UNIVERSIDAD TECNOLOGICA NACIONAL

# Tecnicatura Universitaria en Programación

## Universidad Tecnológica Nacional

**TRABAJO FINAL INTEGRADOR**

**PROGRAMACIÓN II**

*Coordinador*

*Carlos Martínez*

*Profesora*

*Cinthia Rigoni*

*Tutor*

*Emmanuel Avellaneda*

*Alumnos y roles:*

*Cornejo, Diego Gastón (Desarrollador)*

*Cuquejo, Mauro Maximiliano (Líder técnico y Gestión de Repositorio)*

*Dantur, Daniel Fernando Jesús (Desarrollador)*

*Díaz de Quintana, Melisa Magalí (Desarrolladora)*

*Comisión: Ag25-2C-12*

*Grupo 175 – Pedido-Envio*

*Fecha de Entrega: Noviembre 2025*

# ÍNDICE

## Resumen Ejecutivo

El presente informe detalla el diseño, desarrollo e implementación de un programa para la gestión de **Pedidos y Envíos** para un e-commerce. El proyecto consiste en una aplicación de consola, desarrollada en lenguaje Java, diseñada para gestionar de forma integral el ciclo de vida de pedidos y envíos de una empresa ficticia.

El objetivo del proyecto es desarrollar un sistema de gestión, en este caso de pedidos y envíos, implementando arquitectura en capas, patrones de diseño y buenas prácticas de programación orientada a objetos con persistencia en base de datos MySQL. La conexión y manipulación de los datos desde Java se realiza mediante la API de JDBC (Java Database Connectivity) y la implementación del patrón de diseño DAO (Data Access Object), garantizando que la lógica de negocio permanezca independiente de la lógica de acceso a datos.

## Elección y Justificación del Dominio

Se seleccionó el dominio de un **Sistema de Gestión de Pedidos** (logística) debido a su alta relevancia en el mundo actual y su idoneidad para aplicar los conceptos clave de la Programación Orientada a Objetos y la persistencia de datos.

Este dominio presenta un escenario ideal para modelar un problema de negocio de complejidad media, permitiéndonos implementar:

1. **Modelo de Entidades Claras:** Clases como Pedido, Envio, EmpresaEnvio y EstadoPedido, con atributos y relaciones bien definidas.
2. **Lógica de Negocio Compleja:** Reglas para la creación de pedidos, validaciones de datos (ej. tracking no nulo), y la gestión de cambios de estado (ej. de "En tránsito" a "Entregado").
3. **Persistencia Transaccional:** La necesidad de guardar el estado de un envío en una base de datos (MySQL) lo convierte en un caso de estudio óptimo para practicar la arquitectura por capas, el patrón DAO (Data Access Object) y el manejo de excepciones SQL.

El dominio nos permite construir una aplicación robusta que simula un proceso de negocio real, integrando diseño de clases, lógica de servicios y conectividad a bases de datos.

## Arquitectura

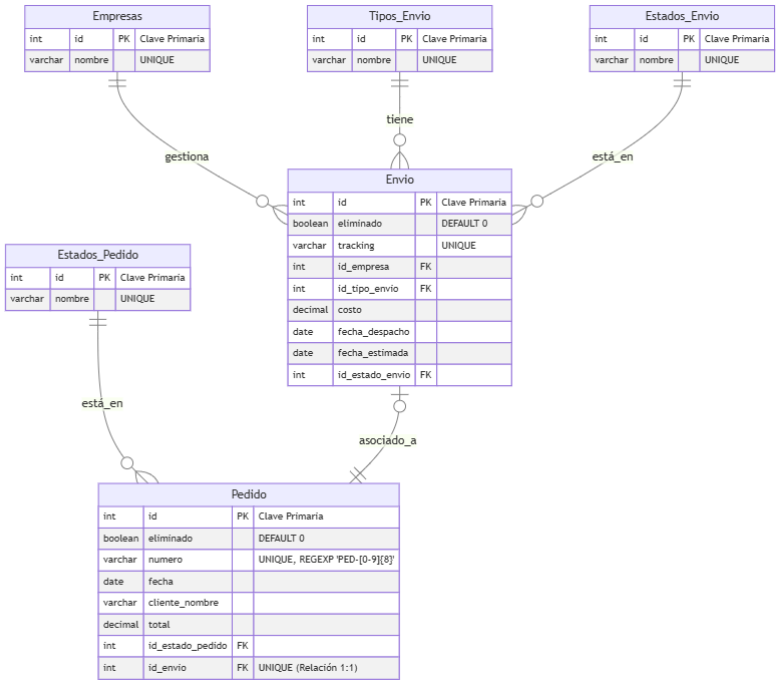
El diseño del sistema se centra en una arquitectura por capas desacoplada, facilitando la mantenibilidad, escalabilidad y la separación de responsabilidades.

Decisiones Clave de Diseño de la base de datos:

El núcleo del dominio gira en torno a la relación entre Pedidos y Envíos.

1. **Relación Pedido-Envío (1 a 0..1):** La decisión de diseño fundamental es que el sistema se centra en la **Gestión de Pedidos**. Un Pedido se crea inicialmente y existe por sí mismo. El Envio es una entidad opcional que se crea *después* y se asocia a un pedido existente. Un pedido no puede tener más de un envío.
2. **Implementación (Foreign Key):** Para modelar esto, la tabla envios contiene una Clave Foránea (id_pedido) que referencia a la Clave Primaria (id) de la tabla pedidos. Esta columna (id_pedido) en la tabla envios se define como **ÚNICA (UNIQUE)**, garantizando así que un pedido solo pueda estar asociado a un envío, cumpliendo la relación 1:1.

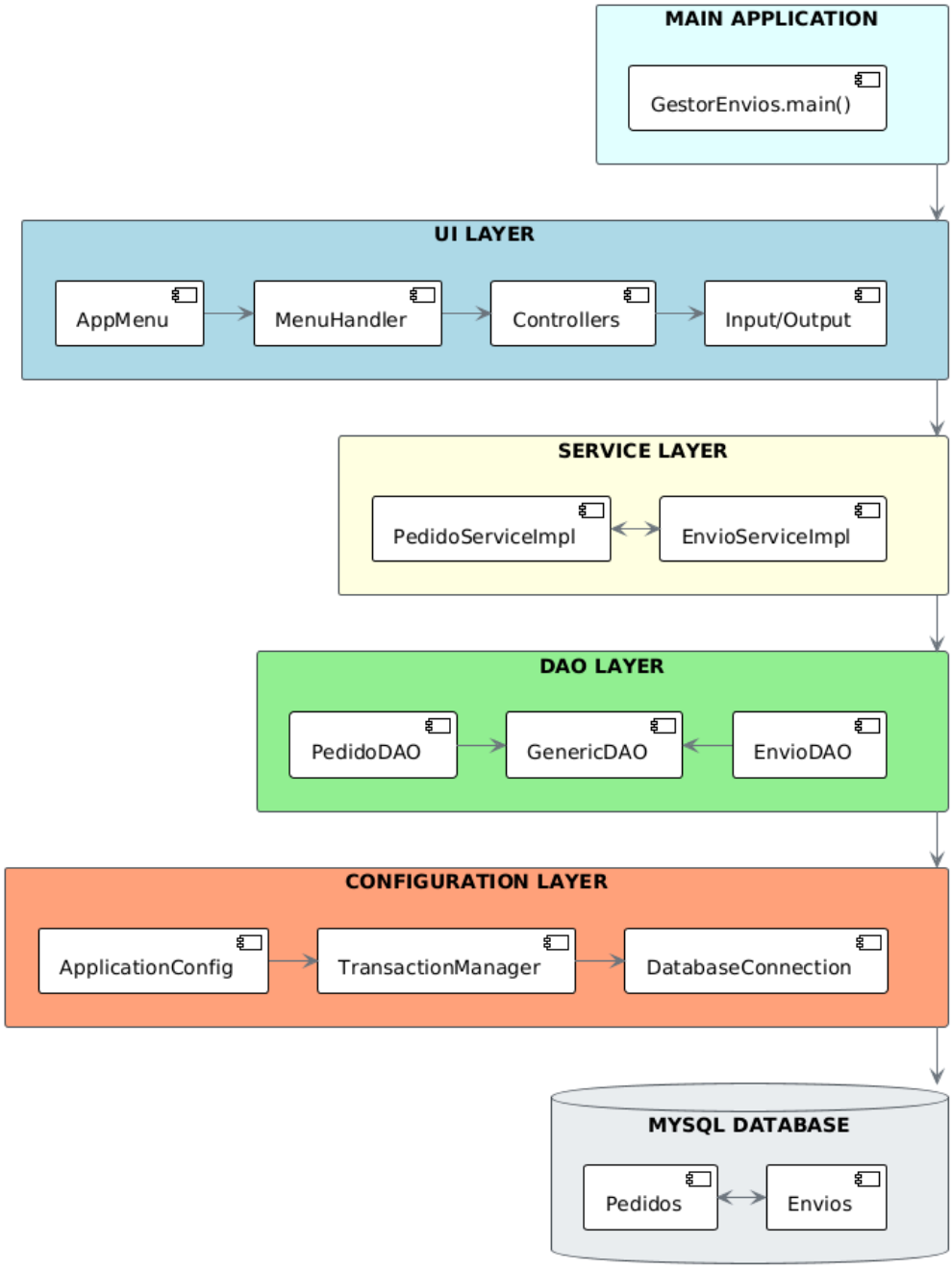
Diagrama Entidad Relación (DER):



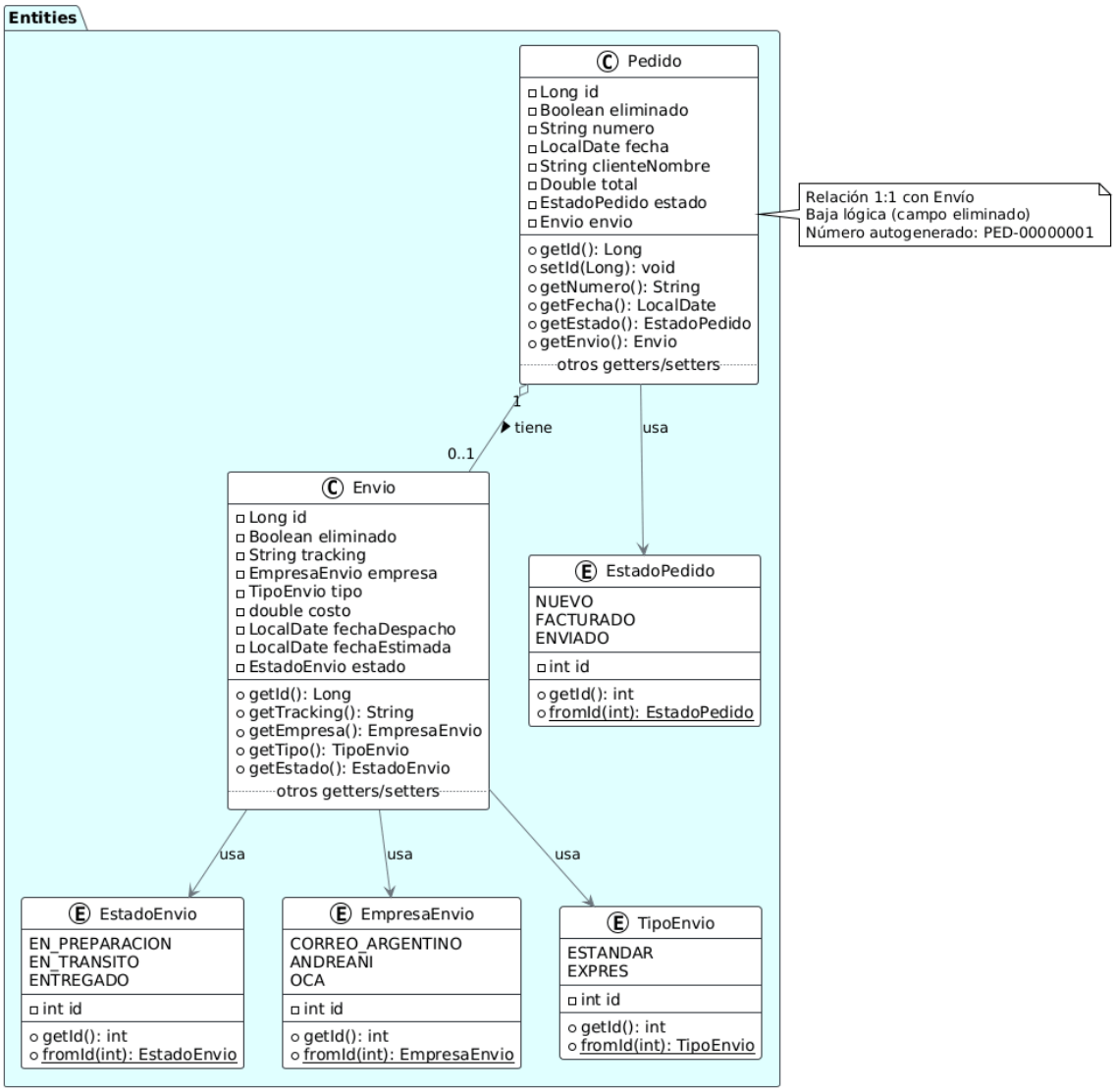
## Arquitectura por Capas

El proyecto sigue una arquitectura de software organizada por paquetes, donde cada uno tiene una responsabilidad claramente definida.

Vista simplificada por capas:

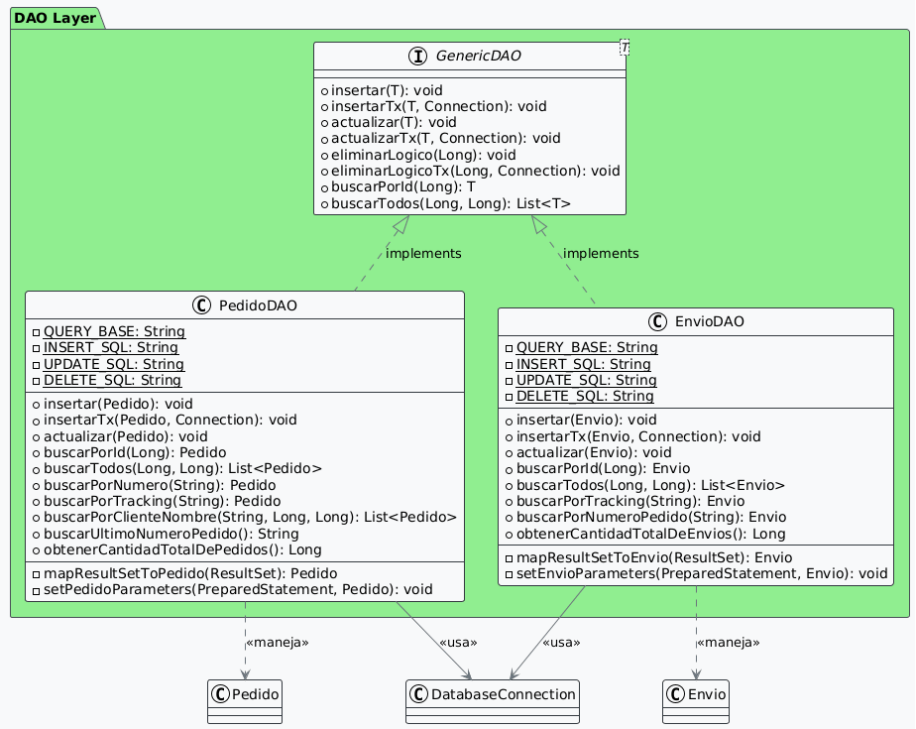


- **Capa de Dominio:**
  - Responsabilidad: Contiene las clases DTO que modelan los datos del problema.
  - Clases clave: Pedido, Envio, EmpresaEnvio, EstadoPedido, TipoEnvio. Estas clases no contienen lógica de negocio ni acceso a datos, solo atributos, constructores, getters y setters.



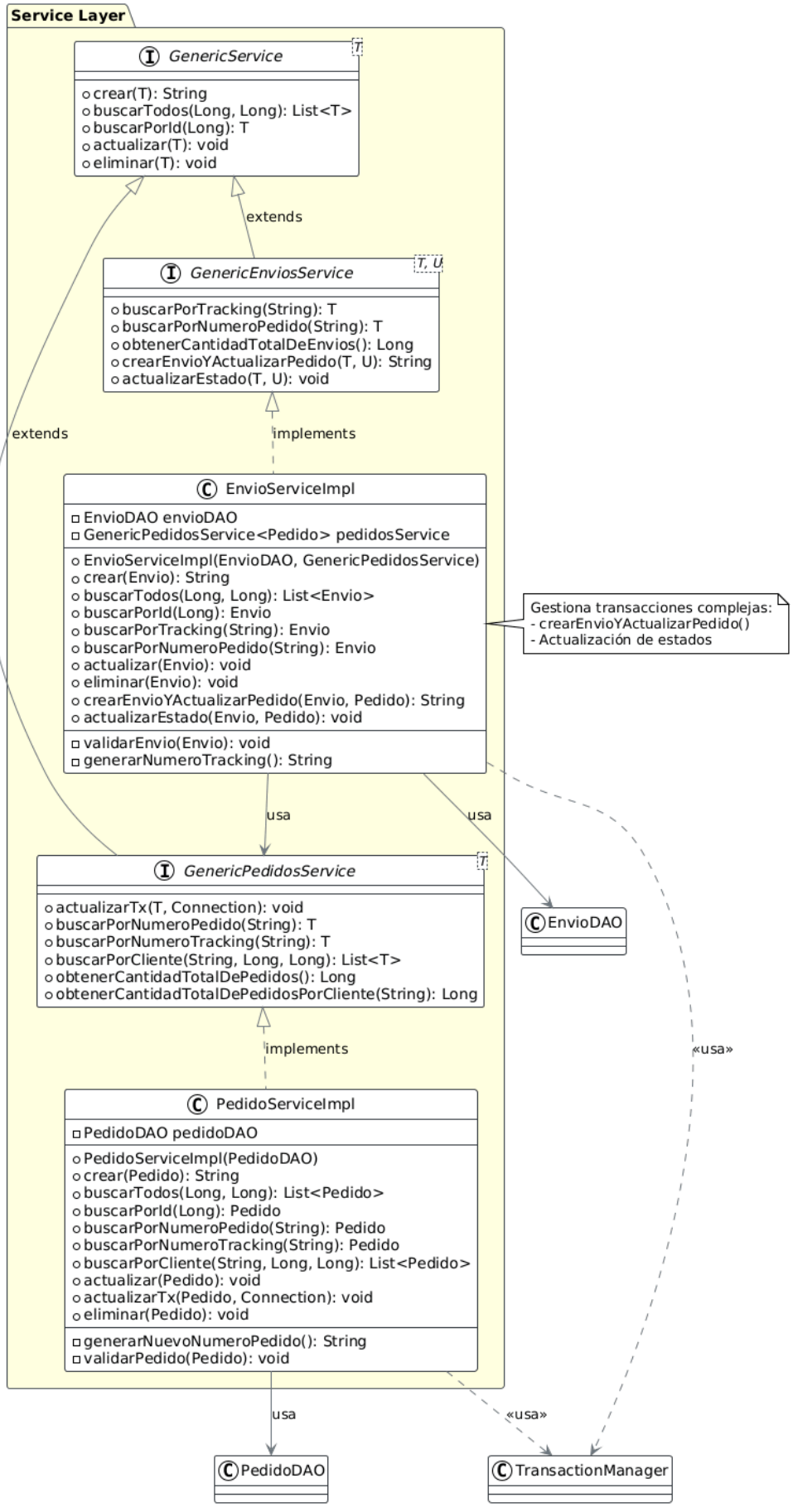
```
entities/
├── Pedido.java              # Entidad principal con relación 1:1 a Envío
├── Envio.java               # Entidad de envío
├── EstadoPedido.java        # 3 estados posibles
├── EstadoEnvio.java         # 3 estados posibles
├── EmpresaEnvio.java        # 3 empresas disponibles
└── TipoEnvio.java           # 2 tipos de servicio
```

- **Capa de Acceso a Datos - DAO:**
  - Responsabilidad**:** Abstraer la lógica de persistencia. Define qué operaciones se pueden realizar sobre la base de datos (CRUD) sin exponer cómo se hacen.
  - Clases clave: GenericDAO.java (interfaz genérica), PedidoDAO.java, EnvioDAO.java, clases que implementan la interfaz genérica.



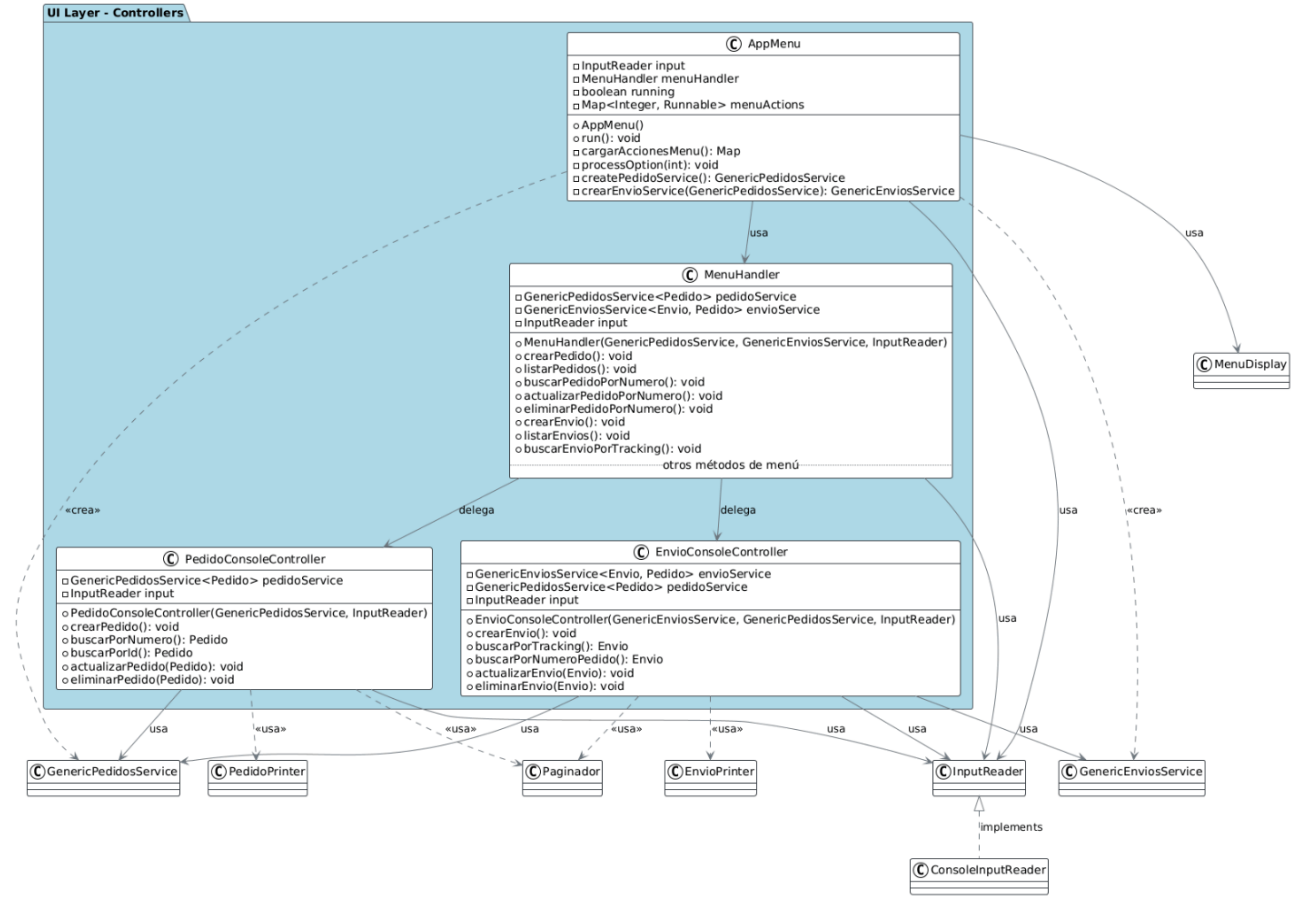
```
dao/
├── GenericDAO.java          # Interfaz con operaciones CRUD + Tx
├── PedidoDAO.java           # 12+ métodos de consulta
└── EnvioDAO.java            # 10+ métodos de consulta
```

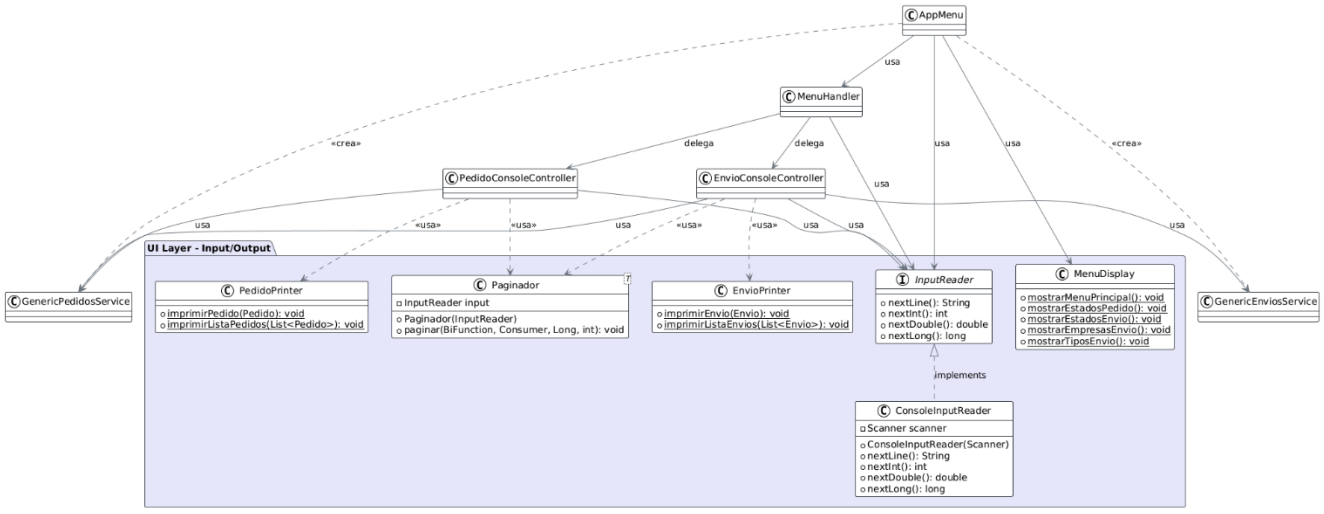
- **Capa de Lógica de Negocio:**
    - Responsabilidad: Orquestar las operaciones y aplicar las reglas de negocio. Esta capa decide cuándo y cómo llamar al DAO y maneja las transacciones.
    - Clases clave: PedidoServiceImpl.java, EnvioServiceImpl.java. Implementan las interfaces GenericPedidosService.java. y GenericEnviosService.java, respectivamente, y que estas últimas a su vez extienden de GenericService. Realizamos esta separación con el fin de desacoplar el contrato de operaciones básicas de los contratos especializados para envíos y pedidos.


Service Layer
GenericService
T
crear(T): String
buscarTodos(Long, Long): List<T>
buscarPorId(Long): T
actualizar(T): void
eliminar(T): void
extends
GenericEnviosService
T, U
buscarPorTracking(String): T
buscarPorNumeroPedido(String): T
obtenerCantidadTotalDeEnvios(): Long
crearEnvioYActualizarPedido(T, U): String
actualizarEstado(T, U): void
extends
implements
EnvioServiceImpl
EnvioDAO envioDAO
GenericPedidosService<Pedido> pedidosService
EnvioServiceImpl(EnvioDAO, GenericPedidosService)
crear(Envio): String
buscarTodos(Long, Long): List<Envio>
buscarPorId(Long): Envio
buscarPorTracking(String): Envio
buscarPorNumeroPedido(String): Envio
actualizar(Envio): void
eliminar(Envio): void
crearEnvioYActualizarPedido(Envio, Pedido): String
actualizarEstado(Envio, Pedido): void
validarEnvio(Envio): void
generarNumeroTracking(): String
Gestiona transacciones complejas:
- crearEnvioYActualizarPedido()
- Actualización de estados
usa
usa
GenericPedidosService
T
actualizarTx(T, Connection): void
buscarPorNumeroPedido(String): T
buscarPorNumeroTracking(String): T
buscarPorCliente(String, Long, Long): List<T>
obtenerCantidadTotalDePedidos(): Long
obtenerCantidadTotalDePedidosPorCliente(String): Long
EnvioDAO
«usa»
implements
PedidoServiceImpl
PedidoDAO pedidoDAO
PedidoServiceImpl(PedidoDAO)
crear(Pedido): String
buscarTodos(Long, Long): List<Pedido>
buscarPorId(Long): Pedido
buscarPorNumeroPedido(String): Pedido
buscarPorNumeroTracking(String): Pedido
buscarPorCliente(String, Long, Long): List<Pedido>
actualizar(Pedido): void
actualizarTx(Pedido, Connection): void
eliminar(Pedido): void
generarNuevoNumeroPedido(): String
validarPedido(Pedido): void
usa
«usa»
PedidoDAO
TransactionManager

```
services/
├── GenericService.java             # CRUD básico
├── GenericPedidosService.java      # Servicios específicos pedidos
├── GenericEnviosService.java       # Servicios específicos envíos
├── PedidoServiceImpl.java          # Lógica de negocio + validaciones
└── EnvioServiceImpl.java           # Transacciones complejas
```

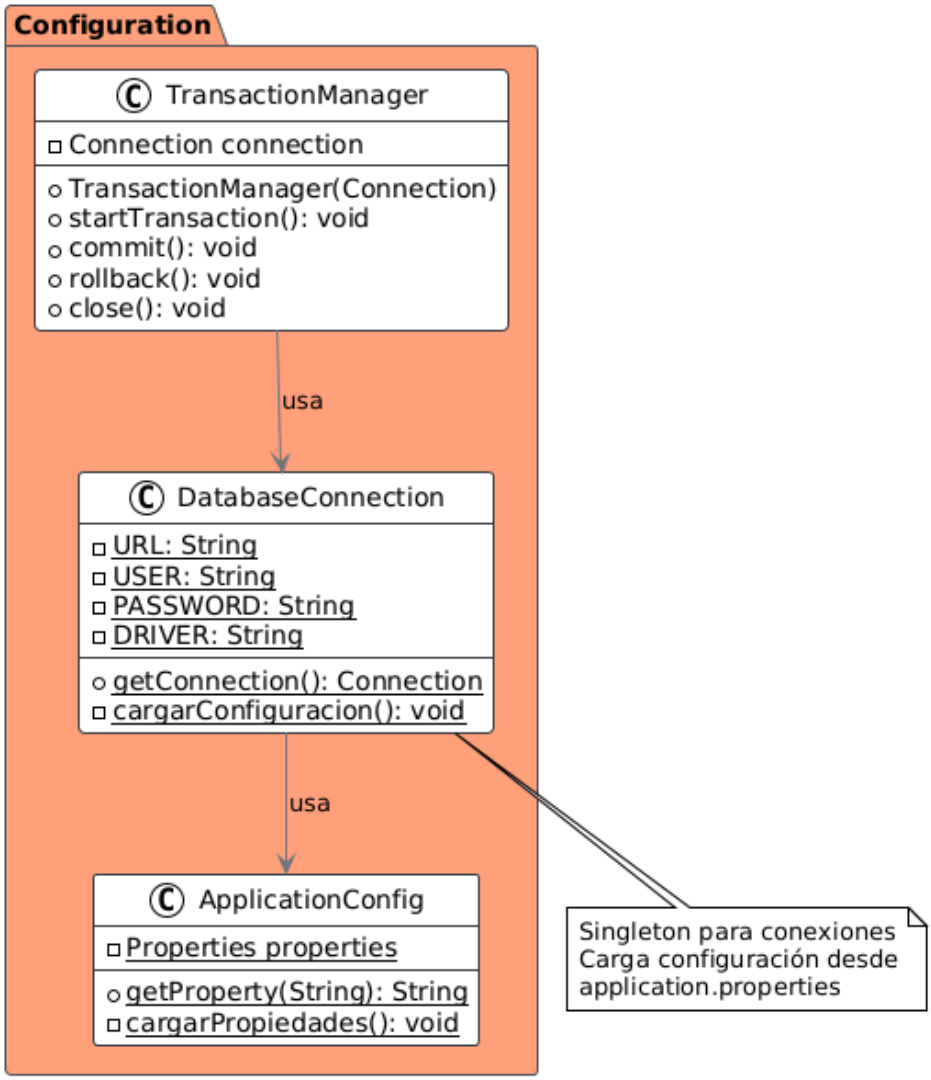
- **Capa de Presentación / Vista:**
    - Responsabilidad: Gestionar toda la interacción visual con el usuario en la consola.
    - Clases clave: AppMenu.java (maneja el bucle principal) y se apoya en sub-paquetes:
        - **ui.console.controllers:** Contiene PedidoConsoleController y EnvioConsoleController, que actúan como puente entre la vista (menú) y la capa de servicios (services).
        - **ui.console.input:** Clases utilitarias (ConsoleInputReader, MenuDisplay) para leer datos del usuario de forma segura.
        - **ui.console.output:** Clases (EnvioPrinter, PedidoPrinter) dedicadas a formatear y mostrar las entidades en la consola.
        - **ui.console.utils:** Contiene utilitarios como el Paginador.java para las listas.


UI Layer - Controllers
AppMenu
- InputReader input
- MenuHandler menuHandler
- boolean running
- Map<Integer, Runnable> menuActions
+ AppMenu()
+ run(): void
- cargarAccionesMenu(): Map
- processOption(int): void
- createPedidoService(): GenericPedidosService
- crearEnvioService(GenericPedidosService): GenericEnviosService
MenuHandler
- GenericPedidosService<Pedido> pedidoService
- GenericEnviosService<Envio, Pedido> envioService
- InputReader input
+ MenuHandler(GenericPedidosService, GenericEnviosService, InputReader)
+ crearPedido(): void
+ listarPedidos(): void
+ buscarPedidoPorNumero(): void
+ actualizarPedidoPorNumero(): void
+ eliminarPedidoPorNumero(): void
+ crearEnvio(): void
+ listarEnvios(): void
+ buscarEnvioPorTracking(): void
otros métodos de menú
PedidoConsoleController
- GenericPedidosService<Pedido> pedidoService
- InputReader input
+ PedidoConsoleController(GenericPedidosService, InputReader)
+ crearPedido(): void
+ buscarPorNumero(): Pedido
+ buscarPorId(): Pedido
+ actualizarPedido(Pedido): void
+ eliminarPedido(Pedido): void
EnvioConsoleController
- GenericEnviosService<Envio, Pedido> envioService
- GenericPedidosService<Pedido> pedidoService
- InputReader input
+ EnvioConsoleController(GenericEnviosService, GenericPedidosService, InputReader)
+ crearEnvio(): void
+ buscarPorTracking(): Envio
+ buscarPorNumeroPedido(): Envio
+ actualizarEnvio(Envio): void
+ eliminarEnvio(Envio): void
MenuDisplay
GenericPedidosService
PedidoPrinter
Paginador
EnvioPrinter
InputReader
GenericEnviosService
ConsoleInputReader
usa
«crea»
delega
«usa»
implements


AppMenu
MenuHandler
PedidoConsoleController
EnvioConsoleController
UI Layer - Input/Output
GenericPedidosService
PedidoPrinter
- imprimirPedido(Pedido): void
- imprimirListaPedidos(List<Pedido>): void
Paginador<T>
- InputReader input
+ Paginador(InputReader)
+ paginar(BiFunction, Consumer, Long, int): void
EnvioPrinter
- imprimirEnvio(Envio): void
- imprimirListaEnvios(List<Envio>): void
InputReader
+ nextLine(): String
+ nextInt(): int
+ nextDouble(): double
+ nextLong(): long
MenuDisplay
+ mostrarMenuPrincipal(): void
+ mostrarEstadosPedido(): void
+ mostrarEstadosEnvio(): void
+ mostrarEmpresasEnvio(): void
+ mostrarTiposEnvio(): void
GenericEnviosService
ConsoleInputReader
- Scanner scanner
+ ConsoleInputReader(Scanner)
+ nextLine(): String
+ nextInt(): int
+ nextDouble(): double
+ nextLong(): long
usa
«crea»
delega
«usa»
implements

```
ui/console/
├── AppMenu.java                    # Loop principal
├── controllers/
│   ├── MenuHandler.java            # 22+ métodos de menú
│   ├── PedidoConsoleController.java # CRUD de pedidos
│   └── EnvioConsoleController.java  # CRUD de envíos
├── input/
│   ├── InputReader.java            # Interfaz
│   ├── ConsoleInputReader.java     # Implementación
│   └── MenuDisplay.java            # Formateo de menús
├── output/
│   ├── PedidoPrinter.java          # Pretty print pedidos
│   └── EnvioPrinter.java           # Pretty print envíos
└── utils/
    └── Paginador.java              # Genérico reutilizable
```

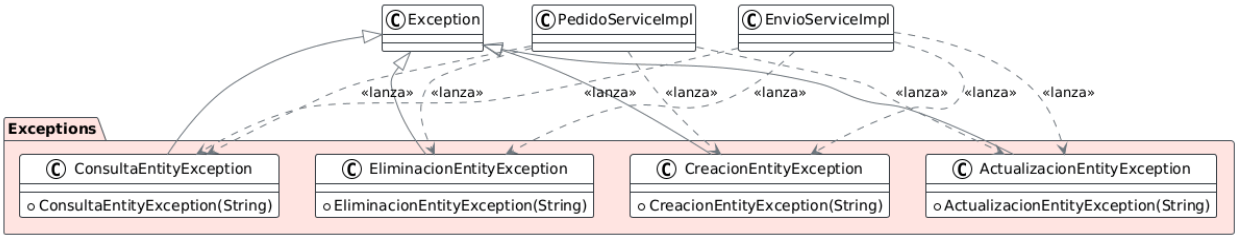
- **Configuración:**
    - Responsabilidad: Gestionar la configuración externa y el ciclo de vida de la conexión.
    - Clases clave: ApplicationConfig.java (lee .properties), DatabaseConnection.java (obtiene la conexión JDBC) y TransactionManager.java (clase dedicada a gestionar el commit y rollback).




```
config/
├── ApplicationConfig.java          # Singleton de propiedades
├── DatabaseConnection.java         # Pool de conexiones
└── TransactionManager.java         # ACID transactions
```

- **exceptions.envios y exceptions.pedidos (Manejo de Errores):**
    - Responsabilidad: Definir excepciones personalizadas (ej. CreacionEnvioException, ConsultaPedidoException) que la capa de

servicios lanza cuando una regla de negocio falla. Esto permite a la UI capturarlas y mostrar mensajes amigables al usuario.




```
models/exceptions/
├── CreacionEntityException.java
├── ConsultaEntityException.java
├── ActualizacionEntityException.java
└── EliminacionEntityException.java
```

- **Main (Punto de Entrada):**
  - Responsabilidad: Contener la clase GestorEnvios.java con el método public static void main(String[] args), que inicializa y ejecuta la aplicación (AppMenu).

## Persistencia

La capa de persistencia constituye el núcleo de la integridad de datos del sistema. Se ha implementado utilizando **JDBC (Java Database Connectivity)** puro bajo el patrón de diseño **DAO (Data Access Object)**, lo que permite un control granular sobre las sentencias SQL y el manejo de conexiones.

### Estructura y Orden de Operaciones

El modelo relacional en MySQL refleja fielmente el diseño de clases del dominio, destacando la relación 1 a 0..1 entre Pedidos y Envíos:

- Tabla pedidos: Entidad fuerte. Clave primaria id (AUTO_INCREMENT).
- Tabla envios: Entidad débil/dependiente. Posee una columna id_pedido con restricción de Clave Foránea (Foreign Key) y restricción de Unicidad (UNIQUE). Esto garantiza a nivel de motor de base de datos que un pedido no pueda tener múltiples envíos activos simultáneamente.

Para crear un Envio, es requisito que exista previamente un Pedido. La aplicación primero persiste el pedido y, en una operación separada, crea el envío asociándolo. Si intentamos crear un envío sin asignarle un pedido, la operación falla.

```
Seleccione una opción:
10
CREAR NUEVO ENVÍO POR NUMERO PEDIDO ===============
Ingrese el Numero del Pedido (PED-XXXXXXXX) al que asignar el envío o 'q' para salir:
PED-88888890
❌ El pedido con número PED-88888890 no existe. Intente nuevamente.
Ingrese el Numero del Pedido (PED-XXXXXXXX) al que asignar el envío o 'q' para salir:
PED-88888889

... Configurando datos del Envío ...
Costo del envío:
2500
Seleccione Empresa de Envío:
1. CORREO_ARGENTINO
2. ANDREANI
3. OCA
Opción:
1
Seleccione Tipo de Envío:
1. ESTANDAR
2. EXPRES
Opción:
2
✅ Envío Tracking Nro: TRK-00000001 creado exitosamente.

Presione Enter para continuar...
```

Gracias a las restricciones de Clave Foránea (Foreign Key) en la base de datos, el sistema impide el borrado (lógico) de un Pedido si este tiene un Envio asociado. El usuario primero debe eliminar el Envio antes de poder eliminar el Pedido.

```
  8  - Eliminar pedido por número
  9  - Eliminar pedido por ID

ENVIOS ========================================
 10  - Crear envío
 11  - Listar envíos
 12  - Buscar envío por tracking
 13  - Buscar envío por número de pedido
 14  - Buscar envío por ID
 15  - Actualizar estado envío por tracking
 16  - Actualizar estado envío por número de pedido
 17  - Actualizar estado envío por ID
 18  - Eliminar envío por tracking
 19  - Eliminar envío por número de pedido
 20  - Eliminar envío por ID

  0  - Salir

Seleccione una opción:
8
ELIMINAR PEDIDO POR NÚMERO ========================
Ingrese Numero del pedido a eliminar (PED-XXXXXXXX) o 'q' para salir:
PED-88888889
⚠ El pedido tiene un envío asociado. Elimine primero el envío: TRK-00000001

Presione Enter para continuar...
```

Elimino entonces el envío:

```
ELIMINAR ENVÍO POR NÚMERO DE PEDIDO ===============
Ingrese el NÚMERO de PEDIDO:
PED-88888889
¿Está seguro que desea eliminar el envío TRK-00000001? (s/n):
s
✓ Envío eliminado.
```

Y ahora puedo eliminar el pedido asociado:

```
  8  - Eliminar pedido por número
  9  - Eliminar pedido por ID

ENVIOS ==========================================
 10  - Crear envío
 11  - Listar envíos
 12  - Buscar envío por tracking
 13  - Buscar envío por número de pedido
 14  - Buscar envío por ID
 15  - Actualizar estado envío por tracking
 16  - Actualizar estado envío por número de pedido
 17  - Actualizar estado envío por ID
 18  - Eliminar envío por tracking
 19  - Eliminar envío por número de pedido
 20  - Eliminar envío por ID

  0  - Salir

Seleccione una opción:
8
ELIMINAR PEDIDO POR NÚMERO ========================
Ingrese Numero del pedido a eliminar (PED-XXXXXXXX) o 'q' para salir:
PED-88888889
¿Está seguro que desea eliminar el pedido PED-88888889? (s/n):
s
 Pedido eliminado correctamente.

Presione Enter para continuar...
```

Gestión de Transacciones (Commit y Rollback)

Para mantener la integridad de los datos, las operaciones de base de datos deben ser transaccionales. En nuestra arquitectura, la responsabilidad de manejar la transacción (iniciar, hacer commit o rollback) recae en la **Capa de Servicios** (ej. PedidoServiceImpl), la cual utiliza el TransactionManager de la capa de configuración.

El flujo es el siguiente:

1. La UI (Consola) solicita una operación (ej. "crear envío").
2. El Controlador (UI) llama a la Capa de Servicio (ej. envioService.crearEnvio...).
3. El Servicio solicita al TransactionManager que inicie una transacción. Esto obtiene una conexión (DatabaseConnection.getConnection()) y deshabilita el auto-commit (conn.setAutoCommit(false)).
4. El Servicio llama a los métodos del DAO (ej. envioDAO.save(pedido, conn)), pasándoles la conexión activa.
5. Si todas las operaciones del DAO finalizan sin errores, el Servicio le pide al TransactionManager que ejecute commit() para confirmar los cambios.
6. Si se produce una SQLException o una ValidationException durante el proceso, el Servicio captura la excepción y le pide al TransactionManager que ejecute rollback() para revertir todos los cambios.
7. Finalmente, la conexión se cierra en un bloque finally dentro del TransactionManager o del Servicio.

## Validaciones y Reglas de Negocio

Las reglas de negocio definen la lógica y las restricciones del dominio. Estas validaciones se implementan principalmente en la Capa de Servicios, antes de enviar los datos a la capa DAO.

- Pedido-Envío: Un Pedido puede existir sin un Envio asociado (ej. en estado "Pendiente" o "En Preparación").
- Envío-Pedido: Un Envio no puede existir si no está asociado a un Pedido válido.
- Tracking Obligatorio: Al momento de crear o actualizar un Envio, este debe poseer un número de seguimiento (trackingId). El servicio valida que este campo no sea nulo ni esté vacío.
- Datos de Pedido: Al crear un Pedido, se valida que los datos esenciales (como el nombre del cliente o la descripción) no estén vacíos.
- Integridad Referencial) El sistema (y la BD) impiden la eliminación de entidades maestras (como EmpresaEnvio o TipoEnvio) si están siendo referenciadas por algún Envio existente.
- Excepciones Personalizadas: Para manejar errores de negocio de forma limpia, se utiliza un paquete de excepciones personalizadas (ej. CreacionEnvioException, ConsultaPedidoException). Si una regla de negocio falla, el Servicio lanza una de estas excepciones, que es capturada por la UI para mostrar un mensaje amigable al usuario en lugar de un error de SQL.

## Pruebas Realizadas

Para verificar el correcto funcionamiento del sistema, se realizaron pruebas funcionales manuales cubriendo los principales casos de uso (CRUD) tanto desde la aplicación de consola como directamente en la base de datos.

### Pruebas de Consola

Se muestra la ejecución de algunas de las opciones CRUD del menú principal para validar la interacción entre la UI, los Servicios y el DAO.

**Menú principal:**

```
GESTOR DE ENVIOS ====================================
MENU ================================================

PEDIDOS =============================================
  1  - Crear pedido
  2  - Listar pedidos
  3  - Buscar pedido por número
  4  - Buscar pedido por tracking
  5  - Buscar pedido por ID
  6  - Buscar pedido por Cliente
  7  - Actualizar pedido por número
  8  - Eliminar pedido por número
  9  - Eliminar pedido por ID

ENVIOS ==============================================
 10  - Crear envío
 11  - Listar envíos
 12  - Buscar envío por tracking
 13  - Buscar envío por número de pedido
 14  - Buscar envío por ID
 15  - Actualizar estado envío por tracking
 16  - Actualizar estado envío por número de pedido
 17  - Actualizar estado envío por ID
 18  - Eliminar envío por tracking
 19  - Eliminar envío por número de pedido
 20  - Eliminar envío por ID

  0  - Salir
```

**Prueba 1: Creación de Pedido y Envío** Se registra un nuevo pedido, se verifica su creación, y posteriormente se le asigna un envío.

Creación de Pedido:

```
Seleccione una opción:
1
CREAR NUEVO PEDIDO ================================
Nombre del Cliente:
Mick Jagger
Total del pedido:
60000
✅ Pedido Tracking Nro: PED-88888890 creado exitosamente.

Presione Enter para continuar...
```

Creación de Envío asignando el pedido nuevo:

```
Seleccione una opción:
10
CREAR NUEVO ENVÍO POR NUMERO PEDIDO ===============
Ingrese el Numero del Pedido (PED-XXXXXXXX) al que asignar el envío o 'q' para salir:
PED-88888890

... Configurando datos del Envío ...
Costo del envío:
2500
Seleccione Empresa de Envío:
1. CORREO_ARGENTINO
2. ANDREANI
3. OCA
Opción:
3
Seleccione Tipo de Envío:
1. ESTANDAR
2. EXPRES
Opción:
2
✅ Envío Tracking Nro: TRK-00000002 creado exitosamente.
```

**Prueba 1.b: Prueba de Restricción (Envío existente – Relación 1:1)**

```
Seleccione una opción:
10
CREAR NUEVO ENVÍO POR NUMERO PEDIDO ===============
Ingrese el Numero del Pedido (PED-XXXXXXXX) al que asignar el envío o 'q' para salir:
PED-88888890
 El pedido con número PED-88888890 ya tiene un envío asignado. Intente nuevamente.
Ingrese el Numero del Pedido (PED-XXXXXXXX) al que asignar el envío o 'q' para salir:
q
 Operación cancelada por el usuario.

Presione Enter para continuar...
```

**Prueba 2:** Se prueba la funcionalidad de findAll() para listar todos los pedidos y envíos existentes.

Lista de Pedidos:

```
Seleccione una opción:
2
LISTA DE PEDIDOS ==================================
Total de pedidos registrados: 10002

ID     | Cliente             | Nº           | Estado       | Envío                                | Estado envío
-------------------------------------------------------------------------------------------------------------------
1      | Cliente 05729       | PED-00000001 | FACTURADO    | 9a581181-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
2      | Cliente 94346       | PED-00000002 | ENVIADO      | 9a58bcda-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
3      | Cliente 99376       | PED-00000003 | FACTURADO    | 9a598358-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
4      | Cliente 75214       | PED-00000004 | FACTURADO    | 9a59fe33-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
5      | Cliente 16934       | PED-00000005 | FACTURADO    | 9a5a677d-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
6      | Cliente 63391       | PED-00000006 | ENVIADO      | 9a5ae669-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
7      | Cliente 55045       | PED-00000007 | ENVIADO      | 9a5b44ae-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
8      | Cliente 55925       | PED-00000008 | NUEVO        | 9a5ba541-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
9      | Cliente 70722       | PED-00000009 | ENVIADO      | 9a5c041e-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
10     | Cliente 25915       | PED-00000010 | FACTURADO    | 9a5c6256-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
11     | Cliente 03886       | PED-00000011 | FACTURADO    | 9a5cb6f7-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
12     | Cliente 40397       | PED-00000012 | NUEVO        | 9a5d12fc-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
13     | Cliente 41078       | PED-00000013 | FACTURADO    | 9a5d76b0-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
14     | Cliente 77139       | PED-00000014 | ENVIADO      | 9a5dfe70-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
15     | Cliente 16509       | PED-00000015 | ENVIADO      | 9a5eb10a-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
16     | Cliente 61426       | PED-00000016 | FACTURADO    | 9a5f1f63-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
17     | Cliente 48072       | PED-00000017 | FACTURADO    | 9a5f85d2-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
18     | Cliente 85533       | PED-00000018 | ENVIADO      | 9a5fdc91-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
19     | Cliente 86207       | PED-00000019 | ENVIADO      | 9a604558-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
20     | Cliente 91513       | PED-00000020 | ENVIADO      | 9a609931-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
21     | Cliente 52747       | PED-00000021 | FACTURADO    | 9a60f65c-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
22     | Cliente 37428       | PED-00000022 | FACTURADO    | 9a615a53-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
23     | Cliente 71490       | PED-00000023 | ENVIADO      | 9a61ba63-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
24     | Cliente 75984       | PED-00000024 | ENVIADO      | 9a6236cb-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
25     | Cliente 28275       | PED-00000025 | ENVIADO      | 9a62c2fb-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
26     | Cliente 97422       | PED-00000026 | ENVIADO      | 9a63684b-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
27     | Cliente 52856       | PED-00000027 | ENVIADO      | 9a63de52-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
28     | Cliente 61548       | PED-00000028 | ENVIADO      | 9a643d92-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
```

```
41     | Cliente 03759        | PED-00000041 | ENVIADO      | 9a69c5f5-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
42     | Cliente 73927        | PED-00000042 | ENVIADO      | 9a6a1ae8-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
43     | Cliente 77572        | PED-00000043 | ENVIADO      | 9a6a7742-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
44     | Cliente 95517        | PED-00000044 | ENVIADO      | 9a6ace0e-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
45     | Cliente 28409        | PED-00000045 | FACTURADO    | 9a6b299b-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
46     | Cliente 61685        | PED-00000046 | NUEVO        | 9a6b88e5-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
47     | Cliente 10253        | PED-00000047 | ENVIADO      | 9a6beb11-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
48     | Cliente 98801        | PED-00000048 | ENVIADO      | 9a6c6f1d-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
49     | Cliente 95446        | PED-00000049 | NUEVO        | 9a6d4917-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
50     | Cliente 80732        | PED-00000050 | FACTURADO    | 9a6dd84d-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
Presione Enter para ver más o 'q' para salir:
```

Lista de envíos:

```
Seleccione una opción:
11
LISTA DE ENVÍOS ==================================
Total de envíos registrados: 10001

ID     | Empresa              | Tracking Nº                          | Estado
-------------------------------------------------------------------------------
1      | CORREO_ARGENTINO     | 9a581181-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
2      | ANDREANI             | 9a58bcda-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
3      | ANDREANI             | 9a598358-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
4      | ANDREANI             | 9a59fe33-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
5      | CORREO_ARGENTINO     | 9a5a677d-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
6      | OCA                  | 9a5ae669-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
7      | CORREO_ARGENTINO     | 9a5b44ae-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
8      | ANDREANI             | 9a5ba541-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
9      | ANDREANI             | 9a5c041e-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
10     | ANDREANI             | 9a5c6256-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
11     | ANDREANI             | 9a5cb6f7-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
12     | OCA                  | 9a5d12fc-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
13     | OCA                  | 9a5d76b0-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
14     | CORREO_ARGENTINO     | 9a5dfe70-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
15     | ANDREANI             | 9a5eb10a-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
16     | OCA                  | 9a5f1f63-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
17     | CORREO_ARGENTINO     | 9a5f85d2-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
18     | OCA                  | 9a5fdc91-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
19     | OCA                  | 9a604558-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
20     | ANDREANI             | 9a609931-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
21     | OCA                  | 9a60f65c-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
22     | ANDREANI             | 9a615a53-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
23     | CORREO_ARGENTINO     | 9a61ba63-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
24     | OCA                  | 9a6236cb-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
25     | OCA                  | 9a62c2fb-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
```

```
43     | CORREO_ARGENTINO     | 9a6a7742-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
44     | ANDREANI             | 9a6ace0e-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
45     | CORREO_ARGENTINO     | 9a6b299b-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION
46     | CORREO_ARGENTINO     | 9a6b88e5-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
47     | OCA                  | 9a6beb11-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
48     | ANDREANI             | 9a6c6f1d-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
49     | ANDREANI             | 9a6d4917-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO
50     | CORREO_ARGENTINO     | 9a6dd84d-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO
Presione Enter para ver más o 'q' para salir:
```

**Prueba 3:** Búsqueda de pedido y envío por número de pedido

```
Seleccione una opción:
3
BUSCAR PEDIDO POR NÚMERO ========================
Ingrese Numero de pedido (PED-XXXXXXXX) o 'q' para salir:
PED-00000031
ID     | Cliente             | Nº           | Estado       | Envío                                | Estado envío
-------------------------------------------------------------------------------------------------------------
31     | Cliente 26705       | PED-00000031 | NUEVO        | 9a655264-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO

Presione Enter para continuar...
```

```
Seleccione una opción:
13
BUSCAR ENVIO POR NUMERO PEDIDO ====================
Ingrese Numero de pedido (PED-XXXXXXXX) o 'q' para salir:
PED-00000034
ID     | Empresa              | Tracking Nº                          | Estado
-----------------------------------------------------------------------------------
34     | CORREO_ARGENTINO     | 9a665c67-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION

Presione Enter para continuar...
```

**Prueba 3.a:** búsqueda de pedidos y envíos por ID

```
Seleccione una opción:
5
BUSCAR PEDIDO POR ID ============================
Ingrese ID de pedido:
151
ID     | Cliente             | Nº           | Estado       | Envío                                | Estado envío
-------------------------------------------------------------------------------------------------------------
151    | Cliente 56482       | PED-00000151 | NUEVO        | 9a99789f-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION

Presione Enter para continuar...
```

```
Seleccione una opción:
14
BUSCAR ENVIO POR ID ================================
Ingrese ID de pedido:
500
ID     | Empresa              | Tracking Nº                          | Estado
-----------------------------------------------------------------------------------
500    | ANDREANI             | 9b2d3758-b01e-11f0-a2d7-36fb9e5c8605 | ENTREGADO

Presione Enter para continuar...
```

**Prueba 4:** Eliminado lógico

```
Seleccione una opción:
8
ELIMINAR PEDIDO POR NÚMERO ========================
Ingrese Numero del pedido a eliminar (PED-XXXXXXXX) o 'q' para salir:
PED-00000032
⚠ El pedido tiene un envío asociado. Elimine primero el envío: 9a65a453-b01e-11f0-a2d7-36fb9e5c8605

Presione Enter para continuar...
```

Eliminamos primero el envío:

```
Seleccione una opción:
18
ELIMINAR ENVÍO POR TRACKING ========================
Ingrese Tracking del envío a modificar:
9a65a453-b01e-11f0-a2d7-36fb9e5c8605
¿Está seguro que desea eliminar el envío 9a65a453-b01e-11f0-a2d7-36fb9e5c8605? (s/n):
s
✔ Envío eliminado.

Presione Enter para continuar...
```

Verificamos estado, sin envío

```
Seleccione una opción:
3
BUSCAR PEDIDO POR NÚMERO ==========================
Ingrese Numero de pedido (PED-XXXXXXXX) o 'q' para salir:
PED-00000032
ID     | Cliente             | Nº           | Estado       | Envío                                | Estado envío
-------------------------------------------------------------------------------------------------------------
32     | Cliente 86932       | PED-00000032 | NUEVO        | Sin envío                            | -

Presione Enter para continuar...
```

Ahora eliminamos

```
Seleccione una opción:
8
ELIMINAR PEDIDO POR NÚMERO ========================
Ingrese Numero del pedido a eliminar (PED-XXXXXXXX) o 'q' para salir:
PED-00000032
¿Está seguro que desea eliminar el pedido PED-00000032? (s/n):
s
✔ Pedido eliminado correctamente.

Presione Enter para continuar...
```

**Prueba 5:** Actualizar el estado de un envío por número de pedido

```
Seleccione una opción:
13
BUSCAR ENVIO POR NUMERO PEDIDO ====================
Ingrese Numero de pedido (PED-XXXXXXXX) o 'q' para salir:
PED-00000030
ID     | Empresa              | Tracking Nº                          | Estado
--------------------------------------------------------------------------------
30     | ANDREANI             | 9a64e558-b01e-11f0-a2d7-36fb9e5c8605 | EN_PREPARACION

Presione Enter para continuar...
```

```
Seleccione una opción:
16
ACTUALIZAR ESTADO DE ENVÍO POR NÚMERO DE PEDIDO ===
Ingrese el NÚMERO de PEDIDO:
PED-00000030
Seleccione Estado del Envío:
1. EN_PREPARACION
2. EN_TRANSITO
3. ENTREGADO
Opción:
2
 Envío actualizado de Tracking 9a64e558-b01e-11f0-a2d7-36fb9e5c8605 actualizado correctamente.

Presione Enter para continuar...
```

```
Seleccione una opción:
13
BUSCAR ENVIO POR NUMERO PEDIDO ====================
Ingrese Numero de pedido (PED-XXXXXXXX) o 'q' para salir:
PED-00000030
ID     | Empresa              | Tracking Nº                          | Estado
--------------------------------------------------------------------------------
30     | ANDREANI             | 9a64e558-b01e-11f0-a2d7-36fb9e5c8605 | EN_TRANSITO

Presione Enter para continuar...
```

## Consultas SQL de Verificación

Se utilizó un cliente de base de datos (MySQL Workbench) para verificar que los datos persistidos por la aplicación Java fueran correctos.

Primero generamos pedidos sin envío:

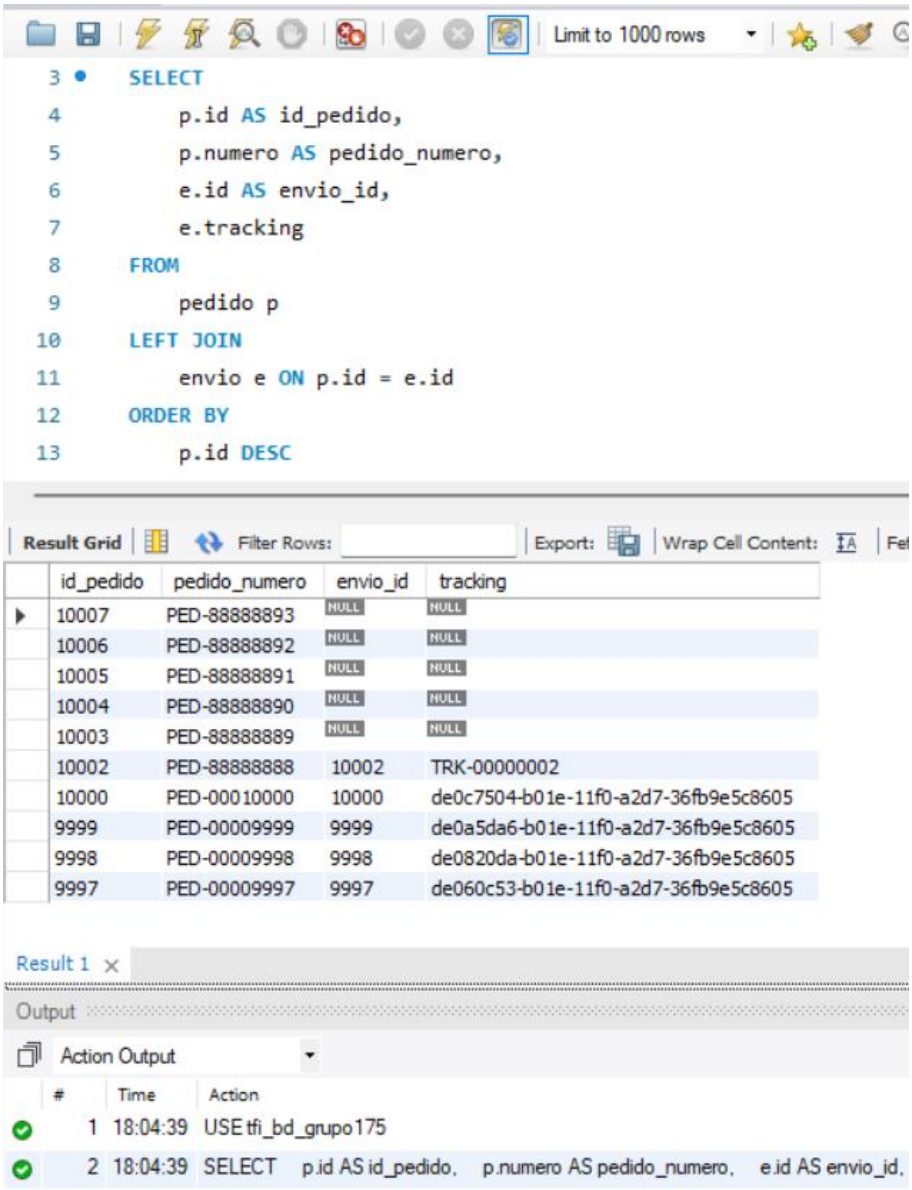
```
Seleccione una opción:
1
CREAR NUEVO PEDIDO ==================================
Nombre del Cliente:
Charlie Watts
Total del pedido:
5300
 Pedido Tracking Nro: PED-88888891 creado exitosamente.

Presione Enter para continuar...
```

```
Seleccione una opción:
1
CREAR NUEVO PEDIDO ==================================
Nombre del Cliente:
Keith Richards
Total del pedido:
8300
 Pedido Tracking Nro: PED-88888892 creado exitosamente.

Presione Enter para continuar...
```

```
Seleccione una opción:
1
CREAR NUEVO PEDIDO ==================================
Nombre del Cliente:
Ronnie Wood
Total del pedido:
7200
 Pedido Tracking Nro: PED-88888893 creado exitosamente.

Presione Enter para continuar...
```

Consulta: Verificación de últimos Pedido-Envío (JOIN)

Pedidos con y sin envío (orden descendente ya que fueron creados los últimos pedidos sin envío)



## Conclusiones y Mejoras

Aunque el sistema es funcional, existen varias áreas de mejora para futuras iteraciones:

- Reemplazar JDBC por JPA: Migrar la capa de persistencia de JDBC puro a una solución de ORM (Object-Relational Mapping) como **JPA/Hibernate**. Esto reduciría drásticamente el código repetitivo (boilerplate) en la capa DAO y simplificaría las consultas.
- API REST: Exponer la capa de servicios como una **API REST (Spring Boot)**, permitiendo que el sistema sea consumido por clientes web (React, Angular) o móviles.
- Testing Unitario: Implementar pruebas unitarias formales con **JUnit** y **Mockito** para automatizar la validación de la capa de servicios y DAO, garantizando la fiabilidad del código ante futuros cambios.

El desarrollo de este proyecto permitió aplicar de manera práctica un sistema de consola funcional, capaz de gestionar el ciclo de vida de pedidos y envíos de acuerdo con los requisitos del dominio, la arquitectura en capas y la persistencia de datos con JDBC.

El logro principal del proyecto es la implementación exitosa de una arquitectura de capas desacoplada. La estricta separación de responsabilidades entre la capa de presentación (ui.console), la capa de lógica de negocio (services) y la capa de acceso a datos (dao), ha demostrado ser un pilar fundamental para la mantenibilidad y escalabilidad del código.

Destaca la implementación de un manejo de persistencia robusto. Mediante el uso del patrón DAO, JDBC y un TransactionManager centralizado, el sistema garantiza la integridad de los datos (ACID) a través de operaciones de commit y rollback. El uso de excepciones personalizadas (models.exceptions) complementa esta robustez, permitiendo un manejo de errores elegante entre la capa de servicio y la interfaz de usuario.

El proyecto cumple exitosamente con los objetivos planteados:

1. Arquitectura sólida: Separación clara de responsabilidades en capas bien definidas
2. Patrones de diseño: Aplicación correcta de múltiples patrones reconocidos
3. Persistencia robusta: Manejo profesional de base de datos con transacciones
4. Código mantenible: Buenas prácticas y documentación exhaustiva
5. Funcionalidad completa: Sistema operativo con todas las operaciones CRUD

El sistema demuestra un dominio sólido de los conceptos de programación orientada a objetos, arquitectura de software y persistencia de datos, cumpliendo con los estándares de calidad esperados, y sentando una base sólida para las futuras mejoras propuestas.

## Fuentes y Herramientas Utilizadas

- **Lenguaje:** Java (JDK 21).
- **IDE:** Apache NetBeans.
- **Base de Datos:** MySQL 8.
- **Driver:** MySQL Connector/J (Driver JDBC).
- **Cliente DB:** MySQL Workbench.
- **Control de Versiones:** Git y GitHub.
- **Asistencia de IA:** Se utilizaron dos asistentes de IA, Gemini y Microsoft Copilot para la depuración de código, resolución de problemas de configuración de JDBC.