

Guía lógica y con fines de mantenimiento  
del programa de administración de ventas y  
productos de INTELAF

# MANUAL TECNICO

PROGRAMA DE  
ADMINISTARCION DE  
PRODUCTOS Y VENTAS DE  
INTELAF

Desarrollador Carlos Pac 201931012



## Contenido

INTRODUCCION .....	3
PLATAFORMA DE DESARROLLO .....	4
REQUISITOS DE SISTEMA .....	4
ALCANCES TECNICOS.....	4
ENTIDADES REPRESENTADAS EN EL PROGRAMA .....	5
Entidades.....	5
1. Clase Users: .....	5
2. Clase Client:.....	7
3. Clase Empleoyee:.....	8
4. Clase Pedido: .....	8
5. Clase Product: .....	11
6. Clase Store: .....	13
7. Clase TimeStoreToStore:.....	16
8. Clase Ventas: .....	17
Base de Datos.....	18
1. ConexionDB .....	19
2. ConsultasDB .....	19
3. ModificacionesDB .....	22
4. RegistroDB .....	23
Generación De Archivos.....	25
Lectura de Archivo de Carga .....	27
Interfaz Grafica .....	28
DIAGRAMAS DE LA ESTRUCTURACION DEL SOFTWARE.....	29
Diagrama de clases UML:.....	29
Diagrama Entidad Relación: .....	29
ESTRUCTURACION DE LA BASE DE DATOS .....	29

## INTRODUCCION

En el presente documento encontrara un guía a nivel técnico del funcionamiento de administración de ventas y productos de la empresa INTELAF adjuntando la lógica de programación con fines de mantenimiento a futuro o actualización en expansión del programa.

Se utilizará simplificaciones de términos con fines de entendimiento y no sobrecargar la explicación de la lógica.

## **PLATAFORMA DE DESARROLLO**

- Lenguaje Java en su versión 11 LTS ORACLE.
- Lenguaje Java en su distribución OPEN-JDK
- Sistema de gestión de proyecto Apache Maven 3.6.3.
- Máquina Virtual (build 11.0.7+8-LTS) Windows.
- Máquina Virtual () Linux Ubuntu y derivados.

## **REQUISITOS DE SISTEMA**

- Windows 10 / Linux Ubuntu 20.04 y derivados.
- 4 Gb en RAM.
- Conexión de área local si el servidor es manejado por otro equipo.
- Procesador a 2.5 GHz
- Mysql (8.0.21) Windows/Linux

## **ALCANCES TECNICOS**

El alcance planteado al programa es la solución al momento de manejar pedidos ventas y existencias de los productos vendidos por la empresa, así como la generación de reportes de los mismos para tener un control del manejo del producto que ofrece la empresa.

## ENTIDADES REPRESENTADAS EN EL PROGRAMA

El programa basado en POO por lo cual está dividido en los siguientes espacios orientados para el trabajo en conjunto alimentando lo que sea necesario durante el programa:

- Entidades
- Base de Datos
- Generación De Archivos
- Lectura de Archivo de Carga
- Interfaz Grafica

### Entidades

El grupo de entidades contiene los siguientes objetos denominados de la siguiente manera:

Nombre de Objeto	Descripción de la Objeto
Users	Clase padre de los objetos Client y Empleoyee, donde se heredan algunas de sus características
Client	Clase hija de Users en donde se abstrae la información de los clientes de la tienda
Empleoyee	Clase hija de Users en donde se abstrae información que comparte con cualquier usuario pero ciertas características únicas de un empleado
Pedido	Clase en la cual se abstrae la información necesaria de un pedido, para más adelante ser utilizada
Product	Clase en la cual se abstrae la información necesaria de un producto para más adelante se utilizada
Store	Clase en la cual se abstrae la información necesaria de un tienda para más adelante ser utilizada
TimeStoreToStore	Clase la cual se abstrae la información de la movilización de un producto de una tienda a otra
Ventas	Clase en la cual se abstrae la informa necesaria de una ventas que más adelante se registra en el programa

A continuación, se presenta los métodos que contiene cada objeto perteneciente a Entidades:

#### 1. Clase Users:

```
Users

public Users(String name,
             String phoneNumber,
             String nit,
             String dpi,
             String email,
             String direction)

Constructor principal de usuarios el cual representa características de personas en el programa almacena las características que comparten las entidades de clientes y empleados

Parameters:
name -
phoneNumber -
nit -
dpi -
email -
direction -
```

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
String	getDirection()	Retorna la direccion del usuario
String	getDpi()	Retorna el numero de DPI del usuario
String	getEmail()	Retorna el correo electronico del usuario
String	getName()	Retorna el nombre del usuario
String	getNit()	Retorna el nit del usuario
String	getPhoneNumber()	Retorna el numero del telefono del usuario
void	setDirection(String direction)	Asigna la direcccion del usuario
void	setDpi(String dpi)	Asgina el numero de DPI del usuario
void	setEmail(String email)	Asigna el correo electronico del usuario
void	setName(String name)	Asigna el nombre del usuario
void	setNit(String nit)	Asigna el nit al usuario
void	setPhoneNumber(String phoneNumber)	Asigna el numero de telefono del usuario

#### setName

```
public void setName(String name)
```

Asigna el nombre del usuario

**Parameters:**

name -

#### getName

```
public String getName()
```

Retorna el nombre del usuario

**Returns:**

#### setDirection

```
public void setDirection(String direction)
```

Asigna la direcccion del usuario

**Parameters:**

direction -

#### getDirection

```
public String getDirection()
```

Retorna la direccion del usuario

**Returns:**

#### setDpi

```
public void setDpi(String dpi)
```

Asgina el numero de DPI del usuario

**Parameters:**

dpi -

#### getDpi

```
public String getDpi()
```

Retorna el numero de DPI del usuario

**Returns:**

#### setEmail

```
public void setEmail(String email)
```

Asigna el correo electronico del usuario

**Parameters:**

email -

**getEmail**

```
public String getEmail()  
Retorna el correo electronico del usuario  
Returns:
```

**setNit**

```
public void setNit(String nit)  
Asigna el nit al usuario  
Parameters:  
nit -
```

**getNit**

```
public String getNit()  
Retorna el nit del usuario  
Returns:
```

**setPhoneNumber**

```
public void setPhoneNumber(String phoneNumber)  
Asigna el numero de telefono del usuario  
Parameters:  
phoneNumber -
```

**getPhoneNumber**

```
public String getPhoneNumber()  
Retorna el numero del telefono del usuario
```

## 2. Clase Client:

**Client**

```
public Client(String name,  
             String phoneNumber,  
             String nit,  
             String dpi,  
             float CreditoCompra,  
             String email,  
             String direction)
```

Constructor registro de cliente en tienda

**Parameters:**

name -  
phoneNumber -  
nit -  
dpi -  
CreditoCompra -  
email -  
direction -

**All Methods**   **Instance Methods**   **Concrete Methods**

Modifier and Type	Method	Description
float	getCreditoCompra()	Rerturna la cantidad de credito que tiene el cliente
void	setCreditoCompra(float creditoCompra)	Asigna una cantidad de credito a un cliente

#### getCreditoCompra

```
public float getCreditoCompra()  
Rertorna la cantidad de credito que tiene el cliente  
Returns:
```

#### setCreditoCompra

```
public void setCreditoCompra(float creditoCompra)  
Asigna un cantidad de credito a un cliente  
Parameters:  
creditoCompra -
```

### 3. Clase Empleoyee:

#### Employee

```
public Employee(String employeeCode,  
                String name,  
                String phoneNumber,  
                String nit,  
                String dpi,  
                String email,  
                String direction)
```

Constructor para empleado de la empresa

Parameters:  
employeeCode -  
name -  
phoneNumber -  
nit -  
dpi -  
email -  
direction -

#### All Methods    Instance Methods    Concrete Methods

Modifier and Type	Method	Description
float	<code>getCreditoCompra()</code>	Rertorna la cantidad de credito que tiene el cliente
void	<code>setCreditoCompra(float creditoCompra)</code>	Asigna un cantidad de credito a un cliente

#### getEmployeeCode

```
public String getEmployeeCode()  
Retorna el codigo de empleado  
Returns:
```

#### setEmployeeCode

```
public void setEmployeeCode(String employeeCode)  
Asigna el codigo de empleado  
Parameters:  
employeeCode -
```

### 4. Clase Pedido:

## Pedido

```
public Pedido(String codigo,
             String tienda1,
             String tienda2,
             String fecha,
             String cliente,
             String producto,
             int cantidad,
             float total,
             float anticipo)
```

Constructor del objeto pedido

Parameters:

codigo -  
tienda1 -  
tienda2 -  
fecha -  
cliente -  
cantidad -  
total -  
anticipo -

### All Methods    Instance Methods    Concrete Methods

Modifier and Type	Method	Description
float	<code>getAnticipo()</code>	Retorna el anticipo asignado al pedido
int	<code>getCantidad()</code>	Retorna la cantidad del producto del pedido
String	<code>getCliente()</code>	Retorna el cliente asignado al pedido
String	<code>getCodigo()</code>	Retorna el codigo asignado al pedido
String	<code>getFecha()</code>	Retorna la fecha asignada al producto
String	<code>getProducto()</code>	Retorna el codigo del producto
String	<code>getTienda1()</code>	Retorna el codigo de tienda donde saldra el producto
String	<code>getTienda2()</code>	Retorna el codigo de tienda a donde llegara el producto
float	<code>getTotal()</code>	Retorna el total asignado al producto
void	<code>setAnticipo(float anticipo)</code>	Asigna un anticipo al pedido
void	<code>setCantidad(int cantidad)</code>	Asigna una cantidad del producto al pedido
void	<code>setCliente(String cliente)</code>	Asigna el cliente de la compra
void	<code>setCodigo(String codigo)</code>	Asigna un codigo al pedido
void	<code>setFecha(String fecha)</code>	Asigna una fecha al pedido
void	<code>setFecha(String fecha)</code>	Asigna una fecha al pedido
void	<code>setProducto(String producto)</code>	Asigna el codigo del producto
void	<code>setTienda1(String tienda1)</code>	Asigna el codigo de tienda de donde saldra el producto
void	<code>setTienda2(String tienda2)</code>	Asigna el codigo de tienda a donde irea el producto
void	<code>setTotal(float total)</code>	Asigna el total del pedido

### setAnticipo

```
public void setAnticipo(float anticipo)
```

Asigna un anticipo al pedido

Parameters:  
anticipo -

### getAnticipo

```
public float getAnticipo()
```

Retorna el anticipo asignado al pedido

Returns:

**setCantidad**

```
public void setCantidad(int cantidad)
```

Asigna una cantidad del producto al pedido

**getCantidad**

```
public int getCantidad()
```

Retorna la cantidad del producto del pedido

Returns:

**setCliente**

```
public void setCliente(String cliente)
```

Asigna el cliente de la compra

Parameters:

cliente -

**getCliente**

```
public String getCliente()
```

Retorna el cliente asignado al pedido

Returns:

**setCodigo**

```
public void setCodigo(String codigo)
```

Asigna un código al pedido

Parameters:

codigo -

**getCodigo**

```
public String getCodigo()
```

Retorna el código asignado al pedido

Returns:

**setFecha**

```
public void setFecha(String fecha)
```

Asigna una fecha al pedido

Parameters:

fecha -

**getFecha**

```
public String getFecha()
```

Retorna la fecha asignada al producto

**setTienda1**

```
public void setTienda1(String tienda1)
```

Asigna el código de tienda de donde saldrá el producto

Parameters:

tienda1 -

**getTienda1**

```
public String getTienda1()
```

Retorna el código de tienda donde saldrá el producto

Returns:

**setTienda2**

```
public void setTienda2(String tienda2)  
Asigna el codigo de tienda a donde irea el producto  
Parameters:  
tienda2 -
```

**getTienda2**

```
public String getTienda2()  
Retorna el codigo de tienda a donde llegara el producto  
Returns:
```

**setTotal**

```
public void setTotal(float total)  
Asigna el total del pedido  
Parameters:  
total -
```

**getTotal**

```
public float getTotal()  
Retorna el total asignado al producto
```

**setProducto**

```
public void setProducto(String producto)  
Asigna el codigo del producto  
Parameters:  
producto -
```

**getProducto**

```
public String getProducto()  
Retrona el codigo del producto  
Returns:
```

## 5. Calse Product:

**Product**

```
public Product(String name,  
              String manufacturer,  
              String code,  
              int cantidad,  
              float price,  
              String description,  
              int garantia,  
              String tienda)
```

Constructor principal de registro de productos

**Parameters:**  
name - nombre del producto  
manufacturer - fabricante del producto  
code - codigo del producto  
cantidad - cantidad en bodega del producto  
price - precio del producto  
description - descripcion del producto  
garantia - garantia en meses del producto

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
int	<code>getCantidad()</code>	Retorna la cantidad de producto
<code>String</code>	<code>getCode()</code>	Retorna el codigo del producto
<code>String</code>	<code>getDescription()</code>	Retrona la descripcion del producto
int	<code>getGarantia()</code>	Retrona la garantia del producto
<code>String</code>	<code>getManufacturer()</code>	Retorna el fabricante del producto
<code>String</code>	<code>getName()</code>	Retorna el nombre del producto
float	<code>getPrice()</code>	Retorna el precio del producto
<code>String</code>	<code>getTienda()</code>	Retorna el codigo de tienda donde se encuentra el producto
void	<code>setCantidad(int cantidad)</code>	Asigna la cantida de producto
void	<code>setCode(String code)</code>	Asigna el codigo del producto
void	<code>setDescription(String description)</code>	Asigna la descripcion del producto
void	<code>setGarantia(int garantia)</code>	Asigna la garantia del producto
void	<code>setManufacturer(String manufacturer)</code>	Asigna el fabricamte del producto
void	<code>setName(String name)</code>	Asigna el nombre del producto
void	<code>setPrice(float price)</code>	Asigna el precio del producto
void	<code>setTienda(String tienda)</code>	Asigna el codigo de tienda donde esta el producto

#### `getCantidad`

```
public int getCantidad()
Retorna la cantidad de producto
```

#### `setCantidad`

```
public void setCantidad(int cantidad)
Asigna la cantida de producto
Parameters:
cantidad -
```

#### `getCode`

```
public String getCode()
Retorna el codigo del producto
Returns:
```

#### `setCode`

```
public void setCode(String code)
Asigna el codigo del producto
Parameters:
code -
```

#### `getDescription`

```
public String getDescription()
Retrona la descripcion del producto
Returns:
```

#### `setDescription`

```
public void setDescription(String description)
Asigna la descripcion del producto
Parameters:
description -
```

**getGarantia**

```
public int getGarantia()  
Retorna la garantia del producto  
Returns:
```

**setGarantia**

```
public void setGarantia(int garantia)  
Asigna la garantia del producto  
Parameters:  
garantia -
```

**getManufacturer**

```
public String getManufacturer()  
Retorna el fabricante del producto  
Returns:
```

**setManufacturer**

```
public void setManufacturer(String manufacturer)  
Asigna el fabricante del producto  
Parameters:  
manufacturer -
```

**getName**

```
public String getName()  
Retorna el nombre del producto
```

**setName**

```
public void setName(String name)  
Asigna el nombre del producto  
Parameters:  
name -
```

**getPrice**

```
public float getPrice()  
Retorna el precio del producto
```

**setPrice**

```
public void setPrice(float price)  
Asigna el precio del producto
```

**getTienda**

```
public String getTienda()  
Retorna el codigo de tienda donde se encuentra el producto  
Returns:
```

**setTienda**

```
public void setTienda(String tienda)  
Asigna el codigo de tienda donde esta el producto  
Parameters:  
tienda -
```

## 6. Clase Store:

## Store

```
public Store(String nombre,
            String Direccion,
            String StoreCode,
            String Phone1,
            String Phone2,
            String Email,
            String Horario)
```

Constructor principal del objeto tienda

Parameters:

nombre -

Direccion -

StoreCode -

Phone1 -

Phone2 -

Email -

Horario -

### All Methods    Instance Methods    Concrete Methods

Modifier and Type	Method	Description
String	<code>getDireccion()</code>	Retorna la direccion de la tienda
String	<code>getEmail()</code>	Retorna el email de la tienda
String	<code>getHorario()</code>	Retorna el horario en texto de la tienda
String	<code>getNombre()</code>	Retorna el nombre de la tienda
String	<code>getPhone1()</code>	Retorna el numero te telefono 1 de la tienda
String	<code>getPhone2()</code>	Retorna el numero de telefono 2 de la tienda
String	<code>getStoreCode()</code>	Retorna el codigo de tienda
void	<code>setDireccion(String direccion)</code>	Asigna la direccion de la tienda
void	<code>setEmail(String email)</code>	Asigna el email de la tienda
void	<code>setHorario(String horario)</code>	Asigna el horario de la tienda en texto
void	<code>setNombre(String nombre)</code>	Asigna el nombre de la tienda
void	<code>setPhone1(String phone1)</code>	Asigna el numero de telefono 1 de la tienda
void	<code>setPhone2(String phone2)</code>	Asigna el numero de telefono 2 de la tienda
void	<code>setStoreCode(String storeCode)</code>	Asigna el codigo de tienda

#### getNombre

```
public String getNombre()
```

Retorna el nombre de la tienda

Returns:

#### setNombre

```
public void setNombre(String nombre)
```

Asigna el nombre de la tienda

Parameters:

nombre -

#### getDireccion

```
public String getDireccion()
```

Retorna la direccion de la tienda

Returns:

**setDireccion**

```
public void setDireccion(String direccion)
```

Asigna la direcccion de la tienda

**Parameters:**

direccion -

**getEmail**

```
public String getEmail()
```

Retorna el email de la tienda

**Returns:****setEmail**

```
public void setEmail(String email)
```

Asigna el email de la tienda

**getHorario**

```
public String getHorario()
```

Retorna el horario en texto de la tienda

**Returns:****setHorario**

```
public void setHorario(String horario)
```

Asigna el horario de la tienda en texto

**Parameters:**

horario -

**getPhone1**

```
public String getPhone1()
```

Retorna el numero de telefono 1 de la tienda

**Returns:****setPhone1**

```
public void setPhone1(String phone1)
```

Asigna el numero de telefono 1 de la tienda

**Parameters:**

phone1 -

**getPhone2**

```
public String getPhone2()
```

Retorna el numero de telefono 2 de la tienda

**Returns:****setPhone2**

```
public void setPhone2(String phone2)
```

Asigna el numero de telefono 2 de la tienda

**Parameters:**

phone2 -

**getStoreCode**

```
public String getStoreCode()
```

Retorna el codigo de tienda

**Returns:**

#### **setStoreCode**

```
public void setStoreCode(String storeCode)
```

Asigna el codigo de tienda

**Parameters:**

storeCode -

## 7. Clase TimeStoreToStore:

#### **TimeStoreToStore**

```
public TimeStoreToStore(String tienda1Codigo,
                      String tienda2Codigo,
                      int tiempo)
```

Costruttore Principal de la entidad tiempo

**Parameters:**

tienda1Codigo -

tienda2Codigo -

tiempo -

#### **All Methods    Instance Methods    Concrete Methods**

Modifier and Type	Method	Description
int	<code>getId()</code>	Retorna el Id de tiempo
String	<code>getStoreCode1()</code>	Retorna el codigo de tienda1
String	<code>getStoreCode2()</code>	Retorna el codigo de tienda2
int	<code>getTime()</code>	Retorna el tiempo entre ambas tiendas
void	<code>setId(int id)</code>	Asigna el Id de tiempo
void	<code>setStoreCode1(String StoreCode1)</code>	Asigna el codigo de tienda1
void	<code>setStoreCode2(String StoreCode2)</code>	Asigna el tiempo de tienda2
void	<code>setTime(int time)</code>	Asigna el tiempo entre ambas tiendas

#### **getStoreCode1**

```
public String getStoreCode1()
```

Retorna el codigo de tienda1

**Returns:**

#### **setStoreCode1**

```
public void setStoreCode1(String StoreCode1)
```

Asigna el codigo de tienda1

**Parameters:**

StoreCode1 -

#### **getStoreCode2**

```
public String getStoreCode2()
```

Retorna el codigo de tienda2

**Returns:**

#### **setStoreCode2**

```
public void setStoreCode2(String StoreCode2)
```

Asigna el tiempo de tienda2

**Parameters:**

StoreCode2 -

**getTime**

```
public int getTime()  
Retorna el tiempo entre ambas tiendas  
Returns:
```

**setTime**

```
public void setTime(int time)  
Asigna el tiempo entre ambas tiendas  
Parameters:  
time -
```

**getId**

```
public int getId()  
Retorna el Id de tiempo  
Returns:
```

**setId**

```
public void setId(int id)  
ASigna el Id de tiempo  
Parameters:  
id -
```

## 8. Clase Ventas:

**Ventas**

```
public Ventas(String codigoProducto,  
             String codigoTienda,  
             String nitCliente,  
             int cantidad,  
             String fecha)
```

CONSTRUCTOR PRINCIPAL PARA VENTAS

Parameters:  
codigoProducto -  
codigoTienda -  
nitCliente -  
cantidad -  
fecha -

**getCantidad**

```
public int getCantidad()  
Retorna la cantidad que fue vendida del producto  
Returns:
```

**setCantidad**

```
public void setCantidad(int cantidad)  
Asigna la cantidad que fue vendida de un producto  
Parameters:  
cantidad -
```

**getCodigoProducto**

```
public String getCodigoProducto()  
Retorna el codigo del producto que fue vendido  
Returns:
```

**setCodigoProducto**

```
public void setCodigoProducto(String codigoProducto)
```

Asigna el codigo del producto que fue vendido

**Parameters:**

codigoProducto -

**getCodigoTienda**

```
public String getCodigoTienda()
```

Retorna el codigo de la tienda donde se realizo la venta

**Returns:****setCodigoTienda**

```
public void setCodigoTienda(String codigoTienda)
```

Asigna el codigo de tienda donde se realiza la venta

**Parameters:**

codigoTienda -

**getFecha**

```
public String getFecha()
```

Retorna la fecha en que se realizo una venta

**Returns:****setFecha**

```
public void setFecha(String fecha)
```

Asigna la fecha en que se realiza una venta

**Parameters:**

fecha -

**getNitCliente**

```
public String getNitCliente()
```

Retorna el nit del cliente quien realizo la compra

**Returns:****setNitCliente**

```
public void setNitCliente(String nitCliente)
```

asgina el nit del cliente quien realizo la compra

**Parameters:**

nitCliente -

## Base de Datos

La conexión a la base de datos Mysql se encuentra en el paquete DBSuport donde la clase ConexiónDB es la encargada de manejar los accesos para poder acceder a la información de la base de datos.

Nombre de Objeto	Descripción de la Objeto
ConexionDB	Clase que maneja la conexión hacia la base de datos es la única abierta durante la ejecución del todo programa.
ConsultasDB	Clase que maneja las Sentencias de consultas SQL de la cual se alimenta todo el programa.
ModificacionesDB	Clase que maneja la modificación de la información ya introducida en la base de datos.
RegistroDB	Clase encargada del registro de las nuevas entidades en la base de datos.

Cabe mencionar que ConsultasDB, ModificacionesDB y RegistroDB, en todos sus métodos poseen similitudes para la entrega de datos, por lo cual se describirá el funcionamiento de una sola, ya que la lógica es la misma en todas, solamente se aclarara la consulta que se hace en el programa.

## 1. ConexionDB

### ConexionDB

```
public ConexionDB()
    throws ClassNotFoundException,
        SQLException
```

Constructor principal de la clase conexion Genera la conexion de la base de datos al principio

**Throws:**  
ClassNotFoundException  
SQLException

### getConexion

```
public Connection getConexion()
```

Retorna la conexion de la base de datos

**Returns:**

### cerrarConexion

```
public void cerrarConexion()
```

Cierra la conexion con la base de datos del programa

**Throws:**  
SQLException

## 2. ConsultasDB

Posee un constructor vacío.

All Methods	Instance Methods	Concrete Methods	
Modifier and Type	Method	Description	
	<code>ArrayList&lt;String&gt; codigosDeTiendas(</code> <code>Connection conexion)</code>	Retorna los codigos de las tiendas que estan en la base de datos	
<code>String</code>	<code>codigoTienda(String nombre, </code> <code>Connection conexion)</code>	Retrona el codigo de la tienda segun el nombre de la misma	
	<code>ArrayList&lt;String[]&gt; comprasClienteReporte(String nitCliente, </code> <code>Connection conexion)</code>	Da informacion de las compras de un cliente en especifico	
	<code>ArrayList&lt;String&gt; consultaDeTiendas(</code> <code>Connection conexion)</code>	Retrona el nombre de las tiendas que estan en el sistema de base de datos	
<code>String</code>	<code>consultarCodigoTienda(String nombre, </code> <code>Connection conexion)</code>	Retorna el codigo de una tienda segun su nombre	
<code>boolean</code>	<code>consultarExistencia(String codigoTienda, String codigoProducto, </code> <code>Connection conexion)</code>	Verifica si ya existe un registro con una configuracion dada	
<code>int</code>	<code>consultaUsuarios(String nombre, String codeNIT, </code> <code>Connection conexion)</code>	Retrona si los datos ingresados pertenecen a un empleado o a un cliente en el sistema	
<code>int</code>	<code>contarPedidos(String codigoPedido, </code> <code>Connection conexion)</code>	Cuenta el numero de pedidos asignado a un codigo	
	<code>ArrayList&lt;String&gt; datosCliente(String nit, </code> <code>Connection conexion)</code>	Retorna todos los datos de un cliente segun si nit de ingreso	
	<code>ArrayList&lt;String&gt; datosEmpleado(String codigo, </code> <code>Connection conexion)</code>	Retorna los datos de un empleado	
<code>String</code>	<code>datosExistenciaProducto(String codigo, String codigoTienda, </code> <code>Connection conexion)</code>	Retorna el valor de existencia del producto en String	
	<code>ArrayList&lt;String&gt; datosPedido(String codigo, </code> <code>Connection conexion)</code>	Esta clase funciona solemnete para el apartado de acceso al usuario Retorna datos de un pedido segun el codigo generado	
	<code>ArrayList&lt;String&gt; datosProducto(String codigo, </code> <code>Connection conexion)</code>	Retornar los datos de producto segun codigo el codigo del mismo	
	<code>ArrayList&lt;String&gt; datosTienda(String codigo, </code> <code>Connection conexion)</code>	Retrona los datos de la tienda segun el codigo de la tienda	
	<code>ArrayList&lt;String[]&gt; diezProductosMasVendidos(</code> <code>Connection conexion)</code>	Retorna los 10 productos mas vendidos contando todas las tiendas	
	<code>ArrayList&lt;String[]&gt; diezProductosMasVendidosIntervalo(String fechaInferior, String fechaSuperior, </code> <code>Connection conexion)</code>	Retorna los 10 productos mas vendidos contando todas las tiendas en un intervalo de tiempo	
	<code>ArrayList&lt;String&gt; estadoPedido(String codigoPedido, </code> <code>Connection conexion)</code>	Retorna el estado de un pedido	
	<code>ArrayList&lt;String[]&gt; pedidosClienteReporte(String nitCliente, </code> <code>Connection conexion)</code>	Da informacion de los pedidos de un cliente en especifico	
	<code>ArrayList&lt;String[]&gt; pedidosReporte(String codigoTienda, String estadoDePedido, </code> <code>Connection conexion)</code>	Da la informacion necesaria para generar un reporte	
	<code>ArrayList&lt;String[]&gt; pedidosSalidaReporte(String codigoTienda, </code> <code>Connection conexion)</code>	Da la informacion de los pedidos que salieron de la tienda	
	<code>ArrayList&lt;String&gt; productoDeUnPedido(String codigo, </code> <code>Connection conexion)</code>	Retorna el producto de un pedido	
	<code>ArrayList&lt;String[]&gt; ProductosMasVendidosTienda(String codigoTienda, </code> <code>Connection conexion)</code>	Retorna los productos mas vendidos por la tienda en que se realizo el reporte	
	<code>ArrayList&lt;String[]&gt; ProductosMasVendidosTiendaIntervalo(String codigoTienda, String fechaInferior, String fechaSuperior, </code> <code>Connection conexion)</code>	Retorna los productos mas vendidos en una tienda especifica y un rango de tiempo	
	<code>ArrayList&lt;String[]&gt; productosNoVendidosPorTienda(String codigoTienda, </code> <code>Connection conexion)</code>	Retorna los productos que nunca se han vendido segun la tienda que se seleccione	
	<code>ArrayList&lt;Pedido&gt; retornoDePedidos(String codigoPedido, </code> <code>Connection conexion)</code>	Retorno de los pedidos segun el codigo enlazado al mismo	
<code>float</code>	<code>sumaAnticipoPedido(String codigo, </code> <code>Connection conexion)</code>	Suma total del anticipo de un pedido	
<code>float</code>	<code>sumaTotalPedido(String codigo, </code> <code>Connection conexion)</code>	Retorna la suma total del pedido	
	<code>ArrayList&lt;String&gt; tiempoEntreTiendas(String codigoTienda1, String codigoTienda2, </code> <code>Connection conexion)</code>	Retorna el tiempo entre las tiendas segun el codigo de tienda ingresados	

Método	Consulta de SQL
<code>codigosDeTiendas</code>	<code>SELECT codigo FROM TIENDA</code>
<code>codigoTienda</code>	<code>SELECT codigo FROM TIENDA WHERE nombre = ?</code>
<code>comprasClienteReporte</code>	<code>SELECT TIENDA_codigo,PRODUCTO_codigo,fecha_venta,cantidad_producto FROM VENTAS WHERE CLIENTE_nit = ?</code>
<code>consultaDeTiendas</code>	<code>SELECT nombre FROM TIENDA</code>
<code>consultarCodigoTienda</code>	<code>SELECT codigo FROM TIENDA WHERE nombre = ?</code>
<code>consultarExistencia</code>	<code>SELECT id FROM EXISTENCIA WHERE TIENDA_codigo = ? AND PRODUCTO_codigo = ?</code>
<code>consultaUsuarios</code>	<code>SELECT nombre,nit FROM CLIENTE WHERE nit = ? AND nombre = ?</code>
<code>contarPedidos</code>	<code>SELECT COUNT(*) FROM PEDIDO WHERE codigo = ?</code>
<code>datosCliente</code>	<code>SELECT nombre,telefono,credito,dpi,email,direccion FROM CLIENTE WHERE nit = ?</code>

datosEmpleado	SELECT nombre,telefono,dpi,nit,email,direccion FROM EMPLEADO WHERE codigo = ?
datosExistenciaProducto	SELECT cantidad FROM EXISTENCIA WHERE PRODUCTO_codigo = ? AND TIENDA_codigo = ?
datosPedido	SELECT anticipo, TIENDA_codigo_salida, TIENDA_codigo_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1
datosProducto	SELECT anticipo, TIENDA_codigo_salida, TIENDA_codigo_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1
datosTienda	SELECT nombre, direccion, telefono_1, telefono_2, email, horario FROM TIENDA WHERE codigo = ?
diezProductosMasVendidos	SELECT PRODUCTO.codigo, PRODUCTO.nombre, PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO_codigo FROM VENTAS GROUP BY PRODUCTO_codigo ORDER BY count(PRODUCTO_codigo) DESC) LIMIT 10
diezProductosMasVendidosIntervalo	SELECT PRODUCTO.codigo, PRODUCTO.nombre, PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO_codigo FROM VENTAS WHERE fecha_venta BETWEEN ? AND ? GROUP BY PRODUCTO_codigo ORDER BY count(PRODUCTO_codigo) DESC) LIMIT 10
estadoPedido	SELECT cantidad, total, anticipo, fecha_orden, estado_pedido, CLIENTE_nit, PRODUCTO_codigo, TIENDA_codigo_salida, TIENDA_codigo_llegada FROM PEDIDO WHERE codigo = ? LIMIT 1
pedidosClienteReporte	SELECT codigo, PRODUCTO_codigo, cantidad, TIENDA_codigo_salida, TIENDA_codigo_llegada FROM PEDIDO WHERE CLIENTE_nit = ? AND estado_pedido != 'entregado'
pedidosReporte	SELECT PEDIDO.codigo, CLIENTE.nombre, PEDIDO.CLIENTE_nit, PEDIDO.TIENDA_codigo_salida FROM PEDIDO, CLIENTE WHERE PEDIDO.TIENDA_codigo_llegada = ? AND PEDIDO.CLIENTE_nit = CLIENTE.nit AND PEDIDO.estado_pedido = ?
pedidosSalidaReporte	SELECT PEDIDO.codigo, CLIENTE.nombre, PEDIDO.CLIENTE_nit, PEDIDO.TIENDA_codigo_llegada FROM PEDIDO, CLIENTE WHERE PEDIDO.TIENDA_codigo_salida = ? AND PEDIDO.CLIENTE_nit = CLIENTE.nit AND PEDIDO.estado_pedido = 'ET'
productoDeUnPedido	SELECT PRODUCTO_codigo FROM PEDIDO WHERE codigo = ?
ProductosMasVendidosTienda	SELECT PRODUCTO.codigo, PRODUCTO.nombre, PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO_codigo FROM VENTAS WHERE VENTAS.TIENDA_codigo = ? GROUP BY PRODUCTO_codigo ORDER BY count(PRODUCTO_codigo) DESC)
ProductosMasVendidosTiendaIntervalo	SELECT PRODUCTO.codigo, PRODUCTO.nombre, PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO_codigo FROM VENTAS WHERE VENTAS.TIENDA_codigo = ? AND fecha_venta BETWEEN ? AND ? GROUP BY PRODUCTO_codigo ORDER BY count(PRODUCTO_codigo) DESC)
productosNoVendidosPorTienda	SELECT PRODUCTO.codigo, PRODUCTO.nombre, PRODUCTO.fabricante FROM PRODUCTO WHERE codigo IN (SELECT PRODUCTO_codigo FROM EXISTENCIA WHERE ((SELECT COUNT(*) FROM VENTAS WHERE VENTAS.TIENDA_codigo = ? AND VENTAS.PRODUCTO_codigo = EXISTENCIA.PRODUCTO_codigo) = 0) AND EXISTENCIA.TIENDA_codigo = ?)
retornoDePedidos	SELECT cantidad, total, anticipo, fecha_orden, estado_pedido, CLIENTE_nit, PRODUCTO_codigo, TIENDA_codigo_salida, TIENDA_codigo_llegada FROM PEDIDO WHERE codigo = ?
sumaAnticipoPedido	SELECT SUM(anticipo) FROM PEDIDO WHERE codigo = ?
sumaTotalPedido	SELECT SUM(total) FROM PEDIDO WHERE codigo = ?
tiempoEntreTiendas	SELECT tiempo, codigo FROM TIEMPO_TRASLADO WHERE TIENDA_codigo1 = ? AND TIENDA_codigo2 = ?

### 3. ModificacionesDB

#### Constructor de la clase vacío

All Methods	Instance Methods	Concrete Methods	
Modifier and Type	Method		Description
String	<code>modificacionEstadoPedido(String codigoPedido, String estadoPedido, Connection conexion)</code>		Cambia el estado del pedido es entrega "ET" en trafico, "T" en tienda, "R" retraso, "E" entegado
String	<code>modificarCliente(Client cliente, Connection conexion)</code>		Modifica los datos existentes de un cliente
String	<code>modificarCreditoCliente(String nit, String creditoNuevo, Connection conexion)</code>		Modificacion del credito del cliente segun el nit
String	<code>modificarEmpleado(Employee empleado, Connection conexion)</code>		Modifica los datos de un empleado
String	<code>modificarExistencia(Product producto, Connection conexion)</code>		Modifica la existencia de un producto
String	<code>modificarPedido(Pedido pedido, Connection conexion)</code>		Modificacion de datos de un pedido
String	<code>modificarProducto(Product producto, Connection conexion)</code>		Modifica los datos de producto
String	<code>modificarTiempo(TimeStoreToStore tiempoDeTransporte, Connection conexion)</code>		Modificacion del tiempo entre tiendas
String	<code>modificarTienda(Store tienda, Connection conexion)</code>		Modifica los datos de una tienda
<b>modificarCliente</b>			
public String modificarCliente(Client cliente, Connection conexion)			
Modifica los datos existentes de un cliente			
<b>Parameters:</b>			
cliente -			
conexion -			
<b>Returns:</b>			
<b>modificarTienda</b>			
public String modificarTienda(Store tienda, Connection conexion)			
Modifica los datos de una tienda			
<b>Parameters:</b>			
tienda -			
<b>Returns:</b>			
<b>modificarEmpleado</b>			
public String modificarEmpleado(Employee empleado, Connection conexion)			
Modifica los datos de un empleado			
<b>Parameters:</b>			
empleado -			
<b>Returns:</b>			
<b>modificarProducto</b>			
public String modificarProducto(Product producto, Connection conexion)			
Modifica los datos de producto			
<b>Parameters:</b>			
producto -			
<b>Returns:</b>			
<b>modificarExistencia</b>			
public String modificarExistencia(Product producto, Connection conexion)			
Modifica la existencia de un producto			
<b>Parameters:</b>			
producto -			
<b>Returns:</b>			

**modificarTiempo**

```
public String modificarTiempo(TimeStoreToStore tiempoDeTransporte, Connection conexion)
```

Modificacion del tiempo entre tiendas

Parameters:

tiempoDeTransporte -

Returns:

**modificarPedido**

```
public String modificarPedido(Pedido pedido, Connection conexion)
```

Modificacion de datos de un pedido

Parameters:

pedido -

Returns:

**modificarCreditoCliente**

```
public String modificarCreditoCliente(String nit, String creditoNuevo, Connection conexion)
```

Modificacion del credito del cliente segun el nit

Parameters:

nit -

creditoNuevo -

Returns:

**modificacionEstadoPedido**

```
public String modificacionEstadoPedido(String codigoPedido, String estadoPedido, Connection conexion)
```

Cambia el estado del pedido es entrega "ET" en trafico, "T" en tienda, "R" retraso, "E" entegado

Parameters:

codigoPedido -

estadoPedido -

Returns:

Método	Sentencia de SQL
modificacionEstadoPedido	UPDATE PEDIDO SET estado_pedido = ? WHERE codigo = ?
modificarCliente	UPDATE CLIENTE SET nombre = ?,telefono = ?,credito = ?,dpi = ?,email = ?,direccion = ? WHERE nit = ?
modificarCreditoCliente	UPDATE CLIENTE SET credito = ? where nit= ?
modificarEmpleado	UPDATE EMPLEADO SET nombre = ?,telefono = ?,dpi = ?,nit = ?,email = ?,direccion = ? WHERE codigo = ?
modificarExistencia	UPDATE EXISTENCIA SET cantidad = ? WHERE TIENDA_codigo = ? AND PRODUCTO_codigo = ?
ModificarPedido	UPDATE PEDIDO SET cantidad = ?,total = ?,anticipo = ?,fecha_orden = ?,CLIENTE_nit = ?,PRODUCTO_codigo = ?,TIENDA_codigo_salida = ?,TIENDA_codigo_llegada = ? WHERE codigo = ?
modificarProducto	UPDATE PRODUCTO SET nombre = ?,fabricante = ?,precio = ?,descripcion = ?,garantia = ? WHERE codigo = ?
modificarTiempo	UPDATE TIEMPO_TRASLADO SET tiempo = ? WHERE TIENDA_codigo1 = ? AND TIENDA_codigo2 = ?
modificarTienda	UPDATE TIENDA SET nombre = ?,direccion = ?,telefono_1 = ?,telefono_2 = ?,email = ?,horario = ? WHERE codigo = ?

#### 4. RegistroDB

La clase posee un constructor vacío.

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
String	registroCliente(Client cliente, Connection conexion)	Metodo en el cual se registran los clientes en la base de datos
String	registroEmpleado(Employee empleado, Connection conexion)	Metodo en el cual se registran los datos de los empleados en la base de datos
String	registroExistencia(Product producto, Connection conexion)	Metodo en el cual se registran la existencia de un producto determinado
String	registroPedido(Pedido pedido, Connection conexion)	Metodo en el cual se registran los pedidos en base de datos
String	registroProducto(Product producto, Connection conexion)	Metodo en el cual se registran los productos en la base de datos
String	registroTiempo(TimeStoreToStore tiempoDeTransporte, Connection conexion)	Se registra el tiempo entre las tiendas
String	registroTienda(Store tienda, Connection conexion)	Metodo en el cual se registran las tiendas en una base de datos
String	registroVenta(Ventas venta, Connection conexion)	Metodo en el cual se registran las ventas en la base de datos

#### registroCliente

```
public String registroCliente(Client cliente, Connection conexion)
```

Metodo en el cual se registran los clientes en la base de datos

**Parameters:**

cliente -

conexion -

**Returns:**

#### registroTienda

```
public String registroTienda(Store tienda, Connection conexion)
```

Metodo en el cual se registran las tiendas en una base de datos

**Parameters:**

tienda -

conexion -

**Returns:**

#### registroEmpleado

```
public String registroEmpleado(Employee empleado, Connection conexion)
```

Metodo en el cual se registran los datos de los empleados en la base de datos

**Parameters:**

empleado -

conexion -

**Returns:**

#### registroProducto

```
public String registroProducto(Product producto, Connection conexion)
```

Metodo en el cual se registran los productos en la base de datos

**Parameters:**

producto -

conexion -

**Returns:**

#### registroExistencia

```
public String registroExistencia(Product producto, Connection conexion)
```

Metodo en el cual se registran la existencia de un producto determinado

**Parameters:**

producto -

conexion -

**Returns:**

**registroTiempo**

```
public String registroTiempo(TimeStoreToStore tiempoDeTransporte, Connection conexion)
```

Se registra el tiempo entre las tiendas

**Parameters:**

tiempoDeTransporte -

conexion -

**Returns:**

**registroPedido**

```
public String registroPedido(Pedido pedido, Connection conexion)
```

Metodo en el cual se registran los pedidos en base de datos

**Parameters:**

pedido -

conexion -

**Returns:**

**registroVenta**

```
public String registroVenta(Ventas venta, Connection conexion)
```

Metodo en el cual se registran las ventas en la base de datos

**Parameters:**

venta -

conexion -

**Returns:**

Método	Sentencia de SQL
registroCliente	INSERT INTO CLIENTE VALUES (?,?,?,?,?,?)
registroTienda	INSERT INTO TIENDA VALUES (?,?,?,?,?,?)
registroEmpleado	INSERT INTO EMPLEADO VALUES (?,?,?,?,?,?)
registroProducto	INSERT INTO PRODUCTO VALUES (?,?,?,?,?,?)
registroExistencia	INSERT INTO EXISTENCIA (TIENDA_codigo,PRODUCTO_codigo,cantidad) VALUES (?,?,?)
registroTiempo	INSERT INTO TIEMPO_TRASLADO (TIENDA_codigo1, TIENDA_codigo2, tiempo) VALUES (?,?,?)
registroPedido	INSERT INTO PEDIDO (codigo,cantidad,total,anticipo,fecha_orden,estado_pedido,CLIENTE_nit,PRODUCTO_codigo, TIENDA_codigo_salida, TIENDA_codigo_llegada) VALUES (?,?,?,?,?,?,?,?,?,?)
registroVenta	INSERT INTO VENTAS (TIENDA_codigo,PRODUCTO_codigo,CLIENTE_nit,fecha_venta,cantidad_producto) VALUES (?,?,?,?,?)

## Generación De Archivos

La generación de archivos es utilizada al momento de generar los reportes del programa, en este caso generada través de un archivo con formato HTML para poder visualizar la información que está dentro del programa, mediante la Clase GenerateHTML la cual escribe el formato del archivo con los métodos que contiene.

GenerateHTML es una clase que se apoya por medio de las consultas descritas anteriormente desde el programa a la base de datos.

#### GenerateHTML

```
public GenerateHTML(File archivo,
                    Connection conexionBaseDatos)
```

All Methods   Instance Methods   Concrete Methods

Modifier and Type	Method	Description
void	Generate(String tipoReporte)	Genera el archivo HTML segun el tipo de reporte que se selecciono
String	getCodigoTienda()	Retorna el codigo de la tienda en la cual se realiza el reporte
String	getNITcliente()	Retorna el nit del cliente en el cual se realizara el reporte
String	getTiempoSuperior()	Retorna la fecha limite superior en la cual se evalua un reporte
String	getTimepoInferior()	Retorna la fecha inferior en la cual se realiza un reporte
void	setCodigoTienda(String codigoTienda)	Asigna el codigo de la tienda en la cual se realiza el reporte
void	setNITcliente(String NITcliente)	Asigna el nit del cliente en el cual se realizara un reporte
void	setTiempoSuperior(String tiempoSuperior)	Asigna la fecha limite superior en la cual se evalua un reporte
void	setTimepoInferior(String timepoInferior)	Asigna la fecha inferior en la cual se evalua un reporte

#### Generate

```
public void Generate(String tipoReporte)
```

Genera el archivo HTML segun el tipo de reporte que se selecciono

Parameters:  
tipoReporte -

#### getCodigoTienda

```
public String getCodigoTienda()
```

Retorna el codigo de la tienda en la cual se realiza el reporte

Returns:

#### setCodigoTienda

```
public void setCodigoTienda(String codigoTienda)
```

Asigna el codigo de la tienda en la cual se realiza el reporte

Parameters:  
codigoTienda -

#### getNITcliente

```
public String getNITcliente()
```

Retorna el nit del cliente en el cual se realizara el reporte

Returns:

#### setNITcliente

```
public void setNITcliente(String NITcliente)
```

Asigna el nit del cliente en el cual se realizara un reporte

Parameters:  
NITcliente -

#### getTiempoSuperior

```
public String getTiempoSuperior()
```

Retorna la fecha limite superior en la cual se evalua un reporte

Returns:

#### setTiempoSuperior

```
public void setTiempoSuperior(String tiempoSuperior)
```

Asigna la fecha limite superior en la cual se evalua un reporte

Parameters:  
tiempoSuperior -

```
getTimepoInferior
```

```
public String getTimepoInferior()
Rertona la fecha inferior en la cual se realiza un reporte
```

Returns:

```
setTimepoInferior
```

```
public void setTimepoInferior(String timepoInferior)
```

Asigna la fecha inferior en la cual se evalua un reporte

Parameters:

timepoInferior -

El conjunto anterior de métodos solo representa los atributos de la clase, los encargados de realizar el armado del archivo HTML son los métodos del tipo PrintWriter.

Nombre del método	Tipo de Reporte encargado a realizar
htmlContentPedidosTienda	Realiza el archivo HTML en el cual reporta que pedidos llegaran a la tienda.
htmlContentPedidosAtrazadosTienda	Realiza el archivo HTML en el cual reporta que pedidos llegaron con atraso a la tienda.
htmlContentPedidosExpendidosPorTienda	Realiza el archivo HTML en el cual reporta que pedidos salieron de la tienda.
htmlPedidosPorCliente	Realiza el archivo HTML en el cual reporta que pedidos fueron hechos por un cliente.
htmlDiezProductosMasVendidos	Realiza el archivo HTML en el cual reporta cuales son los 10 productos mas vendidos.
htmlDiezProductosMasVendidosIntervalo	Realiza el archivo HTML en el cual reporta cuales son los 10 productos mas vendidos en un intervalo de tiempo.
htmlProductosMasVendidosTienda	Realiza el archivo HTML en el cual reporta cuales son los producto mas vendidos por una tienda.
htmlProductosMasVendidosTiendaIntervalo	Realiza el archivo HTML en el cual reporta cuales son los productos mas vendidos en un tienda en un intervalo de tiempo.
htmlProductosNoVendidosPorTienda	Realiza el archivo HTML en el cual reporta cuales son los productos que no se han vendido.

## Lectura de Archivo de Carga

Este apartado es el encargado de realizar la lectura de un archivo de información de la tienda e introducirlo en la base de datos.

La clase encargada de esto es **InputText**.

```
InputText
```

```
public InputText(Connection conexionBaseDatos)
```

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
ArrayList<String>	getErroresDeDatos()	Regresa la cadena con los errores correspondientes de la base de datos
ArrayList<String>	getLineasDeTexto()	Devuelve el array con los datos del procesado
void	LecturaEIngreso(File archivoSeleccionado)	Ingreso de los datos existentes en una base de datos

```
LecturaEIngreso
```

```
public void LecturaEIngreso(File archivoSeleccionado)
```

Ingreso de los datos existentes en una base de datos

**getLineasDeTexto**

```
public ArrayList<String> getLineasDeTexto()
```

Devuelve el array con los datos del procesado

Returns:

**getErroresDeDatos**

```
public ArrayList<String> getErroresDeDatos()
```

Regresa la cadena con los errores correspondientes de la base de datos

Returns:

La descomposición lógica del archivo para la carga de información está dada en reconocer principalmente de una palabra clave para el inicio de la información, el resto de operación se realizan de la siguiente manera:

1. Identificar la cantidad de separadores de información.
2. Cada palabra clave tiene su respectivo número de datos.
3. Realizar Filtros de información, con los cuales pueda identificar texto, números, teléfonos en las posiciones requeridas.
4. Utilizar la entidad relacionada con la información procesada para ingresarla en la base de datos
5. Manejar los errores al momento de ingresar los datos en base de datos.

## Interfaz Grafica

Información de los elementos pertenecientes a la interfaz gráfica del sistema.

<b>Constructor</b>	<b>Description</b>
<code>CaracteristicaReportesJDialog(Frame parent, boolean modal, String tienda, Connection conexionBaseDatos)</code>	Configura los reportes y adjunta la infomacion que ingresa el usuario para luego exportarlos por medio de HTML
<b>Constructor</b>	<b>Description</b>
<code>CatalogoJDialog(Frame parent, boolean modal, Connection conexionBaseDatos)</code>	Genera una ventana en la cual por medio de una tablas y por metodo de ordenamiento de base de datos podemos visualizar todo el catalogo de productos
<b>Constructor</b>	<b>Description</b>
<code>EntregaProductoJDialog(Frame parent, boolean modal, String tienda, Connection conexionBaseDatos)</code>	Formulario en el cual se ingresan datos de pedidos para su entrega posteriormente realizar su transaccion venta y registro en el sistema
<b>Constructor</b>	<b>Description</b>
<code>LogInFrame(PrincipalFrame principal, Connection conexionBaseDeDatos)</code>	Formulario de acceso al software en el cual se hace el filtro de empleados y clientes
<b>Constructor</b>	<b>Description</b>
<code>RastreoPedidoJDialog(Frame parent, boolean modal, Connection conexionBaseDatos)</code>	Formulario apartado para el rastreo e informacion de un pedido mediante codigo de pedido del mismo
<b>Constructor</b>	<b>Description</b>
<code>RealizarPedidoJDialog(Frame parent, boolean modal, String nombreTienda, Connection conexionBaseDatos)</code>	Formulario en el cual se hace un pedido introduciendo los datos de lo que se quiere solicitar
<b>Constructor</b>	<b>Description</b>
<code>RegistrarProductoJDialog(Frame parent, boolean modal)</code>	Formulario mediante se hace el registro y modificacion de los productos y una lista de los productos que estan registrados en la base de datos
<b>Constructor</b>	<b>Description</b>
<code>RegistroCliente(PrincipalFrame principal, Connection conexionBaseDeDatos)</code>	Formulario para el registro para nuevos clientes desde la pestaña principal
<b>Constructor</b>	<b>Description</b>
<code>RegistroClienteJDialog(Frame parent, boolean modal, Connection conexionBaseDatos)</code>	Formulario para registro y modificacion de infromacion de clientes, asi una lista de todos los clientes registrados en el sistema
<b>Constructor</b>	<b>Description</b>
<code>RegistroEmpleadoJDialog(Frame parent, boolean modal, Connection conexionBaseDatos)</code>	Formulario para registro y modificacion de los empleados, y lista de los empleados ya encontrados en la base de datos del sistema

Constructor	Description
RegistroTiendaJDialog(Frame parent, boolean modal, String tienda, Connection conexionBaseDatos)	Registro y modificacion de las tiendas y lista de las tienda ya registradas en la base de datos
Constructor	Description
TiendaClienteJFrame(String tienda, Connection conexionBaseDatos)	Espacio de acceso al usuario en el cual puede visualizar solo, rastreo de pedidos y el catalogo de todos los productos
Constructor	Description
TiendaEmpleadoJFrame(String tienda, Connection conexionBaseDatos)	Espacio de software en el cual solo pueden acceder los empleados y tienen a sus dispocion el registro y modificacion de infromacion y exptacion de reportes y lo referido a pedidos
Constructor	Description
VentaJDialog(Frame parent, boolean modal, Pedido pedidoProcesar)	
VentaJDialog(Frame parent, boolean modal, String tiendaSleccionada, Connection conexionBaseDatos)	Formulario mediante el cual se hacen las ventas de productos del ctaalogo de la empresa
Constructor	Description
VisualizadorDeErroresJDialog(Frame parent, boolean modal, ArrayList<String> erroresDeDatos)	Formulario en el cual se muestran los errores de cargar del archivo de informacion para la base de datos del producto

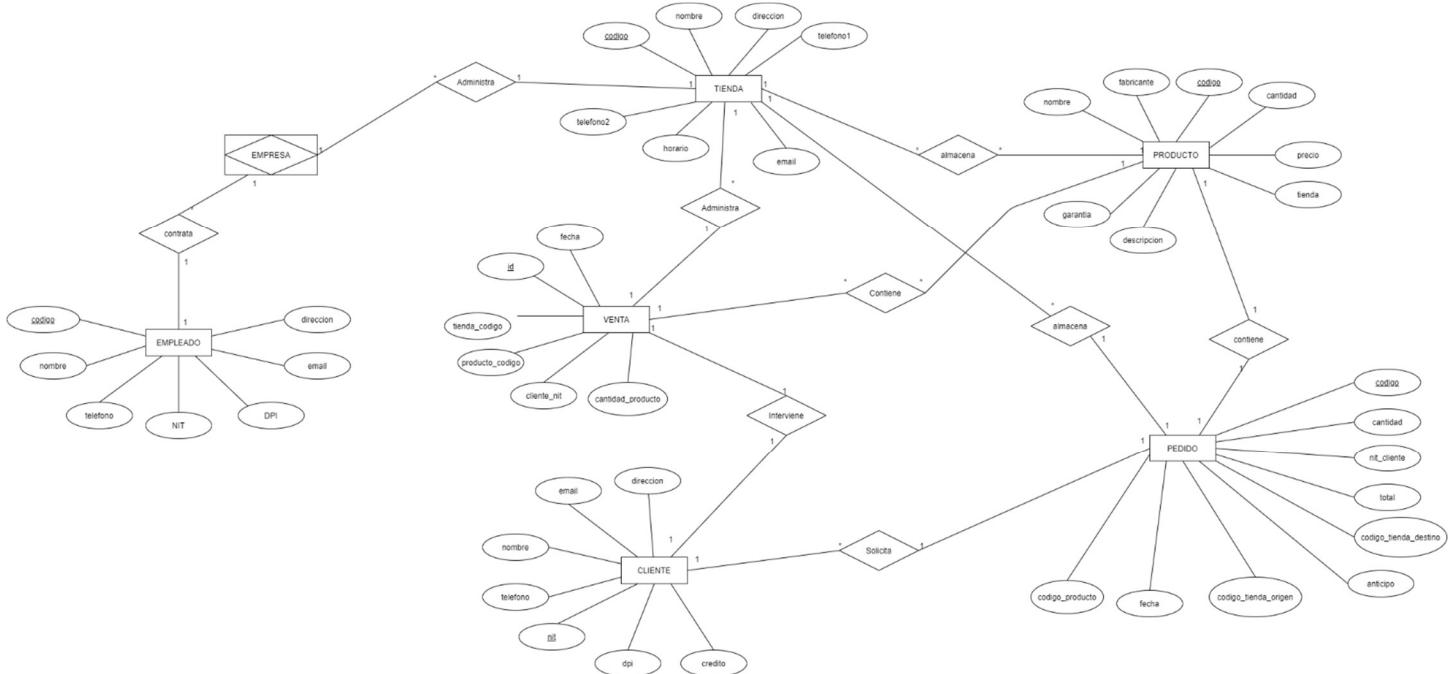
## DIAGRAMAS DE LA ESTRUCTURACION DEL SOFTWARE

### Diagrama de clases UML:

Debido a su gran tamaño en diagrama es detallado en un archivo diferente con formato HTML para poder visualizar de mejor manera la relacione que existen entre las clases del software realizado.

### Diagrama Entidad Relación:

Aparte de este medio se proporciona un Diagrama de Entidad Relación en formato PNG



## ESTRUCTURACION DE LA BASE DE DATOS

Estructuración de la base de datos Mysql donde se presentan las tablas donde se almacenan los datos:

CREATE SCHEMA IF NOT EXISTS mi\_empresa\_proyecto1;

USE mi\_empresa\_proyecto1;

```
CREATE TABLE IF NOT EXISTS TIENDA(
```

```
    codigo VARCHAR(15) NOT NULL,  
    nombre VARCHAR(20) NOT NULL,  
    direccion VARCHAR(100) NOT NULL,  
    telefono_1 VARCHAR(8) NOT NULL,  
    telefono_2 VARCHAR(8),  
    email VARCHAR(20),  
    horario VARCHAR(100),  
    PRIMARY KEY (codigo),  
    UNIQUE (codigo,nombre)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS CLIENTE (
```

```
    nit VARCHAR(10) NOT NULL,  
    nombre VARCHAR(20) NOT NULL,  
    telefono VARCHAR(8) NOT NULL,  
    credito DOUBLE,  
    dpi VARCHAR(10),  
    email VARCHAR(30),  
    direccion VARCHAR(100),  
    PRIMARY KEY (nit),  
    UNIQUE (nit,dpi)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS EMPLEADO (
```

```
    codigo VARCHAR(20) NOT NULL,  
    nombre VARCHAR(20) NOT NULL,  
    telefono VARCHAR(8) NOT NULL,  
    dpi VARCHAR(10) NOT NULL,  
    nit VARCHAR(20),  
    email VARCHAR(30) NOT NULL,  
    direccion VARCHAR(100) NOT NULL,  
    PRIMARY KEY (codigo),  
    UNIQUE (codigo,dpi,nit)
```

```
);
```

```
CREATE TABLE IF NOT EXISTS PRODUCTO(
    codigo VARCHAR(15) NOT NULL,
    nombre VARCHAR(20) NOT NULL,
    fabricante VARCHAR(20) NOT NULL,
    precio DOUBLE NOT NULL,
    descripcion VARCHAR(100),
    garantia INT,
    PRIMARY KEY (codigo),
    UNIQUE (codigo)
);

CREATE TABLE IF NOT EXISTS EXISTENCIA(
    id INT NOT NULL AUTO_INCREMENT,
    TIENDA_codigo VARCHAR(15) NOT NULL,
    PRODUCTO_codigo VARCHAR(15) NOT NULL,
    cantidad INT NOT NULL,
    PRIMARY KEY (id),
    UNIQUE (id),
    FOREIGN KEY (TIENDA_codigo) REFERENCES TIENDA(codigo),
    FOREIGN KEY (PRODUCTO_codigo) REFERENCES PRODUCTO(codigo)
);

CREATE TABLE IF NOT EXISTS PEDIDO(
    id INT NOT NULL AUTO_INCREMENT,
    codigo VARCHAR(15) NOT NULL,
    cantidad INT NOT NULL,
    total DOUBLE NOT NULL,
    anticipo DOUBLE NOT NULL,
    fecha_orden DATE NOT NULL,
    estado_pedido VARCHAR(10) NOT NULL,
    CLIENTE_nit VARCHAR(10) NOT NULL,
    PRODUCTO_codigo VARCHAR(15) NOT NULL,
    TIENDA_codigo_salida VARCHAR(15) NOT NULL,
    TIENDA_codigo_llegada VARCHAR(15) NOT NULL,
    PRIMARY KEY (id),
```

```
FOREIGN KEY (CLIENTE_nit) REFERENCES CLIENTE(nit),
FOREIGN KEY (PRODUCTO_codigo) REFERENCES PRODUCTO(codigo),
FOREIGN KEY (TIENDA_codigo_salida) REFERENCES TIENDA(codigo),
FOREIGN KEY (TIENDA_codigo_llegada) REFERENCES TIENDA(codigo)
);
```

```
CREATE TABLE IF NOT EXISTS TIEMPO_TRASLADO(
    id INT NOT NULL AUTO_INCREMENT,
    TIENDA_codigo1 VARCHAR(15) NOT NULL,
    TIENDA_codigo2 VARCHAR(15) NOT NULL,
    tiempo INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (TIENDA_codigo1) REFERENCES TIENDA(codigo),
    FOREIGN KEY (TIENDA_codigo2) REFERENCES TIENDA(codigo)
);
```

```
CREATE TABLE IF NOT EXISTS VENTAS(
    id INT NOT NULL AUTO_INCREMENT,
    TIENDA_codigo VARCHAR(15) NOT NULL,
    PRODUCTO_codigo VARCHAR(15) NOT NULL,
    CLIENTE_nit VARCHAR(10) NOT NULL,
    fecha_venta DATE NOT NULL,
    cantidad_producto INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (CLIENTE_nit) REFERENCES CLIENTE(nit),
    FOREIGN KEY (TIENDA_codigo) REFERENCES TIENDA(codigo),
    FOREIGN KEY (PRODUCTO_codigo) REFERENCES PRODUCTO(codigo)
);
```