

APUNTES DEL PROYECTO VOTACIONES CON ORIENTACIÓN A OBJETOS

BBDD

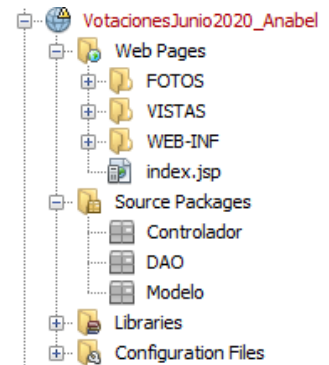
Lo primero que vamos a hacer, es crear la base de datos que ya va a contener a los candidatos de los partidos y el resto de información menos los votantes.

```
-- -----
-- Host: 127.0.0.1
-- Versión del servidor: 10.4.6-MariaDB - mariadb.org binary distribution
-- SO del servidor: Win64
-- HeidiSQL Versión: 10.2.0.5709
-- -----

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
-- Volcando estructura de base de datos para bd_votantes
DROP DATABASE IF EXISTS `bd_votantes`;
CREATE DATABASE IF NOT EXISTS `bd_votantes` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `bd_votantes`;
-- Volcando estructura para tabla bd_votantes.candidatos
DROP TABLE IF EXISTS `candidatos`;
CREATE TABLE IF NOT EXISTS `candidatos` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nombre_apellidos` varchar(1000) NOT NULL,
  `orden` int(11) NOT NULL,
  `id_partido` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_partido` (`id_partido`),
  CONSTRAINT `FK_partido` FOREIGN KEY (`id_partido`) REFERENCES `partido` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=41 DEFAULT CHARSET=latin1;
-- Volcando datos para la tabla bd_votantes.candidatos: ~20 rows (aproximadamente)
/*!40000 ALTER TABLE `candidatos` DISABLE KEYS */;
REPLACE INTO `candidatos` (`id`, `nombre_apellidos`, `orden`, `id_partido`) VALUES
(1, 'Carmen Navarro Lacooba', 1, 1),
(2, 'Manuel Serrano López', 2, 1),
(3, 'Manuel Serena Fernández', 3, 1),
(4, 'Cristina García Martínez', 4, 1),
(5, 'Manuel Gabriel González Ramos', 1, 2),
(6, 'Maria Luisa Vilches Ruiz', 2, 2),
(7, 'José Carlos Díaz Rodríguez', 3, 2),
(8, 'Estefanía Escribano Villena', 4, 2),
(9, 'Maria Dolores Arteaga Espinosa de los Monteros', 1, 4),
(10, 'Hugo Gabriel Guillen Malagón', 2, 4),
(11, 'Ana Isabel Martínez Molina', 3, 4),
(12, 'Cristian Cuerda González', 4, 4),
(13, 'Maria Pérez Segovia', 1, 3),
(14, 'Emilio Zamora Martínez', 2, 3),
(15, 'Darcy Gioconda Cárdenas Barrera', 3, 3),
(16, 'Sergio León Bullón', 4, 3),
(17, 'Rafael Fernández-Lomana Gutiérrez', 1, 5),
(18, 'Juan Francisco Robles Descalzo', 2, 5),
(19, 'Maria Remedios Gil Martínez', 3, 5),
(20, 'Maria Teresa Fernández Lara', 4, 5);
/*!40000 ALTER TABLE `candidatos` ENABLE KEYS */;
-- Volcando estructura para tabla bd_votantes.parametros
DROP TABLE IF EXISTS `parametros`;
CREATE TABLE IF NOT EXISTS `parametros` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `circunscripcion` varchar(50) NOT NULL,
  `candidatos` int(11) NOT NULL DEFAULT 0,
  `tipoConsulta` varchar(50) NOT NULL,
  `fechaConsulta` date NOT NULL,
  `estadoEscrutinió` enum('Cerrado','Abierto','Finalizado') NOT NULL DEFAULT 'Cerrado',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
-- Volcando datos para la tabla bd_votantes.parametros: ~1 rows (aproximadamente)
/*!40000 ALTER TABLE `parametros` DISABLE KEYS */;
REPLACE INTO `parametros` (`id`, `circunscripcion`, `candidatos`, `tipoConsulta`, `fechaConsulta`, `estadoEscrutinió`) VALUES
(1, 'ALBACETE', 4, 'ELECCIONES', '2019-11-10', 'Cerrado');
/*!40000 ALTER TABLE `parametros` ENABLE KEYS */;
-- Volcando estructura para tabla bd_votantes.partido
DROP TABLE IF EXISTS `partido`;
CREATE TABLE IF NOT EXISTS `partido` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nombrepartido` varchar(90) NOT NULL,
  `siglas` varchar(50) NOT NULL,
  `logo` varchar(50) NOT NULL,
  `votos` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `nombrepartido` (`nombrepartido`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
-- Volcando datos para la tabla bd_votantes.partido: ~5 rows (aproximadamente)
/*!40000 ALTER TABLE `partido` DISABLE KEYS */;
REPLACE INTO `partido` (`id`, `nombrepartido`, `siglas`, `logo`, `votos`) VALUES
(1, 'PARTIDO POPULAR', 'PP', 'logos/pp.png', 26589),
(2, 'PARTIDO SOCIALISTA OBRERO ESPAÑOL', 'PSOE', 'logos/psoe.png', 27080),
(3, 'UNIDAS PODEMOS', 'UP', 'logos/podemos.png', 10220),
(4, 'CIUDADANOS', 'CS', 'logos/cs.jpg', 8711),
(5, 'VOX', 'VOX', 'logos/vox.jpg', 20479);
/*!40000 ALTER TABLE `partido` ENABLE KEYS */;
-- Volcando estructura para tabla bd_votantes.votante
DROP TABLE IF EXISTS `votante`;
CREATE TABLE IF NOT EXISTS `votante` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nif` varchar(9) NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `apellidos` varchar(100) NOT NULL,
  `domicilio` varchar(100) NOT NULL,
  `fechanac` date NOT NULL,
  `password` varbinary(16) NOT NULL,
  `votado` enum('V','NV') DEFAULT 'NV',
  `rol` enum('Administrador','Votante') DEFAULT 'Votante',
  PRIMARY KEY (`id`),
  UNIQUE KEY `nif` (`nif`)
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=latin1;
-- Volcando datos para la tabla bd_votantes.votante: ~2 rows (aproximadamente)
/*!40000 ALTER TABLE `votante` DISABLE KEYS */;
REPLACE INTO `votante` (`id`, `nif`, `nombre`, `apellidos`, `domicilio`, `fechanac`, `password`, `votado`, `rol`) VALUES
(11, '06290540V', 'Anabel', 'Morales Nuñez', 'Pasaje San Luis', '1999-08-08', _binary 0xBEAC8D7B8B1732CC80A1A3F937F24856, 'V', 'Administrador'),
(17, '06254578C', 'Daniel', 'Romero Muñoz', 'Calle Noguera 85', '1985-05-08', _binary 0x0A636278918A58A937AD1A457CB5AF5, 'NV', 'Votante');
/*!40000 ALTER TABLE `votante` ENABLE KEYS */;
/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

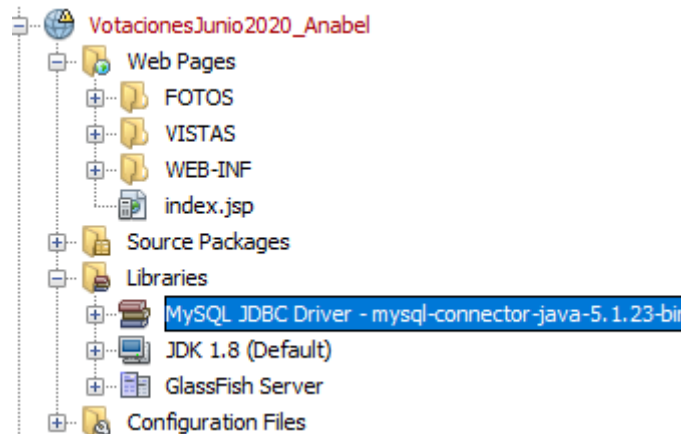
ESTRUCTURA DEL PROYECTO EN CARPETAS

Vamos a estructurar el proyecto. Dentro de Web Pages, vamos a tener las carpetas que contengan las vistas, el contenido multimedia y todos los CSS y Javascript que necesitemos. Después dentro de Source Packages, creamos 3 paquetes: Controlador (incluyen todos los controladores), DAO (incluye la conexión y el DAOOperaciones) y Modelo (incluye las clases).



CONEXIÓN CON LA BASE DE DATOS

Antes de que se nos olvide, vamos a importar la librería MYSQL.



Dentro de la carpeta DAO, vamos a crear el archivo Conexión.java, es un archivo Java Class. En este archivo vamos a conectar la base de datos con el proyecto.

```
public class Conexion {
    private java.sql.Connection conexion;
    private static Conexion conexionunica = null;

    public Conexion() throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/bd_votantes";
        this.conexion = (com.mysql.jdbc.Connection) DriverManager.getConnection(url, "root", "");
    }

    public synchronized static Conexion Connect() throws ClassNotFoundException, SQLException {
        if (conexionunica == null) {
            conexionunica = new Conexion();
        }
        return conexionunica;
    }

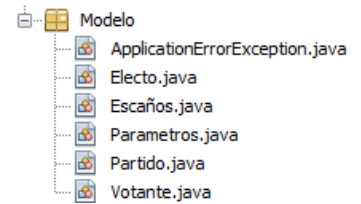
    public java.sql.Connection getConexion() {
        return conexion;
    }

    public void destroy() throws SQLException {
        conexion.close();
    }
}
```

CREAR LAS CLASES EN MODELO Y DAO OPERACIONES EN DAO

Cogemos los atributos que tenemos en la base de datos y creamos sus respectivas clases en Java. Además, les añadimos los sets y get de todos los parámetros y conforme necesitemos iremos creando los constructores.

ApplicationErrorException.java, es la excepción que creamos nosotros para capturar las excepciones y controlarlas.



```
public class ApplicationErrorException extends Exception{
    private String lugar;
    private int codigo;
    public ApplicationErrorException(String mensaje,int codigo, String lugar) {
        super(mensaje); //ESTAMOS HEREDANDO DE EXCEPTION
        this.codigo=codigo;
        this.lugar=lugar;
    }
    public String getLugar() {
        return lugar;
    }

    public void setLugar(String lugar) {
        this.lugar = lugar;
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
}
```

Creamos el DAOOperaciones en el paquete DAO, que va a contener todas las funciones y métodos de nuestro proyecto.

FUNCIONALIDAD DEL PROYECTO

REGISTRAR USUARIO

El proyecto debe empezar con el inicio de sesión o el registro de los votantes, para que ellos puedan votar o introduzcan sus datos y voten.

En este caso, he hecho una interfaz para iniciar el proyecto que contiene el registro y el inicio de sesión juntos.

Bienvenido a las Votaciones Generales de Junio-2020

Iniciar Sesión

NIF/NIE

Password

INICIAR SESIÓN

Hola, nuevo votante!

Si todavía no dispones de cuenta para votar, es la hora.

REGISTRARSE

Como es la primera vez que vamos a ejecutar el proyecto, entendemos que no tenemos ningún usuario, por lo que empezamos por el registro.

Introducimos los datos de nuestro nuevo votante, clicamos sobre registrar. Es muy importante recordar que, en este caso para rescatar en el controlador, lo introducido en el input, tenemos que añadirle el atributo name=""

```
<form action="./ControladorInsertarVotante" method="POST">
  <h1>Crear Cuenta</h1>
  <input type="text" placeholder="NIF/NIE" id="nif" name="nif"/>
  <input type="text" placeholder="Nombre" id="nombre" name="nombre"/>
  <input type="text" placeholder="Apellidos" id="apellidos" name="apellidos"/>
  <input type="text" placeholder="Domicilio" id="domicilio" name="domicilio"/>
  <input type="date" name="fechanac" id="fechanac" />
  <input type="password" placeholder="Password" id="password" name="password"/>
  <button type="submit">Registrarse</button>
</form>
```

Una vez que hemos enviado los datos al controlador, lo primero es recoger los valores y crear un objeto. La conexión, en este trabajo, la establecemos así, y siempre hay que pasarle al método la conexión.

También es muy importante añadir los dos catch porque si no, no funciona. Otra cosa que es muy importante, son las dos primeras líneas, que nos permiten insertar en la base de datos tanto tildes como ñ o caracteres raros. Vamos a trabajar con fechas LOCALDATE.

```
response.setContentType("text/html;charset=UTF-8");
request.setCharacterEncoding("UTF-8");
try (PrintWriter out = response.getWriter()) {
    String nif = request.getParameter("nif");
    String nombre = request.getParameter("nombre");
    String apellidos = request.getParameter("apellidos");
    String domicilio = request.getParameter("domicilio");
    LocalDate fechanac = LocalDate.parse(request.getParameter("fechanac"));
    String password = request.getParameter("password");

    Votante objVotante = new Votante(nif, nombre, apellidos, domicilio, fechanac, password)
    try {
        Conexion miConexion = Conexion.Connect();
        Connection conexion = (Connection) miConexion.getConexion();
        new DAOOperaciones().insertarVotante(objVotante, conexion);
        response.sendRedirect("./VISTAS/Menu.jsp");
    } catch (ApplicationErrorException AE) {
        response.sendRedirect("./VISTAS/Solucion.jsp?codigo=error-insertar-votante");
    } catch (ClassNotFoundException | SQLException ex) {
        response.sendRedirect("./VISTAS/Solucion.jsp?codigo=error2-insertar-votante");
    }
}
```

En cuanto al método insertarVotante(), al cual le pasamos el objeto de tipo Votante y la conexión, lo hacemos de esta forma. **MUY IMPORTANTE AÑADIR LA EXCEPCIÓN A CADA MÉTODO.**

```
public void insertarVotante(Votante _objVotante, Connection _Conexion) throws ApplicationErrorException {
    try {
        String sql = "INSERT INTO votante(nif, nombre, apellidos, domicilio, fechanac, password) VALUES (" +
            _objVotante.getNif() + ", " + _objVotante.getNombre() + ", " + _objVotante.getApellidos() + ", " +
            _objVotante.getDomicilio() + ", " + _objVotante.getFechanac() + ", " + _objVotante.getPassword() + ")";
        PreparedStatement PrepStm = _Conexion.prepareStatement(sql);
        PrepStm.setString(1, _objVotante.getNif());
        PrepStm.setString(2, _objVotante.getNombre());
        PrepStm.setString(3, _objVotante.getApellidos());
        PrepStm.setString(4, _objVotante.getDomicilio());
        PrepStm.setDate(5, java.sql.Date.valueOf(_objVotante.getFechanac()));
        PrepStm.setString(6, _objVotante.getPassword());

        //SE HACE EXECUTEUPDATE PORQUE ES UN INSERT
        int filas = PrepStm.executeUpdate();
        if (filas != 1) {
            throw new ApplicationErrorException("VOTANTE NO INSERTADO", 0, "DAOOperaciones.insertar");
        }
    } catch (SQLException SQLE) {
        String mensaje = SQLE.getMessage();
        int codigo = SQLE.getErrorCode();
        throw new ApplicationErrorException(mensaje, codigo, "ERROR EN LA INSERCIÓN DE VOTANTES");
    }
}
```

INICIAR SESIÓN CON EL USUARIO REGISTRADO

Introducir el nif y la contraseña de nuestro usuario registrado. Creamos un controlador, el cual va a recoger los datos introducidos.

```
try (PrintWriter out = response.getWriter()) {
    HttpSession session = request.getSession(true);
    String nif = request.getParameter("dni");
    String password = request.getParameter("contraseña");

    try {
        Conexion miConexion = Conexion.Connect();
        Connection conexion = (Connection) miConexion.getConnection();
        Votante objVotante = new DAOOperaciones().buscarVotante(nif, password, conexion);
        session.setAttribute("objVotante", objVotante);
        response.sendRedirect("../VISTAS/Menu.jsp");
    } catch (ApplicationErrorException ae) {
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-login-votante");
    } catch (ClassNotFoundException | SQLException ex) {
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error2-login-votante");
    }
}
```

Creamos un objeto votante, al cual, le vamos a pasar la información entera del votante mediante un método que va a devolver un objeto Votante.

Guardamos en sesión el objeto votante para poder hacer uso de contenido.

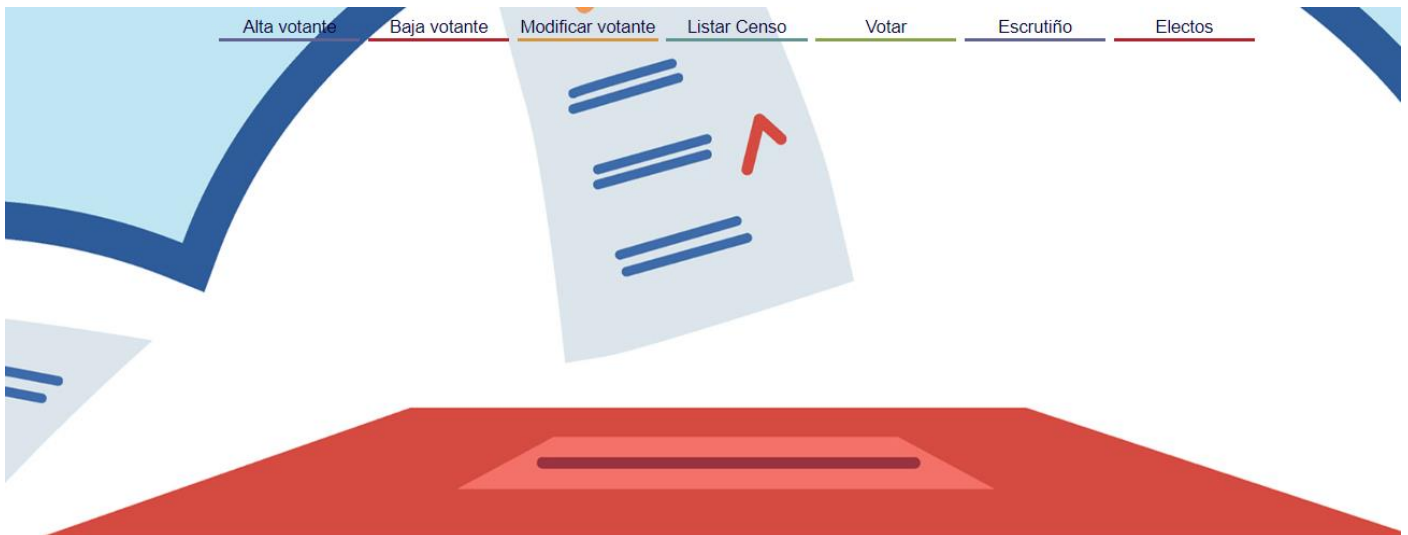
```
public Votante buscarVotante(String _nif, String _password, Connection _conexion) throws Application
    try {
        PreparedStatement PrepStm = _conexion.prepareStatement("SELECT * FROM votante WHERE nif=? AND password=?");
        PrepStm.setString(1, _nif);
        PrepStm.setString(2, _password);

        ResultSet rs = PrepStm.executeQuery();
        if (rs.next()) {
            String nif = rs.getString("nif");
            String nombre = rs.getString("nombre");
            String apellidos = rs.getString("apellidos");
            String domicilio = rs.getString("domicilio");
            LocalDate fechanac = rs.getDate("fechanac").toLocalDate();
            String password = rs.getString("password");
            String votado = rs.getString("votado");
            String rol = rs.getString("rol");

            return new Votante(nif, nombre, apellidos, domicilio, fechanac, password, votado, rol);
        } else {
            throw new ApplicationErrorException("NO SE HA ENCONTRADO EL VOTANTE", 0, "DAOOperaciones.");
        }
    } catch (SQLException SQLE) {
        String mensaje = SQLE.getMessage();
        int codigo = SQLE.getErrorCode();
        throw new ApplicationErrorException(mensaje, codigo, "ERROR EN EL SELECT");
    }
}
```

USUARIO VOTANTE

OPCIONES VALIDAS DE MENU



BAJA VOTANTE

En este aparte, entiendo que ese votante que se ha logueado es el que va a darse de baja. Solamente podrá darse de baja si no ha votado.

BAJA
¿Estás seguro de que quiere darse de baja? Recuerda que no podrás darte de baja si ya has votado.

DAR DE BAJA

Tenemos que recoger los datos y comprobar que son del votante que hemos logueado.

```
try (PrintWriter out = response.getWriter()) {  
  
    HttpSession session = request.getSession(true);  
    String nif = request.getParameter("dni");  
    String password = request.getParameter("contraseña");  
    Parametros objParametros = (Parametros) session.getAttribute("objParametros");  
  
    try {  
        Conexion miConexion = Conexion.Connect();  
        Connection Conexion = (Connection) miConexion.getConexion();  
  
        new DAOOperaciones().borrarVotante(nif, password, Conexion);  
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=baja-votante");  
    } catch (ApplicationErrorException AE) {  
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-baja-votante");  
    } catch (ClassNotFoundException | SQLException ex) {  
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-baja-votante");  
    }  
}
```

El método para borrar un votante es el siguiente:

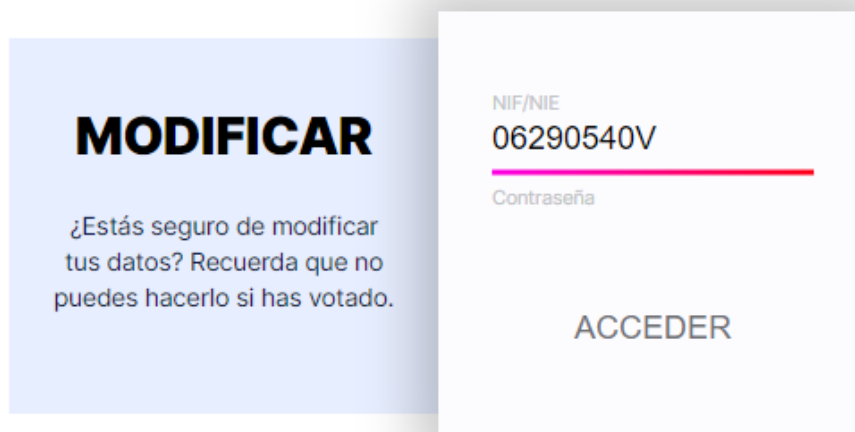
```
public void borrarVotante(String _nif, String _password, Connection _Conexion) throws ApplicationException {
    try {
        String sql = "DELETE FROM votante WHERE nif=? AND votado='NV' AND AES_DECRYPT(password,'anabel')=?";
        PreparedStatement PrepStm = _Conexion.prepareStatement(sql);
        PrepStm.setString(1, _nif);
        PrepStm.setString(2, _password);

        //SE HACE EXECUTEUPDATE PORQUE ES UN DELETE
        int filasBorradas = PrepStm.executeUpdate();
        if (filasBorradas != 1) {
            throw new ApplicationException("VOTANTE NO BORRADO", 0, "DAOoperaciones.borrarVotante");
        }
    } catch (SQLException SQLE) {
        String mensaje = SQLE.getMessage();
        int codigo = SQLE.getErrorCode();
        throw new ApplicationException(mensaje, codigo, "ERROR EN EL BORRADO DE VOTANTES");
    }
}
```

MODIFICAR VOTANTE

Para modificar los datos de un votante tenemos que comprobar la contraseña y el nif. Si todo es correcto y no ha votado, se pueden modificar todos los datos menos el NIF. Por el contrario, si los datos no son correctos, pues sacaremos el mensaje de error.

La interfaz será la misma que la anterior (Borrar Votante).



Esto lo enviamos a un controlador, donde vamos a recoger el nif y la contraseña, pero en este caso ya disponemos de un método que nos va a comprobar si es ese votante o no. Vamos a guardar todos los datos en un objeto Votante y se lo pasamos a la sesión.

```
try (PrintWriter out = response.getWriter()) {
    HttpSession session = request.getSession(true);
    String nif = request.getParameter("dni");
    String password = request.getParameter("contraseña");

    try {
        Conexion miConexion = Conexion.Connect();
        Connection Conexion = (Connection) miConexion.getConexion();
        Votante objVotante = new DAOoperaciones().buscarVotante(nif, password, Conexion);
        session.setAttribute("objVotante", objVotante);
        response.sendRedirect("../VISTAS/ModificarVotante2.jsp");
    } catch (ApplicationErrorException AE) {
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-modificar-votante");
    } catch (ClassNotFoundException | SQLException ex) {
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-modificar-votante");
    }
}
```


Como le hemos pasado todos los datos del objeto Votante a la sesión, ahora para hacerlo de una manera más sencilla, lo que vamos a hacer es pintarlos en un formulario con opción a modificarlos, así el usuario si quiere modificar los datos y si no, no. Como el NIF no se puede modificar, pues no lo muestro.

DATOS PERSONALES

	<small>Nombre</small> ANABEL
	<small>Apellidos</small> MORALES
	<small>Domicilio</small> AVENIDA DE ESPAÑA 35, 3ºDRCH
	<small>Contraseña</small>
	08/08/1999

ACEPTAR >

Estos son los datos que vamos a utilizar, pero claro necesitamos el nif para modificar los datos, por lo que lo muestro con un input-hidden para poder rescatarlo en el controlador.

```
<input type="hidden" value="<%= objVotante.getNif()%>" name="nif" id="nif">
```

Ahora en el controlador, vamos a recoger todos los datos, incluido el nif y creamos un objeto Votante con todos esos datos, el cual se lo vamos a pasar el método que va a actualizar los datos.

```
try (PrintWriter out = response.getWriter()) {
    HttpSession session = request.getSession(true);

    String nif=request.getParameter("nif");
    String nombre = request.getParameter("nombre");
    String apellidos = request.getParameter("apellidos");
    String domicilio = request.getParameter("domicilio");
    LocalDate fechanac = LocalDate.parse(request.getParameter("fechanac"));
    String password = request.getParameter("password");

    Votante objVotante = new Votante(nif, nombre, apellidos, domicilio, fechanac, password);
    try {
        Conexion miConexion = Conexion.Connect();
        Connection conexion = (Connection) miConexion.getConexion();
        new DAOOperaciones().modificarVotante(objVotante, conexion);
        session.setAttribute("objVotante", objVotante);
        response.sendRedirect("../VISTAS/Menu.jsp");
    } catch (ApplicationErrorException AE) {
        String mensaje = AE.getMessage();
        int codigo = AE.getCodigo();
        out.print(mensaje+"-"+codigo);
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-modificar-votante");
    } catch (ClassNotFoundException | SQLException ex) {
        response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error2-modificar-votante");
    }
}
```


En cuanto al método, mediante la sentencia UPDATE en SQL, vamos a poder modificar los valores. Le vamos a pasar todos y los que sean iguales, los dejará iguales y los que cambien, los cambiará.

```
public void modificarVotante(Votante _objVotante, Connection _Conexion) throws ApplicationErrorException
{
    try {
        String sql = "UPDATE votante SET nombre=?, apellidos=?, domicilio=?, fechanac=?, password=AES";
        PreparedStatement PrepStm = _Conexion.prepareStatement(sql);
        PrepStm.setString(1, _objVotante.getNombre());
        PrepStm.setString(2, _objVotante.getApellidos());
        PrepStm.setString(3, _objVotante.getDomicilio());
        PrepStm.setDate(4, java.sql.Date.valueOf(_objVotante.getFechanac()));
        PrepStm.setString(5, _objVotante.getPassword());
        PrepStm.setString(6, _objVotante.getNif());

        //SE HACE EXECUTEUPDATE PORQUE ES UN INSERT
        int filas = PrepStm.executeUpdate();
        if (filas != 1) {
            throw new ApplicationErrorException("VOTANTE NO MODIFICADO", 0, "DAOOperaciones.modificarV");
        }
    } catch (SQLException SQLE) {
        String mensaje = SQLE.getMessage();
        int codigo = SQLE.getErrorCode();
        throw new ApplicationErrorException(mensaje, codigo, "ERROR EN LA MODIFICACIÓN DE VOTANTES");
    }
}
```

LISTADO DE CENSO

En este listado vamos a mostrar la información sobre todos los votantes, para ella vamos a empezar creando un controlador, que será donde saquemos un ArrayList que contenga toda la información necesaria.

Creamos una sesión nueva donde almacenaremos toda la información que hemos sacado.

```
try (PrintWriter out = response.getWriter()) {

    ArrayList<Votante> ListadoCenso = new ArrayList();
    try {
        //SINGLETON
        Conexion miConexion = Conexion.Connect();
        Connection Conexion = (Connection) miConexion.getConexion();
        ListadoCenso = new DAOOperaciones().getCenso(Conexion);

        HttpSession session = request.getSession(true);
        session.setAttribute("ListadoCenso", ListadoCenso);

        response.sendRedirect("./VISTAS/ListadoCenso.jsp");
    } catch (ApplicationErrorException app) {
        response.sendRedirect("./VISTAS/Solucion.jsp?codigo=error-listado-censo");
    } catch (ClassNotFoundException | SQLException ex) {
        response.sendRedirect("./VISTAS/Solucion.jsp?codigo=error-listado-censo");
    } finally {
        out.close();
    }
}
```

En el método getCenso().

```
public ArrayList<Votante> getCenso(Connection _Conexion) throws ApplicationErrorException {
    ArrayList<Votante> ListadoCenso = new ArrayList<>();
    try {
        Statement s = _Conexion.createStatement();
        ResultSet rs = s.executeQuery("SELECT nif,nombre,apellidos,domicilio,fechanac,votado FROM votante");

        while (rs.next()) {
            ListadoCenso.add(
                new Votante(
                    rs.getString("nif"),
                    rs.getString("nombre"),
                    rs.getString("apellidos"),
                    rs.getString("domicilio"),
                    rs.getDate("fechanac").toLocalDate(),
                    rs.getString("votado")
                )
            );
        }
    } catch (SQLException ex) {
        throw new ApplicationErrorException(ex.getMessage(), ex.getErrorCode(), "ERROR SQL - LISTADO CENSO");
    }

    return ListadoCenso;
}
```

Si todo es correcto, nos llevara a la vista ListadoCenso donde vamos a pintar la información mediante un bucle.

```
<% HttpSession session = request.getSession(true);
    ArrayList<Votante> ListadoCenso = (ArrayList) session.getAttribute("ListadoCenso");
    for (Votante objVotante : ListadoCenso) {
%>

<tr>
    <th scope="row"><%= objVotante.getNif() %></th>
    <td><%= objVotante.getNombre() %></td>
    <td><%= objVotante.getApellidos() %></td>
    <td><%= objVotante.getDomicilio() %></td>
    <td><%= objVotante.getFechaNacString() %></td>
    <td><%= objVotante.getVotado() %></td>
</tr>
<% } %>
```

LISTAR PARTIDOS-VOTAR

Lo que vamos a hacer es sacar un listado de los partidos junto a su información para así poder votar a uno de ellos. Volvemos a crear un ArrayList en el cual vamos a incluir la información completa sobre los partidos. Creamos una sesión nueva donde vamos a almacenar la información.

```
try (PrintWriter out = response.getWriter()) {
    //SINGLETON
    Conexion miConexion = Conexion.Connect();
    Connection Conexion = (Connection) miConexion.getConexion();

    ArrayList<Partido> ArrayPartidos = new DAOOperaciones().getPartidos(Conexion);
    HttpSession session = request.getSession(true);
    session.setAttribute("ArrayPartidos", ArrayPartidos);
    response.sendRedirect("../VISTAS/ListarPartidos.jsp");
} catch (ApplicationErrorException AEX) {
    response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-listar-partidos");
} catch (ClassNotFoundException | SQLException ex) {
    response.sendRedirect("../VISTAS/Solucion.jsp?codigo=error-listar-partidos");
}
```

En cuanto al método, es igual al anterior que hemos hecho sobre los votantes.

```
public ArrayList<Partido> getPartidos(Connection _Conexion) throws ApplicationException {  
  
    ArrayList<Partido> ArrayPartidos = new ArrayList<>();  
    try {  
        Statement s = _Conexion.createStatement();  
        ResultSet rs = s.executeQuery("SELECT * FROM partido");  
  
        while (rs.next()) {  
            ArrayPartidos.add(  
                new Partido(  
                    rs.getString("nombrepartido"),  
                    rs.getString("siglas"),  
                    rs.getString("logo"),  
                    rs.getInt("votos")  
                )  
            );  
        }  
    } catch (SQLException ex) {  
        throw new ApplicationException(ex.getMessage(), ex.getErrorCode(), "ERROR SQL - LISTADO PARTIDOS");  
    }  
  
    return ArrayPartidos;  
}
```

Si todo es correcto, lo pintamos en una vista. He intentado de varias maneras, pero la mas eficaz es esta:

```
<% ArrayList<Partido> ArrayPartidos = (ArrayList) session.getAttribute("ArrayPartidos");  
    for (Partido objPartido : ArrayPartidos) {  
%>  
  
    <tr>  
        <td></td>  
        <td><%= objPartido.getSiglas() %></td>  
        <td><%= objPartido.getNombrepartido() %></td>  
        <td><input type="radio" id="siglas" name="siglas" value="<%=objPartido.getSiglas() %>" checked/></td>  
    </tr>  
    <% } %>  
  
    <tr>  
        <td colspan="4"><button value="submit" name="Votar">VOTAR</button></td>  
    </tr>
```

Todos estos datos, los vamos a recoger en un controlador. Primero vamos a rescatar si el botón que hemos pulsado es el de VOTAR o bien, el de volver atrás. Si es el de votar, recogemos el objVotante que se encuentra en la sesión, después recogemos las siglas del partido seleccionado y creamos un objeto Partido (su constructor hay que crearlo nuevo para que solo tenga las siglas). Volvemos a crear otra sesión y lo metemos, para saber cual es el partido que ha votado.

```
try (PrintWriter out = response.getWriter()) {  
    Conexion miConexion = Conexion.Connect();  
    Connection Conexion = (Connection) miConexion.getConexion();  
    if (request.getParameter("Votar") != null) {  
        HttpSession session = request.getSession(true);  
        Votante objVotante = (Votante) session.getAttribute("objVotante");  
  
        String siglas = request.getParameter("siglas");  
        Partido objPartido = new Partido(siglas);  
  
        session.setAttribute("PartidoVotado", objPartido);  
        DAOOperaciones objOperaciones = new DAOOperaciones();  
  
        try {  
            Conexion.setAutoCommit(false);  
            objOperaciones.votar(objVotante, Conexion);  
            objOperaciones.registrarVoto(objPartido, Conexion);  
            Conexion.commit();  
            Conexion.setAutoCommit(true);  
        } catch (ApplicationErrorException AE) {  
            response.sendRedirect("./VISTAS/Solucion.jsp?codigo=error-registro-voto");  
        } catch (SQLException SQLE) {  
            out.println(SQLE.getMessage());  
  
            if (Conexion != null) {  
                try {  
                    Conexion.rollback();  
                } catch (SQLException SQLE2) {  
                    out.println(SQLE2.getMessage());  
                }  
            }  
        }  
    }  
}
```

Hacemos los métodos por separado, primero el votante cuando haya votado, cambia su rol y aparece como V.

```
public void votar(Votante _votante, Connection _conexion) throws ApplicationException {
    int resultado;
    try {
        Statement s = _conexion.createStatement();
        resultado = s.executeUpdate("UPDATE votante SET votado='V' WHERE nif='" + _votante.getNif() + "'");
        if (resultado != 1) {
            throw new ApplicationException("NO SE HA VOTADO CORRECTAMENTE", 0, "DAOperaciones.votar()");
        }
    } catch (SQLException SQLE) {
        String mensaje = SQLE.getMessage();
        int codigo = SQLE.getErrorCode();
        throw new ApplicationException(mensaje, codigo, "ERROR EN VOTAR");
    }
}
```

Y después, sumamos un voto al partido seleccionado, mediante las siglas.

```
public void registrarVoto(Partido _partido, Connection _conexion) throws ApplicationException {
    int resultado;
    try {
        Statement s = _conexion.createStatement();
        resultado = s.executeUpdate("UPDATE partido SET votos=votos+1 WHERE siglas='" + _partido.getSiglas() + "'");
        if (resultado != 1) {
            throw new ApplicationException("NO SE HA SUMADO EL VOTO AL PARTIDO", 0, "DAOperaciones.registrarVoto()");
        }
    } catch (SQLException EX) {
        String mensaje = EX.getMessage();
        int codigo = EX.getErrorCode();
        throw new ApplicationException(mensaje, codigo, "ERROR EN LA SENTENCIA SQL");
    }
}
```

RESULTADO ELECTOS

En esta parte tenemos que tener en cuenta que los candidatos van a ser 4, que es un valor que tenemos en la clase Parámetros. Debemos recoger la sesión, que se supone que se debe crear al principio, yo la he creado en esta fase porque antes lo había hecho muy básico.

Creamos un arrayList, al cual le vamos a pasar el número de escaños. Vamos a utilizar la clase Electo, que es la que hemos creado al principio de todo para esta fase. Y después, el resultado se lo pasamos a una variable de sesión.

```
try (PrintWriter out = response.getWriter()) {
    //SINGLETON
    Conexion miConexion = Conexion.Connect();
    Connection Conexion = (Connection) miConexion.getConexion();

    HttpSession session=request.getSession(true);
    Parametros objParametros=(Parametros)session.getAttribute("objParametros");

    int numeroEscaños=objParametros.getCandidatos();
    ArrayList<Electo> ArrayElectos=new DAOperaciones().getEscaños(numeroEscaños,Conexion);

    session.setAttribute("ArrayElectos", ArrayElectos);
    response.sendRedirect("../VISTAS/ResultadoElectos.jsp");

} catch (ApplicationException AE) {
    response.sendRedirect("../VISTAS/Solucion.jsp");
} catch (ClassNotFoundException | SQLException ex) {
    response.sendRedirect("../VISTAS/Solucion.jsp");
}
```

El método es muy complicado, ya que jugamos con ArrayList, añadiendo y quitando.

```
public ArrayList<Electo> getEscalaños(int _numescaños, Connection _Conexion) throws ApplicationException {

    ArrayList<Partido> ArrayPartidos = new ArrayList<>();
    ArrayList<Electo> ArrayElecto = new ArrayList<>();
    String sql = "SELECT * FROM partido";

    try {
        Statement s = _Conexion.createStatement();
        ResultSet rs = s.executeQuery(sql);
        while (rs.next()) {
            Integer id = rs.getInt("id");
            String nombrepartido = rs.getString("nombrepartido");
            String siglas = rs.getString("siglas");
            String logo = rs.getString("logo");
            Integer votos = rs.getInt("votos");
            Partido objPartido = new Partido(id, nombrepartido, siglas, logo, votos);
            ArrayPartidos.add(objPartido);
        }
        //ORDENAMOS ARRAYLIST PARTIDOS
        ArrayPartidos.sort(Collections.reverseOrder());

        //CREAMOS ARRAYLIST ESCAÑOS
        ArrayList<Escalaños> ArrayEscalaños = new ArrayList<>();
        //RECORREMOS
        for (int i = 0; i < _numescaños; i++) {
            int id_partido = ArrayPartidos.get(0).getId();
            String siglas = ArrayPartidos.get(0).getSiglas();
            String logo = ArrayPartidos.get(0).getLogo();
            //LOS VOTOS SE LOS INCREMENTAMOS DIRECTAMENTE AL OBJETO
            Escalaños objEscalaño = new Escalaños(id_partido, siglas, logo, 1);

            int indice = ArrayEscalaños.indexOf(objEscalaño);
            if (indice == -1) {
                ArrayEscalaños.add(objEscalaño);
            } else {
                objEscalaño = ArrayEscalaños.get(indice);
                objEscalaño.setNumescaños(objEscalaño.getNumescaños() + 1);
                ArrayEscalaños.set(indice, objEscalaño);
            }
            ArrayPartidos.get(0).setVotos(ArrayPartidos.get(0).getVotos() / 2);
            //VOLVEMOS A ORDENAR EL ARRAY
            ArrayPartidos.sort(Collections.reverseOrder());
        }
        for (Escalaños objEscalaño : ArrayEscalaños) {
            String sql2 = "SELECT c.id ,c.nombre_apellidos, p.siglas, p.logo FROM candidatos c, partido p WHERE p.id=c.id_partido AND";
            Statement sr = _Conexion.createStatement();
            ResultSet rs2 = sr.executeQuery(sql2);

            while (rs2.next()) {
                ArrayElecto.add(
                    new Electo(
                        rs2.getInt("id"),
                        rs2.getString("siglas"),
                        rs2.getString("logo"),
                        rs2.getString("nombre_apellidos")
                    )
                );
            }
        }
    } catch (SQLException SQLE) {
```

Con esto obtendríamos el resultado de los 4 candidatos.

La funcionalidad de la aplicación estaría hecha, ahora hay que organizarla para que, dependiendo del estado del escrutinio, el rol del usuario y si ha votado o no, pueda elegir o no, ciertas cosas.

Yo lo he hecho de esta forma, que es la más fácil que me ha parecido. Desde el menú, voy comprobando todos los estados y ya muestro, las opciones que puede hacer.

```
<div id="menu">
    <% Parametros objParametros = (Parametros) session.getAttribute("objParametros");
       Votante objVotante = (Votante) session.getAttribute("objVotante");
       if (objVotante.getRol().equals("Votante") && objVotante.getVotado().equals("NV") && objParametros.getEstadoEscrutinio().equals("A")) {
    <ul id="nav">
        <li class="tutorials"><a href="BajaVotante.jsp">Baja votante</a></li>
        <li class="about"><a href="ModificarVotante.jsp">Modificar votante</a></li>
        <li class="news"><a href=" ../index.jsp">Cerrar Sesión</a></li>
    </ul>
    <% } else if (objVotante.getRol().equals("Votante") && objVotante.getVotado().equals("V") && objParametros.getEstadoEscrutinio().equals("A")) {
    <div id="estado">
        <p>EL ESCRUTINIO SE ENCUENTRA CERRADO, NO SE PUEDE REALIZAR NINGUNA OPCIÓN</p>
        <p><a id="atras" href=" ../index.jsp">CERRAR SESIÓN</a></p>
    </div>
    <% } else if (objVotante.getRol().equals("Administrador") && objVotante.getVotado().equals("V") && objParametros.getEstadoEscrutinio().equals("A")) {
    <ul id="nav">
        <li class="news"><a href=" ../ControladorListarCenso">Listar Censo</a></li>
        <li class="home"><a href="Escrutinio.jsp">Escrutinio</a></li>
        <li class="tutorials"><a href=" ../index.jsp">Cerrar Sesión</a></li>
    </ul>
    <% } else if (objVotante.getRol().equals("Administrador") && objVotante.getVotado().equals("NV") && objParametros.getEstadoEscrutinio().equals("A")) {
    <ul id="nav">
        <li class="tutorials"><a href="BajaVotante.jsp">Baja votante</a></li>
        <li class="about"><a href="ModificarVotante.jsp">Modificar votante</a></li>
        <li class="news"><a href=" ../ControladorListarCenso">Listar Censo</a></li>
        <li class="home"><a href="Escrutinio.jsp">Escrutinio</a></li>
        <li class="tutorials"><a href=" ../index.jsp">Cerrar Sesión</a></li>
    </ul>
```