

# **CURSO HIBERNATE Y JPA**

## **EJERCICIO**

### **API DE CRITERIA LAZY E EAGER LOADING**



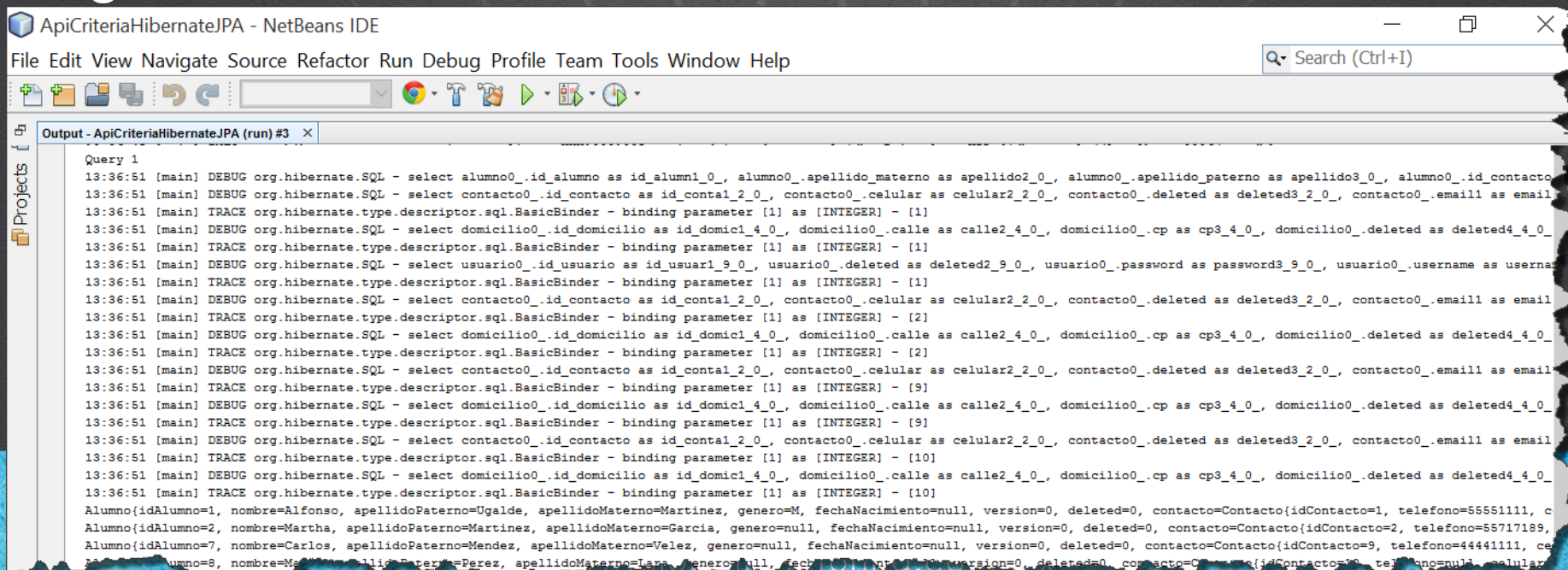
Experiencia y Conocimiento para tu vida

**CURSO HIBERNATE Y JPA**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# OBJETIVO DEL EJERCICIO

Utilizar el API de Criteria para poner en práctica el concepto de Lazy e Eager Loading. Al finalizar deberemos observar lo siguiente:



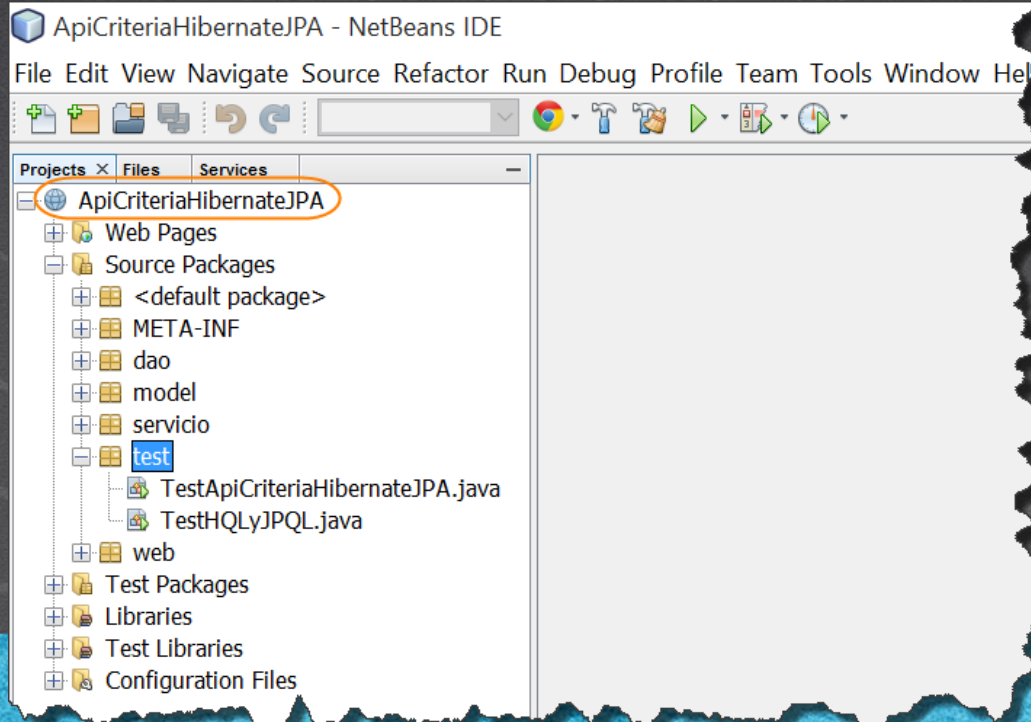
The screenshot shows the NetBeans IDE interface with the title bar 'ApiCriteriaHibernateJPA - NetBeans IDE'. The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. A search bar on the right contains 'Search (Ctrl+I)'. The toolbar shows various icons for file operations and running. The 'Output - ApiCriteriaHibernateJPA (run) #3' window is active, displaying a log of Hibernate SQL queries and trace messages. The log starts with 'Query 1' and shows several SQL select statements for fetching 'alumno0', 'domicilio0', and 'usuario0' entities, along with their associated 'contacto0' entities. The log also includes trace messages for the 'BasicBinder' binding parameters. At the bottom, the log shows the initialization of three 'Alumno' objects with their respective attributes.

```
Query 1
13:36:51 [main] DEBUG org.hibernate.SQL - select alumno0_.id_alumno as id_alumn1_0_, alumno0_.apellido_materno as apellido2_0_, alumno0_.apellido_paterno as apellido3_0_, alumno0_.id_contacto
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email1 as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select usuario0_.id_usuario as id_usuari_9_0_, usuario0_.deleted as deleted2_9_0_, usuario0_.password as password3_9_0_, usuario0_.username as usern
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email1 as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [2]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [2]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email1 as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [9]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [9]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email1 as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [10]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [10]
Alumno{idAlumno=1, nombre=Alfonso, apellidoPaterno=Ugalde, apellidoMaterno=Martinez, genero=M, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=1, telefono=55551111, c
Alumno{idAlumno=2, nombre=Martha, apellidoPaterno=Martinez, apellidoMaterno=Garcia, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=2, telefono=55717189,
Alumno{idAlumno=7, nombre=Carlos, apellidoPaterno=Mendez, apellidoMaterno=Velez, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=9, telefono=44441111, ce
Alumno{idAlumno=8, nombre=Martha, apellidoPaterno=Perez, apellidoMaterno=Lara, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=10, telefono=null, celular
```



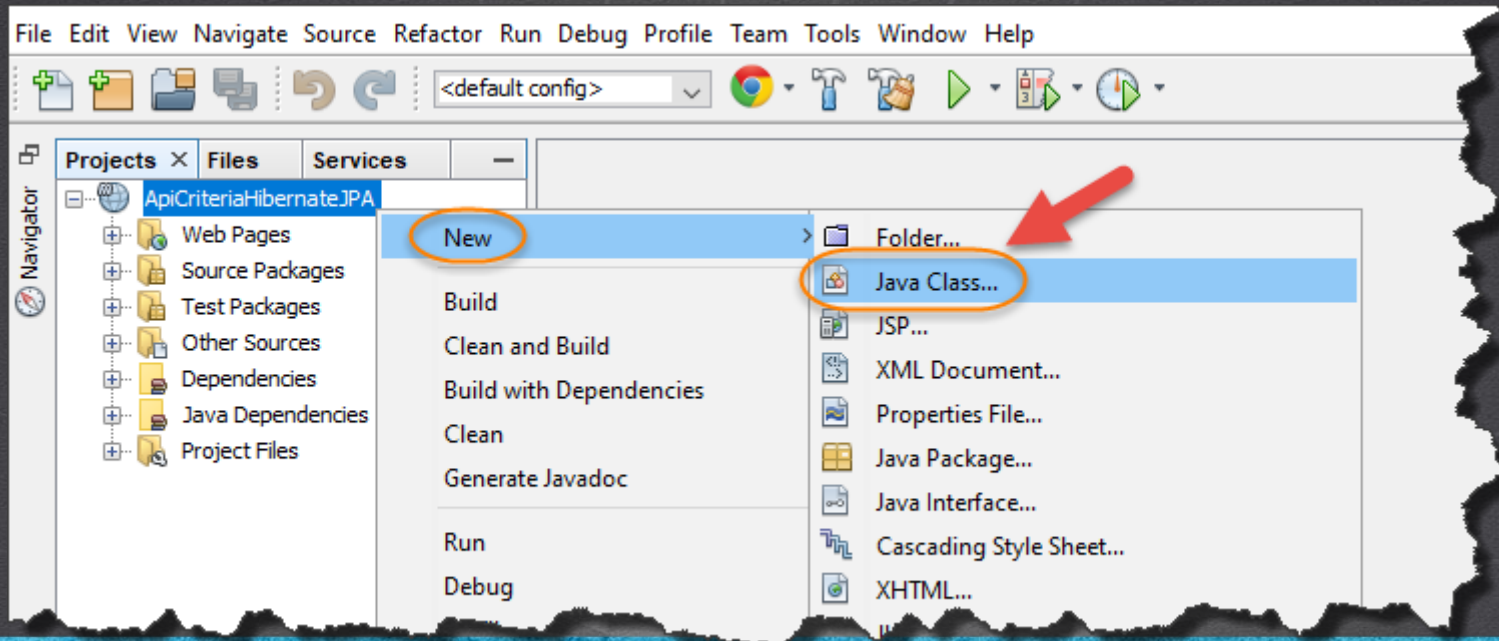
# PASO 1. USAMOS EL PROYECTO

Abrimos el proyecto ApiCriteriaHibernateJPA:



# PASO 2. CREAR UNA CLASE

Creamos la clase TestApiCriteriaLazyEagerLoading.java:



**CURSO HIBERNATE Y JPA**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 2. CREAR UNA CLASE

Creamos la clase TestApiCriteriaLazyEagerLoading.java:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:



# PASO 3. MODIFICAMOS EL CÓDIGO

## Archivo TestApiCriteriaLazyEagerLoading.java:

Clic para  
descargar el  
código

```
package test;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.EntityGraph;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Join;
import javax.persistence.criteria.JoinType;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import model.Alumno;
import model.Contacto;
import model.Domicilio;

public class TestApiCriteriaLazyEagerLoading {

    public static void main(String[] args) {
        /*Utilizamos la Unidad de Persistencia de JPA*/
        EntityManagerFactory fabrica = Persistence.createEntityManagerFactory("HibernateJpaPU");
        EntityManager em = fabrica.createEntityManager();

        //Variables de ayuda
        CriteriaBuilder cb = em.getCriteriaBuilder();
        List<Alumno> alumnos = null;
```

# PASO 3. MODIFICAMOS EL CÓDIGO

## Archivo TestApiCriteriaLazyEagerLoading.java:

Clic para  
descargar el  
código

```
//Por default los queries son tipo lazy
//es decir no levantan las relaciones
//Left Join con API Criteria
//Query 1
System.out.println("\nQuery 1");
CriteriaQuery<Alumno> qb1 = cb.createQuery(Alumno.class);
Root<Alumno> c1 = qb1.from(Alumno.class);
c1.join("domicilio", JoinType.LEFT);
c1.join("contacto", JoinType.LEFT);

alumnos = em.createQuery(qb1).getResultList();
imprimirAlumnos(alumnos);
```

```
//1.JPQL equivalente
from Alumno a LEFT JOIN a.domicilio d LEFT JOIN a.contacto c
```

# PASO 3. MODIFICAMOS EL CÓDIGO

## Archivo TestApiCriteriaLazyEagerLoading.java:

Clic para  
descargar el  
código

```
//Query 2
System.out.println("\nQuery 2");
//Definimos el query
CriteriaQuery<Alumno> qb2 = cb.createQuery(Alumno.class);
//Definimos la raiz del query
Root<Alumno> c2 = qb2.from(Alumno.class);
//Especificamos el join
Join<Alumno, Domicilio> dom = c2.join("domicilio");
//De manera opcional agregamos la restriccion usando el join
qb2.where(cb.equal(dom.<Integer>get("idDomicilio"), 1));
//Definimos un Entity Graph para especificar el Fetch join
EntityGraph<Alumno> fetchGraph = em.createEntityGraph(Alumno.class);
//Especificamos la relación a levantar de manera eager (anticipada)
fetchGraph.addSubgraph("domicilio");
//loadgraph agrega lo definimo más lo que ya se especificó en la clase de entidad
//fetchgraph ignora lo definido en la clase de entidad y agrega solo lo nuevo
em.createQuery(qb2).setHint("javax.persistence.loadgraph", fetchGraph);

TypedQuery<Alumno> q2 = em.createQuery(qb2);
alumnos = q2.getResultList();
imprimirAlumnos(alumnos);
```

//2.JPQL equivalente

```
SELECT a FROM Alumno a JOIN FETCH a.domicilio d where d.idDomicilio = 1
```



# PASO 3. MODIFICAMOS EL CÓDIGO

## Archivo TestApiCriteriaLazyEagerLoading.java:

Clic para  
descargar el  
código

```
// Query 3
System.out.println("\nQuery 3");
CriteriaQuery<Alumno> qb3 = cb.createQuery(Alumno.class);
Root<Alumno> c3 = qb3.from(Alumno.class);
Join<Alumno, Domicilio> domicilio = c3.join("domicilio");
Join<Alumno, Contacto> contacto = c3.join("contacto");
List<Predicate> conditions = new ArrayList();
Integer idAlumno = 1;
conditions.add(cb.equal(c3.get("idAlumno"), idAlumno));
conditions.add(cb.isNotNull(domicilio.get("calle")));
conditions.add(cb.isNotNull(contacto.get("telefono")));

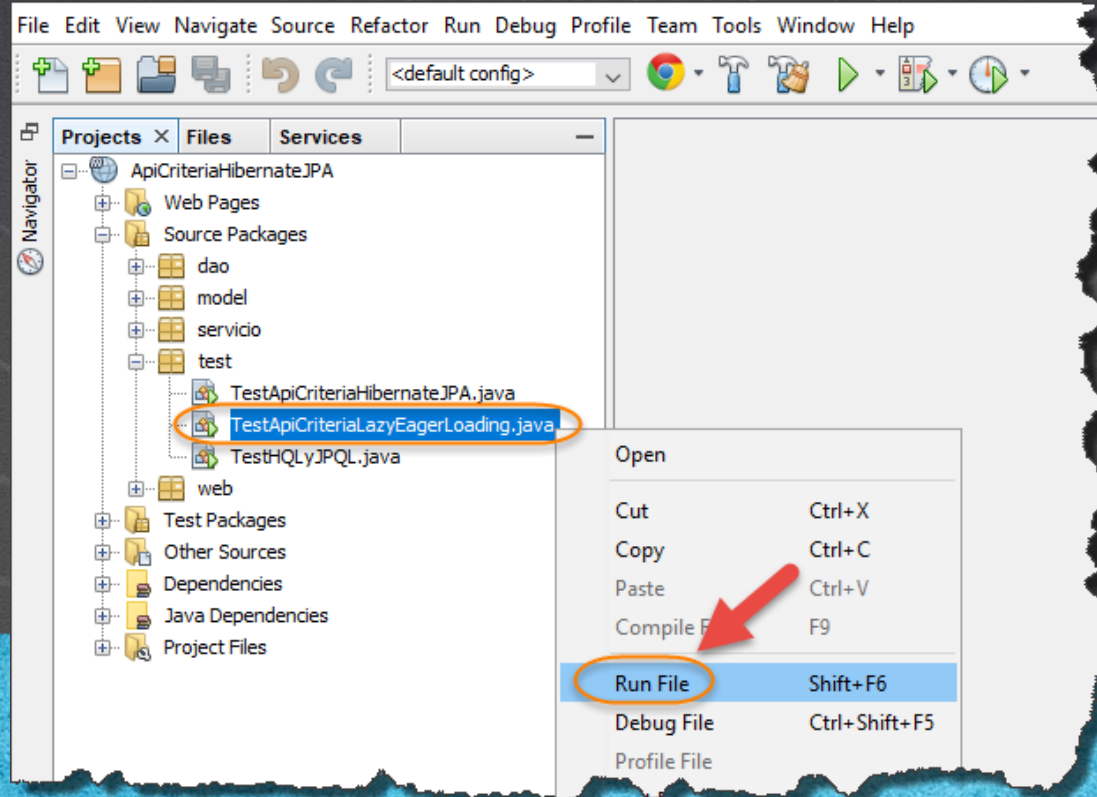
TypedQuery<Alumno> q3 = em.createQuery(
    qb3
    .select(c3)
    .where(conditions.toArray(new Predicate[]{}))
    .orderBy(cb.asc(c3.get("idAlumno")))
    .distinct(true)
);
imprimirAlumnos(q3.getResultList());
}
```

//3.JPQL equivalente

```
SELECT distinct a FROM Alumno a JOIN FETCH a.domicilio d JOIN FETCH a.contacto c
where a.idAlumno = 1 and d.calle is not null and c.telefono is not null order by a.idAlumno asc
```

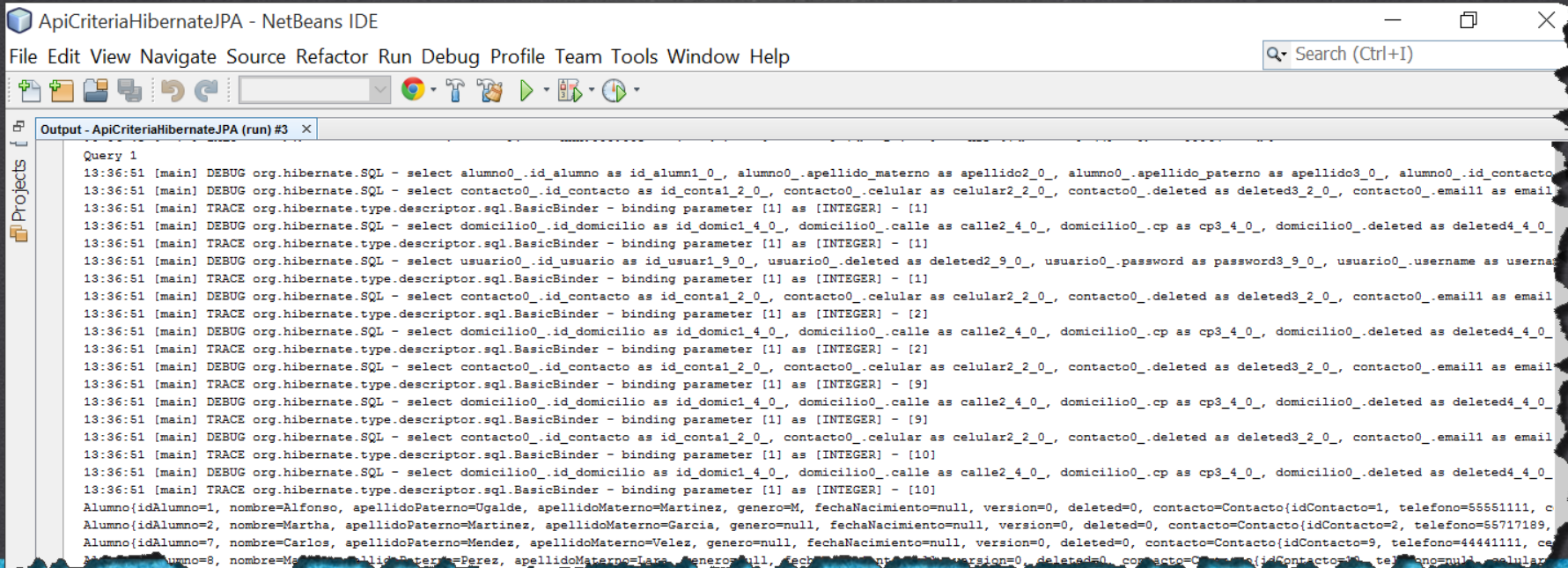
# PASO 4. EJECUTAMOS EL PROYECTO

Ejecutamos cada una de los queries del proyecto:



# PASO 4. EJECUTAMOS EL PROYECTO

Ejecutamos cada una de los queries del proyecto:



```
Output - ApiCriteriasHibernateJPA (run) #3 x
Query 1
13:36:51 [main] DEBUG org.hibernate.SQL - select alumno0_.id_alumno as id_alumn1_0_, alumno0_.apellido_materno as apellido2_0_, alumno0_.apellido_paterno as apellido3_0_, alumno0_.id_contacto
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select usuario0_.id_usuario as id_usuar1_9_0_, usuario0_.deleted as deleted2_9_0_, usuario0_.password as password3_9_0_, usuario0_.username as usern
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [2]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [2]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [9]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [9]
13:36:51 [main] DEBUG org.hibernate.SQL - select contacto0_.id_contacto as id_conta1_2_0_, contacto0_.celular as celular2_2_0_, contacto0_.deleted as deleted3_2_0_, contacto0_.email as email
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [10]
13:36:51 [main] DEBUG org.hibernate.SQL - select domicilio0_.id_domicilio as id_domic1_4_0_, domicilio0_.calle as calle2_4_0_, domicilio0_.cp as cp3_4_0_, domicilio0_.deleted as deleted4_4_0_
13:36:51 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [10]
Alumno{idAlumno=1, nombre=Alfonso, apellidoPaterno=Ugalde, apellidoMaterno=Martinez, genero=M, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=1, telefono=55551111, c
Alumno{idAlumno=2, nombre=Martha, apellidoPaterno=Martinez, apellidoMaterno=Garcia, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=2, telefono=55717189,
Alumno{idAlumno=7, nombre=Carlos, apellidoPaterno=Mendez, apellidoMaterno=Velez, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=9, telefono=44441111, ce
Alumno{idAlumno=8, nombre=Martha, apellidoPaterno=Perez, apellidoMaterno=Lara, genero=null, fechaNacimiento=null, version=0, deleted=0, contacto=Contacto{idContacto=10, telefono=null, celular
```



# EJERCICIOS EXTRA

- Ejecutar cada una de las consultas y compararlas con su equivalente de HQL/JPQL para empezar a detectar cuando es más conveniente utilizar una u otra forma de creación de consultas, ya sea con HQL/JPQL o con el API de Criteria de Hibernate/JPA.



Experiencia y Conocimiento para tu vida

**CURSO HIBERNATE Y JPA**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos ejecutado varias de las consultas con el API de Criteria de Hibernate/JPA aplicando el concepto de Fetch o Eager Loading.
- Por default las consultas son de tipo Lazy en Hibernate/JPA, sin embargo hemos visto cómo hacer que se carguen las relaciones de manera anticipada (eager) aplicando el concepto de Fetch a las consultas.
- Con esto ya podemos comparar y decidir si utilizamos el lenguaje de HQL/JPQL o el API de Criteria.
- Cada uno tiene sus ventajas y desventajas, pero todo dependerá de lo que necesitemos en nuestra aplicación para saber si utilizamos otra solución.



**CURSO ONLINE**

# **HIBERNATE & JPA**

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

**CURSO HIBERNATE Y JPA**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)