UNIVERSIDAD DE CARTAGENA

Facultad de Ingeniería

Programa de Ingeniería de Software

Diseño de una Base de Datos para la Automatización de Procesos Comerciales en una Empresa de Productos Informáticos

II Semestre

Presentado por:

Bella Samai Botello Meza

Docente:

John Carlos Arrieta Arrieta

Fecha:

**Cartagena de Indias, 20 de mayo de 2025**

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

1. Introducción ................................................................................................ 3
2. Objetivos ...................................................................................................... 4  
     2.1. Objetivo General  
     2.2. Objetivos Específicos
3. Justificación ............................................................................................... 5
4. Clases Identificadas en el Ejercicio ............................................................ 6
5. Requerimientos Funcionales ........................................................................ 7  
     5.1. Gestión de Productos en Venta  
     5.2. Gestión de Proveedores  
     5.3. Gestión de Productos para Alquiler  
     5.4. Gestión de Productos de Alta Tecnología  
     5.5. Gestión del Soporte Técnico entre Fabricantes  
     5.6. Gestión de Servicios Técnicos para Impresoras
6. ¿Qué es un CRUD? ..................................................................................... 7
7. Lista de Archivos CRUD Implementados ....................... .......................... 8
8. Explicación de la Estructura de los CRUD ................................................. 9
9. Detalles de Cada Clase manejando los CRUD ........................................... 14
10. Explicación de los CRUD en el main ......................................................... 18
11. Conceptos de Java Aprendidos ....................................................................20
12. Enlace de GitHub .........................................................................................22
13. Bibliografía ................................................................................................... 56

Introducción

Actualmente, la tecnología se ha vuelto un elemento fundamental en todas las áreas, incluyendo el comercio. Una tienda informática que desea modernizar la gestión de sus procesos comerciales puede beneficiarse enormemente del diseño de una base de datos que facilite la administración del inventario, las relaciones con proveedores, el alquiler de equipos y los servicios técnicos ofrecidos a los clientes.

Esto no solo brindará a la tienda la oportunidad de automatizar y optimizar sus procesos internos, sino que también proporcionará una plataforma robusta para gestionar relaciones comerciales, mejorar la atención al cliente y aumentar los ingresos mediante la diversificación de productos y servicios.

**Objetivos**

**Objetivo General:**

Diseñar una base de datos que permita a una tienda informática automatizar y optimizar la gestión de su inventario, sus relaciones con proveedores, el alquiler de equipos y los servicios técnicos ofrecidos a los clientes.

**Objetivos Específicos:**

* Analizar los procesos comerciales de la tienda relacionados con ventas, alquiler y soporte técnico.
* Identificar las entidades y relaciones clave para el diseño de la base de datos.
* Diseñar un modelo relacional que integre y organice la información de forma eficiente.
* Mejorar el control interno y la atención al cliente mediante el uso de la base de datos.

**Justificación**

En la actualidad, una tienda informática necesita manejar muchas cosas a la vez, como el inventario, los proveedores, los alquileres y el servicio técnico. Si todo se hace manualmente, es fácil que haya errores o que se pierda información importante, lo que puede afectar el rendimiento del negocio.

Por eso, diseñar una base de datos que organice toda esta información ayudará a que los procesos sean más rápidos y claros. Además, facilitará el control del inventario, gestionará las relaciones comerciales, mejorará la atención al cliente y aumentará los ingresos a través de la diversificación de productos y servicios.

Esta base de datos no solo hará que la tienda funcione mejor, sino que también permitirá ofrecer un mejor servicio y aumentar las ventas, ayudando a que el negocio crezca y se adapte a lo que hoy exige el mercado.

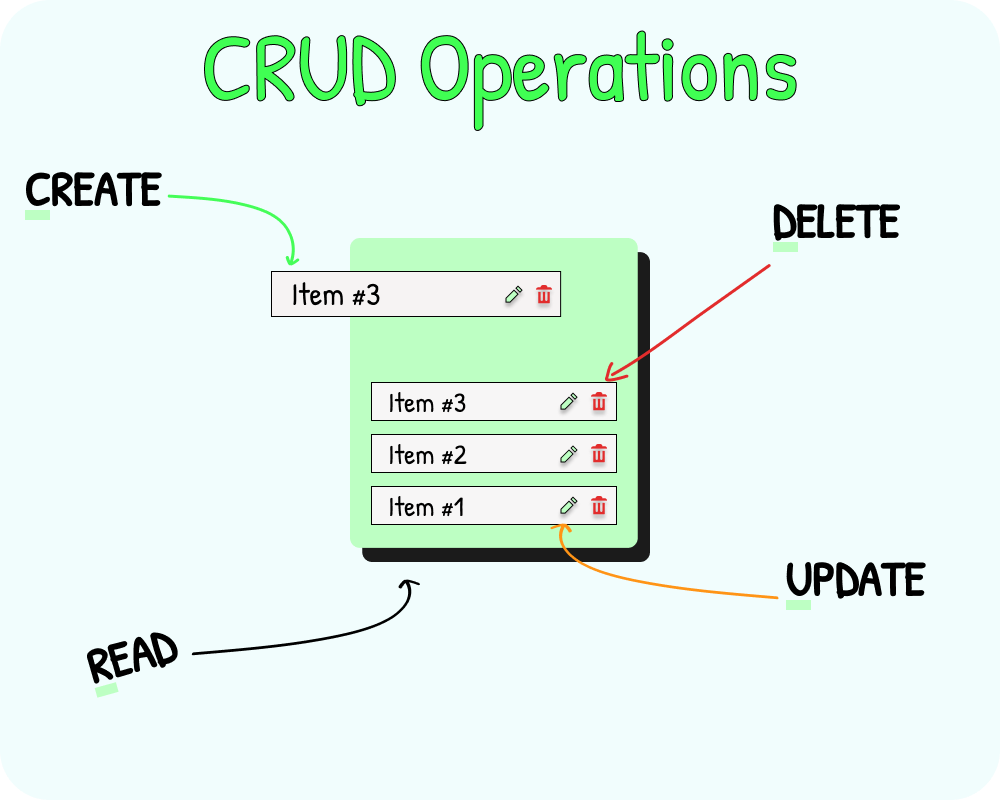
**Clases identificadas en el ejercicio**

1. Producto.
2. Monitor.
3. Impresora.
4. CPU.
5. DiscoDuro.
6. OtroProducto.
7. ProductoProveedor.
8. Proveedor.
9. ProductoAlquiler.
10. ProductoAltaTecnologia.
11. SoporteTecnicoFabricante.
12. EmpresaFabricante.
13. ServicioTecnico.
14. PrecioServicioTecnico.
15. Cliente.
16. ClienteImpresora.

¿Qué es un CRUD?

Un CRUD lo que hacen es llevar a cabo las funciones que un usuario necesita para crear y gestionar datos. Operaciones que se adaptan perfectamente a los requisitos tanto del usuario como del propio sistema:

* Create (Crear registros): se trata de introducir información
* Read bzw. Retrieve (Leer registros): consultar la información de un registro o de un conjunto de ellos.
* Update (Actualizar registros): modificar cualquiera de las columnas de la base de datos.
* Delete bzw. Destroy (Eliminar registros): borrar datos del sistema de almacenamiento.



**Lista de Archivos CRUD Implementados en la actividad**

Cada una de las siguientes clases cuenta con su respectiva clase CRUD, responsable de administrar sus objetos dentro de una lista interna:

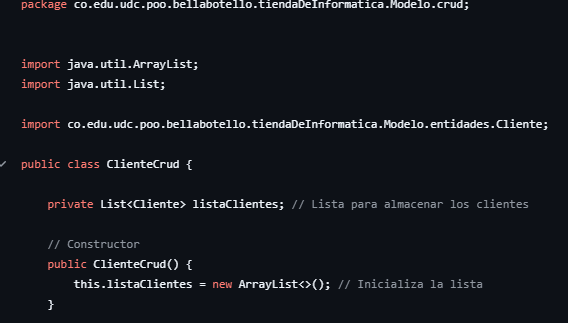
* ProductoCrud.java
* MonitorCrud.java
* ImpresoraCrud.java
* CPUCrud.java
* DiscoDuroCrud.java
* OtroProductoCrud.java
* ProductoProveedorCrud.java
* ProveedorCrud.java
* ProductoAlquilerCrud.java
* ProductoAltaTecnologiaCrud.java
* SoporteTecnicoFabricanteCrud.java
* EmpresaFabricanteCrud.java
* ServicioTecnicoCrud.java
* PrecioServicioTecnicoCrud.java
* ClienteCrud.java
* ClienteImpresoraCrud.java

**Explicación de la estructura de los CRUD**

A continuación, tomaremos como ejemplo clase cliente (ClienteCrud.java)

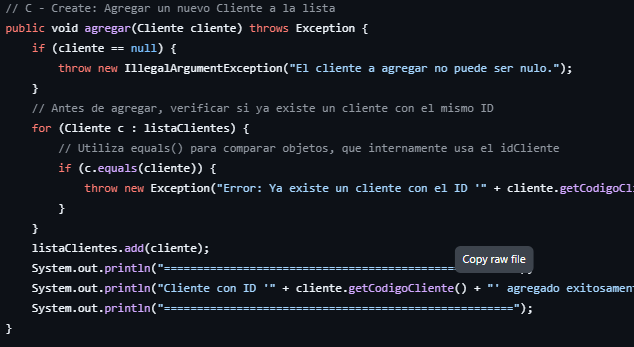
Objetivo:

gestionar en memoria los objetos de tipo Cliente, permitiendo simular el comportamiento de una base de datos utilizando una lista (ArrayList). A través de este CRUD se pueden realizar operaciones de creación, lectura, actualización y eliminación (CRUD) de clientes dentro del sistema de gestión de una tienda informática.

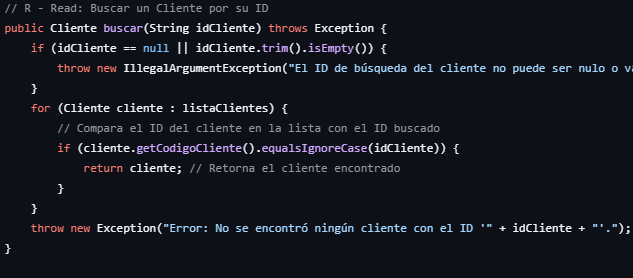


* Declara una variable listaClientes, que almacenará todos los clientes del sistema como una "base de datos en memoria".
* En el constructor, esta lista se inicializa como un nuevo ArrayList. Esto permite que cada vez que se cree un objeto ClienteCrud, tenga su propia lista lista para usar.

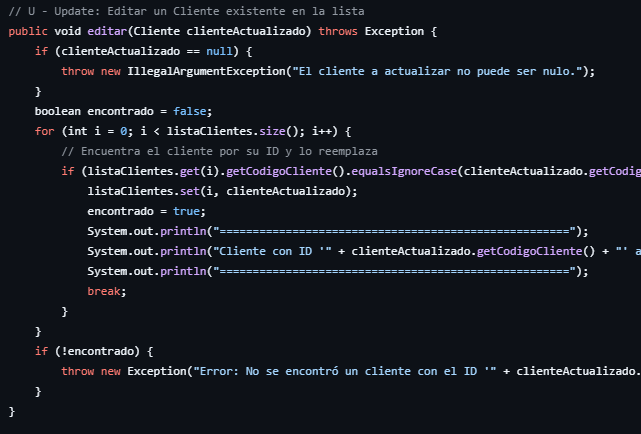
**Agregar**



* Se verifica que el objeto cliente no sea null (nulo).
* Recorre la lista para comprobar si existe un cliente igual
* Si existe, lanza una excepción personalizada.
* ("Error: Ya existe un cliente con el ID '" + cliente.getCodigoCliente() + "'.");
* Si no existe, lo agrega a la lista y muestra un mensaje confirmando la acción.

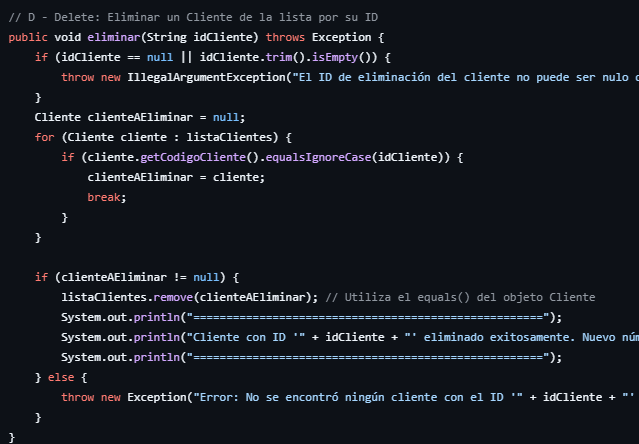
**Buscar**

* Se comprueba que el ID no sea nulo ni vacío.
* Se va recorrer la lista buscando un cliente con un ID que coincida sin importar mayúsculas o minúsculas (equalsIgnoreCase).
* Si lo encuentra, lo retorna.
* Si no se lanzara un comentario indicando que no se encontró el cliente.

**Editar  
  
**

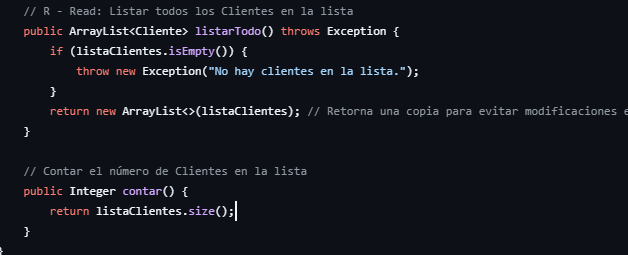
* Se verifica que el nuevo objeto cliente no sea nulo.
* Se recorre la lista para buscar un cliente con el mismo ID.
* Si encuentra el objeto en este caso cliente, reemplaza el objeto viejo por el nuevo con set(i, nuevoObjeto).
* Si no lo encuentra, lanza una excepción indicando que no se puede actualizar porque no existe.

**Eliminar**



* Valida que el ID sea válido.
* Busca el cliente por su ID.
* Si lo encuentra, lo elimina con remove.
* Si no, lanza una excepción.
* Además, muestra el nuevo número de clientes tras eliminar.

**Método listarTodo y contar**



* Listartodo ()

Devolverá una lista de todos los objetos guardados en clientes, si esta vacía lanzara un comentario ("No hay clientes en la lista.").

* contar() retorna la cantidad total de elementos en la lista.

Esta es la estructura base que siguen todas las clases CRUD implementadas en esta actividad. Cada una de ellas se encarga de gestionar una entidad específica del sistema tales como: Monitor, CPU, Proveedor, etc, aplicando el mismo patrón de operaciones fundamentales: agregar, buscar, editar, eliminar, listar y contar objetos. Todas utilizan internamente una lista de tipo ArrayList para almacenar los datos en memoria.s

**Detalles de cada clase manejando los CRUD**

* **CPUCrud.java**

Gestiona los objetos de tipo CPU, que heredan de Producto. Cada CPU se identifica por un código único. Este CRUD permite agregar, editar, buscar, eliminar y listar CPUs. Además, los CPU pueden estar asociados opcionalmente a un objeto de tipo ProductoAltaTecnologia.

* **ClienteCrud.java**

Administra la información de los clientes generales del sistema. Cada cliente se identifica mediante un codigoCliente. El CRUD permite realizar operaciones completas sobre la lista de clientes y es esencial para registrar quienes compran o solicitan servicios.

* **ClienteImpresoraCrud.java**

Especializa a Cliente para los casos donde el cliente tiene asociada una impresora específica. Se utiliza para gestionar clientes que reciben servicios técnicos sobre impresoras. Hereda los atributos de Cliente y añade una relación directa con un objeto Impresora.

* **DiscoDuroCrud.java**

CRUD para los objetos DiscoDuro, que también heredan de Producto. Los discos duros pueden estar marcados como de alta tecnología (ProductoAltaTecnologia) o disponibles para alquiler (ProductoAlquiler). Este CRUD permite gestionar completamente su existencia en el sistema.

* **EmpresaFabricanteCrud.java**

Gestiona la información de empresas que fabrican productos tecnológicos. Estas empresas están asociadas a ProductoAltaTecnologia y a SoporteTecnicoFabricante. Este CRUD permite registrar datos como nombre, dirección y número de empleados.

* **ImpresoraCrud.java**

Gestiona objetos Impresora, también subclase de Producto. Cada impresora puede tener asociado un objeto ServicioTecnico, lo cual permite llevar un control detallado de las reparaciones o mantenimientos realizados. El CRUD maneja su ciclo completo de vida.

* **MonitorCrud.java**

CRUD que administra los Monitor, que heredan de Producto. Cada monitor puede estar asociado a un ProductoAlquiler, permitiendo así su uso temporal bajo condiciones de arriendo. Se gestiona su resolución, modelo y disponibilidad.

* **OtroProductoCrud.java**

Maneja los productos que no encajan en categorías específicas como CPU, monitor, impresora o disco duro. Este CRUD es útil para registrar productos variados del inventario. Hereda de Producto.

* **PrecioServicioTecnicoCrud.java**

Permite registrar los precios que se aplican a servicios técnicos. Esta información se vincula desde la clase ServicioTecnico mediante la propiedad precioAplicado. Es útil para mantener el historial de tarifas aplicadas en distintas fechas.

* **ProductoAlquilerCrud.java**

CRUD especializado en manejar objetos relacionados con alquiler. Se vincula con Monitor y DiscoDuro para permitir su alquiler temporal. Registra atributos como el precio por hora y otras condiciones del arrendamiento.

* **ProductoAltaTecnologiaCrud.java**

Administra productos que poseen características avanzadas o tecnológicas. Estos productos se asocian a una EmpresaFabricante y pueden ser sujetos de SoporteTecnicoFabricante. Es común que se relacione con CPUs y discos duros.

* **ProductoCrud.java**

Clase base para todos los tipos de producto del sistema. Es una clase abstracta que no se instancia directamente. El CRUD asociado puede servir como plantilla base o herramienta común para pruebas y validaciones.

* **ProductoProveedorCrud.java**

CRUD que representa la relación entre productos y proveedores. Es una clase intermedia que almacena información como fechas de adquisición y cantidades suministradas. Facilita el control de abastecimiento y logísticas.

* **ProveedorCrud.java**

Gestiona los datos de proveedores que suministran productos a la tienda. Cada proveedor tiene un identificador (idProveedor) y un NIT. Este CRUD asegura la trazabilidad del origen de los productos.

* **ServicioTecnicoCrud.java**

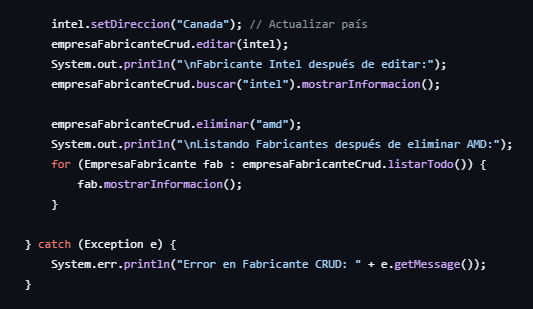
Administra los servicios técnicos realizados, especialmente los que involucran impresoras. Cada servicio técnico está relacionado con un ClienteImpresora, una Impresora y un PrecioServicioTecnico. Registra fecha, detalles del trabajo y monto facturado.

* **SoporteTecnicoFabricanteCrud.java**

CRUD que gestiona los servicios técnicos especializados ofrecidos por una EmpresaFabricante para un ProductoAltaTecnologia. Permite comparar precios entre fabricantes y definir condiciones específicas de soporte.

**Explicación de los CRUD en el main**

A continuación, se mostrará una prueba completa de operaciones CRUD para los objetos de la clase EmpresaFabricante



Primero, Dentro de un bloque try, se crean dos objetos EmpresaFabricante: uno para "Intel" y otro para "AMD", con sus respectivas direcciones, países y cantidades. Ambos objetos se agregan al sistema mediante el método agregar.

A continuación, se listan todos los fabricantes actuales usando , y se imprime su información con el método 

Después probamos el método editar realizando una actualización: se cambia la dirección de Intel a "Canadá" usando  se guarda el cambio con el método editar() y se muestra la información actualizada del fabricante usando buscar().

Luego, se elimina el fabricante "AMD" usando el método eliminar, y se vuelve a listar a los fabricantes restantes para comprobar que AMD ha sido removido.

Finalmente, si ocurre algún error durante estas operaciones, se captura con un bloque catch y se imprime el mensaje de error correspondiente.

A continuación, comentare los conceptos de Java que no conocía bien o que logré comprender mejor gracias al desarrollo de esta actividad. Los explico con mis propias palabras según cómo los usé:

1. **CRUD**   
   No conocía cómo se aplicaban de forma completa en Java. Esta actividad me permitió implementar y entender cómo agregar, listar, editar y eliminar objetos de una clase.
2. **Ciclo for**  
   Aunque ya lo había visto, con esta actividad reforcé su uso para recorrer listas de objetos.
3. **Métodos get y set**  
   Pude comprender mejor cómo se usan para acceder y modificar los atributos privados de una clase.
4. **Organización por clases**  
   En actividades anteriores, como opte por usar JSON, no tenía el código tan dividido. Aquí aprendí que es buena práctica tener todo separado: clases con sus atributos, métodos CRUD, y el main solo para las pruebas. Esto hace que el proyecto sea más entendible y fácil de modificar.
5. **Formato con printf**  
   En esta actividad intenté que la salida en consola fuera más ordenada usando System.out.printf("%-25s: %s\n", ...), lo cual me ayudó a visualizar los datos de forma clara.
6. **Refuerzo de la Actividad 1**  
   Gracias a este ejercicio pude repasar conceptos fundamentales que vi en la primera actividad, pero ahora aplicados de forma más práctica y con una mejor organización del código.

Todos los temas abordados tanto en la primera actividad como en esta segunda han sido fundamentales para avanzar en mi proceso de aprendizaje. La primera actividad me permitió conocer los conceptos básicos, mientras que esta segunda me ayudó a aplicarlos de una manera mucho más estructurada. Sin los conocimientos adquiridos previamente, no habría podido comprender ni ejecutar esta actividad.

Enlace de GitHub:

https://github.com/Bella200625/ActividadUnidad2Finished.git

Video:

<https://youtu.be/TAvTHZTqfqA?feature=shared>

Bibliografía

Código Mentor. (2021, 8 de septiembre). CRUD en Java con ArrayList [Video]. YouTube. <https://www.youtube.com/watch?v=WXtMS0DiwRY>

Programación ATS. (2020, 6 de abril). CRUD en Java con clases y objetos [Video]. YouTube. <https://www.youtube.com/watch?v=rFq8JgF_1Rk>

Fazt. (2020, 14 de enero). Java CRUD - Operaciones básicas con ArrayList [Video]. YouTube. <https://www.youtube.com/watch?v=vV2HykDgbbQ>

Baeldung. (s.f.). Introduction to CRUD in Java. Baeldung. https://www.baeldung.com/java-crud-operations

GeeksforGeeks. (2023, 15 de mayo). CRUD Operations in Java using Collections. GeeksforGeeks. https://www.geeksforgeeks.org/crud-operations-in-java-using-collections/

Oracle. (s.f.). Creating and Using CRUD Operations in Java Applications. Oracle Docs. <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>