

# **Lenguaje de Programación Java**

## **– Trabajo Práctico de Desarrollo**

*Comisión N°: 3E02.*

*Integrantes:*

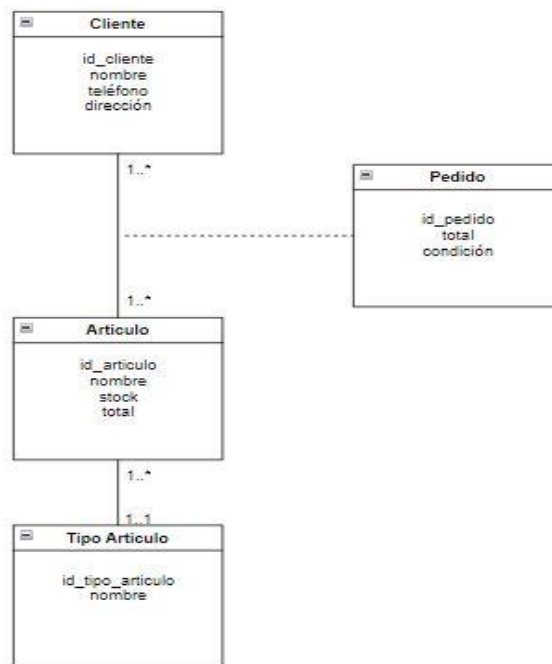
46449 - Bolatti, Martín Francisco - martinbolatti14@gmail.com.

46158 - Santagati, Juan Pablo - juanpiisantagati@gmail.com.

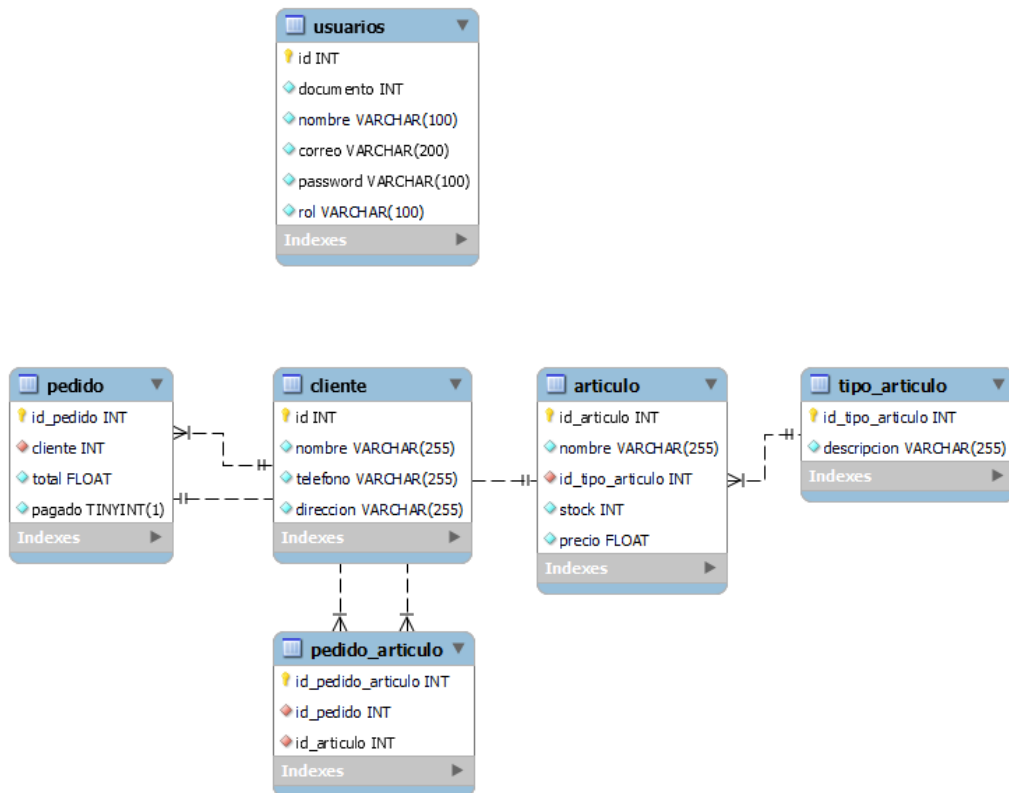
# Índice

<b>Modelo de Dominio</b> .....	2
<b>Modelo de Datos</b> .....	3
<b>Casos de Uso</b> .....	3
Caso de Uso Resumen Re-Estructurado .....	3
Caso de Uso Usuarios .....	3
<b>Captura de Pantalla</b> .....	4
Pantalla de Inicio .....	4
Sobre Nosotros. ....	5
Login. ....	5
Pantalla Principal “Distribuidora” .....	6
Pantalla “Clientes” .....	6
Pantalla “Articulos” .....	7
Pantalla “Pedidos” .....	7
Pantalla “Tipo Artículo” .....	8
Pantalla “Ventas” .....	8
<b>Fragmentos de código</b> .....	9
Controlador Pedido .....	9
ClienteDAO .....	17
ArticuloDAO .....	18
ArticulopedidoDAO .....	19
PedidoDAO .....	19

## ***Modelo de Dominio***



# Modelo de Datos



## Casos de Uso

### Caso de Uso Resumen Re-Estructurado

Nombre: CUR15 - Realizar Venta.

Camino Básico:

1. El vendedor busca al cliente en el sistema invocando **CUU04 - Buscar cliente**.
2. El vendedor ingresa en el sistema los artículos solicitados por el usuario con sus respectivas cantidades invocando **CUU12 - Buscar Artículo**.
3. Una vez finalizada la carga de los productos, el vendedor registra el pedido invocando **CUU13 - Registrar Pedido**.
4. Cuando el cliente realiza el pago del pedido, el vendedor registra el pago invocando **CUU14- Registrar pago**.

### Caso de Uso Usuarios

Nombre: CUU04 - Buscar cliente

Camino Básico:

1. El vendedor ingresa el id del cliente en el sistema.

2. El sistema muestra por pantalla los datos del cliente.

*Nombre* CUU12 - Buscar Artículo.

*Camino Básico:*

1. El vendedor ingresa el id del producto y su respectiva cantidad solicitada por el cliente.
2. El sistema muestra por pantalla los datos del producto.

*Nombre:* CUU13 - Registrar Pedido.

*Camino Básico:*

1. Una vez realizada la carga de el cliente y de todos los articulos, el vendedor realiza la carga del pedido e imprime el mismo, el sistema lo registra.

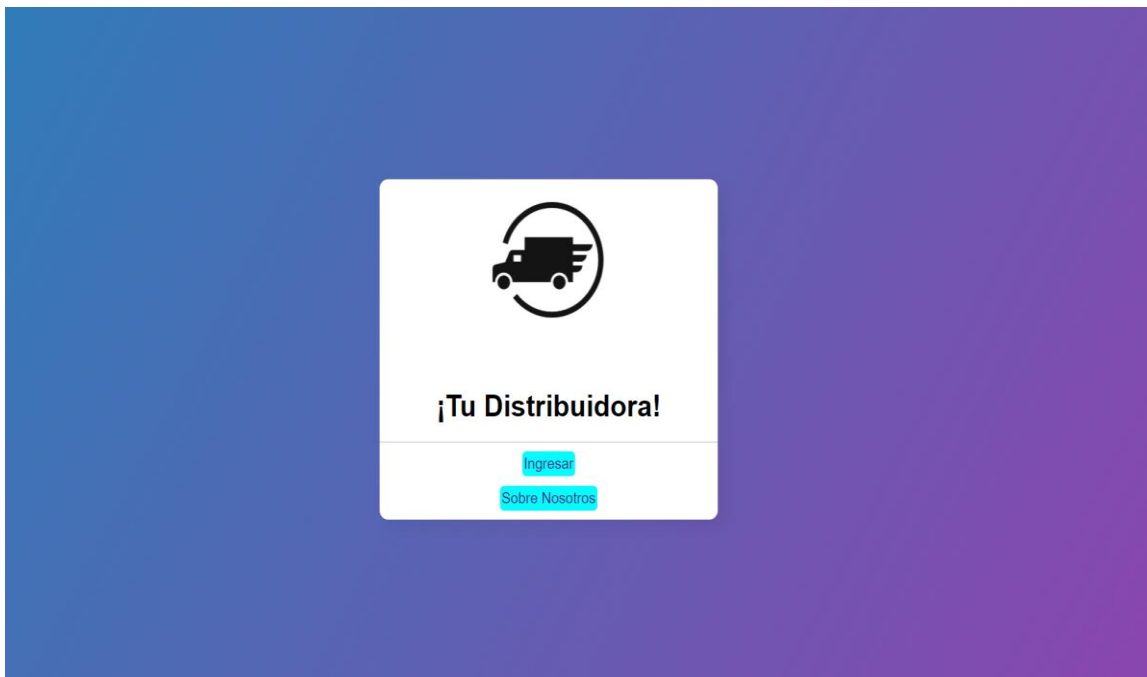
*Nombre:* CUU14 - Registrar Pago.

*Camino Básico:*

1. Cuando el cliente realiza el pago, el vendedor actualiza la condición de pago del pedido y el sistema lo registra.

## ***Captura de Pantalla***

### **Pantalla de Inicio**



## Sobre Nosotros.



### ¿Quienes Somos?

Somos una distribuidora, una pequeña pyme familiar del rubro alimentario, comprometida con la calidad y el servicio excepcional. Ofrecemos soluciones personalizadas para nuestros clientes, con una amplia variedad de productos de primera calidad y entrega oportuna. Somos un equipo de profesionales altamente capacitados, comprometidos con la satisfacción de nuestros clientes y el desarrollo sostenible de nuestra comunidad.

### Nuestra Mision

Nuestra misión es ser un socio estratégico confiable y comprometido, que brinda soluciones integrales de alta calidad y valor agregado a nuestros clientes, al mismo tiempo que contribuimos al bienestar de nuestra comunidad y nuestro planeta. Brindando a nuestros clientes la posibilidad de obtener los productos al mejor precio del mercado, como así también en desarrollar una relación comercial duradera y de calidad con cada una de los mismos.

[Volver al Inicio](#)

## Login.



### ¡Bienvenido!


[¿Olvidaste tu Contraseña?](#)[Ingresar](#)[Volver al Inicio](#)

# Pantalla Principal “Distribuidora”.

Distribuidora

CientesArticulosPedidosTipo ArticuloVentas

Admin



Tu Distribuidora

Somos una empresa dedicada a la distribución de productos de alta calidad en todo el país. Contamos con un amplio catálogo de productos para diferentes sectores, la industria alimentaria es nuestra especialidad.

© 2023 Tu Distribuidora. Todos los derechos reservados.

# Pantalla“Clientes”.


Distribuidora

CientesArticulosPedidosTipo ArticuloVentas

Admin

Cientes

En este panel podras gestionar los datos de los Cientes



Nombre

Telefono

Direccion

Agregar

Actualizar

id	Nombre	Telefono	Direccion	Acciones	
1	Juan Pablo	44444444	Zeballos 1341	Editar	Eliminar
2	Martin Bolatti	55555555	Mitre 1334	Editar	Eliminar

## Pantalla “Articulos”.


Distribuidora

CientesArticulosPedidosTipo ArticuloVentas

Admin

Articulos

En este panel podras Gestionar los Articulos



Nombre

Tipo Articulo

Stock

Precio

Agregar

Actualizar

Filtrar por: Seleccionar una opción

Filtrar

Id	Tipo Articulo	Nombre	Stock	Precio	Acciones	
1	Caramelo	Masticable Misky x 1kg	124	483.32	Editar	Eliminar
2	Chocolate	Dos Corazones x 20u	72	48.23	Editar	Eliminar
3	Chocolate	Marroc x 60u	272	70.02	Editar	Eliminar
4	Galletita	Oreo x 36u	105	280.46	Editar	Eliminar
5	Gaseosa	Coca Cola Botella 500ml Original Pack x 12	184	230.21	Editar	Eliminar
6	Galletita	Vainilla Sweet x18u	70	261.38	Editar	Eliminar
7	Caramelo	Masticable Misky x 1kg	27	712.01	Editar	Eliminar
8	Tostada	Riera Mesa x 18u	30	214.22	Editar	Eliminar

## Pantalla “Pedidos”.

Distribuidora

CientesArticulosPedidosTipo ArticuloVentas

Admin

Datos cliente

Id

BuscarCliente

Nombre

Telefono

Direccion

Datos producto

Id

BuscarProducto


Nombre Articulo

\$ 0000.00

1

AgregarProducto

Actualizar



Codigo	Producto	Precio	Cantidad	Total	Acciones
<div>Generar Venta</div> <div>Nueva Venta</div>				\$ 00.000.00	



# Pantalla “Tipo Artículo”.

Distribuidora

Cientes

Articulos

Pedidos


Tipo Artículo

Ventas

Admin

Tipo Artículo

En este panel podras Gestionar los datos de los Tipos de Articulos



Descripcion

Agregar

Actualizar

Id	Descripcion	Acciones	
1	Caramelo	Editar	Eliminar
2	Galletita	Editar	Eliminar
3	Chocolate	Editar	Eliminar
4	Tostada	Editar	Eliminar
5	Chupaleta	Editar	Eliminar
6	Gaseosa	Editar	Eliminar

# Pantalla “Ventas”.

Distribuidora

Cientes

Articulos

Pedidos


Tipo Artículo

Ventas

Admin

Ventas

En este panel podras Gestionar el pago de las Ventas



Id Pedido

Cliente

Total

Condición de Pago

No Pagado

Actualizar

Id	Cliente	Total	Condición de Pago		
1	Juan Pablo	8553.4	No Pagado	Pago	Eliminar
2	Juan Pablo	8465.92	No Pagado	Pago	Eliminar
3	Martin Bolatti	5526.52	Pagado	Pago	Eliminar
4	Martin Bolatti	3954.64	No Pagado	Pago	Eliminar

# Fragmentos de código

## Controlador Pedido

```
if (menu.equals("Pedidos")) {  
    switch (accion) {  
  
        case "BuscarCliente":  
            try {  
  
                idCliente = Integer.parseInt(request.getParameter("txtidcliente"));  
                cliente = clienteDAO.buscarClienteld(idCliente);  
  
                if(cliente.getNombre().isEmpty()) {  
                    throw new Excepcion("Datos Incorrectos");  
                }  
            } catch (Exception e) {  
                String errorIngreso = e.getMessage();  
                request.setAttribute("errorIngreso", e.getMessage());  
                request.getRequestDispatcher("error.jsp").forward(request, response);  
            }  
            request.setAttribute("lista_articulos", articulos_pedidos);  
            request.setAttribute("totalapagar", total);  
            request.setAttribute("cliente", cliente);  
            break;  
            case "BuscarProducto":  
  
            try {  
  
                idArticulo = Integer.parseInt(request.getParameter("txtidproducto"));  
                articulo = articuloDAO.buscarArticulold(idArticulo);
```

```

if(articulo.getNombre().isEmpty()) {
    throw new Excepcion("Datos Incorrectos");
}

} catch (Exception e) {
String errorIngreso = e.getMessage();
    request.setAttribute("errorIngreso", e.getMessage());
    request.getRequestDispatcher("error.jsp").forward(request, response);
}

request.setAttribute("cliente", cliente);
request.setAttribute("articulo", articulo);
request.setAttribute("lista_articulos", articulos_pedidos);
request.setAttribute("totalapagar", total);
    break;

case "AgregarProducto":
int existente = 0;
try {

    request.setAttribute("cliente", cliente);
    totalpedido = 0;
    articulopedido = new ArticuloPedido();
    request.setAttribute("cliente", cliente);
    request.setAttribute("lista_articulos", articulos_pedidos);
    idArticulo = Integer.parseInt(request.getParameter("txtidproducto"));
    nombre = request.getParameter("txtnombreproducto");
    precio = Double.parseDouble(request.getParameter("txtprecioproducto"));
    cantidad = Double.parseDouble(request.getParameter("cantidadproducto"));

```

```

try {
    if (cantidad > Double.parseDouble(articulo.getStock()) ) {
        throw new Excepcion("Stock Insuficiente");
    }
} catch (Excepcion e) {

    String error = e.getMessage();
    request.setAttribute("error", e.getMessage());
    request.getRequestDispatcher("error.jsp").forward(request, response);
    id--;
    break;
}

id++;
subtotal = precio * cantidad;
articulopedido.setId(id);
articulopedido.setNombre(nombre);
articulopedido.setCantidad(cantidad);
articulopedido.setPrecio(precio);
articulopedido.setSubtotal(subtotal);
articulopedido.setId(idArticulo);
if (articulos_pedidos.size() == 0) {
    articulos_pedidos.add(articulopedido);
} else {
    for (int i= 0; i < articulos_pedidos.size();i++) {
        if (articulos_pedidos.get(i).getNombre().equals(articulopedido.getNombre())) {
            existente = 1;
        }
    }
}

try {

```

```

        if (existente == 1) {
            throw new Excepcion("El Articulo Ya Existe");
        } else {
            articulos_pedidos.add(articulopedido);
        }
    } catch (Excepcion e) {

        String error1 = e.getMessage();
        request.setAttribute("error1", e.getMessage());
        request.getRequestDispatcher("error.jsp").forward(request, response);
    }

}

request.setAttribute("lista_articulos", articulos_pedidos);
for (int i = 0; i < articulos_pedidos.size(); i++) {
    totalpedido += articulos_pedidos.get(i).getSubtotal();
}

NumberFormat formatoNumero1 = NumberFormat.getNumberInstance();
total = formatoNumero1.format(totalpedido);
request.setAttribute("totalapagar", total);

break;

} catch (NumberFormatException e2) {
    String errorMessage2 = e2.getMessage();
    request.setAttribute("errorMessage2", e2.getMessage());
    request.getRequestDispatcher("error.jsp").forward(request, response);
    break;
}

```

```

case "Eliminar":

    int idB = 0;

    idArticulo = Integer.parseInt(request.getParameter("id"));
    articulo = articuloDAO.buscarArticuloId(idArticulo);
    for (int i = 0; i < articulos_pedidos.size(); i++) {
        if (articulos_pedidos.get(i).getId() == articulo.getId()) {
            idB = i;
        }
    }

    articulos_pedidos.remove(idB);
    request.setAttribute("cliente", cliente);
    request.setAttribute("lista_articulos", articulos_pedidos);
    request.setAttribute("totalapagar", total);
    break;

```

```

case "Actualizar":

    try {
        request.setAttribute("cliente", cliente);
        request.setAttribute("articulo", articulo);
        request.setAttribute("lista_articulos", articulos_pedidos);
        ArtículoPedido articuloPedidoU = new ArtículoPedido();
        int idE = 0;
        idArticulo = Integer.parseInt(request.getParameter("txtidproducto"));
        nombre = request.getParameter("txtnombreproducto");
        precio = Double.parseDouble(request.getParameter("txtprecioproducto"));
        cantidad = Double.parseDouble(request.getParameter("cantidadproducto"));
        articulo = articuloDAO.buscarArticuloId(idArticulo);
    }

```

```

        if (cantidad > Double.parseDouble(articulo.getStock())) {

            throw new Excepcion("Stock Insuficiente");

        }
    } catch (Excepcion e) {
        String error = e.getMessage();
        request.setAttribute("error", e.getMessage());
        request.getRequestDispatcher("error.jsp").forward(request, response);
        id--;
    }

    for (int i = 0; i < articulos_pedidos.size(); i++) {
        if (articulos_pedidos.get(i).getId() == articulo.getId()) {
            idE = i;
        }
    }

    id++;

    subtotal = precio * cantidad;
    articuloPedidoU.setId(id);
    articuloPedidoU.setNombre(nombre);
    articuloPedidoU.setCantidad(cantidad);
    articuloPedidoU.setPrecio(precio);
    articuloPedidoU.setSubtotal(subtotal);
    articuloPedidoU.setId(idArticulo);
    articulos_pedidos.set(idE, articuloPedidoU);
    request.setAttribute("totalapagar", total);
    break;
} catch (NumberFormatException e2) {
    String errorMessage2 = e2.getMessage();

```

```

        request.setAttribute("errorMessage2", e2.getMessage());
        request.getRequestDispatcher("error.jsp").forward(request, response);
        break;
    }

    case "Cargar":
        idArticulo = Integer.parseInt(request.getParameter("id"));
        articulo = articuloDAO.buscarArticuloId(idArticulo);
        request.setAttribute("cliente", cliente);
        request.setAttribute("articulo", articulo);
        request.setAttribute("lista_articulos", articulos_pedidos);
        request.setAttribute("totalapagar", total);
    }
    request.getRequestDispatcher("Pedidos.jsp").forward(request, response);
}

if (menu.equals("GenerarVenta")) {
    switch (accion) {
        case "GenerarVenta":
            int idPedido;

            pedidoDAO.AgregarPedido(cliente, totalpedido);
            idPedido = pedidoDAO.DevolverIdPedido();
            articulopedidoDAO.AgregarArticulos(articulos_pedidos, idPedido);
            articuloDAO.ActualizarStock(articulos_pedidos);

            try {
                throw new Excepcion("Venta Generada Correctamente");
            } catch (Excepcion e) {

                String Venta = e.getMessage();

                request.setAttribute("Venta", e.getMessage());

                request.getRequestDispatcher("error.jsp").forward(request, response);
            }
        }
    }
}

```



```

        id--;

    }

    request.setAttribute("cliente", cliente);

    request.setAttribute("articulo", articulo);

    request.setAttribute("lista_articulos", articulos_pedidos);

    break;

case "NuevaVenta":

    articulos_pedidos.removeAll(articulos_pedidos);

    request.getRequestDispatcher("Pedidos.jsp").forward(request, response);

}

request.getRequestDispatcher("Pedidos.jsp").forward(request, response);

}

if (menu.equals("Ventas")) {

    switch (accion) {

        case "Listar":

            List lista= pedidoDAO.Listar();

            request.setAttribute("pedido", lista);


            break;


        case "Cargar":

            idPedido = Integer.parseInt(request.getParameter("id"));

            Pedido pedidoseleccionado = pedidoDAO.buscarPedidoId(idPedido);

            request.setAttribute("pedidoseleccionado", pedidoseleccionado);

            request.getRequestDispatcher("Controlador?menu=Ventas&accion=Listar").forward(request,
response);

            break;

            case "Actualizar":

                pedido = new Pedido();

```

```

String condicion = request.getParameter("txtcondicion");

pedido = pedidoDAO.buscarPedidoId(idPedido);

if (condicion.equals("Pagado")|| condicion.equals("pagado") ) {

    pedido.setCondicion(true);

} else {

    pedido.setCondicion(false);

}

pedidoDAO.Actualizar(pedido);

request.getRequestDispatcher("Controlador?menu=Ventas&accion=Listar").forward(request,
response);

break;

}

request.getRequestDispatcher("Ventas.jsp").forward(request, response);

}

```

## ClienteDAO

```

public Cliente buscarClienteId(int id) {

    Cliente cliente = new Cliente();

    String consulta = "SELECT * FROM cliente WHERE id= ?";

    con = cn.Conexion();

    try {

        ps = con.prepareStatement(consulta);

        ps.setInt(1, id);

        rs = ps.executeQuery();

        while (rs.next()) {

            cliente.setId(rs.getInt("id"));

            cliente.setNombre(rs.getString("nombre"));

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }

    return cliente;

}

```

```

    cliente.setTelefono(rs.getString("telefono"));

    cliente.setDireccion(rs.getString("direccion"));

}

} catch (SQLException ex) {

    Logger.getLogger(UsuarioDAO.class.getName()).log(Level.SEVERE, null, ex);

}

        return cliente;

}

```

## ArticuloDAO

```

public void ActualizarStock(List<ArticuloPedido> articulos_pedidos ) {

    String sentencia = "UPDATE articulo set stock=? WHERE id_articulo=?";

    try {

        con = cn.Conexion();

        ps = con.prepareStatement(sentencia);

        for (ArticuloPedido articulo : articulos_pedidos) {

            Articulo articulo2 = new Articulo();

            int stock = 0;

            articulo2 = buscarArticuloId(articulo.getId());

            stock = Integer.parseInt(articulo2.getStock()) - (articulo.getCantidad().intValue());

            PreparedStatement ps = con.prepareStatement(sentencia);

            ps.setInt(1,stock);

            ps.setInt(2, articulo2.getId());

            ps.executeUpdate();

        }

    } catch (SQLException ex) {

        Logger.getLogger(ArticuloDAO.class.getName()).log(Level.SEVERE, null, ex);

    }
}

```

## ArticuloPedidoDAO

```
public int AgregarArticulos(List<ArticuloPedido> articulos_pedidos, int id) {  
    String sentencia =  
        "INSERT INTO pedido_articulo (id_pedido, id_articulo) VALUES (?,?)"  
        ; try {  
            con = cn.Conexion();  
            ps = con.prepareStatement(sentencia);  
            for (ArticuloPedido articuloPedido : articulos_pedidos) {  
                ps.setInt(1, id);  
                ps.setInt(2, articuloPedido.getId());  
                ps.executeUpdate();  
            }  
        } catch (SQLException ex) {  
            Logger.getLogger(ArticuloPedidoDAO.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    return r;  
}
```

## PedidoDAO

```
public int AgregarPedido(Cliente cliente, double totalpedido) {  
    String sentencia =  
        "INSERT INTO pedido (cliente,total,pagado) VALUES (?,?,?)"  
        ; try {  
            con = cn.Conexion();  
            ps = con.prepareStatement(sentencia);  
            ps.setInt(1, cliente.getId());  
            ps.setDouble(2, totalpedido);  
            ps.executeUpdate();  
        } catch (SQLException ex) {  
            Logger.getLogger(PedidoDAO.class.getName()).log(Level.SEVERE, null, ex);  
        }  
    return r;  
}
```

```

        ps.setDouble(2, totalpedido);

        ps.setBoolean(3, false);

        ps.executeUpdate();

    } catch (SQLException ex) {

        Logger.getLogger(ArticuloDAO.class.getName()).log(Level.SEVERE, null, ex); }

    return r;

}

public List Listar() {

    String consulta = "SELECT * FROM pedido";

    List<Pedido> lista = new ArrayList();

    try {

        con = cn.Conexion();

        ps = con.prepareStatement(consulta);

        rs = ps.executeQuery();

        while (rs.next()) {

            Pedido pedido = new Pedido();

            Cliente c = new Cliente();

            ClienteDAO cDAO = new ClienteDAO();

            pedido.setId_pedido(rs.getInt("id_pedido"));

            c = cDAO.buscarClientId(rs.getInt("cliente"));

            pedido.setCliente(c);

            pedido.setTotal(rs.getDouble("total"));

            pedido.setCondicion(rs.getBoolean("pagado"));

            lista.add(pedido);

        }

    } catch (SQLException ex) {

        Logger.getLogger(ArticuloDAO.class.getName()).log(Level.SEVERE, null,

ex);

    }

    return lista;

```

```
}
```

```
public Pedido buscarPedidold(int id) {  
    con = cn.Conexion();  
    String consulta = "SELECT * FROM pedido WHERE id_pedido=" + id;  
    Pedido pedido = new Pedido();  
  
    try {  
        ps = con.prepareStatement(consulta);  
        rs = ps.executeQuery();  
        while (rs.next()) {  
            Cliente cliente = new Cliente();  
            ClienteDAO clienteDAO = new ClienteDAO();  
            pedido.setId_pedido(rs.getInt("id_pedido"));  
            cliente = clienteDAO.buscarClienteld(rs.getInt("cliente"));  
            pedido.setCliente(cliente);  
            pedido.setTotal(rs.getDouble("total"));  
            pedido.setCondicion(rs.getBoolean("pagado"));  
        }  
        catch (SQLException ex) {  
            Logger.getLogger(PedidoDAO.class.getName()).log(Level.SEVERE, null, ex);  
        }  
  
        return pedido;  
    }  
}
```

```
public void Actualizar(Pedido pedido) {  
    String sentencia = "UPDATE pedido set pagado=? WHERE id_pedido=?";  
    try {  
        con = cn.Conexion();  
        ps = con.prepareStatement(sentencia);  
    }
```

```
ps.setBoolean(1, pedido.isCondicion());  
ps.setInt(2, pedido.getId_pedido());  
  
ps.executeUpdate();  
  
} catch (SQLException ex) {  
    Logger.getLogger(PedidoDAO.class.getName()).log(Level.SEVERE, null, ex);  
}  
}
```