Licenciatura en Sistemas - Orientación a Objetos II – 2024

Prof. Titular: Mg. María Alejandra Vranić

Prof. Ayudantes: Esp. Lic. Gustavo Siciliano
Lic. Ezequiel Scordamaglia
Lic. Oscar Ruina

Índice

DER y Diagrama de clases

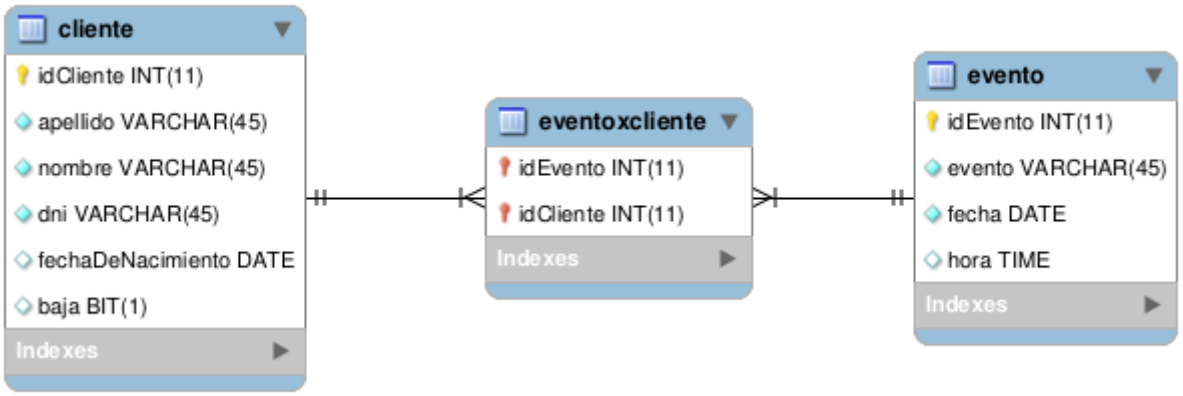

Diagrama entidad relación

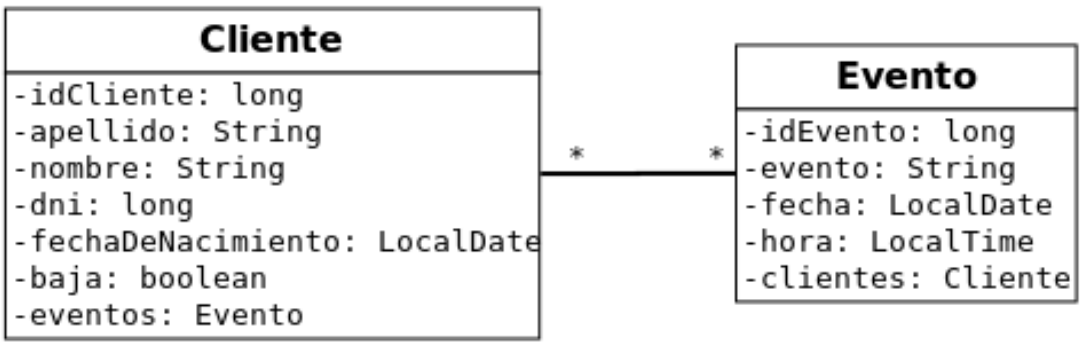

Diagrama de clases

Mapeos de Cliente y Evento

Cliente.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
<class name="datos.Cliente" table="cliente">
    <id column="idCliente" name="idCliente">
      <generator class="identity"/>
    </id>
    <property column="apellido" name="apellido" type="string"/>
    <property column="nombre" name="nombre" type="string"/>
    <property column="dni" name="dni" type="int"/>
    <property column="fechaDeNacimiento" name="fechaDeNacimiento" type="LocalDate"/>
    <property column="baja" name="baja" type="boolean"/>
    <set table="eventoxcliente" name="eventos" outer-join="true">
       <key column="idCliente"/>
       <many-to-many column="idEvento" class="datos.Evento"/>
       </set>
 </class>
</hibernate-mapping>
```

Evento.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="datos.Evento" table="evento">
    <id column="idEvento" name="idEvento">
      <generator class="identity"/>
    </id>
    <property column="evento" name="evento" type="string"/>
    <property column="fecha" name="fecha" type="LocalDate"/>
    <set table="eventoxcliente" name="clientes" outer-join="true">
       <key column="idEvento"/>
    <many-to-many column="idCliente" class="datos.Cliente"/>
    </set>
</class>
</hibernate-mapping>
```

Lo único nuevo en estos archivos es el tag de set. Además, no hay que olvidarse de agregar los mapeos y seleccionar la base de datos correcta en el archivo hibernate.cfg.xml.

Clase Cliente.java

```java
package datos;
import java.time.LocalDate;
import java.util.Iterator;
import java.util.Set;
import negocio.Funciones;

public class Cliente {
	private long idCliente;
	private String apellido;
	private String nombre;
	private int dni;
	private LocalDate fechaDeNacimiento;
	private boolean baja;
	private Set<Evento> eventos;

	public Cliente(){}

	public Cliente(String apellido, String nombre, int dni, LocalDate fechaDeNacimiento) {
		this.apellido = apellido;
		this.nombre = nombre;
		this.dni= dni;
		this.fechaDeNacimiento = fechaDeNacimiento;
		this.baja = false;
	}

	public long getIdCliente() {
		return idCliente;
	}
	protected void setIdCliente(long idCliente) {
		this.idCliente = idCliente;
	}
	public String getApellido() {
		return apellido;
	}
	public void setApellido(String apellido) {
		this.apellido = apellido;
	}
	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public int getDni() {
		return dni;
	}
	public void setDni(int dni) {
		this.dni = dni;
	}
	public LocalDate getFechaDeNacimiento() {
		return fechaDeNacimiento;
	}
	public void setFechaDeNacimiento(LocalDate fechaDeNacimiento) {
		this.fechaDeNacimiento = fechaDeNacimiento;
	}
	public boolean isBaja() {
		return baja;
	}
	public void setBaja(boolean baja) {
		this.baja = baja;
	}
	public Set<Evento> getEventos() {
		return eventos;
	}
	protected void setEventos(Set<Evento> eventos) {
		this.eventos = eventos;
	}
```

```
	@Override
	public int hashCode() {
		return Objects.hash(idCliente);
	}
	@Override
	public boolean equals(Object obj) {
		Cliente other = (Cliente) obj;
		return idCliente == other.idCliente;
	}

	public boolean agregar(Evento evento){
		boolean agregar=false;
		if (! (eventos.contains(evento))) {
			agregar=eventos.add(evento);
		}
		return agregar;
	}

	public boolean eliminar(Evento evento){
		Evento borrar = null;
		boolean eliminar = false;
		Iterator<Evento> it = eventos.iterator();
		while ((it.hasNext()) && (borrar==null)){
			Evento e = it.next();
			if (e.equals(evento)) borrar = e;
		}
		eliminar=eventos.remove(borrar);
		return eliminar;
	}

	public String toString() {
		return "Cliente [idCliente=" + idCliente + ", apellido=" + apellido + ", nombre=" + nombre +
", dni=" + dni + ", fechaDeNacimiento=" + fechaDeNacimiento + ", baja=" + baja + "]";
	}

}
```

Crear el método de actualizar y consumir todos los métodos desde la capa de negocio. ¿Qué cambios se podrían agregar al consumir los métodos desde la capa de negocio?

Clase Evento.java

```
package datos;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Iterator;
import java.util.Set;

public class Evento {

	private long idEvento;
	private String evento;
	private LocalDate fecha;
	private LocalTime hora;
	private Set<Cliente> clientes;

	public Evento(){}

	public Evento(String evento, LocalDate fecha, LocalTime hora, Set<Cliente> clientes) {
		this.evento = evento;
		this.fecha = fecha;
		this.hora = hora;
		this.clientes = clientes;
	}

	public long getIdEvento() {
		return idEvento;
	}
	protected void setIdEvento(long idEvento) {
		this.idEvento = idEvento;
	}
	public String getEvento() {
		return evento;
	}
	public void setEvento(String evento) {
		this.evento = evento;
	}
	public LocalDate getFecha() {
		return fecha;
	}
	public void setFecha(LocalDate fecha) {
		this.fecha = fecha;
	}

	public LocalTime getHora() {
		return hora;
	}
	public void setHora(LocalTime hora) {
		this.hora = hora;
	}
	public Set<Cliente> getClientes() {
		return clientes;
	}
	public void setClientes(Set<Cliente> clientes) {
		this.clientes = clientes;
	}

	public boolean equals(Evento evento){
		return (idEvento==evento.getIdEvento());
	}

	public boolean agregar(Cliente cliente){
		boolean agregar=false;
		if (! (clientes.contains(cliente))) {
		agregar=clientes.add(cliente);
		}
		return agregar;
	}
```

```
  public boolean eliminar(Cliente cliente){
            Cliente borrar=null;
            boolean eliminar=false;
            Iterator<Cliente> it = clientes.iterator();
            while ((it.hasNext()) && (borrar==null)){
                        Cliente c=it.next();
                         if (c.equals(cliente) ) borrar=c;
            }
            eliminar=clientes.remove(borrar);
            return eliminar;
}

@Override
public String toString() {
            return "Evento [idEvento=" + idEvento + ", evento=" + evento + ", fecha=" + fecha + ", hora=" + hora + "]";
}
}
```

Al igual que con la clase anterior, crear el método de actualizar y consumir todos los métodos desde la capa de negocio. ¿Qué cambios se podrían agregar al consumir los métodos desde la capa de negocio?

Métodos nuevos en ClienteDao

```
public Cliente traerClienteYEventos(long idCliente) {
    Cliente objeto = null;
    try {
        iniciaOperacion();
        String hql = "from Cliente c where c.idCliente =:idCliente";
        objeto = (Cliente) session.createQuery(hql).setParameter("idCliente", idCliente)
                .uniqueResult();
        Hibernate.initialize(objeto.getEventos());
    } finally {
        session.close();
    }
    return objeto;
}
```

Desarrollo pendiente

1. Desarrollar los comentarios en las clases de datos Cliente y Evento.
2. Crear las clases de ABM para Cliente y Evento.
3. Crear las clases de DAO para Cliente y Evento.
4. Testear los métodos para cada clase:
    a. agregar.
    b. traer por id.
    c. traer por id con la información de la lista relacionada.
    d. traer todos.
    e. actualizar.
    f. eliminar.