Laboratorio 12:

GRAPHQL

1. Creamos un nuevo proyecto Spring Starter Project. Llámelo “PrjGraphQL” y use como paquetes/group com.cenfotec.graphql
2. En las dependencias agregamo Spring Web, Spring JPA, Lombok y H2
3. Una vez cargado el proyecto, en el POM agregamos las siguientes dependencias

```
<dependency>
    <groupId>com.graphql-java</groupId>
    <artifactId>graphql-spring-boot-starter</artifactId>
    <version>5.0.2</version>
</dependency>
<dependency>
    <groupId>com.graphql-java</groupId>
    <artifactId>graphql-java-tools</artifactId>
    <version>5.2.4</version>
</dependency>
<dependency>
    <groupId>com.graphql-java</groupId>
    <artifactId>graphiql-spring-boot-starter</artifactId>
    <version>5.0.2</version>
</dependency>
```

4. Creamos la clase Vehicle en el paquete com.cenfotec.graphql.entities

5. En dicha clase digitamos lo siguiente

```
@Data
@EqualsAndHashCode
@Entity
public class Vehicle implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Column(name = "ID", nullable = false)
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(name = "type", nullable = false)
    private String type;

    @Column(name = "model_code", nullable = false)
    private String modelCode;

    @Column(name = "brand_name")
    private String brandName;
```

```
    @Column(name = "launch_date")
    private LocalDate launchDate;

    private transient  String formattedDate;

    public String getFormattedDate() {
        return getLaunchDate().toString();
    }
}
```

6. Los import quedan así

```
import java.io.Serializable;
import java.time.LocalDate;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.Data;
import lombok.EqualsAndHashCode;
```

7. Ahora creamos un nuevo paquete com.cenfotec.graphql.repositories y en él creamos la clase VehicleRepository, que tendrá la siguiente declaración

```
@Repository
public interface VehicleRepository extends JpaRepository<Vehicle, Integer> {
}
```

8. Ahora creamos un nuevo paquete llamado com.cenfotec.graphql.services y en él creamos la clase VehicleService. **COLOCAMOS EN LA DECLARACIÓN DE DICHA CLASE LA ANOTACIÓN @Service**

9. En la nueva clase digitamos lo siguiente

```
@Autowired
VehicleRepository vehicleRepo;

public List<Vehicle> getAllVehicles(int count) {
      return
this.vehicleRepo.findAll().stream().limit(count).collect(Collectors.toList());

}

public Optional<Vehicle> getVehicle(int id) {
      return this.vehicleRepo.findById(id);
}
```

```
public Vehicle createVehicle(String type, String modelCode,
             String brandName, final String launchDate) {

      Vehicle vehicle = new Vehicle();
      vehicle.setType(type);
      vehicle.setModelCode(modelCode);
      vehicle.setBrandName(brandName);
      vehicle.setLaunchDate(LocalDate.parse(launchDate));
      return this.vehicleRepo.save(vehicle);
 }
```

10. Los imports quedan asi:

```
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;

import com.cenfotec.graphql.entities.Vehicle;
import com.cenfotec.graphql.repository.VehicleRepository;
```

11. Ahora en src/main/resources creamos un nuevo archivo (click derecho sobre dicho folder → New → File) y le llamamos vehicleql.graphqls

12. En el nuevo archivo digitamos:

```
type Vehicle {
      id: ID!,
      type: String,
      modelCode: String,
      brandName: String,
      launchDate: String
}

type Query {
      vehicles(count: Int):[Vehicle]
      vehicle(id: ID):Vehicle
}

type Mutation {
      createVehicle(type: String!, modelCode: String!, brandName: String, launchDate:
String):Vehicle
}
```

13. Creamos un nuevo paquete llamado com.cenfotec.graphql.query y en él creamos una nueva clase llamada VehicleQuery
14. Modificamos la declaración de la clase que quede asi:

```
@Component
public class VehicleQuery implements GraphQLQueryResolver {
```

…

15. Ahora dentro de la clase, escribimos lo siguiente:

```
@Autowired
 private VehicleService vehicleService;

 public List<Vehicle> getVehicles(int count) {
     return this.vehicleService.getAllVehicles(count);
 }

 public Optional<Vehicle> getVehicle(int id) {
     return this.vehicleService.getVehicle(id);
 }
```

16. Los imports quedan asi

```
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import com.cenfotec.graphql.entities.Vehicle;
import com.cenfotec.graphql.services.VehicleService;
import com.coxautodev.graphql.tools.GraphQLQueryResolver;
```

17. Ahora creamos un nuevo paquete com.cenfotec.graphql.mutation y en él creamos la clase VehicleMutation.
18. Modificamos la declaración de la clase para que sea así:

```
@Component
public class VehicleMutation implements GraphQLMutationResolver {
```

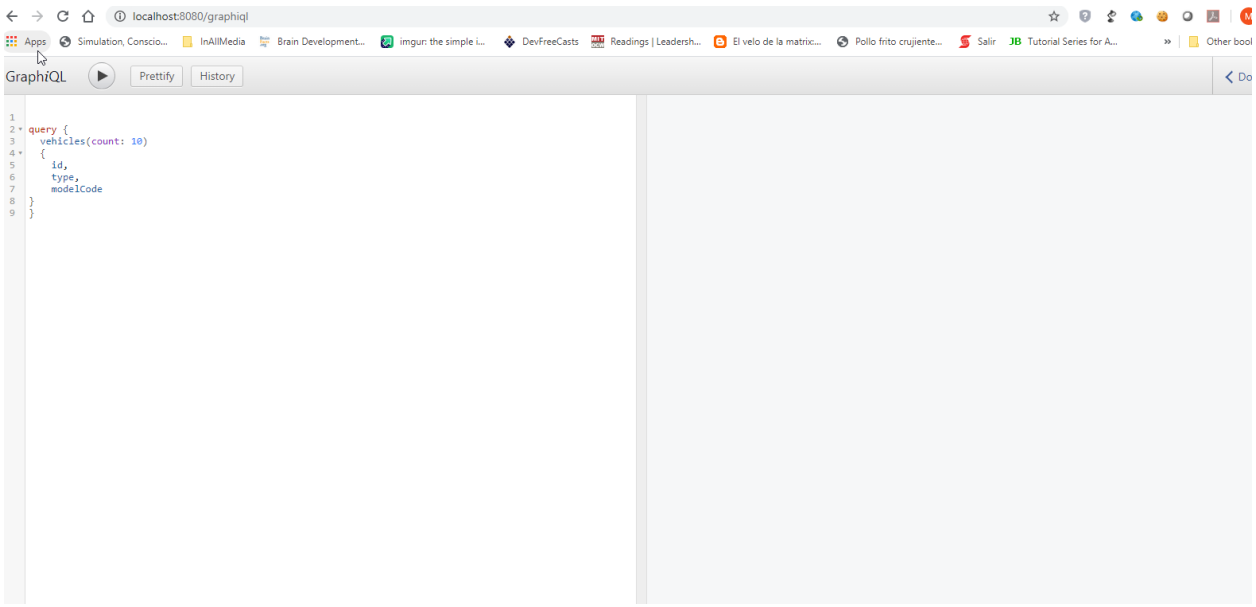
19. Dentro de la clase, escribimos:

```
@Autowired
private VehicleService vehicleService;

public Vehicle createVehicle(String type, String,
                    modelCode, String brandName,
                     String launchDate) {
   return this.vehicleService.createVehicle(type, modelCode,
brandName, launchDate);

}
```

20. Los imports quedan asi

```
import com.cenfotec.graphql.entities.Vehicle;
import com.cenfotec.graphql.services.VehicleService;
import com.coxautodev.graphql.tools.GraphQLMutationResolver;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import java.time.LocalDate;
```

21. Ahora, ponemos a correr la aplicación y abrimos localhost:8080/graphiql



22. En el lado izquierdo, escribimos:

    query {
    vehicles(count: 10)
    {
    id,
    type,
    modelCode
    }
    }

    Y le damos PRETTIFY. Vea la diferencia.
    Ahora le damos RUN

23. Borramos lo que está a la izquierda y escribimos

```
mutation {
 createVehicle(type: "ban", modelCode: "123queso", brandName: "toyota", launchDate: "2020-04-16")
 {
  id
 }
}
```

24. Veamos lo que retorna → NOTE EL VALOR DEL ID
25. Ahora repitamos el paso 22. Observe lo que se retorna.
26. Ahora mofiquemos el query eliminando la línea “modelCode”.