Universidad Mariano Gálvez de Guatemala

Centro Regional Boca del Monte

Carrera:

Ingeniería en sistemas

Curso:

**Progra 2**

Catedrático:

LUIS FERNANDO ALVARADO CRUZ

Sección: A

Manual Técnico



Integrantes

Lester Navil Pérez marroquin

NO. carnet: 7690-20-14011

INDICE

[Introduction 3](#_Toc179000758)

[Especificaciones técnicas 4](#_Toc179000759)

[Instrucciones de instalación 4](#_Toc179000760)

[Instalación de JDK : 4](#_Toc179000761)

[Instalación de eclipse 5](#_Toc179000762)

[Código 6](#_Toc179000763)

# Introduction

En la presente documento se tratara de explicar detalladamente el funcionamiento interno del proyecto la cual tiene como objetico aplicar y consolidar los conocimiento sobre la base de datos a través de este trabajo la cual busca simular el proceso json almacenar archivos y el control de productos

# Especificaciones técnicas

# Instrucciones de instalación

# Instalación de JDK :

Instalar JDK (Java Development Kit):

NetBeans requiere tener el JDK (Java Development Kit) instalado.

Descarga la versión más reciente del JDK desde el sitio oficial de Oracle: https://www.oracle.com/java/technologies/javase-jdk11-downloads.html (o la versión que prefieras en nuestro caso se uso la versión 22 del JDK ).

Instala el JDK siguiendo las instrucciones del instalador.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

# Instalación de eclipse

Una vez que hayas descargado eclipse, abre el archivo de instalación que descargaste.

Sigue las instrucciones del asistente de instalación:

Acepta los términos de la licencia.

Selecciona la ubicación de instalación.

Verifica que eclipse detecte tu instalación de JDK. Si no lo hace automáticamente, selecciona la ruta de instalación del JDK manualmente.

Haz clic en Instalar y espera a que el proceso finalice.

(Acepto condiciones y Darle a Todo next )

Configurar la Conexión

# Código

**public class Agotado**

public class Agotado implements EstadoProducto {

public void manejar(Producto producto) {

System.out.println("El producto está agotado.");

}

}

**public class Almacenamiento**

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

public class Almacenamiento {

public static void guardarFactura(Factura factura, String archivo) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(archivo))) {

writer.write(factura.toString()); // Aquí se personalizar el formato de la factura

} catch (IOException e) {

e.printStackTrace();

}

}

}

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import java.io.\*;

import java.util.List;

**public class CatalogoProductos {**

public static List<Producto> cargarProductosDesdeJSON(String archivo) throws FileNotFoundException {

Gson gson = new Gson();

try (Reader reader = new FileReader(archivo)) {

return gson.fromJson(reader, new TypeToken<List<Producto>>() {}.getType());

} catch (IOException e) {

e.printStackTrace(); // Manejo de excepciones

return null;

}

}

}

import java.util.ArrayList;

import java.util.List;

p**ublic class Cliente {**

private String nombre;

private String direccion;

private List<OrdenDeCompra> ordenesDeCompra;

public Cliente(String nombre, String direccion) {

this.nombre = nombre;

this.direccion = direccion;

this.ordenesDeCompra = new ArrayList<>();

}

public void realizaPedido(List<Producto> productos) {

OrdenDeCompra nuevaOrden = new OrdenDeCompra();

nuevaOrden.setEstado("pendiente");

nuevaOrden.setFecha(new java.util.Date());

for (Producto producto : productos) {

nuevaOrden.agregarProducto(producto);

}

Factura factura = nuevaOrden.generarFactura();

nuevaOrden.setFactura(factura);

ordenesDeCompra.add(nuevaOrden);

System.out.println("Pedido realizado con éxito para el cliente: " + nombre);

}

public void cancelaPedido(OrdenDeCompra orden) {

if (orden != null && ordenesDeCompra.contains(orden)) {

orden.setEstado("cancelado");

System.out.println("Pedido cancelado para el cliente: " + nombre);

} else {

System.out.println("Orden no encontrada o ya cancelada.");

}

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getDireccion() {

return direccion;

}

public void setDireccion(String direccion) {

this.direccion = direccion;

}

public List<OrdenDeCompra> getOrdenesDeCompra() {

return ordenesDeCompra;

}

public void setOrdenesDeCompra(List<OrdenDeCompra> ordenesDeCompra) {

this.ordenesDeCompra = ordenesDeCompra;

}

}

public class Disponible implements EstadoProducto {

public void manejar(Producto producto) {

System.out.println("El producto está disponible.");

}

}

public class EnProduccion implements EstadoProducto {

public void manejar(Producto producto) {

System.out.println("El producto está en producción.");

}

}

public interface EstadoProducto {

void manejar(Producto producto);

}

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

**public class Factura** {

private int id;

private Date fecha;

private double montoTotal;

private List<Producto> productos;

// Constructor privado para forzar el uso del Builder

private Factura(Builder builder) {

this.id = builder.id;

this.fecha = new Date(); // Se genera la fecha automáticamente

this.productos = builder.productos;

this.montoTotal = calcularMontoTotal(); // Calcular el monto total

}

// Método para calcular el monto total de la factura

private double calcularMontoTotal() {

double total = 0.0;

for (Producto producto : productos) {

total += producto.getCantidad() \* producto.getPrecio();

}

return total;

}

// Getters

public int getId() {

return id;

}

public Date getFecha() {

return fecha;

}

public double getMontoTotal() {

return montoTotal;

}

public List<Producto> getProductos() {

return productos;

}

@Override

public String toString() {

return "Factura{" +

"id=" + id +

", fecha=" + fecha +

", montoTotal=" + montoTotal +

", productos=" + productos +

'}';

}

// Clase estática interna Builder

public static class Builder {

private int id;

private List<Producto> productos = new ArrayList<>();

// Constructor del Builder

public Builder(int id) {

this.id = id;

}

// Método para agregar productos

public Builder agregarProducto(Producto producto) {

productos.add(producto);

return this;

}

// Método para construir la factura

public Factura build() {

return new Factura(this);

}

}

public void setFecha(Date date) {

// TODO Auto-generated method stub

}

public void setMontoTotal(double montoTotal2) {

// TODO Auto-generated method stub

}

}

import java.util.Date;

**public class Importacion** {

private String paisDeOrigen;

private Date fechaDeLlegada;

private int cantidad;

public Importacion(String paisDeOrigen, Date fechaDeLlegada, int cantidad) {

this.paisDeOrigen = paisDeOrigen;

this.fechaDeLlegada = fechaDeLlegada;

this.cantidad = cantidad;

}

public void recibirProductos() {

if (fechaDeLlegada != null && fechaDeLlegada.before(new Date())) {

System.out.println("Productos recibidos desde " + paisDeOrigen + " el " + fechaDeLlegada);

System.out.println("Cantidad de productos recibidos: " + cantidad);

} else {

System.out.println("Los productos aún no han llegado o la fecha de llegada es inválida.");

}

}

public String getPaisDeOrigen() {

return paisDeOrigen;

}

public void setPaisDeOrigen(String paisDeOrigen) {

this.paisDeOrigen = paisDeOrigen;

}

public Date getFechaDeLlegada() {

return fechaDeLlegada;

}

public void setFechaDeLlegada(Date fechaDeLlegada) {

this.fechaDeLlegada = fechaDeLlegada;

}

public int getCantidad() {

return cantidad;

}

public void setCantidad(int cantidad) {

this.cantidad = cantidad;

}

@Override

public String toString() {

return "Importacion{" +

"paisDeOrigen='" + paisDeOrigen + '\'' +

", fechaDeLlegada=" + fechaDeLlegada +

", cantidad=" + cantidad +

'}';

}

}

import java.util.ArrayList;

import java.util.List;

**public class Inventario** {

private int cantidadDisponible;

private int cantidadEnProduccion;

// Instancia estática única del Inventario (Singleton)

private static Inventario instancia;

private List<Producto> productos = new ArrayList<>();

// Constructor privado para evitar la creación de nuevas instancias

private Inventario() {

this.cantidadDisponible = 0;

this.cantidadEnProduccion = 0;

}

// Método estático que devuelve la única instancia de Inventario

public static Inventario getInstancia() {

if (instancia == null) {

instancia = new Inventario(); // Crear la instancia si no existe

}

return instancia;

}

// Método para actualizar inventarios (producción y ventas)

public void actualizarInventarios(int productosProducidos, int productosVendidos) {

if (productosProducidos > 0) {

this.cantidadDisponible += productosProducidos;

this.cantidadEnProduccion -= productosProducidos;

}

if (productosVendidos > 0) {

if (productosVendidos <= this.cantidadDisponible) {

this.cantidadDisponible -= productosVendidos;

} else {

System.out.println("No hay suficiente stock para cubrir la venta.");

}

}

System.out.println("Inventario actualizado:");

System.out.println("Cantidad disponible: " + this.cantidadDisponible);

System.out.println("Cantidad en producción: " + this.cantidadEnProduccion);

}

// Getters y setters

public int getCantidadDisponible() {

return cantidadDisponible;

}

public void setCantidadDisponible(int cantidadDisponible) {

this.cantidadDisponible = cantidadDisponible;

}

public int getCantidadEnProduccion() {

return cantidadEnProduccion;

}

public void setCantidadEnProduccion(int cantidadEnProduccion) {

this.cantidadEnProduccion = cantidadEnProduccion;

}

@Override

public String toString() {

return "Inventario{" +

"cantidadDisponible=" + cantidadDisponible +

", cantidadEnProduccion=" + cantidadEnProduccion +

'}';

}

}

import com.google.gson.Gson;

import java.io.FileWriter;

import java.io.IOException;

**public class JsonFileWriter** {

public static void guardarDatosEnJson(String archivo, Object objeto) throws IOException {

Gson gson = new Gson();

FileWriter writer = new FileWriter(archivo);

gson.toJson(objeto, writer);

writer.close();

}

}

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

import java.io.InputStream;

import java.io.IOException;

**public class Main** {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

List<Cliente> clientes = new ArrayList<>();

List<Producto> productosDisponibles = new ArrayList<>();

// Cargar productos desde un archivo JSON

InputStream inputStream = Main.class.getResourceAsStream("/productos.json");

if (inputStream != null) {

try {

productosDisponibles = CatalogoProductos.cargarProductosDesdeJSON(inputStream);

if (productosDisponibles == null || productosDisponibles.isEmpty()) {

System.out.println("No se pudieron cargar productos, utilizando productos predeterminados.");

cargarProductosPredeterminados(productosDisponibles);

}

} catch (IOException e) {

System.out.println("Error al cargar el archivo de productos: " + e.getMessage());

cargarProductosPredeterminados(productosDisponibles); // Cargar productos predeterminados en caso de error

}

} else {

System.out.println("El archivo no se encontró, utilizando productos predeterminados.");

cargarProductosPredeterminados(productosDisponibles);

}

while (true) {

System.out.println("\n--- Menú Principal ---");

System.out.println("1. Agregar Cliente");

System.out.println("2. Realizar Pedido");

System.out.println("3. Cancelar Pedido");

System.out.println("4. Listar Clientes y Pedidos");

System.out.println("5. Listar Productos Disponibles");

System.out.println("6. Salir");

System.out.print("Seleccione una opción: ");

int opcion = scanner.nextInt();

scanner.nextLine(); // Limpiar

switch (opcion) {

case 1:

// Agregar Cliente

System.out.print("Ingrese el nombre del cliente: ");

String nombreCliente = scanner.nextLine();

System.out.print("Ingrese la dirección del cliente: ");

String direccionCliente = scanner.nextLine();

clientes.add(new Cliente(nombreCliente, direccionCliente));

System.out.println("Cliente agregado con éxito.");

break;

case 2:

// Realizar Pedido

if (clientes.isEmpty()) {

System.out.println("No hay clientes registrados. Por favor, agregue un cliente primero.");

break;

}

System.out.println("Seleccione el cliente (Ingrese el número correspondiente):");

for (int i = 0; i < clientes.size(); i++) {

System.out.println((i + 1) + ". " + clientes.get(i).getNombre());

}

int clienteIndex = scanner.nextInt() - 1;

scanner.nextLine(); // Limpiar

if (clienteIndex < 0 || clienteIndex >= clientes.size()) {

System.out.println("Cliente no válido.");

break;

}

Cliente cliente = clientes.get(clienteIndex);

List<Producto> productosParaPedido = new ArrayList<>();

System.out.println("Seleccione los productos para el pedido (Ingrese 0 para finalizar):");

for (int i = 0; i < productosDisponibles.size(); i++) {

Producto producto = productosDisponibles.get(i);

System.out.println((i + 1) + ". " + producto.getNombre() + " - $" + producto.getPrecio() + " (Stock: " + producto.getCantidad() + ")");

}

while (true) {

int productoIndex = scanner.nextInt() - 1;

scanner.nextLine();

if (productoIndex == -1) break; // Finalizar la selección

if (productoIndex < 0 || productoIndex >= productosDisponibles.size()) {

System.out.println("Producto no válido.");

continue;

}

Producto productoSeleccionado = productosDisponibles.get(productoIndex);

System.out.print("Ingrese la cantidad para " + productoSeleccionado.getNombre() + ": ");

int cantidad = scanner.nextInt();

scanner.nextLine();

if (cantidad <= 0 || cantidad > productoSeleccionado.getCantidad()) {

System.out.println("Cantidad no válida. Máximo disponible: " + productoSeleccionado.getCantidad());

continue;

}

productosParaPedido.add(new Producto(productoSeleccionado.getNombre(), productoSeleccionado.getDescripcion(),

productoSeleccionado.getTipo(), cantidad, productoSeleccionado.getPrecio()));

}

// patrón Builder para generar la factura

Factura factura = new Factura.Builder(cliente.getNombre())

.agregarProductos(productosParaPedido)

.build();

cliente.realizaPedido(factura);

System.out.println("Pedido realizado con éxito.");

break;

case 3:

// Cancelar Pedido

if (clientes.isEmpty()) {

System.out.println("No hay clientes registrados.");

break;

}

System.out.println("Seleccione el cliente (Ingrese el número correspondiente):");

for (int i = 0; i < clientes.size(); i++) {

System.out.println((i + 1) + ". " + clientes.get(i).getNombre());

}

int clienteIndexCancelar = scanner.nextInt() - 1;

scanner.nextLine();

if (clienteIndexCancelar < 0 || clienteIndexCancelar >= clientes.size()) {

System.out.println("Cliente no válido.");

break;

}

Cliente clienteCancelar = clientes.get(clienteIndexCancelar);

List<OrdenDeCompra> ordenes = clienteCancelar.getOrdenesDeCompra();

if (ordenes.isEmpty()) {

System.out.println("El cliente no tiene pedidos.");

break;

}

System.out.println("Seleccione el pedido a cancelar (Ingrese el número correspondiente):");

for (int i = 0; i < ordenes.size(); i++) {

System.out.println((i + 1) + ". Pedido ID: " + ordenes.get(i).getId() + ", Estado: " + ordenes.get(i).getEstado());

}

int ordenIndex = scanner.nextInt() - 1;

scanner.nextLine();

if (ordenIndex < 0 || ordenIndex >= ordenes.size()) {

System.out.println("Pedido no válido.");

break;

}

clienteCancelar.cancelaPedido(ordenes.get(ordenIndex));

break;

case 4:

// Listar Clientes y Pedidos

if (clientes.isEmpty()) {

System.out.println("No hay clientes registrados.");

} else {

for (Cliente c : clientes) {

System.out.println("Cliente: " + c.getNombre() + ", Dirección: " + c.getDireccion());

List<OrdenDeCompra> pedidos = c.getOrdenesDeCompra();

if (pedidos.isEmpty()) {

System.out.println(" No tiene pedidos.");

} else {

for (OrdenDeCompra orden : pedidos) {

System.out.println(" Pedido ID: " + orden.getId() + ", Estado: " + orden.getEstado());

}

}

}

}

break;

case 5:

// Listar Productos Disponibles

System.out.println("Productos disponibles:");

for (Producto producto : productosDisponibles) {

System.out.println(producto);

}

break;

case 6:

// Salir del programa

System.out.println("Saliendo del programa...");

scanner.close();

return;

default:

System.out.println("Opción no válida. Intente de nuevo.");

break;

}

}

}

private static void cargarProductosPredeterminados(List<Producto> productosDisponibles) {

// Cargar productos predeterminados si no se pueden cargar desde el JSON

productosDisponibles.add(new Producto("globos", "grande", "rojo", 100, 5.00));

productosDisponibles.add(new Producto("Sombrilla", "grande", "negra", 200, 50.00));

productosDisponibles.add(new Producto("taza", "taza de 1.5 ml", "térmica", 1500, 25.00));

productosDisponibles.add(new Producto("boli", "Bolígrafos personalizados", "de colores", 1500, 25.00));

productosDisponibles.add(new Producto("Blocs de notas", "Con diseños personalizados y hojas de diferentes colores", "hojas", 1500, 25.00));

productosDisponibles.add(new Producto("Llaveros", "Llaveros personalizados", "colgante", 1500, 25.00));

productosDisponibles.add(new Producto("usb", "USB personalizado", "negro", 1050, 50.00));

productosDisponibles.add(new Producto("power bank", "personalizado", "negro", 1500, 250.00));

productosDisponibles.add(new Producto("gorras", "personalizadas", "de colores", 1500, 200.00));

productosDisponibles.add(new Producto("camisas", "personalizadas", "fosforescentes", 1050, 250.00));

}

}

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

**public class OrdenDeCompra** {

private static int idCounter = 1;

private int id;

private Date fecha;

private String estado;

private List<Producto> productos;

private Factura factura;

public OrdenDeCompra() {

this.id = ++idCounter;

this.fecha = new Date();

this.productos = new ArrayList<>();

}

public void agregarProducto(Producto producto) {

if (producto != null) {

productos.add(producto);

System.out.println("Producto agregado: " + producto.getNombre());

} else {

System.out.println("El producto no puede ser nulo.");

}

}

public void quitarProducto(Producto producto) {

if (productos.contains(producto)) {

productos.remove(producto);

System.out.println("Producto quitado: " + producto.getNombre());

} else {

System.out.println("El producto no se encuentra en la orden.");

}

}

public Factura generarFactura() {

Factura nuevaFactura = nuevaFactura();

nuevaFactura.setFecha(new Date());

double montoTotal = 0;

for (Producto producto : productos) {

montoTotal += producto.getCantidad() \* producto.getPrecio();

}

nuevaFactura.setMontoTotal(montoTotal);

System.out.println("Factura generada con un monto total de: " + montoTotal);

return nuevaFactura;

}

private Factura nuevaFactura() {

// TODO Auto-generated method stub

return null;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public Date getFecha() {

return fecha;

}

public void setFecha(Date fecha) {

this.fecha = fecha;

}

public String getEstado() {

return estado;

}

public void setEstado(String estado) {

this.estado = estado;

}

public List<Producto> getProductos() {

return productos;

}

public void setProductos(List<Producto> productos) {

this.productos = productos;

}

public Factura getFactura() {

return factura;

}

public void setFactura(Factura factura) {

this.factura = factura;

}

}

import java.util.Date;

**public class Produccion** {

private String estado;

private Date fechaInicio;

private Date fechaFin;

public Produccion() {

this.estado = "No Iniciado";

this.fechaInicio = null;

this.fechaFin = null;

}

public void iniciarProduccion() {

if (!"En Proceso".equals(estado)) {

this.estado = "En Proceso";

this.fechaInicio = new Date();

System.out.println("Producción iniciada el " + fechaInicio);

} else {

System.out.println("La producción ya está en proceso.");

}

}

public void finalizarProduccion() {

if ("En Proceso".equals(estado)) {

this.estado = "Finalizado";

this.fechaFin = new Date();

System.out.println("Producción finalizada el " + fechaFin);

} else {

System.out.println("La producción no está en proceso o ya ha finalizado.");

}

}

public String getEstado() {

return estado;

}

public void setEstado(String estado) {

this.estado = estado;

}

public Date getFechaInicio() {

return fechaInicio;

}

public void setFechaInicio(Date fechaInicio) {

this.fechaInicio = fechaInicio;

}

public Date getFechaFin() {

return fechaFin;

}

public void setFechaFin(Date fechaFin) {

this.fechaFin = fechaFin;

}

}

public class Producto {

private String nombre;

private String descripcion;

private String tipo;

private int cantidad;

private double precio;

private EstadoProducto estado;

public Producto(String nombre, String descripcion, String tipo, int cantidad, double precio) {

this.nombre = nombre;

this.descripcion = descripcion;

this.tipo = tipo;

this.cantidad = cantidad;

this.precio = precio;

}

public Producto(EstadoProducto estadoInicial) {

this.estado = estadoInicial;

}

public void setEstado(EstadoProducto nuevoEstado) {

this.estado = nuevoEstado;

}

public void manejarEstado() {

estado.manejar(this);

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getDescripcion() {

return descripcion;

}

public void setDescripcion(String descripcion) {

this.descripcion = descripcion;

}

public String getTipo() {

return tipo;

}

public void setTipo(String tipo) {

this.tipo = tipo;

}

public int getCantidad() {

return cantidad;

}

public void setCantidad(int cantidad) {

this.cantidad = cantidad;

}

public double getPrecio() {

return precio;

}

public void setPrecio(double precio) {

this.precio = precio;

}

@Override

public String toString() {

return "Producto{" +

"nombre='" + nombre + '\'' +

", descripcion='" + descripcion + '\'' +

", tipo='" + tipo + '\'' +

", cantidad=" + cantidad +

", precio=" + precio +

'}';

}

}