

```

Lab02Net > graphql_movies > src > JS index.js > ...
1  import { ApolloServer } from "apollo-server";
2  import { typeDefs } from "../typeDefs";
3  import { resolvers } from "../resolvers";
4
5  const server = new ApolloServer({ typeDefs, resolvers });
6
7  server.listen({ port: process.env.PORT || 4000 }).then(({ url }) => {
8    console.log(`🚀 Server ready at ${url}`);
9  });
10

```

```

Lab02Net > graphql_movies > src > JS resolvers.js > ...
1  const movies = [
2    {
3      _id: "12345",
4      title: "Sinder Twindler",
5      year: 2022,
6      rating: 6.5,
7    },
8  ];
9
10 export const resolvers = {
11   Query: {
12     getMovies: (_root, _args, _context, _info) => {
13       return movies;
14     },
15     getMovie: (_root, { id }, _context, _info) => {
16       return movies.find(({ _id }) => _id === id);
17     },
18   },
19   Mutation: {
20     createMovie: (_root, args, _context, _info) => {
21       const randomId = Math.random().toString().split(".")[1];
22       const newMovie = { ...args, _id: randomId };
23       movies.push(newMovie);
24       return newMovie;
25     },
26   },
27 };
28

```

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

```
1  import { gql } from "apollo-server";
2
3  export const typeDefs = gql`
4    type Movie {
5      _id: ID!
6      title: String!
7      rating: Float!
8      year: Int!
9    }
10
11    type Query {
12      getMovies: [Movie!]!
13      getMovie(id: ID!): Movie!
14    }
15
16    type Mutation {
17      createMovie(title: String!, rating: Float!, year: Int!): Movie!
18    }
19  `;
20
```

Lab02Net > graphql_movies > B .babelrc > ...

```
1  {
2    "presets": [
3      [
4        "@babel/preset-env",
5        {
6          "useBuiltIns": "usage",
7          "corejs": "3.0.0"
8        }
9      ]
10   ]
11 }
```

```

{
  "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-qRj8nE/1PjQqZDp01b+O1JJFgHx6U1H3VU3fAz5tB5oMPPOpN4ZeqJntB9yZWQdNS3N4T2EpS4BD0XaUlQ==",
      "dev": true,

```

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

```

1  {
2    "name": "graphql_movies",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start:dev": "babel-node src/index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "apollo-datasource-mongodb": "^0.5.4",
14     "apollo-server": "^3.11.1",
15     "dotenv": "^16.0.3",
16     "graphql": "^15.8.0",
17     "mongoose": "^6.8.1",
18     "rimraf": "^3.0.2"
19   },
20   "devDependencies": {
21     "@babel/cli": "^7.19.3",
22     "@babel/core": "^7.20.5",
23     "@babel/node": "^7.20.5",
24     "@babel/preset-env": "^7.20.2"
25   }
26 }
27

```

Operation

CreateMovie

```
1  mutation CreateMovie($title: String!, $rating: Float!, $year: Int!) { ...
2    createMovie(title: $title, rating: $rating, year: $year) {
3      rating
4      title
5      year
6    }
7  }
```

Variables

Headers

1 { "title": "Avatar",
2 "rating": 5.6,
3 "year": 2022
4 }

JSON

Response

STATUS 200 | 17.0ms | 69B

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

Documentation

Root > Query

← Query

Fields

getMovie(_): Movie!

getMovies: [Movie!]!

Operation

ExampleQuery

```
1  query ExampleQuery {
2    getMovies {
3      _id
4      title
5      year
6      rating
7    }
8  }
```

Variables

Headers

1

JSON

Response

STATUS 200 | 13.0ms | 161B

```
{
  "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 > ...
1  import { MongoDataSource } from "apollo-datasource-mongodb";
2
3  export default class Movies extends MongoDataSource {
4    async getMovies() {
5      return await this.model.find();
6    }
7
8    async getMovie(id) {
9      return await this.findOneById(id);
10   }
11
12   async createMovie({ title, rating, year }) {
13     return await this.model.create({ title, rating, year });
14   }
15 }
16
```

Lab02Net > graphql_movies > src > models > JS moviejs > ...

```
1 import mongoose from "mongoose";
2
3 export const Movie = mongoose.model("Movie", {
4   title: String,
5   rating: Number,
6   year: Number,
7 });
8
```

Lab02Net > graphql_movies > src > JS indexjs > [0] main

```
1 import "dotenv/config";
2 import mongoose from "mongoose";
3 import { ApolloServer } from "apollo-server";
4
5 import { typeDefs } from "./typeDefs";
6 import { resolvers } from "./resolvers";
7 import { Movie as MovieModel } from "./models/movie";
8 import Movies from "./dataSources/movies";
9
10 const uri = process.env.MONGODB_URI;
11 const main = async () => {
12   await mongoose.connect(uri, {
13     useNewUrlParser: true,
14     useUnifiedTopology: true,
15   });
16 };
17
18 main()
19   .then(console.log("🔗 connected to database successfully"))
20   .catch((error) => console.error(error));
21
22 const dataSources = () => ({
23   movies: new Movies(MovieModel),
24 });
25
26 const server = new ApolloServer({ typeDefs, resolvers, dataSources });
27
28 server.listen({ port: process.env.PORT || 4000 }).then(({ url }) => {
29   console.log(`🚀 Server ready at ${url}`);
30 });
31
```

Lab02Net 1 MONGODB_URI = "mongodb+srv://chung123:chung123@apollogql-demo.kk9qw.mongodb.net/Lab02?retryWrites=true&w=majority"

graphql_movies
node_modules
src
dataSources
models
JS indexjs
JS resolversjs
JS typeDefsjs
babelrc
.env

Atlas Hung's Org - ... Access Manager Billing All Clusters Get Help Hung

tutorial-graphql Data Services App Services Charts

DEPLOYMENT Database Data Lake PREVIEW SERVICES Triggers Data API Data Federation SECURITY Quickstart Database Access Network Access Advanced Goto

Overview Real Time Metrics Collections Search Profiler Performance Adviser Online Archive Cmd Line Tools

DATABASES: 1 COLLECTIONS: 1

+ Create Database

Search Namespaces

Lab02

listmovies

Lab02.listmovies

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

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: "value" } OPTIONS Apply Reset

QUERY RESULTS: 1-1 OF 1

```

{
  "_id": ObjectId("63b9554e5816519a12c388d9")
  "name": "avt2"
}

```

```

TS index.ts  tsconfig.json  schema.prisma  TS schema.ts 9+  TS company.ts  TS branch.ts x  TS employee.ts
demo > my-graphql-server > src > data > TS branch.ts > ...
1  [
2    {
3      id: 456,
4      title: 'New Zealand Branch',
5      city: 'Christchurch',
6      country: 'New Zealand',
7      companyID: 123,
8    },
9  ];

```

```

demo > my-graphql-server > src > data > TS company.ts > ...
1  [
2    {
3      id: 123,
4      name: 'Welcome, Developer Enterprise',
5      description: 'Just a data example!',
6    },
7  ];

```

```

demo > my-graphql-server > src > data > TS employee.ts > ...
1  [
2    {
3      id: 789,
4      firstName: 'Dan',
5      lastName: 'Castro',
6      role: 'Developer',
7      branchID: 456,
8    },
9  ];

```

```
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!
  }
`
```

```

26 type Employee {
27   id: Int!
28   firstName: String!
29   lastName: String!
30   role: String
31   branch: Branch!
32   company: Company!
33 }
34
35 type Query {
36   companies: [Company]
37   branches: [Branch]
38   employees: [Employee]
39 }
40
41
42 const schemaDef = makeExecutableSchema({
43   typeDefs,
44   resolvers,
45 });
46

```

```
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}`);
});
```

```

2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 datasource db {
5   provider = "sqlite"
6   url = "file:../src/data/weldev.db"
7 }
8
9 generator client {
10   provider = "prisma-client-js"
11 }
12
13 datasource db {
14   provider = "postgresql"
15   url = env("DATABASE_URL")
16 }
17
18 model Company {
19   id Int @default(autoincrement()) @id
20   name String
21   description String?

```

```

19   id Int @default(autoincrement()) @id
20   name String
21   description String?
22   branches Branch[]
23 }
24
25 model Branch {
26   id Int @default(autoincrement()) @id
27   title String
28   city String?
29   country String
30   company Company @relation(fields: [companyId], references: [id])
31   companyId Int
32   employees Employee[]
33 }
34
35 model Employee {
36   id Int @default(autoincrement()) @id
37   firstName String
38   lastName String
39   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:internal/modules/cjs/loader:956:15)

at Function.resolve (node:internal/modules/cjs/helpers:108:19)

at requireResolveNonCached (C:\Users\USER\AppData\Local\npm-cache\npx\1bf7c3c15bf47d04\node

Usage

\$ prisma [command]

Commands

init	Set up Prisma for your app
generate	Generate artifacts (e.g. Prisma Client)
db	Manage your database schema and 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 commands

Examples

Set up a new Prisma project

\$ prisma init

Generate artifacts (e.g. Prisma Client)

✓ Your Prisma schema was created at `prisma/schema.prisma`

You can now open it in your favorite editor.

Next steps:

1. Set the `DATABASE_URL` in the `.env` file to point to your existing database. If your database has no tables yet, read <https://pris.ly/d/getting-started>
2. Set the `provider` of the `datasource` block in `schema.prisma` to `mysql`, `sqlite`, `sqlserver`, `mongodb` or `cockroachdb`.
3. Run `prisma db pull` to turn your database schema into a Prisma schema.
4. Run `prisma generate` to generate the Prisma Client. 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: GetConfigError: Prisma schema validation - (get-config was not found)
```