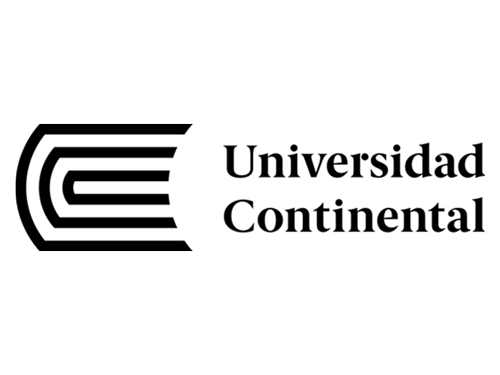
**UNIVERSIDAD CONTINENTAL**

**FACULTAD DE INGENIERÍA**



**INFORME DE ESTRUCTURA DE DATOS**

**NRC: 29901**

**Asignatura:** Estructura de datos

**Docente:** Osorio Contreras Rosario Delia

Proyecto Grupal

**“SISTEMA DE GESTIÓN DE PROCESOS”**

**INTEGRANTES:**

* García Betancourt Israel Jedidias
* Leon Armas Luis Aram
* Ramírez Quillatupa Juan Diego
* Veliz Durand Vieri Del Piero

**2025 - 10**

**Huancayo - Perú**

**ÍNDICE**

[**1. Resumen** 1](#_Toc199365406)

[**2. Introducción** 1](#_Toc199365407)

[**3. Objetivos** 1](#_Toc199365408)

[**3.1. Objetivo general** 1](#_Toc199365409)

[**3.2. Objetivos específicos** 1](#_Toc199365410)

[**4. Marco teórico** 2](#_Toc199365411)

[**4.1 Listas Enlazadas** 2](#_Toc199365412)

[**4.2 Pilas (Stack)** 2](#_Toc199365413)

[**5. Planificación y diseño** 3](#_Toc199365414)

[**5.1. Análisis del problema y diseño de solución** 3](#_Toc199365415)

[**5.2. Diagrama de las estructuras de datos a implementar** 5](#_Toc199365416)

[**5.3. Pseudocódigo de las principales operaciones** 5](#_Toc199365417)

[**6. Implementación y desarrollo** 6](#_Toc199365418)

[**6.1. Captura de pantallas de las primeras ventanas de ejecución con pruebas** 6](#_Toc199365419)

[**6.2. Implementación de las estructuras de datos básicas** 6](#_Toc199365420)

[**6.3. Desarrollo de las principales funcionalidades** 7](#_Toc199365421)

[**7. Conclusión** 11](#_Toc199365422)

[**8. Anexos** 12](#_Toc199365423)

**Sistema de gestión de procesos en un sistema operativo**

# **1. Resumen**

Este informe presenta el desarrollo de un sistema que simula la gestión de procesos, inspirado en el funcionamiento de un sistema operativo. Para ello, se utilizaron estructuras de datos dinámicas como listas enlazadas, colas de prioridad y pilas, todas implementadas desde cero. El documento describe el análisis del problema, el diseño del sistema, la implementación de cada estructura y la creación de una interfaz sencilla para el usuario. También se incorporó un sistema de guardado de datos para conservar el estado entre sesiones. Finalmente, se muestran los resultados de las pruebas realizadas, junto con la organización del equipo y la presentación técnica del proyecto.

# **2. Introducción**

La gestión de procesos es fundamental en los sistemas operativos para organizar las tareas y usar bien los recursos. En este proyecto, se creó un sistema que simula esta gestión usando estructuras de datos dinámicas como listas enlazadas, pilas y colas, todas hechas desde cero. Esto permite entender mejor cómo funcionan y cómo ayudan a administrar procesos, planificar su ejecución y manejar la memoria. El informe explica el análisis, diseño, desarrollo y pruebas del sistema, y destaca el trabajo en equipo y la documentación para garantizar su buen funcionamiento.

# **3. Objetivos**

## **3.1. Objetivo general**

Desarrollar un sistema de gestión de procesos que simule las funciones básicas de un sistema operativo, usando estructuras de datos dinámicas lineales creadas desde cero. El sistema debe optimizar la administración, planificación y gestión de memoria de los procesos.

## **3.2. Objetivos específicos**

* Diseñar e implementar una lista enlazada dinámica para registrar, buscar, modificar y eliminar procesos de forma eficiente.
* Crear una cola de prioridad para organizar la planificación y ejecución de procesos según su importancia.
* Implementar una pila dinámica para manejar la asignación y liberación de bloques de memoria.
* Desarrollar una interfaz fácil de usar que permita gestionar procesos y memoria de manera sencilla.
* Integrar un sistema para guardar y recuperar datos que mantenga el estado del sistema entre sesiones.
* Realizar pruebas y análisis para asegurar que el sistema funcione correctamente, sea eficiente y estable.

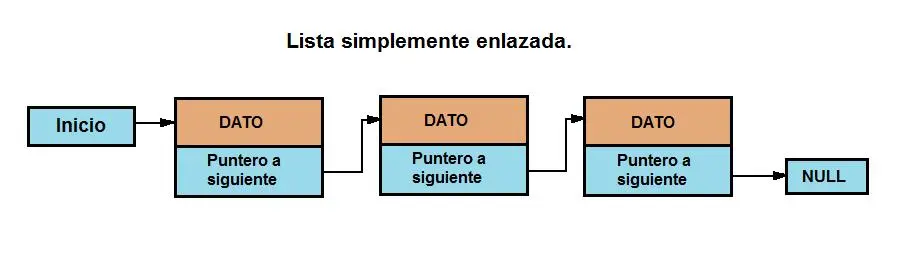
# **4. Marco teórico**

## **4.1 Listas Enlazadas**

Una lista enlazada es una estructura de datos dinámica compuesta por nodos, donde cada nodo contiene un dato y un puntero al siguiente nodo de la lista. Este tipo de estructura permite una gestión flexible de memoria y facilita operaciones como inserción y eliminación de elementos sin necesidad de desplazar otros elementos, a diferencia de los arreglos.

Las funciones principales que operan sobre esta lista son:

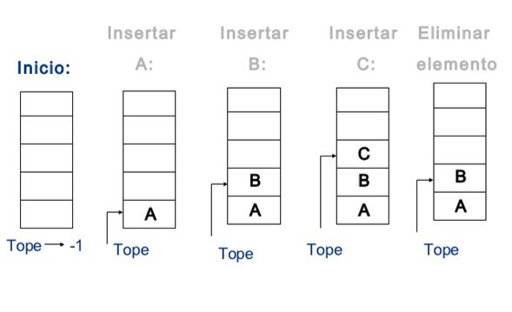
* Inserción de procesos: agrega un nuevo nodo al final de la lista.
* Eliminación de procesos: busca y elimina un nodo que coincide con un PID específico.
* Búsqueda: permite localizar un proceso por su PID o nombre.
* Modificación: actualiza la prioridad de un proceso dado su PID.
* Recorrido: muestra todos los procesos presentes en la lista.



## **4.2 Pilas (Stack)**

Una pila es una estructura de datos que sigue el principio LIFO (Last In, First Out), donde el último elemento insertado es el primero en ser eliminado. Las operaciones básicas son:

* Push: insertar un elemento en el tope de la pila.
* Pop: eliminar el elemento del tope.
* Top/Peek: obtener el elemento del tope sin eliminarlo.



# **5. Planificación y diseño**

## **5.1. Análisis del problema y diseño de solución**

**5.1.1 Análisis del problema**

En los sistemas operativos modernos, la gestión de procesos y el control de memoria son tareas fundamentales que permiten mantener la estabilidad y eficiencia del sistema. Sin embargo, comprender cómo funcionan internamente estos mecanismos puede resultar complejo. Por ello, este proyecto tiene como propósito simular de manera básica cómo opera un sistema operativo en cuanto al manejo de procesos y la memoria, a través de un programa desarrollado en C++.

El problema a resolver consiste en crear un programa que permita al usuario:

* Administrar una lista de procesos (crear, eliminar, buscar, modificar y visualizar).
* Gestionar la memoria asignada a esos procesos, simulando una pila (estructura LIFO).
* Interactuar mediante menús simples en consola.

Este sistema simulado servirá como herramienta educativa para entender el comportamiento de dos estructuras internas típicas de un sistema operativo: la lista de procesos y la pila de asignación de memoria.

**5.1.2. Diseño de solución**

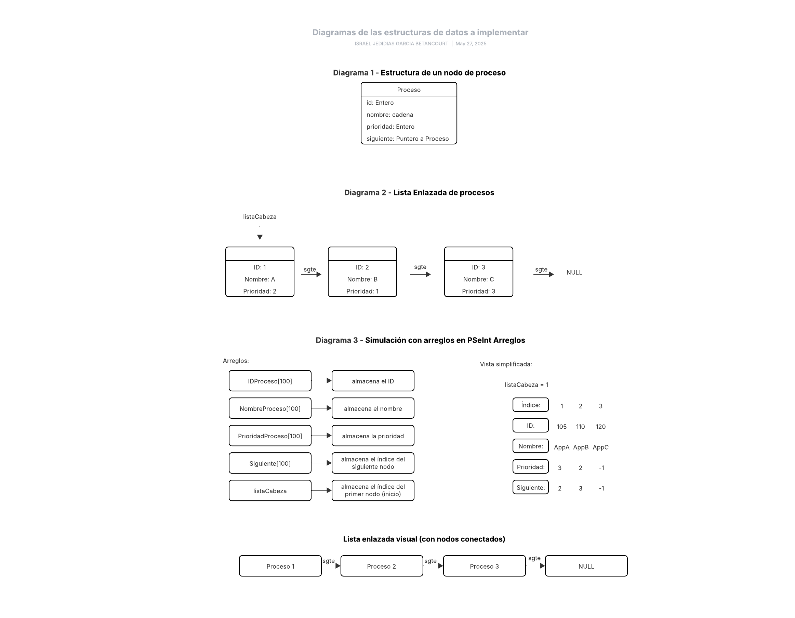
El programa está desarrollado en C++ y busca simular algunas funciones básicas que un sistema operativo realiza para administrar procesos y memoria.

Para lograr esto, el programa está dividido en dos módulos principales:

* **Gestor de Procesos**  
  Esta parte se encarga de manejar los procesos activos, permitiendo al usuario agregar nuevos procesos, eliminarlos, buscar información sobre ellos, modificar sus prioridades y visualizar todos los procesos registrados. Para almacenar la información, se utiliza una estructura de lista enlazada que permite insertar y eliminar procesos de manera dinámica.
* **Gestor de Memoria**  
  Aquí se controla la asignación y liberación de memoria a los procesos. La memoria se administra con una estructura tipo pila (LIFO), donde los bloques de memoria asignados se apilan y desapilan según sea necesario, permitiendo así manejar la memoria de forma ordenada y eficiente.

Ambos módulos están integrados en un menú principal interactivo, que facilita la navegación y permite al usuario seleccionar las acciones que desea realizar. Además, se incluyen submenús para cada gestor, asegurando que el usuario pueda ejecutar múltiples operaciones sin salir del sistema hasta que decida finalizar.

## **5.2. Diagrama de las estructuras de datos a implementar**



**Figura 1.** En la imagen se muestra el diagrama de las estructuras de datos a implementar de nuestro pseudocódigo realizado. [https://lucid.app/lucidchart/457c69ba-9151-4eca-a34e-6c8fec21cab6/edit?page=0\_0&invitationId=inv\_95398363-859b-4f55-9687-98137b853153#](https://lucid.app/lucidchart/457c69ba-9151-4eca-a34e-6c8fec21cab6/edit?page=0_0&invitationId=inv_95398363-859b-4f55-9687-98137b853153)

## **5.3. Pseudocódigo de las principales operaciones**

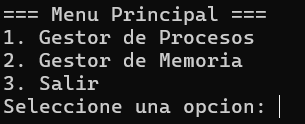


**Figura 2.** En la imagen se muestra el Pseudocódigo de las principales operaciones de listas y pilas. <https://sistemagestionprocesosg2.netlify.app/>

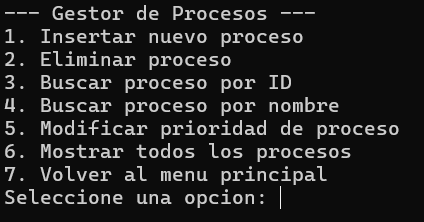
# **6. Implementación y desarrollo**

## **6.1. Captura de pantallas de las primeras ventanas de ejecución con pruebas**

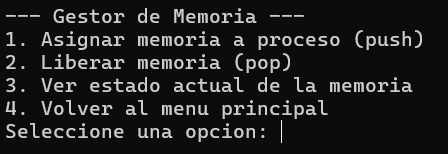
**6.1.1 Menú Principal**



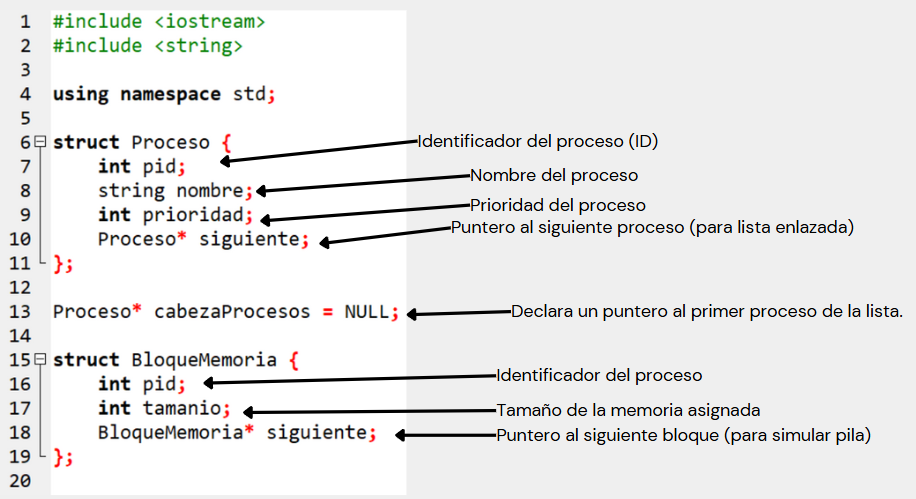
**6.1.2 Menú Gestor de Procesos (listas)**



**6.1.3 Menú Gestor de Memoria (pilas)**

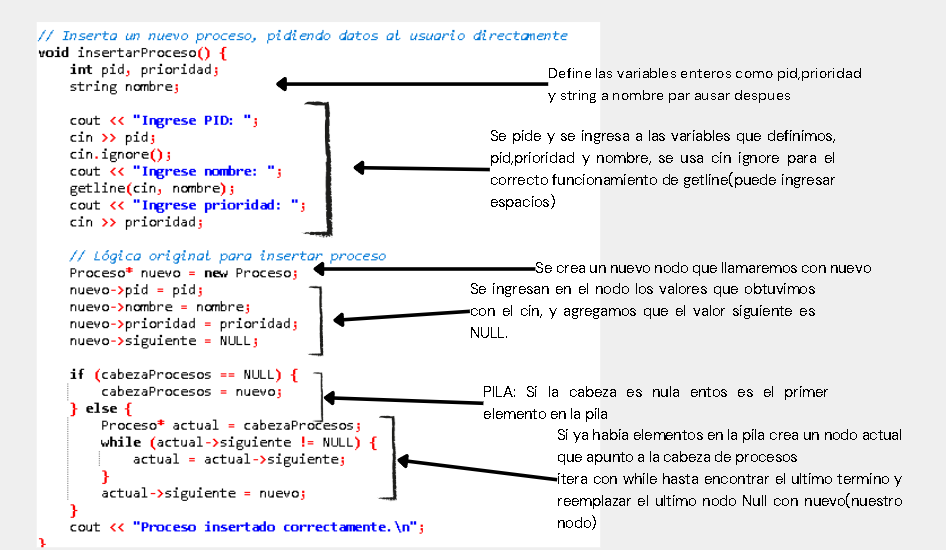


## **6.2. Implementación de las estructuras de datos básicas**

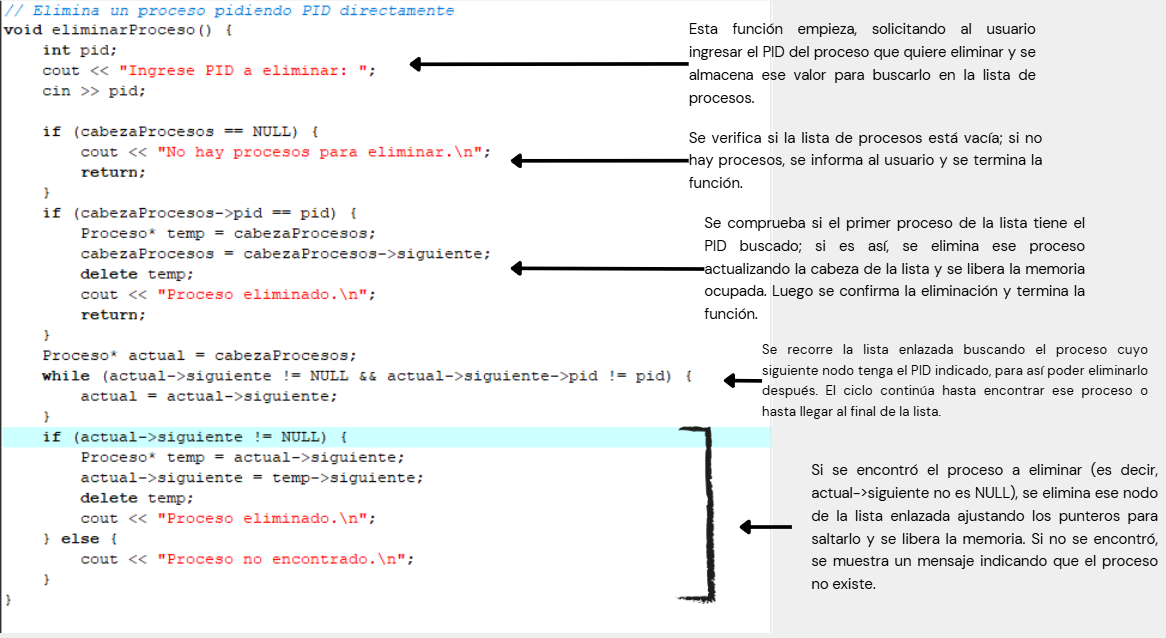


## **6.3. Desarrollo de las principales funcionalidades**

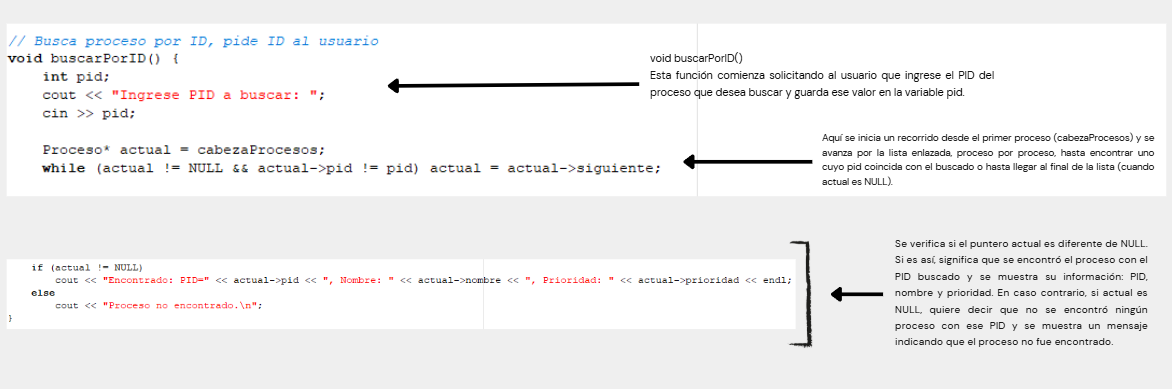
**6.3.1 Void insertarProceso():** Inserta el proceso en la lista y en la pila con los valores de PID (ID único), nombre y prioridad.



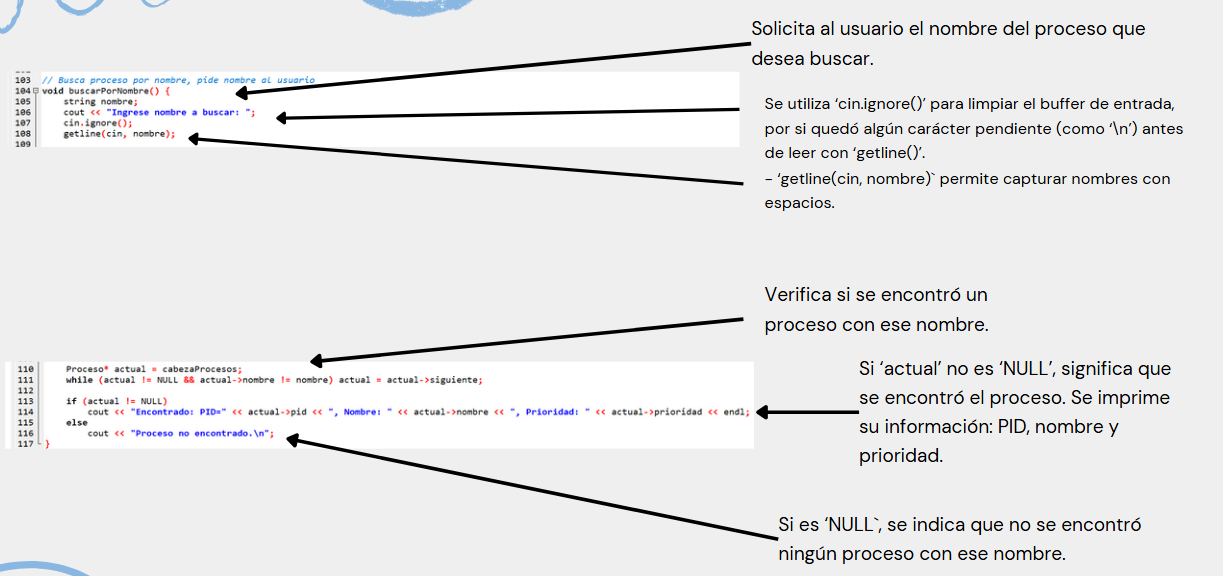
**6.3.2 Void eliminarProceso()** , Elimina un proceso de la lista por su PID ingresado, y muestra si fue eliminado o no encontrado.



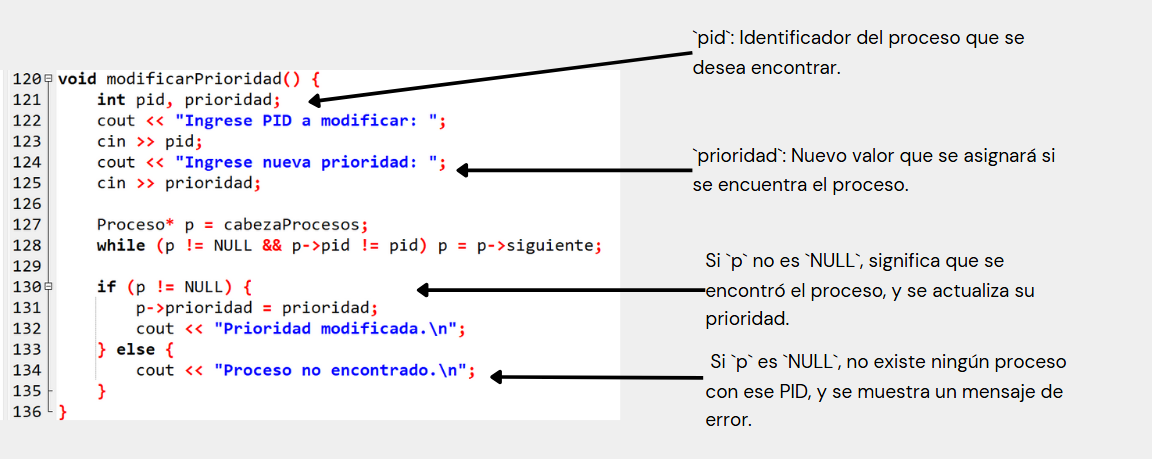
**6.3.3 Void buscarPorID()** , Busca un proceso en la lista por el PID ingresado y muestra su información si lo encuentra, o un mensaje si no existe.



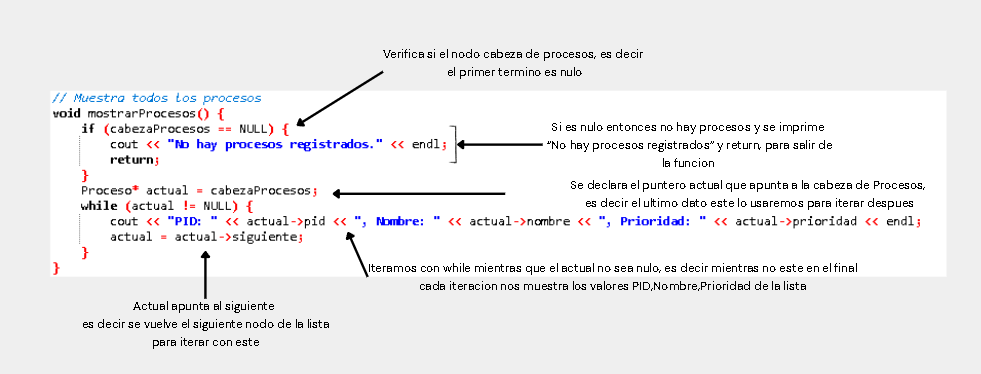
**6.3.4. Void buscarPorNombre():** Esta función permite buscar de manera secuencial un proceso específico por nombre dentro de una lista enlazada simple.



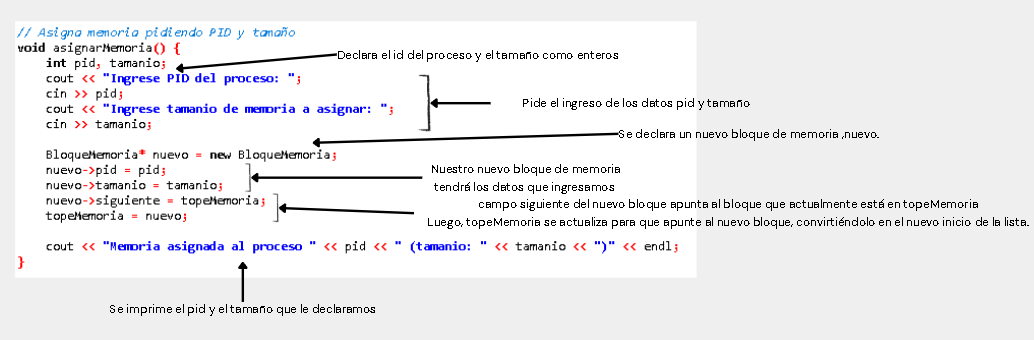
**6.3.5. Void ModificarPrioridad():** Esta función permite modificar la prioridad de un proceso dentro de una lista enlazada de procesos, buscando dicho proceso por su PID (identificador único).



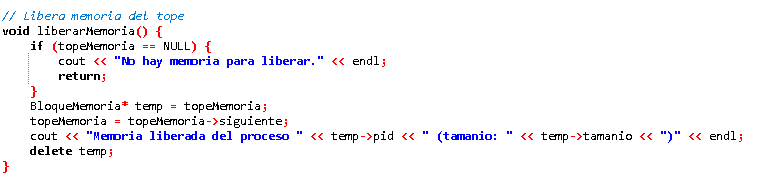
**6.3.6 Void mostrarProcesos():** Esta función permite mostrar todos los procesos registrados, cada proceso es mostrado con su PID(id único), nombre y prioridad.



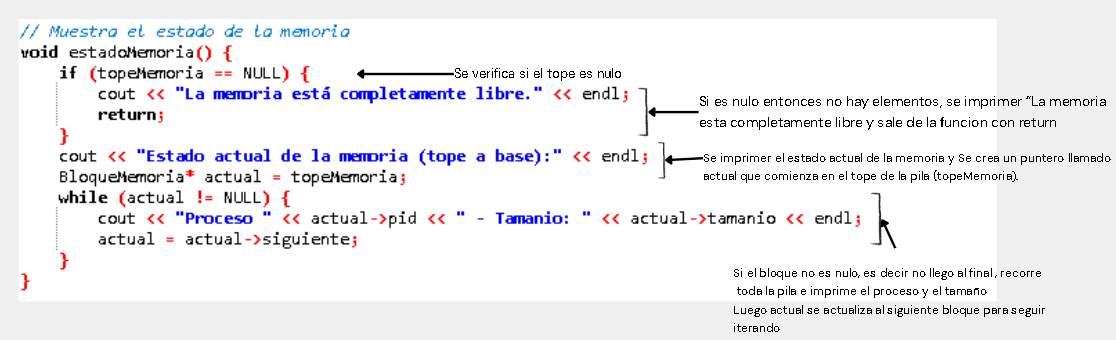
**6.3.7 Void asignarMemoria ():** Permite asignar un tamaño a un proceso, despues de identificar el proceso por su PID(id único) e ingresar su tamaño.



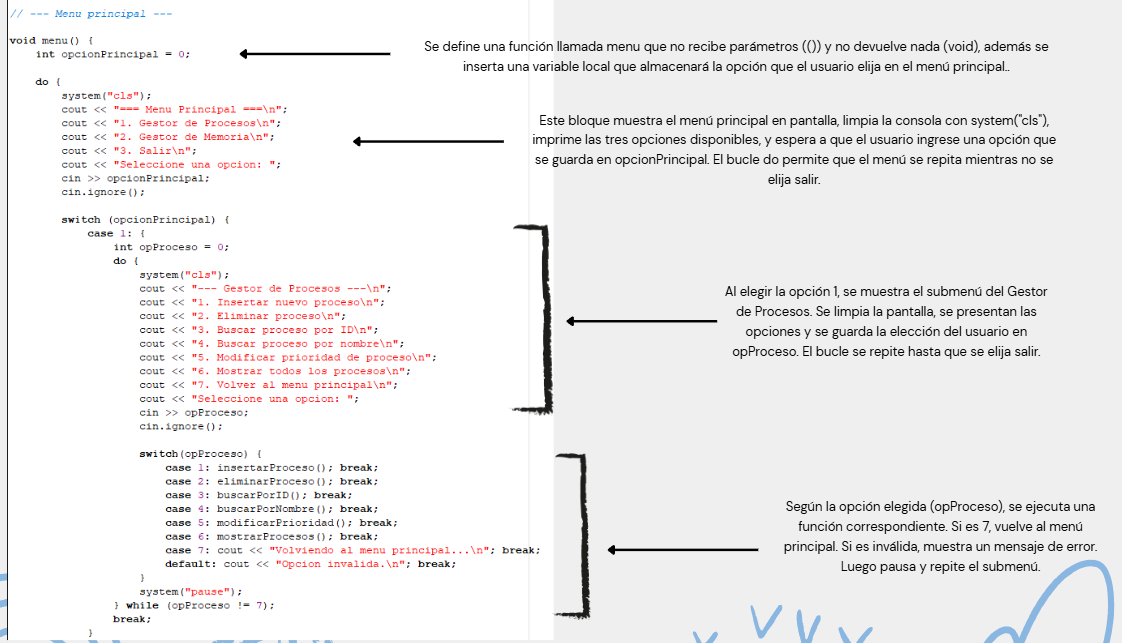
**6.3.8 Void liberarMemoria ():** Permite liberar la memoria del último proceso que se asignó memoria, si es que no hay memoria para liberar solo imprime que no hay que liberar.

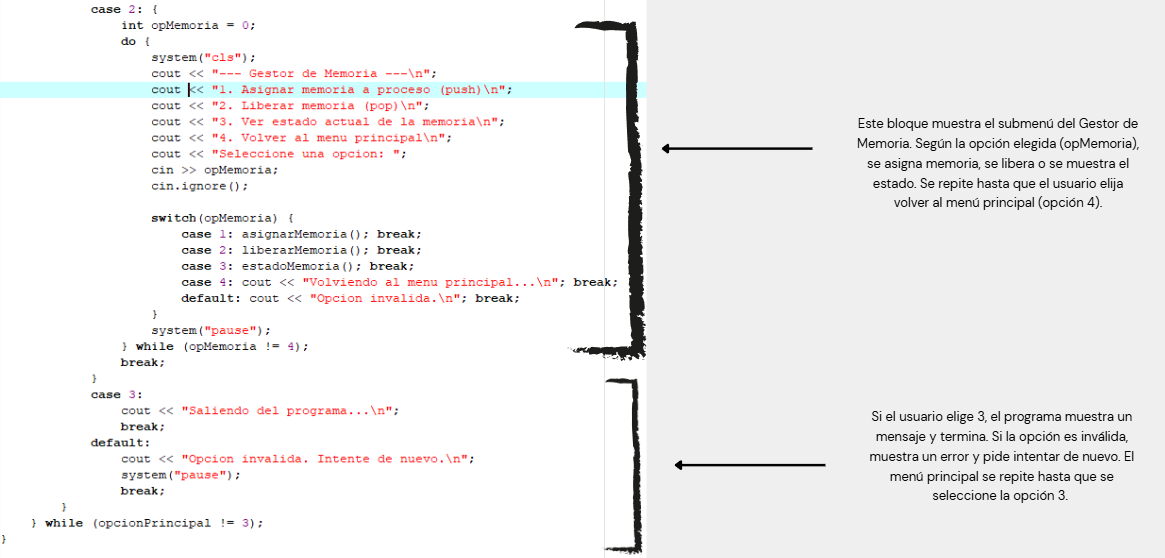


**6.3.9 Void estadoMemoria ():** Muestra todos los elementos de la pila, junto con su PID y su tamaño, en orden desde el tope hasta el final.



**6.3.10 Void menu ():** Muestra el menú principal del programa





**6.3.11 int main ():** Finalmente, este int main(), hara que todo se inicialice correctamente

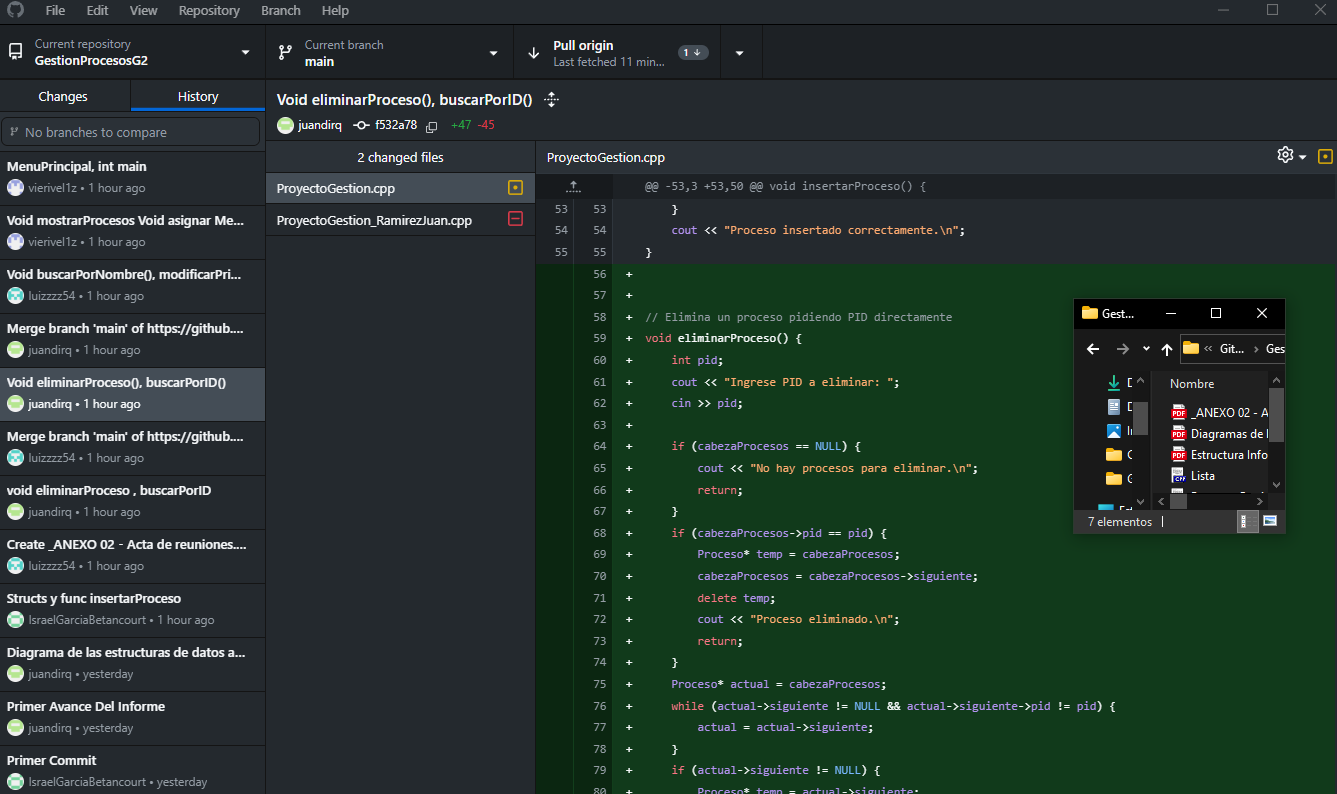


# **7. Conclusión**

El programa desarrollado ofrece una solución eficiente para la gestión de procesos y memoria utilizando estructuras dinámicas (listas enlazadas y pilas). Permite al usuario insertar, buscar, modificar y eliminar procesos, así como asignar y liberar memoria de manera organizada. La implementación de un menú interactivo facilita la navegación y el manejo de las diferentes funcionalidades, garantizando un control adecuado y práctico sobre los recursos del sistema. En conjunto, este diseño cumple con los requerimientos planteados, demostrando un manejo efectivo de estructuras de datos y la interacción con el usuario.

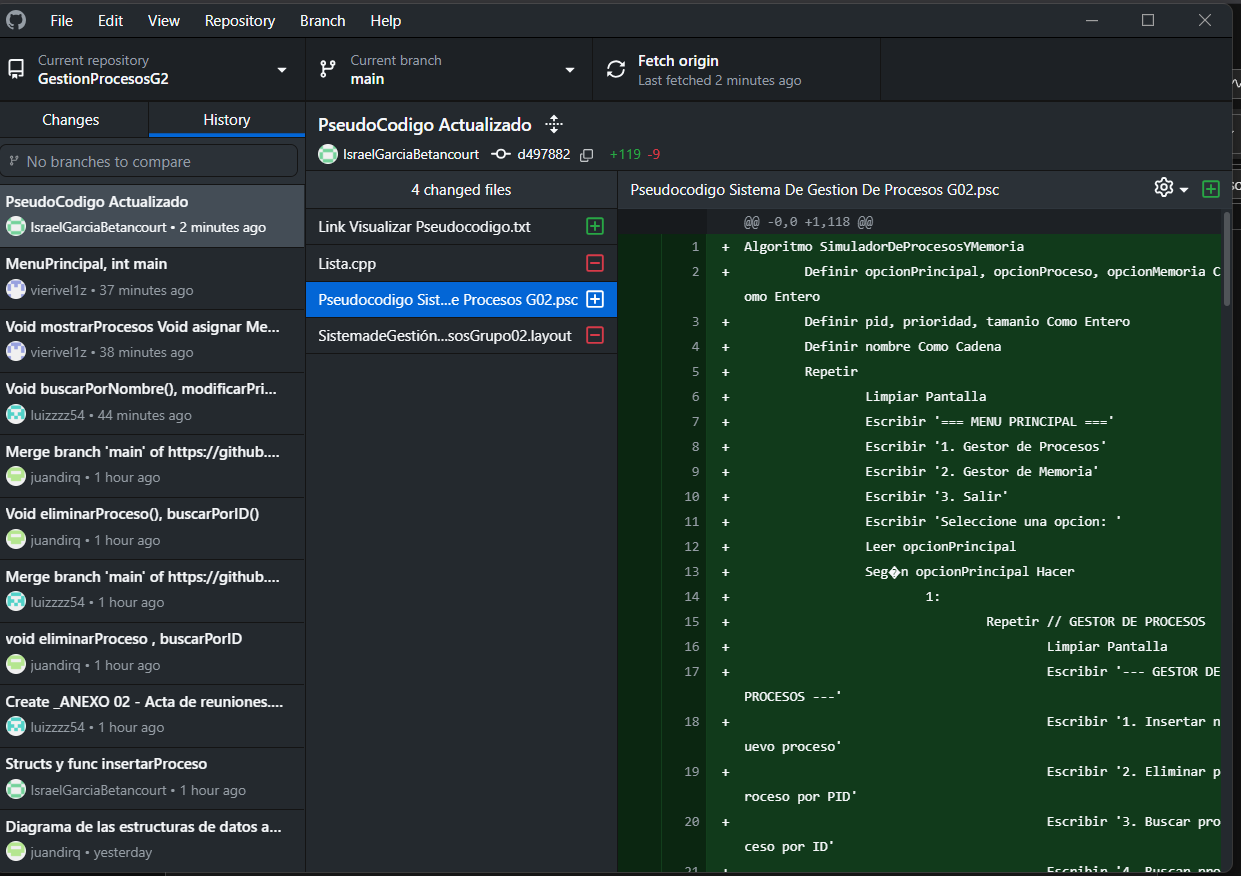
# **8. Anexos**

**Anexo 1 – Captura de participación en el GitHub**



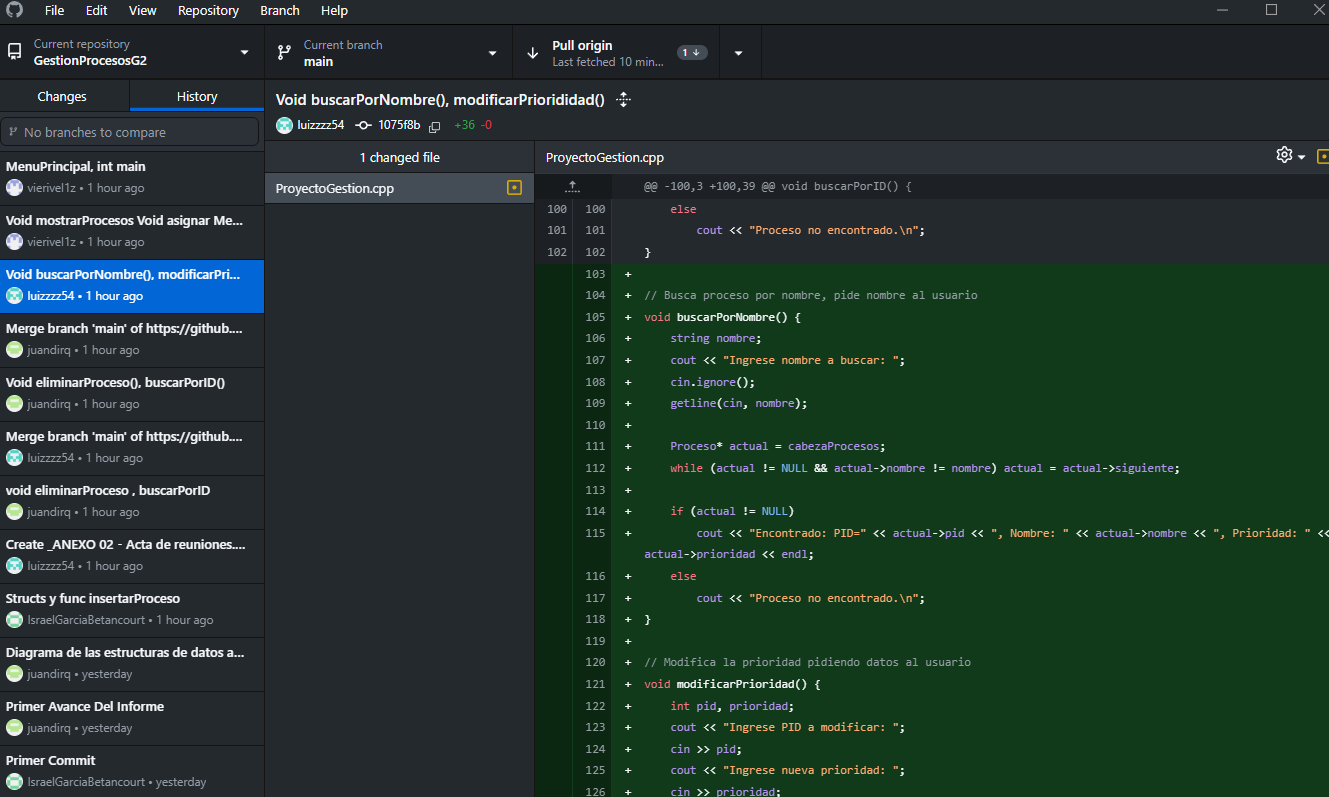
**(Ramírez Quillatupa Juan Diego)**

**Anexo 2 – Captura de participación en el GitHub**



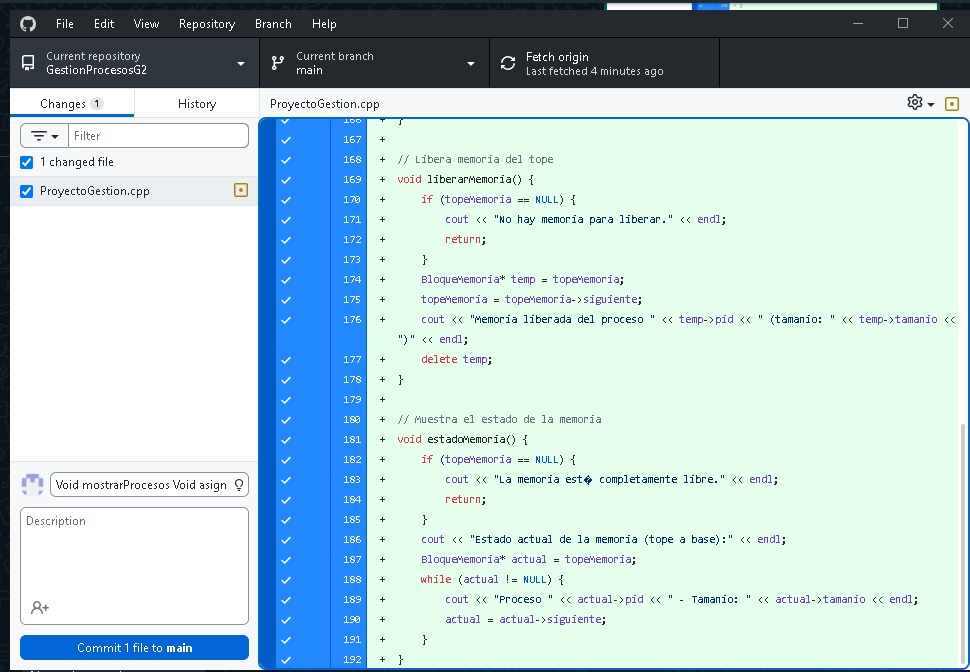
**(García Betancourt Israel Jedidias)**

**Anexo 3 – Captura de participación en el GitHub**



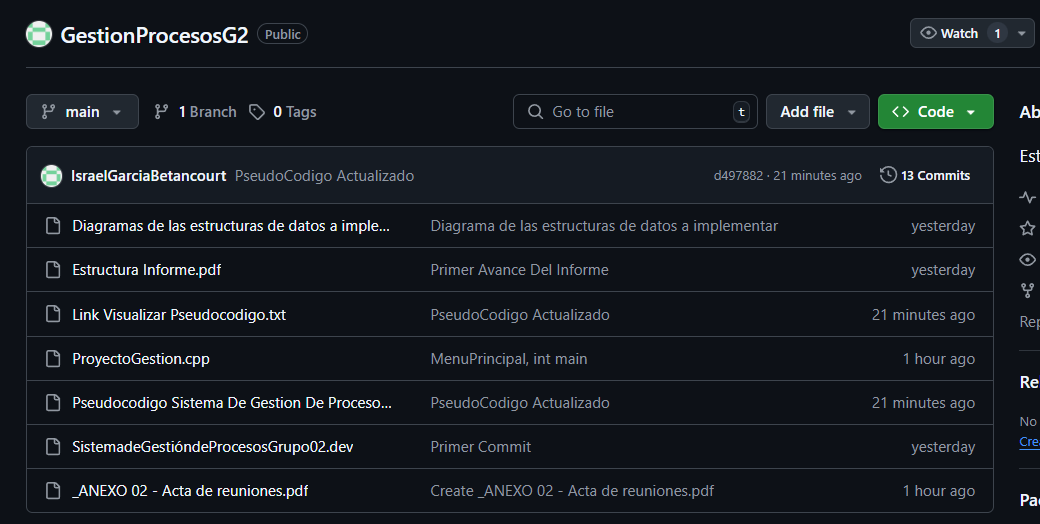
**(Leon Armas Luis Aram)**

**Anexo 4 – Captura de participación en el GitHub**



**(Veliz Durand Vieri Del Piero)**

**Anexo 4 – Enlace al GitHub**



<https://github.com/IsraelGarciaBetancourt/GestionProcesosG2>

**Anexo 5 – Link del Canva (imágenes a mejor calidad)**

<https://www.canva.com/design/DAGos7CEUA0/SLzwxEIwK6lQjbI2S3WHAQ/edit?utm_content=DAGos7CEUA0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>