CONSULTAS SQL ASOCIADAS A LA INTERFAZ DEL SISTEMA

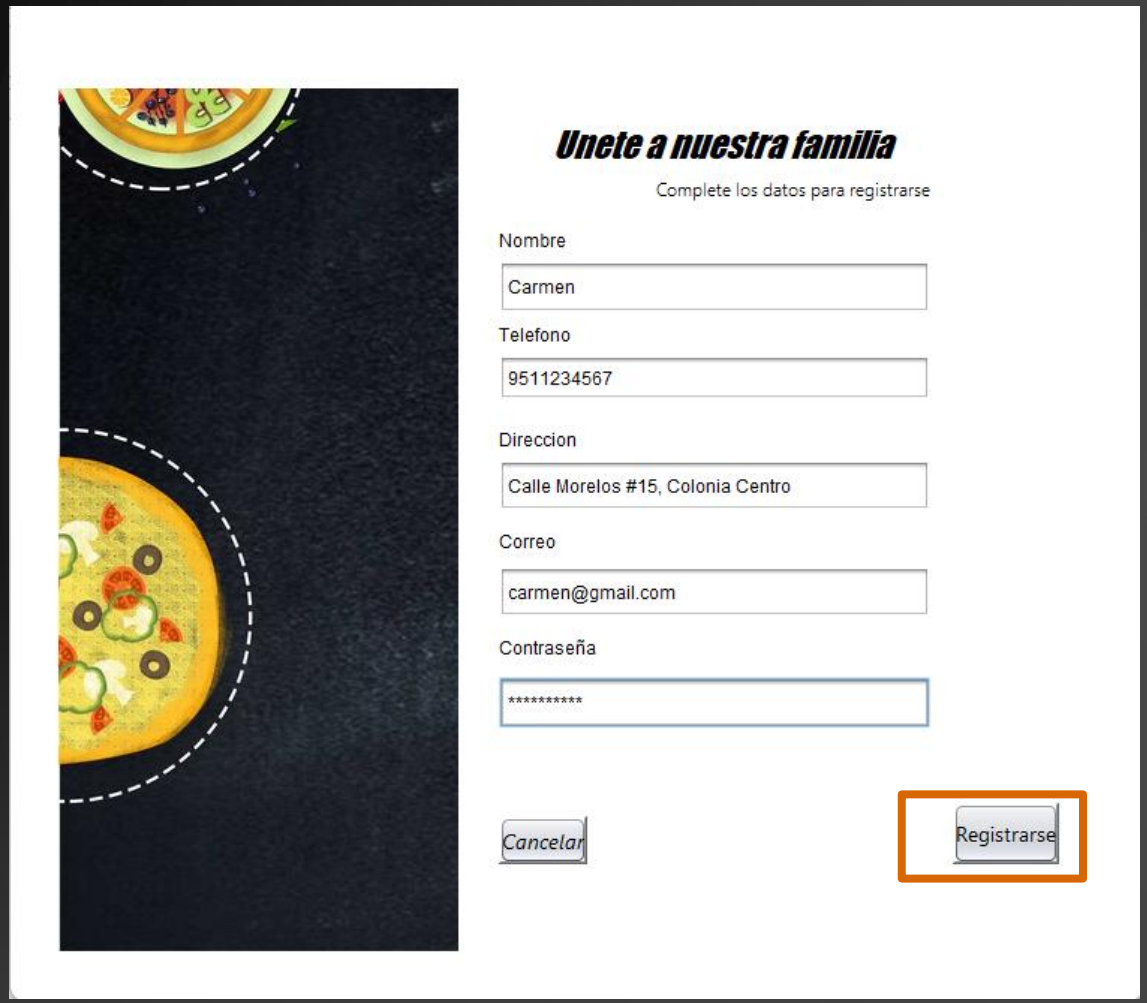
EQUIPO 6:

ANTONIO CONTRERAS ALAN

CRUZ CID DEYANIRA

RAMÍREZ GARCÍA CITLALLI BELEN

BOTÓN DE REGISTRO: **BTNREGISTRAR**



Esta consulta verifica si el correo ingresado ya existe en la base de datos y, si no, inserta un nuevo registro en la tabla Usuario con los datos del formulario.

Funcionamiento:

1. El sistema obtiene los datos ingresados por el usuario (nombre, teléfono, dirección, correo y contraseña).
2. Encripta la contraseña antes de guardarla.
3. Ejecuta una consulta SELECT para comprobar si el correo ya existe.
4. Si no existe, utiliza UsuarioJpaController para insertar el nuevo usuario en la base de datos.
5. Si el registro es exitoso, muestra un mensaje de confirmación y redirige a la ventana de inicio de sesión.

TABLA USUARIO

id_usuario [PK] integer	nombre character varying (100)	correo character varying (100)	contraseña character varying (64)	direccion text	telefono character varying (20)
1	Ana Torres	ana.torres@example.com	h12j34dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79	Calle Luna 12	5551112222
2	Luis Pérez	luis.perez@example.com	hja12sd5d083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79	Av. Sol 45	5553334444
3	Marta Gómez	marta.gomez@example....	8c7f2a1b3d5e0f9c4b6a7d8e9f0c1b2a3d4e5f6c7b8a9d0e1f2c3b4a5d6e7f8c	Calle Nube 8	5555556666
4	Carlos Ruiz	carlos.ruiz@example.com	e9d8c7b6a5f4e3d2c1b0a9f8e7d6c5b4a3f2e1d0c9b8a7f6e5d4c3b2a1f0e9d8	Av. Estrella 99	5557778888
5	Laura Díaz	laura.diaz@example.com	1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b	Calle Cielo 23	5559990000
6	Delivery Central	delivery@example.com	f0e1d2c3b4a5f6e7d8c9b0a1f2e3d4c5b6a7f8e9d0c1b2a3f4e5d6c7b8a9f0e1	Centro	5550001111
10	Belen	ejemplo@gmail.com	b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb...	djklasj	9212880322
11	a	example@gmail.com	a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3	q	1
12	Carmen	carmen@gmail.com	c775e7b757ede630cd0aa1113bd102661ab38829ca52a6422ab782862f2686...	Calle Morelos #15, Colonia Centro	9511234567

CONSULTAS:

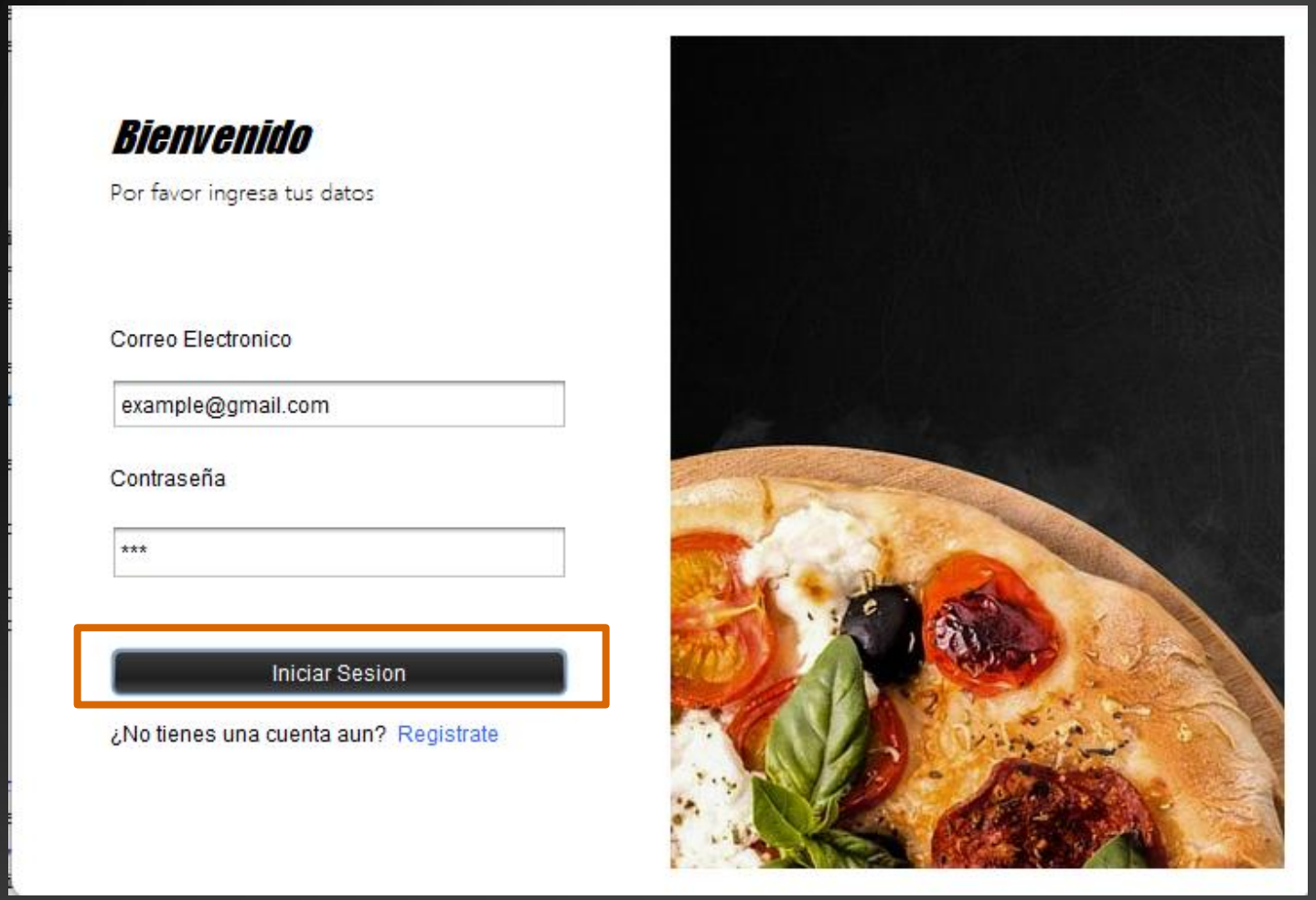
- Verificación del correo:

```
SELECT * FROM Usuario WHERE correo = ?;
```

- Inserción del nuevo usuario (implícita en JPA):

```
INSERT INTO Usuario (nombre, telefono, direccion, correo, contraseña)
VALUES (?, ?, ?, ?, ?);
```

BOTÓN DE INICIO DE SESIÓN: **BTNINICIAR**



Esta consulta busca en la base de datos un usuario cuyo correo y contraseña coincidan con los datos ingresados en la interfaz.

Funcionamiento:

1. El sistema obtiene el correo y la contraseña del formulario.
2. Encripta la contraseña para compararla con la que está guardada en la base de datos.
3. Usa el controlador UsuarioJpaController para recuperar todos los registros de la tabla Usuario.
4. Recorre la lista y verifica si existe un usuario con el mismo correo y contraseña encriptada.
5. Si lo encuentra, permite acceder al menú principal; si no, muestra un mensaje de error.

TABLA USUARIO

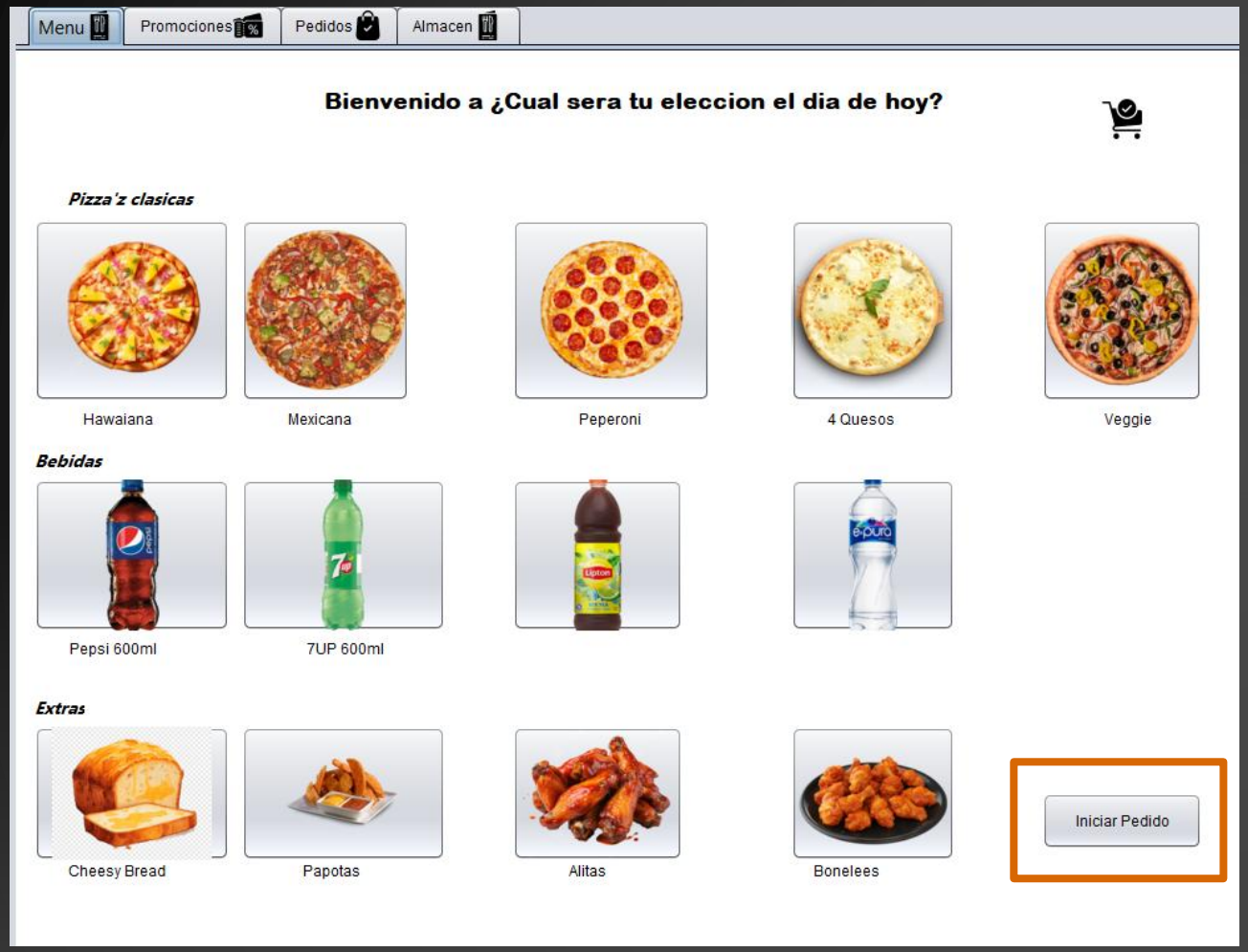
id_usuario [PK] integer	nombre character varying (100)	correo character varying (100)	contraseña character varying (64)	direccion text	telefono character varying (20)
1	Ana Torres	ana.torres@example.com	h12j34dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79	Calle Luna 12	5551112222
2	Luis Pérez	luis.perez@example.com	hja12sd5d083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79	Av. Sol 45	5553334444
3	Marta Gómez	marta.gomez@example....	8c7f2a1b3d5e0f9c4b6a7d8e9f0c1b2a3d4e5f6c7b8a9d0e1f2c3b4a5d6e7f8c	Calle Nube 8	5555556666
4	Carlos Ruiz	carlos.ruiz@example.com	e9d8c7b6a5f4e3d2c1b0a9f8e7d6c5b4a3f2e1d0c9b8a7f6e5d4c3b2a1f0e9d8	Av. Estrella 99	5557778888
5	Laura Díaz	laura.diaz@example.com	1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b	Calle Cielo 23	5559990000
6	Delivery Central	delivery@example.com	f0e1d2c3b4a5f6e7d8c9b0a1f2e3d4c5b6a7f8e9d0c1b2a3f4e5d6c7b8a9f0e1	Centro	5550001111
10	Belen	ejemplo@gmail.com	b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb...	djklasj	9212880322
11	a	example@gmail.com	a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3	q	1
12	Carmen	carmen@gmail.com	c775e7b757ede630cd0aa1113bd102661ab38829ca52a6422ab782862f2686...	Calle Morelos #15, Colonia Centro	9511234567

CONSULTAS:

- Implícita (generada por JPA):

SELECT * FROM Usuario;

BOTON PARA INICIAR PEDIDO: **BTNPEDIDO**



Esta consulta revisa si el usuario actual ya tiene un pedido en estado "Creado" y, si no, inserta un nuevo registro en la tabla Pedidos con el estado inicial "Creado".

Funcionamiento:

1. El sistema obtiene todos los pedidos almacenados mediante PedidosJpaController.
2. Recorre la lista para verificar si el usuario actual ya tiene un pedido en proceso (estado = 'Creado').
3. Si no hay pedidos en curso, crea un nuevo objeto Pedidos con los siguientes datos:
   - ID del usuario actual.
   - Fecha actual.
   - Estado inicial: "Creado".
   - Total: 0.
4. El nuevo pedido se guarda en la base de datos mediante el método create() del controlador JPA.
5. Se muestra un mensaje confirmando que el pedido fue creado exitosamente.

TABLA PEDIDO

id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS:

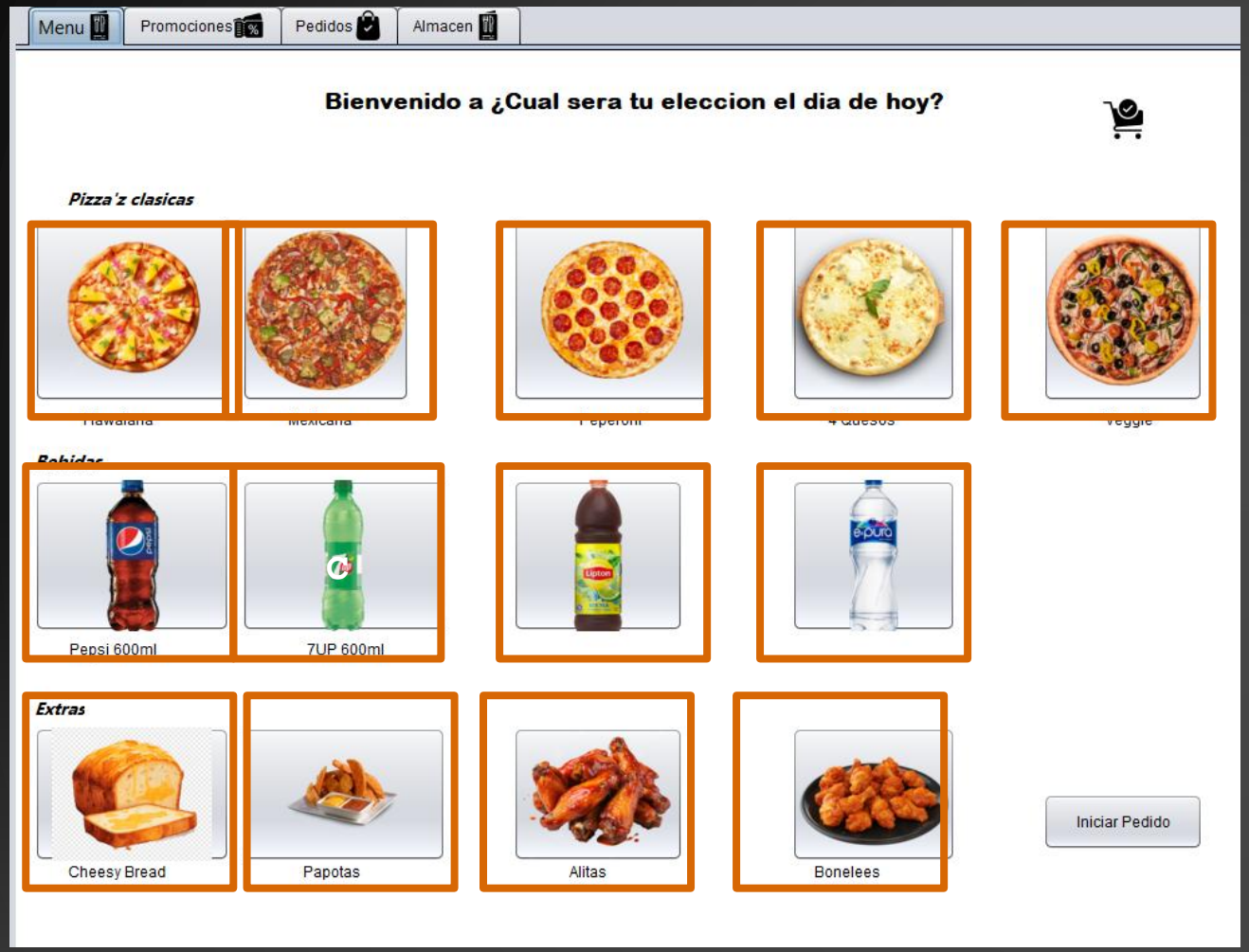
- Verificación pedidos existentes:

SELECT * FROM Pedidos;

- Insertar nuevo pedido (implícita en JPA):

INSERT INTO Pedidos (idUsuario, fecha, estado, total) VALUES (?, CURRENT_DATE, 'Creado', 0);

SELECCIÓN DE PRODUCTOS



Cada botón correspondiente a un producto o promoción realiza una consulta para obtener la información de un producto específico desde la base de datos, usando su identificador (idProducto).

Funcionamiento:

1. Cuando el usuario da clic en un botón (por ejemplo, Peperoni o Canela), el sistema asigna un ID de producto fijo (por ejemplo, 31 o 64).
2. Se crea un controlador ProductoJpaController que usa JPA para comunicarse con la base de datos.
3. El método findProducto(idProducto) ejecuta una consulta SELECT para buscar el producto con ese ID.
4. Si el producto existe, se abre la ventana PrePedido mostrando los datos del producto seleccionado.
5. Si no se encuentra, se muestra un mensaje de error.

SELECCIÓN DE PROMOCIONES

Cada botón correspondiente a un producto o promoción realiza una consulta para obtener la información de un producto específico desde la base de datos, usando su identificador (idProducto).

Funcionamiento:

1. Cuando el usuario da clic en un botón (por ejemplo, Peperoni o Canela), el sistema asigna un ID de producto fijo (por ejemplo, 31 o 64).
2. Se crea un controlador ProductoJpaController que usa JPA para comunicarse con la base de datos.
3. El método findProducto(idProducto) ejecuta una consulta SELECT para buscar el producto con ese ID.
4. Si el producto existe, se abre la ventana PrePedido mostrando los datos del producto seleccionado.
5. Si no se encuentra, se muestra un mensaje de error.

TABLA PRODUCTO

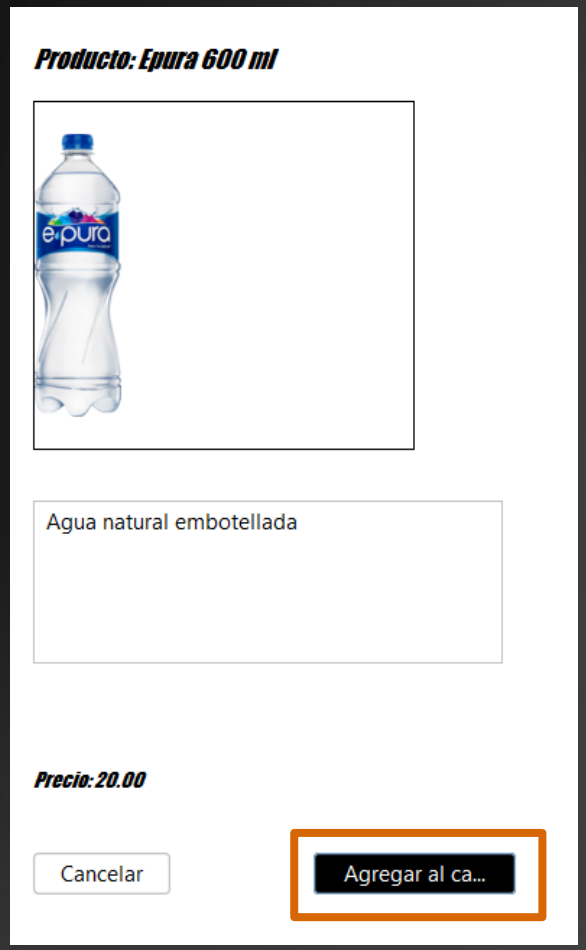
id_producto [PK] integer	nombre character varying (50)	descripcion text	precio numeric (10,2)	codigo character varying (10)
29	Hawaiana	Pizza con jamón, piña y queso mozzarella	139.00	029HAWA
30	Mexicana	Pizza con chorizo, frijoles, jalapeño y queso	149.00	030MEXA
31	Pepperoni	Pizza clásica con pepperoni y queso mozzarella	139.00	031PEPI
32	4 Quesos	Mezcla de mozzarella, gouda, parmesano y cheddar	149.00	0324 QS
33	Veggie	Pizza con pimientos, champiñones, cebolla y aceitunas	139.00	033VEGE
34	Jamón y Queso	Pizza con jamón de pierna y mozzarella	129.00	034JAMO
35	Suprema	Pizza con pepperoni, salchicha italiana, champiñones y pimient...	159.00	035SUPA
36	BBQ Chicken	Pollo en salsa BBQ con cebolla morada y mozzarella	159.00	036BBQN
37	Carnes Frías	Pepperoni, jamón y salchicha italiana	159.00	037CARS
38	Mar y Tierra	Atún, camarones, jamón y queso	169.00	038MARA
39	Caprese	Tomate cherry, albahaca y mozzarella fresca	139.00	039CAPE
40	Italiana	Pepperoni, aceitunas negras y champiñones	149.00	040ITAA
41	Ranchera	Carne molida, cebolla y chile poblano	149.00	041RANA

CONSULTAS:

- Implícita (generada por JPA):

SELECT * FROM Producto WHERE idProducto = ?;

BOTÓN PARA AGREGAR AL CARRITO: **BTNAGREGARC**



Esta consulta inserta un nuevo registro en la tabla PedidoProducto, relacionando el pedido actual con el producto seleccionado, y actualiza el total del pedido en la tabla Pedidos.

Funcionamiento:

1. El sistema verifica que exista un pedido activo. Si no hay, muestra un mensaje indicando que primero debe crearse un pedido.
2. Crea un objeto PedidoProducto con la siguiente información:
   - El pedido actual (idPedido).
   - El producto seleccionado (idProducto).
   - Cantidad del producto (por defecto, 1).
   - Subtotal, calculado con el precio del producto.
3. Guarda el nuevo registro en la tabla PedidoProducto usando el método create().
4. Actualiza el total del pedido sumando el precio del nuevo producto.
5. Finalmente, se modifica el pedido en la base de datos con el nuevo total mediante PedidosJpaController.edit().

TABLA PEDIDO-PRODUCTO

id_pedido_producto [PK] integer	id_pedido integer	id_producto integer	cantidad integer	subtotal numeric (10,2)
1	12	31	1	139.00
2	12	30	1	149.00
3	12	91	1	299.00
4	12	49	1	79.00
5	13	76	1	25.00
6	15	77	1	20.00
7	16	77	1	20.00
8	17	32	1	149.00
9	22	77	1	20.00

CONSULTAS:

- Inserción del producto en el pedido (generada por JPA):

```
INSERT INTO PedidoProducto (idPedido, idProducto, cantidad, subtotal)
VALUES (?, ?, 1, ?);
```

TABLA PEDIDO

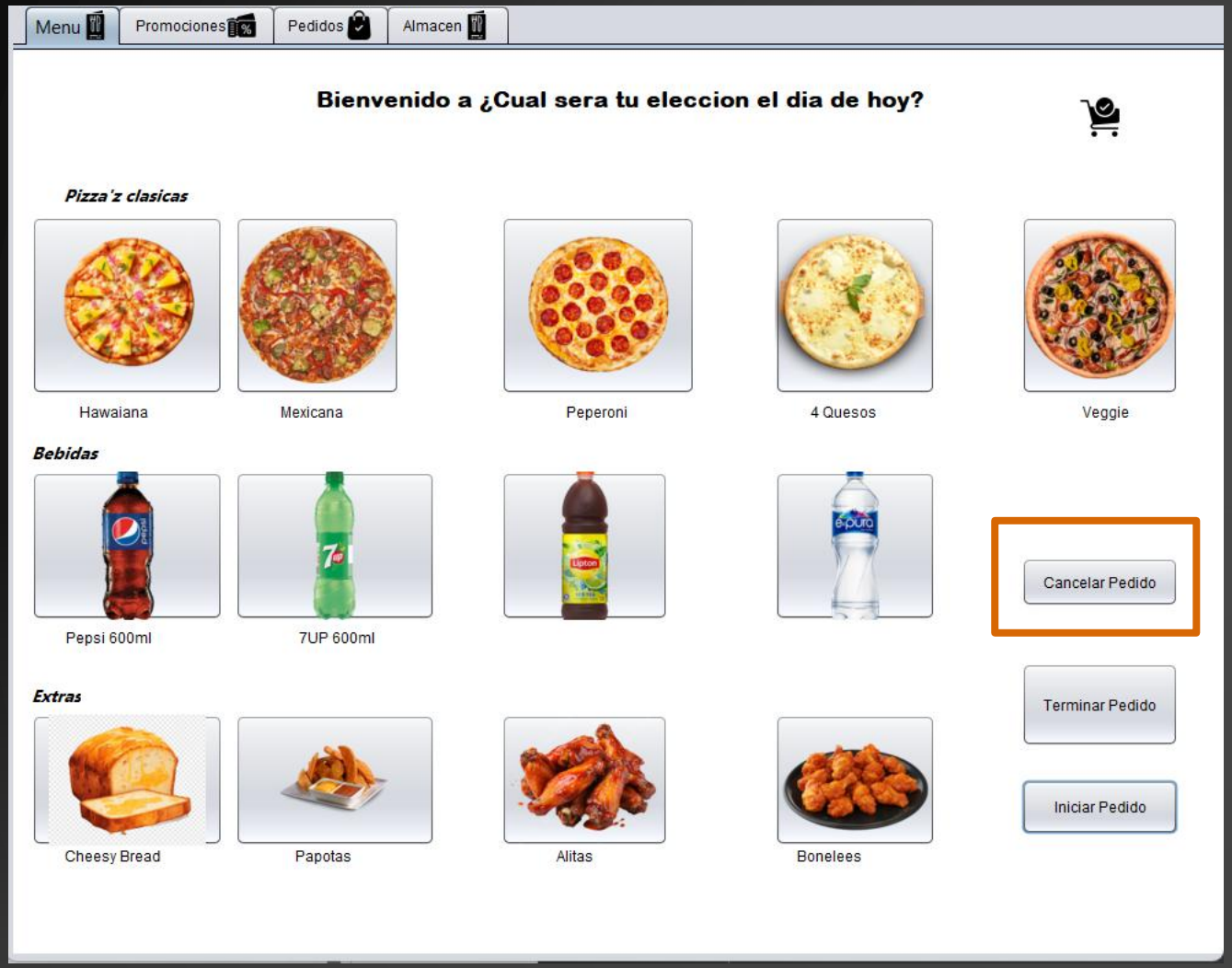
id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS (generada por JPA)**:**

- Actualización del total del pedido:

```
UPDATE Pedidos
SET total = total + ?
WHERE idPedido = ?;
```

BOTON PARA CANCELAR PEDIDO: **BTNCANCELARP**



Esta consulta busca el pedido activo del usuario (en estado "Creado") y, si lo encuentra, actualiza su estado a "Cancelado" en la base de datos.

Funcionamiento:

1. El sistema obtiene todos los pedidos mediante PedidosJpaController.
2. Recorre la lista para localizar un pedido del usuario actual con estado "Creado".
3. Si no hay pedidos activos, se muestra un mensaje informando que no hay nada que cancelar.
4. Si se encuentra un pedido, el sistema cambia su estado a "Cancelado".
5. Se actualiza el registro en la base de datos con el método edit() del controlador JPA.
6. Se confirma la cancelación mediante un mensaje.

TABLA PEDIDO

id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS:

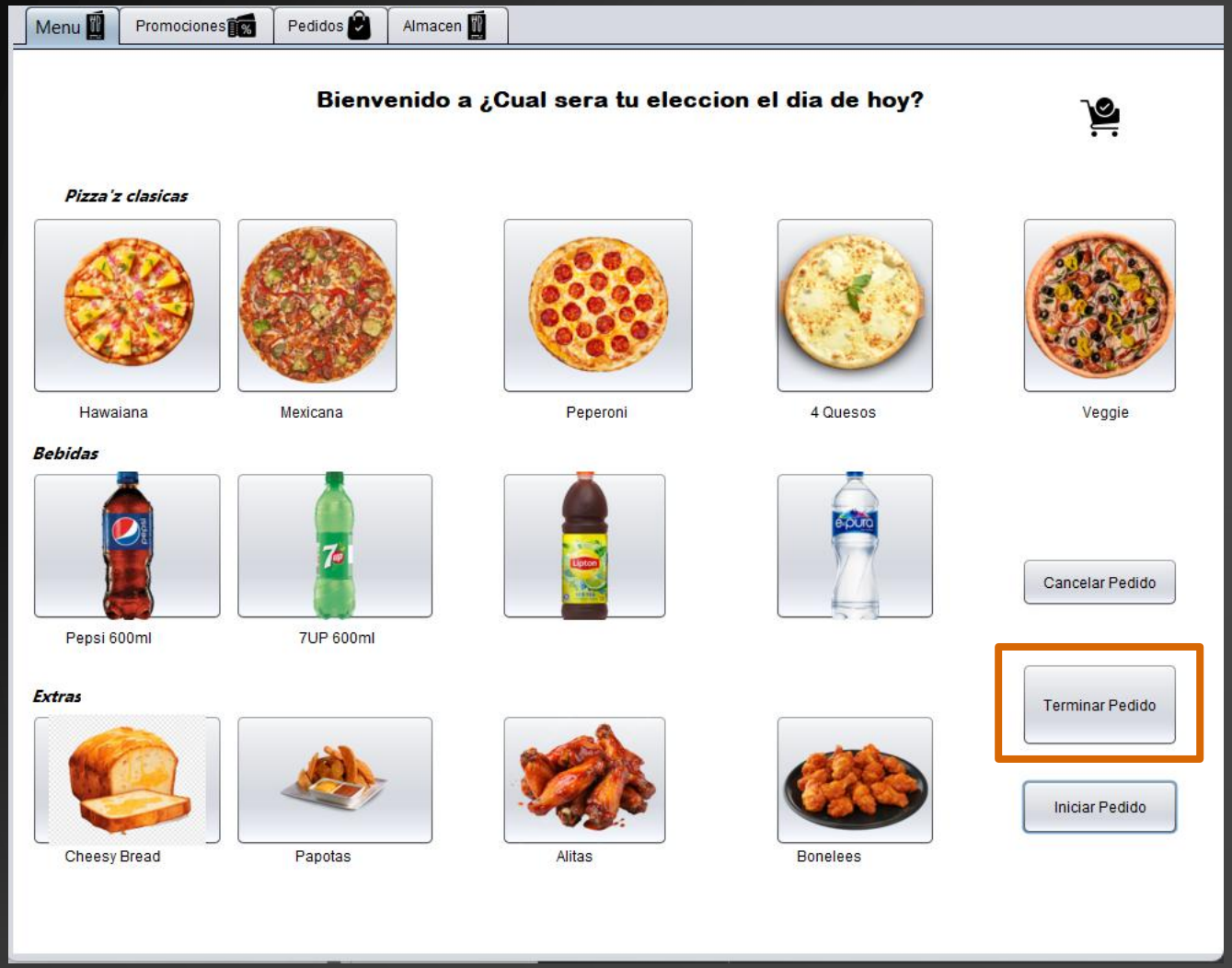
- Búsqueda del pedido activo:

```
SELECT * FROM Pedidos;
```

- Actualización del estado:

```
UPDATE Pedidos
SET estado = 'Cancelado'
WHERE idPedido = ?;
```

BOTON PARA TERMINAR PEDIDO: **BTNTERMINAR**



Esta consulta verifica que el pedido tenga productos agregados y, si es así, actualiza el estado del pedido a "Pendiente de Pago" dentro de la base de datos.

Funcionamiento:

1. El sistema valida que el pedido actual tenga un total mayor que cero (es decir, que contenga productos).
2. Si el pedido está vacío, muestra un mensaje indicando que debe agregar productos antes de continuar.
3. Si el pedido es válido, se obtiene nuevamente desde la base de datos mediante findPedidos(idPedido).
4. Se actualiza su estado a "Pendiente de Pago" usando el método edit() del controlador JPA.
5. Se muestra un mensaje confirmando que el pedido ha sido finalizado y queda listo para proceder al pago.
6. Finalmente, se habilita el botón de pago y se desactiva el de crear pedido.

TABLA PEDIDO

id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS (implícitas en JPA):

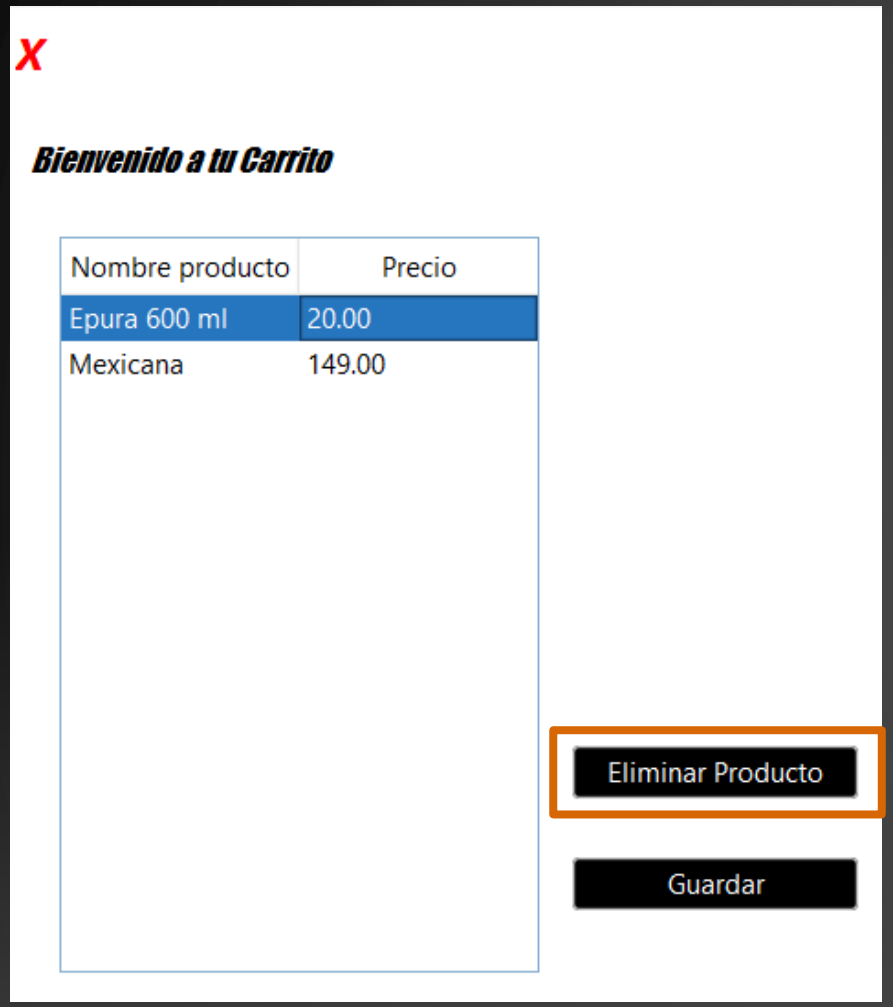
- Búsqueda del pedido actual:

```
SELECT * FROM Pedidos WHERE
idPedido = ?;
```

- Actualización del estado:

```
UPDATE Pedidos
SET estado = 'Pendiente de Pago'
WHERE idPedido = ?;
```

BOTÓN PARA ELIMINAR DEL CARRITO: **BTNELIMINAR**



Esta consulta elimina de la base de datos el producto seleccionado del carrito (tabla PedidoProducto), actualizando visualmente la lista de productos mostrada al usuario.

Funcionamiento:

1. El sistema obtiene la fila seleccionada en la tabla del carrito (contenidoP).
2. Recupera el ID del registro correspondiente en la tabla PedidoProducto.
3. Utiliza el método destroy(idPedidoProducto) del PedidoProductoJpaController para eliminar el registro de la base de datos.
4. Muestra un mensaje de confirmación indicando que el producto fue eliminado correctamente.
5. Llama al método enlistarProductos() para actualizar la tabla visual con los productos restantes.
6. Si no se selecciona ningún producto, se muestra un mensaje de advertencia.

TABLA PEDIDO-PRODUCTO

id_pedido_producto [PK] integer	id_pedido integer	id_producto integer	cantidad integer	subtotal numeric (10,2)
1	12	31	1	139.00
2	12	30	1	149.00
3	12	91	1	299.00
4	12	49	1	79.00
5	13	76	1	25.00
6	15	77	1	20.00
7	16	77	1	20.00
8	17	32	1	149.00
9	22	77	1	20.00

CONSULTAS (implícita en JPA):

```
DELETE FROM PedidoProducto
WHERE idPedidoProducto = ?;
```

BOTÓN PARA GUARDAR CAMBIOS: **BTNGUARDAR**

Este botón actualiza el total del pedido en la base de datos después de realizar cambios en los productos del carrito. Calcula nuevamente el total y lo guarda en la tabla pedidos.

Funcionamiento:

1. Se obtiene el nuevo total del pedido mediante el método calcularNuevoTotal().
2. Se busca el pedido correspondiente en la base de datos usando su idPedido.
3. Se actualiza el campo total con el nuevo valor.
4. Se ejecuta la edición del registro mediante controlador.edit(actualizacion).
5. Se muestra un mensaje confirmando la actualización o un mensaje de error si falla.

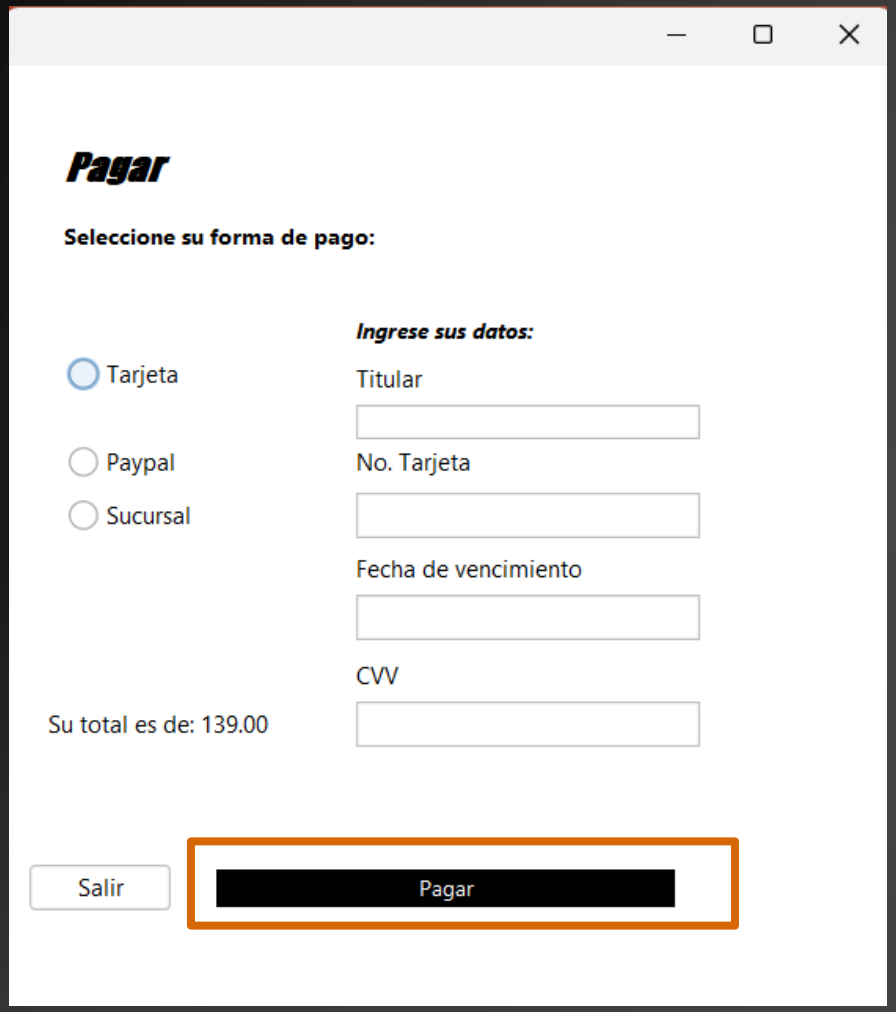
TABLA PEDIDO

id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS (implícitas en JPA)**:**

```
UPDATE pedidos
SET total = ?
WHERE id_pedido = ?
```

BOTÓN PAGAR: **BTNPAGAR**



Este botón valida los datos ingresados según el método de pago elegido (Tarjeta, PayPal o Sucursal). Si los datos son correctos, registra el pago del pedido, cambia su estado a "Pagado" y genera una notificación para el usuario.

Funcionamiento:

1. Se identifica el método de pago seleccionado entre las opciones: Tarjeta, PayPal o Sucursal.
2. Se validan los campos requeridos según la opción elegida; si falta alguno, el sistema muestra un mensaje y detiene el proceso.
3. Se crea la conexión con la base de datos a través del EntityManagerFactory.
4. Se busca el pedido actual en la base de datos con ultimopedido.findPedidos(pedidoActual.getIdPedido()).
5. Si el pedido existe y su estado es "Pendiente de Pago", se actualiza su estado a "Pagado".
6. Se crea un nuevo registro en la tabla Notificaciones con un mensaje informando al usuario del pago exitoso.
7. Se muestra un mensaje de confirmación al usuario ("Se pagó").

TABLA PEDIDO

id_pedido [PK] integer	id_usuario integer	fecha timestamp without time zone	estado character varying (20)	total numeric (10,2)
1	1	2025-10-22 18:07:52.94	Pagado	350.00
2	2	2025-10-22 18:07:52.94	Pendiente de Pago	215.00
3	3	2025-10-22 18:07:52.94	En Preparación	260.00
4	6	2025-10-22 18:07:52.94	Enviado	295.00
5	5	2025-10-22 18:07:52.94	Entregado	420.00
6	1	2025-10-22 18:55:10.94	Pendiente de Pago	355.00
7	10	2025-10-27 14:01:16.984	Pagado	815.00
8	10	2025-10-27 21:16:00.706	Cancelado	0.00
9	10	2025-10-27 21:23:33.644	Cancelado	0.00
10	10	2025-10-27 21:23:42.225	Cancelado	0.00
11	10	2025-10-27 22:06:34.194	Cancelado	0.00
12	11	2025-10-28 18:30:34.438	Cancelado	666.00

CONSULTAS (implícitas en JPA)**:**

```
UPDATE pedidos
SET estado = 'Pagado'
WHERE id_pedido = ?;
```

TABLA NOTIFICACIONES

id_notificacion [PK] integer	id_usuario integer	mensaje text	fecha timestamp without time zone
1	1	Tu pedido #1 ha sido confirmado y está en preparación.	2025-10-22 18:07:52.94
2	2	Tienes un pedido pendiente de pago. Por favor completa el pag...	2025-10-22 18:07:52.94
3	3	Tu pedido #3 está en preparación.	2025-10-22 18:07:52.94
4	5	Tu pedido #5 ha sido entregado. ¡Gracias!	2025-10-22 18:07:52.94
5	6	Pedido asignado para reparto #4.	2025-10-22 18:07:52.94
6	10	Tu pedido 7 Ha sido pagado medianteTarjeta.	2025-10-22 18:07:52.94

CONSULTAS:

- Implícita (generada por JPA):

```
INSERT INTO notificaciones (id_usuario, mensaje)
VALUES (?, ?);
```

TABLA PAGO

id_pago [PK] integer	id_pedido integer	metodo_pago character varying (20)	estado_pago character varying (20)	fecha_pago timestamp without time zone	total_pagado numeric (10,2)
1	1	Tarjeta	Aprobado	2025-10-22 18:07:52.94	245.00
2	2	Efectivo	Pendiente	2025-10-22 18:07:52.94	205.00
3	3	PayPal	Aprobado	2025-10-22 18:07:52.94	320.00
4	4	Tarjeta	Aprobado	2025-10-22 18:07:52.94	245.00
5	5	Efectivo	Aprobado	2025-10-22 18:07:52.94	420.00

CONSULTAS:

- Implícita (generada por JPA):

```
INSERT INTO pagos (id_pedido, metodo_pago, estado_pago, fecha_pago,
total_pago)
VALUES (?, ?, ?, ?, ?);
```