Informe del Proyecto de Lenguajes de Programación

Nombre: Francisco Fabian Gonzalez Suarez

3ro Ciencias de la Computación

Título del Proyecto: Sistema de Control de Gastos Personales con Interfaz Gráfica

Repositorio: https://github.com/Fabian11GS/finance-proyect

# Introducción General

El proyecto desarrollado es una aplicación de escritorio orientada a la gestión de finanzas personales. Su objetivo principal es permitir a los usuarios planificar metas de ahorro, registrar gastos, controlar avances y visualizar reportes detallados. Esta aplicación fue construida completamente en Python, utilizando la biblioteca `customtkinter` para la interfaz gráfica y el módulo `json` para persistencia de datos.  
  
El diseño del proyecto toma como base la integración de varios paradigmas de programación, lo cual aporta robustez, mantenibilidad y flexibilidad al sistema. En este informe se analizan los aspectos técnicos del código desde la óptica de los paradigmas de programación estructurado, orientado a objetos, funcional, concurrente y basado en eventos.

# Paradigma de Programación Estructurada

La programación estructurada constituye el fundamento lógico del sistema. Este paradigma se caracteriza por la organización secuencial del código, el uso de funciones para segmentar tareas y la implementación de estructuras de control.

- Funciones puras y auxiliares:  
 - guardar\_datos(datos): recibe un diccionario y lo guarda en un archivo JSON.  
 - cargar\_datos(): carga los datos almacenados en el archivo y los devuelve como diccionario Python.  
- Estructuras de control:  
 - Condicionales if y try-except para manejo de errores.  
 - Ciclo while en monitorear\_planes() para ejecuciones periódicas.

# Programación Orientada a Objetos (POO)

El corazón de la aplicación está basado en el paradigma orientado a objetos, lo cual permite encapsular datos y comportamientos relacionados dentro de estructuras lógicas reutilizables llamadas clases.

Clase principal: ControlGastosApp  
- Hereda de ctk.CTk y encapsula lógica y datos.  
- Principios aplicados:  
 - Encapsulación de atributos y métodos.  
 - Herencia de funcionalidades de la librería.  
 - Modularidad con métodos específicos como crear\_plan(), mostrar\_planes(), etc.

# Programación Funcional

Se incorporan elementos funcionales que mejoran la legibilidad y expresividad del código.  
- Uso de filter() para filtrar listas.  
- Lambdas para acciones rápidas en eventos y filtros.

# Programación Concurrente

La concurrencia permite ejecutar procesos en segundo plano sin bloquear la interfaz gráfica.  
- Uso de threading para iniciar monitoreo continuo sin bloquear UI.  
- time.sleep() controla la frecuencia de verificación de eventos.

# Programación Basada en Eventos

Este paradigma se evidencia en la interacción usuario-sistema.  
- Cada botón de la UI lanza un método específico.  
- Formularios y ventanas emergentes manejan eventos para editar, crear y eliminar tareas.

# Conclusión

Este proyecto refleja una implementación completa y funcional de varios paradigmas de programación. La combinación estructurada, orientada a objetos, funcional, concurrente y basada en eventos ha dado lugar a una aplicación sólida, interactiva y mantenible, con potencial de crecimiento futuro.