UNIVERSIDAD SIGLO 21

Cozy Books:

Gestor de ventas para librerías

Estudiante: Daiana Micaela Arena

Carrera: Licenciatura en informática

Cursada: EDH

Materia: Seminario de práctica de informática

Prof.: ANA CAROLINA FERREYRA

DNI: 38629115

Índice

Índice 1
Objetivo 4
Consigna 4
Resolución 6
Sistema de Gestión de Ventas para Librerías
Caso aplicado: Cozy Books 6
Introducción 6
Justificación 6
Definiciones del Proyecto y del Sistema 7
Elicitación 7
Competencia 8
Relevamiento 8
Propuesta de Solución 10
Propuesta Funcional 10
Propuesta Técnica 10
Casos de Uso 11
Listado de Casos de Uso 11
Matriz de trazabilidad RF - CU 12
Diagrama UML de casos de uso 14
Análisis 15
Registrar libro (CU11) 15
Registrar venta (CU16) 16
Diseño 17
Diagrama de clases 17
Entidades del dominio 18
Lógica de negocio 18
Acceso a datos 18
Relaciones 19
Implementación 20
Diagrama de implementación 20
Nodos principales 20
Componentes de la aplicación 20
Estructura de la base de datos 21
Relaciones importantes 22
Pruebas 22
Introducción 22
Alcance 22
Estrategia 23
Casos de prueba 23
Definición de base de datos para el sistema 32

Sistema de gestión de Base de Datos....32
Tablas....32
Diagrama entidad-relación....36
Creación de tablas SQL....36
Inserción, consulta y borrado de registros....40
Presentación de consultas SQL....56
Definiciones de comunicación....56
Desarrollo en Java....57
Estructura del proyecto....57
Presentación del desarrollo....57
Compilación y ejecución....58
Características implementadas....58
1. Correcta utilización de sintaxis, tipos de datos y estructuras de control....58
2. Tratamiento y manejo de excepciones....59
3. Encapsulamiento,polimorfismo y abstracción....60
4.Menu de selección....62
5.Estructuras condicionales y repetitivas....62
6.Declaración y creación de objetos en Java....63
7.Utilización de constructores para inicializar objetos....64
8.Algoritmos de ordenación y búsqueda (opcional)....64
Integración con la Base de Datos (JDBC)....66

Objetivo

Esta actividad busca que seas capaz de plantear una solución problemática que pueda resolverse mediante la realización de un proyecto informático.
Para llevar adelante este desafío, podrás aplicar muchos de los conceptos más importantes abordados durante el desarrollo del módulo 1, que recupera tu trabajo en materias troncales de la carrera.
Durante su desarrollo, lograrás aplicar tus conocimientos para alcanzar los siguientes objetivos:

- Definir el alcance y justificación de un proyecto informático para dar solución a una situación problemática.
- Realizar la justificación de un proyecto y la definición de objetivos.
- Aplicar el proceso unificado de desarrollo (PUD).
- Realizar el análisis del modelo de negocio.
- Plantear requerimientos funcionales y no funcionales.

En esta actividad, deberás seleccionar el problema a resolver, realizar el análisis del área problemática, definir el título y objetivos del proyecto, realizar la elicitación, plantear la propuesta de solución y comenzar con la etapa de análisis.

Consigna

Considera que te encuentras trabajando en la organización cuyo problema definiste resolver con un proyecto informático, y te solicitan liderar el equipo de desarrollo que va a llevar adelante el mismo.

El proyecto informático que definas debe ser distinto al considerado en la lectura y el entregable final debe cumplir con los siguientes objetivos:

- Realizar el análisis del modelo de negocios para definir y justificar el proyecto.
- Aplicar el proceso unificado de desarrollo (PUD) para garantizar la calidad, escalabilidad y eficiencia en el ciclo de desarrollo.
- Utilizar una base de datos MySQL para la persistencia de los datos.
- Emplear Java como lenguaje de programación para el desarrollo del sistema.

A los fines del trabajo, para la implementación de la base de datos y el desarrollo con Java, puedes presentar un prototipo, que es “es la creación de un modelo operacional que incluya solo algunas características del sistema final” (Kendall & Kendall, 2011).

El concepto de operacional es clave, ya que no se trata de un simple modelo, sino que permite desarrollar módulos que se van integrando en la versión final del sistema.

Esta primera actividad te permitirá realizar el análisis del área problemática planteada y definir el proyecto informático que le dará solución, incluyendo su justificación y definición de objetivos. Adicionalmente, comenzarás con la aplicación del proceso unificado de desarrollo (PUD), que proporciona una estructura sistemática para el desarrollo del *software* y una división del mismo en etapas bien definidas.

Es fundamental que puedas presentar un análisis del modelo de negocio relacionado con el proyecto para comprender a fondo los procesos y las operaciones del negocio, identificar las necesidades y los problemas a resolver, definir los objetivos y los resultados esperados del sistema. Este análisis proporcionará una base sólida para el diseño y desarrollo del sistema.

Finalmente, deberás considerar la realización de un modelo de requerimientos para capturar de forma precisa y completa los requisitos del sistema, funcionales y no funcionales. Esto implica

identificar y documentar las funcionalidades, las características esperadas, así como las restricciones y los criterios de calidad.

Los entregables a presentar son los siguientes:

- Título del proyecto.
- Introducción.
- Justificación.
- Definiciones del proyecto y del sistema.
- Elicitación.
- Conocimiento del negocio.
- Propuesta de solución.

Inicio del análisis: casos de uso.

Resolución

Sistema de Gestión de Ventas para Librerías
Caso aplicado: Cozy Books

Introducción

El presente trabajo se enfoca en el análisis, diseño y propuesta de desarrollo de un sistema de gestión de ventas orientado a librerías, tomando como caso de estudio a Cozy Books, una tienda en línea de libros fundada en 2013 por una familia apasionada por la lectura. A través de este sistema se busca digitalizar los procesos de venta y gestión de inventario actualmente gestionados manualmente, optimizando la eficiencia operativa y reduciendo errores. Este documento detalla el abordaje del problema, la comprensión del contexto organizacional, la propuesta de solución y los elementos iniciales del análisis funcional del sistema.

Cozy Books es una librería en línea dedicada a la comercialización de libros en diferentes formatos (físicos, digitales y audiolibros). Actualmente, sus operaciones de venta, gestión de inventario y registro de transacciones se gestionan mediante planillas de cálculo, lo que ha comenzado a resultar ineficiente debido al aumento en el volumen de ventas y la diversificación de productos.

El proyecto propuesto consiste en analizar, diseñar y desarrollar un sistema de escritorio basado en **Java** y **MySQL** que permita registrar y gestionar autores, clientes, libros y ventas.

Justificación

Este proyecto aborda una problemática frecuente en negocios en crecimiento: la falta de sistemas que acompañen la evolución operativa y la diversificación de productos. En el caso de Cozy Books, la gestión manual a través de planillas de cálculo no solo implica demoras sino también riesgos operativos como errores de registro de ventas, pérdida de trazabilidad de inventario y dificultades en el seguimiento del historial de compras por cliente. La implementación de un sistema de gestión de ventas personalizado permitirá no solo optimizar procesos internos, sino también sentar las bases para futuras expansiones o integraciones con otras herramientas digitales del comercio electrónico (tengamos en cuenta que, en la actualidad, sus ventas se realizan a través de redes sociales).

Los principales beneficios del sistema serán:

- Optimizar los procesos de venta y gestión de inventario.
- Minimizar errores manuales en el registro de transacciones.
- Mejorar la disponibilidad de información para la toma de decisiones comerciales, por ejemplo, saber los gustos de un cliente para enviarle ofertas.
- Reducir tiempos de procesamiento de ventas y generación de tickets.
- Brindar una solución a medida, más económica y personalizada que adquirir un software comercial de gestión.
- Facilitar el seguimiento del historial de compras por cliente.

Este proyecto tiene un impacto tecnológico relevante, ya que permitirá a la librería comenzar su proceso de digitalización integral, mejorando su competitividad en el mercado del comercio electrónico de libros y proporcionando una base sólida para el crecimiento del negocio.

Definiciones del Proyecto y del Sistema

- **Objetivo del Proyecto:**
  Analizar, diseñar y desarrollar un sistema de **gestión de ventas de una librería** que administre autores, clientes, libros y transacciones comerciales para Cozy Books.
- **Objetivo General del Sistema:**
  El sistema a desarrollar será una aplicación de escritorio con interfaz de consola, diseñada para registrar y administrar entidades claves como autores, clientes, libros (en sus diferentes formatos: físico, digital y audiolibro) y ventas. Con una estructura modular orientada a objetos que facilite su mantenimiento y futura evolución. El sistema priorizará la simplicidad, eficiencia operativa y trazabilidad de transacciones, adaptándose a las necesidades particulares de Cozy Books en el manejo de su catálogo.

Elicitación

Para la comprensión profunda del dominio del problema y de las necesidades específicas de Cozy Books, se aplicaron las siguientes técnicas de elicitación:

● Entrevistas: Se realizaron entrevistas con las propietarias de Cozy Books para comprender los procesos actuales, identificar puntos críticos y conocer las expectativas respecto al sistema.
● Observación directa: Se observaron in situ los procesos de venta, registro de inventario y gestión de clientes para identificar las prácticas reales, flujos de trabajo y posibles ineficiencias no verbalizadas durante las entrevistas.
● Cuestionarios: Se aplicaron cuestionarios estructurados para obtener información cuantitativa sobre volúmenes de operación, frecuencia de ventas, cantidad de libros en catálogo y diversidad de autores representados.

Actividad del Cliente

El dominio de las librerías, tanto físicas como en línea, se caracteriza por la gestión de productos editoriales (libros en diversos formatos) que vinculan autores, editoriales y clientes finales. Las operaciones principales de una librería incluyen:

● Gestión de catálogo: Registro y clasificación de libros según categorías, géneros, autores y formatos (físico, digital, audiolibro). Cada libro posee atributos específicos como ISBN, título, autor(es), editorial, precio, y formato.

● Gestión de inventario: Control de stock disponible, especialmente crítico para libros físicos. Los formatos digitales y audiolibros no presentan restricciones de stock físico, pero sí requieren control de licencias o accesos.

● Gestión de autores: Registro de información sobre autores y la relación con sus obras, facilitando búsquedas y reportes.

● Gestión de clientes: Alta de clientes, seguimiento de historial de compras, preferencias literarias y datos de contacto para acciones de marketing personalizado.

● Procesamiento de ventas: Registro de transacciones comerciales incluyendo cliente, libro(s) adquirido(s), fecha, método de pago y monto total. Generación de comprobantes de venta (tickets o facturas).

● Reporting: Consultas sobre ventas por período, libros más vendidos, autores destacados, comportamiento de clientes, entre otros indicadores de gestión comercial.

En el contexto actual del comercio electrónico, las librerías en línea deben gestionar múltiples formatos de productos (físico con envío, digital con descarga inmediata, audiolibro con acceso streaming o descarga) y adaptar sus procesos logísticos y de entrega según el tipo de producto vendido.

Competencia

En el mercado existen diversas soluciones estándar de gestión para librerías como Bookmanager, BookTrakker o sistemas ERP genéricos como SAP o Oracle, que integran funcionalidades administrativas y de ventas. No obstante, estas plataformas presentan dos limitaciones clave en relación con Cozy Books:

- **Costo elevado:** representan una inversión considerable poco viable para una librería familiar en crecimiento.
- **Complejidad funcional**: incluyen múltiples módulos que exceden las necesidades reales de la empresa, dificultando su adopción y uso efectivo.

Dada esta situación, el desarrollo de un sistema de gestión de ventas a medida representa una alternativa más adecuada tanto en lo económico como en lo funcional.

Relevamiento

El relevamiento de la situación actual de Cozy Books permitió caracterizar tanto el contexto organizacional como las métricas operativas clave:

Contexto Organizacional:

● Cozy Books es una librería familiar en línea fundada en 2013, que comercializa libros en tres formatos: físico, digital y audiolibro.

● Las ventas se realizan principalmente a través de redes sociales (Instagram, Facebook) y contacto directo vía WhatsApp.

● El equipo operativo está conformado por los propietarios y dos colaboradores part-time.

● Los procesos actuales se gestionan manualmente mediante planillas de cálculo de Excel, sin sistema integrado de gestión.

Métricas Operativas:

● Catálogo actual: aproximadamente 350 títulos distribuidos en los tres formatos.

● Diversidad de autores: 180 autores representados en el catálogo.

● Base de clientes activos: alrededor de 250 clientes registrados informalmente.

● Volumen de ventas mensual: entre 40 y 60 transacciones mensuales, con picos en temporadas específicas (inicio de clases, fin de año).

● Crecimiento: incremento del 25% en ventas año a año en los últimos dos años.

Problemática Identificada:

A partir de la observación directa y el análisis de la documentación actual, se identificaron las siguientes deficiencias operativas:

● Los datos de autores, clientes y libros se almacenan de forma dispersa en múltiples hojas de cálculo, sin validaciones ni consistencia entre registros.

● Las ventas se procesan manualmente, muchas veces sin un registro sistemático de las transacciones. En ocasiones no se registran cambios en el stock, generando problemas de disponibilidad y sobreventa de libros físicos.

● No existe un registro estructurado de preferencias de compra o historial detallado por cliente, lo que dificulta acciones de marketing personalizado y envío de ofertas segmentadas.

● La facturación se realiza manualmente, generando tickets básicos sin un formato estandarizado, lo cual dificulta la trazabilidad y el control contable.

● La búsqueda de información (libros por autor, ventas por cliente, stock disponible) requiere navegación manual en múltiples archivos, consumiendo tiempo operativo valioso.

Propuesta de Solución

Propuesta Funcional

El sistema deberá permitir:

- RF01: Registrar nuevos autores.
- RF02: Actualizar información de autores.
- RF03: Eliminar autores.
- RF04: Listar todos los autores registrados.
- RF05: Buscar un autor específico.
- RF06: Registrar nuevos clientes.
- RF07: Actualizar información de clientes.
- RF08: Eliminar clientes.
- RF09: Listar todos los clientes registrados.
- RF10: Buscar un cliente específico.
- RF11: Registrar libros.
- RF12: Actualizar información de libros.
- RF13: Eliminar libros.
- RF14: Listar todos los libros.
- RF15: Buscar un libro específico.
- RF16: Registrar ventas.
- RF17: Actualizar ventas.
- RF18: Eliminar ventas.
- RF19: Buscar ventas.
- RF20: Listar ventas.
- RF21: Generar tickets en formato de archivo de texto.
- RF22: Reporte de Libros por Autor.

Propuesta Técnica

- **Plataforma:** Aplicación de escritorio basada en Java.
- **Base de Datos:** MySQL.
- **Interfaz de usuario:** Menú interactivo en consola.
- **Arquitectura:**
    - Separación clara entre lógica de negocios, acceso a datos y presentación.
    - Uso de POO (Programación Orientada a Objetos).
- **Requerimientos no funcionales:**
    - RNF01: El sistema debe ser seguro, solicitando confirmaciones antes de eliminar registros.
    - RNF02: El tiempo de respuesta para consultas y reportes debe ser inferior a 3 segundos.

Casos de Uso

Listado de Casos de Uso

Código	Nombre	Descripción
CU01	Registrar autor	Permite al usuario registrar un nuevo autor en el sistema.
CU02	Actualizar Autor	Permite modificar la información de un autor existente.
CU03	Eliminar Autor	Permite eliminar un autor registrado del sistema.
CU04	Listar Autores	Permite listar todos los autores registrados en el sistema.
CU05	Buscar Autor	Permite buscar un autor por algún criterio (nombre, id).
CU06	Registrar Cliente	Permite al usuario registrar un nuevo Cliente en el sistema.
CU07	Actualizar Cliente	Permite modificar la información de un Cliente existente.
CU08	Eliminar Cliente	Permite eliminar un Cliente registrado del sistema.
CU09	Listar Cliente	Permite listar todos los Clientes registrados en el sistema. (**incluye** CU20: Buscar ventas)
CU10	Buscar Clientes	Permite buscar un Cliente por algún criterio (nombre, DNI).
CU11	Registrar Libro	Permite al usuario registrar un nuevo Libro en el sistema.
CU12	Actualizar Libro	Permite modificar la información de un Libro existente.
CU13	Eliminar Libro	Permite eliminar un Libro registrado del sistema.

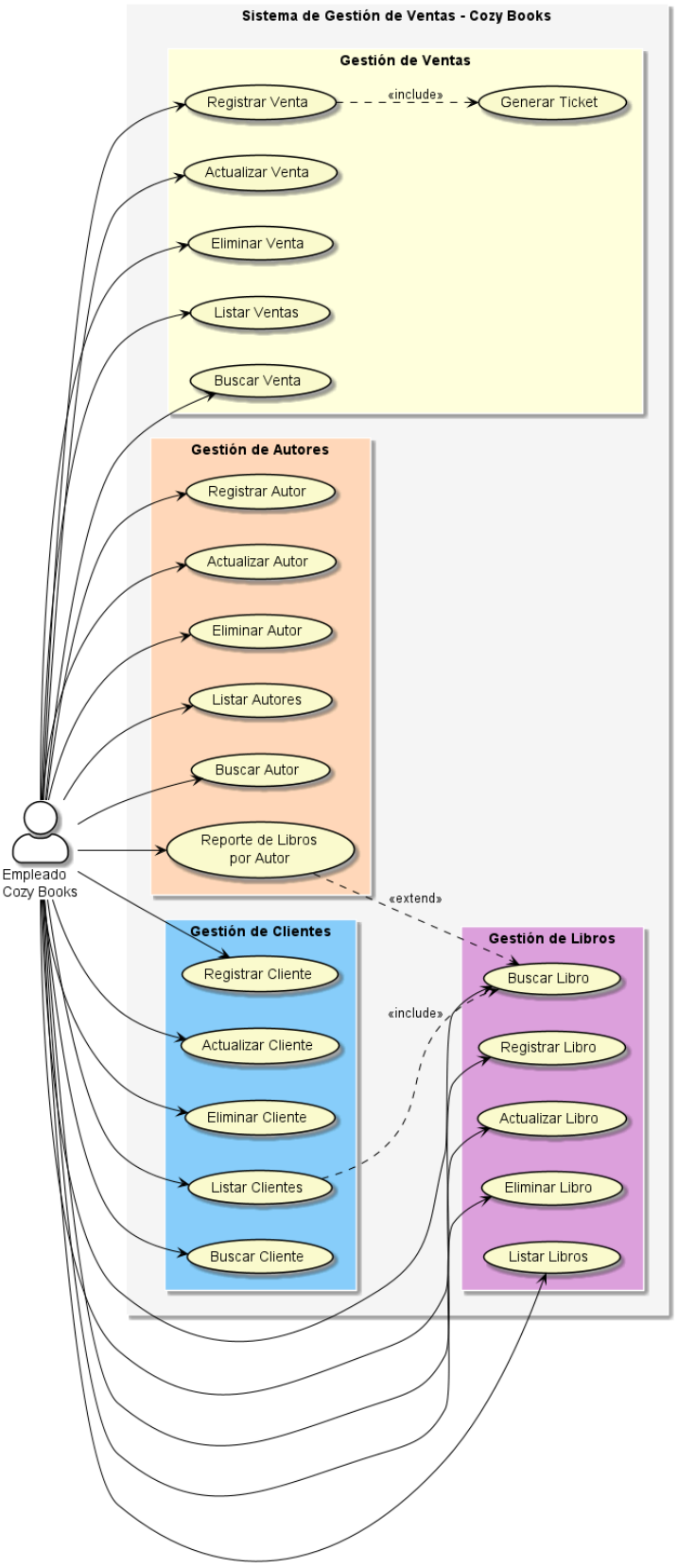
CU14	Listar Libros	Permite listar todos los Libros registrados en el sistema agrupándolos por tipo (audiolibro/físico/digital).
CU15	Buscar Libro	Permite buscar libros por título o autor.
CU16	Registrar Venta	Permite al usuario registrar una nueva Venta en el sistema. (**incluye** CU21: Generar Ticket)
CU17	Actualizar Venta	Permite modificar la información de una Venta existente.
CU18	Eliminar Venta	Permite eliminar una Venta registrada del sistema.
CU19	Listar Ventas	Permite listar todas las ventas registradas en el sistema.
CU20	Buscar Venta	Permite buscar ventas por algún criterio (cliente, fecha, monto)
CU21	Generar Ticket	Permite generar facturas en formato de archivo de texto.
CU22	Reporte de Libros por Autor	Permite consultar todos los libros escritos por un autor específico. (**extiende** CU15: Buscar Libro)

Matriz de trazabilidad RF - CU

Requerimiento Funcional	**Caso de Uso Asociado**
RF01	CU01
RF02	CU02
RF03	CU03
RF04	CU04
RF05	CU05
RF06	CU06
RF07	CU07

RF08	CU08
RF09	CU09
RF10	CU10
RF11	CU11
RF12	CU12
RF13	CU13
RF14	CU14
RF15	CU15
RF16	CU16
RF17	CU17
RF18	CU18
RF19	CU19
RF20	CU20
RF21	CU21
RF22	CU22

Diagrama UML de casos de uso



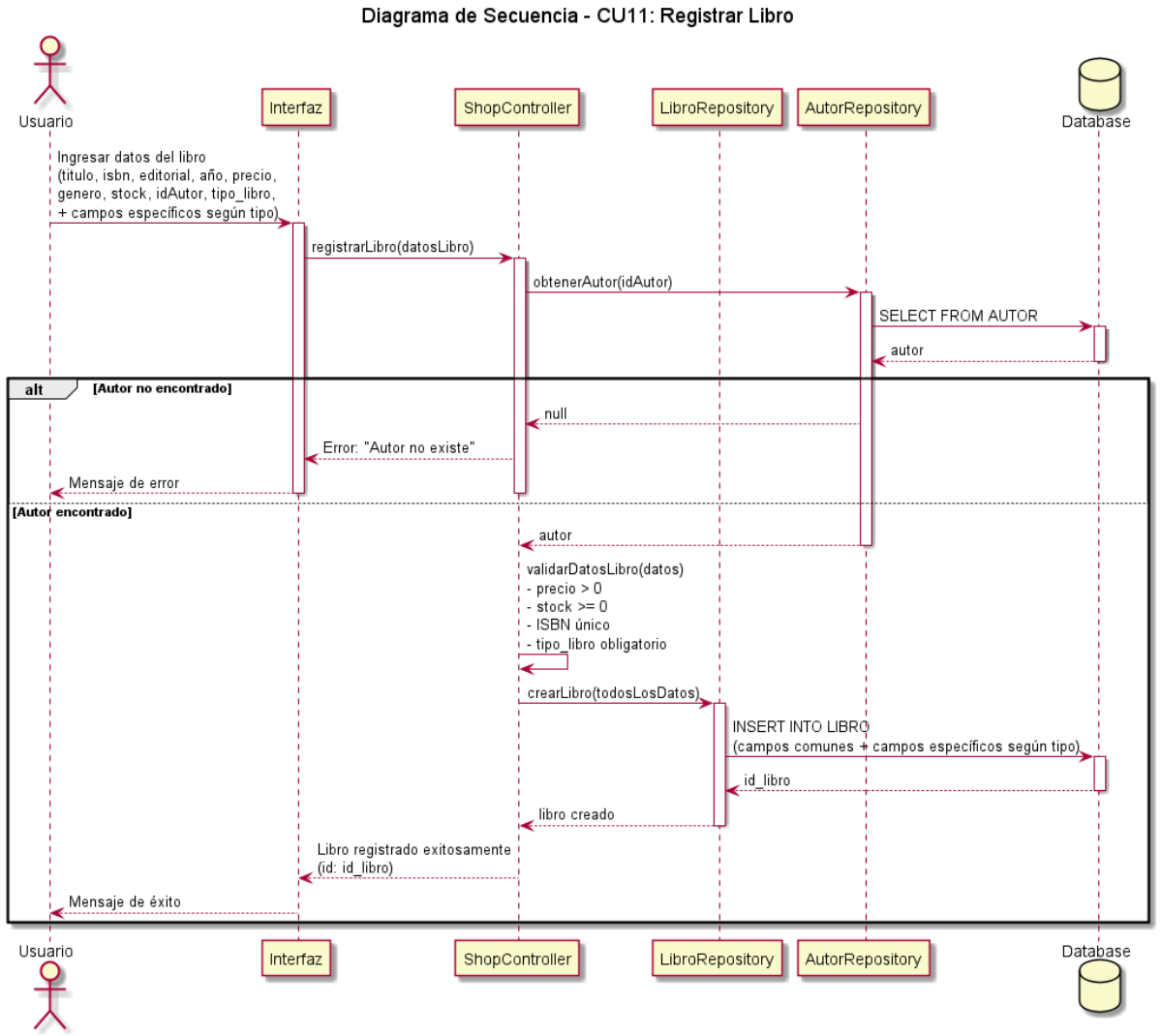
Análisis

En esta etapa detallaremos en profundidad el funcionamiento del sistema propuesto. A continuación se modelan dos de los casos de uso más representativos, detallando las interacciones entre los actores y los componentes del sistema. Esto permite describir la estructura lógica del software y sentar las bases para las etapas posteriores de diseño e implementación.

Para este proyecto se han seleccionado dos casos de uso clave: Registrar libro (CU11) y Registrar venta (CU16), por ser representativos de las funcionalidades de carga y consulta del sistema.

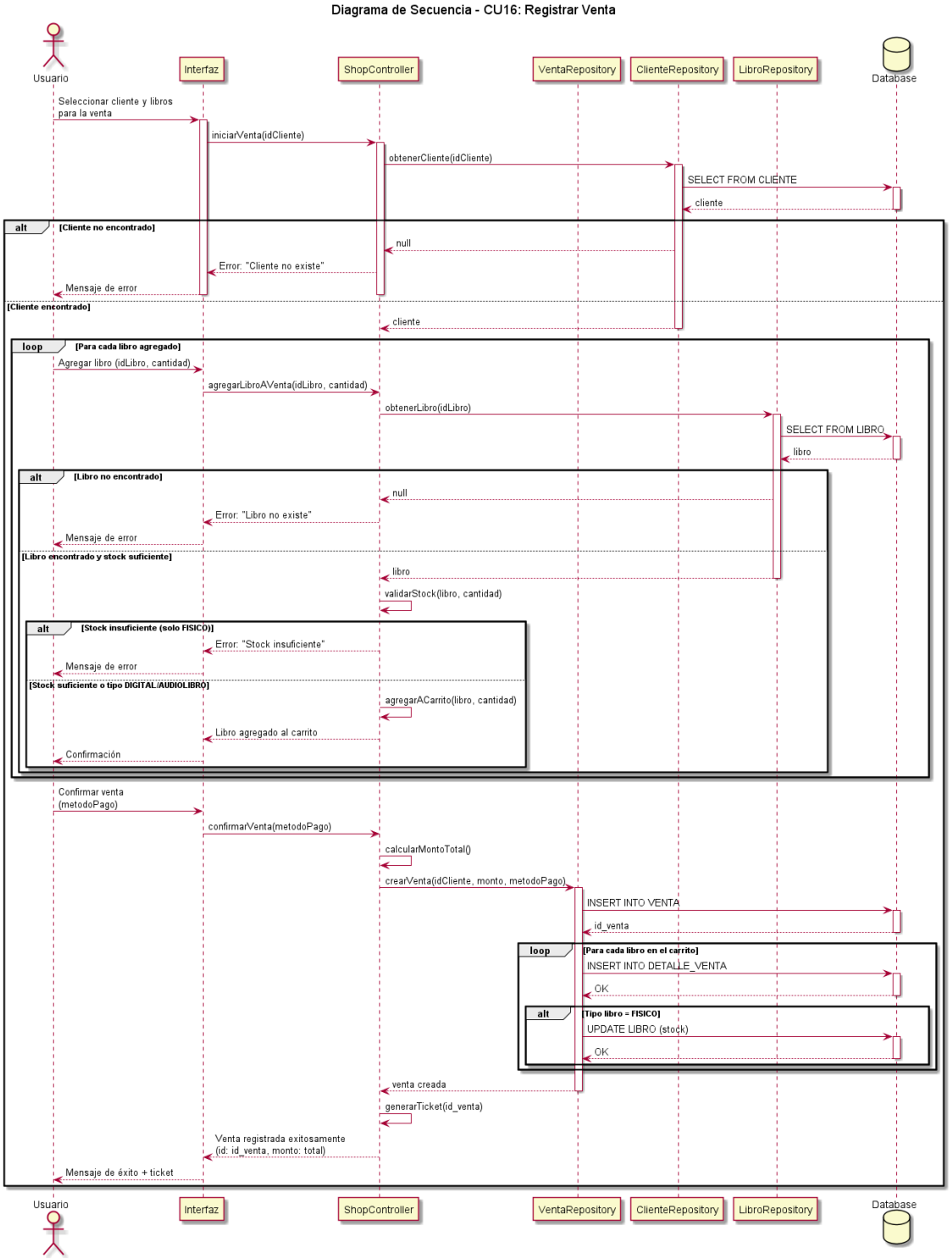
Registrar libro (CU11)

Este caso de uso permite al usuario registrar un nuevo libro en el sistema, ingresando los datos requeridos y almacenándolos de forma persistente.



Registrar venta (CU16)

Este caso de uso permite al usuario registrar una nueva venta en el sistema, seleccionando un cliente existente, agregando uno o más libros con sus cantidades correspondientes, validando la disponibilidad de stock (para libros físicos), calculando el monto total y almacenándolos de forma persistente en la base de datos.



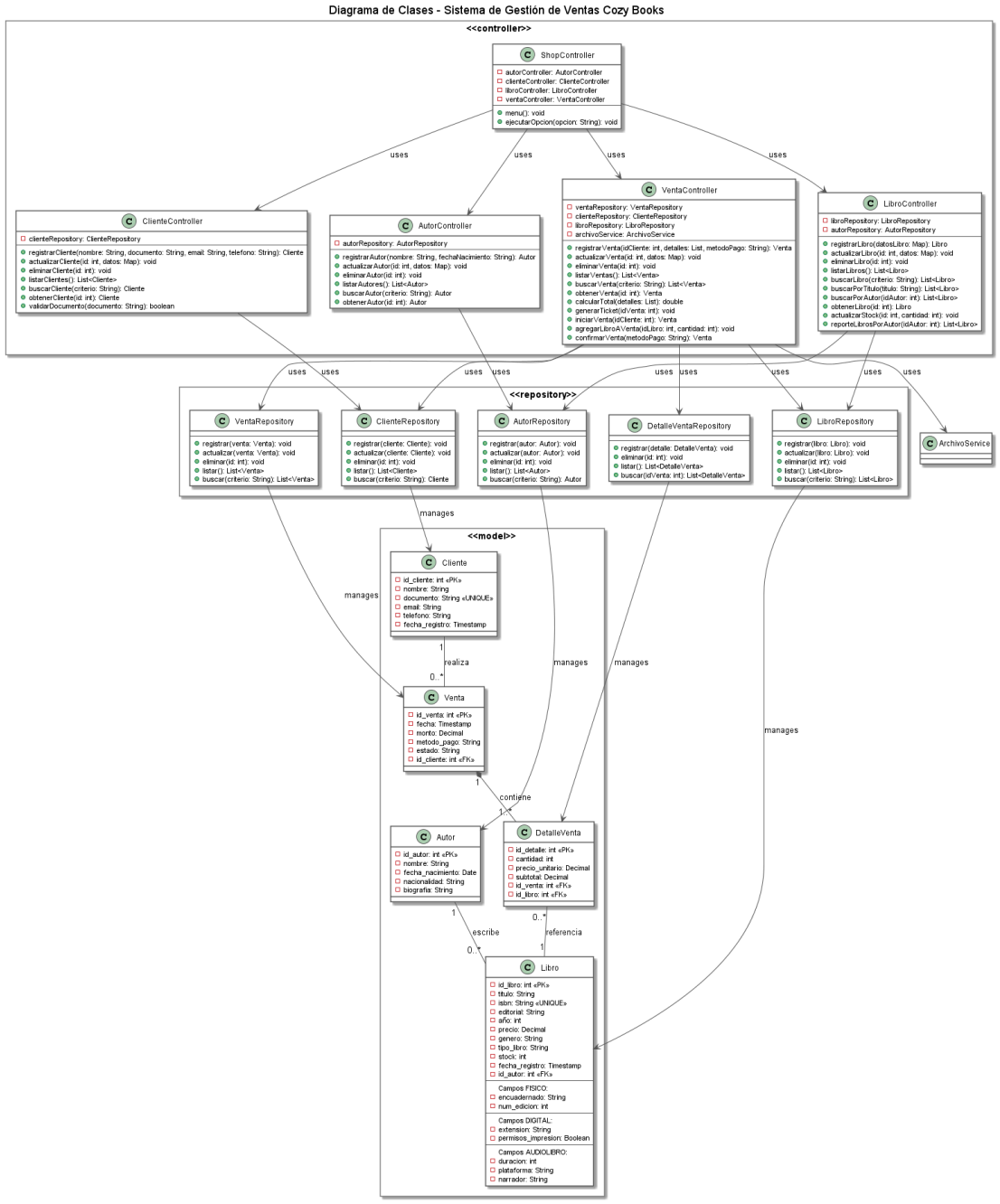
Al finalizar, el sistema genera automáticamente un ticket de compra en formato de archivo de texto (.txt) que incluye: fecha de la venta, datos del cliente, detalle de los libros vendidos (título, cantidad,

precio unitario), subtotales y el monto total de la transacción. Este archivo se guarda en el sistema de archivos local para su posterior impresión o envío al cliente.

Diseño

En esta etapa definimos la arquitectura interna del sistema desde el punto de vista de la **programación orientada a objetos**. Se identifican las clases principales que componen la solución, sus responsabilidades, atributos y métodos, así como las relaciones entre ellas. Esto proporciona una guía para la implementación y asegura coherencia entre el análisis y el desarrollo.

Diagrama de clases



Este diagrama de clases representa los componentes principales del sistema de ventas de Cozy Books: sigue una arquitectura en capas (MVC) para asegurar la separación de responsabilidades y hacer al sistema más mantenible y escalable.

Entidades del dominio

- **Autor** y **Cliente** son las clases que representan a los autores de libros y clientes de la librería, respectivamente. Autor contiene información básica como nombre, fecha de nacimiento, nacionalidad y biografía. Cliente incluye datos personales como nombre, documento único, email, teléfono y fecha de registro en el sistema.

- **Libro** es la clase que centraliza toda la información de los productos comercializados por la librería. Contiene atributos comunes a todos los tipos de libro como título, ISBN, editorial, año, precio, género, stock y fecha de registro. Además, incluye el campo tipo_libro que define si es físico, digital o audiolibro. Esta clase implementa campos condicionales que se utilizan según el tipo: para libros físicos (encuadernado, num_edicion), para libros digitales (extension, permisos_impresion) y para audiolibros (duracion, plataforma, narrador). Esta estructura permite gestionar los tres formatos de producto en una única tabla, optimizando el modelo de datos.

- **Venta** y **DetalleVenta** representan las transacciones comerciales del sistema. Venta es la clase principal que registra información de cada operación: fecha, monto total, método de pago, estado de la transacción y el cliente asociado. DetalleVenta es una clase intermedia que permite modelar la relación muchos-a-muchos entre ventas y libros, almacenando para cada línea de venta la cantidad vendida, el precio unitario del libro al momento de la transacción y el subtotal calculado. Esta separación permite mantener un historial preciso de precios y facilita la generación de reportes detallados por venta.

Lógica de negocio

- **AutorController**, **ClienteController**, **LibroController** y **VentaController** son responsables de la lógica de negocio relacionada con la gestión de autores, clientes, libros y ventas respectivamente. Cada controller implementa las operaciones CRUD completas (registrar, actualizar, eliminar, listar y buscar) correspondientes a los casos de uso CU01-CU05 (Autores), CU06-CU10 (Clientes), CU11-CU15 (Libros) y CU16-CU20 (Ventas). LibroController además gestiona la relación con autores mediante la validación de existencia antes del registro y provee funcionalidades especializadas como el reporte de libros por autor (CU22). VentaController coordina operaciones complejas que involucran múltiples entidades: valida la existencia de clientes y libros, gestiona el carrito de compras temporal, calcula montos totales, controla el stock de libros físicos y delega en ArchivoService la generación automática de tickets de venta en formato de archivo de texto (CU21).
- **ShopController** actúa como controlador principal del sistema, orquestando el menú interactivo de consola y delegando las operaciones específicas a los controllers correspondientes según la opción seleccionada por el usuario. Este patrón de diseño permite una clara separación de responsabilidades y facilita el mantenimiento del código.

Acceso a datos

AutorRepository, **ClienteRepository**, **LibroRepository**, **VentaRepository** y **DetalleVentaRepository** abstraen las operaciones de persistencia hacia la base de datos MySQL. Estas clases encapsulan completamente el acceso a datos y proporcionan una interfaz uniforme de operaciones CRUD

(Create, Read, Update, Delete) para cada entidad del modelo. Cada repository expone cinco métodos estándar: registrar() para insertar nuevos registros, actualizar() para modificar registros existentes, eliminar() para borrar registros por ID, listar() para obtener todos los registros de una entidad, y buscar() para consultas específicas por diferentes criterios. Esta capa de abstracción permite que los controllers trabajen con objetos del dominio sin preocuparse por los detalles de implementación de SQL, facilitando el mantenimiento del código y posibilitando futuros cambios en la tecnología de persistencia sin afectar la lógica de negocio.

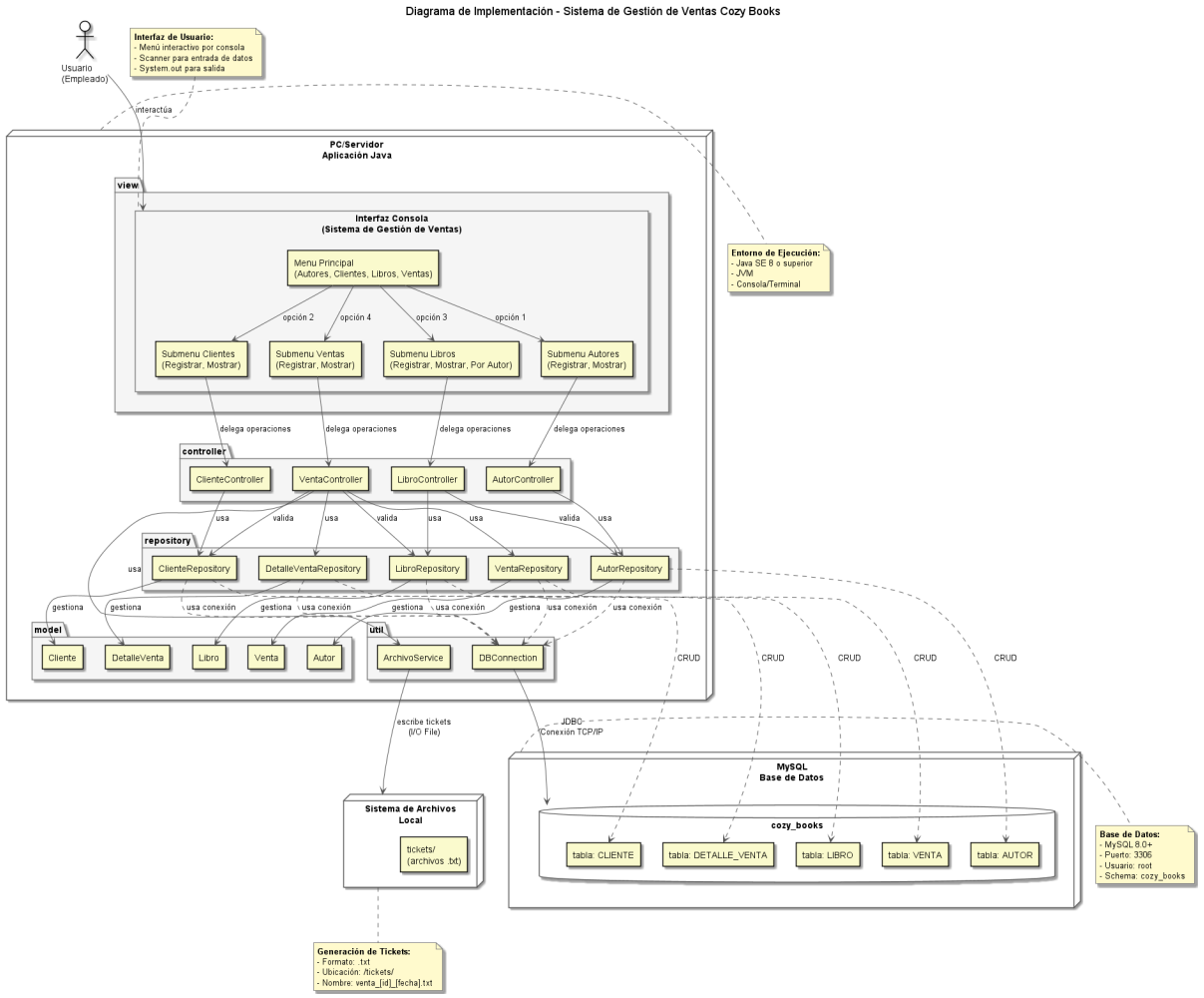
Relaciones

- Un Autor puede escribir múltiples Libros (relación 1:N), y un Cliente puede realizar múltiples Ventas (relación 1:N), representando relaciones de tipo "uno a muchos" que reflejan la naturaleza del negocio donde cada autor tiene un catálogo de obras y cada cliente mantiene un historial de compras.
- Una Venta contiene obligatoriamente uno o más DetalleVenta (relación de composición 1:N), y cada DetalleVenta referencia exactamente un Libro (relación N:1). Esta estructura permite modelar la relación muchos-a-muchos entre ventas y libros, donde una venta puede incluir múltiples libros y un libro puede aparecer en múltiples ventas, manteniendo además el precio histórico de cada transacción.
- La clase Libro implementa una estructura de tabla única con campos condicionales según el tipo_libro (FISICO, DIGITAL, AUDIOLIBRO), lo que permite gestionar diferentes formatos de productos sin necesidad de herencia de clases, simplificando el modelo y optimizando las consultas a la base de datos.
- Los controladores (AutorController, ClienteController, LibroController, VentaController) dependen de sus respectivos repositorios para acceder a los datos. LibroController consulta además AutorRepository para validar la existencia del autor antes de registrar un libro, y VentaController consulta ClienteRepository y LibroRepository para validar clientes y libros antes de procesar una venta, además de utilizar ArchivoService para generar tickets. Esta arquitectura en capas garantiza la separación de responsabilidades y facilita el mantenimiento del sistema.

Todo este diseño modular y orientado a objetos facilita la evolución del sistema y la integración de nuevas funcionalidades en el futuro.

Implementación

Diagrama de implementación



Nodos principales

- **PC Cliente**: Representa la computadora donde se ejecuta la aplicación de escritorio.
- **Aplicación Java:** El sistema de gestión de ventas, empaquetado como un archivo ejecutable (App.jar).
- **MySQL:** Sistema de gestión de base de datos relacional que almacena la información del sistema.

Componentes de la aplicación

- **Interfaz de Usuario:** Capa de presentación basada en consola, agrupada en el paquete view. Proporciona un menú interactivo jerárquico que permite al usuario navegar entre las diferentes funcionalidades del sistema mediante entrada por teclado (Scanner) y salida estándar (System.out). La estructura de navegación incluye un Menú Principal con cuatro opciones (Autores, Clientes, Libros, Ventas), cada una con su respectivo submenú especializado que contiene las operaciones específicas de cada módulo.
- **Controladores:** Implementan la lógica de negocio y están agrupados en el paquete controller (AutorController, ClienteController, LibroController, VentaController). Cada controlador es

especializado en su entidad correspondiente y maneja directamente las operaciones CRUD completas, validaciones de negocio (documentos únicos, stock disponible, existencia de autores, etc.) y la coordinación con los repositorios. La arquitectura elimina la necesidad de un controlador coordinador, permitiendo que la vista delegue directamente a cada controlador especializado.

- **Modelo de Datos:** Clases que representan las entidades del sistema, agrupadas en el paquete model (Autor, Cliente, Libro, Venta, DetalleVenta). Estas clases contienen únicamente los atributos que reflejan la estructura de las tablas de la base de datos según el diseño del ERD, sin lógica de negocio. Libro implementa una estructura de tabla única con campos condicionales para los tres tipos de formato (físico, digital, audiolibro).
- **Repositorios:** Encargados del acceso y persistencia de datos, agrupados en el paquete repository (AutorRepository, ClienteRepository, LibroRepository, VentaRepository, DetalleVentaRepository). Cada repository implementa las cinco operaciones estándar: registrar, actualizar, eliminar, listar y buscar. Utilizan JDBC para conectarse directamente a MySQL, abstraen completamente las operaciones SQL y proporcionan una interfaz orientada a objetos para la persistencia. Los repositorios manejan automáticamente las claves generadas, las transacciones y el manejo de errores de base de datos.
- **Utilidades:** Componentes auxiliares agrupados en el paquete util. DBConnection gestiona las conexiones con MySQL mediante JDBC, incluyendo la configuración automática de la base de datos y la verificación de conectividad. ArchivoService se encarga de generar y escribir los tickets de venta en formato de archivo de texto (.txt) en el sistema de archivos local, almacenándolos con nomenclatura estandarizada (ticket1.txt, ticket2.txt, etc.).

Estructura de la base de datos

Tablas:

AUTOR, CLIENTE, LIBRO, VENTA, DETALLE_VENTA. Cada tabla representa una entidad clave del dominio de gestión de ventas de la librería Cozy Books.

Relaciones:

- Un autor puede escribir múltiples libros (relación uno a muchos), pero cada libro pertenece a un único autor. La relación se establece mediante la clave foránea id_autor en la tabla LIBRO.
- Un cliente puede realizar múltiples ventas (relación uno a muchos), pero cada venta pertenece a un único cliente. La relación se establece mediante la clave foránea id_cliente en la tabla VENTA.
- Una venta contiene obligatoriamente uno o más detalles de venta (relación uno a muchos de composición), y cada detalle pertenece a una única venta. La relación se establece mediante la clave foránea id_venta en la tabla DETALLE_VENTA.
- Un libro puede aparecer en múltiples detalles de venta a lo largo del tiempo (relación muchos a uno), permitiendo mantener un historial de ventas por producto. Cada detalle referencia un libro mediante la clave foránea id_libro en DETALLE_VENTA, guardando además el precio unitario al momento de la transacción para mantener la integridad histórica de los datos ante futuros cambios de precio.
- La tabla DETALLE_VENTA actúa como tabla intermedia que resuelve la relación muchos a muchos entre VENTA y LIBRO, almacenando información adicional específica de cada línea de venta (cantidad, precio unitario, subtotal).

Relaciones importantes

La aplicación Java se ejecuta sobre el JRE (Java Runtime Environment) en el PC del usuario, proporcionando una interfaz de consola interactiva para el personal de Cozy Books.

La interfaz de usuario (paquete view) captura las entradas del usuario mediante menús interactivos y delega las operaciones al ShopController, que coordina la ejecución distribuyéndola entre los controladores especializados (AutorController, ClienteController, LibroController, VentaController).

Los controladores implementan la lógica de negocio, realizan validaciones (documentos únicos, stock disponible, existencia de relaciones), manipulan las entidades del modelo de datos y utilizan los repositorios para acceder a la capa de persistencia. VentaController además coordina operaciones complejas que involucran múltiples repositorios y delega la generación de tickets a ArchivoService.

Los repositorios (paquete repository) abstraen completamente las operaciones SQL, implementando las cinco operaciones estándar (registrar, actualizar, eliminar, listar, buscar) para cada entidad. Utilizan DBConnection para obtener conexiones a la base de datos y ejecutar sentencias SQL mediante JDBC.

La conexión a la base de datos MySQL se realiza mediante JDBC (Java Database Connectivity), gestionada centralizadamente por el componente DBConnection del paquete util, que maneja el pool de conexiones, parámetros de conexión y control de transacciones.

Las tablas mantienen integridad referencial mediante claves foráneas: LIBRO referencia a AUTOR mediante id_autor, VENTA referencia a CLIENTE mediante id_cliente, y DETALLE_VENTA referencia tanto a VENTA (id_venta) como a LIBRO (id_libro), asegurando la consistencia de los datos y previniendo operaciones que violen las relaciones establecidas.

ArchivoService interactúa con el sistema de archivos local para generar tickets de venta en formato de texto plano (.txt), almacenándolos en el directorio /tickets/ con nomenclatura que incluye el ID de venta y la fecha de la transacción.

Pruebas

Introducción

En esta etapa, se presenta el plan de pruebas para el Sistema de Ventas para Cozy Books. Se detallan las estrategias, objetivos, recursos y casos de prueba específicos para validar la funcionalidad del sistema previo a su implementación en producción.

Alcance

Nos centraremos en particular en los casos de uso desarrollados anteriormente:

- Registrar libro (CU11)
- Registrar venta (CU16)

Estrategia

Se aplicarán los siguientes tipos de pruebas:

- Pruebas unitarias: Validación de componentes individuales
- Pruebas de integración: Validación de integración entre componentes
- Pruebas funcionales: Validación de funcionalidades completas del sistema
- Pruebas de aceptación: Validación de requisitos de usuario

Casos de prueba

-Registrar libro (CU11)

ID	**CP-CU11-01**
Caso de Prueba	Registro de libro con datos válidos
Caso de Uso	Registrar libro (CU11)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. El autor del libro ya existe en la base de datos (id_autor válido) 4. El ISBN del libro no existe previamente en el sistema
Datos de Entrada	● Título: "Cien Años de Soledad" ● ISBN: "978-0307474728" ● Editorial: "Sudamericana" ● Año: 1967 ● Precio: 15500.00 ● Género: "Realismo Mágico" ● Tipo de libro: "FISICO" ● Stock: 25 ● ID Autor: 5 (Gabriel García Márquez - existente) ● Encuadernado: "Tapa dura" ● Número de edición: 1
Pasos	1. Acceder al menú principal 2. Seleccionar opción "3. Registrar Libro" 3. Seleccionar tipo de libro "1. Físico" 4. Ingresar el ID del autor (5) 5. Ingresar título del libro 6. Ingresar ISBN 7. Ingresar editorial 8. Ingresar año de publicación 9. Ingresar precio 10. Ingresar género 11. Ingresar stock inicial 12. Ingresar tipo de encuadernado

	13. Ingresar número de edición 14. Confirmar el registro
Resultado Esperado	Libro registrado exitosamente con mensaje de confirmación "Libro registrado exitosamente (id: [id_libro generado])"
Criterio de Aceptación	● El libro debe aparecer en la tabla LIBRO de la base de datos con todos los campos correctamente almacenados ● El libro debe poder ser consultado posteriormente mediante la opción "7. Mostrar Libros" ● Los campos específicos de libro físico (encuadernado, num_edicion) deben estar completos ● Los campos de libro digital y audiolibro deben quedar NULL ● El campo tipo_libro debe ser "FISICO"
Prioridad	Alta

ID	**CP-CU11-02**
Caso de Prueba	Registro de libro con autor inexistente
Caso de Uso	Registrar libro (CU11)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema
Datos de Entrada	● ID Autor: 999 (no existe en la base de datos) ● Título: "Libro de Prueba" ● ISBN: "978-1234567890" ● Editorial: "Editorial Test" ● Año: 2023 ● Precio: 10000.00 ● Género: "Ficción" ● Tipo de libro: "FISICO" ● Stock: 10 ● Encuadernado: "Tapa blanda" ● Número de edición: 1
Pasos	1. Acceder al menú principal 2. Seleccionar opción "3. Registrar Libro" 3. Seleccionar tipo de libro "1. Físico" 4. Ingresar ID de autor inexistente (999) 5. Intentar continuar con el registro
Resultado Esperado	El sistema muestra mensaje de error: "Error: Autor no existe" y no permite continuar con el registro del libro

Criterio de Aceptación	● El libro NO debe ser registrado en la base de datos ● El sistema debe mostrar un mensaje de error claro ● El sistema debe retornar al menú anterior sin crear registros huérfanos ● La validación debe ocurrir antes de solicitar los demás datos del libro
Prioridad	Alta

ID	**CP-CU11-03**
Caso de Prueba	Registro de libro con ISBN duplicado
Caso de Uso	Registrar libro (CU11)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. El autor del libro existe en la base de datos 4. Ya existe un libro con el ISBN "978-0307474728" en el sistema
Datos de Entrada	● Título: "Nuevo Libro" ● ISBN: "978-0307474728" (duplicado) ● Editorial: "Nueva Editorial" ● Año: 2024 ● Precio: 12000.00 ● Género: "Novela" ● Tipo de libro: "FISICO" ● Stock: 15 ● ID Autor: 5 (existente) ● Encuadernado: "Tapa dura" ● Número de edición: 2
Pasos	1. Acceder al menú principal 2. Seleccionar opción "3. Registrar Libro" 3. Seleccionar tipo de libro "1. Físico" 4. Ingresar el ID del autor 5. Ingresar título del libro 6. Ingresar ISBN duplicado 7. Intentar continuar con el registro
Resultado Esperado	El sistema muestra mensaje de error: "Error: ISBN ya existe en el sistema" y no permite completar el registro
Criterio de Aceptación	● El libro NO debe ser registrado en la base de datos ● El sistema debe validar la unicidad del ISBN antes de confirmar el registro ● El sistema debe mostrar un mensaje de error descriptivo
Prioridad	Alta

ID	**CP-CU11-04**
Caso de Prueba	Registro de libro con precio inválido (negativo)
Caso de Uso	Registrar libro (CU11)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. El autor del libro existe en la base de datos
Datos de Entrada	● Título: "Libro Test" ● ISBN: "978-9876543210" ● Editorial: "Editorial Prueba" ● Año: 2024 ● Precio: -500.00 (valor inválido) ● Género: "Ficción" ● Tipo de libro: "FISICO" ● Stock: 10 ● ID Autor: 5 (existente) ● Encuadernado: "Tapa blanda" ● Número de edición: 1
Pasos	1. Acceder al menú principal 2. Seleccionar opción "3. Registrar Libro" 3. Seleccionar tipo de libro "1. Físico" 4. Ingresar el ID del autor 5. Ingresar título del libro 6. Ingresar ISBN 7. Ingresar editorial 8. Ingresar año 9. Ingresar precio negativo (-500.00) 10. Intentar continuar
Resultado Esperado	El sistema muestra mensaje de error: "Error: El precio debe ser mayor a 0" y solicita reingresar el precio
Criterio de Aceptación	● El libro NO debe ser registrado con precio negativo o cero ● El sistema debe validar que precio > 0 ● El sistema debe permitir reingresar el dato correcto sin perder la información ya ingresada ● La validación debe ocurrir inmediatamente al ingresar el precio
Prioridad	Media

ID	**CP-CU11-05**

Caso de Prueba	Registro de libro físico con stock negativo
Caso de Uso	Registrar libro (CU11)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. El autor del libro existe en la base de datos
Datos de Entrada	● Título: "Libro Stock Test" ● ISBN: "978-1111111111" ● Editorial: "Editorial Test" ● Año: 2024 ● Precio: 10000.00 ● Género: "Técnico" ● Tipo de libro: "FISICO" ● Stock: -5 (valor inválido) ● ID Autor: 3 (existente) ● Encuadernado: "Tapa dura" ● Número de edición: 1
Pasos	1. Acceder al menú principal 2. Seleccionar opción "3. Registrar Libro" 3. Seleccionar tipo de libro "1. Físico" 4. Ingresar todos los datos requeridos 5. Ingresar stock negativo (-5) 6. Intentar confirmar el registro
Resultado Esperado	El sistema muestra mensaje de error: "Error: El stock debe ser mayor o igual a 0" y solicita reingresar el stock
Criterio de Aceptación	● El libro NO debe ser registrado con stock negativo ● El sistema debe validar que stock >= 0 para libros físicos ● El sistema debe permitir stock = 0 (producto sin existencias) ● Para libros digitales y audiolibros, el stock no debe ser requerido.
Prioridad	Media

-Registrar venta (CU16)

ID	**CP-CU16-01**
Caso de Prueba	Registro de venta exitosa con múltiples libros
Caso de Uso	Registrar venta (CU16)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. Existe al menos un cliente registrado en el sistema (id_cliente válido)

	4. Existen libros disponibles en el catálogo con stock suficiente (para físicos)
Datos de Entrada	● ID Cliente: 15 (María González - existente) ● Libros a vender: ○ Libro 1: ID 5 (Físico - "Cien Años de Soledad"), Cantidad: 2, Precio unitario: 15500.00 ○ Libro 2: ID 12 (Digital - "Clean Code"), Cantidad: 1, Precio unitario: 8900.00 ○ Libro 3: ID 8 (Audiolibro - "Sapiens"), Cantidad: 1, Precio unitario: 12500.00 ● Método de pago: "TARJETA" ● Monto total calculado: 52400.00
Pasos	1. Acceder al menú principal 2. Seleccionar opción "4. Registrar Venta" 3. Ingresar ID del cliente (15) 4. Sistema muestra datos del cliente para confirmar 5. Agregar primer libro: ID 5, cantidad 2 6. Sistema confirma agregado y muestra subtotal 7. Indicar que desea agregar más libros (S) 8. Agregar segundo libro: ID 12, cantidad 1 9. Indicar que desea agregar más libros (S) 10. Agregar tercer libro: ID 8, cantidad 1 11. Indicar que NO desea agregar más libros (N) 12. Sistema muestra resumen de la venta y monto total 13. Ingresar método de pago "TARJETA" 14. Confirmar la venta
Resultado Esperado	● Venta registrada exitosamente con mensaje "Venta registrada exitosamente (id: [id_venta], monto: $52400.00)" ● Sistema genera y guarda ticket en archivo .txt automáticamente ● Mensaje de confirmación: "Ticket generado: /tickets/venta_[id][fecha].txt"
Criterio de Aceptación	1. La venta debe aparecer en la tabla VENTA con estado "COMPLETADA" 2. Se deben crear 3 registros en DETALLE_VENTA (uno por cada libro) 3. El stock del libro físico (ID 5) debe reducirse en 2 unidades 4. El stock de libros digitales y audiolibros no debe modificarse 5. El archivo de ticket debe generarse en el directorio /tickets/ 6. El ticket debe contener: fecha, datos del cliente, detalle de libros, cantidades, precios, subtotales y total 7. La venta debe poder consultarse posteriormente
Prioridad	Alta

ID	**CP-CU16-02**

Caso de Prueba	Intento de venta con cliente inexistente
Caso de Uso	Registrar venta (CU16)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema
Datos de Entrada	● ID Cliente: 999 (no existe en la base de datos)
Pasos	1. Acceder al menú principal 2. Seleccionar opción "4. Registrar Venta" 3. Ingresar ID del cliente inexistente (999) 4. Intentar continuar con el registro
Resultado Esperado	El sistema muestra mensaje de error: "Error: Cliente no existe" y retorna al menú principal sin iniciar la venta
Criterio de Aceptación	1. NO se debe crear ningún registro en la tabla VENTA 2. NO se debe crear ningún registro en DETALLE_VENTA 3. El sistema debe validar la existencia del cliente antes de solicitar libros 4. El mensaje de error debe ser claro y descriptivo 5. El sistema debe permitir reintentar con un ID válido
Prioridad	Alta

ID	**CP-CU16-03**
Caso de Prueba	Intento de venta con libro inexistente
Caso de Uso	Registrar venta (CU16)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. Existe un cliente registrado en el sistema
Datos de Entrada	● ID Cliente: 15 (María González - existente) ● ID Libro: 888 (no existe en la base de datos) ● Cantidad: 1
Pasos	1. Acceder al menú principal 2. Seleccionar opción "4. Registrar Venta" 3. Ingresar ID del cliente válido (15) 4. Sistema muestra datos del cliente 5. Intentar agregar libro con ID inexistente (888) 6. Ingresar cantidad: 1

Resultado Esperado	El sistema muestra mensaje de error: "Error: Libro no existe" y solicita ingresar otro ID de libro válido sin perder el progreso de la venta
Criterio de Aceptación	● El sistema debe validar la existencia del libro antes de agregarlo al carrito ● La venta no debe cancelarse completamente, solo rechazar el libro inválido ● El sistema debe permitir continuar agregando otros libros válidos ● No se debe crear el detalle de venta con el libro inexistente
Prioridad	Alta

ID	**CP-CU16-04**
Caso de Prueba	Intento de venta de libro físico con stock insuficiente
Caso de Uso	Registrar venta (CU16)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. Existe un cliente registrado en el sistema 4. Existe un libro físico con stock = 3 unidades (ID 20)
Datos de Entrada	● ID Cliente: 10 (Juan Pérez - existente) ● ID Libro: 20 (Físico - stock actual: 3) ● Cantidad solicitada: 5 (mayor al stock disponible)
Pasos	1. Acceder al menú principal 2. Seleccionar opción "4. Registrar Venta" 3. Ingresar ID del cliente válido (10) 4. Sistema muestra datos del cliente 5. Intentar agregar libro físico ID 20 6. Ingresar cantidad: 5 7. Intentar confirmar
Resultado Esperado	El sistema muestra mensaje de error: "Error: Stock insuficiente. Stock disponible: 3" y solicita ingresar una cantidad válida o seleccionar otro libro
Criterio de Aceptación	● El sistema debe validar que la cantidad solicitada <= stock disponible (solo para libros físicos) ● El libro no debe agregarse al detalle con cantidad mayor al stock ● El sistema debe mostrar el stock disponible en el mensaje de error ● Para libros digitales y audiolibros, NO debe validar stock (ilimitado) ● La venta no debe procesarse si hay items con stock insuficiente
Prioridad	Alta

ID	**CP-CU16-05**
Caso de Prueba	Registro de venta exitosa solo con libros digitales (sin restricción de stock)
Caso de Uso	Registrar venta (CU16)
Precondiciones	1. El sistema está operativo 2. El usuario tiene acceso al sistema 3. Existe un cliente registrado en el sistema 4. Existen libros digitales y/o audiolibros en el catálogo
Datos de Entrada	● ID Cliente: 25 (Ana Martínez - existente) ● Libros a vender: ○ Libro 1: ID 30 (Digital - "JavaScript: The Good Parts"), Cantidad: 1, Precio: 7500.00 ○ Libro 2: ID 42 (Audiolibro - "Atomic Habits"), Cantidad: 2, Precio: 9800.00 ● Método de pago: "EFECTIVO" ● Monto total calculado: 27100.00
Pasos	1. Acceder al menú principal 2. Seleccionar opción "4. Registrar Venta" 3. Ingresar ID del cliente (25) 4. Sistema muestra datos del cliente 5. Agregar libro digital: ID 30, cantidad 1 6. Indicar que desea agregar más libros (S) 7. Agregar audiolibro: ID 42, cantidad 2 8. Indicar que NO desea agregar más libros (N) 9. Sistema muestra resumen de la venta 10. Ingresar método de pago "EFECTIVO" 11. Confirmar la venta
Resultado Esperado	● Venta registrada exitosamente con mensaje de confirmación ● Sistema genera ticket automáticamente ● NO se actualiza stock (libros digitales/audiolibros no tienen restricción)
Criterio de Aceptación	● La venta debe aparecer en la tabla VENTA con estado "COMPLETADA" ● Se deben crear 2 registros en DETALLE_VENTA ● El campo stock de los libros digitales/audiolibros NO debe modificarse ● El sistema NO debe validar stock para tipo_libro = "DIGITAL" o "AUDIOLIBRO" ● El ticket debe generarse correctamente con todos los detalles ● La venta debe poder consultarse en el historial
Prioridad	Alta

Definición de base de datos para el sistema

A continuación se detalla la estructura de tablas, relaciones, y restricciones que soportarán las funcionalidades del sistema.

Sistema de gestión de Base de Datos

Para la implementación se utilizará MySQL 8.0, un sistema de gestión de bases de datos relacional de código abierto ampliamente utilizado, que ofrece un buen equilibrio entre rendimiento, facilidad de uso y características necesarias para una PYME.

Tablas

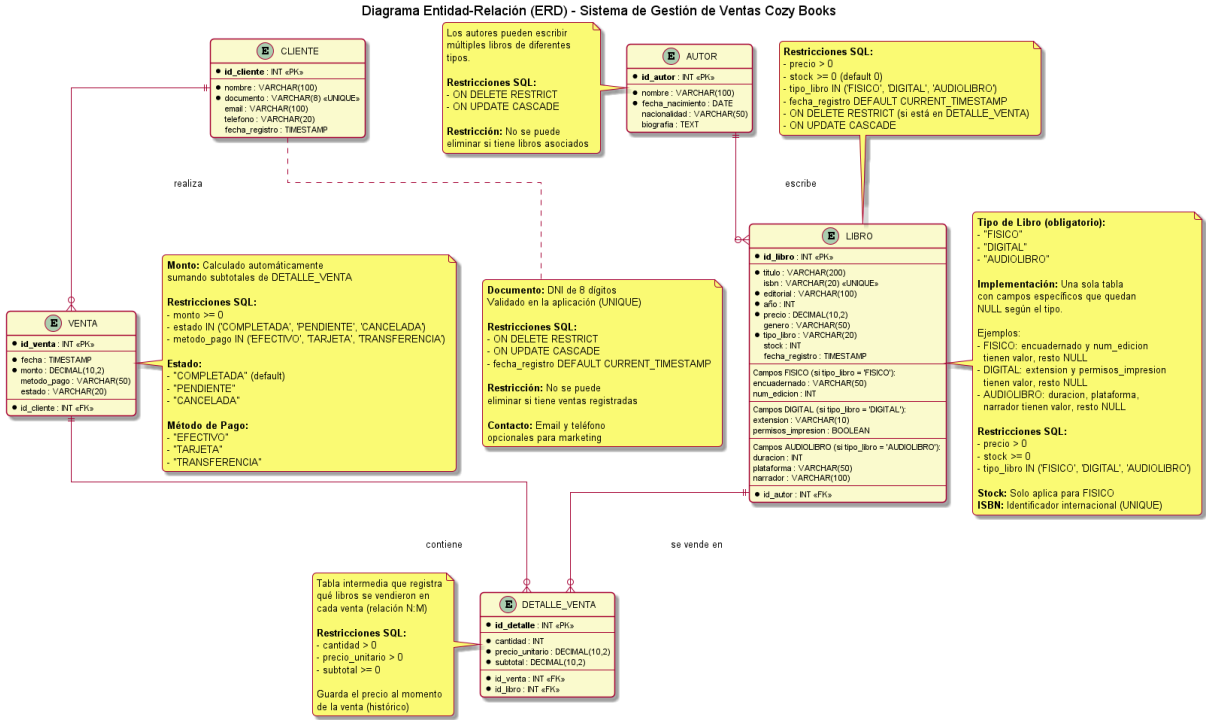
Tabla	Campo	Tipo de Dato	Descripción
AUTOR	id_autor	INT, PK, AUTO	Identificador único del autor
	nombre	VARCHAR(100)	Nombre completo del autor
	fecha_nacimiento	DATE	Fecha de nacimiento del autor
	nacionalidad	VARCHAR(50)	Nacionalidad del autor
	biografia	TEXT	Biografía del autor
CLIENTE	id_cliente	INT, PK, AUTO	Identificador único del cliente
	nombre	VARCHAR(100)	Nombre completo del cliente
	documento	VARCHAR(8), UNIQUE	DNI del cliente (8 dígitos)
	email	VARCHAR(100)	Correo electrónico del cliente

	telefono	VARCHAR(20)	Teléfono de contacto del cliente
	fecha_registro	TIMESTAMP	Fecha y hora de registro en el sistema
LIBRO	id_libro	INT, PK, AUTO	Identificador único del libro
	titulo	VARCHAR(200)	Título del libro
	isbn	VARCHAR(20), UNIQUE	Código ISBN del libro (identificador internacional)
	editorial	VARCHAR(100)	Editorial que publicó el libro
	año	INT	Año de publicación del libro
	precio	DECIMAL(10,2)	Precio de venta del libro
	genero	VARCHAR(50)	Género literario del libro
	tipo_libro	VARCHAR(20)	Tipo de formato: 'FISICO', 'DIGITAL' o 'AUDIOLIBRO'
	stock	INT	Cantidad disponible en inventario (solo para físicos)
	fecha_registro	TIMESTAMP	Fecha y hora de registro del libro en el sistema
	id_autor	INT, FK	Referencia al autor del libro (clave foránea a AUTOR)

	encuadernado	VARCHAR(50)	Tipo de encuadernado (solo si tipo_libro = 'FISICO')
	num_edicion	INT	Número de edición (solo si tipo_libro = 'FISICO')
	extension	VARCHAR(10)	Formato del archivo digital (solo si tipo_libro = 'DIGITAL')
	permisos_impresion	BOOLEAN	Indica si permite impresión (solo si tipo_libro = 'DIGITAL')
	duracion	INT	Duración en minutos (solo si tipo_libro = 'AUDIOLIBRO')
	plataforma	VARCHAR(50)	Plataforma de distribución (solo si tipo_libro = 'AUDIOLIBRO')
	narrador	VARCHAR(100)	Nombre del narrador (solo si tipo_libro = 'AUDIOLIBRO')
VENTA	id_venta	INT, PK, AUTO	Identificador único de la venta
	fecha	TIMESTAMP	Fecha y hora de la transacción de venta
	monto	DECIMAL(10,2)	Monto total de la venta
	metodo_pago	VARCHAR(50)	Método de pago: 'EFECTIVO', 'TARJETA' o 'TRANSFERENCIA'

	estado	VARCHAR(20)	Estado de la venta: 'COMPLETADA', 'PENDIENTE' o 'CANCELADA'
	id_cliente	INT, FK	Referencia al cliente que realizó la compra (clave foránea a CLIENTE)
DETALLE_VENTA	id_detalle	INT, PK, AUTO	Identificador único del detalle de venta
	cantidad	INT	Cantidad de unidades vendidas del libro
	precio_unitario	DECIMAL(10,2)	Precio del libro al momento de la venta (histórico)
	subtotal	DECIMAL(10,2)	Subtotal calculado (cantidad × precio_unitario)
	id_venta	INT, FK	Referencia a la venta (clave foránea a VENTA)
	id_libro	INT, FK	Referencia al libro vendido (clave foránea a LIBRO)

Diagrama entidad-relación



Creación de tablas SQL

CREATE SCHEMA `cozy_books` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci ;

✓	2	02:15:16	Apply changes to cozy_books	Changes applied

USE cozy_books;

✓	4	02:18:01	USE cozy_books	0 row(s) affected

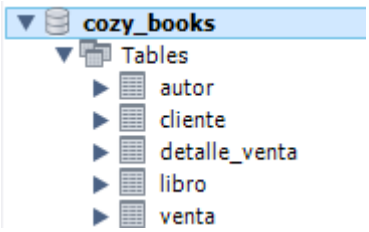
```
-- ========================================
-- SISTEMA DE GESTIÓN DE VENTAS - COZY BOOKS
-- Base de Datos: MySQL 8.0
-- ========================================

USE cozy_books;

-- ========================================
-- TABLA: AUTOR
-- ========================================
CREATE TABLE AUTOR (
    id_autor INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    nacionalidad VARCHAR(50),
    biografia TEXT,
    INDEX idx_autor_nombre (nombre)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-- ============================================
-- TABLA: CLIENTE
-- ============================================
CREATE TABLE CLIENTE (
    id_cliente INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    documento VARCHAR(8) NOT NULL UNIQUE,
    email VARCHAR(100),
    telefono VARCHAR(20),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_cliente_nombre (nombre),
    INDEX idx_cliente_documento (documento)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- TABLA: LIBRO
-- ============================================
CREATE TABLE LIBRO (
    id_libro INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(200) NOT NULL,
    isbn VARCHAR(20) UNIQUE,
    editorial VARCHAR(100) NOT NULL,
    año INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL CHECK (precio > 0),
    genero VARCHAR(50),
    tipo_libro VARCHAR(20) NOT NULL CHECK (tipo_libro IN ('FISICO', 'DIGITAL', 'AUDIOLIBRO')),
    stock INT DEFAULT 0 CHECK (stock >= 0),
    fecha_registro TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    -- Campos para LIBRO FISICO
    encuadernado VARCHAR(50),
    num_edicion INT,

    -- Campos para LIBRO DIGITAL
    extension VARCHAR(10),
    permisos_impresion BOOLEAN,

    -- Campos para AUDIOLIBRO
    duracion INT,
    plataforma VARCHAR(50),
    narrador VARCHAR(100),

    -- Clave foránea
    id_autor INT NOT NULL,

    INDEX idx_libro_titulo (titulo),
    INDEX idx_libro_isbn (isbn),
    INDEX idx_libro_tipo (tipo_libro),
    INDEX idx_libro_autor (id_autor),
```

```
    CONSTRAINT fk_libro_autor
        FOREIGN KEY (id_autor)
        REFERENCES AUTOR(id_autor)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================
-- TABLA: VENTA
-- ========================================
CREATE TABLE VENTA (
    id_venta INT AUTO_INCREMENT PRIMARY KEY,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    monto DECIMAL(10,2) NOT NULL CHECK (monto >= 0),
    metodo_pago VARCHAR(50) CHECK (metodo_pago IN ('EFECTIVO', 'TARJETA', 'TRANSFERENCIA')),
    estado VARCHAR(20) DEFAULT 'COMPLETADA' CHECK (estado IN ('COMPLETADA', 'PENDIENTE',
'CANCELADA')),

    -- Clave foránea
    id_cliente INT NOT NULL,

    INDEX idx_venta_fecha (fecha),
    INDEX idx_venta_cliente (id_cliente),
    INDEX idx_venta_estado (estado),

    CONSTRAINT fk_venta_cliente
        FOREIGN KEY (id_cliente)
        REFERENCES CLIENTE(id_cliente)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ========================================
-- TABLA: DETALLE_VENTA
-- ========================================
CREATE TABLE DETALLE_VENTA (
    id_detalle INT AUTO_INCREMENT PRIMARY KEY,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    precio_unitario DECIMAL(10,2) NOT NULL CHECK (precio_unitario > 0),
    subtotal DECIMAL(10,2) NOT NULL CHECK (subtotal >= 0),

    -- Claves foráneas
    id_venta INT NOT NULL,
    id_libro INT NOT NULL,

    INDEX idx_detalle_venta (id_venta),
    INDEX idx_detalle_libro (id_libro),

    CONSTRAINT fk_detalle_venta
```

```
    FOREIGN KEY (id_venta)
    REFERENCES VENTA(id_venta)
    ON DELETE CASCADE
    ON UPDATE CASCADE,

  CONSTRAINT fk_detalle_libro
    FOREIGN KEY (id_libro)
    REFERENCES LIBRO(id_libro)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================
-- COMENTARIOS EN TABLAS
-- ============================================

ALTER TABLE AUTOR
COMMENT = 'Almacena información de los autores de libros';

ALTER TABLE CLIENTE
COMMENT = 'Almacena información de los clientes de la librería';

ALTER TABLE LIBRO
COMMENT = 'Tabla única que almacena todos los tipos de libros (físicos, digitales y audiolibros)';

ALTER TABLE VENTA
COMMENT = 'Registra las transacciones de venta realizadas';

ALTER TABLE DETALLE_VENTA
COMMENT = 'Tabla intermedia que relaciona ventas con libros (N:M) y guarda precio histórico';
```

6	02:18:42	CREATE TABLE AUTOR (id_autor INT AUTO_INCREMENT PRIMARY KEY, no...	0 row(s) affected		0.047 sec
7	02:18:42	CREATE TABLE CLIENTE (id_cliente INT AUTO_INCREMENT PRIMARY KEY, ...	0 row(s) affected		0.031 sec
8	02:18:42	CREATE TABLE LIBRO (id_libro INT AUTO_INCREMENT PRIMARY KEY, titulo ...	0 row(s) affected		0.031 sec
9	02:18:42	CREATE TABLE VENTA (id_venta INT AUTO_INCREMENT PRIMARY KEY, fe...	0 row(s) affected		0.047 sec
10	02:18:42	CREATE TABLE DETALLE_VENTA (id_detalle INT AUTO_INCREMENT PRIMAR...	0 row(s) affected		0.047 sec
11	02:18:42	ALTER TABLE AUTOR COMMENT = 'Almacena información de los autores de libros'	0 row(s) affected	Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
12	02:18:42	ALTER TABLE CLIENTE COMMENT = 'Almacena información de los clientes de la libr...	0 row(s) affected	Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
13	02:18:42	ALTER TABLE LIBRO COMMENT = 'Tabla única que almacena todos los tipos de libr...	0 row(s) affected	Records: 0 Duplicates: 0 Warnings: 0	0.016 sec
14	02:18:42	ALTER TABLE VENTA COMMENT = 'Registra las transacciones de venta realizadas'	0 row(s) affected	Records: 0 Duplicates: 0 Warnings: 0	0.031 sec
15	02:18:42	ALTER TABLE DETALLE_VENTA COMMENT = 'Tabla intermedia que relaciona vent...	0 row(s) affected	Records: 0 Duplicates: 0 Warnings: 0	0.000 sec



Inserción, consulta y borrado de registros

```
-- ============================================
-- SECCIÓN 1: INSERCIÓN (INSERT)
-- Operaciones para registrar nuevos datos
-- ============================================

-- ------------------------------------------
-- CU01: REGISTRAR AUTOR
-- ------------------------------------------
-- Insertar un nuevo autor en el sistema con todos los datos
INSERT INTO AUTOR (nombre, fecha_nacimiento, nacionalidad, biografia)
VALUES ('Gabriel García Márquez', '1927-03-06', 'Colombia', 'Premio Nobel de Literatura 1982');

-- Insertar autor con información mínima (solo campos obligatorios)
INSERT INTO AUTOR (nombre, fecha_nacimiento)
VALUES ('Roberto Bolaño', '1953-04-28');

-- Verificar que el autor fue insertado
SELECT * FROM AUTOR WHERE nombre = 'Gabriel García Márquez';
SELECT * FROM AUTOR WHERE nombre = 'Roberto Bolaño';


-- ------------------------------------------
-- CU06: REGISTRAR CLIENTE
-- ------------------------------------------
-- Insertar un nuevo cliente con todos los datos
INSERT INTO CLIENTE (nombre, documento, email, telefono)
VALUES ('Juan Pérez', '12345678', 'juan.perez@email.com', '011-1234-5678');

-- Insertar cliente con información mínima (solo campos obligatorios)
INSERT INTO CLIENTE (nombre, documento)
VALUES ('María González', '23456789');

-- IMPORTANTE: Verificar que el documento sea único ANTES de insertar
SELECT COUNT(*) as existe
FROM CLIENTE
WHERE documento = '12345678';

-- Verificar que los clientes fueron insertados
SELECT * FROM CLIENTE WHERE documento = '12345678';
SELECT * FROM CLIENTE WHERE documento = '23456789';


-- ------------------------------------------
-- CU11: REGISTRAR LIBRO
-- ------------------------------------------
-- Registrar libro FÍSICO
INSERT INTO LIBRO (
    titulo, isbn, editorial, año, precio, genero,
```

```
    tipo_libro, stock, id_autor,
    encuadernado, num_edicion
) VALUES (
    'Cien Años de Soledad',
    '978-0307474728',
    'Sudamericana',
    1967,
    15500.00,
    'Realismo Mágico',
    'FISICO',
    25,
    1,
    'Tapa dura',
    1
);

-- Registrar libro DIGITAL
INSERT INTO LIBRO (
    titulo, isbn, editorial, año, precio, genero,
    tipo_libro, stock, id_autor,
    extension, permisos_impresion
) VALUES (
    'Clean Code',
    '978-0132350884',
    'Prentice Hall',
    2008,
    8900.00,
    'Programación',
    'DIGITAL',
    0,
    1,
    'PDF',
    true
);

-- Registrar AUDIOLIBRO
INSERT INTO LIBRO (
    titulo, isbn, editorial, año, precio, genero,
    tipo_libro, stock, id_autor,
    duracion, plataforma, narrador
) VALUES (
    'Sapiens: De animales a dioses',
    '978-6073149327',
    'Debate',
    2014,
    12500.00,
    'Historia',
    'AUDIOLIBRO',
    0,
    1,
```

```
    945,
    'Audible',
    'Carlos Manuel Vesga'
);

-- IMPORTANTE: Verificar que el ISBN sea único ANTES de insertar
SELECT COUNT(*) as existe
FROM LIBRO
WHERE isbn = '978-0307474728';

-- Verificar que los libros fueron insertados
SELECT * FROM LIBRO WHERE isbn = '978-0307474728';
SELECT * FROM LIBRO WHERE isbn = '978-0132350884';
SELECT * FROM LIBRO WHERE isbn = '978-6073149327';


-- ------------------------------------------
-- CU16: REGISTRAR VENTA (Proceso complejo)
-- ------------------------------------------
-- Este es un proceso que involucra múltiples tablas y requiere transacción

-- Paso 1: Verificar que el cliente existe
SELECT id_cliente, nombre
FROM CLIENTE
WHERE id_cliente = 1;

-- Paso 2: Verificar que los libros existen y hay stock (para físicos)
SELECT id_libro, titulo, tipo_libro, stock, precio
FROM LIBRO
WHERE id_libro IN (1, 2, 3);

-- Paso 3: Validar stock para libros físicos
SELECT
    id_libro,
    titulo,
    stock,
    CASE
        WHEN tipo_libro = 'FISICO' AND stock >= 2 THEN 'OK'
        WHEN tipo_libro IN ('DIGITAL', 'AUDIOLIBRO') THEN 'OK'
        ELSE 'Stock insuficiente'
    END as validacion
FROM LIBRO
WHERE id_libro = 1;

-- Paso 4: Insertar la venta (CON TRANSACCIÓN)
START TRANSACTION;

-- Insertar registro de venta
INSERT INTO VENTA (fecha, monto, metodo_pago, estado, id_cliente)
VALUES (NOW(), 52400.00, 'TARJETA', 'COMPLETADA', 1);
```

```
-- Obtener el ID de la venta recién creada
SET @id_venta = LAST_INSERT_ID();

-- Paso 5: Insertar detalles de venta
INSERT INTO DETALLE_VENTA (cantidad, precio_unitario, subtotal, id_venta, id_libro)
VALUES
   (2, 15500.00, 31000.00, @id_venta, 1),  -- Libro físico
   (1, 8900.00, 8900.00, @id_venta, 2),    -- Libro digital
   (1, 12500.00, 12500.00, @id_venta, 3);  -- Audiolibro

-- Paso 6: Actualizar stock (solo para libros físicos)
UPDATE LIBRO
SET stock = stock - 2
WHERE id_libro = 1 AND tipo_libro = 'FISICO';

-- Confirmar transacción
COMMIT;

-- Verificar la venta creada
SELECT
   v.id_venta,
   v.fecha,
   c.nombre as cliente,
   v.monto,
   v.metodo_pago,
   v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.id_venta = @id_venta;

-- Verificar detalles de la venta
SELECT
   dv.id_detalle,
   l.titulo as libro,
   dv.cantidad,
   dv.precio_unitario,
   dv.subtotal
FROM DETALLE_VENTA dv
INNER JOIN LIBRO l ON dv.id_libro = l.id_libro
WHERE dv.id_venta = @id_venta;


-- ==========================================
-- SECCIÓN 2: CONSULTA (SELECT)
-- Operaciones para listar y buscar datos
-- ==========================================

-- ------------------------------------------
-- CU04: LISTAR AUTORES
```

```
-- ------------------------------------------
-- Listar todos los autores
SELECT * FROM AUTOR
ORDER BY nombre;

-- Listar autores con cantidad de libros
SELECT
   a.id_autor,
   a.nombre,
   a.nacionalidad,
   COUNT(l.id_libro) as cantidad_libros
FROM AUTOR a
LEFT JOIN LIBRO l ON a.id_autor = l.id_autor
GROUP BY a.id_autor, a.nombre, a.nacionalidad
ORDER BY cantidad_libros DESC, a.nombre;


-- ------------------------------------------
-- CU05: BUSCAR AUTOR
-- ------------------------------------------
-- Buscar autor por nombre (búsqueda parcial)
SELECT * FROM AUTOR
WHERE nombre LIKE '%García%'
ORDER BY nombre;

-- Buscar autor por ID
SELECT * FROM AUTOR
WHERE id_autor = 1;

-- Buscar autor por nacionalidad
SELECT * FROM AUTOR
WHERE nacionalidad = 'Colombia'
ORDER BY nombre;


-- ------------------------------------------
-- CU09: LISTAR CLIENTES
-- ------------------------------------------
-- Listar todos los clientes
SELECT * FROM CLIENTE
ORDER BY nombre;

-- Listar clientes con su historial de compras
SELECT
   c.id_cliente,
   c.nombre,
   c.documento,
   c.email,
   COUNT(v.id_venta) as total_compras,
   COALESCE(SUM(v.monto), 0) as monto_total_gastado
```

```
FROM CLIENTE c
LEFT JOIN VENTA v ON c.id_cliente = v.id_cliente
GROUP BY c.id_cliente, c.nombre, c.documento, c.email
ORDER BY monto_total_gastado DESC;


-- ---------------------------------------------
-- CU10: BUSCAR CLIENTE
-- ---------------------------------------------
-- Buscar cliente por nombre (búsqueda parcial)
SELECT * FROM CLIENTE
WHERE nombre LIKE '%Pérez%'
ORDER BY nombre;

-- Buscar cliente por documento (DNI)
SELECT * FROM CLIENTE
WHERE documento = '12345678';

-- Buscar cliente por ID
SELECT * FROM CLIENTE
WHERE id_cliente = 1;

-- Buscar cliente por email
SELECT * FROM CLIENTE
WHERE email LIKE '%@email.com'
ORDER BY nombre;


-- ---------------------------------------------
-- CU14: LISTAR LIBROS
-- ---------------------------------------------
-- Listar todos los libros
SELECT * FROM LIBRO
ORDER BY tipo_libro, titulo;

-- Listar libros con información del autor
SELECT
   l.id_libro,
   l.titulo,
   a.nombre as autor,
   l.editorial,
   l.año,
   l.precio,
   l.tipo_libro,
   l.stock
FROM LIBRO l
INNER JOIN AUTOR a ON l.id_autor = a.id_autor
ORDER BY l.tipo_libro, l.titulo;

-- Listar libros agrupados por tipo
```

```
SELECT
   tipo_libro,
   COUNT(*) as cantidad,
   AVG(precio) as precio_promedio,
   SUM(CASE WHEN tipo_libro = 'FISICO' THEN stock ELSE 0 END) as stock_total
FROM LIBRO
GROUP BY tipo_libro;


-- ------------------------------------------
-- CU15: BUSCAR LIBRO
-- ------------------------------------------
-- Buscar libro por título (búsqueda parcial)
SELECT
   l.*,
   a.nombre as nombre_autor
FROM LIBRO l
INNER JOIN AUTOR a ON l.id_autor = a.id_autor
WHERE l.titulo LIKE '%Soledad%'
ORDER BY l.titulo;

-- Buscar libro por ISBN
SELECT * FROM LIBRO
WHERE isbn = '978-0307474728';

-- Buscar libro por género
SELECT * FROM LIBRO
WHERE genero LIKE '%Realismo%'
ORDER BY titulo;

-- Buscar libro por tipo
SELECT * FROM LIBRO
WHERE tipo_libro = 'FISICO'
ORDER BY titulo;

-- Buscar libros con stock bajo (menos de 10 unidades)
SELECT
   l.titulo,
   l.stock,
   a.nombre as autor,
   l.precio
FROM LIBRO l
INNER JOIN AUTOR a ON l.id_autor = a.id_autor
WHERE l.tipo_libro = 'FISICO' AND l.stock < 10
ORDER BY l.stock ASC;


-- ------------------------------------------
-- CU19: LISTAR VENTAS
-- ------------------------------------------
```

```
-- Listar todas las ventas
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    v.monto,
    v.metodo_pago,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
ORDER BY v.fecha DESC;

-- Listar ventas con detalles completos
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    c.documento,
    l.titulo as libro,
    dv.cantidad,
    dv.precio_unitario,
    dv.subtotal,
    v.monto as total_venta,
    v.metodo_pago,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
INNER JOIN DETALLE_VENTA dv ON v.id_venta = dv.id_venta
INNER JOIN LIBRO l ON dv.id_libro = l.id_libro
ORDER BY v.fecha DESC, v.id_venta, dv.id_detalle;

-- Resumen de ventas por día
SELECT
    DATE(fecha) as dia,
    COUNT(*) as cantidad_ventas,
    SUM(monto) as total_vendido
FROM VENTA
WHERE estado = 'COMPLETADA'
GROUP BY DATE(fecha)
ORDER BY dia DESC;


-- ------------------------------------------
-- CU20: BUSCAR VENTA
-- ------------------------------------------
-- Buscar ventas por cliente (nombre parcial)
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
```

```
    v.monto,
    v.metodo_pago,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE c.nombre LIKE '%Pérez%'
ORDER BY v.fecha DESC;

-- Buscar venta por ID
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    c.documento,
    c.email,
    v.monto,
    v.metodo_pago,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.id_venta = 1;

-- Buscar ventas por rango de fechas
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    v.monto,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.fecha BETWEEN '2024-01-01' AND '2024-12-31'
ORDER BY v.fecha DESC;

-- Buscar ventas por monto (mayor a cierto valor)
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    v.monto,
    v.metodo_pago
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.monto > 50000.00
ORDER BY v.monto DESC;

-- Buscar ventas por método de pago
SELECT
    v.id_venta,
    v.fecha,
```

```
    c.nombre as cliente,
    v.monto,
    v.metodo_pago,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.metodo_pago = 'EFECTIVO'
ORDER BY v.fecha DESC;

-- Buscar ventas por estado
SELECT
    v.id_venta,
    v.fecha,
    c.nombre as cliente,
    v.monto,
    v.estado
FROM VENTA v
INNER JOIN CLIENTE c ON v.id_cliente = c.id_cliente
WHERE v.estado = 'COMPLETADA'
ORDER BY v.fecha DESC;


-- ---------------------------------------------
-- CU22: REPORTE DE LIBROS POR AUTOR
-- ---------------------------------------------
-- Consultar todos los libros de un autor específico con detalles
SELECT
    l.id_libro,
    l.titulo,
    l.editorial,
    l.año,
    l.precio,
    l.tipo_libro,
    l.stock,
    l.genero,
    -- Campos específicos según tipo
    CASE
        WHEN l.tipo_libro = 'FISICO' THEN CONCAT('Encuadernado: ', l.encuadernado, ', Edición: ',
l.num_edicion)
        WHEN l.tipo_libro = 'DIGITAL' THEN CONCAT('Formato: ', l.extension, ', Impresión: ',
IF(l.permisos_impresion, 'Sí', 'No'))
        WHEN l.tipo_libro = 'AUDIOLIBRO' THEN CONCAT('Duración: ', l.duracion, ' min, Plataforma: ',
l.plataforma, ', Narrador: ', l.narrador)
    END as detalles_especificos
FROM LIBRO l
WHERE l.id_autor = 1
ORDER BY l.tipo_libro, l.titulo;

-- Reporte resumen por autor
SELECT
```

```
    a.nombre as autor,
    COUNT(l.id_libro) as total_libros,
    SUM(CASE WHEN l.tipo_libro = 'FISICO' THEN 1 ELSE 0 END) as fisicos,
    SUM(CASE WHEN l.tipo_libro = 'DIGITAL' THEN 1 ELSE 0 END) as digitales,
    SUM(CASE WHEN l.tipo_libro = 'AUDIOLIBRO' THEN 1 ELSE 0 END) as audiolibros,
    AVG(l.precio) as precio_promedio
FROM AUTOR a
LEFT JOIN LIBRO l ON a.id_autor = l.id_autor
WHERE a.id_autor = 1
GROUP BY a.id_autor, a.nombre;


-- ---------------------------------------------
-- CONSULTAS ADICIONALES ÚTILES
-- ---------------------------------------------

-- Obtener historial completo de un cliente (CU09 con detalle)
SELECT
    c.nombre as cliente,
    c.documento,
    v.id_venta,
    v.fecha,
    l.titulo as libro,
    dv.cantidad,
    dv.precio_unitario,
    dv.subtotal,
    v.monto as total_venta,
    v.metodo_pago
FROM CLIENTE c
INNER JOIN VENTA v ON c.id_cliente = v.id_cliente
INNER JOIN DETALLE_VENTA dv ON v.id_venta = dv.id_venta
INNER JOIN LIBRO l ON dv.id_libro = l.id_libro
WHERE c.id_cliente = 1
ORDER BY v.fecha DESC, dv.id_detalle;

-- Calcular total de ventas (para reportes)
SELECT
    COUNT(*) as total_ventas,
    SUM(monto) as ganancia_total,
    AVG(monto) as venta_promedio
FROM VENTA
WHERE estado = 'COMPLETADA';

-- Libros más vendidos
SELECT
    l.id_libro,
    l.titulo,
    a.nombre as autor,
    l.tipo_libro,
    SUM(dv.cantidad) as unidades_vendidas,
```

```
    SUM(dv.subtotal) as ingresos_generados
FROM LIBRO l
INNER JOIN AUTOR a ON l.id_autor = a.id_autor
LEFT JOIN DETALLE_VENTA dv ON l.id_libro = dv.id_libro
GROUP BY l.id_libro, l.titulo, a.nombre, l.tipo_libro
HAVING unidades_vendidas > 0
ORDER BY unidades_vendidas DESC
LIMIT 10;

-- Clientes más activos (mayor cantidad de compras)
SELECT
    c.id_cliente,
    c.nombre,
    c.documento,
    COUNT(v.id_venta) as total_compras,
    SUM(v.monto) as total_gastado,
    AVG(v.monto) as promedio_por_compra
FROM CLIENTE c
LEFT JOIN VENTA v ON c.id_cliente = v.id_cliente
GROUP BY c.id_cliente, c.nombre, c.documento
HAVING total_compras > 0
ORDER BY total_gastado DESC
LIMIT 10;

-- Stock crítico (libros físicos con menos de 5 unidades)
SELECT
    l.id_libro,
    l.titulo,
    a.nombre as autor,
    l.stock,
    l.precio
FROM LIBRO l
INNER JOIN AUTOR a ON l.id_autor = a.id_autor
WHERE l.tipo_libro = 'FISICO'
  AND l.stock < 5
ORDER BY l.stock ASC;


-- ============================================
-- SECCIÓN 3: ACTUALIZACIÓN (UPDATE)
-- Operaciones para modificar datos existentes
-- ============================================

-- ---------------------------------------------
-- CU02: ACTUALIZAR AUTOR
-- ---------------------------------------------
-- Actualizar información completa de un autor
UPDATE AUTOR
SET nombre = 'Gabriel José de la Concordia García Márquez',
    nacionalidad = 'Colombia',
```

```
   biografia = 'Escritor, periodista y premio Nobel de Literatura 1982. Máximo exponente del realismo mágico.'
WHERE id_autor = 1;

-- Actualizar solo algunos campos
UPDATE AUTOR
SET biografia = 'Escritor y poeta chileno, considerado uno de los más importantes de la literatura en español.'
WHERE nombre = 'Roberto Bolaño';

-- Verificar actualización
SELECT * FROM AUTOR WHERE id_autor = 1;
SELECT * FROM AUTOR WHERE nombre = 'Roberto Bolaño';


-- ---------------------------------------------
-- CU07: ACTUALIZAR CLIENTE
-- ---------------------------------------------
-- Actualizar información completa del cliente
UPDATE CLIENTE
SET nombre = 'Juan Carlos Pérez',
   email = 'juancarlos.perez@email.com',
   telefono = '011-9999-8888'
WHERE id_cliente = 1;

-- Actualizar solo email
UPDATE CLIENTE
SET email = 'maria.gonzalez@newemail.com'
WHERE documento = '23456789';

-- Verificar actualización
SELECT * FROM CLIENTE WHERE id_cliente = 1;
SELECT * FROM CLIENTE WHERE documento = '23456789';


-- ---------------------------------------------
-- CU12: ACTUALIZAR LIBRO
-- ---------------------------------------------
-- Actualizar información general del libro
UPDATE LIBRO
SET precio = 16000.00,
   stock = 30
WHERE id_libro = 1;

-- Actualizar campos específicos de libro físico
UPDATE LIBRO
SET encuadernado = 'Tapa blanda',
   num_edicion = 2
WHERE id_libro = 1 AND tipo_libro = 'FISICO';
```

```
-- Actualizar precio de todos los libros de un autor (aumento del 10%)
UPDATE LIBRO
SET precio = precio * 1.10
WHERE id_autor = 1;

-- Verificar actualización
SELECT * FROM LIBRO WHERE id_libro = 1;
-- Verificar actualización de precio para libros del autor
SELECT id_libro, titulo, precio FROM LIBRO WHERE id_autor = 1;


-- ------------------------------------------
-- CU17: ACTUALIZAR VENTA
-- ------------------------------------------
-- Actualizar estado de la venta
UPDATE VENTA
SET estado = 'CANCELADA'
WHERE id_venta = 1;

-- Actualizar método de pago
UPDATE VENTA
SET metodo_pago = 'EFECTIVO'
WHERE id_venta = 1;

-- Verificar actualización simple
SELECT * FROM VENTA WHERE id_venta = 1;

-- Ejemplo COMPLETO: Actualizar detalles de venta (proceso complejo)
-- Supongamos que queremos cambiar la cantidad del libro 1 en la venta 1 de 2 a 3 unidades
START TRANSACTION;

-- 1. Obtener datos actuales del detalle
SELECT @cantidad_actual := cantidad, @id_libro_actual := id_libro
FROM DETALLE_VENTA
WHERE id_venta = 1 AND id_detalle = 1;

-- 2. Revertir stock del libro físico (sumar la cantidad que se había restado)
UPDATE LIBRO
SET stock = stock + @cantidad_actual
WHERE id_libro = @id_libro_actual AND tipo_libro = 'FISICO';

-- 3. Actualizar el detalle con la nueva cantidad y recalcular subtotal
UPDATE DETALLE_VENTA dv
INNER JOIN LIBRO l ON dv.id_libro = l.id_libro
SET dv.cantidad = 3,
   dv.subtotal = 3 * dv.precio_unitario
WHERE dv.id_detalle = 1 AND dv.id_venta = 1;

-- 4. Descontar el nuevo stock
UPDATE LIBRO
```

```
SET stock = stock - 3
WHERE id_libro = @id_libro_actual AND tipo_libro = 'FISICO';

-- 5. Recalcular el monto total de la venta
UPDATE VENTA v
SET v.monto = (
   SELECT SUM(dv.subtotal)
   FROM DETALLE_VENTA dv
   WHERE dv.id_venta = v.id_venta
)
WHERE v.id_venta = 1;

COMMIT;

-- Verificar actualización del detalle
SELECT * FROM DETALLE_VENTA WHERE id_venta = 1;
SELECT * FROM VENTA WHERE id_venta = 1;


-- ============================================
-- SECCIÓN 4: BORRADO (DELETE)
-- Operaciones para eliminar datos
-- ============================================

-- ---------------------------------------------
-- CU03: ELIMINAR AUTOR
-- ---------------------------------------------
-- IMPORTANTE: Primero verificar si tiene libros asociados
SELECT COUNT(*) as libros_asociados
FROM LIBRO
WHERE id_autor = 1;

-- Eliminar un autor (solo si NO tiene libros asociados)
DELETE FROM AUTOR
WHERE id_autor = 99;

-- Verificar eliminación
SELECT COUNT(*) as existe
FROM AUTOR
WHERE id_autor = 99;

-- Nota: Si el autor tiene libros, la operación fallará por la restricción FK


-- ---------------------------------------------
-- CU08: ELIMINAR CLIENTE
-- ---------------------------------------------
-- IMPORTANTE: Primero verificar si tiene ventas
SELECT COUNT(*) as ventas_registradas
FROM VENTA
```

```
WHERE id_cliente = 1;

-- Eliminar un cliente (solo si NO tiene ventas registradas)
DELETE FROM CLIENTE
WHERE id_cliente = 99;

-- Verificar eliminación
SELECT COUNT(*) as existe
FROM CLIENTE
WHERE id_cliente = 99;

-- Nota: Si el cliente tiene ventas, la operación fallará por la restricción FK


-- ------------------------------------------
-- CU13: ELIMINAR LIBRO
-- ------------------------------------------
-- IMPORTANTE: Primero verificar si está en ventas
SELECT COUNT(*) as veces_vendido
FROM DETALLE_VENTA
WHERE id_libro = 1;

-- Eliminar un libro (solo si NO está en ninguna venta)
DELETE FROM LIBRO
WHERE id_libro = 99;

-- Verificar eliminación
SELECT COUNT(*) as existe
FROM LIBRO
WHERE id_libro = 99;

-- Nota: Si el libro está en DETALLE_VENTA, la operación fallará por FK


-- ------------------------------------------
-- CU18: ELIMINAR VENTA
-- ------------------------------------------
-- Eliminar una venta con reversión de stock (CON TRANSACCIÓN)
START TRANSACTION;

-- 1. OPCIONAL: Revertir stock antes de eliminar (solo para físicos)
UPDATE LIBRO l
INNER JOIN DETALLE_VENTA dv ON l.id_libro = dv.id_libro
SET l.stock = l.stock + dv.cantidad
WHERE dv.id_venta = 99
  AND l.tipo_libro = 'FISICO';

-- 2. Eliminar la venta (los detalles se eliminan automáticamente por CASCADE)
DELETE FROM VENTA
WHERE id_venta = 99;
```

```
COMMIT;

-- Verificar eliminación
SELECT COUNT(*) as existe
FROM VENTA
WHERE id_venta = 99;

-- Verificar que los detalles también se eliminaron
SELECT COUNT(*) as existe
FROM DETALLE_VENTA
WHERE id_venta = 99;
```

Presentación de consultas SQL

Todas las consultas detalladas previamente pueden ser encontradas en el repositorio de GitHub creado para este proyecto:

https://github.com/DaianaArena/cozy-books

Definiciones de comunicación

El sistema de Cozy Books está diseñado como una **aplicación de escritorio monolítica** que se ejecuta íntegramente en un único proceso dentro del PC del usuario:

- Plataforma: Aplicación Java standalone
- Interfaz: Consola interactiva local (sin interfaz web)
- Despliegue: Archivo ejecutable único (App.jar)
- Ejecución: Todo el código Java se ejecuta en el mismo JRE (Java Runtime Environment)

En una arquitectura monolítica local:

- Todos los componentes (View, Controller, Repository, Model, Util) residen en el mismo espacio de memoria
- La comunicación entre capas se realiza mediante invocaciones de métodos directas (llamadas a funciones Java)
- No existen barreras de red, procesos o fronteras de serialización
- No hay necesidad de protocolos de comunicación como HTTP, REST, SOAP, gRPC, etc.

Toda la comunicación entre las capas del sistema se realiza mediante invocaciones síncronas de métodos:

- View → Controller: `shopController.registrarVenta(cliente, libros)`
- Controller → Repository: `ventaRepository.registrar(venta)`
- Repository → Database: `connection.prepareStatement(sql)`

No hay necesidad de:
- Serialización/deserialización de objetos
- Formatos de mensaje (JSON, XML, Protocol Buffers)
- Contratos de API (OpenAPI, WSDL)
- Gestión de timeout o reintentos
- Manejo de latencia de red

Desarrollo en Java

El sistema fue desarrollado en Java versión 20.0.2, utilizando el paradigma orientado a objetos y el patrón de arquitectura MVC (Modelo-Vista-Controlador), junto con una base de datos MySQL 8.0.34. El objetivo fue crear una aplicación de consola para la gestión integral de una librería, permitiendo administrar autores, clientes, libros, ventas y reportes.
Repositorio en GitHub: https://github.com/DaianaArena/cozybooks-project

Estructura del proyecto

La aplicación implementa una arquitectura MVC bien definida con las siguientes capas:

```
cozybooks-project/
├── src/
│   └── com/
│       └── cozybooks/
│           ├── model/          # Entidades del dominio
│           ├── repository/     # Acceso a datos
│           ├── controller/     # Lógica de negocio
│           ├── util/          # Utilidades
│           └── Main.java      # Punto de entrada
├── lib/                       # Librerías externas
├── tickets/                   # Tickets generados
├── sql/                       # Scripts de base de datos
└── documentacion/             # Documentación del proyecto
```

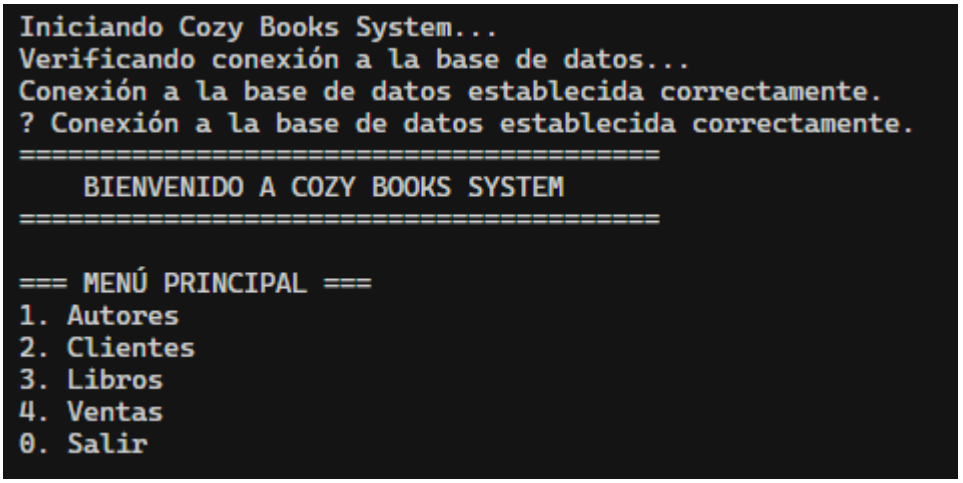
- **Modelo (Model)**: Clases que representan las entidades del dominio (Autor, Cliente, Libro, Venta, DetalleVenta)
- **Vista (View)**: Interfaz de usuario por consola con menús y submenús interactivos (MenuView)
- **Controlador (Controller)**: Lógica de negocio y coordinación entre vista y modelo (AutorController, ClienteController, LibroController, VentaController)
- **Repositorio (Repository)**: Capa de acceso a datos con operaciones CRUD (AutorRepository, ClienteRepository, LibroRepository, VentaRepository, DetalleVentaRepository)
- **Utilidades (Util)**: Servicios auxiliares para conexión a base de datos y manejo de archivos (DBConnection, ArchivoService)

Presentación del desarrollo

El desarrollo se centró en hacer el código modular y escalable. Se utilizaron clases y métodos bien definidos, aplicando el patrón de diseño Modelo-Vista-Controlador (MVC) y los principios de programación orientada a objetos (POO).
El menú principal y los submenús permiten al usuario interactuar con el sistema de manera sencilla y clara, utilizando estructuras condicionales y repetitivas para la navegación y la ejecución de operaciones.

Compilación y ejecución

El programa compila y se ejecuta correctamente en cualquier entorno compatible con Java 20.0.2. El punto de entrada es la clase Main.java, que inicia el menú principal a través de la vista MenuView. El sistema verifica automáticamente la conexión a la base de datos al iniciar y muestra mensajes informativos sobre el estado de la conexión:



Características implementadas

1. Correcta utilización de sintaxis, tipos de datos y estructuras de control

-Se emplean tipos de datos adecuados para cada entidad y atributo en las clases del modelo, incluyendo:

- int para identificadores únicos y cantidades enteras (ej. id en todas las entidades, stock en Libro)
- double para valores monetarios (ej. precio en Libro, monto en Venta)
- String para nombres, descripciones, direcciones, números de identificación y otros textos
- LocalDate para manejar fechas de manera eficiente (ej. fecha en Venta)

```java
public class Cliente {
    private int idCliente;
    private String nombre;
    private String documento; // DNI de 8 dígitos, único
    private String email;
    private String telefono;
    private LocalDateTime fechaRegistro;

    // Constructores
    public Cliente() {}

    public Cliente(String nombre, String documento) {
```

```java
public class Venta {
    private int idVenta;
    private LocalDateTime fecha;
    private BigDecimal monto;
    private MetodoPago metodoPago;
    private EstadoVenta estado;
    private int idCliente;
    private List<DetalleVenta> detalles;
```

2. Tratamiento y manejo de excepciones

-El sistema implementa un manejo robusto de excepciones en todos los niveles de la aplicación, validando la entrada del usuario y controlando errores de base de datos para evitar fallos del programa.

-El sistema garantiza que todas las entradas del usuario sean validadas antes del procesamiento, evitando fallos del programa y proporcionando mensajes de error claros y específicos para cada situación.

```java
private void procesarOpcion() {
    try {
        int opcion = Integer.parseInt(scanner.nextLine().trim());

        switch (opcion) {···

        if (ejecutando && opcion != 0) {
            System.out.println("\nPresione Enter para continuar...");
            scanner.nextLine();
        }

    } catch (NumberFormatException e) {
        System.out.println("Error: Debe ingresar un número válido.");
    } catch (Exception e) {
        System.out.println("Error inesperado: " + e.getMessage());
    }
}
```

Validación de campos obligatorios:

```
23      public void registrarLibro() {
24          try {
25              System.out.println(x:"\n═══ REGISTRAR LIBRO ═══");
26
27              System.out.print(s:"Título del libro: ");
28              String titulo = scanner.nextLine().trim();
29              if (titulo.isEmpty()) {      You, 2 hours ago • Add Cozy Books sa
30                  System.out.println(x:"Error: El título es obligatorio.");
31                  return;
32              }
```

Validación de rangos numéricos:

```
52              System.out.print(s:"Precio: ");
53              BigDecimal precio = new BigDecimal(scanner.nextLine().trim());
54              if (precio.compareTo(BigDecimal.ZERO) ≤ 0) {
55                  System.out.println(x:"Error: El precio debe ser mayor a 0.");
56                  return;
57              }
```

Validación de formato de fecha:

```
39              try {
40                  fechaNacimiento = LocalDate.parse(fechaStr, DateTimeFormatter.ISO_LOCAL
41              } catch (DateTimeParseException e) {
42                  System.out.println(x:"Error: Formato de fecha inválido. Use YYYY-MM-DD"
43                  return;
44              }
```

Control de transacciones:

```
14  public class VentaController {
266     public void eliminarVenta() {
287             if (confirmacion.equals(anObject:"s") || confirmacion.equals(anObject:"si")
288                 DBConnection.beginTransaction();      You, 2 hours ago • Add Cozy Books
289
290                 try {
291                     List<DetalleVenta> detalles = detalleVentaRepository.buscarPorVenta
292
293                     for (DetalleVenta detalle : detalles) {···
299
300                     ventaRepository.eliminar(id);
301
302                     DBConnection.commitTransaction();
303                     System.out.println(x:"Venta eliminada exitosamente.");
304
305                 } catch (Exception e) {
306                     DBConnection.rollbackTransaction();
307                     throw e;
308                 }
309             } else {
310                 System.out.println(x:"Operación cancelada.");
311             }
```

3. Encapsulamiento,polimorfismo y abstracción

-**Encapsulamiento**: Todos los atributos de las clases de dominio son privados y se accede a ellos mediante métodos públicos.

```java
public class Autor {
    private int idAutor;
    private String nombre;
    private LocalDate fechaNacimiento;
    private String nacionalidad;
    private String biografia;

    // Constructores
    public Autor() {}

    public Autor(String nombre, LocalDate fechaNacimiento) {…

    public Autor(String nombre, LocalDate fechaNacimiento, String na

    // Getters y Setters
    public int getIdAutor() {…

    public void setIdAutor(int idAutor) {…

    public String getNombre() {…
```

-**Polimorfismo**: Las listas y métodos que operan sobre diferentes tipos de libros pueden trabajar con libros físicos, digitales y audiolibros indistintamente a través del enum TipoLibro.

```java
public class LibroController {
    public void registrarLibro() {
        int stock = 0;

        switch (tipoOpcion) {
            case 1:
                tipoLibro = Libro.TipoLibro.FISICO;
                System.out.print(s:"Stock inicial: ");
                stock = Integer.parseInt(scanner.nextLine().trim());
                if (stock < 0) {
                    System.out.println(x:"Error: El stock no puede ser negativo.");
                    return;
                }
                break;
            case 2:
                tipoLibro = Libro.TipoLibro.DIGITAL;
                break;
            case 3:
                tipoLibro = Libro.TipoLibro.AUDIOLIBRO;
                break;
            default:
                System.out.println(x:"Error: Opción inválida.");
```

-**Abstracción**: Los repositorios y controladores abstraen la lógica de acceso a datos y negocio, ocultando la complejidad de las operaciones subyacentes.

```java
public class LibroRepository {

    public Libro registrar(Libro libro) throws SQLException {
        if (libro.getIsbn() ≠ null && !libro.getIsbn().isEmpty() && existeIsbn(libro.g
            throw new SQLException("Ya existe un libro con el ISBN: " + libro.getIsbn()
        }

        String sql = "INSERT INTO LIBRO (titulo, isbn, editorial, año, precio, genero,
                     "encuadernado, num_edicion, extension, permisos_impresion, duracion
                     "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        try (Connection conn = DBConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql, Statement.RETURN_GENER
```

4.Menu de selección

-El sistema cuenta con un menú principal y submenús para cada funcionalidad, permitiendo al usuario navegar y seleccionar operaciones.

```
========================================
    BIENVENIDO A COZY BOOKS SYSTEM
========================================

=== MENÚ PRINCIPAL ===
1. Autores
2. Clientes
3. Libros
4. Ventas
0. Salir

Seleccione una opción: 1

=== GESTIÓN DE AUTORES ===
1. Registrar Autor
2. Actualizar Autor
3. Eliminar Autor
4. Listar Autores
5. Buscar Autor
6. Reporte de Libros por Autor
0. Volver al menú principal
```

5.Estructuras condicionales y repetitivas

- Se utilizan estructuras de control como `if`, `switch` y ‘while’ para la lógica de los menús y operaciones

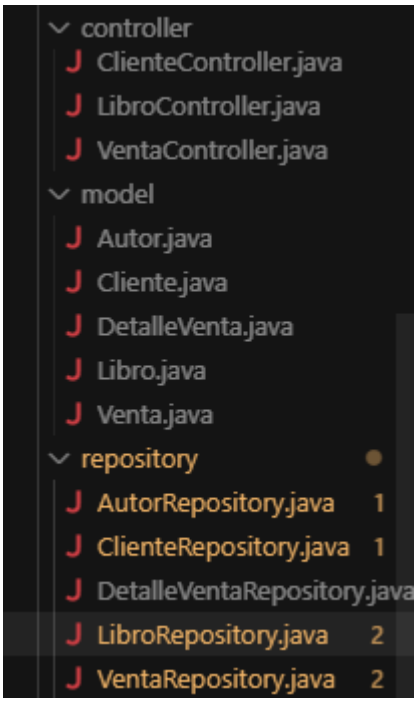
```java
88              if (ejecutando && opcion ≠ 0) {      You, 2 hours ago • Add Cozy Bo
89                  System.out.println(x:"\nPresione Enter para continuar...");
90                  scanner.nextLine();
91              }
```

```java
public class MenuView {
     */
    private void procesarOpcion() {
        try {
            int opcion = Integer.parseInt(scanner.nextLine().trim());

            switch (opcion) {
                case 1:
                    mostrarSubmenuAutores();
                    break;
                case 2:
                    mostrarSubmenuClientes();
                    break;
                case 3:
                    mostrarSubmenuLibros();
                    break;
                case 4:
                    mostrarSubmenuVentas();
                    break;
                case 0:
                    ejecutando = false;      You, 2 hours ago • Add Cozy Books sale
```

```java
    public void iniciar() {
        System.out.println("═══════════════════════════════════════════");
        System.out.println("    BIENVENIDO A COZY BOOKS SYSTEM");
        System.out.println("═══════════════════════════════════════════");

        while (ejecutando) {      You, 2 hours ago • Add Cozy Books sales man
            mostrarMenuPrincipal();
            procesarOpcion();
        }

        System.out.println("¡Gracias por usar Cozy Books System!");
    }
```

6.Declaración y creación de objetos en Java

- Se crean objetos de las clases de dominio (Autor, Cliente, Libro, Venta, DetalleVenta) y de los controladores y repositorios en los lugares apropiados.



7.Utilización de constructores para inicializar objetos

- Todas las clases de dominio cuentan con constructores que inicializan sus atributos.

```java
// Constructores
public DetalleVenta() {}

public DetalleVenta(int cantidad, BigDecimal precioUnitario, int idVenta, int idLibro) {
    this.cantidad = cantidad;
    this.precioUnitario = precioUnitario;
    this.idVenta = idVenta;
    this.idLibro = idLibro;
    this.subtotal = precioUnitario.multiply(BigDecimal.valueOf(cantidad));
}
```

8.Algoritmos de ordenación y búsqueda (opcional)

-Se implementan búsquedas por múltiples criterios en todas las entidades del sistema y algoritmos de ordenación para presentar los datos de manera organizada.Ejemplos de algoritmos de búsqueda implementados:

Búsqueda por nombre en autores:

```java
public Autor buscar(String criterio) throws SQLException {
    String sql = "SELECT * FROM AUTOR WHERE nombre LIKE ? ORDER BY nombre LIMIT 1";

    try (Connection conn = DBConnection.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(parameterIndex:1, "%" + criterio + "%");

        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return mapearResultSetAAutor(rs);
            }
        }
    }

    return null;
}
```

Búsqueda múltiple en clientes:

```java
public Cliente buscar(String criterio) throws SQLException {
    String sql = "SELECT * FROM CLIENTE WHERE nombre LIKE ? OR documento LIKE ?

    try (Connection conn = DBConnection.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {

        String criterioBusqueda = "%" + criterio + "%";
        stmt.setString(parameterIndex:1, criterioBusqueda);
        stmt.setString(parameterIndex:2, criterioBusqueda);

        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return mapearResultSetACliente(rs);
            }
        }
    }

    return null;
}
```

Búsqueda múltiple en libros:

```java
public List<Libro> buscar(String criterio) throws SQLException {
    String sql = "SELECT * FROM LIBRO WHERE titulo LIKE ? OR isbn LIKE ? OR genero LIKE ? ORDER BY titulo";
    List<Libro> libros = new ArrayList<>();

    try (Connection conn = DBConnection.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {

        String criterioBusqueda = "%" + criterio + "%";
        stmt.setString(parameterIndex:1, criterioBusqueda);
        stmt.setString(parameterIndex:2, criterioBusqueda);
        stmt.setString(parameterIndex:3, criterioBusqueda);

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                libros.add(mapearResultSetALibro(rs));
            }
        }
    }

    return libros;
}
```

Búsqueda múltiple en ventas:

```java
public List<Venta> buscar(String criterio) throws SQLException {
    String sql = "SELECT * FROM VENTA WHERE CAST(id_venta AS CHAR) LIKE ? OR CAST(id_cliente AS CHAR
                 "OR metodo_pago LIKE ? OR estado LIKE ? ORDER BY fecha DESC";
    List<Venta> ventas = new ArrayList<>();

    try (Connection conn = DBConnection.getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {

        String criterioBusqueda = "%" + criterio + "%";
        stmt.setString(parameterIndex:1, criterioBusqueda);
        stmt.setString(parameterIndex:2, criterioBusqueda);
        stmt.setString(parameterIndex:3, criterioBusqueda);
        stmt.setString(parameterIndex:4, criterioBusqueda);

        try (ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                ventas.add(mapearResultSetAVenta(rs));
            }
        }
    }

    return ventas;
}
```

Integración con la Base de Datos (JDBC)

La clase DBConnection se encarga de crear y gestionar la conexión con una base de datos MySQL local corriendo en el puerto 3306, utilizando el usuario root con contraseña 1234.

```java
public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/cozy_books?useS
    private static final String USER = "root";
    private static final String PASSWORD = "1234";

    private static Connection connection = null;

    /**
     * Obtiene una conexión a la base de datos
     * @return Connection objeto de conexión
     * @throws SQLException si hay error al conectar
     */
    public static Connection getConnection() throws SQLException {
        if (connection == null || connection.isClosed()) {
            try {
                // Cargar el driver de MySQL
                Class.forName(className:"com.mysql.cj.jdbc.Driver");
                connection = DriverManager.getConnection(URL, USER, PASSWORD);
                System.out.println(x:"Conexión a la base de datos establecida corr
            } catch (ClassNotFoundException e) {
                throw new SQLException("Driver de MySQL no encontrado: " + e.getMe
```