Integrantes del equipo:

Product Owner (PO): Juan Pablo Padilla Ramírez - A00574398 Líder Técnico (TL): Daniel Santino Alejandri Cure - A00573882 Desarrollador UI: David Alejandro Flores Cruz - A00573996 Desarrollador de Datos/API: Rodrigo Otero Juárez - A00573780

QA/Analista: Alfredo de Alba Ulloa - A01541178

Resumen sesión 2

Problema Central: Los jóvenes y adultos con interés en mejorar su situación económica carecen de herramientas accesibles para administrar y planificar sus finanzas personales, lo que limita su capacidad de ahorro y estabilidad económica.

ODS Seleccionado: ODS 8: Trabajo decente y crecimiento económico.

Población Objetivo: Jóvenes y personas interesadas en mejorar su situación financiera.

Criterios de Éxito del Prototipo:

- -Registrar ingresos y gastos de forma sencilla.
- -Mostrar al menos un reporte visual (gráfica) de ingresos vs. gastos.
- -Clasificar gastos en categorías básicas.
- -Guardar y recuperar la información del usuario de forma persistente.
- -Generar una recomendación básica de ahorro.

Actividad 1: Lluvia de ideas

App de Registro de Gastos Simplificada: Una aplicación minimalista enfocada en la rapidez y facilidad para que el usuario anote sus ingresos y gastos diarios. El objetivo es volverlo muy fácil de registrar para fomentar el hábito del registro.

Asistente de Presupuestos Basado en Metas (Regla 50/30/20): Una herramienta que, a partir del ingreso mensual del usuario, automáticamente propone un presupuesto distribuido en necesidades (50%), deseos (30%) y ahorros (20%). El usuario luego clasifica sus gastos y la app le indica cómo va su progreso respecto al plan.

Analizador de Finanzas con recompensa: Una aplicación que convierte el ahorro en un juego. El usuario gana puntos, medallas o desbloquea logros por cumplir sus metas de ahorro, mantener rachas de registro diario o reducir gastos en ciertas categorías. El objetivo es aumentar la motivación a través de recompensas.

Planificador Financiero con Proyecciones a Futuro: Una herramienta que no solo registra el pasado, sino que permite al usuario fijar metas a largo plazo como por ejemplo ahorrar para el enganche de un coche y simula cómo sus hábitos de gasto actuales impactarán el tiempo necesario para alcanzar dichas metas.

Técnica MoSCoW:

Debe tener:

Registrar ingresos: El usuario debe poder añadir una entrada de dinero con monto y fecha.

Registrar gastos: El usuario debe poder añadir una salida de dinero con monto, fecha y categoría.

Almacenamiento persistente: Los datos de ingresos y gastos deben guardarse en un archivo (ej. CSV, JSON) para que no se pierdan al cerrar la app.

Cálculo y visualización del balance: La app debe mostrar en la pantalla principal el resultado de (Total Ingresos - Total Gastos).

Reporte visual de gastos: Debe tener algún apoyo visual para que el usuario pueda ver representadas sus finanzas.

Debería tener:

Editar/Eliminar transacciones: El usuario debería poder corregir errores o eliminar registros incorrectos.

Generar recomendación de ahorro: La app debería mostrar un mensaje simple, como "Estás destinando un X% de tus ingresos al ahorro, con esto puedes llegar a la meta".

Ver historial de transacciones: Debería haber una vista o tabla donde se listen todos los registros guardados.

Podría tener:

Personalizar categorías: El usuario podría tener la opción de añadir, editar o eliminar las categorías de gastos.

Filtrar historial por fechas: El usuario podría seleccionar un rango de fechas para ver sólo las transacciones y reportes de ese período.

Exportar datos a CSV: Podría existir un botón para guardar un reporte de las transacciones en un archivo CSV.

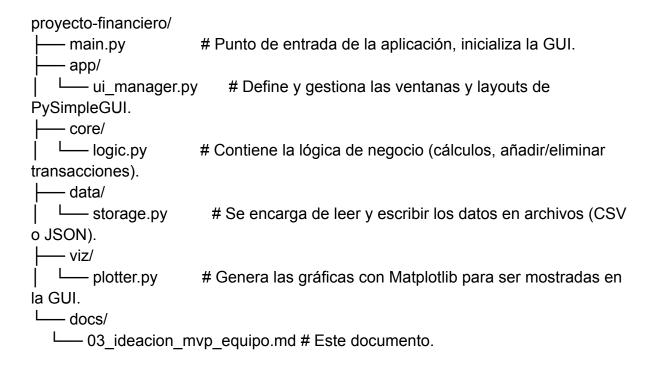
No va a tener:

Sincronización en la nube: El prototipo no se conectará a internet ni guardará datos en la nube.

Conexión con APIs bancarias: No se conectará a bancos para obtener transacciones automáticamente. Toda la entrada es manual.

Múltiples perfiles de usuario: La aplicación estará diseñada para un único usuario por instalación.

Arquitectura Mínima (Módulos del Proyecto):



main.py: Orquesta la aplicación. Inicia el bucle principal de la interfaz de usuario y llama a los otros módulos cuando es necesario (ej. al hacer clic en un botón).

app/ui_manager.py: Responsable exclusivo de la presentación. Contiene las funciones que crean la ventana principal, las ventanas modales de ingreso/gasto y actualizan los elementos visuales.

core/logic.py: El cerebro de la aplicación. Realiza operaciones como calcular el balance total, procesar los datos para las gráficas y gestionar la lista de transacciones en memoria. No depende de PySimpleGUI.

data/storage.py: El sistema de persistencia. Provee funciones como guardar_datos(lista_transacciones) y cargar_datos() que interactúan con el disco duro.

viz/plotter.py: Módulo especializado en visualización. Recibe datos procesados por el core y devuelve un objeto de gráfica (figura de Matplotlib) que el app puede mostrar.