		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2312	Práctica	1	Fecha	25/02/2024
Alumno/a		Gómez, Llaneza, Ana			
Alumno/a		de la Cruz, Cirujano, Laura			

Práctica 1: Arquitectura de Jakarta EE

Ejercicio número 1:

Prepare e inicie una máquina virtual a partir de la plantilla si2srv con: 1GB de RAM asignada, y 1 CPU. A continuación:

- Modifique los ficheros que considere necesarios en el proyecto para que se despliegue tanto la aplicación web como la base de datos contra la dirección asignada a la pareja de prácticas.

En la modificación de ficheros para que se despliegue hemos completado la variable as.host de build.propiedades con la IP de nuestra máquina virtual. Estas IP son 10.2.8.1 en el caso de Laura y 10.2.8.2 en el caso de Ana. Y las variables db.host y db.client.host del fichero postgresql.propiedades con las mismas IP que en el otro archivo.

Cuando ya tenemos estas IP configuradas desplegamos el servicio.

```
ana@ana:~/Documentos/SI/p1.1/P1-base$ ant replugar limpiar-todo unsetup-db todo
Buildfile: /home/ana/Documentos/SI/p1.1/P1-base/build.xml

replugar:
[exec] Command undeploy executed successfully.

limpiar-todo:
[delete] Deleting directory /home/ana/Documentos/SI/p1.1/P1-base/build
[delete] Deleting directory /home/ana/Documentos/SI/p1.1/P1-base/dist
delete-resource-local:
```

```

create-pool-local:
[echo] Registering jdbc-connection-pool VotoPool.
[echo] ds=org.postgresql.ds.PGConnectionPoolDataSource

create-jdbc-connection-pool:
[exec] JDBC connection pool VotoPool created successfully.
[exec] Command create-jdbc-connection-pool executed successfully.

create-resource-local:
[echo] Registering jdbc resource jdbc/VotoDB.

create-jdbc-resource:
[exec] JDBC resource jdbc/VotoDB created successfully.
[exec] Command create-jdbc-resource executed successfully.

delete-db:
[echo] driver=org.postgresql.Driver
[echo] url=jdbc:postgresql://10.2.8.2:5432/voto
[echo] user=alumnodb
[echo] password=****
[exec] dropdb: error: database removal failed: ERROR: database "voto" does not exist
[exec] Result: 1

create-db:
[sql] Executing resource: /home/ana/Documentos/SI/p1.1/P1-base/sql/create.sql
[sql] Executing resource: /home/ana/Documentos/SI/p1.1/P1-base/sql/insert.sql
[sql] 1003 of 1003 SQL statements executed successfully

setup-db:

montar-jerarquia:
[mkdir] Created dir: /home/ana/Documentos/SI/p1.1/P1-base/build
[mkdir] Created dir: /home/ana/Documentos/SI/p1.1/P1-base/dist
[mkdir] Created dir: /home/ana/Documentos/SI/p1.1/P1-base/build/WEB-INF/classes
[mkdir] Created dir: /home/ana/Documentos/SI/p1.1/P1-base/build/WEB-INF/lib

compilar:
[javac] Compiling 6 source files to /home/ana/Documentos/SI/p1.1/P1-base/build/WEB-INF/classes

preparar-web-inf:
[copy] Copying 8 files to /home/ana/Documentos/SI/p1.1/P1-base/build

empaquetar:
[jar] Building jar: /home/ana/Documentos/SI/p1.1/P1-base/dist/P1.war

desplegar:
[exec] Application deployed with name P1.
[exec] Command deploy executed successfully.

BUILD SUCCESSFUL
Total time: 9 seconds
ana@ana:~/Documentos/SI/p1.1/P1-base$

```

Posteriormente, desplegamos la aplicación web y la base de datos con el comando

\$> ant desplegar

```

ana@ana:~/Documentos/SI/p1.1/P1-base$ ant desplegar
Buildfile: /home/ana/Documentos/SI/p1.1/P1-base/build.xml

desplegar:
[exec] Command deploy failed.
[exec] remote failure: Error occurred during deployment: Application with name P1 is already registered. Either specify that redeployment must be forced, or redeploy the application. Or if this is a new deployment, pick a different name. Please see server.log for more details.
[exec] Result: 1

BUILD SUCCESSFUL
Total time: 1 second
ana@ana:~/Documentos/SI/p1.1/P1-base$

```

- Registre un voto contra la aplicación web empleando el navegador en la ruta <http://10.X.Y.Z:8080/P1>. Conéctese a la base de datos (usando el cliente DBeaver por ejemplo) y obtenga evidencias de que el voto se ha registrado.

Accedemos al enlace <http://10.2.8.2:8080/P1> y registramos un voto:

Complete la información sobre el voto:

Id Mesa electoral:	<input type="text"/>
Id Circunscripción:	<input type="text"/>
Id Proceso electoral:	<input type="text"/>
Nombre candidato/a votado/a:	<input type="text"/>
<input type="button" value="Enviar"/>	

Complete la información sobre el voto:

Id Mesa electoral:	<input type="text" value="1212"/>
Id Circunscripción:	<input type="text" value="2323"/>
Id Proceso electoral:	<input type="text" value="3434"/>
Nombre candidato/a votado/a:	<input type="text" value="Fernando García"/>
<input type="button" value="Enviar"/>	

Y posteriormente completamos la información del censo.

Complete la información sobre el censo

Número de DNI:	<input type="text" value="39739740E"/>
Nombre y Apellidos:	<input type="text" value="Jose Moreno Locke"/>
Fecha de Nacimiento:	<input type="text" value="09/04/66"/>
Código Autorización:	<input type="text" value="729"/>
<input type="button" value="Volver"/>	<input type="button" value="Registrar Voto"/>

Si el voto se ha ejecutado correctamente, podemos consultar el voto y vemos la información sobre el registro del voto correspondiente.

Información Sobre el Registro del Voto:

Código Respuesta: 000
Id Voto: 1
Id Proceso electoral: 3434
Marca Tiempo Voto: 2024-02-14 19:40:39.095015
Número DNI: 39739740E

Accedemos a DBeaver y observamos que se registra el voto correctamente en la base de datos.

4.488	alumnodb	voto	idle	2024-02-14 19:40:39.09573	select idVoto, codRespuesta, marcaTiempo from voto ...
4.544	alumnodb	postgres	DBeaver 23.3.4 - View sessions	active	2024-02-14 19:41:57.261294 SELECT sa.* FROM pg_catalog.pg_stat_activity sa
833					
832					

- Acceda a la página de pruebas extendida, <http://10.X.Y.Z:8080/P1/testbd.shtml>. Compruebe que la funcionalidad de listado de y borrado de votos funciona correctamente. Elimine el voto anterior.

Accedemos a <http://10.2.8.2/P1/testbd.shtml> y en primer lugar comprobamos que existe un voto con el id del proceso electoral que hemos creado anteriormente, para borrarlo ahora.

Lista de votos del proceso electoral 3434

idVoto	idCircunscripcion	idMesaElectoral	idProcesoElectoral	Candidato/a Votado/a	Marca de Tiempo	codRespuesta
1	2323	1212	3434	Fernando García	2024-02-14 19:40:39.095015	000

[Volver](#)

Una vez que nos hemos asegurado que dicho voto existe, rellenamos el campo de “Id Proceso Electoral” en “Borrado de votos”.

Complete la información sobre el voto:

Voto:
Id Mesa electoral:
Id Circunscripción:
Id Proceso electoral:
Nombre candidato/a votado/a:

Censo:
Número de DNI:
Nombre y Apellidos:
Fecha de Nacimiento:
Código Autorización:
Debug: ☒ Si ☐ No
Prepared Statements: ☒ Si ☐ No
Conexión directa al a BD: ☐ Si ☒ No

[Registrar Voto](#)

Consulta de Votos:

Id Proceso electoral:
[Consultar Votos](#)

Borrado de Votos:

Id Proceso electoral:
[Borrar Votos](#)

Posteriormente cuando pulsemos el botón “Borrar Votos” se muestra la siguiente página en la que confirma que el voto ha sido eliminado correctamente.

Se han borrado correctamente 1 votos del proceso electoral con identificador 3434

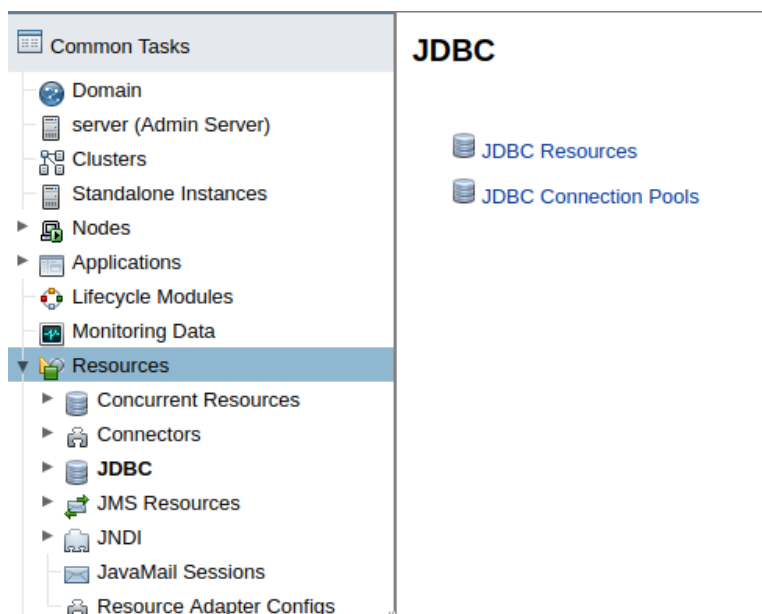
[Volver](#)

Y para comprobarlo, accedemos con Dbeaver a la base de datos y vemos que se ha eliminado correctamente con el mensaje que aparece en la parte superior “delete ...”.

4.550	alumnodb	voto	idle	2024-02-14 19:49:47.820785	delete from voto where idProcesoElectoral=\$1
4.573	alumnodb	voto	idle		
4.598	alumnodb	postgres	DBeaver 23.3.4 - View sessions	active	2024-02-14 19:50:32.59113
833					SELECT sa.* FROM pg_catalog.pg_stat_activity sa
832					
835					

Ejercicio número 2:

La clase VotoDAO, especificada en VotoDAO.java, implementa los dos tipos de conexión descritos anteriormente, los cuales son heredados de la clase DBTester. Sin embargo, la configuración de la conexión utilizando la conexión directa es incorrecta. Se pide completar la información necesaria para llevar a cabo la conexión directa de forma correcta. Para ello habrá que fijar los atributos a los valores correctos. En particular, el nombre del driver JDBC a utilizar, el JDBC connection string que se debe corresponder con el servidor postgresql, y el nombre de usuario y la contraseña de la base de datos. Es necesario consultar el apéndice 10 para ver los detalles de cómo se obtiene una conexión de forma correcta. Una vez completada la información, acceda a la página de pruebas extendida, <http://10.2.8.2:8080/P1/testbd.xhtml> y pruebe a registrar un voto utilizando la conexión directa, y pruebe a listarlo y eliminarlo. Adjunte en la memoria evidencias de este proceso, incluyendo capturas de pantalla.



Para configurar la conexión directa de forma correcta, cambiamos el valor de las variables JDBC_DRIVER, JDBC_CONNSTRING, JDBC_USER y JDBC_PASSWORD con los valores que se indican en las imágenes siguientes.

```
// Información de conexión
// Para conexiones directas, requerimos: driver, cadena de conexión,
// usuario y clave
private static final String JDBC_DRIVER =
    "org.postgresql.Driver";
```

```
// TODO: Definir la cadena de conexion a la base de datos
/*****/
private static final String JDBC_CONNSTRING =
    "jdbc:postgresql://10.2.8.2:5432/voto";
/*****/

/*****/
private static final String JDBC_USER = "alumnodb";
private static final String JDBC_PASSWORD = "alumnodb";
/*****/
```

Después de realizar la modificación en DBTester, accedemos a <http://10.2.8.2:8080/P1/testbd.shtml> registramos un voto con la opción de “Conexión directa a la BD” y pulsamos el botón de “Registrar voto”.

Complete la información sobre el voto:

Voto:

Id Mesa electoral:
 Id Circunscripción:
 Id Proceso electoral:
 Nombre candidato/a votado/a:

Censo:

Número de DNI:
 Nombre y Apellidos:
 Fecha de Nacimiento:
 Código Autorización:
 Debug: ☒ Si ☐ No
 Prepared Statements: ☒ Si ☐ No
 Conexión directa al a BD: ☒ Si ☐ No

Comprobamos que el voto se ha registrado correctamente, haciendo una consulta.

Información Sobre el Registro del Voto:

Código Respuesta: 000
 Id Voto: 2
 Id Proceso electoral: 3434
 Marca Tiempo Voto: 2024-02-16 09:40:19.327287
 Número DNI: 39739740E

Lista de votos del proceso electoral 3434

idVoto	idCircunscripcion	idMesaElectoral	idProcesoElectoral	Candidato/a Votado/a	Marca de Tiempo	codRespuesta
1	2323	1212	3434	Fernando Garcia	2024-02-16 11:48:21.291915	000

Y posteriormente lo borramos.

Se han borrado correctamente 1 votos del proceso electoral con identificador 3434

Ejercicio número 3:

Examinar el archivo `postgresql.properties` para determinar el nombre del recurso JDBC correspondiente al `DataSource` y el nombre del pool. Acceda a la Consola de Administración. Compruebe que los recursos JDBC y pool de conexiones han sido correctamente creados. Realice un Ping JDBC a la base de datos. Anote en la memoria de la práctica los valores para los parámetros Initial and Minimum Pool Size, Maximum Pool Size, Pool Resize Quantity, Idle Timeout, Max Wait Time. Comente razonadamente qué impacto considera que pueden tener estos parámetros en el rendimiento de la aplicación. Consulte la documentación sobre Glassfish (<https://glassfish.org/docs/latest/performance-tuning-guide.html#jdbc-connection-pool-settings>).

Examinamos el archivo `postgresql.properties` y vemos que el nombre del recurso JDBC es “jdbc/VotoDB” y el nombre del pool es “VotoPool”, correspondientes a la primera y segunda imagen respectivamente.

```
db.jdbc.resource.name=jdbc/VotoDB
```

```
db.pool.name=VotoPool
```

Accedemos a la página <http://10.2.8.2:4848>, posteriormente a `Resources>JDBC>VotoPool` y vemos los valores de los parámetros siguientes:

Initial and Minimum Pool Size = 8

Maximum Pool Size = 32

Pool Resize Quantity = 2

Idle Timeout = 300

Max Wait Time = 60000

The screenshot shows the 'Edit JDBC Connection Pool' page for 'VotoPool' in the Glassfish Administration Console. The 'General' tab is selected, and a 'Ping Succeeded' message is displayed at the top right. The page includes a sidebar with a tree view of the system's resources, and a main content area with various configuration fields. The 'General Settings' section is expanded, showing fields for Pool Name, Resource Type, Datasource Classname, Driver Classname, Ping, Deployment Order, and Description. The 'Pool Name' is 'VotoPool', 'Resource Type' is 'javax.sql.ConnectionPoolDataSource', 'Datasource Classname' is 'org.postgresql.ds.PGConnectionPoolDataSource', 'Driver Classname' is 'org.postgresql.Driver', 'Ping' is checked, 'Deployment Order' is '100', and 'Description' is empty. The 'Advanced' and 'Additional Properties' tabs are also visible at the top.

Pool Settings

Initial and Minimum Pool Size:	<input type="text" value="8"/>	Connections	Minimum and initial number of connections maintained in the pool
Maximum Pool Size:	<input type="text" value="32"/>	Connections	Maximum number of connections that can be created to satisfy client requests
Pool Resize Quantity:	<input type="text" value="2"/>	Connections	Number of connections to be removed when pool idle timeout expires
Idle Timeout:	<input type="text" value="300"/>	Seconds	Maximum time that connection can remain idle in the pool
Max Wait Time:	<input type="text" value="60000"/>	Milliseconds	Amount of time caller waits before connection timeout is sent

El impacto que podrían tener estos parámetros es debido al coste elevado de la creación de la base de datos. Esto conlleva a que el parámetro “Initial and Minimum Pool Size” no tiene que ser ni muy alto ni muy elevado, ya que no sería eficiente. Con “Maximum Pool Size” tampoco debería tener un valor muy alto ya que el sistema se colapsaría al tener demasiadas conexiones a la vez en el servidor.

Ejercicio número 4:

Localice los siguientes fragmentos de código SQL dentro del proyecto proporcionado (P1-base) correspondientes a los siguientes procedimientos:

- Consulta de si la información del censo es correcta.

La función que consulta si la información de censo es getQryCompruebaCenso(Censo censo), ya que recibe un objeto de tipo Censo por argumento y comprueba que sus campos (numeroDNI, nombre, fechaNacimiento y codigoAutorizacion) sean los correctos.

```
/**
 * getQryCompruebaCenso
 */
private String getQryCompruebaCenso(CensoBean censo) {
    String qry = "select * from censo "
        + "where numeroDNI='" + censo.getNumeroDNI()
        + "' and nombre='" + censo.getNombre()
        + "' and fechaNacimiento='" + censo.getFechaNacimiento()
        + "' and codigoAutorizacion='" + censo.getCodigoAutorizacion() + "'";
    return qry;
}
```

```
private static final String SELECT_CENSO_QRY =
    "select * from censo " +
    "where numeroDNI=? " +
    " and nombre=? " +
    " and fechaNacimiento=? " +
    " and codigoAutorizacion=? ";
```

- Registro del voto.

La función que hace el registro del voto es getQryInsertVoto(VotoBean voto) ya que hace una consulta en sql con un INSERT en el que introduce el voto con los parámetros que tiene el voto que son: idCircunscripcion, idMesaElectoral, idProcesoElectoral, nombreCandidatoVotado y numeroDNI.

```
/**
 * getQryInsertVoto
 */
private String getQryInsertVoto(VotoBean voto) {
    String qry = "insert into voto("
        + "idCircunscripcion,"
        + "idMesaElectoral,idProcesoElectoral,"
        + "nombreCandidatoVotado, numeroDNI)"
        + " values ("
        + "'" + voto.getIdCircunscripcion() + "',"
        + "'" + voto.getIdMesaElectoral() + "',"
        + "'" + voto.getIdProcesoElectoral() + "',"
        + "'" + voto.getNombreCandidatoVotado() + "',"
        + "'" + voto.getCenso().getNumeroDNI() + "'"
        + ")";
    return qry;
}
```

```
private static final String INSERT_VOTO_QRY =
    "insert into voto(" +
    "idCircunscripcion,idMesaElectoral,idProcesoElectoral,nombreCandidatoVotado,numeroDNI)" +
    " values (?, ?, ?, ?, ?)";
```

Ejercicio número 5:

Edite el fichero VotoDAO.java y localice el método errorLog. Compruebe en qué partes del código se escribe en log utilizando dicho método. Realice el registro de un voto utilizando la página testbd.xhtml con la opción de debug activada. Visualice el log del servidor de aplicaciones y compruebe que dicho log contiene información adicional sobre las acciones llevadas a cabo en VotoDAO.java. Incluya en la memoria capturas de pantalla del log del servidor, accediendo a él tanto desde el terminal como del portal web.

```
/**
 * Imprime traza de depuracion
 */
private void errorLog(String error) {
    if (isDebug())
        System.err.println("[directConnection=" + this.isDirectConnection() + "] " +
                             error);
}
```

El método errorLog se utiliza en los métodos compruebaCenso, registraVoto, getVotos y delVotos. Y también se ejecuta errorLog fuera de métodos, como antes de ejecutar la consulta SQL para borrar los votos asociados a un proceso electoral.

Accediendo por el servidor web para ver el log del servidor de aplicaciones, accedemos a la url <http://10.2.8.2:4848>.

Log Viewer
View, search, and filter a server log file using basic and advanced options. Refer to the Log Levels page for information about log levels you can filter here.

[Advanced Search](#)

Search Criteria

Text search:

Only log entries containing the specified text will be displayed. Search is case sensitive.

Timestamp: ☒ Most Recent ☐ Specific Range:

Log Level: ☐ Do not include more severe messages

Log entries are limited to those stored in the log file. Set appropriate log level in the Log Level page to ensure data is logged.

[Modify Search](#) [Search](#) [Close](#)

Instance: Log File:

Log Viewer Results (40)

Records before 1064 Log File Record Numbers 1064 through 1103 Records after 1103

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
1103	INFO	[LOG INFO] Voto registrado correctamente.(details)	jakarta.enterprise.logging.stdout	16 Feb 2024 18:42:13.150	()
1102	SEVERE	[directConnection=false] select idVoto, codRespuesta, marcaTiempo from voto where idProcesoElectoral... (details)	jakarta.enterprise.logging.stderr	16 Feb 2024 18:42:13.149	()
1101	SEVERE	[directConnection=false] insert into voto(idCircunscripcion,idMesaElectoral,idProcesoElectoral,nombr... (details)	jakarta.enterprise.logging.stderr	16 Feb 2024 18:42:13.145	()
1100	SEVERE	[directConnection=false] select * from censo where numeroDNI=? and nombre=? and fechaNacimiento=? ... (details)	jakarta.enterprise.logging.stderr	16 Feb 2024 18:42:13.140	()
1099	INFO	[LOG INFO] Solicitado el registro del voto.(details)	jakarta.enterprise.logging.stdout	16 Feb 2024 18:42:13.138	()

Ejercicio número 6:

Realícense las modificaciones necesarias en VotoDAOWS.java para que implemente de manera correcta un servicio web. Los siguientes métodos y todos sus parámetros deberán ser publicados como métodos del servicio.

- compruebaCenso()
- registraVoto()
- isDebug() / setDebug()
- isPrepared() / setPrepared()

Deberemos publicar así mismo:

- isDirectConnection() / setDirectConnection()

que son métodos heredados de la clase DBTester

Para ello, implemente estos métodos también en la clase hija. Es decir, haga un override de Java, implementando estos métodos en VotoDAOWS mediante invocaciones a la clase padre (super). En ningún caso se debe añadir ni modificar nada de la clase DBTester.

Modifique así mismo el método registraVoto() para que éste devuelva el voto modificado tras el correcto o incorrecto registro del voto:

- Con identificador de voto y código de respuesta correcto en caso de haberse realizado.
- Con null en caso de no haberse podido realizar.

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones requeridas. Por último, conteste a la siguiente pregunta: ¿Por qué se ha de alterar el parámetro de retorno del método registraVoto() para que devuelva el voto modificado el lugar de un boolean?

Hacemos @Override de los métodos isDirectConnection() y setDirectConnection() como se nos pide en el enunciado, estas funciones las sacamos de la clase DBTester.

```
/**
 * @return the pooled
 */
@Override
@WebMethod()
public boolean isDirectConnection()
{
    return super.isDirectConnection();
}

/**
 * @param directConnection valor de conexión directa o indirecta
 */
@Override
@WebMethod()
public void setDirectConnection(boolean directConnection)
{
    super.setDirectConnection(directConnection);
}
```

En primer lugar hemos cambiado el nombre de la clase y de su constructor por VotoDAOWS, y seguidamente hemos añadido las anotaciones @WebService() en la clase y @WebMethod() en los métodos.

```

30 @WebService()
31 @WebMethod()
32 public class VotoDAOWS extends DBTester {
33
34     private boolean debug = false;
35
36     /**
37      * TODO: Declarar un atributo booleano "prepared"
38      * que indique si hay que usar prepared statements o no
39      * Utilizar statements preparados o no
40      */
41     private boolean prepared = true;
42     //*****
43
44     /** Para prepared statements, usamos cadenas constantes
45      * Esto agiliza el trabajo del pool de statements */
46     private static final String DELETE_VOTOS_QRY =
47         "delete from voto " +
48         "where idProcesoElectoral=?";
49     private static final String SELECT_VOTOS_QRY =
50         "select * from voto " +
51         "where idProcesoElectoral=?";
52
53     /**
54      * TODO: Declarar consultas SQL estaticas
55      * para uso con prepared statement
56      * Hay que convertir las consultas de get0ryXXX()
57      */
58     //private static final String ...//
59     //*****
60     private static final String SELECT_CENSO_QRY =
61         "select * from censo " +
62         "where numeroDNI=? " +
63         "and nombre=? " +
64         "and fechaNacimiento=? " +
65         "and codigoAutorizacion=? ";
66
67     private static final String INSERT_VOTO_QRY =
68         "insert into voto(" +
69         "idCircunscripcion,idMesaElectoral,idProcesoElectoral,nombreCandidatoVotado,numeroDNI) " +
70         "values (?,?,?,?,?)";
71
72     private static final String SELECT_VOTO_TRANSACCION_QRY =
73         "select idVoto, codRespuesta, marcaTiempo " +
74         "from voto " +
75         "where idProcesoElectoral = ? " +
76         "and numeroDNI = ?";
77     //*****
78
79     /**
80      * Constructor de la clase
81      */
82     public VotoDAOWS() {
83         return;
84     }
85
86
87

```

Para la modificación de registraVoto() hemos cambiado el valor de retorno de boolean por VotoBean ya que tenemos que devolver el voto modificado. En consecuencia el valor de ret pasa de ser un boolean a ser un VotoBean, en las sentencias que antes eran ret = false pasan a ser ret = null y las sentencias que antes eran ret = true pasan a ser ret = voto.

```

/**
@WebMethod()
public synchronized VotoBean registraVoto(VotoBean voto) {
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    String codRespuesta = "999"; // En principio, voto invalido
    VotoBean ret = null;

    PreparedStatement pstmt = null;

    // Comprobamos campos voto

    if (voto.getIdCircunscripcion() == null || voto.getIdMesaElectoral() == null ||
        voto.getIdProcesoElectoral() == null || voto.getNombreCandidatoVotado() == null ||
        voto.getCenso() == null || voto.getCenso().getNumeroDNI() == null) {
        errorLog(";El voto tiene campos vacíos!");
        return null;
    }

    // Registrar el voto en la base de datos

    try {
        // Obtener conexion
        con = getConnection();

        // Insertar en la base de datos el voto

        if (isPrepared() == true) {
            String insert = INSERT_VOTO_QRY;
            errorLog(insert);
            pstmt = con.prepareStatement(insert);
            pstmt.setString(1, voto.getIdCircunscripcion());
            pstmt.setString(2, voto.getIdMesaElectoral());
            pstmt.setString(3, voto.getIdProcesoElectoral());
            pstmt.setString(4, voto.getNombreCandidatoVotado());
            pstmt.setString(5, voto.getCenso().getNumeroDNI());
            ret = null;

            if (!pstmt.execute() && pstmt.getUpdateCount() == 1) {
                ret = voto;
            }
        } else {
            stmt = con.createStatement();
            String insert = getQryInsertVoto(voto);
            errorLog(insert);

            if (!stmt.execute(insert) && stmt.getUpdateCount() == 1) {
                errorLog("Voto insertado correctamente 2");
                ret = voto;
            }
        }
    }
}

```

```

public synchronized VotoBean registraVoto(VotoBean voto) {
    try {
        if (ret != null) {

            if (isPrepared() == true) {
                String select = SELECT_VOTO_TRANSACCION_QRY;
                errorLog(select);
                pstmt = con.prepareStatement(select);
                pstmt.setString(1, voto.getIdProcesoElectoral());
                pstmt.setString(2, voto.getCenso().getNumeroDNI());
                rs = pstmt.executeQuery();
            } else {
                String select = getQryBuscaVotoTransaccion(voto);
                errorLog(select);
                rs = stmt.executeQuery(select);
            }

            if (rs.next()) {
                // Completamos la información que se genera al insertar el
                // voto en la base de datos
                voto.setIdVoto(String.valueOf(rs.getInt("idVoto")));
                voto.setCodigoRespuesta(rs.getString("codRespuesta"));
                voto.setMarcaTiempo(rs.getString("marcaTiempo"));
                ret = voto;
            } else {
                errorLog("No se encontró el voto insertado en la base de datos.");
                return null;
            }

        }

    } catch (Exception e) {
        errorLog(e.toString());
        ret = null;
    } finally {
        try {
            if (rs != null) {
                rs.close(); rs = null;
            }
            if (stmt != null) {
                stmt.close(); stmt = null;
            }
            if (pstmt != null) {
                pstmt.close(); pstmt = null;
            }
            if (con != null) {
                closeConnection(con); con = null;
            }
        } catch (SQLException e) {
        }
    }

    return ret;
}

```

Ejercicio número 7:

Despliegue el servicio con la regla correspondiente en el build.xml. Acceda al WSDL remotamente con el navegador e inclúyalo en la memoria de la práctica (habrá que asegurarse que la URL contiene la dirección IP de la máquina virtual donde se encuentra el servidor de aplicaciones y no su nombre de host, que solo es conocido por la propia VM, pero no por el ordenador host). Comente en la memoria aspectos relevantes del código XML del fichero WSDL y su relación con los métodos Java del objeto del servicio, argumentos recibidos y objetos devueltos 2 .

Conteste a las siguientes preguntas:

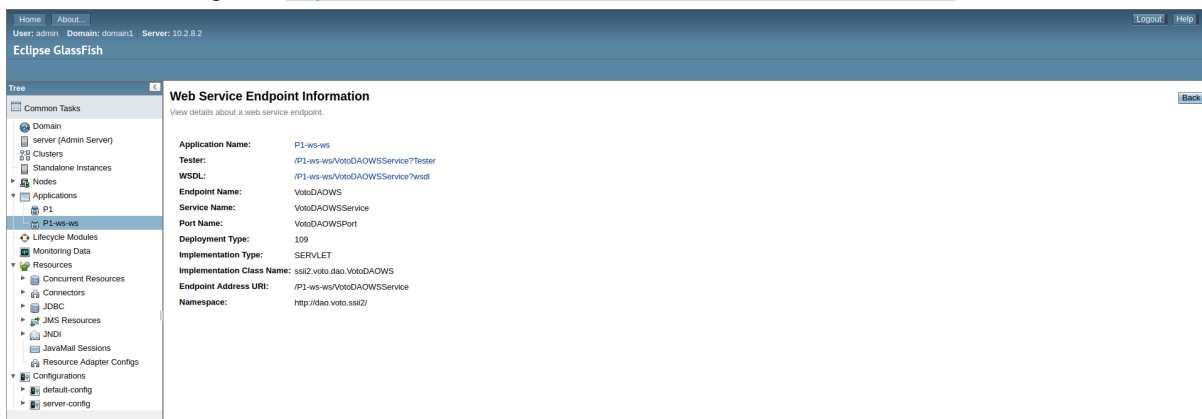
Desplegamos el servicio con los comandos:

ant compilar-servicio

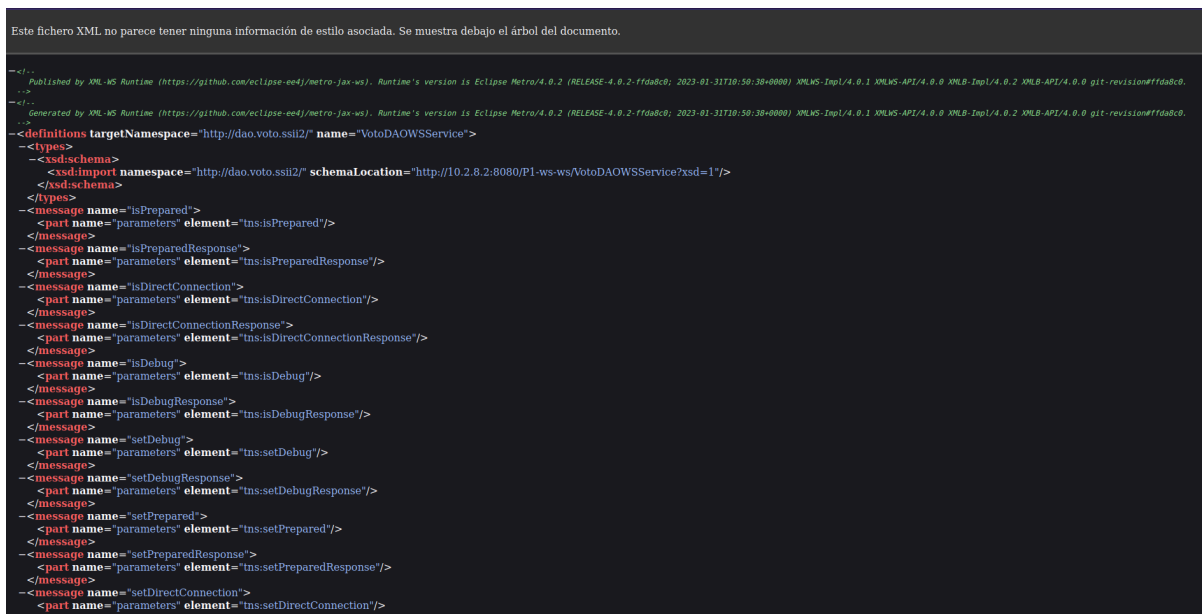
ant empaquetar-servicio

ant desplegar-servicio

Y accedemos a la url <http://10.2.8.2:4848>, nos dirigimos a Applications>P1-ws-ws y copiamos el enlace de WSDL en el navegador <http://10.2.8.2:8080/P1-ws-ws/VotoDAOWSService?wsdl>.



Cuando accedemos al enlace <http://10.2.8.2:8080/P1-ws-ws/VotoDAOWSService?wsdl>, vemos la siguiente imagen.



- ¿En qué fichero están definidos los tipos de datos intercambiados con el webservice?

Los tipos de datos están definidos en el fichero wsdl que indica el enlace de la foto inferior:

```
schemaLocation="http://10.2.8.2:8080/P1-ws-ws/VotoDAOWSService?xsd=1"/>
```

Si nos dirigimos al enlace anterior encontramos el siguiente contenido:

```

<!-- Published by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is Eclipse Metro/4.0.2 (RELEASE-4.0.2-fd0a8c9; 2023-01-31T10:50:39-0000) XMLWS-Impl/4.0.1 XMLWS-API/4.0.0 XMLB-Impl/4.0.2 XMLB-API/4.0.0 git-revision#fd0a8c9.
-->
<xs:schema version="1.0" targetNamespace="http://dao.voto.ssii2/">
  <xs:element name="compruebaCenso" type="tns:compruebaCenso"/>
  <xs:element name="compruebaCensoResponse" type="tns:compruebaCensoResponse"/>
  <xs:element name="delVotos" type="tns:delVotos"/>
  <xs:element name="delVotosResponse" type="tns:delVotosResponse"/>
  <xs:element name="getVotos" type="tns:getVotos"/>
  <xs:element name="getVotosResponse" type="tns:getVotosResponse"/>
  <xs:element name="isDebug" type="tns:isDebug"/>
  <xs:element name="isDebugResponse" type="tns:isDebugResponse"/>
  <xs:element name="isDirectConnection" type="tns:isDirectConnection"/>
  <xs:element name="isDirectConnectionResponse" type="tns:isDirectConnectionResponse"/>
  <xs:element name="isPrepared" type="tns:isPrepared"/>
  <xs:element name="isPreparedResponse" type="tns:isPreparedResponse"/>
  <xs:element name="registraVoto" type="tns:registraVoto"/>
  <xs:element name="registraVotoResponse" type="tns:registraVotoResponse"/>
  <xs:element name="setDebug" type="tns:setDebug"/>
  <xs:element name="setDebugResponse" type="tns:setDebugResponse"/>
  <xs:element name="setDirectConnection" type="tns:setDirectConnection"/>
  <xs:element name="setDirectConnectionResponse" type="tns:setDirectConnectionResponse"/>
  <xs:element name="setPrepared" type="tns:setPrepared"/>
  <xs:element name="setPreparedResponse" type="tns:setPreparedResponse"/>
  <xs:complexType name="getVotos">
    <xs:sequence>
      <xs:element name="arg0" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getVotosResponse">
    <xs:sequence>
      <xs:element name="return" type="tns:votoBean" nillable="true" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="votoBean">
    <xs:sequence>
      <xs:element name="censo" type="tns:censoBean" form="qualified" minOccurs="0"/>
      <xs:element name="codigoRespuesta" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="idCircunscripcion" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="idMesaElectoral" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="idProcesoElectoral" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="idVoto" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="marcaTiempo" type="xs:string" form="qualified" minOccurs="0"/>
      <xs:element name="nombreCandidatoVotado" type="xs:string" form="qualified" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

- ¿Qué tipos de datos predefinidos se usan?

Los datos predefinidos que encontramos son todos aquellos que comienzan por xs, como por ejemplo xs:string, xs:boolean, xs: int.

```

-<xs:complexType name="votoBean">
  <xs:sequence>
    <xs:element name="censo" type="tns:censoBean" form="qualified" minOccurs="0"/>
    <xs:element name="codigoRespuesta" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="idCircunscripcion" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="idMesaElectoral" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="idProcesoElectoral" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="idVoto" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="marcaTiempo" type="xs:string" form="qualified" minOccurs="0"/>
    <xs:element name="nombreCandidatoVotado" type="xs:string" form="qualified" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
-<xs:complexType name="censoBean">
  <xs:sequence>

```

- ¿Cuáles son los tipos de datos nuevos que se definen?

Los nuevos tipos de datos que se definen son todos aquellos que comienzan por tns, como por ejemplo tns:compruebaCenso, tns:compruebaCensoResponse, tns:delVotos...

```

<!-- Published by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is Eclipse Metro/4.0.2 (RELEASE-4.0.2-fd0a8c9; 2023-01-31T10:50:39-0000) XMLWS-Impl/4.0.1 XMLWS-API/4.0.0 XMLB-Impl/4.0.2 XMLB-API/4.0.0 git-revision#fd0a8c9.
-->
<xs:schema version="1.0" targetNamespace="http://dao.voto.ssii2/">
  <xs:element name="compruebaCenso" type="tns:compruebaCenso"/>
  <xs:element name="compruebaCensoResponse" type="tns:compruebaCensoResponse"/>
  <xs:element name="delVotos" type="tns:delVotos"/>
  <xs:element name="delVotosResponse" type="tns:delVotosResponse"/>
  <xs:element name="getVotos" type="tns:getVotos"/>
  <xs:element name="getVotosResponse" type="tns:getVotosResponse"/>
  <xs:element name="isDebug" type="tns:isDebug"/>
  <xs:element name="isDebugResponse" type="tns:isDebugResponse"/>
  <xs:element name="isDirectConnection" type="tns:isDirectConnection"/>
  <xs:element name="isDirectConnectionResponse" type="tns:isDirectConnectionResponse"/>
  <xs:element name="isPrepared" type="tns:isPrepared"/>
  <xs:element name="isPreparedResponse" type="tns:isPreparedResponse"/>
  <xs:element name="registraVoto" type="tns:registraVoto"/>
  <xs:element name="registraVotoResponse" type="tns:registraVotoResponse"/>
  <xs:element name="setDebug" type="tns:setDebug"/>
  <xs:element name="setDebugResponse" type="tns:setDebugResponse"/>
  <xs:element name="setDirectConnection" type="tns:setDirectConnection"/>
  <xs:element name="setDirectConnectionResponse" type="tns:setDirectConnectionResponse"/>
  <xs:element name="setPrepared" type="tns:setPrepared"/>
  <xs:element name="setPreparedResponse" type="tns:setPreparedResponse"/>
  <xs:complexType name="getVotos">

```

- **¿Qué etiqueta está asociada a los métodos invocados en el webservice?**
La etiqueta asociada a los métodos invocados es <message>.
- **¿Qué etiqueta describe los mensajes intercambiados en la invocación de los métodos del webservice?**
La etiqueta asociada a los mensajes intercambiados es <operation>.
- **¿En qué etiqueta se especifica el protocolo de comunicación con el webservice?**
La etiqueta que especifica el protocolo de comunicación es <binding>.
- **¿En qué etiqueta se especifica la URL a la que deberá acceder un cliente para acceder al webservice?**
La etiqueta en la que se especifica la URL en la que el cliente accede al webservice es <service>.

Ejercicio número 8:

Realícese las modificaciones necesarias en ControladorBean.java para que implemente de manera correcta la llamada al servicio web mediante stubs estáticos, usando el objeto VotoDAOWS dao.

Téngase en cuenta que:

- El nuevo método registraVoto() ahora no devuelve un boolean, sino el propio objeto VotoBean modificado.
- Las llamadas remotas pueden generar nuevas excepciones que deberán ser tratadas en el código cliente.
- Antes de llamar a registraVoto() y compruebaCenso(), se ha de traducir la información del voto contenida en VotoBean a ssii2.servicio.VotoBean, usando el método traducirVotoParaServicio descrito más arriba. Será necesario actualizar el identificador del voto, la marca de tiempo, y el código de respuesta del voto, que se obtienen al llamar a registraVoto.

Incluye en la memoria una captura con dichas modificaciones.

En ControladorBean.java sustituimos todas las sentencias anteriores de "VotoDAO dao = new VotoDAO();" por las sentencias de la imagen siguiente:

```
VotoDAOWSService service = new VotoDAOWSService();
VotoDAOWS dao = service.getVotoDAOWSPort();
```

Cuando llamamos a registraVoto() cambiamos la comprobación "false" por "null" ya que hemos cambiado el retorno de la función.

```

    if (dao.registraVoto(traducirVotoParaServicio(this.voto)) == null) {
        String error_msg = "¡No se ha podido registrar el voto!";
        if (this.interaccion.getDebug() == true) {
            this.escribirLog(error_msg);
        }
        this.setMensajeError(error_msg);
        return "error";
    }
}

```

Cuando llamamos a los métodos registraVoto() y compruebaCenso() metemos por argumento la llamada a la función traducirVotoParaServicio(this.voto).

Ejercicio número 9:

Modifique la llamada al servicio para que la ruta (URL) al servicio remoto se obtenga del fichero de configuración web.xml. Para saber cómo hacerlo consulte el apéndice 14.1 para más información y edite el fichero web.xml y analice los comentarios que allí se incluyen.

En ControladorBean.java añadimos las tres sentencias de después de la creación de dao, para que la llamada al servicio remoto se obtenga de web.xml.

```

VotoDAOWebService service = new VotoDAOWebService();
VotoDAOWS dao = service.getVotoDAOWSPort();

String url = FacesContext.getCurrentInstance().getExternalContext().getInitParameter("url");
BindingProvider bp = (BindingProvider) dao;
bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, url);

```

En el fichero web.xml añadimos un nuevo parámetro con la url del servidor.

```

<context-param>
    <param-name>url</param-name>
    <param-value>http://10.2.8.2:8080/P1-ws-ws/VotoDAOWebService</param-value>
</context-param>

```


Ejercicio número 10:

Siguiendo el patrón de los cambios anteriores, adaptar los siguientes métodos en ControladorBean.java para que toda la funcionalidad de la página de pruebas testbd.xhtml se realice a través del servicio web. Esto afecta al menos a los siguientes métodos de ControladorBean.java:

- borrarVotos(): La operación dao.delVotos() debe implementarse en el servicio web.
- consultarVotos(): La operación dao.getVotos() debe implementarse en el servicio web.

Tenga en cuenta que no todos los tipos de datos son compatibles con JAXB (especifica como codificar clases java como documentos XML), por lo que es posible que tenga que modificar el valor de retorno de alguno de estos métodos. Más específicamente, se tiene que modificar la declaración actual del método getVotos(), que devuelve un VotoBean[], por:

public ArrayList getVotos(String idProcesoElectoral)

Hay que tener en cuenta que la página listavotos.xhtml espera recibir un array del tipo VotoBean[]. Por ello, es conveniente, una vez obtenida la respuesta, convertir el ArrayList a un array de tipo VotoBean[] utilizando el método toArray() de la clase ArrayList. No es necesario traducir los objetos ssii2.servicio.VotoBean devueltos, y se pueden almacenar directamente en la sesión para su lectura por listavotos.xhtml. Incluye en la memoria una captura con las adaptaciones realizadas.

En la clase VotoDAOWS.java cambiamos el retorno del método getVotos(), también cambiamos las variables de ret y de votos por ArrayList<VotoBean>.

```
public ArrayList<VotoBean> getVotos(String idProcesoElectoral) {  
    PreparedStatement pstmt = null;  
    Connection pcon = null;  
    ResultSet rs = null;  
    ArrayList<VotoBean> ret = null;  
    ArrayList<VotoBean> votos = null;  
    String qry = null;  
  
    try {  
        // Crear una conexion u obtenerla del pool  
        pcon = getConnection();  
        qry = SELECT_VOTOS_QRY;  
        errorLog(qry + "[idProcesoElectoral=" + idProcesoElectoral + "]");  
  
        // La preparacion del statement  
        // es automaticamente tomada de un pool en caso  
        // de que ya haya sido preparada con anterioridad  
  
        pstmt = pcon.prepareStatement(qry);  
        pstmt.setString(1, idProcesoElectoral);  
        rs = pstmt.executeQuery();  
  
        votos = new ArrayList<VotoBean>();  
  
        while (rs.next()) {  
            CensoBean c = new CensoBean();  
            VotoBean v = new VotoBean();  
            v.setIdVoto(rs.getString("idVoto"));  
            v.setIdCircunscripcion(rs.getString("idCircunscripcion"));  
            v.setIdMesaElectoral(rs.getString("idMesaElectoral"));  
            v.setIdProcesoElectoral(rs.getString("idProcesoElectoral"));  
            v.setNombreCandidatoVotado(rs.getString("nombreCandidatoVotado"));  
            v.setMarcaTiempo(rs.getString("marcaTiempo"));  
            v.setCodigoRespuesta(rs.getString("codRespuesta"));  
  
            votos.add(v);  
        }  
  
        ret = votos;  
  
        // Cerramos / devolvemos la conexion al pool  
  
        pcon.close();  
    }  
}
```

Y en ControladorBean.java también cambiamos el tipo de la variable en el que recogemos el resultado de la llamada al método dao.getVotos().

```
public String consultarVotos() {  
    /* Instanciamos el objeto que presta la lógica de negocio de la aplicación */  
    VotoDAOWSService service = new VotoDAOWSService();  
    VotoDAOWS dao = service.getVotoDAOWSPort();  
  
    String url = FacesContext.getCurrentInstance().getExternalContext().getInitParameter("url");  
    BindingProvider bp = (BindingProvider) dao;  
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, url);  
  
    // Obtenemos los votos  
    ssii2.servicio.VotoBean v = traducirVotoParaServicio(this.voto);  
  
    List<ssii2.servicio.VotoBean> votos = dao.getVotos(v.getIdProcesoElectoral());  
  
    FacesContext.getCurrentInstance().getExternalContext().getSessionMap().  
        put("votosObtenidos", votos);  
    return "listadoVotos";  
}
```

Para comprobar que las modificaciones no han alterado nuestra aplicación, accedemos al servidor y consultamos un voto.

Información Sobre el Registro del Voto:

Código Respuesta: 000
Id Voto: 7
Id Proceso electoral: 3434
Marca Tiempo Voto: 2024-02-23 11:31:29.750894
Número DNI: 39739740E

Lista de votos del proceso electoral 3434

idVoto	idCircunscripcion	idMesaElectoral	idProcesoElectoral	Candidato/a Votado/a	Marca de Tiempo	codRespuesta
7	2323	1212	3434	Fernando Garcia	2024-02-23 11:31:29.750894	000

[Volver](#)

Se han borrado correctamente 1 votos del proceso electoral con identificador 3434

[Volver](#)

Ejercicio número 11:

Realice una importación manual del WSDL del servicio sobre el directorio de clases local. Anote en la memoria qué comando ha sido necesario ejecutar en la línea de comandos, qué clases han sido generadas y por qué. Téngase en cuenta que el servicio debe estar previamente desplegado.

```
wsimport -d build/client/WEB-INF/classes -p ssii2.servicio
```

<http://10.2.8.2:8080/P1-ws-ws/VotoDAOWSService?wsdl>

```
ana@ana:~/Documentos/SI/P1-ws$ wsimport -d build/client/WEB-INF/classes -p ssii2.servicio http://10.2.8.2:8080/P1-ws-ws/VotoDAOWSService?wsdl
analizando WSDL...

Generando código...

Compilando código...

Note: /home/ana/Documentos/SI/P1-ws/build/client/WEB-INF/classes/ssii2/servicio/VotoDAOWSService.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
ana@ana:~/Documentos/SI/P1-ws$
```

Se generan las clases

```

v build
  v client/WEB-INF
    v classes/ssii2/servicio
      J CensoBean.class
      J CompruebaCenso.class
      J CompruebaCensoResponse.class
      J DelVotos.class
      J DelVotosResponse.class
      J GetVotos.class
      J GetVotosResponse.class
      J IsDebug.class
      J IsDebugResponse.class
      J IsDirectConnection.class
      J IsDirectConnectionResponse.class
      J IsPrepared.class
      J IsPreparedResponse.class
      J ObjectFactory.class
      J package-info.class
      J RegistraVoto.class
      J RegistraVotoResponse.class
      J SetDebug.class
      J SetDebugResponse.class
      J SetDirectConnection.class
      J SetDirectConnectionResponse.class
      J SetPrepared.class
      J SetPreparedResponse.class
      J VotoBean.class
      J VotoDAOWS.class
      J VotoDAOWSService.class
```

Ejercicio número 12:

Complete el target generar-stubs definido en build.xml para que invoque a wsimport (utilizar la funcionalidad de ant exec para ejecutar aplicaciones). Téngase en cuenta que:

- El raíz del directorio de salida del compilador para la parte cliente ya está definido en build.properties como \${build.client}/WEB-INF/classes
- El paquete Java raíz (ssii2) ya está definido como \${paquete} (usar \${paquete}.servicio para el paquete).
- La URL ya está definida como \${wsdl.url}

Revisar la documentación disponible en <https://ant.apache.org/manual/Tasks/exec.html> para saber cómo invocar un programa dentro de un objetivo de ant mediante la etiqueta ant exec.

```
<target name="generar-stubs" depends="montar-jerarquia" description="Genera los stubs del cliente a partir del archivo WSDL">
  <!-- TODO - Implementar llamada wsimport -->
  <exec executable="wsimport">
    <arg line="-d ${build.client}/WEB-INF/classes" />
    <arg line="-p ${paquete}.voto" />
    <arg line="${wsdl.url}" />
  </exec>
  <delete file="${build}/${tmpvisaclientjar}" />
  <jar jarfile="${build}/${tmpvisaclientjar}" >
    <fileset dir="${build.client}/WEB-INF/classes" />
  </jar>
  <move file="${build}/${tmpvisaclientjar}" todir="${build.client}/WEB-INF/lib" />
</target>
```

Ejercicio número 13:

- Realice un despliegue de la aplicación completa en dos nodos tal y como se explica en la Figura 8. Habrá que tener en cuenta que ahora en el fichero build.properties hay que especificar la dirección IP del servidor de aplicaciones donde se desplegará la parte del cliente de la aplicación y la dirección IP del servidor de aplicaciones donde se desplegará la parte del servidor. Las variables as.host.client y as.host.server deberán contener esta información.

En build.properties especificamos en as.host.client la dirección IP donde se desplegará la parte del cliente con la IP 10.2.8.1 y en as.host.server la dirección IP donde se desplegará la parte del servidor con 10.2.8.2.

```
1  # Propiedades de despliegue de aplicacion de Voto
2  nombre=P1-ws
3  build=${basedir}/build
4  build.client=${build}/client
5  build.server=${build}/server
6  dist=${basedir}/dist
7  dist.client=${dist}/client
8  dist.server=${dist}/server
9  src=${basedir}/src
10 src.client=${src}/client
11 src.server=${src}/server
12 web=${basedir}/web
13 conf=${basedir}/conf
14 conf.serverws=${conf}/serverws
15 paquete=ssii2
16 war=${nombre}-cliente.war
17 wswar=${nombre}-ws.war
18 jar=${nombre}.jar
19 tmpvotoclientjar=${nombre}-clientstubs.jar
20 asadmin=${as.home}/bin/asadmin
21 as.home=${env.JEE_HOME}
22 as.lib=${as.home}/lib
23 as.user=admin
24 as.host.client=10.2.8.1
25 as.host.server=10.2.8.2
26 as.port=4848
27 as.passwordfile=${basedir}/passwordfile
28 as.target=server
29 wsdl.url=http://${as.host.server}:8080/${nombre}-ws/VotoDAOWebService?wsdl
30 wsimport=wsimport
```

```

BUILD SUCCESSFUL
Total time: 0 seconds
ana@ana:~/Documentos/SI/P1-ws$ ant compilar-cliente
Buildfile: /home/ana/Documentos/SI/P1-ws/build.xml

montar-jerarquia:

compilar-cliente:
[javac] Compiling 4 source files to /home/ana/Documentos/SI/P1-ws/build/client/WEB-INF/classes

BUILD SUCCESSFUL
Total time: 1 second
ana@ana:~/Documentos/SI/P1-ws$ ant empaquetar-servicio
Buildfile: /home/ana/Documentos/SI/P1-ws/build.xml

preparar-meta-inf-servicio:

empaquetar-servicio:
[delete] Deleting: /home/ana/Documentos/SI/P1-ws/dist/server/P1-ws-ws.war
[jar] Building jar: /home/ana/Documentos/SI/P1-ws/dist/server/P1-ws-ws.war

BUILD SUCCESSFUL
Total time: 0 seconds
ana@ana:~/Documentos/SI/P1-ws$ ant empaquetar-cliente
Buildfile: /home/ana/Documentos/SI/P1-ws/build.xml

preparar-web-inf-cliente:
[copy] Copying 8 files to /home/ana/Documentos/SI/P1-ws/build/client

empaquetar-cliente:
[jar] Building jar: /home/ana/Documentos/SI/P1-ws/dist/client/P1-ws-cliente.war

BUILD SUCCESSFUL
Total time: 0 seconds
ana@ana:~/Documentos/SI/P1-ws$ ant desplegar-servicio
Buildfile: /home/ana/Documentos/SI/P1-ws/build.xml

desplegar-servicio:
[exec] Application deployed with name P1-ws.
[exec] Command deploy executed successfully.

BUILD SUCCESSFUL
Total time: 2 seconds
ana@ana:~/Documentos/SI/P1-ws$ ant desplegar-cliente
Buildfile: /home/ana/Documentos/SI/P1-ws/build.xml

desplegar-cliente:
[exec] Application deployed with name P1-ws.
[exec] Command deploy executed successfully.

BUILD SUCCESSFUL
Total time: 11 seconds
ana@ana:~/Documentos/SI/P1-ws$ 

```

- Probar a registrar votos correctos a través de la página testbd.xhtml. Ejecutar las consultas SQL necesarias para comprobar que se realiza el registro del voto. Anotar en la memoria práctica los resultados en forma de consulta SQL y resultados sobre la tabla de votos.

Incluye evidencias en la memoria de la realización del ejercicio.

Vamos a la url <http://10.2.8.1:8080/P1-ws-cliente/testdb.xhtml> y registramos un voto.

Complete la información sobre el voto:

Voto:

Id Mesa electoral: 1212
Id Circunscripción: 2323
Id Proceso electoral: 3434
Nombre candidato/a votado/a: Fernando Garcia

Censo:

Número de DNI: 39739740E
Nombre y Apellidos: Jose Moreno Locke
Fecha de Nacimiento: 09/04/66
Código Autorización: 729
Debug: ☒ SI ☐ No
Prepared Statements: ☒ SI ☐ No
Conexión directa al a BD: ☐ SI ☒ No

Registrar Voto

Consulta de Votos:

Id Proceso electoral:
Consultar Votos

Borrado de Votos:

Id Proceso electoral:
Borrar Votos

Vemos que el voto se ha registrado correctamente, mostrando la información sobre su registro.

10.2.8.1:8080/P1-ws-cliente/censo.xhtml?sessionId=f62e13fabcd0d089403a17ffffa3

Gmail YouTube Maps

Información Sobre el Registro del Voto:

Código Respuesta: 000

Id Voto: 1

Id Proceso electoral: 3434

Marca Tiempo Voto: 2024-02-25 09:31:07.146663

Número DNI: 39739740E

Comprobamos que el voto se ha guardado correctamente.

10.2.8.1:8080/P1-ws-cliente/testbd.xhtml?sessionId=f64fb63a0a4c26916838a066f4d6

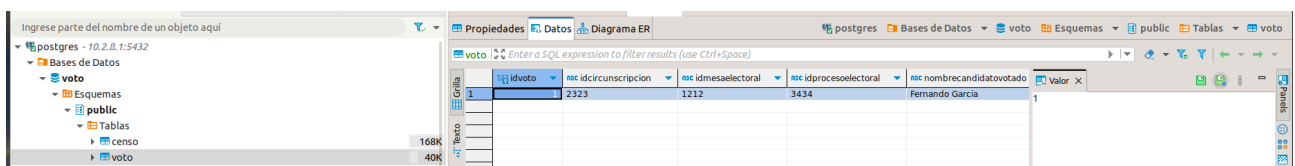
Gmail YouTube Maps

Lista de votos del proceso electoral 3434

idVoto	idCircunscripcion	idMesaElectoral	idProcesoElectoral	Candidato/a Votado/a	Marca de Tiempo	codRespuesta
1	2323	1212	3434	Fernando Garcia	2024-02-25 09:31:07.146663	000

Volver

En la aplicación DBeaver en la que tenemos conectada la base de datos, miramos que se ha añadido el voto correctamente.



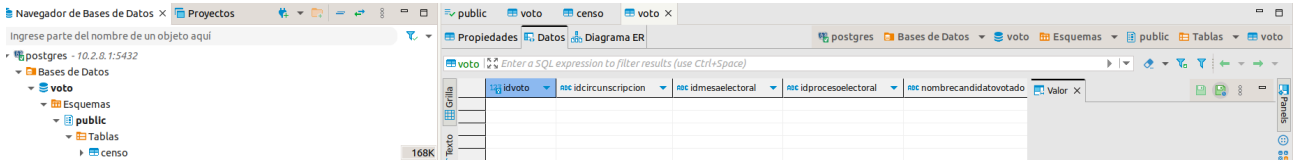
idvoto	idcircunscripcion	idmesaelectoral	idprocesoelectoral	nombrecandidatovotado	valor
1	2323	1212	3434	Fernando Garcia	1

Borramos el voto.

Se han borrado correctamente 1 votos del proceso electoral con identificador 3434

[Volver](#)

Y comprobamos en DBeaver que el voto también se ha borrado de la base de datos.



Cuestión número 1:

Teniendo en cuenta el diagrama de la Figura 3, indicar las páginas xhtml y métodos, por los que se pasa para registrar un voto desde voto.xhtml, pero en el caso de uso en que se introduzca una información censal incorrecta. Es decir, de una persona que no se encuentre en la tabla de censo.

La página principal es voto.xhtml en la que el usuario introduce su información del censo, esta página valida parámetros introducidos por el usuario. Entonces si no está registrado en el censo, se redirige a censo.xhtml el cual en la clase ControladorBean.java con el método compruebaCenso() comprueba si el usuario se encuentra en el censo. Si el usuario no se encuentra en el censo se mostrará un mensaje de error con error.xhtml.

Cuestión número 2:

De los diferentes Managed Beans y páginas xhtml que se usan en la aplicación, ¿podría indicar cuáles son las páginas xhtml encargadas de obtener la información sobre el voto y el censo cuando se usa voto.xhtml para realizar el registro del voto, y cuáles son los Managed Beans encargados de almacenarla? ¿Qué información se obtiene en cada página xhtml sobre el voto?

Las páginas encargadas de obtener la información sobre el voto:
voto.html: se encarga de obtener la información del voto.

Los Managed Beans encargados de almacenarla:

VotoBean: se almacena la información del voto.

CensoBean: se almacena la información del censo.

ControladorBean: se encarga de registrar el voto, borrar los votos y consultar los votos.

Cuestión número 3 :

Respecto a la información sobre el voto que maneja la aplicación (voto en sí y censo), ¿cómo accede a ella el controlador ControladorBean?

El ControladorBean accede a esta información a través del método traducirVotoParaServicio que se encuentra en VotoBean y CensoBean.

Cuestión número 4 :

Enumere las diferencias que existen en la invocación de páginas xhtml, a la hora de realizar el registro del voto, cuando se utiliza la página de pruebas extendida testbd.xhtml frente a cuando se usa voto.xhtml. ¿Podría indicar por qué funciona correctamente el registro del voto cuando se usa testbd.xhtml a pesar de las diferencias observadas?

Si comparamos testbd.xhtml con voto.xhtml. testbd.xhtml solicita información tanto del voto como del censo mientras que voto.xhtml solo de voto.

testbd.xhtml tiene más funcionalidades que voto.xhtml como por ejemplo consultar y eliminar votos.

Las dos registran un voto, por lo tanto, las dos funcionan correctamente.