**Informe del Proyecto: Cronómetro Pomodoro Personalizable**

## **Contexto**

En la actualidad, la gestión eficiente del tiempo representa un desafío constante, especialmente en contextos que requieren concentración sostenida, como el estudio o el trabajo intelectual. El método Pomodoro ha demostrado ser una estrategia útil para mejorar la productividad al dividir el tiempo en intervalos de trabajo y descanso. Sin embargo, muchas de las herramientas disponibles en el mercado presentan limitaciones, particularmente en cuanto a la personalización de los tiempos y la integración con recursos externos como música.

## **Definición del Problema**

Se ha identificado que diversas aplicaciones Pomodoro no ofrecen la flexibilidad necesaria para ajustarse a las necesidades específicas de cada usuario. En particular, suelen carecer de:

* Intervalos de tiempo configurables.
* Integración directa con plataformas de música.
* Experiencias unificadas, lo que obliga al usuario a cambiar de plataforma constantemente.

Esto interrumpe el flujo de concentración y reduce la eficacia de las sesiones Pomodoro.

## **Propuesta de Solución**

Se desarrolló una aplicación de consola que permite configurar un cronómetro Pomodoro de manera totalmente personalizable. Las características principales de esta solución son:

* Permitir al usuario definir la duración de los ciclos de trabajo y descanso.
* Reproducir playlists personalizadas desde YouTube Music para cada fase del ciclo.
* Mostrar frases motivacionales al finalizar cada intervalo.
* Almacenar las preferencias del usuario en un archivo de configuración.
* Permitir modificar la configuración en cualquier momento.

Este enfoque centraliza todas las funcionalidades en una sola herramienta, eliminando la necesidad de alternar entre múltiples aplicaciones.

## **Justificación de la Solución**

La implementación de esta aplicación mejora la experiencia del usuario al permitirle mantener el enfoque sin distracciones. Consolidar las funciones de temporización, música y personalización en un solo entorno:

* Aumenta la eficiencia en el uso del tiempo.
* Facilita la adopción de hábitos de estudio o trabajo saludable.
* Proporciona un entorno adaptable a cada persona, incluso sin conexión a internet o sin interfaces gráficas complejas.

## **Diseño de Clases**

La aplicación está implementada en Java, con las siguientes clases principales:

**Main**: Clase principal que inicia la aplicación. Crea instancias de las demás clases y ejecuta el flujo principal.

### **Usuario**: Gestiona la configuración del usuario, incluyendo tiempos personalizados y URLs de playlists. Lee y guarda esta configuración en un archivo de texto (config.txt), y valida los datos ingresados.

### **Temporizador**: Simula el paso del tiempo para los intervalos de trabajo y descanso. Incluye la funcionalidad de temporización con salida por consola.

### **Musica**:Reproduce y detiene la música en el navegador mediante un proceso del sistema. Soporta la carga de diferentes playlists según la etapa del ciclo.

### **Interfaz**: Coordina la interacción con el usuario. Controla la lógica completa del ciclo Pomodoro, desde la reproducción de música hasta la presentación de frases motivacionales.

### **TiempoInvalidoException y UrlInvalidaException**: Excepciones personalizadas que permiten validar correctamente los valores ingresados por el usuario.

## **Reflexión sobre los Principios SOLID**

### **Principio de Responsabilidad Única (SRP),** cada clase tiene una única responsabilidad bien definida:

* Usuario maneja la configuración.
* Temporizador ejecuta el conteo de tiempo.
* Musica se encarga de la reproducción musical.
* Interfaz orquesta el flujo de uso.
* Main coordina el inicio de la aplicación.

### **Principio Abierto/Cerrado (OCP),** las clases pueden extenderse sin necesidad de ser modificadas. Por ejemplo:

* Usuario podría guardar nuevas configuraciones sin alterar su funcionamiento actual.
* Musica puede extenderse para incluir otros servicios de música.

### **Principio de Sustitución de Liskov (LSP),** no se aplica directamente, ya que no se utilizó herencia funcional. Todas las clases son concretas y no dependen de supertipos.

### **Principio de Segregación de Interfaces (ISP),** no se definieron interfaces explícitas, pero las clases están organizadas para no obligar a depender de funcionalidades que no usan.

### **Principio de Inversión de Dependencias (DIP),** no fue aplicado en esta versión, ya que las dependencias se instancian directamente. Sin embargo, el diseño modular permite incorporar este principio fácilmente en una versión más avanzada.

## **Diseño de la Persistencia**

La persistencia de datos se basa en archivos de texto plano para asegurar que la configuración del usuario se mantenga entre sesiones.

### **Archivo config.txt:** Este archivo guarda las preferencias del usuario en formato clave=valor. Es gestionado por la clase Usuario.

### **Archivo frases.txt:** Contiene frases motivacionales, una por línea, que son cargadas al inicio y presentadas aleatoriamente tras cada ciclo Pomodoro.

## 

## **Diagrama de Clases**

