Java Developer

Taller de Desarrollo Web

Java Persistence API

(J P A)

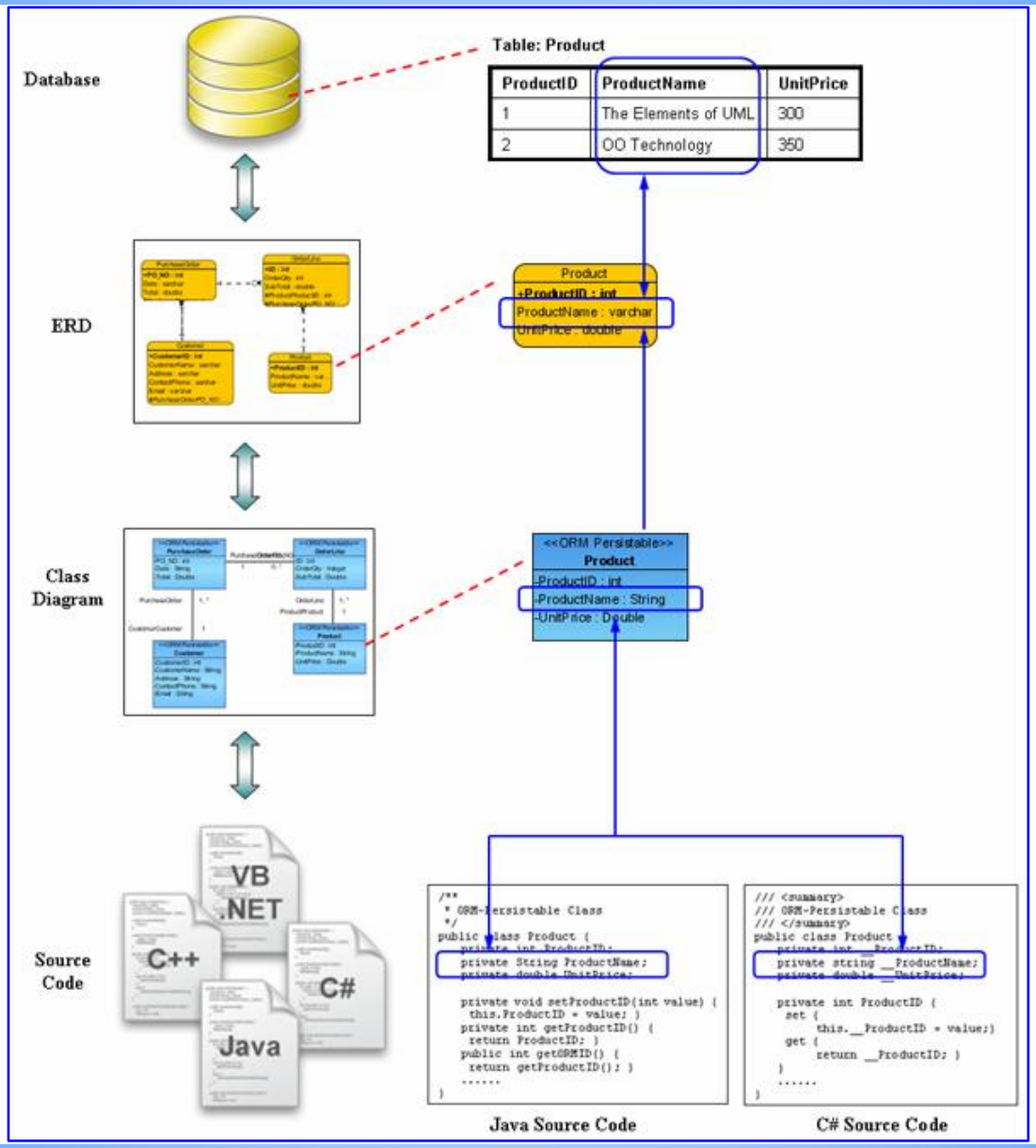
Eric Gustavo Coronel Castillo
gcoronelc@gmail.com

Ricardo Walter Marcelo Villalobos
ricardomarcelo@hotmail.com

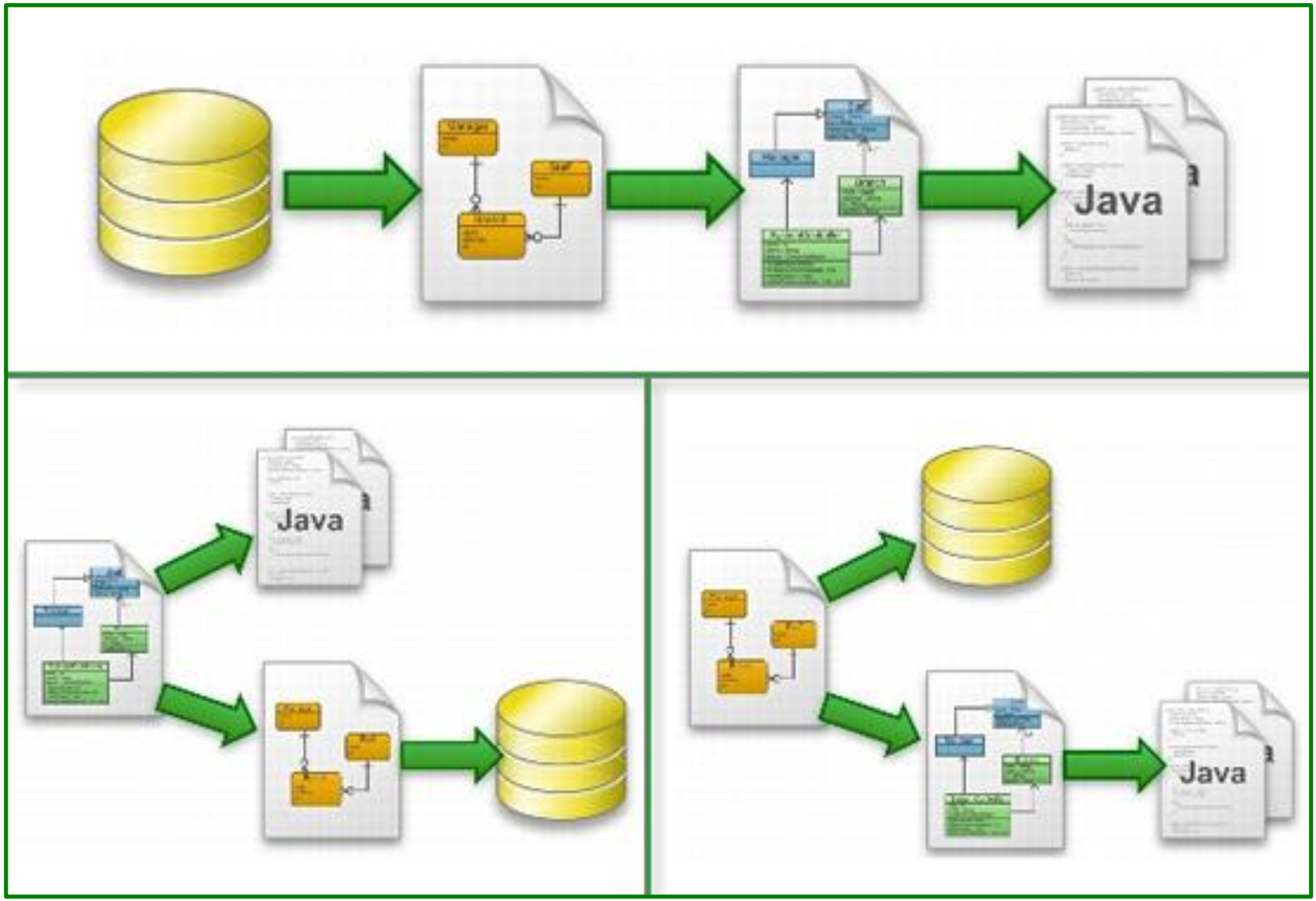
INDICE

- Object Relational Mapping
- JPA (Java Persistence API)
- Entity: Mapeo de Objetos
- Entity Manager

Object Relational Mapping

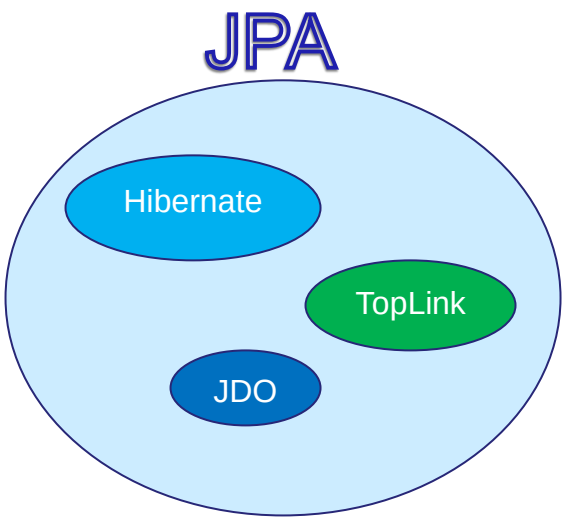


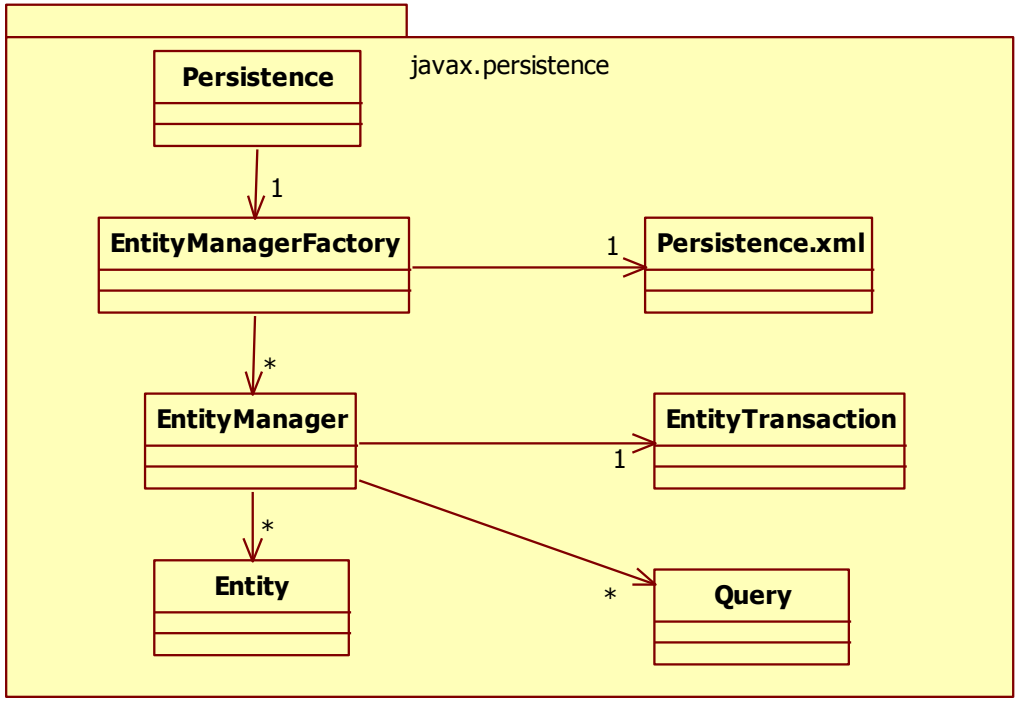
Object Relational Mapping



JPA (Java Persistence API)

Estándar Java para el mapeo objeto relacional (ORM), ha combinado ideas y conceptos de los principales frameworks de persistencia, como Hibernate, TopLink y JDO.





Entity: Mapeo de Objetos

- Clase POJO (Plain Old Java Object), serializable.
- Metadatos de mapeo:
  - Anotaciones en la propia clase (preferido)
  - Ficheros de mapeo xml externos
- Una entidad lleva anotado:
  - **@Entity**
  - **@Id** (al atributo que es Primary key en la BD)

Entity: Mapeo de Objetos

- Atributos que referencian a otras entidades (Foreign Key) mediante:
  - **@OneToOne**, **OneToMany**, **ManyToOne**, **ManyToMany**
- No es necesario anotar las tablas y las columnas de manera explicita (nombres por defecto)
- Anotaciones para sobreescribir nombres
  - **@Table**
  - **@Column**
  - **@JoinColumn**

Entity Manager

- **EntityManager** es la clase principal del JPA
  - Crear entidades
  - Crear consultas que devuelven entidades
  - Actualizar y borrar entidades
- Dos tipos de **EntityManger**
  - **Container-Managed:** Proporcionado por Service Provider Interface de JPA (mediante inyección de dependencia)

```
@PersistenceContext(unitName="VentasPU")
EntityManager em;
```

  - **Non-Managed:** Desde cualquier implementacion JPA, usando factorías.

```
import javax.persistence.*;
...
EntityManagerFactory emf =
     Persistence.createEntityManagerFactory("VentasPU");
EntityManager em = emf.createEntityManager();
```

Entity Manager

- **EntityManager** proporciona métodos **CRUD** sobre las entidades

```
Vendedor ven = new Vendedor("Juan Perez", 5000);
em.persist(ven);
```

  - ***persist(Object entity):*** Almacena el objeto *entity* en la base de datos.
  - ***merge(T entity):*** Actualiza las modificaciones en la entidad devolviendo la lista resultante.
  - ***remove(Object entity):*** Elima la entidad.
  - ***find(Class<T> entity, Object primaryKey):*** Busca la entidad a través de su clave primaria.
  - ***flush():*** Sincroniza las entidades con el contenido de la base de datos.
  - ***refresh(Object entity):*** Refresca el estado de la entidad con su contenido en la base de datos.
  - ***createQuery(String queryJPQL):*** Crea una query utilizando el lenguaje JPQL.
  - ***createNativeQuery(String querySQL):*** Crea una query utilizando el lenguaje SQL.
  - ***isOpen():*** Comprueba si está; abierto el EntityManager.
  - ***close():*** Cierra el EntityManager.

Transacciones

- Dos tipos de transacionalidad
  - **JTA container:** Administrado por el contenedor
  - **Resource Local:** Administrado programáticamente
- **EntityTransaction:** begin, commit, rollback

```
em.getTransaction().begin();
Vendedor ven = em.find(Vendedor.class, 1);
em.remove(ven);
em.getTransaction().commit();
```

Consultas

- Dos tipos principales de consultas con JPQL (OO):
  - **Dynamic Query:** Usa EntityManager

```
Query q = em.createQuery("select v from Vendedor v");
List<Vendedor> vendedores = q.getResultList();
```

  - **Named Query:** Definida estáticamente, instanciada y ejecutada.

```
@NamedQuery(name="Vendedor.traerTodo",
        query="select v from Vendedor v")
@Entity
public class Vendedor {
  ...
}
```

Configuración

- Unidad de persistencia y configuración de acceso a un origen de datos con JPA (**META-INF/persistence.xml**)

- 2 Tipos de acceso

  - **Managed Container** (JNDI – Pool de conexiones)

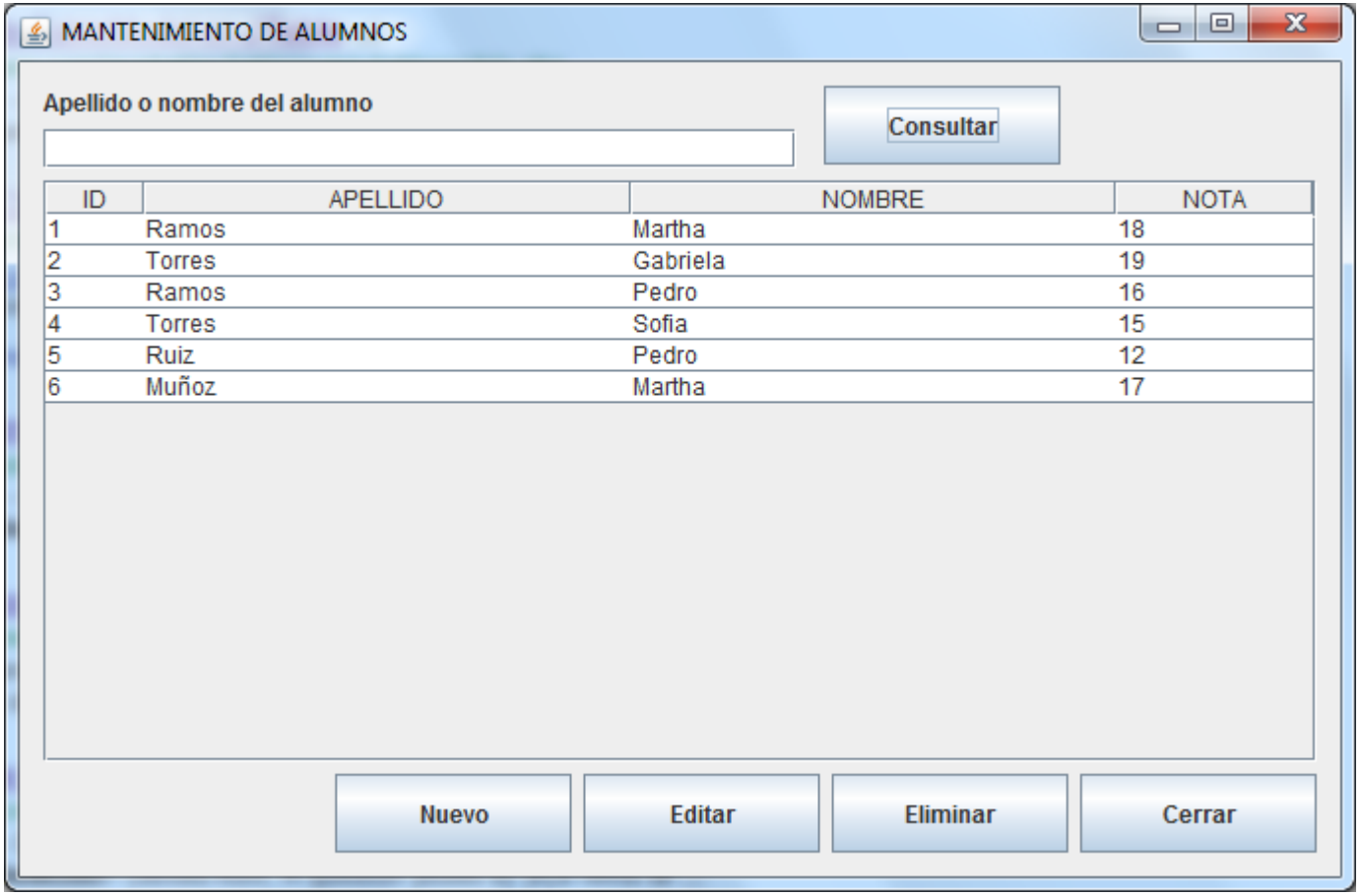
```
<persistence-unit name="VentasPU">
  <jta-data-source>jdbc/Ventas</jta-data-source>
</persistence-unit>
```

Configuración

- **Non-Managed** (Conexión directa)

```
<persistence-unit name="VentasPU">
  <class>com.ecommerce.model.entity.producto</class>
  <properties>
    <property name="hibernate.connection.username"
       value="root"/>
    <property name="hibernate.connection.driver_class"
       value="com.mysql.jdbc.Driver"/>
    <property name="hibernate.connection.password"
       value="adminadmin"/>
    <property name="hibernate.connection.url"
       value="jdbc:mysql://localhost/test"/>
    <property name="hibernate.hbm2ddl.auto" value="update"/>
</properties>
</persistence-unit>
```

Demo


MANTENIMIENTO DE ALUMNOS
Apellido o nombre del alumno
Consultar
ID
APELLIDO
NOMBRE
NOTA
1 Ramos Martha 18
2 Torres Gabriela 19
3 Ramos Pedro 16
4 Torres Sofia 15
5 Ruiz Pedro 12
6 Muñoz Martha 17
Nuevo
Editar
Eliminar
Cerrar

Ejercicio

- Desarrolle una aplicación que permita darle mantenimiento a la tabla de artículos.
- Su estructura es la siguiente:

IdArticulo	Código de articulo, es auto numérico.
Nombre	Nombre del artículo.
PreCosto	Precio de costo.
PreVenta	Precio de venta.
Stock	Stock.

Gracias . . .