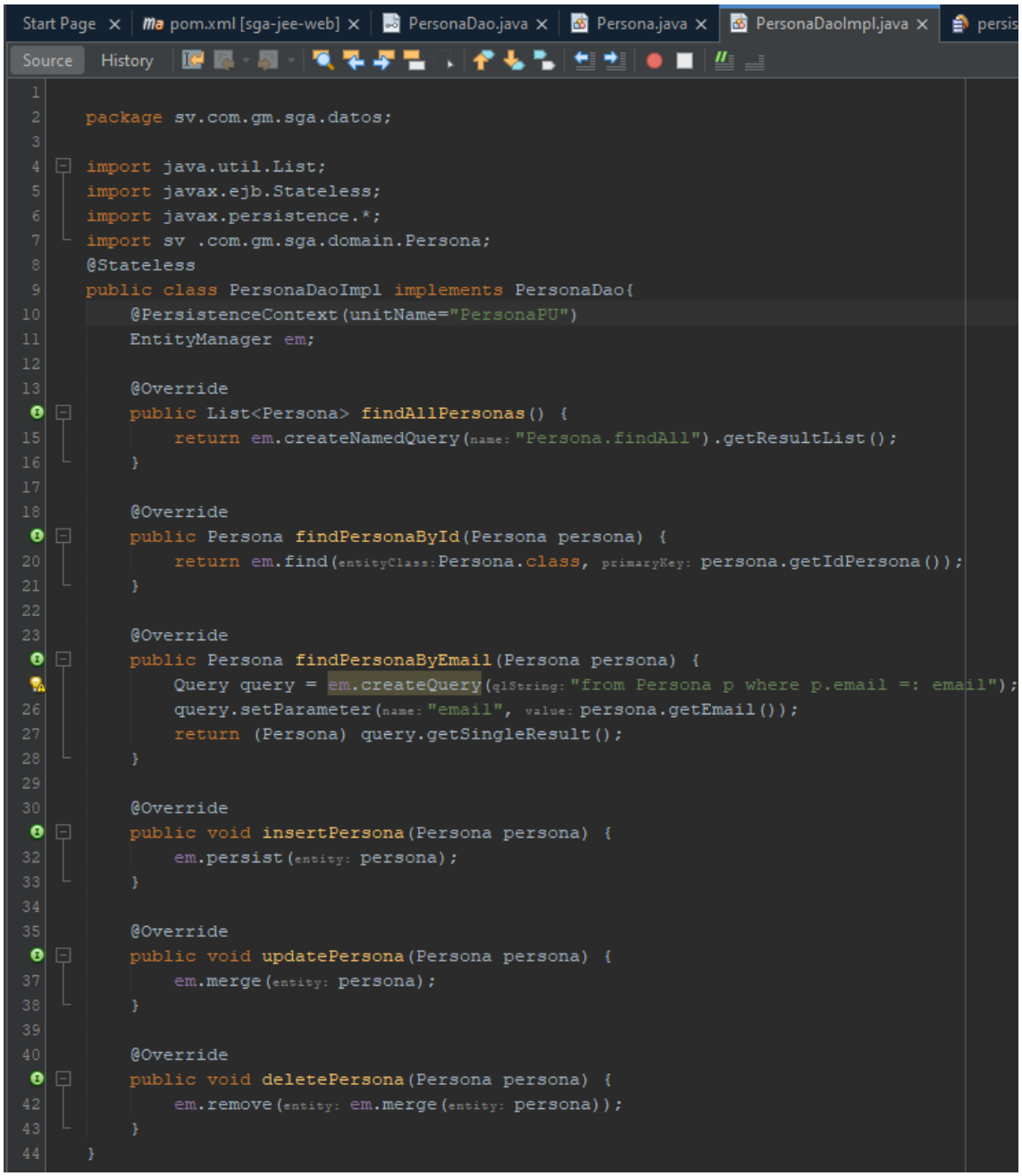
Ejercicio sgaWEB con JPA
(se modifico el ejercicio de la lección 2 “sga-jee-web”)

Dependencias:

```
<dependencies>
    <dependency>
        <groupId>javax</groupId>
        <artifactId>javaee-api</artifactId>
        <version>${jakartaee}</version>
        <scope>provided</scope>
    </dependency>
        <dependency>
        <groupId>javax</groupId>
        <artifactId>javaee-api</artifactId>
        <version>8.0.1</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>javax.persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
    <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>eclipselink</artifactId>
        <version>2.7.4</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.17</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.12.0</version>
    </dependency>
```

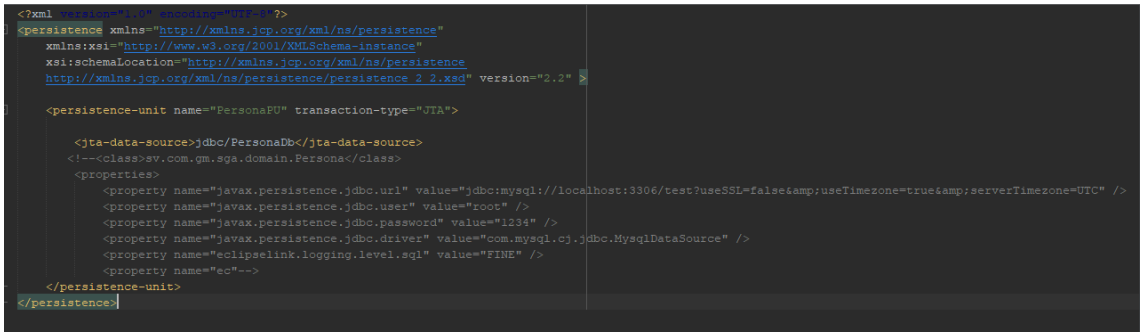
```
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.12.0</version>
    </dependency>
</dependencies>
```

Implementacion de Persona:




```java
package sv.com.gm.sga.datos;

import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.*;
import sv .com.gm.sga.domain.Persona;
@Stateless
public class PersonaDaoImpl implements PersonaDao{
    @PersistenceContext(unitName="PersonaPU")
    EntityManager em;

    @Override
    public List<Persona> findAllPersonas() {
        return em.createNamedQuery(name: "Persona.findAll").getResultList();
    }

    @Override
    public Persona findPersonaById(Persona persona) {
        return em.find(entityClass:Persona.class, primaryKey: persona.getIdPersona());
    }

    @Override
    public Persona findPersonaByEmail(Persona persona) {
        Query query = em.createQuery(qlString: "from Persona p where p.email =: email");
        query.setParameter(name: "email", value: persona.getEmail());
        return (Persona) query.getSingleResult();
    }

    @Override
    public void insertPersona(Persona persona) {
        em.persist(entity: persona);
    }

    @Override
    public void updatePersona(Persona persona) {
        em.merge(entity: persona);
    }

    @Override
    public void deletePersona(Persona persona) {
        em.remove(entity: em.merge(entity: persona));
    }
}
```

Persistence.xml



```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd" version="2.2" >

    <persistence-unit name="PersonaPU" transaction-type="JTA">

        <jta-data-source>jdbc/PersonaDb</jta-data-source>
      <!--<class>sv.com.gm.sga.domain.Persona</class>
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/test?useSSL=false&amp;useTimezone=true&amp;serverTimezone=UTC" />
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="1234" />
            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.MysqlDataSource" />
            <property name="eclipselink.logging.level.sql" value="FINE" />
            <property name="ec"-->
    </persistence-unit>
</persistence>
```

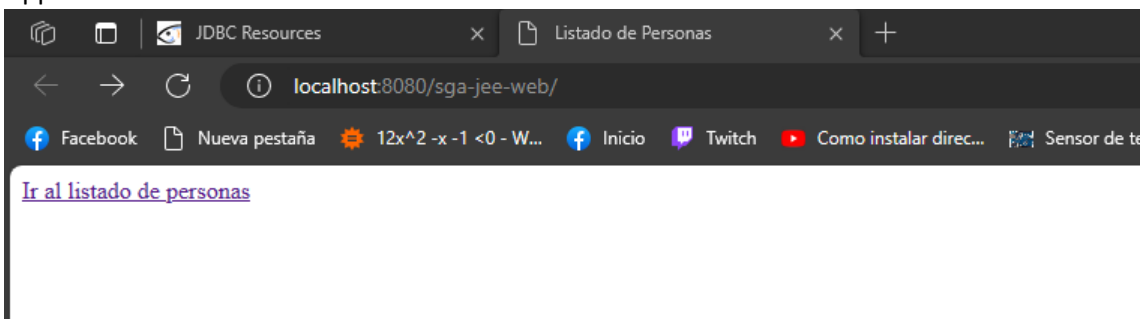
PersonaServiceImpl:

```java
@Stateless
public class PersonaServiceImpl implements PersonaServiceRemote, PersonaService{


    @Inject
    private PersonaDao personaDao;

    @Override
    public List<Persona> listarPersonas() {
       return personaDao.findAllPersonas();
    }

    @Override
    public Persona encontrarPersonaPorId(Persona persona) {
        return personaDao.findPersonaById(persona);
    }

    @Override
    public Persona encontrarPersonaPorEmail(Persona persona) {
        return personaDao.findPersonaByEmail(persona);
    }

    @Override
    public void registrarPersona(Persona persona) {
        personaDao.insertPersona(persona);
    }

    @Override
    public void modificarPersona(Persona persona) {
        personaDao.updatePersona(persona);
    }

    @Override
    public void eliminarPersona(Persona persona) {
        personaDao.deletePersona(persona);
```

PersonaServlet:

```java
package sv.com.gm.sga.web;
import java.io.IOException;
import java.util.List;
import javax.inject.Inject;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import sv.com.gm.sga.domain.Persona;
import sv.com.gm.sga.servicio.PersonaService;

@WebServlet("/personas")
public class PersonaServlet extends HttpServlet{

    @Inject
    PersonaService personaService;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
        List<Persona> personas = personaService.listarPersonas();
        System.out.println("personas:" + personas);
        request.setAttribute(name: "personas", o: personas);
        request.getRequestDispatcher(path: "/listadoPersonas.jsp").forward(request, response);
    }
}
```

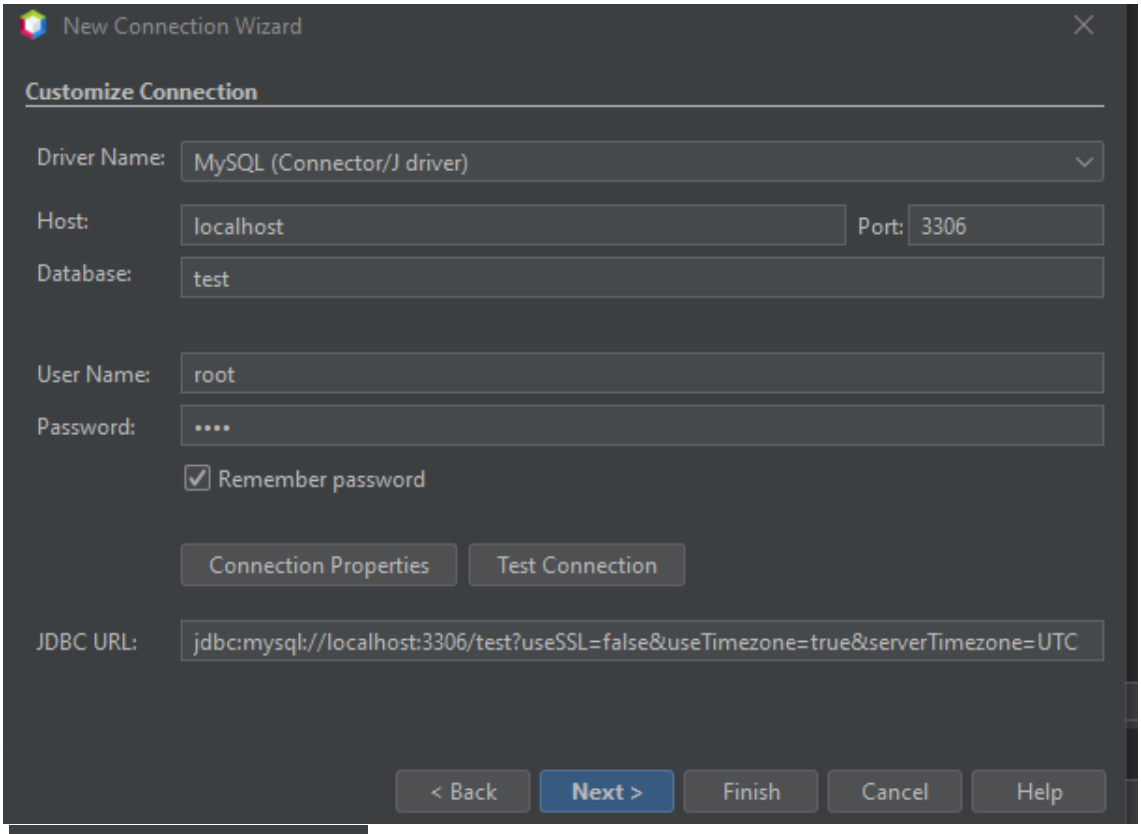
App funcionando:

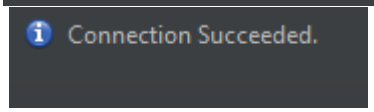


Listado de Personas

- Juan Perez
- Karla Gomez
- Maria Gutierrez

Crear una conexión de base de datos desde apache netbeans:





Creacion de clases de entidad automáticamente:

```java
package sv.com.gm.sga.domain;

import java.io.Serializable;
import java.util.Collection;
import javax.persistence.*;
import javax.validation.constraints.Size;

@Entity
@NamedQueries({
    @NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p"),
    @NamedQuery(name = "Persona.findByIdPersona", query = "SELECT p FROM Persona p WHERE p.idPersona = :idPersona"),
    @NamedQuery(name = "Persona.findByNombre", query = "SELECT p FROM Persona p WHERE p.nombre = :nombre"),
    @NamedQuery(name = "Persona.findByApellido", query = "SELECT p FROM Persona p WHERE p.apellido = :apellido"),
    @NamedQuery(name = "Persona.findByEmail", query = "SELECT p FROM Persona p WHERE p.email = :email"),
    @NamedQuery(name = "Persona.findByTelefono", query = "SELECT p FROM Persona p WHERE p.telefono = :telefono")})
public class Persona implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_persona")
    private Integer idPersona;
    @Size(max = 45)
    private String nombre;
    @Size(max = 45)
    private String apellido;
    // @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])
    @Size(max = 45)
    private String email;
    @Size(max = 45)
    private String telefono;
    @OneToMany(mappedBy = "persona")
    private Collection<Usuario> usuarioCollection;

    public Persona() {
    }
```

```java
public Persona(String nombre, String apellido, String email, String telefono) {
    this.nombre = nombre;
    this.apellido = apellido;
    this.email = email;
    this.telefono = telefono;
}



public Persona(Integer idPersona) {
    this.idPersona = idPersona;
}

public Integer getIdPersona() {
    return idPersona;
}

public void setIdPersona(Integer idPersona) {
    this.idPersona = idPersona;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
```

```java
    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public Collection<Usuario> getUsuarioCollection() {
        return usuarioCollection;
    }

    public void setUsuarioCollection(Collection<Usuario> usuarioCollection) {
        this.usuarioCollection = usuarioCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPersona != null ? idPersona.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Persona)) {
            return false;
        }
        Persona other = (Persona) object;
        if ((this.idPersona == null && other.idPersona != null) || (this.idPersona != null && !this.idPersona.equals(obj:other.idPersona))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Persona{" + "idPersona=" + idPersona + ", nombre=" + nombre + ", apellido=" + apellido + ", email=" + email + ", telefono=" + telefono + '}';
    }
}
```

Clase usuario:

```java
package sv.com.gm.sga.domain;

import java.io.Serializable;
import javax.persistence.*;
import javax.validation.constraints.Size;

@Entity
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u FROM Usuario u WHERE u.idUsuario = :idUsuario"),
    @NamedQuery(name = "Usuario.findByUsername", query = "SELECT u FROM Usuario u WHERE u.username = :username"),
    @NamedQuery(name = "Usuario.findByPassword", query = "SELECT u FROM Usuario u WHERE u.password = :password")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_usuario")
    private Integer idUsuario;
    @Size(max = 45)
    private String username;
    @Size(max = 45)
    private String password;
    @JoinColumn(name = "id_persona", referencedColumnName = "id_persona")
    @ManyToOne
    private Persona persona;

    public Usuario() {
    }

    public Usuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public Usuario(String username, String password) {
        this.username = username;
        this.password = password;
    }
```

```java
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.idUsuario == null && other.idUsuario != null) || (this.idUsuario != null && !this.idUsuario.equals(obj:other.idUsuario))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Usuario{" + "idUsuario=" + idUsuario + ", username=" + username + ", password=" + password + ", persona=" + persona + '}';
    }
```

Laboratorio uno: realizar todas las capas para la entidad usuario:

UsuarioDaoImpl:

```java
package sv.com.gm.sga.datos;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.*;
import sv .com.gm.sga.domain.Persona;
import sv.com.gm.sga.domain.Usuario;

@Stateless
public class UsuarioDaoImpl implements UsuarioDao{
    @PersistenceContext(unitName="PersonaPU")
    EntityManager em;

    @Override
    public List<Usuario> findAllUsuarios() {
        return em.createNamedQuery(name: "Usuario.findAll").getResultList();
    }

    @Override
    public Usuario findUsuarioById(Usuario usuario) {
        return em.find(entityClass:Usuario.class, primaryKey: usuario.getIdUsuario());
    }

    @Override
    public Usuario findUsuarioByUsername(Usuario usuario) {
        Query query = em.createQuery(qlString: "from Usuario p where p.username =: username");
        query.setParameter(name: "username", value: usuario.getUsername());
        return (Usuario) query.getSingleResult();
    }

    @Override
    public void insertUsuario(Usuario usuario) {
        em.persist(entity: usuario);
    }

    @Override
    public void updateUsuario(Usuario usuario) {
        em.merge(entity: usuario);
    }

    @Override
    public void deleteUsuario(Usuario usuario) {
        em.remove(entity: usuario);
    }
```

UsuarioServiceImpl:

```java
import java.util.List;
import javax.ejb.Stateless;
import javax.inject.Inject;
import sv.com.gm.sga.datos.UsuarioDao;
import sv.com.gm.sga.domain.Usuario;
@Stateless

public class UsuarioServiceImpl implements UsuarioService{
   @Inject
    private UsuarioDao usuarioDao;
    @Override
    public List<Usuario> listarUsuarios() {
         return usuarioDao.findAllUsuarios();
    }

    @Override
    public Usuario encontrarUsuarioPorId(Usuario usuario) {
        return usuarioDao.findUsuarioById(usuario);
    }

    @Override
    public Usuario encontrarUsuarioPorUsername(Usuario usuario) {
        return usuarioDao.findUsuarioByUsername(usuario);
    }

    @Override
    public void registrarUsuario(Usuario usuario) {
        usuarioDao.insertUsuario(usuario);
    }

    @Override
    public void modificarUsuario(Usuario usuario) {
        usuarioDao.updateUsuario(usuario);

    }

    @Override
    public void eliminarUsuario(Usuario usuario) {
        usuarioDao.deleteUsuario(usuario);
    }

}
```

Servlet:

```java
package sv.com.gm.sga.web;
import java.io.IOException;
import java.util.List;
import javax.inject.Inject;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import sv.com.gm.sga.domain.Usuario;

import sv.com.gm.sga.servicio.UsuarioService;

@WebServlet("/usuarios")
public class UsuarioServlet extends HttpServlet{
        @Inject
        UsuarioService usuarioService;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException{
        List<Usuario> usuario = usuarioService.listarUsuarios();
        System.out.println("usuarios:" + usuario);
        request.setAttribute(name: "usuarios", o: usuario);
        request.getRequestDispatcher(path: "/listadoUsuarios.jsp").forward(request, response);
    }

}
```

```jsp
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Listado de usuarios</title>
    </head>
    <body>
        <h1>Listado de usuarios</h1>
        <ul>
            <c:forEach items="${usuarios}" var="usuario">
                <li>${usuario.username} ${usuario.idUsuario} </li>
            </c:forEach>
        </ul>
    </body>
</html>
```