



MANUAL DEL ERP

**DAM – SISTEMAS DE GESTIÓN
EMPRESARIAL**

(GRUPO 10)

Integrantes:

- *Cristina Silván Sadoc*
- *Joaquín Domínguez Santana*
- *Ana Isabel Guzmán Bandera*

ÍNDICE

Introducción

- 1.1. Objetivo del documento*
- 1.2. Tecnologías utilizadas*

Modelado de la Base de Datos

- 2.1. Modelo Entidad-Relación (E/R)*
- 2.2. Modelo Lógico*
- 2.3. Descripción de las tablas*
- 2.4. Explicación de claves primarias y foráneas*

Descripción del Sistema

- 3.1. Funcionalidades principales del ERP*
- 3.2. Arquitectura del sistema*

Interfaces del Programa

- 4.1. Interfaz de acceso (Login)*
- 4.2. Menú principal del ERP*
- 4.3. Gestión de clientes*
- 4.4. Gestión de proveedores*
- 4.5. Gestión de productos*
- 4.6. Gestión de empleados*
- 4.7. Gestión de pedidos y ventas*

Procesos CRUD

- 5.1. CRUD de Clientes*
- 5.2. CRUD de Proveedores*
- 5.3. CRUD de Productos*
- 5.4. CRUD de Usuarios*

Conclusión y Futuras Mejoras

INTRODUCCIÓN

1.1. Objetivo del documento

Con este documento, tratamos de desarrollar un manual técnico del ERP en proceso.

Detallamos la estructura de la base de datos, cómo funcionará la aplicación echándole un vistazo al wireframe realizado como prototipo, la estructura del código fuente y los procesos CRUD implementados.

El ERP desarrollado está diseñado para la gestión integral de una empresa, permitiendo administrar la información de clientes, proveedores, productos y empleados. Entre las funcionalidades principales tenemos:

- Registro, consulta, actualización y eliminación de clientes, proveedores, productos y empleados.
- Acceso mediante una interfaz de login.
- Un menú principal desde el cual se accede a las diferentes secciones del sistema.
- Interfaz gráfica para facilitar la gestión de la información.

1.2. Tecnologías utilizadas

Para el desarrollo del ERP se han empleado las siguientes tecnologías:

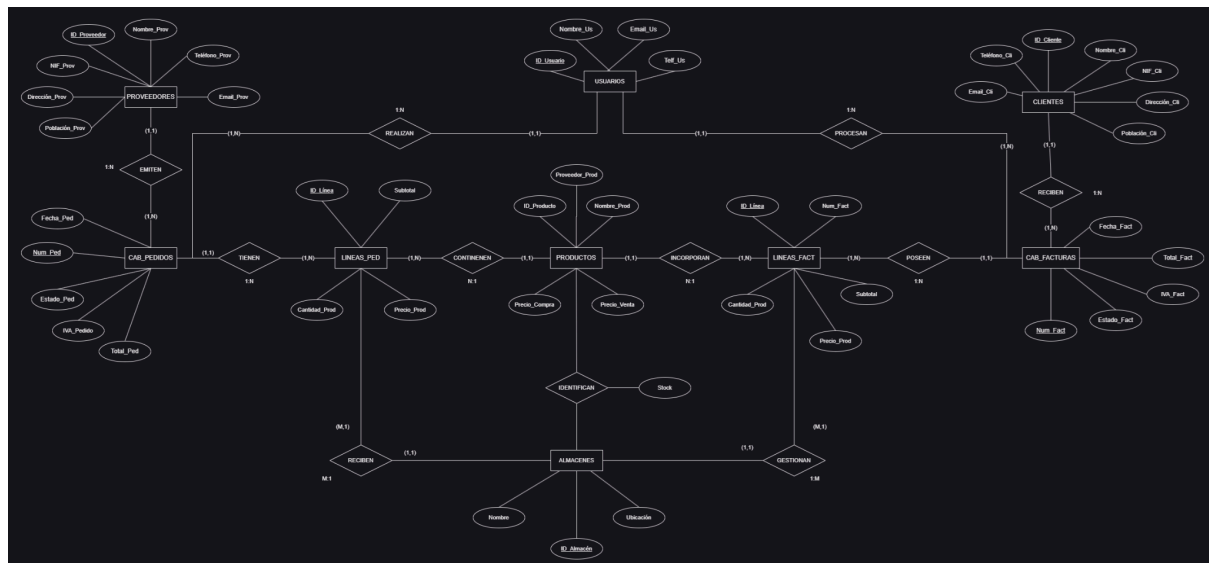
- **Lenguaje de programación:** Java
- **Entorno de desarrollo:** Visual Studio Code
- **Gestor de bases de datos:** SQLite
- **Frameworks y Librerías:** JDBC para la conexión con la base de datos
- **Control de versiones:** Git y GitHub
- **Patrón de diseño:** DAO (Data Access Object)

MODELADO DE LA BASE DE DATOS

Para comenzar con la realización de este proyecto, debemos crear un registro de todos los aspectos que nos serán de utilidad y cómo se relacionan entre sí. En concreto modelamos los datos referentes a: clientes, proveedores, facturas, usuarios, almacenes, pedidos...

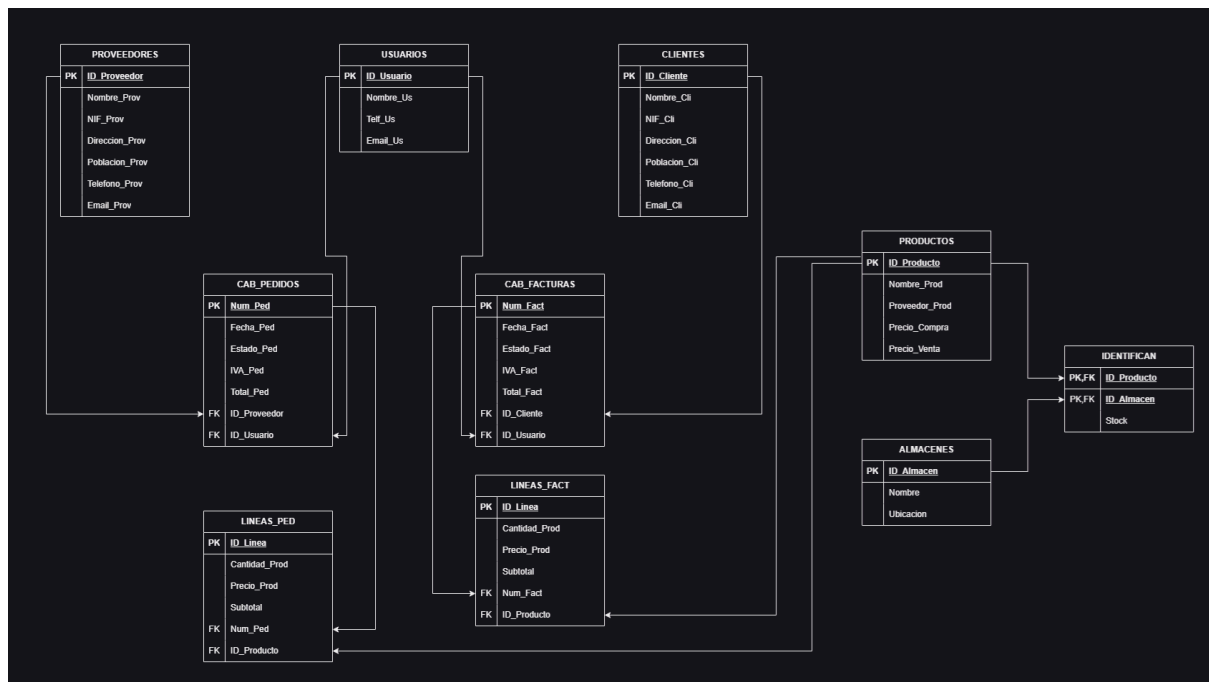
2.1. Modelo Entidad-Relación (E/R)

Con el modelo Entidad-Relación hacemos visible de forma gráfica cómo se relacionan las distintas entidades comentadas anteriormente y sus atributos (distintos tipos de datos que necesitaremos referentes a cada entidad).



2.2. Modelo Lógico

Como paso intermedio para crear el modelo físico de nuestra base de datos, haremos la conversión del modelo Entidad-Relación al modelo lógico o relacional, estableciendo claramente las claves primarias y foráneas de cada entidad y sus relaciones.



2.3. Descripción de las tablas

Una vez creado el modelo relacional, podemos elaborar nuestra base de datos donde incluiremos cada tabla con sus atributos y características en lenguaje SQL.

```
CREATE TABLE sqlite_sequence(name,seq);

CREATE TABLE CLIENTES (
  Id_Cliente INTEGER PRIMARY KEY AUTOINCREMENT,
  Nombre_Cli VARCHAR(100),
  NIF_Cli VARCHAR(50),
  Direccion_Cli VARCHAR(255),
  Poblacion_Cli VARCHAR(100),
  Telefono_Cli VARCHAR(20),
  Email_Cli VARCHAR(100)
);

CREATE TABLE PRODUCTOS (
  Id_Prod INTEGER PRIMARY KEY AUTOINCREMENT,
  Nombre_Prod VARCHAR(100),
  Proveedor_Prod VARCHAR(100),
  Precio_Compra DECIMAL(10, 2),
  Precio_Venta DECIMAL(10, 2)
);

CREATE TABLE PROVEEDORES(
  Id_Proveedor INTEGER PRIMARY KEY AUTOINCREMENT,
  Nombre_Prov VARCHAR(100),
  NIF_Prov VARCHAR(50),
  Direccion_Prov VARCHAR(255),
  Poblacion_Prov VARCHAR(100),
  Telefono_Prov VARCHAR(20),
  Email_Prov VARCHAR(100)
);

CREATE TABLE USUARIOS (
  Id_Usuario INTEGER PRIMARY KEY AUTOINCREMENT,
  Nombre_Us VARCHAR(100),
  Telefono_Us VARCHAR(20),
  Email_Us VARCHAR(100)
);

CREATE TABLE ALMACEN (
  Id_Almacen INTEGER PRIMARY KEY AUTOINCREMENT,
  Nombre_Alm VARCHAR(100),
  Ubicacion_Alm VARCHAR(255)
);
```

2.4. Explicación de claves primarias y foráneas

Junto con la descripción de las tablas añadimos las claves foráneas que relacionan unas entidades con otras y las claves primarias que nos permitirán acceder a cada registro.

```
CREATE TABLE CAB_PEDIDOS (
  Num_Ped INTEGER PRIMARY KEY AUTOINCREMENT,
  Fecha_Ped DATE,
  Estado_Ped VARCHAR(50),
  IVA_Ped DECIMAL(5, 2),
  Total_Ped DECIMAL(10, 2),
  Id_Proveedor INT,
  Id_Usuario INT,
  FOREIGN KEY (Id_Proveedor) REFERENCES PROVEEDORES(Id_Proveedor),
  FOREIGN KEY (Id_Usuario) REFERENCES USUARIOS(Id_Usuario)
);

CREATE TABLE CAB_FACTURAS (
  Num_Fact INTEGER PRIMARY KEY AUTOINCREMENT,
  Fecha_Fact DATE,
  Estado_Fact VARCHAR(50),
  IVA_Fact DECIMAL(5, 2),
  Total_Fact DECIMAL(10, 2),
  Id_Cliente INT,
  Id_Usuario INT,
  FOREIGN KEY (Id_Cliente) REFERENCES CLIENTES(Id_Cliente),
  FOREIGN KEY (Id_Usuario) REFERENCES USUARIOS(Id_Usuario)
);

CREATE TABLE PRODUCTOS_ALMACEN (
  Id_Producto INT,
  Id_Almacen INT,
  Stock INT,
  PRIMARY KEY (Id_Producto, Id_Almacen),
  FOREIGN KEY (Id_Producto) REFERENCES PRODUCTOS(Id_Prod),
  FOREIGN KEY (Id_Almacen) REFERENCES ALMACEN(Id_Almacen)
);

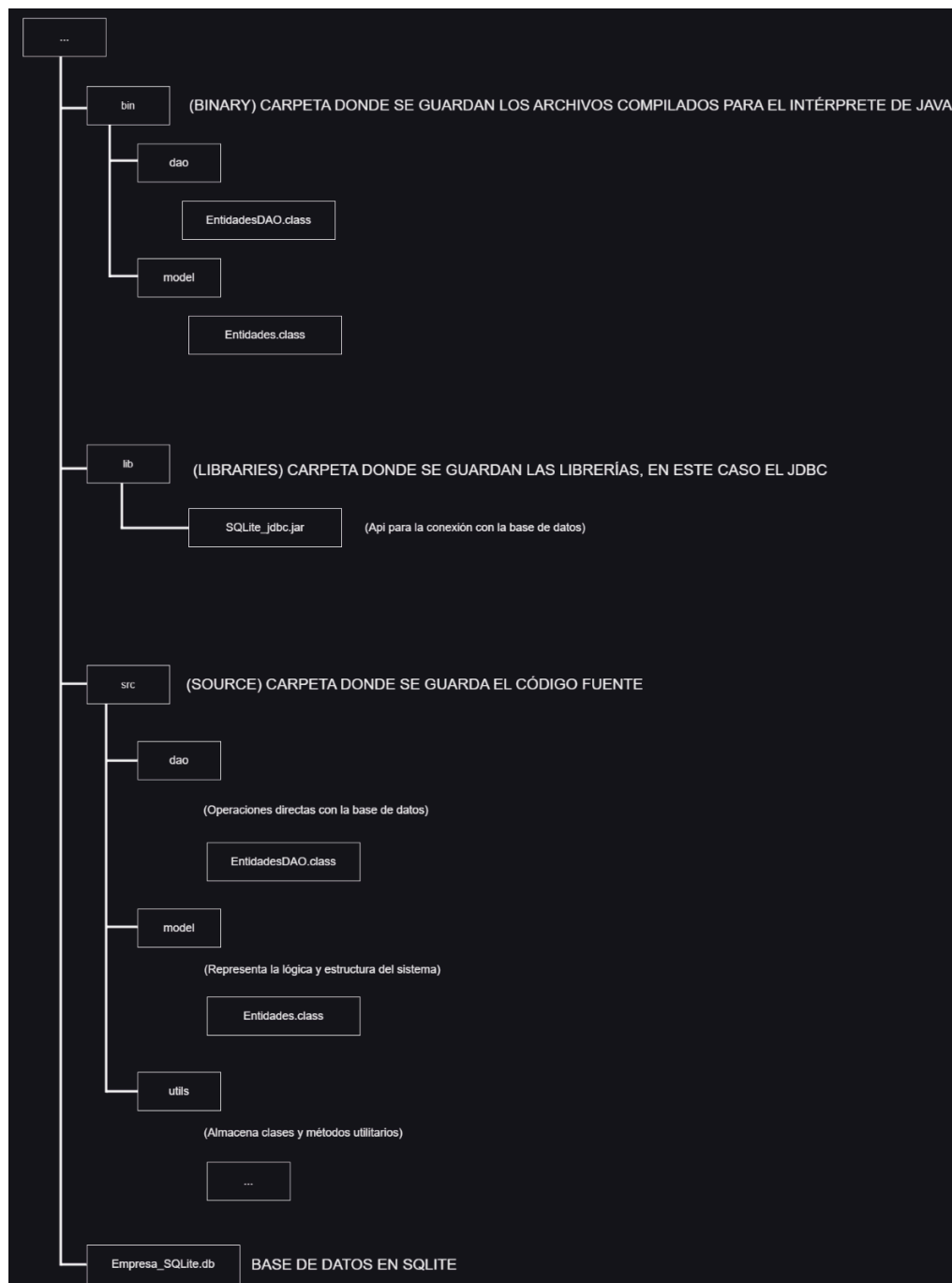
CREATE TABLE LINEAS_PED (
  Id_Linea_Ped INT PRIMARY KEY,
  Cantidad_Prod_Ped INT,
  Precio_Prod_Ped DECIMAL(10, 2),
  Subtotal_Ped DECIMAL(10, 2),
  Num_Ped INT,
  Id_Producto INT,
  Id_Almacen INT,
  FOREIGN KEY (Num_Ped) REFERENCES CAB_PEDIDOS(Num_Ped),
  FOREIGN KEY (Id_Producto) REFERENCES PRODUCTOS(Id_Prod),
  FOREIGN KEY (Id_Almacen) REFERENCES ALMACEN(Id_Almacen)
);

CREATE TABLE LINEAS_FACT (
  Id_Linea_Fact INT PRIMARY KEY,
  Cantidad_Prod_Fact INT,
  Precio_Prod_Fact DECIMAL(10, 2),
  Subtotal_Fact DECIMAL(10, 2),
  Num_Fact INT,
  Id_Producto INT,
  Id_Almacen INT,
  FOREIGN KEY (Num_Fact) REFERENCES CAB_FACTURAS(Num_Fact),
  FOREIGN KEY (Id_Producto) REFERENCES PRODUCTOS(Id_Prod),
  FOREIGN KEY (Id_Almacen) REFERENCES ALMACEN(Id_Almacen)
);
```

DESCRIPCIÓN DEL SISTEMA

3.2. Arquitectura del sistema

El proyecto está diseñado según el patrón DAO, quedando de la forma siguiente:



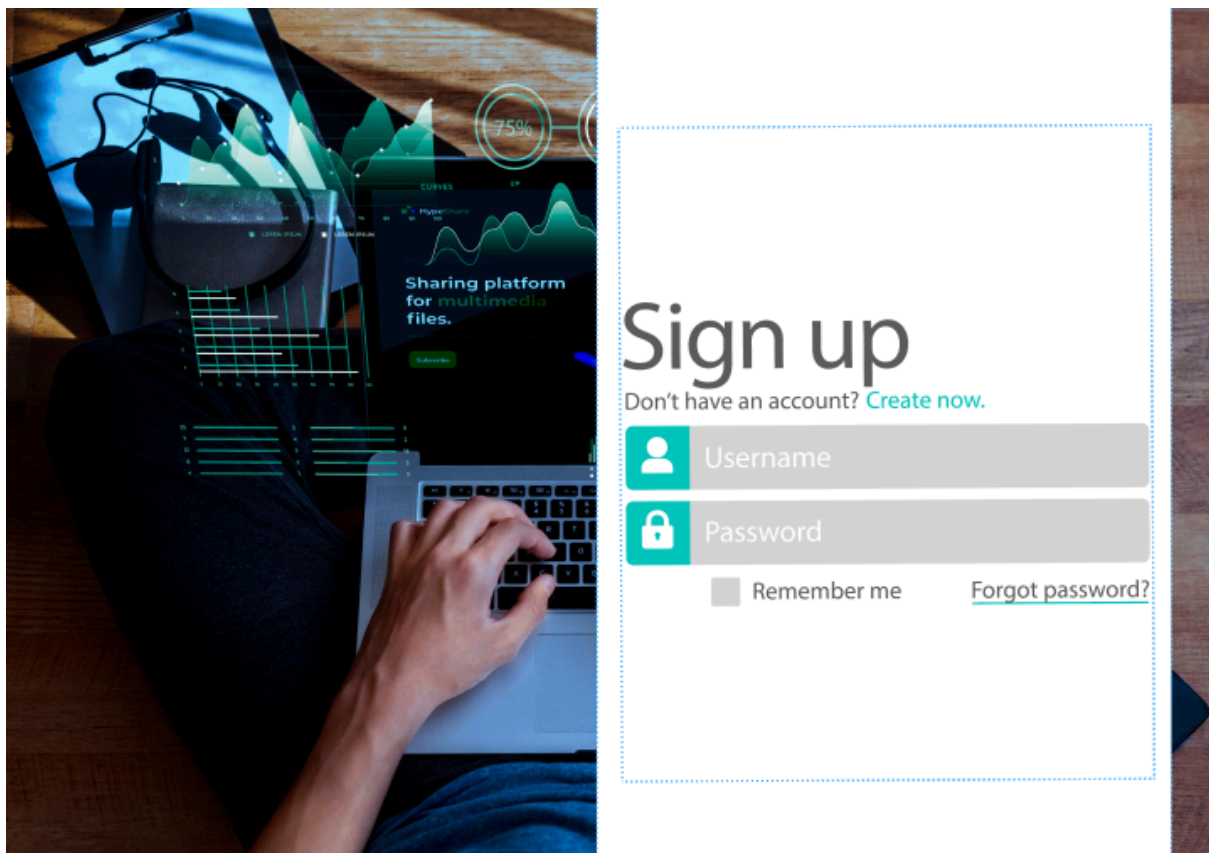
INTERFACES DEL PROGRAMA

Para comenzar con la parte gráfica de nuestro proyecto hemos diseñado las distintas interfaces del programa con Figma, incluyendo todos los aspectos necesarios para la completa funcionalidad.

Para ello hemos incorporado un menú donde de forma sencilla podamos acceder a cada apartado de nuestro programa. En ellos podemos consultar, añadir, eliminar información almacenada en nuestra base de datos sobre los distintos aspectos necesarios para nuestro ERP: clientes, proveedores, facturas, productos...

A continuación mostramos cómo quedarían los bocetos de los distintos apartados de nuestro programa.

4.1. Interfaz de acceso (login)



[illegible]

[illegible]

Home

Cientes

Productos

Facturas

Proveedores

Empleados

Clientes

Modificar cliente

Modificar

Deshacer

¿Qué cliente deseas modificar?

Indica sus datos

Nombre

Email

NIF

Teléfono

Dirección

Población


Buscar

Modificar

Deshacer


ID_Cliente	Nombre	Email	NIF	Teléfono	Dirección	Población
1	John Doe	john.doe@example.com	123456789	1234567890	123 Main St	New York


4.4. Gestión de proveedores





Home
Clientes
Productos
Facturas
Proveedores
Empleados

Q 🔔

 Usuario ▾

Proveedores 

 Añadir proveedor

 Buscar proveedor

ID_Proveedor	Nombre	Email	NIF	Teléfono	Dirección	Población
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum



Home
Clientes
Productos
Facturas
Proveedores
Empleados

Q 🔔

 Lorem Ipsum ▾

Proveedores 

Añadir proveedor

Guardar

Deshacer

¿Qué proveedor deseas añadir?
Indica sus datos

Nombre

Email

NIF

Teléfono


Dirección



Población



Guardar

Deshacer

4.5. Gestión de productos





 Usuario 

Home


Cientes



Productos

Facturas

Proveedores

Empleados


Productos 



 



Añadir producto

Buscar producto

ID_Producto	Nombre	Proveedor	Precio compra	Precio venta
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum
Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum	Lorem Ipsum





 Lorem Ipsum 

Home


Cientes

Productos

Facturas

Proveedores

Empleados

Productos 

Añadir producto

Guardar

Deshacer

¿Qué producto deseas añadir?
Indica sus datos

Nombre

Proveedor

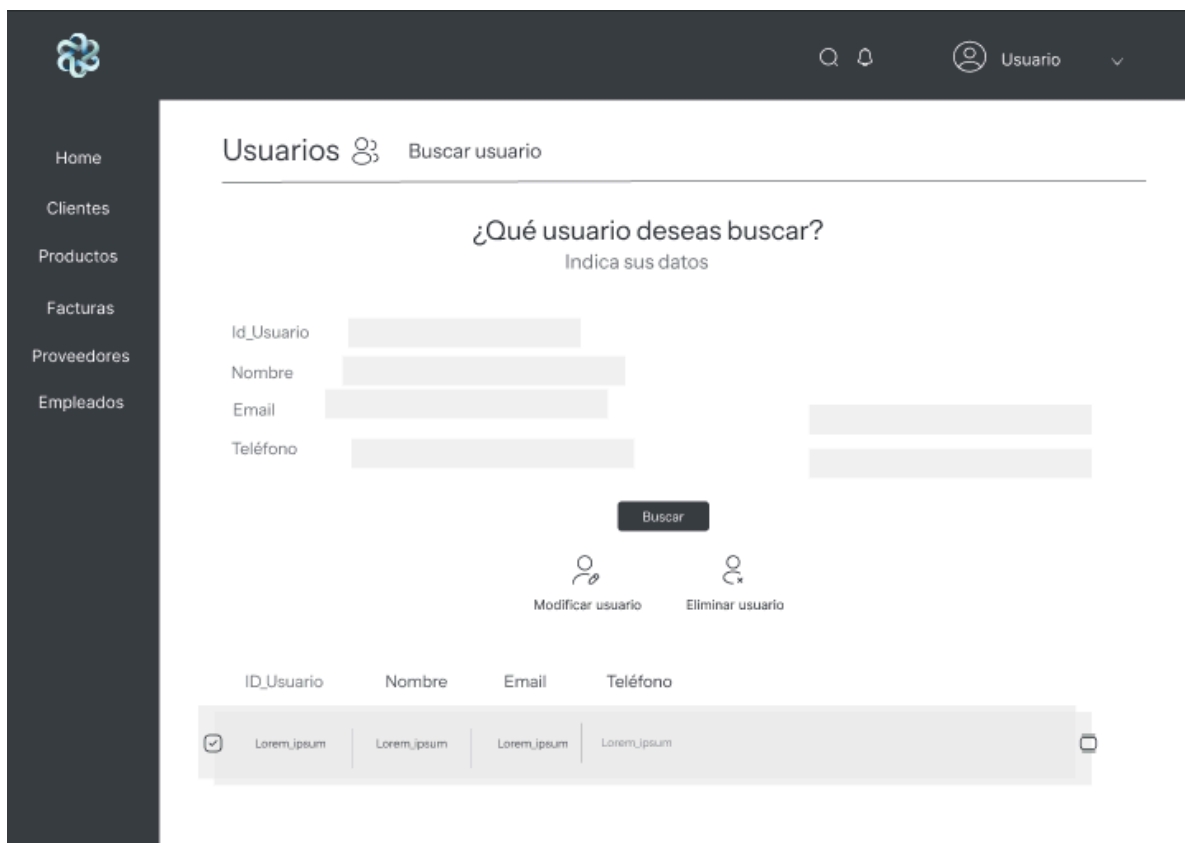
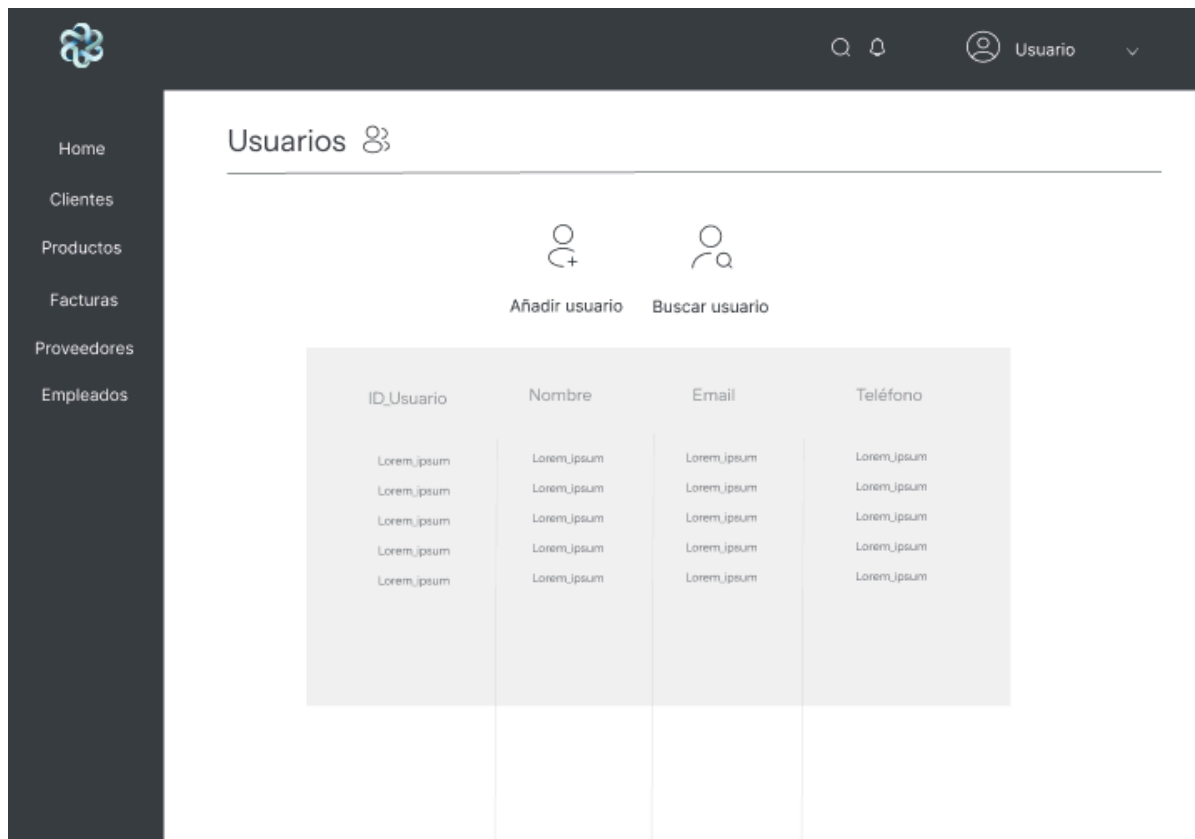
Precio compra

Precio venta

Guardar

Deshacer

4.6. Gestión de usuarios



[illegible]

[Home](#)

Cientes

Productos

Facturas

Proveedores

Empleados

Facturas



Añadir factura

¿Qué factura deseas añadir?

Indica sus datos

Número factura

Fecha

Cliente

ID_Producto	Nombre	Cantidad	Precio	Subtotal	IVA	Total
	Lorem_ipsum		Lorem_ipsum	Lorem_ipsum	Lorem_ipsum	Lorem_ipsum

Guendel

Dashacer

IVA_Fact

Lorem ipsum

Total Factura

Lorem ipsum

PROCESOS CRUD

5.1. CRUD de Clientes

1. En **src/model** se encuentra el diseño de la entidad CLIENTES:

```
public class Clientes {
    private int id_cliente;
    private String nombre_cliente;
    private String nif_cliente;
    private String direccion_cliente;
    private String poblacion_cliente;
    private String telefono_cliente;
    private String email_cliente;

    //CONSTRUCTORES
    public Clientes(){

    }

    public Clientes(int id, String nombre, String nif, String direccion, String poblacion, String telefono, String email){
        this.id_cliente = id;
        this.nombre_cliente = nombre;
        this.nif_cliente = nif;
        this.direccion_cliente = direccion;
        this.poblacion_cliente = poblacion;
        this.telefono_cliente = telefono;
        this.email_cliente = email;
    }
}
```

```
//GETTERS
public int getId(){
    return this.id_cliente;
}
public String getNombre(){
    return this.nombre_cliente;
}
public String getNif(){
    return this.nif_cliente;
}
public String getDireccion(){
    return this.direccion_cliente;
}
public String getPoblacion (){
    return this.poblacion_cliente;
}
public String getTelefono(){
    return this.telefono_cliente;
}
public String getEmail(){
    return this.email_cliente;
}
```

```
//SETTERS
public void setId(int id){
    this.id_cliente = id;
}
public void setNombre(String nombre){
    this.nombre_cliente = nombre;
}
public void setNif(String nif){
    this.nif_cliente = nif;
}
public void setDireccion(String direccion){
    this.direccion_cliente = direccion;
}
public void setPoblacion(String poblacion){
    this.poblacion_cliente = poblacion;
}
public void setTelefono(String telefono){
    this.telefono_cliente = telefono;
}
public void setEmail(String email){
    this.email_cliente = email;
}
```

```
//TO-STRING
@Override
public String toString(){
    return "CLIENTE " + id_cliente + "/// Nombre: " + nombre_cliente + ", NIF: " + nif_cliente
    + ", Direccion: " + direccion_cliente + ", Poblacion: " + poblacion_cliente + ", Tlf: " + telefono_cliente +
    ", Email: " + email_cliente;
}
```

Teniendo una estructura básica como cualquier clase estándar.

2. En **src/dao** por otra parte, tenemos el CRUD para la entidad CLIENTES...

- Método CREATE

```
import java.util.ArrayList;
import java.util.List;
import model.Clientes;
import model.ConexionBD;
import java.sql.*;

public class ClientesDAO {

    //CREATE
    public void crearCliente(Clientes nuevo_cliente){
        String sql = "INSERT INTO clientes (Id_Cliente, Nombre_Cli, NIF_Cli, Direccion_Cli, Poblacion_Cli, Telefono_Cli, Email_Cli) VALUES (?, ?, ?, ?, ?, ?, ?)";

        try (Connection conn = ConexionBD.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(parameterIndex:1, nuevo_cliente.getId());
            stmt.setString(parameterIndex:2, nuevo_cliente.getNombre());
            stmt.setString(parameterIndex:3, nuevo_cliente.getNif());
            stmt.setString(parameterIndex:4, nuevo_cliente.getDireccion());
            stmt.setString(parameterIndex:5, nuevo_cliente.getPoblacion());
            stmt.setString(parameterIndex:6, nuevo_cliente.getTelefono());
            stmt.setString(parameterIndex:7, nuevo_cliente.getEmail());

            stmt.executeUpdate();
            System.out.println("(CLIENTE AGREGADO CORRECTAMENTE) " + nuevo_cliente);
        } catch (SQLException e){
            e.printStackTrace();
            System.out.println(x:"(NO SE HA PODIDO CREAR EL CLIENTE)...");
        }
    }
}
```

- Método READ

```
// READ (Devuelve un solo cliente mediante su id)
public Clientes obtenerClienteMedianteID(int id){
    String sql = "SELECT * FROM clientes WHERE Id_Cliente = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            Clientes cliente = new Clientes();
            cliente.setId(rs.getInt(columnLabel:"Id_Cliente"));
            cliente.setNombre(rs.getString(columnLabel:"Nombre_Cli"));
            cliente.setNif(rs.getString(columnLabel:"NIF_Cli"));
            cliente.setDireccion(rs.getString(columnLabel:"Direccion_Cli"));
            cliente.setPoblacion(rs.getString(columnLabel:"Poblacion_Cli"));
            cliente.setTelefono(rs.getString(columnLabel:"Telefono_Cli"));
            cliente.setEmail(rs.getString(columnLabel:"Email_Cli"));
            return cliente;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(CLIENTE NO ENCONTRADO)...");
    return null;
}
```

- Método UPDATE

```
// UPDATE
public boolean actualizarCliente(int id, String nuevo_nombre, String nuevo_NIF, String nuevo_direccion, String nuevo_poblacion, String nuevo_telefono, String nuevo_email){
    String sql = "UPDATE clientes SET Nombre_Cli = ?, NIF_Cli = ?, Direccion_Cli = ?, Poblacion_Cli = ?, Telefono_Cli = ?, Email_Cli = ? WHERE Id_Cliente = ?";

    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(parameterIndex:1, nuevo_nombre);
        stmt.setString(parameterIndex:2, nuevo_NIF);
        stmt.setString(parameterIndex:3, nuevo_direccion);
        stmt.setString(parameterIndex:4, nuevo_poblacion);
        stmt.setString(parameterIndex:5, nuevo_telefono);
        stmt.setString(parameterIndex:6, nuevo_email);
        stmt.setInt(parameterIndex:7, id);

        int filasActualizadas = stmt.executeUpdate();
        if (filasActualizadas > 0) {
            System.out.println(x:"(CLIENTE ACTUALIZADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return false;
}
```

- Método DELETE

```
public boolean eliminarCliente(int id){
    String sql = "DELETE FROM clientes WHERE Id_Cliente = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);

        int filasEliminadas = stmt.executeUpdate();
        if (filasEliminadas > 0) {
            System.out.println(x:"(CLIENTE ELIMINADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

    System.out.println(x:"(CLIENTE NO ENCONTRADO)...");
    return false;
}
```

5.2. CRUD de Proveedores

Las estructuras del resto de entidades, siguen el mismo diseño.

1. En **src/model** se encuentra el diseño de la entidad **PROVEEDORES**:

```
public class Proveedores {
    private int id_proveedor;
    private String nombre_proveedor;
    private String nif_proveedor;
    private String direccion_proveedor;
    private String poblacion_proveedor;
    private String telefono_proveedor;
    private String email_proveedor;

    //CONSTRUCTORES
    public Proveedores(){

    }

    public Proveedores(int id, String nombre, String nif, String direccion, String poblacion, String telefono, String email){
        this.id_proveedor = id;
        this.nombre_proveedor = nombre;
        this.nif_proveedor = nif;
        this.direccion_proveedor = direccion;
        this.poblacion_proveedor = poblacion;
        this.telefono_proveedor = telefono;
        this.email_proveedor = email;
    }
}
```

```
//GETTERS
public int getId(){
    return this.id_proveedor;
}
public String getNombre(){
    return this.nombre_proveedor;
}
public String getNif(){
    return this.nif_proveedor;
}
public String getDireccion(){
    return this.direccion_proveedor;
}
public String getPoblacion (){
    return this.poblacion_proveedor;
}
public String getTelefono(){
    return this.telefono_proveedor;
}
public String getEmail(){
    return this.email_proveedor;
}
```

```
//SETTERS
public void setId(int id){
    this.id_proveedor = id;
}
public void setNombre(String nombre){
    this.nombre_proveedor = nombre;
}
public void setNif(String nif){
    this.nif_proveedor = nif;
}
public void setDireccion(String direccion){
    this.direccion_proveedor = direccion;
}
public void setPoblacion(String poblacion){
    this.poblacion_proveedor = poblacion;
}
public void setTelefono(String telefono){
    this.telefono_proveedor = telefono;
}
public void setEmail(String email){
    this.email_proveedor = email;
}
```

```
//TO-STRING
@Override
public String toString(){
    return "PROVEEDOR " + id_proveedor + " // Nombre: " + nombre_proveedor + ", NIF: " + nif_proveedor +
        ", Direccion: " + direccion_proveedor + ", Poblacion: " + poblacion_proveedor + ", Tlf: " + telefono_proveedor +
        ", Email: " + email_proveedor;
}
```

2. En **src/dao** tenemos el CRUD para la entidad PROVEEDORES...

- Método CREATE

```
public class ProveedoresDAO{

    //CREATE
    public void crearProveedor(Proveedores nuevo_proveedor){
        String sql = "INSERT INTO productos (Id_Proveedor, Nombre_Prov, NIF_Prov, Direccion_Prov, Poblacion_Prov, Telefono_Prov, Email_Prov) VALUES (?, ?, ?, ?, ?, ?, ?)";

        try (Connection conn = ConexionBD.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(parameterIndex:1, nuevo_proveedor.getId());
            stmt.setString(parameterIndex:2, nuevo_proveedor.getNombre());
            stmt.setString(parameterIndex:3, nuevo_proveedor.getNif());
            stmt.setString(parameterIndex:4, nuevo_proveedor.getDireccion());
            stmt.setString(parameterIndex:5, nuevo_proveedor.getPoblacion());
            stmt.setString(parameterIndex:6, nuevo_proveedor.getTelefono());
            stmt.setString(parameterIndex:7, nuevo_proveedor.getEmail());

            stmt.executeUpdate();
            System.out.println("PROVEEDOR AGREGADO CORRECTAMENTE " + nuevo_proveedor);

            stmt.executeUpdate();
            System.out.println(x:"(PROVEEDOR AGREGADO CORRECTAMENTE)");
        } catch (SQLException e){
            e.printStackTrace();
            System.out.println(x:"(NO SE HA PODIDO CREAR EL PROVEEDOR)...");
        }
    }
}
```

- Método READ

```
// READ (Devuelve un solo proveedor mediante su id)
public Proveedores obtenerProveedorMedianteID(int id){
    String sql = "SELECT * FROM productos WHERE Id_Proveedor = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            Proveedores proveedor = new Proveedores();
            proveedor.setId(rs.getInt(columnLabel:"Id_Proveedor"));
            proveedor.setNombre(rs.getString(columnLabel:"Nombre_Prov"));
            proveedor.setNif(rs.getString(columnLabel:"NIF_Prov"));
            proveedor.setDireccion(rs.getString(columnLabel:"Direccion_Prov"));
            proveedor.setPoblacion(rs.getString(columnLabel:"Poblacion_Prov"));
            proveedor.setTelefono(rs.getString(columnLabel:"Telefono_Prov"));
            proveedor.setEmail(rs.getString(columnLabel:"Email_Prov"));
            return proveedor;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(PROVEEDOR NO ENCONTRADO)...");
    return null;
}
```

- Método UPDATE

```
// UPDATE
public boolean actualizarProveedor(int id, String nuevo_nombre, String nuevo_NIF, String nuevo_direccion, String nuevo_poblacion, String nuevo_telefono, String nuevo_email){
    String sql = "UPDATE productos SET Nombre_Prov = ?, NIF_Prov = ?, Direccion_Prov = ?, Poblacion_Prov = ?, Telefono_Prov = ?, Email_Prov = ? WHERE Id_Proveedor = ?";

    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(parameterIndex:1, nuevo_nombre);
        stmt.setString(parameterIndex:2, nuevo_NIF);
        stmt.setString(parameterIndex:3, nuevo_direccion);
        stmt.setString(parameterIndex:4, nuevo_poblacion);
        stmt.setString(parameterIndex:5, nuevo_telefono);
        stmt.setString(parameterIndex:6, nuevo_email);
        stmt.setInt(parameterIndex:7, id);

        int filasActualizadas = stmt.executeUpdate();
        if (filasActualizadas > 0) {
            System.out.println(x:"(PROVEEDOR ACTUALIZADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

- Método DELETE

```
public boolean eliminarProveedor(int id){
    String sql = "DELETE FROM productos WHERE Id_Proveedor = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);

        int filasEliminadas = stmt.executeUpdate();
        if (filasEliminadas > 0) {
            System.out.println(x:"(PROVEEDOR ELIMINADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(PROVEEDOR NO ENCONTRADO)...");
    return false;
}
```

5.3. CRUD de Productos

1. En **src/model** se encuentra el diseño de la entidad PRODUCTOS:

```
public class Productos{
    private int id_producto;
    private String nombre_producto;
    private String proveedor_producto;
    private float precio_compra_producto;
    private float precio_venta_producto;

    //CONSTRUCTORES
    public Productos(){

    }

    public Productos(int id, String nombre, String proveedor, float compra, float venta){
        this.id_producto = id;
        this.nombre_producto = nombre;
        this.proveedor_producto = proveedor;
        this.precio_compra_producto = compra;
        this.precio_venta_producto = venta;
    }

    //EL RESTO DEL CÓDIGO FUENTE ES IGUAL QUE LAS CLASES ANTERIORES
}
```

2. En **src/dao** tenemos el CRUD para la entidad PRODUCTOS...

- Método CREATE

```
public class ProductosDAO{

    //CREATE
    public void crearProducto(Productos nuevo_producto){
        String sql = "INSERT INTO productos (Id_Prod, Nombre_Prod, Proveedor_Prod, Precio_Compra, Precio_Venta) VALUES (?, ?, ?, ?, ?)";

        try (Connection conn = ConexionBD.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(parameterIndex:1, nuevo_producto.getId());
            stmt.setString(parameterIndex:2, nuevo_producto.getNombre());
            stmt.setString(parameterIndex:2, nuevo_producto.getProveedor());
            stmt.setFloat(parameterIndex:2, nuevo_producto.getPrecioCompra());
            stmt.setFloat(parameterIndex:2, nuevo_producto.getPrecioVenta());

            stmt.executeUpdate();
            System.out.println("(PRODUCTO AGREGADO CORRECTAMENTE) " + nuevo_producto);

            stmt.executeUpdate();
            System.out.println(x: "(PRODUCTO AGREGADO CORRECTAMENTE)");
        } catch (SQLException e){
            e.printStackTrace();
            System.out.println(x: "(NO SE HA PODIDO CREAR EL PRODUCTO)...");
        }
    }
}
```


- Método READ

```
// READ (Devuelve un solo producto mediante su id)
public Productos obtenerProductoMedianteID(int id){
    String sql = "SELECT * FROM productos WHERE Id_Prod = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            Productos producto = new Productos();
            producto.setId(rs.getInt(columnLabel:"Id_Prod"));
            producto.setNombre(rs.getString(columnLabel:"Nombre_Prod"));
            producto.setProveedor(rs.getString(columnLabel:"Proveedor_Prod"));
            producto.setPrecioCompra(rs.getFloat(columnLabel:"Precio_Compra"));
            producto.setPrecioVenta(rs.getFloat(columnLabel:"Precio_Venta"));
            return producto;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(PRODUCTO NO ENCONTRADO)...");
    return null;
}
```

- Método UPDATE

```
// UPDATE
public boolean actualizarProducto(int id, String nuevo_nombre, String nuevo_proveedor, Float nuevo_precio_compra, Float nuevo_precio_venta){
    String sql = "UPDATE productos SET Nombre_Prod = ?, Proveedor_Prod = ?, Precio_Compra = ?, Precio_Venta = ? WHERE Id_Prod = ?";

    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(parameterIndex:1, nuevo_nombre);
        stmt.setString(parameterIndex:2, nuevo_proveedor);
        stmt.setFloat(parameterIndex:3, nuevo_precio_compra);
        stmt.setFloat(parameterIndex:4, nuevo_precio_venta);
        stmt.setInt(parameterIndex:5, id);

        int filasActualizadas = stmt.executeUpdate();
        if (filasActualizadas > 0) {
            System.out.println(x:"(PRODUCTO ACTUALIZADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

- Método DELETE

```
public boolean eliminarProducto(int id){
    String sql = "DELETE FROM productos WHERE Id_Prod = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);

        int filasEliminadas = stmt.executeUpdate();
        if (filasEliminadas > 0) {
            System.out.println(x:"(PRODUCTO ELIMINADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(PRODUCTO NO ENCONTRADO)...");
    return false;
}
```

5.4. CRUD de Usuarios

1. En **src/model** se encuentra el diseño de la entidad USUARIOS:

```
public class Usuarios {
    private int id_usuario;
    private String nombre_usuario;
    private String telefono_usuario;
    private String email_usuario;

    //CONSTRUCTORES
    public Usuarios(){

    }

    public Usuarios(int id, String nombre, String telefono, String email){
        this.id_usuario = id;
        this.nombre_usuario = nombre;
        this.telefono_usuario = telefono;
        this.email_usuario = email;
    }

    //EL RESTO DEL CÓDIGO FUENTE ES IGUAL QUE LAS CLASES ANTERIORES
}
```

2. En **src/dao** tenemos el CRUD para la entidad USUARIOS...

- Método CREATE

```
public class UsuariosDAO {

    //CREATE
    public void crearUsuario(Usuarios nuevo_usuario){
        String sql = "INSERT INTO usuarios (Id_Usuario, Nombre_Us, Telefono_Us, Email_Us) VALUES (?, ?, ?, ?)";

        try (Connection conn = ConexionBD.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(parameterIndex:1, nuevo_usuario.getId());
            stmt.setString(parameterIndex:2, nuevo_usuario.getNombre());
            stmt.setString(parameterIndex:3, nuevo_usuario.getTelefono());
            stmt.setString(parameterIndex:4, nuevo_usuario.getEmail());

            stmt.executeUpdate();
            System.out.println("(USUARIO AGREGADO CORRECTAMENTE) " + nuevo_usuario);
        } catch (SQLException e){
            e.printStackTrace();
            System.out.println(x:"(NO SE HA PODIDO CREAR EL USUARIO)...");
        }
    }
}
```

- Método READ

```
// READ (Devuelve un solo usuario mediante su id)
public Usuarios obtenerUsuarioMedianteID(int id){
    String sql = "SELECT * FROM usuarios WHERE Id_Usuario = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            Usuarios usuario = new Usuarios();
            usuario.setId(rs.getInt(columnLabel:"Id_Usuario"));
            usuario.setNombre(rs.getString(columnLabel:"Nombre_Us"));
            usuario.setTelefono(rs.getString(columnLabel:"Telefono_Us"));
            usuario.setEmail(rs.getString(columnLabel:"Email_Us"));
            return usuario;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(USUARIO NO ENCONTRADO)...");
    return null;
}
```

- Método UPDATE

```
// UPDATE
public boolean actualizarUsuario(int id, String nuevo_nombre, String nuevo_telefono, String nuevo_email){
    String sql = "UPDATE usuarios SET Nombre_Us = ?, Telefono_Us = ?, Email_Us = ? WHERE Id_Usuario = ?";

    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(parameterIndex:1, nuevo_nombre);
        stmt.setString(parameterIndex:2, nuevo_telefono);
        stmt.setString(parameterIndex:3, nuevo_email);
        stmt.setInt(parameterIndex:4, id);

        int filasActualizadas = stmt.executeUpdate();
        if (filasActualizadas > 0) {
            System.out.println(x:"(USUARIO ACTUALIZADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return false;
}
```

- Método DELETE

```
public boolean eliminarUsuario(int id){
    String sql = "DELETE FROM usuarios WHERE Id_Usuario = ?";
    try (Connection conn = ConexionBD.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(parameterIndex:1, id);

        int filasEliminadas = stmt.executeUpdate();
        if (filasEliminadas > 0) {
            System.out.println(x:"(USUARIO ELIMINADO)");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println(x:"(USUARIO NO ENCONTRADO)...");
    return false;
}
```

Adicionalmente, hemos desarrollado la clase para la conexión con la base de datos SQLite y una clase para la consulta de la estructura

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexionBD {
    private static final String URL = "jdbc:sqlite:Empresa_SQLite.db";

    public static Connection getConnection() {
        Connection conexion = null;
        try {
            Class.forName(className:"org.sqlite.JDBC");
            conexion = DriverManager.getConnection(URL);
            System.out.println(x:"(CONEXION ESTABLECIDA CON LA BASE DE DATOS)");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println(x:"(NO SE ENCONTRO LA BASE DE DATOS)");
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println(x:"(LA CONEXION NO SE PUDO ESTABLECER)");
        }
        return conexion;
    }
}
```

```
public class ConsultaEstructura {

    public static void verEstructuraTabla(String tabla) {
        String sql = "PRAGMA table_info(" + tabla + ")";

        System.out.println(x:"\n");
        try (Connection conn = ConexionBD.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery()) {
            System.out.println(x:"\n...\n");
            System.out.println("/TABLA " + tabla.toUpperCase() + "/" );
            while (rs.next()) {
                int cid = rs.getInt(columnLabel:"cid");
                String name = rs.getString(columnLabel:"name");
                String type = rs.getString(columnLabel:"type");
                int notnull = rs.getInt(columnLabel:"notnull");
                String dflt_value = rs.getString(columnLabel:"dflt_value");
                int pk = rs.getInt(columnLabel:"pk");

                System.out.println("CID: " + cid + " ||| Columna: " + name + " ||| Tipo: "
                    + type + " ||| Not Null: " + notnull + " ||| Default: " + dflt_value + " ||| Clave primaria: " + pk + " |||");
                /*
                CID es el índice de la columna (comienza desde 0).
                NAME es el nombre de la columna.
                TYPE es el tipo de datos de la columna.
                NOTNULL indica si la columna permite valores nulos (0 es permitido, 1 no lo es).
                DFLT_VALUE es el valor predeterminado de la columna (si tiene uno).
                PK indica si la columna es una clave primaria (1 es clave primaria, 0 no lo es).
                */
            }
            System.out.println(x:"...\n");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

CONCLUSIÓN Y FUTURAS MEJORAS

- **Conclusiones:**

- El patrón de diseño DAO es eficiente para proyectos Java de gran tamaño gracias a su modularidad y su estructura, que es fácilmente visible y accesible
- El jdbc es más cómodo que el Gradle o el Maven gracias a su curva de aprendizaje para principiantes
- SQLite es ideal para juniors como nosotros por su facilidad para de configuración, además de tener toda la base de datos compactada en un solo archivo portátil

- **Futuras mejoras:**

- Tenemos pendiente adaptar el CRUD para el campo id autoincremental que está diseñado como autoincremental
- Tenemos pendiente desarrollar las tablas para las relaciones entre las entidades