Proyecto de clase de **Python aplicado**.

Fundamentos de Programación

**Nombre del Proyecto:   
Construye un sistema de recomendación de películas**

**1. Información General**

* **Nombre estudiantes:**  
  **1.**Juan Sebastián Toro Ramírez  
  **2.**Juan Manuel Patarroyo Forero  
  **3.**Juan David Aguilera Sastre  
  **4.**Tania Alejandra Rojas Naranjo
* **Curso / Grupo:**Fundamentos de Programación C
* **Fecha de entrega:**  
  16 de octubre del 2025
* **Profesor:**  
    
  Pablo Enrique Carreño Hernandez

**2. Título del Proyecto**

Sistema de recomendación de películas personalizado con Machine Learning.

**3. Descripción del Proyecto**

El proyecto consiste en desarrollar un sistema de recomendación que ayude a las personas a encontrar películas según sus gustos, solucionando el problema de exceso de opciones en las plataformas de streaming. El Proyecto está dirigido a las personas que disfrutan ver películas en su tiempo libre pero no logran hallar que películas visualizar por la cantidad excesiva de opciones cinematográficas. El resultado esperado es un modelo que sugiere películas personalizadas, mejorando la experiencia del usuario al momento de utilizar una plataforma de streaming.

**4. Objetivos**

**General:**

* Desarrollar un sistema de recomendación de películas personalizado que permita a las personas encontrar fácilmente contenidos cinematográficos que se acomoden a sus gustos, con el fin de mejorar su experiencia al utilizar plataformas de streaming y reducir el tiempo dedicado a la búsqueda de películas.

**Específicos:**

• Implementar estructuras de datos adecuadas para almacenar y gestionar la información de las películas y las preferencias de las personas

• Aplicar funciones y algoritmos de filtrado colaborativo o basado en contenido para generar recomendaciones personalizadas

• Utilizar técnicas de análisis de datos para identificar relaciones entre las preferencias de los usuarios y las características de cada película

• Desarrollar una interfaz sencilla e interactiva que permita al usuario ingresar sus gustos y visualizar las recomendaciones generadas por el sistema.

**5. Requisitos**

* Herramientas y tecnologías utilizadas (Python, librerías, etc.)

1. Lenguaje Python
2. Librerias:

* pandas → para manejar y limpiar los datos.
* numpy → para operaciones numéricas.
* ast → para convertir texto en estructuras de datos de Python.
* scikit-learn (CountVectorizer, cosine\_similarity) → para procesar el texto y calcular similitudes entre películas.
* streamlit → para crear la interfaz web interactiva.
* requests → para obtener las imágenes de los afiches desde la API de TMDB.
* pickle → para guardar y cargar los datos procesados.
* Requisitos de instalación o ejecución

1. Tener instalado **Python 3.8 o superior**.
2. Instalar las siguientes librerías (puede hacerse desde la terminal):

**pip install pandas numpy scikit-learn streamlit requests**

1. Tener los archivos tmdb\_5000\_movies.csv y tmdb\_5000\_credits.csv en la misma carpeta del proyecto.
2. Ejecutar el archivo principal del sistema (por ejemplo, app.py) con el siguiente comando:

streamlit run app.py

1. El sistema se abrirá en el navegador mostrando la interfaz del recomendador.

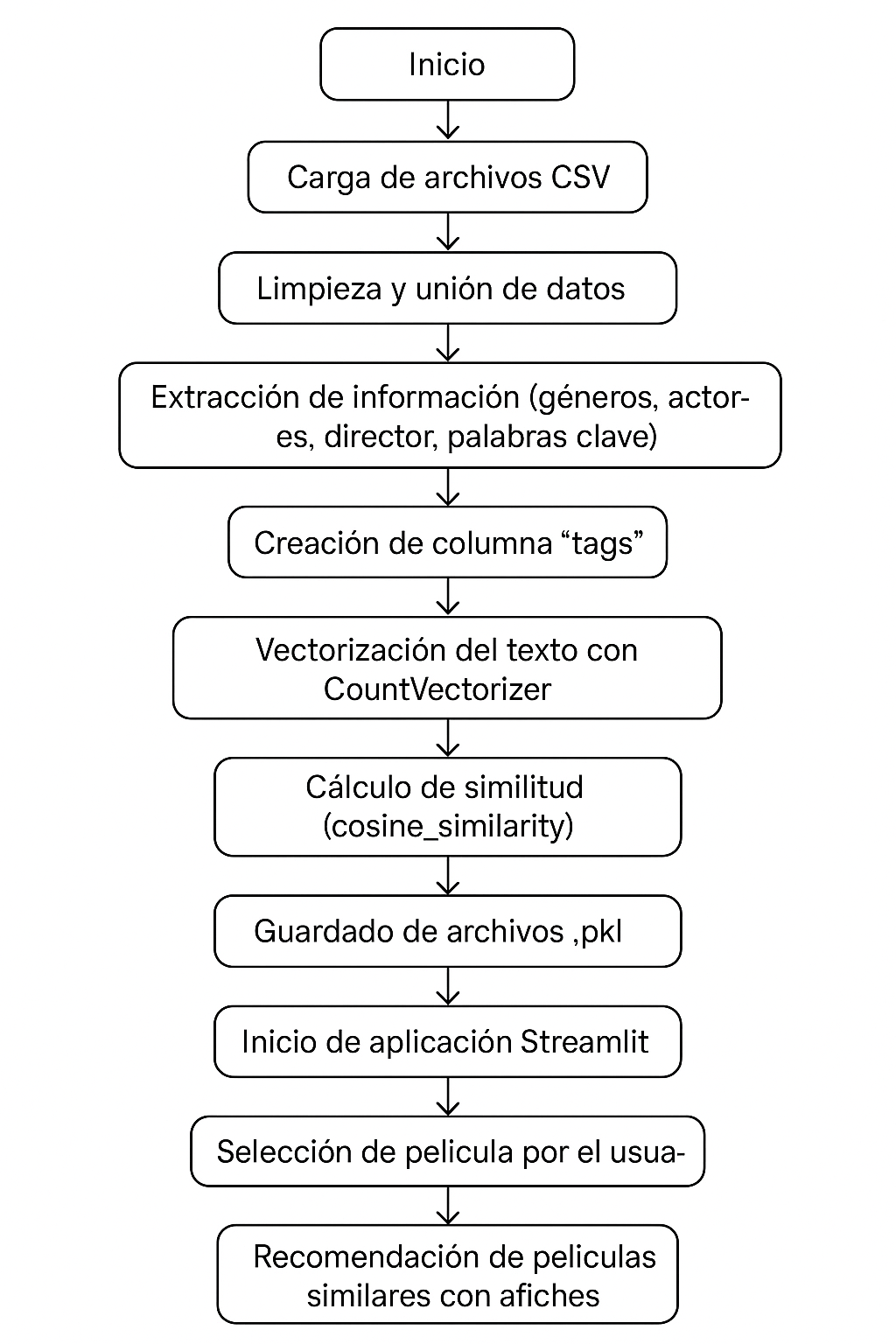
**6. Diseño del Proyecto**

* **Arquitectura o estructura del programa:** (modularización, funciones, clases, etc.)

El programa está dividido en **dos partes principales**:

1. **Procesamiento de datos y creación del modelo:**
   * Limpieza y unión de los archivos de películas y créditos.
   * Extracción de información relevante (géneros, reparto, director, palabras clave, sinopsis).
   * Creación de una columna “tags” con toda esa información combinada.
   * Conversión de texto en vectores usando CountVectorizer.
   * Cálculo de similitudes con cosine\_similarity.
   * Guardado de los resultados en archivos .pkl para reutilizarlos.
2. **Interfaz en Streamlit:**
   * Carga de los datos procesados y del modelo.
   * Función que muestra las películas más parecidas a la seleccionada.
   * Visualización de títulos y afiches en una interfaz limpia y moderna.

* **Diagrama de flujo**



Fin

* **Interfaz (si aplica):** descripción o imagen de la interfaz gráfica o consola

El sistema cuenta con una interfaz web sencilla creada con **Streamlit**.  
El usuario puede seleccionar una película desde una lista desplegable y luego presionar un botón para ver las recomendaciones.  
Cada recomendación se muestra con su título y una imagen del póster obtenida automáticamente desde la API de TMDB.

**7. Desarrollo**

* Explicación paso a paso de cómo se desarrolló el proyecto

**Importar y unir los datos:**  
Se cargaron los archivos tmdb\_5000\_movies.csv y tmdb\_5000\_credits.csv, y se unieron mediante el título de la película.

**Limpieza y selección de columnas:**  
Se eliminaron valores nulos y se seleccionaron las columnas más importantes: movie\_id, title, overview, genres, keywords, cast y crew.

**Transformación de datos en listas:**  
Se usó la librería ast para convertir los textos con formato JSON en listas de Python, y se extrajeron nombres de géneros, actores, directores y palabras clave.

**Creación de los “tags”:**  
Toda la información relevante se unió en una sola columna de texto llamada tags.

**Vectorización y similitud:**  
Se aplicó CountVectorizer para convertir los textos en vectores numéricos y luego se calculó la similitud entre películas usando cosine\_similarity.

**Guardado del modelo:**  
Se guardaron los datos procesados y la matriz de similitud en archivos .pkl con pickle.

**Creación de la interfaz:**  
En Streamlit se programó una interfaz donde el usuario selecciona una película y recibe cinco recomendaciones visuales con sus afiches.

* Fragmentos de código relevantes comentados

# Combinar las columnas de texto en una sola lista

movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']

# Unir las palabras en un solo string

new['tags'] = new['tags'].apply(lambda x: " ".join(x))

# Vectorizar el texto (transformarlo en números)

cv = CountVectorizer(max\_features=5000, stop\_words='english')

vector = cv.fit\_transform(new['tags']).toarray()

# Calcular la similitud entre todas las películas

similarity = cosine\_similarity(vector)

# Función para recomendar películas

def recommend(movie):

index = new[new['title'] == movie].index[0]

distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

for i in distances[1:6]:

print(new.iloc[i[0]].title)

* Descripción de las funciones principales

**convert(text)** → Convierte texto con formato JSON en una lista con los nombres de géneros o palabras clave.

**fetch\_director(text)** → Extrae el nombre del director de cada película.

**collapse(L)** → Elimina espacios y deja las palabras en formato limpio.

**recommend(movie)** → Calcula y muestra las películas más parecidas a la seleccionada.

**fetch\_poster(movie\_id)** → Usa la API de TMDB para obtener la imagen del afiche de cada película.

**8. Pruebas y Resultados**

* Cómo se probó el programa
* Capturas de pantalla o ejemplos de ejecución
* Resultados obtenidos
* Manual de usuario

**9. Conclusiones**

* Lecciones aprendidas
* Dificultades encontradas y cómo se resolvieron
* Posibles mejoras o ideas futuras

**10. Bibliografía / Recursos**

* Sitios web, documentación, libros o videos utilizados, mínimo 10