La API JDBC

Clases DAO (Data Access Object)

Usar DAO para abstraer y encapsular todos los accesos a los datos. El DAO maneja la conexión con las fuentes de datos (como Base de Datos) para obtener y almacenar datos.

Para ejemplificar un DAO, supongamos que tenemos las clases Usuario y Hospital como parte del dominio de una aplicación. También tenemos una clase que recupera el DataSource de la aplicación.

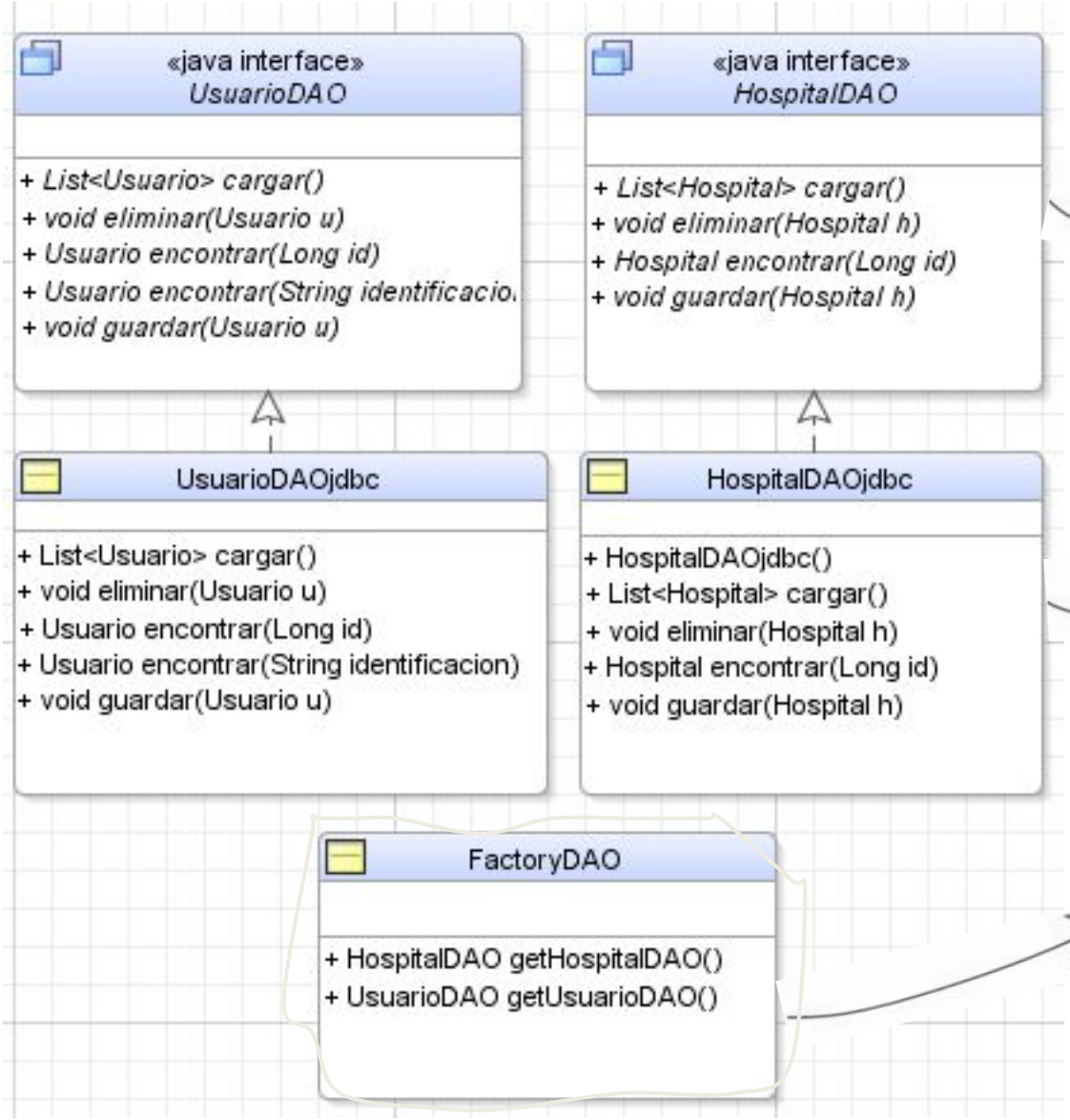
```
public class Usuario {
 Long id;
 String identificacion;
 String clave;
 String perfil;
 public String getIdentificacion() {
```

```
public class Hospital {
Long id;
String nombre;
String domicilio;
String telefonos;
public Long getId() {
  return id;
```

```
public class MiDataSource {

 private static DataSource dataSource = null;
 static {
    try {
     dataSource = (DataSource)new InitialContext().lookup("java:comp/env/jdbc/sbarra");
    } catch (NamingException e) {
     e.printStackTrace();
    }
 }
 public static DataSource getDataSource(){
    return dataSource;
 }

}
```

La API JDBC

Clases DAO



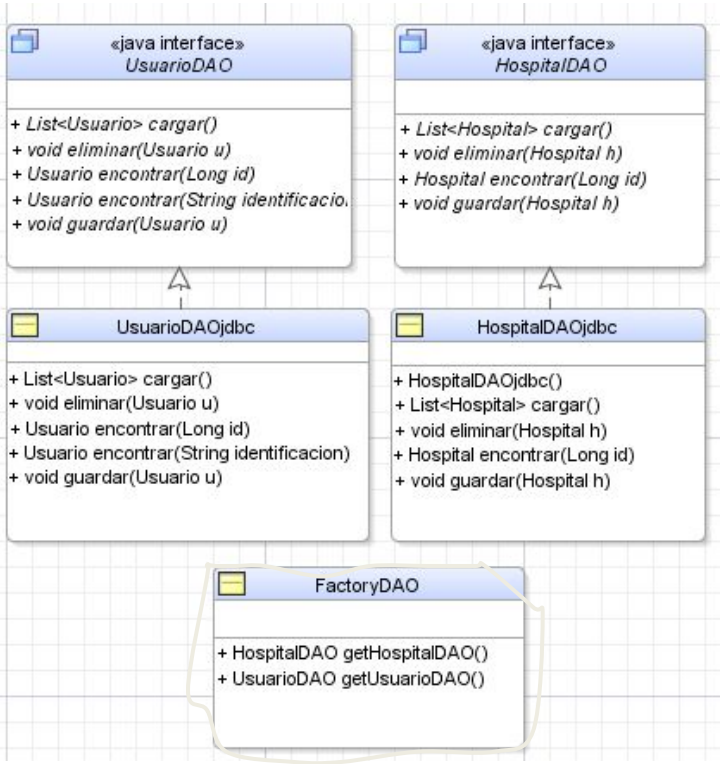
Las interfaces xxxDAO tienen operaciones comunes de acceso a datos.

Implementaciones de las interfaces xxxDAO usando JDBC

Esta clase crea objetos xxxDAO. Nos provee de objetos que implementan las distintas interfaces xxxDAO Estos objetos son usados para acceder a la capa de datos

La API JDBC

Clases DAO




```
public class FactoryDAO {
    public static UsuarioDAO getUsuarioDAO(){
        return new UsuarioDAOjdbc();
    }
    . . .
}
```

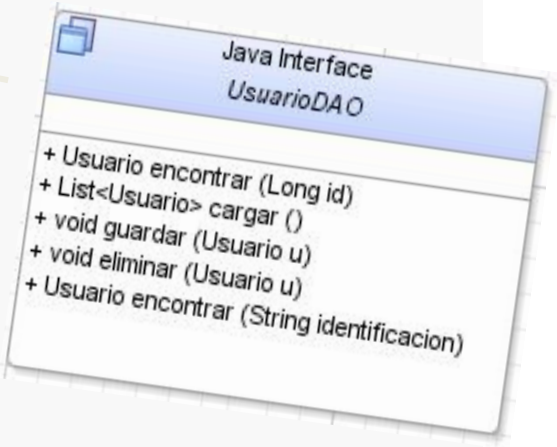
```
package dao.implJDBC;

public class UsuarioDAOjdbc implements UsuarioDAO {

public Usuario encontrar(String identificacion) {
  Usuario usuario = null;
  try{
   Connection con = MiDataSource.getDataSource().getConnection();
   Statement st = con.createStatement();
   ResultSet rs= st.executeQuery("Select u from Usuarios
     where u.identificacion='"+identificacion+"'");
     if (rs.next()==true) {
      usuario = new Usuario();
      usuario.setMatricula(rs.getInt(1));
      usuario.setApeynom(rs.getString(2));
      // más setters
      }
    rs.close();
    st.close();
    con.close();
  } catch (java.sql.SQLException e) {
    System.out.println("Error de SQL: "+e.getMessage());
  }
  return usuario;
 }

 public List<Usuario> cargar() {…}
 public void eliminar(Usuario u) {…}
 public Usuario encontrar(Long id) {…}
 public void guardar(Usuario u) {…}
}
```

La API JDBC

Clases DAO

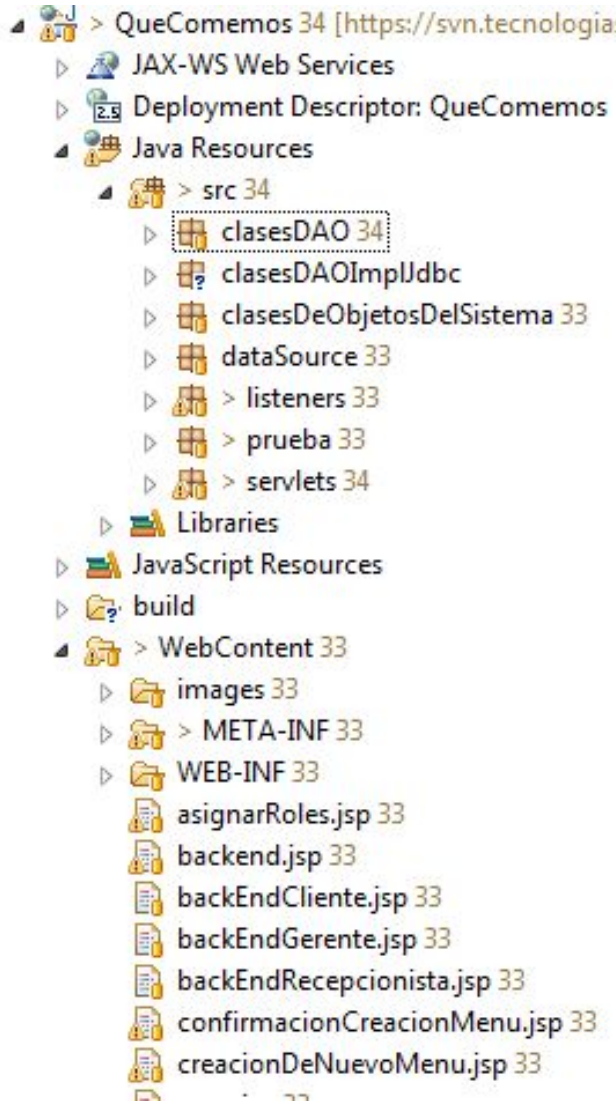
En una aplicación web tradicional, implementada con las componentes estándares de J2EE y sin framework, los Servlets podrían implementar la **lógica de negocios**. Estos comúnmente acceden a la capa de datos a través de objetos DAO y utilizan método de la interface DAO para comunicarse con la base de datos.

```
public class ServletLogin extends HttpServlet {

 public void doGet(HttpServletRequest request, HttpServletResponse response) {
    Perfil perfil = null;
    RequestDispatcher rd = null;
    UsuarioDAO uDAO = FactoryDAO.getUsuarioDAO();
    Usuario u = uDAO.encontrar(request.getParameter("identificacion"));
    if (u != null) {
        perfil=u.getPerfil();
        HttpSession sesion = request.getSession();
        sesion.setAttribute("perfil", perfil);
        result ="/mostrarMenu";
      } else
        result = "/registracion.jsp";

    rd = getServletContext().getRequestDispatcher(result);
    rd.forward(request, response);

   }
}
```



La API JDBC

Organización de las cosas



Agrupar los fuentes en paquetes:

- Paquete para las interfaces DAO
- Paquetes para las implementaciones de las interfaces DAO. Pueden existir más de una implementación.
- Paquetes para los listeners.
- Paquetes para los Servlets (por ahora).