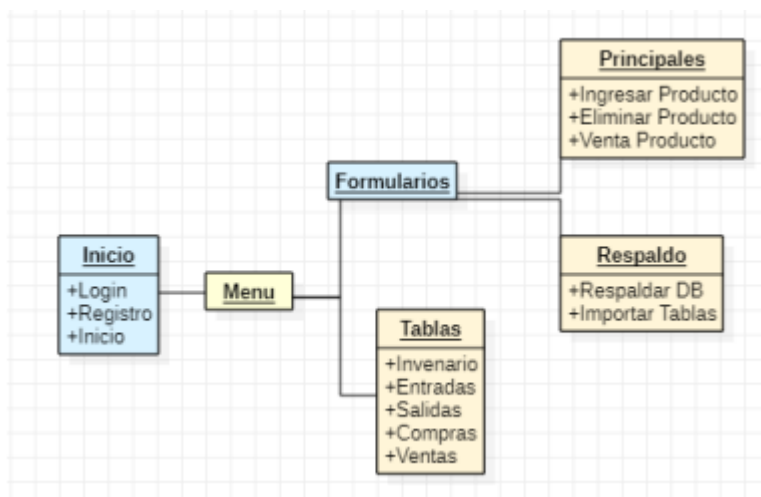


Pruebas por módulos

Tos de pruebas a realizar

- **Pruebas exploratorias** -> Verificar de forma general su funcionamiento, tanto la parte de diseño y la ejecución (detectar errores y los casos de uso de cada prueba).
- **Pruebas de Casos de Uso** -> Verificar que cumpla de manera correcta con el caso de uso que corresponda
- **Pruebas de caja negra** -> Se llevan a cabo sobre la interfaz del software; obviando el comportamiento interno y la estructura del programa. Los casos de uso de prueba de la caja negra pretenden demostrar que las funciones del software son operativas, la entrada se acepta de forma correcta, si se produce una salida correcta y si la integridad de la información externa se mantiene.
- **Pruebas de caja blanca** -> Se centran en los detalles procedimentales del software (el código funciona correctamente)



Módulo de Inicio

Historial de revisiones:

Revisión	Responsable(s)	Descripción	Fecha
Módulo de Inicio	Jesús Gerardo E7	Pruebas exploratorias	16/11/2021
Módulo de Inicio	Jesús Gerardo E7	Pruebas de Casos de Uso	17/11/2021
Módulo de Inicio	Jesús Gerardo E7	Pruebas de caja negra	18/11/2021
Módulo de Inicio	Jesús Gerardo E7	Pruebas de caja negra	19/11/2021
Módulo de Inicio	Jesús Gerardo E7	Modificaciones (pruebas de caja negra y caja blanca)	21/11/2021

Pruebas exploratorias revisión 1

1. Revisión general del módulo.

El programa inicia con el **formulario de sesión**. Se tiene que insertar el usuario y la contraseña.

Usuario:
Nombre de usuario

Contraseña:
.....

[]

Iniciar

Mensajes que muestra según los errores (se muestran en el espacio en blanco).

Datos incorrectos Llenar todos los campos

Después de equivocarse 3 veces se debe esperar 30 sec.

Esperar 27 segundos

En caso de ingresar correctamente los datos, pasa a la siguiente interfaz.

Hay 2 caminos a tomar si el usuario es tipo administrador (único usuario) para a la interfaz Inicio, si es usuario normal, pasa directo al menú del sistema.

El **menú del administrador** (Inicio) muestra un mensaje de bienvenida y contiene 3 botones, el **primero** manda directo al **menú** inicial, el **segundo** a una interfaz de **registro** de nuevo usuario y el **tercero** es para **iniciar** con otro **usuario** ya creado.

Si se presiona uno de los dos primeros botones se abre la interfaz correspondiente pero no se cierra la interfaz Inicio, lo que pasa es que se bloquea el botón y no deja abrir de nuevo la interfaz abierta hasta cerrarla.

Bienvenido

Proyecto Sistema
de Inventario

Menú

Registrar

Iniciar

El **formulario de registro** es parecido al de Iniciar sesión, en este se ingresa el nombre del nuevo usuario,

Usuario: — ✕

Nombre de usuario

Contraseña:

.....

Confirmar

.....

Iniciar

Los mensajes que se despliegan según los errores son los siguientes:

Error al guardar Completar todos los campos
 Las contraseñas no coinciden Usuario ya existente

2. Detección de errores

Fallos detectados	Beneficios obtenidos
Los botones no se bloquean por completo en la interfaz inicio	Los errores básicos que se obtienen de un mal ingreso de datos o algún otro error se despliegan directamente en las interfaces.
No es un error en sí, pero lo de esperar 30 seg. se esperaba que aumentara al doble después de la 3 equivocación, pero se reinicia en 0.	El diseño es agradable y los colores no cansan a la vista.
No se muestra el logo de la aplicación en el proceso de la interfaz	

3. Posibles soluciones a errores o cambios

- Modificar lo de los botones
- Buscar la forma de arreglar lo del tiempo de espera
- Agregar logo a todas las interfaces

Pruebas de caso de uso

1. Revisión general del módulo.

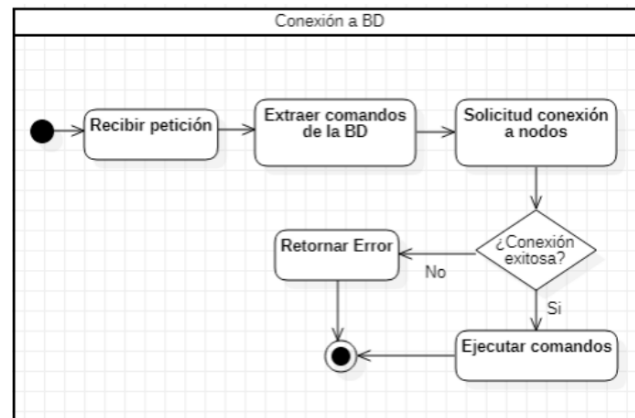
Inicio de sesión

Los siguientes son los diagramas que se establecieron inicialmente para el sistema. Al principio como solo era el diseño de la interfaz los datos de esta parte han sido modificados.

El proceso de inicio solo se tenía contemplado para un solo usuario y una vez confirmado pasaba directo al menú.

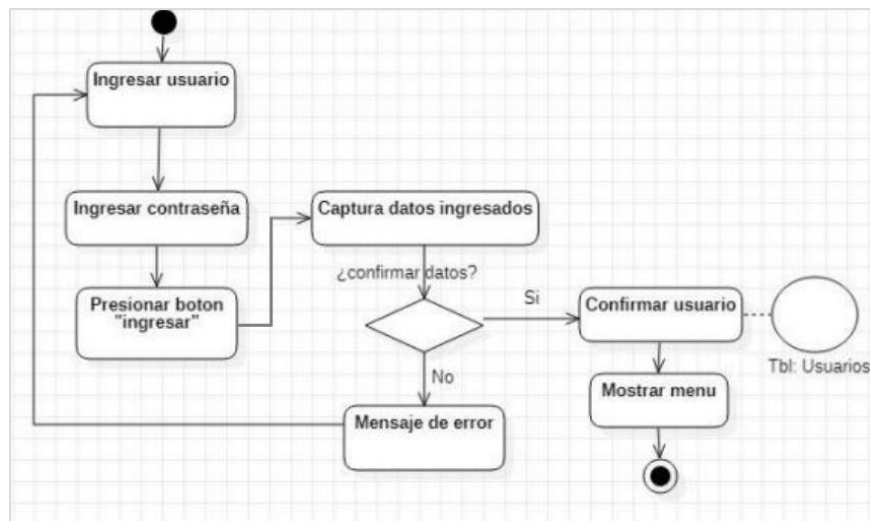
Para los mensajes de error solo se mostraba 1 mensaje de error por si existía algún problema, pero no especificaba el porqué.

Caso de uso para conexión: Se necesita de la conexión para poder utilizar las tablas.



Caso de uso para Iniciar sesión

- Diagrama de actividades inicial



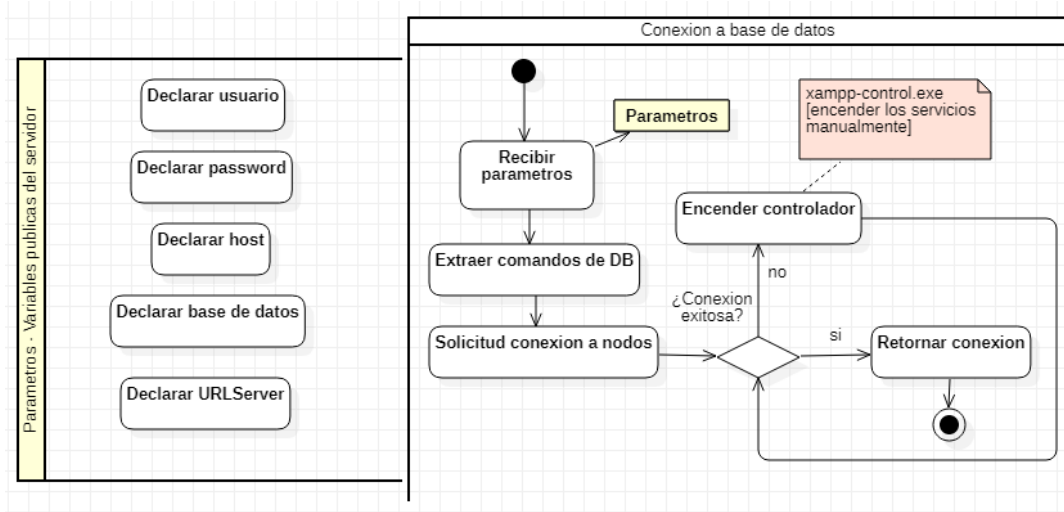
2. Detección de errores

Fallos detectados	Beneficios obtenidos
Los casos de uso se necesitan modificar en cuanto a los usuarios y caminos a tomar	Los cambios a hacer en esta parte son mínimos y no presentan algún problema
Modificar los caminos a tomar	
En la conexión, es mejor crear variables para cada parte de la conexión	
Falta un protocolo en caso de que no esté encendido el servidor (dentro de la conexión)	

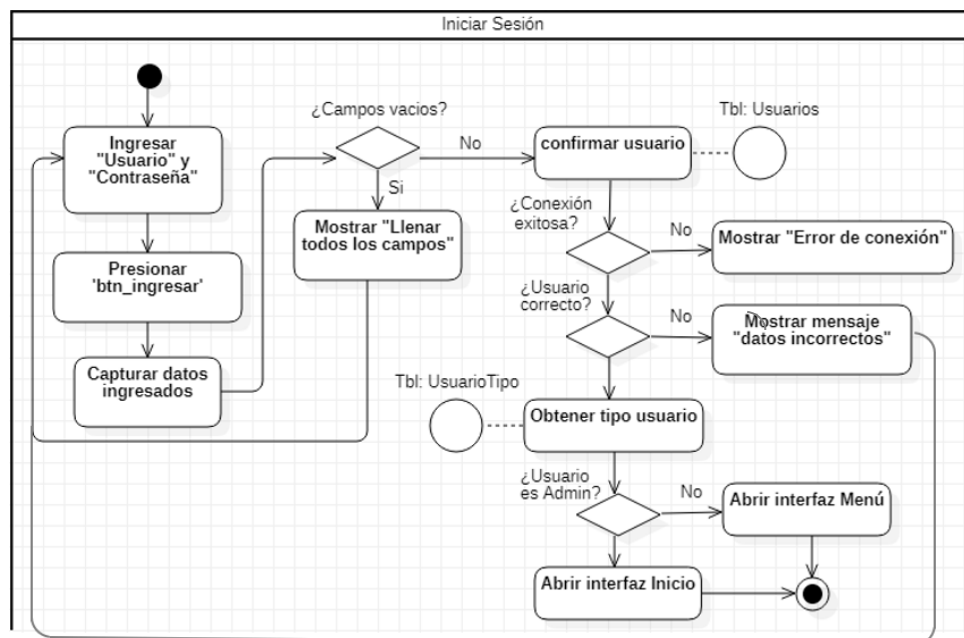
3. Posibles soluciones o cambios.

Los nuevos diagramas para este módulo se generaron a partir de estos casos de uso creados para mejorar el sistema según las pruebas, los cuales son los siguientes.

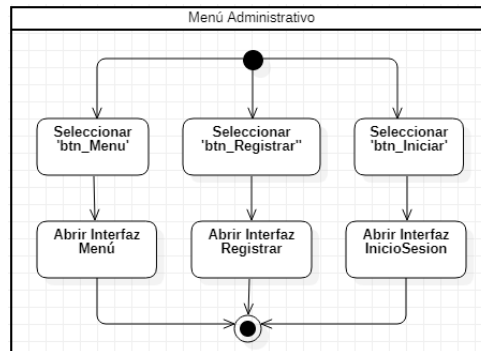
- **Conexión con base de datos:** Se hicieron modificaciones para los parámetros y errores
- Nuevo diagrama de actividades



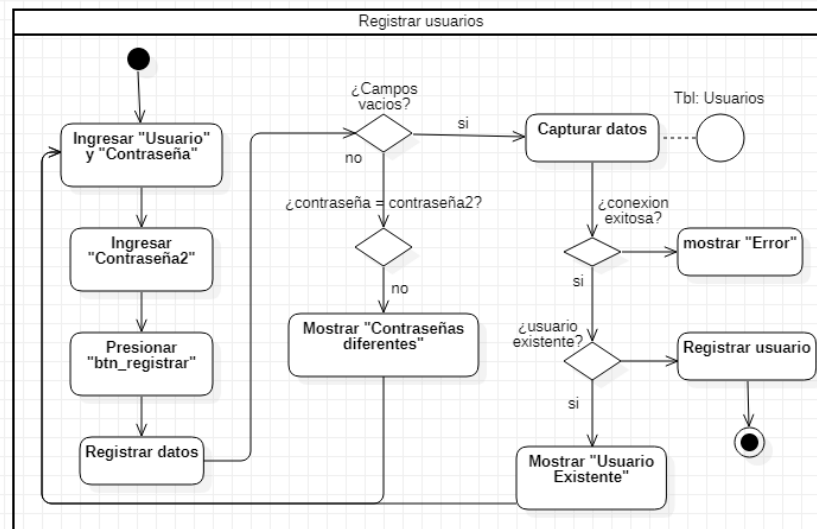
- **Inicio de sesión** (Caso de uso original): Ahora puede pasar a la interfaz menú o a la de administrador según corresponda
- Nuevo diagrama actividades



- **Inicio:** Interfaz principal del administrador, con 3 botones (btn_menu, btn_InicioSesion, btn_RegistrarUsuario).
- Diagrama de actividades:



- **Registro de usuario:** para generar nuevos usuarios no administradores.
- Diagrama de actividades



Pruebas de caja negra según los nuevos casos de uso

Pruebas con los casos de uso editados y fallos detectados anteriormente ya modificados dentro del programa

Inicio de sesión con servidor encendido, Nombre de usuario: Admin, Contraseña: admin.

Id_usuario	Usuario	Contraseña	Fecha_ingreso	Id_tipo
7	Admin	d033e22ae348aeb5660fc2140aec35850c4da997	2000-11-01 11:11:11	1


Prueba 1: Presionando iniciar

Sin modificar datos	Con campos vacíos	Usuario no registrado
Usuario: — ✕ <input type="text" value="Nombre de usuario"/>	Usuario: — ✕ <input type="text" value="user"/>	Usuario: — ✕ <input type="text" value="asd"/>
Contraseña: — ✕ <input type="password" value="....."/>	Contraseña: — ✕ <input type="password"/>	Contraseña: — ✕ <input type="password" value="..."/>
Datos incorrectos	Llenar todos los campos	Datos incorrectos
Iniciar	Iniciar	Iniciar

Prueba 2: Bloqueo despues de 3 intentos fallidos. Se bloquea el sistema hasta que pase el tiempo de espera.

Cuarto intento	Quinto intento	Sexto intento
Usuario: — ✕ <input type="text" value="Nombre de usuario"/>	Usuario: — ✕ <input type="text" value="Nombre de usuario"/>	Usuario: — ✕ <input type="text" value="Nombre de usuario"/>
Contraseña: — ✕ <input type="password" value="....."/>	Contraseña: — ✕ <input type="password" value="....."/>	Contraseña: — ✕ <input type="password" value="....."/>
Esperar 30 segundos	Esperar 90 segundos	Esperar 640 segundos
Iniciar	Iniciar	Iniciar

Prueba 3: Entrar como admin

Usuario: — ✕ <input type="text" value="Admin"/>	Bienvenido — ✕  Proyecto Sistema de Inventario
Contraseña: — ✕ <input type="password" value="....."/>	Menú
Iniciar	Registrar
	Iniciar

Se guarda la ultima fecha en la que ingreso al sistema.

Id_usuario	Usuario	Contraseña	Fecha_ingreso	Id_tipo
7	Admin	d033e22ae348aeb5660fc2140aec35850c4da997	2021-11-18 13:30:43	1

Prueba 4: Crear nuevo usuario

Muestra mensaje de confirmación y limpia los campos

Menú

Registrar

Iniciar

Usuario:

Usuario

Contraseña:

.....

Confirmar

.....

Iniciar

Usuario:

Usuario

Contraseña:

.....

Confirmar

.....

Registro nuevo guardado

Iniciar

Id_usuario	Usuario	Contraseña	Fecha_ingreso	Id_tipo
7	Admin	d033e22ae348aeb5660fc2140aec35850c4da997	2021-11-18 13:30:02	1
21	usuario	8cb2237d0679ca88db6464eac60da96345513964	2021-11-18 13:42:33	2

Despliegue de errores

Sin modificar datos

Usuario:

Nombre de usuario

Contraseña:

.....

Confirmar

.....

Error al guardar

Iniciar

Con campos vacíos

Usuario:

Contraseña:

.....

Confirmar

.....

Completar todos los campos

Iniciar

Contraseñas diferentes

Usuario:

Usuario

Contraseña:

...

Confirmar

....

Las contraseñas no coinciden

Iniciar

Usuario ya existente

Usuario:

Usuario

Contraseña:

.....

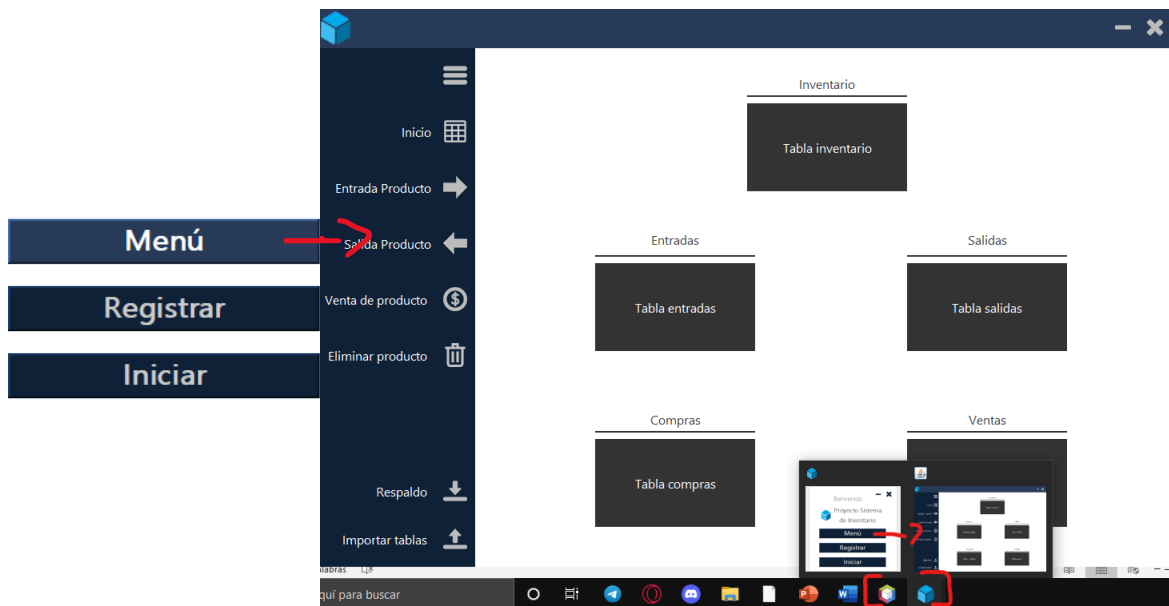
Confirmar

.....

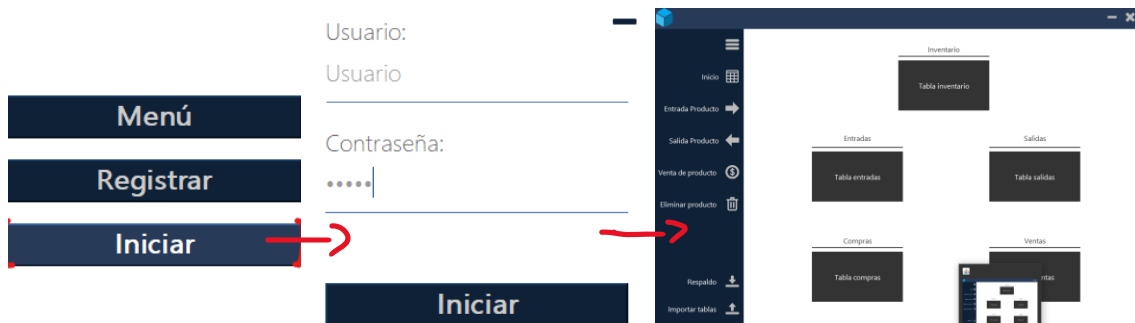
Usuario ya existente

Iniciar

Prueba 5: Entrar al menú



Prueba 6: Iniciar sesión con el usuario nuevo



Fallos detectados	Beneficios obtenidos
Falta una opción para eliminar usuarios.	Se arreglo el tiempo de espera, la conexión y el bloqueo de los botones
Aún falta poner bien el logo en la barra de tareas	El manejo de usuarios funciona correctamente.
	No se lanzaron fallos al momento de cada proceso.

Pruebas de caja blanca y modificaciones realizadas o a realizar.

Después de realizar ciertas correcciones y cambios conforme se hacían las pruebas anteriores, el código quedo casi completo.

Los cambios que siguen son para un mejor orden dentro de la estructura de este, mas que nada para que quede limpio.

Nota: Comentar las tareas que realizan las instrucciones para evitar confusiones.

Se necesita buscar clases o partes dentro de clases que no se estén utilizando en el sistema.

Inicio de sesión

Prueba de intentos: Se imprimió en consola los tiempos para ver si funcionaba la espera en los intentos. Se eliminan los mensajes de consola al final.

Paquete: inicio, **clase:** LogUser.java.

```
private void intento() {
    frmBtn = frmBtn + 1;
    System.out.println("intento " + frmBtn);
    if (frmBtn == 3) {
        // BTNINICIAR1.setEnabled(false);
        // if (!t.isRunning()) {
        espera = 30;
        System.out.println("tiempo " + espera + " sec");
        TXERROR.setText("Esperar " + espera + " segundos");

        // t.start();
        // }
    } else if (frmBtn > 3) {
        // BTNINICIAR1.setEnabled(false);
        // if (!t.isRunning()) {
        espera = (int) (10 * (Math.pow(frmBtn - 1, frmBtn - 2)));
        System.out.println("tiempo " + espera + " sec");
        TXERROR.setText("Esperar " + espera + " segundos");

        // t.start();
        // }
    }
}
```

intento 1
intento 2
intento 3
tiempo 30 sec
intento 4
tiempo 90 sec
intento 5
tiempo 640 sec
intento 6
tiempo 6250 sec

Así quedaría de forma funcional

```
private void BTNINICIAR1ActionPerformed(java.awt.event.ActionEvent evt) {
    validarUsuario();
    intento();
}
```

```
//contador de intentos
private void intento() {
    frmBtn = frmBtn + 1;
    // 3er intento bloqueo
    if (frmBtn == 3) {
        BTNINICIAR1.setEnabled(false);
        if (!t.isRunning()) {
            espera = 30;
            TXERROR.setText("Esperar " + espera + " segundos");
            t.start();
        }
    }
    //incremento de bloqueos
    else if (frmBtn > 3) {
        BTNINICIAR1.setEnabled(false);
        if (!t.isRunning()) {
            espera = (int) (10 * (Math.pow(frmBtn - 1, frmBtn - 2)));
            TXERROR.setText("Esperar " + espera + " segundos");
            t.start();
        }
    }
}
```

```
private void btn_salirMouseClicked(java.awt.event.MouseEvent evt) {
    // bloquear si el intento de inicio es mayor a 3
    if (frmBtn > 3) {
    } else {
        System.exit(0);
    }
}
```

Prueba de que el usuario se guarda en el sistema

El modelo de la base de datos y el modelo de los registros se mandan a llamar dentro de la clase principal.

```
public class LogUser extends javax.swing.JFrame {

    //Llamada al modelo de las variables de cada tabla y modelo sql de usuarios
    columnasTab mod = new columnasTab();
    SqlUsuarios modSql = new SqlUsuarios();
```

Se guardo el nombre del usuario dentro del modelo, posteriormente se usara get para guardarlo hasta que se cierre el sistema.

```
mod.setUsuario_responsableENTRADA(TXTUSUARIO.getText());|
```

Entradas según tipo de usuario. Dentro de private void validarUsuario(), línea: 279

Se manda a llamar a la clase varUser con el modelo para guardar el usuario

```
// Entrada usuario administrador
if (mod.getId_tipoUSUARIOS() == 1) {
    Inicio abrir = new Inicio();
    abrir.setVisible(true);
    new VarUser(mod).setVisible(true);

    // Entrada usuario normal
} else if (mod.getId_tipoUSUARIOS() == 2) {
    Menu abrir = new Menu();
    abrir.setVisible(true);
    new VarUser(mod).setVisible(true);
}
```

Paquete: Principal, clase: VarUser.java.

Se guarda el usuario en la variable frmUsr para poder manejar el sistema y tener registro

```
public class VarUser extends javax.swing.JFrame {

    /* Creates new form ABC */
    public VarUser() {
        initComponents();
    }

    //Declaracion de la variable usuario
    columnasTab mod;
    public static String frmUsr;

    public VarUser(columnasTab mod) {
        initComponents();
        setLocationRelativeTo(null);

        //frmUsr toma el valor guardado en el modelo por el JtexpForm de LogUser
        frmUsr = mod.getUsuarioUSUARIOS();

        //System.out.println("Prueba Usuario: " + mod.getUsuarioUSUARIOS());
    }
```

Para hacer la prueba se manda a imprimir en la consola

```

public VarUser(columnasTab mod) {
    initComponents();
    setLocationRelativeTo(null);

    //frmUsr toma el valor guardado en el modelo por el JtexForm de LogUser
    frmUsr = mod.getUsuarioUSUARIOS();

    System.out.println("Prueba Usuario 1" + mod.getUsuarioUSUARIOS());
    System.out.println("Prueba Usuario 2" + frmUsr);
}

```

```

compile:
run:
Prueba Usuario 1Admin
Prueba Usuario 2Admin
Prueba Usuario 1usuario
Prueba Usuario 2usuario
BUILD SUCCESSFUL (total time: 28 seconds)

```

Controlar los botoes de la interfaz Inicio

En la clase principal de este se declaran 3 variables esticas, una para cada iterfaz a la que se retorna

```

public class Inicio extends javax.swing.JFrame {

    // variables publicas de botones
    public static registro frmReg;
    public static Menu frmMen;
    public static LogUser frmLog;
}

```

Despues, en la accion de seleccionar el boton correspondiente se guardan como llamadas a otras clases para que no sean nulas.

```

private void BTNIMENUActionPerformed(java.awt.event.ActionEvent evt) {
    if (frmMen == null) {
        frmMen = new Menu();
        frmReg = new registro();
        frmLog = new LogUser();

        frmMen.setVisible(true);
    }
}

```

```

private void BTNREGISTRARActionPerformed(java.awt.event.ActionEvent evt) {
    if (frmReg == null) {
        frmMen = new Menu();
        frmReg = new registro();
        frmLog = new LogUser();
        frmReg.setVisible(true);
    }
}

private void BTNINICIARActionPerformed(java.awt.event.ActionEvent evt) {
    if (frmLog == null) {
        frmMen = new Menu();
        frmReg = new registro();
        frmLog = new LogUser();
        frmLog.setVisible(true);
        this.dispose();
    }
}

```

De esta manera pasan a la siguiente interfaz correspondiente.

Y dentro de las otras interfaces se cambia a nulo para que los botones se vuelvan a activar, cuando las acciones sean, cerrar la ventan.

Paquete: Principal, clase: Menu.java.

```
private void cerrarMousePressed(java.awt.event.MouseEvent evt) {  
    if (frmUsr == null) {  
        System.exit(0);  
    } else if (frmUsr.equals("Admin")) {  
        Inicio.frmMen = null;  
        Inicio.frmReg = null;  
        Inicio.frmLog = null;  
        this.setVisible(false);  
    } else {  
        System.exit(0);  
    }  
}  
  
private void formWindowClosing(java.awt.event.WindowEvent evt) {  
    if (frmUsr.equals("Admin")) {  
        Inicio.frmMen = null;  
        Inicio.frmReg = null;  
        Inicio.frmLog = null;  
    } else {  
    }  
}
```

Paquete: Inicio, clase: Registro.java.

```
private void formWindowClosing(java.awt.event.WindowEvent evt) {  
    Inicio.frmMen = null;  
    Inicio.frmReg = null;  
    Inicio.frmLog = null;  
}  
  
private void btn_salirMousePressed(java.awt.event.MouseEvent evt) {  
    Inicio.frmMen = null;  
    Inicio.frmReg = null;  
    Inicio.frmLog = null;  
    this.setVisible(false);  
}
```

Paquete: Inicio, clase: LogUser.java.

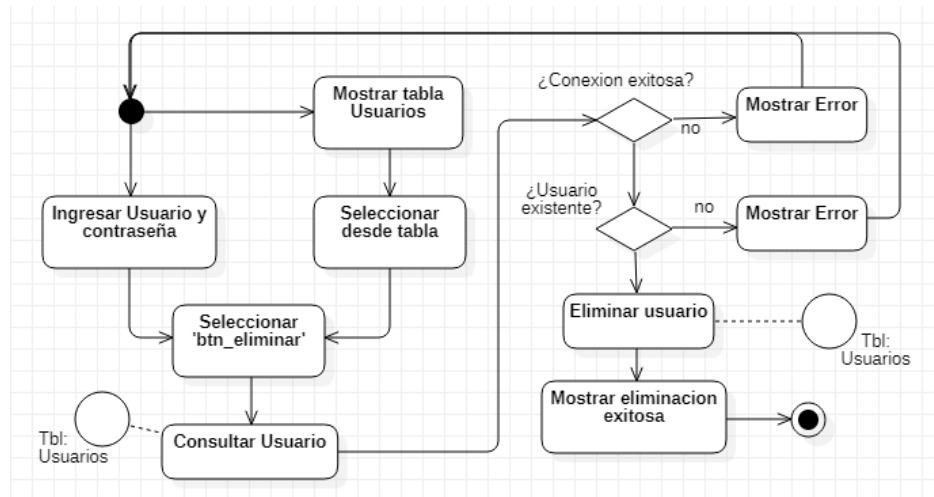
```
private void formWindowClosing(java.awt.event.WindowEvent evt) {  
    Inicio.frmMen = null;  
    Inicio.frmReg = null;  
    Inicio.frmLog = null;  
}  
  
private void btn_salirMouseClicked(java.awt.event.MouseEvent evt) {  
    Inicio.frmMen = null;  
    Inicio.frmReg = null;  
    Inicio.frmLog = null;  
    if (frmBtn > 3) {  
    } else {  
        System.exit(0);  
    }  
}
```

Modificacion – nuevo caso de uso e interfaz eliminacion de usuarios.

Durante las pruebas de caja negra nos percatamos de que falta una opcion para eliminar usuarios, en esta se ven perjudicadas las partes:

- Interfaz de Inicio: Crear nueva variable privada frmEliUsr / Modificar interfaz - Crear boton .
- Interfaz EliminarUsuario: Crear interfaz y caso de uso.
- Clase EliminarUsuarioSQL: Crear modelo sql para eliminar usuarios no administradores.

Digrama de secuencias Interfaz Eliminar Usuario



Diseño de la interfaz

Id

Usuario Contraseña

Id_Usuario	Usuario	Fecha_Ingreso

Para la programacion se opto por eliminar el modelo sql y dejarlo todo en la clase de la interfaz.

Para verificar que funciona correctamente cada una de las consultas sql se uso la consola de phpMyAdmin

Mostrar ventana de consultas SQL

✓ Mostrando filas 0 - 4 (total de 5, La consulta tardó 0,0023 segundos.)

```
SELECT Id_usuario,Usuario,Fecha_ingreso FROM usuarios WHERE Id_tipo = 2
```

■ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

■ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Sort by key: Ninguna

+ Opciones

				Id_usuario	Usuario	Fecha_ingreso
■	✎ Editar	📄 Copiar	🗑️ Borrar	21	usuario	2021-11-24 21:39:03
■	✎ Editar	📄 Copiar	🗑️ Borrar	22	asdasd	2021-11-24 21:44:52
■	✎ Editar	📄 Copiar	🗑️ Borrar	23	123	2021-11-24 21:44:58
■	✎ Editar	📄 Copiar	🗑️ Borrar	24	raul	2021-11-24 21:45:06
■	✎ Editar	📄 Copiar	🗑️ Borrar	25	mani	2021-11-24 21:45:14

Mostrar ventana de consultas SQL

✓ 1 fila afectada. (La consulta tardó 0,1707 segundos.)

```
DELETE FROM usuarios WHERE Id_usuario= 25
```

En la clase Inicio.Inicio.java se agrego la nueva variable

```
public static registro frmReg;
public static Menu frmMen;
public static LogUser frmLog;
public static ElimUsuario frmEliU;
```

y el nuevo boton

```
private void BTNELIMINARActionPerformed(java.awt.event.ActionEvent evt) {
    if (frmUsr == null) {

    } else if (frmEliU == null) {

        frmEliU = new ElimUsuario();
        frmMen = new Menu();
        frmReg = new registro();
        frmLog = new LogUser();

        frmEliU.setVisible(true);
        this.dispose();
    }
}
```

Igual que en las demas clases en las acciones de salir (menu.java, inicio.java, registrar.java, ElimUser.java)

```
private void salirMouseClicked(java.awt.event.MouseEvent evt) {
    Inicio.frmEliU = null;
    Inicio.frmMen = null;
    Inicio.frmReg = null;
    Inicio.frmLog = null;
    this.dispose();
}
```

Y así queda la nueva interfaz:

Bienvenido

Proyecto Sistema de Inventario

Menú

Registrar

Iniciar

Editar Usuarios

Id **Buscar**

Usuario Contraseña

Nombre de usuario Contraseña

Id_usuario	Usuario	Fecha_ingreso
21	usuario	2021-11-24 21:39:0...
22	asdasd	2021-11-24 21:44:5...
23	123	2021-11-24 21:44:5...
24	raul	2021-11-24 21:45:0...

Editar **Eliminar Usuario**

Mensajes de error según fallos

Usuario no existente **Error Falta Id**

Llenar todos los campos **Error al modificar usuario**

Modulo de Ingreso, Eliminacion y ventas

Historial de reviciones:

Módulo de Ingresar, Eliminar y Ventas	Gonzalo Safkiel E7	Pruebas exploratorias de Ingresar, Eliminar y Ventas	18/11/2021
Módulo de Ingresar, Eliminar y Ventas	Gonzalo Safkiel E7	Pruebas de Casos de Uso de Ingresar, Eliminar y Ventas	22/11/2021
Módulo de Ingresar, Eliminar y Ventas	Gonzalo Safkiel E7	Pruebas de Pruebas de caja negra y caja blanca	24/11/2021
Módulo de Ingresar, Eliminar y Ventas	Gonzalo Safkiel E7	Modificaciones (pruebas de caja negra y caja blanca)	25/11/2021

Pruebas exploratorias revisión 1

Revisión general de Ingresar.

La parte de la interfaz Ingresar comienza con el formulario de Ingresar, este se enfoca en introducir valores(Producto,Proveedor,Fecha,Precio,Estado del Producto y la Categoría) nesarios para registrar un nuevo producto en el sistema.



Nota: el campo de Precio y Cantidad solo pueden recibir numeros.

Los siguientes son los mensajes de error de diferentes casos:

Completar todos los campos

Error al guardar Producto

Producto Existente

Revisión general de Eliminar.

La siguiente interfaz se usa para eliminar un producto permanentemente de la base de datos si el proveedor y el nombre del producto son iguales.

Producto

Proveedor

Nombre del proveedor

Fecha:

Eliminar producto

Los siguientes son los mensajes de error de diferentes casos:

Completar todos los campos

Error al registrar salida

Producto Borrado

Revisión general de Ventas.

La siguiente interfaz se enfoca en realizar la venta de productos, la cantidad que ingresa aquí se restara de la cantidad de inventario y se sumara a las salidas que están igualmente en inventario. Pide el Nombre, el proveedor, Comprador, Fecha de venta, Precio y Cantidad.

Producto

Nombre del producto

Proveedor

Nombre del proveedor

Comprador

Nombre del Comprador

Fecha:

Precio

Cantidad

\$\$\$\$

00000

Realizar Venta

Completar todos los campos

Error

Venta realizada

Detección de errores

Fallos detectados	Beneficios obtenidos
El mensaje de error “Producto Existente” no aparece al cumplir con la condición solicitada	Lo demás funciona bien

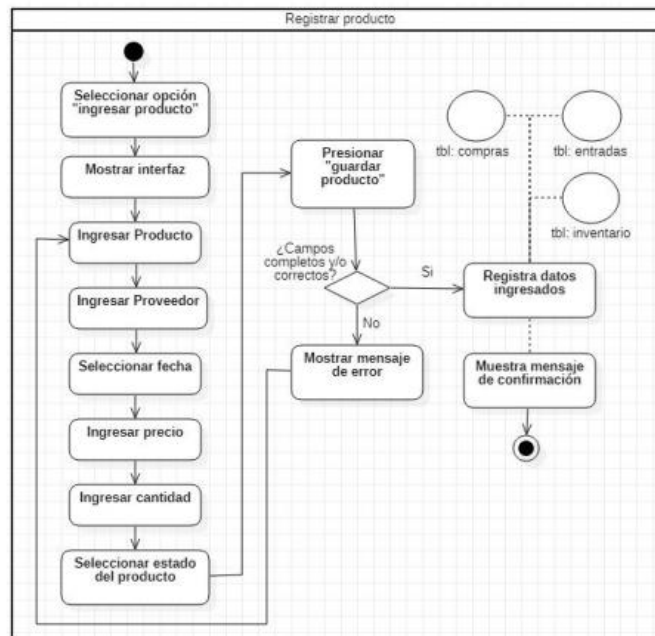
Posibles soluciones o cambios.

-Modificar el código para que el mensaje aparezca cuando la condición se cumpla

Pruebas de caso de uso

Ingresar producto

El siguiente diagrama se enfocaba en ingresar un producto, los ingresaba con los datos correctos, pero este no introduce la categoría a la que está sujeto el producto. Además, solo tiene un mensaje de error el cual se daba si faltaban campos.

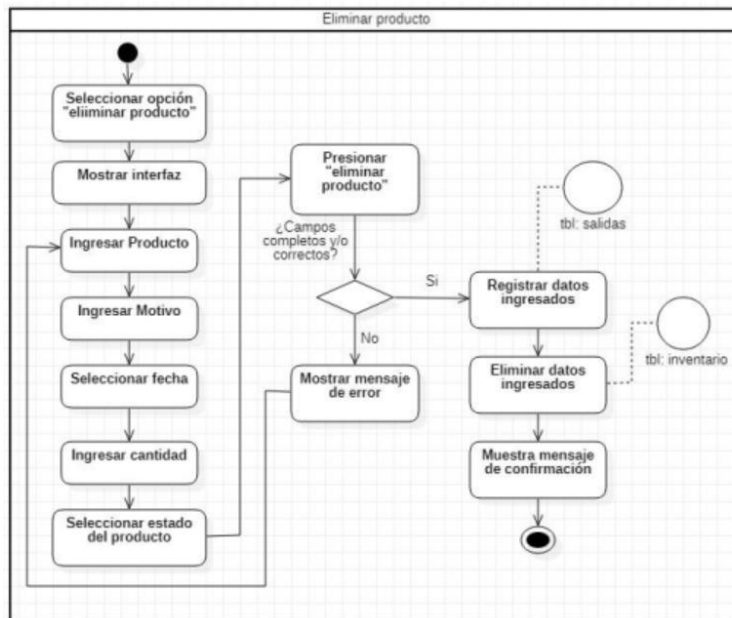


Eliminar Producto

El diagrama de actividades de eliminar estaba incorrecto, el caso de uso era correcto, pero mal implementado, esto debido a que para eliminar un producto era mejor hacerlo de una manera sencilla y que solo borrara el producto permanentemente. Pero en este caso, este pedía datos para la tabla salidas y esto creaba un conflicto interno en el código para obtener datos.

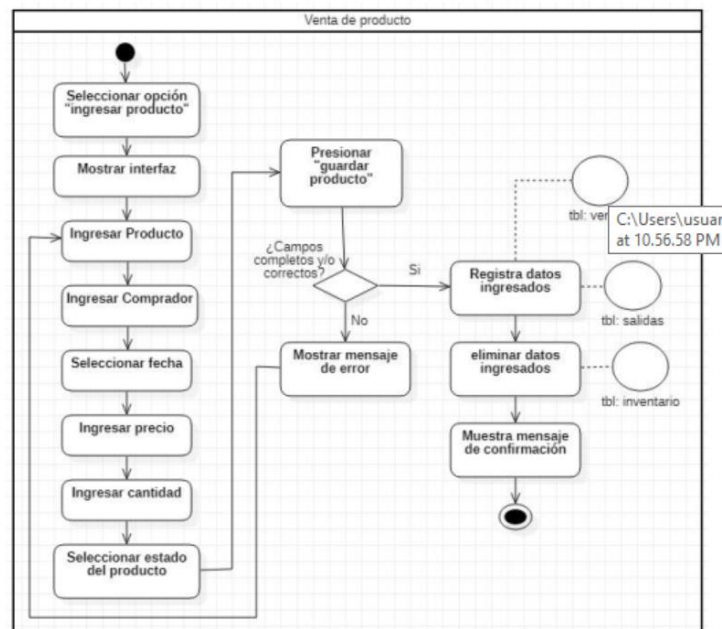
Por lo tanto, se optó por separar la eliminación con el registro de salidas. Esto provocará la creación de un nuevo caso de uso únicamente para salidas, el de eliminación se modificará.

El siguiente es el caso de uso antiguo de la eliminación de producto.



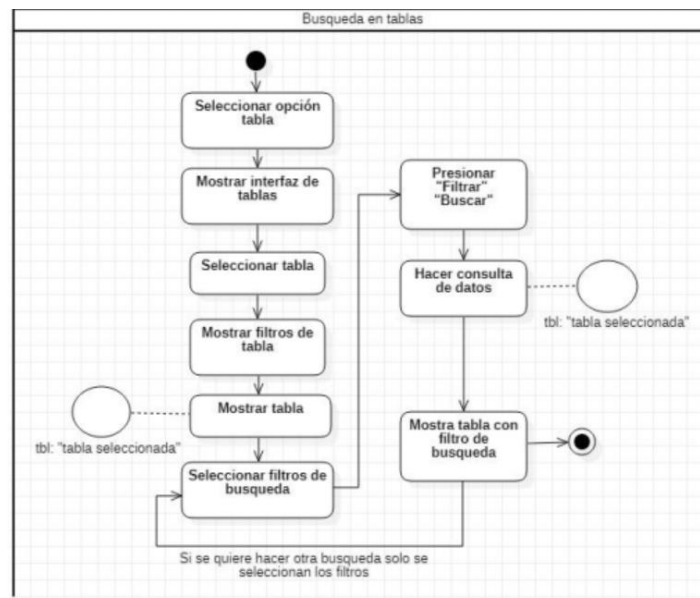
Venta de Producto

El siguiente diagrama es el del caso de uso de la venta, este tiene una mala interpretación ya que necesita de más actividades para poder hacer correctamente la introducción de datos en la tabla ventas y en otras.



Búsqueda de datos

El diagrama siguiente es el de la búsqueda, este no será modificado, aunque solo se aplicará para poder buscar productos en Ingresar.java, esto debido que en los otros no será necesario como antes se tenía previsto.



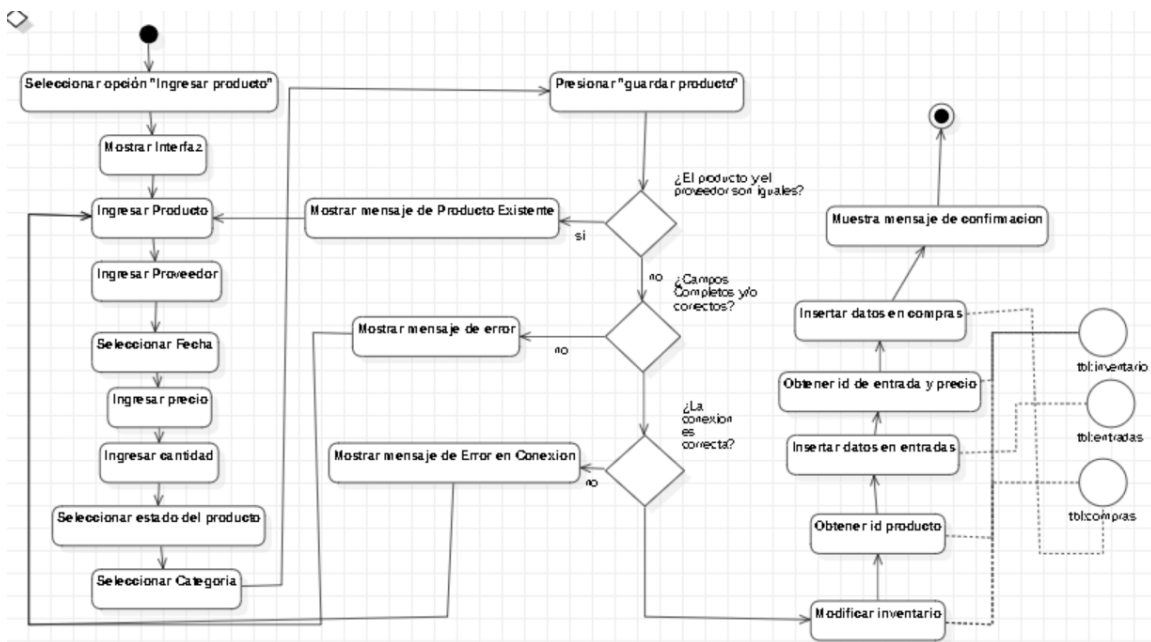
Detección de errores

Fallos detectados	Beneficios obtenidos
El caso de uso necesita modificarse en cuanto el camino y actividades nuevas.	El cambio representa una mayor facilidad de obtención de valores y reciclaje de código
Modificar los caminos a tomar	
Faltan variables para poder obtener mejor los valores	

Posibles soluciones o cambios.

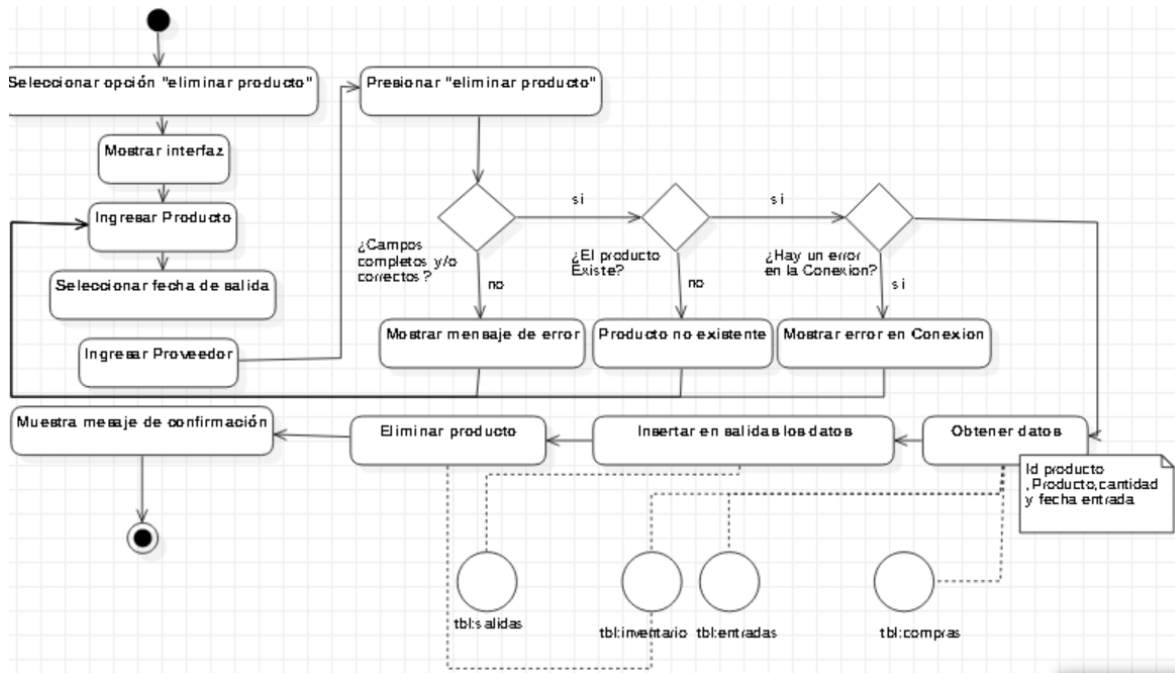
Nuevo Caso de uso Ingresar

El siguiente diagrama fue generado a partir del anterior, este ingresa la categoría y además suma otros métodos en orden para almacenar datos u obtener datos.



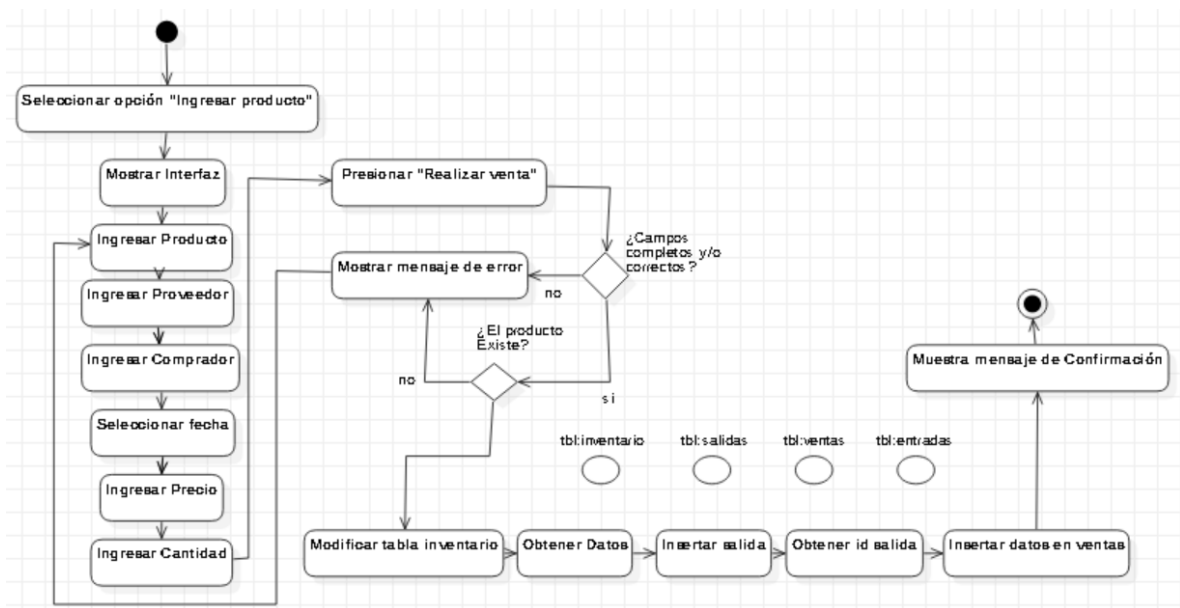
Eliminar

Debido a la separación de eliminación y ventas este nuevo caso de uso tiene más actividades para el completo funcionamiento de la eliminación del producto.



Nuevo caso uso de Ventas

Este diagrama cumple con los requerimientos para que sirva correctamente las ventas realizadas con más actividades a realizar.



Pruebas de caja negra modulo Ingresar, Eliminar y Ventas

Pruebas con el caso de uso modificado

Ingresar

Como ejemplo se agregará un nuevo producto, con los siguientes datos:

Este tendrá nombre “Cable UTP”.

Con un proveedor “Stereon”.

Fecha de entrada de 30 de noviembre del 2021.

Con precio de \$250 y con una cantidad de 50.

En estado de Madurez

Y una categoría en Tecnología.

✖
Buscar

Producto

Proveedor

Fecha:

Precio

Cantidad

Estado del producto

Categoría

Ingresar producto

El mensaje de verificación es el siguiente:

Producto Guardado

Según el modelo de datos esto debería suceder cuando el producto a sido correctamente introducido en la base de datos.

Para verificar esto se tendrá que visualizar el tabla en MySQL:

▼	Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
rar	554	Cable UTP	Tecnologia	50	0	Fase de Madurez	250	50

Si en algunos campos no hay caracteres se mostrara el mensaje de error propuesto:

Producto

Proveedor

Fecha:

30 nov. 2021

Precio

Cantidad

Estado del producto

Fase de Madurez

Categoria

Completar todos los campos

Ingresar producto

En caso de que el servidor falle se mostrara el siguiente mensaje:

Buscar

Producto

Proveedor

Fecha:

11 nov. 2021

Precio

Cantidad

Estado del producto

Fase de Desarrollo

Categoria

rrrrr

Error al guardar Producto

Ingresar producto

De esta manera podemos comprobar que función correctamente en relación al caso de uso modificado.

Eliminar

Se usará de ejemplo la eliminación de un producto ya ingresado llamado "PRO" de las tablas inventario.

Primeramente, se tiene

Opciones									
	Id_producto	Producto	Categoría	Entrada	Salida	Estado_producto	Precio	Cantidad	
	585	PRO	rverr	2332	0	Fase de Desarrollo	4224	2332	

Y se introducen el nombre del producto, el proveedor y la fecha de salida:

Producto

PRO

Proveedor

www

Fecha:

30 nov. 2021

Eliminar producto

El resultado es el siguiente

Fecha:

30 nov. 2021

Salida Exitosa

Eliminar producto

+ Opciones							
 	Id_producto	Producto	Categoría	Entrada	Salida	Estado_producto	Precio Cantidad
<div> Seleccionar todo</div> <div>Para los elementos que están marcados:</div> <div> Editar</div> <div> Copiar</div> <div> Borrar</div> <div></div>							

Por lo tanto, la eliminación es exitosa.

Si alguno de los campos está vacío en la eliminación sucede lo siguiente:

Producto

Proveedor

Fecha:

Completar todos los campos

Eliminar producto

Si existe un error en el sistema parece el siguiente mensaje:

Producto

eeec

Proveedor

ecwec

Fecha:

10 nov. 2021

Error al registrar salida

Eliminar producto

Ventas

Se ingresará una venta nueva del producto PRO, con el mismo proveedor, pero en este caso el Comprador será Comprador1 con una cantidad de 15.

Producto

PRO

Proveedor

PRO

Comprador

Comprador1

Fecha:

26 nov. 2021

Precio

15

Cantidad

15

Venta Exitosa

Realizar Venta

El Resultado es:

15	805	566	Comprador1	1	15	222.0	2021-11-26 23:5...
----	-----	-----	------------	---	----	-------	--------------------

Si hay campos vacíos se mostrara el siguiente mensaje:

Producto	<input type="text"/>
Proveedor	<input type="text"/>
Comprador	<input type="text"/>
Fecha:	<input type="text"/>
Precio	Cantidad
<input type="text"/>	<input type="text"/>
Completar todos los campos	
Realizar Venta	

Si existe un error en el sistema se mostrará el mensaje:

11 nov. 2021
Precio Cantidad
<input type="text"/>
Error
Realizar Venta

Fallos detectados	Beneficios obtenidos
Ninguno	El funcionamiento es más seguro
	No hay fallos cuando el servidor está correcto
	No se lanzaron fallos al momento de cada proceso.

Pruebas de caja blanca y modificaciones realizadas o a realizar.

Después de las correcciones de las pruebas anteriores se realizarán las pruebas del código

Ingresar producto

Prueba de valores: Se imprimirá en consola los valores obtenidos de los Campos de la clase Ingresar.java para saber si no existen valores perdidos.

Paquete: Modelos, **clase:** columnasTab.java.

La Id del producto de prueba es : 565

En la tabla inventario:

+ Opciones								
	Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
	565	PRO	rverr	2332	0	Fase de Desarrollo	4224	2332

En la tabla entradas:

	565	2021-11-04 23:15:43	1	2332
--	-----	---------------------	---	------

Prueba de eliminar un producto desde código:

Paquete: Modelos, clase: EliminarProductoSQL.java.

Para probar que la eliminación es correcta se borrara un producto desde el código de la siguiente manera:

Primero se modificará siguiente código para saber si se puede eliminar un producto desde dentro

```
String delete = "DELETE inventario FROM inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto "
                + " WHERE inventario.Producto = ? and compras.Proveedor = ?";
ps = (PreparedStatement) con.prepareStatement(delete);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();
return true;
```

Se modificará para que borre el producto llamado “PRO”

```
String delete = "DELETE inventario FROM inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto "
                + " WHERE inventario.Producto = ? and compras.Proveedor = ?";
ps = (PreparedStatement) con.prepareStatement(delete);
ps.setString(1, "PRO");
ps.setString(2, "PRO");
```

El resultado es el siguiente:

+ Opciones								
	Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
Seleccionar todo Para los elementos que están marcados: Editar Copiar Borrar								

Prueba de comprobación la reducción en cantidad en la tabla inventario:

Paquete: Modelos, clase: VnetaSQL.java.

Según el funcionamiento de las ventas, la cantidad introducida en el número de ventas deberá disminuir la cantidad en el inventario, por lo tanto, se probará su funcionalidad.

Se ingresará un producto con el nombre de “Prueba”, proveedor “Prueba” y con una cantidad de 40.

Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
567	Prueba	f	40	0	Fase de Desarrollo	1300	40

El código incluirá la cantidad 15 para imprimirla y saber si esta cantidad se restara de inventario y se suma en salidas.

```

int cantidad = 15;
System.out.println("La cantidad que disminuira es la siguiente: "+cantidad);

String sql = "UPDATE inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto SET "
+ "inventario.Salida = inventario.Salida+" + cantidad + ", inventario.Cantidad = inventario.Cantidad-"
+ cantidad + " WHERE inventario.Producto = ? AND compras.Proveedor = ?;";
ps = (PreparedStatement) con.prepareStatement(sql);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();

```

El resultado es el siguiente en la tabla inventario:

Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
567	Prueba	f	40	15	Fase de Desarro...	1300	25

El resultado es el siguiente en la tabla salidas:

806	567	Prueba	2021-11-30 00:2...	2021-11-11 00:2...	venta	1	15
-----	-----	--------	--------------------	--------------------	-------	---	----

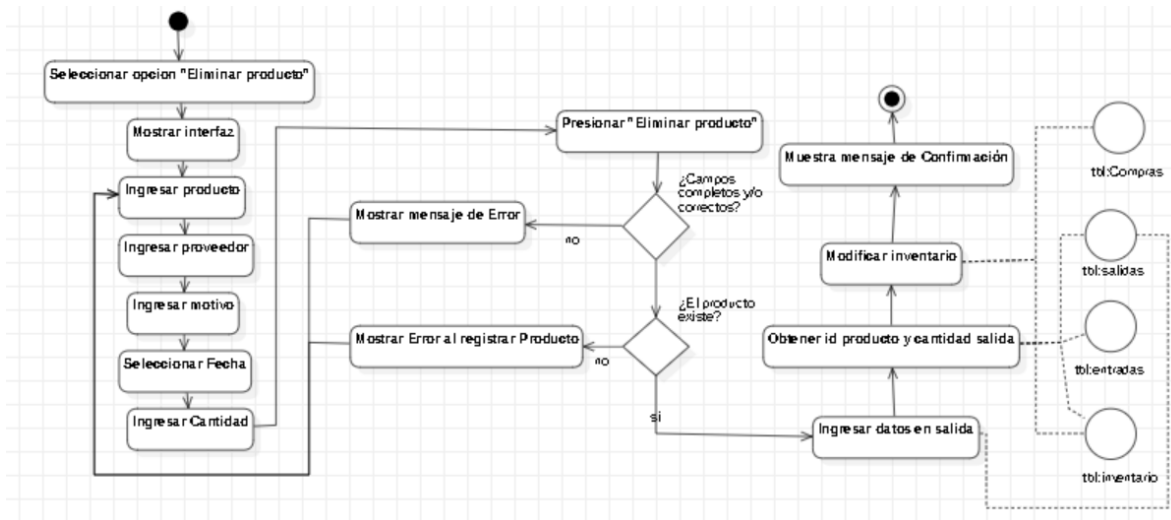
Modificacion – nuevo caso de uso e interfaz Salidas.

Debido a que el caso de uso eliminar fue dividido se creo otro caso de uso para el registro de salidas para no eliminar un producto. las siguientes son las partes modificadas:

- Interfaz de Menu: Crea un nuevo apartado dentro del menu.
- Clase VentasSQL: Modifica el metodo ventas y crea nuevas variables y consultas

Diagrama de secuencias Salidas

Este es el nuevo diagrama de casos de uso del apartado salidas, no elimina ningún producto, pero si modifica las demás tablas.



Diseño de Interfaz

Producto

Nombre del producto

Proveedor

Nombre del proveedor

Motivo

Nombre del proveedor

Fecha:

Cantidad

00000

Eliminar producto

Se realizo la consulta en sql para verificar el funcionamiento de VentasSQL dentro de la consola de phpMyAdmin::

```

1 fila afectada. (La consulta tardó 0,0001 segundos.)

UPDATE inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto SET inventario.Salida = inventario.Salida+25,
inventario.Cantidad = inventario.Cantidad- 25 WHERE inventario.Producto = 'P' AND compras.Proveedor = 'P';

[ Editar en línea ] [ Editar ] [ Crear código PHP ]

```

```

SELECT inventario.Id_producto,inventario.Salida ,entradas.Fecha_entrada FROM compras INNER JOIN inventario ON
compras.Id_producto = inventario.Id_producto INNER JOIN entradas ON inventario.Id_producto = entradas.Id_producto WHERE
inventario.Producto = 'P' AND compras.Proveedor = 'P';

```

■ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas:

Opciones

Id_producto	Salida	Fecha_entrada
568	25	2021-11-04 00:47:45

```

1 fila insertada. (La consulta tardó 0,0368 segundos.)

insert into salidas (Id_salida,Id_producto,Fecha_entrada,Fecha_salida,Motivo,usuario_responsable,Cantidad, Producto
VALUES(111111,111111,25-01-24,15-01-24,'n','e',2,'dd');

[ Editar en línea ] [ Editar ] [ Crear código PHP ]

```

Warning: #1264 Out of range value for column 'Fecha_entrada' at row 1

Warning: #1265 Datos truncados para columna 'Fecha_salida' en la línea 1

```

MySQL ha devuelto un conjunto de valores vacío (es decir: cero columnas). (La consulta tardó 0,0046 segundos.)

SELECT salidas.Id_salida FROM salidas INNER JOIN inventario ON salidas.Id_producto = inventario.Id_producto INNER JOIN compras
ON inventario.Id_producto = compras.Id_producto WHERE inventario.Producto = 'P' AND compras.Proveedor = 'P';

■ Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]

```

Id_salida

```

insert into ventas (Id_venta, Id_salida, Id_producto, Comprador, Responsable, Cantidad, Precio_venta, Fecha_venta)
VALUES(1111,111111,1111,'n','f',3,3,23-12-11);

[ Editar en línea ] [ Editar ] [ Crear código PHP ]

```

Warning: #1264 Out of range value for column 'Fecha_venta' at row 1

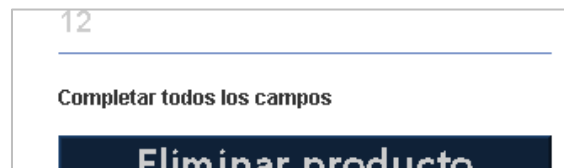
Prueba de Interfaz Salidas

Se introducirá la salida del producto P y proveedor P:



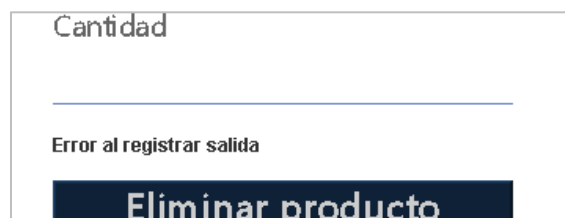
A dialog box with a title bar containing a minus sign and a close button (X). The form inside has the following fields: 'Producto' with value 'P', 'Proveedor' with value 'P', 'Motivo' with value 'ninguno', 'Fecha:' with a date picker showing '3 nov. 2021', and 'Cantidad' with value '12'. Below these fields, it says 'Salida Exitosa' and has a large dark blue button labeled 'Eliminar producto'.

Si hay campos vacíos aparecerá el siguiente mensaje:



A dialog box showing the value '12' in the 'Cantidad' field. Below the field, it says 'Completar todos los campos' and has a large dark blue button labeled 'Eliminar producto'.

Si hay un error en la salida se mostrara lo siguiente:



A dialog box showing the 'Cantidad' field. Below the field, it says 'Error al registrar salida' and has a large dark blue button labeled 'Eliminar producto'.

Código de Salidas

Para salidas se modificó la parte de código para llamar a una nueva clase:

```
if (eliSQL.Salidas(us)) {  
    limpiar();  
}  
else {  
    TXTERROR.setText("Error al registrar salida");  
}  
TXTERROR.setText("Error al registrar salida");
```

Creación de EliminarSQL.java

El código se basa en el funcionamiento de las salidas como en el caso de uso, primeramente, tenemos la parte del código que modifica la tablas inventarios para introducir la cantidad de salida de la venta esta se obtiene del Campo para introducir la cantidad.

```
int cantidad = in.getCantidadSALIDA();
String sql = "UPDATE inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto SET inventario.Salida = inventario.Salida
+ " WHERE inventario.Producto = ? AND compras.Proveedor = ?;";
ps = (PreparedStatement) con.prepareStatement(sql);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();
```

Después es necesario obtenerla Id del producto y la salida, de las tablas inventarios y entradas. Esto para posterior mente insertarlas en las tablas salidas.

```
String sql2 = "SELECT inventario.Id_producto, inventario.Salida ,entradas.Fecha_entrada FROM compras "
+ " INNER JOIN inventario ON compras.Id_producto = inventario.Id_producto "
+ " INNER JOIN entradas ON inventario.Id_producto = entradas.Id_producto "
+ " WHERE inventario.Producto = ? AND compras.Proveedor = ?;";
ps = (PreparedStatement) con.prepareStatement(sql2);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());

rs = ps.executeQuery();
```

Al obtenerlas creamos unas variables(idp,sal) para poder insertar todo en salidas y hacer la inserción

```
if (rs.next()) {
    int idp = rs.getInt("Id_producto");
    int sal = rs.getInt("Salida");
    String fech = rs.getString("Fecha_entrada");
    java.util.Date fech2 = rs.getDate("Fecha_entrada");
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String Y = df.format(fech2);
    int idSalida = in.getId_salidasSALIDA();
    //Fecha salida
    java.util.Date fecha = in.getFecha_salidaSALIDA();
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String X = df.format(fecha);
    String sql3 = "INSERT INTO salidas(Id_salida,Id_producto,Fecha_entrada,Fecha_salida,Motivo,usuario_responsable,Cantidad,Producto)"
+ " VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
    ps = (PreparedStatement) con.prepareStatement(sql3);
    ps.setInt(1, idSalida);
    ps.setInt(2, idp);
    ps.setString(3, fech);
    ps.setString(4, X);
    ps.setString(5, in.getMotivoSALIDA());
    ps.setString(6, in.getUser());
    ps.setInt(7, sal);
    ps.setString(8, in.getProductoSALIDA());
    ps.execute();
    return true;
}
```

Modificación Código Ingresar

En esta parte se creó una nueva clase llamadas EntradasSQL.java para poner correctamente los datos.

Se modifica la tabla inventario con dos variables de cantidad y precio, esto para usarlos dentro de la consulta.

```
int cantidad = in.getCantidadINGRESAR();
int precio = in.getPrecioINGRESAR();
String sql = "UPDATE inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto"
+ " SET inventario.Entrada = inventario.Entrada + " + cantidad + ","
+ " inventario.Cantidad = inventario.Cantidad + " + cantidad + ","
+ " inventario.Precio = " + precio + ""
+ " WHERE inventario.Producto = ? AND compras.Proveedor = ?;";
```

Esto mientras el proveedor y el producto sean iguales.

```
ps = (PreparedStatement) con.prepareStatement(sql);
ps.setString(1, in.getProductoINGRESAR());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();
```

Después es necesario consultar la Id del producto, esto debido a que posterior mente se usara en la siguiente consulta, se usara la tabla inventario y compras. De igual manera el proveedor y el producto sean iguales.

```
//Consulta el id del producto
String sql2 = "SELECT inventario.Id_producto FROM inventario "
            + " INNER JOIN compras ON inventario.Id_producto = compras.Id_producto "
            + " WHERE inventario.Producto = ? AND compras.Proveedor = ? ";
ps = (PreparedStatement) con.prepareStatement(sql2);
ps.setString(1, in.getProductoINGRESAR());
ps.setString(2, in.getProveedorCOMPRA());
rs = ps.executeQuery();
```

Se obtienen y además se modificará fecha obtenida del campo a un formato utilizable en java.

```
if (rs.next()) {
    //Guarda el id en variable
    int idp = rs.getInt("Id_producto");

    System.out.println("Id del producto : " + idp);

    //Fecha de entrada del producto
    java.util.Date fecha = in.getFecha_entradaENTRADA();
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String X = df.format(fecha);
```

Se inserta los datos y las variables obtenidas anteriormente en la tabla entradas.

```
//Genera id entrada
int idEntrada = in.getId_entradaENTRADA();
//Ingresa datos en La tabla Entradas
String sql3 = "insert into entradas (Id_entrada,Id_producto,Fecha_entrada,usuario_responsable,Cantidad) "
            + "VALUES(?, ?, ?, ?, ?)";
ps = (PreparedStatement) con.prepareStatement(sql3);
ps.setInt(1, idEntrada);
ps.setInt(2, idp);
ps.setString(3, X);
ps.setString(4, fnUser);
ps.setInt(5, cantidad);
ps.execute();
```

Consulta la id entrada y el precio para después usarlos en entradas.

```
String sql4 = "SELECT entradas.Id_entrada FROM inventario "
            + " INNER JOIN entradas ON inventario.Id_producto = entradas.Id_producto "
            + " WHERE entradas.Id_producto = " + idp + " ";
ps = (PreparedStatement) con.prepareStatement(sql4);
rs = ps.executeQuery();
if (rs.next()) {
    //Guarda los datos en variables
    int ide = rs.getInt("Id_entrada");
```

Y por último ingresa todo en compras.

```
//Genera id compra
int idcompra = in.getId_compraCOMPRA();
//Ingresar datos en tabla compras
String sql5 = "insert into compras (Id_compra,Id_entrada,Id_producto,Proveedor,Responsable,Cantidad,Precio_compra,fecha_compra) "
            + "VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
ps = (PreparedStatement) con.prepareStatement(sql5);
ps.setInt(1, idcompra);
ps.setInt(2, ide);
ps.setInt(3, idp);
ps.setString(4, in.getProveedorCOMPRA());
ps.setString(5, fnUser);
ps.setInt(6, cantidad);
ps.setInt(7, precio);
ps.setString(8, X);
ps.execute();
```

Modificación de Código Eliminar

Se modifiko la clase EliminarSQL.java siguiendo el caso de uso.

Primera mente el código obtiene la Id del producto, el producto y la cantidad a eliminar.

```
// inventario.compras inner
String sql2 = "SELECT inventario.Id_producto, inventario.Producto, inventario.Cantidad ,entradas.Fecha_entrada "
+ " FROM compras "
+ " INNER JOIN inventario ON compras.Id_producto = inventario.Id_producto "
+ " INNER JOIN entradas ON inventario.Id_producto = entradas.Id_producto "
+ " WHERE inventario.Producto = ? AND compras.Proveedor = ?";
ps = (PreparedStatement) con.prepareStatement(sql2);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
rs = ps.executeQuery();
```

Y obtiene los datos y los introduce a variables.

```
if (rs.next()) {
    int idp = rs.getInt("Id_producto");
    String prod = rs.getString("Producto");
    int sal = rs.getInt("Cantidad");
    String fech = rs.getString("Fecha_entrada");

    System.out.println("Id producto " + idp + " salida, " + sal + " fecha: " + fech + " prdc " + prod);

    int idSalida = in.getId_salidaSALIDA();
    String motivo = "eliminacion";
    //Fecha salida
    java.util.Date fecha = in.getFecha_salidaSALIDA();
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String X = df.format(fecha);
}
```

Al obtener las variables y valores se realiza una inserción en la tabla salidas.

```
String sql3 = "INSERT INTO salidas(Id_salida,Id_producto, Producto, Fecha_entrada,Fecha_salida,Motivo,usuario_responsable,Cantidad)"
+ " VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
ps = (PreparedStatement) con.prepareStatement(sql3);
ps.setInt(1, idSalida);
ps.setInt(2, idp);
ps.setString(3, prod);
ps.setString(4, fech);
ps.setString(5, X);
ps.setString(6, motivo);
ps.setString(7, username);
ps.setInt(8, sal);
ps.execute();
```

Por último, borra el producto si el nombre y el proveedor son iguales.

```
String delete = "DELETE inventario FROM inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto "
+ " WHERE inventario.Producto = ? and compras.Proveedor = ?";
ps = (PreparedStatement) con.prepareStatement(delete);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();
return true;
```

Modificación Código Ventas

Se modifiko la clase ventasSQL para poder cumplir con el caso de uso.

Primeramente, se obtiene la cantidad de Venta.java esto para poder ponerlo en una variable y posteriormente usarlo en la modificación de la tabla inventario.

```
int cantidad = in.getCantidadVENTA();
```

La cantidad obtenida dentro se suma a el campo Salida de inventario y se resta la cantidad de el campo cantidad igualmente de inventario.

```
String sql = "UPDATE inventario INNER JOIN compras ON inventario.Id_producto = compras.Id_producto SET "
+ " inventario.Salida = inventario.Salida+ " + cantidad + ", inventario.Cantidad = inventario.Cantidad-"
+ cantidad + " WHERE inventario.Producto = ? AND compras.Proveedor = ?";
ps = (PreparedStatement) con.prepareStatement(sql);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());
ps.execute();
```

Posteriormente tenemos que obtener la Id del producto y la salida para usarlos en otra consulta o modificación.

```
String sql2 = "SELECT inventario.Id_producto,inventario.Salida ,entradas.Fecha_entrada FROM compras "
+ "INNER JOIN inventario ON compras.Id_producto = inventario.Id_producto"
+ " INNER JOIN entradas ON inventario.Id_producto = entradas.Id_producto WHERE inventario.Producto = ? AND compras.Proveedor = ?:";
ps = (PreparedStatement) con.prepareStatement(sql2);
ps.setString(1, in.getProductoSALIDA());
ps.setString(2, in.getProveedorCOMPRA());

rs = ps.executeQuery();
```

Se obtienen los valores y se hace la variable Id_salida para usarla después.

```
if (rs.next()) {
    int idp = rs.getInt("Id_producto");
    int sal = rs.getInt("Salida");
    String fech = rs.getString("Fecha_entrada");

    int idSalida = in.getId_salidaSALIDA();
    //Fecha salida
    java.util.Date fecha = in.getFecha_ventaVENTA();
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String X = df.format(fecha);
```

Se insertan los valores en la tabla salidas con los valores preparados para introducirlos.

```
String sql3 = "insert into salidas (Id_salida,Id_producto,Fecha_entrada,Fecha_salida,Motivo,usuario_responsable,Cantidad, Producto) "
+ "VALUES(?,?,?,?,?,?,?)";
ps = (PreparedStatement) con.prepareStatement(sql3);
ps.setInt(1, idSalida);
ps.setInt(2, idp);
ps.setString(3, fech);
ps.setString(4, X);
ps.setString(5, motivo);
ps.setString(6, Usuario);
ps.setInt(7, cantidad);
ps.setString(8, in.getProductoSALIDA());
ps.execute();
```

Obtenemos la Id de salida para poder ponerlo en la siguiente consulta.

```
String sql3 = "insert into salidas (Id_salida,Id_producto,Fecha_entrada,Fecha_salida,Motivo,usuario_responsable,Cantidad, Producto) "
+ "VALUES(?,?,?,?,?,?,?)";
ps = (PreparedStatement) con.prepareStatement(sql3);
ps.setInt(1, idSalida);
ps.setInt(2, idp);
ps.setString(3, fech);
ps.setString(4, X);
ps.setString(5, motivo);
ps.setString(6, Usuario);
ps.setInt(7, cantidad);
ps.setString(8, in.getProductoSALIDA());
ps.execute();
```

Por último, insertamos en ventas todos los valores obtenidos en la tabla ventas para poner fin al registro de las ventas.

```
int idvent = in.getId_ventaVENTA();
String sql4 = "insert into ventas (Id_venta, Id_salida, Id_producto, Comprador, Responsable, Cantidad, Precio_venta, Fecha_venta) VALUES(?,?,?,?,?,?,?,?)";
ps = (PreparedStatement) con.prepareStatement(sql4);
ps.setInt(1, idvent);
ps.setInt(2, ids);
ps.setInt(3, idp);
ps.setString(4, in.getCompradorVENTA());
ps.setString(5, Usuario);
ps.setInt(6, cantidad);
ps.setInt(7, in.getPrecio_ventaVENTA());
ps.setString(8, X);

ps.execute();
```

Módulo de Tablas a Buscador y Filtro

Historial de revisiones:

Revisión	Responsable(s)	Descripción	Fecha
Módulo de tablas a buscador y filtro.	Araceli	Pruebas exploratorias	18/11/2021
Módulo de tablas a buscador y filtro.	Victoria	Pruebas de Casos de Uso	20/11/2021
Módulo de tablas a buscador y filtro.	Araceli	Pruebas de caja negra	22/11/2021
Módulo de tablas a buscador y filtro.	Victoria	Pruebas de caja blanca	24/11/2021

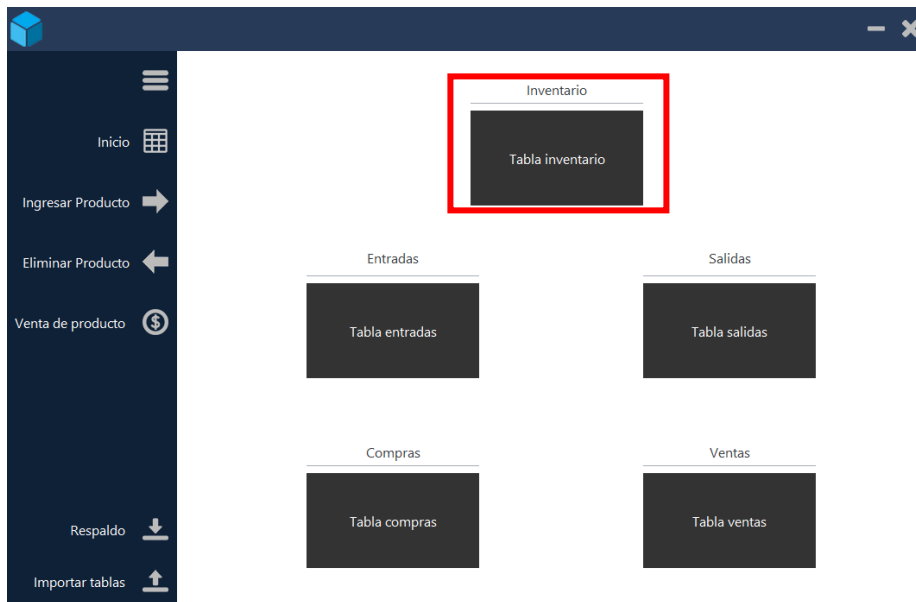
Pruebas exploratorias

1. Revisión general del modulo

Para acceder a las tablas del inventario seleccionamos inicio. Podemos observar que se encuentran las tablas de entradas, salidas, compras, ventas e inventario general.



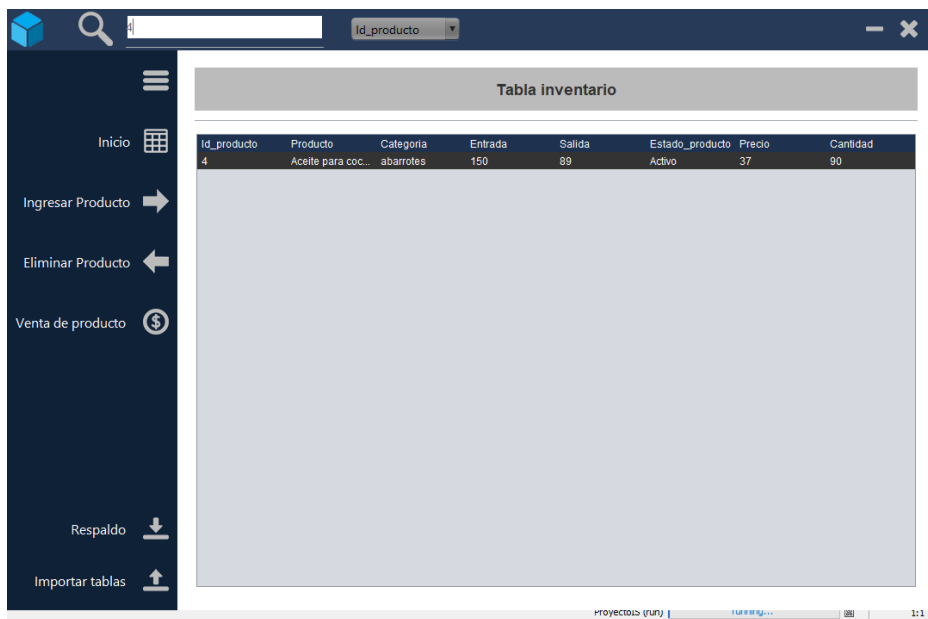
Al seleccionar cualquiera de las tablas nos despliega la información que contiene cada una. En este caso seleccionamos la tabla de inventario.



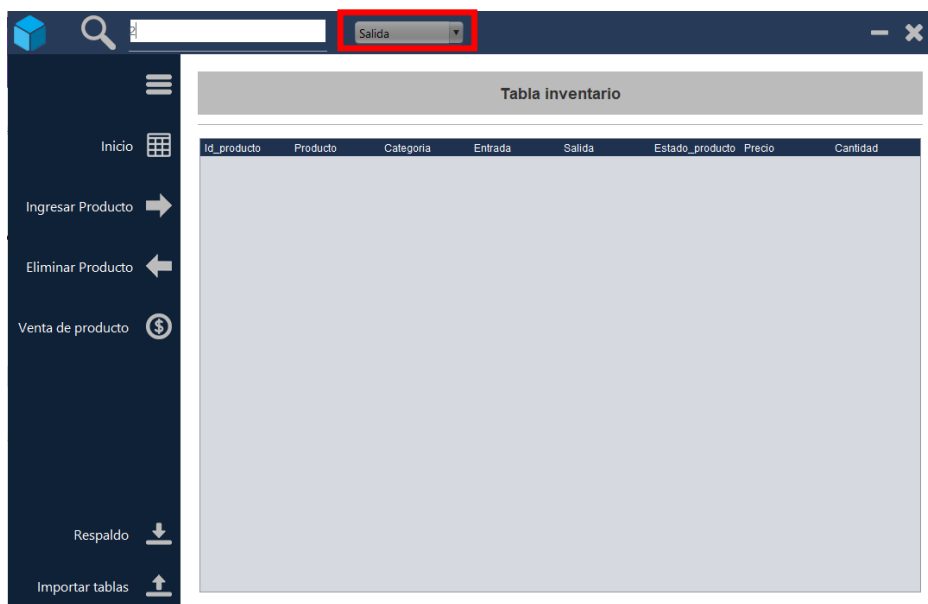
Se muestran las columnas que contienen la información agregada desde la base de datos.

Tabla inventario							
Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
1	Detergente	limpieza	200	80	Activo	12	65
2	Shampoo	higiene personal	10	12		9	19
3	Desinfectante ...	limpieza	500	250	Activo	25	65
4	Aceite para coc...	abarrotes	150	89	Activo	37	90

Para consultar algún registro, accedemos a la barra de búsqueda e ingresamos el id del producto. Ingresamos el id_producto 4 y nos despliega solamente el dato solicitado.



Al usar el filtro para realizar una búsqueda de solo un determinado dato de una columna, no se despliega ninguna información. Puesto que solo funciona con el id_producto.



2. Detección de errores

Fallos detectados	Beneficios obtenidos
<ul style="list-style-type: none"> No realiza la búsqueda al momento de ingresar un dato distinto que no sea el id de algún producto. 	<ul style="list-style-type: none"> Permite buscar los datos con el id específico.
<ul style="list-style-type: none"> No se puede realizar búsquedas a través del filtro. 	<ul style="list-style-type: none"> Conecta correctamente a la base de datos.

	<ul style="list-style-type: none"> Se despliegan los datos correctamente.
--	--

3. Posibles soluciones de los errores:
 - Arreglar el filtro para que funcione.
 - Arreglar la barra de búsqueda.

Pruebas de caso de uso

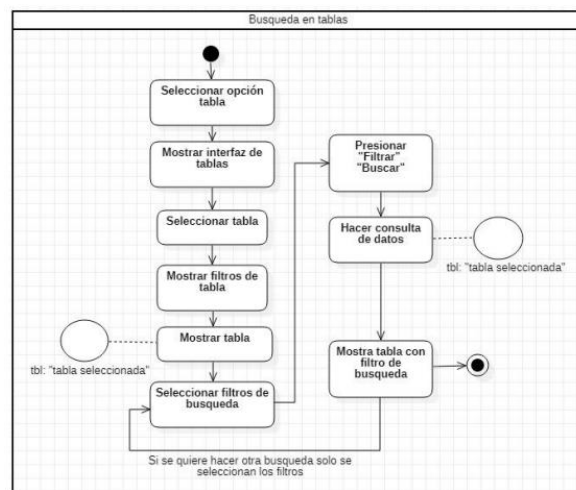
1. Revisión general del modulo

Búsqueda en tablas

El siguiente diagrama de actividad se establece la forma en que se realiza la consulta de algún dato contenido en las tablas.

El proceso de consulta permitía el especificar algún dato y que al momento de consultarlo solo desplegara la información solicitada.

Si había algún error en la búsqueda, no se desplegaba ninguna información y se mostraba la tabla vacía.



2. Detección de errores

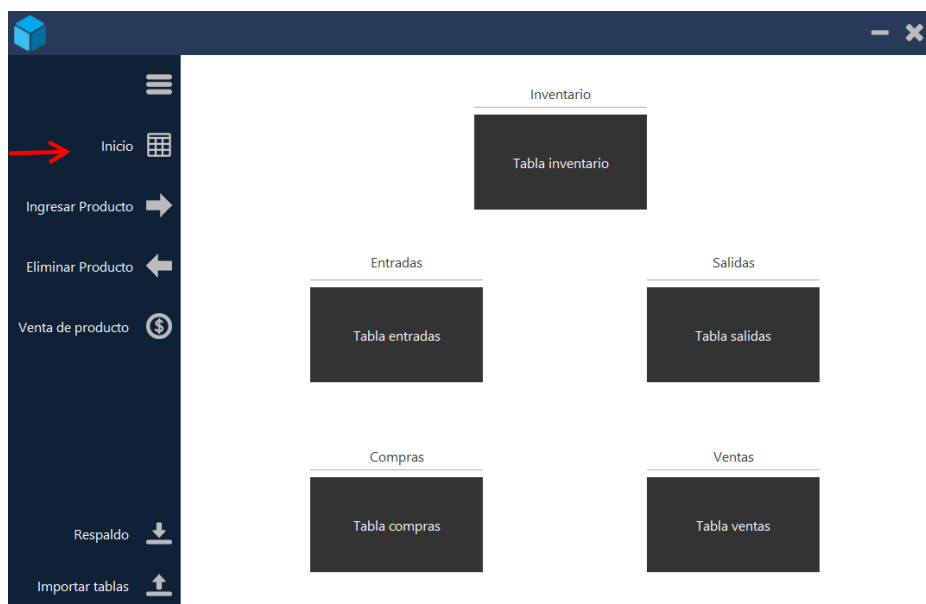
Fallos detectados	Beneficios obtenidos
<ul style="list-style-type: none"> No realiza lo especificado en el diagrama la interfaz. 	<ul style="list-style-type: none"> No hay cambios dentro del diagrama de actividades.

3. Posibles soluciones de los errores:

El diagrama de actividades es correcto en base a su proceso.

Pruebas de caja negra

Prueba 1: Ingresar a las tablas desde el menú de inicio, a continuación se muestra la interfaz de tablas.



Prueba 2:

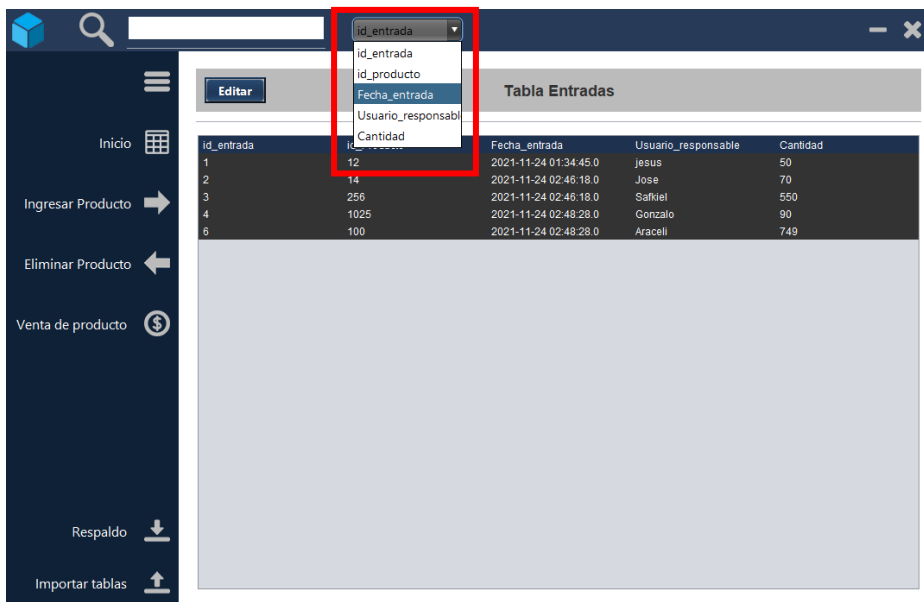
Seleccionamos cualquiera de las tablas para acceder a los datos que contiene cada una. Se muestran los datos desplegados que fueron agregados con anterioridad desde la base de datos.

The screenshot shows the application interface with the 'Tabla Entradas' table selected and highlighted with a red border. The table has 5 columns: 'id_entrada', 'id_Producto', 'Fecha_entrada', 'Usuario_responsable', and 'Cantidad'. The data is as follows:

id_entrada	id_Producto	Fecha_entrada	Usuario_responsable	Cantidad
1	12	2021-11-24 01:34:45.0	jesus	50
2	14	2021-11-24 02:46:18.0	Jose	70
3	256	2021-11-24 02:46:18.0	Safriel	550
4	1025	2021-11-24 02:48:28.0	Gonzalo	90
6	100	2021-11-24 02:48:28.0	Araceli	749

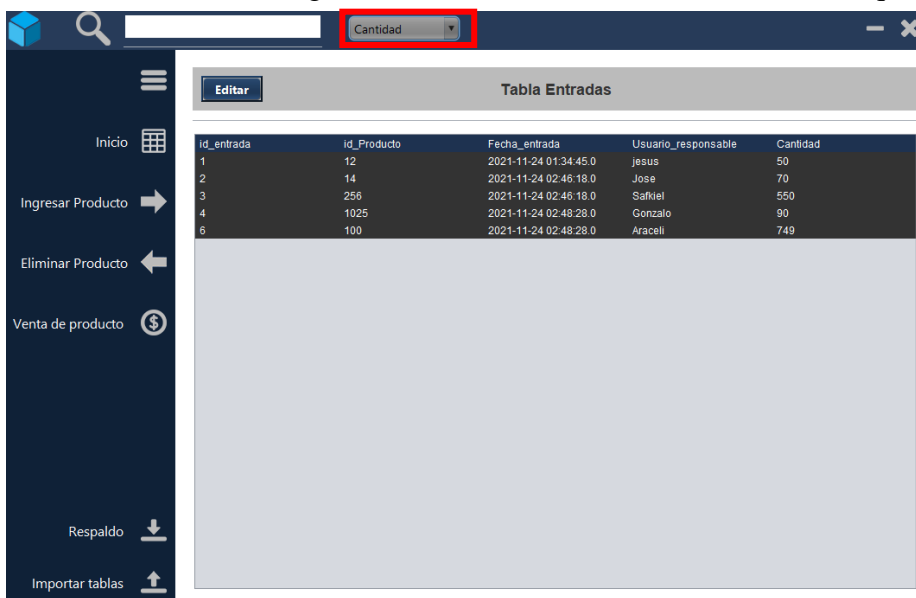
Prueba 3:

Seleccionamos la lista desplegable la cual contiene los títulos de cada apartado de la tabla.



Seleccionamos un filtro para observa algún cambio.

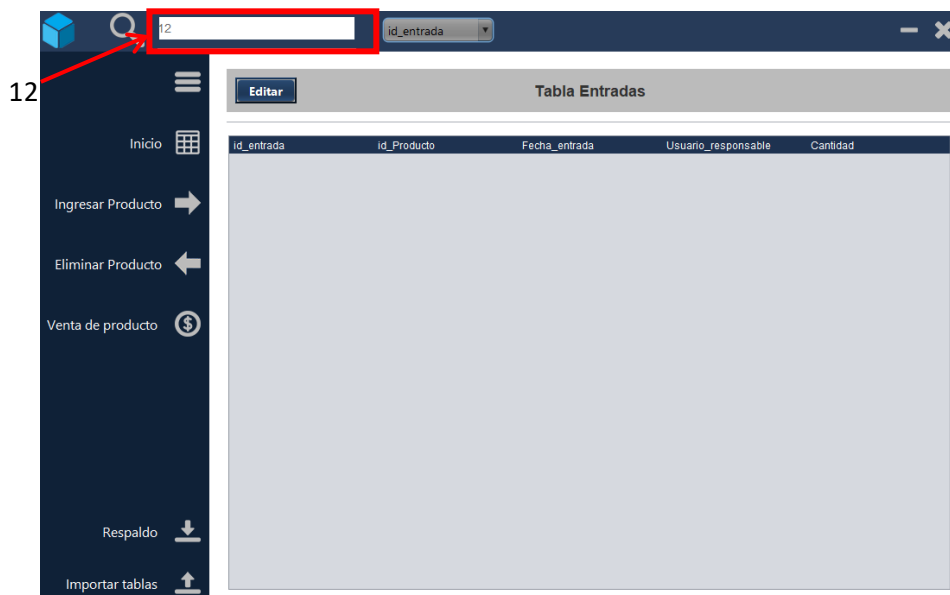
La tabla no muestra ningún cambio al momento de seleccionar cualquier filtro.



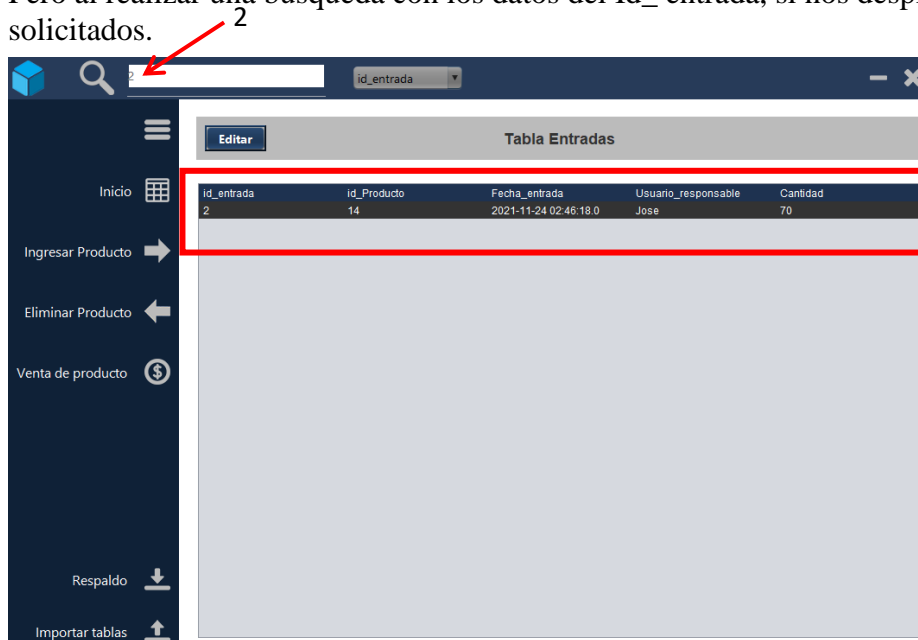
Prueba 4:

Para realizar una consulta de algún dato, nos ubicamos en la barra de búsqueda e ingresamos el dato solicitado. Ejemplo: 12, contenido en la barra Id_Producto.

Damos clic en el botón con forma de lupa.



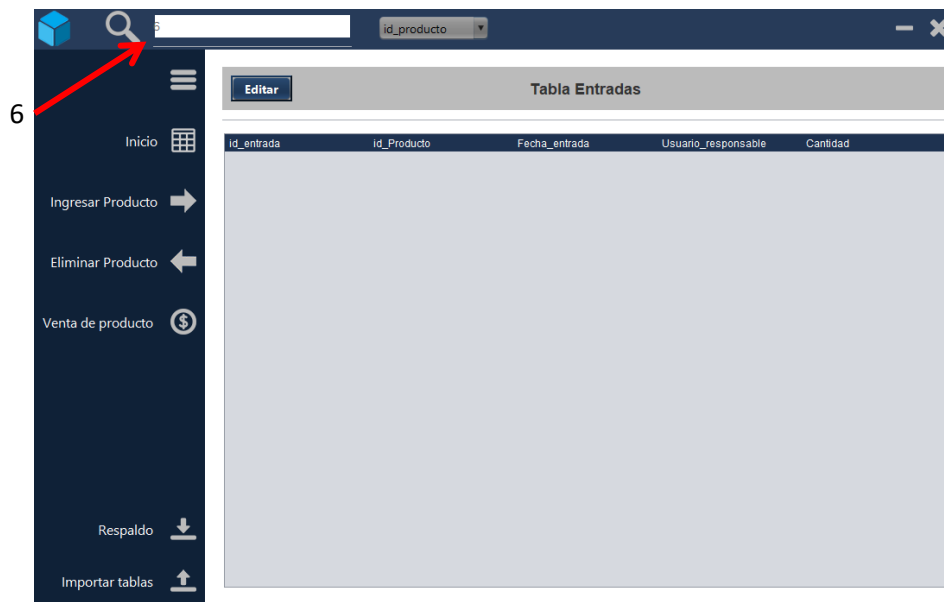
Pero al realizar una búsqueda con los datos del Id_ entrada, si nos despliega los datos solicitados.



Prueba 5:

Realizamos una consulta con la barra de búsqueda y el filtro.

Se observa que al momento de usar un dato del Id_ entrada y un filtro distinto al Id_ entrada, no despliega ningún dato.



Fallos detectados	Beneficios obtenidos
<ul style="list-style-type: none"> • EL filtro no permite realizar búsquedas diferentes de Id_ entrada. • La barra de búsqueda no permite realizar consultas diferentes de Id_ entrada. 	<ul style="list-style-type: none"> • Se realizaran cambios luego de haber descubierto errores al momento de realizar consulta. • Hacer que el filtro funcione correctamente.

Pruebas de caja blanca

Tablas a buscador y filtro

Prueba de intentos:

Conexión de la base de datos.

```
PreparedStatement ps = null;
ResultSet rs = null;
ConexionSQL conn = new ConexionSQL();
Connection con = conn.getConexion();
```

Se mandan llamar desde la base de datos los registros de las tablas.

```

setLocationRelativeTo(null);
desplace = new Desface();

try {
    DefaultTableModel modelo = new DefaultTableModel();
    jtInventario.setModel(modelo);

    PreparedStatement ps = null;
    ResultSet rs = null;
    ConexionSQL conn = new ConexionSQL();
    Connection con = conn.getConexion();

    String sql = "Select id_producto, producto, categoria, entrada, salida, estado_producto, precio, cantidad FROM inventario";
    ps = con.prepareStatement(sql);
    rs = ps.executeQuery();

    ResultSetMetaData rsMd = rs.getMetaData();
    int cantidadColumnas = rsMd.getColumnCount();

    modelo.addColumn("Id_producto");
    modelo.addColumn("Producto");
    modelo.addColumn("Categoria");
    modelo.addColumn("Entrada");
    modelo.addColumn("Salida");
    modelo.addColumn("Estado_producto");
    modelo.addColumn("Precio");
    modelo.addColumn("Cantidad");

    while (rs.next()) {
        Object[] fila = new Object[cantidadColumnas];
    }
}

```

Se muestra la funcionalidad del botón de búsqueda que permite realizar las consultas de los datos de las tablas.

```

private void buscar() {
    String campo = txtCampo.getText();

    String filtro = CBoxFiltro.getSelectedItem().toString();

    System.out.println(""+filtro);
    String where = "";

    if (!"".equals(campo)) { //si el campo no esta vacio
        where = " WHERE "+filtro+"=" + campo + " ";
    }

    try {
        DefaultTableModel modelo = new DefaultTableModel();
        jtInventario.setModel(modelo);

        PreparedStatement ps = null;
        ResultSet rs = null;
        ConexionSQL conn = new ConexionSQL();
        Connection con = conn.getConexion();

        String sql = "Select id_producto, producto, categoria, entrada, salida, estado_producto, precio, cantidad FROM inventario "
            + where;
        System.out.println(sql);
        ps = con.prepareStatement(sql);
        rs = ps.executeQuery();

        ResultSetMetaData rsMd = rs.getMetaData();
        int cantidadColumnas = rsMd.getColumnCount();
    }
}

```

Modificación:

Agregamos un comando el cual permite el funcionamiento del filtro.

```

private void CBoxFiltroActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String[] catalogo = {" "};
    JComboBox<String> productos = new JComboBox<String>(catalogo);
}

```

Podemos observar que las consultas funcionan correctamente al momento de utilizar un filtro.

Shampoo

Producto

Tabla inventario

Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
2	Shampoo	higiene personal	10	12		9	19

Inicio

Ingresar Producto

Eliminar Producto

Venta de producto

Respaldo

Importar tablas

Si colocamos un dato no contenido en la columna seleccionada en el filtro, no nos muestra ningún dato.

30

Producto

Tabla inventario

Id_producto	Producto	Categoria	Entrada	Salida	Estado_producto	Precio	Cantidad
-------------	----------	-----------	---------	--------	-----------------	--------	----------

Inicio

Ingresar Producto

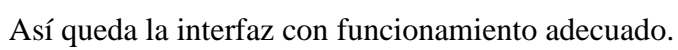
Eliminar Producto

Venta de producto

Respaldo

Importar tablas

Probamos con otra tabla para comprobar la funcionalidad de la barra de búsqueda y filtros.
Ejemplo: Safkiel y filtro Usuario_responsable



Así queda la interfaz con funcionamiento adecuado.

Módulo de Exportar BDD

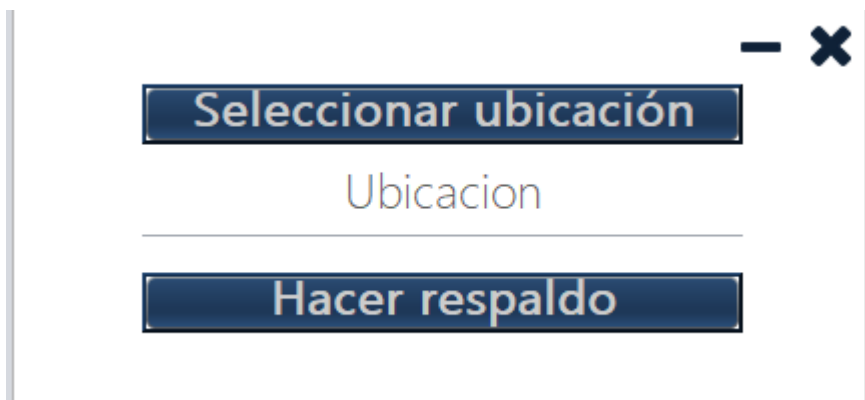
Historial de revisiones:

Revisión	Responsable(s)	Descripción	Fecha
Exportar BDD	Adal Bautista E7	Pruebas exploratorias	17/11/2021
Exportar BDD	Adal Bautista E7	Pruebas de Casos de Uso	18/11/2021
Importar BDD	Adal Bautista E7	Pruebas exploratorias	19/11/2021
Importar BDD	Adal Bautista E7	Pruebas de Casos de Uso	21/11/2021

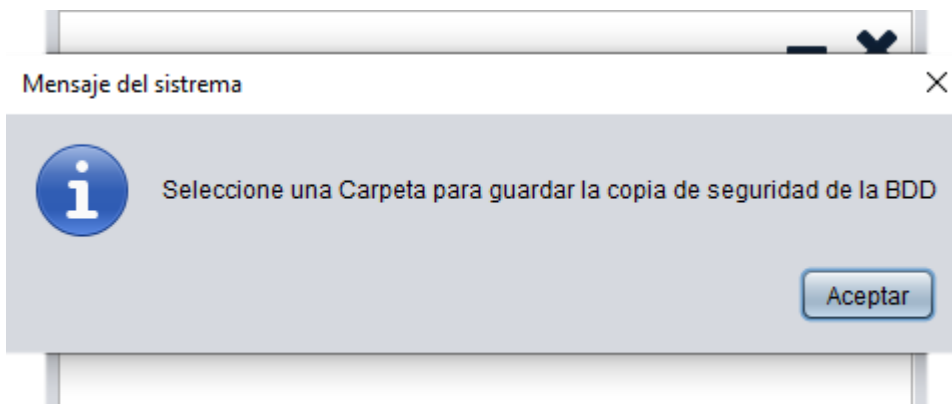
Pruebas Exploratorias Revisión 1

1. Revisión General Del Módulo.

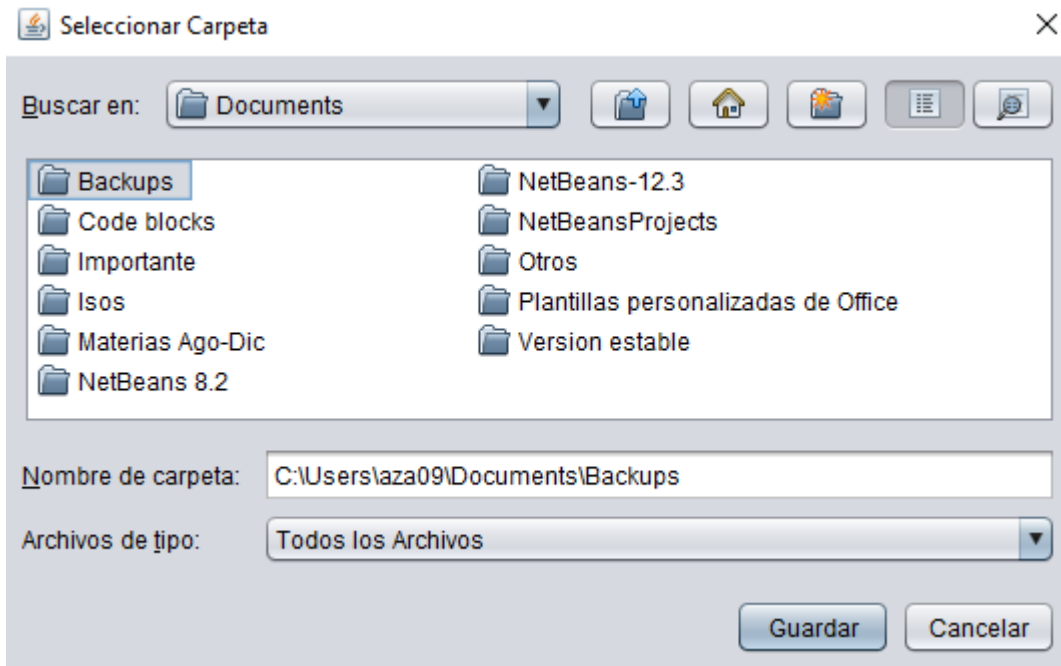
El programa despliega la interfaz de **Exportar Base De Datos**. Se tiene seleccionar el lugar donde se guardará el respaldo de la base de datos.



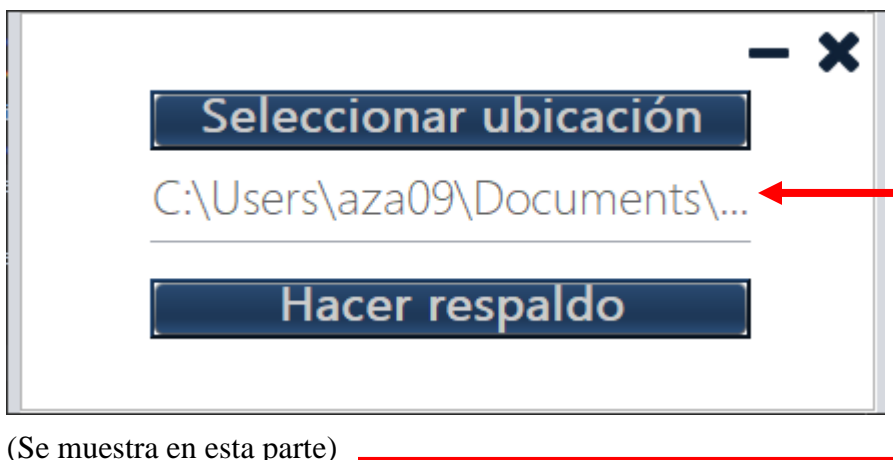
En caso de no seleccionar una carpeta el programa desplegará un mensaje para informar al usuario que no ha seleccionado una carpeta



Al momento en que el usuario seleccione una carpeta donde guardara la copia de seguridad

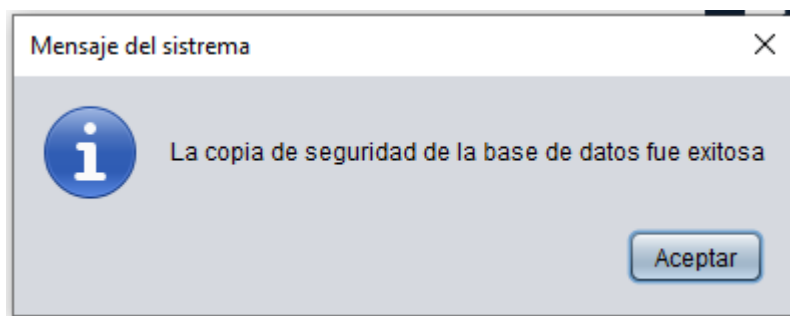


El sistema mostrara la ruta de dicha carpeta



(Se muestra en esta parte)

Al momento en que el usuario haga clic sobre “Hacer Respaldo” el sistema genera una copia de la base de datos y despliega un mensaje informando al usuario si se ha guardado la base de datos o no (en este caso se guardó correctamente)



1. Detección De Errores

Fallos detectados	Beneficios obtenidos
El programa muestra el mensaje de éxito aun cuando no se ha seleccionado una carpeta para hacer el respaldo	Al encontrar este error del mal despliegue del mensaje podemos solucionarlo para que el sistema funcione correctamente

2. Posibles Soluciones A Errores O Cambios

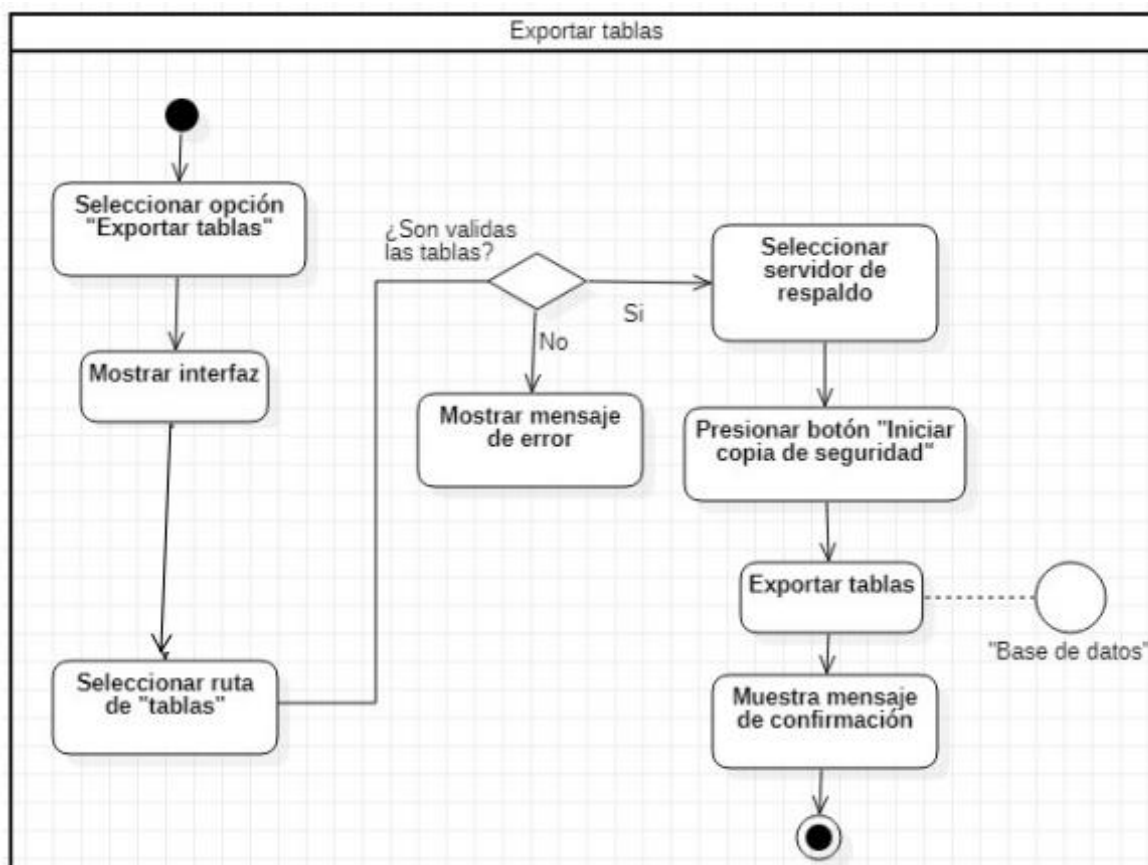
- Examinar el código para cambiar el Else y así poder solucionar el problema
- Cambiar el modo en que se despliegan los mensajes del sistema

Pruebas De Caso De Uso

1. Revisión General Del Módulo

Copia De Seguridad

El siguiente diagrama se estableció inicialmente para el sistema que genera la base de datos. Como se puede observar el sistema sigue los pasos indicados y no necesita algún cambio



2. Detección De Errores

Fallos detectados	Beneficios obtenidos
Ninguno	No se requiere modificación

3. Posibles Soluciones O Cambios.

- El sistema no requiere correcciones

Módulo de Importar BDD

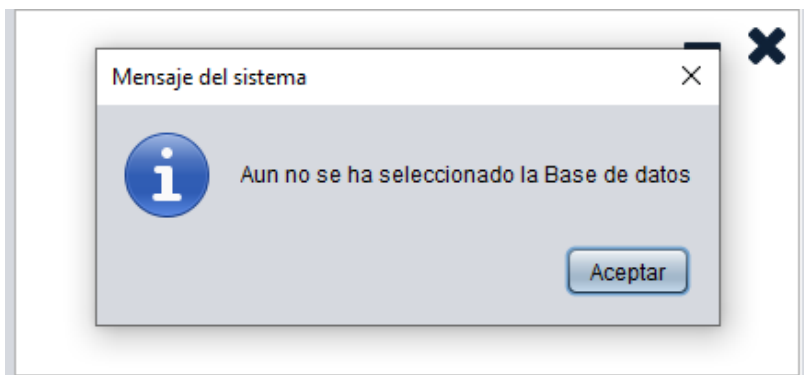
Pruebas Exploratorias Revisión 1

2. Revisión General Del Módulo.

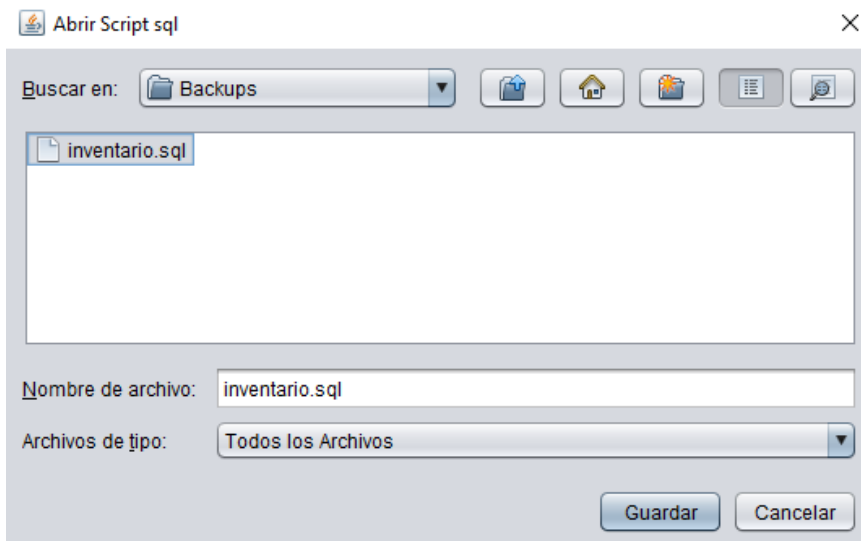
El programa despliega la interfaz de **Importar Base De Datos**. Se tiene seleccionar la carpeta donde se guardó el respaldo de la base de datos.



En caso de no seleccionar una Base de datos el programa desplegará un mensaje para informar al usuario.



Al momento en que el usuario seleccione el archivo script donde este la copia de seguridad

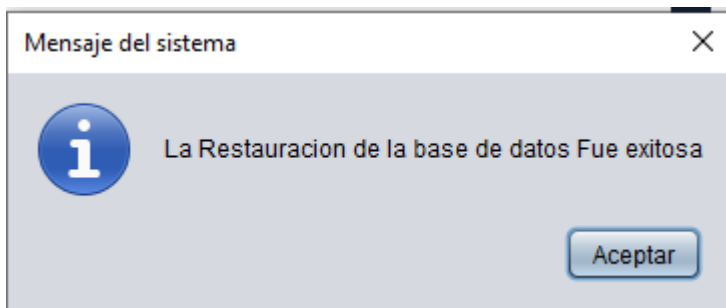


El sistema mostrara la ruta de dicho archivo



(Se muestra en esta parte)

Al momento en que el usuario haga clic sobre “Importar tablas” el sistema restaurara la copia de seguridad de la base de datos y despliega un mensaje informando al usuario si se restauró la base de datos o no (en este caso se restauró correctamente)



3. Detección De Errores

Fallos detectados	Beneficios obtenidos
El programa muestra el mensaje de éxito aun cuando no se ha seleccionado un archivo para	Al encontrar este error del mal despliegue del mensaje podemos solucionarlo para que el

hacer la restauración	sistema funcione correctamente
-----------------------	--------------------------------

4. Posibles Soluciones A Errores O Cambios

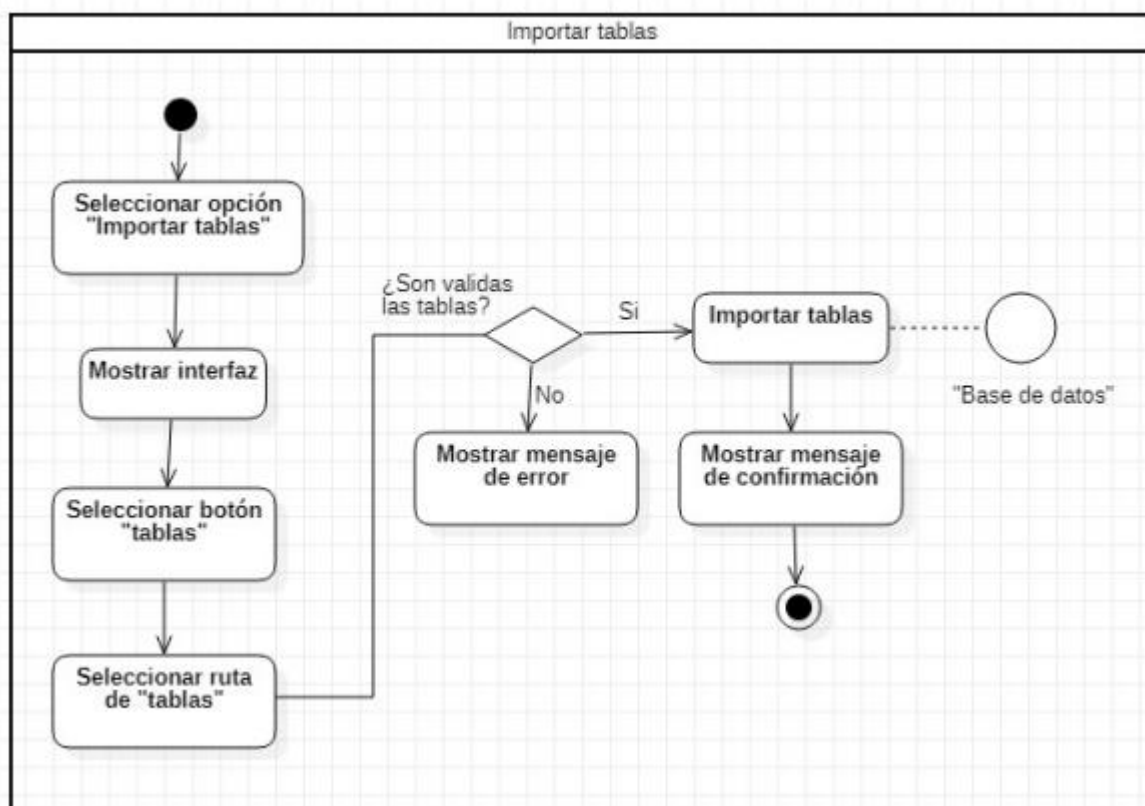
- Examinar el código para cambiar el Else y así poder solucionar el problema
- Cambiar el modo en que se despliegan los mensajes del sistema

Pruebas De Caso De Uso

4. Revisión General Del Módulo

Copia De Seguridad

El siguiente diagrama se estableció inicialmente para el sistema que genera la base de datos. Como se puede observar el sistema sigue los pasos indicados y no necesita algún cambio



5. Detección De Errores

Fallos detectados	Beneficios obtenidos
Ninguno	No se requiere modificación

6. Posibles Soluciones O Cambios.

El sistema no requiere correcciones

Pruebas del sistema completo

Pruebas de rendimiento: son las pruebas que se realizan, desde una perspectiva, para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo.

Prueba de volumen: se realiza para verificar el rendimiento de la base de datos frente a un gran volumen de datos en la base de datos. La prueba de carga se realiza cambiando las cargas del usuario para los recursos y verificando el rendimiento de los recursos.

Pruebas de esfuerzo encuentran fallas en la aplicación que simplemente no se mostrarían en una sola instancia

Prueba de usabilidad por parte del usuario es una técnica usada en el diseño de interacciones centrado en el usuario para evaluar un producto mediante pruebas con los usuarios mismos