CURSO DE HIBERNATE 5

OpenWebinars

HIBERNATE

(1) INTRODUCCION

Persistencia, desfase objecto-relacional, ORM. Productos y estándares

(2) HIBERNATE

Más que un ORM. Comparativa con otros productos. JPA. Maven. Módulos

(3) PRIMER PROYECTO

Hibernate.cfg.xml, EntityManager y persistence.xml

(4) ENTIDADES

Definición del modelo del dominio. Entidades y ciclo de vida. XML y anotaciones. Tipos de datos.

(5) ASOCIACIONES

ManyToOne, OneToMany, OneToOne, ManyToMany

HIBERNATE

(6) ELEMENTOS AVANZADOS

Campos calculados, herencia.

(7) COLECCIONES

Mapeo de colecciones. Tipos (list, set, map). Colecciones ordenadas (sorted vs. ordered).

(8) GENERACION DEL ESQUEMA

Customización del proceso de generación del esquema.

(9) CONTEXTO DE PERSISTENCIA

Almacenamiento, recuperación y borrado de entidades.

(10) TRANSACCIONES

Control de concurrencia. Patrones y antipatrones.

HIBERNATE

(12)
ENVERS

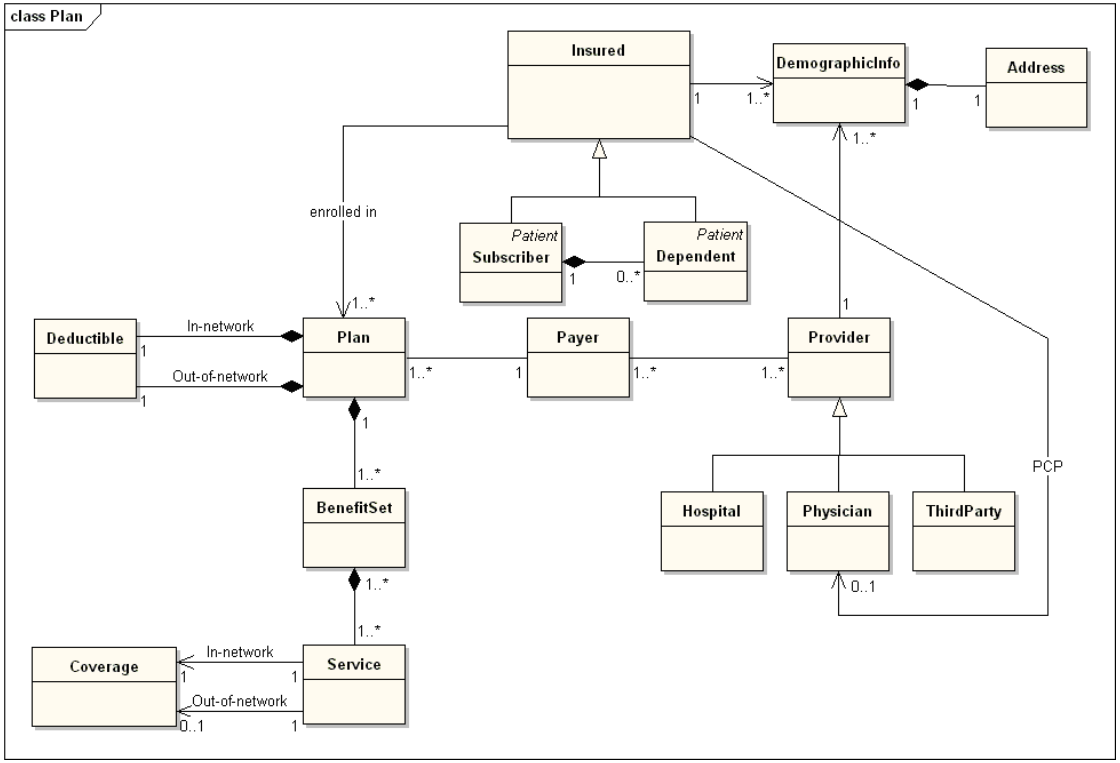
Introducción a la
auditoria de entidades.

(11)
CONSULTAS
HPQL VS JPQL

Consultas con
parámetros,
Anotaciones. SQL nativo

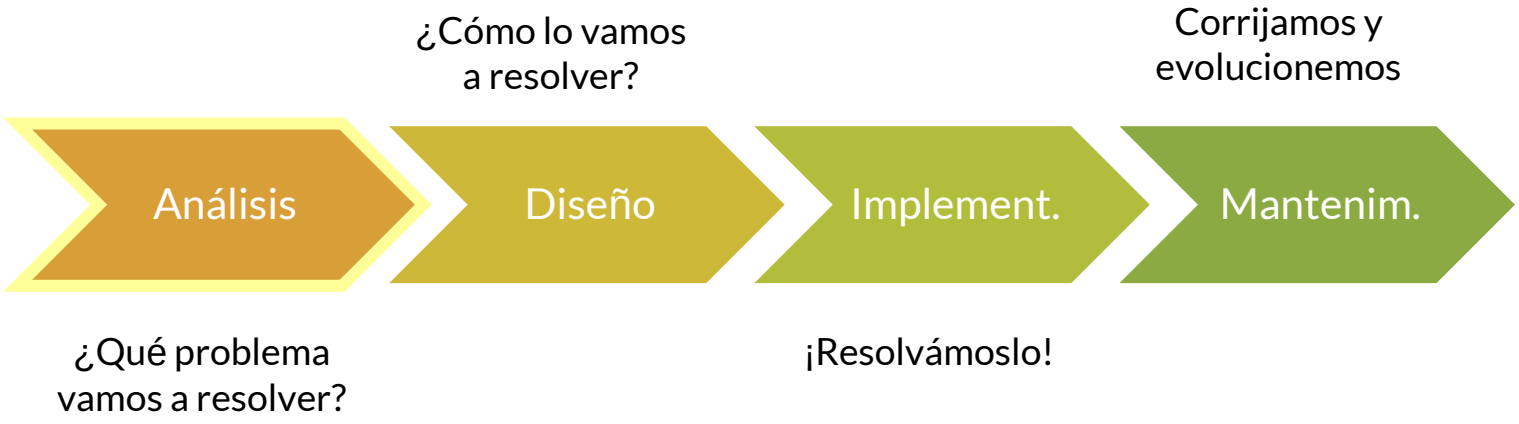
1.

MODELO DE DOMINIO



Clases que representan el problema (sistema) que estamos tratando de resolver.

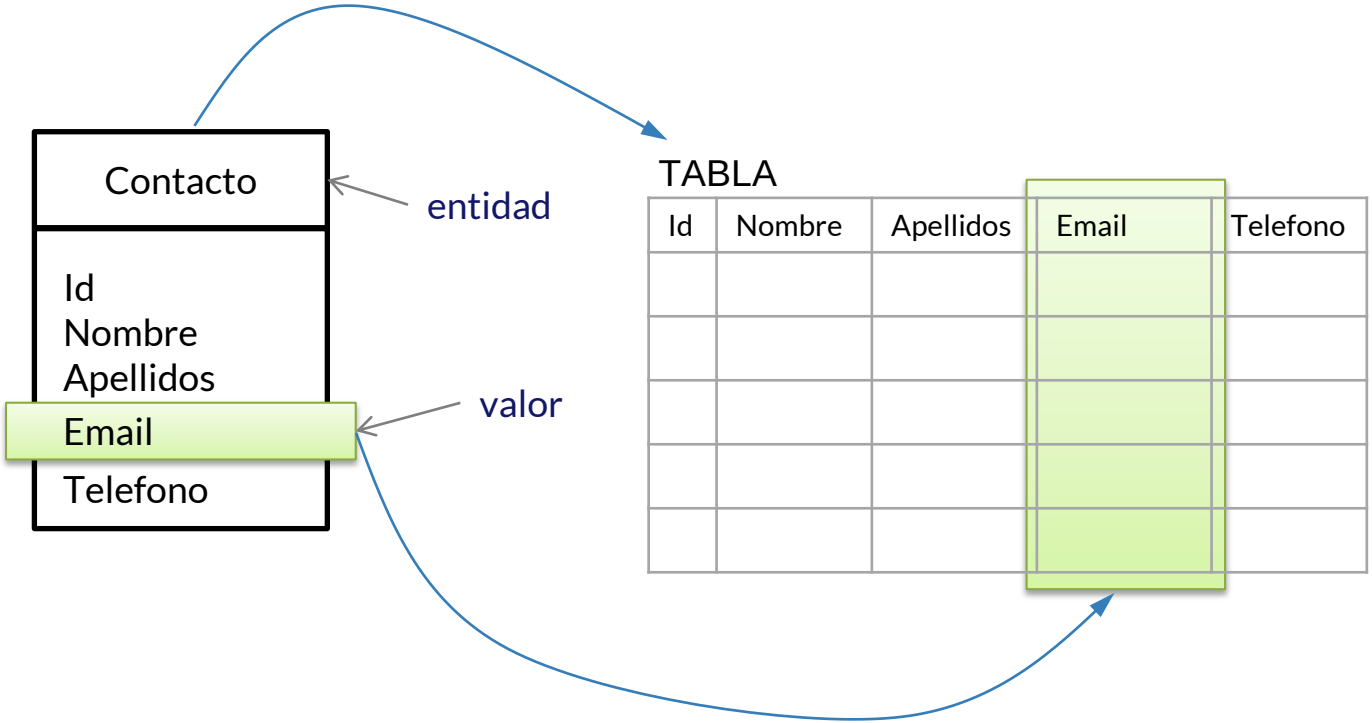
FASES DEL DESARROLLO DE SOFTWARE



2.

VALORES Y ENTIDADES

ENTIDADES Y VALORES



ENTIDADES Y VALORES

Valores

- Tipos básicos
- Tipos *embeddable*
- Tipos *collection*

Entidades

- Representan una clase del modelo.
- Tienen ciclo de vida.

3.

MAPEO DE ENTIDADES

ENTIDADES IDENTIFICADAS

Todas las entidades JPA deben tener un identificador.

Identidad

- Asociado a ==
- Misma representación en memoria.

Igualdad

- Asociada a *equals()*.
- También llamada equivalencia.
- Estados *equivalentes*

Identidad BD

- Asociado a PK
- Dos objetos son idénticos si están en la misma tabla y tienen la misma PK.

ANOTACIONES MÍNIMAS

- *@Entity*
  Anotamos sobre la clase.
- *@Id*
  Podemos anotar sobre propiedad o sobre método *getter*.

```
@Entity
public class MyEntity {

    @Id
    private long id;


    public long getId() {
        return id;
    }

}
```

ELECCIÓN DEL IDENTIFICADOR (CLAVE PRIMARIA)

- Es habitual utilizar campos *artificiales*, con el mismo nombre (*id)* y de tipo entero (*long*).
- JPA se puede encargar de generarlo (*@GeneratedValue*).
  - AUTO: Hibernate escoge la estrategia.
  - SEQUENCE: se utiliza una secuencia
  - IDENTITY: se utiliza un campo autonumérico
  - TABLE: se utiliza una tabla extra especial.

CONTROL DE NOMBRES

- *@Entity*

La tabla se llamará igual que la clase.

- *@Entity*
  @Table(name="…")

Podemos modificar el nombre de la tabla generada.

```
@Entity
@Table(name= "MY_ENT")
public class MyEntity {

    @Id
    private long id;


    public long getId() {
        return id;
    }

}
```

4.

MAPEO DE VALORES

TIPOS DE DATOS MAPEADOS

- Tipos básicos
- Envoltorios de tipos básicos (*Integer*, *Double*, *…)*
- *String*, *BigInteger*, *BigDecimal*, *java.util.Date*, *java.util.Calendar*, *java.sql.Date*, *java.sql .Time*, *java.sql.Timestamp*, *byte[]*, *Byte[]*, *char[]*, o *Character[]*
- *java.io.Serializable* (representación en bytes)*
- *@Embeddable*

El resto de tipos de datos generarán por defecto un error.

ANOTACIÓN @Column

- Por defecto JPA (Hibernate) mapea todos los atributos (o *getter*).
- *@Column* nos permite definir algunas propiedades
  - *Nullable:* si permite almacenar nulos
  - *Name*: nombre de la columna en la BD
  - *Insertable, updatable:* define si la entidad puede ser o no insertable o actualizable.
  - *Length:* tamaño que tendrá el campo en la BD.

TIPOS DE DATOS TEMPORALES

- *@Temporal*
- *java.util.Date, java.util.Calendar, java.sql.Date, java.sql.Time, java.util.Timestamp*
- *java.time* de JDK 8
- Propiedad TemporalType (DATE, TIME, TIMESTAMP).

```java
@Entity
@Table(name= "MY_ENT")
public class MyEntity {

    @Id
    private long id;

  @Temporal(TemporalType.TIMESTAMP)
  private Date createdOn;

  //...

}
```

5.

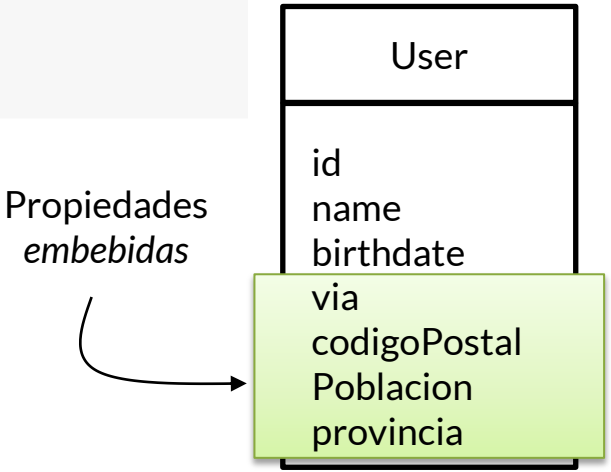
TIPOS EMBEBIDOS

TIPOS DE DATOS @Embeddable

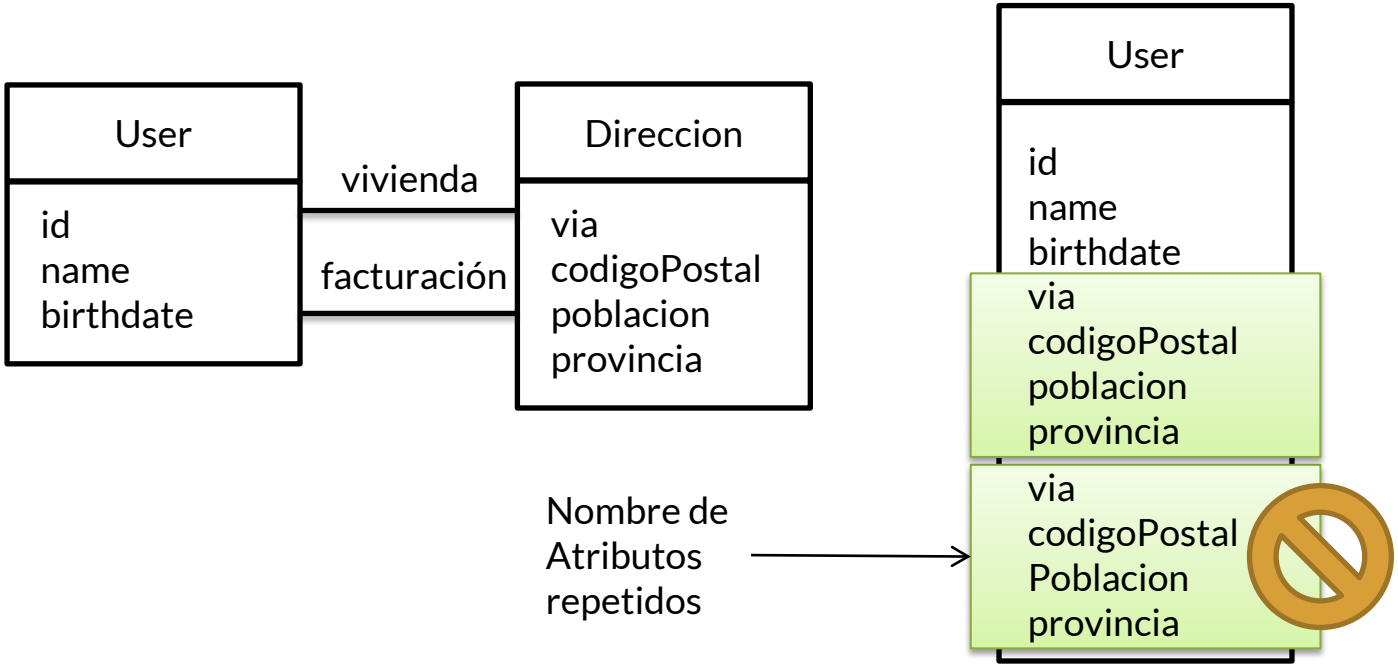
```java
@Embeddable
public class Direccion {

  @Column(nullable=false)
  private String via;

  @Column(nullable=false, length = 5)
  private String codigoPostal;

  @Column(nullable=false)
  private String poblacion;

  @Column(nullable=false)
  private String provincia;

  //...


}
```

```java
@Entity
@Table(name="USERCONEMBEDD")
public class User {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private long id;

    private String name;

  @Temporal(TemporalType.DATE)
    private Date birthDate;

  private Direccion address;

    //...
}
```

- La clase embebida no es una entidad, pero podemos tratarla como objeto (*null*).



SOBRESCRITURA CON @Embedded



6.

CICLO DE VIDA DE LAS ENTIDADES

CICLO DE VIDA DE UNA ENTIDAD