**Guía Paso a Paso para Crear una Base de Datos en PostgreSQL**

Esta guía paso a paso te ayudará a crear una base de datos para la "Vuelta a Colombia" utilizando PostgreSQL. A continuación, se describen cada uno de los pasos junto con fragmentos del script SQL proporcionado.

**Paso 1: Creación de la Base de Datos**

El primer paso es crear la base de datos donde almacenaremos todas las tablas y datos relacionados con la "Vuelta a Colombia".

-- Crear la base de datos si no existe

CREATE DATABASE vueltaaColombia;

**Paso 2: Creación de Secuencias**

Las secuencias son objetos de base de datos vitales que facilitan la generación de números únicos, generalmente utilizados para crear identificadores primarios. En este paso, estableceremos secuencias para los IDs autoincrementables que se utilizarán en varias tablas, garantizando así la integridad y la unicidad de los datos.

-- Crear secuencias para los IDs autoincrementables

CREATE SEQUENCE edition\_id\_seq START WITH 1 INCREMENT BY 1 MAXVALUE 100000 NO CYCLE;

CREATE SEQUENCE sponsor\_nit\_seq START WITH 1 INCREMENT BY 1 MAXVALUE 100000 NO CYCLE;

**Paso 3: Creación de Tablas**

El siguiente paso es la creación de tablas bien estructuradas que almacenarán diferentes tipos de datos. Cada tabla debe diseñarse con una consideración cuidadosa de los tipos de datos, restricciones y relaciones para garantizar una recuperación de datos eficiente y una integridad de datos sólida.

**Tabla Edition**

Esta tabla servirá como el núcleo para almacenar información detallada sobre las distintas ediciones de la "Vuelta a Colombia", incluyendo detalles como el año, la fecha de inicio y la fecha de fin.

-- Crear tabla Edition

CREATE TABLE Edition (

id INT PRIMARY KEY DEFAULT nextval('edition\_id\_seq'),

year INT,

start\_date DATE,

end\_date DATE

);

**Tabla Sponsor**

La tabla Sponsor mantendrá un registro meticuloso de los patrocinadores del evento, almacenando detalles como un NIT único y el nombre del patrocinador, además de información adicional que puede ser útil para la gestión de relaciones con los patrocinadores.

-- Crear tabla Sponsor

CREATE TABLE Sponsor (

NIT INT PRIMARY KEY DEFAULT nextval('sponsor\_nit\_seq'),

name VARCHAR(255)

);

**Relación entre Atletas y Equipos**

Esta relación se materializa en la tabla Athlete\_Team, que almacena las referencias a las tablas Edition\_Team y Athlete mediante claves foráneas. Esta tabla representa la relación dinámica entre los atletas y los equipos en una edición específica del evento.

-- Crear tabla Athlete\_Team

CREATE TABLE Athlete\_Team (

edition\_team\_id INT REFERENCES Edition\_Team(id),

athlete\_id INT REFERENCES Athlete(id),

PRIMARY KEY (edition\_team\_id, athlete\_id)

);

**Relación entre Etapas y Ediciones**

La tabla Stage es fundamental para registrar información sobre las diferentes etapas de una edición específica del evento. Se establece una relación con la tabla Edition a través de una clave foránea, facilitando la vinculación de cada etapa con una edición específica y permitiendo análisis detallados de cada etapa.

-- Crear tabla Stage

CREATE TABLE Stage (

id INT PRIMARY KEY DEFAULT nextval('stage\_id\_seq'),

edition\_id INT REFERENCES Edition(id),

origin VARCHAR(255),

destination VARCHAR(255),

length DECIMAL,

stage\_type VARCHAR(255)

);

**Relación entre Podio y Etapas/Atletas**

La tabla Podium es crucial para capturar los resultados de cada etapa, registrando detalles sobre los atletas que alcanzaron el podio y el tiempo que lograron. Se establecen relaciones con las tablas Stage y Athlete a través de claves foráneas, creando un registro detallado de cada resultado del podio.

-- Crear tabla Podium

CREATE TABLE Podium (

stage\_id INT REFERENCES Stage(id),

athlete\_id INT REFERENCES Athlete(id),

position INT,

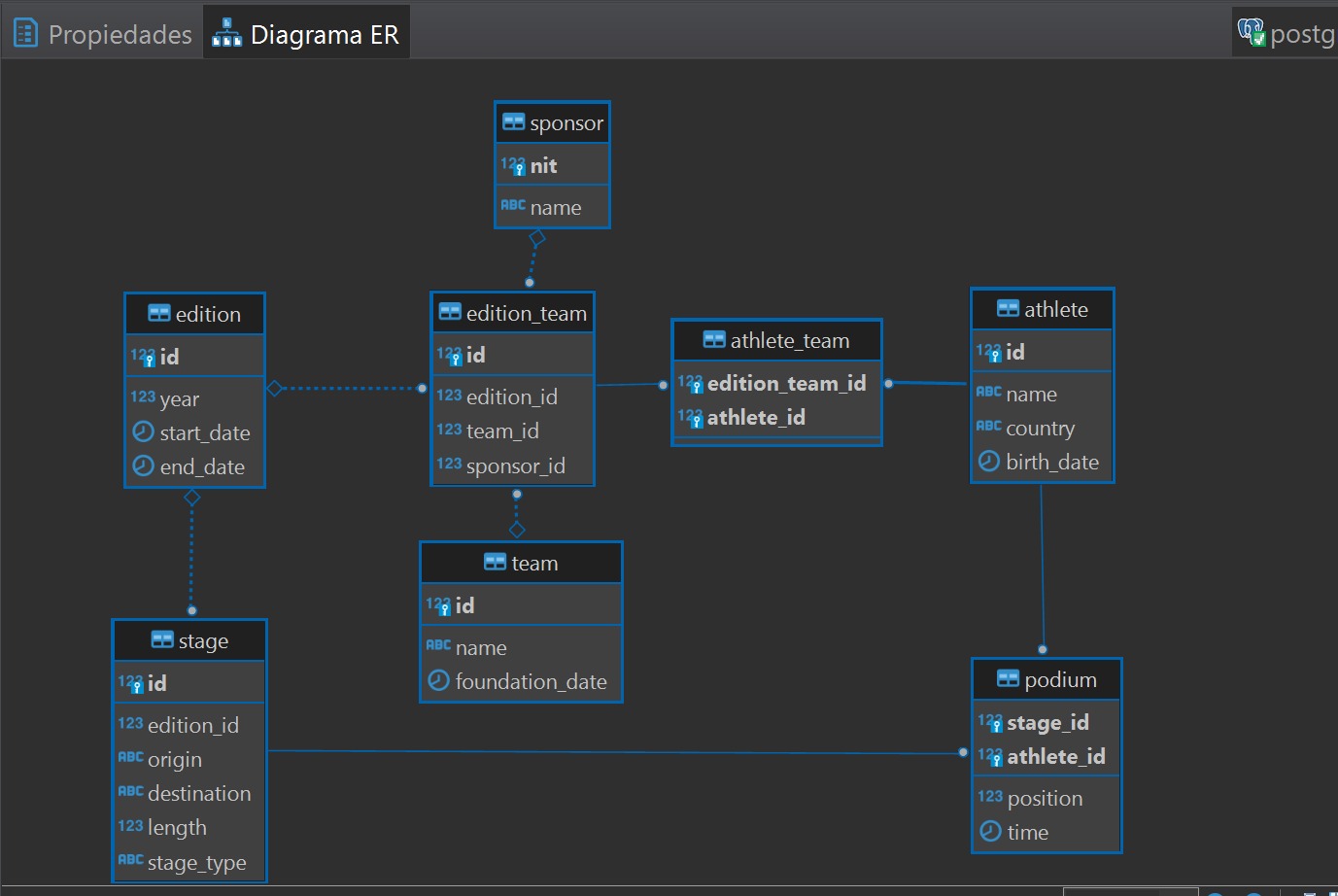
time TIME,

PRIMARY KEY (stage\_id, athlete\_id)

);

**Paso 5: Finalización**

Una vez que hayas ejecutado todos los fragmentos de script anteriores en tu sistema de gestión de bases de datos PostgreSQL, habrás configurado con éxito la base de datos para la "Vuelta a Colombia".



**Guía para Integrar ElasticSearch con Spring Boot**

**1. Configuración de Dependencias**

Para empezar, es esencial configurar las dependencias en el archivo **build.gradle**. Asegúrese de incluir las siguientes:

Gradle code

implementation 'org.springframework.boot:spring-boot-starter-data-jpa' implementation 'org.springframework.boot:spring-boot-starter-web' testImplementation 'org.springframework.boot:spring-boot-starter-test' compileOnly 'org.projectlombok:lombok' annotationProcessor 'org.projectlombok:lombok' implementation 'net.logstash.logback:logstash-logback-encoder:7.2'

**2. Ajustes en Logstash**

Para que Logstash funcione correctamente con ElasticSearch, configure el archivo **logstash/config/logstash.yml** de la siguiente manera:

Yml code

http.host: "0.0.0.0" path.config: /usr/share/logstash/pipeline xpack.monitoring.elasticsearch.hosts: [ "localhost:9200" ]

**3. Despliegue con Docker**

Para facilitar el despliegue y conexión entre ElasticSearch, Kibana y Logstash, utilice Docker. A continuación, se muestra una estructura básica del archivo **docker-compose.yaml**:

* **ElasticSearch**: Motor de búsqueda y análisis.
* **Kibana**: Herramienta de visualización para ElasticSearch.
* **Logstash**: Herramienta de procesamiento que envía datos a ElasticSearch.

**Revisar Anexos docker-compose.yaml.**

**4. Implementación de Controladores en Spring Boot**

Dentro del paquete **com.uptc.VueltaColombia.controller**, se ha definido el HomeController.java con varios endpoints que demuestran la funcionalidad de registro:

**/:** Página de inicio.

**/logs:** Visualización de registros.

**/warnings:** Página de advertencias.

**/errors:** Página de errores.

Estos endpoints generan registros con diferentes niveles (INFO, WARN, ERROR) que se canalizan hacia ElasticSearch mediante Logstash.

**Revisar Anexos com.uptc.VueltaColombia.controller.**

**5. Ajustes en Logback**

Para que Spring Boot pueda enviar registros a Logstash, es crucial configurar Logback. Agregue la configuración pertinente en **src/main/resources/logback-spring.xml**:

src/main/resources/logback-spring.xml

<?xml version="1.0" encoding="UTF-8" ?>

<configuration>

<include resource="org/springframework/boot/logging/logback/base.xml"/>

<appender name="logstash" class="net.logstash.logback.appender.LogstashTcpSocketAppender">

<destination>localhost:5000</destination>

<encoder class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">

<providers>

<mdc />

<context />

<logLevel />

<loggerName />

<pattern>

<pattern>

{

"app": "test-log"

}

</pattern>

</pattern>

<threadName />

<message />

<logstashMarkers />

<stackTrace />

</providers>

</encoder>

</appender>

<root level="info">

<appender-ref ref="logstash" />

</root>

</configuration>