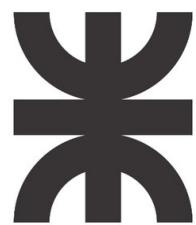


Documentación de software



Lenguaje de programación JAVA

2023

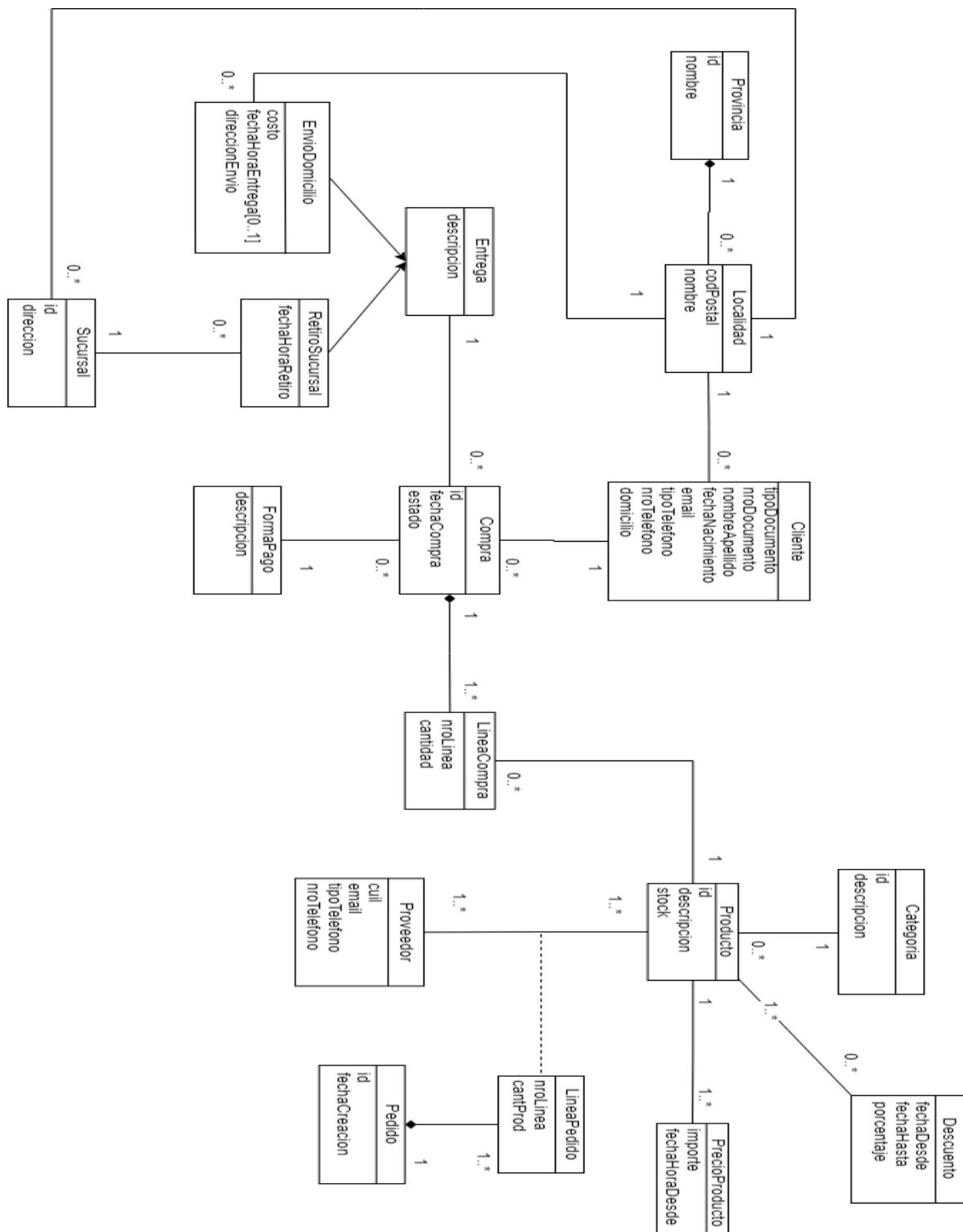
Integrante:
Alvaro Marina

Docentes:
Adrián Meca
Ricardo Tabacman

Introducción

El software desarrollado es un ecommerce. Tiene la funcionalidad típica del carrito de compras, y otras como el crear una cuenta, editar el perfil, y realizar ABMC en el caso de que el usuario sea Administrador.

Diagramas Clases



CUR-1: Realizar compra de productos (reestructurado)

Clasificación del Caso de Uso

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Reestructurado	Sistema	Negra	Real	Semántico

Meta del CASO DE USO: Agregar los productos al carrito de compras y confirmar el pedido

ACTORES

Primario: Cliente

PRECONDICIONES (de negocio): El ecommerce tiene productos y stock de cada uno.

PRECONDICIONES (de sistema): El cliente ya posee una cuenta creada.

DISPARADOR: Una persona ingresa al sitio web para realizar una compra

FLUJO DE SUCESOS:

CAMINO BÁSICO:

1. El cliente se loguea en el sistema según **CUU-1 Logueo**. Sistema valida.
2. El cliente carga los productos en el carrito según **CUU-2 Carga carrito**.
3. El cliente confirma la compra según **CUU-3 Confirma compra**. Sistema registra

CAMINOS ALTERNATIVOS:

1.a<anterior> El cliente ya está logueado:

1.a.1 Sigue en el paso 2.

3.a<durante> El cliente cancela la compra. FIN CU

CUU-1: Logueo

Clasificación del Caso de Uso

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Sin estructurar	Usuario	Negra	Real	Semántico

Meta del CASO DE USO: Acceder a los servicios ofrecidos por la membresía

ACTORES

Primario: *Miembro*

PRECONDICIONES (de negocio):

PRECONDICIONES (de sistema): *El usuario tiene una cuenta registrada.*

DISPARADOR: *El cliente desea loguearse en el sitio*

FLUJO DE SUCESOS:

CAMINO BÁSICO:

1. *El usuario ingresa email y contraseña*
2. *El sistema redirecciona a la pagina principal*

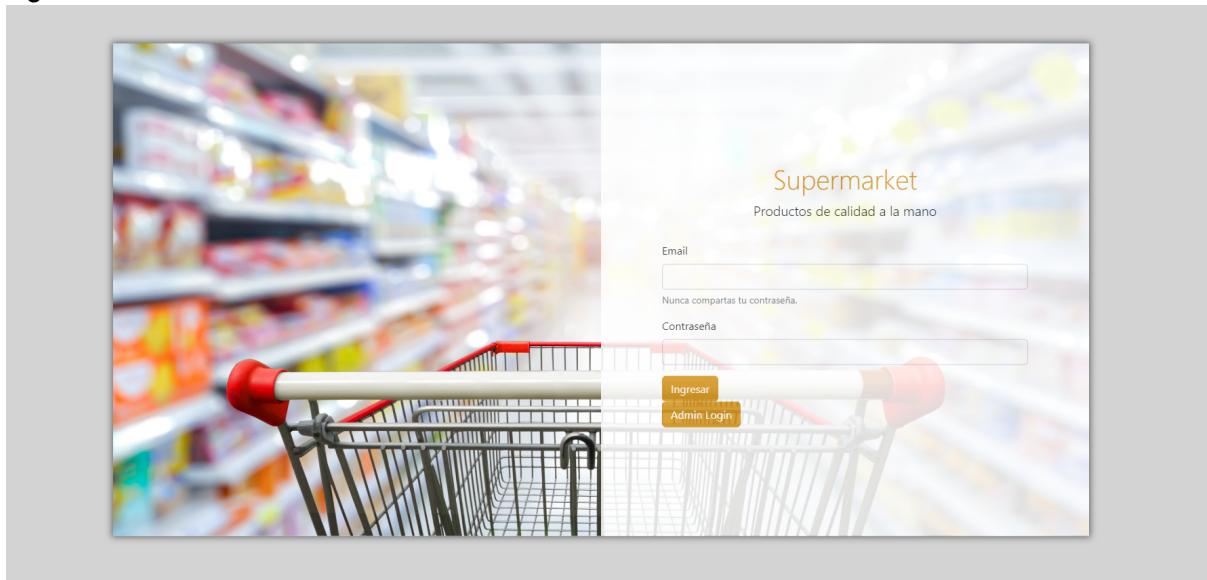
CAMINOS ALTERNATIVOS:

1.a<posterior> *El cliente ingresa datos incorrectos*

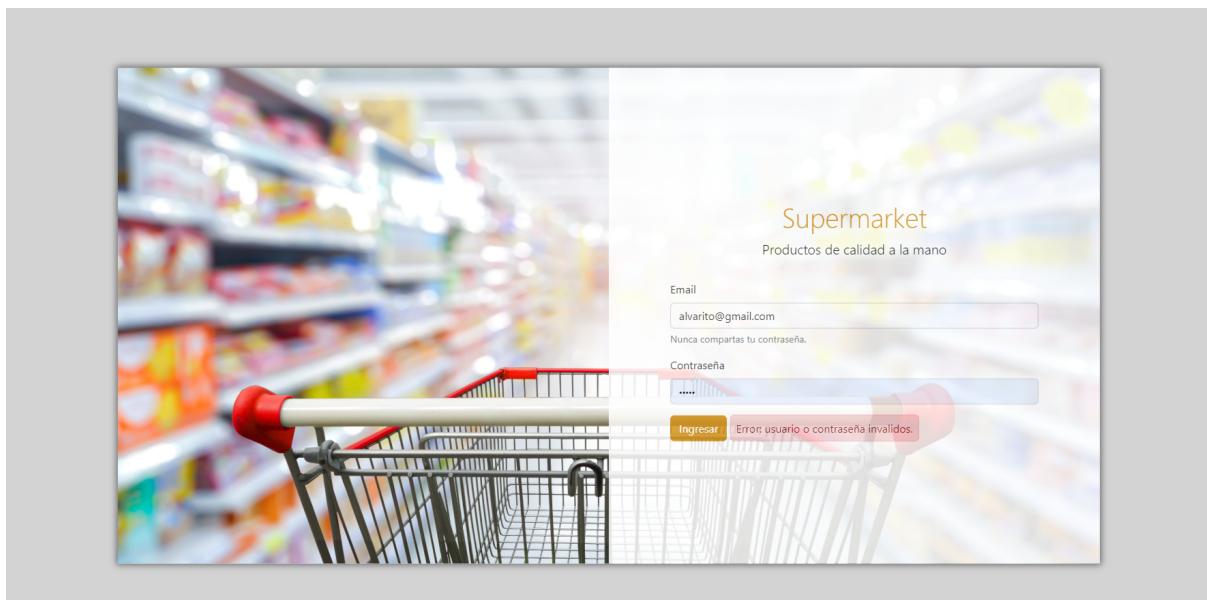
1.a.1 *El sistema da una notificación. Vuelve al paso 1.*

Interfaz de usuario

Ingresar al sistema



Sistema valida datos del usuario



Redireccion a home

Home Categorías ▾ Ofertas Sucursales Ayuda Acerca de

Login Carrito Mi cuenta Salir Dropdown link ▾



Supermarket
Productos de calidad a la mano

Buscar



CUU-2: Cargar carrito

Clasificación del Caso de Uso

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Sin estructurar	Usuario	Negra	Real	Semántico

Meta del CASO DE USO: Cargar los productos deseados dentro del carrito

ACTORES

Primario: Cliente

PRECONDICIONES (de negocio):

PRECONDICIONES (de sistema): El usuario ya está logueado

DISPARADOR: El cliente realiza búsqueda de productos

FLUJO DE SUCESOS:

CAMINO BÁSICO:

1. El cliente busca producto por descripción o por categoría
2. El cliente agrega el producto al carrito
3. Se repiten 1 y 2 hasta que el cliente ingrese todos los productos que desee al carrito.

CAMINOS ALTERNATIVOS:

2.a<durante> El cliente solicita una cantidad mayor al stock existente

2.a.1 El sistema da una notificación

2.a.1.a El cliente elige una nueva cantidad. Sigue en paso 3

2.a.1.b El cliente elige otro producto. Sigue en paso 3

Interfaz de usuario

Busqueda por filtro de categoria

Home Categorías ▾ Ofertas Sucursales Ayuda Acerca de Login Carrito Mi cuenta Salir Dropdown link ▾

Supermarket

Productos de calidad a la mano

Buscar

 Nombre: Fideos Stock: 22 Id: 3 Ingresar la cantidad: <input type="text"/> Precio: \$102.24 Agregar al carrito Eliminar del carrito	 Nombre: Block Stock: 10 Id: 5 Ingresar la cantidad: <input type="text"/> Precio: \$197.94 Agregar al carrito Eliminar del carrito	 Nombre: Raviol Stock: 9 Id: 10 Ingresar la cantidad: <input type="text"/> Precio: \$61.85 Agregar al carrito Eliminar del carrito	 Nombre: Variedad Stock: 6 Id: 11 Ingresar la cantidad: <input type="text"/> Precio: \$96.49 Agregar al carrito Eliminar del carrito
---	--	--	---

Busqueda por filtro de descripcion

Home Categorías ▾ Ofertas Sucursales Ayuda Acerca de Login Carrito Mi cuenta Salir Dropdown link ▾

Supermarket

Productos de calidad a la mano

fideo Buscar

 Nombre: Fideos Stock: 22 Id: 3 Ingresar la cantidad: <input type="text"/> Precio: \$102.24 Agregar al carrito Eliminar del carrito

[Confirmar compra](#)

Validación de cantidad < stock



Nombre:

Coca-Cola

Stock:

16

Id:

14

Ingrese la cantidad:

20



Value must be less than or equal to 16.

[Continuar del carrito](#)

busquedaProducto.java

```
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.util.LinkedList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import Entities.Categoría;
import Entities.LineaCompra;
import Entities.Producto;
import Entities.Provincia;
import data.DbHandlerCategorias;
import data.DbHandlerProductos;
import data.DbHandlerProvincias;
import data.DbHandlerPedidos;
import data.DbHandlerLineaPedido;
import data.DbHandlerLocalidades;

/**
 *
 * * Servlet implementation class busquedaProducto
 */
@WebServlet(name = "busquedaProducto", urlPatterns = {
    "/busquedaProducto" })
public class busquedaProducto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public busquedaProducto() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request,
```

```

HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    HttpSession session = request.getSession();
    DbHandlerProductos dbProductos = new DbHandlerProductos();
    DbHandlerPedidos dbPedido = new DbHandlerPedidos();
    DbHandlerLineaPedido dbLineaPedido = new
DbHandlerLineaPedido();

    String buscar = request.getParameter("descripcion");
    String cartOrder = request.getParameter("order");
    String idCat = request.getParameter("catId");

    LinkedList<Producto> productos = new LinkedList<Producto>();
    LinkedList<Producto> prodsdeLista = new
LinkedList<Producto>();

    if(buscar != null && buscar != "") {
        productos =
dbProductos.selectProductoByDescripcion(buscar);
    } else if(idCat != null) {
        productos =
dbProductos.selectProductByCategory(Integer.parseInt(idCat));
        buscar = "";
    }
    String pedidoCreado = (String)
session.getAttribute("pedidoCreado");
    Integer idPedido = (Integer)
session.getAttribute("idPedido");
    LinkedList<LineaCompra> listaLC = (LinkedList<LineaCompra>)
session.getAttribute("listaLC");
    System.out.println("pedido Creado "+ pedidoCreado);
    System.out.println("idPedido "+ idPedido);
    System.out.println("listaLC "+ listaLC);
    if(pedidoCreado == null) {
        pedidoCreado = "creado";
}

```

```

        idPedido = dbPedido.getLastId()+1;
        listaLC = new LinkedList<LineaCompra>();
        System.out.println("entre al if e inicialice las
variables, ahora pedidoCreado = "+pedidoCreado+ " idPedido = "+idPedido+
listaLC = "+listaLC);
                session.setAttribute("idPedido", idPedido);
                session.setAttribute("pedidoCreado", pedidoCreado);
                session.setAttribute("listaLC", listaLC);
            }
            if(cartOrder != null &&
cartOrder.equalsIgnoreCase("add-to-cart")){
                System.out.println("entre al add to cart");
                Integer prod =
Integer.parseInt(request.getParameter("p-id"));
                Integer cantidad =
Integer.parseInt(request.getParameter("cantidad"));
                String descripcion =
request.getParameter("p-descripcion");
                Double precio =
Double.parseDouble(request.getParameter("precio"));
                System.out.println("id Pedido =
"+idPedido+ " idProducto = "+prod+ " cantidad = "+cantidad);
                LineaCompra lc = new LineaCompra();
                lc.setIdProducto(prod);
                lc.setNroPedido(idPedido);
                lc.setCantidad(cantidad);
                lc.setPrecio(precio);
                lc.setDescripcion(descripcion);
                listaLC.add(lc);
//                dbLineaPedido.addLineaPedido(idPedido,
prod, cantidad, precio, descripcion);
            }
            prodsdeLista = dbLineaPedido.selectProductosList(idPedido);

            if(listaLC != null) {
//                System.out.println("primer elemento "+listaLC.getFirst());
                if(productos != null && listaLC.size() > 0) {
                    for(LineaCompra lc: listaLC){
                        for(Producto p: productos) {
                            if(lc.getIdProducto() == p.getId()) {
                                productos.remove(p);
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        request.setAttribute("productos", productos);
        request.setAttribute("idCat", idCat);
        request.setAttribute("descripcion", buscar);
        request.setAttribute("idPedido", idPedido);

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

```

buscaFotos.java

```

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import Entities.Producto;
import data.DbHandlerProductos;

/*
 * Servlet implementation class buscaFotos
 */
@WebServlet(name = "buscaFotos", urlPatterns = { "/buscaFotos" })
public class buscaFotos extends HttpServlet {
    private static final long serialVersionUID = 1L;

/*
 * @see HttpServlet#HttpServlet()
 */
public buscaFotos() {
    super();
    // TODO Auto-generated constructor stub
}

/*
 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
*/
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
}

```

```

        DbHandlerProductos dbProductos = new DbHandlerProductos();
        Integer id =
    Integer.parseInt(request.getParameter("idProd"));
        Producto p = dbProductos.selectProducto(id);
        response.setContentType("image/jpeg");
        response.setContentLength(p.getImageBytes().length);
        response.getOutputStream().write(p.getImageBytes());
        response.getOutputStream().flush();
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse response)
    */
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
}

```

```

package data;

import java.awt.Image;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Base64;
import java.util.LinkedList;

import javax.swing.ImageIcon;

import Entities.Categoría;
import Entities.Producto;
import Entities.Proveedor;

```

```

public class DbHandlerProductos extends DbHandler{

    public DbHandlerProductos() {
        try {
            Class.forName(driver);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public LinkedList<Producto> selectProducto(String order) throws
IOException { // Devuelve todos los productos

        Statement stmt = null;
        ResultSet rs = null;
        Connection conn;

        try {
            conn = this.getConnection();
            LinkedList<Producto> productos = new
LinkedList<Producto>();
            stmt = conn.createStatement();
            String query = "select pr.valor, pr.idProducto,
p.descripcion as 'prod-desc', p.foto as 'foto', stock, p.idCategoria as
'idCategoria', cat.descripcion as 'cat-desc',\r\n"
                    + " prov.idProveedor as 'idProveedor',
Nombre, cuil, tipoTelefono, nroTelefono from producto p\r\n"
                    + "inner join proveedor prov on
prov.idProveedor = p.idProveedor\r\n"
                    + "inner join categoria cat on
cat.idCategoria = p.idCategoria\r\n"
                    + "inner join (select max(fechaDesde) as
fec, idProducto from precio group by precio.idProducto) maxprec2 on
p.idProducto = maxprec2.idProducto \r\n"
                    + "inner join precio pr on pr.idProducto =
p.idProducto and\r\n" + "pr.fechaDesde = maxprec2.fec";

            rs = stmt.executeQuery(query);
            while (rs != null && rs.next()) {
                Producto producto = new Producto();
                Categoria categoria = new Categoria();
                Proveedor proveedor = new Proveedor();
                Blob clob = rs.getBlob("foto");
                byte[] byteArr =

```

```

clob.getBytes(1,(int)clob.length());
        byte[] imageBytes = rs.getBytes("foto");

        categoria.setId(rs.getInt("idCategoria"));

categoria.setDescripcion(rs.getString("cat-desc"));

proveedor.setId(rs.getInt("idProveedor"));

proveedor.setTipoTelefono(rs.getString("tipoTelefono"));

proveedor.setNroTelefono(rs.getString("nroTelefono"));
        proveedor.setNombre(rs.getString("Nombre"));
        proveedor.setCuil(rs.getString("cuil"));

        producto.setId(rs.getInt("idProducto"));

producto.setDescripcion(rs.getString("prod-desc"));
        producto.setStock(rs.getInt("stock"));
        producto.setPrecio(rs.getDouble("valor"));
        producto.setCategoria(categoria);
        producto.setProveedor(proveedor);
        producto.setImageBytes(imageBytes);

        productos.add(producto);
    }
    return productos;
} catch (SQLException e) {
    e.printStackTrace();
    return null;
} finally {
    try {
        if (rs != null)
            rs.close();
        if (stmt != null)
            stmt.close();
        this.releaseConnection();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

public Producto selectProducto(Producto prod) { // Devuelve un
producto

```

```
PreparedStatement stmt = null;
ResultSet rs = null;
Connection conn;

try {
    conn = this.getConnection();
    LinkedList<Producto> productos = new
LinkedList<Producto>();
    String query = "select precio.valor, max(fechaDesde),
precio.idProducto, producto.descripcion as 'prod-desc', stock,
producto.idCategoria as 'idCategoria', "
        + "categoria.descripcion as 'cat-desc',
proveedor.idProveedor as 'idProveedor', Nombre, cuil, tipoTelefono,
nroTelefono\r\n"
        + "from precio inner join producto on
producto.idProducto = precio.idProducto\r\n"
        + "inner join categoria on
producto.idCategoria = categoria.idCategoria\r\n"
        + "inner join proveedor on
producto.idProveedor = proveedor.idProveedor\r\n"
        + "where producto.idProducto = ? and
precio.fechaDesde = (select max(fechaDesde) from precio p where
p.idProducto = ?) group by precio.idProducto\r\n"
        + "";
    stmt = conn.prepareStatement(query);
    stmt.setInt(1, prod.getId());
    stmt.setInt(2, prod.getId());
    rs = stmt.executeQuery();

    while (rs != null && rs.next()) {
        Categoria categoria = new Categoria();
        Proveedor proveedor = new Proveedor();

        categoria.setId(rs.getInt("idCategoria"));

        categoria.setDescripcion(rs.getString("cat-desc"));

        proveedor.setId(rs.getInt("idProveedor"));

        proveedor.setTipoTelefono(rs.getString("tipoTelefono"));

        proveedor.setNroTelefono(rs.getString("nroTelefono"));
            proveedor.setNombre(rs.getString("nombre"));
            proveedor.setCUIL(rs.getString("cuil"));
    }
}
```

```

        prod.setId(rs.getInt("idProducto"));
        prod.setDescripcion(rs.getString("prod-desc"));
        prod.setStock(rs.getInt("stock"));
        prod.setPrecio(rs.getDouble("precio.valor"));

        prod.setCategoria(categoría);
        prod.setProveedor(proveedor);

        productos.add(prod);
    }
    return prod;

} catch (SQLException e) {
    e.printStackTrace();
    return null;
} finally {
    try {
        if (rs != null)
            rs.close();
        if (stmt != null)
            stmt.close();
        this.releaseConnection();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

public Producto selectProducto(Integer id) { // Devuelve un
producto

    PreparedStatement stmt = null;
    ResultSet rs = null;
    Connection conn;
    Producto prod = new Producto();
    try {
        conn = this.getConnection();
//        LinkedList<Producto> productos = new
LinkedList<Producto>();
        String query = "select precio.valor, max(fechaDesde),
precio.idProducto, producto.foto as 'foto', producto.descripcion as
'prod-desc', stock, producto.idCategoria as 'idCategoria', "
                    + "categoria.descripcion as 'cat-desc',
proveedor.idProveedor as 'idProveedor', Nombre, cuil, tipoTelefono,

```

```

nroTelefono\r\n"
                + "from precio inner join producto on
producto.idProducto = precio.idProducto\r\n"
                + "inner join categoria on
producto.idCategoria = categoria.idCategoria\r\n"
                + "inner join proveedor on
producto.idProveedor = proveedor.idProveedor\r\n"
                + "where producto.idProducto = ? and
precio.fechaDesde = (select max(fechaDesde) from precio p where
p.idProducto = ?) group by precio.idProducto\r\n"
                + "";
stmt = conn.prepareStatement(query);
stmt.setInt(1, id);
stmt.setInt(2, id);
rs = stmt.executeQuery();

while (rs != null && rs.next()) {
    Categoria categoria = new Categoria();
    Proveedor proveedor = new Proveedor();
    byte[] imageBytes = rs.getBytes("foto");

    categoria.setId(rs.getInt("idCategoria"));

    categoria.setDescripcion(rs.getString("cat-desc"));

    proveedor.setId(rs.getInt("idProveedor"));

    proveedor.setTipoTelefono(rs.getString("tipoTelefono"));

    proveedor.setNroTelefono(rs.getString("nroTelefono"));
    proveedor.setNombre(rs.getString("nombre"));
    proveedor.setCuil(rs.getString("cuil"));

    prod.setId(rs.getInt("idProducto"));
    prod.setDescripcion(rs.getString("prod-desc"));
    prod.setStock(rs.getInt("stock"));
    prod.setPrecio(rs.getDouble("precio.valor"));
    prod.setImageBytes(imageBytes);
    prod.setCategoria(categoria);
    prod.setProveedor(proveedor);

    //
    // productos.add(prod);
}

return prod;

} catch (SQLException e) {

```

```

        e.printStackTrace();
        return null;
    } finally {
        try {
            if (rs != null)
                rs.close();
            if (stmt != null)
                stmt.close();
            this.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public LinkedList<Producto> selectProductoByDescripcion(String
desc) { // Devuelve lista de productos que coinciden con una descripcion

    PreparedStatement stmt = null;
    ResultSet rs = null;
    Connection conn;
    try {
        conn = this.getConnection();
        LinkedList<Producto> productos = new
LinkedList<Producto>();
        String query = "select precio.valor,
precio.idProducto, producto.descripcion as 'prod-desc', stock,
producto.idCategoria as 'idCategoria', \r\n"
                    + "categoria.descripcion as 'cat-desc',
foto\r\n"
                    + "from precio inner join producto on
producto.idProducto = precio.idProducto\r\n"
                    + "inner join categoria on
producto.idCategoria = categoria.idCategoria\r\n"
                    + "inner join (select p.idProducto,
max(fechaDesde) as fecha from precio p group by p.idProducto) maxFechas
on maxFechas.idProducto = precio.idProducto and maxFechas.fecha =
precio.fechaDesde\r\n"
                    + "where producto.idProducto IN (select
distinct p.idProducto from producto p where\r\n"
                    + "p.descripcion like ?) group by
precio.idProducto";
        stmt = conn.prepareStatement(query);
        stmt.setString(1, "%" + desc + "%");
        rs = stmt.executeQuery();
    }
}

```

```

        while (rs != null && rs.next()) {
            Categoria categoria = new Categoria();
            Producto prod = new Producto();
            // Blob clob = rs.getBlob("foto");
            // byte[] byteArr =
            clob.getBytes(1,(int)clob.length());

            categoria.setId(rs.getInt("idCategoria"));

            categoria.setDescripcion(rs.getString("cat-desc"));

            prod.setId(rs.getInt("idProducto"));
            prod.setDescripcion(rs.getString("prod-desc"));
            prod.setStock(rs.getInt("stock"));
            prod.setPrecio(rs.getDouble("precio.valor"));
            prod.setCategoria(categoria);
            prod.setByteArr(byteArr);

            productos.add(prod);
        }
        return productos;

    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    } finally {
        try {
            if (rs != null)
                rs.close();
            if (stmt != null)
                stmt.close();
            this.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public void addProducto(Producto prod) {

    PreparedStatement stmt = null;
    ResultSet keyRS = null;
}

```

```

Connection conn = null;
int idprod;

try {
    conn = this.getConnection();
    stmt = conn.prepareStatement(
        "insert into producto(descripcion, stock,
idCategoria, idProveedor) " + "values(?, ?, ?, ?)",
        Statement.RETURN_GENERATED_KEYS);
    stmt.setString(1, prod.getDescripcion());
    stmt.setInt(2, prod.getStock());
    stmt.setInt(3, prod.getCategoría().getId());
    stmt.setInt(4, prod.getProveedor().getId());

    stmt.executeUpdate();
    keyRS = stmt.getGeneratedKeys();

    if (keyRS != null && keyRS.next()) {
        idprod = keyRS.getInt(1);
        this.addPrecioNuevoProducto(prod.getPrecio(),
idprod);

    }
}

} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        if (keyRS != null)
            keyRS.close();
        if (stmt != null)
            stmt.close();
        this.releaseConnection();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

public void addPrecioNuevoProducto(double precio, int idprod) {

PreparedStatement stmt = null;
ResultSet keyRS = null;
Connection conn = null;

```

```

        try {
            conn = this.getConnection();
            stmt = conn.prepareStatement("insert into
precio(valor, idProducto)\r\n" + "\r\n"
                + "values(?,?)",
Statement.RETURN_GENERATED_KEYS);
            stmt.setDouble(2, idprod);
            stmt.setDouble(1, precio);

            stmt.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (keyRS != null)
                    keyRS.close();
                if (stmt != null)
                    stmt.close();
                this.releaseConnection();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

public void addPrecio(double precio, int idprod) {

    PreparedStatement stmt = null;
    ResultSet keyRS = null;
    Connection conn = null;

    try {
        conn = this.getConnection();
        stmt = conn.prepareStatement("insert into
precio(valor, idProducto)\r\n" + "\r\n"
            + "select TRUNCATE(IF(pr.valor*(30/100 + 1)
< ?,pr.valor*(30/100 + 1), ?),2), pr.idProducto from producto p\r\n"
            + "inner join proveedor prov on
prov.idProveedor = p.idProveedor\r\n"
            + "inner join categoria cat on
cat.idCategoria = p.idCategoria\r\n"
            + "inner join (select max(fechaDesde) as
fec, idProducto from precio where YEAR(fechaDesde) =
(YEAR(current_date)) and idProducto = ? group by precio.idProducto)
maxprec2 on p.idProducto = maxprec2.idProducto \r\n"

```

```

        + "inner join precio pr on pr.idProducto =
p.idProducto and\r\n"
        + "pr.fechaDesde = maxprec2.fec;\r\n" + "",
Statement.RETURN_GENERATED_KEYS);
stmt.setInt(3, idprod);
stmt.setDouble(2, precio);
stmt.setDouble(1, precio);

stmt.executeUpdate();

} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        if (keyRS != null)
            keyRS.close();
        if (stmt != null)
            stmt.close();
        this.releaseConnection();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

public void delete(Producto prod) {
    Connection conn = null;
    PreparedStatement stmt = null;
    try {
        conn = this.getConnection();
        stmt = conn.prepareStatement("delete from producto
where idProducto = ?");
        stmt.setInt(1, prod.getId());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null)
                stmt.close();
            this.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
}

```

```

public void modifyProduct(Producto prod) {
    Connection conn = null;
    PreparedStatement stmt = null;
    try {
        conn = this.getConnection();
        stmt = conn.prepareStatement(
            "UPDATE producto SET descripcion = ?, stock
 = ?, idProveedor = ?, idCategoria = ? WHERE (idProducto = ?);");
        stmt.setString(1, prod.getDescripcion());
        stmt.setInt(2, prod.getStock());
        stmt.setInt(3, prod.getProveedor().getId());
        stmt.setInt(4, prod.getCategoría().getId());
        stmt.setInt(5, prod.getId());
        stmt.executeUpdate();
        this.addPrecio(prod.getPrecio(), prod.getId());
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null)
                stmt.close();
            this.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public void inflation(Double val) throws IOException {
    LinkedList<Producto> prods = this.selectProducto("a");
    for (Producto p : prods) {
        this.addPrecio(p.getPrecio() * (val / 100 + 1),
p.getId());
    }
}

public LinkedList<Producto> selectProductsOffers() {
    PreparedStatement stmt = null;
    ResultSet rs = null;
    Connection conn;
    try {
        conn = this.getConnection();
        LinkedList<Producto> productos = new
LinkedList<Producto>();
        String query = "select precio.valor,

```

```

precio.idProducto, precio.oferta, precio.fechaDesde,
producto.descripcion as 'prod-desc', stock, producto.idCategoria as
'idCategoria', \r\n"
                    + "categoria.descripcion as 'cat-desc',
foto from precio inner join producto on producto.idProducto =
precio.idProducto\r\n"
                    + "inner join categoria on
producto.idCategoria = categoria.idCategoria\r\n"
                    + "inner join (select p.idProducto,
max(fechaDesde) as fecha from precio p group by p.idProducto) maxFechas
on maxFechas.idProducto = precio.idProducto and maxFechas.fecha =
precio.fechaDesde\r\n"
                    + "where producto.idProducto IN (select
distinct p.idProducto from producto p) and precio.oferta = 1 group by
precio.idProducto";
stmt = conn.prepareStatement(query);
rs = stmt.executeQuery();

while (rs != null && rs.next()) {
    Categoria categoria = new Categoria();
    Producto prod = new Producto();
    byte[] imageBytes = rs.getBytes("foto");

    categoria.setId(rs.getInt("idCategoria"));

    categoria.setDescripcion(rs.getString("cat-desc"));

    prod.setId(rs.getInt("idProducto"));
    prod.setDescripcion(rs.getString("prod-desc"));
    prod.setStock(rs.getInt("stock"));
    prod.setPrecio(rs.getDouble("precio.valor"));
    prod.setCategoria(categoria);
    prod.setImageBytes(imageBytes);

    productos.add(prod);
}
return productos;

} catch (SQLException e) {
    e.printStackTrace();
    return null;
} finally {
    try {
        if (rs != null)
            rs.close();
    }
}

```

```

                if (stmt != null)
                    stmt.close();
                this.releaseConnection();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public LinkedList<Producto> selectProductByCategory(Integer idCat)
    { // Devuelve lista de productos que coinciden con una categoria

        PreparedStatement stmt = null;
        ResultSet rs = null;
        Connection conn;
        try {
            conn = this.getConnection();
            LinkedList<Producto> productos = new
LinkedList<Producto>();
            String query = "select precio.valor,
precio.idProducto, producto.descripcion as 'prod-desc', stock,
producto.idCategoria as 'idCategoria',\r\n"
                           + "categoria.descripcion as 'cat-desc',
foto\r\n"
                           + "from precio inner join producto on
producto.idProducto = precio.idProducto\r\n"
                           + "inner join categoria on
producto.idCategoria = categoria.idCategoria\r\n"
                           + "inner join (select p.idProducto,
max(fechaDesde) as fecha from precio p group by p.idProducto) maxFechas
on maxFechas.idProducto = precio.idProducto\r\n"
                           + "and maxFechas.fecha =
precio.fechaDesde\r\n"
                           + "where producto.idProducto IN (select
distinct p.idProducto from producto p where\r\n"
                           + "p.idCategoria like ?) group by
precio.idProducto";
            stmt = conn.prepareStatement(query);
            stmt.setInt(1, idCat);
            rs = stmt.executeQuery();

            while (rs != null && rs.next()) {
                Categoria categoria = new Categoria();
                Producto prod = new Producto();

                categoria.setId(rs.getInt("idCategoria"));

```

```

        categoria.setDescripcion(rs.getString("cat-desc"));

                prod.setId(rs.getInt("idProducto"));
                prod.setDescripcion(rs.getString("prod-desc"));
                prod.setStock(rs.getInt("stock"));
                prod.setPrecio(rs.getDouble("precio.valor"));
                prod.setCategoria(categoria);

            productos.add(prod);
        }
        return productos;

    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    } finally {
        try {
            if (rs != null)
                rs.close();
            if (stmt != null)
                stmt.close();
            this.releaseConnection();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

public void updateStock(Integer idProducto, Integer cantidad) {
    Connection conn = null;
    PreparedStatement stmt = null;
    try {
        conn = this.getConnection();
        stmt = conn.prepareStatement(
            "UPDATE producto SET stock = stock-? WHERE
(idProducto = ?)");
        stmt.setInt(1, cantidad);
        stmt.setInt(2, idProducto);
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null)

```

```
        stmt.close();
        this.releaseConnection();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

CUU-3: Confirma compra

Clasificación del Caso de Uso

Clasificación del Caso de Uso					
Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Sin estructurar	Usuario	Negra	Real	Semántico

Meta del CASO DE USO: Cumplir metas y objetivos de entrenamiento del usuario miembro

ACTORES

ACTORES
Primario: *Staff*

Otros: *Miembro*

PRECONDICIONES (de negocio): Existen personal de staff en horario

PRECONDICIONES (de negocio): Existen personal de statt en horario
PRECONDICIONES (de sistema): El carrito del usuario ya está cargado

DISPARADOR: *El miembro desea confirmar la compra*

FLUJO DE SUCESOS:

PESSO DE SUCESSO CAMINHO BÁSICO:

1. *El cliente confirma los productos que posee en el carrito y la cantidad*
 2. *El cliente selecciona tipo de entrega a domicilio*
 3. *El cliente selecciona forma de pago con tarjeta*
 4. *El cliente ingresa los datos de tarjeta*

CAMINOS ALTERNATIVOS:

1.a<durante> El cliente desea agregar mas productos:

1-a-1 El cliente vuelve a cargar productos. Vuelve al paso 1

2.a<durante> El cliente selecciona tipo de entrega “retiro por sucursal”

2.a.1 El cliente selecciona sucursal. Sigue en paso 3.

3.a<durante> El cliente selecciona efectivo. Sigas en paso 3.

Confirmacion de productos y tipo de entrega/forma de pago

Home Categorías Ofertas Sucursales Ayuda Acerca de Login Carrito Mi cuenta Salir Dropdown link ▾

Supermarket
Productos de calidad a la mano

Descripción	Precio	Cantidad	Subtotal
Fideos	\$102.24	5	\$511.2
<input type="button" value="Eliminar"/>			

Precio total: \$511.2

Para realizar la compra debe tener por lo menos un producto

Tipo de entrega: Entrega a domicilio Retiro por sucursal

Forma de pago: Efectivo Tarjeta

Confirmacion de direccion de entrega y datos de tarjeta

Home Categorías Ofertas Sucursales Ayuda Acerca de Login Carrito Mi cuenta Salir Dropdown link ▾

Supermarket
Productos de calidad a la mano

Nombre	Código postal
Alvaro	2000
Email	Dirección de entrega
alvarito@gmail.com	Junin 2332
DNI	Fecha Nacimiento
12345678	06/24/1995
Teléfono	Número de tarjeta
3416512954	XXXX XXXX XXXX XXXX
Provincia	Tipo de tarjeta
Santa Fe	Visa
Localidad	CVV Mes/Año vencimiento
Rosario	XXX XX XXXX

Confirmacion de sucursal a elegir y datos de tarjeta

Home Categorías Ofertas Sucursales Ayuda Acerca de Login Carrito Mi cuenta Salir Dropdown link ▾

Supermarket
Productos de calidad a la mano

Nombre	Código postal
Alvaro	2000
Email	Sucursal
alvarito@gmail.com	Roca 2376, Capital Fedex
DNI	Fecha Nacimiento
12345678	06/24/1995
Teléfono	Número de tarjeta
3416512954	XXXX XXXX XXXX XXXX
Provincia	Tipo de tarjeta
Santa Fe	Visa
Localidad	CVV Mes/Año vencimiento
Rosario	XXX XX XXXX

