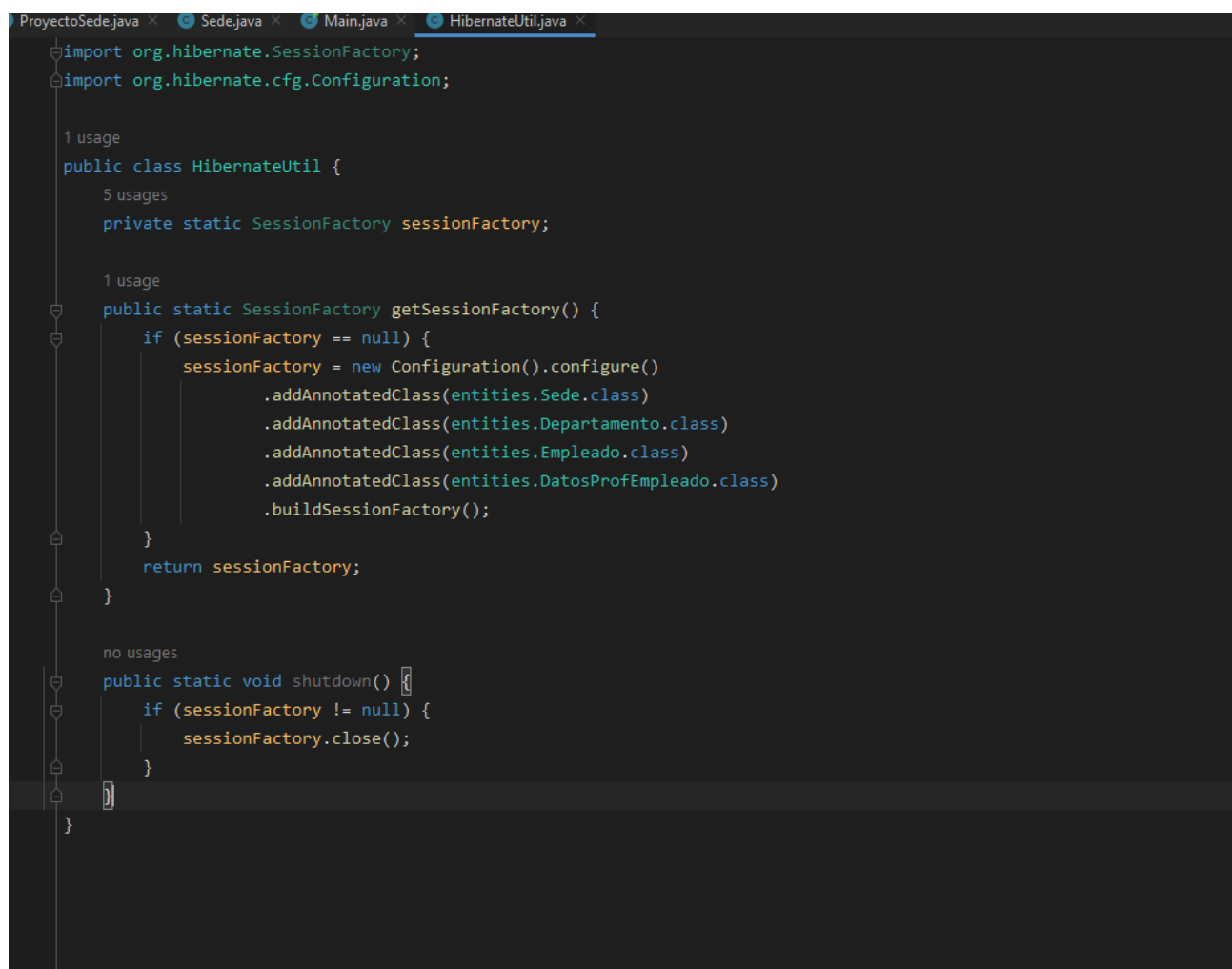


Tenemos dos maneras de hacer las inserciones, dependiendo de si utilizamos XML o Anotaciones JPA.

1. Utilizando XML

1.1 Creación de clase Hibernate Utils.

En esta clase crearemos los métodos para abrir la sesión, añadir las clases Java a la base de datos y finalmente cerrar la sesión



```
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

1 usage
public class HibernateUtil {
    5 usages
    private static SessionFactory sessionFactory;

    1 usage
    public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            sessionFactory = new Configuration().configure()
                .addAnnotatedClass(entities.Sede.class)
                .addAnnotatedClass(entities.Departamento.class)
                .addAnnotatedClass(entities.Empleado.class)
                .addAnnotatedClass(entities.DatosProfEmpleado.class)
                .buildSessionFactory();
        }
        return sessionFactory;
    }

    no usages
    public static void shutdown() {
        if (sessionFactory != null) {
            sessionFactory.close();
        }
    }
}
```

Figura 1: Clase HibernateUtils

1.2 La clase main

Una vez hecho esto, nos iremos a nuestra clase main y lo primero que deberemos hacer será

- Obtener el sessionFactory con el método que hemos creado en HibernateUtils
- Abrir la sesión
- Declarar una variable de tipo "Transaction" a null.

```
public class Main {  
    public static void main(String[] args) {  
        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();  
  
        Session session = sessionFactory.openSession();  
  
        Transaction transaction = null;
```

Figura 2: Clase sessionFactory

Luego, dentro de un try-catch abriremos la transacción con la sesión que hemos creado antes y empezaremos a introducir los datos creando instancias de las clases Java y utilizando los setters.

Es importante que cada vez que introduzcamos un dato guardemos con session.save().

Se vería de la siguiente forma:

```
try {  
    transaction = session.beginTransaction();  
  
    Sede sede = new Sede();  
    sede.setNombre("Sede principal");  
  
    session.save(sede);  
  
    Departamento depto1 = new Departamento();  
    depto1.setNombre("Departamento de Ventas");  
    depto1.setSede(sede);  
    session.save(depto1);  
  
    Departamento depto2 = new Departamento();  
    depto2.setNombre("Departamento de Marketing");  
    depto2.setSede(sede);  
    session.save(depto2);  
  
    Empleado emp1 = new Empleado();  
    emp1.setDni("12345678A");  
    emp1.setDepartamento(depto1);  
    emp1.setNombre("Carlos Martínez");  
    session.save(emp1);  
  
    Empleado emp2 = new Empleado();  
    emp2.setDni("12345677B");  
    emp2.setDepartamento(depto1);  
    emp2.setNombre("Andrés Iniesta");  
    session.save(emp2);  
}
```

Figura 3: Inserción de datos (XML)

Al final, deberemos hacer commit para efectuar la transacción y en el catch pondremos que si pasa algún tipo de problema, la transacción haga rollback y no se ejecute nada.

```
        session.save(emp4);

        transaction.commit();

    } catch (Exception e) {
        if (transaction != null) {
            transaction.rollback();
        }

        throw new RuntimeException(e);
    } finally {
        session.close();
    }
}
}
```

Figura 4: Commit y cerrar sesión

Utilizando este método puede que de error por incompatibilidad entre Hibernate y Java 22. Para solucionar esto deberemos ir a Run → Edit Configuration → Add new run configuration → Application y debemos poner lo siguiente:

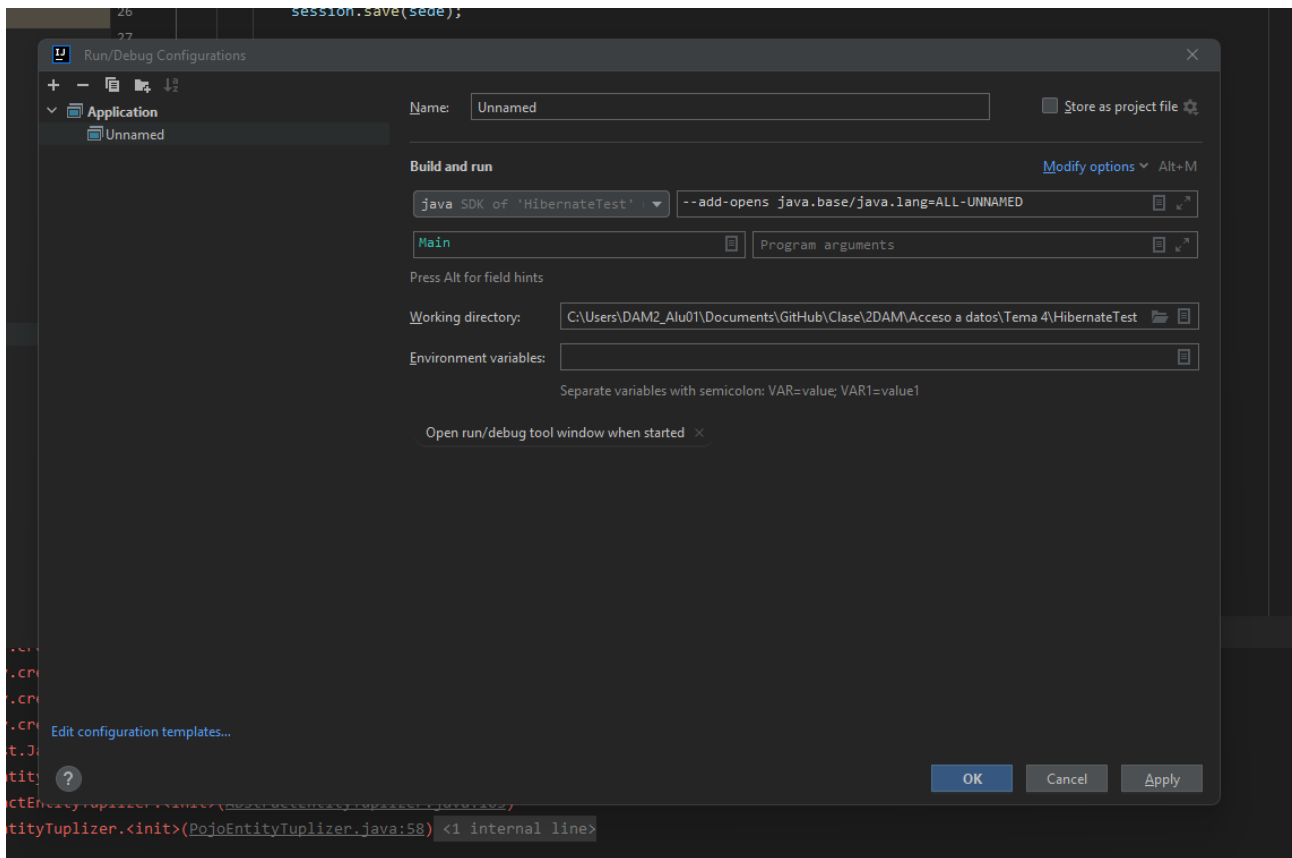


Figura 5: Run configuration

De esta forma forzaremos al programa a ejecutarse.

2.- Utilizando anotaciones JPA

Si utilizamos este método de mapeo, deberemos utilizar EntityManagerFactory y la librería de jakarta.persistence (Hibernate 6 o más) o javax.persistence (para versiones inferiores)

2.1 Creación del documento persistence.xml

Lo primero que deberemos hacer es crear un archivo llamado "persistence.xml" y tiene que estar situado dentro de src/main/resources/META-INF o (src/resources/META-INF si no utilizamos Maven). En ese archivo .xml pondremos todo lo relacionado con la configuración de la conexión (Driver, url, nombre de usuario de la bd...)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">
  <persistence-unit name="SedePU">
    <properties>
      <property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
      <property name="jakarta.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/proyecto_orm?serverTimezone=UTC"/>
      <property name="jakarta.persistence.jdbc.user" value="usuarioGlobal"/>
      <property name="jakarta.persistence.jdbc.password" value="abc123."/>

      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL8Dialect"/>

      <property name="hibernate.hbm2ddl.auto" value="update"/>

      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

Figura 6: Documento persistence.xml

2.2 La clase Main

Esta vez, en la clase main, utilizaremos la clase EntityManagerFactory con la cual enlazaremos nuestro archivo persistence.xml. A continuación, crearemos la entity manager con el método .createEntityManager()

```
public class Main {
  public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("dataPersistence");
    EntityManager em = emf.createEntityManager();
  }
}
```

Figura 7: Clase EntityManagerFactory

En cuanto a la inserción de datos, es muy parecido a la otra forma, lo único que cambia es que en vez de usar `session.save(entidad)` utilizaremos el método `.persist(entidad)`

```
EntityManager em = emf.createEntityManager();

try {
    em.getTransaction().begin();

    Sede nuevaSede = new Sede();
    nuevaSede.setNombre("Sede Central");
    em.persist(nuevaSede);

    Departamento depto1 = new Departamento();
    depto1.setNombre("Recursos Humanos");
    depto1.setSede(nuevaSede);
    em.persist(depto1);

    Departamento depto2 = new Departamento();
    depto2.setNombre("Tecnología");
    depto2.setSede(nuevaSede);
    em.persist(depto2);

    Empleado emp1 = new Empleado();
    emp1.setDni("12345678A");
    emp1.setNombre("Juan Pérez");
    emp1.setDepartamento(depto1);
    em.persist(emp1);

    Empleado emp2 = new Empleado();
    emp2.setDni("87654321B");
    emp2.setNombre("Ana Gómez");
    emp2.setDepartamento(depto1);
    em.persist(emp2);
}
```

Figura 8: Inserción de datos (JPA)

Como abrimos también una transacción aquí, deberemos cerrarla en el bloque “finally”:

```
2         em.getTransaction().commit();
3         System.out.println("Datos insertados correctamente.");
4
5     } catch (Exception e) {
6         em.getTransaction().rollback();
7         e.printStackTrace();
8     } finally {
9         em.close();
10        emf.close();
11    }
12 }
13 }
14 }
```

Figura 9: Commit y cierre de transacción