

# Sistema de Gestión Financiera Personal

Proyecto Final - Fundamentos de Programación

**Castro-Palomino, María de las Nieves; Espinoza-Gobea, Oscar Martin; Hoces-Azañero, Reynaldo Emilio; Montalvo-García, Antony Abel; Quispe-Vera, Rudi Nelson; y Vicente-Puicon, Jean Pierre**

*Universidad Autónoma del Perú  
Facultad de Arquitectura e Ingeniería  
Escuela Profesional de Ingeniería de Software  
Ciclo I - 2025*

2025

# Índice

- 1 Introducción**
- 2 Marco teórico**
  - Definición de términos básicos
- 3 Diseño del sistema**
- 4 Implementación**
- 5 Metodología**
- 6 Resultados y Demostración**
- 7 Conclusiones**
- 8 Recomendaciones**

## Introducción

# Introducción



## 🎓 Contexto académico

- Proyecto del curso **Fundamentos de Programación**.
- Aplicación de programación estructurada y modelo EPS.
- Uso de PSeint y Python como herramientas.
- Integración de gestión financiera personal.

## ◎ Objetivo principal

- Diseñar e implementar un sistema de gestión financiera.
- Registrar ingresos (fijos/variables) y gastos clasificados.
- Aplicar la regla 50-30-20 para recomendaciones.
- Promover buenas prácticas financieras y de programación.

## 💡 Relevancia del proyecto

Aborda la problemática de desorganización financiera personal mediante una solución computacional básica pero funcional. Demuestra cómo algoritmos sencillos pueden apoyar la toma de decisiones financieras informadas, integrando conceptos de ingeniería de software y educación financiera.

## Marco teórico

# Fundamentos teóricos

## ◁▷ Programación Estructurada

- Paradigma basado en secuencia, selección y repetición.
- Código claro, legible y mantenible.
- Minimiza errores lógicos.
- Promueve buenas prácticas desde el inicio.

## ↳ Modelo EPS

- **Entrada:** Datos del usuario.
- **Proceso:** Cálculos y transformaciones.
- **Salida:** Resultados y recomendaciones.
- Organización clara del flujo del programa.

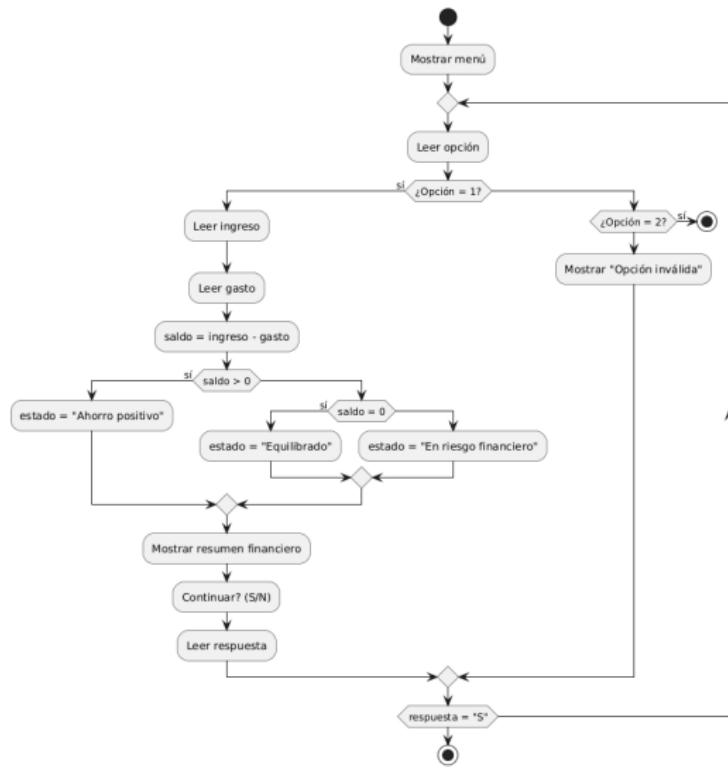
## ⌚ Regla 50-30-20

- Modelo de planificación financiera personal.
- **50 %** Gastos necesarios.
- **30 %** Gastos discretionariales.
- **20 %** Ahorro e inversión.
- Referencia para evaluar distribución financiera.

## 🕒 Calidad del Código

- Legibilidad y documentación adecuada.
- Nombres descriptivos de variables.
- Comentarios explicativos.
- Modularidad y reutilización.

# Fundamentos teóricos



# Definición de términos básicos

## Conceptos fundamentales

- **Algoritmo:** Conjunto ordenado de pasos para resolver un problema.
- **Variable:** Espacio en memoria con valor modificable.
- **Constante:** Valor fijo durante la ejecución.
- **Tipo de dato:** Clasificación del valor almacenado.

## Estructuras de control

- **Secuencial:** Ejecución ordenada.
- **Condicional (if, else):** Toma de decisiones.
- **Repetitiva (for, while):** Iteraciones controladas.
- **Recursividad:** Función que se llama a sí misma.

## Modularidad

- **Función:** Bloque de código reutilizable con retorno.
- **Procedimiento:** Bloque sin retorno explícito.
- **Módulo:** Archivo que agrupa funciones/variables.
- **Librería:** Conjunto de módulos predefinidos.

## Finanzas personales

- **Ingresos fijos/variables:** Fuentes de dinero.
- **Gastos necesarios:** Necesidades básicas.
- **Gastos discretionales:** Ocio y entretenimiento.
- **Ahorro:** Reserva para futuro.

## Diseño del sistema

# Arquitectura del sistema - Modelo EPS



## → Entrada

- Ingresos fijos y variables
- Gastos clasificados
- Monto de ahorro
- Validación de datos

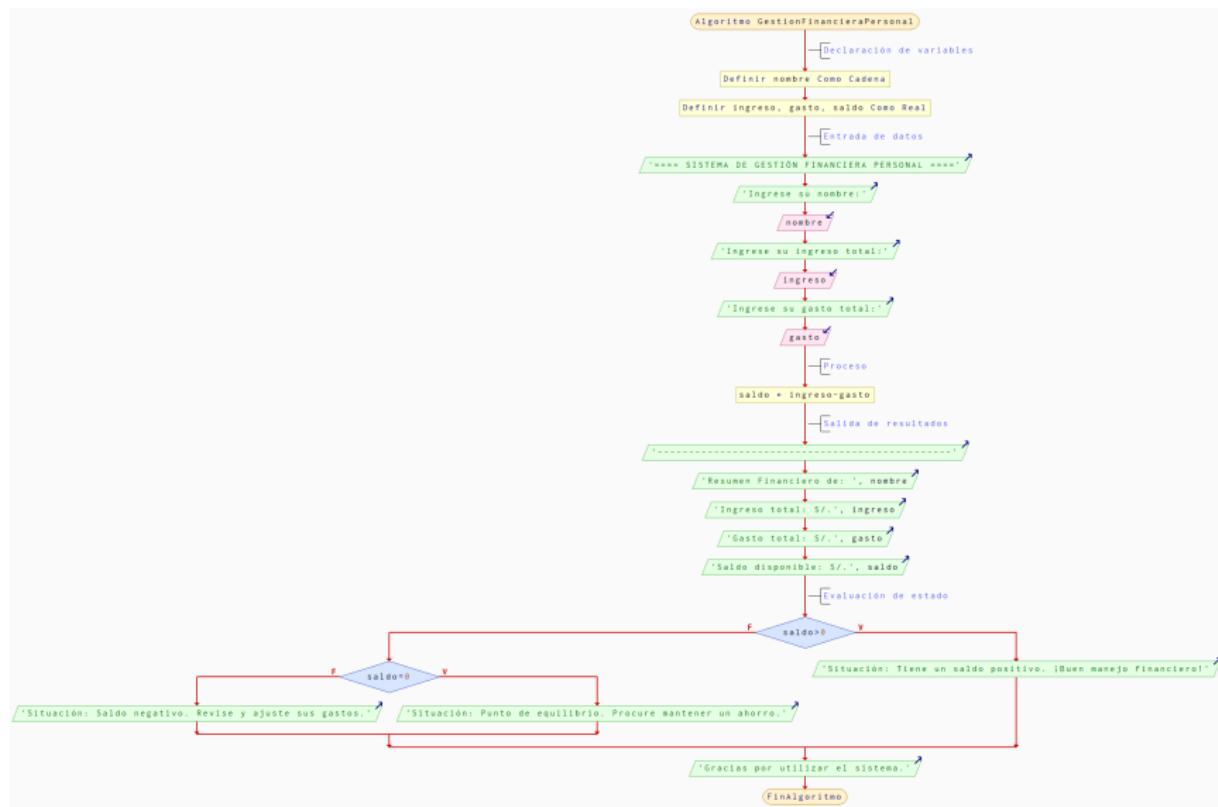
## ⚙️ Proceso

- Cálculo de totales
- Clasificación por categorías
- Cálculo de porcentajes
- Evaluación vs regla 50-30-20

## 🖨️ Salida

- Resumen financiero
- Distribución porcentual
- Recomendaciones
- Gráficos y reportes

# Diagrama de flujo del sistema



# Diagrama de flujo del sistema

## ► Inicio y Entrada

- Declaración de variables
- Captura de datos financieros
- Clasificación de gastos
- Validación básica

## ▣ Proceso y Salida

- Cálculos aritméticos
- Evaluación vs 50-30-20
- Generación de reportes
- Recomendaciones personalizadas

## Implementación

# Estructuras de control utilizadas

## ⌚ Estructuras Condicionales

**if, elif, else para:**

- Evaluar balance financiero
- Comparar con regla 50-30-20
- Generar recomendaciones
- Validar entradas del usuario

## ⟳ Estructuras Repetitivas

**while para:**

- Menú principal interactivo
- Validación de entradas inválidas
- Ejecución continua del sistema

**for para:**

- Recorrer categorías de gastos
- Procesar listas de datos

## ⊕ Modularidad

**Funciones con retorno:**

- Cálculo de totales
- Conversión de porcentajes
- Evaluación financiera

**Procedimientos:**

- Mostrar menú principal
- Presentar resultados
- Generar reportes

**Librerías utilizadas:**

- `datetime`: Fechas
- `matplotlib`: Gráficos
- `reportlab`: PDFs

## ∞ Recursividad

**Aplicación didáctica:**

- Cálculo de ahorro acumulado
- Proyecciones financieras
- Validación recursiva de datos

# Pseudocódigo Implementado (PSelnt) - Parte 1

## Sección Principal del Algoritmo

```
1 Proceso Sistema_Gestion_Financiera_Personal
2 // =====
3 // DECLARACIÓN DE VARIABLES
4 // =====
5 Definir ingreso_fijo, ingreso_variable, ingreso_total Como Real
6 Definir gastos_necesarios, gastos_discrecionales Como Real
7 Definir ahorro_actual Como Real
8 Definir porcentaje_necesarios, porcentaje_discrecionales, porcentaje_ahorro Como Real
9
10 // =====
11 // ENTRADA DE DATOS
12 // =====
13 Escribir "___ SISTEMA DE GESTIÓN FINANCIERA PERSONAL ___"
14 Escribir "Ingrese su ingreso fijo mensual:"
15 Leer ingreso_fijo
16 Escribir "Ingrese su ingreso variable mensual:"
17 Leer ingreso_variable
18 ingreso_total ← ingreso_fijo + ingreso_variable
19
20 // Entrada de gastos necesarios y discretionales...
```

# Pseudocódigo Implementado (PSelnt) - Parte 2

## Sección Principal del Algoritmo (continuación)

```
22 // Cálculos de totales...
23
24 // =====
25 // PROCESO
26 // =====
27 porcentaje_necesarios ← (gastos_necesarios / ingreso_total) * 100
28 porcentaje_discrecionales ← (gastos_discrecionales / ingreso_total) * 100
29 porcentaje_ahorro ← (ahorro_actual / ingreso_total) * 100
30
31 // =====
32 // SALIDA DE RESULTADOS
33 // =====
34 Escribir "==== RESUMEN FINANCIERO MENSUAL ==="
35 Escribir "Ingreso total: S/ ", ingreso_total
36 Escribir "Gastos necesarios: S/ ", gastos_necesarios, " (", porcentaje_necesarios, "%)"
37 Escribir "Gastos discretionales: S/ ", gastos_discrecionales, " (",
porcentaje_discrecionales, "%)"
38 Escribir "Ahorro mensual: S/ ", ahorro_actual, " (", porcentaje_ahorro, "%)"
39
40 // Evaluación según regla 50-30-20...
41 FinProceso
42
```

# Características Técnicas del Sistema

Característica	Implementación	Beneficio	Estado
❖ Pseudocódigo	PSelnt compatible	Ejecutable en diversas versiones, pedagógico	✓
❖ Python	Notebook Jupyter	Interactivo, con gráficos y reportes PDF	✓
❖ Visualización	Matplotlib	Gráficos de barras y circulares comparativos	✓
❖ Reportes	ReportLab	Generación automática de PDFs profesionales	✓
❖ Persistencia	Archivos CSV	Almacenamiento de historial financiero	⌚
❖ Interfaz	Gráfica (Tkinter)	Mejor experiencia de usuario	★

## 🔒 Validaciones

- Entradas numéricas no negativas
- Control de opciones del menú
- Prevención de división por cero
- Mensajes de error descriptivos

## ❖ Compatibilidad

- Python 3.7+
- PSelnt 2023+
- Multiplataforma
- Sin dependencias externas críticas

## Metodología

# Metodología de desarrollo

## Q Análisis y Diseño

- Estudio de modelos financieros (50-30-20)
- Revisión bibliográfica de programación estructurada
- Diseño del modelo EPS
- Definición de variables y estructuras

## Metodología Ágil

- Enfoque iterativo e incremental
- Inspirado en Scrum y XP
- Reuniones de seguimiento semanales
- Retroalimentación continua

## </> Implementación

- Desarrollo incremental
- Prototipado en PSeInt
- Migración a Python
- Integración de módulos

## Pruebas y Validación

- Casos de prueba representativos
- Validación de cálculos financieros
- Pruebas de usabilidad
- Corrección de errores iterativa

# Metodología de desarrollo

🕒 Cronograma de Actividades del Proyecto

Actividad	S1	S2	S3	S4	S5	S6	S7	S8
🔍 Análisis del problema y objetivos	✓	✓						
📖 Revisión bibliográfica	✓	✓	✓					
➡️ Diseño del sistema			✓	✓				
➡️ Diseño de diagramas (EPS, flujo)				✓	✓			
⚡ Implementación en Python					✓	✓		
📝 Pruebas y ajustes						✓	✓	
📄 Documentación del informe						✓	✓	
✉️ Revisión final y entrega							✓	🚩

## 👥 Metodología Ágil

- Desarrollo iterativo e incremental
- Inspirado en Scrum y XP
- Reuniones semanales de seguimiento
- Retroalimentación continua

## 🕒 Seguimiento

- Actas de reunión semanales
- Control de versiones (Git)
- Tablero Kanban digital
- Revisiones por pares

## Resultados y Demostración

# Evidencia de Ejecución

=====  
SISTEMA DE GESTIÓN FINANCIERA PERSONAL  
Basado en la Regla 50-30-20  
=====

PASO 1: INGRESOS MENSUALES

Ingreso total: S/. 3,000.00

--- PASO 2: GASTOS NECESARIOS (50% recomendado) ---

Incluye: vivienda, alimentación, servicios, transporte, educación, salud

--- VIVIENDA ---

Si vive en casa propia, registre solo gastos de mantenimiento.  
Si alquila, registre el monto del alquiler.

--- OTROS GASTOS NECESARIOS ---

Total gastos necesarios: S/ 1,370.00

--- PASO 3: GASTOS DISCRECIONALES (30% recomendado) ---

...  
Le falta ahorrar S/ 600.00 para cumplir la meta  
Intente incrementar su ahorro mensual gradualmente

Generando gráficos...

*Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output settings...*

# Reporte PDF Generado

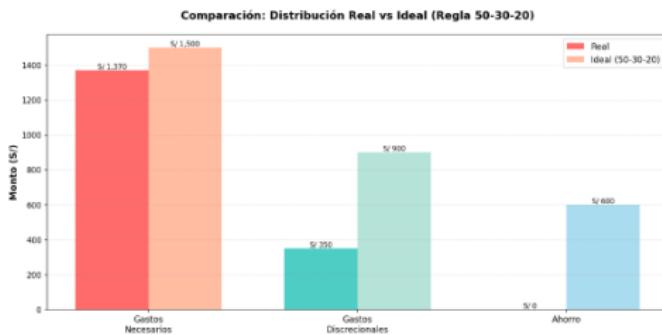
## Reporte de Gestión Financiera Personal

Fecha de elaboración: 19/12/2025 01:11

### Resumen Financiero Mensual

Concepto	Monto (S/.)	% del Ingreso
Ingreso total	3,000.00	100%
Gastos necesarios	1,370.00	45.7%
Gastos discrecionales	350.00	11.7%
Ahorro	0.00	0.0%
Balance	1,280.00	—

### Visualización de la Distribución Financiera



## Conclusiones

# Conclusiones

## ✓ Logros Técnicos

- Implementación exitosa del modelo EPS
- Uso adecuado de estructuras de control
- Código modular y documentado
- Integración de múltiples tecnologías
- Sistema funcional y educativo

## 🎓 Valor Formativo

- Aplicación de conceptos teóricos en práctica
- Desarrollo de pensamiento algorítmico
- Mejora de habilidades de programación
- Comprensión de gestión financiera básica

## 💡 Aprendizajes Obtenidos

- Importancia de la planificación en desarrollo de software
- Valor de la documentación y buenas prácticas
- Utilidad de la programación para resolver problemas reales
- Necesidad de validación y manejo de errores

## ⌚ Proyecciones Futuras

- Interfaz gráfica de usuario
- Base de datos para historial
- Análisis de tendencias financieras
- Versión web/móvil
- Integración con APIs financieras

## Recomendaciones

# Recomendaciones

## Para Usuarios

- Registrar ingresos y gastos periódicamente
- Aplicar gradualmente la regla 50-30-20
- Revisar reportes mensuales para ajustes
- Priorizar el hábito de ahorro constante
- Utilizar el sistema como herramienta educativa

## Para Desarrollo

- Mantener código documentado y legible
- Implementar más validaciones de datos
- Agregar exportación a formatos adicionales
- Incluir análisis de tendencias históricas
- Mejorar manejo de excepciones

## Para Docentes

- Utilizar el proyecto como ejemplo práctico
- Enfatizar la relación teoría-práctica
- Fomentar buenas prácticas de programación
- Integrar conceptos de finanzas personales
- Promover trabajo colaborativo en equipo

## Para la Institución

- Incorporar proyectos similares en el currículo
- Fomentar interdisciplinariedad (software + finanzas)
- Proveer herramientas actualizadas para desarrollo
- Organizar concursos de proyectos prácticos
- Establecer vínculos con el sector financiero

# Referencias I

- [1] R. S. Pressman y B. R. Maxim, **Software Engineering: A Practitioner's Approach**. New York, NY: McGraw-Hill Education, 2014.
- [2] I. Sommerville, **Software Engineering**. Harlow, UK: Pearson Education, 2015.
- [3] S. McConnell, **Code Complete**. Redmond, WA: Microsoft Press, 2004.
- [4] R. C. Martin, **Clean Code: A Handbook of Agile Software Craftsmanship**. Upper Saddle River, NJ: Prentice Hall, 2008.
- [5] BBVA, «La regla del ahorro 50-30-20,» [En línea]. dirección: <https://www.bbva.es/finanzas-vistazo/ef/cuentas/regla-ahorro-50-30-20.html>
- [6] S. Perú, «Cómo organizar tus finanzas personales con la regla 50-30-20,» [En línea]. dirección: <https://www.scotiabank.com.pe/blog/organizar-finanzas-personales-regla-50-30-20>
- [7] E. Gamma, R. Helm, R. Johnson y J. Vlissides, **Design Patterns: Elements of Reusable Object-Oriented Software**. Boston, MA: Addison-Wesley, 1994.
- [8] D. E. Knuth, **The Art of Computer Programming, Vol. 1: Fundamental Algorithms**. Reading, MA: Addison-Wesley, 1997.
- [9] L. J. Aguilar, «Fundamentos de Programación: Algoritmos, estructura de datos y objetos,» [En línea]. dirección: <https://openlibra.com/es/book/fundamentos-de-programacion>

## Referencias II

- [10] E. L. de Python, «Módulos en Python,» [En línea]. dirección: <https://ellibrodepython.com/modulos-python>
- [11] R. G. Duque, «Python para todos,» [En línea]. dirección: <https://persoal.citius.usc.es/eva.cernadas/informaticaparacientificos/material/libros/Python%20para%20todos.pdf>
- [12] U. O. de Catalunya, «Introducción a la ingeniería del software,» [En línea]. dirección: <https://openaccess.uoc.edu/bitstream/10609/147761/1/Introducci%C3%B3n%20a%20la%20ingenier%C3%ADA%20del%20software.pdf>
- [13] E. Warren y A. W. Tyagi, **All Your Worth: The Ultimate Lifetime Money Plan.** New York, NY: Free Press, 2005.
- [14] K. Schwaber y J. Sutherland, «The Scrum Guide,» [En línea]. dirección: <https://scrumguides.org>
- [15] K. Beck, **Extreme Programming Explained: Embrace Change.** Boston, MA: Addison-Wesley, 2000.

# ¡Gracias por su atención!

🗣 ¿Preguntas?

**Contacto:**  
Equipo de Desarrollo - Gestión  
Financiera Personal  
Fundamentos de Programación - Ciclo I  
2025  
Universidad Autónoma del Perú