



UNIVERSIDAD POPULAR DEL CESAR

FACULTAD DE INGENIERIAS

PROGRAMA INGENIERÍA DE SISTEMAS

Proyecto de Aula

Asignatura:

Programación de Computadores III

Integrantes:

Oscar Daniel Duque Paz

Josheph Javier Martínez Tapias

Oscar Mauricio Parra Canabate

Docente:

Jhon Patiño

Valledupar – Cesar

2025

Tabla de contenido

1. Identificación y formulación del problema o necesidad.....	3
2. Estado del arte / Vigilancia tecnológica	4
3. Descripción de la solución.....	5
3.1. ¿Cuál es su solución?.....	4
3.2. ¿Por qué esta solución?	5
3.3. ¿Para que esta solución?.....	5
3.4. Factor Diferenciador	6
3.5. Lista de requerimientos funcionales de la aplicación.....	6
4. Definición de la arquitectura de paquetes	8
4.1. Diseño UML.....	10
5. Bibliografía.....	16

1. IDENTIFICACIÓN Y FORMULACIÓN DEL PROBLEMA O NECESIDAD

Problema o Necesidad	Causas	CONSECUENCIAS	AREA DE APLICACION
<p>Los pequeños emprendedores del sector de la moda en Colombia enfrentan grandes dificultades para gestionar sus ventas de ropa en línea de manera profesional. Actualmente, la mayoría depende exclusivamente de redes sociales como Instagram, Facebook y WhatsApp para comercializar sus productos, lo que genera una gestión desorganizada, sin control de inventario, sin registros históricos de ventas, sin facturación electrónica legal, y con una experiencia de compra deficiente para sus clientes.</p>	<p>Falta de plataforma profesional: Los emprendedores no tienen acceso a sistemas de e-commerce accesibles y adaptados a la venta de ropa.</p> <p>Gestión informal en redes sociales: Instagram y WhatsApp no ofrecen control de inventario, historial de ventas ni facturación legal.</p> <p>Desconocimiento técnico: Los emprendedores no saben cómo implementar soluciones tecnológicas para sus negocios.</p> <p>Costos prohibitivos: Plataformas como Shopify tienen costos mensuales que los pequeños negocios no pueden sostener.</p>	<p>Pérdida de ventas: Sin control de stock en tiempo real, venden prendas que ya no tienen disponibles.</p> <p>Desorganización financiera: No pueden saber cuánto venden realmente por día, semana o mes, dificultando la toma de decisiones.</p> <p>Experiencia deficiente: Clientes deben preguntar por WhatsApp disponibilidad, tallas, colores, y el proceso es lento y frustrante.</p> <p>Imposibilidad de escalar: Sin datos históricos y reportes, no pueden identificar productos exitosos ni planificar crecimiento.</p>	<p>El proyecto se enmarca en el ámbito del comercio electrónico de moda, específicamente dirigido a pequeños emprendedores y microempresas del sector textil en Colombia que desean profesionalizar sus operaciones de venta en línea.</p>

2. Estado Del Arte / Vigilancia Tecnológica.

En la actualidad, existen diversas plataformas de e-commerce especializadas en la venta de ropa. Sin embargo, la mayoría están diseñadas para grandes empresas con presupuestos elevados. A continuación, se presenta una comparación de las principales soluciones del mercado colombiano e internacional:

Funcionalidades	Falabella.com	Shein	Shopify
Catálogo con variantes (tallas/colores)	😊	😊	😊
Gestión de inventario en tiempo real	😊	😊	😊
Carrito de compras profesional	😊	😊	😊
Facturación electrónica DIAN	😊	😐	😐
Reportes detallados (diarios/mensuales/anuales)	😊	😊	😕
Sistema de cupones y descuentos	😊	😊	😊
Alertas de stock bajo	😊	😊	😕
Try-On Virtual (Prueba virtual de ropa)	😐	😐	😐
Notificaciones por correo automatizadas	😊	😊	😊
Accesible para pequeños emprendedores	😐	😕	😐

3. Descripción de la Solución

Clothix es una plataforma integral de e-commerce especializada en la venta de ropa, diseñada específicamente para pequeños emprendedores colombianos del sector moda que actualmente operan de manera informal a través de redes sociales como Instagram, Facebook y WhatsApp.

3.1. ¿Qué es la solución?

Se propone el desarrollo de un sistema completo de comercio electrónico enfocado en la profesionalización de negocios de moda emergentes, con énfasis en la automatización de procesos operativos y el cumplimiento de obligaciones legales. La plataforma permitirá la gestión de catálogo de prendas con variantes (tallas y colores), procesamiento de pedidos, facturación electrónica válida ante la DIAN, generación de reportes detallados, sistema de cupones promocionales, alertas de inventario y una innovadora funcionalidad de prueba virtual de ropa (Try-On) mediante realidad aumentada.

3.2. ¿Por qué es necesaria esta solución?

Actualmente, los pequeños emprendedores del sector moda en Colombia enfrentan serios obstáculos para profesionalizar sus operaciones de venta. La gran mayoría depende exclusivamente de redes sociales para comercializar sus productos, lo que genera múltiples problemáticas. Nuestro software busca resolver todos estos problemas de raíz, proporcionando una plataforma especializada que permita al emprendedor ser un aliado estratégico que acompaña su proceso de transformación digital y crecimiento empresarial.

3.3. ¿Para qué se implementará?

El sistema Clothix está diseñado para facilitar la gestión completa de negocios de moda mediante una plataforma integral, intuitiva y profesional. Los objetivos específicos de implementación son:

- Profesionalizar operaciones comerciales
- Optimizar la gestión de catálogo de ropa
- Generar inteligencia de negocio con reportes detallados
- Controlar inventario en tiempo real
- Aumentar conversión de ventas con innovación tecnológica
- Impulsar ventas con estrategias promocionales
- Automatizar comunicación con clientes
- Mejorar experiencia del cliente
- Escalar el negocio de manera sostenible

3.4. Factor diferenciador

Clothix se distingue por ofrecer una gestión integral y tecnológicamente avanzada del proceso comercial de venta de ropa. A diferencia de plataformas genéricas como Shopify o marketplace tradicionales como Falabella, Shein y Dafiti, Clothix integra de manera nativa la innovadora funcionalidad de Try-On Virtual mediante una IA, permitiendo a los clientes "probarse" virtualmente las prendas antes de comprar, una tecnología que ninguna competencia local o internacional de tamaño medio ofrece actualmente. Esta característica única, combinada con la facturación electrónica automática válida ante la DIAN, un sistema de reportes detallado capaz de generar análisis de cualquier período alertas inteligentes de stock por variante específica (talla y color), y un sistema avanzado de cupones promocionales.

3.5. Lista de requerimientos funcionales de la aplicación:

Requerimiento Funcional	Descripción
Control de acceso	Sistema de registro y autenticación. Dos roles: Cliente (compra ropa) y Administrador (emprendedor que gestiona la tienda).
Gestión completa de catálogo de ropa	CRUD de productos con variantes múltiples imágenes, categorías marca, precio, descripción, material. Control de stock independiente por variante.
Try-On Virtual	Tecnología de realidad aumentada que permite a los clientes probarse virtualmente las prendas antes de comprar. Visualización realista de cómo queda la prenda en diferentes tallas y colores.
Carrito de compras	Agregar prendas con selección de talla y color. Modificar cantidades. Eliminar productos. Cálculo automático de subtotal, IVA (19%) y total.
Sistema de cupones y descuentos	Creación y gestión de cupones (código, porcentaje/valor de descuento, fecha de vigencia, número de usos). Aplicación automática en checkout. Validación de vigencia y usos disponibles.
Gestión de direcciones de envío	Usuarios pueden registrar múltiples direcciones. Selección de ciudad de un catálogo colombiano. Dirección completa, barrio, código postal, referencia.
Lista de favoritos	Clientes pueden marcar prendas favoritas. Lista persistente de productos deseados. Acceso rápido para compra futura.

Facturación electrónica DIAN	Integración con API de Factus. Generación de factura electrónica válida. Almacenamiento de factura en sistema.
Notificaciones por correo	Al cliente: Confirmación de registro, confirmación de pedido, cambios de estado del pedido, confirmación de envío.
Sistema de alertas de stock	Monitoreo automático de niveles de inventario. Generación de alertas cuando una variante (talla/color específico) está por agotarse (umbral configurable). Panel de gestión de alertas para el administrador. Notificación por correo.
Sistema de reportes en PDF	Generación de reportes personalizados en PDF
Mi perfil (Cliente)	Ver y editar información personal. Gestionar direcciones de envío. Ver historial de compras. Ver cupones disponibles. Cambiar contraseña.

4. Definición de la arquitectura (Diagrama de paquetes)

Clothix implementa una arquitectura en capas profesional que separa responsabilidades y facilita el mantenimiento, escalabilidad y estabilidad del sistema. La plataforma está construida con tecnologías modernas de Microsoft (Blazor Server para la interfaz de usuario, C# como backend, y Oracle Database 19c como motor de base de datos empresarial), implementando patrones de diseño reconocidos como Repository Pattern (a través de los DAOs), Data Transfer Object (DTO) Pattern para la transferencia de datos entre capas, Service Layer Pattern para centralizar la lógica de negocio, y Dependency Injection para el desacoplamiento de componentes. Esta arquitectura multicapa garantiza que cada componente tenga una responsabilidad única y bien definida, permitiendo que cambios en una capa no afecten a las demás, facilitando así el mantenimiento evolutivo y la incorporación de nuevas funcionalidades sin comprometer la estabilidad del sistema.

El proyecto está dividido en cuatro paquetes principales:

Nombre Paquete	Función
ENTITY	Contiene las clases que representan el modelo de dominio del negocio, es decir, las entidades que encapsulan los datos y comportamientos fundamentales del e-commerce de moda. Entre estas clases se encuentran los objetos que representan Usuario, Artículo, Pedido, Factura, etc. Este paquete es responsable de definir la estructura y reglas básicas de los datos manejados por la aplicación, incluyendo validaciones de atributos y relaciones entre entidades. Además, aquí se definen los DTOs utilizados para transferir información entre capas sin exponer directamente las entidades de base de datos.
DAL (Data Access Layer)	Agrupa las clases encargadas de la persistencia y acceso a datos. Aquí se implementan los DAOs (Data Access Objects) que siguen el patrón Repository, encapsulando toda la lógica de comunicación con la base de datos Oracle. Cada DAO es responsable de ejecutar los procedimientos almacenados definidos en los paquetes PL/SQL de Oracle, mapear los resultados a DTOs y manejar las transacciones de base de datos. Esta separación permite abstraer la tecnología de almacenamiento utilizada, facilitando futuras migraciones o cambios en la infraestructura de datos sin impactar la lógica de negocio.
BLL (Business Logic Layer)	Incluye la capa de servicios que actúa como intermediaria entre la capa de presentación (GUI) y la capa de acceso a datos (DAL). En esta capa se centralizan todas las reglas de negocio, validaciones complejas y operaciones que requieren coordinación entre múltiples repositorios. Los servicios son responsables de validar datos de entrada, aplicar lógica de negocio, coordinar operaciones entre múltiples DAOs, y encapsular integraciones externas. Esta capa facilita la reutilización de la lógica para diferentes tipos de interfaces o clientes futuros.

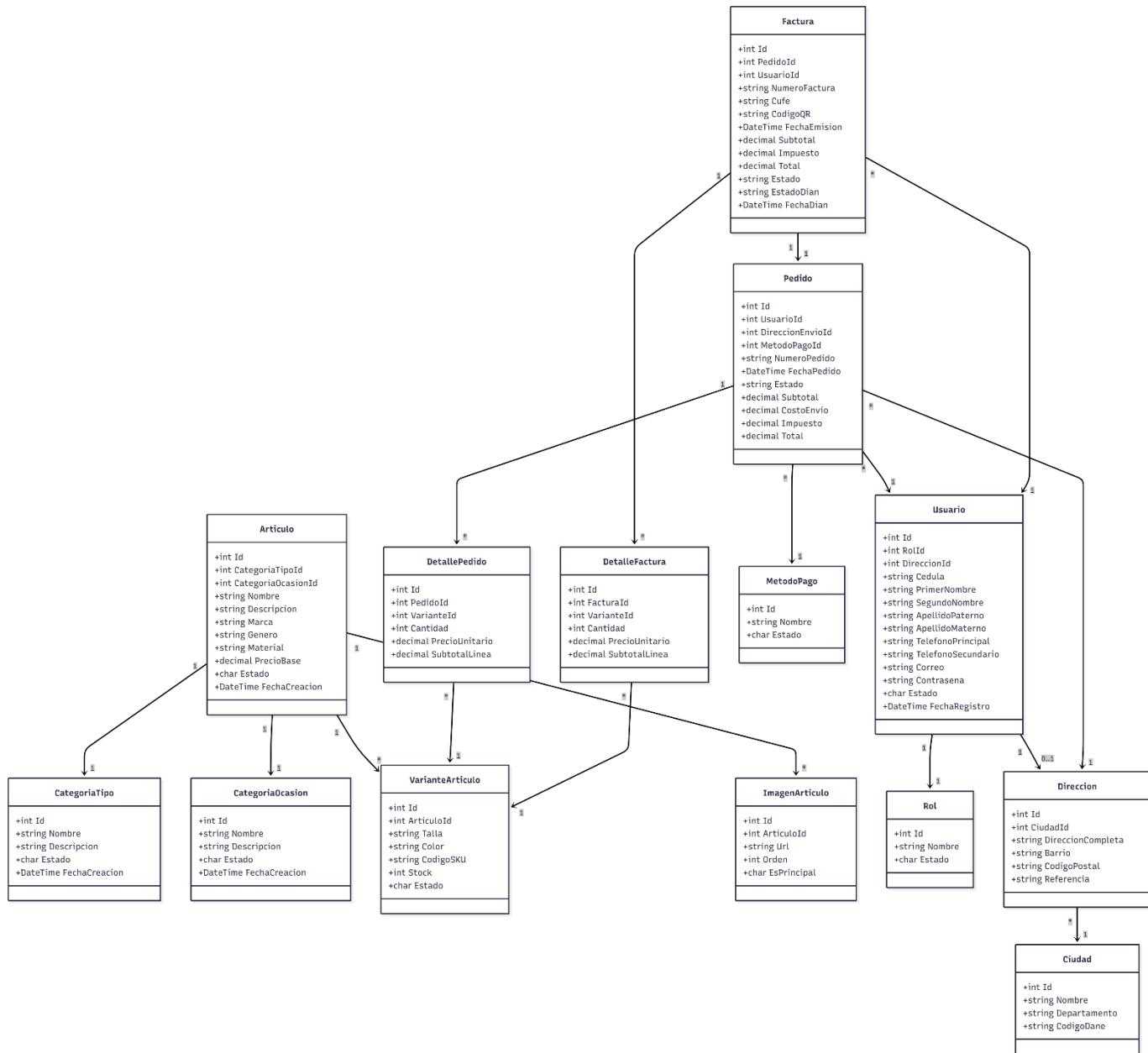
GUI (Graphical User Interface)	<p>Contiene los componentes de interfaz de usuario desarrollados con Blazor Server utilizando el patrón MVVM (Model-View-ViewModel). Aquí se gestionan todas las páginas razor que interactúan con el usuario final, tanto del lado del cliente como del lado del administrador/emprendedor. Cada componente Blazor inyecta los servicios necesarios de la capa BLL mediante Dependency Injection, maneja eventos generados por el usuario, actualiza el estado reactivo de la aplicación, y renderiza dinámicamente la interfaz según el rol del usuario autenticado. Esta división asegura que los detalles de presentación y experiencia de usuario no se mezclen con la lógica del negocio ni con la persistencia de datos, manteniendo una separación limpia de responsabilidades.</p>
--------------------------------	---

Ventajas de la arquitectura

Esta organización en capas aporta múltiples beneficios al proyecto CLOTHIX:

- **Separación clara de responsabilidades:** Cada capa tiene un propósito específico y bien definido, lo que mejora la legibilidad del código y facilita la colaboración entre desarrolladores, permitiendo que diferentes miembros del equipo trabajen en capas distintas sin generar conflictos.
- **Escalabilidad y flexibilidad:** Cada capa puede evolucionar de forma independiente. Por ejemplo, se puede cambiar la interfaz de usuario de Blazor a React sin afectar la lógica de negocio, o migrar de Oracle a SQL Server modificando únicamente la capa DAL sin tocar BLL ni GUI.
- **Reutilización de componentes y lógica:** La capa de servicios (BLL) puede ser consumida por diferentes interfaces: la actual aplicación web Blazor, una futura aplicación móvil, o incluso una API REST pública, maximizando el retorno de inversión del código desarrollado.
- **Facilidad para pruebas unitarias y de integración:** Al estar los componentes desacoplados mediante interfaces y dependency injection, es posible crear mocks de cada capa para realizar pruebas aisladas, mejorando significativamente la calidad y confiabilidad del software.
- **Mantenibilidad a largo plazo:** Los cambios en requisitos de negocio se localizan en capas específicas (generalmente BLL), reduciendo el riesgo de introducir bugs en otras partes del sistema y facilitando la incorporación de nuevas funcionalidades.
- **Cumplimiento con principios SOLID y patrones de diseño reconocidos:** La arquitectura implementa Single Responsibility Principle (cada clase tiene una responsabilidad), Dependency Inversion Principle (dependencias hacia abstracciones, no implementaciones concretas), Repository Pattern, DTO Pattern, Service Layer Pattern, todos ampliamente aceptados en el desarrollo de aplicaciones empresariales robustas y escalables.

4.1. Diagrama UML



El diagrama de clases representa la estructura estática del sistema Clothix, mostrando las entidades principales, sus atributos y las relaciones entre ellas. El sistema está organizado en cinco módulos que trabajan de forma integrada para gestionar todo el proceso de venta de ropa en línea.

El módulo de Artículos define los productos disponibles en la tienda con información base como nombre, marca, género, material y precio. Cada artículo pertenece a una CategoríaTipo que indica el tipo de prenda (camisa, pantalón, vestido, etc.) y una CategoríaOcasión que define para qué ocasión se usa (casual, formal, deportiva, etc.). Los artículos tienen múltiples VarianteArtículo que representan las combinaciones de talla, color y stock disponible, además de ImagenArtículo que almacena las fotografías del producto con un orden específico.

El módulo de Usuarios gestiona la información de las personas registradas en el sistema. Cada usuario tiene un Rol asignado que define sus permisos de acceso (Cliente o Administrador). Los usuarios almacenan datos personales como cédula, nombres, apellidos, teléfonos y correo electrónico, y pueden tener una Dirección asociada para facilitar el proceso de compra.

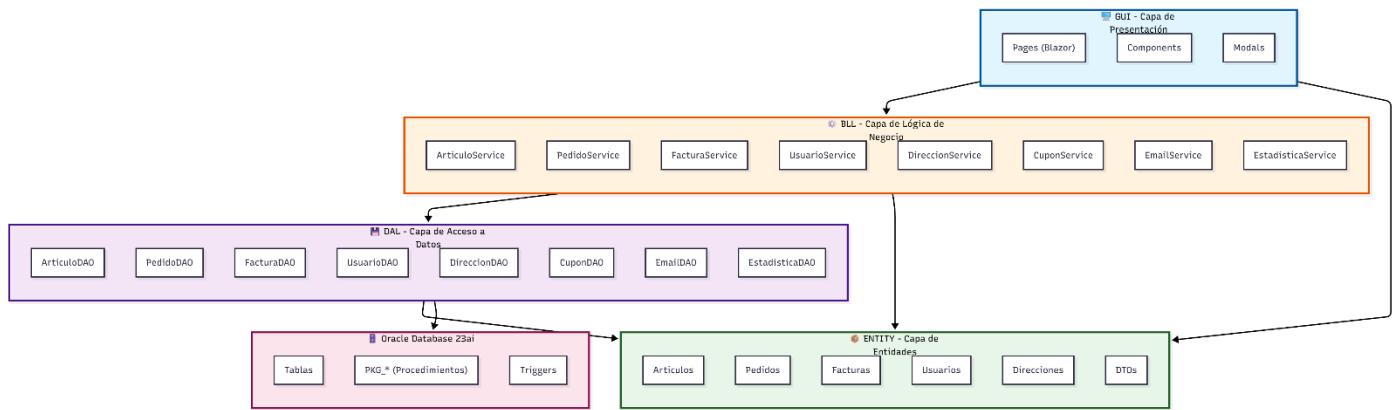
El módulo de Direcciones almacena las direcciones de envío de los pedidos. Cada Dirección está vinculada a una Ciudad que contiene información del departamento y código DANE, lo cual es importante para el cálculo de costos de envío y para el cumplimiento de requisitos de facturación electrónica en Colombia.

El módulo de Pedidos controla todo el flujo de compra desde que el cliente agrega productos al carrito hasta la confirmación final. Un Pedido incluye información del Usuario que lo realiza, la DirecciónEnvío, el MétodoPago seleccionado y los montos calculados (subtotal, costo de envío, impuesto y total). Los DetallePedido registran cada producto comprado, haciendo referencia a la VarianteArtículo específica con su cantidad y precio unitario al momento de la compra.

El módulo de Facturación genera facturas electrónicas vinculadas a cada pedido para cumplir con los requisitos tributarios colombianos. Cada Factura contiene el CUFE (Código Único de Facturación Electrónica), el código QR, y registra el estado tanto interno del sistema como el estado reportado por la DIAN. Los DetalleFactura replican la información de los productos facturados para mantener la trazabilidad completa de cada transacción.

Las relaciones principales entre las entidades son: un Artículo tiene muchas Variantes (1:N), un Pedido pertenece a un Usuario (N:1), un Pedido tiene muchos DetallePedido (1:N), una Factura está asociada a un único Pedido (1:1), y tanto DetallePedido como DetalleFactura referencian VarianteArtículo para mantener el control de stock y la consistencia de los datos.

4.2. Diagrama de paquetes



La capa GUI (Interfaz Gráfica de Usuario) es la capa de presentación construida con Blazor Server. Esta capa contiene las páginas web (Pages) que el usuario ve en su navegador, los componentes reutilizables (Components) como tarjetas de productos o ítems del carrito, y las ventanas modales (Modals) para interacciones específicas. Esta capa es responsable únicamente de mostrar información al usuario y capturar sus acciones, sin contener lógica de negocio. Para ejecutar operaciones depende de la capa BLL y utiliza la capa ENTITY para trabajar con los modelos de datos.

La capa BLL (Business Logic Layer) implementa toda la lógica de negocio del sistema. Aquí se encuentran los servicios que realizan validaciones, calculan el IVA, aplican cupones de descuento, procesan pagos y manejan las reglas específicas del negocio. Los servicios principales son ArticuloService, PedidoService, FacturaService, UsuarioService, DireccionService, CuponService, EmailService y EstadisticaService. Esta capa actúa como intermediario entre la GUI y el DAL, orquestando operaciones complejas que pueden involucrar múltiples consultas a la base de datos o llamadas a servicios externos como el API de FACTUS para facturación electrónica.

La capa DAL (Data Access Layer) es responsable de la comunicación directa con la base de datos Oracle. Cada DAO (Data Access Object) encapsula las operaciones CRUD de su módulo correspondiente, como ArticuloDAO, PedidoDAO, FacturaDAO, etc. Esta capa se encarga de mapear los datos entre los DTOs que usa el sistema y los parámetros que requieren los procedimientos almacenados de Oracle, además de convertir los resultados devueltos por la base de datos en objetos que las capas superiores puedan utilizar.

La capa ENTITY define las clases de dominio y los DTOs (Data Transfer Objects) para transferencia de datos. Las clases de dominio representan las entidades del negocio como Articulo, Pedido, Factura, Usuario y Direccion, mientras que los DTOs son versiones simplificadas o especializadas de estas entidades diseñadas para transferir datos entre capas. Esta capa es compartida por todas las demás capas para mantener consistencia en los modelos de datos. Es importante destacar que esta capa no contiene lógica de negocio, solo define la estructura de los datos.

La base de datos Oracle 23ai actúa como la capa de persistencia del sistema. Contiene las tablas que almacenan toda la información, los procedimientos almacenados organizados en paquetes (PKG_ARTICULOS, PKG_PEDIDOS, PKG_FACTURACION, etc.) que encapsulan la lógica de base de datos, y los triggers que automatizan ciertas operaciones como la generación de números de pedido o el registro de auditoría. Los procedimientos almacenados concentran operaciones importantes como validaciones de stock, cálculos de totales y manejo de transacciones complejas.

El flujo de dependencias en el sistema es unidireccional: GUI depende de BLL, BLL depende de DAL, y DAL se comunica con Oracle Database. Todas las capas utilizan ENTITY para transferir datos. Este diseño respeta el principio de bajo acoplamiento ya que cada capa solo conoce la capa inmediatamente inferior, lo que facilita el mantenimiento y permite realizar cambios en una capa sin afectar a las demás. Los principios SOLID aplicados incluyen el Single Responsibility Principle donde cada capa tiene una única responsabilidad claramente definida, y el Dependency Inversion Principle donde las capas superiores dependen de abstracciones mediante interfaces en lugar de implementaciones concretas.

Bibliografía

1. Microsoft. (2025). *Blazor Server Documentation*. Recuperado de <https://learn.microsoft.com/en-us/aspnet/core/blazor/>
2. Oracle. (2025). *Oracle Database 19c Documentation*. Recuperado de <https://docs.oracle.com/en/database/oracle/oracle-database/19/>
3. Factus. (2025). *API de Facturación Electrónica - Documentación*. Recuperado de <https://factus.co/api-documentacion>
4. Shopify. (2025). *E-commerce Platform Comparison - Fashion Industry*. Recuperado de <https://www.shopify.com/retail/fashion-ecommerce>
5. Falabella.com. (2025). *Plataforma de E-commerce Colombia*. Recuperado de <https://www.falabella.com.co/>
6. Shein. (2025). *Global Fashion E-commerce Platform*. Recuperado de <https://www.shein.com/>

