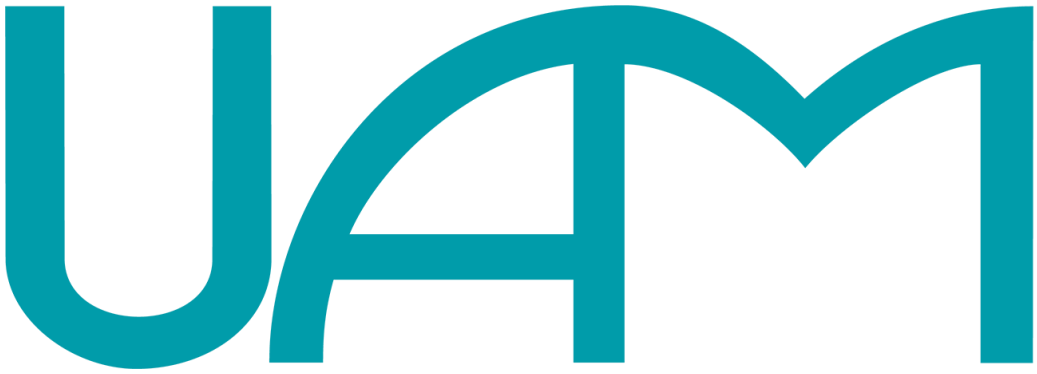


Universidad Americana

Facultad de Ingeniería y Arquitectura

Introducción a la Programación

Gestor de Pagos Personales y Empresariales —- PayControl

Autores:

Kate Valentina Ramírez Urbina

Mery Nohemy López Aguirre

Alex Josué Fonseca Velásquez

Oscar Isaac Duran Guadrón

Docente:

Lic. Silvia Gigdalia Ticay López

04 de Julio 2025

INTRODUCCIÓN

Hoy en día, tanto las personas como los pequeños negocios enfrentan dificultades para llevar un control ordenado de sus pagos. La falta de herramientas simples y accesibles puede causar olvidos, recargos por atraso o desorganización financiera. Esto se vuelve un problema tanto en el hogar como en el entorno empresarial, donde una mala gestión de pagos afecta directamente la toma de decisiones y la estabilidad económica.

Con el fin de resolver esta necesidad, se propuso desarrollar un programa en el lenguaje de Python que permita registrar, organizar y consultar pagos, ya sean personales o empresariales. La idea principal es facilitar al usuario el manejo de sus finanzas mediante un programa sencillo, que permita llevar control de fechas, montos y tipos de pagos, así como distinguir cuáles ya se han realizado y cuáles están pendientes.

El desarrollo del proyecto se llevó a cabo de forma progresiva. Primero, se analizó el problema y se identificaron los elementos esenciales que debía tener el sistema. A partir de ese análisis, se elaboró un algoritmo, Posteriormente, se implementó el sistema utilizando el lenguaje Python, el cual fue elegido por su facilidad de uso, sintaxis clara y su capacidad para trabajar con estructuras de datos simples y archivos de texto. Se trabajó por módulos, separando cada parte del sistema para hacerlo más ordenado y funcional: manejo de archivos, gestión de usuarios, pagos y navegación por menús.

El proyecto se desarrolló en el marco de la asignatura Introducción a la Programación y permitió aplicar conocimientos fundamentales como el diseño algorítmico, el uso de estructuras de control, la organización modular del código y la persistencia de datos.

Este programa busca ser una herramienta útil para mejorar la organización financiera en la vida personal o empresarial. Al mismo tiempo, representa una experiencia significativa que conecta la teoría con la práctica, y demuestra que la tecnología puede estar al alcance de todos para facilitar la vida diaria.

OBJETIVOS

i. Objetivo general

Diseñar e implementar un programa en Python que permita registrar, gestionar y consultar pagos personales o empresariales de forma eficiente, con el fin de promover el orden financiero.

ii. Objetivos específicos

1. Identificar las necesidades básicas en la gestión de pagos personales o empresariales.
2. Analizar los requerimientos funcionales de un sistema digital de control financiero.
3. Desarrollar un algoritmo que optimice el procesamiento de datos relacionados con pagos personales o empresariales.
4. Implementar el algoritmo en Python utilizando estructuras adecuadas.

JUSTIFICACIÓN

En la actualidad, tanto individuos como pequeños emprendimientos enfrentan desafíos constantes en la gestión de sus finanzas. La falta de herramientas accesibles, simples y adaptadas a su realidad provoca desorganización, pérdidas económicas por recargos o atrasos en los pagos, y una falta de control general sobre sus obligaciones financieras. Esta situación se agrava cuando se carece de conocimientos técnicos o acceso a sistemas sofisticados.

Frente a esta necesidad, surge la idea de desarrollar un programa en Python que permita gestionar de manera básica y funcional los pagos personales o empresariales. Este proyecto se justifica desde dos perspectivas: la educativa y la social/práctica. En el plano educativo, permite a los estudiantes aplicar sus primeros conocimientos en programación, desde el diseño algorítmico hasta la implementación y documentación de un sistema real. En el ámbito práctico, representa una herramienta útil para la organización financiera, al alcance de personas con pocos recursos tecnológicos.

Además, esta iniciativa promueve el desarrollo del pensamiento lógico, la resolución de problemas y el uso de la tecnología como medio para dar respuesta a necesidades del entorno inmediato. De esta manera, se conecta el aprendizaje en el aula con la vida diaria, fortaleciendo no solo las habilidades técnicas, sino también el compromiso con el desarrollo de soluciones accesibles y contextualizadas.

DEFINICIÓN Y ALCANCE DEL CASO DE ESTUDIO

Este caso de estudio consiste en desarrollar un programa computacional que permita a los usuarios registrar sus pagos financieros, categorizarlos según tipo, monto y fecha, y consultar su historial de pagos. El objetivo es brindar una herramienta básica que contribuya a la planificación y organización económica.

iii. Alcance:

•Registro de pagos con fecha, tipo y monto.

•Clasificación de pagos como realizados o pendientes.

•Consulta del historial de pagos.

•Uso de archivos de texto plano para guardar los datos.

•Interfaz basada en línea de comandos.

•No se incluye interfaz gráfica ni integración con bases de datos externas.

ACTIVIDADES DE LA PRÁCTICA DE LA FAMILIARIZACIÓN

iv. Descripción del problema o necesidad

Para comenzar, se observaron diversas necesidades en el entorno cotidiano de las personas. Como grupo, una situación nos llamó especialmente la atención: la dificultad que enfrentan las personas y las pequeñas empresas para gestionar sus pagos de forma organizada, sin olvidar fechas ni confundir montos durante el proceso de realizar un pago.

En la vida diaria, tanto individuos como microempresarios suelen experimentar problemas relacionados con el manejo de sus finanzas, ya sea por falta de organización, herramientas inadecuadas o simples olvidos. Esta situación puede provocar recargos, retrasos e incluso pérdida de información importante.

A partir de esta problemática, se detectó la necesidad de una herramienta digital simple y accesible que permita registrar pagos, visualizar su historial y distinguir entre los pagos realizados y los pendientes. Esta solución busca apoyar a personas y empresas en la gestión eficiente de sus compromisos financieros.

v. Análisis del problema

Para empezar, se detectó la necesidad de mejorar las gestiones de pagos financieros, se sugirió analizar a las personas y empresas de cómo resuelven estos problemas. Se valoraron las diferentes soluciones existentes como aplicaciones móviles, hojas de cálculo, métodos manuales, concluyendo que muchas son complejas o inaccesibles para quienes no tienen formación técnica.

Aparte, se identificó que diferentes sistemas de este tipo requieren un costo, lo cual la mayoría de las personas limita sus posibilidades de encontrar buenas soluciones simples y funcionales.

Se definieron los requerimientos básicos: facilidad de uso, almacenamiento local y operación en terminal. Estos criterios permitirán que cualquiera de los usuarios pueda utilizar el sistema sin tener conocimiento avanzado y de forma eficiente.

Como solución se definieron funciones prioritarias para responder a las necesidades detectadas como: añadir pago, listar estos pagos por estado, consultar todos los pagos y guardar la información en un archivo. Estas decisiones se tomaron con el compromiso de una solución accesible, práctica y funcional para todos orientadas a usuarios y empresas reales con necesidades cotidianas.

Diseño del algoritmo

Se elaboró un algoritmo paso a paso que organiza el funcionamiento del programa. Se diseñó una estructura con funciones para:

1. Ingresar datos del pago.
2. Consultar pagos registrados.
3. Guardar y cargar datos de un archivo.
4. Menú interactivo para el usuario.

Como resultado se obtuvo un pseudocódigo claro y modular que facilita la posterior implementación del programa.

vi. Lista de requerimientos:

vii. Requerimientos Funcionales:

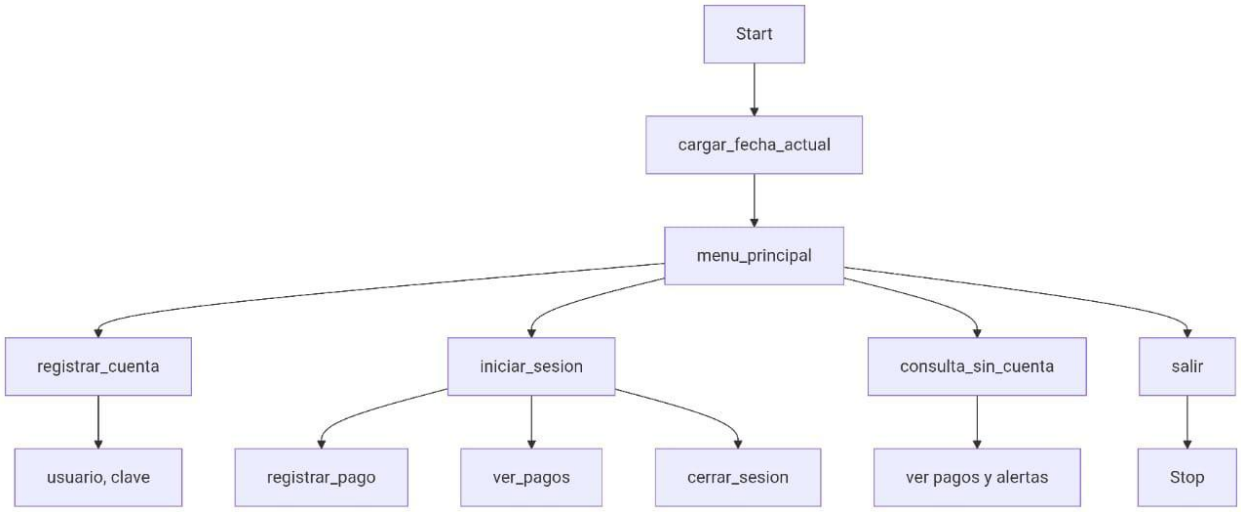
1. Cargar la fecha actual al iniciar el sistema.
2. Mostrar un menú principal con varias opciones.
3. Permitir al usuario registrar una cuenta proporcionando usuario y clave.
4. Permitir al usuario iniciar sesión con sus credenciales.
5. Una vez iniciada la sesión, permitir: Registrar pagos
6. Ver pagos realizados
7. Cerrar sesión
8. Permitir consultar pagos y alertas sin necesidad de cuenta.

9. Opción de salir del sistema.

i. Requerimiento no Funcionales

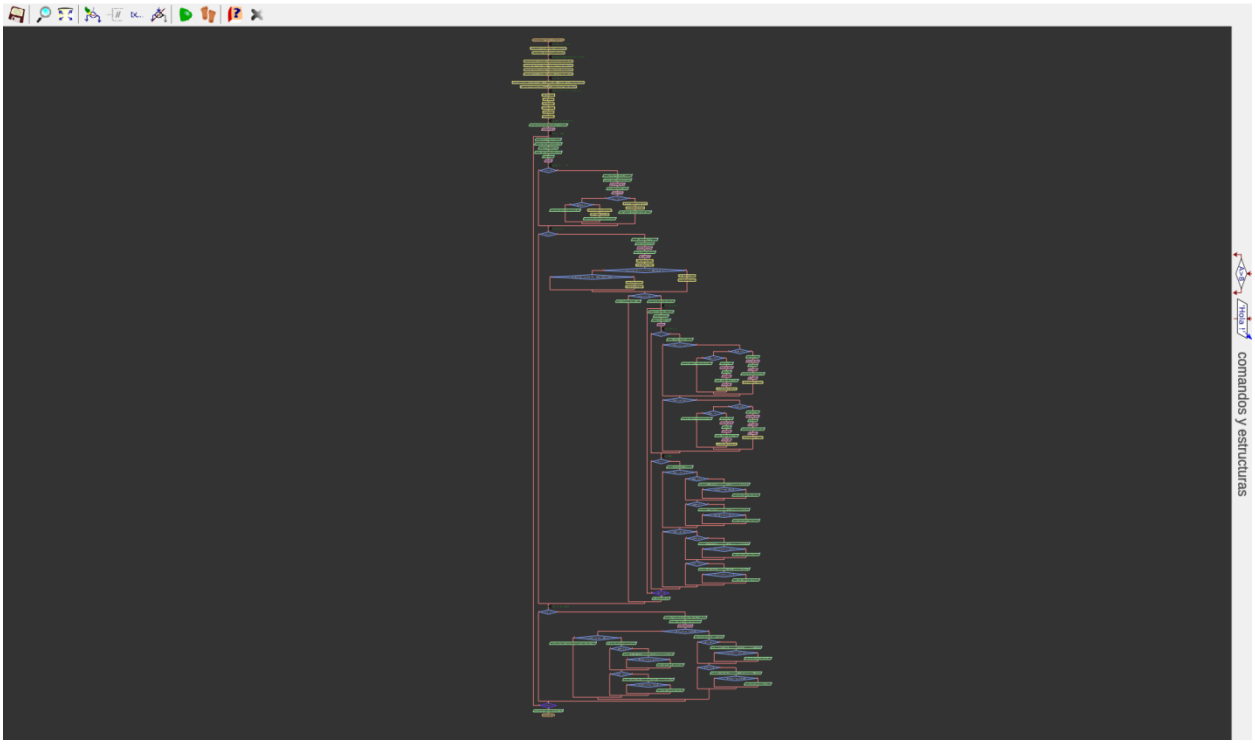
1. Los datos deben almacenarse de forma persistente, preferiblemente en archivos de texto para facilitar su manejo.
2. La estructura de los archivos debe ser clara y compresible para que los usuarios y desarrolladores puedan interpretarla fácilmente.
3. La interfaz del sistema debe ser sencilla y textual para facilitar su uso por parte de cualquier usuario.
4. El sistema debe ser compatible con múltiples dispositivos y ejecutarse correctamente usando Python.
5. El sistema debe proteger la información del usuario, como claves y daros personales, mediante un manejo seguro.
6. El sistema debe ser fácil de mantener y actualizar sin afectar los datos anteriores ni las funcionalidades existentes.

ii. Diagrama de Estructura.




```
goritmo SistemaDePagosMultiples

  // Variables de usuario
  Definir usuario1, usuario2 Como Cadena
  Definir clave1, clave2 Como Cadena

  // Variables de pagos para dos usuarios
  Definir proveedor1u1, monto1u1, fecha1u1 Como Cadena
  Definir proveedor2u1, monto2u1, fecha2u1 Como Cadena

  Definir proveedor1u2, monto1u2, fecha1u2 Como Cadena
  Definir proveedor2u2, monto2u2, fecha2u2 Como Cadena

  // Variables de control
  Definir opcion, sesion_activa, indice_usuario, pagos_u1, pagos_u2 Como Entero
  Definir usuario_actual, contraseña, fecha_actual Como Cadena

  // Inicializar usuarios y contadores
  usuario1 ← ""; clave1 ← ""; pagos_u1 ← 0
  usuario2 ← ""; clave2 ← ""; pagos_u2 ← 0

  // Ingresar la fecha actual
  Escribir "Ingrese la fecha de hoy (dd/mm/aaaa):"
  Leer fecha_actual

  // Menú principal
  Repetir
      Escribir "=== SISTEMA DE PAGOS ==="
      Escribir "1. Registrar nueva cuenta"
      Escribir "2. Iniciar sesión"
      Escribir "3. Ver pagos (sin cuenta)"
      Escribir "4. Salir"
      Leer opcion

      // Registro de cuenta
      Si opcion = 1 Entonces
          Escribir "--- REGISTRO DE USUARIO ---"
          Escribir "Ingrese nombre de usuario:"
          Leer usuario_actual
          Escribir "Ingrese contraseña:"
          Leer contraseña

          Si usuario1 = "" Entonces
              usuario1 ← usuario_actual
```

```
PSeInt - Ejecutando proceso SISTEMADEPAGOSMULTIPLES
*** Ejecución Iniciada. ***
Ingrese la fecha de hoy (dd/mm/aaaa):
>
línea 24 instrucción 1
```



iii. Codificación, ejecución, verificación y depuración.

La codificación se llevó a cabo utilizando algunas herramientas de programación como el lenguaje python el cual es ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning.

un sistema de gestión de pagos personales en Python, asegurando que tenga una estructura bien definida y que se ejecute en el entorno de Visual Studio Code.

Herramientas utilizadas:

Lenguaje: Python.

IDE: Visual Studio Code.

Archivos de almacenamiento:

Archivos.	Funcionalidades principales.
Archivos.py	Manejo de archivos: carga y guardado de usuarios y pagos
Auth.py	Registro e inicio de sesión de usuarios
Pagos.py	Registro de pagos y visualización de pendientes
Main.py	Punto de entrada del programa y menú inicial de navegación

Algunos de los procesos de Desarrollo son:

Codificación: Se dividió el proyecto en módulos lógicos, desarrollados por separado para facilitar pruebas y mantenimiento.

Ejecución: Se ejecutó el archivo main.py desde Visual Studio Code, enlazando los subprogramas correctamente.

Verificación: Se ingresaron usuarios y pagos de prueba para comprobar el flujo correcto del sistema.

Depuración: Se realizaron pruebas con entradas incorrectas y revisión manual para capturar errores lógicos o de formato.

iv. Documentación del proyecto

Una vez implementado y verificado el sistema de gestión de pagos, se procedió a la elaboración de la documentación técnica y funcional del proyecto. Esta documentación tiene como propósito dejar constancia del proceso seguido, describir las funcionalidades del programa, su estructura modular y la forma adecuada de uso por parte del usuario. Se detallaron los módulos principales, sus funciones específicas, el flujo de interacción del sistema y las instrucciones básicas para instalación y ejecución.

También se integraron buenas prácticas de programación, como el modularidad, el uso de comentarios en el código, nombres de variables claros y estructuras de control eficientes, siguiendo recomendaciones como las indicadas por Beazley & Jones (2013) y Sweigart (2019), y apoyándose en recursos como W3Schools y Real Python para el manejo adecuado de archivos.

Se generó archivos estructurados con los siguientes contenidos:

Archivos.py: Manejo de almacenamiento de datos.

Auth.py: Autenticación de usuarios.

Pagos.py: Registro y consulta de pagos.

Main.py: Navegación del sistema.

CONCLUSIÓN

El desarrollo del programa en el Lenguaje de Python logró cumplir satisfactoriamente con los objetivos planteados al inicio del proyecto. Se construyó una herramienta funcional que permite el registro, organización y consulta de pagos personales o empresariales, brindando una solución práctica para mejorar el control financiero en contextos donde frecuentemente se carece de herramientas digitales accesibles.

El sistema implementado cuenta con funcionalidades clave como el registro de pagos con fecha, monto y tipo, la clasificación de pagos realizados o pendientes, la consulta del historial completo y el almacenamiento de datos en archivos de texto. Estas características permiten un uso eficiente del programa sin requerir conocimientos técnicos avanzados ni conexión a bases de datos externas. La interfaz en línea de comandos resultó ser clara y adecuada para el propósito del sistema.

El programa fue diseñado de forma modular, lo que facilitó su implementación, prueba y mantenimiento. Cada parte del sistema, autenticación de usuarios, gestión de pagos, manejo de archivos y navegación. Fue desarrollada como un componente independiente, lo que aporta orden y escalabilidad. Durante las pruebas, el sistema respondió de forma correcta, mostrando fluidez en la ejecución de los distintos procesos, sin errores críticos.

Además, se documentó adecuadamente la estructura del programa, lo cual permite su fácil comprensión y futuras modificaciones. El sistema demostró ser una herramienta útil para apoyar la organización financiera tanto en el ámbito personal como en pequeños entornos empresariales.

El proyecto resultó funcional, estable y efectivo en su propósito. Representa una base sólida sobre la cual se pueden desarrollar futuras mejoras, como la inclusión de una interfaz gráfica, la integración con monedas extranjeras, o nuevas funciones como la edición y eliminación de pagos registrados. La solución propuesta responde de forma concreta a una necesidad real y aporta valor mediante su sencillez, utilidad y adaptabilidad.

RECOMENDACIONES

Para futuras mejoras del programa, se recomienda implementar una interfaz gráfica que facilite la interacción del usuario y haga el sistema más intuitivo. También sería conveniente agregar funciones para editar o eliminar pagos, lo cual permitiría una gestión más completa y flexible de la información. Incluir soporte para múltiples monedas, como córdobas o dólares, aumentaría su utilidad en distintos contextos económicos. Además, se sugiere permitir la clasificación de pagos por categorías personalizadas, así como la incorporación de filtros de búsqueda por fecha, tipo o monto, que faciliten la consulta de datos específicos. Otra mejora importante sería la integración de alertas o recordatorios automáticos para pagos pendientes, a través de notificaciones o correo electrónico. En cuanto al almacenamiento, se recomienda considerar el uso de bases de datos como SQLite para una gestión más robusta y escalable de la información. Finalmente, sería útil desarrollar versiones diferenciadas del programa para uso personal y empresarial, adaptando las funciones según las necesidades de cada tipo de usuario.

Enlaces de GitHUB

Oscar Isaac Duran Guadrón: https://github.com/OscarDU24/Trabajo-FInal-xd

Mery Nohemy López Aguirre: https://github.com/nohemy24/Trabajo-Final-de-INtro-A-la-prog.git

Kate Valentina Ramírez Urbina: https://github.com/kateV1013/Trabajo-Final.git

Alex Josué Fonseca Velásquez: https://github.com/alexfonseca1/Trabajo-final1.git

BIBLIOGRAFIA

Beazley, D. M., & Jones, B. K. (2013). Python Cookbook: Recipes for Mastering Python 3. O'Reilly Media. https://www.oreilly.com/library/view/python-cookbook/9781449340377/

Sweigart, A. (2019). Automate the Boring Stuff with Python: Practical Programming for Total Beginners. No Starch Press. https://automatetheboringstuff.com/

W3Schools. (s.f.). Python File Handling. https://www.w3schools.com/python/python_file_handling.asp

Real Python. (s.f.). Working with Files in Python. https://realpython.com/working-with-files-in-python/