**UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA**

**FACULTAD DE CIENCIAS Y TECNOLOGÍA**

**CARRERA DE INGENIERÍA DE SISTEMAS**



**MICROSERVICIOS (COM600)**

**PRACTICA #4 GRAPHQL**

**DOCENTE:** Ing. Montellano Barriga Carlos

**ESTUDIANTE:** Miranda Veniz Marcela

**SUCRE-BOLIVIA**

**Práctica uso de Graphql**

**Nombre Estudiante:** Miranda Veniz Marcela

**CU:** 35-5382

**g. Ejemplos de consultas en GraphQL Playground** Consultar todos los padrones con su mesa

query {

getPadrones {

id

nombres

apellidos

numero\_documento

mesa {

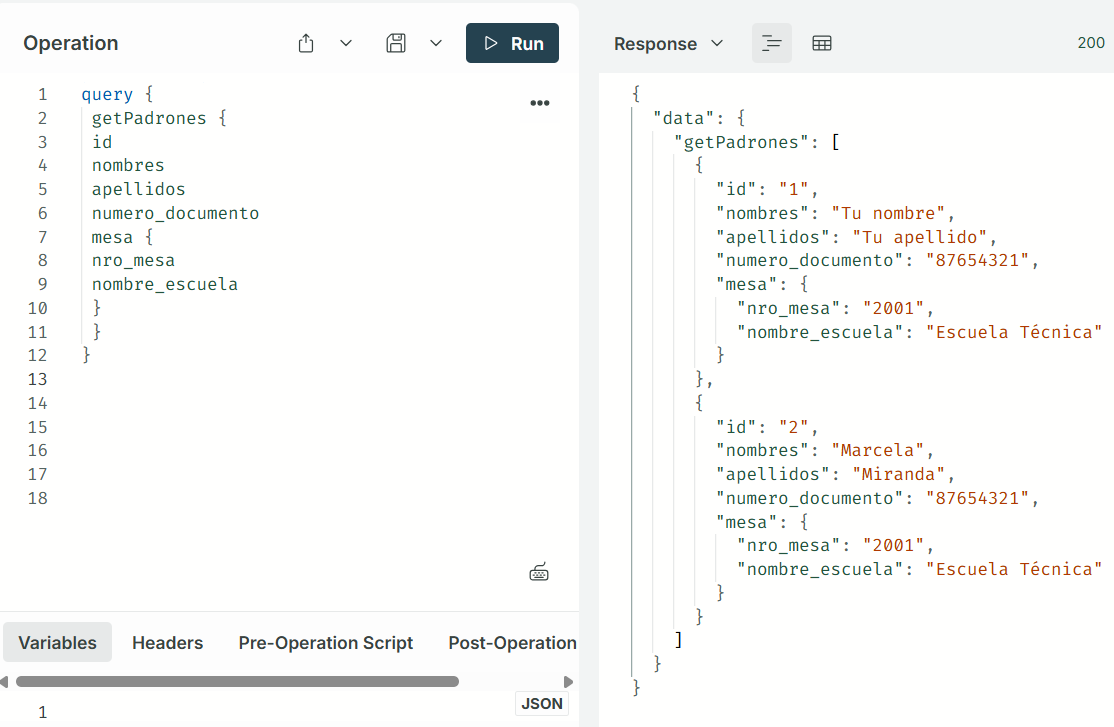
nro\_mesa

nombre\_escuela

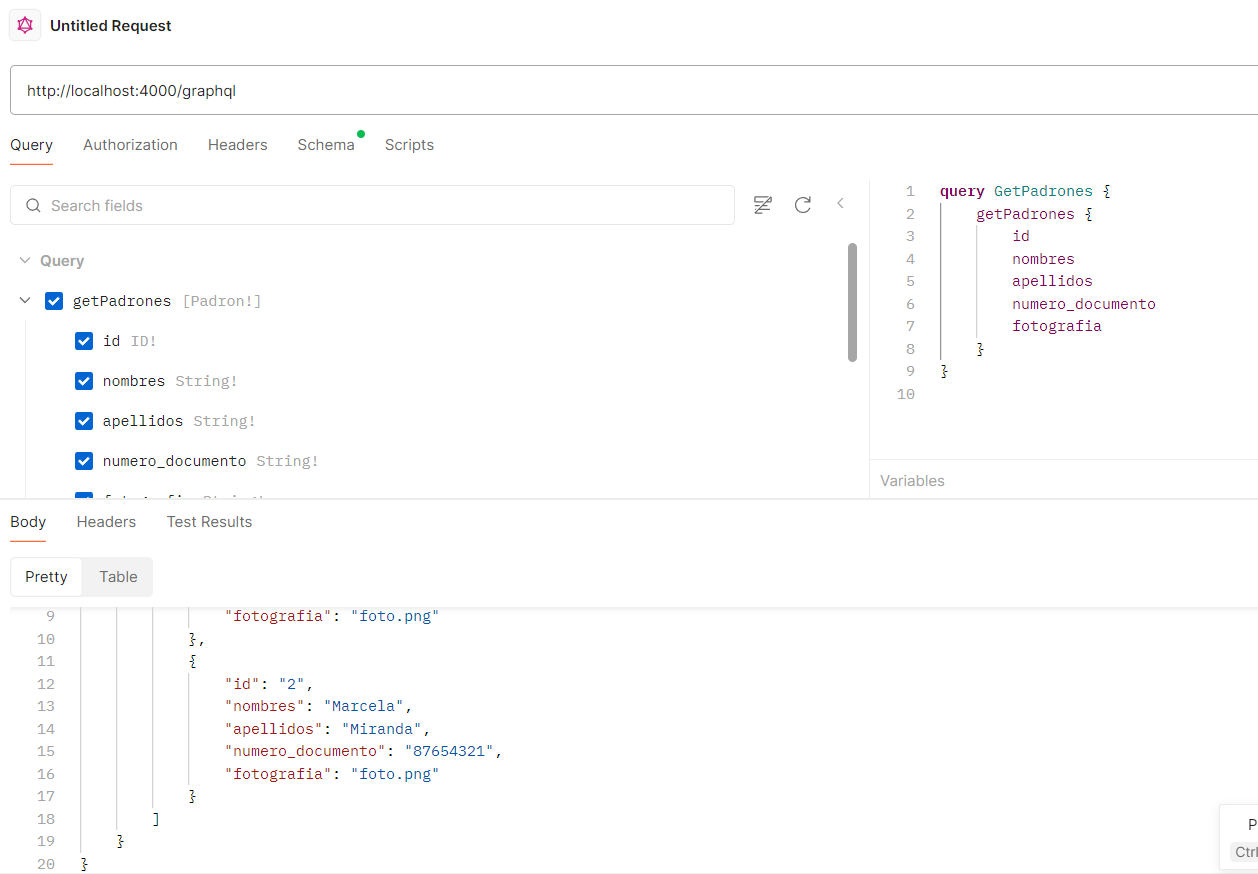
}

}

}



POSTMAN:



Consultar mesas con sus padrones

query {

getMesas {

nro\_mesa

nombre\_escuela

padrones {

nombres

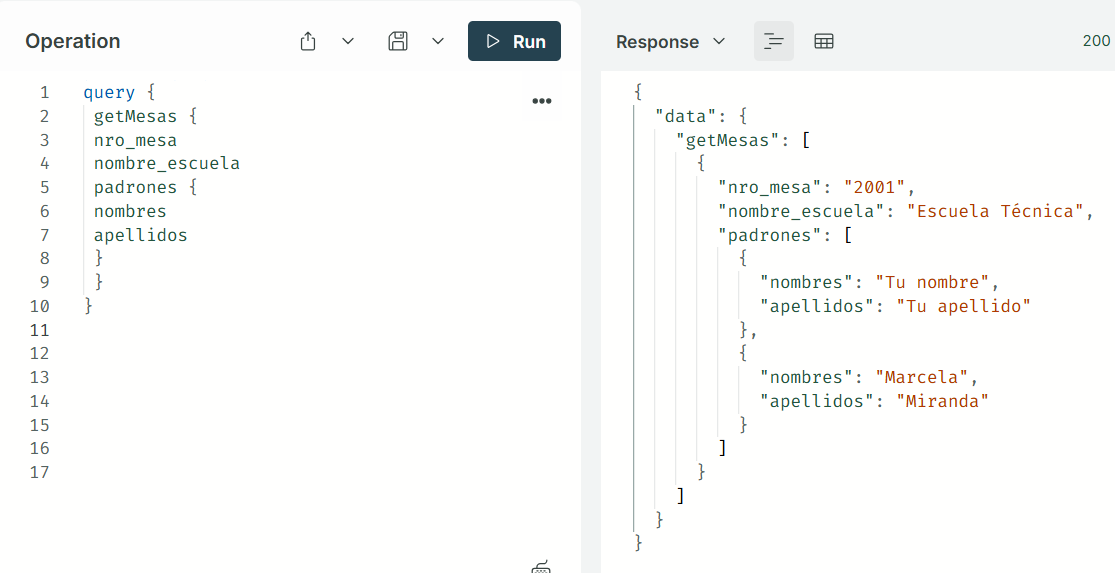
apellidos

}

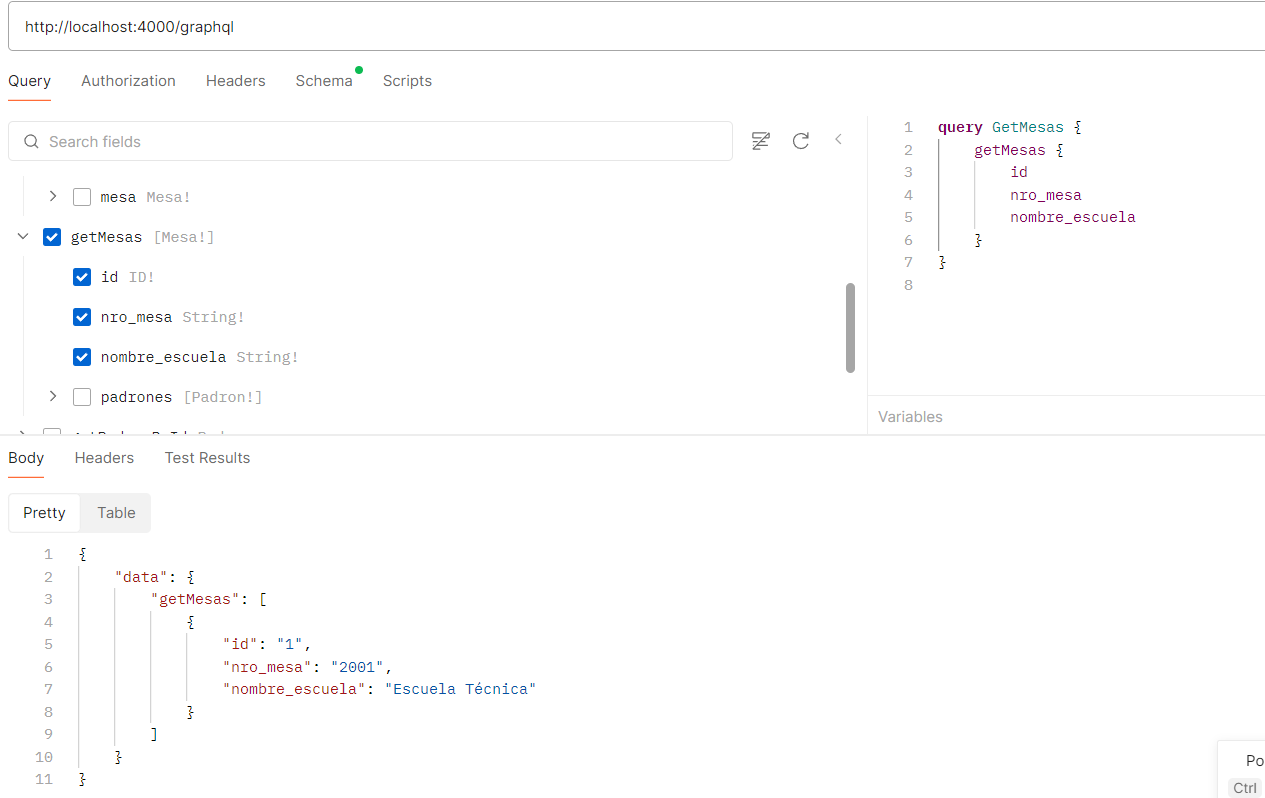
}

}





POSTMAN:



Crear una mesa

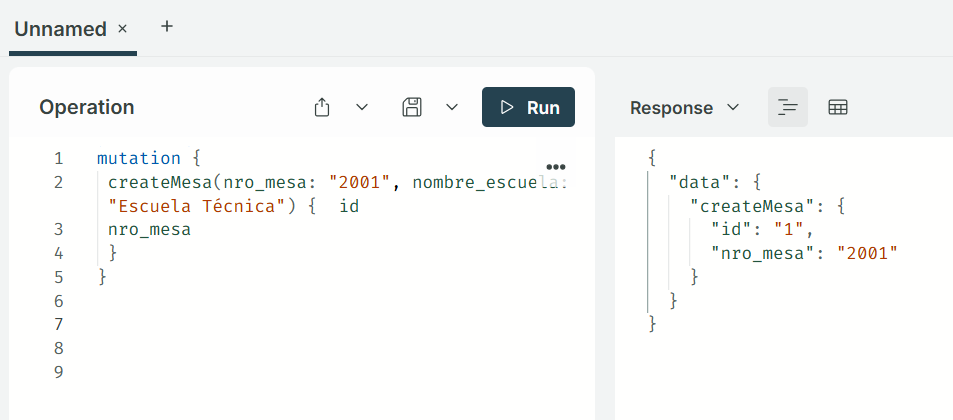
mutation {

createMesa(nro\_mesa: "2001", nombre\_escuela: "Escuela Técnica") { id

nro\_mesa

}

}



Crear un padron

mutation {

createPadron(

nombres: "Tu nombre"

apellidos: "Tu apellido"

numero\_documento: "87654321"

fotografia: "foto.png"

mesaId: 1

) {

id

nombres

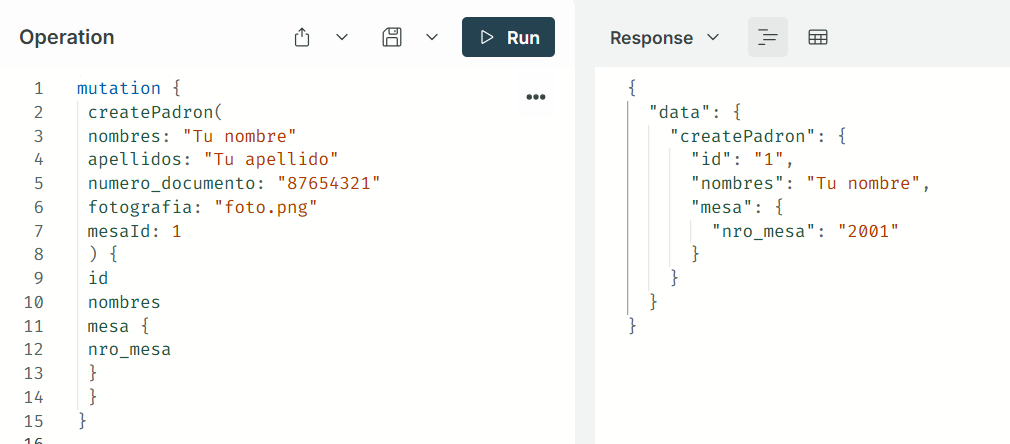
mesa {

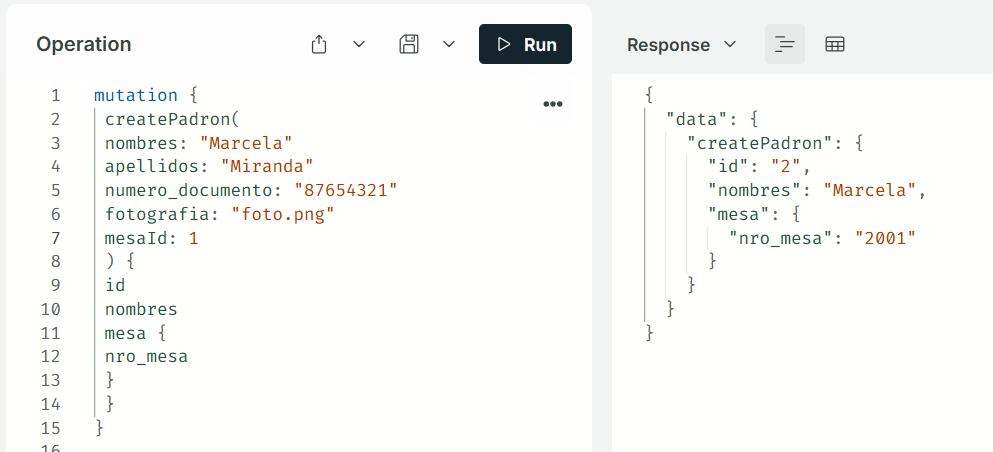
nro\_mesa

}

}

}





1. Haz correr el ejercicio y realizar captura de pantalla de la ejecución de las consultas tanto en postman como en la misma herramienta , colocalo en un documento word

**Enunciado del Ejercicio – Sistema de Biblioteca**

La **Biblioteca Central** necesita un sistema para **gestionar los libros y los préstamos realizados por los usuarios**.

El sistema debe permitir:

• **Registrar libros** con los campos: titulo, autor, isbn, anio\_publicacion. • **Registrar préstamos** (Prestamo) que guarden el usuario, la fecha\_prestamo, la fecha\_devolucion y el libro asociado (Libro). • **Consultar la lista de préstamos**, mostrando el usuario, fechas y los datos del libro prestado.

• **Consultar los libros disponibles** junto con el historial de préstamos asociados.

**Tareas que debes realizar**

a. **Esquema GraphQL**

o Define los **tipos** Libro y Prestamo.

o Define consultas (Query) para:

▪ Obtener todos los libros (getLibros).

▪ Obtener todos los préstamos (getPrestamos).

▪ Obtener un préstamo por su id (getPrestamoById).

o Define mutaciones (Mutation) para:

▪ Crear un libro (createLibro).

▪ Registrar un préstamo (createPrestamo).

Graphql-biblioteca/src/entity/Libro.js

const { EntitySchema } = require("typeorm");

module.exports = new EntitySchema({

  name: "Libro",

  tableName: "libros",

  columns: {

    id: { type: Number, primary: true, generated: true },

    titulo: { type: String },

    autor: { type: String },

    isbn: { type: String },

    anio\_publicacion: { type: Number }

  },

  relations: {

    prestamos: {

      type: "one-to-many",

      target: "Prestamo",

      inverseSide: "libro"

    }

  }

});

Graphql-biblioteca/src/entity/Prestamo.js

const { EntitySchema } = require("typeorm");

module.exports = new EntitySchema({

  name: "Prestamo",

  tableName: "prestamos",

  columns: {

    id: { type: Number, primary: true, generated: true },

    usuario: { type: String },

    fecha\_prestamo: { type: "date" },

    fecha\_devolucion: { type: "date" }

  },

  relations: {

    libro: {

      type: "many-to-one",

      target: "Libro",

      joinColumn: true,

      eager: true

    }

  }

});

Graphql-biblioteca/src/schema/resolvers.js

const { AppDataSource } = require("../database");

const resolvers = {

  Query: {

    getLibros: async () => {

      return await AppDataSource.getRepository("Libro").find({ relations: ["prestamos"] });

    },

    getPrestamos: async () => {

      return await AppDataSource.getRepository("Prestamo").find({ relations: ["libro"] });

    },

    getPrestamoById: async (\_, { id }) => {

      return await AppDataSource.getRepository("Prestamo").findOne({

        where: { id },

        relations: ["libro"]

      });

    },

    getPrestamosByUsuario: async (\_, { usuario }) => {

      return await AppDataSource.getRepository("Prestamo").find({

        where: { usuario },

        relations: ["libro"]

      });

    }

  },

  Mutation: {

    createLibro: async (\_, { titulo, autor, isbn, anio\_publicacion }) => {

      const repo = AppDataSource.getRepository("Libro");

      const libro = repo.create({ titulo, autor, isbn, anio\_publicacion });

      return await repo.save(libro);

    },

    createPrestamo: async (\_, { usuario, fecha\_prestamo, fecha\_devolucion, libroId }) => {

      const repoPrestamo = AppDataSource.getRepository("Prestamo");

      const repoLibro = AppDataSource.getRepository("Libro");

      const libro = await repoLibro.findOneBy({ id: libroId });

      if (!libro) throw new Error("Libro no encontrado");

      const prestamo = repoPrestamo.create({

        usuario,

        fecha\_prestamo,

        fecha\_devolucion,

        libro

      });

      return await repoPrestamo.save(prestamo);

    }

  }

};

module.exports = resolvers;

Graphql-biblioteca/src/schema/typeDefs.js

const { gql } = require("apollo-server-express");

const typeDefs = gql`

  type Libro {

    id: ID!

    titulo: String!

    autor: String!

    isbn: String!

    anio\_publicacion: Int!

    prestamos: [Prestamo!]

  }

  type Prestamo {

    id: ID!

    usuario: String!

    fecha\_prestamo: String!

    fecha\_devolucion: String!

    libro: Libro!

  }

  type Query {

    getLibros: [Libro!]

    getPrestamos: [Prestamo!]

    getPrestamoById(id: ID!): Prestamo

    getPrestamosByUsuario(usuario: String!): [Prestamo!] # desafío extra

  }

  type Mutation {

    createLibro(titulo: String!, autor: String!, isbn: String!, anio\_publicacion: Int!): Libro

    createPrestamo(usuario: String!, fecha\_prestamo: String!, fecha\_devolucion: String!, libroId: ID!): Prestamo

  }

`;

module.exports = typeDefs;

Graphql-biblioteca/src/database.js

require("reflect-metadata");

const { DataSource } = require("typeorm");

const Prestamo = require("./entity/Prestamo");

const Libro = require("./entity/Libro");

const AppDataSource = new DataSource({

  type: "mysql",

  host: "localhost",

  port: 3306,

  username: "root",

  password: "SerCu05/M20.", // tu contraseña si tienes

  database: "graphql\_biblioteca",

  synchronize: true,

  logging: false,

  entities: [Libro, Prestamo],

});

module.exports = { AppDataSource };

Graphql-biblioteca/src/index.js

require("reflect-metadata");

const express = require("express");

const { ApolloServer } = require("apollo-server-express");

const { AppDataSource } = require("./database");

const typeDefs = require("./schema/typeDefs");

const resolvers = require("./schema/resolvers");

async function startServer() {

  const app = express();

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

  await server.start();

  server.applyMiddleware({ app });

  await AppDataSource.initialize();

  console.log("✅ Conectado a la base de datos");

  app.listen(4000, () => {

    console.log(`🚀 Servidor listo en http://localhost:4000${server.graphqlPath}`);

  });

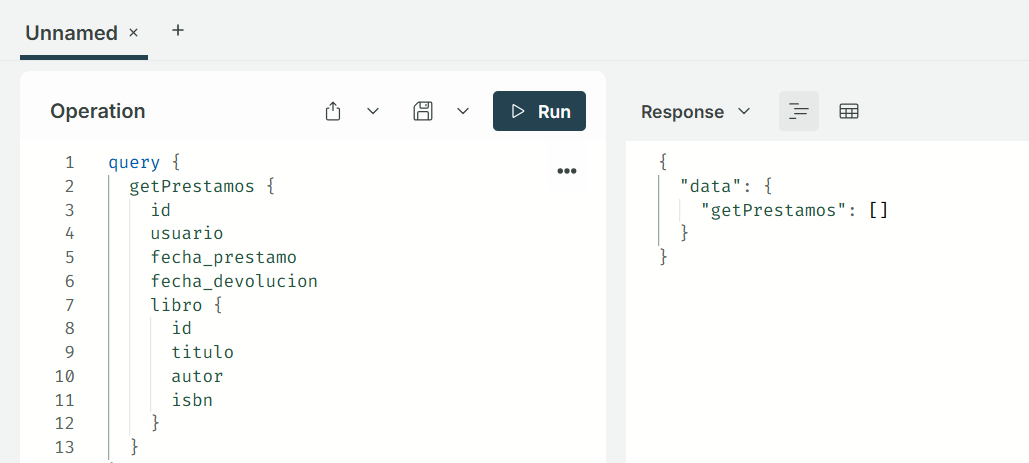
}

startServer();

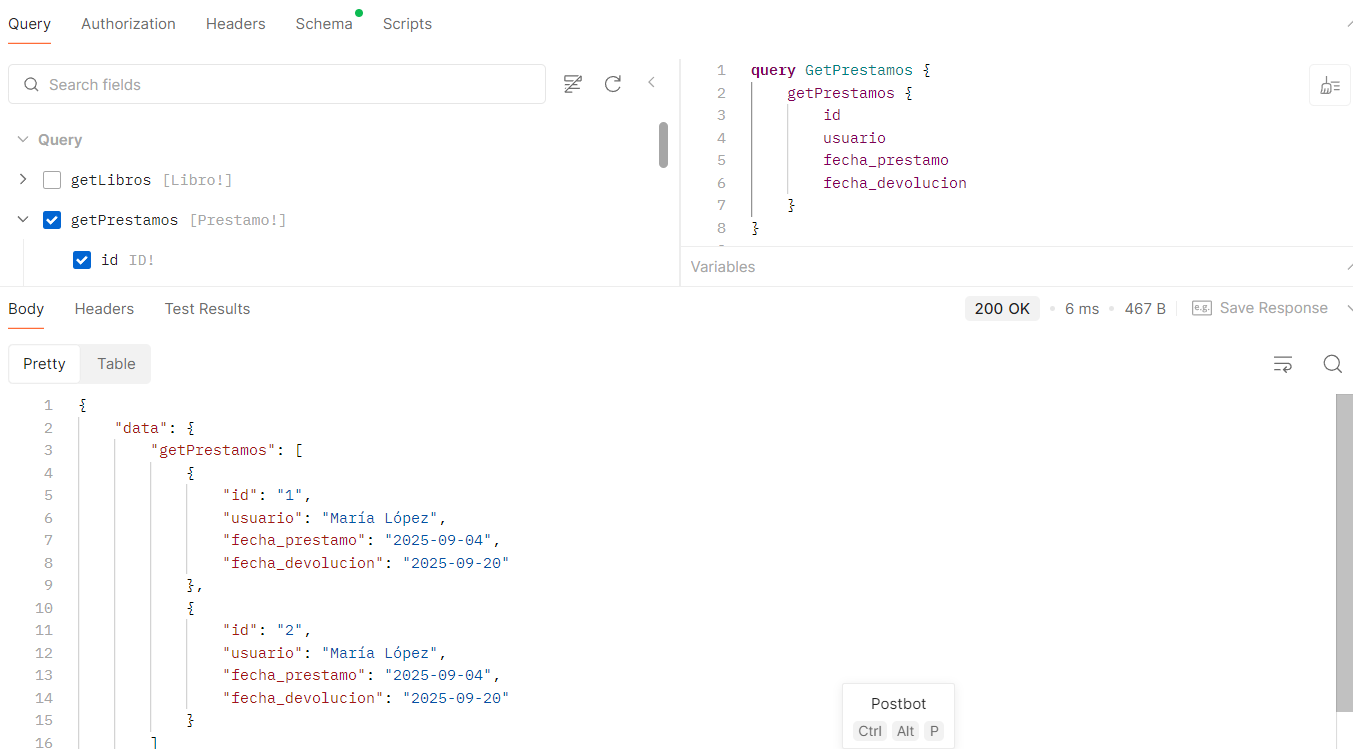
b. **Consultas GraphQL (Queries)**

o Listar todos los préstamos con el libro correspondiente.

HERRAMIENTA:

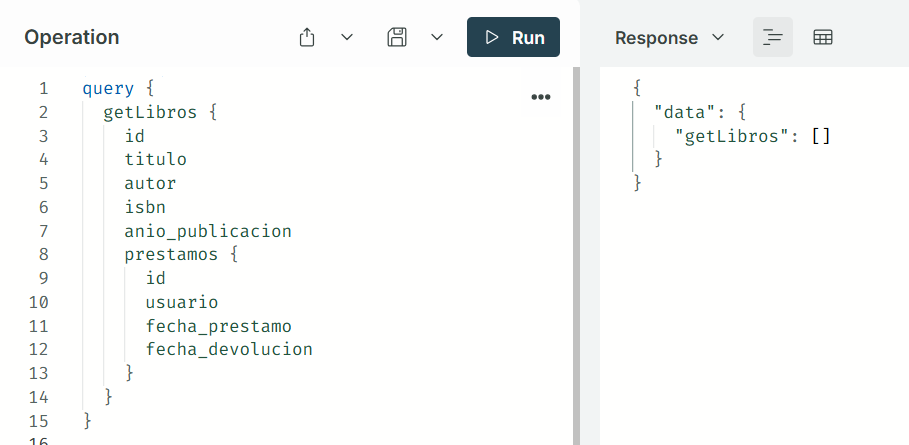


POSTMAN:

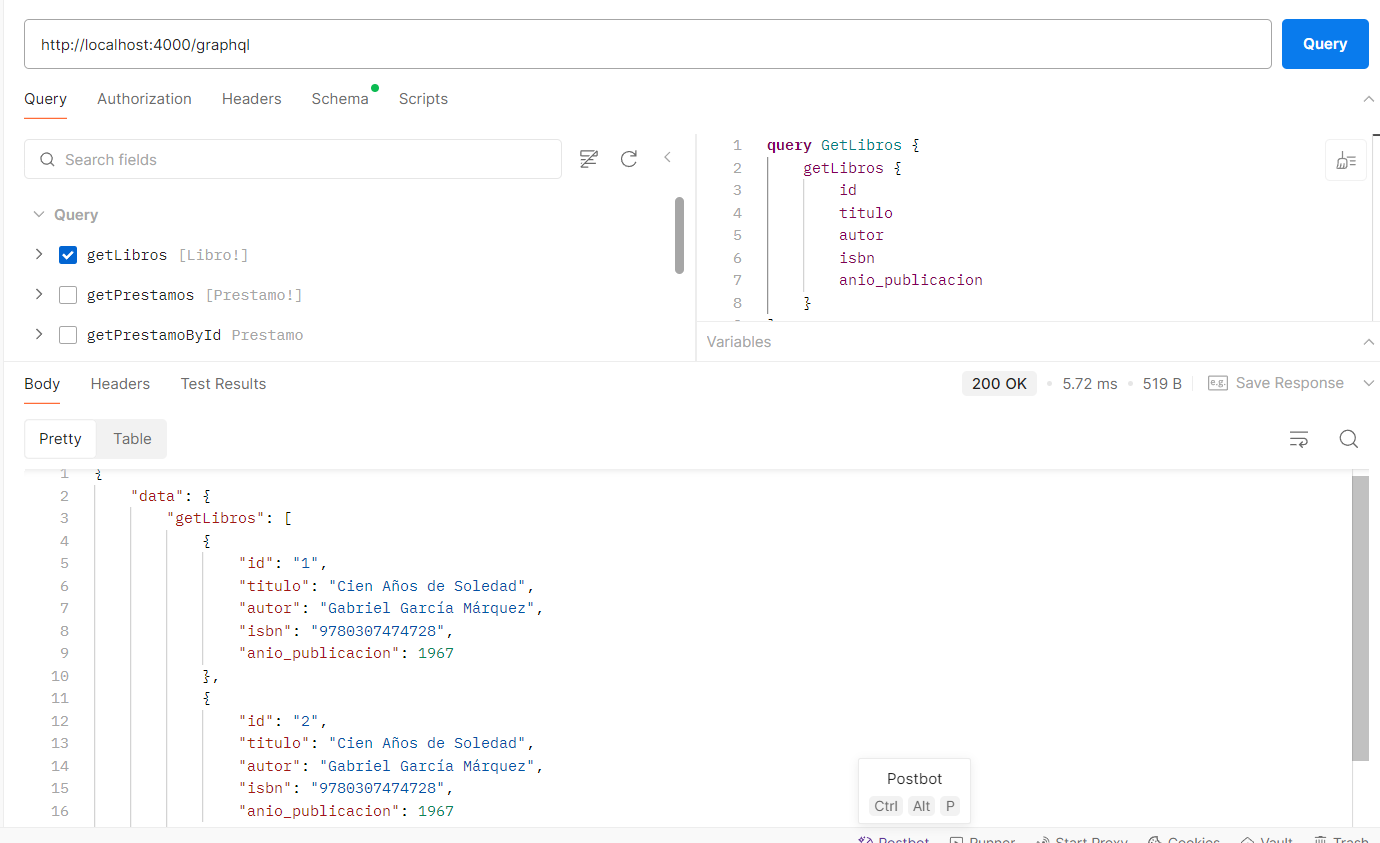


o Listar todos los libros junto con el historial de préstamos realizados.

HERRAMIENTA:



POSTMAN:



c. **Mutaciones GraphQL**

o Crear un libro nuevo:

▪ titulo: Cien Años de Soledad

▪ autor: Gabriel García Márquez

▪ isbn: 9780307474728

▪ anio\_publicacion: 1967.

HERRAMIENTA:



POSTMAN:



o Crear un préstamo con los siguientes datos:

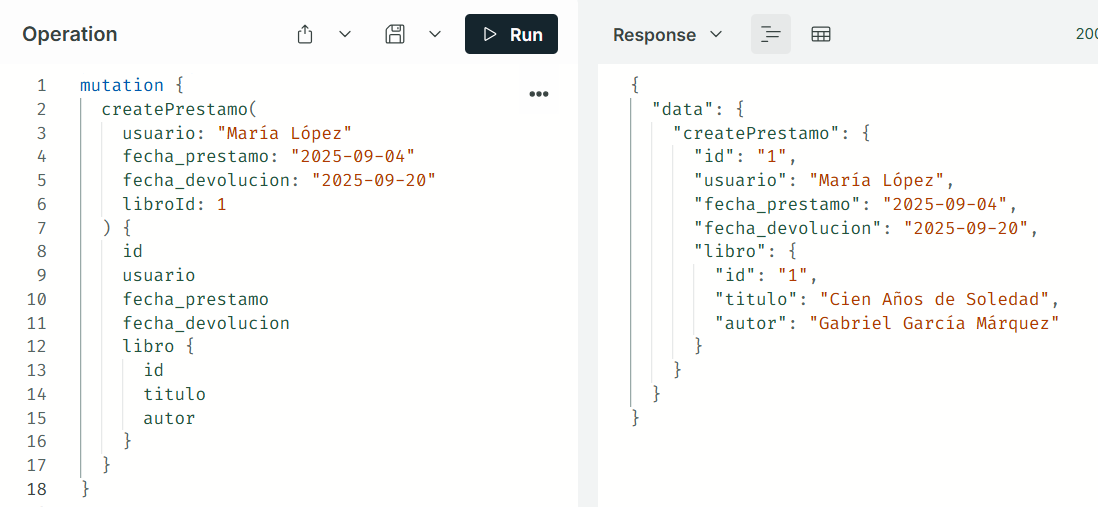
▪ usuario: María López

▪ fecha\_prestamo: 2025-09-04

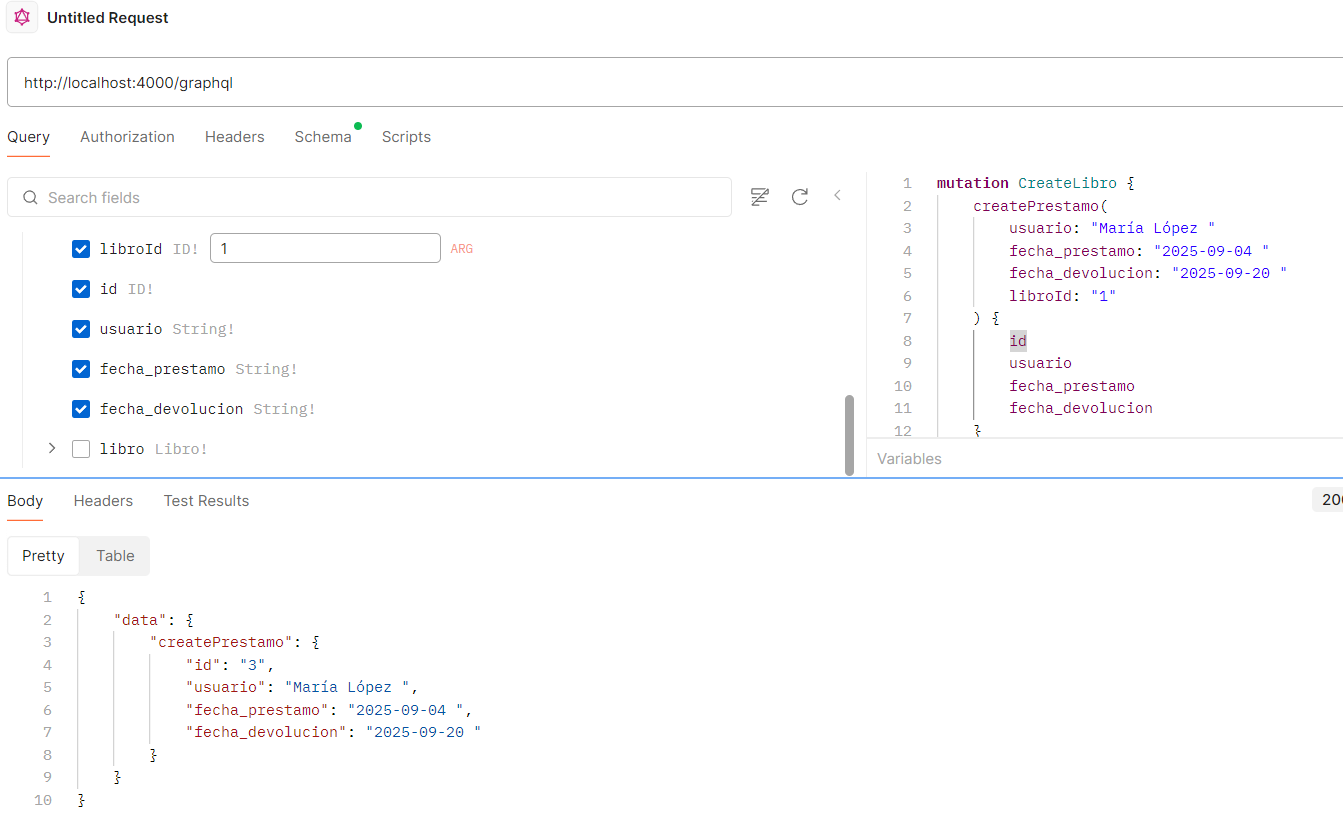
▪ fecha\_devolucion: 2025-09-20

▪ libroId: 1 (libro recién creado).

HERRAMIENTA:



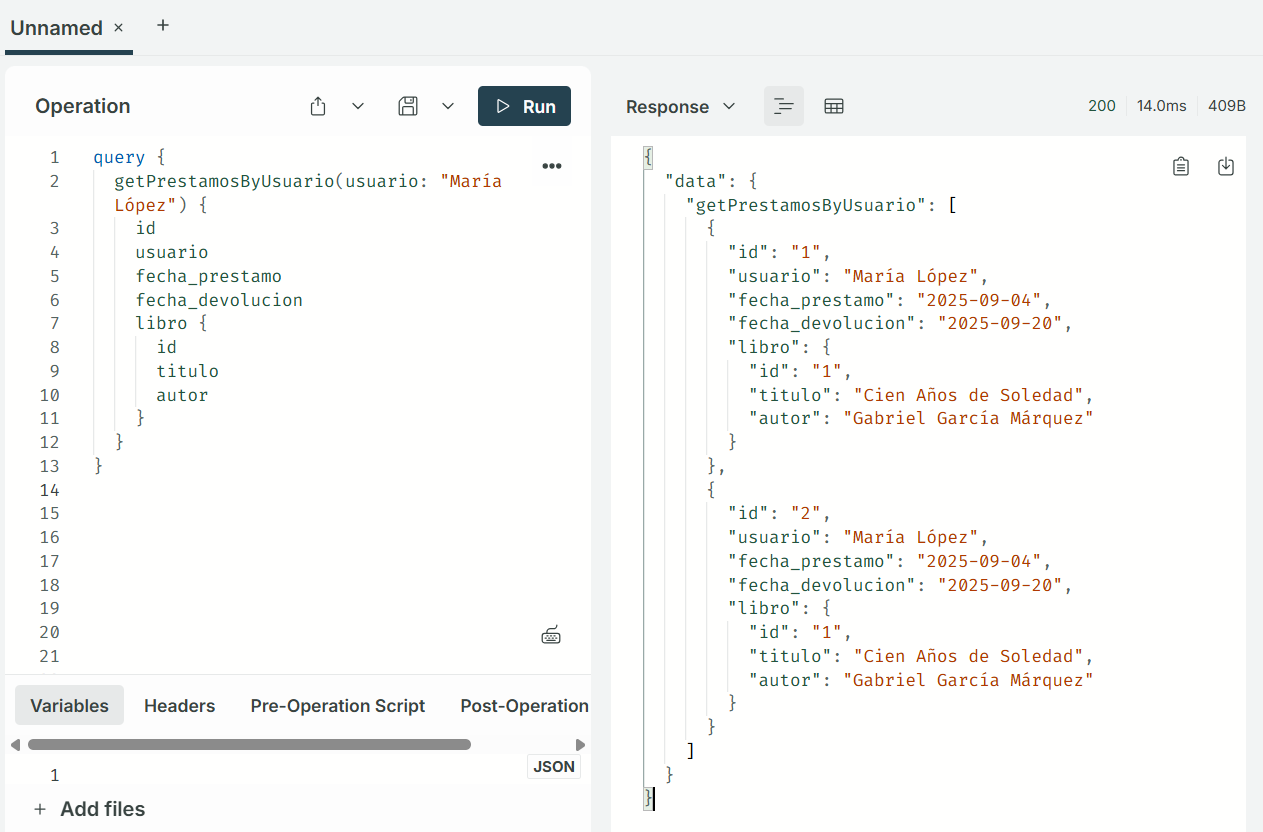
POSTMAN:



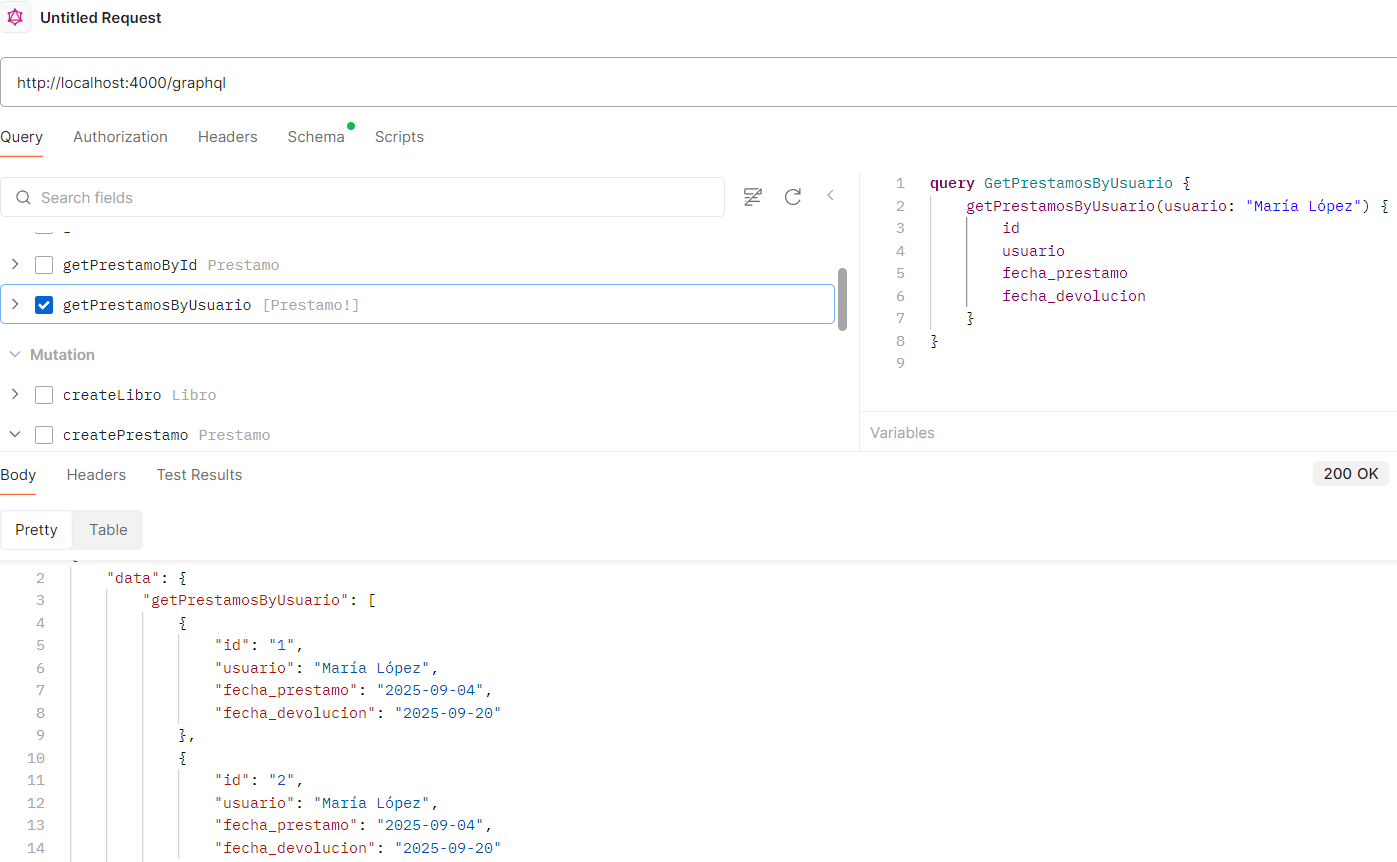
d. **Desafío extra**

o Crea una consulta que devuelva los **préstamos filtrados por nombre de usuario**.

HERRAMIENTA:



POSTMAN:



2. Sube a tu repositorio el código generado y en el documento Word el resultado de hacer las consultas tanto en la herramienta como en postman