09/05/2025

# Gestor de Restaurante

**Asignatura:**

Programación y bases de datos

**Profesor:**

Saul Moreno Mancebón.


Trabajo realizado por:

Javier Beltrán Artal (CEO)

Juan José Berdejo Tenorio (Tester)

Pablo Gargallo Sanz (Dev)

Eder Moros Rodero (Dev)

Joel Díaz Cano (Dev)

Carlos Delgado Zambrana (Dev)

Carlos de Val Escalante (Documentador)


# Índice

0.1 Introducción_____Pág-3

1.0 Documentación Técnica_____Pág-3

1.1 Tablas principales de la base de datos_____Pág-3

1.2 Triggers_____Pág-4

1.3 Procedimientos almacenados_____Pág-4

1.4 Vistas_____Pág-4

2.0 Funcionamiento_____Pág-5

2.1 Explicación de cada parte del programa_____Pág-5

2.1.1 bin_____Pág-5

2.1.2 SQL_____Pág-6

2.1.3 dao_____Pág-6

2.1.4 main_____Pág-7

2.1.5 model_____Pág-8

2.1.6 view_____Pág-8

2.1.7 Gitignore, Config.json, LICENSE, README, Run.bat__Pág-10-11

3.0 Requisitos_____Pág-10

4.0 Diseño_____Pág-11

5.0 Programa Fuente_____Pág-12

6.0 Guía de instalación _____Pág-12

7.0 Guías de uso y forma de uso de la Guía_____Pág-12

8.0 Descripción de la aplicación _____Pág-12

9.0 Especificaciones del sistema físico instalado_____Pág-13

10.0 Forma de comenzar la ejecución de la aplicación_____Pág-13

11.0 Orden en que se desarrollan los procesos_____Pág-13

12.0 Descripción de los procesos_____Pág-13

13.0 Ejemplos de uso del programa_____Pág-14

14.0 Documentación de pruebas (issues)_____

## 0.1 **Introducción**

Somos una consultora especializada en brindar soluciones tecnológicas innovadoras a empresas de diversos sectores. Nos enfocamos en entender las necesidades de cada cliente para desarrollar herramientas que optimicen sus procesos y mejoren su eficiencia.

En esta ocasión, hemos desarrollado una aplicación integral diseñada especialmente para la gestión eficiente de un bar/Restaurante. Esta aplicación permite centralizar operaciones como administración de pedidos, gestión de mesas, atención al cliente...

Con esto lo que haremos es facilitar una experiencia ágil para el personal.

## 1.0 **Documentación Técnica**

Base de datos: Esta base de datos está diseñada para gestionar de forma eficiente nuestra aplicación de restaurante incluyendo sus operaciones, administración, seguimiento y control de los datos históricos.

## 1.1 **Tablas principales de la base de datos:**

-Mesa: Esta registraría cada mesa del restaurante, dentro de mesa se encuentra número (es un identificador único), estado (indica si está ocupado o libre) y activo (si la mesa está activa en el sistema).

-Categoria: Organiza los productos en categorías (Entrantes, Bebidas...)

-Producto: Este incluye nombre, descripción, precio y categoría.

-Pedido: Esta tabla maneja los pedidos que hacen los clientes.

-Pedido_plato: Esta tabla guarda cada producto dentro de un pedido.

-Empleado: Esta tabla guarda la información de los empleados (nombre, puesto, salario...)

-Flujo_caja: Esta tabla registra ingresos y gastos del restaurante con descripción y fecha.

-Usuarios: Tabla de control de accesos del sistema, tiene un solo usuario administrador y usuarios normales.

-Historial_sesiones: Lleva el registro de inicios y cierres de sesión de cada usuario.

-Estadisticas_venta: Esta tabla registra cuántas unidades de cada producto se han vendido.

-AuditoríaGeneral: Esta tabla registra operaciones de borrado en Mesa

## 1.2 **Triggers:**

--**Triggers:** Estos se ejecutarán cuando ocurran otros eventos.

- auditar_borrado_mesa_general: Esto guarda en la tabla AuditoriaGeneral quién eliminó una mesa.

-prevenir_multiples_admins: Impide que exista más de un usuario con permisos de administrador.

-actualizar_total_pedido: Actualiza el total de un pedido cuando se agregan productos.

-agregar_actualizar_estadisticas_venta: Actualiza la cantidad vendída de cada producto.

1.3 **Procedimientos almacenados:**

--**Procedimientos almacenados:** Esto permite realizar operaciones de forma estructurada y repetida.

-Crear_pedido: Crea un nuevo pedido.

-agregar_producto_a_pedido: Cambia la cantidad de un producto.

-eliminar_producto_de_pedido: Elimina un producto del pedido de una mesa y ajusta el total.

-CrearOModificarMesa: Si La mesa existe y esta inactiva, se activa, si la mesa no existe, crea una mesa.

1.4 **Vistas:**

--**Vistas:** Consultas que permiten ver información de forma estructurada.

**-**vista_ventas_detalladas: Muestra detalles de cliente, productos y totales.

-vista_productos_activos: Lista productos que estan activos con sus respectivas categorías.

-vista_historial_pedidos: Son pedidos que ya han sido pagados ordenados por fecha.

-vista_pedidos_detallados: Pedidos con productos y datos para el seguimiento.

2.0 **Funcionamiento**

El sistema carga la configuración del restaurante desde un archivo externo usando ConfigLoader lo que permite cambiar ajustes sin modificar código.

Este se conecta a la base de datos usando ConexionBD, si hay errores los muestra por consola.

funcionamiento de la aplicación:

Se carga la configuración, se conecta a la base de datos, y desde ahí ya podemos hacer las gestiones necesarias para utilizar la aplicación, dentro de esta aplicación tendremos un administrador el cual tendrá todos los permisos y habrá más usuarios en este caso serán los camareros que tendrán permisos para tomar comandas y hacer cobros.

## 2.1 **Explicación de cada parte del programa**

### 2.1.1 **bin**

La clase bin contiene los archivos compilados del código fuente

En la carpeta lib tenemos 4 archivos que estos son:

**jackson-annotations-3.0-rc3.jar**: contiene anotaciones Java utilizadas por las otras bibliotecas de Jackson que tenemos.

**jackson-core-2.19.0-rc2.jar:** Este nos ofrece las clases fundamentales para el procesamiento de bajo nivel de JSON

**jackson-databind-2.19.0-rc2.jar:** Este archivo combina los dos anteriores para poder serializar o deserializar objetos.

**mysql-connector-j-9.2.0.jar:** Con este archivo podemos conectar el archivo java y la base de datos para que funcionen de forma conjunta.

### 2.1.2 **SQL**

Después, nos encontramos la carpeta SQL, dentro de esta tenemos el archivo(seed.sql), este archivo será nuestra base de datos con las tablas como están organizadas.

Ahora nos encontramos la carpeta src en la cual tendremos las principales carpetas del programa, comenzare explicando la carpeta **config** en la que tenemos las clases:

**ColorPaleta.java:** esta clase lo que hace es actuar como una paleta de colores para nuestra aplicación, específicamente para nuestra interfaz gráfica de la tpv.

**Config.java:** esta clase actúa como estructura de datos la cual almacena y accede a la configuración de nuestra aplicación, esto quiere decir que almacena toda la configuración relevante del restaurante.

**ConfigCaja.java:** esta clase se utiliza para representar la configuración de la caja registradora de nuestro sistema, esta tiene diversas propiedades que son:(dar cambios, permitir descuentos y un máximo de descuento).

**ConfigLoader.java:** esta clase nos permite que el resto de la aplicación de nuestro restaurante de una forma que esté lista para usar y de forma estructurada.

**Horario.java:** sirve para definir los turnos de trabajo y el horario de atención al cliente, es decir los turnos de desayunos, comidas y cenas.

**Impresora.java:** esta clase representa la impresora que tenemos conectada a nuestro sistema del restaurante, lo que nos permite registrar múltiples impresoras, esta impresora será configurable para poder indicar su ubicación en el restaurante (es decir, si está en la cocina o en barra)

**Tax.java:** esta clase sirve para poner los impuestos correspondientes por zona entre la barra, terraza y comedor.

### 2.1.3 **dao**

Ahora nos encontramos las carpetas **dao**, estas carpetas contienen los siguientes archivos:

**CategoriaDAO.java:** esta clase gestiona el poder crear, borrar, modificar, mostrar y obtener categoría

**ConexionBD.java:** esta clase nos permite conectarnos con la base de datos, también define los parámetros de conexión.

**HistorialSesionesDAO.java:** se encarga de registrar en la base de datos las sesiones de los usuarios.

**MesaDAO.java:** Esta clase maneja las operaciones relacionadas con la tabla Mesa en la base de datos, esto quiere decir que encapsula la lógica para poder crear, leer, actualizar y eliminar mesas.

**PedidoDAO.java:** se encarga de las operaciones relacionadas con pedidos, en este tenemos las opciones de insertar, consultar, modificar y eliminar información de pedidos y de platos asociados a una mesa.

**ProductoDAO:java:** Encapsula todas las operaciones de acceso a la base de datos relacionadas con los productos, lo que puede crear, borrar, modificar, mostrar y buscar productos.

**UserDataDAO.java:** gestiona los usuarios en la base de datos, este tiene cuatro métodos que nos permiten: validarCredenciales (si el nombre de usuario y la contraseña son correctos), obtenerUsuario (si este existe nos lo devuelve), existenUsuarios (esta consulta cuantos usuarios existen) y newUsuario( esta inserta un nuevo usuario.

```
String sql = "SELECT contraseña FROM Usuarios WHERE BINARY nombre_usuario = ?";
```

Cuando estábamos probando el inicio de sesión nos dimos cuenta de que el encoding de caracteres predeterminado de MySQL no es sensible a mayúsculas y minúsculas al comparar cadenas de texto.

La solución más rápida que encontramos es hacer una comparación binaria ya que al traducirla a código binario una 'A' mayúscula y una 'a' minúscula tiene distinto valor.

### 2.1.4 **main**

Ahora pasamos a la carpeta **main** en la cual tenemos las siguientes clases:

**App.java:** esta clase será nuestro punto de entrada de la aplicación, si no existe un usuario nos creará un usuario administrador inicial, después de esto se nos solicitará crear un usuario administrador. sí existe usuario se solicitará el inicio de este con la validación de sus credenciales.

Después se establece una conexión con la base de datos y cuando un usuario inicia sesión se registra en la base de datos.

Esta aplicación tiene un JFrame principal y usa JDialog para el inicio de sesión y la creación de usuarios.

**TPVMain.java:** será nuestra clase principal del sistema de punto de venta, TPVMain será nuestro contenedor visual que nos permite gestionar la navegación visual entre las distintas secciones de nuestra app.

### 2.1.5 **model**

**Carta.java:** esta clase representa el menú del restaurante, la tenemos estructurada en categorías de productos.

**Categoria.java:** Representa una sección del menú dentro de (Bebidas, entrantes y postres) conteniendo una lista de productos pertenecientes a esta.

**LogSesion.java:** aquí se guardan un historial de sesiones de usuario

**Mesa.java:** esta clase representa las mesas del restaurante, nos permite saber si la mesa está ocupada o libre.

**Pedido.java:** la clase pedido representa un pedido realizado por una mesa, esta registra que mesa ha hecho el pedido, guarda fecha y hora a la que se ha realizado ese pedido.

**PedidoPlato.java:** relación entre un pedido y un producto,Pedidoid es el identificador del pedido,codigoProducto es el código del plato y cantidad es el número de unidades del plato que se han pedido.

**Producto.java:** es un plato del menú el cual incluye su información básica y estado de disponibilidad.

**UserData:java:** esta clase representa un usuario del sistema incluyendo su información y rol dentro de la empresa.

2.1.6 **view**

**Boton.java:** esta clase la hemos diseñado para crear botones con estilos visuales personalizados, lo que hará esta clase es que mejorará la apariencia y la experiencia del usuario.

**CartaVIEW.java:** esta clase nos va a mostrar los menús que nos permite interactuar con la carta, esta clase tiene diversas opciones: mostrar la carta, modificar la carta, eliminar la carta y crear la carta.

**CategoriaVIEW.java:** esta clase se encarga de gestionar las interacciones del usuario con la carta, esta clase nos permite: crear una nueva categoría, modificar una categoría que ya exista y eliminar una categoría.

**MesaVIEW.java:** esta clase nos proporciona una interfaz de texto para que el usuario pueda gestionar las mesas del restaurante, esta maneja la lógica de acceso a los datos. Esta clase puede: crear una mesa, cambiar el estado de una mesa y eliminar una mesa, buscar una mesa y mostrar una mesa

**ProductoVIEW.java:** La clase producto nos va a ayudar a gestionar productos en nuestra aplicación, esta clase nos permitirá: crear un producto, modificar un producto, eliminar un producto y mostrar producto

**VistaCobro.java:** esta clase es una interfaz gráfica la cual se encargar de mostrar al usuario que lo utilice los detalles del pedido de una mesa y poder gestionar el pago mediate efectivo o tarjeta.

**VistaConfiguracion.java:** Permite al usuario realizar tareas administrativas como crear, modificar y eliminar (productos, platos, mesas y generar un resumen diario).

**VistaMenu.java:** esta clase permitirá seleccionar una categoría y un producto y mostrarlo en el pedido, es decir, le permitirá al camarero tomar un pedido de una mesa.

**VistaMesas.java:** esta clase es la responsable de mostrar la gestión de mesas y permitir la interacción con ellas.

Con esta clase obtenemos un botón correspondiente a cada mesa y al hacer click en el botón de una mesa, navegamos hacía el menú correspondiente a esa mesa.

## Gitignore

Ahora paso a explicar el .**gitignore:** este se usa para indicar a Git que archivos y carpetas no deben de subirse al repositorio.

## Config.json

**Config.json:** este archivo se carga nada más iniciar el programa y se utiliza para personalizar el comportamiento de la tpv( nos mostrará el nombre del restáurate en pantallas y tickets), 'taxes' nos aplicará impuestos según la ubicación, 'impresoras' configurará impresoras por zonas, 'horarios' define horarios de trabajo,'config_caja' configura reglas de la tpv.

**LICENSE:** archivo de licencia MIT.

**README:** este archivo nos sirve para proporcionar una descripción de lo que hace la aplicación, su organización y como se puede ejecutar.

**Run.bat:** sirve para compilar y ejecutar el programa desde la consola de Windows, Linux o macOS.

## 3.0 Requisitos

Requisitos funcionales:

1. La aplicación debe permitir a los usuarios registrarse y autenticarse

2.Debe guardar los datos en la base de datos de forma correcta y poder consultarlos.

3.Debe permitir (crear, leer, actualizar y eliminar) Productos.

Requisitos no funcionales:

1.La aplicación debe responder con rapidez con cada solicitud.

2.La interfaz es intuitiva y compatible con diferentes dispositivos.

3. El sistema está protegido contra inyecciones SQL.

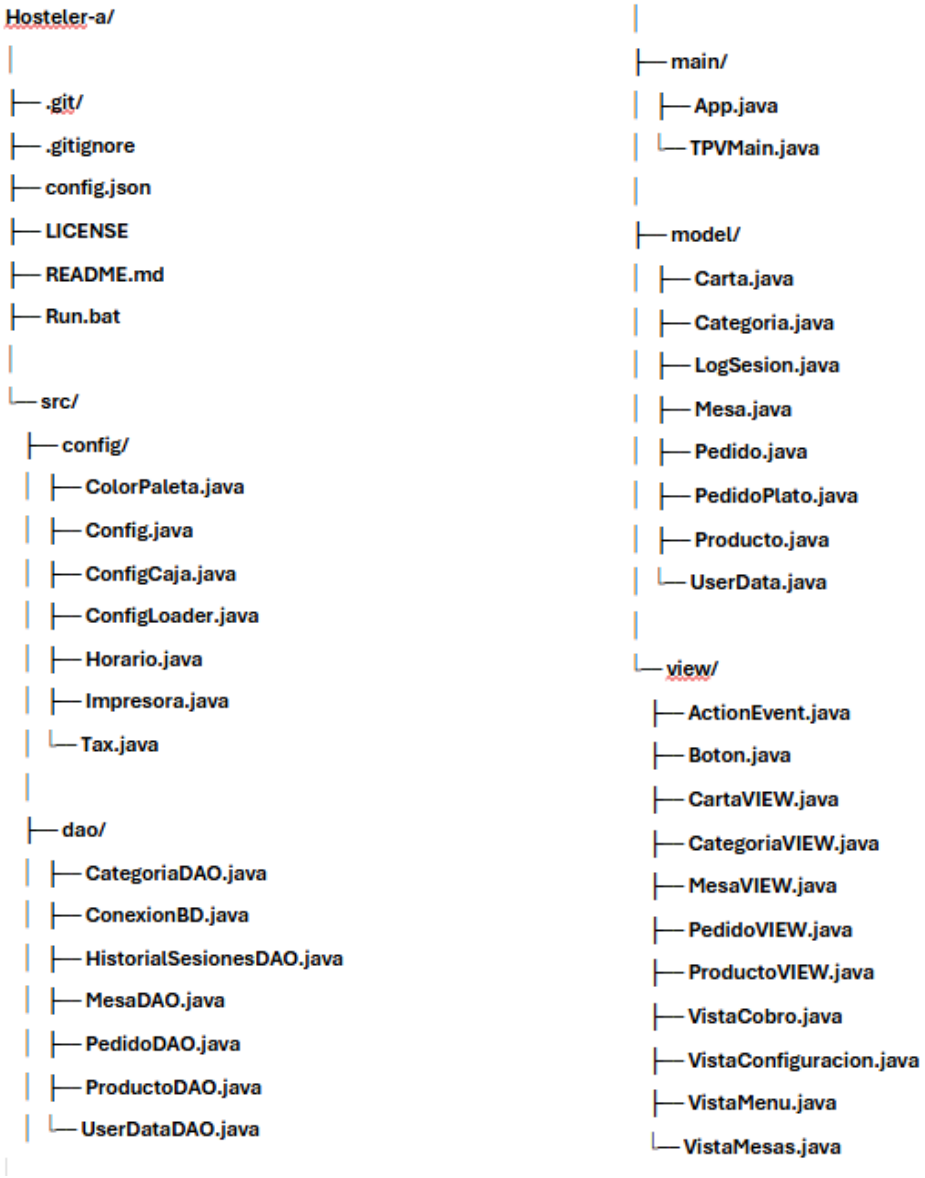
Requisitos técnicos:

- Lenguaje: Java 8 o superior.
- JDK requerido: Java Development kit 8 o superior.
- Entorno de desarrollo recomendado: Eclipse.
- Base de datos: MySQL
- Sistema operativo: Compatible con Windows, macOS y Linux.

4.0 **Diseño**

Este programa está escrito en java y presenta una estructura clara y modular.

La estructura del proyecto seria de esta manera:




## 5.0 **Programa Fuente:**

TPVMain.java: Este es un archivo que actúa como el punto de entrada del programa, este crea una instancia de TPVMain lo cual inicia la aplicación Principal, esta es la interfaz principal y necesita App.java para delegar tareas de backend.

App.java: Punto de entrada principal, maneja la configuración, autenticación y funciones como generar facturas y reportes.

## 6.0 **Guía de instalación:**

Explicación paso a paso de Instalación del programa:

Lo primero de todo debes asegurarte de tener instalado JDK 8 o superior

1. Descomprime el proyecto en tu carpeta de trabajo
2. Abre el proyecto en tu IDE:
   En visual estudio:
   1. Busca en la parte de arriba a la izquierda archivo.
   2. Seleccionas abrir carpeta
   3. Seleccionas la carpeta donde está el proyecto.

3. Compila el proyecto (este se debería compilar automáticamente)
4. Ejecuta el programa utilizando el ./Run.bat o haciendo click sobre el archivo y seleccionando (run file)

## 7.0 **Guía de uso**

1. **Forma de uso de la guía:** esta guía esta creada para el administrador del sistema, camareros y técnicos que deseen instalar o hacer mantenimiento de la aplicación.
   Sigue cada apartado paso por paso para entender como instalar, ejecutar y operar la aplicación de forma correcta.

## 8.0 **Descripción de la aplicación**

Esta es una aplicación diseñada para bares y restaurantes, lo que permite hacer esta aplicación es:

- Controlar mesas y pedidos.
- Gestionar productos y menús
- Cobrar mediante efectivo y tarjeta.
- Generar facturas e informes de ventas de forma diaria.
- Configurar la caja, impuestos y horarios.

9.0 **Especificaciones del sistema físico instalado**

- **S.O**: Windows 10 (o superior), Linux o macOS.
- **Procesador:** i3 o superior.
- **Memoria RAM:** 4GB o superior
- **Almacenamiento:** 1 GB de espacio de almacenamiento libre.

**Requisitos software:**

- Java JDK 8 o superior.
- Conexión a base de datos MSQL.

10.0 **Forma de comenzar la ejecución de la aplicación**

Ejecutar el archivo ./Run.bat

11.0 **Orden en que se desarrollan los procesos**

1. Iniciar aplicación.
2. Cargar la configuración.
3. Hacer la verificación de usuarios: si no existe un admin se crea automáticamente.
4. Inicio de sesión.
5. Carga interfaz de la TPV.
6. Gestión de mesas, pedidos y cobros.
7. Generación de facturas.
8. Cierre de jornada y reporte diario.

12.0 **Descripción de los procesos**

1. Inicio de sesión: usuario inicia sesión con la clave, este registro se guarda en el historial.
2. Gestión de mesas: visualiza las mesas y pedidos activos.
3. Creación de pedidos: se pueden seleccionar productos desde la carta, editar las cantidades y eliminar productos del pedido.
4. Cobro de pedidos: Hace el cierre del pedido, elección del tipo de pago y la generación del ticket.
5. Facturación: Genera un documento HTML con los detalles del pedido.
6. Reporte diario: Genera un resumen de ventas del día.

## 13.0 **Ejemplos de uso del programa**

Camarero toma un pedido:

1. Inicia sesión.
2. Entra en mesas.
3. Selecciona una mesa libre.
4. Agrega productos desde el menú.
5. Confirma pedido.

1. Cobro y cuenta:

1. Seleccionar una mesa con un pedido activo.
2. Entrar en cobrar.
3. Elige efectivo.
4. Imprime ticket.
5. Pedido se marca como cerrado.

Cierre de Jornada

1. Entra en la configuración
2. Selecciona “Resumen del día”
3. Se genera un HTML con los totales.

## 14.0 **Documentación de pruebas (issues)**

Se han hecho comprobaciones en los valores de entrada para que no puedan ser negativos.

Había un error a la hora de eliminar y crear mesa.

Crear botón eliminar producto del pedido.

No se puede eliminar las categorías.

Añadir un botón para imprimir facturas.

Botón de volver al inicio desde configuración tiene un bug el cual te lleva a la vista de mesas en vez de al inicio.

Objeto app no se inicializa en vistacobro.

Error al iniciar el programa, al iniciar el programa se tendría que mostrar un panel de inicio con distintas opciones y sale solo el menú de las mesas.

A la hora de seleccionar productos para el pedido no deja seleccionar dos veces el mismo producto, tienes que salirte a categorías y volver a entrar o añadir otro producto primero.