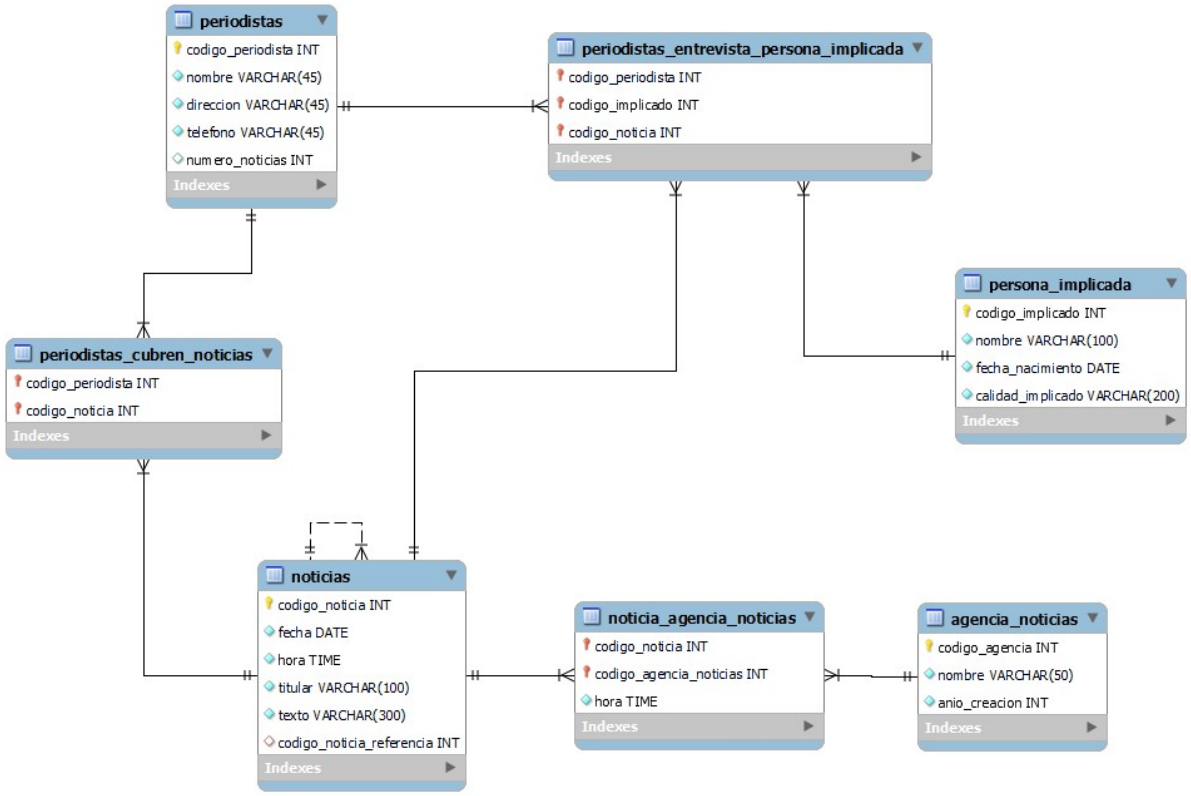
GUÍA PRÁCTICA DE CASO DE ESTUDIO:

Paso 1: Configurar la Base de Datos

1.1. Elegimos un sistema de gestión de bases de datos (por ejemplo, MySQL, PostgreSQL, Oracle) y se configura la base de datos, en nuestro caso de estudio se eligió MySQL.

Descripción del caso de estudio y modelo relacional:



Aplicativo solicitado: El periódico

Un periódico desea tener una base de datos para almacenar la información referente a los periodistas que trabajan el dicho periódico. Además, se almacenará información referente a las noticias que cubren los periodistas y las agencias de noticias que trabajan con el periódico.

De los periodistas se almacenará un código identificativo, nombre, dirección teléfono y el número de noticias que ha cubierto. De las noticias cubiertas por los periodistas, se almacenará un código, la fecha y hora de la noticia, el titular y el texto. Además, se almacenará que noticias están relacionadas entre sí. Hay que tener en cuenta que en una noticia puede haber varios implicados y que un implicado lo puede ser de distintas noticias. De estos implicados, se almacenará un código, nombre, fecha de nacimiento y calidad en la que aparece como implicado en cada noticia.

Cuando se produce una noticia, uno o varios periodistas la pueden cubrir, así como un periodista, por supuesto, puede cubrir varias noticias. A la hora de cubrir la noticia, los periodistas pueden hablar con los implicados en dicha noticia. Un periodista podrá hablar con varios implicados y un implicado ser preguntado por varios periodistas, eso sí, referente a distintas noticias en la que estén implicados. Se desea conocer qué periodista habla con cual implicado respecto a cuál noticia.

Por otro lado, de las agencias de noticias que trabajan con el periódico se desea almacenar un código de agencia, el nombre y el año de creación. Se desea almacenar qué agencia o agencias han dado cada noticia, siendo importante la hora en la que han dado la noticia para comprobar cual agencia es más rápida en dar una determinada noticia.:

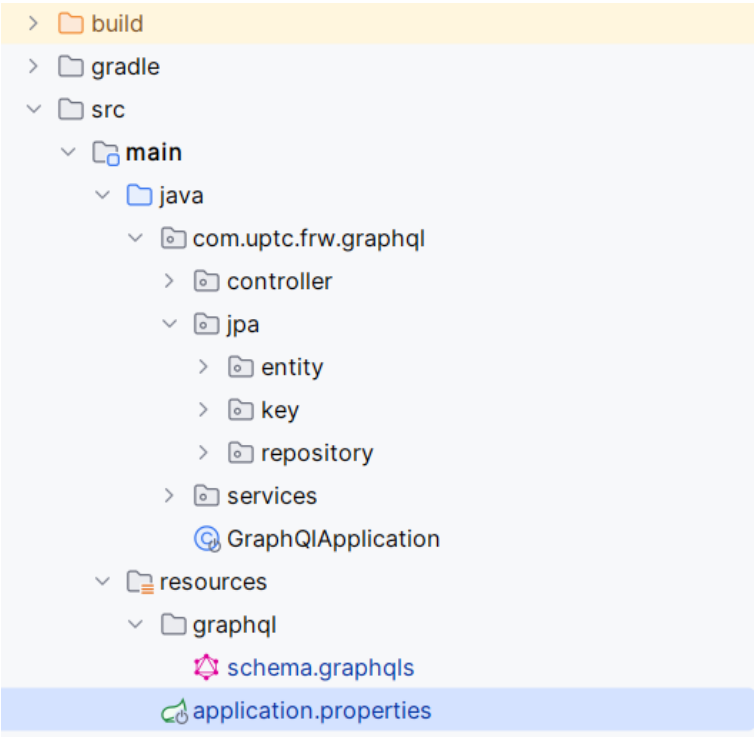
1.2. Creamos las tablas necesarias para almacenar información sobre periodistas, noticias, implicados y agencias de noticias. Asegurándonos de definir correctamente las relaciones entre las tablas:

```sql
create SCHEMA agencia_noticias;
USE agencia_noticias ;

-- -----------------------------------------------------
-- Table PERIODISTAS-----------
-- -----------------------------------------------------
CREATE TABLE PERIODISTAS (
  codigo_periodista INT NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(45) NOT NULL,
  direccion VARCHAR(45) NOT NULL,
  telefono VARCHAR(45) NOT NULL,
  numero_noticias INT NULL,
  PRIMARY KEY (codigo_periodista));


-- -----------------------------------------------------
-- Table NOTICIAS------------------
-- -----------------------------------------------------
CREATE TABLE NOTICIAS (
  codigo_noticia INT NOT NULL AUTO_INCREMENT,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  titular VARCHAR(100) NOT NULL,
  texto VARCHAR(300) NOT NULL,
  codigo_noticia_referencia INT NULL,
  PRIMARY KEY (codigo_noticia),
  CONSTRAINT fk_NOTICIAS_REFERENCIA
    FOREIGN KEY (codigo_noticia_referencia)
    REFERENCES NOTICIAS (codigo_noticia));
```

```sql
-- -----------------------------------------------------
-- Table PERSONA_IMPLICADA----------------
-- -----------------------------------------------------
CREATE TABLE PERSONA_IMPLICADA (
  codigo_implicado INT NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  calidad_implicado VARCHAR(200) NOT NULL,
  PRIMARY KEY (codigo_implicado));


-- -----------------------------------------------------
-- Table AGENCIA_NOTICIAS-----------
-- -----------------------------------------------------
CREATE TABLE AGENCIA_NOTICIAS (
  codigo_agencia INT NOT NULL AUTO_INCREMENT,
  nombre VARCHAR(50) NOT NULL,
  anio_creacion INT NOT NULL,
  PRIMARY KEY (codigo_agencia));


-- -----------------------------------------------------
-- Table PERIODISTAS_CUBREN_NOTICIAS------------
-- -----------------------------------------------------
CREATE TABLE PERIODISTAS_CUBREN_NOTICIAS (
  codigo_periodista INT NOT NULL,
  codigo_noticia  INT NOT NULL,
  PRIMARY KEY (codigo_periodista, codigo_noticia),
  CONSTRAINT fk_PERIODISTAS_NOTICIAS
    FOREIGN KEY (codigo_periodista)
    REFERENCES PERIODISTAS (codigo_periodista),
  CONSTRAINT fk_NOTICIAS_PERIODISTAS
    FOREIGN KEY (codigo_noticia )
    REFERENCES NOTICIAS (codigo_noticia ));
```

```sql
-- -----------------------------------------------------
-- Table NOTICIA_AGENCIA_NOTICIAS------
-- -----------------------------------------------------
CREATE TABLE NOTICIA_AGENCIA_NOTICIAS (
  codigo_noticia  INT NOT NULL,
  codigo_agencia_noticias INT NOT NULL,
  hora TIME NOT NULL,
  PRIMARY KEY (codigo_noticia , codigo_agencia_noticias),
  CONSTRAINT fk_NOTICIAS_AGENCIA_NOTICIAS
    FOREIGN KEY (codigo_noticia)
    REFERENCES NOTICIAS (codigo_noticia ),
  CONSTRAINT fk_AGENCIA_NOTICIAS_NOTICIAS
    FOREIGN KEY (codigo_agencia_noticias)
    REFERENCES AGENCIA_NOTICIAS (codigo_agencia));

-- -----------------------------------------------------
-- Table PERIODISTAS_ENTREVISTA_PERSONA_IMPLICADA-------
-- -----------------------------------------------------
CREATE TABLE PERIODISTAS_ENTREVISTA_PERSONA_IMPLICADA (
  codigo_periodista INT NOT NULL,
  codigo_implicado INT NOT NULL,
  codigo_noticia  INT NOT NULL,
  PRIMARY KEY (codigo_periodista,codigo_implicado, codigo_noticia),
  CONSTRAINT fk_PERIODISTAS_NOTICIAS_PERSONA_IMPLICADA
    FOREIGN KEY (codigo_periodista)
    REFERENCES PERIODISTAS (codigo_periodista),
  CONSTRAINT fk_PERSONA_IMPLICADA_PERIODISTAS_NOTICIA
    FOREIGN KEY (codigo_implicado)
    REFERENCES PERSONA_IMPLICADA (codigo_implicado),
  CONSTRAINT fk_NOTICIAS_PERIODISTAS_PERSONA_IMPLICADA
    FOREIGN KEY (codigo_noticia)
    REFERENCES NOTICIAS (codigo_noticia));
```

Se maneja la siguiente estructura del Proyecto:



Paso 2: Configurar la Conexión a la Base de Datos

2.1. Utilizamos JDBC o un marco de persistencia de Java, como Hibernate, para configurar la conexión a la base de datos.

Se configura el archivo *application.properties,* con la conexión a la Base de Datos

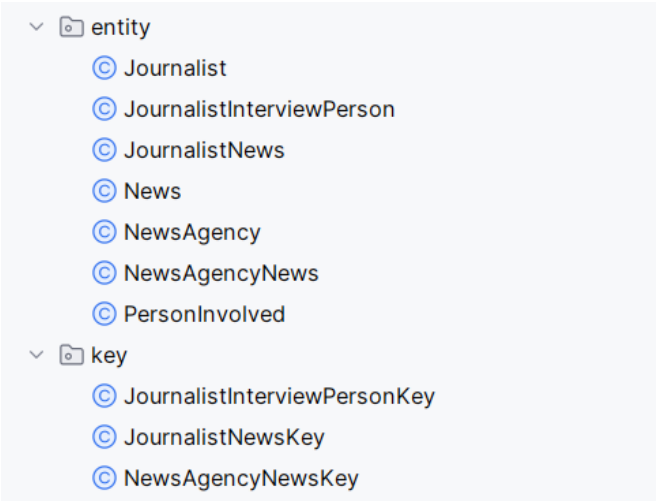
```
server.port=8089
spring.datasource.url=jdbc:mysql://localhost:3306/agencia_noticias
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.graphql.graphiql.enabled=true
```

En el archivo *buil.gradle,* **se agregan las siguientes dependencias**

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-graphql'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'mysql:mysql-connector-java:8.0.33'
    implementation("com.graphql-java-kickstart:graphql-spring-boot-starter:5.10.0")
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

Paso 3: Crear las Entidades

3.1. Creamos clases Java para representar las entidades del problema, como Periodista, Noticia, Implicado, Agencia, etc, donde estas clases reflejen la estructura de las tablas de la base de datos.



```java
@Entity
@Table(name="PERIODISTAS")
public class Journalist {
    @Id
    @Column(name="CODIGO_PERIODISTA")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    4 usages
    @Column(name="NOMBRE")
    private String name;
    4 usages
    @Column(name="DIRECCION")
    private String address;
    4 usages
    @Column(name="TELEFONO")
    private String phone;
    4 usages
    @Column(name="NUMERO_NOTICIAS")
    private int numberNews;
    2 usages
    @OneToMany(mappedBy = "journalist")
    private List<JournalistNews> journalistNews;
    2 usages
    @JsonIgnore
    @OneToMany(mappedBy = "journalist")
    private List<JournalistInterviewPerson> journalistInterviewPeople;
```

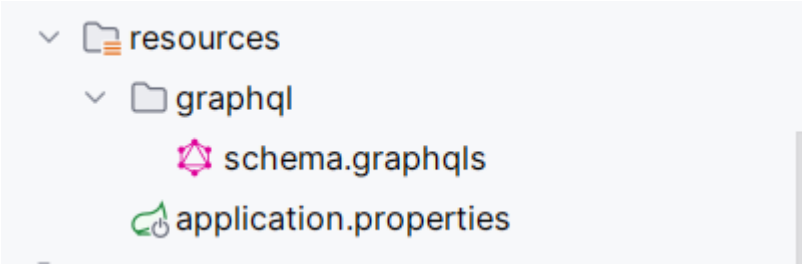
Paso 4: Implementar Operaciones CRUD

4.1. Creamos clases Repository para cada entidad que proporcionen métodos para crear, leer, actualizar y eliminar registros en la base de datos. Estos métodos utilizarán como persistencia Java Persistence API (JPA).

Paso 5: Integrar GraphQL

5.1. Agregamos la dependencia de GraphQL-Java a nuestro proyecto.

5.2. Definimos el esquema GraphQL en un archivo .graphqls con tipos y consultas (Query, Mutation) que reflejen las entidades y operaciones CRUD.



Estructura de los Types:

```
type Journalist{
    id: Int,
    name: String!,
    address: String!,
    phone: String!,
    numberNews: Int
    journalistInterviewPeople:[JournalistInterviewPerson],
    journalistNews: [JournalistNews]
}
type News{
    id:Int,
    dateNew: String! ,
    timeNew: String! ,
    headline: String! ,
    text: String!,
    newsReference: News
    newsAgencyNewsList:[NewsAgencyNews]
    journalistInterviewPeople:[JournalistInterviewPerson],
    journalistNews:[JournalistNews]
}

type NewsAgency{
    id: Int,
    name: String,
    year:Int
    newsAgencyNewsList:[NewsAgencyNews]
}
```

```
type NewsAgencyNews{
    hour:String!
    newsAgency:NewsAgency,
    news:News
}
type PersonInvolved{
    id:Int,
    name:String!,
    birthday:String!,
    qualityInvolved:String!
    journalistInterviewPeople:[JournalistInterviewPerson]
}
type JournalistInterviewPerson{
    journalist:Journalist,
    involved:PersonInvolved
    news:News
}
type JournalistNews{
    journalist:Journalist,
    news:News
}
```

Estructuras de las Query

```
type Query{
    allJournalists:[Journalist]
    getJournalistById(id:Int!):Journalist

    getAllJournalistNew:[JournalistNews]
    getJournalistNewsById(idJournalist:Int!,idNew:Int!):JournalistNews

    getAllNews:[News]
    getNewsById(id:Int!):News

    getAllNewsAgency:[NewsAgency]
    getNewsAgencyById(id:Int!):NewsAgency

    getAllPersonInvolved:[PersonInvolved]
    getPersonInvolvedById(id:Int!):PersonInvolved

    getAllJournalistInterviewPerson:[JournalistInterviewPerson]
    getJournalistInterviewPersonById(idJournalist:Int!,idNew:Int!, idInvolved:Int!):JournalistInterviewPerson

    getAllNewsAgencyNews:[NewsAgencyNews]
    getNewsAgencyNewsById(idJournalist:Int!,idNew:Int!):NewsAgencyNews

}
```

Estructuras de las Mutations

```
type Mutation{
    addJournalist(name: String!, address: String!, phone: String!, numberNews: Int) : Journalist
    updateNameJournalistById(id:Int!,name: String!):Journalist
    deleteJournalistById(id:Int!):String

    createJournalistNews(idJournalist:Int!,idNew:Int!):JournalistNews
    deleteJournalistNews(idJournalist:Int!,idNew:Int!):String

    createNews(dateNew:String!, timeNew:String!, headline:String!, text:String!):News
    addNewsReferences(id:Int!,idNewReference:Int!):News
    updateNewsHeadline(id:Int!, headLine:String!):News
    deleteNewsById(id:Int!):String

    createNewsAgency(name:String!, year:Int!):NewsAgency
    updateNameNewsAgency(id:Int!,name:String!):NewsAgency
    deleteNewsAgencyById(id:Int!):String

    createPersonInvolved(name:String!,birthday:String!,qualityInvolved:String!):PersonInvolved
    updateNamePersonInvolved(id:Int!,name:String!):PersonInvolved
    deletePersonInvolvedById(id:Int!):String

    createJournalistInterviewPerson(idJournalist:Int!,idNew:Int!,idInvolved:Int!):JournalistInterviewPerson
    deleteJournalistInterviewPerson(idJournalist:Int!,idNew:Int!,idInvolved:Int!):String

    createNewsAgencyNews(idnewsAgency:Int!,idNew:Int!,hour:String!):NewsAgencyNews
    deleteNewsAgencyNews(idJournalist:Int!,idNew:Int):String

}
```

Paso 6: Pruebas y Documentación

6.1. Probamos varias veces la aplicación para asegurarnos de que las operaciones CRUD y las consultas GraphQL funcionaran correctamente.

6.2. Documentamos partes del proyecto GraphQL para que otros desarrolladores puedan comprender y utilizar la aplicación.

Estructuras ejemplo de GraphQL para el caso de estudio:

```
--------------------News

mutation{
  createNews(
    dateNew:"2023-10-01",
    timeNew:"14:30:00",
    headline:"Educación en Colombia",
    text:"Educación lidera sectores con mayores recursos aprobados para el Presupuesto 2024"

  ) {
     id, dateNew,timeNew,headline,text, newsReference {
       id
     }
  }

}

query{
  getAllNews{
     id, dateNew,timeNew,headline,text, newsReference {
       id
     }
  }
}


query{
   getNewsById(id:5){
     id, dateNew,timeNew,headline,text, newsReference{
       id
     }
  }
}

mutation{
updateNewsHeadline(id:5,  headLine: "Actualización del Encabezado"){
 id, dateNew,timeNew,headline,text, newsReference{
       id
     }
```

```
mutation{
  addNewsReferences(id:2, idNewReference:4){
  id, dateNew,timeNew,headline,text,newsReference{
    id, dateNew,timeNew,headline,text
  }
}
}

mutation{
  deleteNewsById(id:1)
}

-------------------------NewsAgency

mutation{
   createNewsAgency(name:"Caracol", year:1900){
   id,name,year
  }
}

query{
  getAllNewsAgency {
    id,name,year
  }
}

query{
  getNewsAgencyById(id:2) {
    id,name,year
  }
}

mutation{
  updateNameNewsAgency(id:2,name: "RCN"){
    id,name,year
  }
}

mutation{
  deleteNewsAgencyById(id:3)
}
```

```
-----------------------------------------------------PersonInvolved


mutation{
   createPersonInvolved(name:"Carmen",birthday:"1980-01-05", qualityInvolved:"Estudiante Universitario"){
    id,name,birthday,qualityInvolved
  }
}

query{
  getAllPersonInvolved {
    id,name,birthday,qualityInvolved
  }
}

query{
  getPersonInvolvedById(id:1) {
     id,birthday,qualityInvolved
  }
}

mutation{
  updateNamePersonInvolved(id:5,name: "Estudiante 5"){
      id,name,birthday,qualityInvolved
  }
}

mutation{
  deletePersonInvolvedById(id:3)
}
```

```
-----------------------------------------------------JournalistNews
query{
  getAllJournalistNew {
    journalist{name},
    news{headline}
  }
}

query{
  getJournalistNewsById(idJournalist:2,idNew:3) {
    journalist{name},
    news{headline}
  }
}



mutation{
   createJournalistNews(idJournalist:2,idNew:4){
    journalist{name},
    news{headline}
  }
}


mutation{
  deleteJournalistNews(idJournalist:2,idNew:2)
}
```

```
-------------------------------------------------- JournalistInterviewPerson

query{
  getJournalistInterviewPersonById(idJournalist:2,idNew:3,idInvolved:1) {
    journalist{name},
    news{headline}
  }
}

mutation{
   createJournalistInterviewPerson(idJournalist:2,idNew:3,idInvolved:1){
    journalist{name},
    involved{name},
    news{headline}
  }
}


query{
  getAllJournalistInterviewPerson {
    journalist{name},
    involved{name},
    news{headline}
  }
}

mutation{
  deleteJournalistInterviewPerson(idJournalist:2,idNew:3,idInvolved:1)
}
```