INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
ACADEMIA DE SISTEMAS DISTRIBUIDOS
DEPARTAMENTO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

Número de Práctica:
Nombre de la Práctica: Invocación a métodos remotos RMI Java con Sistema Manejador de Bases de Datos vía ORM's JFreeChart y JasperReport

Problemática:
Desarrollar aplicaciones distribuidas, utilizando RMI Java, Base de Datos relacionales, JasperReport y JFreeChart para Interactuar con Bases de Datos y generar Reportes y Gráficos a partir de los datos utilizando ORM's para la capa de acceso a datos.

Elemento de Competencia:
Utiliza invocación a métodos remotos (Java RMI) además de un Mapeador Objeto Relacional, para persistir los datos en un Sistemas Manejadores de Bases de Datos Relacional, así mismo JFreeChart y JasperReport para generar Reportes PDF y Gráficos.

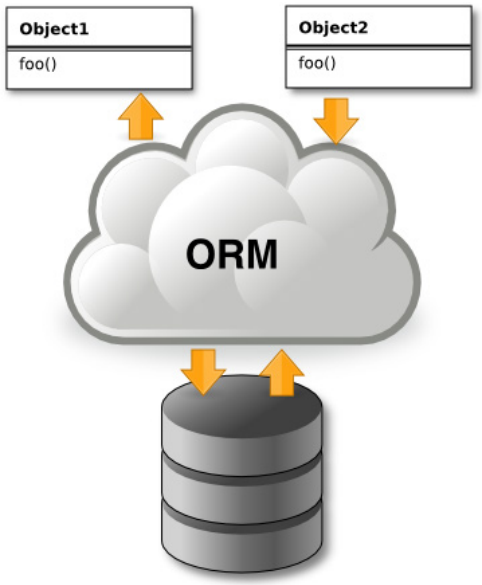
Requerimientos:
Hardware:
- Laboratorio de cómputo
- Red computacional
- Proyector

Software:
- Sistema operativo
- Sistema Manejador de Bases de Datos (MySQL, SQL Server, ORACLE, DB2, etc.)
- Java Development Kit (JSDK 7.x)
- Hibernate
- JasperReport (Jaspersoft iReport Designer)
- JFreeChart

Actividades Previas:
El alumno deberá presentarse en el laboratorio con el diseño lógico de una Base de Datos Relacional, con las características solicitadas por el docente titular de la Unidad de Aprendizaje.

Fundamentos Teóricos:

- ✓ ORM. El mapeo objeto-relacional es una técnica de programación para convertir datos entre bases de datos relacionales y el sistema de tipos utilizado en un lenguaje de programación orientado a objetos, utilizando un motor de persistencia.



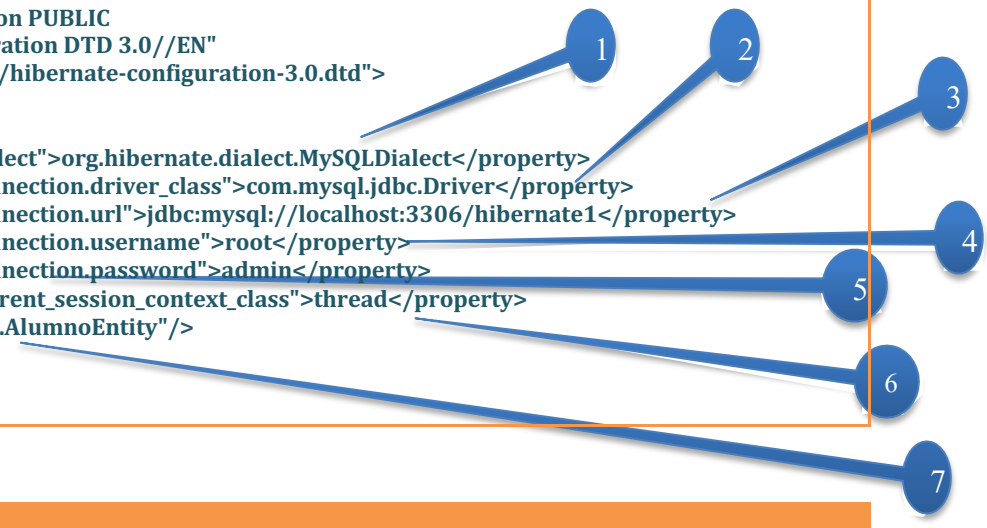
- ✓ ***Hibernate.*** Es una herramienta de Mapeo objeto-relacional para la plataforma Java (disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

- ✓ ***JPA (Java Persistence API - API de Persistencia en Java).*** Es una abstracción sobre JDBC que permite realizar mantener una correlación de forma sencilla, facilitando el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, puede configurarse a través de metadatos (mediante xml o anotaciones).

- ✓ Hibernate
  - Archivos de Configuración

Hibernate

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
 <session-factory>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate1</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password">admin</property>
  <property name="hibernate.current_session_context_class">thread</property>
  <mapping class="com.entidades.AlumnoEntity"/>
 </session-factory>
</hibernate-configuration>
```



Indicador	Funcionalidad
1	**Indica el dialecto especifico para el manejador de bases de datos a utilizar.**
2	**Indica el nombre del driver (Controlador JDBC) especifico para el manejador de bases de datos a utilizar**
3	**Indica la cadena de conexión a la base de datos.**
4	**Indica el nombre de usuario de la base de datos a conectar**
5	**Indica la clave del usuario de la base de datos a conectar**
6	**Indica el mecanismo de administración de las sesiones hibernate**
7	**Indica el nombre de la clase Java mapeada con tabla(relación) en la base de datos**

  - HibernateUtil.- Se encarga de manejar el SessionFactory, aplica el patrón singleton.

HibernateUtil

```
public class HibernateUtil {
   private static final SessionFactory sessionFactory;
   static {
      try {
         sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
      } catch (Throwable ex) {
         System.err.println("Initial SessionFactory creation failed." + ex);
         throw new ExceptionInInitializerError(ex);
      }
   }
   public static SessionFactory getSessionFactory() {
      return sessionFactory;
   }
}
```

- Definir clases entidad para el mapeo de bases de datos utilizando Anotaciones.

Ejemplo: Mapeo de Clase tipo Entidad con Anotaciones

Field
id
nombre
paterno
materno
email
nombreUsuario
claveUsuario
tipoUsuario

Si en la Base de datos se tiene la siguiente estructura para la tabla usuarios

La clase Entidad Java equivalente a la tabla usuarios de la base de datos con las respectivas anotaciones JPA seria la siguiente:
Nota: Tanto Hibernate como JPA utilizan las anotaciones del Api de persistencia Java JPA.

```
@Entity
@Table(name="usuarios")
public class Usuario implements Serializable {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private int id;
    private String nombre;
    private String paterno;
    private String materno;
    private String email;
    private String nombreUsuario;
    private String tipoUsuario;
```

- ✓ JPA
  - Archivo de Configuración

JPA

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
             xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
             http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="JavaApplication13PU"
                      transaction-type="RESOURCE_LOCAL">
        <provider>
            org.eclipse.persistence.jpa.PersistenceProvider
        </provider>
        <class>javaapplication13.Usuarios</class>
        <properties>
            <property name="javax.persistence.jdbc.url"
                      value="jdbc:mysql://localhost:3306/EzjaMVC"/>
            <property name="javax.persistence.jdbc.password"
                      value="admin"/>
            <property name="javax.persistence.jdbc.driver"
                      value="com.mysql.jdbc.Driver"/>
            <property name="javax.persistence.jdbc.user"
                      value="root"/>
        </properties>
    </persistence-unit>
</persistence>
```

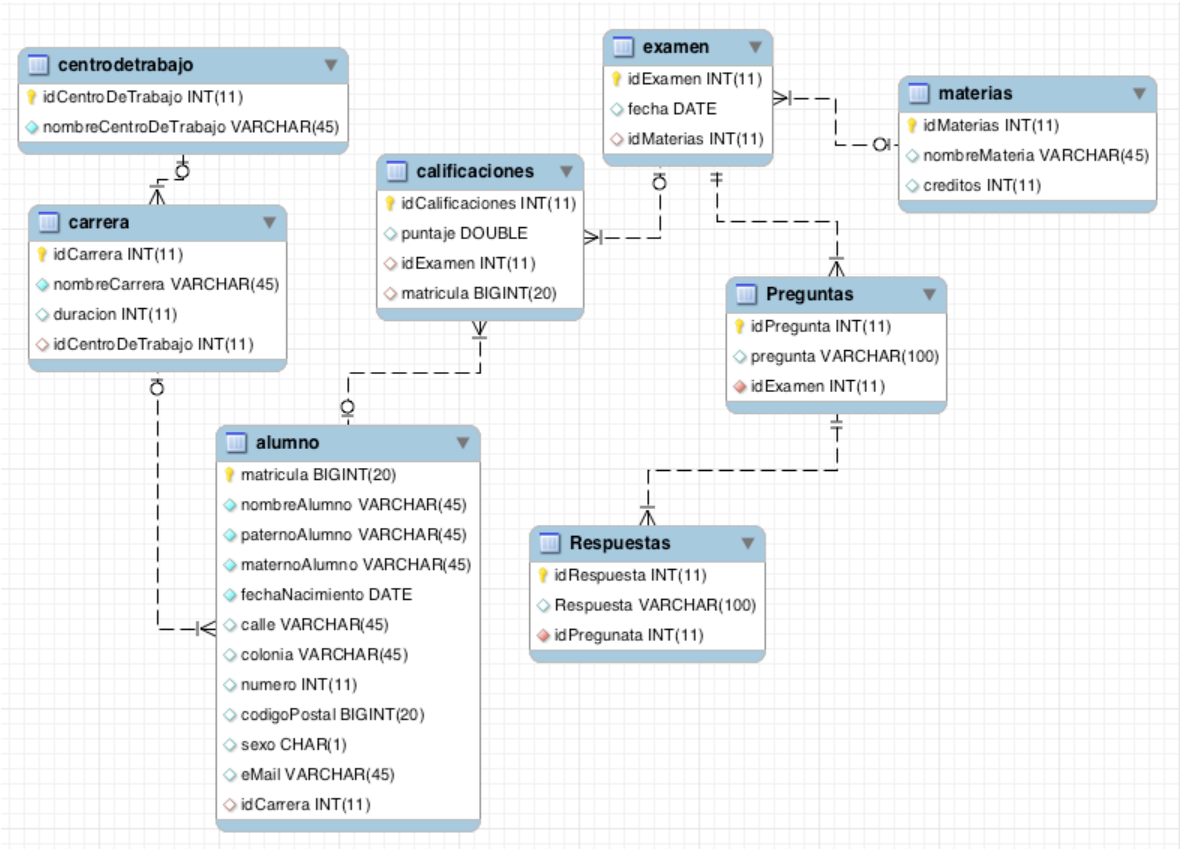
- ✓ Las propiedades de Configuración son similares a Hibernate
- ✓ La Clase de tipo entidad es similar al utilizado en Hibernate

Hibernate	
Método	Funcionalidad
Session session = HibernateUtil.getSessionFactory(). getCurrentSession();	**Obtener la sesión actual de la fabrica de sesiones**
Transaction tr = session.beginTransaction();	**Iniciar una transacción a partir del objeto sesión .**
session.save(objetoEntidad);	**Persiste la instancia del objeto de tipo Entidad en una base de datos relacional.**
session.update(objetoEntidad);	**Actualiza la instancia del objeto de tipo Entidad en una base de datos relacional.**
session.delete(objetoEntidad);	**Elimina la instancia del objeto de tipo Entidad en una base de datos relacional.**
session.get(objetoEntidad.class, PK);	**Busca y Devuelve una instancia del objeto de tipo Entidad. La búsqueda se realiza en base a la clase y a su identificador primario.**
tr.commit();	**Finaliza una transacción, comprometiendo físicamente los datos.**

JPA	
Método	Funcionalidad
EntityManagerFactory emf = Persistence.createEntityManagerFactory ("nombreUnidadDePersistencia");	**Obtener la sesión actual de la fabrica de sesiones**
EntityManager em = emf.createEntityManager ();	**Iniciar el objeto entity manager que tiene la capacidad de interactuar con la base de datos.**
em.getTransaction().begin();	**Iniciar una transacción a partir del objeto Entity Manager**
em.persist(objetoEntidad);	**Persiste la instancia del objeto de tipo Entidad en una base de datos relacional.**
em.merge(objetoEntidad);	**Actualiza la instancia del objeto de tipo Entidad en una base de datos relacional.**
em.remove(objetoEntidad);	**Elimina la instancia del objeto de tipo Entidad en una base de datos relacional.**
em.find(objetoEntidad.class, PK);	**Busca y Devuelve una instancia del objeto de tipo Entidad. La búsqueda se realiza en base a la clase y a su identificador primario.**
em.getTransaction().commit();	**Finaliza una transacción, comprometiendo físicamente los datos.**

Instrucciones:

1. Cree la base de datos a partir del modelo propuesto.



2. Crear una interfaz Gráfica en Java Swing(50 %) y Web (50 %) Utilizando Servlets y/o JSP para dar soporte a las actividades del CRUD (create, retrieve/retrieve all, update y delete), por cada una de las entidades de la base de datos utilizando un mapeador objeto relacional (ORM) como tecnologia para el acceso a datos.
3. Implementar el cliente y el servidor utilizando invocacion a métodos remotos para mantener comunicación con el servidor de bases de datos PostgreSql/MySql/Oracle/ etc y permita dar soporte a las operaciones solicitadas en el punto 2 con sus respectivas validaciones.
4. Generar al menos un Reporte y una Gráfica de cada entidad existente en la base de datos.