Universidad de Cartagena
Fundada en 1827

Asignatura: Bases de Datos 2
Estudiante: David Santiago Escamilla Castro
Código: 7502410012
Fecha: 30 de noviembre de 2025

1. Introducción

El presente documento describe el desarrollo del proyecto final de la asignatura Bases de Datos 2. El objetivo principal del proyecto fue diseñar y desarrollar una aplicación Java **standalone (de escritorio)** organizada estrictamente bajo la **Arquitectura Hexagonal**, conectada a una base de datos MySQL mediante adaptadores JDBC, con capacidad de ejecutar un flujo **CRUDL (Create, Read, Update, Delete, List)** sobre una entidad “Cliente”.

El proyecto integra conceptos fundamentales de bases de datos, construcción de consultas SQL, manejo de conexiones, uso de repositorios, adaptadores, capa de dominio y aplicación de principios de responsabilidad y desac acoplamiento propios de la arquitectura Hexagonal. Adicionalmente, se desarrolló una interfaz gráfica en **Java Swing**, que permite al usuario final interactuar con la base de datos de manera visual y amigable.

2. Funcionamiento del Programa

1. El usuario abre el .jar ejecutable.
2. La ventana Swing aparece.
3. Al hacer clic en **Listar Clientes**:
    - La GUI llama al caso de uso.
    - El caso de uso llama al repositorio JDBC.
    - JDBC consulta MySQL.
    - Los datos se muestran en la tabla.

Todo siguiendo el flujo hexagonal.

3. Estructura del Proyecto

La aplicación se construyó siguiendo rigurosamente las capas de la arquitectura hexagonal:

```
src/main/java/com/example/app
│
├── domain
│   └── Cliente.java
│   └── ClienteService.java
│
├── application
│   └── ListarClientesUseCase.java
│   └── CrearClienteUseCase.java
│   └── ...
│
├── infrastructure
│   └── jdbc
│       └── ClienteRepositoryJdbc.java
│
└── adapters
    └── in
        └── gui
            └── MainWindow.java
```

✔ Dominio (Reglas de negocio)

Contiene:

- La entidad Cliente
- Las reglas que definen el comportamiento del negocio
- Interfaces que describen la estructura de un repositorio

✔ Aplicación (Casos de uso)

Encapsula:

- Listar clientes
- Crear clientes

- Eliminar clientes
- Actualizar clientes

Cada caso de uso **solo llama interfaces**, no implementaciones, cumpliendo el principio de inversión de dependencias.

✔ Infraestructura (Adaptador hacia la BD)

Aquí se implementa:

- JDBC puro
- Carga del archivo config.properties
- Ejecución de consultas SQL reales

✔ Entrada del sistema (GUI Swing)

La ventana gráfica incluye:

- Botón Listar Clientes
- Tabla de visualización
- En versiones extendidas: Insertar / Actualizar / Eliminar

4. Conclusiones

- Se aplicó correctamente la arquitectura Hexagonal, logrando una aplicación desacoplada y modular.
- El uso de casos de uso permite que la lógica de negocio sea totalmente independiente de la interfaz gráfica y de la base de datos.
- La implementación de JDBC demuestra manejo directo de SQL, conexiones y prepared statements.
- La separación en capas facilita:
  - mantenimiento,
  - escalabilidad,
  - pruebas unitarias,
  - reutilización del código.
- La GUI Swing permite una interacción simple y efectiva para validar el CRUDL implementado.