



UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

FACULTAD DE INGENIERIA

EN SISTEMAS DE LA INFORMACIÓN

CURSO: Programación II.

Ing. Jhonny Morales.

TEMA: Proyecto Final.

Integrantes:

Carnet:

Cristian Alexander Claudio del valle

0904 – 19 - 4013

Jeisson Alejandro López Hernández

0904 - 19 - 4766

Johanan Uriel Vásquez López

0904 - 21 - 6857

Jhonatan Emanuel Tebalán García

4490 - 20 – 940

Dicmar Alexander Joaquin López

0904 - 20 - 26899

Huehuetenango, octubre de 2022.

Índice.

Introducción.....	I
1. Proceso de elaboración del software	1
2. Tecnologías utilizadas.....	7
2.1. Gliffy.....	8
2.2. Java Netbean	9
2.3. My SQL.....	10
3. Diagramas	10
3.1.Diagrama de Clases	10
3.2.Diagrama Base de Datos.....	10
4. Manual de usuario	11
Conclusiones	12

I. Introducción.

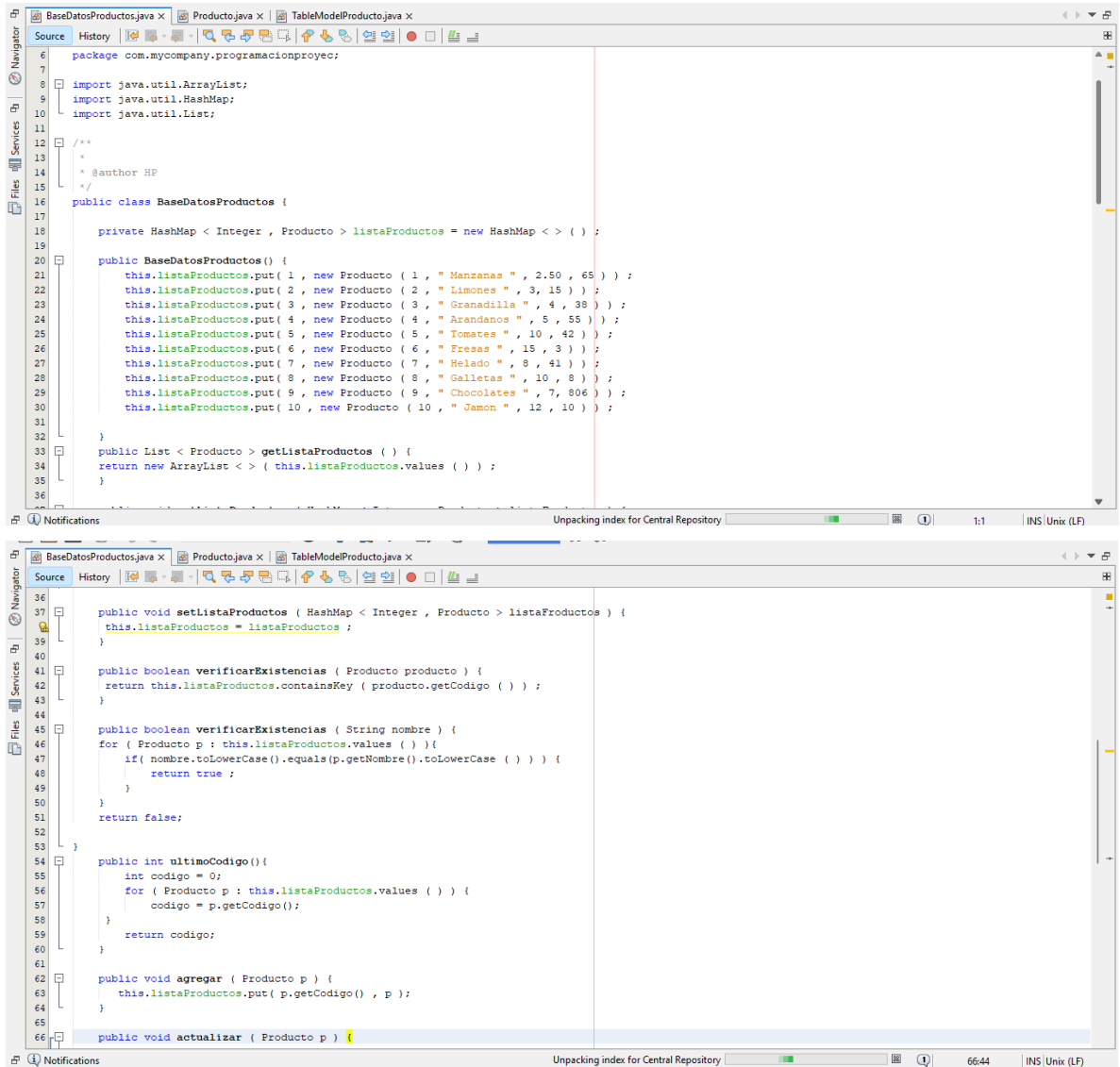
En la programación hay inmensas funciones, así como también muchos procesos que se pueden realizar y dan grandes resultados tanto al programador como al usuario, de esta manera presentamos el proyecto de final de curso, que, aunque no se base en muchas funciones de las herramientas que se utilizaron, tiene una estructura que es en base a la programación orientada a objetos (POO).

El programa consiste en realizar un inventario de productos perecederos en donde podamos introducir productos eliminar, el programa en si tiene una función la cual lleva una estructura que consiste en almacenar datos, así también como crear nuevos datos, poder eliminarlos y al mismo tiempo actualizarlos.

Su función fue programada con todo lo referente a lo que se vio en clases así de la misma manera, dejamos en claro que el software que se creo puede implementar nuevas funciones y poder optar por una actualización de un nivel superior.

1. Proceso de elaboración del software.

– CRUD



```
package com.mycompany.programacionproyec;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 *
 * @author HP
 */
public class BaseDatosProductos {

    private HashMap < Integer , Producto > listaProductos = new HashMap < > ( ) ;

    public BaseDatosProductos() {
        this.listaProductos.put( 1 , new Producto ( 1 , " Manzanas " , 2,50 , 65 ) ) ;
        this.listaProductos.put( 2 , new Producto ( 2 , " Limones " , 3,15 ) ) ;
        this.listaProductos.put( 3 , new Producto ( 3 , " Granadilla " , 4 , 38 ) ) ;
        this.listaProductos.put( 4 , new Producto ( 4 , " Arandanos " , 5 , 55 ) ) ;
        this.listaProductos.put( 5 , new Producto ( 5 , " Tomates " , 10 , 42 ) ) ;
        this.listaProductos.put( 6 , new Producto ( 6 , " Fresas " , 15 , 3 ) ) ;
        this.listaProductos.put( 7 , new Producto ( 7 , " Helado " , 8 , 41 ) ) ;
        this.listaProductos.put( 8 , new Producto ( 8 , " Galletas " , 10 , 8 ) ) ;
        this.listaProductos.put( 9 , new Producto ( 9 , " Chocolates " , 7 , 806 ) ) ;
        this.listaProductos.put( 10 , new Producto ( 10 , " Jamon " , 12 , 10 ) ) ;
    }

    public List < Producto > getListaProductos ( ) {
        return new ArrayList < > ( this.listaProductos.values ( ) ) ;
    }

    public void setListaProductos ( HashMap < Integer , Producto > listaProductos ) {
        this.listaProductos = listaProductos ;
    }

    public boolean verificarExistencias ( Producto producto ) {
        return this.listaProductos.containsKey ( producto.getCodigo ( ) ) ;
    }

    public boolean verificarExistencias ( String nombre ) {
        for ( Producto p : this.listaProductos.values ( ) ) {
            if ( nombre.toLowerCase().equals(p.getNombre().toLowerCase ( ) ) ) {
                return true ;
            }
        }
        return false;
    }

    public int ultimoCodigo(){
        int codigo = 0;
        for ( Producto p : this.listaProductos.values ( ) ) {
            codigo = p.getCodigo();
        }
        return codigo;
    }

    public void agregar ( Producto p ) {
        this.listaProductos.put( p.getCodigo() , p );
    }

    public void actualizar ( Producto p ) { }
```

This screenshot shows the implementation of several methods in the `TableModelProducto.java` file. The methods include `actualizar`, `borrar`, `generarInforme`, and `obtenerMayores`. The `actualizar` method is currently selected and highlighted. The IDE interface includes a Navigator on the left, a Source editor in the center, and a status bar at the bottom.

```
66 public void actualizar ( Producto p ) {
67
68 }
69
70 public void borrar ( Producto p ) {
71     this.listaProductos . remove ( p.getCodigo ( ) ) ;
72 }
73
74 public String generarInforme ( ) {
75     List<Producto> listaM = obtenerMayores();
76     return listaM.get ( 0 ) .getNombre ( ) + " " + listaM.get ( 1 ) .getNombre ( ) + " " + listaM.get ( 2 ) .getNombre ( ) ;
77 }
78
79 private List<Producto> obtenerMayores ( ) {
80     List<Producto> lista = new ArrayList<>(this.listaProductos.values());
81     List<Producto> listaMayores = new ArrayList<>();
82     for (int i = 0; i < 3; i++) {
83         Producto p = new Producto();
84         for (Producto pTemp : lista){
85             if (pTemp.getPrecio() > p.getPrecio()){
86                 p = pTemp;
87             }
88         }
89         listaMayores.add(p);
90         lista.remove(p);
91     }
92     return listaMayores;
93 }
94
95
96
97
98 } //Final
99
```

- CLASES:

This screenshot shows the definition of the `Producto` class in the `Producto.java` file. The class includes private attributes for `codigo`, `nombre`, `precio`, and `inventario`. It also defines a constructor with parameters, a default constructor, and getter/setter methods for the `codigo` attribute. The IDE interface is similar to the previous one, with a Navigator, Source editor, and status bar.

```
6 package com.myccompany.programacionproyec;
7
8 /**
9  *
10  * @author HP
11  */
12 public class Producto {
13
14     private int codigo;
15     private String nombre;
16     private double precio ;
17     private int inventario;
18
19     public Producto(int codigo, String nombre, double precio, int inventario) {
20         this.codigo = codigo;
21         this.nombre = nombre;
22         this.precio = precio;
23         this.inventario = inventario;
24     }
25
26     public Producto() {
27     }
28
29     public int getCodigo() {
30         return codigo;
31     }
32
33     public void setCodigo(int codigo) {
34         this.codigo = codigo;
35     }
36
37 }
```

This screenshot shows the implementation of the `Producto` class in an IDE. The class has several methods for managing product data:

```
36 public String getNombre() {
37     return nombre;
38 }
39
40 public void setNombre(String nombre) {
41     this.nombre = nombre;
42 }
43
44 public double getPrecio() {
45     return precio;
46 }
47
48 public void setPrecio(double precio) {
49     this.precio = precio;
50 }
51
52 public int getInventario() {
53     return inventario;
54 }
55
56 public void setInventario(int inventario) {
57     this.inventario = inventario;
58 }
59
60
61
62
63
64 }
```

The status bar at the bottom indicates "Unpacking index for Central Repository" and "INS| Unix (LF)".

This screenshot shows the implementation of the `TableModelProducto` class, which extends `AbstractTableModel`. It includes package declarations, imports, and methods for row and column counts:

```
6 package com.mycompany.programacionproyec;
7
8 import java.util.ArrayList;
9 import java.util.List;
10 import javax.swing.table.AbstractTableModel;
11
12 /**
13  *
14  * @author HP
15  */
16 public class TableModelProducto extends AbstractTableModel{
17
18     private String[] columns = {"Codigo", "Nombre", "Precio", "Inventario"};
19     private List<Producto> producto = new ArrayList<>();
20
21     public TableModelProducto(List<Producto> prod) {
22         this.producto = prod;
23     }
24
25
26     @Override
27     public int getRowCount() {
28         return this.producto.size();
29     }
30
31     @Override
32     public int getColumnCount() {
33         return this.columns.length;
34     }
35
36     @Override
```

The status bar at the bottom indicates "Unpacking index for Central Repository" and "INS| Unix (LF)".

```

35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
BaseDatosProductos.java x | Producto.java x | TableModelProducto.java x
Source History
@Override
public Object getValueAt(int fila, int columna) {
    Object resp;

    switch(columna){
        case 0:
            resp = this.producto.get(fila).getCodigo();
            break;
        case 1:
            resp = this.producto.get(fila).getNombre();
            break;
        case 2:
            resp = this.producto.get(fila).getPrecio();
            break;
        default:
            resp = this.producto.get(fila).getInventario();
    }
    return resp;
}

@Override
public String getColumnName(int colum) {
    return this.columnas[colum];
}

public void actualizarTabla() {
    fireTableDataChanged();
}

public Producto detalle(int fila){
    return this.producto.get(fila);
}
Notifications Unpacking index for Central Repository 35:1 INS Unix (LF)

```

- JFrame form

Inventario de Productos

Agregar Producto

Codigo:

Nombre:

Precio:

Inventario:

Lista Productos

*Seleccione el dato de la tabla que desea actualizar o Borrar

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
2	Limones	3.0	15
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10

```

BaseDatosProductos.java x Principal.java x Producto.java x TableModelProducto.java x
source Design History
package com.mycocompany.programacionproyec;

import javax.swing.JOptionPane;

/**
 * @author HP
 */
public class Principal extends javax.swing.JFrame {
    public int codigo;
    public String nombre;
    public double precio;
    public int inventario;
    Producto Productos [];
    BaseDatosProductos bdProductos = new BaseDatosProductos();

    /**
     * Creates new form Principal
     */
    public Principal() {
        initComponents();
        this.tblTabla.setModel(new TableModelProducto(bdProductos.getListaProductos()));
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

```

```

source Design History
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    if (validarDatos()){
        bdProductos.actualizar (new Producto (codigo, nombre, precio, inventario));
        this.tblTabla.setModel(new TableModelProducto(bdProductos.getListaProductos()));
        limpiarDatos();
        JOptionPane.showMessageDialog(this, "Producto Actualizado exitosamente", "Confirmacion",1);
    }else{
        JOptionPane.showMessageDialog(this, "Algunos de los valores digitados esta vacio o es erroneo", "Advertencia",2);
    }
}

private void btnAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    if (validarDatos()){
        if (!bdProductos.verificarExistencias(nombre)){
            if (codigo == 0) {
                int codigoAux = bdProductos.ultimoCodigo() + 1;
                bdProductos.agregar (new Producto (codigoAux, nombre, precio, inventario));
                this.tblTabla.setModel(new TableModelProducto(bdProductos.getListaProductos()));
                limpiarDatos();
                JOptionPane.showMessageDialog(this, "Producto Agregado exitosamente", "Confirmacion",1);
            }else{
                JOptionPane.showMessageDialog(this, "Imposible agregar! Esta accion es permitida para Actualizar o Borrar", "Advertencia",2);
            }
        }else {
            JOptionPane.showMessageDialog(this, "El producto ya existe", "Advertencia",2);
        }
    }
}

```



```

261     }else {
262         JOptionPane.showMessageDialog(this, "Algunos de los valores digitados esta vacio o es erroneo","Advertencia",2);
263     }
264 }
265
266
267
268 private void tblTablaMouseClicked(java.awt.event.MouseEvent evt) {
269     Producto producto = ((TableModelProducto) this.tblTabla.getModel()).detalle(this.tblTabla.getSelectedRow());
270     this.txtCodigo.setText(String.valueOf(producto.getCodigo()));
271     this.txtNombre.setText(producto.getNombre());
272     this.txtPrecio.setText(String.valueOf(producto.getPrecio()));
273     this.txtInventario.setText(String.valueOf(producto.getInventario()));
274 }
275
276
277 private void btnBorrarActionPerformed(java.awt.event.ActionEvent evt) {
278     if (validarDatos()){
279         bdProductos.borrar(new Producto(codigo, nombre, precio, inventario));
280         this.tblTabla.setModel(new TableModelProducto(bdProductos.getListaProductos()));
281         limpiarDatos();
282         JOptionPane.showMessageDialog(this, "Producto Eliminado exitosamente", "Confirmacion",1);
283     }else{
284         JOptionPane.showMessageDialog(this, "Algunos de los valores digitados esta vacio o es erroneo","Advertencia",2);
285     }
286 }
287
288
289
290 private void btnInformeActionPerformed(java.awt.event.ActionEvent evt) {
291     JOptionPane.showMessageDialog(this,bdProductos.generarInforme(), "Advertencia",2);}

```

```

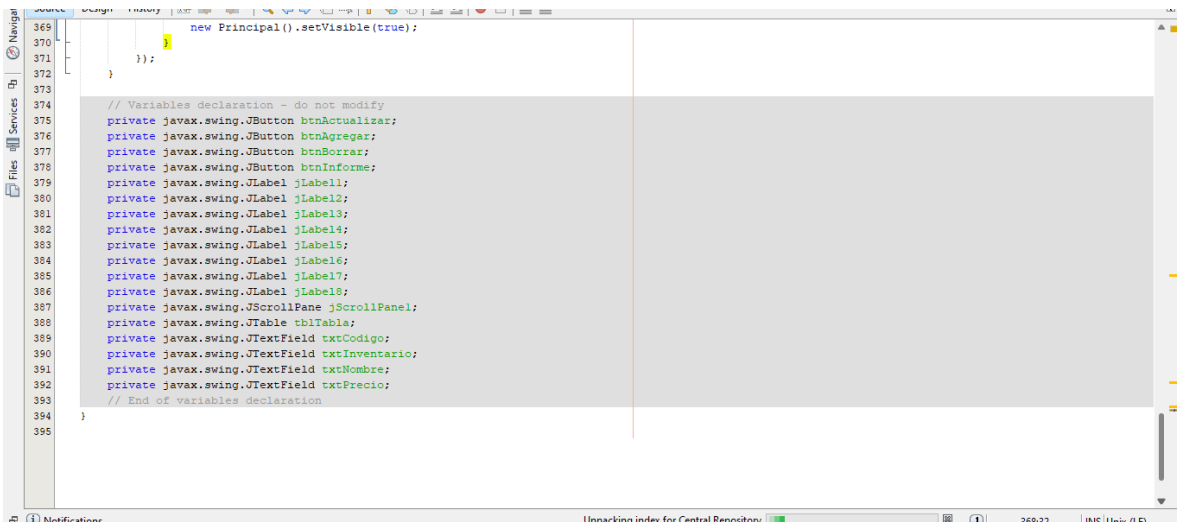
294
295
296
297 public boolean validarDatos() {
298     try {
299         codigo = Integer.parseInt(""+equals(txtCodigo.getText())? "0" : txtCodigo.getText());
300         nombre = txtNombre.getText();
301         precio = Double.parseDouble(txtPrecio.getText());
302         inventario = Integer.parseInt(txtInventario.getText());
303         return true;
304     } catch (Exception e ) {
305         return false;
306     }
307 }
308
309 public void limpiarDatos() {
310     this.txtCodigo.setText("");
311     this.txtNombre.setText("");
312     this.txtPrecio.setText("");
313     this.txtInventario.setText("");
314 }
315
316
317
318
319
320
321
322
323
324

```

```

338
339 /**
340  * @param args the command line arguments
341  */
342 public static void main(String args[]) {
343     /* Set the Nimbus look and feel */
344     //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
345     /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
346      * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
347      */
348     try {
349         for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
350             if ("Nimbus".equals(info.getName())) {
351                 javax.swing.UIManager.setLookAndFeel(info.getClassName());
352                 break;
353             }
354         }
355     } catch (ClassNotFoundException ex) {
356         java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
357     } catch (InstantiationException ex) {
358         java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
359     } catch (IllegalAccessException ex) {
360         java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
361     } catch (javax.swing.UnsupportedLookAndFeelException ex) {
362         java.util.logging.Logger.getLogger(Principal.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
363     }
364     //</editor-fold>
365
366     /* Create and display the form */
367     java.awt.EventQueue.invokeLater(new Runnable() {
368         public void run() {

```



```
369         new Principal().setVisible(true);
370     }
371 }
372
373
374 // Variables declaration - do not modify
375 private javax.swing.JButton btnActualizar;
376 private javax.swing.JButton btnAgregar;
377 private javax.swing.JButton btnBorrar;
378 private javax.swing.JButton btnInforme;
379 private javax.swing.JLabel jLabel1;
380 private javax.swing.JLabel jLabel2;
381 private javax.swing.JLabel jLabel3;
382 private javax.swing.JLabel jLabel4;
383 private javax.swing.JLabel jLabel5;
384 private javax.swing.JLabel jLabel6;
385 private javax.swing.JLabel jLabel7;
386 private javax.swing.JLabel jLabel8;
387 private javax.swing.JScrollPane jScrollPane1;
388 private javax.swing.JTable tblTabla;
389 private javax.swing.JTextField txtCodigo;
390 private javax.swing.JTextField txtInventario;
391 private javax.swing.JTextField txtNombre;
392 private javax.swing.JTextField txtPrecio;
393 // End of variables declaration
394
395 }
```

– BASE DE DATOS.

2. Tecnologías utilizadas.

Durante la elaboración del programa se utilizaron ciertas tecnologías que como tal son las básicas se podría decir que son las que utilizamos para optar por crear una proyecto o programa de esta magnitud la cuales son:

2.1.Gliffy Diagrams:

Es un software en línea, que permite crear fácilmente diferentes diagramas, así como de flujo, diagramas de clases, planos, dibujos técnicos y muchos más, este software ofrece una gran variedad de herramienta para que el usuario modifique y edite de una mejor manera sus diagramas a gusto propio.

Principalmente utilizamos Gliffy para la creación del “Diagrama de Clases” el cual nos permitió dar una idea de como más o menos debemos estructurar el código, así nos hiciera más fácil comprender la lógica y poder empezar a realizar el código con una idea clara de lo que tenia que hacer.

Es de buena ayuda crea diagramas en una calidad y de una forma muy rápida, eso facilita en cierto modo crear diagramas y dejar de utilizar la manera clásica de hacer diagramas en hojas y a lápiz.



2.2.Java Netbeans IDE

Es un Entorno de desarrollo integrado libre (IDE), que está hecho principalmente para el lenguaje de programación Java, es un producto libre y gratuito sin restricciones de uso. NetBeans IDE es libre, código abierto, multiplataforma con soporte integrado para el lenguaje de programación Java.

Este proyecto cuenta con una gran base de usuarios y una comunidad muy constante de código abierto, permite que las aplicaciones que se realicen en el entorno sean a partir de un conjunto de componente de software llamados módulos, los módulos están formados por clases que al momento de utilizarlos permite un control y orden sobre el proyecto que se quiere realizar. NetBeans IDE permite el desarrollo de todos los tipos de aplicación Java, J2SE, web, EJB y aplicaciones móviles.

La plataforma de NetBeans ofrece servicios reusables comunes para las aplicaciones de escritorios dichas características son:

- Gestión de la interfaz de usuario (menús y barras de herramientas)

- Gestión de configuración de usuario
- Gestión de almacenamiento (guardar o cargar algún tipo de dato)
- Gestión de ventana
- Marco Asistente (soporta diálogos paso a paso)
- Librería visual de Neatbeans
- Herramientas de desarrollo integrado.



2.3.MySQL Workbench

Es una herramienta visual para bases de datos a su vez este software permite crear, editar y vincular las bases de datos, además de integrar a algún otro desarrollo de software.

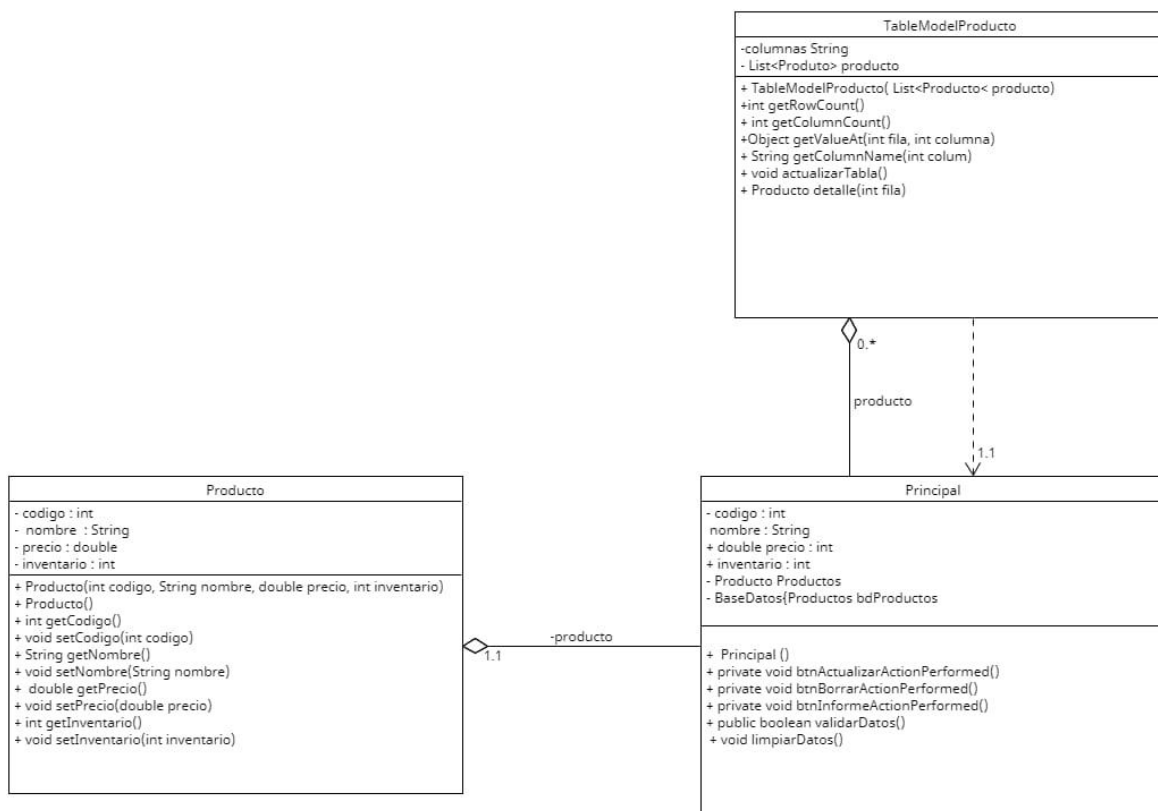
Este software ofrece dos ediciones diferentes una en open source y una edición comercial. Le brinda al usuario una infinidad de actividades que puede realizar en una base de datos, esto hace que el usuario pueda desarrollar todo tipo de SQL y tenga herramientas de administración completas.

Los desarrolladores y administradores de bases de datos van a poder convertir rápida y fácilmente aplicaciones existentes para ejecutar en My SQL en cualquier plataforma o sistema operativo existente.



3. Diagramas.

3.1. Diagrama de Clases (UML)



3.2. Diagrama de Base de Datos.

Manual de usuario

Este software se encarga de hacer un inventario de los productos que el usuario ingrese, para tener un mejor control de los productos que ingresen y salgan del almacén.

El software cuenta con las características de poder ingresar el nombre, el precio y la cantidad que se tenga de un producto, además que es mismo le asigna un código al producto para identificarlo.

Función Agregar

Si se desea agregar un nuevo producto lo que tenemos que hacer es:

1. Ingresar el nombre, precio y el inventario del producto
2. Seleccionar el botón agregar

The screenshot shows the 'Inventario de Productos' application. At the top, there's a title 'Inventario de Productos' and a subtitle 'Agregar Producto'. Below this, there are four input fields: 'Codigo:' with the value '1', 'Nombre:' with 'Manzanas', 'Precio:' with '2.5', and 'Inventario:' with '65'. A red arrow labeled '1' points to the 'Nombre' field, and another red arrow labeled '2' points to the 'Agregar' button. Below the inputs are three buttons: 'Actualizar', 'Agregar', and 'Borrar'. At the bottom, there's a section titled 'Lista Productos' with a table of products. A note says '*Seleccione el dato de la tabla que desea actualizar o Borrar'. An 'Informe' button is also present on the right.

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
2	Limones	3.0	15
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10

Función Borrar

Cuando el inventario de un producto se haya acabado lo que podemos hacer es borrar el registro correspondiente de la tabla

1. Seleccionar el producto en la tabla
2. Seleccionar el botón eliminar

Inventario de Productos

Agregar Producto

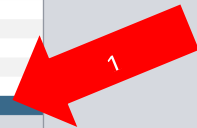

Codigo: 11
Nombre: Limones
Precio: 3.0
Inventario: 15

Agregar

Actualizar Borrar Informe

Lista Productos *Seleccione el dato de la tabla que desea actualizar o Borrar

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10
11	Limones	3.0	15



Inventario de Productos

Agregar Producto

Codigo:
Nombre:
Precio:
Inventario:

Agregar

Actualizar Borrar Informe

Lista Productos *Selección

Confirmación

Producto Eliminado exitosamente

OK

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10

Función actualizar

Cuando entre un nuevo cargamento de inventario de un producto ya existente lo que podemos hacer es usar la función actualizar

Pasos para actualizar un producto:

1. Seleccionar el producto que se quiera actualizar
2. Poner el nuevo precio y el inventario
3. Seleccionar el botón actualizar

Inventario de Productos

Agregar Producto

Codigo: 5

Nombre: Tomates

Precio: 5.0

Inventario: 150

Agregar

Actualizar

Borrar

Informe

*Seleccione el dato de la tabla que desea actualizar o Borrar

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10
11	limones	3.0	15

Lista Productos

Inventario de Productos

Agregar Producto

Codigo:

Nombre:

Precio:

Inventario:

Actualizar

Borrar

Informe

*Seleccione el dato de la tabla que desea actualizar o Borrar

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	5.0	150
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10
11	limones	3.0	15

Lista Productos

Confirmacion

Producto Actualizado exitosamente

OK

Función informe

Si deseamos un informe de los productos solo hay que seleccionar el botón informe

Inventario de Productos

Agregar Producto

Codigo:

Nombre:

Precio:

Inventario:

Lista Productos

*Seleccione el dato de la tabla que desea actualizar o Borrar

Codigo	Nombre	Precio	Inventario
1	Manzanas	2.5	65
2	Limonas	3.0	15
3	Granadilla	4.0	38
4	Arandanos	5.0	55
5	Tomates	10.0	42
6	Fresas	15.0	3
7	Helado	8.0	41
8	Galletas	10.0	8
9	Chocolates	7.0	806
10	Jamon	12.0	10

Advertencia

!

Fresas Jamon Tomates

OK

II. Conclusiones.

1. Una data base, o como se conoce comúnmente en español como base de datos, es la forma de almacenamiento de datos más común que existe, como también hay distintos programas que ejecutan una DB.
2. Una base de datos no es base de datos sin que exista un

Link de GITHUB

<https://github.com/yoanan1996/Proyect-Inventario>