

Entrega Final

Tema: Planificación de consultas ciudadanas

Nombres: Sergio Arriagada Gallo

Miguel Guerrero Ruiz

Sebastián Moyano Riveros

Diego Duran

1.1 Análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.

Se ha pedido realizar un sistema de consultas ciudadanas para ver la opinión de los ciudadanos, para esto hemos diseñado un sistema de consultas que se conformara de los siguientes datos: Consultas(Aquí van consultas sobre algun tema en especifico, por ejemplo: Una consulta sobre el aborto y como funciona), respuestas(Que se conforman por likes o dislikes).

Datos:

Consultas: ID de consulta, título de consulta,título tema, descripción de la consulta, y maneja un conjunto de respuestas

Respuestas: ID de respuesta, contadores a favores y en contra

Las principales funcionalidades serán:

-Clasificar las consultas por temas

-Añadir y eliminar consultas en sus respectivos temas

-Permitir a los ciudadanos expresar su opinión respecto a la consulta que seleccionen

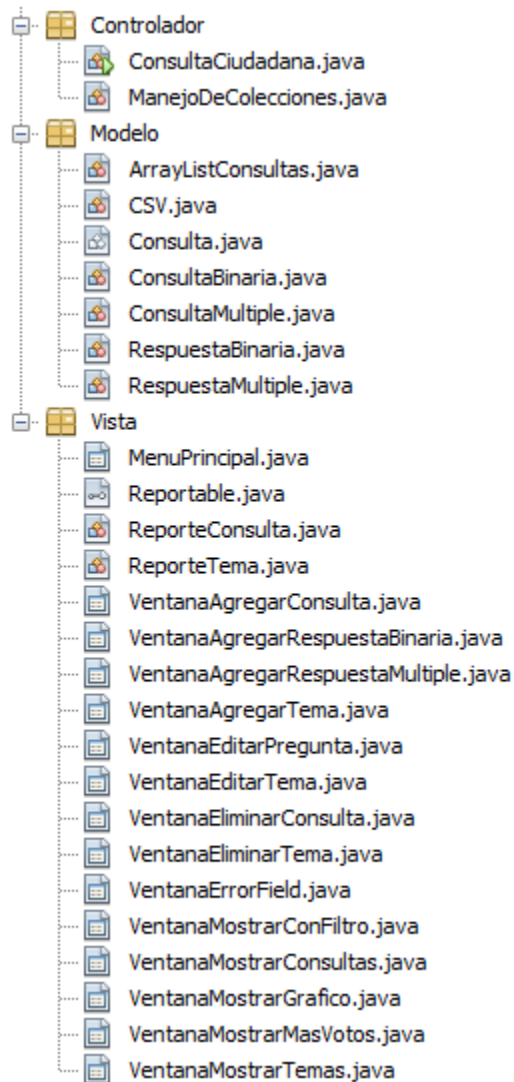
-Permitir a los ciudadanos añadir consultas

-Permitir mostrar una gráfica con los respectivas respuestas de una consulta

Y como extra:

- Leer datos de un archivo Csv, en este caso las consultas

1.2 Diseño conceptual de clases del Dominio y su código en Java



ConsultaCiudadana: Esta clase es “main” de nuestro proyecto es la encargada de generar la clase ManejoDeColecciones, y de llamar al menú principal.

ManejoDeColecciones: Como su nombre lo dice esta clase es la encargada de manejar las colecciones valga la redundancia, es decir, la encargada de buscar, operar, eliminar sobre todas las colecciones que posee el sistema.

ArrayListConsultas: Esta clase contiene en su interior un arrayList que almacena en su interior datos de tipo Consulta.

CSV: Esta clase es la encargada de realizar la lectura de archivos de nuestro programa.

Consulta: Esta clase abstracta define el tipo de dato consulta, el cual es la base del sistema y de donde bifurca a 2 tipo de consulta, pudiendo ser una consulta Binaria (que tendrá de respuesta solo likes & dislikes), o una consulta Múltiple (que almacena varios tipos de reacciones) .

Reportable: Corresponde una interface que tendrá 2 funciones, como lo son generarTxt() y generarExcel(), las cuales se utilizaran en un futuro para generar archivos txt y un excel respectivamente.

ReporteConsulta: Clase que implementa una interfaz para crear un archivo txt y un excel en la VentanaMostrarConsultas, almacenando todo aquello que se muestre en la matriz.

ReporteTema: Clase que implementa la interfaz Reportable que utilizara sus funciones para crear un archivo txt y excel.

ConsultaBinaria: Esta clase, es una subclase de Consulta. ConsultaBinaria tiene el atributo de respuesta binaria.

ConsultaMultiple: Esta clase, es una subclase de Consulta. ConsultaMultiple tiene el atributo de respuesta multiple.

RespuestaBinaria: Esta clase es la encargada de almacenar las aquellas respuestas que se conforman de Likes y dislikes, para las consultas binarias.

RespuestaMultiple: esta clase se encarga de almacenar todas aquellas respuestas que sean a través diferentes tipos de opiniones , pudiendo tomar el valor de Muy de acuerdo, de acuerdo, neutral, en contra, muy en contra.

MenuPrincipal: Es un JFrame. Es la primera ventana que se encuentra al iniciar el programa, esta ventana permite el acceso a las demás ventanas, es el centro de las ventanas visuales.

Reportable: Es una interfaz, la cual otorga el rol de reportable, es decir, que la clase que la implemente, pueda generar un reporte ya sea excel o tipo txt de sus datos.

VentanaAgregarRespuestaBinaria: Es un JFrame. Esta ventana que se encarga darle las opciones al usuario de seleccionar un tema e id de una consulta para emitir una respuesta binaria.

VentanaAgregarRespuestaMultiple: Es un JFrame. Esta ventana que se encarga darle las opciones al usuario de seleccionar un tema e id de una consulta para emitir una respuesta múltiple.

VentanaAgregarConsulta: Es un JFrame. Esta ventana es la encargada de añadir una consulta a la base de datos del sistema.

VentanaAgregarTema: Es un JFrame. Esta ventana es la encargada de permitirle al usuario añadir un tema en la colección.

VentanaEditarPregunta: Es un JFrame. Esta ventana le permite al usuario editar la pregunta que seleccione de su respectiva colección.

VentanaEditarTema: Es un JFrame. Esta ventana se encarga de permitirle al usuario modificar atributos de un tema existente.

VentanaEliminarConsulta: Es un JFrame. Esta ventana permite al usuario eliminar una pregunta de la base de datos.

VentanaEliminarTema: Es un JFrame. Esta ventana permite al usuario eliminar un tema de la base de datos.

VentanaErrorField: Es un JFrame. Esta ventana muestra un pop-up mostrando un string dependiendo de la situación que lo necesite.

VentanaMostrarConFiltro: Es un JFrame. Esta ventana permite mostrar una consulta filtrada de la colección en base a su tema buscado y la id correspondiente.

VentanaMostrarConsultas: Es un JFrame. Esta ventana es la encargada de mostrar la colecciones de consultas almacenada en la base de datos.

VentanaMostrarGrafico: Es un JFrame. Esta ventana es la encargada de mostrar un gráfico con likes y dislikes, de una consulta previamente seleccionada.

VentanaMostrarMasVotos: Es un JFrame. Esta ventana es la encargada de mostrar la consulta que tenga mas votos en un tema seleccionado.

VentanaMostrarTemas: Es un JFrame. Esta ventana muestra los temas almacenados en las keys del mapa.

1.3 **Todos los atributos de todas las clases deben ser privados y poseer sus respectivos métodos de lectura y escritura (getter y setter).**

Todas las clases tiene su respectivo getters and setters exceptuando:

La clase ManejoDeColecciones: Ya que es la encargada de manejar solamente las colecciones que se utiliza en el programa.

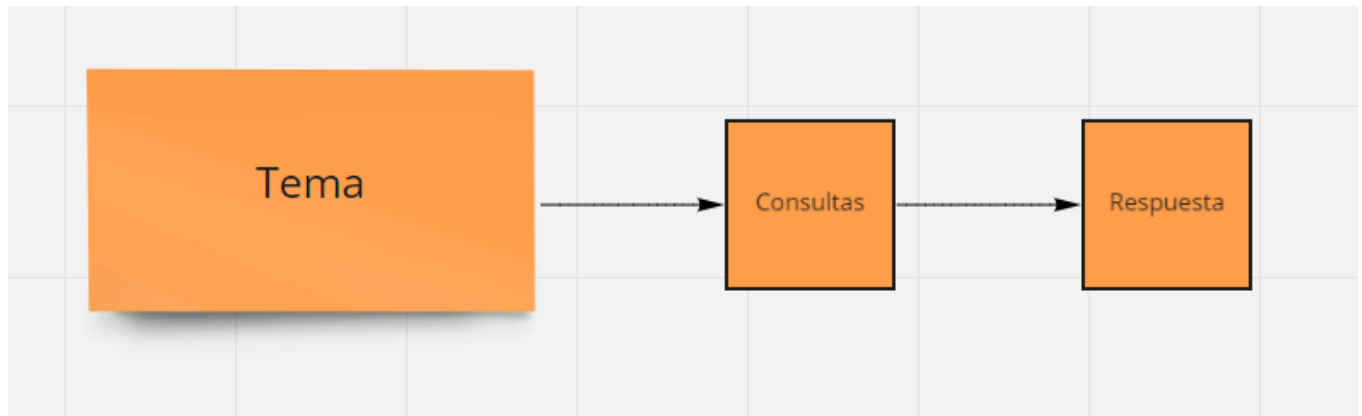
La clase ConsultaCiudadana: Es clase es el “main” encargada de generar la clase ManejoDeColecciones, y de llamar al menú principal, ósea no contiene atributos.

Y todos los JFrame debido que son solo ventanas.

1.4 **Se deben incluir datos iniciales dentro del código.**

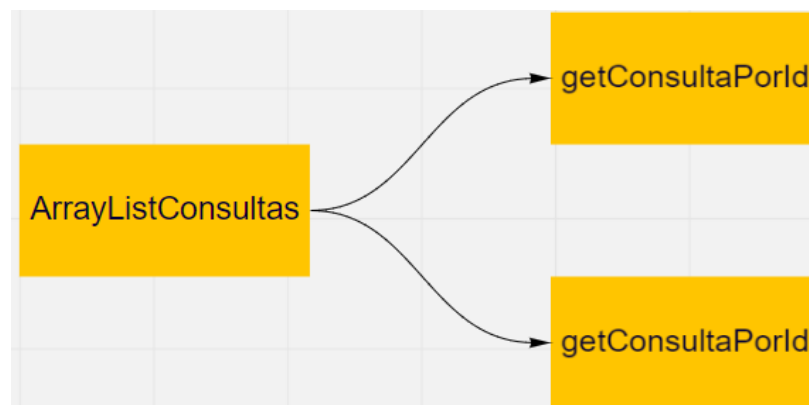
A través de un archivo csv, nuestro programa ya vendrá con algunas consultas.

2.1 Diseño conceptual y codificación de 2 (dos) niveles de anidación de colecciones de objetos.



En nuestro proyecto se utilizara un hashmap teniendo como key los temas de las consultas, el cual tendrá almacenado ArrayListConsultas, la cual almacena todas las consultas del tema en un ArrayList de Consulta, y cada consulta teniendo su respectiva respuesta.

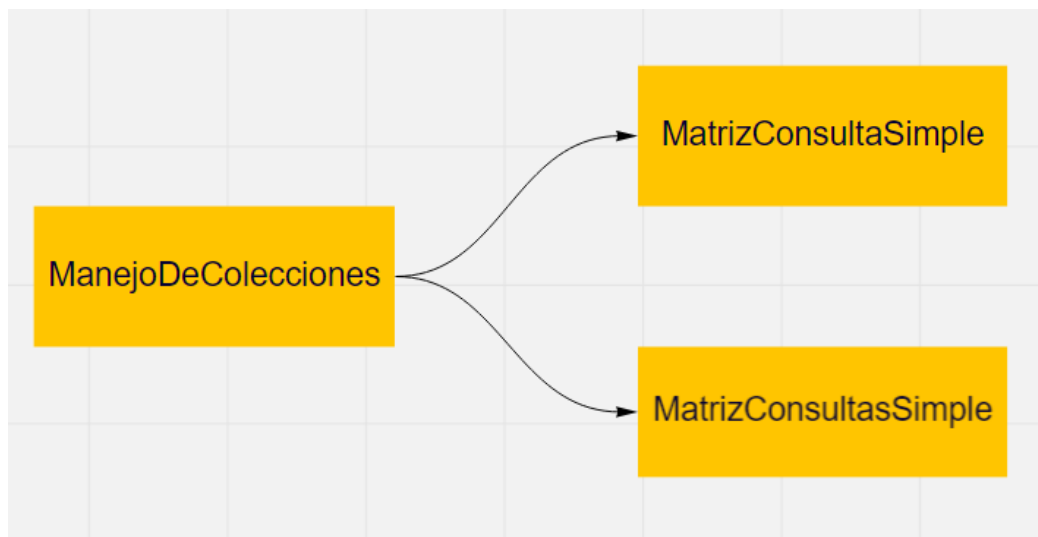
2.2 Diseño conceptual y codificación de 2 (dos) clases que utilicen sobrecarga de métodos



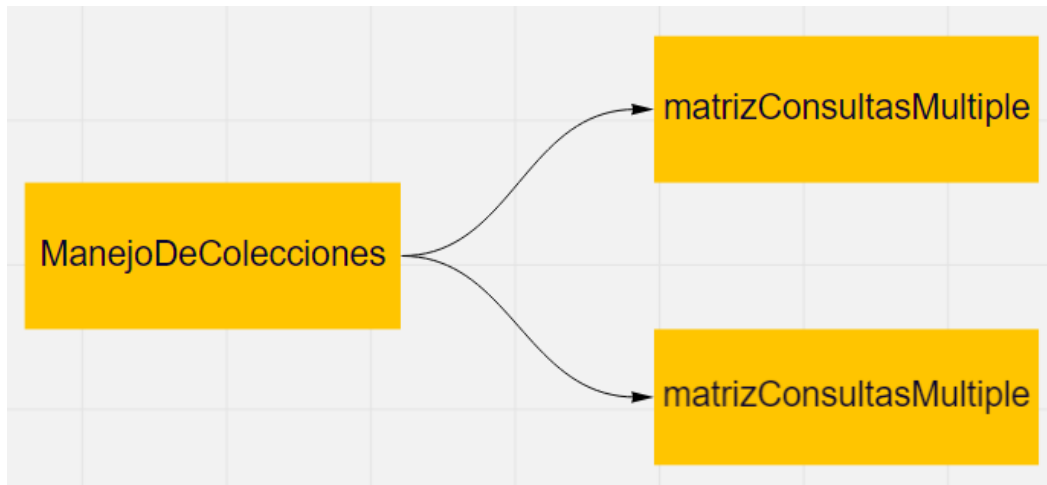
La clase ArrayListConsultas contiene una sobrecarga de métodos: getConsultaPorId, donde una recibe por parámetro un entero que es una id, y el otro recibe una string de la id buscada.



La clase `ManejoDeColecciones` contiene una sobrecarga de método: `agregarConsultas`, donde uno recibe un objeto de la clase tipo `Consulta` y lo agrega al hashmap, y el otro método recibe todo los datos requeridos para crear una nueva consulta y agregarla al hashmap.



La clase `ManejoDeColecciones` contiene una sobrecarga de método: `MatrizConsultasSimple`, donde uno recibe una string como parámetro, la cual crea una matriz para ser utilizada en generar archivos de salida, y la otra no recibe nada, y es utilizada para mostrar las consultas binarias a través de tablas.

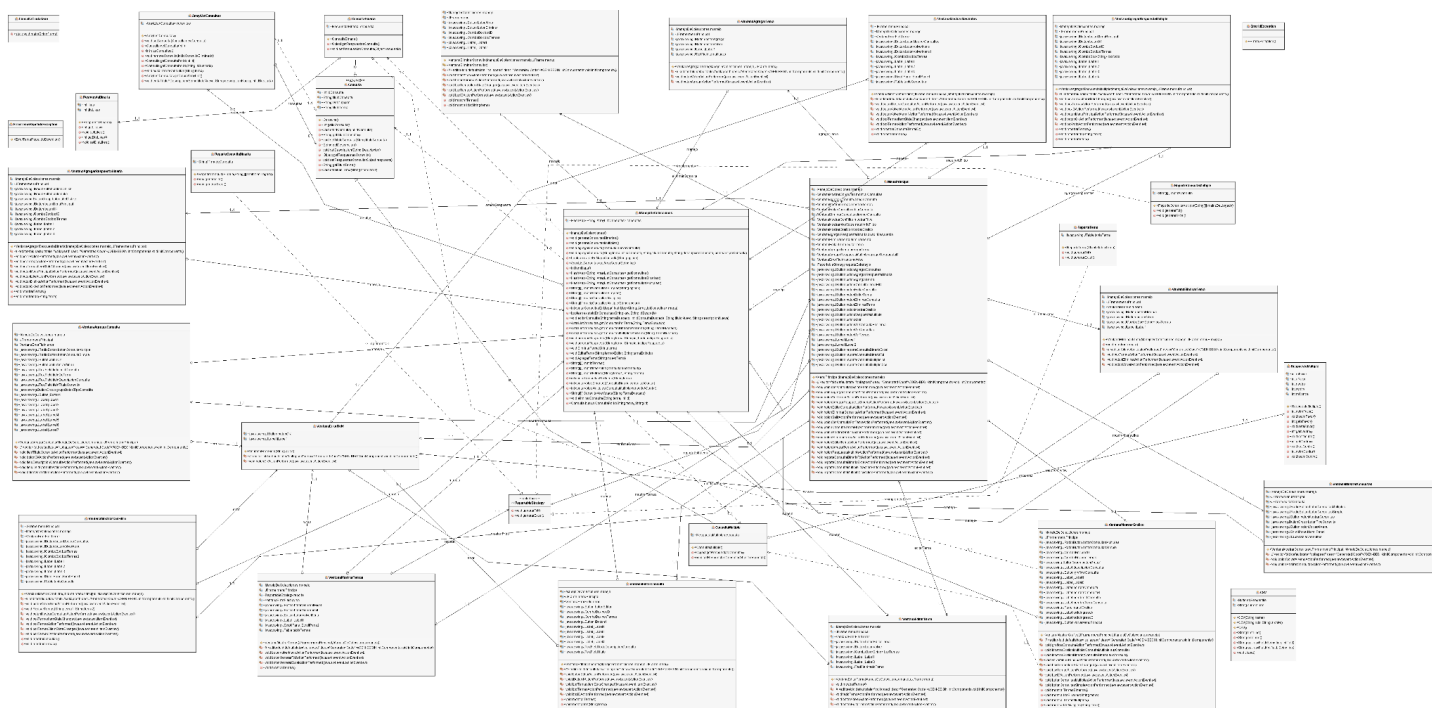


La clase ManejoDeColecciones contiene una sobrecarga de método: matrizConsultasMultiple, donde uno recibe una string como parámetro, la cual crea una matriz para ser utilizada en generar archivos de salida, y la otra no recibe nada, y es utilizada para mostrar las consultas múltiples a través de tablas.

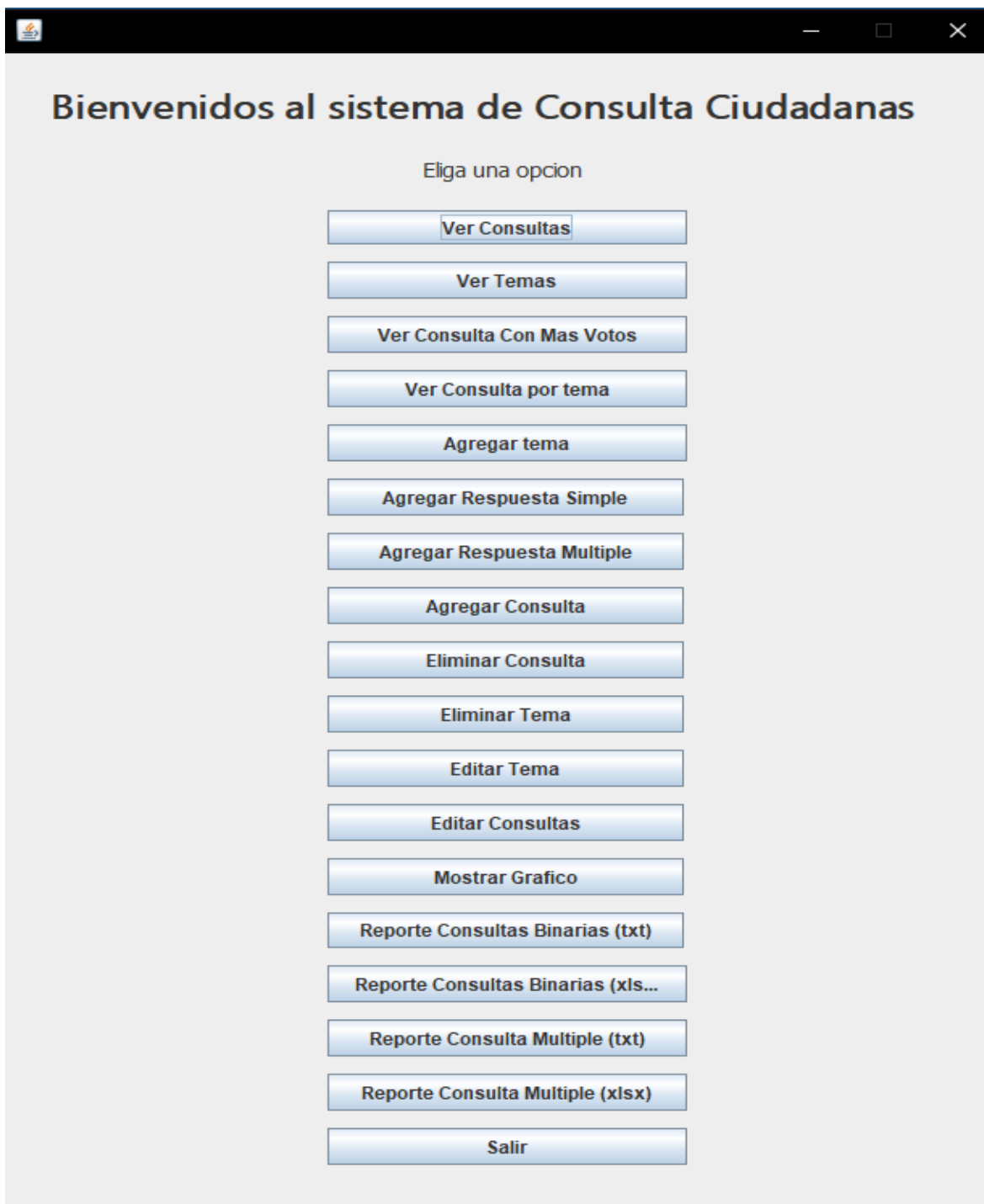
2.3 Diseño conceptual y codificación de al menos una clase mapa del JCF.

El programa posee un mapa de arraylist, y su key es el nombre de tema al cual están asociadas las consultas, de este modo al usar la key del mapa, retorna la colección arraylist de consultas del sistema.

3.1 Diseño de diagrama de clases UML



3.2 Implementación de menú del sistema para funcionalidades



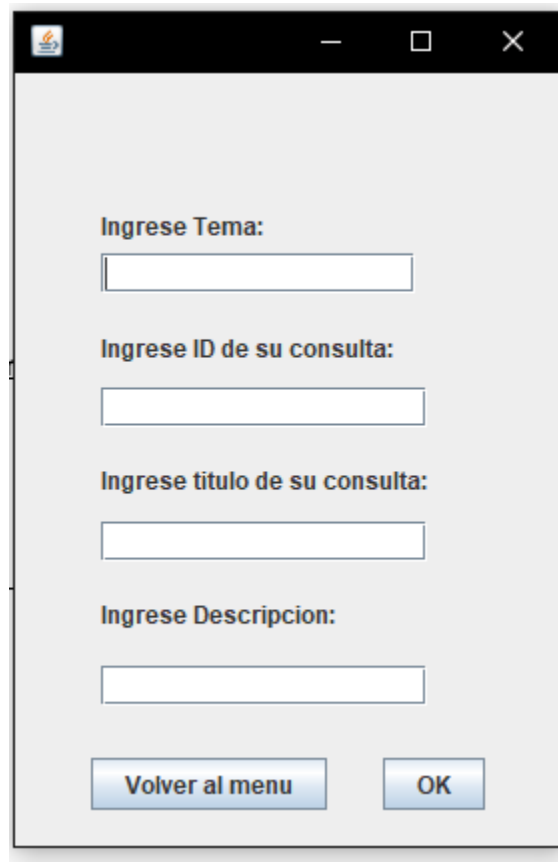
Bienvenidos al sistema de Consulta Ciudadanas

Eliga una opción

- Ver Consultas
- Ver Temas
- Ver Consulta Con Mas Votos
- Ver Consulta por tema
- Agregar tema
- Agregar Respuesta Simple
- Agregar Respuesta Multiple
- Agregar Consulta
- Eliminar Consulta
- Eliminar Tema
- Editar Tema
- Editar Consultas
- Mostrar Grafico
- Reporte Consultas Binarias (txt)
- Reporte Consultas Binarias (xls...)
- Reporte Consulta Multiple (txt)
- Reporte Consulta Multiple (xlsx)
- Salir

En nuestro caso se implementa un menú a través de un JFrame.

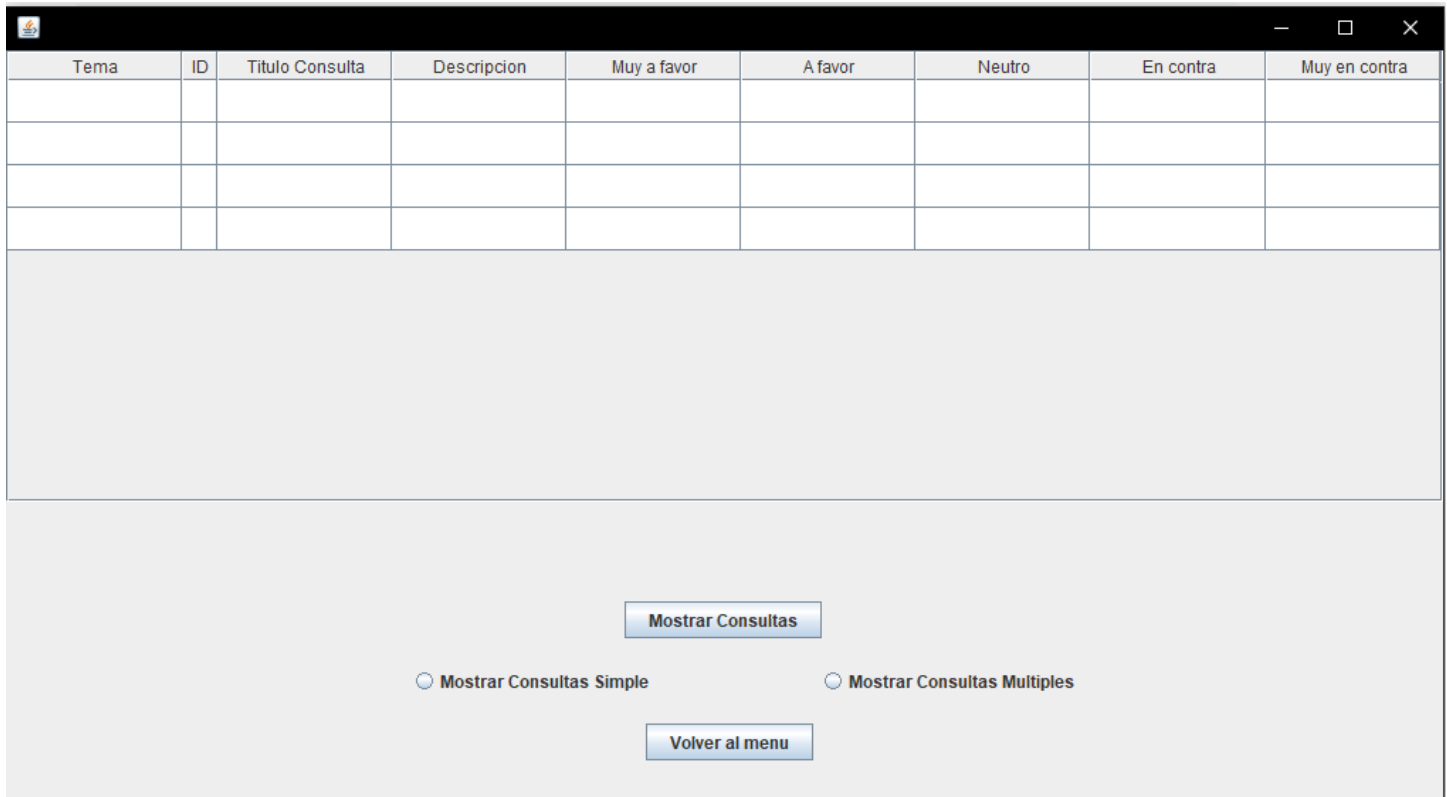
3.2.1 Funcionalidad básica (1): Inserción manual o agregar elemento



The image shows a Java Swing window titled "Inserción manual o agregar elemento". It contains four text input fields with the following labels: "Ingrese Tema:", "Ingrese ID de su consulta:", "Ingrese título de su consulta:", and "Ingrese Descripción:". At the bottom of the window, there are two buttons: "Volver al menú" and "OK".

La funcionalidad de inserción manual se produce pidiendo los campos requeridos los cuales son , Tema, el id de la consulta, el título y su descripción. En caso de ser un tema ya existente, se ingresa la consulta en el ArrayList de las consultas de ese tema, sino en el HashMap se crea una nueva ArrayList con la nueva key la cual es el tema ingresado.

3.2.2 Funcionalidad básica (2): Mostrar por pantalla listado de elementos



Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra

Mostrar Consultas

☐ Mostrar Consultas Simple ☐ Mostrar Consultas Multiples

Volver al menu

La funcionalidad de mostrar proviene directamente en la VentanaMostrarConsulta en donde se almacenan los datos anteriormente guardados, utilizando una matriz para poder mostrar los datos dependiendo el tipo de consulta requerido en una JTable, mostrando Su tema, la id, Titulo Consulta, Descripción, y los respectivos votos del su tipo de consulta.

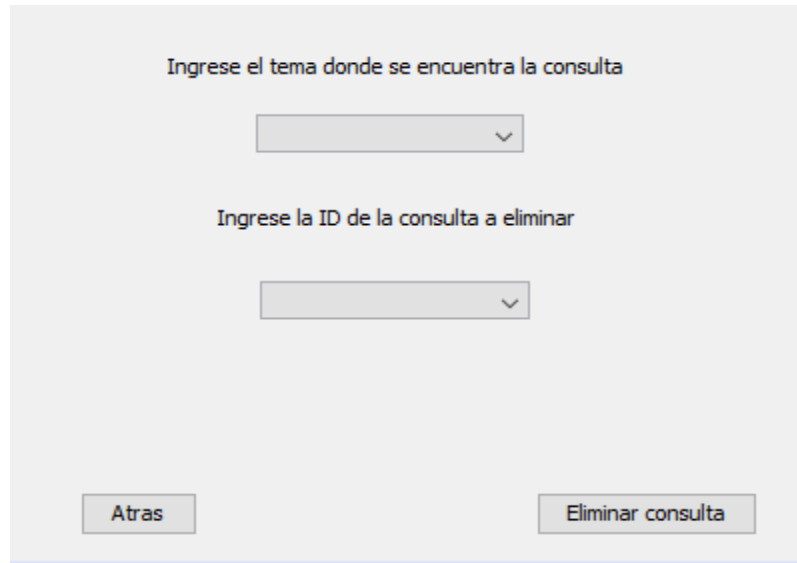
A.2 Implementación de las siguientes funcionalidades en el menú:

A.2.1 Funcionalidad básica (1): Edición/modificación de elemento

Ingrese tema donde se encuentra la consulta	Titulo
<input type="text"/>	<input type="text"/>
Ingrese la id de la consulta que busca	Descripcion
<input type="text"/>	<input type="text"/>
<input type="button" value="Atras"/>	<input type="button" value="Editar"/>

La funcionalidad de editar un elemento proviene de VentanaEditar en donde se pedirá al usuario cual es el tema y la id del elemento que desea editar para luego seleccionar cuáles serán los nuevos valores que tomará , modificandolos internamente gracias a una función proveniente al Manejo de colecciones llamada `editarConsulta(String tema, int IdConsulta, String TituloNuevo, String Descripcion)`

A.2.2 Funcionalidad básica (2): Eliminación del elemento para las 2 colecciones anidadas.



The screenshot shows a window titled 'Eliminar consulta' with a light gray background. It contains two labels: 'Ingrese el tema donde se encuentra la consulta' and 'Ingrese la ID de la consulta a eliminar'. Each label is followed by a text input field with a small downward arrow icon on the right. At the bottom, there are two buttons: 'Atras' on the left and 'Eliminar consulta' on the right.

La función de eliminar un elemento se utiliza en la `VentanaEliminarConsulta` que tiene como función el eliminar una consulta, la cual es seleccionada en el campo de tema y el de id que se desea eliminar. Luego para hacer efectiva la eliminación se debe utilizar el botón eliminar para utilizar la función `removeConsulta` que se obtendrá de la clase `ArrayListConsultas`.

A.3 Se debe generar un reporte en archivo txt/csv que considere mostrar datos de las 2 colecciones anidadas



Dentro del menu general se realizan la generación de txt y excel.

A.5 Todas las funcionalidades pueden ser implementadas mediante consola.

Todas las funcionalidades están implementadas a través de una interfaz gráfica.

EP4.1. Se deben incluir al menos 2 funcionalidades propias que sean de utilidad para el negocio específicamente:

Seleccionar un objeto por criterio: En el menú principal se encuentra la opción de “Mostrar consulta con más votos”, la cual le da opción al usuario de seleccionar un tema y mostrar la consulta que tenga más votos en ese tema:

Ingrese filtros:

Ingrese el tema para buscar:

Tema

Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra

Mostrar Consultas

Volver menu

Subconjunto filtrado por criterio:

En el menú principal se encuentra la opción de “Ver Consulta por tema” la cual permite al usuario seleccionar una colección de consultas las cuales se filtran por dos temas distintos.

Ingrese filtros:

Tema 1

Mostrar Consultas

Tema 2

Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra	Likes	Dislikes

Volver menu

EP4.2. Diseño y codificación de 2 (dos) clases que utilicen sobreescritura de métodos.

Hay dos clases que utilizan sobreescritura de métodos y son ConsultaBinaria y ConsultaMultiple, estas clases sobreescriben los métodos de la clase abstracta Consulta.

Los métodos sobreescritos son:

getRespuestasConsulta y setRespuestasConsulta.

Donde a través de estos métodos se pueden setear y obtener las respuestas de cada tipo de consulta.

EP4.3. Diseño y codificación de 1 (una) clase abstracta que sea padre de al menos 2 (dos) clases. La clase abstracta debe ser utilizada por alguna otra clase (contexto)

Se implementó la clase abstracta "Consulta" la cual tiene dos clases hijas que son "ConsultaBinaria" y "ConsultaMultiple".

Esta clase hereda los atributos idConsulta, tituloConsulta, Descripcion, tituloTema. Y además esta clase padre es utilizada en la clase "ArraylistConsulta", en donde es parte de una Arraylist de consultas(contexto).

EP4.4. Diseño y codificación de 1 (una) interfaz que sea implementada por al menos 2 (dos) clases. La interfaz debe ser utilizada por alguna otra clase (contexto)

Se implementa la interfaz "Reportable" la cual tiene dos métodos "generarTxt" y "generarExcel", permitiendo generar reportes en archivos txt y xlsx.

Además las clases "ReporteConsulta" y "ReporteTema" son las que implementan dicha interfaz, permitiendo generar un reporte de las consultas seleccionadas y temas respectivamente (Implementado los métodos de la interfaz). Dichas clases son utilizadas por las siguientes ventanas "VentanaMostrarConsultas" y "VentanaMostrarTemas"

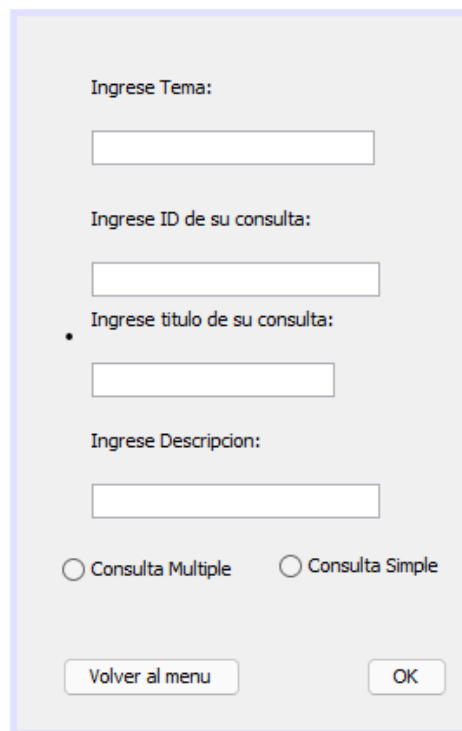
Extra:

-Se genera documentación a través de Javadoc

EPB.2. Se deben implementar al menos 3 ventanas gráficas (GUIs en AWTo SWING): 1 ventana de menú, 1 ventana de agregar elemento y 1 ventana de listar elementos

1.MostrarPrincipal: se encarga de mostrar el menú principal en donde se podrá acceder a diferentes funcionalidades del programa

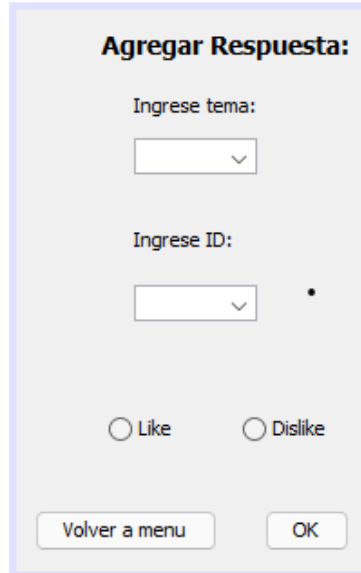
2.VentanaAgregarConsulta: se encarga de agregar un consulta, consultando por el tema de la nueva consulta, el ID que tomará, el título de esta y su descripción. Por último debe seleccionar que tipo de consulta será (Si desea que tenga Likes o Dislikes, o Reacciones).



The image shows a screenshot of a Java Swing dialog box titled 'VentanaAgregarConsulta'. The dialog box has a light gray background and a blue border. It contains the following elements:

- A label 'Ingrese Tema:' followed by a text input field.
- A label 'Ingrese ID de su consulta:' followed by a text input field.
- A label 'Ingrese titulo de su consulta:' followed by a text input field.
- A label 'Ingrese Descripcion:' followed by a text input field.
- Two radio buttons at the bottom: 'Consulta Multiple' and 'Consulta Simple'.
- Two buttons at the bottom: 'Volver al menu' and 'OK'.

3.VentanaAgregarRespuestaBinaria: el objetivo de esta ventana es la de agregar una respuesta a una consulta simple, teniendo la opción de darle un like o un dislike.



Agregar Respuesta:

Ingrese tema:

Ingrese ID:

☐ Like ☐ Dislike

4.VentanaAgregarRespuestaMultiple: el objetivo de esta ventana es la de poder agregar una respuesta a una consulta Multiple, teniendo la opción de escoger si es que se esta Muy a favor, a favor, neutro, en contra o Muy en contra.



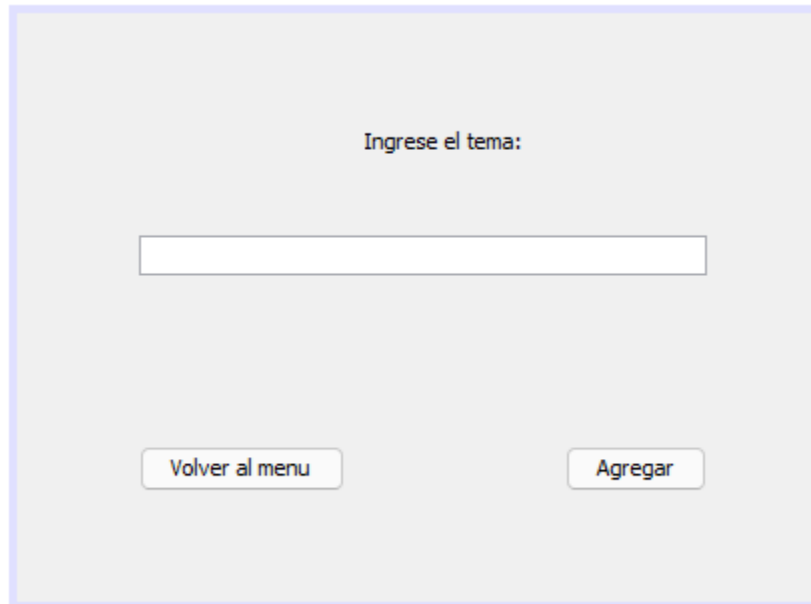
Agregar Respuesta:

Ingrese tema:

Ingrese ID:

Ingrese Voto:

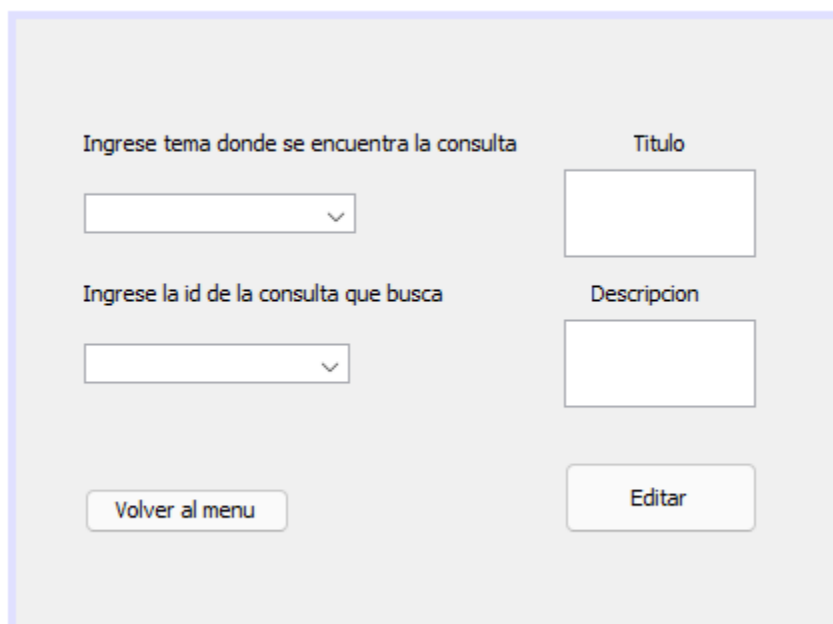
5.VentanaAgregarTema: ventana que tiene como objetivo agregar simplemente un tema sin consultas.



Ingresa el tema:

Volver al menu Agregar

6.VentanaEditarPregunta: esta ventana se encargará de modificar una consulta , teniendo la capacidad de cambiar tanto su titulo como su Descripción.



Ingresa tema donde se encuentra la consulta Titulo

Ingresa la id de la consulta que busca Descripción

Volver al menu Editar

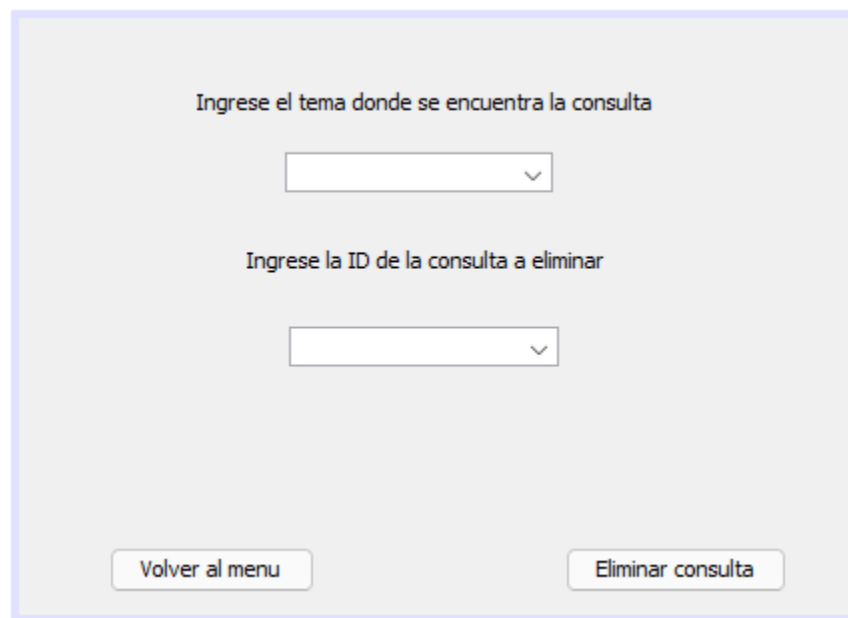
7.VentanaEditarTema: ventana que tiene como funcion el editar el nombre de un tema.



Ingrese el tema que desea editar:

Tema:

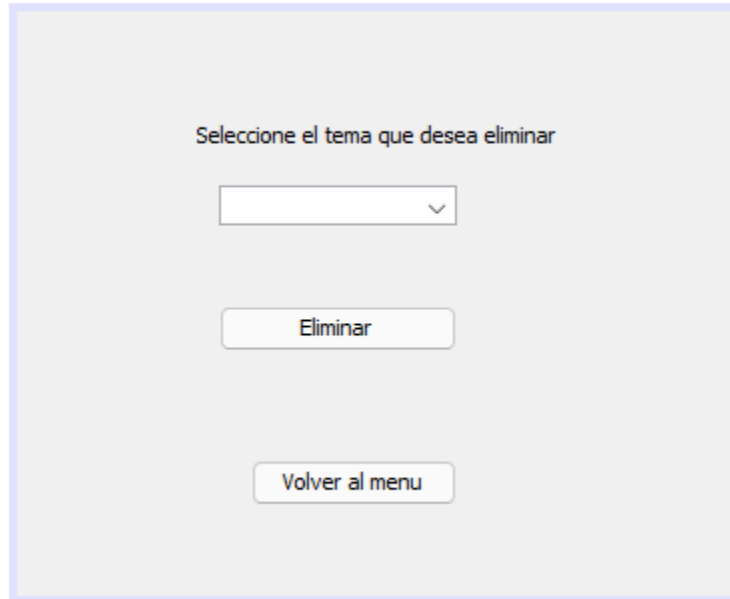
8.VentanaEliminarConsulta: se encargara de eliminar una consulta de un tema.



Ingrese el tema donde se encuentra la consulta

Ingrese la ID de la consulta a eliminar

9.VentanaEliminarTema: ventana que se encargara de eliminar un tema con todas las consultas almacenadas en el.



10.VentanaErrorField: esta ventana se utilizara siempre y cuando el usuario haya cometido un error o en casos especificos cuando se generar archivos del programa (como un archivo TXT o Excel). Este tendra un mensaje personalizado por cada tipo de error, modificando el label que exisiste en el.



•

Ingrese filtros:

Tema 1

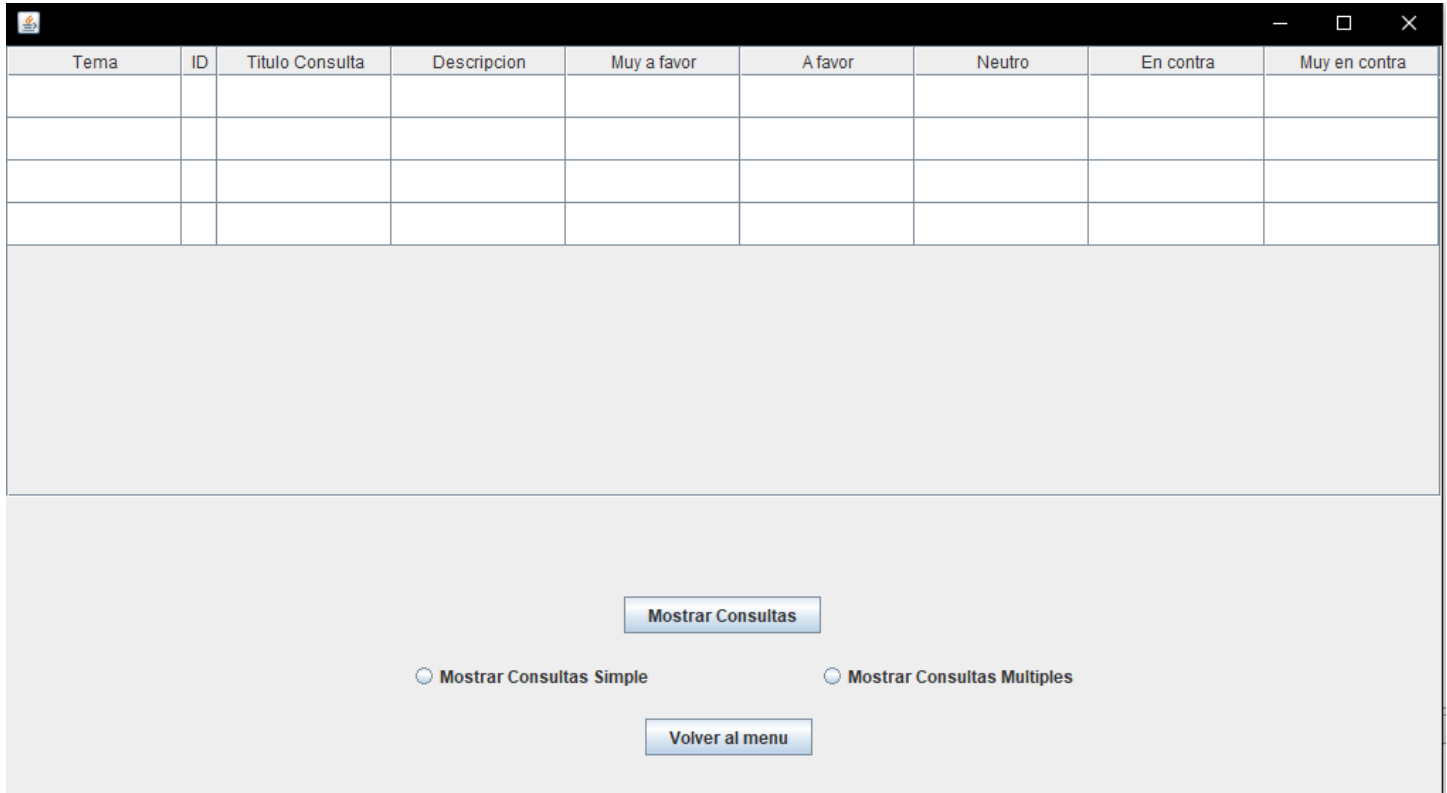
Mostrar Consultas

Tema 2

Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra	Likes	Dislikes

Volver menu

12. VentanaMostrarConsultas: esta ventana se encargara de mostrar todas las consultas almacenadas en el programa, dependiendo si son consultas simples o consultas multiples.



Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra

☐ Mostrar Consultas Simple ☐ Mostrar Consultas Multiples

13. VentanaMostrarGrafico: el objetivo de esta ventana es mostrar un grafico que contabilice y compare los votos de una consulta en especifico seleccionada por el usuario.

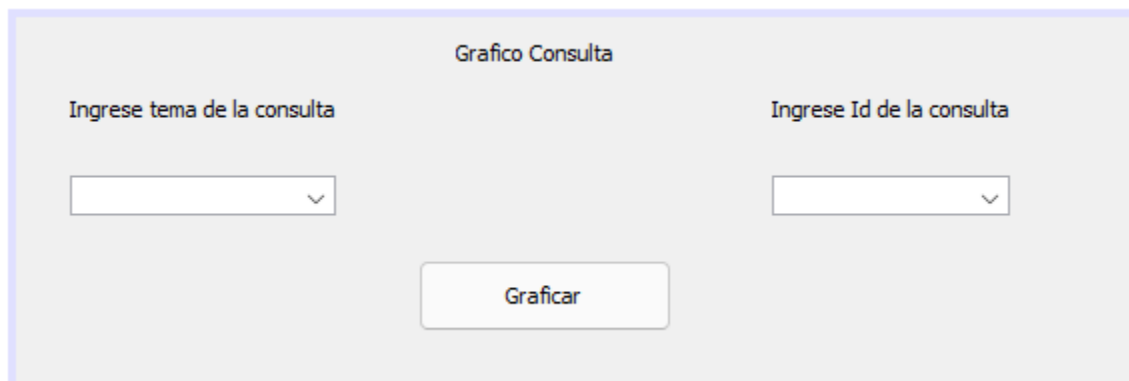


Grafico Consulta

Ingrese tema de la consulta

Ingrese Id de la consulta

14.VentanaMostrarMasVotos: se encarga de mostrar aquella consulta de un tema especifico que tenga mas votos.

Ingrese filtros:
Ingrese el tema para buscar:
Tema

Tema	ID	Titulo Consulta	Descripcion	Muy a favor	A favor	Neutro	En contra	Muy en contra

Mostrar Consultas

Volver menu

15.VentanaMostrarTemas: se encargara todos los temas manejados en el programa.

Lista de todos los temas:

Temas

Generar txt Generar Excel

Volver al menu

EF.2: Implementar el manejo de excepciones capturando errores potenciales específicos mediante Try-catch

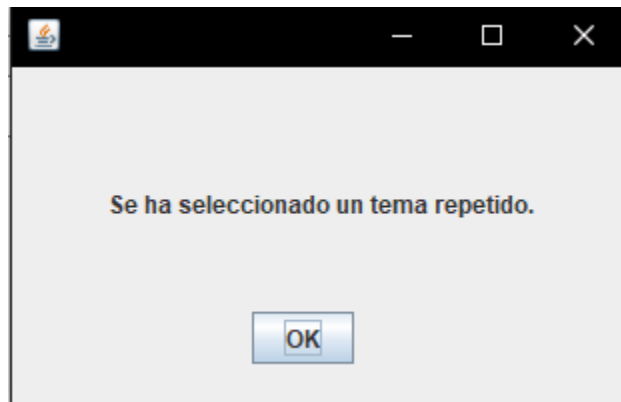
Se implementa Try-Catch para el error de “tema repetido” usado en la ventana “VentanaMostrarConFiltro.java” y también se implementa en la ventana “VentanaAgregarConsulta.java” con un “id repetido”.

```
public void RevisarTemas(String tema1, String tema2) throws ErrorTemaRepetidoException{
    if(tema1.equals(tema2)){
        throw new ErrorTemaRepetidoException();
    }else{
        mostrarCosultas();
    }
}

}

private void botonMostrarConsultasActionPerformed(java.awt.event.ActionEvent evt) {

    try {
        RevisarTemas((String) boxTemas.getSelectedItem(), (String) boxTemas2.getSelectedItem());
    } catch (ErrorTemaRepetidoException e) {
        this.error = new VentanaErrorField("Se ha seleccionado un tema repetido.");
        this.error.setVisible(true);
        return;
    }
}
```



Ej: En caso de ingresar un tema repetido dentro de la ventana de mostrar con filtro, el método “RevisarTemas” lanza un error, y se procede a ejecutar las líneas de código del catch, produciendo que se abra una ventana mostrando el error.

EF.3. Crear 2 clases que extiendan de una Excepción y que se utilicen en el programa.

Clase ErrorTemaRepetidoException que extiende de Exception, clase que se utiliza para funcionar como una excepción, en caso de que se encuentre un tema repetido, este es utilizado en VentanaMostrarConFiltro

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Modelo;

/**
 * Clase para excepcion de Tema
 */
public class ErrorTemaRepetidoException extends Exception {

    /**
     * Constructor
     */
    public ErrorTemaRepetidoException() {
        super("Tema Repetido!");
    }
}
```

Clase ErrorIdException que extiende de Exception, clase que se utiliza para funcionar como excepcion, en caso de que se encuentre un id repetido, este es utilizado en VentanaAgregarConsulta

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Modelo;

/**
 * Clase para excepcion de id
 */
public class ErrorIdException extends Exception {

    /**
     * Constructor
     */
    public ErrorIdException() {
        super("Error Id!");
    }
}
```

EF.4. Aplicación del patrón de diseño Strategy (Estrategia)

Se implementa Strategy con la siguiente interfaz, las cuales las clases “ReporteConsultaBinaria” y “ReporteConsultaMultiple” implementan esta.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Vista;

import java.io.IOException;

/**
 * Clase de tipo Interfaz:
 * Utilizada para los tipos de reportes.
 */
public interface ReportableStrategy {
    public void generarTxt() throws IOException;;
    public void generarExcel() throws IOException;;
}

public class ReporteConsultaBinaria implements ReportableStrategy {

}

public class ReporteConsultaMultiple implements ReportableStrategy {
```



Se hace uso de Strategy dentro de la ventana del menú principal.

Extra.

Se implementa lectura de datos desde archivo CSV.

Se implementa el Modelo Vista Controlador (MVC).