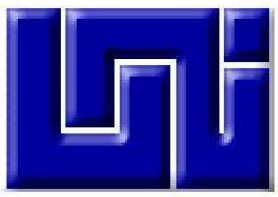
UNIVERSIDAD NACIONAL DE INGENIERIA

RECINTO UNIVERSITARIO SIMON BOLIVAR

FACULTAD DE ELECTROTECNIA Y COMPUTACION



Sistema Gestor de Contenido-Indexación e HibernateSearch (SiGeC-INIBAS).

MANUAL DE TECNICO

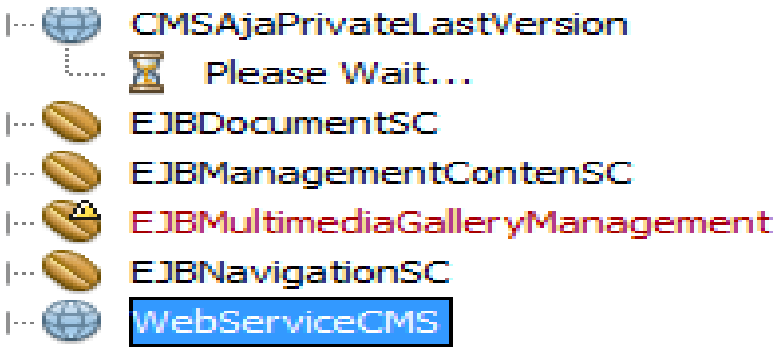
Elaborado por:

- **Br. Johanna del Carmen Vásquez Villalta.**
- **Br. Raquel Sotelo Almendarez.**

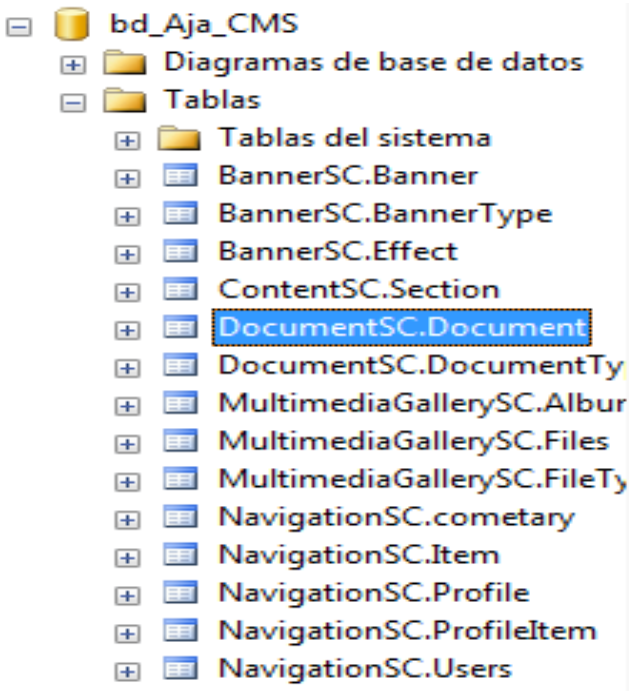
Managua, Nicaragua 27 de Noviembre del 2014

- **Estructura del Sistema:**
  - El sistema se creará bajo el lenguaje de programación Java, haciendo uso del conocido **Framework JSF** (Java Server Faces).
  - Dispone de secciones jerárquicas, se crearán varios módulos independientes el uno del otro, haciendo uso de los conocidos **Enterprise Java Beans** (EJB).

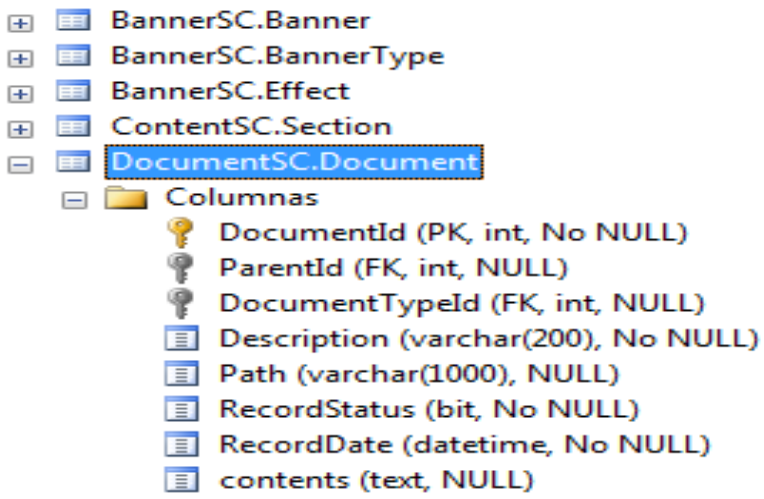
En la imagen siguiente podemos ver una estructura inicial del Sistema.
En esta imagen podemos observar al menos 4 proyectos EJB. El primero EJBDocumenSC, hace referencia al módulo que contiene la capa de datos para el esquema DocumenSC, mismo nombre que tenemos en Sql Server para el esquema correspondiente.



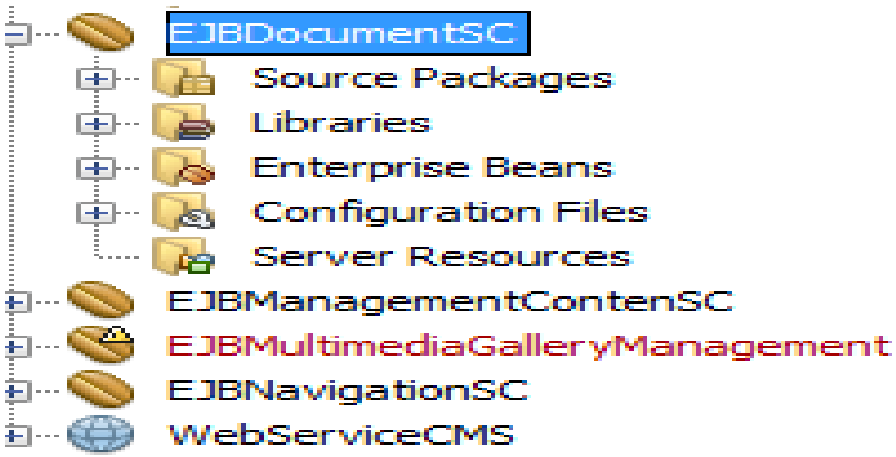
Podemos observar en la siguiente imagen el esquema DocumentSC:



Luego tenemos los atributos de la Clase Document:

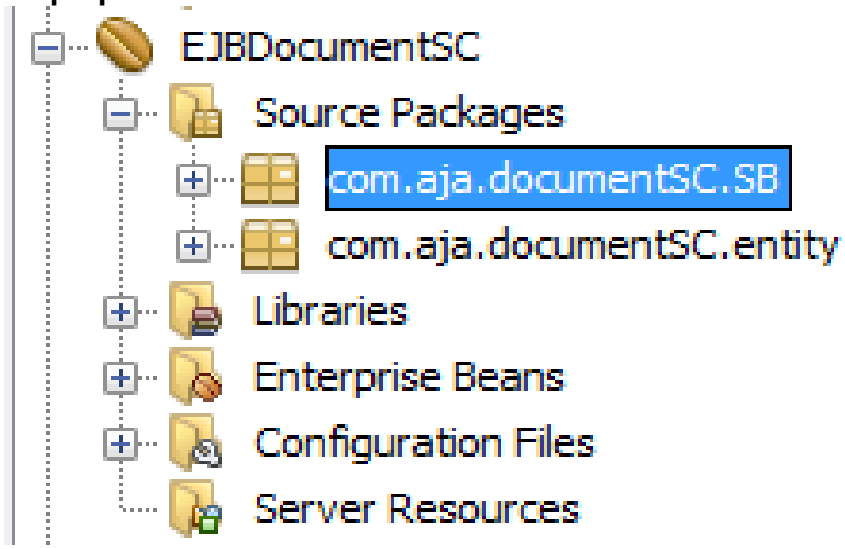


- **Estructura de EJBDocumentSC:**

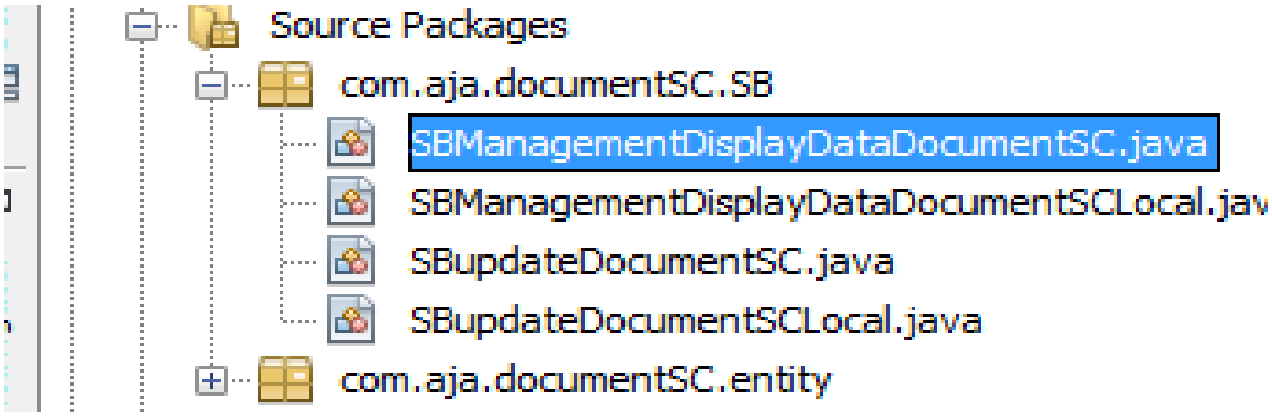


El paquete denominado Source Package alberga las clases desarrolladas por el programador:

1. **Ubicamos dos paquetes conteniendo clases:**

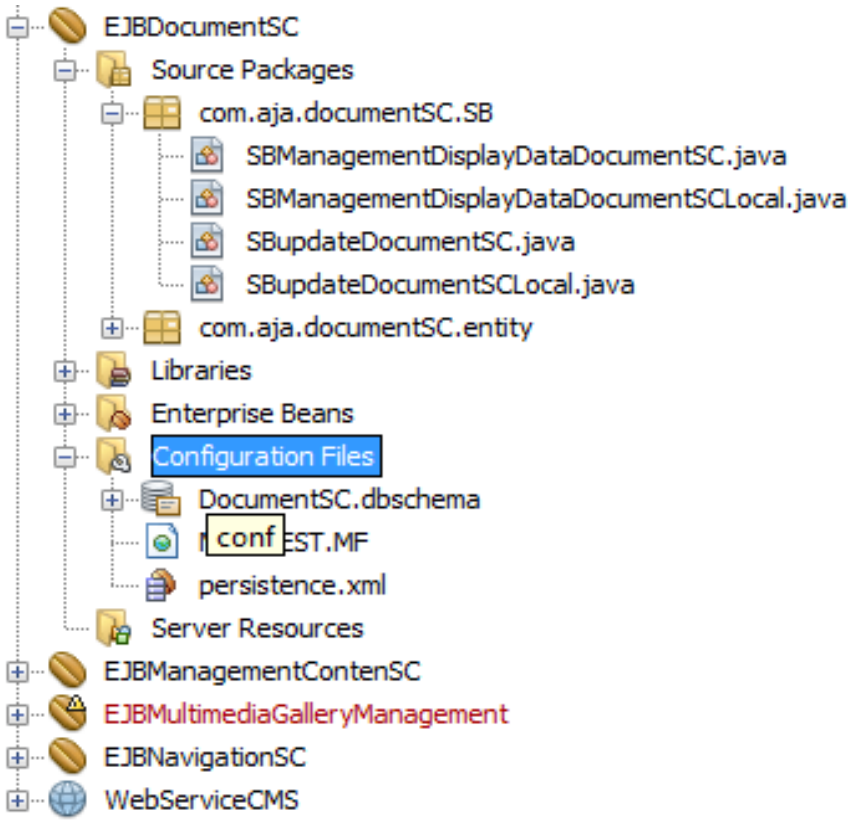


a. Com.aja.documentSC.SB contiene los Session Bean que se crearon para este módulo:

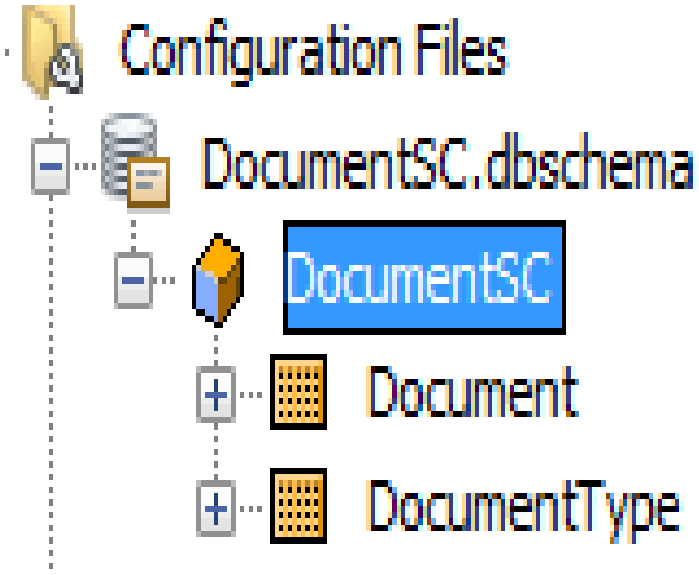


Se crearon dos Session Bean, los dos que se observamos en el paquete que llevan el prefijo Local fueron generados a partir de los dos que creo el programador.

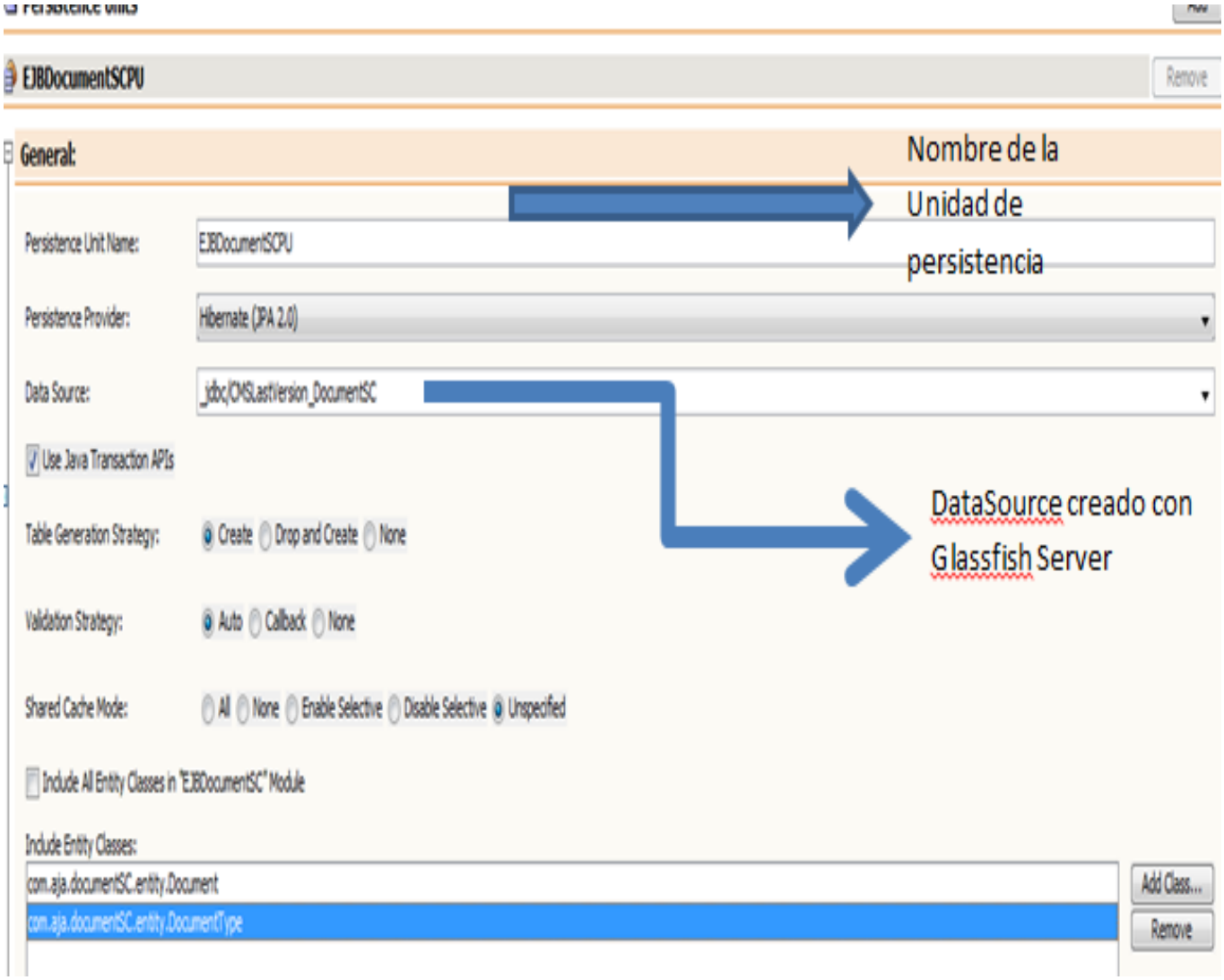
Los Session Bean se alimentan de las clases que se mapearon usando Hibernate. Una parte del este módulo es el segmento de Archivos de configuración, donde tenemos el Esquema correspondiente a este módulo (DocumentSC), y también se ubica el archivo xml que contiene la configuración de nuestra conexión y mapeo de las clases de la base de datos.



❋ **Esquema extendido:**



❋ **Vista gráfica del archivo de persistencia en formato XML:**



- **Documento XML de configuración:**

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="EJBDocumentSCPU" transaction-type="JTA">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>_jdbc/CMSLastVersion_DocumentSC</jta-data-source>
    <class>com.aja.documentSC.entity.Document</class>
    <class>com.aja.documentSC.entity.DocumentType</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>
    <properties>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
  </persistence-unit>
</persistence>
```

Ejemplo de uno de los Sesión bean:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package com.aja.documentSC.SB;

import com.aja.documentSC.entity.Document;
import com.aja.documentSC.entity.DocumentType;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.PersistenceUnit;
import javax.persistence.Query;

/**
 *
 * @author jvasquez
 */
@Stateless
public class SBManagementDisplayDataDocumentSC implements
SBManagementDisplayDataDocumentSCLocal {
@PersistenceUnit(unitName="EJBDocumentSCPU")
private EntityManagerFactory emf =null;
    private EntityManager getEntityManager(){
        return emf.createEntityManager();
    }
 public List<Document> getDocuments(){
     List<Document> lst=new ArrayList<Document>(){};
     try{
```

Clases mapeadas mediante Hibernate.

Características necesarias para aplicar persistencia.

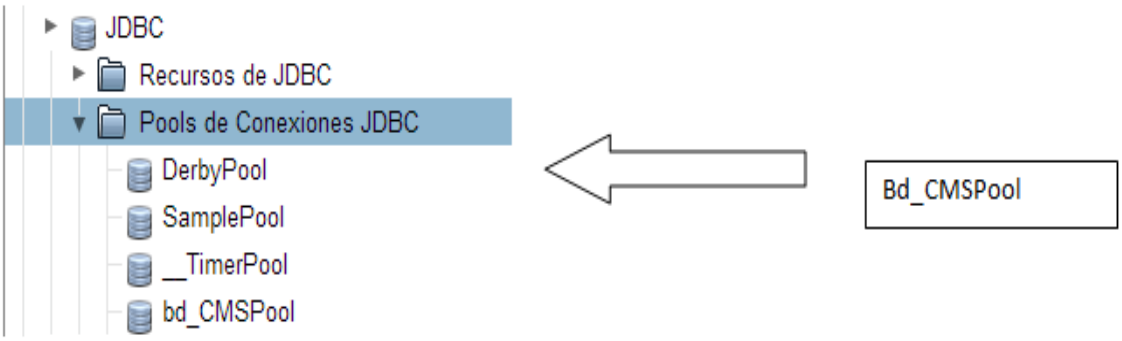
Unidad de Persistencia…

```
        Query q=getEntityManager().createQuery("SELECT d FROM Document d");
        q.setFirstResult(0);
        lst=q.getResultList();
    }catch(Exception ex){
        System.out.println("Ha Ocurrido un Error: " + ex);
    }
    return lst;
}
 public List<Document> getDocumentsByOrderDate(){
     List<Document> lst=new ArrayList<Document>(){};
     try{
         Query q=getEntityManager().createQuery("SELECT d FROM Document d ORDER BY
d.recordDate DESC");
         q.setFirstResult(0);
         lst=q.getResultList();
     }catch(Exception ex){
         System.out.println("Ha Ocurrido un Error: " + ex);
     }

       return lst;
}
 public boolean searchDocumentByDescription(String des){

     boolean var=false;
     try{
         Query q=getEntityManager().createQuery("SELECT d FROM Document d");
         List<Document> lst=q.getResultList();
         for (int i = 0; i < lst.size(); i++) {
             Document section = lst.get(i);

if((section.getDescription().toUpperCase().equals(des.toUpperCase()))||(section.getDescription().toUp
perCase().startsWith(des.toUpperCase()))||(section.getDescription().toUpperCase().endsWith(des.to
UpperCase()))||(section.getDescription().toUpperCase().contains(des.toUpperCase()))){
                 var=true;
                 break;
             }
         }
      }catch(Exception ex){
         System.out.println("Ha Ocurrido un Error: " + ex);
      }
     return var;
}
 public boolean searchDocumentByDescriptionUnique(String descrition){

     boolean var=false;
     try{
         Query q=getEntityManager().createQuery("SELECT d FROM Document d WHERE
d.description = :description").setParameter("description", descrition);

         Document document=(Document) q.getSingleResult();
         if(document!=null){
             var=true;
```

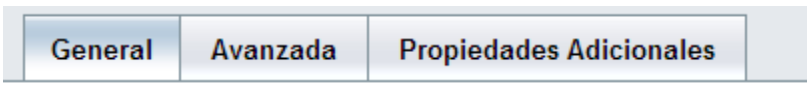
```
            }
        }catch(Exception ex){
            System.out.println("Ha Ocurrido un Error: " + ex);
        }
        return var;
    }
    public List<DocumentType> getDocumentType(){
        List<DocumentType> lst=new ArrayList<DocumentType>(){};
        try{
            Query q=getEntityManager().createQuery("SELECT d FROM DocumentType d ORDER BY
d.documentTypeId DESC");
            q.setFirstResult(0);
            lst=q.getResultList();
        }catch(Exception ex){
            System.out.println("Ha Ocurrido un Error: " + ex);
        }

        return lst;
    }

    public List<Document> getChildsByParents(int parentId){
        EntityManager entitymanager = getEntityManager();
        List<Document> lst=new ArrayList<Document>(){};
        try{
        Document parents=entitymanager.find(Document.class, parentId);
        Query q=getEntityManager().createQuery("SELECT d FROM Document d WHERE d.parentId =
:parentId").setParameter("parentId", parents);
        lst=   q.getResultList();

        }catch(Exception ex){
            System.out.println("Error---  "+ex);
        }

        return lst;
    }

     public List<Document> getParents(){
        List<Document> lst=new ArrayList<Document>(){};
        try{
           List<Document> q=getEntityManager().createQuery("SELECT d FROM Document
d").getResultList();
            for (int i = 0; i < q.size(); i++) {
                Document document = q.get(i);
                if(document.getParentId()==null){
                    lst.add(document);
                }
            }
        }catch(Exception ex){
            System.out.println("Error: " + ex);
        }

        return lst;
```

```
  }
  public List<Document> getChilds(){
     List<Document> lst=new ArrayList<Document>(){};
     try{
        List<Document> q=getEntityManager().createQuery("SELECT d FROM Document
d").getResultList();
         for (int i = 0; i < q.size(); i++) {
            Document document = q.get(i);
            if(document.getParentId()!=null){
               lst.add(document);
            }
         }
     }catch(Exception ex){
         System.out.println("Error: " + ex);
     }

     return lst;
  }
}
```

El servidor web usado para este proyecto fue el denominado **GlassFish**, en él montamos los dos .war que se crearon, uno para el ambiente privado y otro que representa al sitio público.
Lo primero que se hace en la consola de administración de GlassFish es crear nuestro pool de conexión apuntando a nuestra base de datos.
En la imagen podemos observar el pool que ya tenemos creado:



Para cada pool de conexión el servidor nos permite las opciones que observamos en la imagen:



En la pestaña general para este proyecto tenemos la siguiente configuración:

- **Configuración General**

Nombre de Pool: bd_CMSPool
Tipo de Recurso: javax.sql.DataSource
Se debe indicar si la clase de origen de datos implanta más de 1 interfaz.
Nombre de Clase de Origen de Datos:

Nombre de clase específico del proveedor que implanta las API DataSource y/o XADataSource

Nombre de Clase de Controlador:

Nombre de clase específico del proveedor que implanta la interfaz java.sql.Driver.

Ping: Activada

Si se activa, se hace ping en el pool durante la creación o la nueva configuración a fin de identificar y advertir si hay valores erróneos para sus atributos

Orden de Despliegue:

Especifica el orden de carga del recurso al iniciar el servidor. Los números inferiores se cargan en primer lugar.

Descripción:

Configuración de Pool

Tamaño de Pool Inicial y Mínimo: 8 Conexiones

Número mínimo e inicial de conexiones mantenidas en el pool

Tamaño de Pool Máximo: 80 Conexiones

Número máximo de conexiones que se pueden crear para responder a las solicitudes del cliente

Cantidad de Cambio de Tamaño del Pool: 2 Conexiones

Número de conexiones que se van a eliminar cuando caduque el timeout de inactividad del pool

Timeout de Inactividad: 300 Segundos

Tiempo máximo que una conexión puede permanecer inactiva en el pool

Tiempo de Espera Máximo: 60000 Milisegundos

Tiempo que espera el emisor de la llamada antes de que se envíe un mensaje de timeout de conexión

Transacción

Conexiones No Transaccionales: Activada

Devuelve conexiones que no son transaccionales

Aislamiento de Transacción:

Si no se especifica, utiliza el nivel por defecto para el controlador JDBC

Nivel de Aislamiento: Garantizado

Todas las conexiones utilizan el mismo nivel de aislamiento; necesita aislamiento de transacción

Para la pestaña avanzada se dispone de la siguiente configuración:

Editar Atributos Avanzados del Pool de Conexiones JDBC

Modifique un pool de conexiones JDBC existente. Un pool de conexiones JDBC es un grupo de conexiones reutilizables para una determinada base de datos.

Cargar Valores por Defecto

Nombre de Pool: bd_CMSPool

Timeout de Sentencia: -1 Segundos

Propiedad de timeout de una conexión para activar la finalización de las consultas demasiado largas. -1 indica que no se ha activado.

Tamaño de Caché de Sentencia: 0

El almacenamiento en caché se activa cuando se define en un valor positivo distinto de cero (por ejemplo, 10).

SQL de Inicio:

Especifique una cadena SQL para que se ejecute siempre que se cree una conexión desde el pool.

Listeners de Rastreo SQL:

Lista separada por comas de las clases que implantan la interfaz org.glassfish.api.jdbc.SQLTraceListener

Envolver Objetos JDBC: Activada

Si se establece en true, la aplicación obtendrá los objetos jdbc envueltos para Statement, PreparedStatement, CallableStatement, ResultSet y DatabaseMetaData.

Pool: Activada

Si se define en false, desactiva el pool de conexiones

Configuración de Conexión

Validar como Máximo una Vez: 0 Segundos

Especifica el intervalo de tiempo, en segundos, entre solicitudes consecutivas para validar una conexión como máximo una vez. El valor por defecto es 0, lo que significa que el atributo no está activado.

Timeout de Pérdida de Conexión: 0 Segundos

El valor 0 implica que no se detecta ninguna pérdida de conexión

Reclamación de Pérdida de Conexión:

Si está activada, la conexión perdida será reclamada por el pool después de que se produzca el timeout de la pérdida de conexión.

Timeout de Pérdida de Sentencia: 0 Segundos

El valor 0 implica que no se detecta ninguna pérdida de sentencia

Reclamación de Pérdida de Sentencia:

Si está activada, el pool reclamará la sentencia con pérdidas después de que se produzca el timeout de pérdidas de sentencia

Reintentos de Creación: 0

Número de intentos de crear una conexión nueva. 0 indica que no se han producido intentos.

Intervalo de Reintentos: 10 Segundos

Intervalo de tiempo entre los reintentos al intentar crear una conexión. Resulta efectivo cuando los reintentos de creación son mayores que 0.

Asociación Lenta: Activada

Las conexiones se asocian de forma lenta cuando se realiza una operación en ellas

Inscripción Lenta de Conexiones: Activada

Se realiza una inscripción del recurso en la transacción únicamente cuando se utiliza realmente en un método

Asociar a Thread: Activada

Cuando el mismo thread necesita una conexión, se puede volver a utilizar la conexión ya asociada a ese thread

Conexiones Coincidentes: Activada

Activa o desactiva la búsqueda de coincidencias de conexiones del pool

Uso Máximo de Conexión: 0

El pool volverá a utilizar las conexiones durante el número especificado de veces y después se cerrarán. 0 indica que no se ha activado la función.

❋ Validación de Conexión

Validación de Conexión:

☐ Necesario

Valida las conexiones y permite al servidor volver a conectar en caso de fallo

Método de Validación: table

Nombre de Tabla:

Rellenar Nombres de Tabla

Si se selecciona la validación de tabla, seleccione o introduzca el nombre de tabla.

Nombre de Clase de Validación:

Si se selecciona custom-validation, especifique el nombre de clase de validación.

Ante Cualquier Fallo:

☐ Cerrar Todas las Conexiones

Cierra todas las conexiones y vuelve a conectar en caso de fallo; en caso contrario, vuelve a conectar sólo cuando se utilice

Permitir Emisores de Llamada que No Sean de Componentes:

☐ Activada

Activa el pool que van a utilizar los emisores de llamadas que no sean de componentes como, por ejemplo, ServletFilters

Y en la que es de más interés se tiene lo siguiente:

❋ Editar Propiedades de Pool de Conexiones JDBC

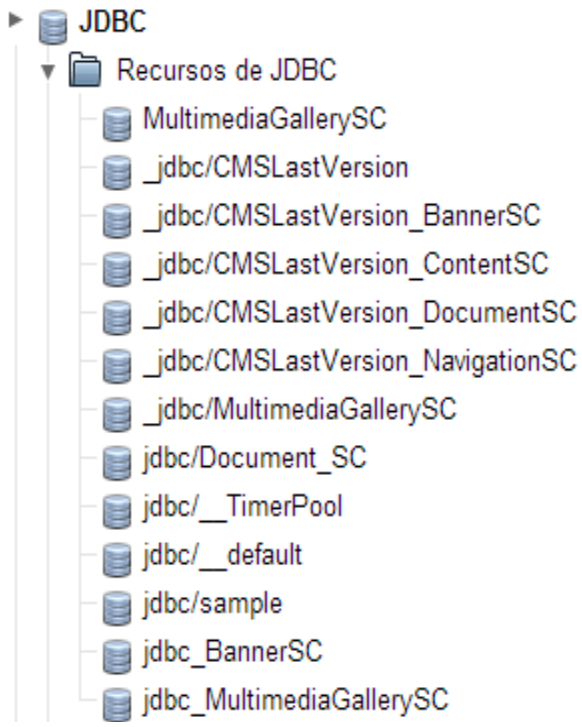
Modifique las propiedades de un pool de conexión JDBC existente.

Nombre de Pool:bd_CMSPool

Nombre	Valor	Descripción
datasourceName	com.microsoft.sqlserver.jdbc.SQLServerD	
User	sa	
DatabaseName	bd_CMS	
ApplicationName	Microsoft JDBC Driver for SQL Server	
LockTimeout	-1	
Password	Temporal1	
SendTimeAsDatetime	true	
PacketSize	8000	
SendStringParametersAsUnicode	true	

ServerName	192.168.2.233	
XopenStates	false	
TrustServerCertificate	false	
URL	jdbc:sqlserver://	
PortNumber	1433	
LoginTimeout	15	
ResponseBuffering	adaptive	
MultiSubnetFailover	false	
SelectMethod	direct	
LastUpdateCount	true	
WorkstationID	desarrollo2	
Encrypt	false	
ApplicationIntent	readwrite	

Lo siguiente que nos interesa en GlassFish es cada uno de los recursos jdbc que se crearon para alimentar las clases que se mapearon usando JPA e Hibernate. En la siguiente imagen se observan los recursos JDBC que se crearon:



- **Características de uno de estos recursos JDBC:**

Nombre JNDI: _jdbc/CMSLastVersion_DocumentSC

Nombre de Pool: bd_CMSPool

Use la página Pools de Conexiones JDBC para crear pools nuevos

Orden de Despliegue: 100

Especifica el orden de carga del recurso al iniciar el servidor. Los números inferiores se cargan en primer lugar.

Descripción:

Estado: Activada