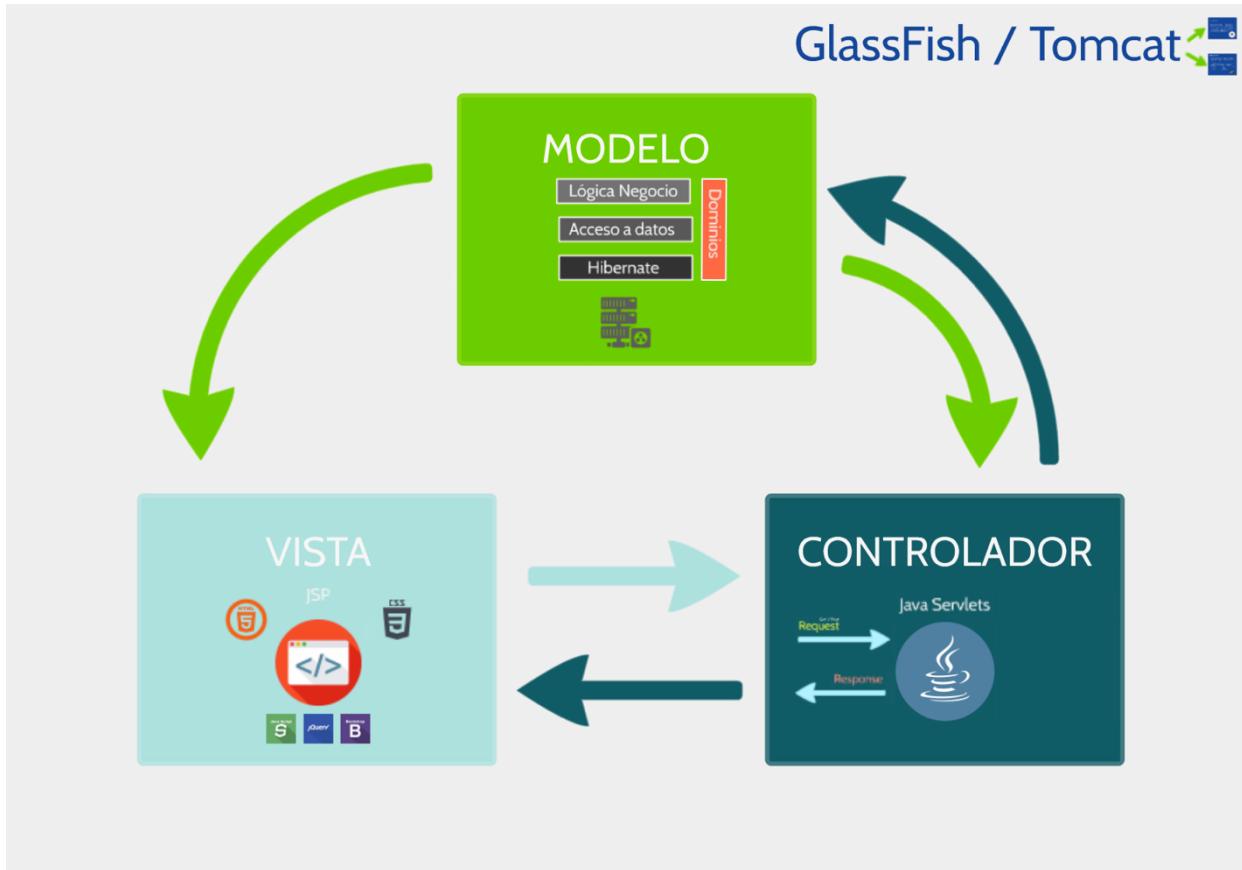


## Laboratorio de la capa DAO

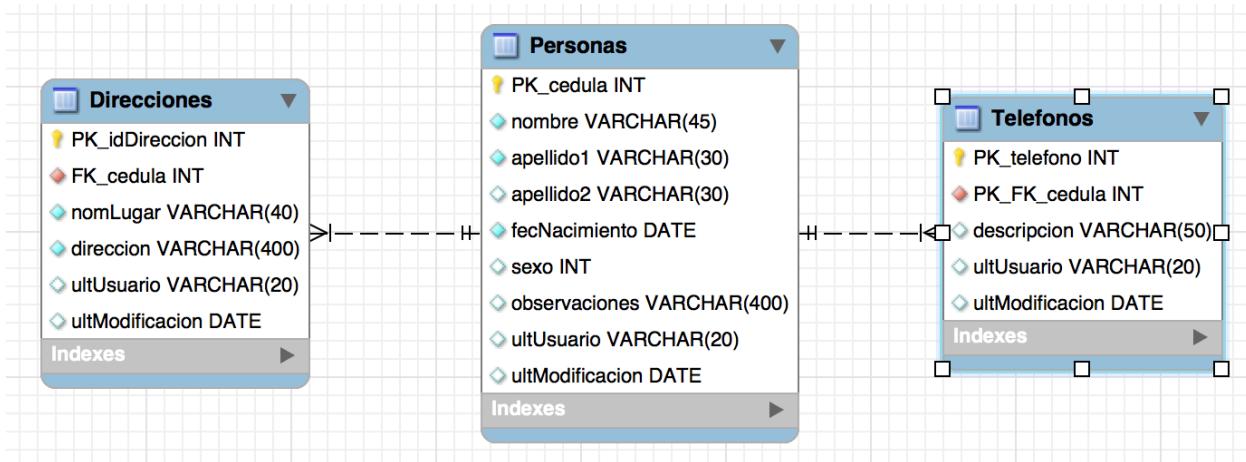
### Arquitectura

Para la elaboración de la arquitectura se deberán crear una serie de capas las cuales de exemplifican en el siguiente gráfico.

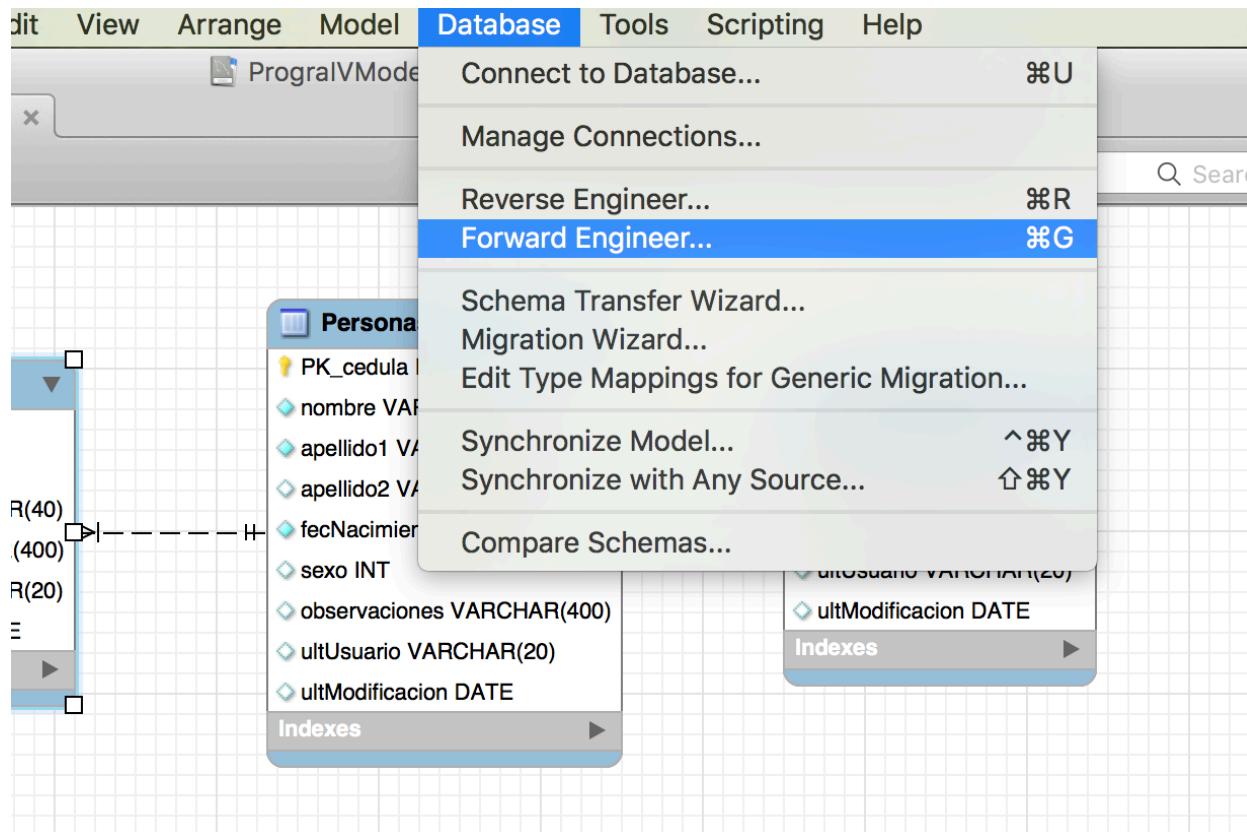


## Creación de la base de datos

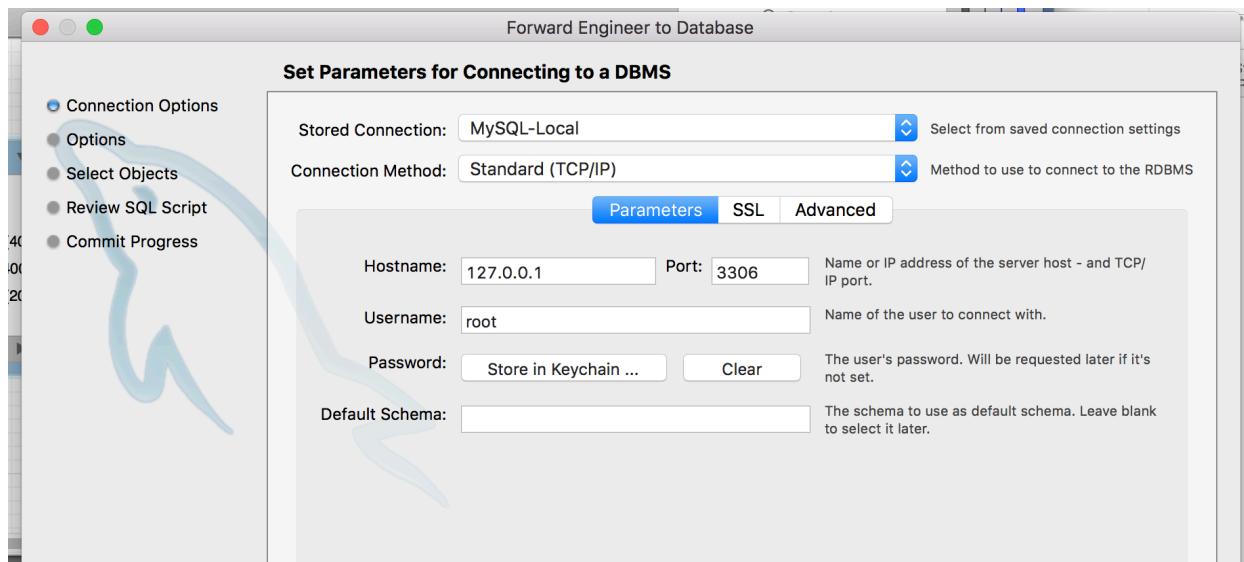
Para el laboratorio deberá crear la siguiente base de datos (esta se encuentra en el sitio del grupo)



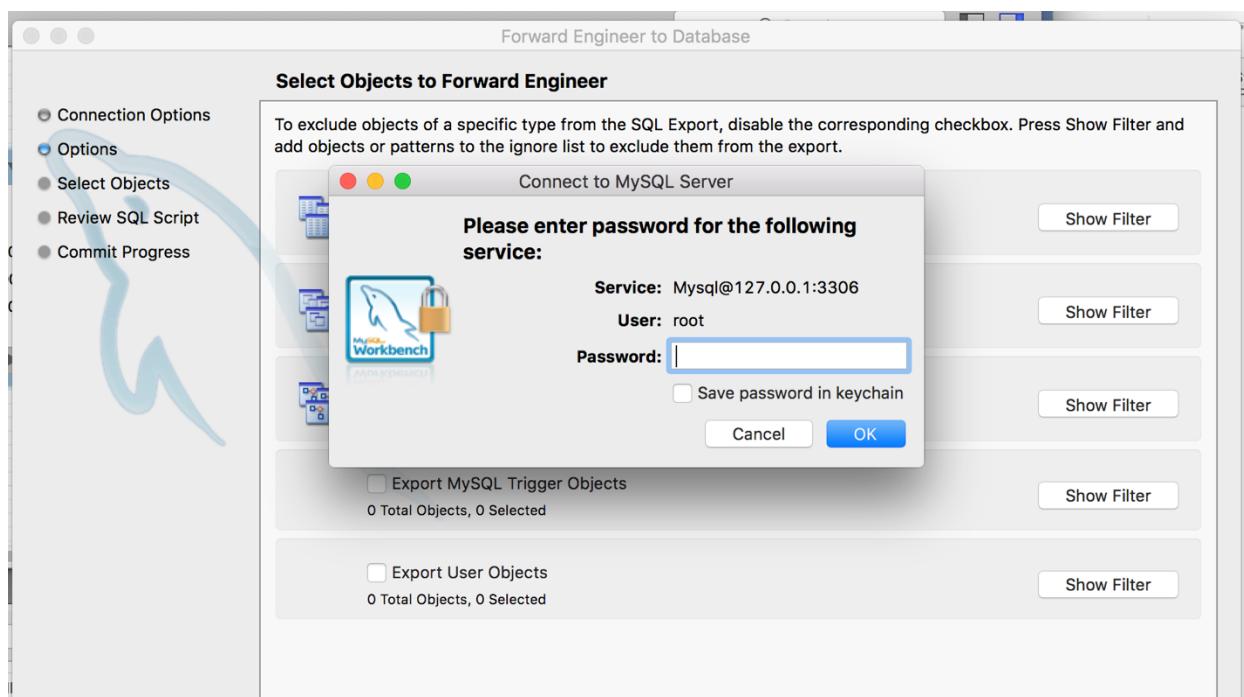
La figura anterior representa el modelo de la base de datos, por lo que se deberá aplicarla en el servidor MySQL utilizando la opción “Forward Engineer...”



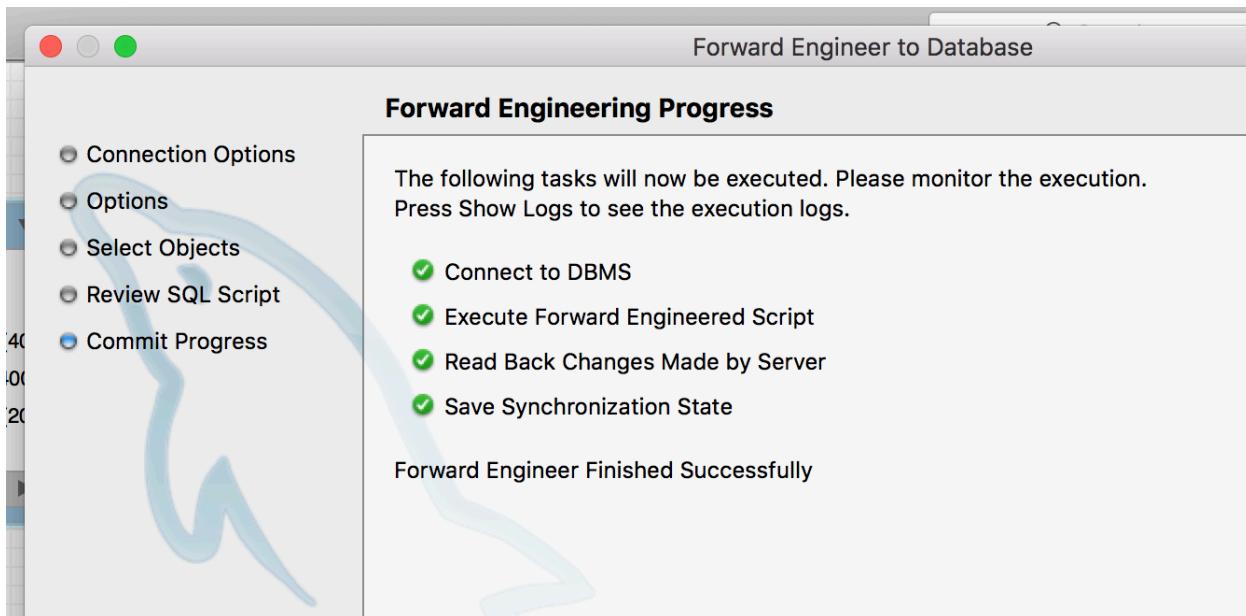
Los parámetros inicialmente son los datos de conexión de la base de datos (Nota: verifique que los servicios del servidor MySQL se están ejecutando).



Indique la contraseña de la base de datos (si el usuario no tiene contraseña, el proceso no la solicitará)

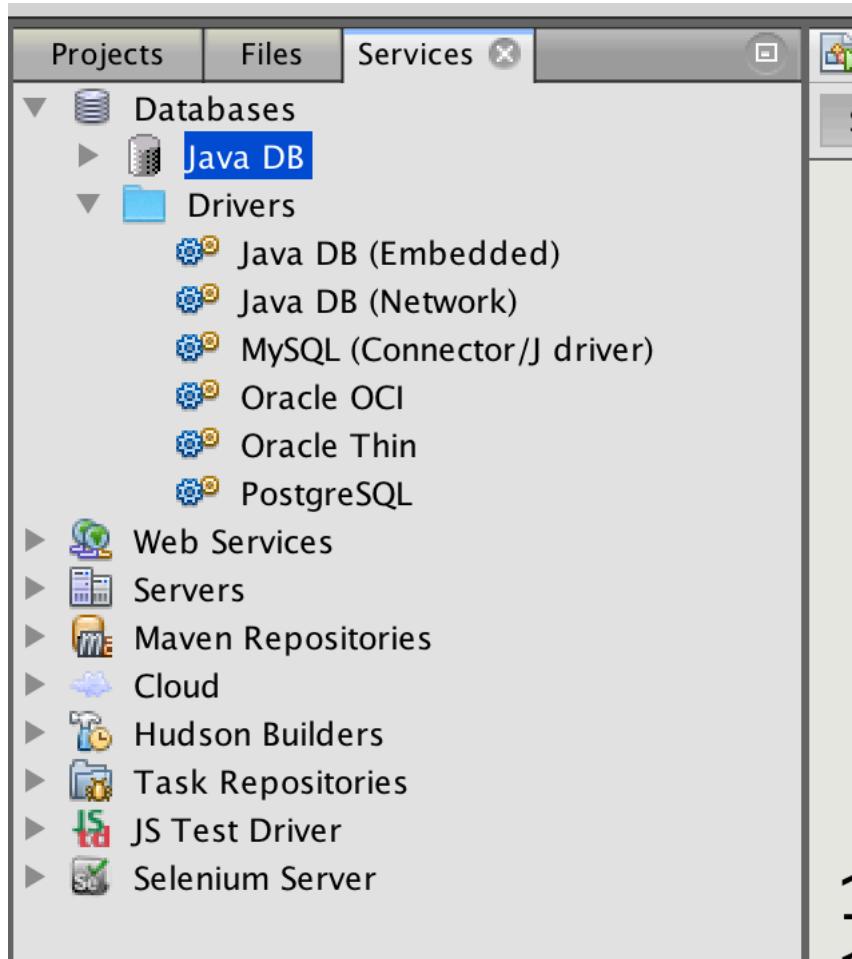


Al finalizar el proceso deberá ver la siguiente pantalla en donde todo deberá tener el “check” en verde, así el log de mensajes de error no deberá aparecer.

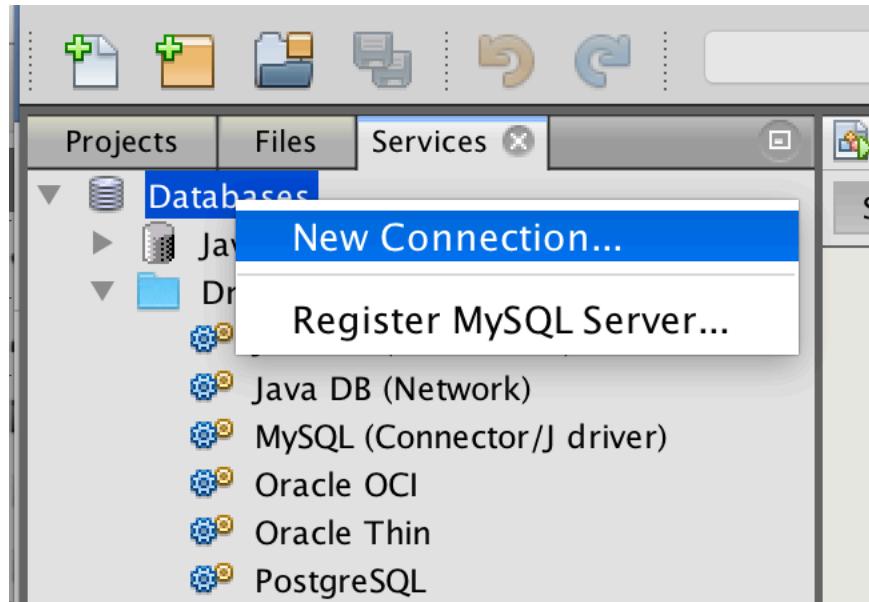


## Creación de la conexión a la base de datos desde NetBeans

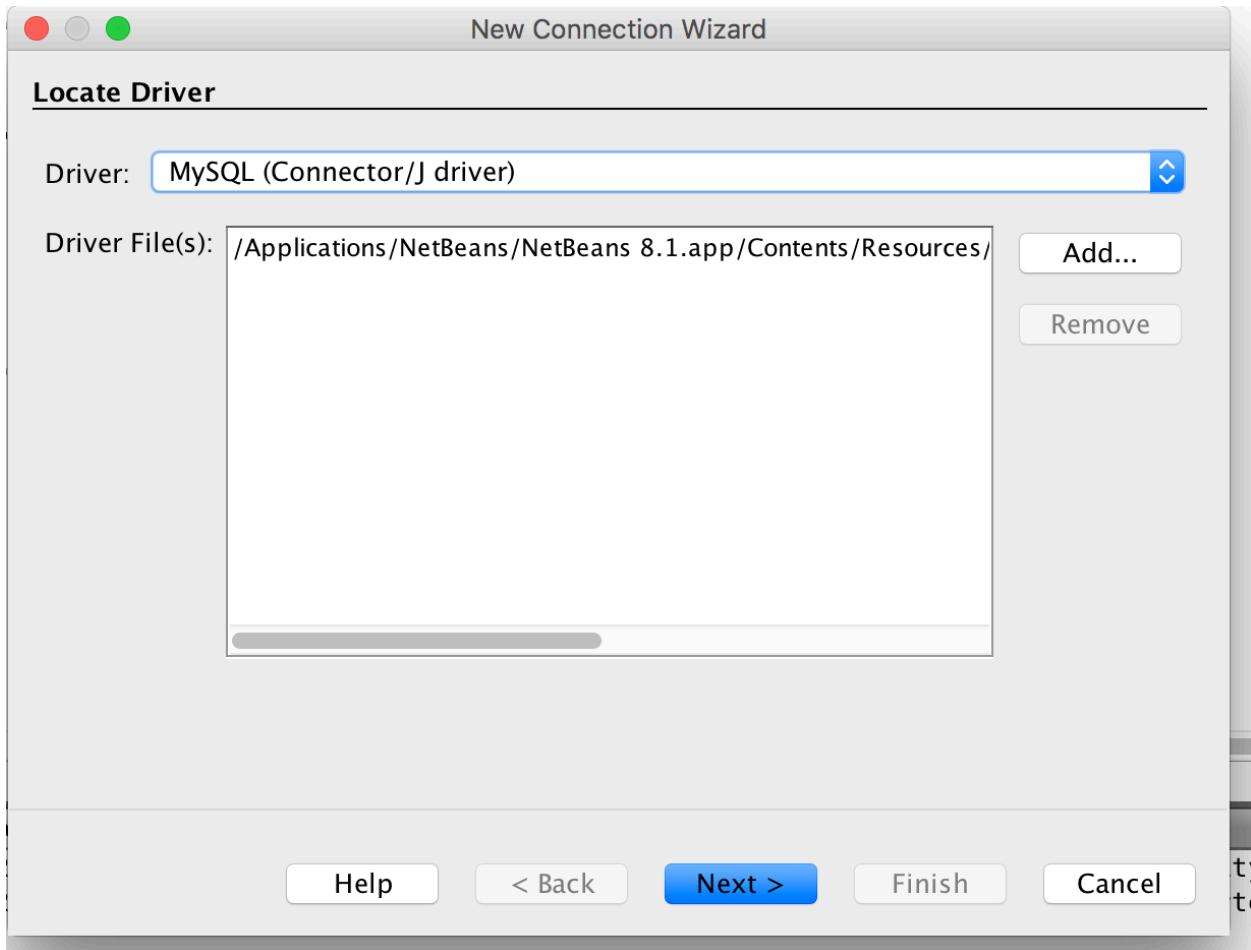
Inicialmente deberá configurar la conexión a la base de datos en NetBeans por lo deberá ir a la pestaña de “Services” (si esta oculta, podrá mostrarla en la opción “Window->Services”), en la cual se muestran los diferentes servicios relacionados al IDE, en los siguientes paso se muestra como crear esta conexión:



Debe dar clic derecho sobre la opción “Databases” y en el menú contextual deberá seleccionar la opción “New Connection...”



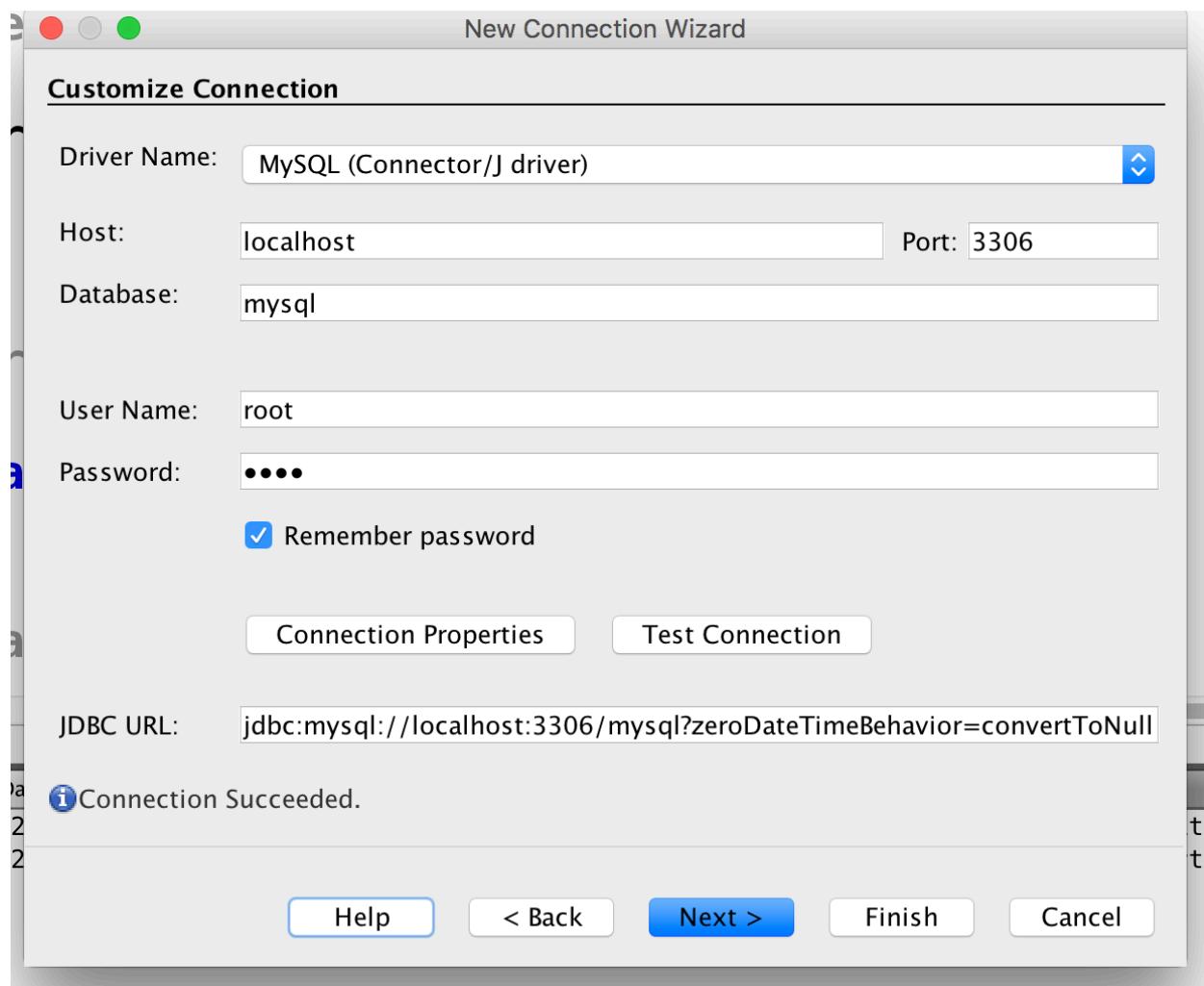
En la siguiente pantalla de configuración de la conexión deberá seleccionar el driver de la base de datos (NetBean tiene algunos Drivers nativos, si no lo encuentra en la el listBox deberá instalarlo en el IDE), para lo que deberá seleccionar el driver correspondiente y dar clic en la opción “Next”.



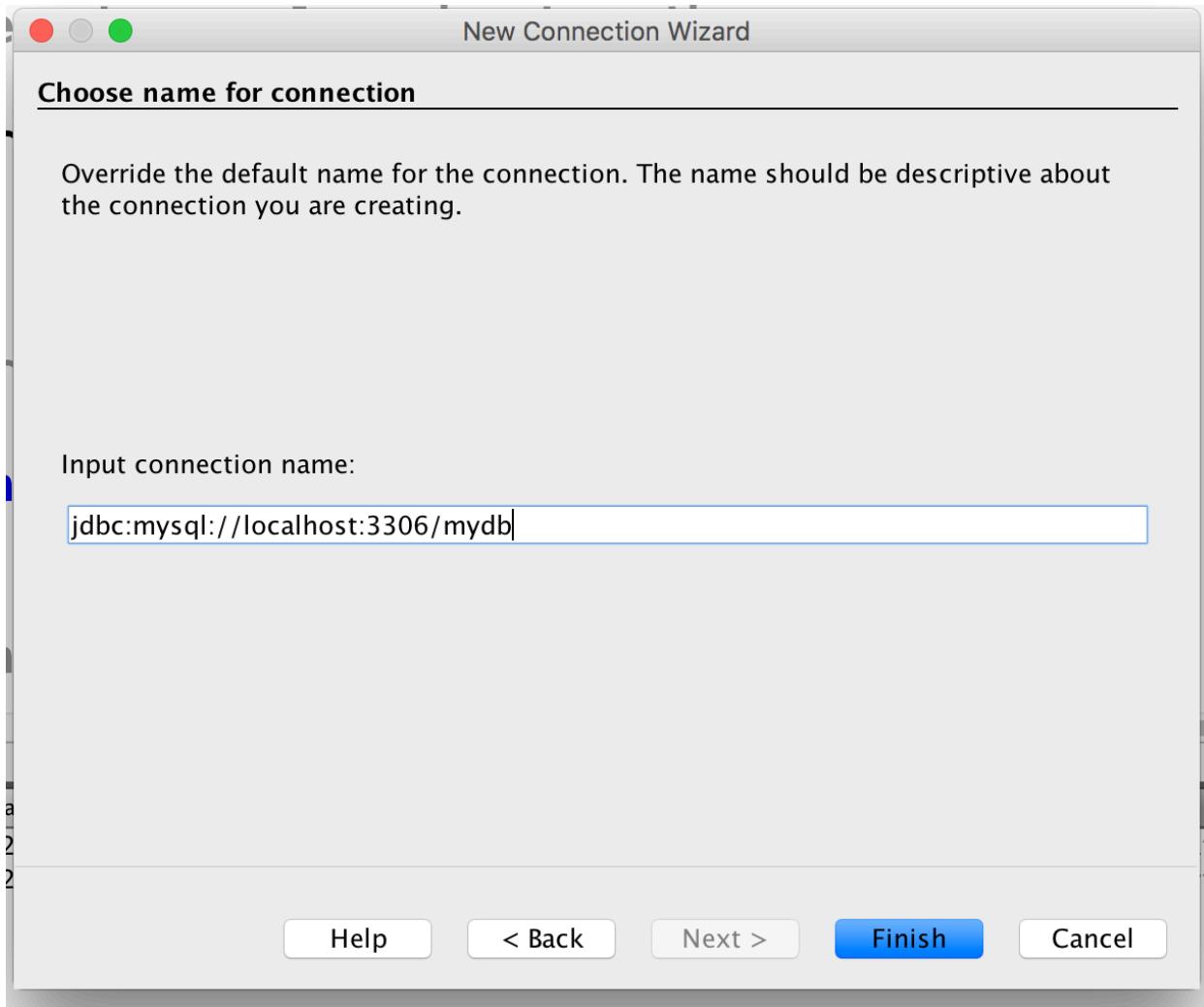
Después, deberá indicar los datos de conexión:

- Host
- Puerto
- Base de datos
- Usuario
- Contraseña

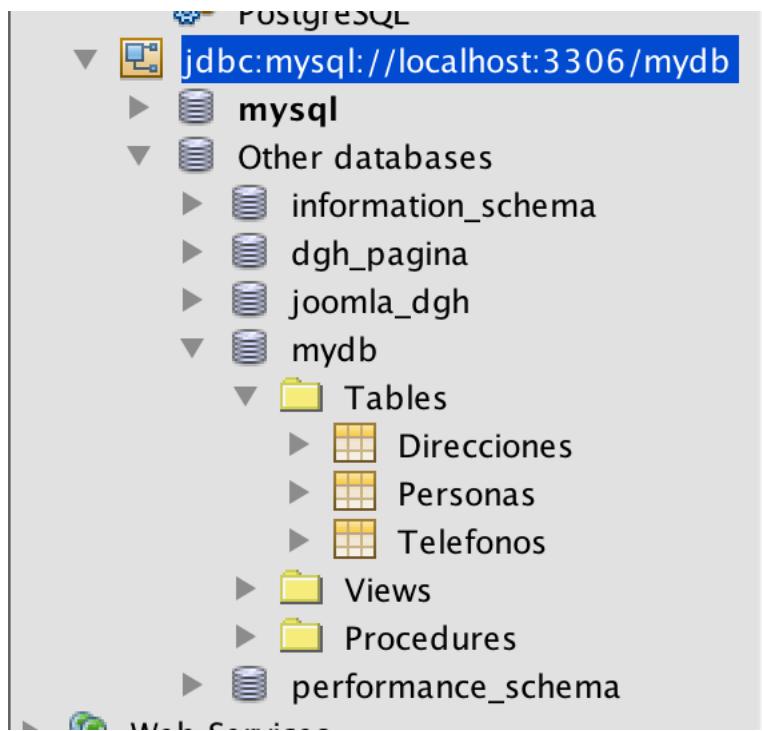
Y dar clic en la opción “Test Connection” para verificar que la conexión a la base de datos se está realizando correctamente.



En la siguiente pantalla se deberá dar la opción “Finish”, para finalizar la configuración de la conexión en NetBeans.



Al finalizar la acción, en servicios se podrá observar la conexión creada, así como explorar desde NetBeans las bases de datos con sus respectivas tablas y campos.

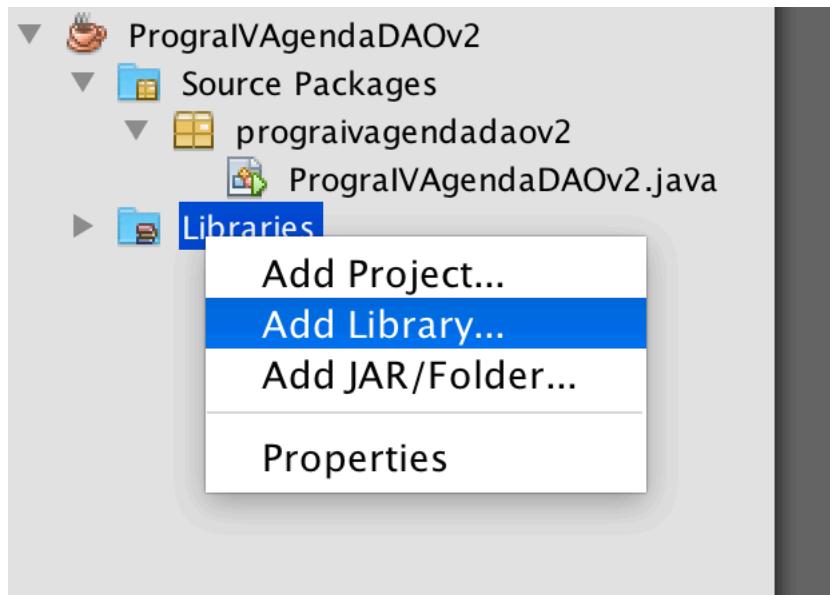


## Creación del proyecto DAO

Para la creación de la capa de acceso a datos (**Data Access Objects**), se creará un proyecto, esto lo que permite es que este pueda ser encapsulado en un archivo .jar y ser usado en varios proyectos.

Para el laboratorio y para efectos del curso se utilizará el framework Hibernate<sup>1</sup> (Tutorial ejemplo: <http://goo.gl/q2lT3V> ), para lo que deberá incluir al proyecto las librerías necesarias para su implementación.

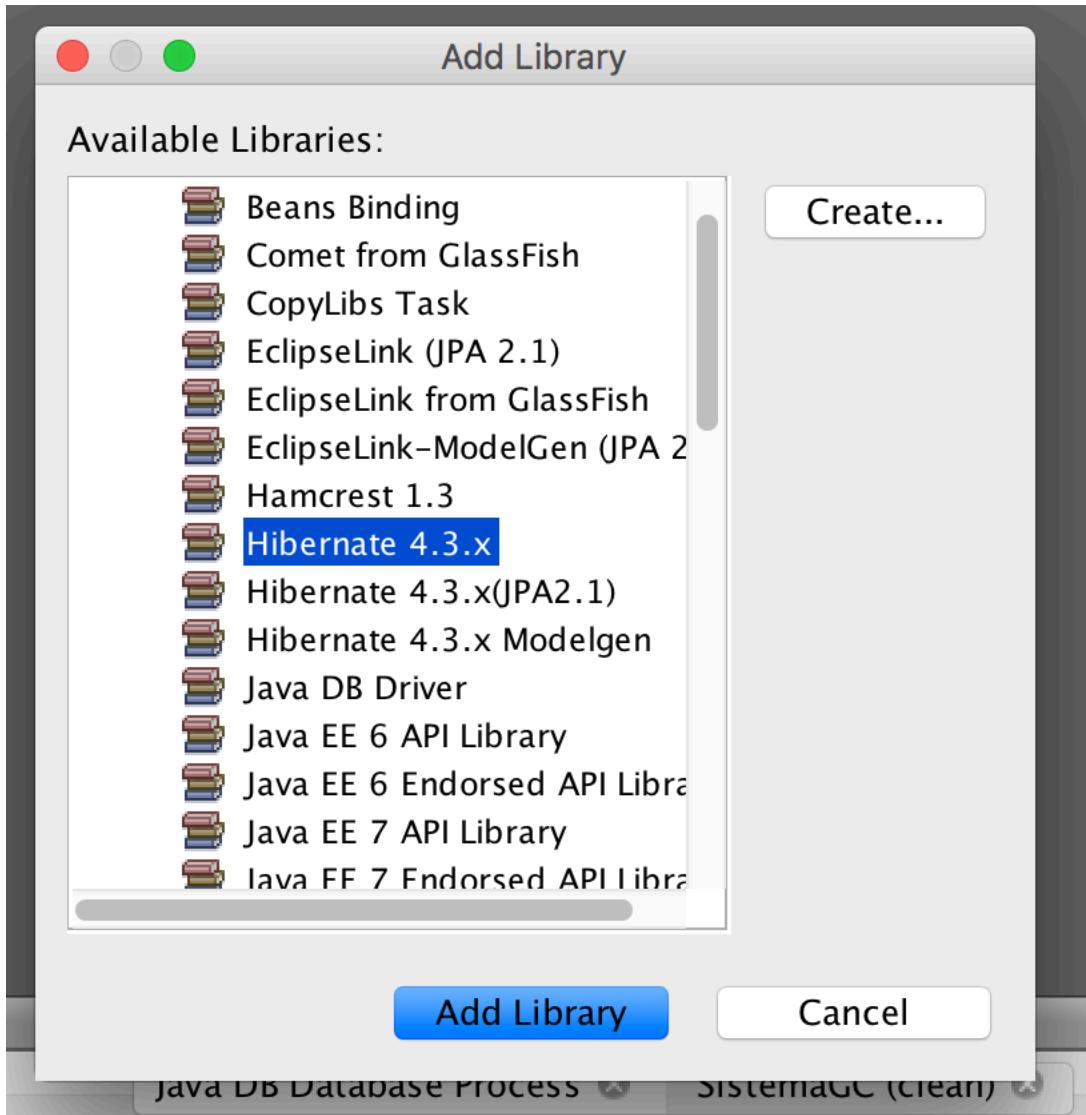
Deberá seleccionar “Libraries” del proyecto creado y dar clic en la opción “Add Library”.



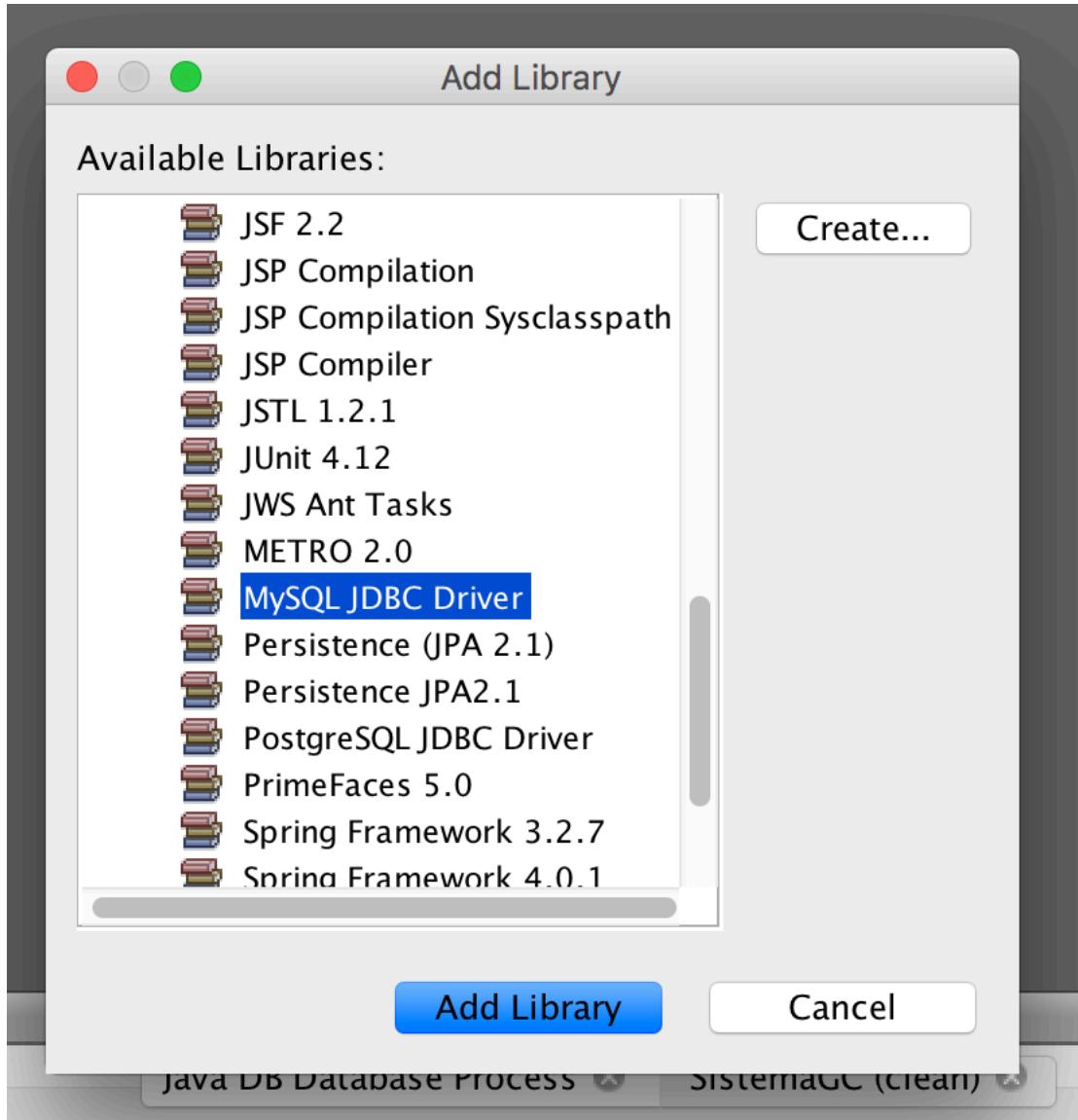
---

<sup>1</sup> Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y 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.

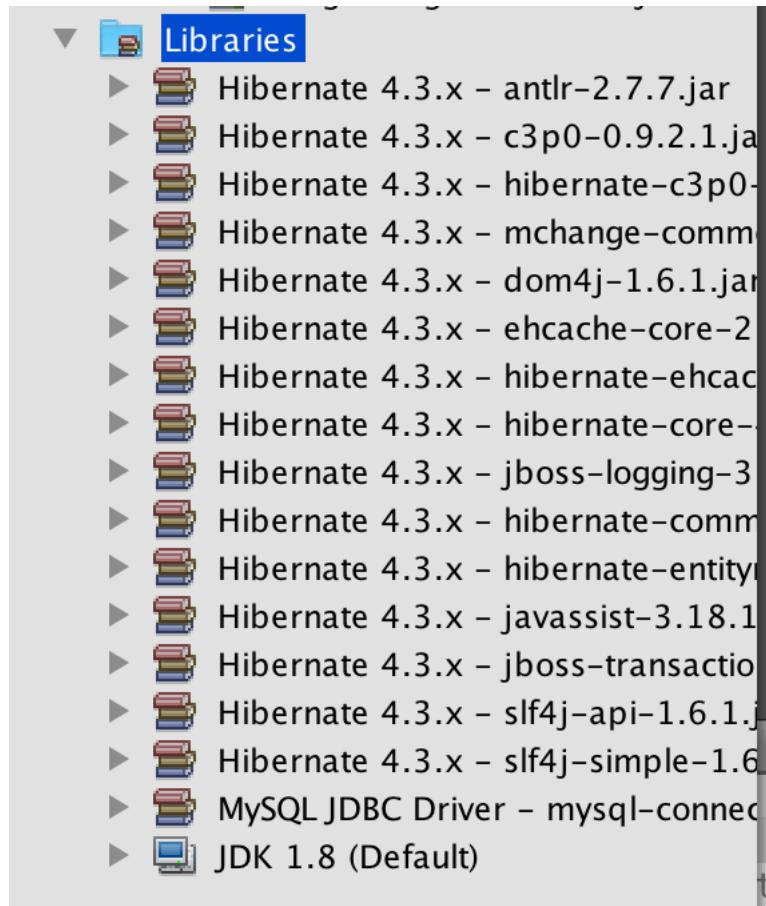
Después, deberá buscar la librería “Hibernate 4.3.x” y dar clic en la opción “Add Library”.



Lo mismo deberá realizar con el driver de conexión de MySQL, realizando el mismo proceso pero incluyendo la librería “MySQL JDBC Driver”.



Al finalizar la inclusión de las librerías, deberá verse de la siguiente manera la carpeta de librerías del proyecto creado para este laboratorio.

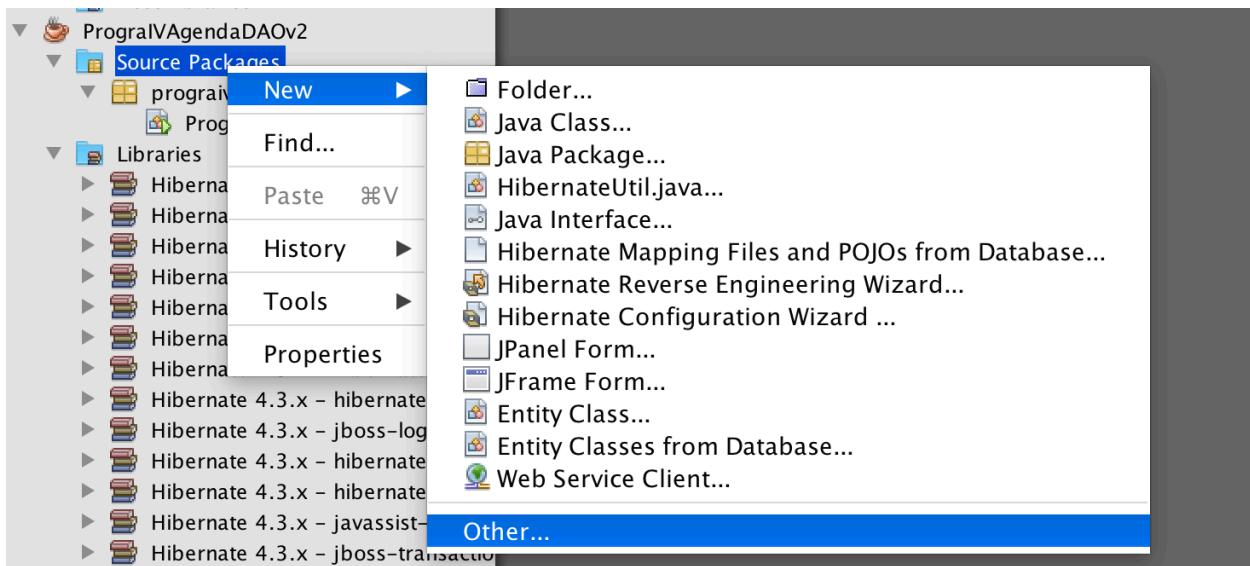


## Creación de los archivos de configuración necesarios para la utilización de Hibernate

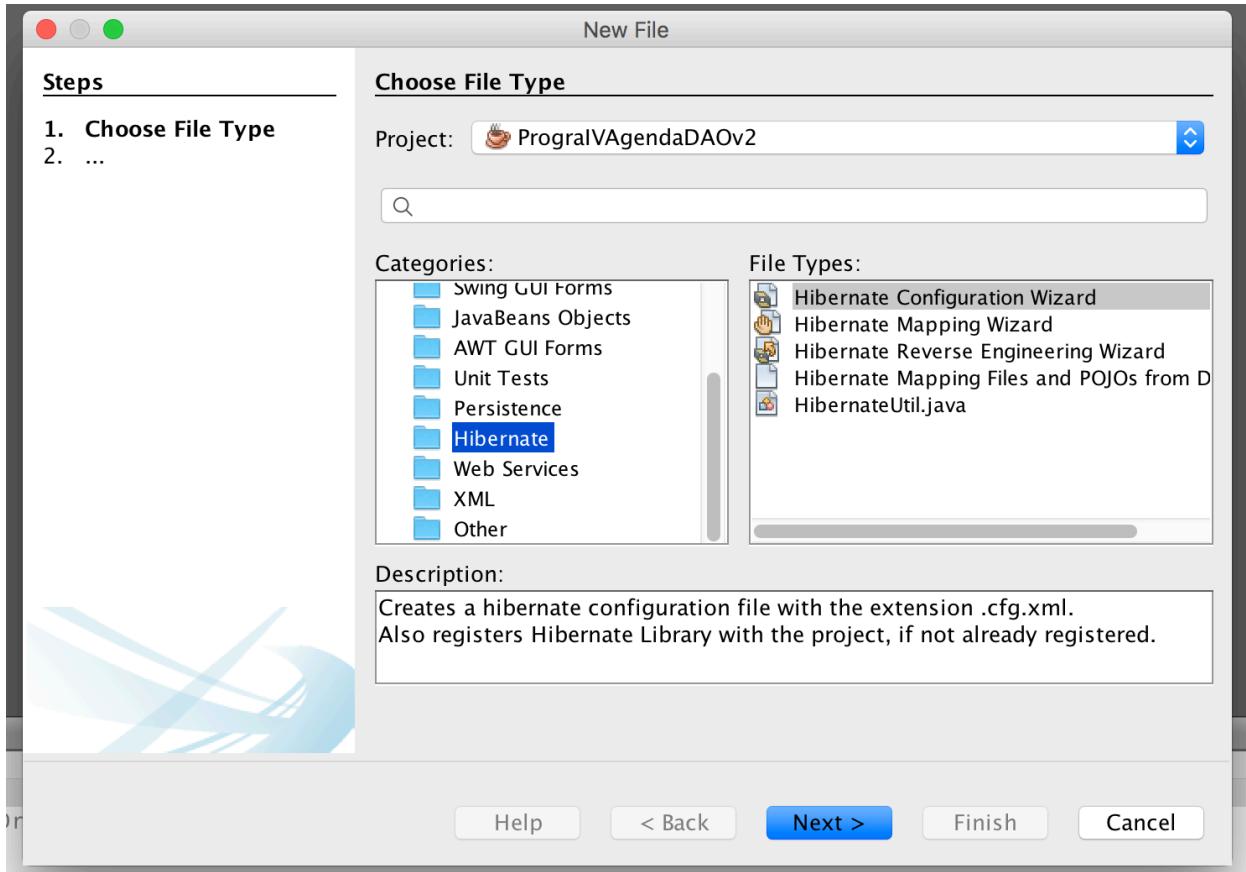
Para la utilización del framework, es necesario la creación de varios archivos de configuración, los cuales son:

- Hibernate Configuration Wizard
- Hibernate Reverse Engineering Wizard

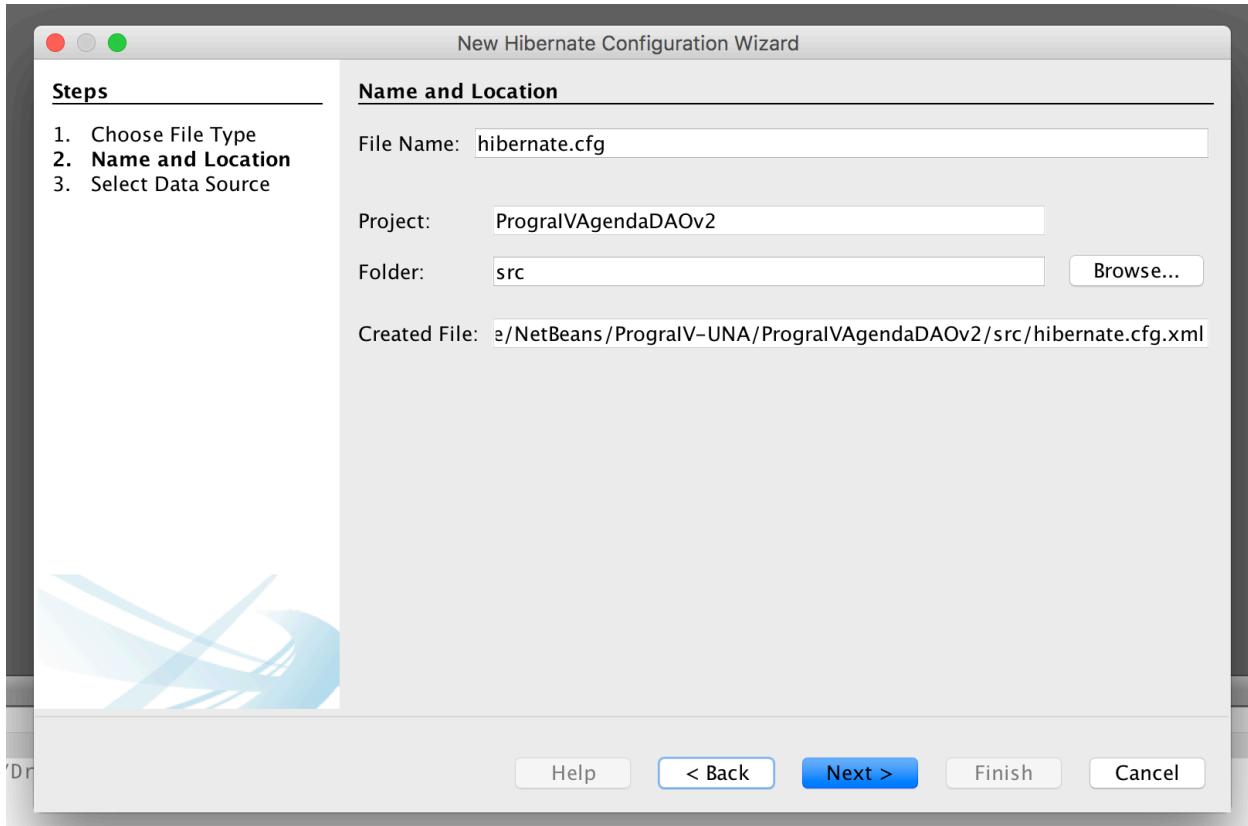
Para esto deberá dar clic en “Source Packages” y en el menú contextual deberá seleccionar “New->Other...”



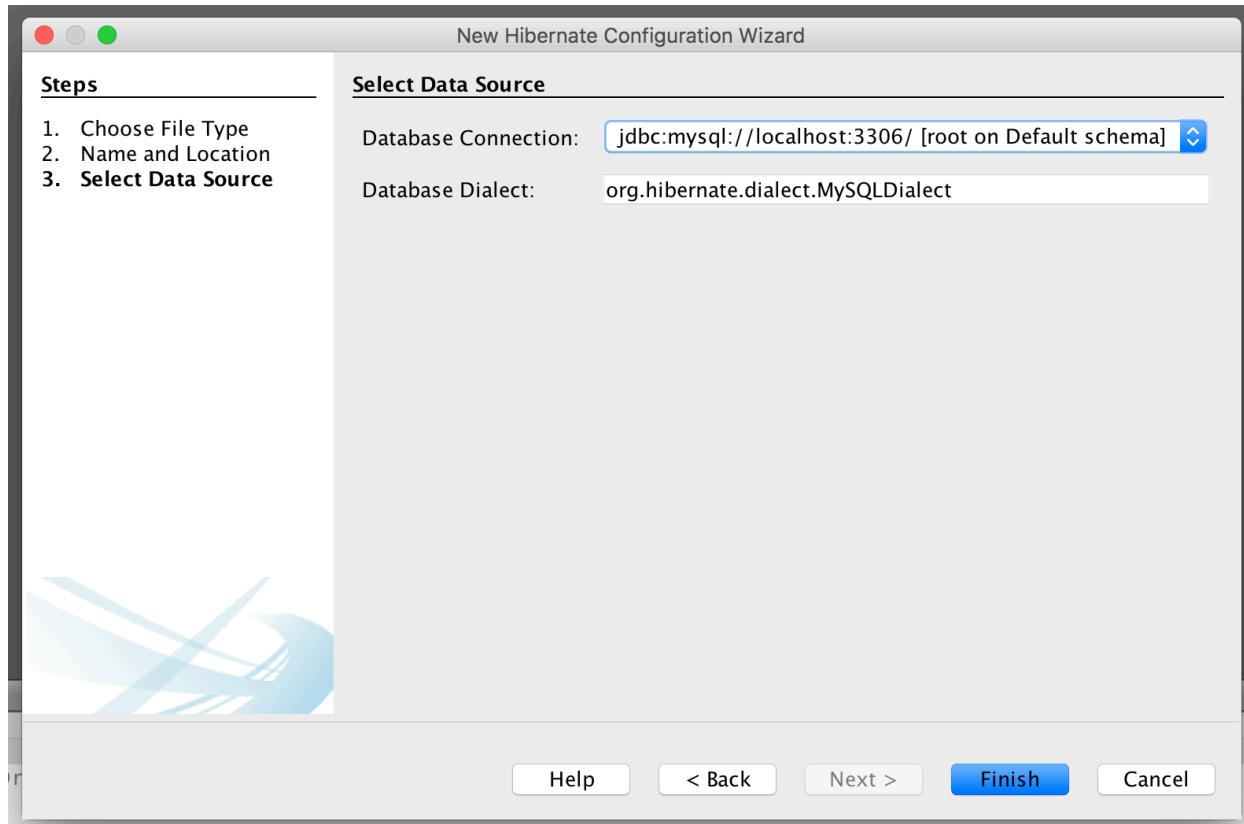
Posteriormente, deberá seleccionar la categoría “Hibernate” y en el tipo de archivo seleccionar la opción “Hibernate Configuration Wizard” y dar clic en el botón “Next”.



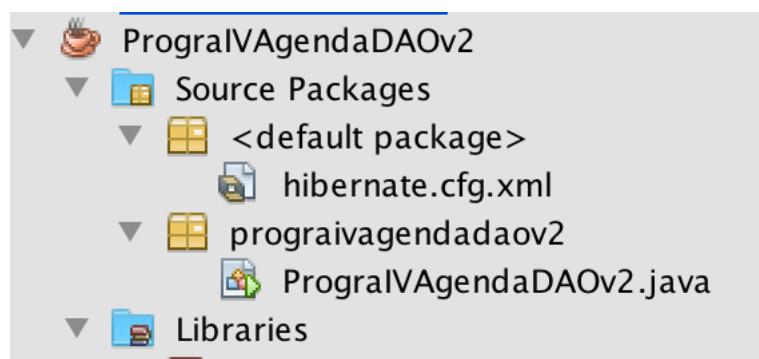
Luego deberá indicar el nombre del archivo de configuración, que para efectos de laboratorio deberá llamarse “hibernate.cfg”.



En la siguiente pantalla deberá seleccionar la conexión a la base de datos creada en Netbeans (uno de los primeros para del presente documento) e indicar el dialecto, recuerdo que Hibernate es un framework de persistencia de bases de datos, por lo que deberá indicar a cual base de datos se va a conectar, ya que el framework implementará el SQL correspondiente de una base de datos u otra.



Después de la creación de este archivo el proyecto deberá mostrarse de la siguiente manera, el archivo “hibernate.cfg.xml” en “<default package>”

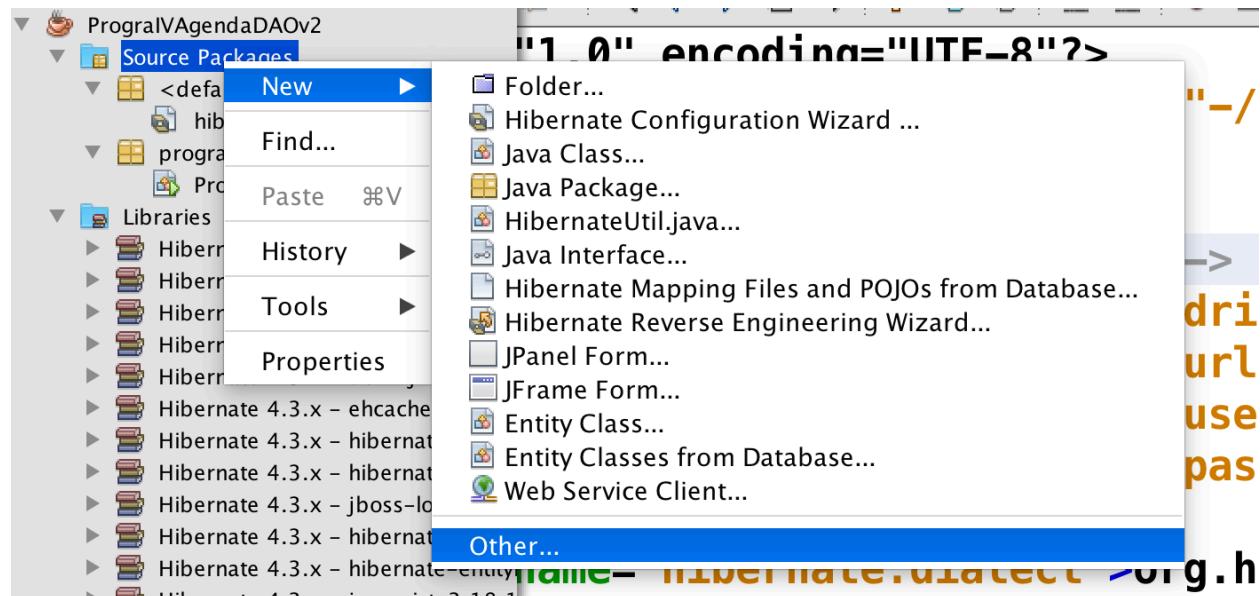


Y el archivo deberá contener la siguiente información, denote, que los datos que posee el archivo son los necesarios para realizar la conexión a la base de datos deseada.

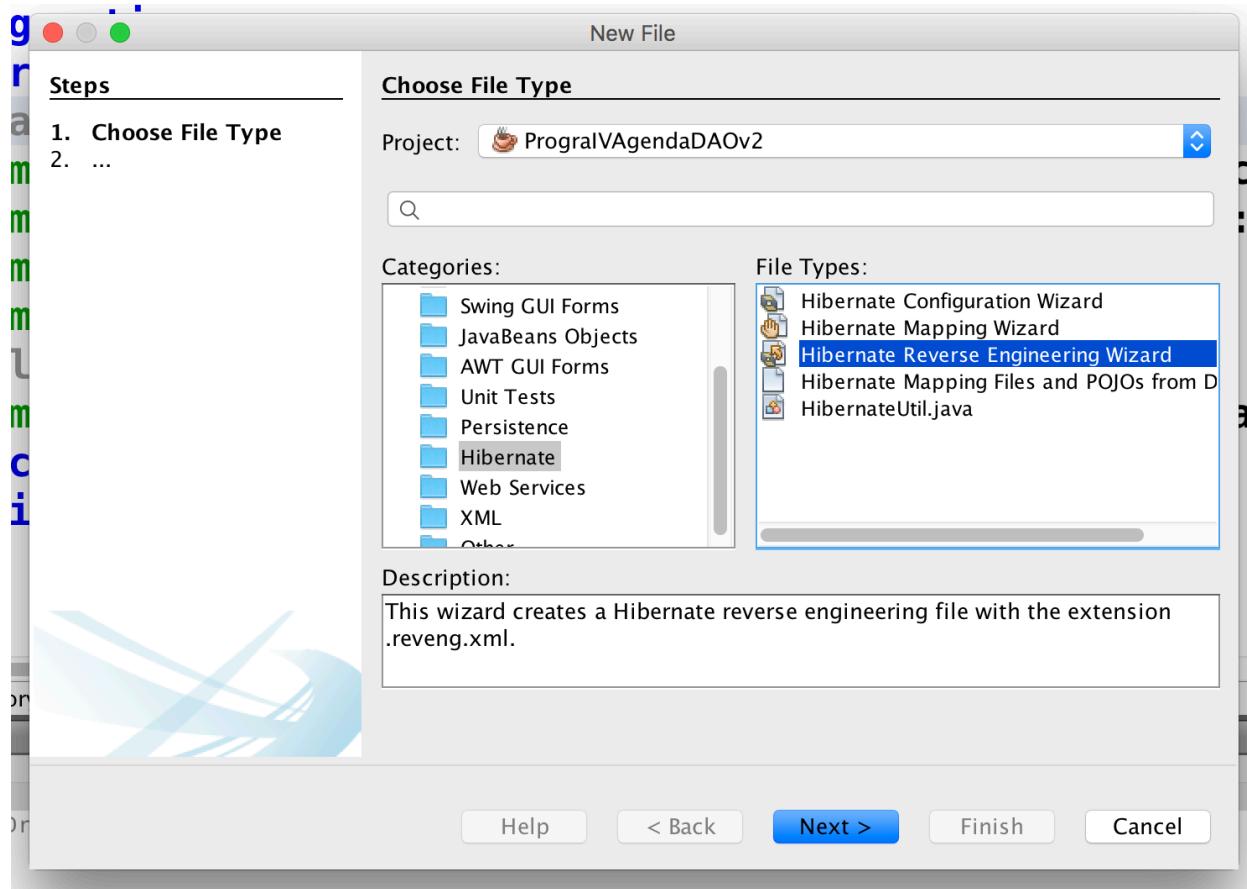
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mydb</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">root</property>
        <!-- SQL dialect -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    </session-factory>
</hibernate-configuration>
```

El siguiente archivo a crea es el “Hibernate Reverse Engineering Wizard”, el cual servirá para aplicar ingeniería reversa a partir de la base de datos, es decir, generar los archivos (clases y xml) según lo que posea la base de datos en cuestión.

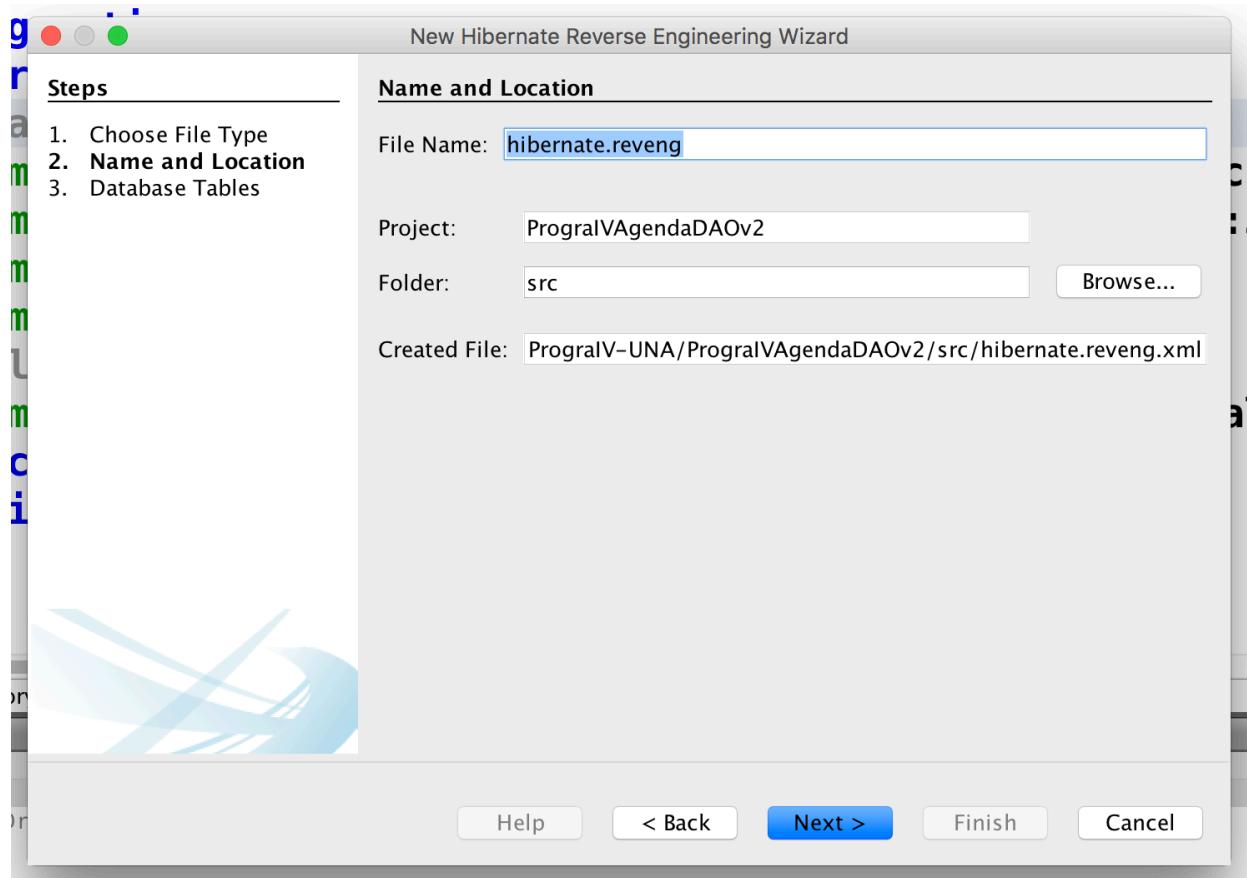
Para lo que deberá dar clic derecho en “Source Package” y en el menú contextual seleccionar la opción “New->Other...”



En la pantalla que se muestra a continuación deberá seleccionar la categoría “Hibernate” y el archivo “Hibernate Reverse Engenieering Wizard”.

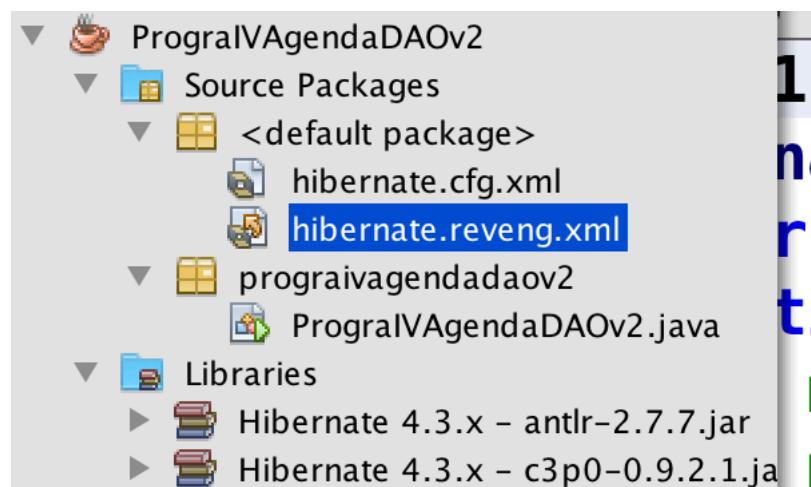
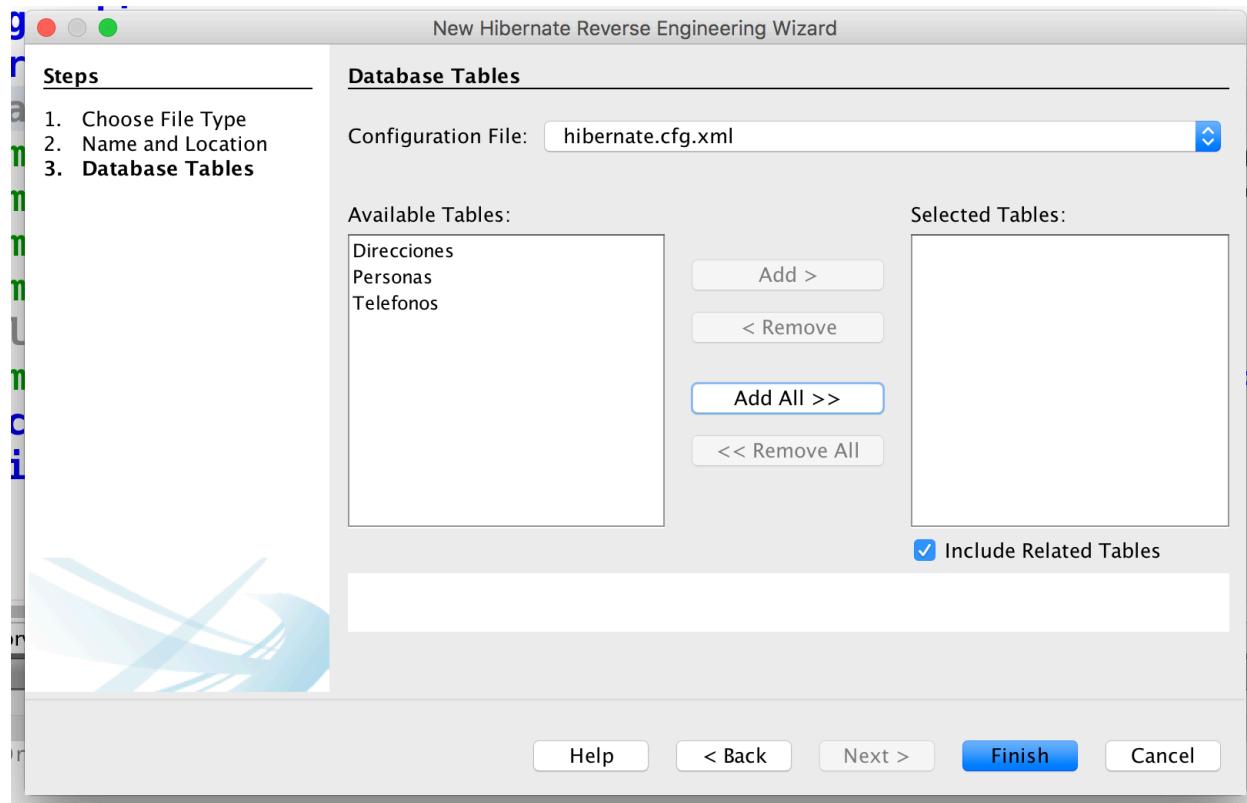


Posteriormente, netBeans solicitará el nombre de archivo de configuración, que para efectos del laboratorio se llamará como lo indica netbeans por defecto “hibernate.reveng”.



En la siguiente pantalla Netbeans, le solicitará seleccionar las tablas a las cuales desea aplicarle la ingeniería reversa (Nota: estos datos de la base de datos son mostrados con los datos de conexión que se indicaron en el archivo anterior, si las tablas no aparecen verifique que estos datos sean los correctos en el archivo hibernate.cfg.xml)

De estar todo correcto se debe proceder a incluir las tablas necesarios. "Add All" para efectos del laboratorio.



En el archivo creado deberá tener la siguiente información.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering D1
<hibernate-reverse-engineering>
  <schema-selection match-catalog="mydb"/>
  <table-filter match-name="Direcciones"/>
  <table-filter match-name="Personas"/>
  <table-filter match-name="Telefonos"/>
</hibernate-reverse-engineering>
```

Para el laboratorio deberá crear los siguientes paquetes:

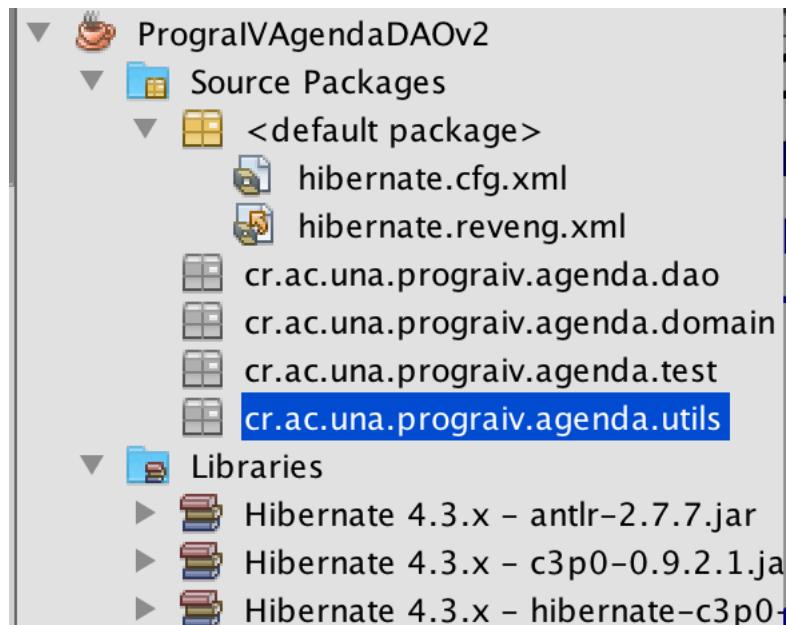
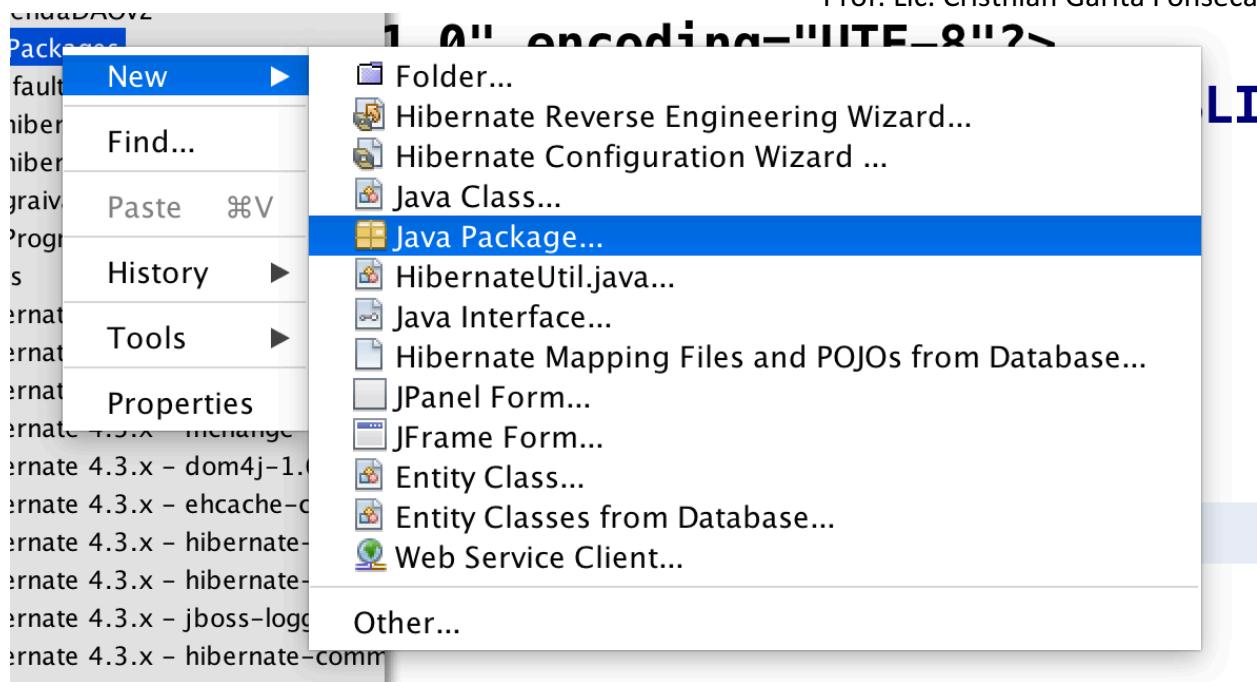
- **cr.ac.una.prograIV.agenda.dao**
  - Objetos de acceso a la base de datos
- **cr.ac.una.prograIV.agenda.domain**
  - Archivos POJO (clases “planas” en java) y los archivos de mapeo con la base de datos por la utilización de hibernate
- **cr.ac.una.prograIV.agenda.test**
  - Capa para crear las clases de prueba
- **cr.ac.una.prograIV.agenda.utils**
  - Objetos utilitarios del proyecto

#### Nota

Para la creación de los paquetes se va utilizará un estándar utilizado para nombrarlos, para lo que se debe incluir el dominio de la empresa o institución en la que se desarrollará la aplicación:



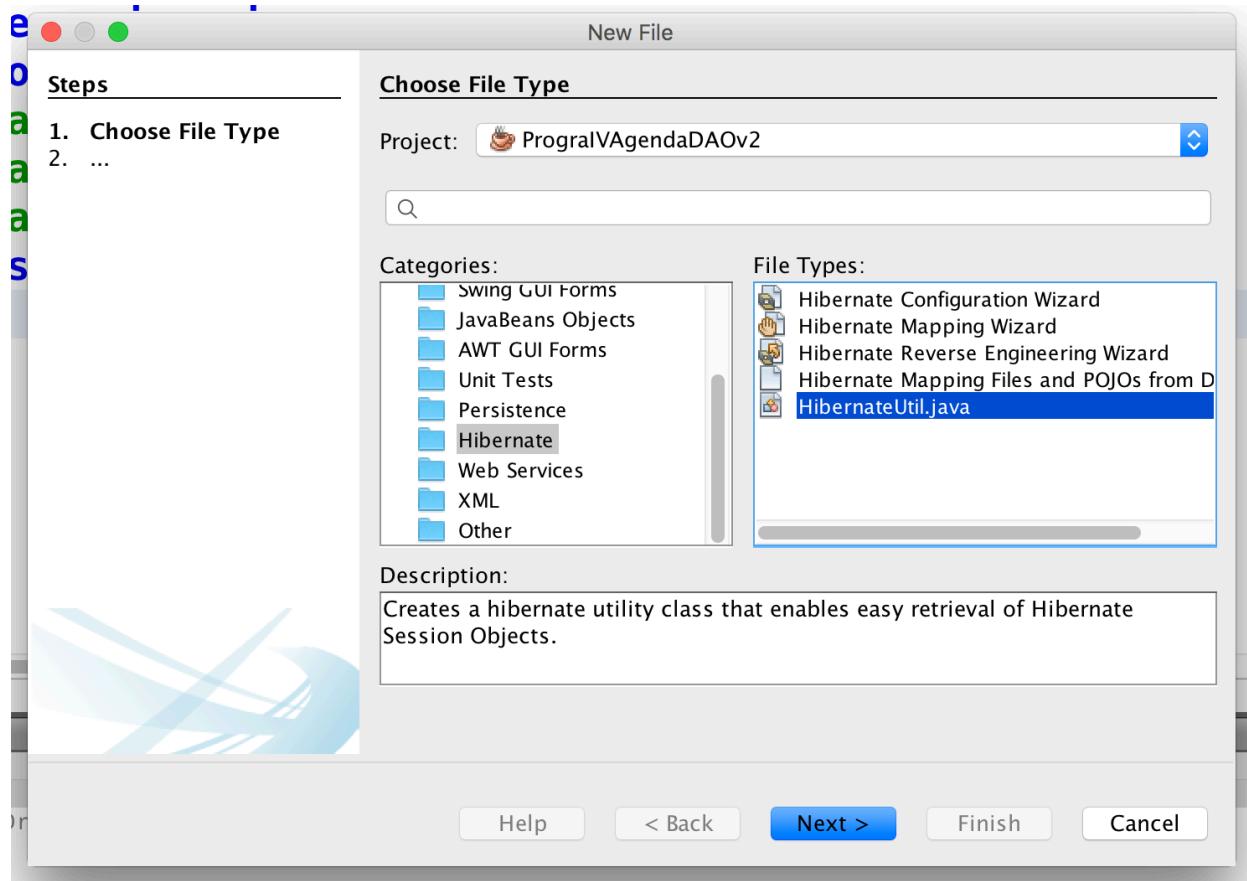
Para lo que deberá dar clic en “Source Package” y crear los paquetes indicados anteriormente.



Creación de la clase “Utilerías de Hibernate”

Para la utilización del hibernate deberá tener acceso al “Session Factory”<sup>2</sup>, para lo cual se creará una clase que la gestione para su utilización en los objetos de acceso a la base de datos (DAO).

Para la creación de esta clase, se deberá dar clic derecho al paquete “utils” y en el menú contextual seleccionar la opción “New->Other..”, después buscar la categoría “Hibernate” y el archivo “HibernateUtil.java” y dar clic en la opción siguiente hasta finalizar con la creación de la clase.



Esta clase deberá contener el siguiente código:

- Métodos a atributos para poder acceder al “sessionfactory”

<sup>2</sup> La **SessionFactory** es aquella que se encarga indicarle al sistema, donde se encuentran todos los ficheros de mapeo de Hibernate y de configuración.

- Un método para iniciar cada operación con la base de datos, nótese que en este método se obtiene el sessionfactory y se inicia una transacción<sup>3</sup> en la base de datos.
- Un método para controlar los errores en las transacciones que se realicen en la base de datos, en caso de fallo, se realiza un rollBack, lo que regresa la base de datos a su estado original antes del inicio de la transacción.

```

public class HibernateUtil {

    private static final SessionFactory sessionFactory;
    private Transaction transac;
    private Session sesion;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    //Metodos para el manejo de transacciones

    public void iniciaOperacion() throws HibernateException {
        this.sesion = HibernateUtil.getSessionFactory().openSession();
        this.transac = sesion.beginTransaction();
    }

    public void manejaExpcion(HibernateException he) throws HibernateException {
        transac.rollback();
        throw new HibernateException("Ocurrió un error en la capa de acceso a datos", he);
    }
}

```

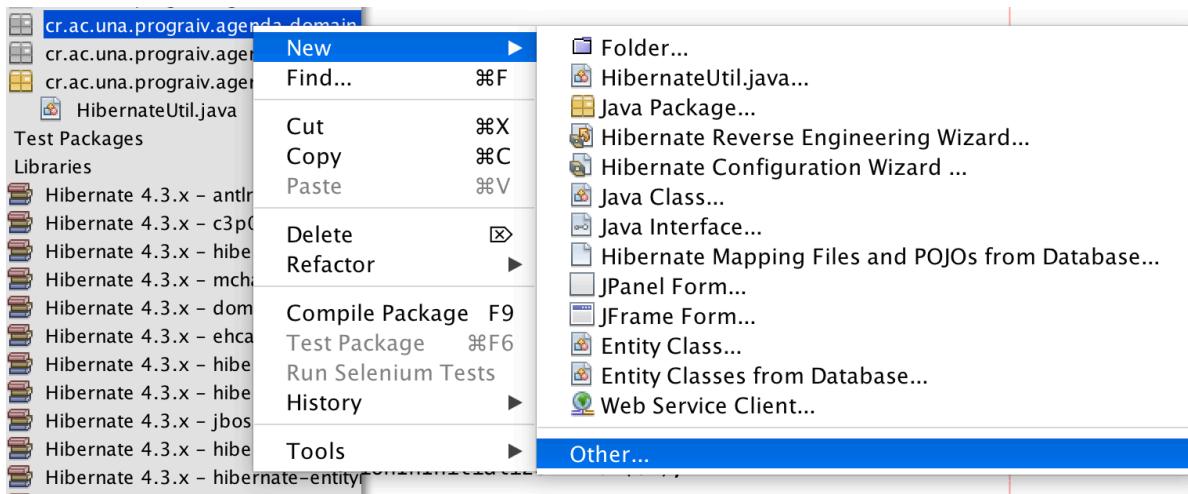
## Generación del mapeo de la base de datos

---

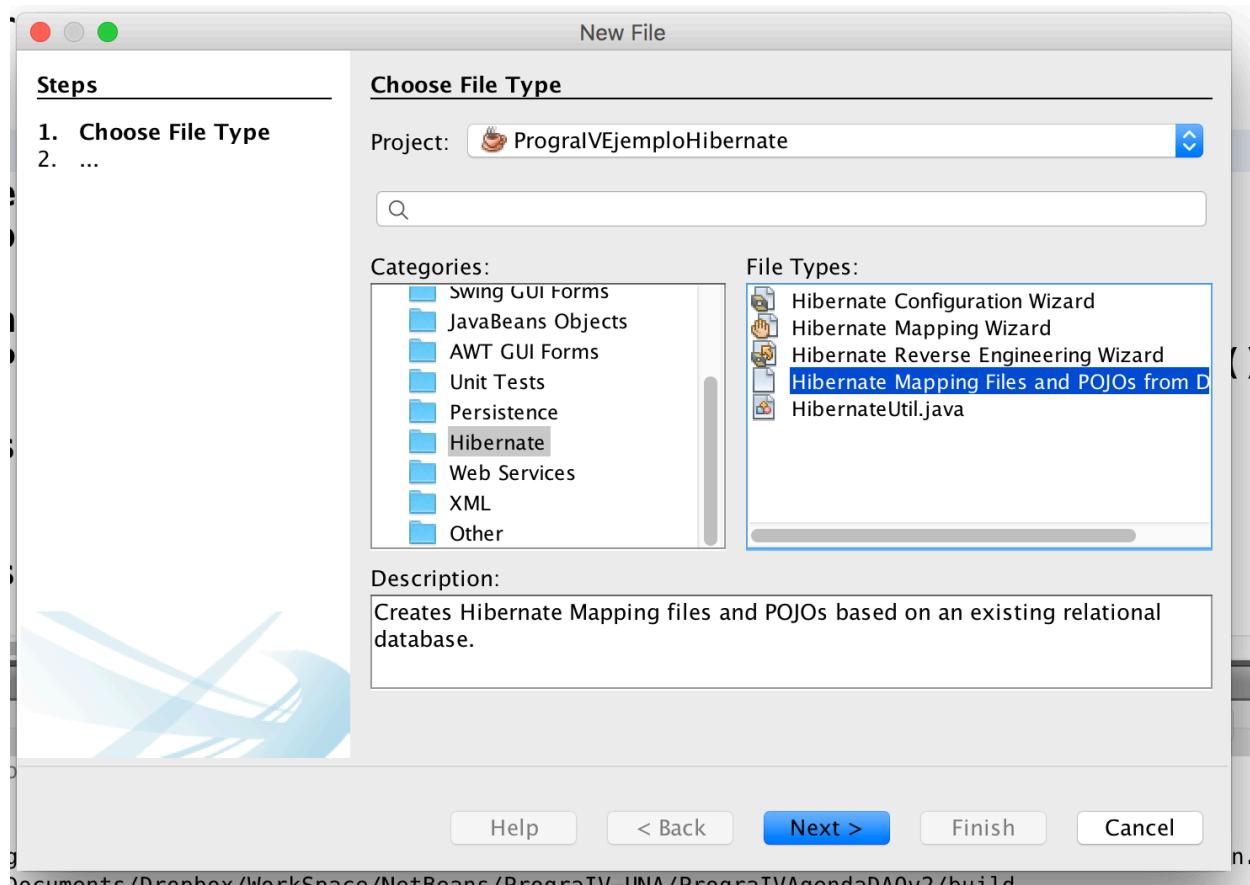
<sup>3</sup> Un SGBD se dice **transacional**, si es capaz de mantener la integridad de los datos, haciendo que estas **transacciones** no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Una de las características de Hibernate es realizar un mapeo de la base de datos y crear los objetos en java relacionados a cada tabla existente en esta, por lo que se dice que con hibernate se programa con una base de datos de objetos.

Para poder realizar el mapeo de la base de datos, deberá dar clic en el paquete “domain”, y en menú contextual deberá seleccionar la opción “New->Other..”

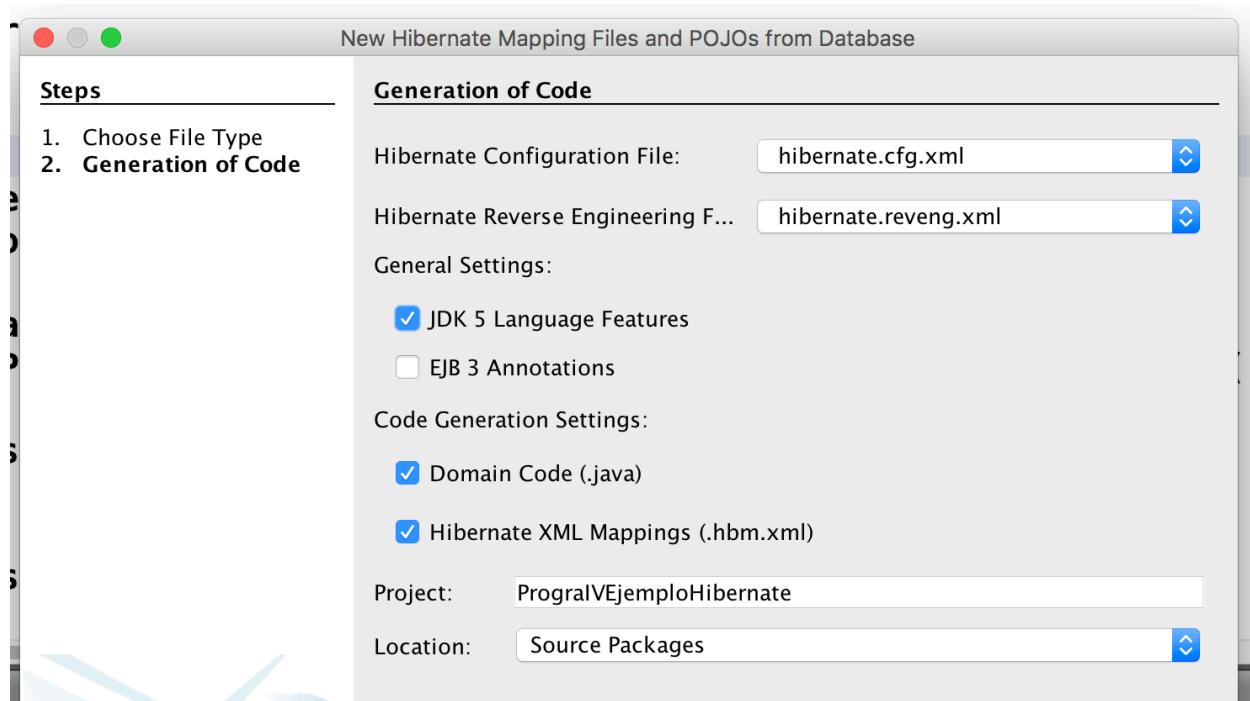


Luego, seleccionar la categoría “Hibernate” y el archivo “Hibernate Mapping Files and POJOs from Data Base”

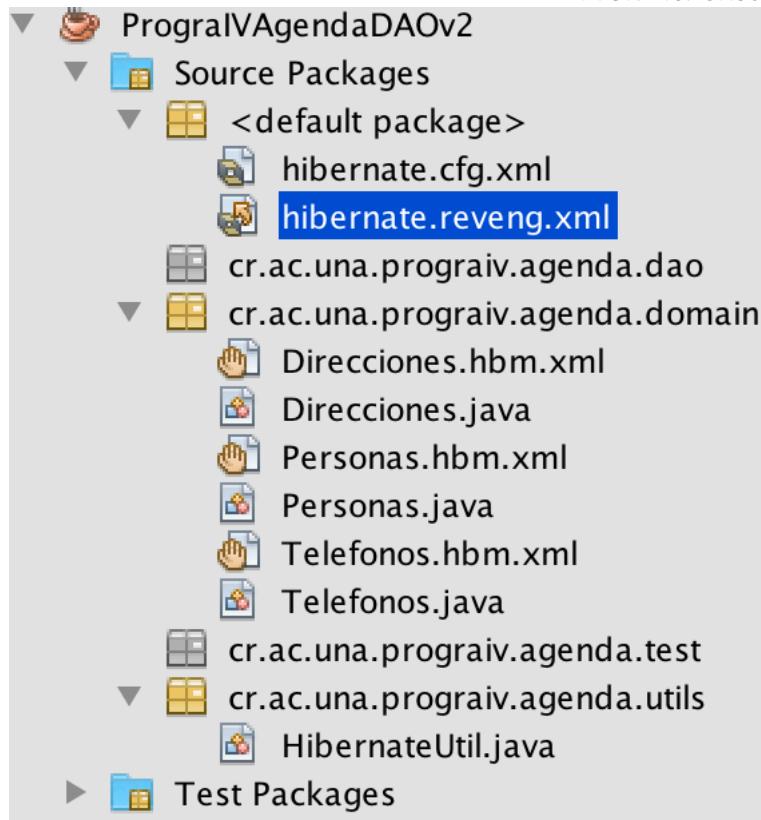


Finalmente deberá seleccionar la opción “JDK 5 Language Features” para que genere las clases con las especificaciones del JDK5.

Nota: note que el wizard busca los archivos de configuración creados anteriormente (hibernate.cfg.xml e hibernate.reveng.xml) si estos están mal configurados la acción no se podrá realizar con éxito.



Una vez generado el mapeo de la base de datos, podrá observar en el paquete “domain”, las clases con sus receptivos mapeos haciendo referencia a la base de datos en cuestión.



```

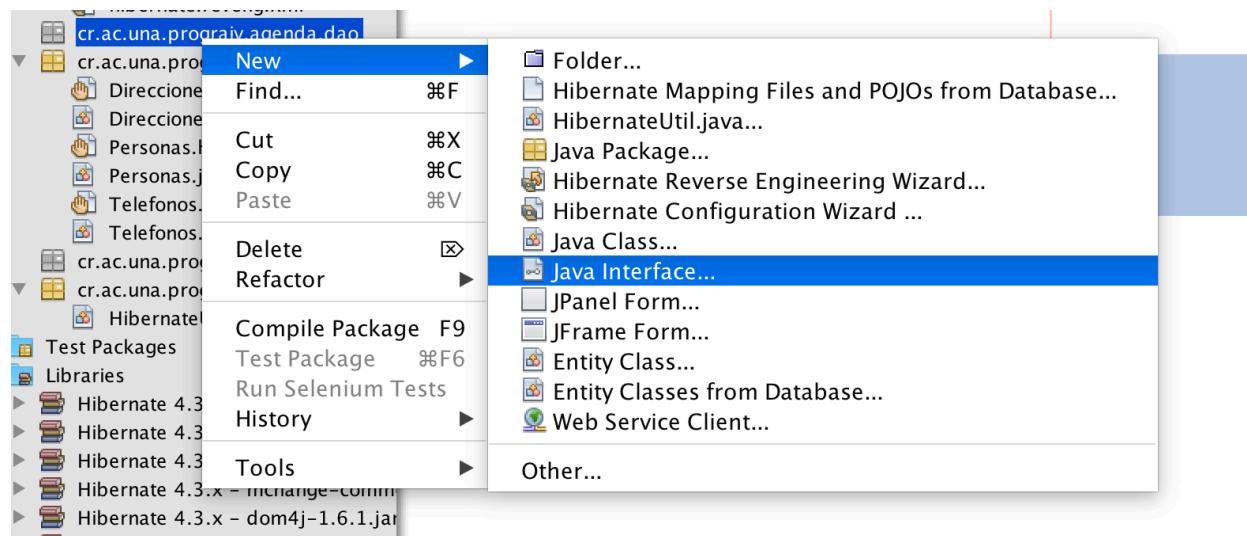
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://
<hibernate-configuration>
<session-factory>
    <!-- Database connection settings -->
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/mydb</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>
    <!-- SQL dialect -->
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <!-- Archivos de mapeo -->
    <mapping resource="cr/ac/unaprograiv/domain/Personas.hbm.xml"/>
    <mapping resource="cr/ac/unaprograiv/domain/Direcciones.hbm.xml"/>
    <mapping resource="cr/ac/unaprograiv/domain/Telefonos.hbm.xml"/>
</session-factory>
</hibernate-configuration>

```

## Creación del paquete DAO

En el paquete DAO, existirán todos los objetos que tienen acceso a la base de datos, como todos los objetos tiene el mismo comportamiento se procederá a la creación de una interfaz que será implementada en cada una de las clases DAO gestionen la información de la base de datos.

Para esto deberá dar clic derecho sobre el paquete DAO, y crear la interfaz “IBaseDAO”, la cual utiliza Java Generics<sup>4</sup> para su desarrollo



```
public interface IBaseDAO <T,K> {
    public abstract void save (T o);
    public abstract T merge (T o);
    public abstract void delete (T o);
    public abstract T findById (K o);
    public abstract List<T> findAll();
}
```

<sup>4</sup> Uso de java generics <http://goo.gl/H7OsQS>

Posteriormente deberá implementar las clases DAO para cada una de las tablas de la base de datos, heredando de la clase “HibernateUtil” e implementando la interfaz “IBaseDAO”.

Con los siguientes métodos:

1. Save: Guarda el objeto en la base de datos.
2. Merge: Verifica el objeto con el registro de la tabla, las diferencias en los atributos del objeto los aplica en la base de datos (update).
3. Delete: Borra el objeto de la base de datos.
4. findById: Busca un objeto por el identificador en la base de datos y lo retorna.
5. findAll: retorna todos los objetos de la base de datos utilizando el HQL<sup>5</sup> como lenguaje de consulta navito utilizado por hibernate.

## SAVE

```
public class PersonasDAO extends HibernateUtil implements IBaseDAO<Personas, java.math.BigInteger>{  
  
    @Override  
    public void save(Personas o) {  
        try {  
            iniciaOperacion();  
            getSession().save(o);  
            getTransac().commit();  
        } catch (HibernateException he) {  
            manejaExcepcion(he);  
            throw he;  
        } finally {  
            getSession().close();  
        }  
    }  
}
```

---

<sup>5</sup> Tutorial de HQL: <http://goo.gl/mKA4sS>

## MERGE

```
@Override
public Personas merge(Personas o) throws HibernateException {
    try {
        iniciaOperacion();
        o = (Personas) getSesion().merge(o);
        getTransac().commit();
    } catch (HibernateException he) {
        manejaExcepcion(he);
        throw he;
    } finally {
        getSesion().close();
    }
    return o;
}
```

## DELETE

```
@Override
public void delete(Personas o) {
    try {
        iniciaOperacion();
        getSesion().delete(o);
        getTransac().commit();
    } catch (HibernateException he) {
        manejaExcepcion(he);
        throw he;
    } finally {
        getSesion().close();
    }
}
```

## FINDBYID

```

@Override
public Personas findById(BigInteger id) {
    Personas personas = null;

    try {
        iniciaOperacion();
        personas = (Personas) getSesion().get(Personas.class, id);
    } finally {
        getSesion().close();
    }
    return personas;
}

```

## FINDALL

```

@Override
public List<Personas> findAll() {
    List<Personas> listaPersonas;
    try {
        iniciaOperacion();
        listaPersonas = getSesion().createQuery("from Personas").list();
    } finally {
        getSesion().close();
    }

    return listaPersonas;
}

```

Al finalizar se deberá crear una clase prueba en donde se ejecuten los diferentes métodos para verificar su correcto funcionamiento.

```

public class Test {

    public static void main (String[]arg){
        PersonasDAO personasDAO = new PersonasDAO();
        Personas p1 = new Personas(112540149, "Cristhian", "Garita", "Fonseca", new Date(), 1, "Prueba", "chgari", new Date());

        //personasDAO.save(p1);
        p1.setNombre("Julio");
        personasDAO.merge(p1);
    }
}

```

Al finalizar este laboratorio el proyecto deberá tener la siguiente estructura de proyecto

