Documentación KinCaps (Sprint #2)

Descripción:

En el segundo sprint de esta aplicación web, se ha implementado el módulo de administración. Este mismo incluye la funcionalidad completa de CRUD para administrar las entidades clave del sistema, como Usuarios, Proveedores, Gorras, Facturas, Empleados, Detalle Factura, Detalle Carrito, Clientes y Carrito. Para realizar esto se hizo el uso de Servlets para monitorear las solicitudes y respuestas, también se hizo el uso de DAOs para gestionar las interacciones con la base de datos, volviéndolo un sistema eficiente y ordenado para el área de administración.

DAOs:

UsuarioDAO:

Autenticación de usuarios mediante procedimiento almacenado sp_Login.

Método destacado:

- Verificar Credenciales: Devuelve una instancia de Cliente o Empleado según el tipo de usuario autenticado.
- Uso: Accede directamente a procedimientos almacenados y aplica casting dinámico.

ClienteDAO:

- Validación y persistencia de clientes.
- email Existe: Verifica si un correo ya existe.
- crear Cliente: Crea un nuevo cliente.
- Uso: Incluye control de errores personalizado.

EmpleadoDAO:

- Manejo completo de empleados.
- Métodos: guardar, buscarPorId, listarTodos, actualizar, eliminar.

GorrasDAO:

- Gestión de productos tipo “gorra”.
- Métodos: guardar, buscarPorId, listarTodas, actualizar, eliminar.

FacturaDAO:

- Administración de facturas emitidas.
- Métodos: guardar, buscarPorId, listarTodas, actualizar, eliminar.

DetalleFacturaDAO:

- Gestión de detalles asociados a una factura.
- Métodos: guardar, buscarPorId, listarTodos, actualizar, eliminar.

CarritoDAO:

- Operaciones sobre carritos de compra.
- Métodos: guardar, buscarPorId, listarTodos, actualizar, eliminar.

DetalleCarritoDAO:

- Gestión de los ítems en un carrito de compras.
- Métodos: guardar, buscarPorId, listarTodos, actualizar, eliminar.

ProveedorDAO:

- Gestión de proveedores del sistema.
- Métodos: guardar, buscarPorId, listarTodos, actualizar, eliminar.

Comportamiento

- Casi todos los DAO exceptuando UsuarioDAO y ClienteDAO comparten un patrón estructural:
- Obtención de EntityManager mediante JPAUtil.
- Transacciones con EntityTransaction.
- Uso de try-finally para asegurar el cierre del EntityManager.
- Rollback automático en caso de errores al persistir o actualizar datos.

Dependencias

- modelo: Clases de entidades JPA (Cliente, Empleado, Carrito, etc.).
- util.JPAUtil: Clase utilitaria para obtener instancias de EntityManager.

Modelo:

Usuario

- Es la superclase base para los tipos de usuarios en el sistema.
- Campos comunes: nombre, apellido, email, teléfono, dirección y contraseña en hash.
- Anotada con @MappedSuperclass para que sus atributos sean heredados por las subclases sin necesidad de crear tabla propia.

Cliente

- Representa a los clientes de la aplicación.
- Tiene un identificador único.
- Hereda los campos básicos del usuario.

Empleado

- Representa a los empleados del sistema.
- Campos adicionales: idEmpleado, puesto y fechaContratacion.
- Hereda campos comunes de Usuario.

Proveedor

- Entidad independiente que representa a los proveedores.
- Campos principales: idProveedor, nombre, contacto y teléfono.

Gorras

- Son los productos en el inventario.
- Campos: idGorra, modelo, marca, color, precio, stock.
- Relación ManyToOne con Proveedor.

Factura

- Es una factura generada.
- Campos: idFactura, cliente, empleado, fechaEmision, total, método de pago.
- Relaciones ManyToOne con Cliente y Empleado.

Detalle Factura

- Detalle de los productos vendidos en una factura.
- Campos: idDetalleFactura, factura, gorra, cantidad y precioVenta.
- Relaciones ManyToOne con Factura y Gorras.

Carrito

- Es el carrito de compras de un cliente.
- Campos: idCarrito, cliente y fechaCreacion.
- Relación ManyToOne con Cliente.

Detalle Carrito

- Detalle de los productos añadidos al carrito.
- Campos: idDetalleCarrito, carrito, gorra, cantidad y precioUnitario.
- Relaciones ManyToOne con Carrito y Gorras.

Aspectos Técnicos Comunes:

- Uso de anotaciones JPA para definir mapeo objeto-relacional:
- @Entity, @Table: para marcar las clases y tablas.
- @Id, @GeneratedValue: para claves primarias auto incrementables.
- @Column: para mapear atributos a columnas con restricciones.
- @ManyToOne y @JoinColumn: para establecer relaciones entre entidades.
- Uso de tipos Java modernos para fechas.
- Uso de BigDecimal para valores monetarios, asegurando precisión.
- Las clases cuentan con constructores por defecto y sobrecargados para facilitar creación de instancias.
- Métodos: getters y setters para acceso a atributos.
- Herencia para compartir atributos comunes en Usuario, Cliente y Empleado.

Servlet:

DetalleCarritoCRUDServlet:

- Maneja el CRUD de los detalles de los productos que están en el carrito.
- Campos manejados: idDetalleCarrito, carrito, gorra, cantidad, precioUnitario.

Funcionalidad:

- Mostrar formulario para agregar o editar detalle de carrito.
- Procesa operaciones de agregar, editar, actualizar, eliminar y listar detalles.

Relación con otras entidades:

- Usa CarritoDAO para obtener carritos.
- Usa GorrasDAO para obtener productos.
- URL base: /mantenimiento/detallecarrito

ClienteCRUDServlet:

- Gestiona el CRUD de clientes.
- Campos manejados: idCliente, nombre, apellido, email, teléfono, dirección, contrasenaHash.

Funcionalidad:

- Mostrar el formulario para agregar o editar cliente.
- Procesa operaciones para crear, actualizar, eliminar y listar clientes.
- Validación básica de campos requerida en formularios.
- URL base: /mantenimiento/clientes

CatalogoServlet:

- Muestra el catálogo de productos disponibles.
- Obtiene la lista de gorras desde GorrasDAO y la muestra en la página.
- Funcionalidad simple: solo listar productos para vista pública o de catálogo.
- URL base: /gorras/catalogo

CarritoCRUDServlet:

- Maneja el CRUD para carritos de compra.
- Campos manejados: idCarrito, cliente, fechaCreacion.

Funcionalidad:

- Mostrar formulario para crear o editar carrito.
- Procesa la creación, actualización, eliminación y listado.
- Interacción con ClienteDAO para obtener datos de clientes.
- URL base: /mantenimiento/carrito

Aspectos Técnicos Comunes:

- Uso de servlets con anotación @WebServlet para definir rutas y nombres.
- Uso de DAOs específicos para acceder y modificar la base de datos.

En los formularios HTML generados dinámicamente:

- Uso de Bootstrap para diseño.
- Validación HTML con atributos.
- Manejo de codificación UTF-8 para evitar problemas con caracteres especiales.
- Uso de RequestDispatcher para enviar datos a JSP para mostrar listados.
- Control de rutas mediante método privado para diferenciar acciones según path info.
- Manejo de excepciones y logging para control de errores.
- Operaciones CRUD completas en todos los servlets.
- Redirecciones después de operaciones para evitar reenvío de formularios.
- Inclusión de un footer común con derechos reservados y enlaces de política y términos.