

CURSO : Desarrollo de Aplicaciones Web I (0265)
PROFESOR : César Enrique Santos Torres
CICLO : Quinto
SECCIÓN : 28257
GRUPO : 2024331933
FECHA : 23/11/2024 10:00am
DURACIÓN : 2 horas

NOTA

ALUMNO (A) : VASQUEZ VASQUEZ

CASO DE LABORATORIO 1 (CL1)

Consideraciones generales:

- El laboratorio consta de 4 partes, cada parte tiene una secuencia de pasos las cuales deberá ir acompañada (De forma obligatoria) de capturas de pantalla de lo implementado.
- Sólo debe subir este documento, con sus evidencias y respuestas en él. El código fuente de ambos proyectos debe ser subido a Github (Adjuntar links del repositorio). No se aceptará código zipeado.
- El nombre del presente archivo deberá tener la siguiente estructura: "DAWI-APELLIDOS-NOMBRES.pdf".

LOGRO DE LA EVALUACIÓN:

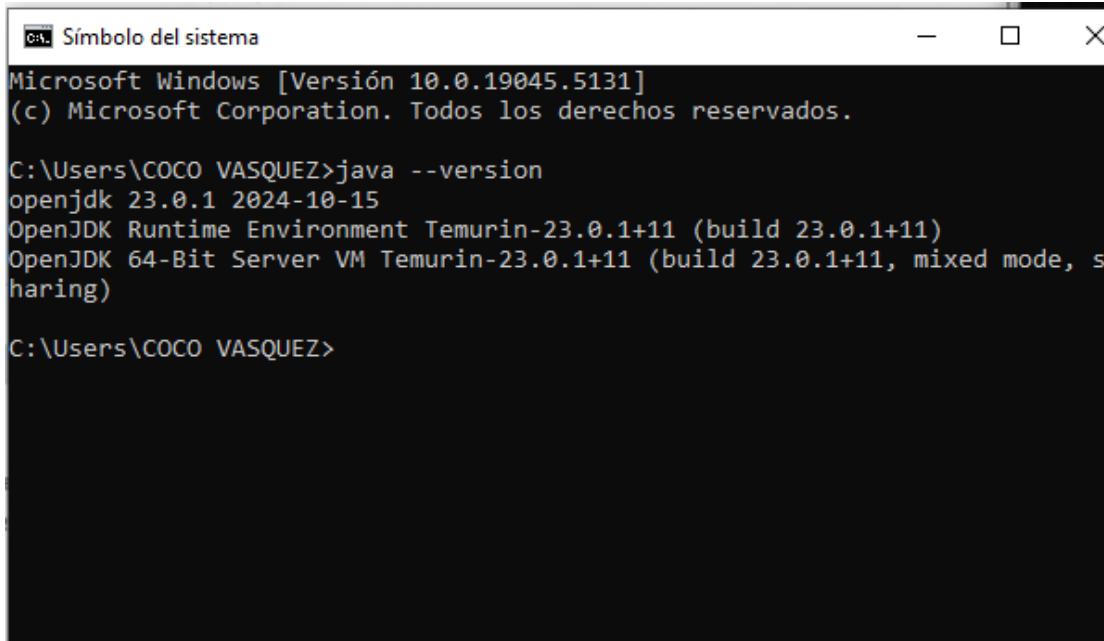
Al término de la evaluación, el alumno implementa las operaciones de mantenimiento sobre una entidad utilizando Java Persistence API.

CONSOLIDADO

Pregunta	Puntaje		Llenar solo en caso de Recalificación justificada	
	Máximo	Obtenido	Sustento	Puntaje
1	5			
2	5			
3	5			
4	5			
Total	20			
Nota Recalificada				

Parte 01 Configuración básica (25%)

- Descargar JDK versión 23 de <https://adoptium.net/es/temurin/releases/>
- Configurar variable de entorno JAVA_HOME y Path
- Validar configuración Java con los siguientes comandos (Use cmd):
 - java -version
 - echo %JAVA_HOME%

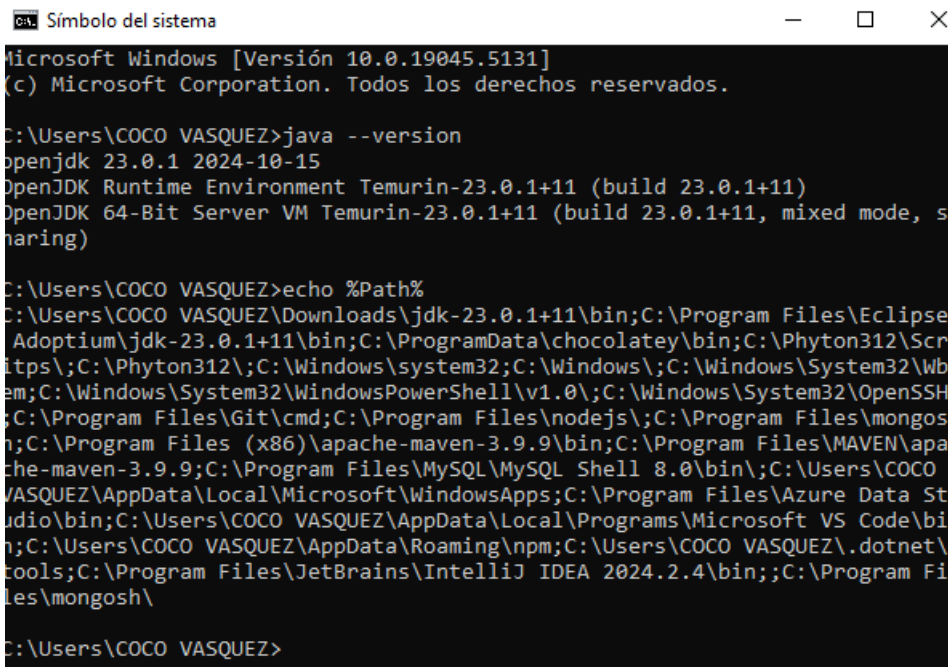


```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\COCO VASQUEZ>java --version
openjdk 23.0.1 2024-10-15
OpenJDK Runtime Environment Temurin-23.0.1+11 (build 23.0.1+11)
OpenJDK 64-Bit Server VM Temurin-23.0.1+11 (build 23.0.1+11, mixed mode, sharing)

C:\Users\COCO VASQUEZ>
```

- echo %Path%



```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\COCO VASQUEZ>java --version
openjdk 23.0.1 2024-10-15
OpenJDK Runtime Environment Temurin-23.0.1+11 (build 23.0.1+11)
OpenJDK 64-Bit Server VM Temurin-23.0.1+11 (build 23.0.1+11, mixed mode, sharing)

C:\Users\COCO VASQUEZ>echo %Path%
C:\Users\COCO VASQUEZ\Downloads\jdk-23.0.1+11\bin;C:\Program Files\Eclipse
\Adoptium\jdk-23.0.1+11\bin;C:\ProgramData\chocolatey\bin;C:\Phyton312\Scr
ipts\;C:\Phyton312\;C:\Windows\system32;C:\Windows\;C:\Windows\System32\Wb
em;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH
;C:\Program Files\Git\cmd;C:\Program Files\nodejs\;C:\Program Files\mongos
h;C:\Program Files (x86)\apache-maven-3.9.9\bin;C:\Program Files\MAVEN\apa
che-maven-3.9.9;C:\Program Files\MySQL\MySQL Shell 8.0\bin\;C:\Users\COCO
VASQUEZ\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Azure Data St
udio\bin;C:\Users\COCO VASQUEZ\AppData\Local\Programs\Microsoft VS Code\bi
n;C:\Users\COCO VASQUEZ\AppData\Roaming\npm;C:\Users\COCO VASQUEZ\.dotnet\
tools;C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.4\bin;;C:\Program Fi
les\mongosh\

C:\Users\COCO VASQUEZ>
```

- Conectar al servidor MySQL usando la terminal (Use cmd):
 - Use el comando: mysql -u root -p

```
Simbolo del sistema - mysql -u root -p

Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\COCO VASQUEZ>mysql --version
mysql Ver 9.1.0 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Users\COCO VASQUEZ>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- Restaurar bd “world” de <https://downloads.mysql.com/docs/world-db.zip> (Use cmd):
 - Use el comando: source <ruta-archivo-world.sql>;

```
Simbolo del sistema - mysql -u root -p

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)


Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

- Usar bd “world” y hacer un select de los primeros 20 registros de la tabla “city” (Use cmd).

 Símbolo del sistema - mysql -u root -p

```
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.


C:\Users\COCO VASQUEZ>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use world
Database changed
mysql> _
```

 Símbolo del sistema - mysql -u root -p

```
mysql> use world
Database changed
mysql> SELECT * FROM city LIMIT 20;
```

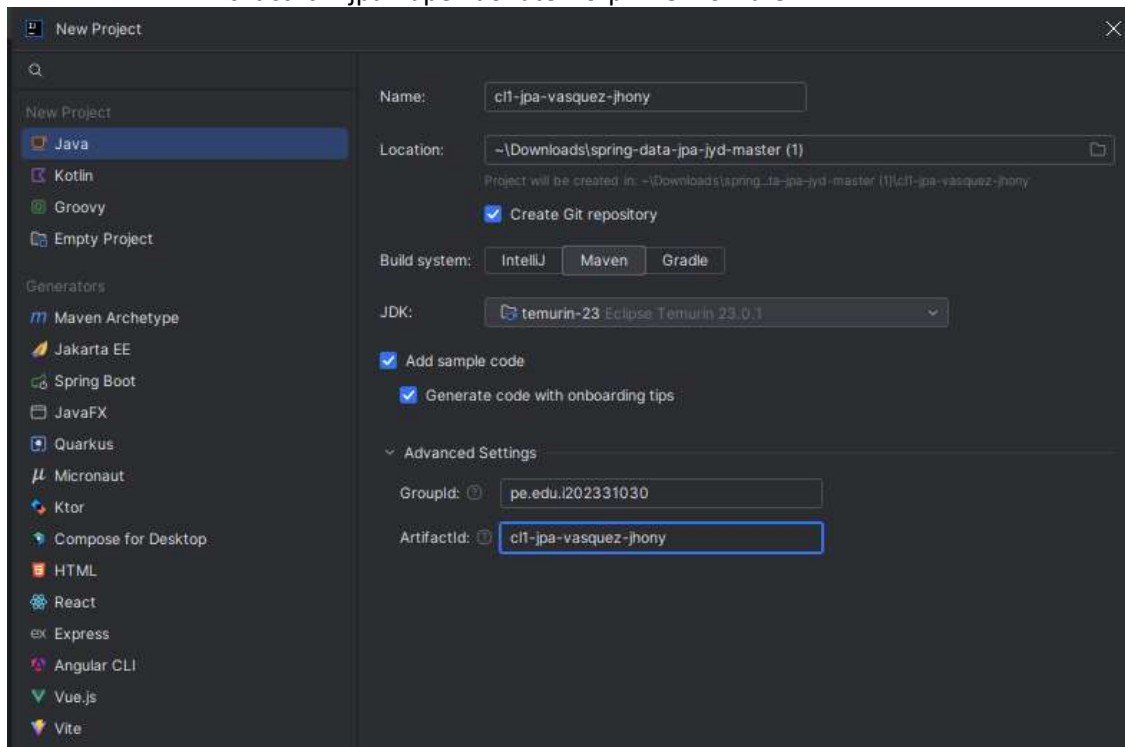
ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020
19	Zaanstad	NLD	Noord-Holland	135621
20	's-Hertogenbosch	NLD	Noord-Brabant	129170

```
20 rows in set (0.01 sec)

mysql>
```

Parte 02 Proyecto JPA-Hibernate (25%)

- Crear un proyecto JPA-Hibernate desde IntelliJ Idea con las siguientes características:
 - **Name:** cl1-jpa-<apellidoPaterno-primerNombre> (Use minúsculas y guión "-")
 - **Location:** Seleccione un directorio con permisos de lectura y escritura
 - **Create Git repository:** Check
 - **Build system:** Maven
 - **JDK:** Eclipse Temurin-23
 - **Advanced Settings:**
 - **GroupId:** pe.edu.<codigoEstudiante>
 - **Artifact:** cl1-jpa-<apellidoPaterno-primerNombre>



- Configurar dependencias en el pom.xml

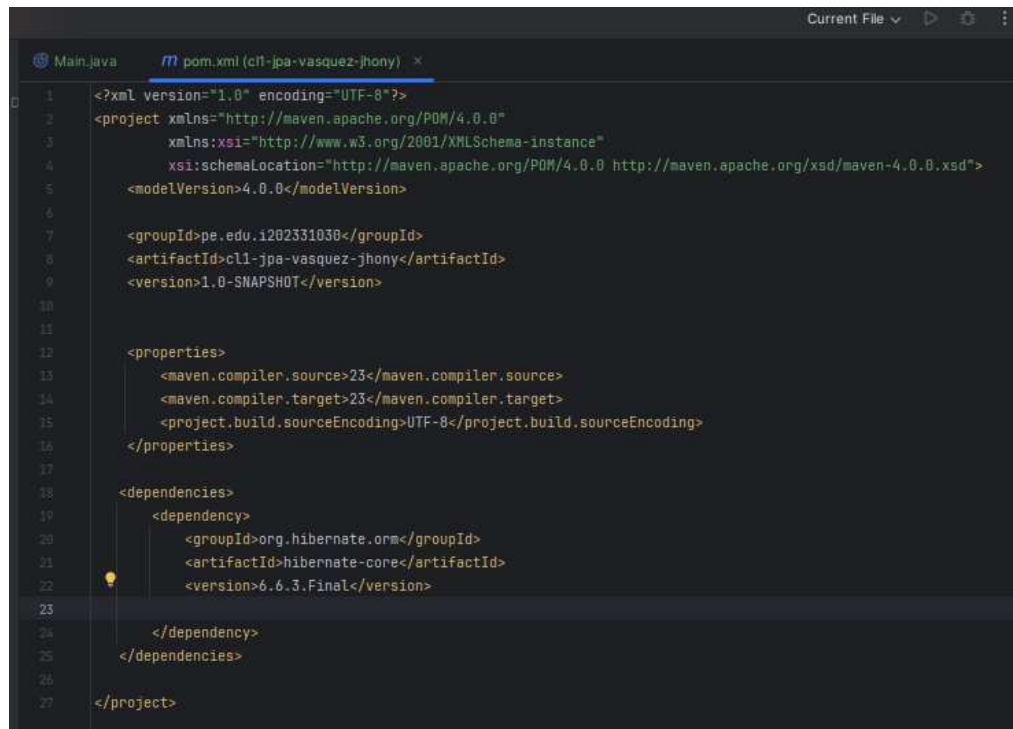
```
<properties>
  <maven.compiler.source>23</maven.compiler.source>
  <maven.compiler.target>23</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
  <dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.6.3.Final</version>
  </dependency>

  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>9.1.0</version>
  </dependency>
</dependencies>

</project>
```

- Hibernate (6.6.2.Final)



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>pe.edu.i202331030</groupId>
8      <artifactId>cll-jpa-vasquez-jhony</artifactId>
9      <version>1.0-SNAPSHOT</version>
10
11
12      <properties>
13          <maven.compiler.source>23</maven.compiler.source>
14          <maven.compiler.target>23</maven.compiler.target>
15          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16      </properties>
17
18      <dependencies>
19          <dependency>
20              <groupId>org.hibernate.orm</groupId>
21              <artifactId>hibernate-core</artifactId>
22              <version>6.6.3.Final</version>
23          </dependency>
24      </dependencies>
25
26
27  </project>

```

- Driver de MySQL (9.1.0)



```

    <properties>
        <maven.compiler.source>23</maven.compiler.source>
        <maven.compiler.target>23</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>6.6.3.Final</version>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <version>9.1.0</version>
        </dependency>
    </dependencies>

</project>

```

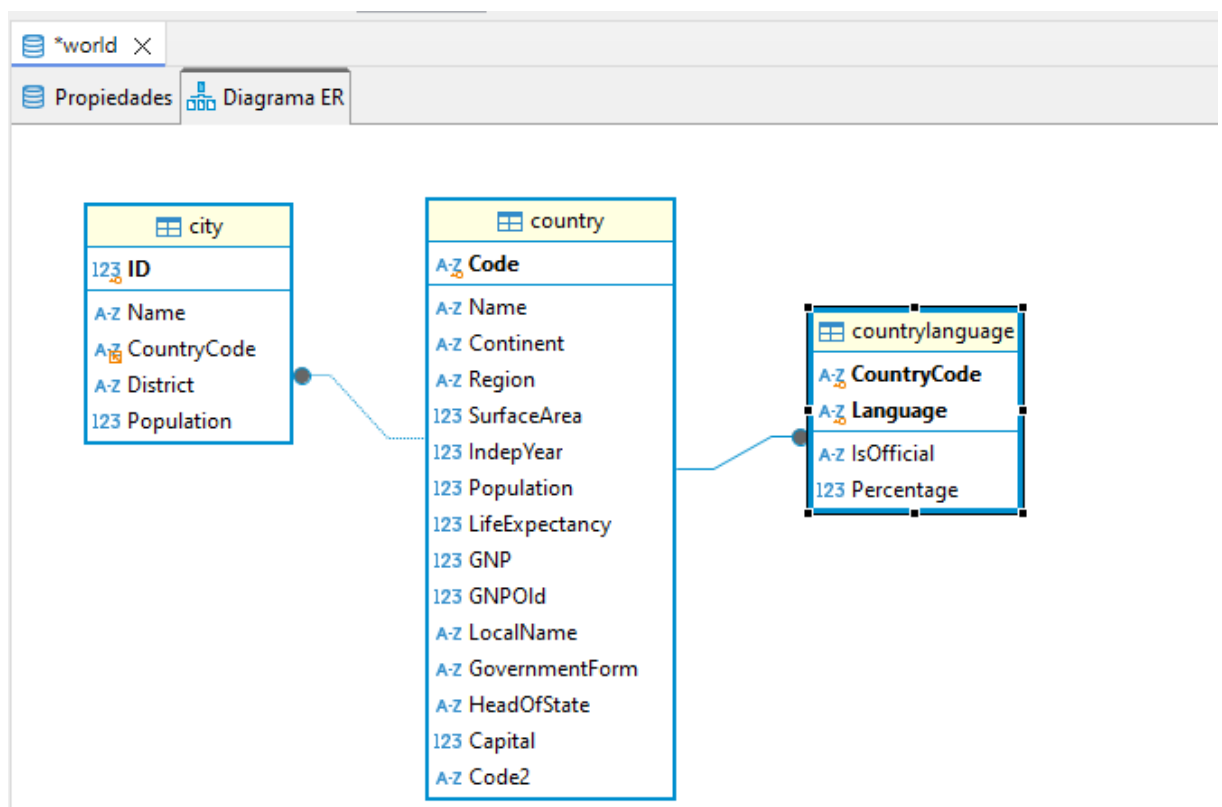
- Configurar unidad de persistencia en archivo "persistence.xml"

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
5
6      <persistence-unit name="JPA_PU" transaction-type="RESOURCE_LOCAL">
7          <properties>
8
9              <!-- Database connection settings -->
10             <property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
11             <property name="jakarta.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/world"/>
12             <property name="jakarta.persistence.jdbc.user" value="root"/>
13             <property name="jakarta.persistence.jdbc.password" value="coco12345"/>
14
15             <!-- Echo all executed SQL to console -->
16             <property name="hibernate.show_sql" value="true"/>
17             <property name="hibernate.format_sql" value="true"/>
18             <property name="hibernate.highlight_sql" value="true"/>
19
20         </properties>
21     </persistence-unit>
22
23 </persistence>

```

- Crear las entidades correspondientes a las siguientes tablas de la bd “world”:



Entidad City

```
package pe.edu.i202331030.entity;

public class City { 12 usages

    private Integer id; 4 usages
    private String name; 4 usages
    private String district; 4 usages
    private Integer population; 4 usages

    public City() { no usages
    }

    public City(Integer id, String name, String district, Integer population) { 3 usages
        this.id = id;
        this.name = name;
        this.district = district;
        this.population = population;
    }

    @Override
    public String toString() {
        return "City{" +
            "id=" + id +
            ", name=" + name + '\'' +
            ", district=" + district + '\'' +
            ", population=" + population +
            '\'';
    }

    public Integer getId() { no usages
        return id;
    }

    public void setId(Integer id) { no usages
```

Entidad Country

```
package pe.edu.i202331030.entity;

public class Country { 8 usages

    private String code; 4 usages
    private String name; 4 usages
    private String continent; 4 usages
    private String region; 4 usages
    private Double surfaceArea; 4 usages
    private Integer indepYear; 4 usages
    private Integer population; 4 usages
    private Double lifeExpectancy; 4 usages
    private Double GNP; 4 usages
    private Double GNPOld; 4 usages
    private String localName; 4 usages
    private String governmentForm; 4 usages
    private String headOfState; 4 usages
    private Integer capital; 4 usages
    private String code2; 4 usages

    public Country(String code, String name, String continent, String region, Double surfaceArea,
        this.code = code;
        this.name = name;
        this.continent = continent;
        this.region = region;
        this.surfaceArea = surfaceArea;
        this.indepYear = indepYear;
        this.population = population;
        this.lifeExpectancy = lifeExpectancy;
        this.GNP = GNP;
        this.GNPOld = GNPOld;
        this.localName = localName;
        this.governmentForm = governmentForm;
        this.headOfState = headOfState;
        this.capital = capital;
```


Entidad Countrylanguage

```
package pe.edu.i202331030.entity;

public class CountryLanguage { 1 usage

    private String language; 4 usages
    private String isOfficial; 4 usages
    private Double percentage; 4 usages

    public CountryLanguage() { no usages
    }

    public CountryLanguage(String language, String isOfficial, Double percentage) { no usages
        this.language = language;
        this.isOfficial = isOfficial;
        this.percentage = percentage;
    }

    @Override
    public String toString() {
        return "CountryLanguage{" +
            "language='" + language + '\'' +
            ", isOfficial='" + isOfficial + '\'' +
            ", percentage=" + percentage +
            '}';
    }

    public String getLanguage() { no usages
        return language;
    }

    public void setLanguage(String language) { no usages
        this.language = language;
    }
}
```

- Mapear las 3 entidades de forma tradicional (Sin Lombok).

```
import jakarta.persistence.*;

@Entity no usages
@Table(name = "city")
public class city {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer ID;

    @Column(name = "Name", nullable = false, length = 35) 4 usages
    private String Name;

    @Column(name = "CountryCode", nullable = false, length = 3) 4 usages
    private String CountryCode;

    @Column(name = "District", nullable = false, length = 20) 4 usages
    private String District;

    @Column(name = "Population", nullable = false) 4 usages
    private Integer Population;

    public city() {
    }
}
```

```

package pe.edu.i202331030.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

import java.util.List;

@Entity 4 usages
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "Code", length = 3, nullable = false)
    private String code;

    @Column(name = "Name", length = 52, nullable = false) 5 usages
    private String name;

    @Column(name = "Continent", nullable = false) 5 usages
    private String continent;

    @Column(name = "Region", length = 26, nullable = false) 5 usages
    private String region;

    @Column(name = "SurfaceArea", precision = 10, scale = 2, nullable = false) 5 usages
    private double surfaceArea;

    @Column(name = "IndepYear") 5 usages
    private Integer indepYear;

    @Column(name = "Population", nullable = false) 5 usages
    private Integer population;
}

```

```

package pe.edu.i202331030.entity;

import jakarta.persistence.*;

@Entity 1 usage
@Table(name = "CountryLanguage")
@IdClass(CountryLanguageId.class)
public class CountryLanguage {

    @Id
    @Column(name = "CountryCode", length = 3, nullable = false)
    private String countryCode;

    @Id
    @Column(name = "Language", length = 30, nullable = false)
    private String language;

    @Column(name = "IsOfficial", nullable = false) 4 usages
    private String isOfficial;

    @Column(name = "Percentage", precision = 4, scale = 1, nullable = false) 4 usages
    private double percentage;

    public CountryLanguage() {
    }

    public CountryLanguage(String language, String isOfficial, Double percentage) { no usages
        this.language = language;
        this.isOfficial = isOfficial;
        this.percentage = percentage;
    }

    @Override
    public String toString() {
        return "CountryLanguage{" +

```

- Definir la estrategia de generación correcta para los PKs.

```
package pe.edu.i202331030.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

import java.util.List;

@Entity 4 usages
@Table(name = "country")
public class Country {

    @Id
    @Column(name = "Code", length = 3, nullable = false)
    private String code;

    @Column(name = "Name", length = 52, nullable = false) 5 usages
    private String name;

    @Column(name = "Continent", nullable = false) 5 usages
    private String continent;
}
```

```
import jakarta.persistence.*;

@Entity 1 usages
public class City {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer id;
    private String name; 4 usages
    private String district; 4 usages
    private Integer population; 4 usages

    public City() {
    }

    public City(Integer id, String name, String district, Integer population) { 4 usages
        this.id = id;
        this.name = name;
        this.district = district;
        this.population = population;
    }
}
```

```
package pe.edu.i202331030.entity;

import jakarta.persistence.*;

@Entity no usages
@Table(name = "CountryLanguage")
public class CountryLanguage {

    @EmbeddedId no usages
    private CountryLanguageId id;

    @Column(name = "IsOfficial", nullable = false, length = 1) 4 usages
    private String isOfficial;

    @Column(name = "Percentage", nullable = false) 4 usages
    private double percentage;

    @ManyToOne no usages
    @MapsId("countryCode") // Vincula CountryCode con el ID embebido
    @JoinColumn(name = "CountryCode", nullable = false)
    private Country country;
}
```

```

import jakarta.persistence.Column;
import jakarta.persistence.Embeddable;
import java.io.Serializable;
import java.util.Objects;

@Embeddable 3 usages
public class CountryLanguageId implements Serializable {

    @Column(name = "CountryCode", length = 3, nullable = false) 5 usages
    private String countryCode;

    @Column(name = "Language", length = 30, nullable = false) 5 usages
    private String language;

    // Rename usages
    // Constructor por defecto
    public CountryLanguageId() {
    }

    // Getters, Setters, equals y hashCode
    public String getCountryCode() { return countryCode; }
}

```

- Considerar el mapeo de las relaciones de forma bidireccional.

```

package pe.edu.i202331030.entity;
import jakarta.persistence.*;

@Entity 14 usages
@Table(name = "city")
public class City {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY) // Estrategia para generar el ID automáticamente
    @Column(name = "ID")
    private Integer id;
    private String name; 4 usages
    private String district; 4 usages
    private Integer population; 4 usages

    @ManyToOne(cascade = CascadeType.REMOVE) 3 usages
    @JoinColumn(name = "CountryCode")
    private Country country;

    public City() {
    }

    public City(String name, String district, Integer population, Country country) { 3 usages
        this.name = name;
        this.district = district;
        this.population = population;
        this.country = country;
    }

    @Override
    public String toString() {
        return "City{" +
            "id=" + id +
            ", name=" + name + '\'' +
}

```

```

package pe.edu.i202331030.entity;

import jakarta.persistence.*;
import java.util.List;

@Entity
public class Country {

    @Id
    private String code;
    private String name;
    private String continent;
    private String region;
    private Double surfaceArea;
    private Integer indepYear;
    private Integer population;
    private Double lifeExpectancy;
    private Double GNP;
    private Double GNPOld;
    private String localName;
    private String governmentForm;
    private String headOfState;
    private Integer capital;
    private String code2;

    @OneToMany(mappedBy = "country", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<City> city;

    @OneToMany(mappedBy = "country", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<CountryLanguage> countryLanguage;

    public Country() {
    }

```

```

package pe.edu.i202331030.entity;
import jakarta.persistence.*;

@Entity
public class CountryLanguage {

    @Id
    private String language;
    private String isOfficial;
    private Double percentage;

    @ManyToOne(cascade = CascadeType.REMOVE)
    @JoinColumn(name = "CountryCode")
    private Country country;

    public CountryLanguage() {
    }

    public CountryLanguage(String language, String isOfficial, Double percentage, Country country) {
        this.language = language;
        this.isOfficial = isOfficial;
        this.percentage = percentage;
        this.country = country;
    }

    @Override
    public String toString() {
        return "CountryLanguage{" +
            "language='" + language + '\'' +
            ", isOfficial='" + isOfficial + '\'' +
            ", percentage=" + percentage +
            '}';
    }

    public String getLanguage() {
        return language;
    }

```

- Crear una clase “JPAPersist” y en ella registre un país imaginario, que tenga 3 ciudades y 2 lenguajes nativos. Sólo debe realizar un llamado al método “persist”.

```

import java.util.ArrayList;
public class JPAPersist {

    public static void main(String[] args) {

        //Inicializar el JPA a la
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("world");
        EntityManager em = emf.createEntityManager();

        //Crear país imaginario
        Country country = new Country("PEU", "Peru", "South America", "South America", 45533.32, 2.345);

        //Crear ciudades
        City city1 = new City("Lima", "Centro", 10000, country);
        City city2 = new City("Arequipa", "Sur", 40000, country);
        City city3 = new City("Iquitos", "Noreste", 40000, country);

        //Registrar ciudades a Country
        country.setCitys(ArrayList.of(city1, city2, city3));

        //Crear 2 lenguajes nativos
        CountryLanguage language1 = new CountryLanguage("Quechua", "T", 2.4, country);
        CountryLanguage language2 = new CountryLanguage("Aymara", "T", 2.4, country);

        country.setCountryLanguages(ArrayList.of(language1, language2));

        //Registrar Country
        em.getTransaction().begin();
        em.persist(country);
        em.getTransaction().commit();

        System.out.println("Registro exitoso");
    }
}

```

```

city
(CountryCode, district, name, population)
values
(?, ?, ?, ?)
[Hibernate]
insert
into
city
(CountryCode, district, name, population)
values
(?, ?, ?, ?)
[Hibernate]
insert
into
CountryLanguage
(CountryCode, isOfficial, percentage, language)
values
(?, ?, ?, ?)
[Hibernate]
insert
into
CountryLanguage
(CountryCode, isOfficial, percentage, language)
values
(?, ?, ?, ?)
Registro exitoso

```

country											
Propiedades		Datos		Diagrama ER		localhost Databases world Tables coun					
country Code IN (PEU)											
Grids											
A.Z. Code A.Z. Name A.Z. Continent A.Z. Region 121 SurfaceArea 122 Indep 123 Population 124 LifeExpect 125 GNP 126 LocalName											
1 PEU Peru South America South America 45,533.32 2,345 415,600 34.2 6,500.5 10,883.2 Peru											

city CountryCode IN ('PEU')						
	123 ID	A-Z Name	A-Z CountryCode	A-Z District	123 Population	
1	4.089	Alegria	PEU	Caridad	16.000	
2	4.090	Marea	PEU	Jornal	45.000	
3	4.091	Juanjo	PEU	Normando	60.000	

countrylanguage CountryCode IN ('PEU')				
	A-Z CountryCode	A-Z Language	A-Z IsOfficial	123 Percentage
1	PEU	Antamino	F	7,4
2	PEU	Rustico	F	2,4

- Crear una clase “JPARemove” y en ella elimine el país imaginario (Previamente creado). La eliminación debe eliminar el rastro de sus 3 ciudades y 2 lenguajes nativos. Sólo debe realizar un llamado al método “remove”. Considere, no afectar la funcionalidad de la clase “JPAPersist”.

```

public static void main(String[] args) {
    //referenciar al EMF y EM
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("persistenceUnitName: world");
    EntityManager em = emf.createEntityManager();

    // Consultar el pais imaginario por su clave primaria (Code)
    Country country = em.find(Country.class, "PEU");

    // Si el pais existe, eliminarlo junto con sus relaciones
    if (country != null) {
        em.getTransaction().begin();

        // Eliminar las ciudades asociadas
        for (City city : country.getCity()) {
            em.remove(city);
        }

        // Eliminar los lenguajes asociados
        for (CountryLanguage language : country.getCountryLanguage()) {
            em.remove(language);
        }

        // Eliminar el pais
        em.remove(country);

        em.getTransaction().commit();
        System.out.println("El pais y sus registros asociados han sido eliminados.");
    } else {
        System.out.println("El pais no existe en la base de datos.");
    }
}

```



```

        ID=?
[Hibernate]
    delete
    from
        city
    where
        ID=?
[Hibernate]
    delete
    from
        CountryLanguage
    where
        language=?
[Hibernate]
    delete
    from
        CountryLanguage
    where
        language=?
[Hibernate]
    delete
    from
        Country
    where
        code=?
El país y sus registros asociados han sido eliminados.

Process finished with exit code 0

```

- Crear una clase “JPAFind” y en ella realice una sola consulta a la entidad “Country” (Busque el código “PER” usando find) y en base al resultado imprima el nombre de las ciudades peruanas con población > 700k. **Deberá usar una función lambda** para discriminar el resultado.

```

import java.util.List;
import java.util.stream.Collectors;

> public class JPAFind {
>
    public static void main(String[] args) {

        //Referenciar al EMF y EM
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("world");
        EntityManager em = emf.createEntityManager();

        // Buscar el país con el código 'PER' usando find
        Country country = em.find(Country.class, "PER");

        if (country != null) {
            System.out.println("Ciudades de " + country.getName() + " con población mayor a 700k:");

            // Filtrar las ciudades de Perú con población mayor a 700,000 usando una función lambda
            List<City> citiesWithLargePopulation = country.getCity().stream()
                .filter(city -> city.getPopulation() > 700000) // Población mayor a 700k
                .collect(Collectors.toList());

            // Imprimir las ciudades filtradas
            citiesWithLargePopulation.forEach(city ->
                System.out.println(city.getName() + " - Población: " + city.getPopulation()));
        } else {
            System.out.println("No se encontró el país con el código 'PER'.");
        }
    }
}

```

```

        c1_0.headOfState,
        c1_0.indepYear,
        c1_0.lifeExpectancy,
        c1_0.localName,
        c1_0.name,
        c1_0.population,
        c1_0.region,
        c1_0.surfaceArea
    from
        Country c1_0
    where
        c1_0.code=?
Ciudades de Peru con población mayor a 700k:
[Hibernate]
    select
        c1_0.CountryCode,
        c1_0.ID,
        c1_0.district,
        c1_0.name,
        c1_0.population
    from
        city c1_0
    where
        c1_0.CountryCode=?
Lima - Población: 6464693
Arequipa - Población: 762000

```

Parte 03 Proyecto Spring Data JPA (25%)

- Generar un proyecto con Spring Data JPA desde <https://start.spring.io/> con las siguientes características:
 - **Project:** Maven
 - **Language:** Java
 - **Spring Boot:** 3.3.5
 - **Group:** pe.edu.<codigoEstudiante>
 - **Artifact:** cl1-jpa-data-<apellidoPaterno-primerNombre>
 - **Packaging:** Jar
 - **Java:** 23
 - **Dependencies:**
 - Spring Data JPA
 - MySQL Driver
 - Lombok



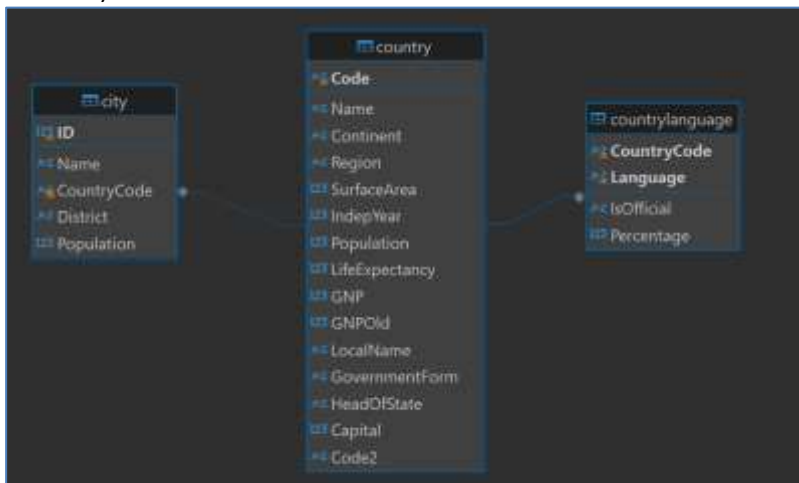
- Configurar el “application.properties” con los datos de conectividad a la bd “world”.

```

application.properties
1  spring.application.name=cl1-jpa-data-vasquez-jhony
2
3  spring.jpa.hibernate.ddl-auto=none
4  spring.datasource.url=jdbc:mysql://localhost:3306/world
5  spring.datasource.username=root
6  spring.datasource.password=coco12345
7  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
8  spring.jpa.show-sql=true
9

```

- Crear las entidades correspondientes a las siguientes tablas (Las mismas del proyecto anterior):



- Mapear las 3 entidades usando Lombok.

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony.entity;

import jakarta.persistence.*;
import lombok.*;

@Entity
@Table(name = "city")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class City {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ID")
    private Integer id;

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

    @Column(name = "District")
    private String district;

    @Column(name = "Population")
    private Integer population;

    @ManyToOne
    @JoinColumn(name = "CountryCode", nullable = false)
    private Country country;
}

```

```

@Entity 3 usages
@Table(name = "country")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Country {

    @Id
    @Column(name = "Code", length = 3)
    private String code;

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

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

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

    @Column(name = "SurfaceArea")
    private Double surfaceArea;

    @Column(name = "IndepYear")
    private Integer indepYear;

    @Column(name = "Population")
    private Integer population;

    @Column(name = "LifeExpectancy")
    private Double lifeExpectancy;

    @Column(name = "GNP")

```

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony;

import jakarta.persistence.*;
import lombok.*;
import pe.edu.i202331030.cl1_jpa_data_vasquez_jhony.entity.Country;

@Entity no usages
@Table(name = "countrylanguage")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CountryLanguage {

    @EmbeddedId
    private CountryLanguageId id;

    @Column(name = "IsOfficial", nullable = false, length = 1)
    private String isOfficial;

    @Column(name = "Percentage", nullable = false)
    private Double percentage;

    @ManyToOne
    @JoinColumn(name = "CountryCode", insertable = false, updatable = false)
    private Country country;
}

```

品

NAME: _____

-

1

```

@Entity 3 usages
@Table(name = "country")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Country {

    @Id
    @Column(name = "Code", length = 3)
    private String code;

```

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony;

import jakarta.persistence.*;
import lombok.*;
import pe.edu.i202331030.cl1_jpa_data_vasquez_jhony.entity.Country;

@Entity no usages
@Table(name = "countrylanguage")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CountryLanguage {

    @EmbeddedId
    private CountryLanguageId id;

```

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony;

import jakarta.persistence.*;
import lombok.*;
import java.io.Serializable;

@Embeddable 1 usage
@Data
@NoArgsConstructor
@AllArgsConstructor
public class CountryLanguageId implements Serializable {

```

- Considerar el mapeo de las relaciones de forma bidireccional.

```

@OneToMany(mappedBy = "country", cascade = CascadeType.ALL, orphanRemoval = true)
private List<City> cities;

@OneToMany(mappedBy = "country", cascade = CascadeType.ALL, orphanRemoval = true)
private List<CountryLanguage> languages;

```



```

13     @Column(name = "District")
14     private String district;
15
16     @Column(name = "Population")
17     private Integer population;
18
19     @ManyToOne
20     @JoinColumn(name = "CountryCode", nullable = false)
21     private Country country;
22

```

```

22     private Double percentage;
23
24     @ManyToOne
25     @MapsId("countryCode")
26     @JoinColumn(name = "CountryCode", insertable = false, updatable = false)
27     private Country country;
28

```

- Crear una interfaz “CountryRepository” que extienda de “CrudRepository”.

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony.CrudRepository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import pe.edu.i202331030.cl1_jpa_data_vasquez_jhony.entity.Country;

public interface CountryRepository extends CrudRepository<Country, String> {
    |
}

```

- Implementar el método “run” definido en la interfaz “CommandLineRunner” desde la clase principal del proyecto de Spring Boot.

```

package pe.edu.i202331030.cl1_jpa_data_vasquez_jhony;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Cl1JpaDataVasquezJhonyApplication implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(Cl1JpaDataVasquezJhonyApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        // Lógica a ejecutar al iniciar la aplicación
        System.out.println("Aplicación iniciada correctamente.");

        // Ejemplo de una operación simple (se puede reemplazar con lógica específica)
        realizarConsultaEjemplo();
    }

    private void realizarConsultaEjemplo() { // !usage
        System.out.println("Ejecutando una consulta inicial o tarea...");
        // Aquí podrías interactuar con tus repositorios o servicios
    }
}

```



```

2024-11-24T23:45:17.124-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.p.s. RegistryConfiguration$Default
2024-11-24T23:45:17.413-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.p.s. RegistryConfiguration$Default
2024-11-24T23:45:17.953-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.p.s. RegistryConfiguration$Default
2024-11-24T23:45:18.054-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] org.hibernate.Version
2024-11-24T23:45:18.089-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.c.internal.BootstrapInitializer
2024-11-24T23:45:18.525-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.c.j.s.BootstrapInitializer$Default
2024-11-24T23:45:18.570-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] com.zaxxer.hikari.HikariDataSource
2024-11-24T23:45:19.129-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] com.zaxxer.hikari.pool.HikariPool
2024-11-24T23:45:19.132-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] com.zaxxer.hikari.HikariDataSource
2024-11-24T23:45:20.714-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.c.j.s.BootstrapInitializer
2024-11-24T23:45:20.718-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] j.LocalContainerEntityManagerFactoryBean
2024-11-24T23:45:21.265-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [main] o.h.c.j.s.BootstrapInitializer
Aplicación iniciada correctamente.
Ejecutando una consulta inicial a tarea...
2024-11-24T23:45:21.379-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean
2024-11-24T23:45:21.777-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
2024-11-24T23:45:21.788-05:00 INFO 10220 --- [c11-jpa-data-vasquez-jhony] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource
Process finished with exit code 0

```

- Deberá implementar las siguientes 3 consultas:
 - **ifPresentOrElse()**
Imprimir en la terminal los nombres de los lenguajes que se hablan en el país “ARG” (Argentina). En caso de no obtener resultado, deberá imprimir los nombres de los lenguajes del país “PER” (Perú).
<imagen2>
 - **deleteAllById()**
Eliminar 2 países: “COL” y “ARG”. La eliminación deberá ser cascada y borrará sus ciudades y lenguajes correspondientes.
<imagen2>
 - Volver a ejecutar la primera consulta, pues al eliminar “ARG”, deberá ejecutarse el flujo alterno. (Deberá restaurar la BD desde la terminal en cada prueba)
<imagen2>

Parte 04 Gestión de conexiones (25%)

- Crear una clase de configuración denominada “ConexionesConfig.java”, en ella deberá implementar una personalización del DataSource de HikariCP. Considere los siguientes valores para el pool de conexiones:
 - MaxmumPoolSize: 30
 - MinimumIdle: 4
 - IdleTimeout: 4 minutos
 - ConnectionTimeout: 45 segundos
- Las credenciales del DataSource (Parámetros de conexión a la BD) no deben ser visibles en el código fuente. Para ello deberá crear las siguientes variables de entorno:
 - DB_WORLD_URL
 - DB_WORLD_USER
 - DB_WORLD_PASS
 - DB_WORLD_DRIVER

<imagen1>

<imagen2>

- Luego de configurar el DataSource, ¿Es necesario proporcionar las credenciales desde el archivo “application.properties”? ¿Por qué? Explique con sus propias palabras.
<Respuesta>
- ¿Por qué debo o no, configurar un JNDI en Spring Boot?
<Respuesta>