PLANETARIO

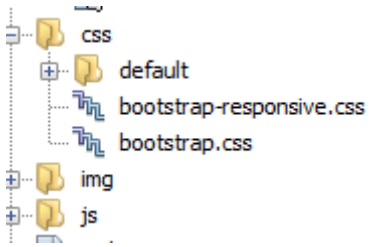
Objetivos:

La idea del proyecto és crear una aplicación web que sea capaz de recoger las tablas e información de una base de datos y procesarlas desde el mismo navegador a través de dicha aplicación.

Habrá una pantalla de login donde el usuario pueda identificarse o registrarse en caso de no tener cuenta gracias a una tabla llamada usuario que tendré en mi mysql la cual contendrá todos los datos de los usuarios

La aplicación estará compuesta por una pantalla de inicio o home, una pantalla para los planetas otra ,para ver los satelites y finalmente otra para los usuarios.

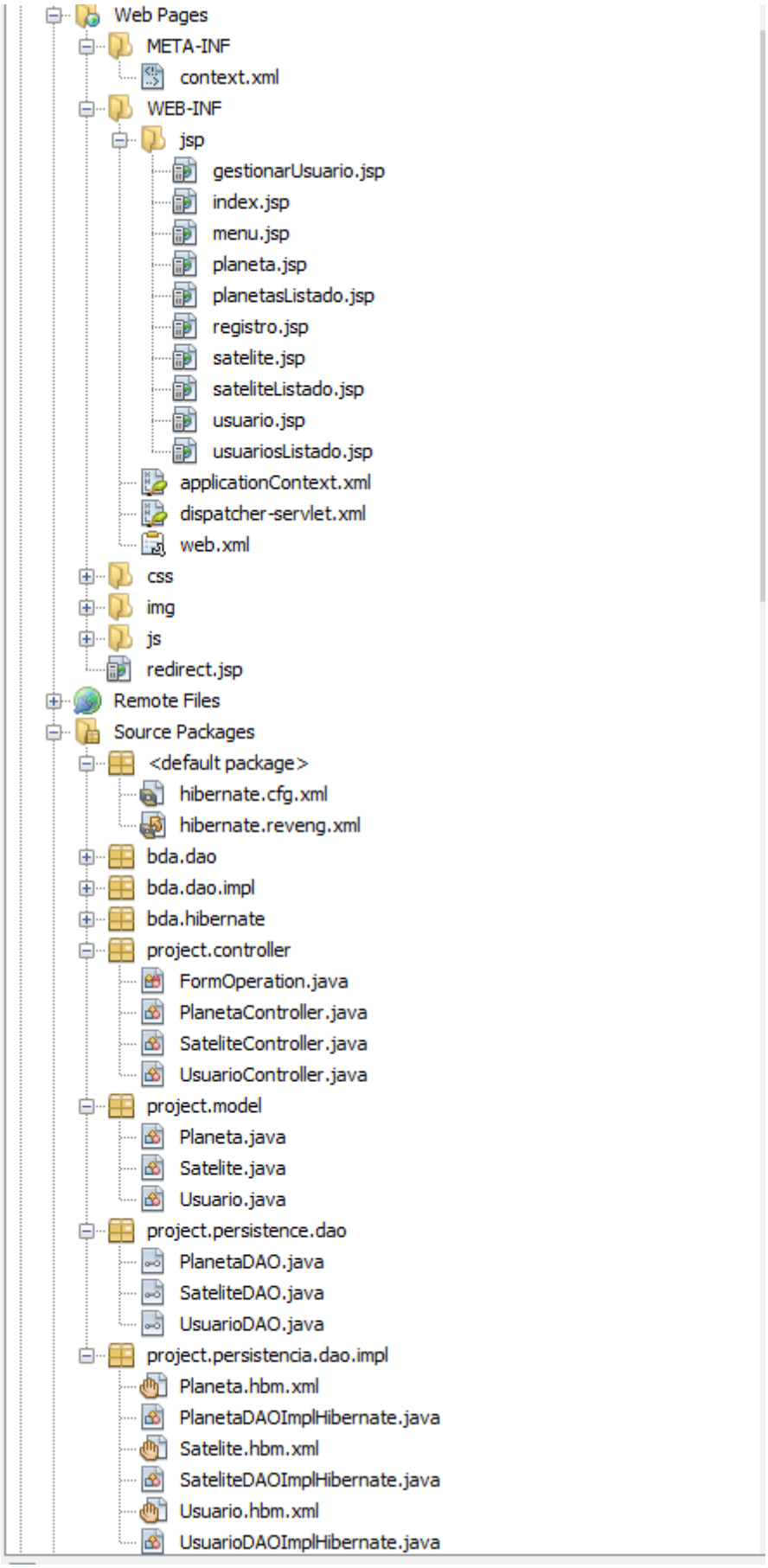
El objectivo principal de dicha aplicación es poder insertar, modificar y elimimar elementos de las tablas de la bases de datos.

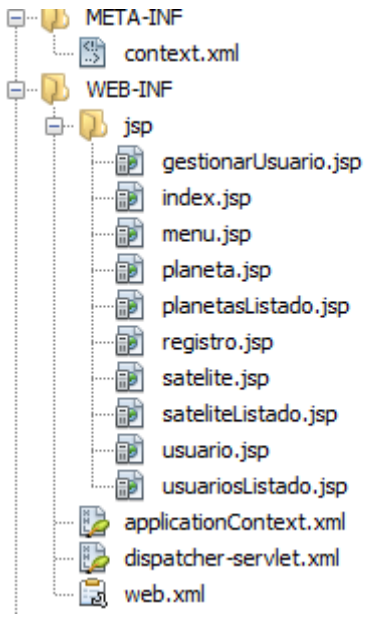


Utilización de bootstrap para el diseño de la web

Edgar Benito Carbó - 2DAM

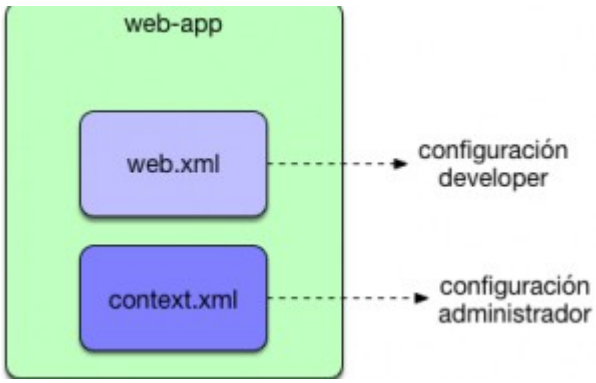
Estructura general de proyecto





- Lo primero que nos encontramos són 2 directorios los cuales contendrán 2 tipos de ficheros de configuración , que podriamos diferenciar por ficheros web y un ficheros por la parte del administrador como es el **context.xml**

-



- En el apartado de web encontramos un directorio llamado jsp y 3 ficheros: **applicationContext.xml**, **dispatcher-servlet.xml** y **web.xml.**

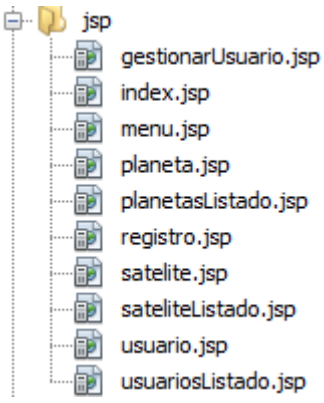
- **dispatcher-servlet:** és la pieza central de cualquier spring, el cual utilizaré para enviar las peticiones a los componentes asignados para manejarlas.

```
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"

    http://www.springframework.org/schema/aop http:
    http://www.springframework.org/schema/tx http:/
    http://www.springframework.org/schema/mvc http:
    http://www.springframework.org/schema/context h
```

```
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="index.htm">indexController</prop>
        </props>
    </property>
</bean>
```

```
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/"
      p:suffix=".jsp" />
```

```
<bean name="indexController"
      class="org.springframework.web.servlet.mvc.ParameterizableViewController"
      p:viewName="index" />
```



- En **jsp** tenemos todos los ficheros jsp.

- Un fichero **JSP** és un fichero **HTML** en el que podemos incluir código **Java**

- A continuación veremos todos los ficheros en detalle y explicaremos su funcionalidad en este proyecto .

1. El fichero **index.jsp** será en primero que veremos al ejecutar la aplicación, es decir será la página inicial del proyecto en mi caso he hecho uso de ella para realizar un **login** en el que los usuarios existentes puedan validarse y en caso de que no estén dados de alta tengan la opción de **registrarse.**

- El código sería el siguiente:

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" 
    <script type="text/javascript"  src="<%=request.getContextPath()%>/js/jquery-1.9.0.js"></script>
    <script type="text/javascript"  src="<%=request.getContextPath()%>/js/bootstrap.js" ></script>
    <script src="js/main.js"></script>
    <title>Iniciar sesión</title>
```

```
<form action="<%=urlAction%>" method="post" id="formInicio">
    <fieldset>
        <div class="form-group">
            <label class="control-label" for="nombre">Usuario:</label>
            <input class="form-control" id="txtUsuario" type="text" name="usuario" "/><br>
        </div>
        <div class="form-group">
            <label class="control-label" for="password">Contraseña:</label>
            <input class="form-control" id="txtPassword" type="password" name="password" "/>
        </div>
    </fieldset>
    <div class="form-actions">
        <input type="button" value="Iniciar Sesión" id="btnIniciarSesion"/>
    </div>
        No tienes una cuenta... <a href="registro.jsp">Registrarme</a>
</form>
```

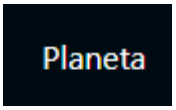
```
<%
    FormOperation formOperation = (FormOperation) request.getAttribute("formOperation");
    String labelButton = "";
    String urlAction = "";
    switch (formOperation) {
        case INSERT:
            labelButton = "Insertar";
            urlAction = request.getContextPath() + "/usuario/insert.html";
            break;
        case LOGIN:
            labelButton = "Login";
            urlAction = request.getContextPath() + "/usuario/login.html";
            break;

    }


%>
```

2. El **menu.jsp**, será la página principal de nuestra proyecto y desde ella podremos acceder a todos los recursos de la web.

-Contendrá una barra de navegación , con enlaces , los cuales nos llevarán a la pantalla deseada en ese momento, también tendrá en la parte central estará el contenido principal de dicha ventana, en mi caso un mensaje de bienvenida al usuario y una imagen de fondo.

- Finalmente un pie de página con el nombre del autor y su fecha de creación.

3. PlanetaListado, será la página que veremos al hacer click en nuestro elemento planeta de la barra de navegación.



- En mi caso dicha página contendrá una lista con todos los planetas y todos sus atributos para que podamos consultar dichos elementos, modificarlos, eliminarlos o añadir nuevos.
- Para ello he hecho uso de un controlador de usuarios en el cual estén implementados todas las operaciones que queremos que se puedan realizar sobre los planetas.
- Como he dicho anteriormente esas operaciones serán **insertar**, **editar** y **borrar** planetas.
- Ejemplo del código:

```
@RequestMapping({"/planeta/newForInsert.html"})
public ModelAndView newForInsert(HttpServletRequest request, HttpServletResponse response) {
    Map<String, Object> model = new HashMap<String, Object>();
    String viewName;

    try {
        Planeta planeta = planetaDAO.create();
        model.put("formOperation", FormOperation.INSERT);
        model.put("planeta", planeta);

        viewName = "planeta";
    } catch (BussinessException ex) {
        model.put("bussinessMessages", ex.getBussinessMessages());
        model.put("backURL", request.getContextPath() + "/index.html");
        viewName = "error";
    }
    return new ModelAndView(viewName, model);
}
```

```java
@RequestMapping({"/planeta/insert.html"})
public ModelAndView insert(HttpServletRequest request, HttpServletResponse response)
    Map<String, Object> model = new HashMap<String, Object>();
    String viewName;

    request.setCharacterEncoding("UTF-8");

    Planeta planeta = null;
    try {
        planeta = planetaDAO.create();
        planeta.setNombre(request.getParameter("nombre"));


        planetaDAO.saveOrUpdate(planeta);

        viewName = "redirect:/planeta.html";
    } catch (BussinessException ex) {
        model.put("bussinessMessages", ex.getBussinessMessages());
        model.put("backURL", request.getContextPath() + "/index.html");
        viewName = "error";
    }
    return new ModelAndView(viewName, model);
}
```

- Haciendo uso de esta implementación conseguiremos introducir planetas a la tabla

```java
List<Planeta> planetas = planetaDAO.findAll();
```

Esta linea de código la he usado para crear una lista de planetas, haciendo uso de la clase PlanetaDAO, está clase lo único que hace es heredar de la clase GenericDAO en la cual estan implementados todos los métodos utilizados

```java
public interface GenericDAO<T,ID extends Serializable> {
    T create() throws BussinessException;
    void saveOrUpdate(T entity) throws BussinessException;
    T get(ID id) throws BussinessException;
    void delete(ID id) throws BussinessException;
    List<T> findAll() throws BussinessException;
}
```

```
FormOperation.INSERT);
```

FormOperation es una clases enum, con las operaciones:
INSERT, DELETE, UPDATE, LOGIN

```
public enum FormOperation {
    INSERT, DELETE, UPDATE, LOGIN
}
```

- El procedimiento será muy similar para el resto de operaciones, adjunto capturas del código:

```
@RequestMapping({"/planeta/delete.html"})
public ModelAndView delete(HttpServletRequest request, HttpServletResponse response) throws UnsupportedEncodingException
    Map<String, Object> model = new HashMap<String, Object>();
    String viewName;

    request.setCharacterEncoding("UTF-8");

    Planeta planeta = null;
    try {
        int id;
        id = Integer.parseInt(request.getParameter("idPlaneta"));

        planeta = planetaDAO.get(id);
        if (planeta == null) {
            throw new BussinessException(new BussinessMessage(null, "Ya no existe el planeta a borrar"));
        }

        planetaDAO.delete(id);

        viewName = "redirect:/planeta.html";
    } catch (BussinessException ex) {
        model.put("bussinessMessages", ex.getBussinessMessages());
        model.put("planeta", planeta);
        model.put("formOperation", FormOperation.DELETE);
        viewName = "planeta";
    }
    return new ModelAndView(viewName, model);
}
```

- Busca el planeta por el id y cuando los encuentra lo elimina de la base de datos y de la tabla

```java
@RequestMapping({"/planeta/update.html"})
public ModelAndView update(HttpServletRequest request, HttpServletResponse response) throws UnsupportedE
    Map<String, Object> model = new HashMap<String, Object>();
    String viewName;

    request.setCharacterEncoding("UTF-8");

    Planeta planeta = null;
    try {
        int id;
        id = Integer.parseInt(request.getParameter("idPlaneta"));

        planeta = planetaDAO.get(id);
        if (planeta == null) {
            throw new BussinessException(new BussinessMessage(null, "Ya no existe el planeta"));
        }
        planeta.setNombre(request.getParameter("nombre"));


        planetaDAO.saveOrUpdate(planeta);

        viewName = "redirect:/planeta.html";
    } catch (BussinessException ex) {
        model.put("bussinessMessages", ex.getBussinessMessages());
        model.put("backURL", request.getContextPath() + "/index.html");
        viewName = "error";
    }
    return new ModelAndView(viewName, model);
}
```

- Busca el planeta por el id y cuando los encuentra lo actualiza en la base de datos y en la tabla
- Visualización en nuestro navegador de como quedaría la tabla de los planetas:

Nuevo Planeta

IdPlaneta	Nombre		
57	Urano	Editar	Borrar
67	Mercurio	Editar	Borrar
68	Tierra	Editar	Borrar
71	Venus	Editar	Borrar
80	jk-45	Editar	Borrar
85	Jupiter	Editar	Borrar
86	Saturno	Editar	Borrar
89	Neptuno	Editar	Borrar
90	Marte	Editar	Borrar

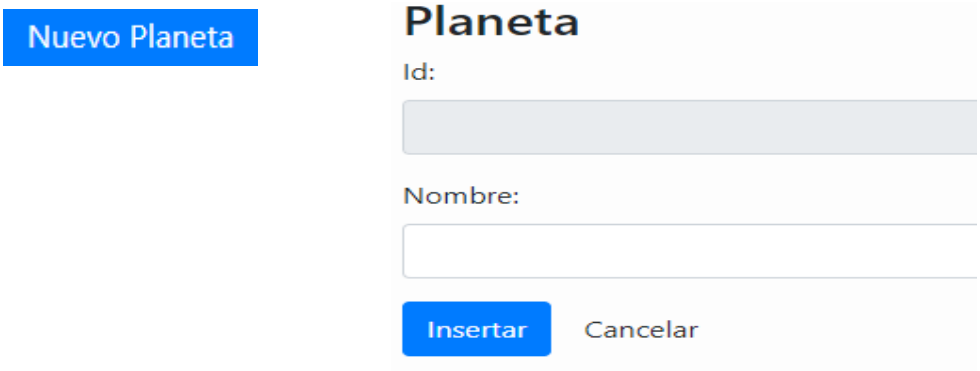
4. Planeta.jsp, será el formulario al que se nos moveremos cuando queramos insertar, modificar o eliminar un planeta.

- Implementación en Netbeans:

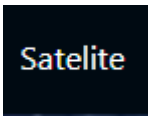
```
<%
    FormOperation formOperation = (FormOperation) request.getAttribute("formOperation");
    String labelButton = null;
    String urlAction = null;
    switch (formOperation) {
        case INSERT:
            labelButton = "Insertar";
            urlAction = request.getContextPath() + "/planeta/insert.html";
            break;
        case UPDATE:
            labelButton = "Actualizar";
            urlAction = request.getContextPath() + "/planeta/update.html";
            break;
        case DELETE:
            labelButton = "Borrar";
            urlAction = request.getContextPath() + "/planeta/delete.html";
            break;
        default:
            throw new RuntimeException("El valor de 'formOperation' no es valido" + formOperation);
    }
%>
```

```
<form action="<%=urlAction%>" method="post" >
    <fieldset>
        <div class="form-group">
            <label class="control-label" for="id">Id:</label>
            <input class="form-control disabled " id="idPlaneta" name="idPlaneta" type="text" v
        </div>

        <div class="form-group">
            <label class="control-label" for="nombre">Nombre:</label>
            <input class="form-control" id="nombre" type="text" name="nombre" value="${planeta.
        </div>
    </fieldset>

    <div class="form-actions">
        <button id="aceptarBtn" class="btn btn-primary" type="submit"><%=labelButton%></button>
        <a class="btn" href="<%=request.getContextPath()%>/planeta.html" >Cancelar</a>
    </div>
</form>
```

- Visualización en el navegador:



5. sateliteListado.jsp, al igual que planetaListado, este fichero contendrá una tabla con todos los satelites almacenados en la base de datos, una barra de navegación y un pie de página.



- Implementación en Netbeans:
- Bucle en el que se recorren todos los satélites de la tabla

```
<%
    for (Satelite satelite : satelites) {
%>
<tr>
    <td><%=satelite.getIdSatelite()%></td>
    <td><%=HtmlUtils.htmlEscape(satelite.getPeriodo())%></td>
    <td><%=HtmlUtils.htmlEscape(satelite.getMedida())%></td>
    <td><%=HtmlUtils.htmlEscape(satelite.getPeso())%></td>
    <td>
        <a href="<%=request.getContextPath()%>/satelite/readForUpdate.html?id=<%=satelite.getIdSatelite()%>"
    </td>
    <td>
        <a href="<%=request.getContextPath()%>/satelite/readForDelete.html?id=<%=satelite.getIdSatelite()%>"
    </td>
</tr>
<%
    }
%>
```

- Barra de navegación

```
<nav class="navbar navbar-expand-lg" style="background:#000a12">
    <a class="navbar-brand" href="#">Home</a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item">
                <a class="nav-link" href="<%=request.getContextPath()%>/planeta.html"
            </li>
            <li class="nav-item active">
                <a class="nav-link" href="<%=request.getContextPath()%>/satelite.html"
            </li>
            <li class="nav-item">
                <a class="nav-link" href="<%=request.getContextPath()%>/usuario.html"
            </li>
        </ul>
    </div>
</nav>
```

- Pie de página

```
<footer class="bg-dark text-center text-lg-start fixed-bottom">
    <div class="text-center p-3 text-light" style="background:#000a12" >
        <h5>Edgar Benito Carbó - 12/02/2021</h5>
    </div>
</footer>
```

- Visualización de la tabla en el navegador:

Nuevo Satelite

IdSatelite	Periodo	Medida	Peso		
1	Lunar	500	1550000	Editar	Borrar
7	Lunar	400	2520000	Editar	Borrar
8	Lunar	300	2520000	Editar	Borrar
9	Lunar	250	2520000	Editar	Borrar

6. satelite.jsp, és el formulario para insertar, editar, o elimiar satelites de la base de datos:

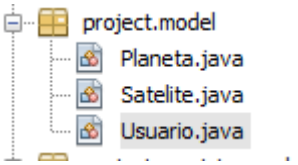
- Implementación:

```
<form action="<%=urlAction%>" method="post" >
    <fieldset>
        <div class="form-group">
            <label class="control-label" for="idSatelite">IdSatelite</label>
            <input class="form-control disabled " id="idSatelite" name="idSatelite" type="text" value="${satel
        </div>

        <div class="form-group">
            <label class="control-label" for="periodo">Periodo</label>
            <input class="form-control" id="periodo" type="text" name="periodo" value="${satelite.periodo}" >
        </div>

        <div class="form-group">
            <label class="control-label" for="medida">Medida</label>
            <input class="form-control" id="medida" type="text" name="medida" value="${satelite.medida}" >
        </div>

        <div class="form-group">
            <label class="control-label" for="peso">Peso</label>
            <input class="form-control" id="peso" type="text" name="peso" value="${satelite.peso}" >
        </div>


    </fieldset>

    <div class="form-actions">
        <button id="aceptarBtn" class="btn btn-primary" type="submit"><%=labelButton%></button>
        <a class="btn" href="<%=request.getContextPath()%>/satelite.html" >Cancelar</a>
    </div>
</form>
iv>
```

- Navegador:

7. Por último nos quedaría echarle un vistazo a los ficheros de configuración y mapeo para las tablas y sus respectivas clases:

- Clases:




```java
public class Planeta  implements java.io.Serializable {


    private Integer idPlaneta;
    private String nombre;
```

```java
public class Satelite  implements java.io.Serializable {


    private Integer idSatelite;
    private String periodo;
    private String peso;
    private String medida;
```

```java
public class Usuario  implements java.io.Serializable {


    private Integer idUsuario;
    private String usuario;
    private String password;
    private String tipoUsuario;
```

- Configuración .cfg y .reveng:

```xml
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.D
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/
    <property name="hibernate.connection.username">astro</property>
    <property name="hibernate.connection.password">astro</property>
    <mapping resource="project/persistencia/dao/impl/Usuario.hbm.xml"/>
    <mapping resource="project/persistencia/dao/impl/Planeta.hbm.xml"/>
    <mapping resource="project/persistencia/dao/impl/Satelite.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

```xml
<hibernate-reverse-engineering>
  <schema-selection match-catalog="astrodb"/>
  <table-filter match-name="planeta"/>
  <table-filter match-name="usuario"/>
  <table-filter match-name="satelite"/>
</hibernate-reverse-engineering>
```

Utilizaremos dichos ficheros para establecer la conexión con la base de datos con el ususario y la contraseña indicados en el mismo.

- Ficheros de mapeo:
  PLANETA

```
<hibernate-mapping>
    <class catalog="astrodb" name="project.model.Planeta" optimi
        <id name="idPlaneta" type="java.lang.Integer">
            <column name="idPlaneta"/>
            <generator class="identity"/>
        </id>
        <property name="nombre" type="string">
            <column length="16" name="nombre"/>
        </property>
        <set name="satelites" cascade="all" inverse="true">
            <key>
                <column name="idPlaneta"/>
            </key>
            <one-to-many class="modelo.Satelite"/>
        </set>
    </class>
</hibernate-mapping>
```

SATÉLITE

```
<hibernate-mapping>
    <class catalog="astrodb" name="project.model.Satelite" optim
        <id name="idSatelite" type="java.lang.Integer">
            <column name="idSatelite"/>
            <generator class="identity"/>
        </id>
        <property name="periodo" type="string">
            <column length="8" name="periodo" not-null="true"/>
        </property>
        <property name="peso" type="string">
            <column length="8" name="peso" not-null="true"/>
        </property>
        <property name="medida" type="string">
            <column length="8" name="medida" not-null="true"/>
        </property>
        <many-to-one name="planeta">
            <column name="idPlaneta"/>
        </many-to-one>
    </class>
</hibernate-mapping>
```

Relación 1 a M entre planetas y satétiles

USUARIO

```
<hibernate-mapping>
  <class catalog="astrodb" name="project.model.Usuario" optimistic-lock="
    <id name="idUsuario" type="java.lang.Integer">
      <column name="idUsuario"/>
      <generator class="identity"/>
    </id>
    <property name="usuario" type="string">
      <column name="usuario" not-null="true"/>
    </property>
    <property name="password" type="string">
      <column name="password" not-null="true"/>
    </property>
    <property name="tipoUsuario" type="string">
      <column length="12" name="tipoUsuario" not-null="true"/>
    </property>
  </class>
</hibernate-mapping>
```

DIAGRAMA DE CLASES PARA LAS TABLAS