🍔 FIDEBURGUESAS-POS - GUÍA COMPLETA DE FUNCIONAMIENTO Y PRUEBAS

📋 ¿QUÉ ES FIDEBURGUESAS-POS?

FideBurguesas-POS es un sistema completo de Point of Sale (Punto de Venta) para restaurantes que demuestra conceptos avanzados de **Programación Cliente-Servidor Concurrente**. El sistema permite gestión completa de inventario, ventas, usuarios y órdenes con procesamiento multihilo y serialización automática.

🎯 FUNCIONALIDADES PRINCIPALES

✅ **Sistema de autenticación** con roles (Admin, Cajero, Cocina)
✅ **Gestión completa de inventario** (productos, categorías, combos)
✅ **Punto de venta funcional** con cálculo automático de totales
✅ **Sistema de órdenes** con estados y procesamiento asíncrono
✅ **Backup automático** con serialización cada 24 horas
✅ **Procesamiento multihilo** con 16 hilos concurrentes
✅ **Manejo de excepciones** personalizadas (stock, validaciones)
✅ **Cache inteligente** para optimización de rendimiento
✅ **Interfaz gráfica moderna** con múltiples ventanas especializadas

🛠️ PREPARACIÓN DEL ENTORNO

REQUISITOS PREVIOS

- **Apache NetBeans IDE 25** (o superior)
- **Java 24** (JDK configurado)
- **Maven** (incluido en NetBeans)
- **Sistema operativo:** Windows, macOS o Linux

CONFIGURACIÓN INICIAL

1. **Abrir NetBeans IDE**
2. **Importar proyecto:** File → Open Project → Seleccionar carpeta `FideBurguesas-POS`
3. **Verificar configuración:** Click derecho en proyecto → Properties → Sources → Source/Binary Format = **JDK 24**
4. **Compilar proyecto:** Build → Clean and Build Project

5. **Verificar dependencias:** Maven descargará automáticamente las librerías necesarias

🚀 CÓMO EJECUTAR EL PROYECTO

PASO 1: INICIAR EL SISTEMA

1. **Ejecutar aplicación:** Click derecho en proyecto → Run

Verificar logs: Revisar consola para confirmaciones:
✅ MultithreadManager iniciado con 16 hilos
✅ Backup automático programado cada 24 horas

2. ✅ Base de datos inicializada correctamente

3. **Pantalla de login:** Se abre automáticamente la ventana de autenticación

PASO 2: AUTENTICACIÓN

1. **Ventana de login** aparece con interfaz moderna FideBurguesas
2. **Verificar conexión:** Sistema carga automáticamente datos de usuarios

---

👤 GESTIÓN DE USUARIOS

USUARIOS PRECARGADOS

El sistema viene con usuarios de diferentes roles: ● **Admin:** `admin` / `admin123` (Gestión completa)
● **Cajero:** `cajero1` / `cajero123` (Punto de venta)
● **Cocina:** `cocina1` / `cocina123` (Gestión de órdenes)

INICIAR SESIÓN

1. **Ingresar credenciales** en la ventana de login
2. **Click "Iniciar Sesión"** o presionar Enter
3. **Autenticación:** Sistema verifica usuario y contraseña
4. **Redirección automática:** Se abre la interfaz correspondiente al rol:
    - **Admin** → `AdminMainFrame` (Gestión completa)
    - **Cajero** → `CajeroMainFrame` (Punto de venta)
    - **Cocina** → `CocinaMainFrame` (Gestión de órdenes)

🖥️ NAVEGACIÓN SEGÚN ROL DE USUARIO

👨‍💼 INTERFAZ DE ADMINISTRADOR

PESTAÑA: DASHBOARD

- **Resumen general** del sistema
- **Estadísticas** de ventas y inventario
- **Indicadores** de rendimiento

PESTAÑA: USUARIOS

- **Lista completa** de usuarios del sistema
- **Crear nuevos usuarios** con roles específicos
- **Editar/Eliminar** usuarios existentes
- **Validaciones:** Username único, roles válidos

PESTAÑA: CATEGORÍAS

- **Gestión de categorías** de productos
- **Crear/Editar/Eliminar** categorías
- **Estado activo/inactivo** de categorías

PESTAÑA: PRODUCTOS

- **Inventario completo** con stock en tiempo real
- **Agregar nuevos productos** con validaciones
- **Control de stock** y alertas de reposición
- **Gestión de precios** y categorías

PESTAÑA: COMBOS

- **Crear combos** con descuentos
- **Gestionar productos** incluidos en combos
- **Cálculo automático** de precios con descuento

PESTAÑA: ÓRDENES

- **Historial completo** de órdenes
- **Estados de órdenes** (Pendiente, En Preparación, Listo, Entregado)
- **Detalles completos** de cada orden

PESTAÑA: REPORTES

- **Reportes de ventas** por período
- **Análisis de productos** más vendidos
- **Estadísticas** del sistema

👨‍💻 INTERFAZ DE CAJERO

PANEL PRINCIPAL: PUNTO DE VENTA

- **Catálogo visual** de productos organizados por categorías
- **Tarjetas modernas** con información del producto:
    - Nombre, precio, stock disponible
    - Imagen del producto (si disponible)
    - Botón "Agregar" funcional

TABLA DE ORDEN ACTUAL

- **Productos agregados** con cantidad y precios
- **Cálculo automático** de subtotal, impuestos y total
- **Botones de acción:** Finalizar Orden, Cancelar Orden
- **Edición inline** de cantidades

GESTIÓN DE COMBOS

- **Visualización de combos** disponibles
- **Precios con descuento** calculados automáticamente
- **Verificación de stock** de todos los productos del combo

👨‍🍳 INTERFAZ DE COCINA

PANEL DE ÓRDENES PENDIENTES

- **Lista de órdenes** en estado "Pendiente" y "En Preparación"
- **Detalles completos** de cada orden
- **Botones de acción:** Iniciar Preparación, Marcar Listo
- **Tiempo de procesamiento** visible

🔍 PRUEBAS FUNCIONALES RECOMENDADAS

PRUEBA 1: HERENCIA Y POLIMORFISMO

Objetivo: Demostrar herencia y polimorfismo implementados

1. **Login como Admin**
2. **Crear usuario nuevo:**
    - Ir a pestaña "Usuarios" → "Agregar Usuario"
    - Nombre: `Maria Rodriguez` ← Heredado de Persona
    - Username: `maria` ← Específico de Usuario
    - Tipo: `CAJERO` ← Enum de Usuario
3. **Crear producto:**
    - Ir a "Productos" → "Agregar Producto"
    - Código: `BURG001` ← Heredado de Item
    - Stock: `50` ← Específico de Producto

4. **Verificar herencia:** Observar propiedades comunes y específicas

PRUEBA 2: EXCEPCIONES PERSONALIZADAS

Objetivo: Demostrar manejo de excepciones StockInsuficienteException

1. **Login como Cajero**
2. **Crear producto con stock 0:**
    - Como Admin, editar un producto y poner stock = 0
3. **Intentar vender producto sin stock:**
    - Como Cajero, intentar agregar el producto a una orden

Resultado esperado: Mensaje de error controlado:
⚠️ Stock insuficiente para [Producto]

- Disponible: 0, Solicitado: 1

PRUEBA 3: COLECCIONES Y CRUD

Objetivo: Demostrar uso de List<> y operaciones CRUD

1. **Gestión de productos:**
    - Crear 3-4 productos diferentes
    - Verificar que aparecen en `List<Producto>`
    - Editar uno, eliminar otro
    - **Verificar:** La lista se actualiza dinámicamente
2. **Gestión de órdenes:**
    - Crear orden con múltiples productos
    - Verificar `List<OrdenDetalle>` se construye correctamente
    - **Observar:** Cálculos automáticos de totales

PRUEBA 4: MULTIHILOS EN ACCIÓN

Objetivo: Demostrar procesamiento concurrente

Verificar logs al inicio:
✅ MultithreadManager iniciado con 16 hilos

1. ✅ Backup automático en hilo: pool-2-thread-1

2. **Crear múltiples órdenes simultáneamente:**
    - Abrir varias ventanas de cajero (simular múltiples terminales)
    - Procesar órdenes en paralelo
    - **Verificar:** Sistema no se bloquea
3. **Backup automático:**
    - **Observar logs:** Backup funciona en segundo plano
    - **Verificar:** Interfaz sigue respondiendo durante backup

PRUEBA 5: SERIALIZACIÓN Y PERSISTENCIA

Objetivo: Demostrar backup automático con serialización

1. **Crear datos de prueba:**
    - Productos, categorías, usuarios, órdenes
2. **Verificar logs de backup:**
   ✅ Backup automático completado exitosamente

3. **Simular reinicio:**
    - Cerrar y reabrir aplicación
    - **Verificar:** Datos se mantienen (fueron serializados)

PRUEBA 6: INTERFAZ GRÁFICA COMPLETA

Objetivo: Demostrar GUI funcional y navegable

1. **Probar todos los roles:**
    - Login con Admin, Cajero, Cocina
    - **Verificar:** Interfaces diferentes según rol
2. **Navegación completa:**
    - Todas las pestañas funcionales
    - Formularios de crear/editar operativos
    - Tablas con datos en tiempo real
3. **Validaciones de formularios:**
    - Campos requeridos
    - Formatos válidos (email, precios, etc.)

📊 MONITOREO DEL SISTEMA

INFORMACIÓN EN CONSOLA

● **Estado multihilos:** Número de hilos activos
● **Backup automático:** Frecuencia y estado
● **Cache:** Estadísticas de uso y rendimiento
● **Órdenes:** Procesamiento asíncrono
● **Timestamp:** Hora de cada evento importante

CONTROLES DISPONIBLES

● **Gestión completa:** Desde interfaz de administrador
● **Backup manual:** Disponible en reportes
● **Estados de orden:** Workflow completo

🗂️ ARCHIVOS GENERADOS

usuarios.dat

- **Ubicación:** Directorio del proyecto
- **Contenido:** Usuarios serializados
- **Propósito:** Persistencia de autenticación

backup_sistema_[fecha].dat

- **Ubicación:** Directorio del proyecto
- **Contenido:** Backup completo serializado
- **Propósito:** Respaldo automático de datos

configuracion_sistema.dat

- **Ubicación:** Directorio del proyecto
- **Contenido:** Configuraciones del sistema
- **Propósito:** Persistencia de configuraciones

⚠️ SOLUCIÓN DE PROBLEMAS COMUNES

"Error de compilación"

- **Verificar:** Import de `StockInsuficienteException` en `CajeroMainFrame`
- **Acción:** Agregar import faltante
- **Recompilar:** Clean and Build Project

"Error en backup automático"

- **Verificar:** Todas las clases modelo implementan `Serializable`
- **Acción:** Agregar `implements Serializable` a clases faltantes
- **Verificar logs:** Confirmar backup exitoso

Ventana no responde

- **Cerrar aplicación** desde NetBeans
- **Verificar:** No hay múltiples instancias ejecutándose
- **Limpiar proyecto:** Clean and Build

Base de datos no inicializa

- **Verificar:** Configuración de base de datos en `DatabaseConfig`
- **Comprobar:** Permisos de escritura en directorio
- **Reiniciar:** Aplicación completa

🎓 CONCEPTOS TÉCNICOS DEMOSTRADOS

HERENCIA

- **Usuario extends Persona:** Propiedades comunes + específicas
- **Producto extends Item:** Comportamiento base + especializado
- **Jerarquía clara:** Reutilización de código

POLIMORFISMO

- **Interface Vendible:** Productos y combos vendibles
- **Interface Procesable:** Órdenes procesables
- **Mismo comportamiento:** Implementación específica por tipo

EXCEPCIONES PERSONALIZADAS

- **StockInsuficienteException:** Control de inventario
- **OrdenInvalidaException:** Validación de órdenes
- **Manejo gracioso:** Usuario informado, sistema estable

MULTIHILOS

- **MultithreadManager:** Gestión centralizada de hilos
- **Backup asíncrono:** No bloquea interfaz
- **Procesamiento concurrente:** Múltiples operaciones simultáneas

SERIALIZACIÓN

- **Backup automático:** Objetos → bytes → archivo
- **Persistencia:** Datos sobreviven reinicio
- **Configuraciones:** Sistema mantiene estado

COLECCIONES

- **List<Producto>:** Inventario dinámico
- **List<OrdenDetalle>:** Gestión de ventas
- **List<Usuario>:** Control de acceso

✅ VERIFICACIÓN DE ÉXITO

El proyecto funciona correctamente si:

✅ FUNCIONALIDADES BÁSICAS

- **Login exitoso** con diferentes roles
- **Interfaces específicas** cargan según usuario
- **CRUD completo** en todas las entidades
- **Navegación fluida** entre pantallas

✅ REQUERIMIENTOS DEL AVANCE 2

- **Clases y objetos:** Instanciación visible en todo el sistema
- **Herencia:** Usuario/Persona, Producto/Item funcionando
- **Polimorfismo:** Interfaces Vendible/Procesable operativas
- **Excepciones:** StockInsuficienteException se dispara correctamente
- **Colecciones:** Lists dinámicas en todas las funciones
- **Serialización:** Backup automático sin errores
- **GUI:** Interfaces completas y funcionales
- **Multihilos:** Logs confirman 16 hilos activos

✅ CALIDAD PROFESIONAL

- **Sin errores de compilación**
- **Logs limpios** sin excepciones no controladas
- **Rendimiento fluido** en todas las operaciones
- **Datos persistentes** entre ejecuciones

🏆 EVIDENCIA PARA EVALUACIÓN

Screenshots Recomendados:

1. **Logs de consola** mostrando multihilos y backup exitoso
2. **Pantalla de login** con autenticación funcional
3. **Interface de Admin** con gestión completa
4. **Interface de Cajero** con punto de venta operativo
5. **Mensaje de excepción** de stock insuficiente
6. **Lista de productos** mostrando herencia visible
7. **Órdenes procesándose** con estados dinámicos

Funcionalidades Clave Demostradas:

✅ **Arquitectura MVC** completa y funcional
✅ **Conceptos OOP** (Herencia, Polimorfismo, Encapsulación)
✅ **Programación concurrente** con multihilos
✅ **Manejo robusto** de excepciones

✅ **Serialización avanzada** para persistencia
✅ **Interfaz gráfica profesional** multiplataforma
✅ **Rendimiento optimizado** con cache y threads

¡Tu proyecto FideBurguesas-POS demuestra DOMINIO COMPLETO de Programación Cliente-Servidor Concurrente! 🚀🍔