



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

Trabajo Final-Unidad II-Proyecto

**“Aplicación del NOSQL en el Proyecto Aplicativo
Web y Sistemas de Ventas Electrodomésticos”**

Curso: Base de Datos II

Docente: Ing. Patrick José Cuadros

INTEGRANTES:

- Linares Chávez, Cesar (2019063854)
- Lira Álvarez , Rodrigo (2019063331)
- Perez Vizcarra, Juan Jose (2019063636)

Tacna – Perú

2022

INDICE GENERAL

RESUMEN	3
ABSTRACT.....	3
1. ANTECEDENTES O INTRODUCCIÓN	4
2. TITULO	4
3. AUTORES	4
4. PLANTEAMIENTO DEL PROBLEMA	4
5. OBJETIVOS	7
6. REFERENTES TEORICOS	7
7. DESARROLLO DE LA PROPUESTA	10
8. BIBLIOGRAFIA	33
9. ANEXOS	33

RESUMEN

La Tienda Nutrifit, ubicada en la ciudad de Tacna, se dedica a la venta de artículos electrodomésticos, opera desde la mañana hasta la tarde tiene un pequeño almacén en su segundo piso y actualmente están buscando una renovación de su sistema.

La empresa necesita una recolección de datos de forma rápida y simple, que desea adquirir a través de la obtención de datos que puedan ser utilizados en las labores de los empleados cuando ellos lo necesiten.

Según sus necesidades se plantean que en el nuevo sistema tenga como prioridad el registro de ventas, el registro de los empleados, el registro de clientes y la impresión de las boletas.

También se hará la integración del aplicativo web que contendrá los artículos que estén en disponibilidad para que los clientes puedan realizar sus compras con facilidad.

ABSTRACT

The Nutrifit Store, located in the city of Tacna, is dedicated to the sale of household appliances, operates from morning to afternoon, has a small warehouse on the second floor and is currently looking for a renovation of its system.

The company needs a quick and simple data collection, which they want to acquire through the collection of data that can be used in the work of employees when they need it.

According to their needs, the new system will have as a priority the registration of sales, the registration of employees, the registration of customers and the printing of receipts.

The integration of the web application that will contain the items that are in availability so that customers can make their purchases easily will also be done.

1. Antecedentes o Introducción

La tienda Nutrifit, es una empresa pequeña de electrodomésticos que se encuentra en el centro de la ciudad de Tacna, ofrece variedad de productos electrónicos especialmente líderes en venta de electrodomésticos, ofreciendo diferentes marcas tales como Sony, LG, Samsung, AIWA y otros.

Agregando también, que poseen muchas categorías de productos de los cuales tienen para ofrecer, en estos momentos, al aumentar las categorías como también productos nuevos y la acumulación de información anteriores y registros, surge la necesidad de realizar nuevas actualizaciones, consultas rápidas y completas para ver la información detallada de los productos, como también funciones para las consultas e inserciones de los empleados, llegando a utilizar diversos modelos para que las consultas de datos se desarrollen con éxito.

2. TITULO

Aplicación del NOSQL en el Proyecto de Aplicativo Web Tienda de Electrodomésticos (ARTEC) y Sistemas de Ventas Electrodomésticos.

3. AUTORES

N	Nombre	Cargo
1	Linares Chávez, Cesar	Jefe de Proyecto
2	Linares Chávez, Cesar	Programador
3	Perez Vizcarra, Juan Jose	Analista
4	Perez Vizcarra, Juan Jose	DBA
5	Lira Álvarez , Rodrigo	Programador

4. PLANTEAMIENTO DEL PROBLEMA

4.1 PROBLEMA

En la actualidad, La Tienda Nutrifit de momento no cuenta con un sistema que tenga los mantenimientos de los empleados, productos y ventas, que sea rápida y eficaz en las consultas de la base de datos, por tal motivo la empresa requiere una nueva mejora en su sistema.

Para realizar tal mejora e implementado el software requerido por la empresa, se ha desempeñado en la utilización de una metodología bastante útil para el desarrollo de base de datos.

Se trata del OR-M o ORM (Objet Relational Mapping) es un modelo de programación que permitirá convertir los datos de los objetos en formato correcto para guardar la información en una base de dato y quedan vinculados a la base de datos (persistencia) pasando por la creación de una base de datos virtual. Es complicado transformar la información que reciba en los objetos de la aplicación, es decir, hacer tablas en ello.

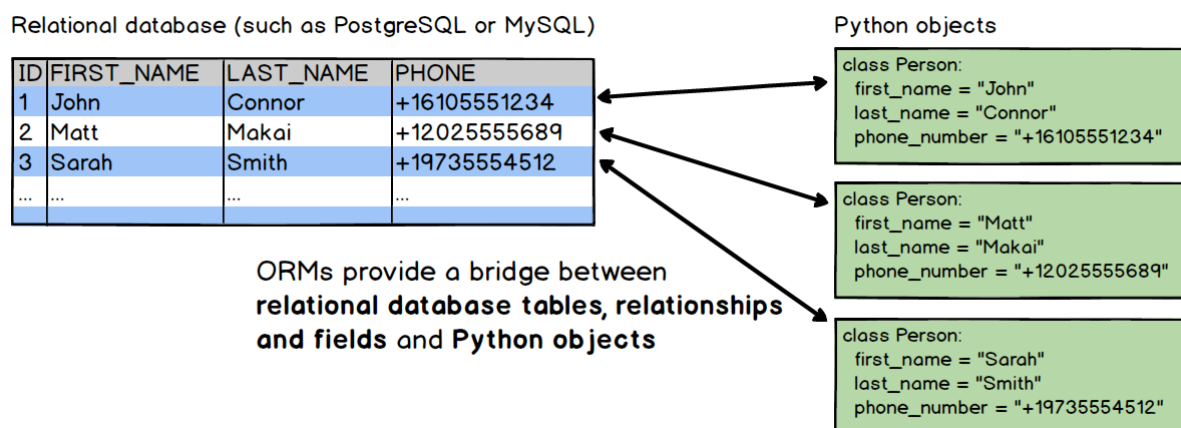
Siendo que este modelo, se basa en las estructuras de la base de datos relacional que son vinculadas con las entidades lógicas o base de datos virtual definida en el ORM, de modo que las acciones CRUD (Create, Read, Update, Delete) se ejecutan sobre la base de datos física y realizan de forma indirecta por medio del ORM.

En resumen, con el modelo de programación mencionado, debemos utilizar las clases entidades y presentaciones, dejando al lado las clases de negocios para poder ahorrar código y consultar los datos de una forma sencilla.

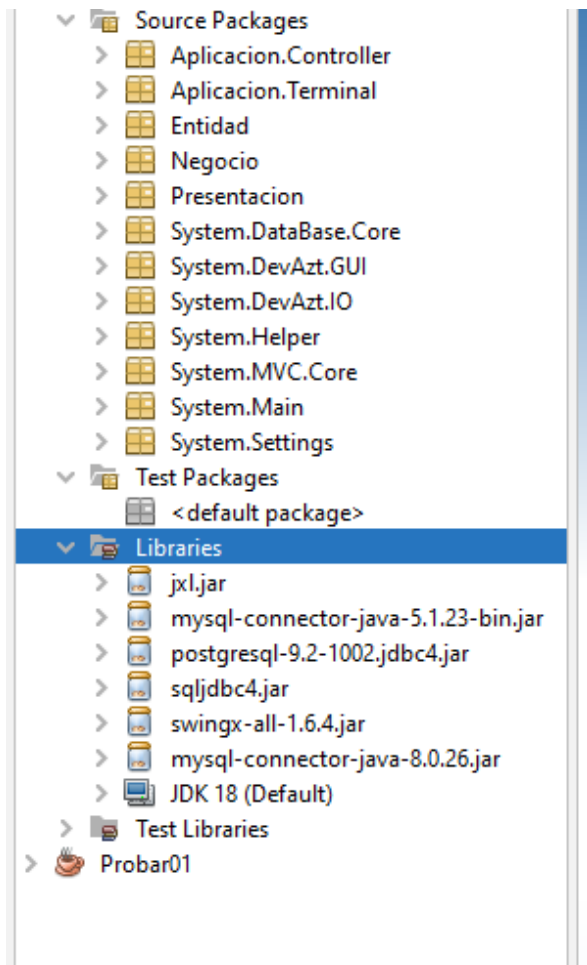
El uso de un ORM en la base de datos permite adaptarse a los nuevos tiempos y darnos una mayor versatilidad a la hora de manejar los datos.

De sus funcionalidades se puede mencionar:

- El ORM funciona como una capa intermedia completamente separada de las bases de datos
- Hace que una migración sea más sencilla.
- Es un código más legible y con menos líneas.
- Otra ventaja importante es la seguridad. Al ser una capa independiente a los datos, nos permite protegerlos de ciberataques al no estar en el mismo nivel.
- Un ORM nos guarda y carga toda la información de una base de datos relacional automáticamente.
- Nos evita escribir a mano consultas de SQL necesarias.



El modelo OR-M, puede aclarar el objetivo principal de todo el problema de solucionar y ahorrar el tiempo de las consultas y la utilización del CRUD sin tener que utilizar el modelo de negocio o Clases de negocios en el proyecto siguiendo el proceso de objeto relacional mapeado



4.2 JUSTIFICACION

La finalidad de realizar este proyecto buscar facilitar las consultas de los datos, mantenimiento de los productos y empleados de la tienda para que así la empresa tenga un manejo óptimo de sus recursos permitiéndoles de manera notable incrementar su confianza al realizar sus ventas.

También el proyecto tiene que realizarse en el plazo de tiempo solicitado por la propia empresa, permitiéndoles obtener nuevos aspectos tecnológicos para la automatización de procesos por medio del sistema planteado, acortando distancias, disminuyendo costos y ofreciendo todo tipo de productos electrónicos para toda la comunidad virtual.

4.3 ALCANCE

El sistema tendrá una disponibilidad y alcance de los mantenimientos de los empleados, productos y ventas en lo que pueda ser utilizado con plenitud, también tendrá consultas incluidas dentro del propio sistema para que se pueda obtener información de los datos correspondientes. El sistema es únicamente para los empleados de la tienda y para los clientes, solo tendrá que acceder a las compras de los productos que ofrece la tienda y que después de ello se guardará los datos de la venta, permitiendo optimizar el proceso de mantenimiento y de todos los procesos relacionados con el mismo,

5. OBJETIVOS

5.1. GENERAL

- Establecer los métodos y tipos de mantenimiento a aplicar, de forma adecuada para procurar maximizar la disponibilidad y confiabilidad de la administración de datos y el mantenimiento de los datos de forma eficiente a través de un software para la Empresa Nutrifit.

5.2. ESPECIFICOS

- Reducir el tiempo de respuesta de las consultas de los datos guardados en la base de datos.
- Reducir los costos de mantenimiento.
- Mantener un control del inventario usando el sistema y aplicativo siendo la principal, para los clientes y el otro, la modificación de productos y mantenimientos para los empleados.
- Aumentar la confiabilidad de administración de datos.
- Aumentar la disponibilidad de administración de datos.

6. REFERENTES TEORICOS

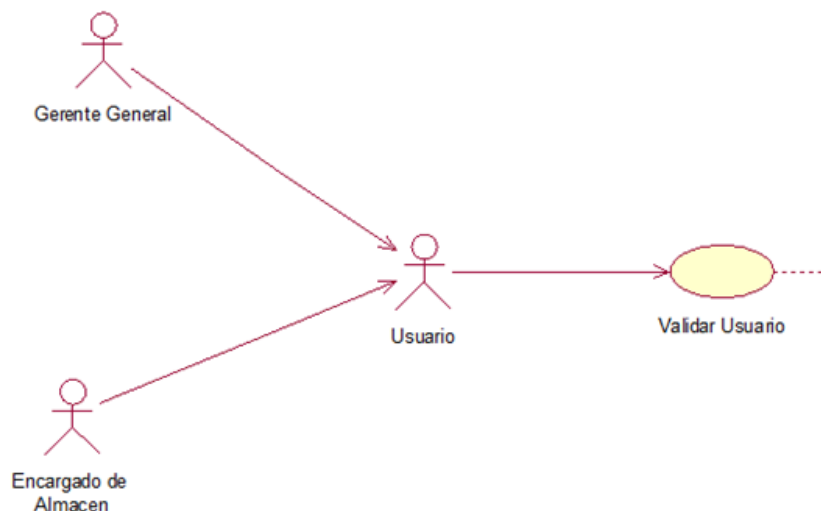
- Diagramas De Casos De Uso
 - Paquete – Login – Sitio Web

Validar Usuario- Sitio Web

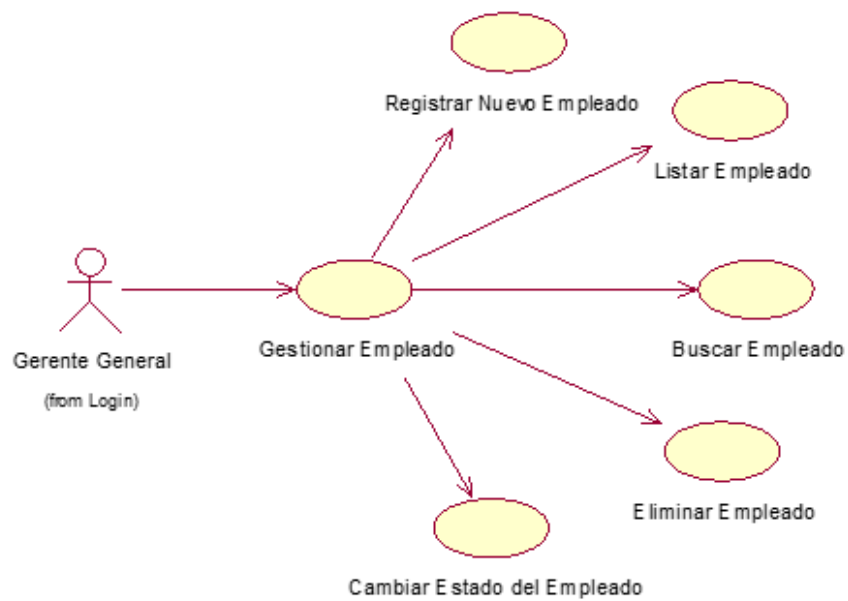


- Paquete-Login

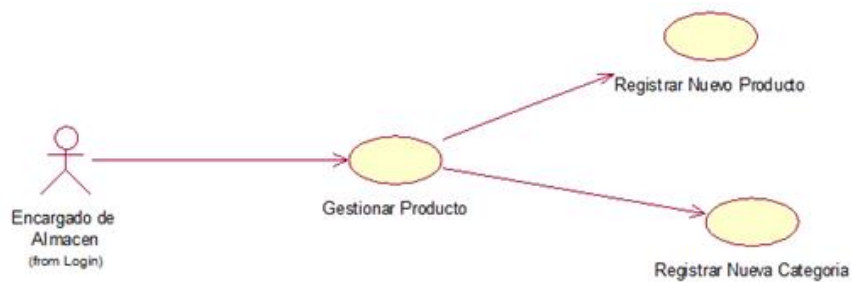
Validar Usuario



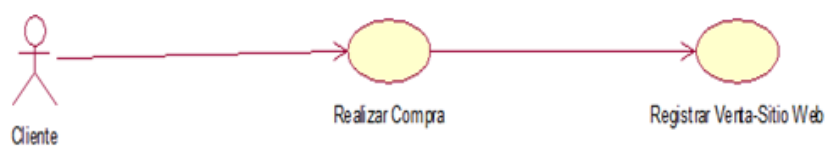
- Paquete-Empleados: Gestionar Empleado



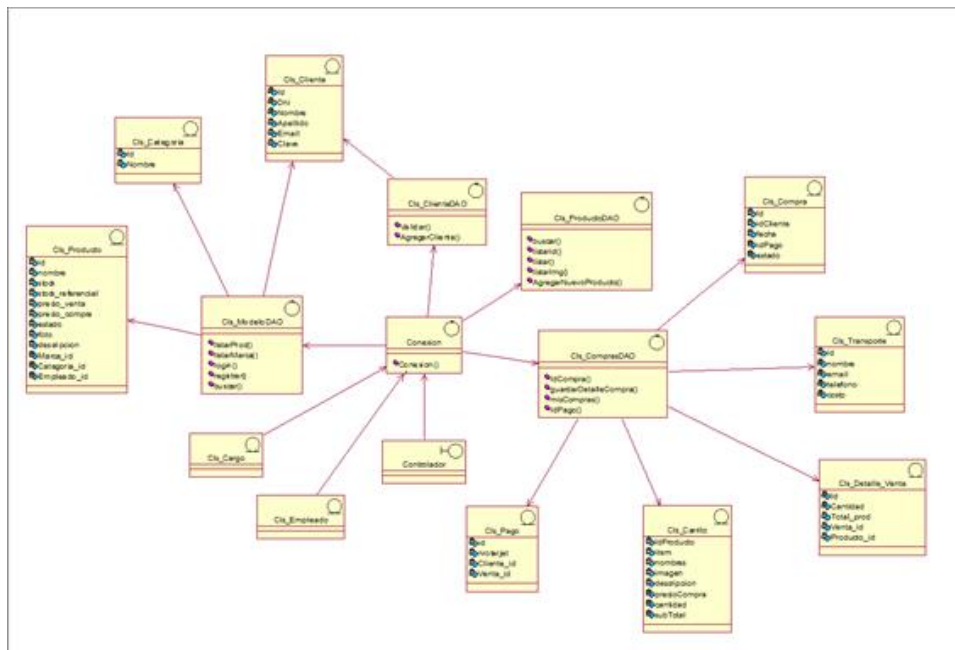
- Paquete-Productos: Gestionar Productos



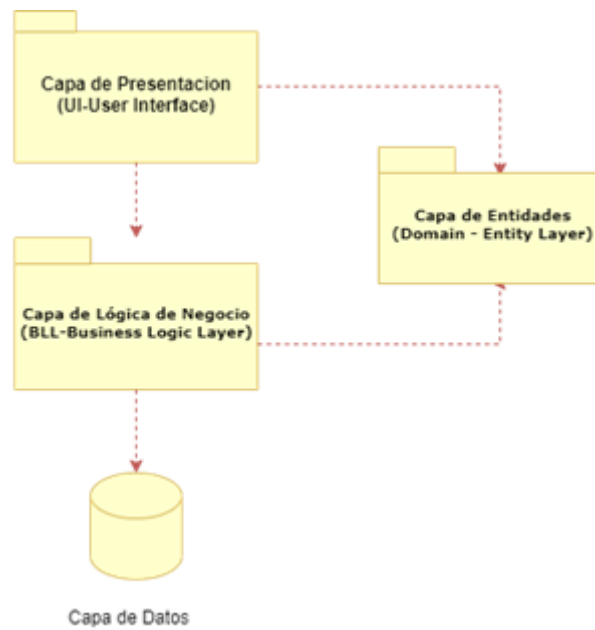
- Paquete-Ventas-Sitio Web
Registrar Venta



- Diagramas de clases

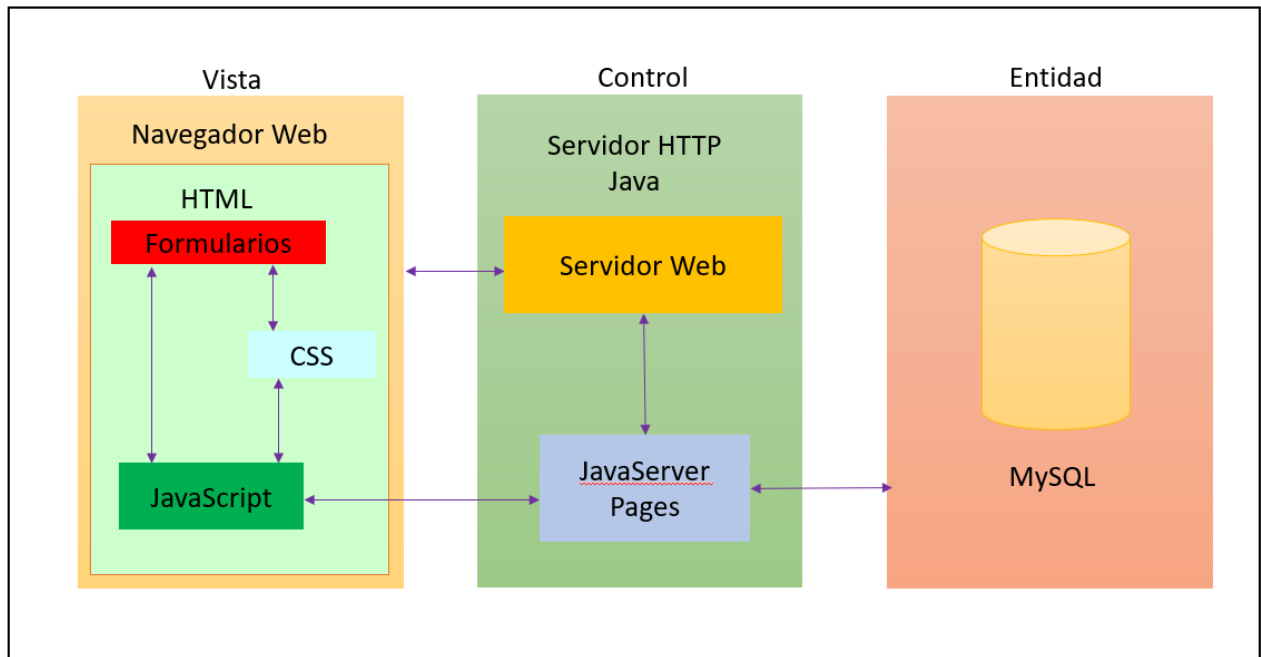


- Diagramas de componentes





























- Diagrama de Arquitectura de la aplicación

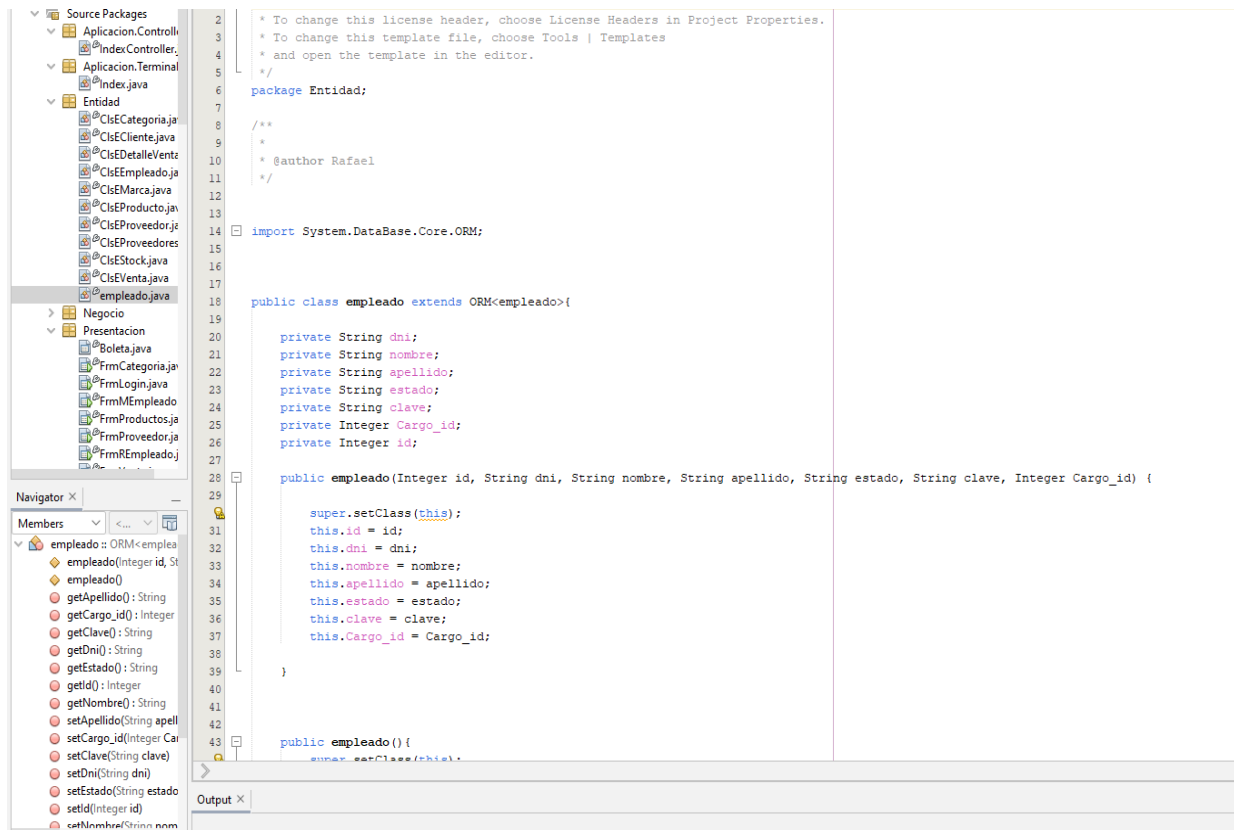
**Aplicativo Web Tienda de Electrodomésticos (ARTEC) y
Sistemas de Ventas Electrodomésticos**

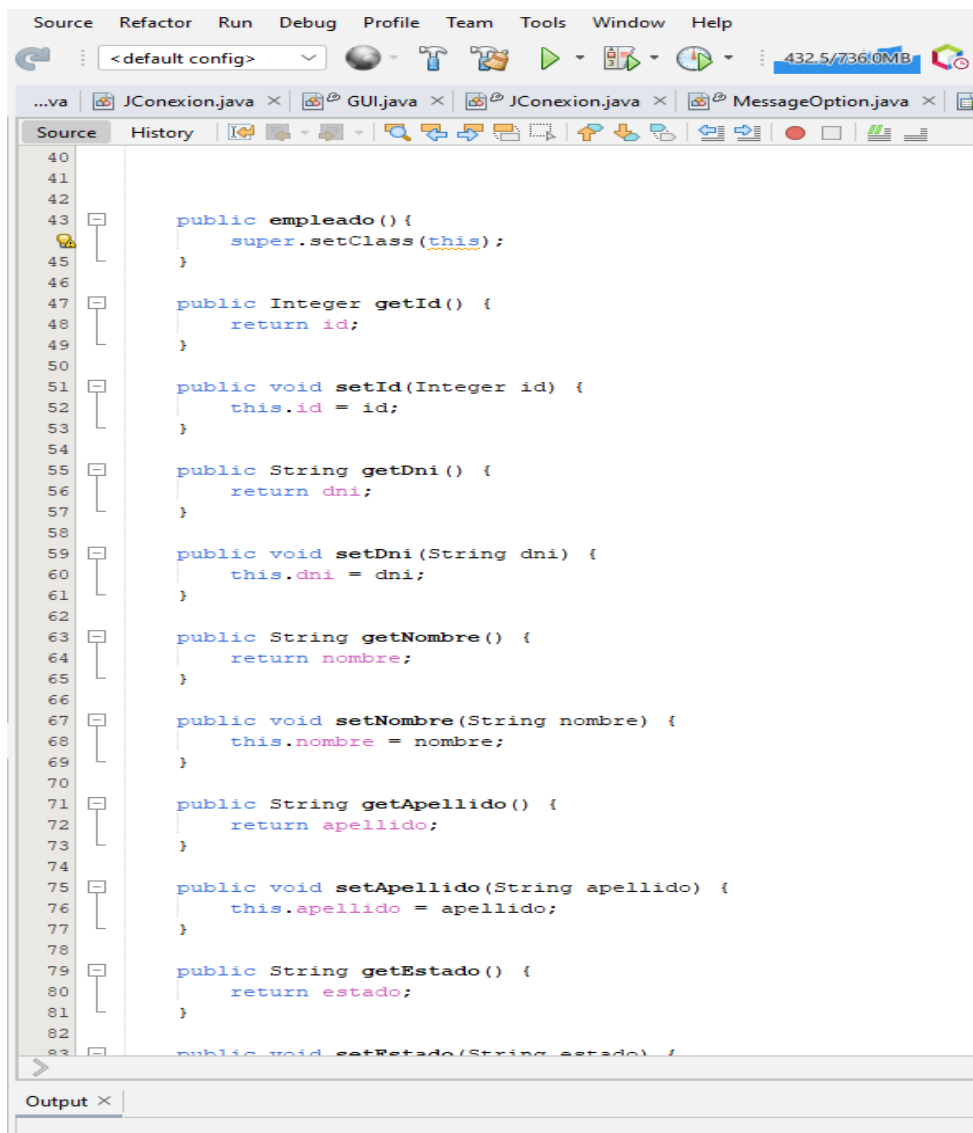


7. DESARROLLO DE LA PROPUESTA

7.1 CAPTURAS DE CODIGO DE APLICACION DEL OR/M

- ▼  AppJConexion
 - ▼  Source Packages
 - >  Aplicacion.Controller
 - >  Aplicacion.Terminal
 - >  Entidad
 - >  Negocio
 - >  Presentacion
 - >  System.DataBase.Core
 - >  System.DevAzt.GUI
 - >  System.DevAzt.IO
 - >  System.Helper
 - >  System.MVC.Core
 - >  System.Main
 - >  System.Settings
 - ▼  Test Packages
 -  <default package>
 - ▼  Libraries
 - >  jxl.jar
 - >  mysql-connector-java-5.1.23-bin.jar
 - >  postgresql-9.2-1002.jdbc4.jar
 - >  sqljdbc4.jar
 - >  swingx-all-1.6.4.jar
 - >  mysql-connector-java-8.0.26.jar
 - >  JDK 18 (Default)
 - >  Test Libraries
- >  Probar01





```
Source Refactor Run Debug Profile Team Tools Window Help
<default config>
JConexion.java x GUI.java x JConexion.java x MessageO
Source History
61 }
62
63 public String getNombre() {
64     return nombre;
65 }
66
67 public void setNombre(String nombre) {
68     this.nombre = nombre;
69 }
70
71 public String getApellido() {
72     return apellido;
73 }
74
75 public void setApellido(String apellido) {
76     this.apellido = apellido;
77 }
78
79 public String getEstado() {
80     return estado;
81 }
82
83 public void setEstado(String estado) {
84     this.estado = estado;
85 }
86
87 public String getClave() {
88     return clave;
89 }
90
91 public void setClave(String clave) {
92     this.clave = clave;
93 }
94
95 public Integer getCargo_id() {
96     return Cargo_id;
97 }
98
99 public void setCargo_id(Integer Cargo_id) {
100     this.Cargo_id = Cargo_id;
101 }
102
103
104
```

```

1  /**
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package Presentacion;
7   //import Entidad.ClsETrabajador;
8
9   import Entidad.empleado;
10  import Negocio.*;
11  import Entidad.ClsEEmpleado;
12  import java.sql.ResultSet;
13  import java.sql.SQLException;
14  import java.util.ArrayList;
15  import javax.swing.JOptionPane;
16
17  /**
18   *
19   * @author Arnold
20   */
21  public class FrmLogin extends javax.swing.JFrame {
22
23      /**
24       * Creates new form FrmLogin
25       */
26      public FrmLogin() {
27          initComponents();
28          this.setLocationRelativeTo(null);
29      }
30
31      String DniEmpleado = "";
32      String NombreEmpleado = "";
33
34      /**
35       * This method is called from within the constructor to initialize the form.
36       * WARNING: Do NOT modify this code. The content of this method is always
37       * regenerated by the Form Editor.
38       */
39      @SuppressWarnings("unchecked")

```

```

117
118
119     private void btnIngresarActionPerformed(java.awt.event.ActionEvent evt) {
120         //
121         //
122         //
123
124         //
125         //
126
127
128         String dni= txtDni.getText();
129         String clave=txtClave.getText();
130
131         Boolean conf=false;
132         String Cargo="";
133         String estado = "";
134
135         empleado u=new empleado();
136
137         ArrayList<empleado> list = u.get();
138
139         for(empleado list1:list){
140
141             if(list1.getDni().equals(dni) & list1.getClave().equals(clave)){
142                 conf=true;
143                 estado=list1.getEstado();
144                 Cargo=list1.getCargo_id().toString();
145             }
146         }
147         if(conf=true){
148             if(estado.equals("A")){
149                 switch(Cargo)
150                 {
151                     case "1" :
152                         JOptionPane.showMessageDialog(null,"inicia Sistema");
153                         DniEmpleado = txtDni.getText();
154                         NombreEmpleado = txtClave.getText();
155                         FrmMEmpleado FEmp = new FrmMEmpleado();
156                         FEmp.labelDni.setText(txtDni.getText());
157                         FEmp.show();
158                         this.dispose();
159                         break;
160                     case "2" :

```



```

144     }
145 }
146
147 if (conf=true) {
148     if (estado.equals("A")) {
149         switch (Cargo)
150         {
151             case "1" :
152                 JOptionPane.showMessageDialog(null, "inicia Sistema");
153                 DniEmpleado = txtDni.getText();
154                 NombreEmpleado = txtClave.getText();
155                 FrmMEmpleado FEmp = new FrmMEmpleado();
156                 FEmp.labelDni.setText(txtDni.getText());
157                 FEmp.show();
158                 this.dispose();
159                 break;
160             case "2" :
161                 JOptionPane.showMessageDialog(null, "inicia Sistema");
162                 DniEmpleado = txtDni.getText();
163                 NombreEmpleado = txtClave.getText();
164                 FrmVenta frm = new FrmVenta();
165                 FrmVenta.labelDniEmpleado.setText(txtDni.getText());
166                 frm.show();
167                 this.dispose();
168                 break;
169             default :
170         }
171     }
172     else {
173         JOptionPane.showMessageDialog(null, "Cuenta suspendida ");
174     }
175 }
176
177 else {
178     JOptionPane.showMessageDialog(null, "Cuenta incorrecta");
179 }
180
181

```

```

248     }
249
250     private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {
251         this.dispose();
252     }
253
254
255     /**
256      * @param args the command line arguments
257      */
258     public static void main(String args[]) {
259         /* Set the Nimbus look and feel */
260         Look and feel setting code (optional)
261         //</editor-fold>
262
263         /* Create and display the form */
264         java.awt.EventQueue.invokeLater(new Runnable() {
265             public void run() {
266                 new FrmLogin().setVisible(true);
267             }
268         });
269     }
270
271     // Variables declaration - do not modify
272     private javax.swing.JButton btnIngresar;
273     private javax.swing.JButton btnSalir;
274     private javax.swing.JLabel jLabel1;
275     private javax.swing.JLabel jLabel2;
276     private javax.swing.JLabel jLabel3;
277     private javax.swing.JTextField txtClave;
278     private javax.swing.JTextField txtDni;
279     // End of variables declaration
280 }
281

```

```

7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
import java.sql.ResultSet;
import java.sql.SQLException;
import Negocio.ClsNEmpleado;
import Entidad.ClsEEmpleado;
import Entidad.empleado;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import javax.swing.JOptionPane;
import javax.swing.Timer;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Arnold
 */
public class FrmMEmpleado extends javax.swing.JFrame {

    /**
     * Creates new form FrmMEmpleado
     */

    ClsNEmpleado objNEmp = new ClsNEmpleado();
    ClsEEmpleado objEEmp = new ClsEEmpleado();

    DefaultTableModel modelo=new DefaultTableModel();

    public FrmMEmpleado() {
        initComponents();
        this.setLocationRelativeTo(null);
        modelo.addColumn("Dni");
        modelo.addColumn("Nombres");
        modelo.addColumn("Apellidos");
        modelo.addColumn("Cargo");
        modelo.addColumn("Estado");
        modelo.addColumn("Clave");
        this.Tabla.setModel(modelo);
        fecha_actual();
    }

```

```

52 public void fecha_actual() {
53     Date sisFecha = new Date();
54     SimpleDateFormat formato= new SimpleDateFormat("dd-MM-YYYY");
55     Fecha.setText(formato.format(sisFecha));
56
57     Timer tiempo = new Timer(100, new FrmMEmpleado.horas());
58     tiempo.start();
59 }
60
61 class horas implements ActionListener{
62     public void actionPerformed(ActionEvent e){
63         Date sistHora = new Date();
64         String pmAm = "hh:mm:ss a";
65         SimpleDateFormat format = new SimpleDateFormat(pmAm);
66         Calendar hoy = Calendar.getInstance();
67         Hora.setText(String.format(format.format(sistHora), hoy));
68     }
69 }
70
71 /**
72  * This method is called from within the constructor to initialize the form.
73  * WARNING: Do NOT modify this code. The content of this method is always
74  * regenerated by the Form Editor.
75  */
76 @SuppressWarnings("unchecked")
77 Generated Code
286
287 private void btnListarActionPerformed(java.awt.event.ActionEvent evt) {
288
289     MtdlimpiarTabla();
290     MtdListar();
291 }
292
293 private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
294     MtdlimpiarTabla();
295     //      ResultSet rs;
296     //
297     //      objEEmp.setB(txtDni.getText());
298     //      int flag = 0;
299
300     String dni= txtDni.getText();
301     empleado u=new empleado();
302
303     ArrayList<empleado> list = u.get();

```

```

294 private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
295     MtdlimpiarTabla();
296     //      ResultSet rs;
297     //
298     //      objEEmp.setB(txtDni.getText());
299     //      int flag = 0;
300
301     String dni= txtDni.getText();
302     empleado u=new empleado();
303
304     ArrayList<empleado> list = u.get();
305
306     for(empleado list1:list){
307
308         if(list1.getDni().equals(dni)){
309             modelo.addRow(new Object[]{list1.getDni(),list1.getNombre(),list1.getApellido(),
310                 list1.getCargo_id(),list1.getEstado(),list1.getClave()});
311             Tabla.setModel(modelo);
312         }
313     }
314
315

```

```

418 private void btnLimpiarActionPerformed(java.awt.event.ActionEvent evt) {
419     MtdlimpiarTabla();
420     txtDni.setText("");
421 }
422
423 private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
424     int fila = Tabla.getSelectedRow();
425     if(fila>=0){
426         // objNEmp.MtdEliminar(Integer.parseInt(Tabla.getValueAt(fila, 0).toString()));
427         String dni=Tabla.getValueAt(fila, 0).toString();
428
429         empleado u=new empleado();
430         ArrayList<empleado> list = u.get();
431         for(empleado list1:list){
432             if(list1.getDni().equals(dni)){
433                 empleado f=new empleado().find(list1.getId());
434                 if(f!=null){
435                     f.delete();
436                 }
437                 modelo.removeRow(fila);
438             }
439         }
440     }
441
442
443
444
445 }
446 else{
447     JOptionPane.showMessageDialog(null, "Seleccione una fila de la tabla");
448 }
449
450 }

```

```

451 private void btnEstadoActionPerformed(java.awt.event.ActionEvent evt) {
453
454     int fila = Tabla.getSelectedRow();
455     if(fila>=0){
456         // objNEmp.MtdEliminar(Integer.parseInt(Tabla.getValueAt(fila, 0).toString
457         String dni=Tabla.getValueAt(fila, 0).toString();
458         String estado=Tabla.getValueAt(fila, 4).toString();
459
460         empleado u=new empleado();
461         ArrayList<empleado> list = u.get();
462         for(empleado list1:list){
463
464             if(list1.getDni().equals(dni)){
465                 if(list1.getEstado().equals("A")){
466                     empleado f=new empleado().find(list1.getId());
467                     f.setEstado("I");
468                     f.update();
469                     MtdlimpiarTabla();
470                     MtdListar();
471
472                 }
473                 else{
474                     empleado f=new empleado().find(list1.getId());
475                     f.setEstado("A");
476                     f.update();
477                 }
478             }
479         }
480     }
481 }
482 else{
483     JOptionPane.showMessageDialog(null, "Seleccione una fila de la tabla");
484 }
485
486

```

```

512
513 private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
514
515     FrmREmpleado FrmREmp = new FrmREmpleado();
516     FrmREmp.show();
517     this.dispose();
518
519 }
520
521 private void btnSalirActionPerformed(java.awt.event.ActionEvent evt) {
522     FrmLogin FrmLog= new FrmLogin();
523     FrmLog.show();
524     this.dispose();
525 }
526
527 private void MtdListar() {
528     //      ResultSet rs;//iniciando nuestro contenedor de datos
529     //      rs=objNEmp.MtdListar_Empleados();//rs contiene el resultado de la consulta al BD
530     //      try {
531     //          while (rs.next())//inicia desde el primer elemento hasta el final de la consulta DB
532     //          {
533     //              modelo.addRow(new Object[]{rs.getInt("dni"),rs.getString("nombre"),rs.getString("apellido"),
534     //              rs.getString("Cargo_id"),rs.getString("estado"),rs.getString("clave")});
535     //          }
536     //          Tabla.setModel(modelo);
537     //      } catch (SQLException ex) {
538     //          System.out.println(ex);
539     //      }
540
541     empleado u=new empleado();
542
543     ArrayList<empleado> list = u.get();
544
545     for(empleado list1:list){
546
547         modelo.addRow(new Object[]{list1.getDni(),list1.getNombre(),list1.getApellido(),
548             list1.getCargo_id(),list1.getEstado(),list1.getClave()});
549         Tabla.setModel(modelo);
550
551     }
552
553 }
554

```

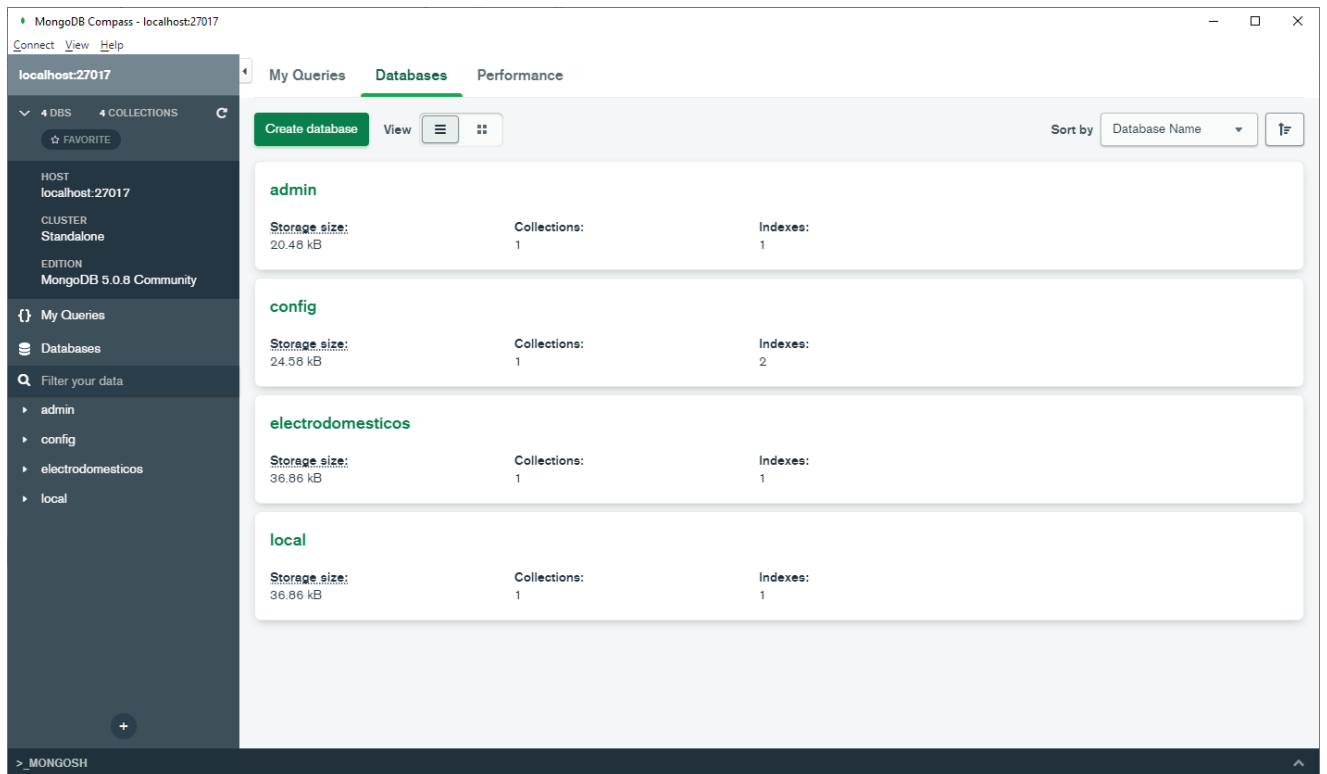
```

554
555 private void MtdlimpiarTabla() {
556     modelo.getDataVector().removeAllElements();
557     Tabla.updateUI();
558 }
559
560
561 /**
562  * @param args the command line arguments
563  */
564 public static void main(String args[]) {
565     /* Set the Nimbus look and feel */
566     Look and feel setting code (optional)
567     //</editor-fold>
568
569     /* Create and display the form */
570     java.awt.EventQueue.invokeLater(new Runnable() {
571         public void run() {
572             new FrmMEmpleado().setVisible(true);
573         }
574     });
575 }
576
577 // Variables declaration - do not modify
578 private javax.swing.JLabel Fecha;
579 private javax.swing.JLabel Hora;
580 private javax.swing.JTable Tabla;
581 private javax.swing.JButton btnBuscar;
582 private javax.swing.JButton btnEliminar;
583 private javax.swing.JButton btnEstado;
584 private javax.swing.JButton btnLimpiar;
585 private javax.swing.JButton btnListar;
586 private javax.swing.JButton btnNuevo;
587 private javax.swing.JButton btnSalir;
588 private javax.swing.JLabel jLabel1;
589 private javax.swing.JLabel jLabel11;
590 private javax.swing.JLabel jLabel15;
591 private javax.swing.JLabel jLabel16;
592 private javax.swing.JLabel jLabel2;
593 private javax.swing.JLabel jLabel3;
594 private javax.swing.JScrollPane jScrollPane1;
595 public static javax.swing.JLabel labelDni;
596 private javax.swing.JTextField txtDni;
597 // End of variables declaration

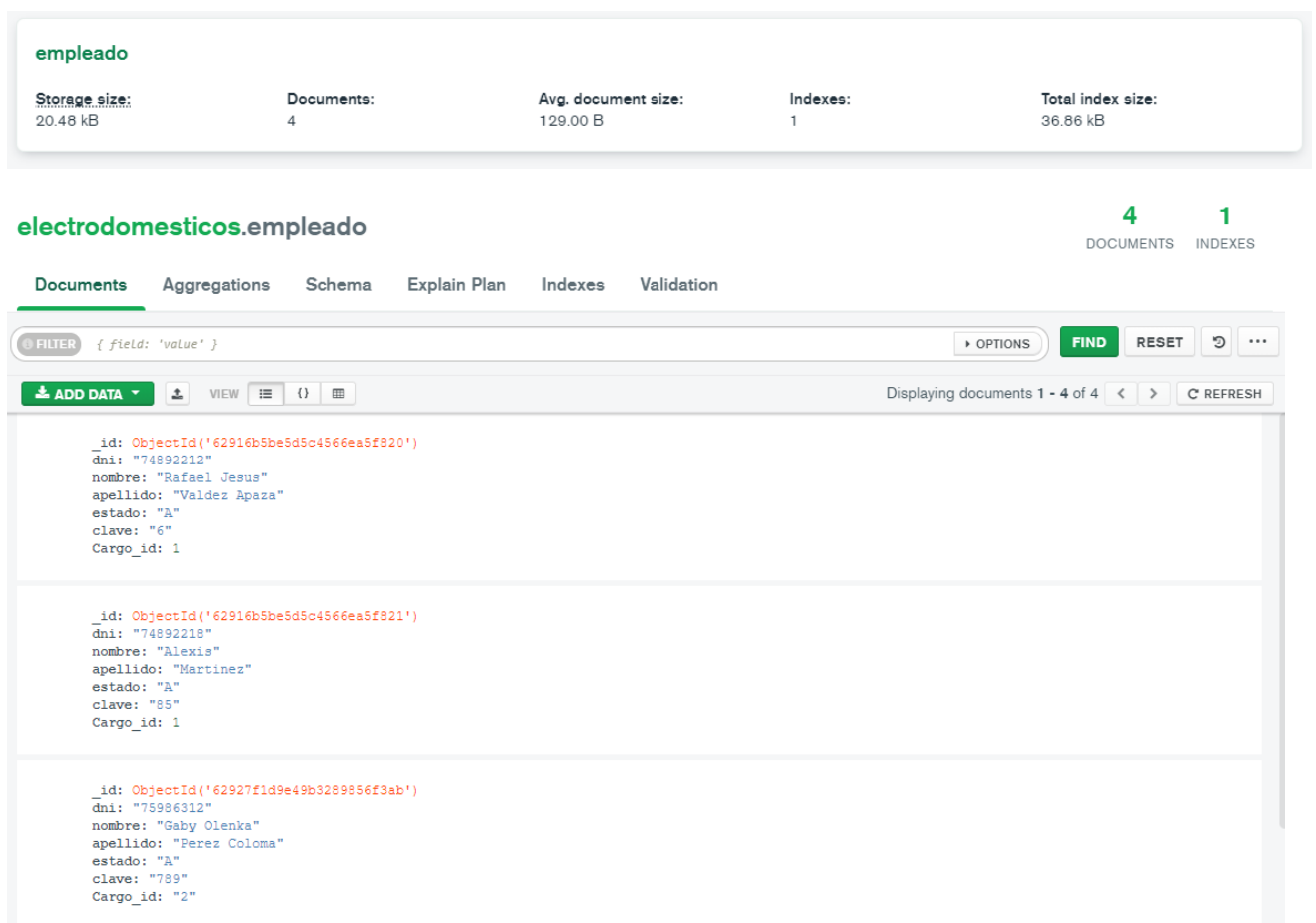
```

7.2 Implementar una parte de su proyecto utilizando una base de datos NoSQL, puede ser un requerimiento funcional o no funcional

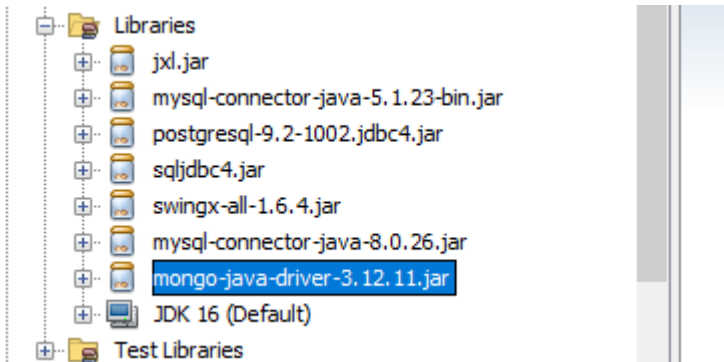
Primero se creó una base de datos llamada electrodomésticos, mismo nombre de la bd SQL.



Después se creó una colección llamada empleado, la cual almacenara documentos con los datos de cada empleado.



Luego en NetBeans se importó el controlador para usar mongodb dentro de nuestro sistema.



Creamos una clase de conexión donde realizamos la conexión con nuestra base de datos “Electrodomésticos”

```
6 package Negocio;
7
8 import com.mongodb.BasicDBObject;
9 import com.mongodb.DB;
10 import com.mongodb.DBCollection;
11 import com.mongodb.DBCursor;
12 import com.mongodb.MongoClient;
13 import com.mongodb.MongoException;
14 import java.util.List;
15 import javax.swing.JOptionPane;
16
17
18 import java.util.logging.Level;
19 import java.util.logging.Logger;
20
21
22 /**
23  *
24  * @author Rafael
25  */
26 public class ClsConexionMongo {
27     public DB db;
28     public DBCollection tabla;
29     public DBCursor cursos=null;
30     public BasicDBObject document=new BasicDBObject();
31
32
33     public static MongoClient conectar(){
34         MongoClient mongo=null;
35         String server="localhost";
36         Integer puerto=27017;
37
38         try {
39             Logger.getLogger("org.mongodb.driver").setLevel(Level.WARNING);
40             mongo=new MongoClient(server,puerto);
41             // db=mongo.getDB("electrodomesticos");
42             // tabla=db.getCollection("empleado");
43
44         } catch (Exception e) {
45             JOptionPane.showMessageDialog(null, "Error"+e.toString());
46         }
47
48         return mongo;
49     }
50
51 }
```

El requerimiento que seleccionamos para utilizar una base de datos NoSQL fue la de gestionar empleado.

Empleado DNI: Nro DNI
Fecha: dd-MM-YYYY
Hora: hh--mm

MANTENIMIENTO DE EMPLEADOS

Listar

Title 1	Title 2	Title 3	Title 4

Ingrese dni:

Buscar

Limpiar

Estado:

Cambiar

Eliminar

Salir

Nuevo

Función Listar

```

19  import com.mongodb.BasicDBObject;
20  import com.mongodb.DB;
21  import com.mongodb.DBCollection;
22  import com.mongodb.DBCursor;
23  import com.mongodb.MongoClient;
24  /**
25   *
26   * @author Arnold
27   */
28  public class ClsNEmpleado {
29      public DB db;
30      public DBCollection tabla;
31      public DBCursor cursor;
32
33
34      MongoClient mongo=ClsConexionMongo.conectar();
35
36
37      public DBCursor Listar(){
38
39          db=mongo.getDB("electrodomesticos");
40          tabla=db.getCollection("empleado");
41          cursor=tabla.find();
42
43          return cursor;
44      }
45  }
46

```

Procedimiento que invoca a la función Listar

```

private void MtdListar() {
    DBCursor cursor;
    cursor=objNEmp.Listar();

    while(cursor.hasNext()){

        modelo.addRow(new Object[]{cursor.next().get("dni"),cursor.curr().get("nombre"),cursor.curr().get("apellido"),cursor.curr().get("Cargo_id"),
            cursor.curr().get("estado"),cursor.curr().get("clave")});
        Tabla.setModel(modelo);
    }
}

```

Función Buscar

```

68 public DBCursor Buscar(ClsEEmpleado objEC){
69     db=mongo.getDB("electrodomesticos");
70     tabla=db.getCollection("empleado");
71
72     BasicDBObject documento=new BasicDBObject();
73     documento.put("dni",objEC.getDni());
74
75     cursor=tabla.find(documento);
76     return cursor;
77 }
78

```

Procedimiento que invoca a la función Buscar

```

private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    MtdlimpiarTabla();
    ResultSet rs;

    //
    // objEEmp.setB(txtDni.getText());
    // int flag = 0;

    ClsNEmpleado objN=new ClsNEmpleado();
    ClsEEmpleado objE=new ClsEEmpleado();
    objE.setDni(txtDni.getText());

    DBCursor cursor;
    cursor=objN.Buscar(objE);

    while(cursor.hasNext()){

        modelo.addRow(new Object[]{cursor.next().get("dni"),cursor.curr().get("nombre"),cursor.curr().get("apellido"),cursor.curr().get("Cargo_id"),
            cursor.curr().get("estado"),cursor.curr().get("clave")});
        Tabla.setModel(modelo);
    }
}

```

Función Eliminar

```

80 public Boolean Eliminar(ClsEEmpleado objEEmp){
81     db=mongo.getDB("electrodomesticos");
82     tabla=db.getCollection("empleado");
83
84     BasicDBObject documento=new BasicDBObject();
85     try {
86         documento.put("dni", objEEmp.getDni());
87         tabla.remove(documento);
88         return true;
89     } catch (Exception e) {
90         return false;
91     }
92 }

```

Procedimiento que invoca a la función Eliminar

```
private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int fila = Tabla.getSelectedRow();
    if(fila>=0){
        objNEmp.MtdEliminar(Integer.parseInt(Tabla.getValueAt(fila, 0).toString()));
        String dni=Tabla.getValueAt(fila, 0).toString();

        ClsNEmpleado objN=new ClsNEmpleado();
        ClsEEmpleado objE=new ClsEEmpleado();
        objE.setDni(dni);

        if(objN.Eliminar(objE)==true){
            JOptionPane.showMessageDialog(null,"Dato Eliminado");
            MtdlimpiarTabla();
            MtdListar();
        }
    }
}
```

LOGIN

DNI usuario:

Contraseña:

Función Loguin

```
26 public class ClsControlLogin {
27     public DB db;
28     public DBCollection tabla;
29     public DBCursor cursor;
30
31     MongoClient mongo=ClsConexionMongo.conectar();
32
33     public DBCursor Loguin(ClsEEmpleado objEC){
34         db=mongo.getDB("electrodomesticos");
35         tabla=db.getCollection("empleado");
36
37         BasicDBObject documento=new BasicDBObject();
38         documento.put("dni",objEC.getDni());
39         documento.put("clave",objEC.getClave());
40
41
42         cursor=tabla.find(documento);
43
44
45         return cursor;
46     }
47 }
```

Procedimiento de Logueo

```

182
183     ClsControlLogin objCl=new ClsControlLogin();
184
185     ClsEEmpleado objEmp=new ClsEEmpleado();
186     objEmp.setDni(txtDni.getText());
187     objEmp.setClave(txtClave.getText());
188
189     DBCursor cursor;
190     cursor=objCl.Loguin(objEmp);
191
192     String estado="";
193     String cargo="";
194
195     if(cursor.hasNext()){
196         System.err.println(""+cursor.next().get("Cargo_id"));
197         estado="" + cursor.curr().get("estado");
198         cargo="" + cursor.curr().get("Cargo_id");
199     }
200
201     if(cursor.count()==1){
202         while(cursor.hasNext()){
203
204             if(estado.equals("A")){
205
206                 switch(cargo)
207                 {
208                     case "1" :
209                         JOptionPane.showMessageDialog(null,"inicia Sistema");
210                         FrmEmpleado FEmp = new FrmEmpleado();
211                         FEmp.labelDni.setText(txtDni.getText());
212                         FEmp.show();
213                         this.dispose();
214                         break;
215                     case "2" :
216                         JOptionPane.showMessageDialog(null,"inicia Sistema");
217                         FrmVenta frm =new FrmVenta();
218                         FrmVenta.labelDniEmpleado.setText(txtDni.getText());
219                         frm.show();
220                         this.dispose();
221                         break;
222                     default :
223
224                 }
225             }
226             else{
227                 JOptionPane.showMessageDialog(null,"Cuenta suspendida ");
228             }
229         }
230         else{
231             JOptionPane.showMessageDialog(null,"Cuenta incorrecta");
232         }
233     }

```

REGISTRAR EMPLEADOS

DNI:

Nombres:

Apellidos:

Cargo:

Estado:

Clave:

Función Insertar

```

48 public boolean Insertar(ClsEEmpleado objEEmp){
49     db=mongo.getDB("electrodomesticos");
50     tabla=db.getCollection("empleado");
51
52     BasicDBObject documento=new BasicDBObject();
53
54     try {
55         documento.put("dni", objEEmp.getDni());
56         documento.put("nombre", objEEmp.getNombres());
57         documento.put("apellido", objEEmp.getApellidos());
58         documento.put("estado", objEEmp.getEstado());
59         documento.put("clave", objEEmp.getClave());
60         documento.put("Cargo_id", objEEmp.getCargo());
61         tabla.insert(documento);
62         return true;
63     } catch (Exception e) {
64         return false;
65     }
66 }

```

Botón de invocación

7.3 DIAGRAMA DE BASE DE DATOS

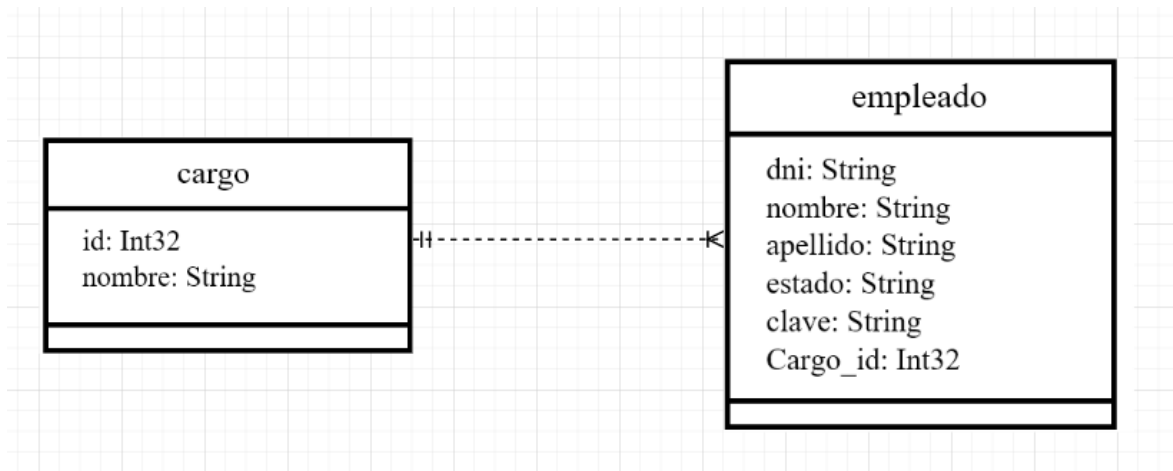
El diagrama de base de datos relacional para el sistema de ventas se compone de las siguientes tablas y relaciones:

- Tabla Cliente:** id INT (PK), dni VARCHAR(10), nombre VARCHAR(45), apellido VARCHAR(45), email VARCHAR(45), clave VARCHAR(45).
- Tabla Venta:** id INT (PK), total DOUBLE, fecha DATETIME, estado VARCHAR(10), Transporte_id INT (FK), Cliente_id INT (FK).
- Tabla Transporte:** id INT (PK), nombre VARCHAR(45), email VARCHAR(45), telefono VARCHAR(45), costo DOUBLE.
- Tabla Pago:** id INT (PK), nrotarjet VARCHAR(25), Cliente_id INT (FK), Venta_id INT (FK).
- Tabla Empleado:** id INT (PK), dni VARCHAR(10), nombre VARCHAR(45), apellido VARCHAR(45), estado CHAR(1), clave VARCHAR(5), Cargo_id INT (FK).
- Tabla Producto:** id INT (PK), nombre VARCHAR(45), stock INT, stock_referencial INT, precio_venta DOUBLE, precio_compra DOUBLE, estado CHAR(1), foto VARCHAR(255), Marca_id INT (FK), Categoria_id INT (FK), Empleado_id INT (FK).
- Tabla Detalle_Venta:** id INT (PK), cantidad INT, total_prod DOUBLE, Venta_id INT (FK), Producto_id INT (FK).
- Tabla Marca:** id INT (PK), nombre VARCHAR(45), email VARCHAR(45), telefono VARCHAR(45).
- Tabla Categoria:** id INT (PK), nombre VARCHAR(45).

Relaciones y cardinalidad:

- Cliente (1) a Venta (N): 1:N
- Venta (1) a Transporte (1): 1:1
- Venta (1) a Pago (N): 1:N
- Pago (1) a Empleado (N): 1:N
- Empleado (1) a Producto (N): 1:N
- Producto (1) a Detalle_Venta (N): 1:N
- Detalle_Venta (1) a Producto (N): 1:N
- Producto (1) a Marca (N): 1:N
- Producto (1) a Categoria (N): 1:N

Requerimiento Gestionar Empleado:



8. BIBLIOGRAFIA

- ESIC Business & Marketing School. (2018). El ORM como herramienta eficiente de trabajo. Esic.edu; ESIC. Recuperado de <https://www.esic.edu/rethink/tecnologia/el-orm-como-herramienta-eficiente-de-trabajo>
- ¿Qué es un ORM? Código Facilito. Recuperado de <https://codigofacilito.com/articulos/orm-explicacion>
- ¿Qué es un ORM? Programarfacil.com. Recuperado de <https://programarfacil.com/blog/que-es-un-orm/>
- ¿Qué es un ORM? (2018). Deloitte Spain. Recuperado de <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>
- ¿Cuándo usar un ORM? (2018). Deloitte Spain. Recuperado de <https://www2.deloitte.com/es/es/pages/technology/articles/cuando-usar-orm.html>

9. ANEXOS

9.1. DICCIONARIO DE DATOS DE SU BASE DE DATOS RELACIONAL

cargo

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id <i>(Primaria)</i>	int(11)	No				
nombre	varchar(45)	No				

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	3	A	No	

categoria

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id <i>(Primaria)</i>	int(11)	No				
nombre	varchar(45)	No				

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	2	A	No	

cliente

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id <i>(Primaria)</i>	int(11)	No				
dni	varchar(10)	No				
nombre	varchar(45)	Sí	NULL			
apellido	varchar(45)	Sí	NULL			
email	varchar(45)	No				
clave	varchar(45)	No				

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	0	A	No	

detalle_venta

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
cantidad	int(11)	Sí	<i>NULL</i>			
total_prod	double	Sí	<i>NULL</i>			
Venta_id	int(11)	No		venta -> id		
Producto_id	int(11)	No		producto -> id		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	2	A	No	
fk_Detalle_Venta_Venta1_idx	BTREE	No	No	Venta_id	2	A	No	
fk_Detalle_Venta_Producto1_idx	BTREE	No	No	Producto_id	2	A	No	

empleado

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
dni	varchar(10)	No				
nombre	varchar(45)	Sí	<i>NULL</i>			
apellido	varchar(45)	Sí	<i>NULL</i>			
estado	char(1)	Sí	<i>NULL</i>			
clave	varchar(5)	Sí	<i>NULL</i>			
Cargo_id	int(11)	No		cargo -> id		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	1	A	No	
fk_Empleado_Cargo1_idx	BTREE	No	No	Cargo_id	1	A	No	

marca

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
nombre	varchar(45)	Sí	<i>NULL</i>			
email	varchar(45)	Sí	<i>NULL</i>			
telefono	varchar(45)	Sí	<i>NULL</i>			

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	2	A	No	

pago

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
nrotarjet	varchar(25)	Sí	<i>NULL</i>			
Cliente_id	int(11)	No		cliente -> id		
Venta_id	int(11)	No		venta -> id		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id	0	A	No	
fk_Tarjeta_Cliente1_idx	BTREE	No	No	Cliente_id	0	A	No	
fk_Tarjeta_Venta1_idx	BTREE	No	No	Venta_id	0	A	No	

producto

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
nombre	varchar(45)	Si	NULL			
stock	int(11)	Si	NULL			
stock_referencial	int(11)	Si	NULL			
precio_venta	double	Si	NULL			
precio_compra	double	Si	NULL			
estado	char(1)	Si	NULL			
foto	varchar(255)	Si	NULL			
descripcion	varchar(255)	Si	NULL			
Marca_id	int(11)	No		marca -> id		
Categoria_id	int(11)	No		categoria -> id		
Empleado_id	int(11)	No		empleado -> id		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id	9	A	No	
fk_Producto_Categoria1_idx	BTREE	No	No	Categoria_id	9	A	No	
fk_Producto_Marca1_idx	BTREE	No	No	Marca_id	9	A	No	
fk_Producto_Empleado1_idx	BTREE	No	No	Empleado_id	2	A	No	

transporte

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
nombre	varchar(45)	No				
email	varchar(45)	Si	NULL			
telefono	varchar(45)	Si	NULL			
costo	double	No				

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id	0	A	No	

venta

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Media type
id (<i>Primaria</i>)	int(11)	No				
total	double	Si	NULL			
fecha	datetime	Si	NULL			
estado	varchar(10)	Si	NULL			
Transporte_id	int(11)	No		transporte -> id		
Cliente_id	int(11)	No		cliente -> id		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id	0	A	No	
fk_Venta_Transporte1_idx	BTREE	No	No	Transporte_id	0	A	No	
fk_Venta_Cliente1_idx	BTREE	No	No	Cliente_id	0	A	No	

9.1.1. NOMBRE DE BASE DE DATOS, DESCRIPCION, TIPO RELACIONAL

NOMBRE DE BASE DE DATOS	DESCRIPCION	TIPO RELACIONAL
electrodomésticos	Base de Datos transaccional. Se encuentra almacenado todos los datos de las tablas del Sistema y Aplicación.	MySQL

9.1.2. TABLAS CON LAS SIGUIENTES CAMPOS: NOMBRE DE COLUMNA, TIPO DE DATO, PERMITE NULOS, DESCRIPCION, CONSTRAINT, DEFAULT

cargo					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Numero de Id del cargo	PRIMARY	NINGUNA
nombre	Varchar(45)	NO	Nombre del cargo		NINGUNA

categoría					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Numero de id de categoría	PRIMARY	NINGUNA
nombre	Varchar(45)	NO	nombre de la categoría		NINGUNA

cliente					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Número del id del cliente	PRIMARY	NINGUNA
dni	Varchar(10)	NO	DNI del cliente		NINGUNA
nombre	Varchar(10)	SI	Nombre del cliente		NULL
apellido	Varchar(10)	SI	Apellido del Cliente		NULL

email	Varchar(10)	NO	Correo Electrónico del Cliente		NINGUNA
contraseña	Varchar(10)	NO	Contraseña del Cliente		NINGUNA

detalle_venta					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Id de detalles de venta	PRIMARY	NINGUNA
cantidad	Int(11)	SI	Cantidad de detalle de ventas		NULL
Total_prod	double	SI	Total, de los productos de los detalles de ventas		NULL
Venta_id	Int(11)	NO	Número del Id de Venta	Fk_Detalle_Venta_Venta	NINGUNA
Producto_id	Int(11)	NO	Número del id del Producto	Fk_Detalle_Venta_Producto	NINGUNA

empleado					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Numero de id del empleado	PRIMARY	NINGUNA
dni	Varchar(10)	NO	DNI del empleado		NINGUNA
nombres	Varchar(45)	SI	Nombres del Empleado		NULL
apellidos	Varchar(45)	SI	Apellidos del Empleado		NULL
estado	Varchar(1)	SI	Estado del Empleado		NULL
clave	Varchar(5)	SI	Contraseña del Empleado		NULL
Cargo_id	Int(11)	NO	Numero de Id de Cargo	FK_Empleado_Cargo	NINGUNA

proveedor					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT

id	Int(11)	NO	Numero de Id de proveedor	PRIMARY	NINGUNA
nombre	Varchar(45)	SI	Nombre del Proveedor		NULL
email	Varchar(45)	SI	Correo Electrónico del Proveedor		NULL
telefono	Varchar(45)	SI	Teléfono del Proveedor		NULL

metodo_pago					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMIT E NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Numero de Id del Método de PAGO	PRIMARY	NINGUNA
nrotarjet	Varchar(25)	SI	Número de Tarjeta		NULL
fechaven	date	SI	Fecha		NULL
cvv	Varchar(4)	SI	Código		NULL
Cliente_id	Int(11)	NO	Número del Id Del Cliente	FK_Tarjeta_Cliente	NINGUNA
Venta_id	Int(11)	NO	Numero de Id de Venta	FK_Tarjeta_Venta	NINGUNA

producto					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMIT E NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Id del producto	PRIMARY	Ninguna
nombre		SI	Nombre del producto		NULL
stock	Int(11)	SI	Stock del producto		NULL
Stock_referencial	Int(11)	SI	Numero referencial de la cantidad del producto		NULL
Precio_venta	double	SI	Venta del producto		NULL
Precio_compra	double	SI	Compra del producto		NULL
estado	Char(1)	SI	estado		NULL
foto	longblob	SI	foto		NULL

descripcion	Varchar(255)	SI	Descripción del producto		NULL
Proveedor_id	Int(11)	NO	Numero de id del Proveedor	FK_Producto_Proveedor	Ninguna
Categoria_id	Int(11)	NO	Numero id de Categoría	FK_Producto_Categoria	Ninguna
Empleado_id	Int(11)	NO	Numero id del Empleado	FK_Producto_Empleado	Ninguna

transporte					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITIR NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	NO	Numero de Id de Transporte	PRIMARY	NINGUNA
nombre	Varchar(25)	NO	Nombre del Transporte		NINGUNA
email	Varchar(25)	SI	Correo Electrónico de Transporte		NULL
teléfono	Varchar(25)	SI	Teléfono de Transporte		NULL
costo	double	NO	Costo del Transporte		NINGUNA

venta					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITIR NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int(11)	No	Numero de Id de Venta	PRIMARY	NINGUNA
total	double	Si	Número Total		NULL
fecha	datetime	Si	Fecha		NULL
estado	Varchar(10)	Si	Estado		NULL
Transporte_id	Int(11)	No	Numero Id de Transporte	FK_Venta_Transporte	NINGUNA
Cliente_id	Int(11)	No	Numero id del Cliente	FK_Venta_Cliente	NINGUNA

9.2. DICCIONARIO DE DATOS DE SU BASE NOSQL

Nombre de la Colección:			empleado			
Descripción de la Colección:			Esta colección se encarga de almacenar documentos con los datos de los empleados registrados.			
Objetivo:			Almacenar datos de los empleados.			
Relaciones con otras Tablas:			Posee una clave foránea de la colección cargo.			
Descripción de los campos						
Nro.	Nombre del campo	Tipo dato longitud	Permite nulos	Clave primaria	Clave foránea	Descripción del campo
1	dni	String	NO			Dni del empleado.
2	nombre	String	NO			Nombre del empleado.
3	apellido	String	NO			Apellido del empleado.
4	estado	String	NO			Estado del empleado (Inactivo o Activo).
5	clave	String	NO			Contraseña del empleado para ingresar al sistema.
6	Cargo_id	Int32	NO		FK	Clave foránea de la colección cargo.

Nombre de la Colección:			cargo			
Descripción de la Colección:			Esta colección se encarga de almacenar documentos con los datos de los cargos disponibles para los empleados.			
Objetivo:			Almacenar los cargos disponibles.			
Relaciones con otras Tablas:			Su id es clave foránea en la colección empleado.			
Descripción de los campos						
Nro.	Nombre del campo	Tipo dato longitud	Permite nulos	Clave primaria	Clave foránea	Descripción del campo
1	id	Int32	NO	PK		Id del documento.
2	nombre	String	NO			Nombre del cargo.

9.2.1. NOMBRE DE BASE DE DATOS, DESCRIPCION, TIPO RELACIONAL

NOMBRE DE BASE DE DATOS	DESCRIPCION	TIPO RELACIONAL
electrodomesticos	Base de Datos NoSQL dentro de mongodb, la cual almacena la colección empleado y cargo.	NoSQL

9.2.2. TABLAS CON LAS SIGUIENTES CAMPOS: NOMBRE DE COLUMNA, TIPO DE DATO, PERMITE NULOS, DESCRIPCION, CONSTRAINT, DEFAULT

empleado					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
dni	String	NO	DNI del empleado		NINGUNA
nombre	String	SI	Nombres del Empleado		NULL
apellido	String	SI	Apellidos del Empleado		NULL
estado	String	SI	Estado del Empleado		NULL
clave	String	SI	Contraseña del Empleado		NULL
Cargo_id	Int32	NO	Numero de Id de Cargo	FK_Empleado_Cargo	NINGUNA

cargo					
NOMBRE DE COLUMNA	TIPO DE DATO	PERMITE NULOS	DESCRIPCION	CONSTRAINT	DEFAULT
id	Int32	NO	Numero de Id del cargo	PRIMARY	NINGUNA
nombre	String	NO	Nombre del cargo		NINGUNA

9.3. ESTANDAR DE PROGRAMACION

electrodomésticos

1. Declaración de variables

Se propone que la declaración de las variables, se ajusten al motivo para la que se requieran. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente:

- La longitud debe ser lo más recomendable posible. No debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 20 caracteres.

- Alcance de la variable

No tenemos un prefijo definido para cada variable en específico por lo que nuestra consideración la hacer una variable es que además de la relación que debe tener con la clase esta se describa a primera vista de que trata esa variable.

- El tipo de dato al que pertenece la variable.

Por lo tanto, la estructura de la variable es como sigue:

Estructura	Descripción de la Variable
Longitud máx.	□ 1 □□ 100 □
Formato	<i>Dependiendo de los productos que desee comprar el cliente, o el número de clientes como tal.</i>
Ejemplo	Registro_De_Venta[] MT = new Registro_De_Venta[100]; Producto[] DA = new Producto[100];
Estructura	Descripción de la Variable
Longitud máx.	□ 1 □□ 100 □
Formato	<i>Dependiendo de los productos que desee comprar el cliente, o el número de clientes como tal.</i>
Ejemplo	Registro_De_Empleado[] MT = new Registro_De_Empleado[100]; Empleado[] DA = new Empleado[100];

1.1 Descripción de la Variable.

Nombre que se le asignará a la variable para que se le identifique y deberá de estar asociada al motivo para la cual se le declara.

- nombre; private String, Registrar Nombre de Producto
- marca; private String, Registrar Marca de Producto
- etiqueta; private double, Registrar Etiqueta Producto
- fecha; private String, Registrar Fecha de Producto
- precio; private double, Registrar Precio de Producto
- cantidad; private double, Registrar Cantidad de Productos
- nombre_cliente; private String, Registrar Nombre Cliente
- apellido_cliente; private String, Registrar Apellido Cliente
- dni; protected String, Registrar DNI
- ID_ListadoProducto; private String, Registrar ID Producto
- nproductos = 0; private Int , Registrar Productos
- Total_De_Productos = 0; protected double, Contar productos
- nombres; private String, Registrar Nombre de Empleado

- apellidos; private String, Registrar Apellidos de Empleado
- cargo; private String, Registrar Cargo de Empleado
- estado; private String, Registrar Estado de Empleado
- clave; private String, Registrar Clave de Empleado
- vrt; String,
- f=false; bool, Confirmación
- op,op1; Int , Variable de opciones múltiples
- n=0; Int, contador
- m=0; Int, contador

1.2 Variables de Tipo Arreglo

En el caso de las definiciones de arreglos de elementos se declarará la variable con el prefijo de “lista”, el cual nos dará entender que se trata de una variable del tipo arreglo la cual contendrá de cero a más datos, según el tamaño declarado.

- listaMT; public; Lista para registrar clientes
- listaDA; public; Lista para registrar ventas de los productos
- listamod; public; Lista para modificaciones de los registros de ventas
- listasecciones; private; Lista de los productos para calcularlos en base a la modificación, aumento o eliminación del registro del producto
- listaME; public; Lista para registrar empleados
- listamode; public; Lista para modificaciones de los empleados

2. Definición de Controles

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo al cual pertenece y la función que cumple dentro de la aplicación.

2.1 Tipo de datos

Tipo de variable	Mnemónico	Descripción
Byte	by	Entero de 8 bits sin signo.
Integer	in	Entero de 32 bits con signo.
Char	ch	Un carácter UNICODE de 16 bits
String	st	Cadena de caracteres
Date	dt	Formato de fecha/hora
Boolean	bl	Valor lógico: verdadero y falso
Float	fl	Comas flotantes, 11-12 dígitos significativos.
Double	db	Coma flotante, 64 bits (15-16 dígitos significativos)
Object	ob	Objeto genérico

2.2 Declaración de variables, atributos y objetos

1. Se debe declarar una variable por línea.

Nombre	Descripción
Sintaxis	[String] [Nombre]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres. El nombre de la variable puede incluir solo 2 nombres de la persona.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,].
Ejemplo	Public String nombre Indica una variable o atributo que guardará un nombre.

Marca	Descripción
Sintaxis	[String] [marca]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 20 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,].
Ejemplo	Public String marca Indica una variable o atributo que guardará una marca para un producto.

Etiqueta	Descripción
Sintaxis	[double] [etiqueta]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 2. Letra Ñ o ñ.

	<p>3. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ' , +, -, *, {, }, [,].</p> <p>4. Caracteres tildados: á, é, í, o, ú.</p>
Ejemplo	<p>Public double etiqueta</p> <p>Indica una variable o atributo que guardará una etiqueta de un producto.</p>

Fecha	Descripción
Sintaxis	[String] [fecha]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de variables o atributos no se deberá utilizar caracteres como:</p> <p>5. Letra Ñ o ñ.</p> <p>6. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ' , +, -, *, {, }, [,].</p> <p>7. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	<p>Public String fecha</p> <p>Indica una variable o atributo que guardará una fecha del producto.</p>

Precio	Descripción
Sintaxis	[double] [precio]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de variables o atributos no se deberá utilizar caracteres como:</p> <p>8. Letra Ñ o ñ.</p> <p>9. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ' , +, -, *, {, }, [,].</p> <p>10. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	<p>Public Double precio</p> <p>Indica una variable o atributo que guardará un valor del precio del Producto.</p>

Cantidad	Descripción
Sintaxis	[double] [cantidad]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de variables o atributos no se deberá utilizar caracteres como:</p> <p>11. Letra Ñ o ñ.</p> <p>12. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ' , +, -, *, {, }, [,].</p> <p>13. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	Public Double cantidad

	Indica una variable o atributo que guardará la cantidad de Productos.
--	---

Nombre del Cliente	Descripción
Sintaxis	[String] [nombre_cliente]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,].
Ejemplo	Public String nombre_cliente Indica una variable o atributo que guardará un nombre de Cliente de los Productos.

Apellido del Cliente	Descripción
Sintaxis	[String] [apellido_cliente]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,].
Ejemplo	Public String apellido_cliente Indica una variable o atributo que guardará un apellido de Cliente de los Productos.

DNI	Descripción
Sintaxis	[String] [dni]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 14. Letra Ñ o ñ. 15. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 16. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String dni Indica una variable o atributo que guardará un dni del Cliente de los Productos.

ID del Listado del Producto	Descripción
Sintaxis	[String] [ID_ListadoProducto]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 17. Letra Ñ o ñ. 18. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 19. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String ID_ListadoProducto Indica una variable o atributo que guardará un ID del listado de los Productos pertenecientes al Cliente.

Cantidad de Productos	Descripción
Sintaxis	[Int] [nproductos]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 20. Letra Ñ o ñ. 21. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 22. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public Int nproductos Indica una variable o atributo que guardará un numero de Productos registrados por el sistema.

Totalidad de los Productos	Descripción
Sintaxis	[double] [Total_De_Productos]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 23. Letra Ñ o ñ. 24. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 25. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public double Total_De_Productos Indica una variable o atributo que guardará la suma total de los Productos de la Lista del Cliente.

Totalidad de los Productos	Descripción
----------------------------	-------------

Sintaxis	[String] [nombres]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String nombres Indica una variable o atributo que guardará los nombres del empleado.

Totalidad de los Productos	Descripción
Sintaxis	[String] [apellidos]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String apellidos Indica una variable o atributo que guardará los apellidos del empleado.

Totalidad de los Productos	Descripción
Sintaxis	[String] [cargo]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String cargo Indica una variable o atributo que guardará el cargo del empleado.

Totalidad de los Productos	Descripción
Sintaxis	[String] [estado]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.

Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String estado Indica una variable o atributo que guardará el estado del empleado.

Totalidad de los Productos	Descripción
Sintaxis	[String] [clave]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String clave Indica una variable o atributo que guardará la clave del empleado.

Verificación del ID del Cliente	Descripción
Sintaxis	[String] [vrt]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 4. Letra Ñ o ñ. 5. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 6. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	String vrt Indica la confirmación del ID del cliente.

Comprobación del Id Cliente y De Productos	Descripción
Sintaxis	[bool] [f]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 7. Letra Ñ o ñ. 8. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,].

	9. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	bool f=false Verifica y Confirma el registro de clientes e Productos del Cliente.

Opciones de Selección	Descripción
Sintaxis	[Int] [op,op1]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 10. Letra Ñ o ñ. 11. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 12. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Int op,op1 Indica la elección de las opciones que tienen en los menús.

Contador Principal	Descripción
Sintaxis	[Int] [n=0]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 13. Letra Ñ o ñ. 14. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 15. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Int n=0 Indica el conteo de cada registro de cliente

Contador Secundario	Descripción
Sintaxis	[Int] [m=0]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 16. Letra Ñ o ñ. 17. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 18. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Int m=0 Indica el conteo de cada registro de producto perteneciente al cliente.

Listado de Registro de Clientes	Descripción
Sintaxis	[public] [MT]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 19. Letra Ñ o ñ. 20. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 21. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Registro_De_Venta[] MT = new Registro_De_Venta[100] Almacena los datos de los clientes

Listado de Registro de los Productos	Descripción
Sintaxis	[public] [DA]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 22. Letra Ñ o ñ. 23. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 24. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Producto[] DA = new Producto[100] Almacena cantidad datos de los productos de cada lista de cada cliente

Listado de las Modificaciones para los Productos	Descripción
Sintaxis	[public] [mod]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 25. Letra Ñ o ñ. 26. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 27. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Producto mod = new Producto() Permite modificar los productos de una lista del cliente

Listado de Productos Registrados para el Calculo	Descripción
Sintaxis	[private] [secciones]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: 28. Letra Ñ o ñ. 29. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 30. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	private Producto[] secciones = new Producto[100] Almacena datos de los cálculos de los productos

2.3 Declaración de clases

Registro de Cliente	Descripción
Sintaxis	[] Class [Registro_De_Venta]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: 31. Letra Ñ o ñ. 32. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 33. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Class Registro_De_Venta Indica una clase Registro de Venta que hace que se registre la venta de un cliente a manos del usuario.

Producto	Descripción
Sintaxis	[] Class [Producto]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: 34. Letra Ñ o ñ. 35. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 36. Caracteres tildados: á, é, í, ó, ú.

Ejemplo	Class Producto Indica una clase Producto para ser registrado por el usuario
----------------	--

Producto	Descripción
Sintaxis	[] Class [Registro_De_Empleado]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Class Registro_De_Empleado Indica una clase empleado para ser registrado por el administrador

Producto	Descripción
Sintaxis	[] Class [Empleado]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 4. Letra Ñ o ñ. 5. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. 6. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Class Registro_De_Empleado Indica una clase empleado para ser registrado por el administrador

2.4 Declaración de métodos

Registro del Producto	Descripción
Sintaxis	Agregar_Empleado [(Vacio)]
Descripción	El nombre del método constará hasta de 25 caracteres. La primera letra de la primera palabra del nombre será escrita en minúscula y las siguientes palabras empezarán con letra mayúscula.

Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	<pre>public void AgregarEmpleado()</pre> <p>Indica un método AgregarEmpleado, que recibe varias variables por valor de Tipo Int y String para el registro del producto.</p>

AgregarEmpleado()

Eliminar_Empleado ()

Cambiar_Estado ()

Listar_Empleado ()

Buscar_Empleado ()

Eliminación del Producto	Descripción
Sintaxis	Eliminar_Empleado [(int Id, String)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,], _. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	<pre>public void Eliminar_Empleado (int Id, String)</pre> <p>Se incluye el método Eliminar para borrar el empleado.</p>

Cambiar Estado	Descripción
Sintaxis	Cambiar_Estado[(String)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <ol style="list-style-type: none"> 1. Letra Ñ o ñ.

	2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Cambiar_Estado (String) Se incluye el método cambiar para modificar el estado del empleado.

Listar Empleado	Descripción
Sintaxis	Listar_Empleado[(String)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Listar_Empleado (Id, String) Se incluye el método listar para mostrar los empleados registrados.

Buscar Empleado	Descripción
Sintaxis	Buscar_Empleado[(Id, String)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 1. Letra Ñ o ñ. 2. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. 3. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Buscar_Empleado (Id, String) Se incluye el método buscar para encontrar el registro de un empleado.

Registro del Producto	Descripción
Sintaxis	AgregarProducto [(Vacio)]
Descripción	El nombre del método constará hasta de 25 caracteres. La primera letra de la primera palabra del nombre será escrita en minúscula y

	las siguientes palabras empezarán con letra mayúscula.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 4. Letra Ñ o ñ. 5. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 6. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void AgregarProducto() Indica un método AgregarProducto, que recibe varias variables por valor de Tipo String, Int y double para el registro del producto.

Cálculo del Precio	Descripción
Sintaxis	CalcularPrecio [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 7. Letra Ñ o ñ. 8. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void CalcularPrecio() Indica un método CalcularPrecio que recibe una variable por valor de tipo double al cálculo del precio del producto

Visualización del Producto	Descripción
Sintaxis	mostrar_Producto [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: <ol style="list-style-type: none"> 9. Letra Ñ o ñ. 10. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 11. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void mostrar_Producto() Indica un método mostrar_Producto que recibe varias variables para visualizarlas en la pantalla dentro de un listado

Totalidad del Precio	Descripción
Sintaxis	Precio_total[(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.

Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 12. Letra Ñ o ñ. 13. Caracteres especiales ;, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 14. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public double Precio_total() Indica un método Precio_total que recibe una variable por valor de tipo double al precio total de los productos

Registro del Cliente	Descripción
Sintaxis	Agregar_Registro0 [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 15. Letra Ñ o ñ. 16. Caracteres especiales ;, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 17. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Agregar_Registro0() Indica un método Agregar_Registro0 que recibe varias variables de Tipo String para el registro del cliente

Visualización del Registro del Cliente	Descripción
Sintaxis	Mostrar_Registro0 [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 18. Letra Ñ o ñ. 19. Caracteres especiales ;, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 20. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Mostrar_Registro0() Indica un método Mostrar_Registro0 que recibe las variables ingresadas del Agregar_Registro0 para poder ser visualizadas en la pantalla.

Identificación de lista de Productos del Cliente	Descripción
Sintaxis	Agregar_Registro [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.

Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <p>21. Letra Ñ o ñ.</p> <p>22. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,].</p> <p>23. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	<p>public void Agregar_Registro()</p> <p>Indica un método Agregar_Registro que recibe una variable por valor de tipo String para la confirmación del ID de listado de productos del Cliente.</p>

Visualización del Listado de los Productos y El Total de la Venta	Descripción
Sintaxis	Mostrar_Registro [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <p>24. Letra Ñ o ñ.</p> <p>25. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,].</p> <p>26. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	<p>public void Mostrar_Registro()</p> <p>Indica un método Mostrar_Registro que recibe una variable ingresada de tipo String junto con la variable double del precio total de los productos para ser visualizadas</p>

Visualización de los Productos del Listado	Descripción
Sintaxis	mostrarProductos [(Vacio)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	<p>En la declaración de métodos no se deberá utilizar caracteres como:</p> <p>27. Letra Ñ o ñ.</p> <p>28. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,].</p> <p>29. Caracteres tildados: á, é, í, ó, ú.</p>
Ejemplo	<p>public void mostrarProductos()</p> <p>Indica un método mostrarProductos que recibe varias variables de tipo String y tipo double para ser mostradas en fila por cada producto registrado</p>

Cálculo de la Totalidad de los Productos	Descripción
Sintaxis	<code>Calcular_Total [(double c)]</code>
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 30. Letra Ñ o ñ. 31. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 32. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	<code>public void Calcular_Total (double c)</code> Indica un método <code>Calcular_Total</code> que recibe una variable por valor de tipo <code>double</code> para sumar la cantidad del precio por cada producto registrado que este dentro de la lista

Visualización del ID del Listado	Descripción
Sintaxis	<code>ID_Mostrar [(Vacio)]</code>
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 33. Letra Ñ o ñ. 34. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 35. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	<code>public String ID_Mostrar()</code> Indica el <code>ID_Mostrar</code> , que devuelve el valor de tipo <code>String</code> para mostrar la identificación del listado de los productos del cliente

Aumento de la cantidad de los Productos	Descripción
Sintaxis	<code>AdherirProducto [(Producto b)]</code>
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 36. Letra Ñ o ñ. 37. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 38. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	<code>public void AdherirProducto(Producto b)</code>

	Indica un método AdherirProducto que recibe la variable arreglo para agregar más productos para ser almacenados
--	---

Modificación de los Productos	Descripción
Sintaxis	Modificar [(Producto s, int Idm, double f1)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 39. Letra Ñ o ñ. 40. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,]. 41. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Modificar (Producto s, int Idm, double f1) Indica el método Modificar que recibe varias variables de tipo String y tipo double para modificar el producto halla dentro de la lista.

Eliminación del Producto	Descripción
Sintaxis	Eliminar [(int Idy, double f)]
Descripción	Todas las variables o atributos tendrán una longitud máxima de 30 caracteres.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: 42. Letra Ñ o ñ. 43. Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,], _. 44. Caracteres tildados: á, é, í, ó, ú.
Ejemplo	public void Eliminar (int Idy, double f) Se incluye el método Eliminar que se borrar(reemplazar) el producto y dejar vacío el producto eliminando y agregando otro producto más.

2.5 Control de versiones de código fuente

Cada modificación realizada será guardada de la forma:

Título	Descripción
Formato	[Registro De Ventas][_] [20210107][_][14:45:52].

Descripción	Se generará archivo con la siguientes extension:..cs. Proyecto_Final.cs
--------------------	--

Título	Descripción
Formato	[Registro De Empleados][_]][20210107][_][14:45:52].
Descripción	Se generará archivo con la siguientes extension:..cs. Proyecto_Final.cs

3. Clases.

El nombre de las clases no tiene prefijos para ser nombrados por que se guían de una lógica que tiene más concordancia con el tema así que no estamos usando los prefijos, ya que buscamos dar el nombre de la clase de manera más directa.

- Registro_De_Venta
- Producto
- Registro_De_Empleado
- Empleado

4. Métodos, procedimientos y funciones definidos por el Usuario.

- AgregarProducto()
- CalcularPrecio()
- Listar_Producto()
- Precio_total()

- AgregarEmpleado()
- Eliminar_Empleado ()
- Cambiar_Estado ()
- Listar_Empleado ()
- Buscar_Empleado ()

- Agregar_Registro0()
- Mostrar_Registro0()
- Mostrar_Registro_Cliente()
- Agregar_Registro()
- Mostrar_Registro()

- ListarProductos()
- Buscar_Productos ()
- Calcular_Total(double c)
- ID_Mostrar()
- AdherirProducto(Producto b)
- Modificar (Producto s, int Idm, double f1)
- Eliminar (int Idy, double f)

5. Beneficios

1. La documentación siempre es importancia para la mayor comprensión del Usuario, Programador y los que se ocupan del Mantenimiento.
2. Si se tiene una documentación bien elaborada, no habrá pierda en caso se necesite mantenimiento al sistema.
3. Ayuda a ver ciertos errores que podría presentar el código, dado a que al analizar todo al detalle, nuestro campo de visión aumenta significativamente.

6. Conclusiones

1. Documentar un programa, es sumamente importante, dado a que ayuda a detectar ciertas fallas que podría presentar el programa que se está desarrollando.
2. El manejo de las variables se debe de trabajarse con mucha cautela, dado a que son eje principal de todo el sistema.