

Universidad Simón Bolívar  
Departamento de Computación  
Taller de Desarrollo de Software

Manual del Sistema  
Sistema de Consultas difusas de la Encuesta de Opinión Estudiantil

Alejandro Ballesta #08-10090  
Jesus Rondon #0437545  
Andres Cordova #0538050

## Indice General

1. Introducción	3
2. Modelo	4
3. Controlador	6
4. Vista	7

## Seccion 1

### Introduccion

El propósito de este manual es proveer la mayor cantidad de información posible sobre cómo se hicieron las aplicaciones. Primero describiremos cómo funciona la aplicación con el manejador de bases de datos difusas PostgreSQLf, y luego explicaremos cómo utilizar la libreria SQLFi en su aplicación. Como la capa de la vista del sistema es la misma para ambos, la explicaremos una sola vez al final.

Los tres elementos más importantes del proyecto son, los archivos dentro de la carpeta Vista, el controlador que es el archivo consultas.Configuracion.java, y el modelo que es distinto dependiendo si se utiliza SQLFi o PostgreSQLf. Si se utiliza el primero, la librería se encarga del manejo de la base de datos a través de las funciones que provee. Cuando se utiliza el último, se utiliza un manejador de base de datos que creamos llamado dbms.DatabaseManager.java

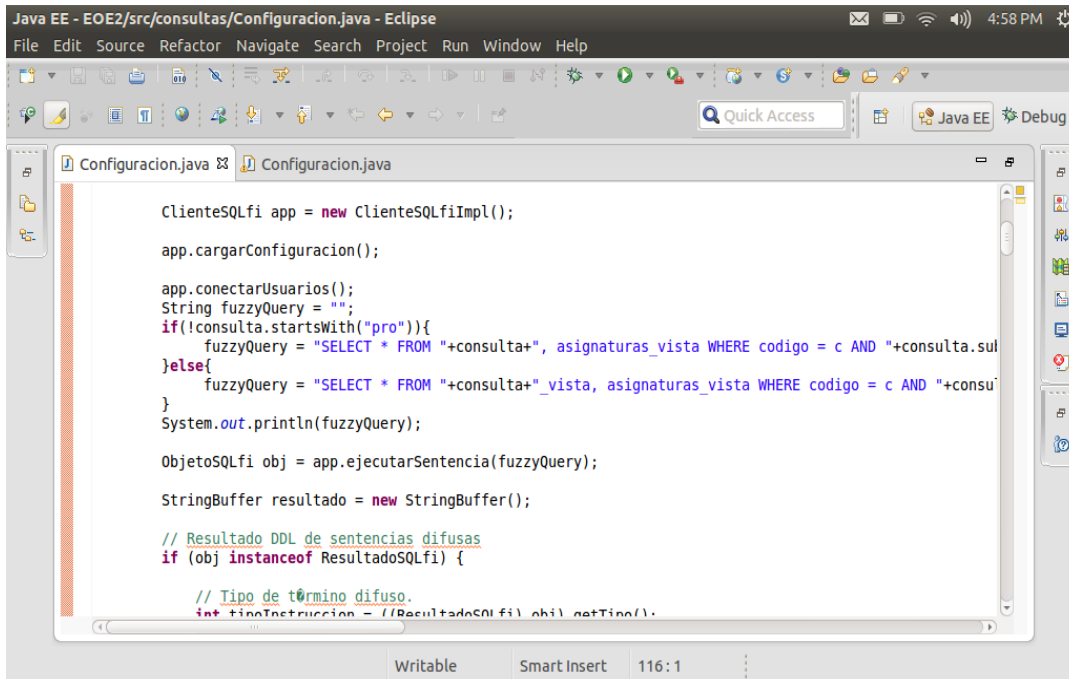
Esperamos que este documento le permita tener un entendimiento integral del sistema.

## Seccion 2

### Modelo

#### En SQLfi

La aplicación en SQLFi ejecuta funciones que se encargan del manejo de la base de datos.



El método cargarConfiguracion(), carga los datos del archivo SQLfi.properties para luego utilizarlos con la función conectarUsuarios, la cual conecta los usuarios a la base de datos. Después se crea la consulta como un string con los parámetros necesarios, y se ejecuta sobre la base de datos con la función ejecutarSentencia. Luego se obtienen los resultados y se envían a la vista.

#### En PostgreSQL

La clase dbms.DatabaseManager.java contiene las siguientes propiedades:

- instance: La instancia de un manejador de base de datos
- conexion: Conexión con la base de datos.

Sus métodos más importantes son los siguientes:

**conectarse():** Se conecta con la base de datos. Este método es muy importante, pues al llamado a la función DriverManager.getConnection() incluye la información a modificar si desea utilizar una base de datos con distinto nombre. Observe:

```
public static boolean conectarse(){
    try{
        Class.forName("org.postgresql.Driver");
        conexion = DriverManager.getConnection(
            "jdbc:postgresql://localhost:5432/E0EPF",
            "postgres",
            "BrownTabl3"
        );
    }
}
```

```

        return true;
    } catch (Exception e){
        System.out.println("*****aquí esta= ");
        e.printStackTrace();
    }
    return false;
}

```

EOEPF es el nombre de la base de datos que se en este caso se creó en PostgreSQL. postgres es el nombre de usuario, y BrownTabl3 es el password del usuario.

**desconectarse():** Cierra la conexión con la base de datos.

Las próximas tres funciones de consulta, crean las consultas difusas como un string, ejecutan la consulta, y luego devuelven el resultado. Crean un Array List de tipo Materia, y luego se construyen las instancias de la clase Materia con los datos de la base de datos, para luego meter las instancias en el Array List de materias. Veamos:

```

while(resultados.next()){
    Materia materia = new
Materia(resultados.getString("codigo"),
resultados.getString("nombre"),
resultados.getString("Gr.Memb."));
    materias.add(materia);
}

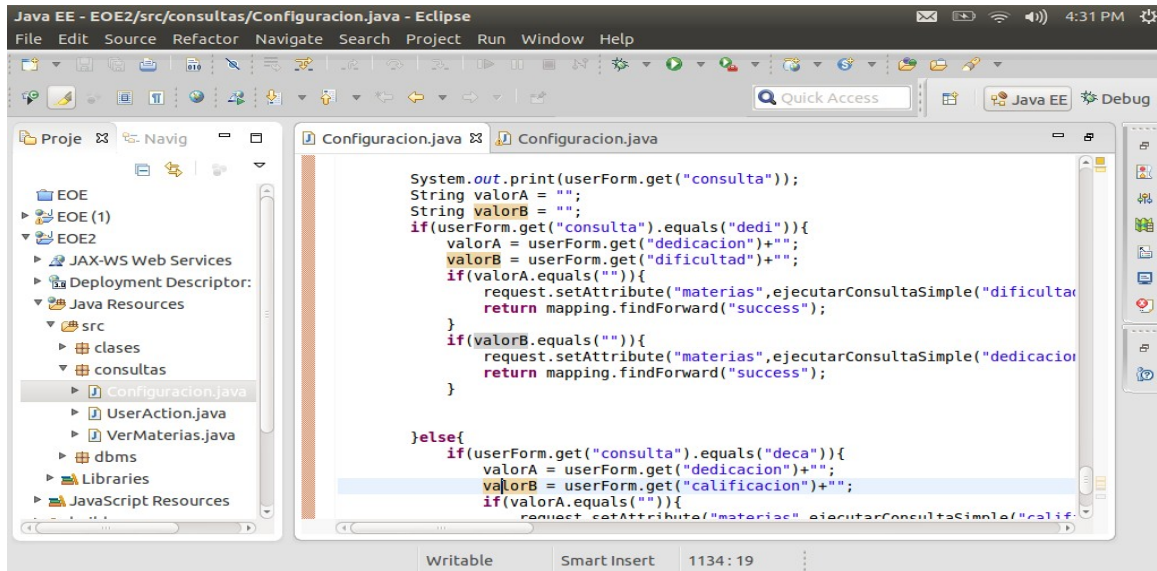
```

La clase dbms.DatabaseManager.java contiene las funciones que ejecutan las consultas a la base de datos. Estas son: consultarDedicacionDificultad, consultarDedicacionCalificacion y consultarProfesorDificultad. Las tres reciben los parámetros que necesitan (predicado difuso elegido por el usuario para hacer la consulta) y luego ejecutan la consulta en la base de datos. Después, devuelven los resultados en un objeto. De allí, se obtienen los valores de cada fila de la tabla y se construye una clase Materia con esos valores. Después, el arreglo de materias se envía al controlador el cual lo envía directamente a la vista.

## Sección 3

### Controlador

La clase consultas.Configuracion.java es el controlador de la aplicación. Se encarga de recibir los parámetros que el usuario introduce en la vista, y pasárselos al manejador de base datos. Luego, obtiene el resultado de la consulta ejecutada por el manejador y la envía de vuelta a la vista.



Primero, recibe un valor escondido de la vista que le indica el tipo de consulta (dedi, deca o prodi: dedicación/dificultad, dedicación/calificación, profesor/dificultad). Luego, la aplicación llama a la función ejecutarConsultaSimple, la cual llama a la función correspondiente del manejador de bases de datos. En SQLFi se utilizan las funciones de esta libreria para ejecutar las consultas mientras que en PostgreSQL se utilizan los métodos de la clase dbms.DatabaseManager.java.

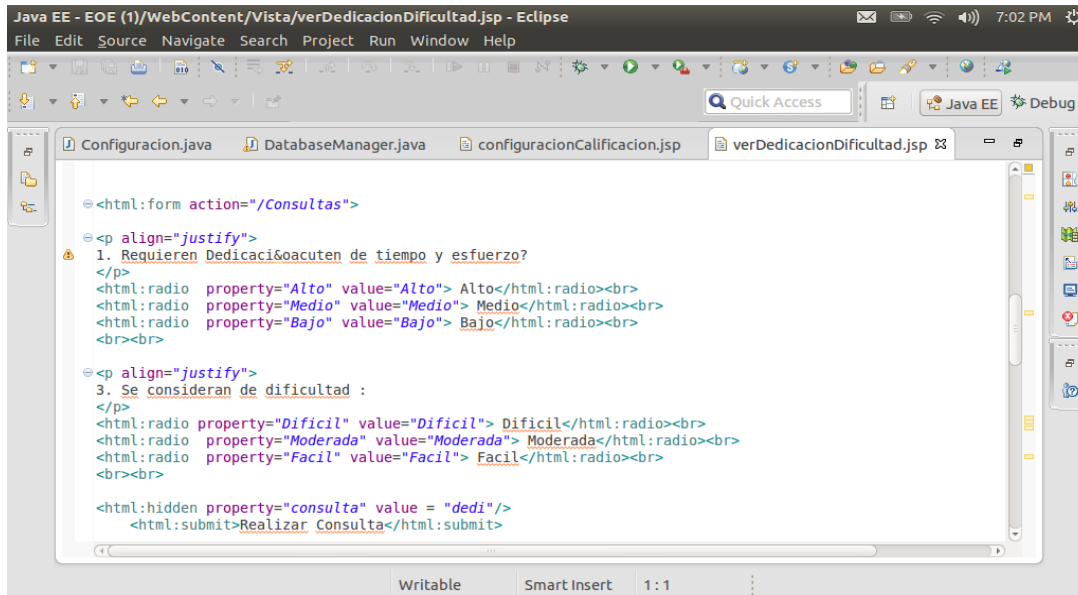
Si se encuentra en la sección de configuración, se verifica si ya existe el predicado, y si existe se actualiza el predicado. De lo contrario, se crea un predicado nuevo con el prefijo del nombre del usuario. Luego la información se pasa a la vista.

## Sección 4

### Vista

El funcionamiento de la vista es el mismo para ambos módulos (enviar y recibir información), por lo tanto, veremos un sólo ejemplo representativo.

Veamos un ejemplo de un archivo de extensión jsp que se encarga de enviar información al controlador. En esta imagen vemos, la parte más importante de la vista verDedicacionDificultad.jsp:



```
Java EE - EOE (1)/WebContent/Vista/verDedicacionDificultad.jsp - Eclipse
File Edit Source Navigate Search Project Run Window Help

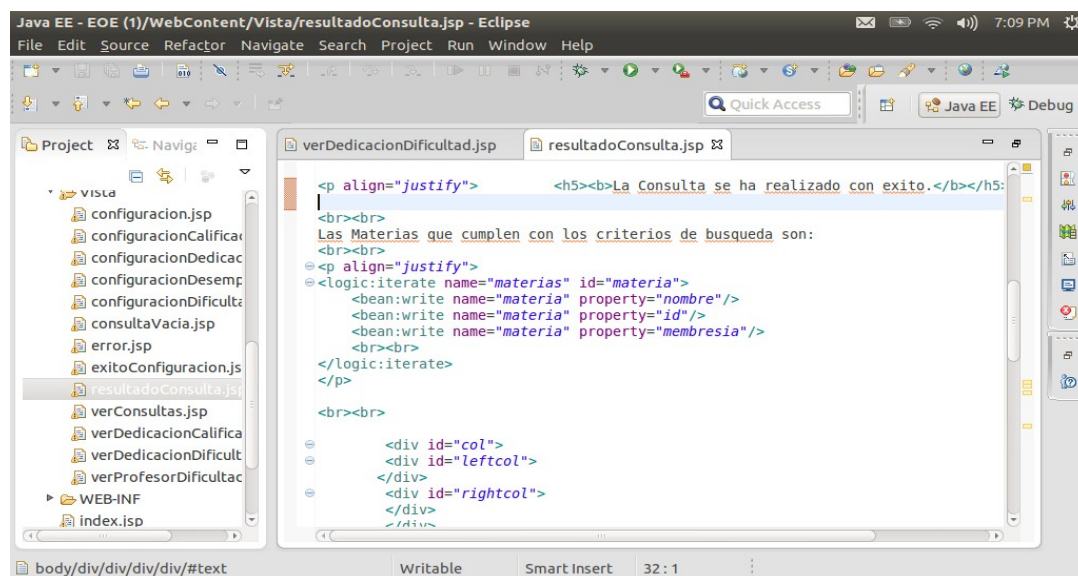
<html:form action="/Consultas">

<p align="justify">
1. Requieren Dedicación o acuten de tiempo y esfuerzo?
</p>
<html:radio property="Alto" value="Alto"> Alto</html:radio><br>
<html:radio property="Medio" value="Medio"> Medio</html:radio><br>
<html:radio property="Bajo" value="Bajo"> Bajo</html:radio><br>
<br><br>

<p align="justify">
3. Se consideran de dificultad :
</p>
<html:radio property="Dificil" value="Dificil"> Dificil</html:radio><br>
<html:radio property="Moderada" value="Moderada"> Moderada</html:radio><br>
<html:radio property="Facil" value="Facil"> Facil</html:radio><br>
<br><br>

<html:hidden property="consulta" value = "dedi"/>
<html:submit>Realizar Consulta</html:submit>
```

Se utilizan radio buttons, y a las elecciones (property) se le asigna el valor (value) respectivo al ser elegidas por el usuario. Luego se envía esta información junto con el valor escondido al controlador. El valor escondido (hidden) le indica al controlador que la consulta es de tipo Dedicación (de)/Dificultad (di). La vista que recibe los datos de las consultas y las muestra al usuario es el archivo resultadoConsulta.jsp. La parte más importante es el ciclo (etiqueta iterate) que imprime las materias.



```
Java EE - EOE (1)/WebContent/Vista/resultadoConsulta.jsp - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help

<p align="justify"> <h5><b>La Consulta se ha realizado con éxito.</b></h5>
<br><br>
Las Materias que cumplen con los criterios de busqueda son:
<br><br>
<p align="justify">
<logic:iterate name="materias" id="materia">
  <bean:write name="materia" property="nombre"/>
  <bean:write name="materia" property="id"/>
  <bean:write name="materia" property="membresia"/>
</logic:iterate>
</p>
<br><br>

<div id="col">
  <div id="leftcol">
  </div>
  <div id="rightcol">
  </div>
</div>
```