

UNIVERSIDAD MARIANO GÁLVEZ

INGENIERÍA EN SISTEMAS

SECCIÓN A

PROGRAMACIÓN II

PROYECTO II

MELANY VERONICA GOMEZ RODRIGUEZ

7690-23-1041

VENTANAS

- Orden de Compra y Venta

Nombre

Nit

Telefono

Direccion

productoid	nombre	stock	estado
1	k	1	k
2	cafe	1	local
3	leche	7	local

ID Productos:

Cantidad productos:

nitCliente	nombreCliente	direccion	celular
11	mela	zona	564
123	edy	petapa	890

En nuestra ventana de compra y venta podemos agregar a nuestro cliente con su nit, numero telefónico y dirección y tenemos la opción de elegir nuestro producto a través de un ID y podemos elegir la cantidad que deseamos.

- Producción e Importación

Producto

Nombre

Cantidad

Estado

productoid	nombre	stock	estado
1	k	1	k
2	cafe	1	local
3	leche	7	local

En esta ventana podemos añadir nuestros productos tenemos la opción de agregar la cantidad y el estado ya sea local o importado.

- Facturación

Pedidoid	Estado	Fecha Orden
1	GENERADO	2024-10-24
2	GENERADO	2024-10-24
3	GENERADO	2024-10-24
4	GENERADO	2024-10-24

En esta ventana tenemos la opción de ver nuestras ordones, viendo el numero de pedido, el estado y la fecha de la orden.

DIAGRAMA ENTIDAD-RELACION

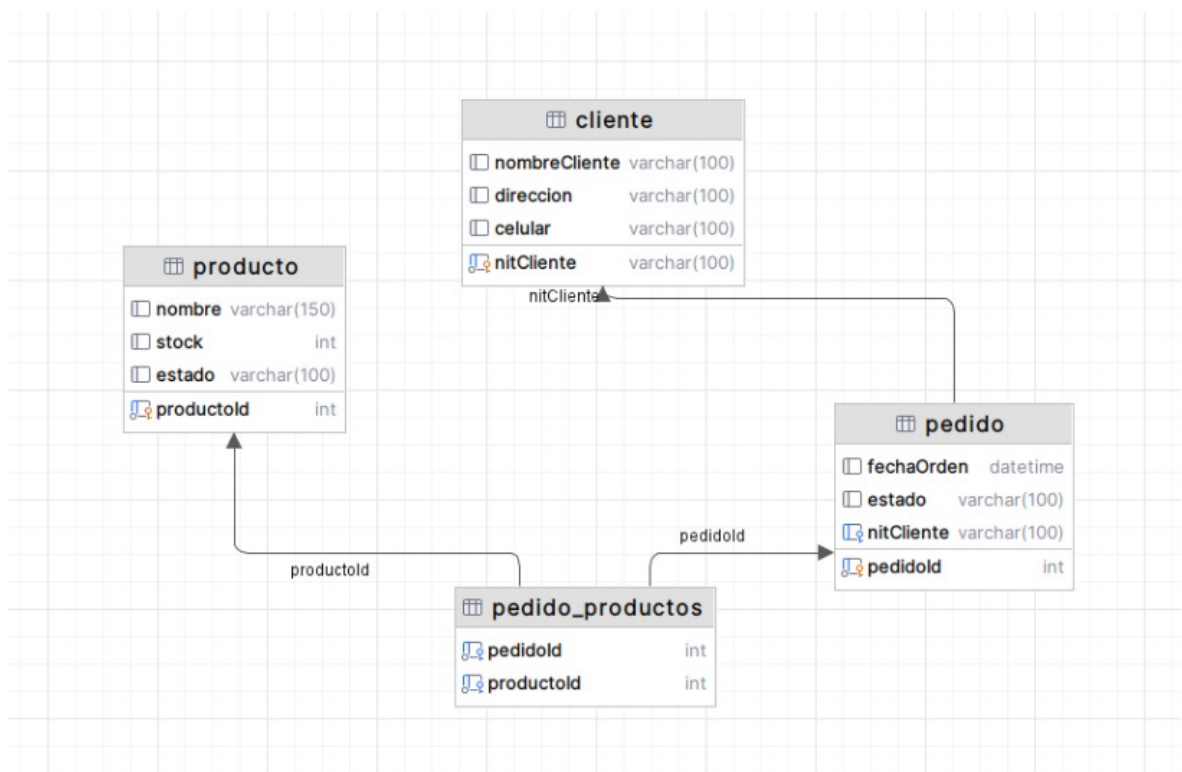
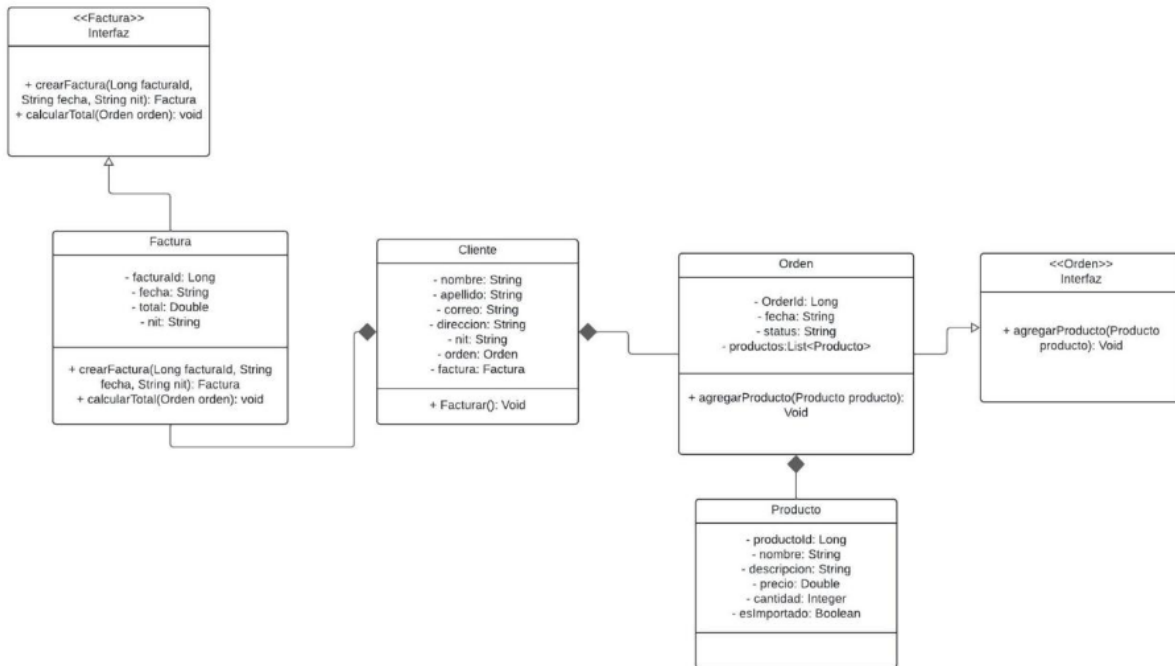
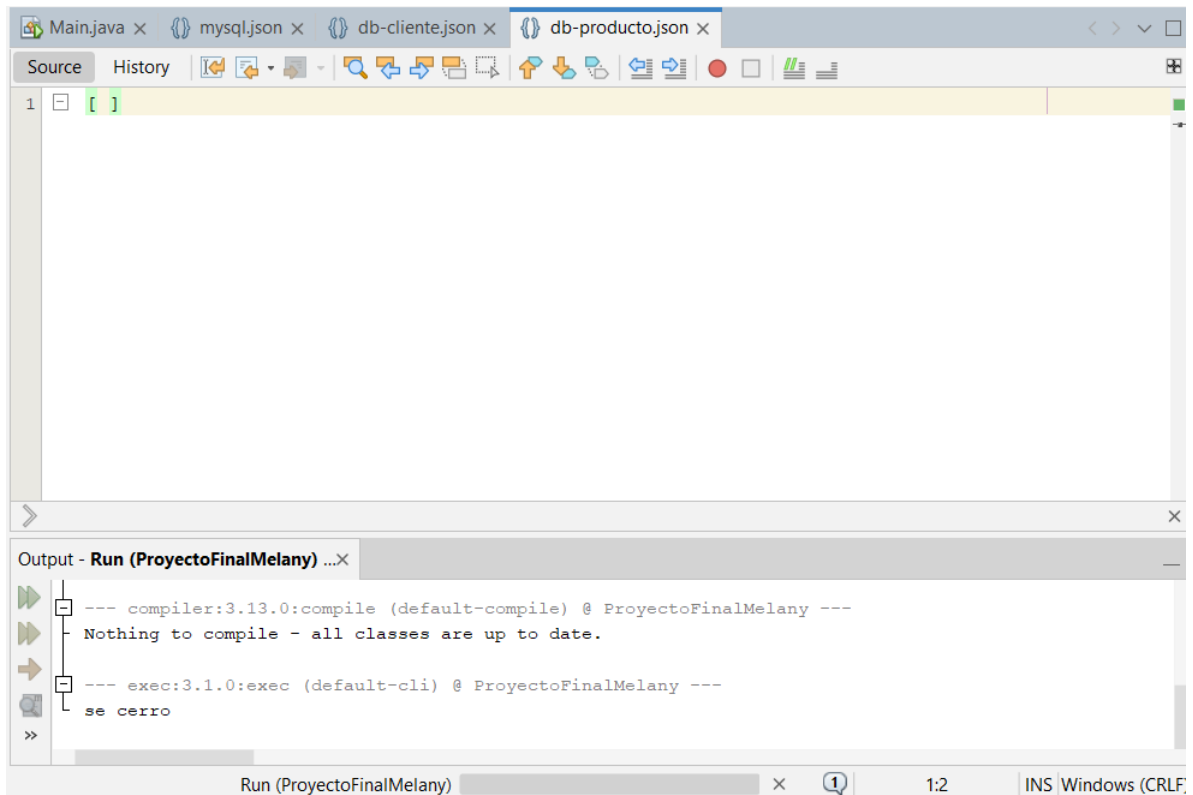


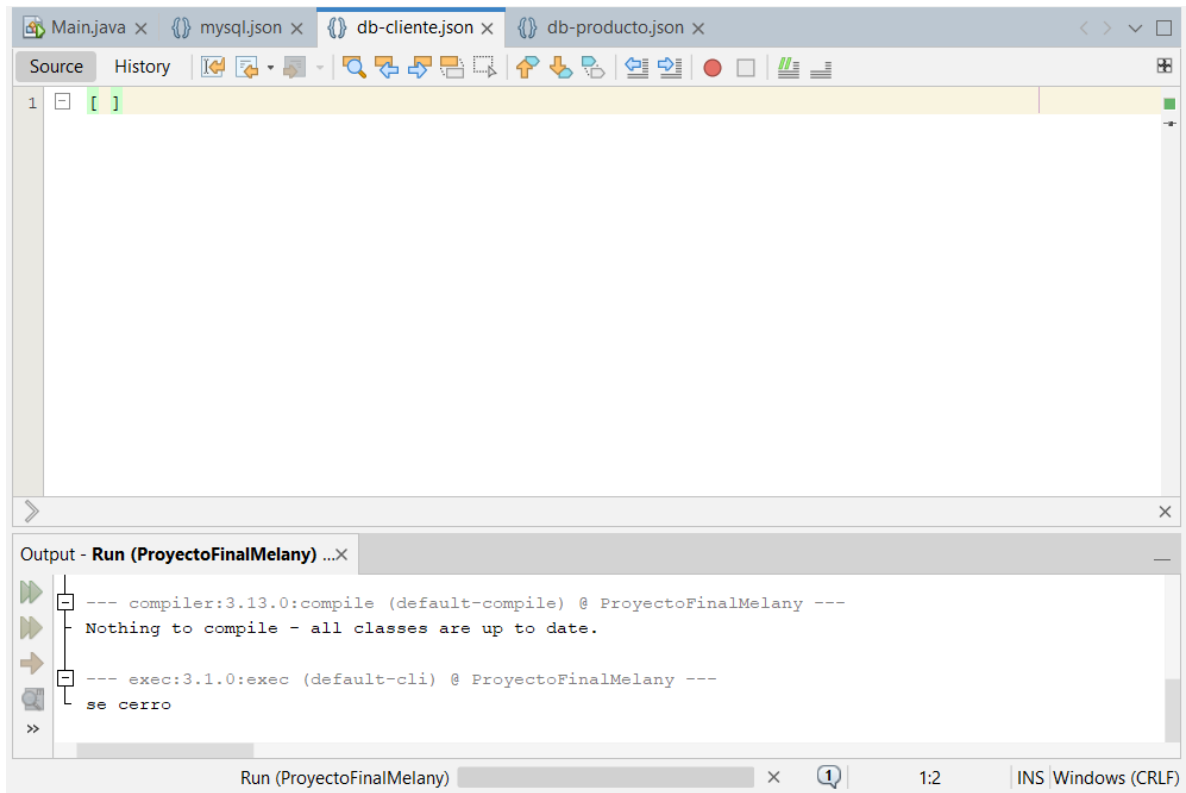
DIAGRAMA DE CLASES



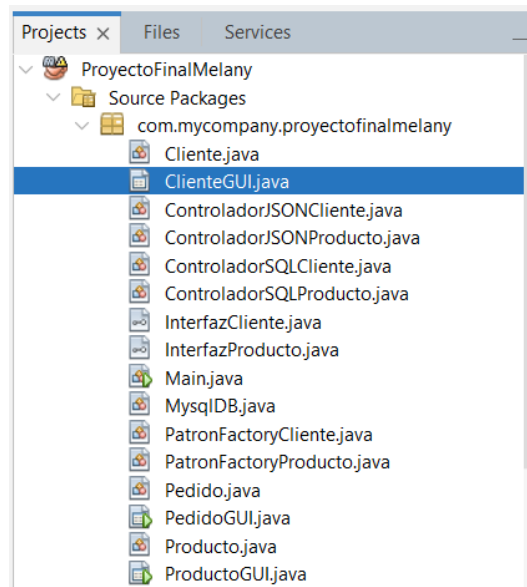
BASE DE DATOS EN JSON DE PRODUCTO



BASE DE DATOS EN JSON DE CLIENTE

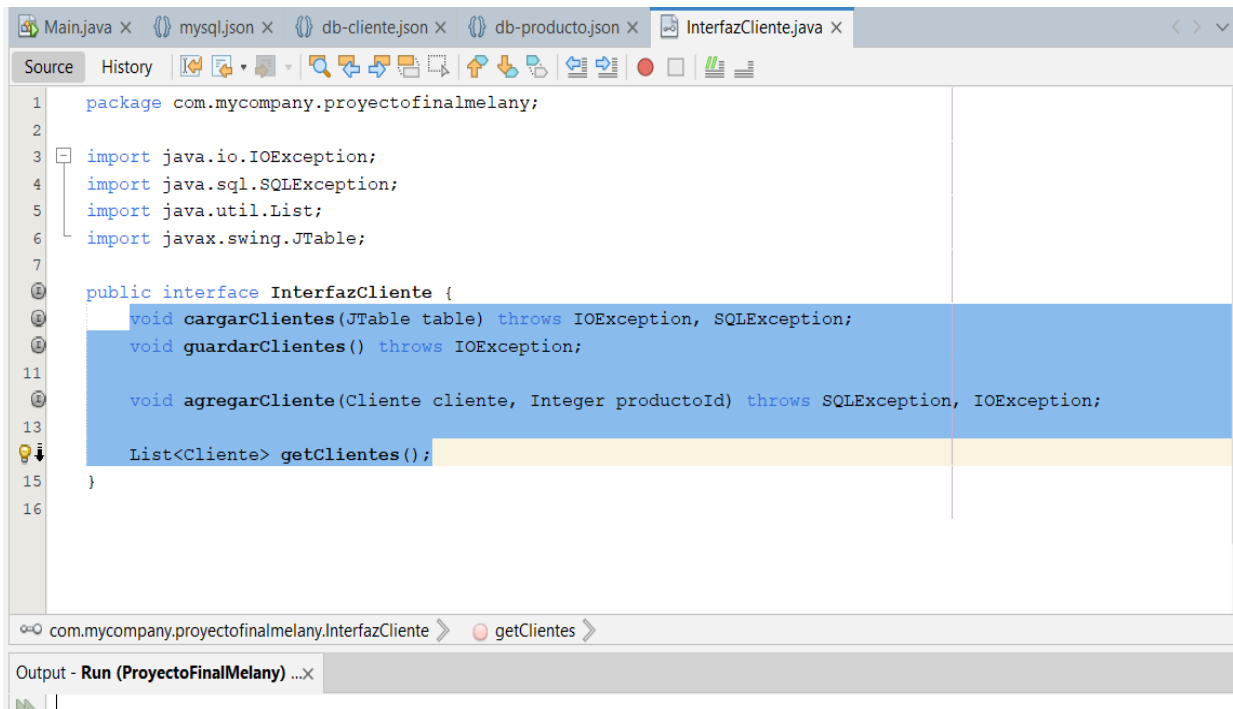


PATRÓN DE DISEÑO ÚNICA RESPONSABILIDAD



En el patrón única responsabilidad creamos archivos solo con la lógica necesaria y no mezclamos lógica entre si

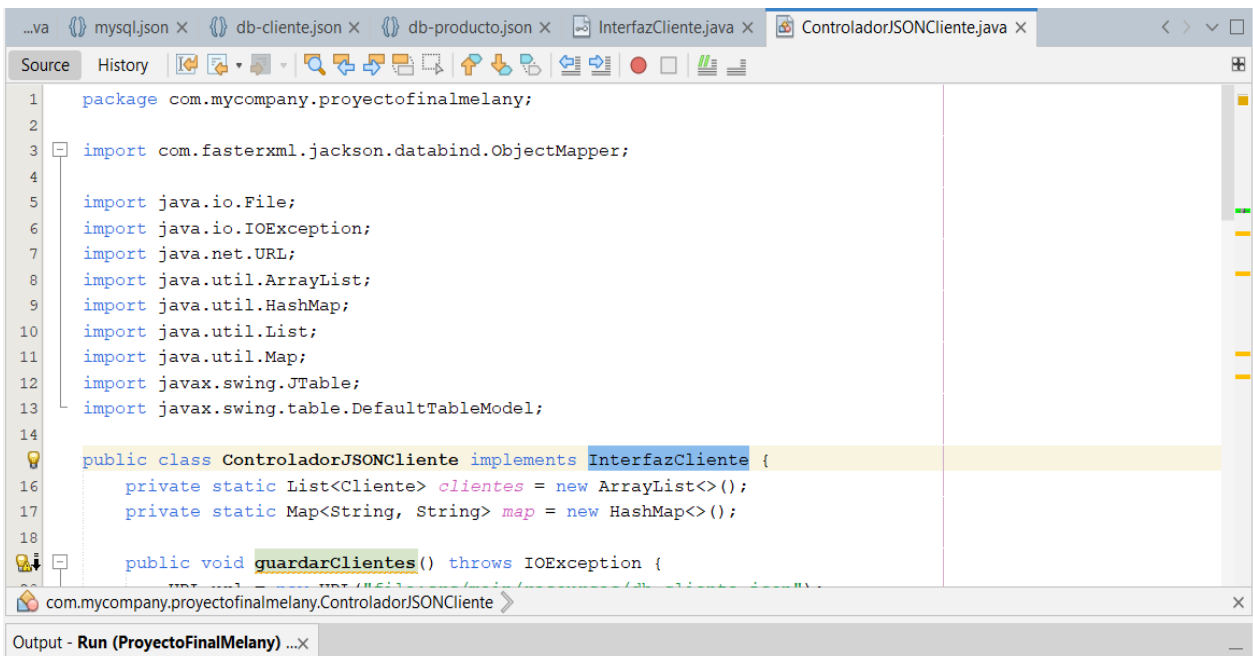
PATRÓN DE SEGREGACIÓN DE INTERFAZ



```
1 package com.myccompany.proyectofinalmelany;
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 InterfazCliente {
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
```

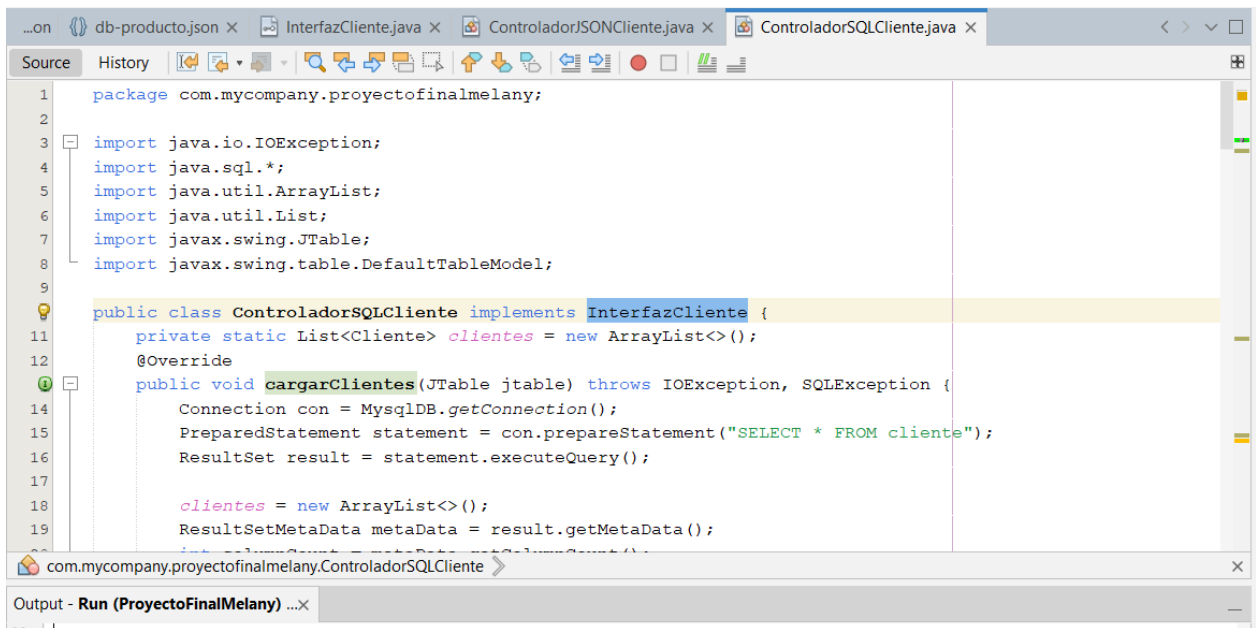
Se crea una interfaz para hacer nuevas clases que la implementen y sean compatibles entre sí.

Se crea una clase de json a partir de la interfaz cliente



```
1 package com.myccompany.proyectofinalmelany;
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 ControladorJSONCliente implements InterfazCliente {
16     private static List<Cliente> clientes = new ArrayList<>();
17     private static Map<String, String> map = new HashMap<>();
18
19     public void guardarClientes() throws IOException {
20         // ...
21     }
22 }
```

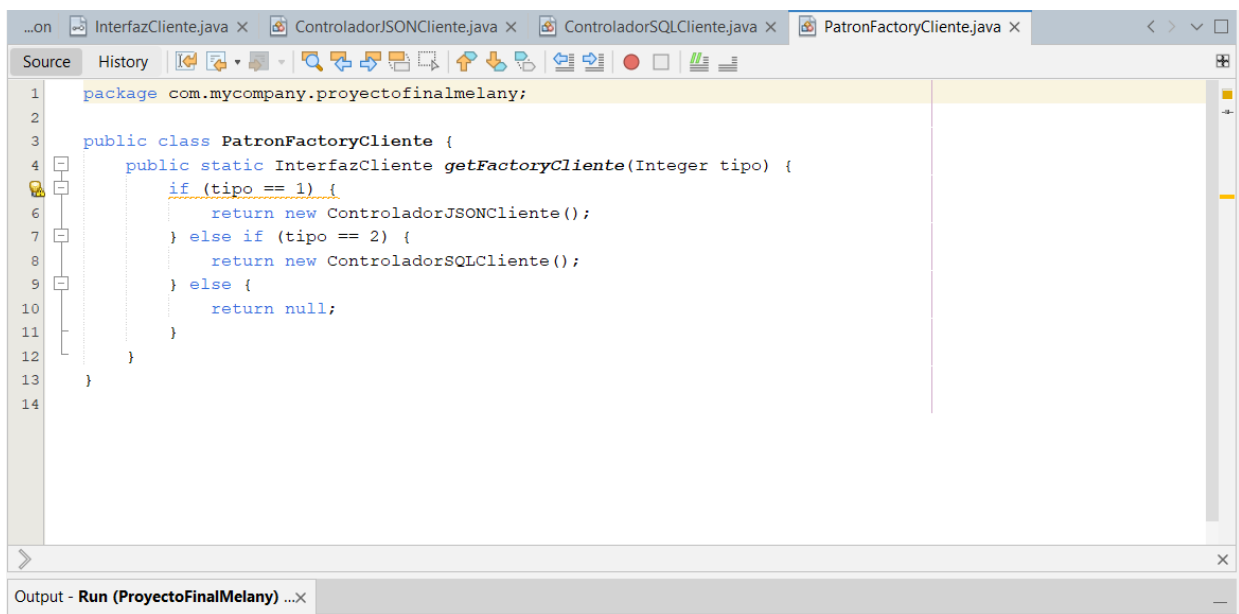
Se crea una clase de MYSQL apartir de la interfaz cliente



The screenshot shows an IDE with several tabs open: db-producto.json, InterfazCliente.java, ControladorJSONCliente.java, and ControladorSQLCliente.java. The active tab is ControladorSQLCliente.java, which contains the following code:

```
1 package com.mycompany.proyectofinalmelany;
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 ControladorSQLCliente implements InterfazCliente {
11     private static List<Cliente> clientes = new ArrayList<>();
12     @Override
13     public void cargarClientes(JTable jTable) throws IOException, SQLException {
14         Connection con = MysqlDB.getConnection();
15         PreparedStatement statement = con.prepareStatement("SELECT * FROM cliente");
16         ResultSet result = statement.executeQuery();
17
18         clientes = new ArrayList<>();
19         ResultSetMetaData metaData = result.getMetaData();
20         int columnCount = metaData.getColumnCount();
21         while (result.next()) {
22             Cliente cliente = new Cliente();
23             for (int i = 1; i <= columnCount; i++) {
24                 cliente.setNombre(result.getString(i));
25             }
26             clientes.add(cliente);
27         }
28         jTable.setModel(new DefaultTableModel(clientes, new String[]{"Nombre"}));
29     }
30 }
```

PATRON FACTORY METHOD



The screenshot shows an IDE with several tabs open: InterfazCliente.java, ControladorJSONCliente.java, ControladorSQLCliente.java, and PatronFactoryCliente.java. The active tab is PatronFactoryCliente.java, which contains the following code:

```
1 package com.mycompany.proyectofinalmelany;
2
3 public class PatronFactoryCliente {
4     public static InterfazCliente getFactoryCliente(Integer tipo) {
5         if (tipo == 1) {
6             return new ControladorJSONCliente();
7         } else if (tipo == 2) {
8             return new ControladorSQLCliente();
9         } else {
10             return null;
11         }
12     }
13 }
14
```

Se crea una fábrica de clientes ya sea de JSON o MYSQL

ESTRUCTURA DE BASE DE DATOS

