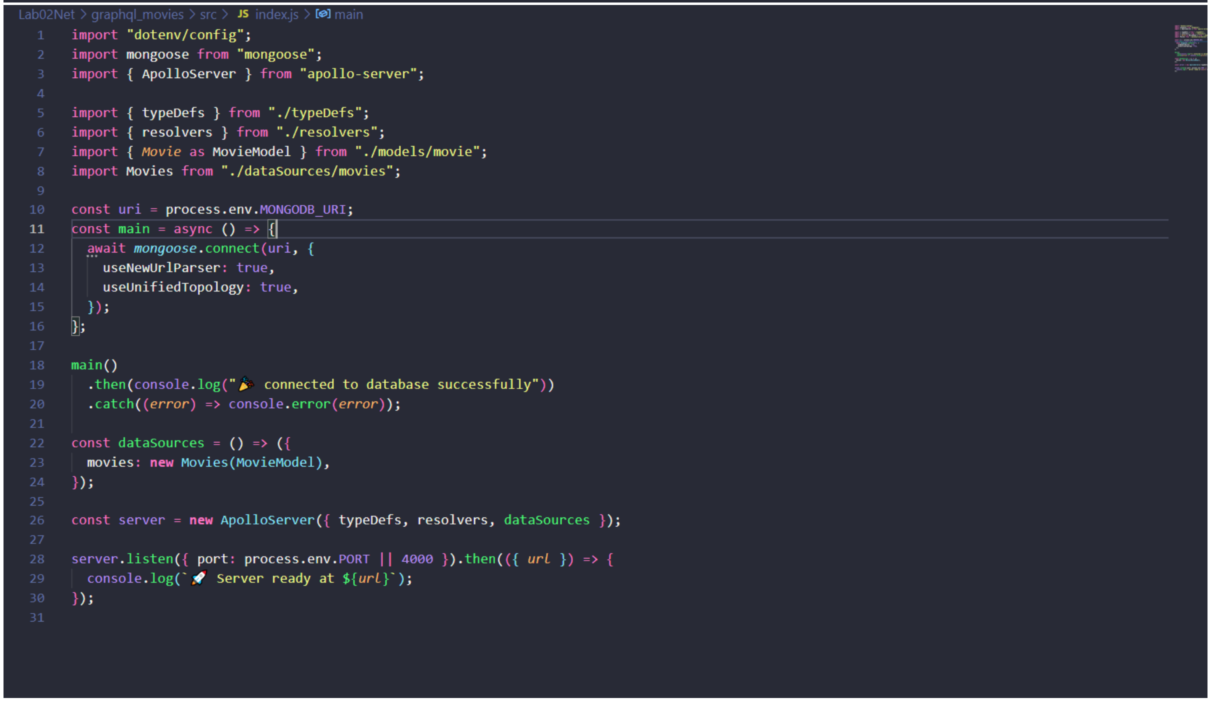
```json
{
  "data": {
    "getMovies": [
      {
        "_id": "12345",
        "title": "Sinder Twindler",
        "year": 2022,
        "rating": 6.5
      },
      {
        "_id": "6281690892742182",
        "title": "Avatar",
        "year": 2022,
        "rating": 5.6
      }
    ]
  }
}
```


Lab02Net > graphql_movies > src > dataSources > JS movies.js > ...

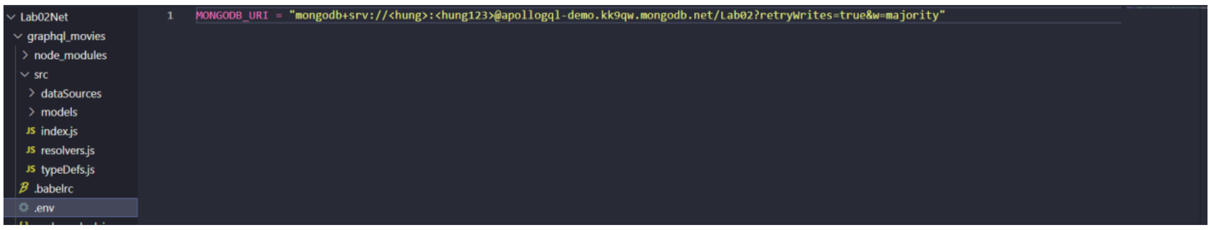
```javascript
import { MongoDataSource } from "apollo-datasource-mongodb";

export default class Movies extends MongoDataSource {
  async getMovies() {
    return await this.model.find();
  }

  async getMovie(id) {
    return await this.findOneById(id);
  }

  async createMovie({ title, rating, year }) {
    return await this.model.create({ title, rating, year });
  }
}

```


Lab02Net > graphql_movies > src > models > JS movie.js > ...

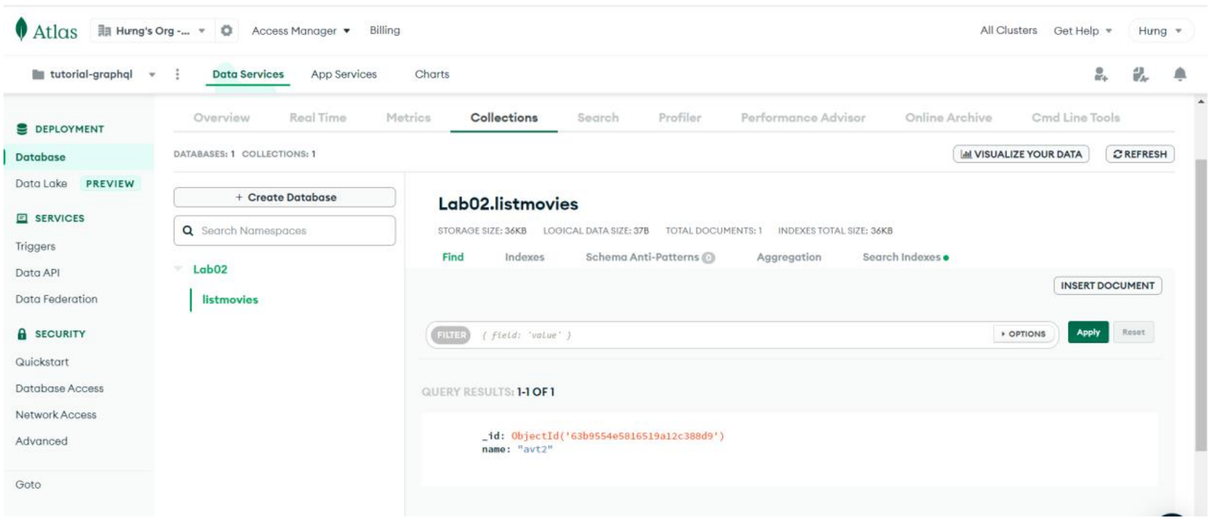
```js
import mongoose from "mongoose";

export const Movie = mongoose.model("Movie", {
  title: String,
  rating: Number,
  year: Number,
});

```


Lab02Net > graphql_movies > src > JS index.js > main

```js
import "dotenv/config";
import mongoose from "mongoose";
import { ApolloServer } from "apollo-server";

import { typeDefs } from "./typeDefs";
import { resolvers } from "./resolvers";
import { Movie as MovieModel } from "./models/movie";
import Movies from "./dataSources/movies";

const uri = process.env.MONGODB_URI;
const main = async () => {
  await mongoose.connect(uri, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  });
};

main()
  .then(console.log("🎉 connected to database successfully"))
  .catch((error) => console.error(error));

const dataSources = () => ({
  movies: new Movies(MovieModel),
});

const server = new ApolloServer({ typeDefs, resolvers, dataSources });

server.listen({ port: process.env.PORT || 4000 }).then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});

```


Lab02Net
graphql_movies
node_modules
src
dataSources
models
index.js
resolvers.js
typeDefs.js
.babelrc
.env

```
MONGODB_URI = "mongodb+srv://<hung>:<hung123>@apollogql-demo.kk9qw.mongodb.net/Lab02?retryWrites=true&w=majority"
```



DATABASES: 1 COLLECTIONS: 1

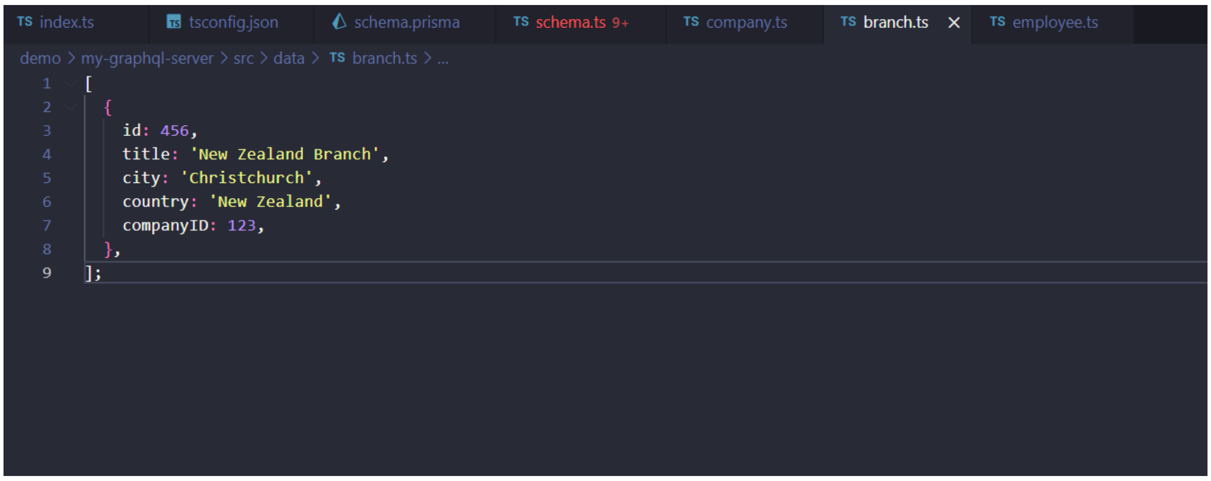
Lab02.listmovies

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 37B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB

`{ field: 'value' }`

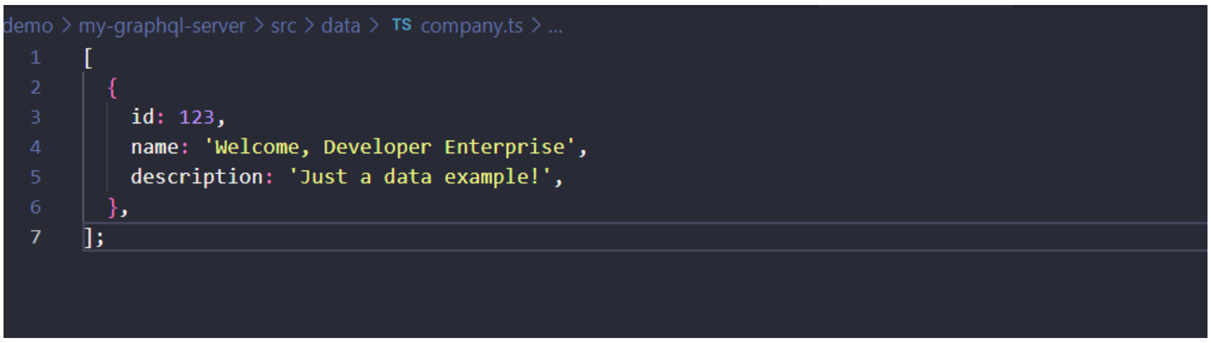
QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('63b9554e5816519a12c388d9')
name: "avt2"
```



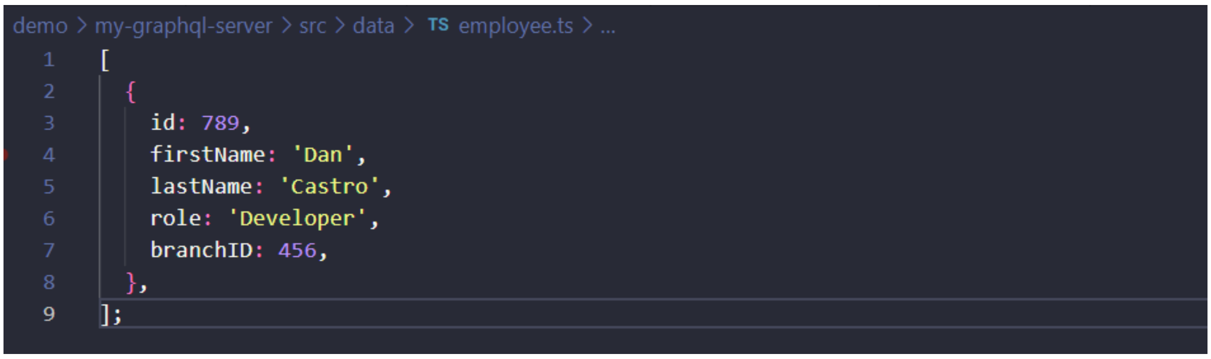
demo > my-graphql-server > src > data > branch.ts > ...

```ts
[
  {
    id: 456,
    title: 'New Zealand Branch',
    city: 'Christchurch',
    country: 'New Zealand',
    companyID: 123,
  },
];
```



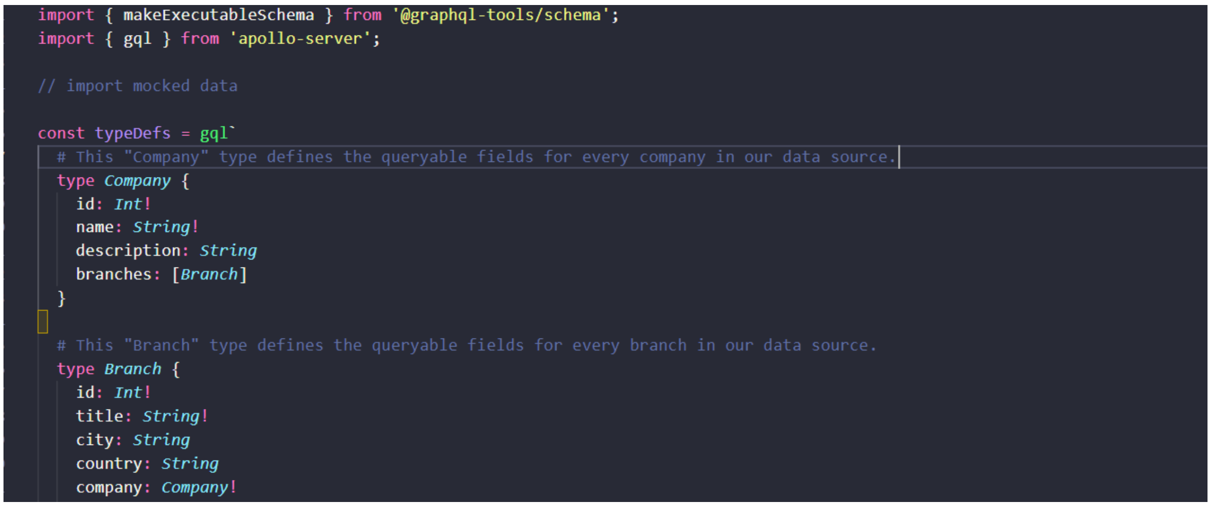
demo > my-graphql-server > src > data > company.ts > ...

```ts
[
  {
    id: 123,
    name: 'Welcome, Developer Enterprise',
    description: 'Just a data example!',
  },
];
```

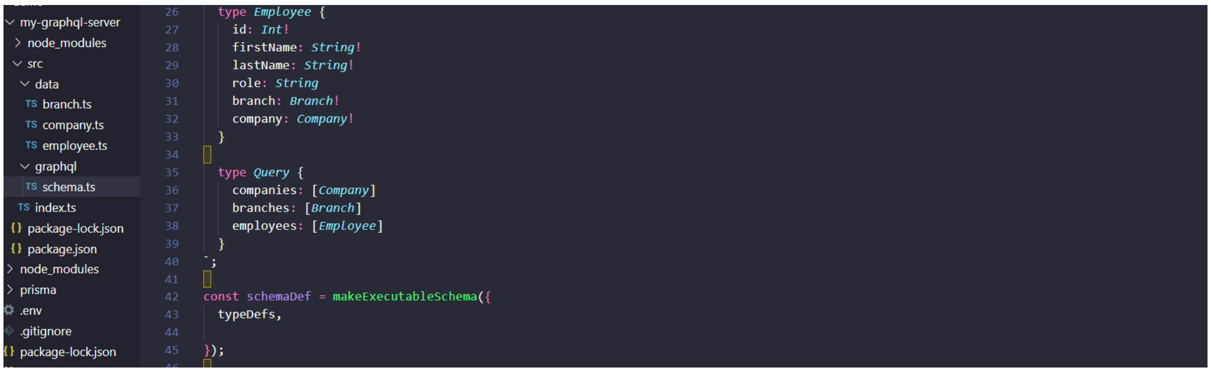


demo > my-graphql-server > src > data > employee.ts > ...

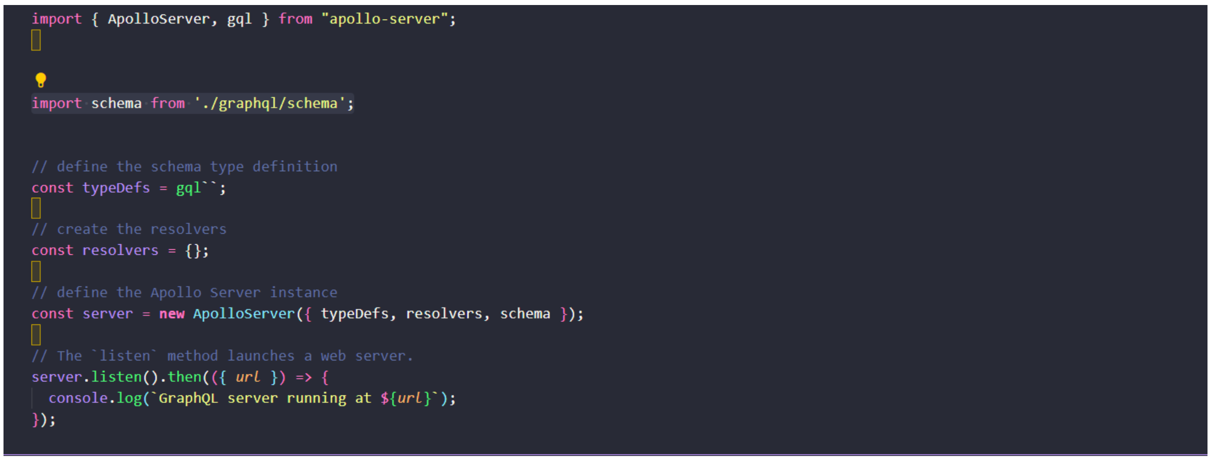
```ts
[
  {
    id: 789,
    firstName: 'Dan',
    lastName: 'Castro',
    role: 'Developer',
    branchID: 456,
  },
];
```



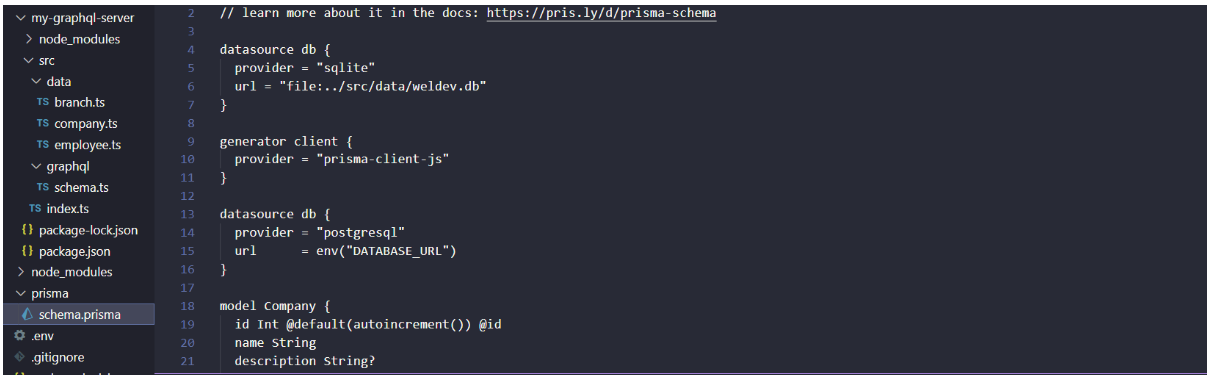
```
import { makeExecutableSchema } from '@graphql-tools/schema';
import { gql } from 'apollo-server';

// import mocked data

const typeDefs = gql`
  # This "Company" type defines the queryable fields for every company in our data source.
  type Company {
    id: Int!
    name: String!
    description: String
    branches: [Branch]
  }

  # This "Branch" type defines the queryable fields for every branch in our data source.
  type Branch {
    id: Int!
    title: String!
    city: String
    country: String
    company: Company!
```



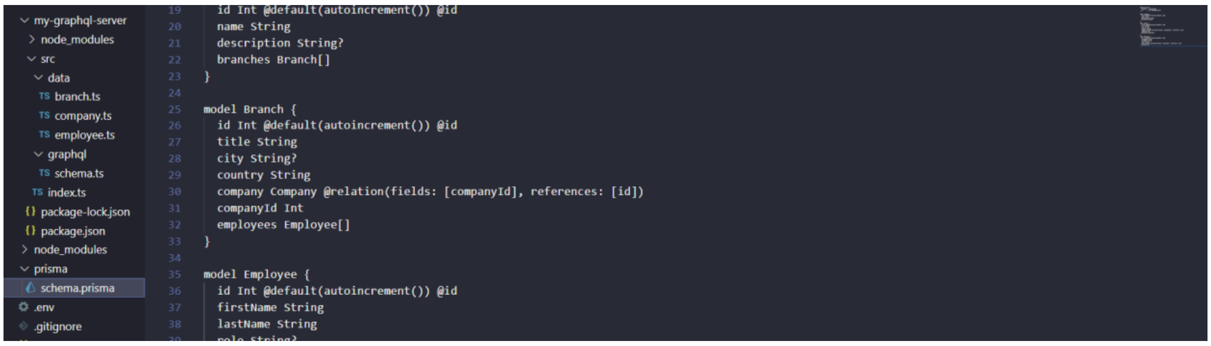
```
  type Employee {
    id: Int!
    firstName: String!
    lastName: String!
    role: String
    branch: Branch!
    company: Company!
  }

  type Query {
    companies: [Company]
    branches: [Branch]
    employees: [Employee]
  }
`;

const schemaDef = makeExecutableSchema({
  typeDefs,

});
```



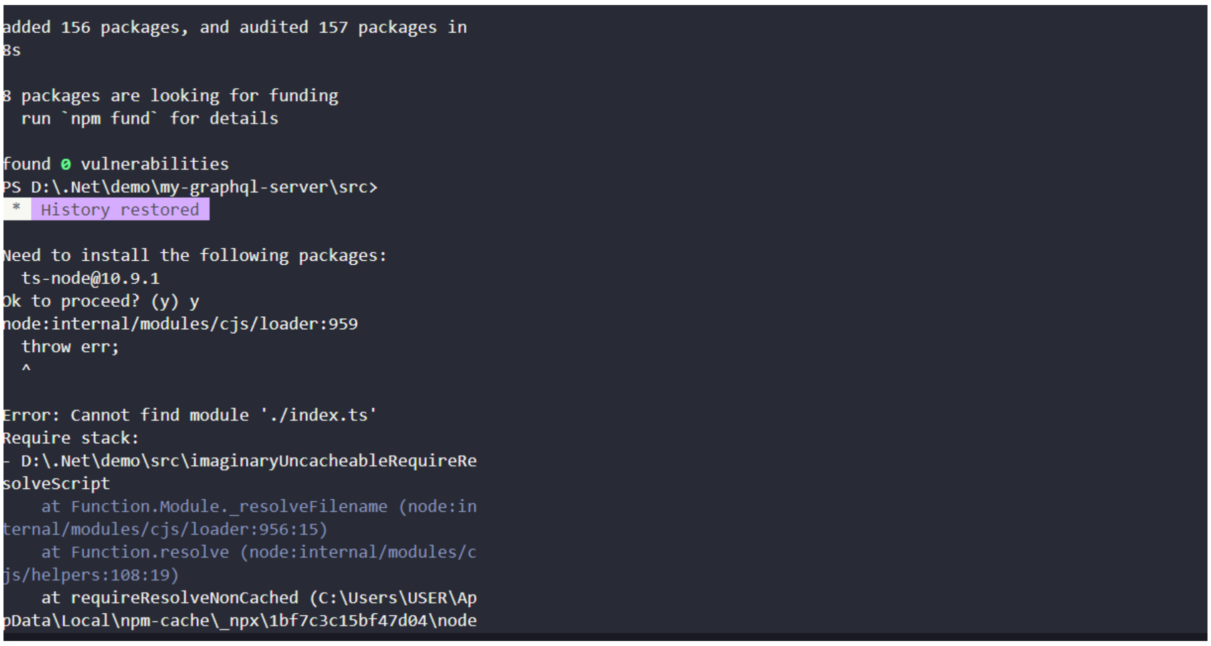
```
import { ApolloServer, gql } from "apollo-server";


import schema from './graphql/schema';


// define the schema type definition
const typeDefs = gql``;

// create the resolvers
const resolvers = {};

// define the Apollo Server instance
const server = new ApolloServer({ typeDefs, resolvers, schema });

// The `listen` method launches a web server.
server.listen().then(({ url }) => {
  console.log(`GraphQL server running at ${url}`);
});
```



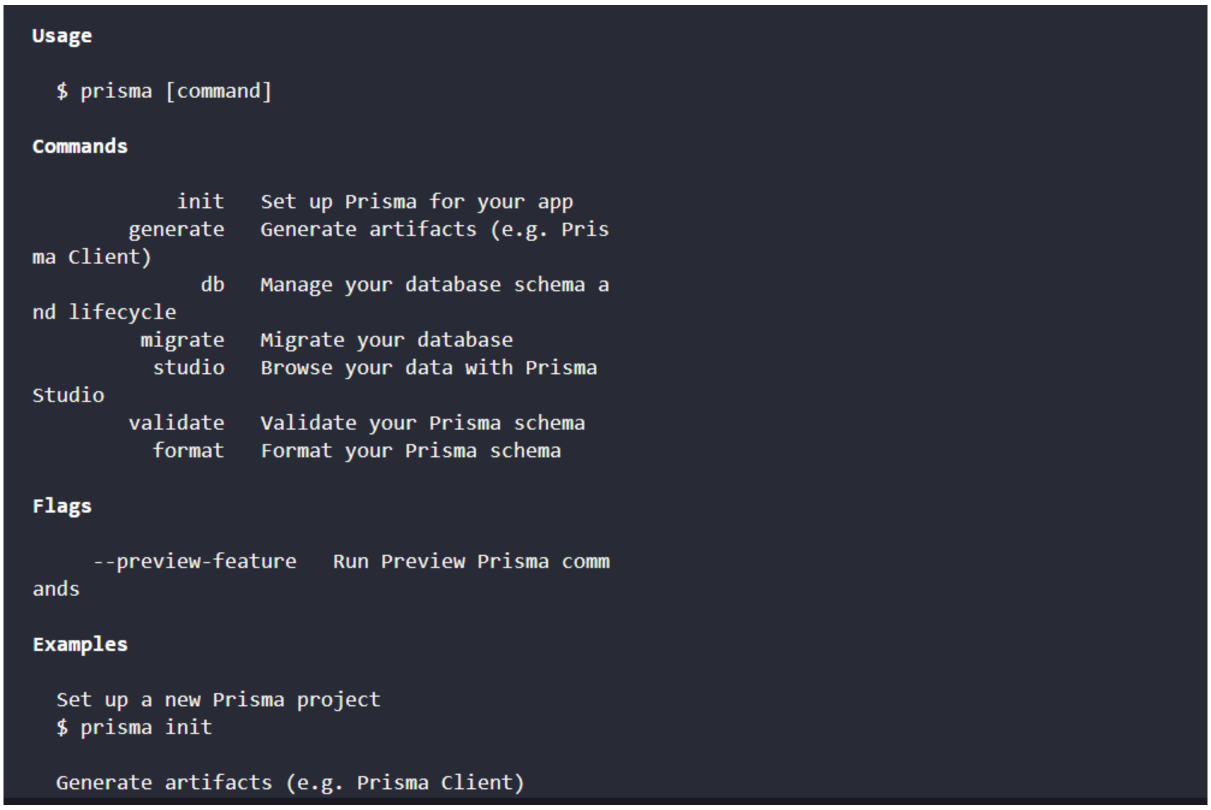
```
// learn more about it in the docs: https://pris.ly/d/prisma-schema

datasource db {
  provider = "sqlite"
  url = "file:../src/data/weldev.db"
}

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model Company {
  id Int @default(autoincrement()) @id
  name String
  description String?
```



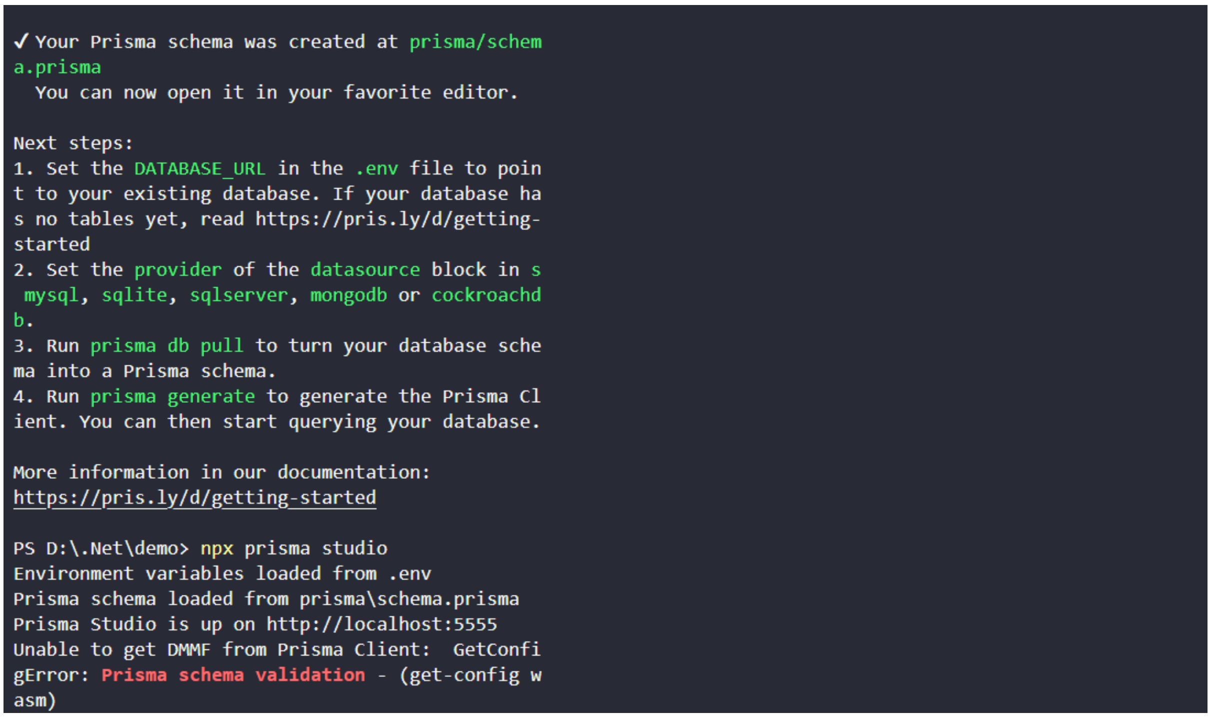
```
  id Int @default(autoincrement()) @id
  name String
  description String?
  branches Branch[]
}

model Branch {
  id Int @default(autoincrement()) @id
  title String
  city String?
  country String
  company Company @relation(fields: [companyId], references: [id])
  companyId Int
  employees Employee[]
}

model Employee {
  id Int @default(autoincrement()) @id
  firstName String
  lastName String
  role String?
```



```
added 156 packages, and audited 157 packages in
8s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\.Net\demo\my-graphql-server\src>
 *  History restored

Need to install the following packages:
  ts-node@10.9.1
Ok to proceed? (y) y
node:internal/modules/cjs/loader:959
  throw err;
  ^

Error: Cannot find module './index.ts'
Require stack:
- D:\.Net\demo\src\imaginaryUncacheableRequireRe
solveScript
    at Function.Module._resolveFilename (node:in
ternal/modules/cjs/loader:956:15)
    at Function.resolve (node:internal/modules/c
js/helpers:108:19)
    at requireResolveNonCached (C:\Users\USER\Ap
pData\Local\npm-cache\_npx\1bf7c3c15bf47d04\node
```



```
Usage

  $ prisma [command]

Commands

            init   Set up Prisma for your app
        generate   Generate artifacts (e.g. Pris
ma Client)
              db   Manage your database schema a
nd lifecycle
         migrate   Migrate your database
          studio   Browse your data with Prisma
Studio
        validate   Validate your Prisma schema
          format   Format your Prisma schema

Flags

     --preview-feature   Run Preview Prisma comm
ands

Examples

  Set up a new Prisma project
  $ prisma init

  Generate artifacts (e.g. Prisma Client)
```



```
✓ Your Prisma schema was created at prisma/schem
a.prisma
  You can now open it in your favorite editor.

Next steps:
1. Set the DATABASE_URL in the .env file to poin
t to your existing database. If your database ha
s no tables yet, read https://pris.ly/d/getting-
started
2. Set the provider of the datasource block in s
 mysql, sqlite, sqlserver, mongodb or cockroachd
b.
3. Run prisma db pull to turn your database sche
ma into a Prisma schema.
4. Run prisma generate to generate the Prisma Cl
ient. You can then start querying your database.

More information in our documentation:
https://pris.ly/d/getting-started

PS D:\.Net\demo> npx prisma studio
Environment variables loaded from .env
Prisma schema loaded from prisma\schema.prisma
Prisma Studio is up on http://localhost:5555
Unable to get DMMF from Prisma Client:  GetConfi
gError: Prisma schema validation - (get-config w
asm)
```