Tecnológico Nacional De México

Instituto Tecnológico de Oaxaca

Ingeniería En Sistemas Computacionales

Diseño e Implementación de Software con Patrones.

7 am – 8 am

Unidad 3

“Patrón de Diseño Singleton”

Presenta:

Implementación de los Patrones en el Sistema Copy Max

Nombres	Numero de Control
Bautista Fabian Max	**C19160532**
Celis Delgado Jorge Eduardo	**21160599**
Flores Guzmán Alan Ismael	**20161193**
García Osorio Bolívar	**20161819**
Pérez Barrios Diego	**21160750**
Pérez Martínez Edith Esmeralda	**21160752**
Sixto Morales Ángel	**21160797**

Periodo Escolar:

Febrero – Julio

Grupo:

7SB

Maestro:

Espinoza Pérez Jacob

Oaxaca de Juárez, Oaxaca **Abril 2025**

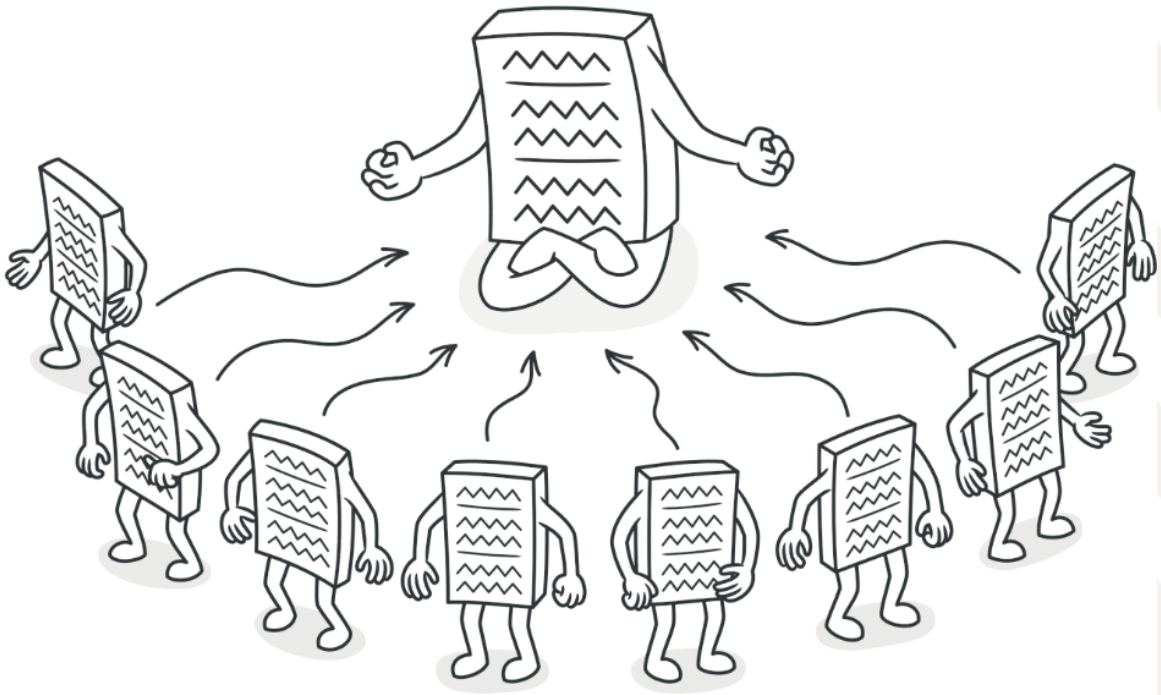
INDICE

INTRODUCCIÓN

El patrón Singleton garantiza que una clase tenga una única instancia accesible de forma global, y al mismo tiempo, proporciona un punto centralizado para su acceso. Por ejemplo, si todos los formularios del sistema necesitan consultar la configuración actual del usuario logueado o acceder a la conexión activa de la base de datos, el Singleton asegura que todos trabajen con la misma instancia, evitando duplicidad de información o errores de sincronización.

PATRÓN DE DISEÑO SINGLETON: IMPLEMENTADO EN COPY MAX

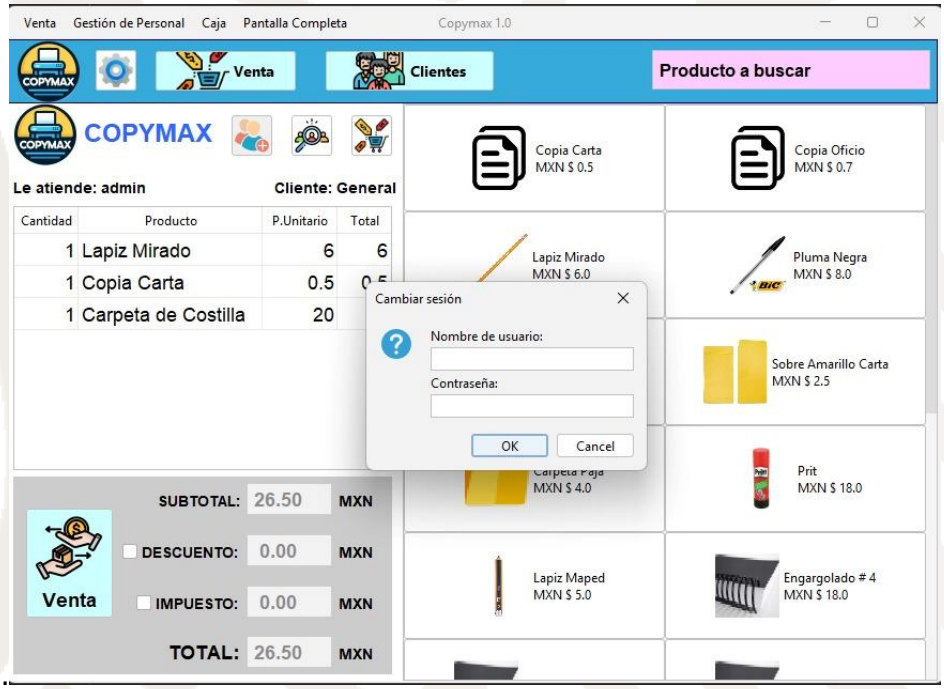
Singleton es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia .



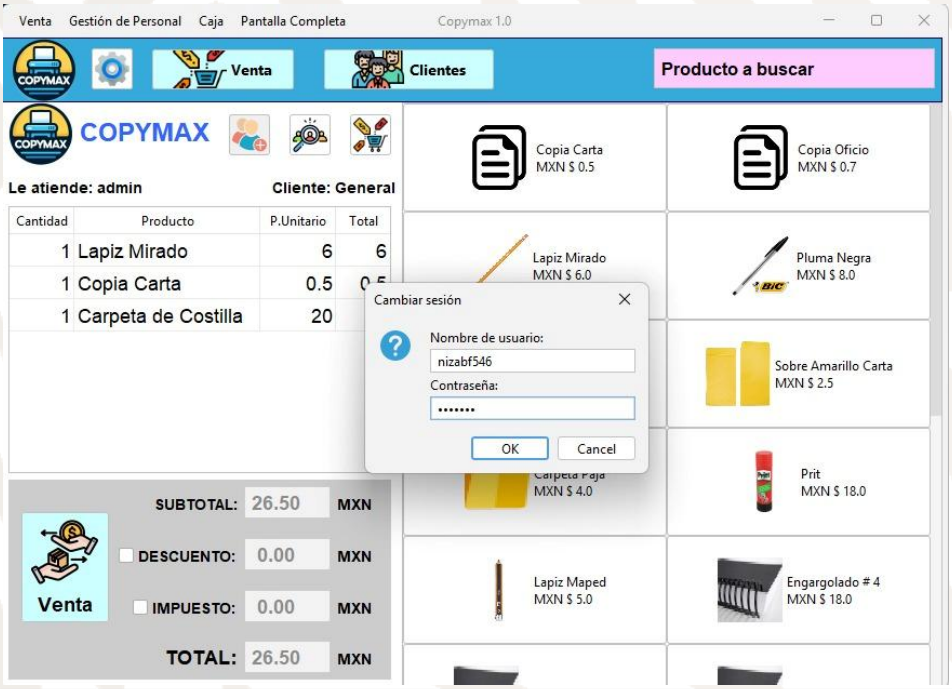
La clase Usuariosesion representa una única sesión de usuario activa en el sistema. Aplica el patrón Singleton para asegurarse de que solo exista una instancia durante el ciclo de vida del sistema Copy Max.

CÓDIGO IMPLEMENTADO

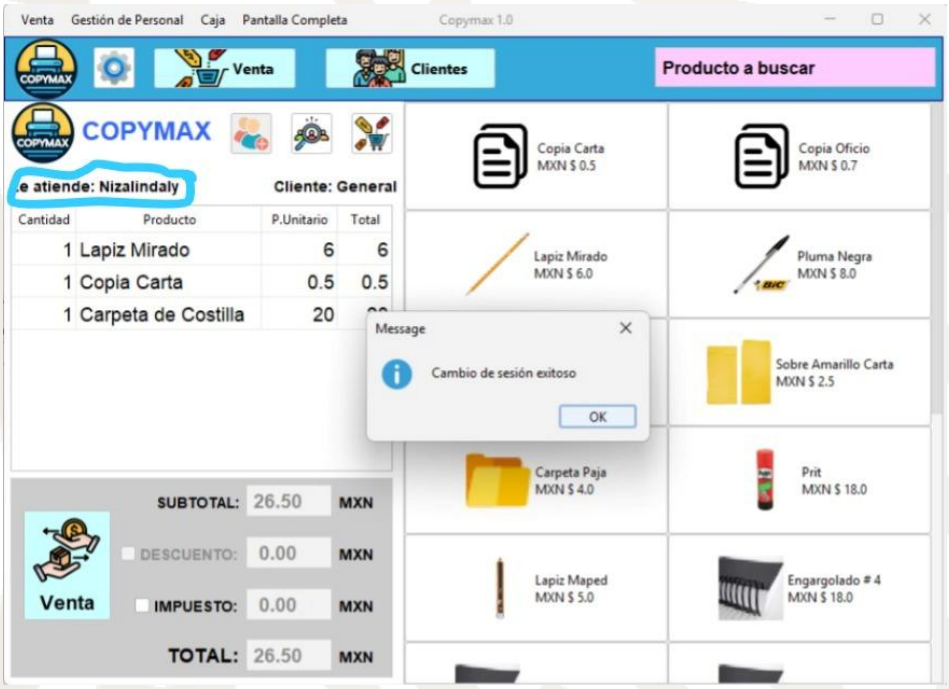
Para revisar la implementación del patrón necesitamos entrar como usuario administrador



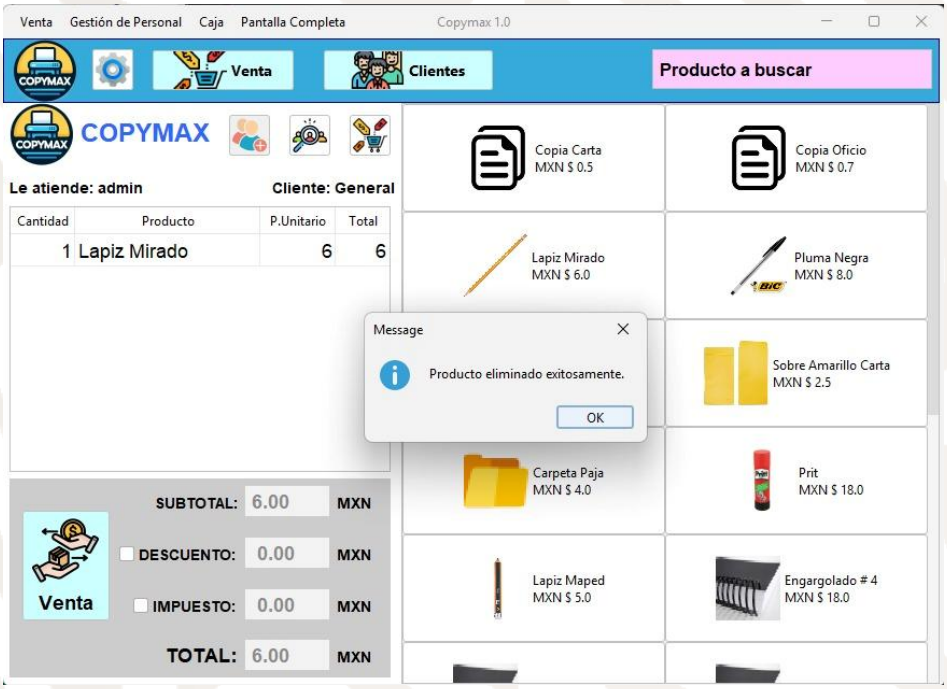
O como también iniciar como un usuario normal.



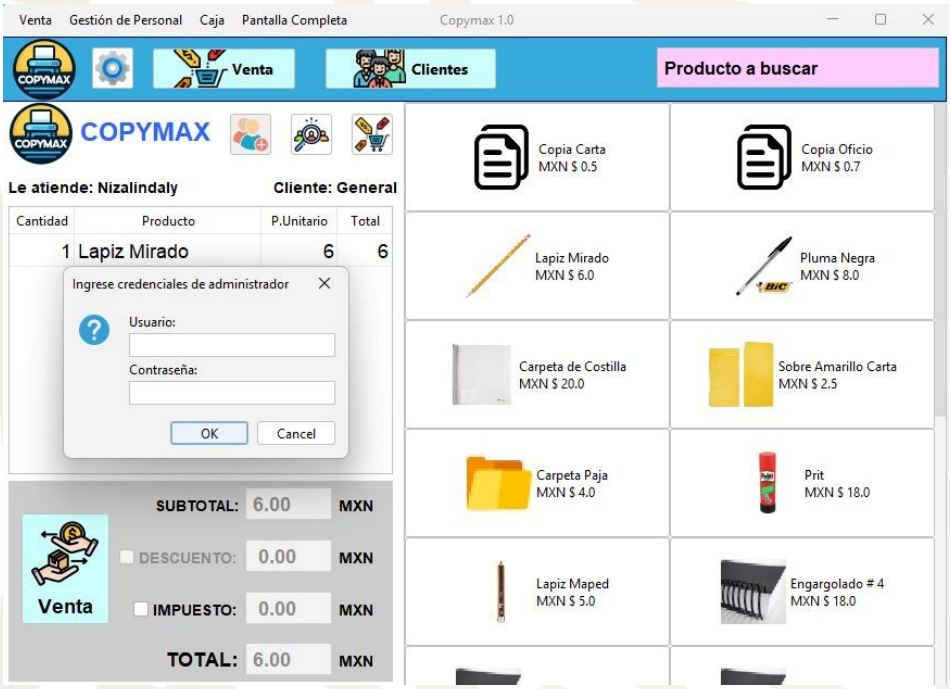
Donde veremos que al ser asi se actualizara el usuario y los permisos del programa dependiendo de cómo se esté iniciando.



En administrador se podrán borrar directamente directo los item de las ventas.



Mientras que un usuario cajero preguntara por la validación del administrador para poder realizar la operación.




```
package Modelo;

import Conexion.Conexion;

import javax.swing.JOptionPane;


public class Usuariosesion {

    private static Usuariosesion instance;

    private String nombre;

    private String nombrereal;

    private int idUsuario;
```

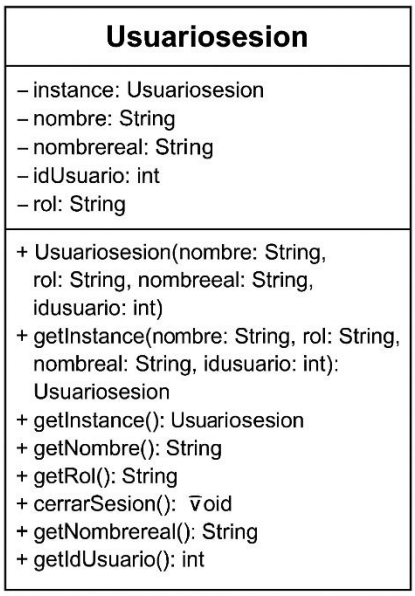
```
    private String rol;

    private Usuariosesion(String nombre, String rol, String nombreeal,int idusuario) {
        this.nombre = nombre;
        this.rol = rol;
        this.nombrereal=nombreeal;
        this.idUsuario=idusuario;
    }

    public static Usuariosesion getInstance(String nombre, String rol, String nombreal, int
idusuario) {
    if (instance == null) {
        instance = new Usuariosesion(nombre, rol, nombreal, idusuario);
    } else {
        instance.setUsuario(nombre, rol, nombreal, idusuario);
    }
    return instance;
}
    public static Usuariosesion getInstance() {
        return instance;
    }

    public String getNombre() {
        return nombre;
```

```
    }

    public String getRol() {
        return rol;
    }

    public void cerrarSesion() {
        instance = null;
    }
     public String getNombrereal() {
        return nombrereal;
    }

     public int getIdUsuario() {
        return idUsuario;
    }

     public void setUsuario(String nombre, String rol, String nombreal, int idusuario) {
    this.nombre = nombre;
    this.rol = rol;
    this.nombrereal = nombreal;
    this.idUsuario = idusuario;
}
}
```

Explicacion de la Clase UsuarioSesion

- La instancia se crea con el método getInstance(String, String, String, int), y solo se permite una sola instancia.
- El método getInstance() sin parámetros permite acceder a la instancia ya existente.
- La clase mantiene información del usuario: nombre, nombrereal, rol e idUsuario.
- El método cerrarSesion() elimina la instancia (permite cerrar la sesión y reiniciar el singleton).
- setUsuario(...) permite actualizar los datos del usuario si ya hay una instancia creada.

ESTRUCTURA UML



VENTAJAS Y DESVENTAJAS

Ventajas	Desventajas
Instancia única garantizada: Evita la creación innecesaria de múltiples objetos.	Difícil de testear: Puede dificultar las pruebas unitarias por su naturaleza global.
Acceso global controlado: Proporciona un punto central de acceso desde cualquier parte del sistema.	Puede generar acoplamiento: Su uso excesivo puede llevar a una dependencia global innecesaria.
Ideal para recursos compartidos: Útil para conexiones, configuración o control de sesión.	No es seguro en hilos si no se implementa correctamente.

CONCLUSIÓN

El patrón Singleton ofrece una solución simple y eficaz para manejar instancias únicas dentro del sistema de una papelería, especialmente cuando se trata de recursos globales como conexiones, configuraciones o sesiones de usuario. Sin embargo, su uso debe ser cuidadoso y justificado, ya que puede derivar en código difícil de testear o mantener si se abusa de él como "puente" entre módulos. Bien implementado, el Singleton mejora la eficiencia y coherencia del sistema, facilitando el control y acceso uniforme a recursos críticos.