

Universidad Mariano Gálvez

Sede Boca del Monte

Facultad de Ingeniería en Sistemas

Sección: A

Catedrático: Ingeniero Luis Alvarado

Tema:

Proyecto Final

Nombre: Walter Yair Ramos Carreto

Carnet: 7690-23-17204

Guatemala, 25 de octubre del 2024

Diagrama de clases (UML)

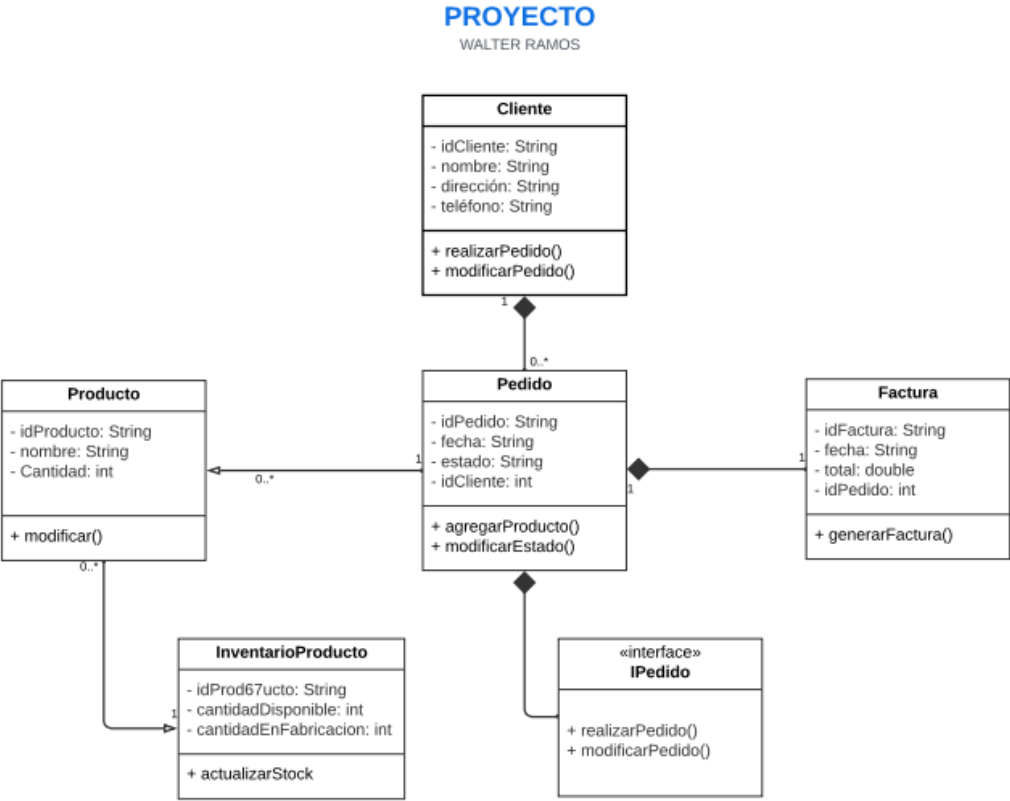
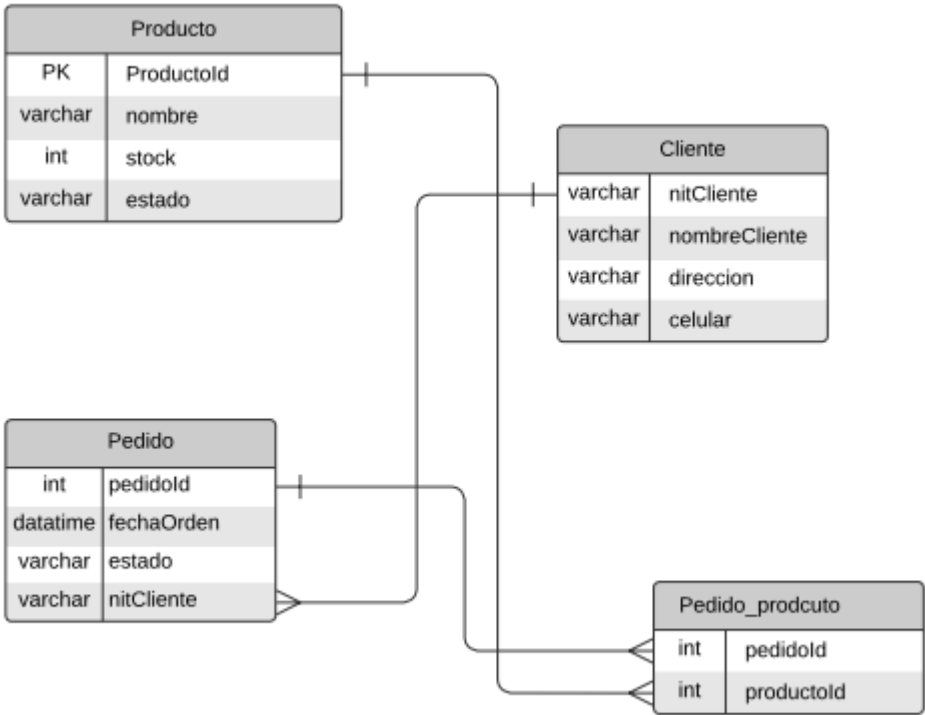
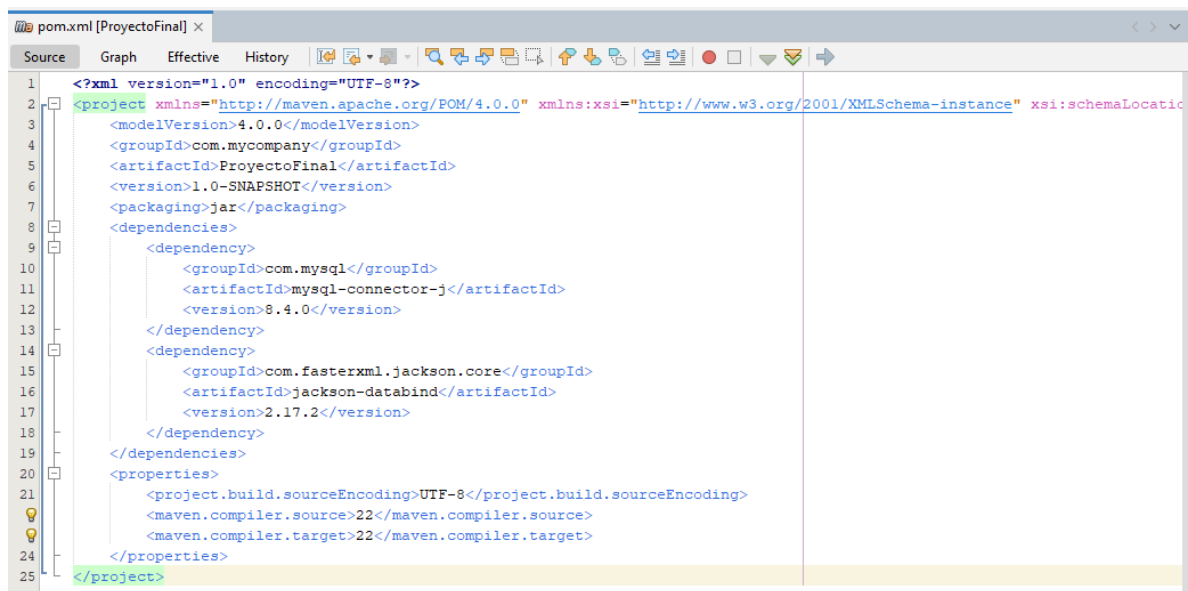


Diagrama Identidad – Relación



pom.xml

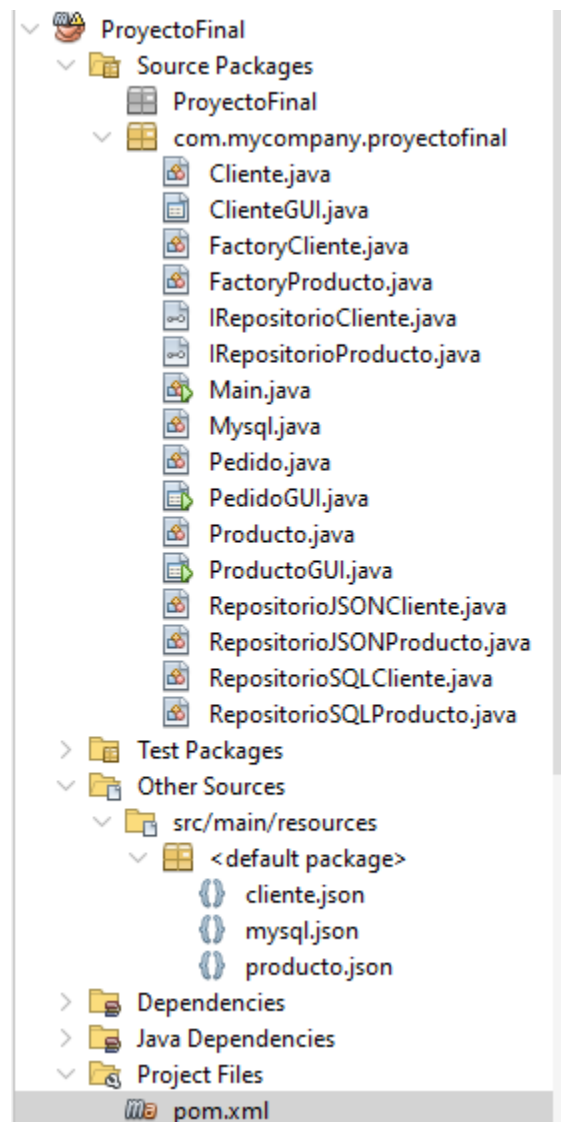
El cual se utilizó Maven con Java 17



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.mycompany</groupId>
5   <artifactId>ProyectoFinal</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <packaging>jar</packaging>
8   <dependencies>
9     <dependency>
10       <groupId>com.mysql</groupId>
11       <artifactId>mysql-connector-j</artifactId>
12       <version>8.4.0</version>
13     </dependency>
14     <dependency>
15       <groupId>com.fasterxml.jackson.core</groupId>
16       <artifactId>jackson-databind</artifactId>
17       <version>2.17.2</version>
18     </dependency>
19   </dependencies>
20   <properties>
21     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22     <maven.compiler.source>22</maven.compiler.source>
23     <maven.compiler.target>22</maven.compiler.target>
24   </properties>
25 </project>
```

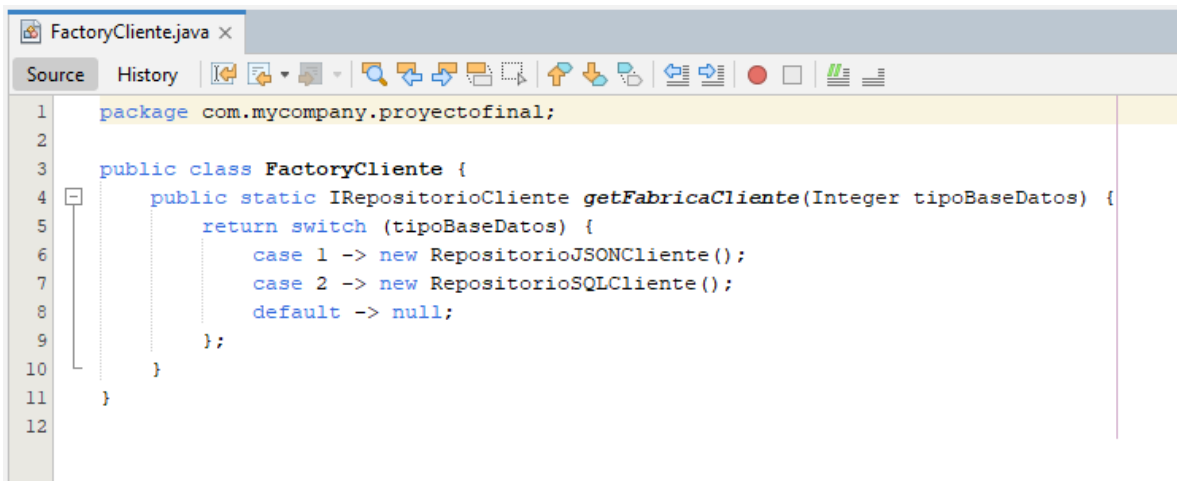
Utilizamos varias librerías como por ejemplo el mysql-connector-j; esta librería la utilizamos para poder establecer una conexión a nuestra base de datos

Al igual utilizamos la librería com.fasterxml.jackson.core; esta librería la utilizamos para manejar nuestras clases que son .json



Patrones de Diseño utilizados:

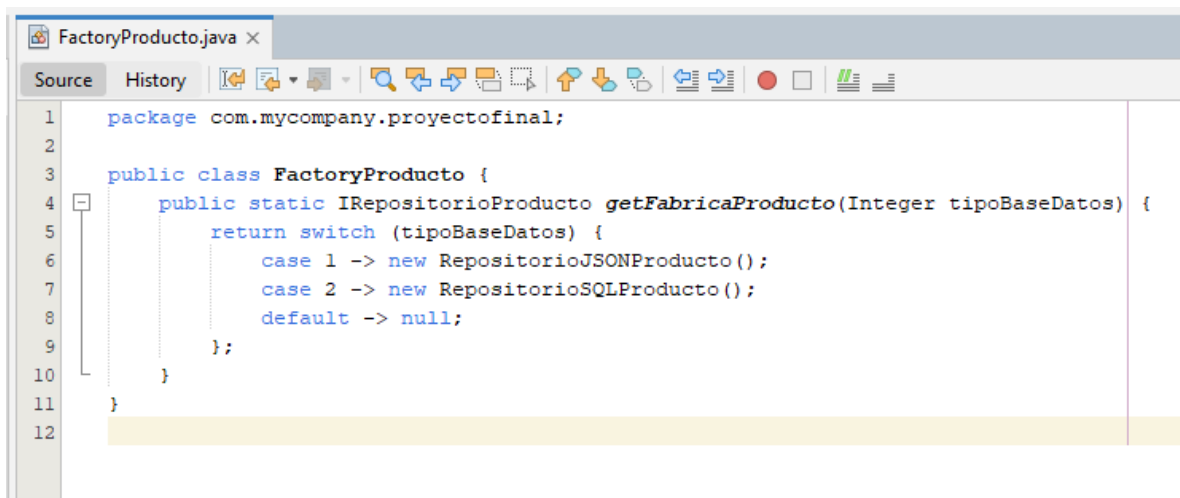
Utilizamos el Patrón Factory Method para Cliente:



```
1 package com.mycompany.proyectoFinal;
2
3 public class FactoryCliente {
4     public static IRepositoryCliente getFabricaCliente(Integer tipoBaseDatos) {
5         return switch (tipoBaseDatos) {
6             case 1 -> new RepositorioJSONCliente();
7             case 2 -> new RepositorioSQLCliente();
8             default -> null;
9         };
10    }
11 }
12
```

Este patrón de diseño nos sirve para crear la fábrica de instancias para guardar, mostrar ya sea en Mysql o Json, en donde seleccionaremos que necesitamos.

Utilizamos el Patrón Factory Method para Producto:



```
1 package com.mycompany.proyectofinal;
2
3 public class FactoryProducto {
4     public static IRepositoryProducto getFabricaProducto(Integer tipoBaseDatos) {
5         return switch (tipoBaseDatos) {
6             case 1 -> new RepositorioJSONProducto();
7             case 2 -> new RepositorioSQLProducto();
8             default -> null;
9         };
10    }
11 }
12
```

Este patrón de diseño nos sirve para crear la fábrica de instancias para guardar, mostrar ya sea en Mysql o Json, en donde seleccionaremos que necesitamos.

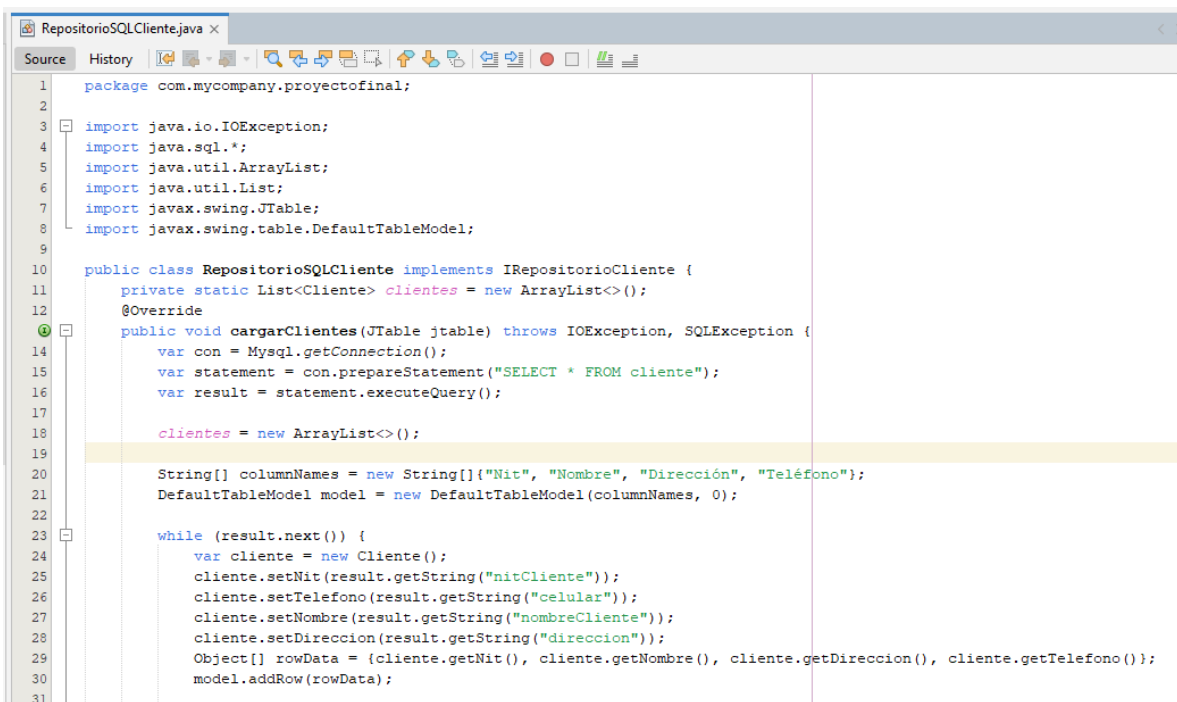
Patrón de diseño Principio de Segregación de Interfaces para clientes:

```
IRepositorioCliente.java x
Source History
1 package com.mycompany.proyectofinal;
2
3 import java.io.IOException;
4 import java.sql.SQLException;
5 import java.util.List;
6 import javax.swing.JTable;
7
8 public interface IRepositorioCliente {
9     void cargarClientes(JTable table) throws IOException, SQLException;
10    void guardarClientes() throws IOException;
11
12    void agregarCliente(Cliente cliente, Integer productoId) throws SQLException, IOException;
13
14    List<Cliente> getClientes();
15 }
16
```

Patrón de diseño Principio de Segregación de Interfaces para producto:

```
IRepositorioProducto.java x
Source History
1 package com.mycompany.proyectofinal;
2
3 import java.io.IOException;
4 import java.sql.SQLException;
5 import java.util.List;
6 import javax.swing.JTable;
7
8 public interface IRepositorioProducto {
9     void cargarProductos(JTable jTable) throws IOException, SQLException;
10    void guardarProductos() throws IOException;
11    void agregarProducto(Producto producto) throws SQLException, IOException;
12    Producto despacharProducto(int idProducto, int cantidadRequerida) throws SQLException, IOException;
13    List<Producto> getProductos();
14    Producto getProducto(int idProducto) throws SQLException, IOException;
15 }
16
```

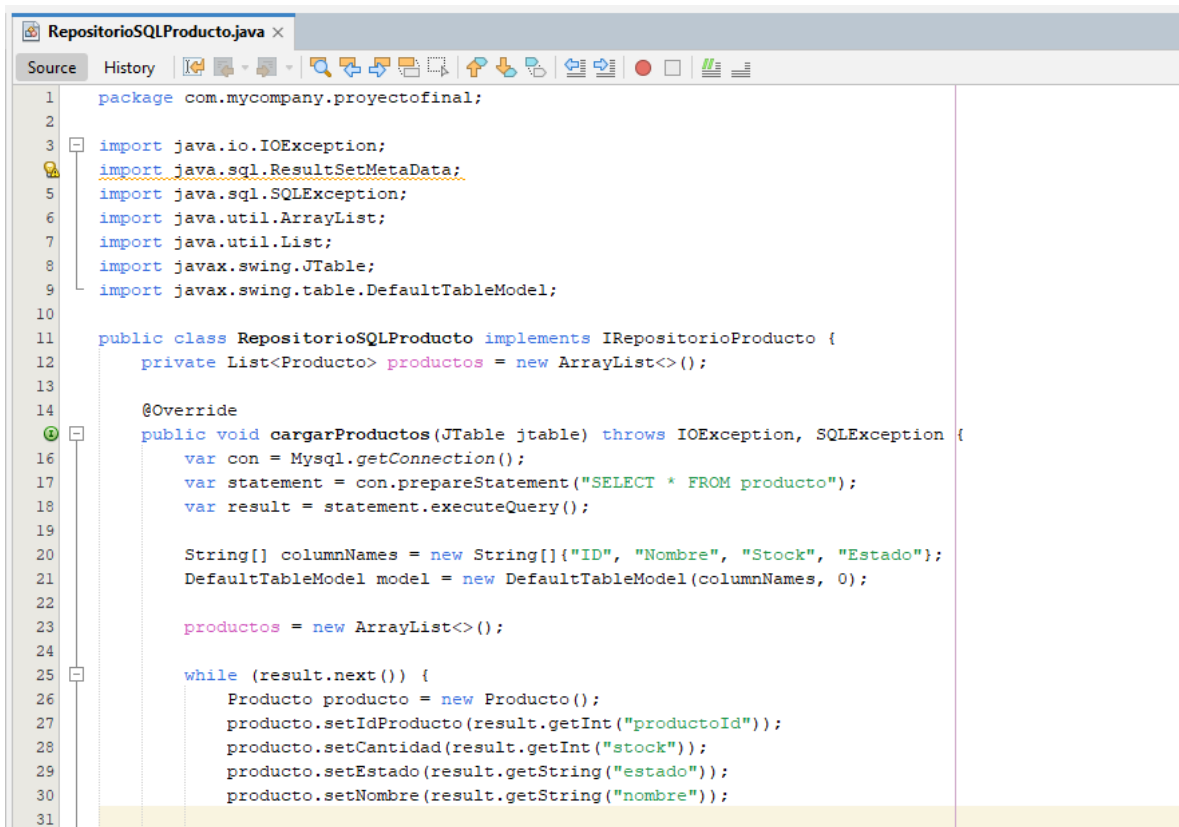

Creación de interfaz según base de datos:



```
1 package com.mycompany.proyectofinal;
2
3 import java.io.IOException;
4 import java.sql.*;
5 import java.util.ArrayList;
6 import java.util.List;
7 import javax.swing.JTable;
8 import javax.swing.table.DefaultTableModel;
9
10 public class RepositorioSQLCliente implements IRepositoryCliente {
11     private static List<Cliente> clientes = new ArrayList<>();
12     @Override
13     public void cargarClientes(JTable jTable) throws IOException, SQLException {
14         var con = Mysql.getConnection();
15         var statement = con.prepareStatement("SELECT * FROM cliente");
16         var result = statement.executeQuery();
17
18         clientes = new ArrayList<>();
19
20         String[] columnNames = new String[]{"Nit", "Nombre", "Dirección", "Teléfono"};
21         DefaultTableModel model = new DefaultTableModel(columnNames, 0);
22
23         while (result.next()) {
24             var cliente = new Cliente();
25             cliente.setNit(result.getString("nitCliente"));
26             cliente.setTelefono(result.getString("celular"));
27             cliente.setNombre(result.getString("nombreCliente"));
28             cliente.setDireccion(result.getString("direccion"));
29             Object[] rowData = {cliente.getNit(), cliente.getNombre(), cliente.getDireccion(), cliente.getTelefono()};
30             model.addRow(rowData);
31         }
32     }
33 }
```

Es la implementación de la interfaz “IRepositorioCliente” para nuestra base de datos
MYSQL

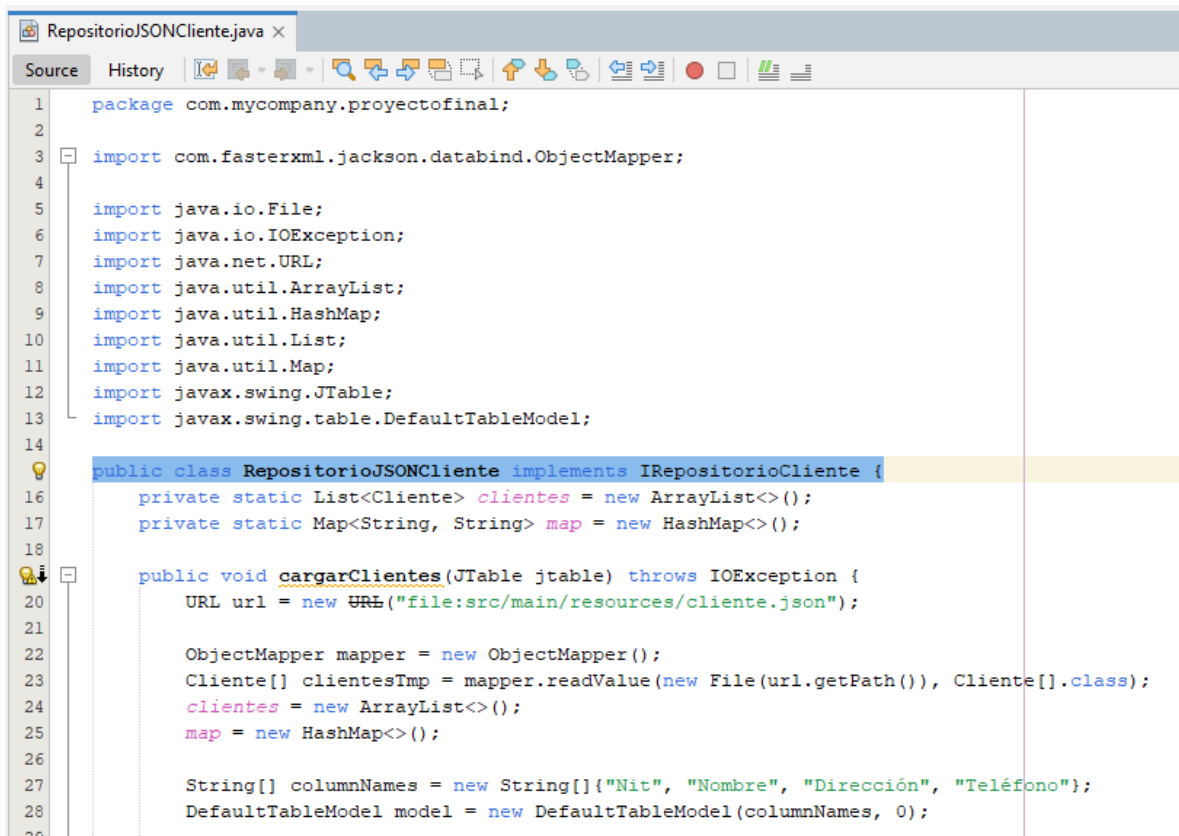
Creación de interfaz según base de datos:



```
1 package com.mycompany.proyectofinal;
2
3 import java.io.IOException;
4 import java.sql.ResultSetMetaData;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.List;
8 import javax.swing.JTable;
9 import javax.swing.table.DefaultTableModel;
10
11 public class RepositorioSQLProducto implements IRepositoryProducto {
12     private List<Producto> productos = new ArrayList<>();
13
14     @Override
15     public void cargarProductos(JTable jTable) throws IOException, SQLException {
16         var con = Mysql.getConnection();
17         var statement = con.prepareStatement("SELECT * FROM producto");
18         var result = statement.executeQuery();
19
20         String[] columnNames = new String[]{"ID", "Nombre", "Stock", "Estado"};
21         DefaultTableModel model = new DefaultTableModel(columnNames, 0);
22
23         productos = new ArrayList<>();
24
25         while (result.next()) {
26             Producto producto = new Producto();
27             producto.setIdProducto(result.getInt("productoId"));
28             producto.setCantidad(result.getInt("stock"));
29             producto.setEstado(result.getString("estado"));
30             producto.setNombre(result.getString("nombre"));
31         }
32     }
33 }
```

Es la implementación de la interfaz “IRepositorioProducto” para nuestra base de datos
MYSQL

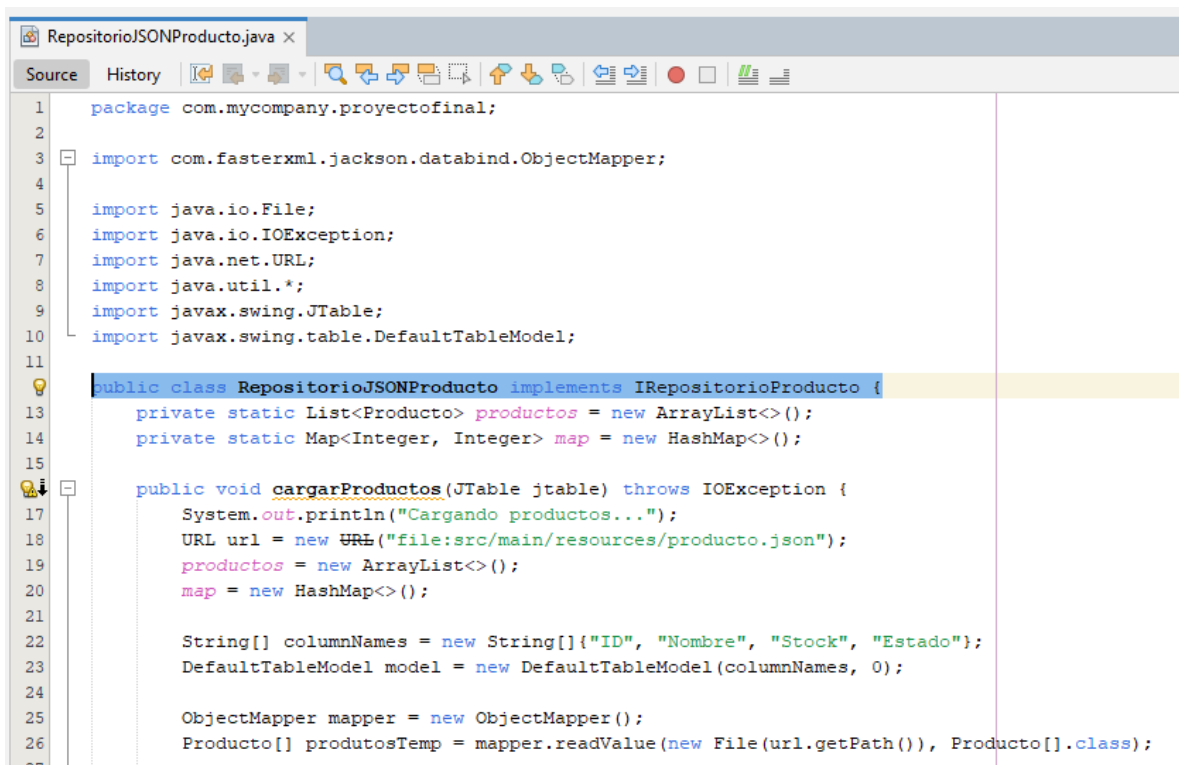
Creación de interfaz según base de datos:



```
1 package com.mycompany.proyectofinal;
2
3 import com.fasterxml.jackson.databind.ObjectMapper;
4
5 import java.io.File;
6 import java.io.IOException;
7 import java.net.URL;
8 import java.util.ArrayList;
9 import java.util.HashMap;
10 import java.util.List;
11 import java.util.Map;
12 import javax.swing.JTable;
13 import javax.swing.table.DefaultTableModel;
14
15 public class RepositorioJSONCliente implements IRepositoryCliente {
16     private static List<Cliente> clientes = new ArrayList<>();
17     private static Map<String, String> map = new HashMap<>();
18
19     public void cargarClientes(JTable jtable) throws IOException {
20         URL url = new URL("file:src/main/resources/cliente.json");
21
22         ObjectMapper mapper = new ObjectMapper();
23         Cliente[] clientesTmp = mapper.readValue(new File(url.getPath()), Cliente[].class);
24         clientes = new ArrayList<>();
25         map = new HashMap<>();
26
27         String[] columnNames = new String[]{"Nit", "Nombre", "Dirección", "Teléfono"};
28         DefaultTableModel model = new DefaultTableModel(columnNames, 0);
29     }
30 }
```

Es la implementación de la interfaz “IRepositoryCliente” para nuestra base de datos en JSON

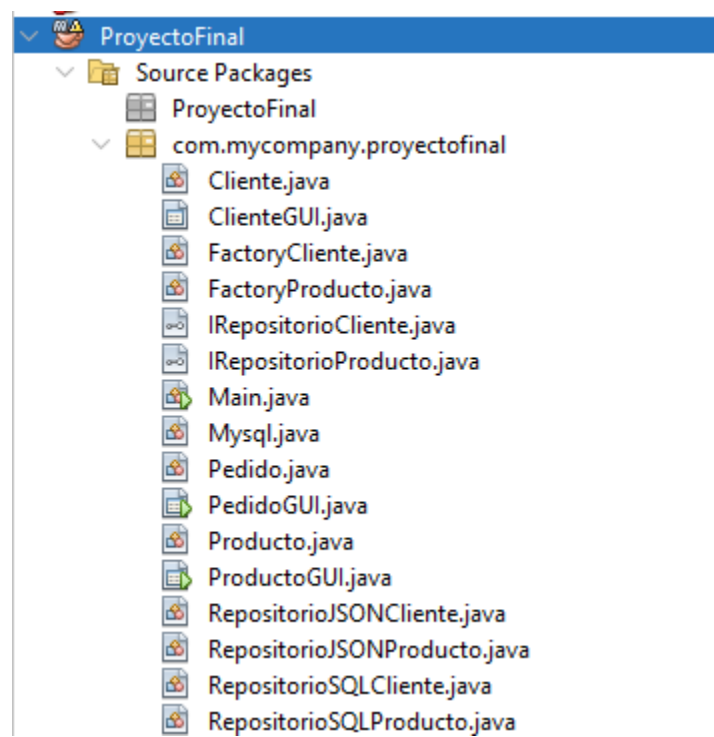
Creación de interfaz según base de datos:



```
1 package com.mycompany.proyectofinal;
2
3 import com.fasterxml.jackson.databind.ObjectMapper;
4
5 import java.io.File;
6 import java.io.IOException;
7 import java.net.URL;
8 import java.util.*;
9 import javax.swing.JTable;
10 import javax.swing.table.DefaultTableModel;
11
12 public class RepositorioJSONProducto implements IRepositoryProducto {
13     private static List<Producto> productos = new ArrayList<>();
14     private static Map<Integer, Integer> map = new HashMap<>();
15
16     public void cargarProductos(JTable jtable) throws IOException {
17         System.out.println("Cargando productos...");
18         URL url = new URL("file:src/main/resources/producto.json");
19         productos = new ArrayList<>();
20         map = new HashMap<>();
21
22         String[] columnNames = new String[]{"ID", "Nombre", "Stock", "Estado"};
23         DefaultTableModel model = new DefaultTableModel(columnNames, 0);
24
25         ObjectMapper mapper = new ObjectMapper();
26         Producto[] productosTemp = mapper.readValue(new File(url.getPath()), Producto[].class);
27     }
28 }
```

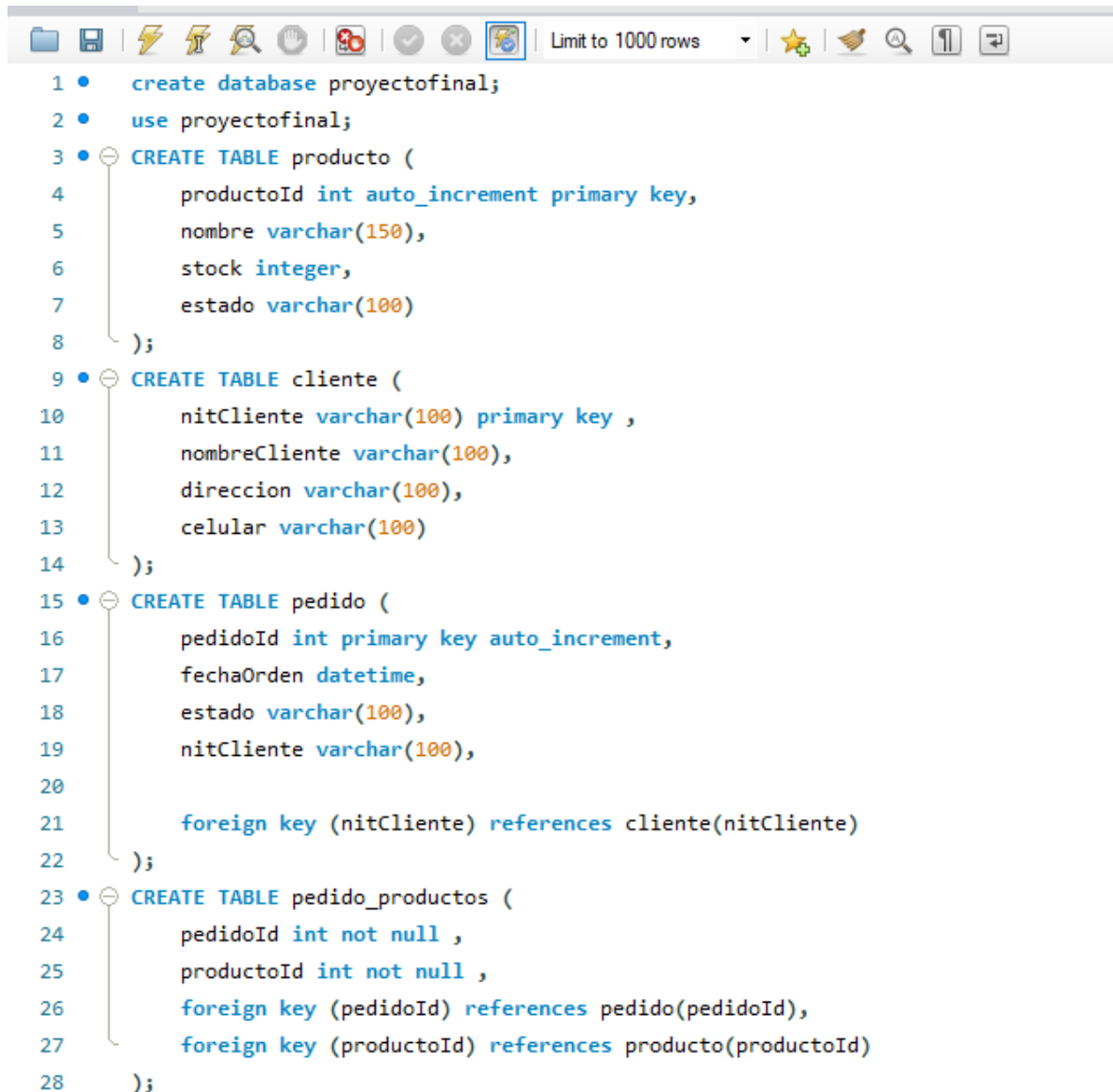
Es la implementación de la interfaz “IRepositorioProducto” para nuestra base de datos en JSON

Tenemos el Patrón de Única Responsabilidad:



Se utiliza el patrón de única responsabilidad separando la lógica por módulos

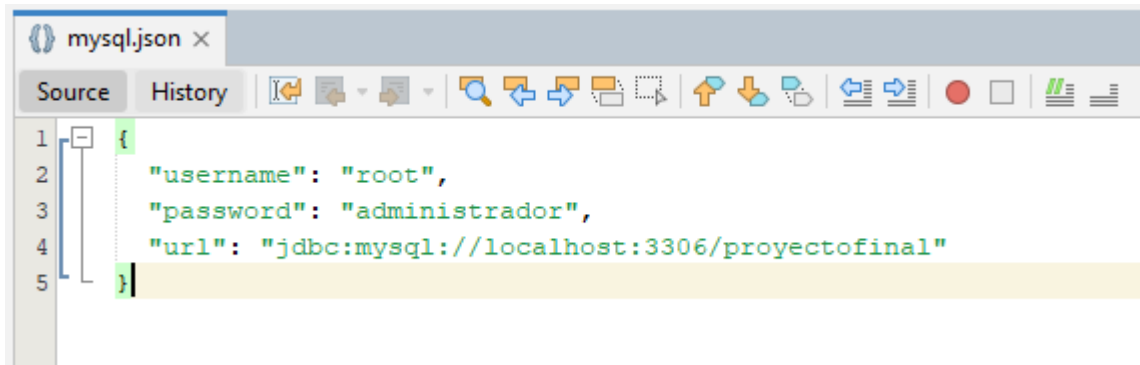
Nuestra Estructura de Base de datos en MYSQL



The image shows a screenshot of a MySQL IDE window. The title bar includes icons for file operations, execution, and search, along with a dropdown menu set to "Limit to 1000 rows". The main area displays SQL code for creating a database and three tables. The code is as follows:

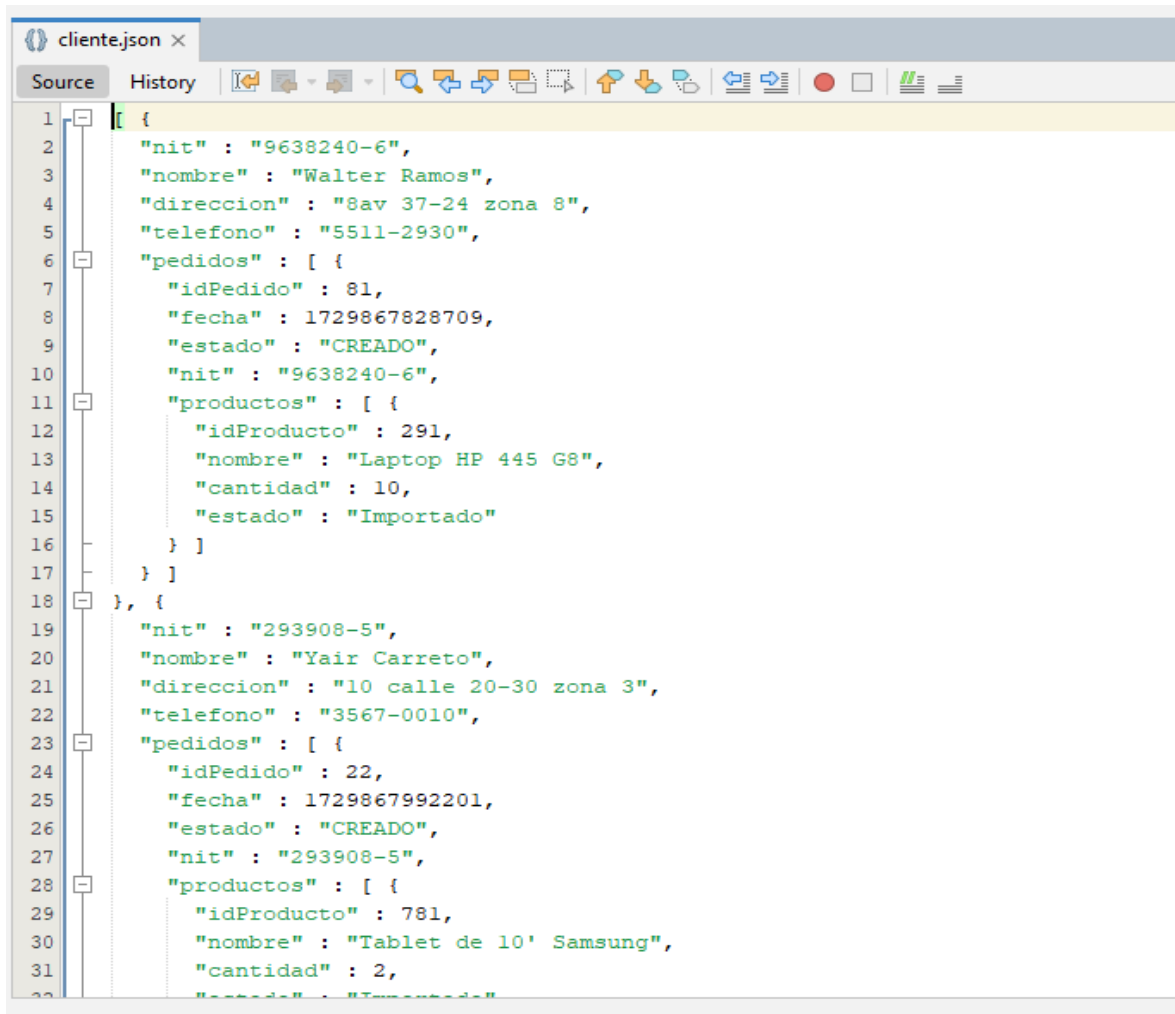
```
1 • create database proyectofinal;
2 • use proyectofinal;
3 • CREATE TABLE producto (
4     productoId int auto_increment primary key,
5     nombre varchar(150),
6     stock integer,
7     estado varchar(100)
8 );
9 • CREATE TABLE cliente (
10     nitCliente varchar(100) primary key ,
11     nombreCliente varchar(100),
12     direccion varchar(100),
13     celular varchar(100)
14 );
15 • CREATE TABLE pedido (
16     pedidoId int primary key auto_increment,
17     fechaOrden datetime,
18     estado varchar(100),
19     nitCliente varchar(100),
20
21     foreign key (nitCliente) references cliente(nitCliente)
22 );
23 • CREATE TABLE pedido_productos (
24     pedidoId int not null ,
25     productoId int not null ,
26     foreign key (pedidoId) references pedido(pedidoId),
27     foreign key (productoId) references producto(productoId)
28 );
```

Tenemos nuestro mysql.json, donde manda a llamar a nuestro usuario y contraseña:



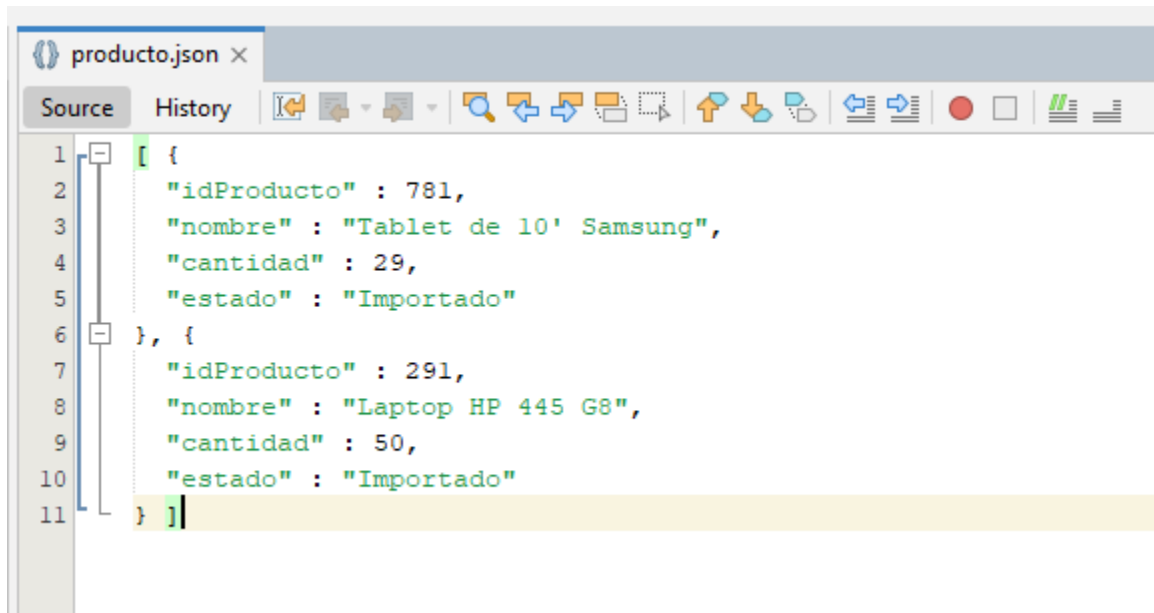
```
1 {  
2   "username": "root",  
3   "password": "administrador",  
4   "url": "jdbc:mysql://localhost:3306/proyectofinal"  
5 }
```

Tenemos nuestro cliente.json en donde se aguarda los datos de nuestros clientes en JSON.



```
1 [ {  
2   "nit" : "9638240-6",  
3   "nombre" : "Walter Ramos",  
4   "direccion" : "8av 37-24 zona 8",  
5   "telefono" : "5511-2930",  
6   "pedidos" : [ {  
7     "idPedido" : 81,  
8     "fecha" : 1729867828709,  
9     "estado" : "CREADO",  
10    "nit" : "9638240-6",  
11    "productos" : [ {  
12      "idProducto" : 291,  
13      "nombre" : "Laptop HP 445 G8",  
14      "cantidad" : 10,  
15      "estado" : "Importado"  
16    } ]  
17  } ]  
18 }, {  
19   "nit" : "293908-5",  
20   "nombre" : "Yair Carreto",  
21   "direccion" : "10 calle 20-30 zona 3",  
22   "telefono" : "3567-0010",  
23   "pedidos" : [ {  
24     "idPedido" : 22,  
25     "fecha" : 1729867992201,  
26     "estado" : "CREADO",  
27     "nit" : "293908-5",  
28     "productos" : [ {  
29       "idProducto" : 781,  
30       "nombre" : "Tablet de 10' Samsung",  
31       "cantidad" : 2,  
32       "estado" : "Importado"
```

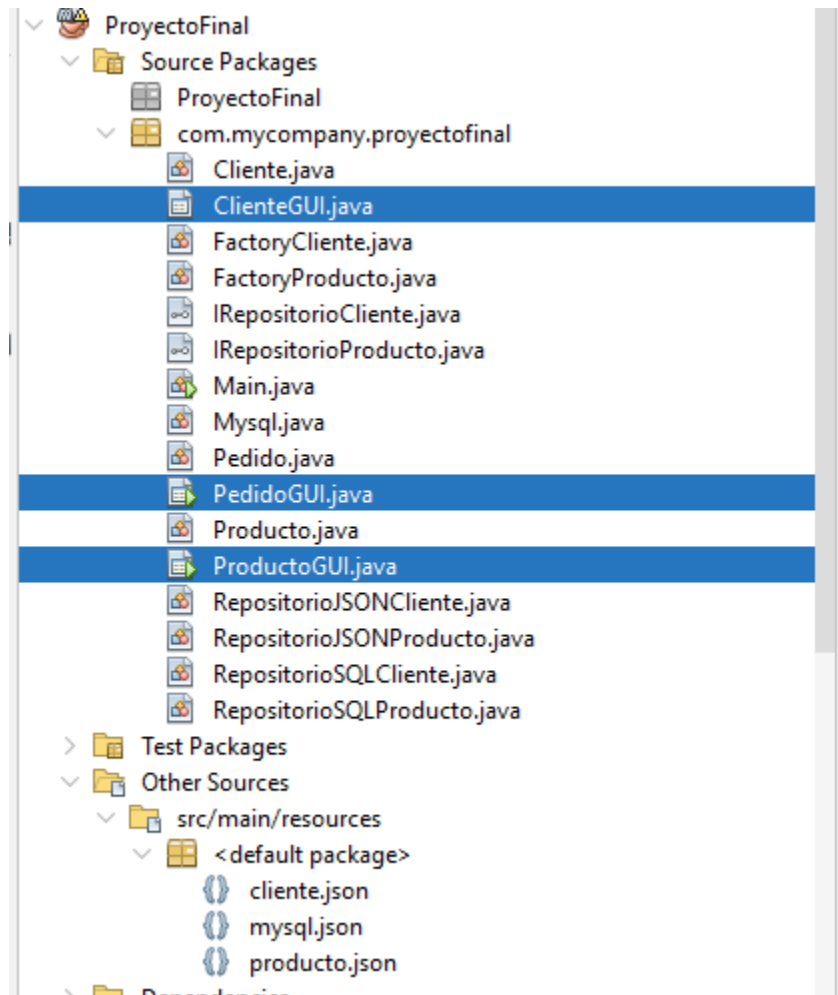
Tenemos nuestro producto.json en donde se guardara todos los datos de nuestros productos:



The image shows a code editor window with a tab labeled 'producto.json'. The editor has a 'Source' tab selected and a toolbar with various icons. The code is a JSON array containing two objects, each representing a product. The first object has an 'idProducto' of 781, a 'nombre' of 'Tablet de 10' Samsung', a 'cantidad' of 29, and an 'estado' of 'Importado'. The second object has an 'idProducto' of 291, a 'nombre' of 'Laptop HP 445 G8', a 'cantidad' of 50, and an 'estado' of 'Importado'. The code is syntax-highlighted, and the array is enclosed in square brackets.

```
1 [ {  
2   "idProducto" : 781,  
3   "nombre" : "Tablet de 10' Samsung",  
4   "cantidad" : 29,  
5   "estado" : "Importado"  
6 }, {  
7   "idProducto" : 291,  
8   "nombre" : "Laptop HP 445 G8",  
9   "cantidad" : 50,  
10  "estado" : "Importado"  
11 } ]
```


Tenemos nuestras clases donde se utilizó JAVA SWING



En nuestro JAVA SWING nos permite hacer el menú de nuestro cliente, producto y pedido, al igual realizar cambios en el mismo, como cambio de color, letra y diseño:

Nuestro CleinteGUI.java

ClienteGUI.java ×

Source Design History

The Preview Design button (in the toolbar) enables you to test the design of the form.

Nombre: **Nit:**

Telefono: **Dirección:**

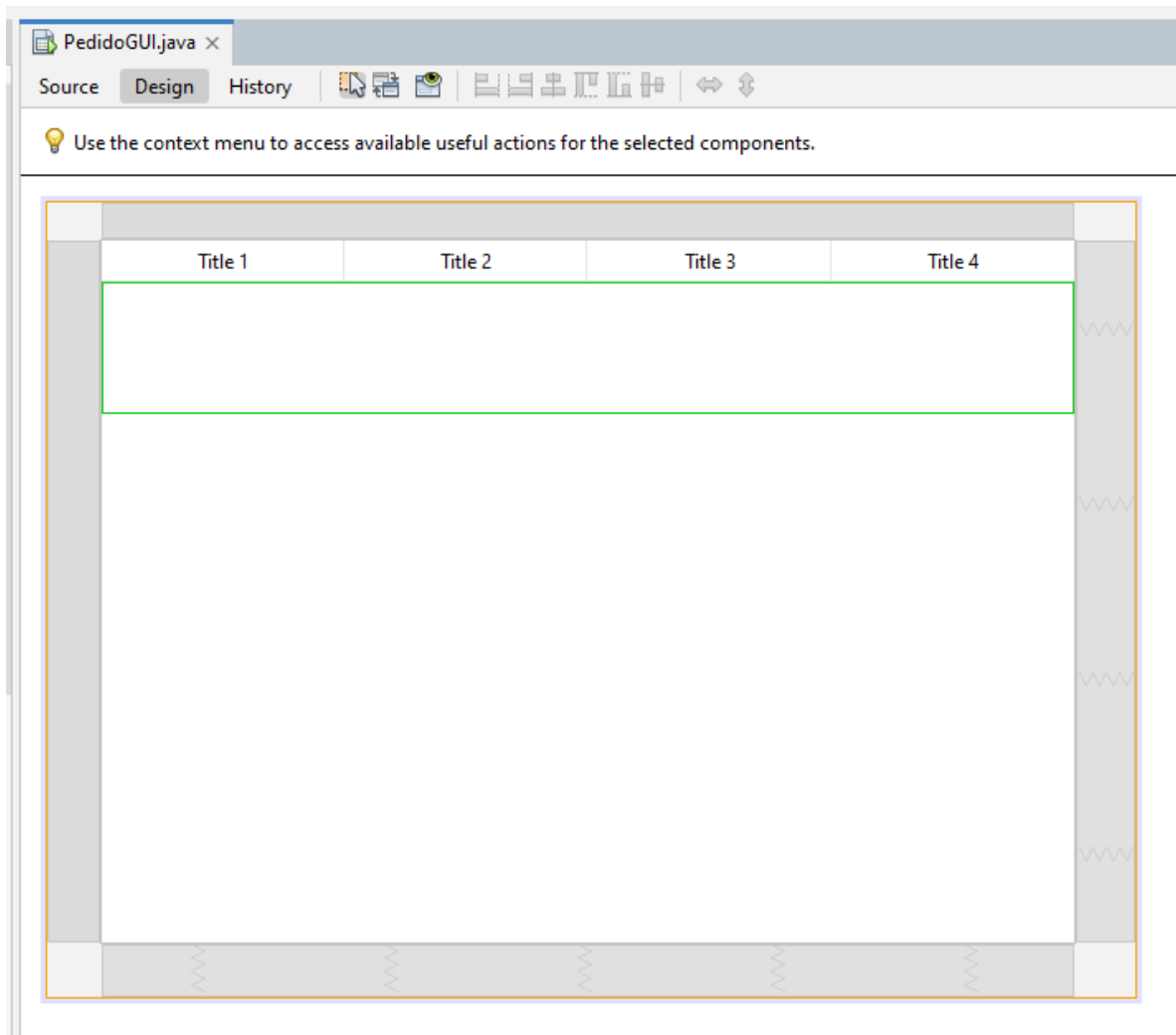
| Title 1 | Title 2 | Title 3 | Title 4 |
|---------|---------|---------|---------|
| | | | |
| | | | |

| Title 1 | Title 2 | Title 3 | Title 4 |
|---------|---------|---------|---------|
| | | | |
| | | | |

ID Productos: **Ver pedidos** **Crear Productos**

Cantidad productos: **JSON** **Guardar**

Nuestro PedidoGUI.java



Nuestro ProductoGUI.java

ProductoGUI.java ×

Source Design History

Select the root node in Navigator to access various useful settings of the form (in Properties).

| | | | | |
|--|------------------|--|----------------------|--|
| | Nombre: | | <input type="text"/> | |
| | Cantidad: | | <input type="text"/> | |
| | Estado: | | <input type="text"/> | |

| | | | |
|---------|---------|---------|---------|
| Title 1 | Title 2 | Title 3 | Title 4 |
|---------|---------|---------|---------|

Guardar