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.
- Interfaz intuitiva que unifique todos los elementos de la experiencia Pomodoro.

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

Propuesta de Solución

Se desarrolló una aplicación Pomodoro personalizable con interfaz gráfica utilizando Java Swing. 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 desde la consola.
- Controlar el ciclo Pomodoro mediante botones visuales: iniciar, pausar y detener.

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

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, motivación y control visual en un solo entorno:

- Aumenta la eficiencia en el uso del tiempo.
- Facilita la adopción de hábitos de estudio o trabajo saludables.
- Proporciona un entorno adaptable a cada persona, incluso sin conexión a internet o sin depender de herramientas externas complejas.

Diseño de Clases

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

- **Main**: Clase principal que inicia la aplicación. Crea la vista y el controlador.
- **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. Permite pausar y detener el conteo.
- **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 y la reproducción de una alarma al finalizar los ciclos.
- **PomodoroView**: Implementa la interfaz gráfica con Swing. Incluye etiquetas para mostrar estado, tiempo y tipo de ciclo, además de botones para controlar el flujo.
- **PomodoroController**: Coordina la lógica entre el modelo (Temporizador, Usuario, Musica) y la vista (PomodoroView), incluyendo la respuesta a eventos de los botones.
- **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.

- **Principio Abierto/Cerrado (OCP)**: Las clases permiten ser extendidas (por ejemplo, el uso de nuevas plataformas de música o ajustes adicionales en la configuración) sin modificar el código base.

- **Principio de Sustitución de Liskov (LSP)**: No se utiliza herencia funcional, por lo tanto, no se aplica directamente.

- **Principio de Segregación de Interfaces (ISP)**: Aunque no se definieron interfaces explícitas, las clases están estructuradas para no depender de funcionalidades que no utilizan.

- **Principio de Inversión de Dependencias (DIP)**: No se aplicó directamente, ya que las dependencias se crean internamente, pero el diseño modular lo permitiría fácilmente.

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**: 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.

La implementación de la interfaz gráfica ha elevado la usabilidad general del sistema, facilitando su adopción por parte de cualquier usuario, sin necesidad de conocimientos técnicos.

Diagrama de Clases