**Universidad Peruana de Ciencias Aplicadas**



**INFORME DEL TRABAJO PARCIAL**

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO ALGORITMO Y ESTRUCTURA DE DATOS

Sección: 1978

AUTORES

Patricio Farias, Ana Camila ( [u20241i469@upc.edu.pe](mailto:u20241i469@upc.edu.pe) )

Sandoval Cueto, Fabian Jesus ( [u20221a132@upc.edu.pe](mailto:u20221a132@upc.edu.pe) )

Caldas Garcia, Sergio Ruben ( [u202411274@upc.edu.pe](mailto:u202411274@upc.edu.pe) )

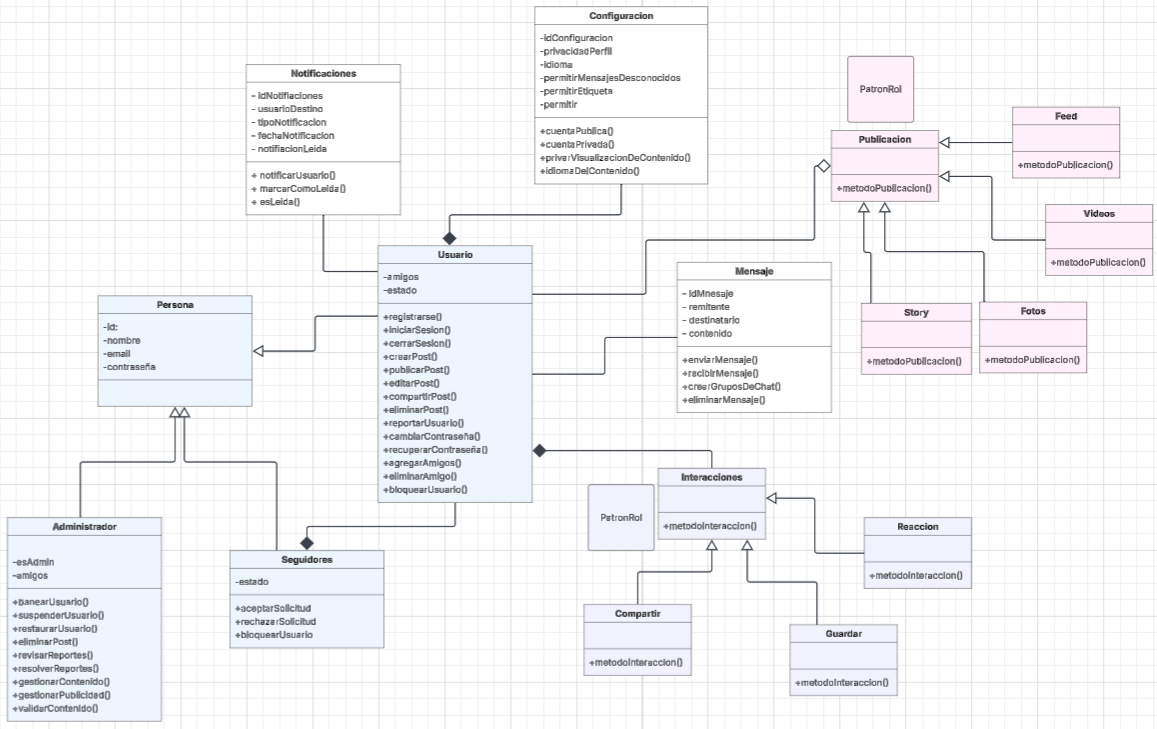
RESUMEN

Las redes sociales basadas en contenido visual se han consolidado como un espacio fundamental para la interacción digital, donde los usuarios comparten experiencias y se conectan de manera dinámica. En este contexto, el presente proyecto plantea el desarrollo de una simulación minimalista en C++, orientada a la implementación de estructuras de datos como listas, algoritmos de ordenamiento y manejo de archivos. El sistema permitirá la gestión de clientes y la publicación de fotos, videos e historias, así como la interacción entre usuarios mediante comentarios, reacciones y mensajes privados, todo desarrollado en consola.

Cabe resaltar que este proyecto no busca desarrollar una aplicación con interfaz gráfica ni un entorno de usuario final completo, sino un prototipo funcional en consola que permita validar los conceptos de estructura, organización y dinámica de interacción propios de una red social. De esta manera, se prioriza la correcta implementación de las estructuras de datos, la lógica de negocio y la simulación de procesos clave, antes de una posible evolución hacia entornos más complejos. Este enfoque minimalista facilita el aprendizaje, la experimentación y la evaluación del sistema, manteniendo la claridad en cada uno de sus componentes.

**Palabras clave:** C++ estructuras de datos, listas enlazadas en C++, colas y pilas en C++, templates en C++, expresiones lambda en C++, recursividad en C++, manejo de archivos en C++, modularidad en C++, algoritmos de ordenamiento C++, búsqueda en C++

**2. Diagrama de Clases:**



**3. Análisis de Complejidad (Big O):**

Adjuntar y realizar el análisis de 3 métodos que consideren más importantes, adjuntando su análisis de complejidad y resultado.

| **N°** | **Clase** | **Descripcion** | **Codigo Fuente** | **Valoración (AA: Big O)** |
| --- | --- | --- | --- | --- |
| 1 | Persona | Asigna valores iniciales a los atributos. |  | O(1) |
| 2 | Pila | Recorre la pila desde la cima hasta la base mostrando los datos |  | O(n) |
| 3 | ListaSimple | Busca un valor específico dentro de una lista enlazada. |  | O(n) |

1. **Uso de Expresiones Lambdas y plantillas (templates):**

Adjunta el código fuente de 9 lambdas y 9 plantillas, donde se evidencie su uso.

| **Expresiones Lambdas** | | | |
| --- | --- | --- | --- |
| **N°** | **Clase** | **Método** | **Código Fuente** |
| 1 | Menu | mostrarMenuUsuario |  |
| 2 | Menu | mostrarMenuUsuario |  |
| 3 | Menu | mostrarMenuUsuario |  |
| 4 | Menu | mostrarMenuUsuario |  |
| 5 | Menu | mostrarMenuUsuario |  |
| 7 | Menu | mostrarMenuAdministrador |  |
| 8 | Menu | mostrarMenuAdministrador |  |
| 9 | Menu | mostrarMenuAdministrador |  |
| **Templates** | | | |
| **N°** | **Clase** | **Método** | **Código Fuente** |
| 1 | Persona   |  | | --- | | class Persona   |  | | --- |  |  | | --- | |  |
| 2 | Seguidores   |  | | --- | | class Seguidores   |  | | --- | |  |
| 3 | Admin | class Admin |  |
| 4 | Notificaciones | clases Notificaciones |  |
| 5 | Usuario | class Usuario |  |
| 6 | Nodo | class Nodo |  |
| 7 | Pila | class Pila |  |
| 8 | QuickSort | class Quicksort |  |
| 9 | QuickSort | class QuickSort |  |

1. **Estructura de Datos:**

Adjuntar código de las 3 estructuras de datos (clases y métodos) en donde se defina y argumente del porque se usó la estructura de datos propuesta: ventajas y desventajas.

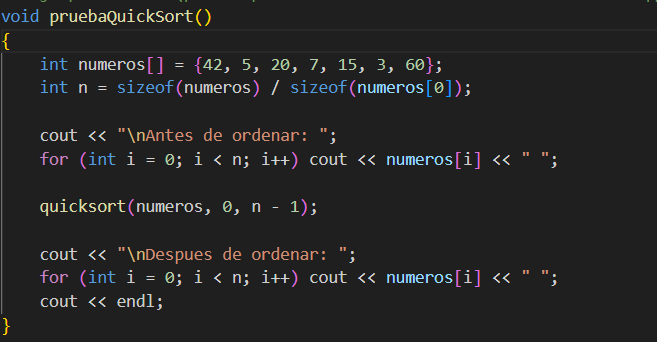
| **N°** | **Clase** | **Estructura de Datos** | **Justificación** |
| --- | --- | --- | --- |
| 1 | ListaSimple.h | Lista simple | Se usa una Lista Simple porque permite manejar colecciones dinámicas de elementos sin conocer su tamaño previamente.  Es útil cuando se requiere insertar o eliminar elementos de forma frecuente sin necesidad de reorganizar toda la memoria |
| 2 | Pilas.h | Pila | Se usa una Pila (Stack) porque su comportamiento LIFO (Last In, First Out) es ideal para:   * Deshacer acciones (undo/redo) * Control de llamadas en recursión * Evaluación de expresiones y análisis de sintaxis |
| 3 | QuickSort.h | Quicksort | Se utiliza QuickSort porque es un algoritmo eficiente para ordenar grandes volúmenes de datos, especialmente cuando el arreglo está desordenado. |

1. **Uso de Recursividad:**

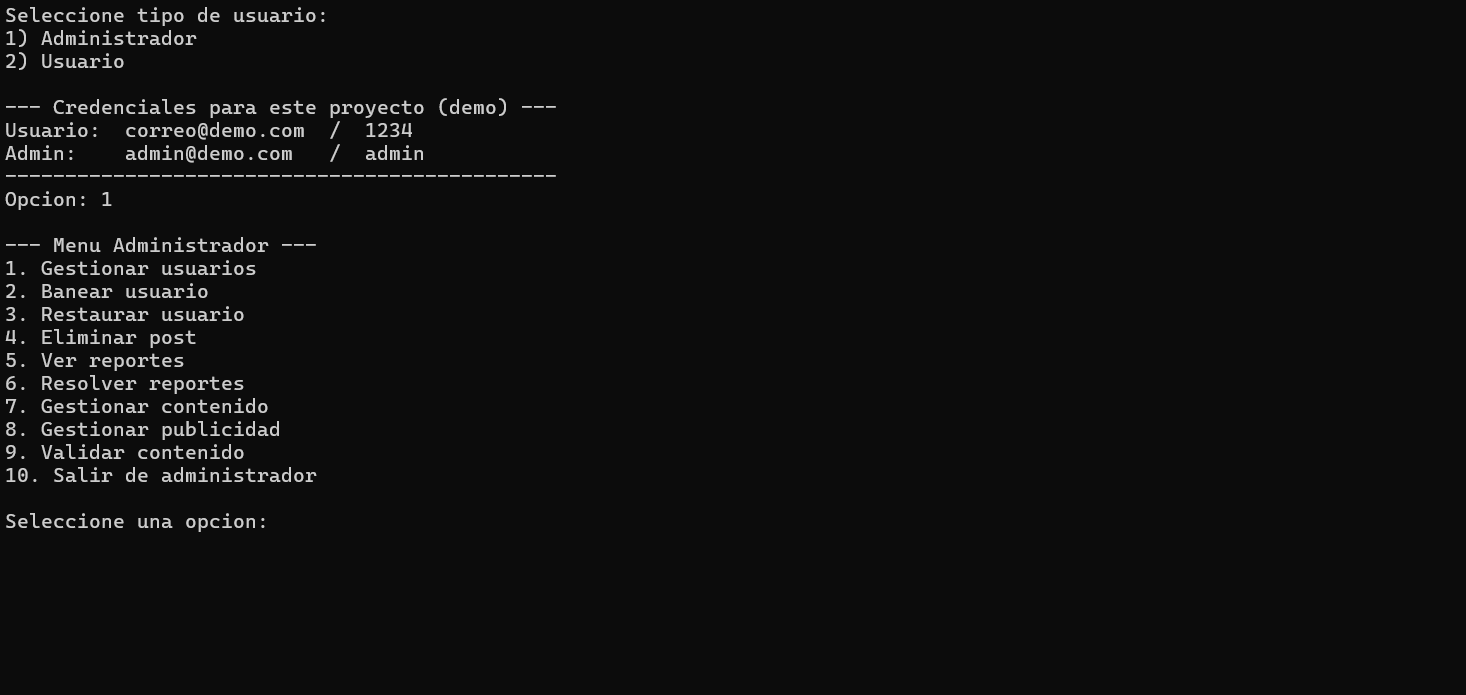
Adjuntar código fuente de 3 funciones de recursividad donde se evidencie su uso.

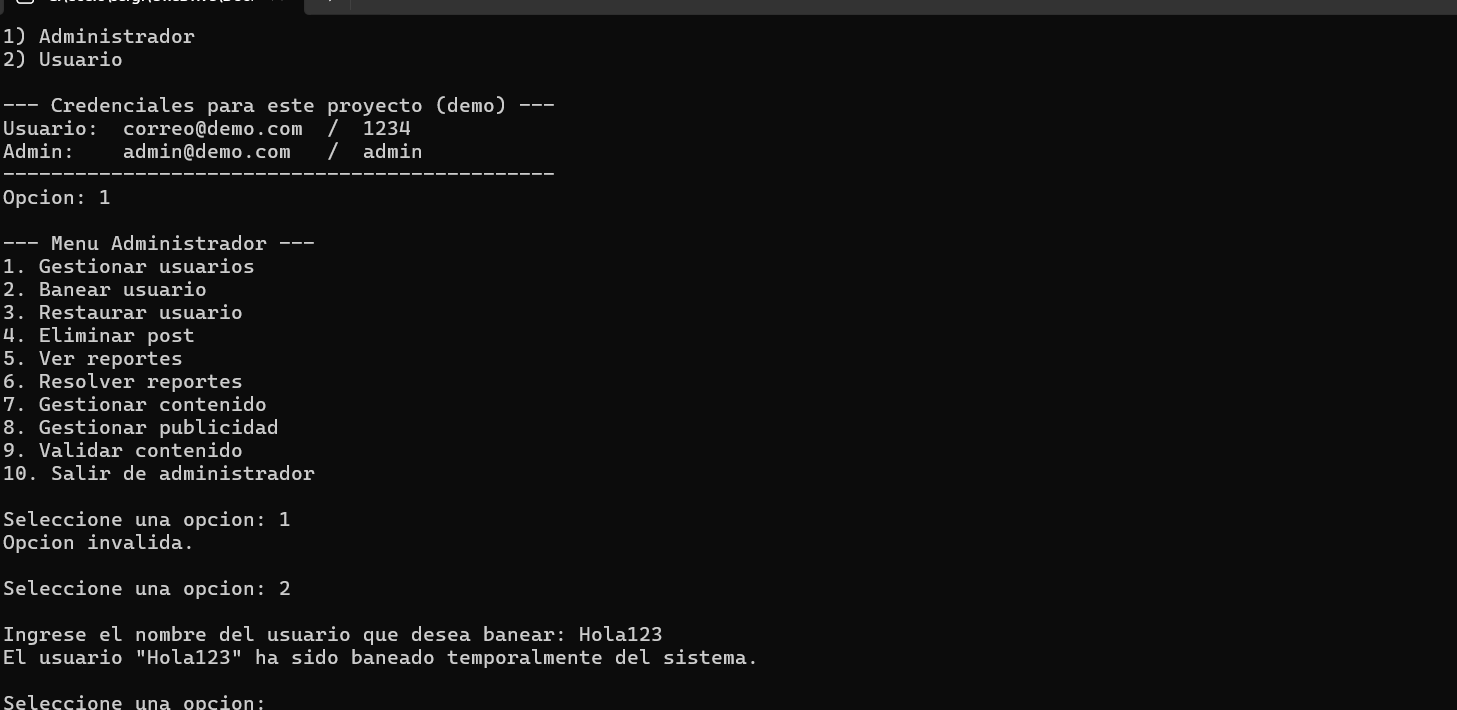
1. **Uso de algoritmo de ordenamiento**

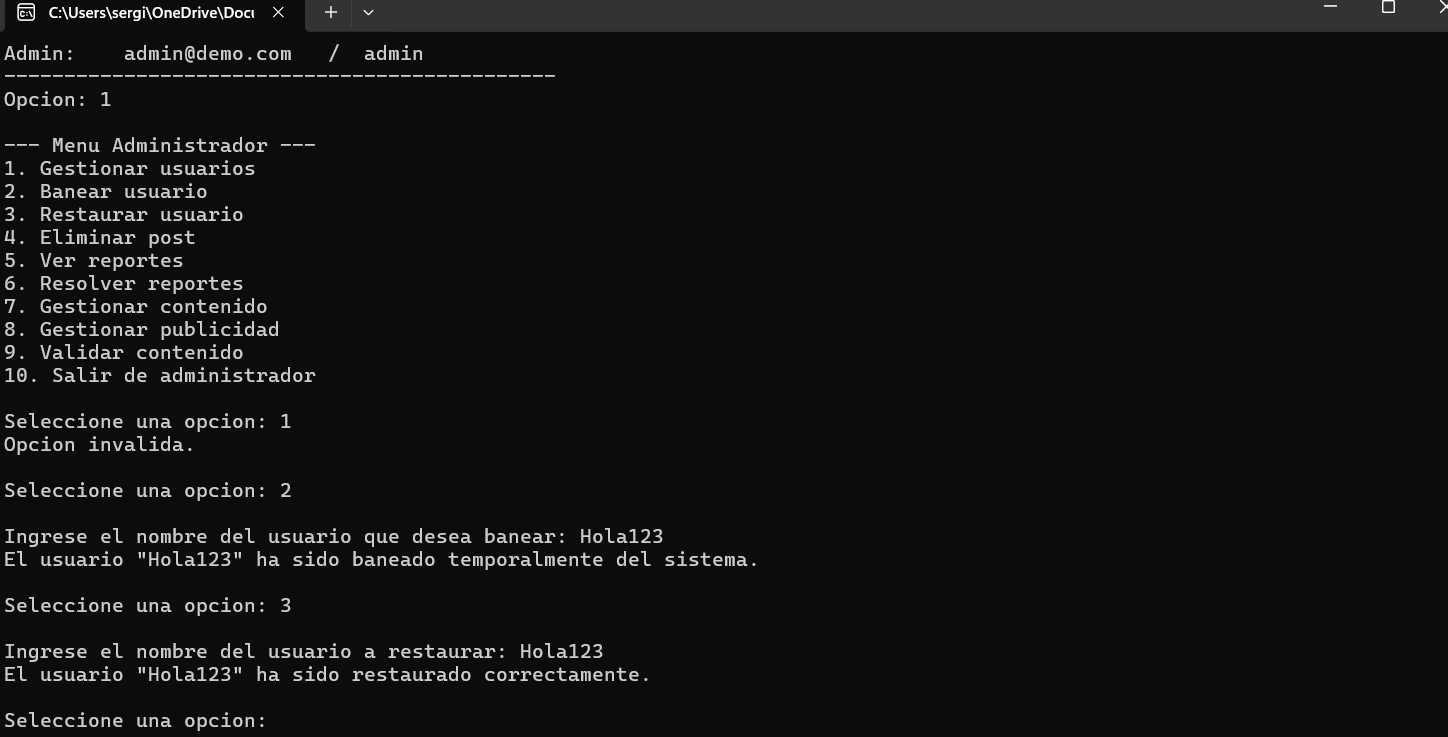
Adjuntar código fuente, en donde se defina y argumente del porque se usó ese algoritmo y donde se evidencie su uso.

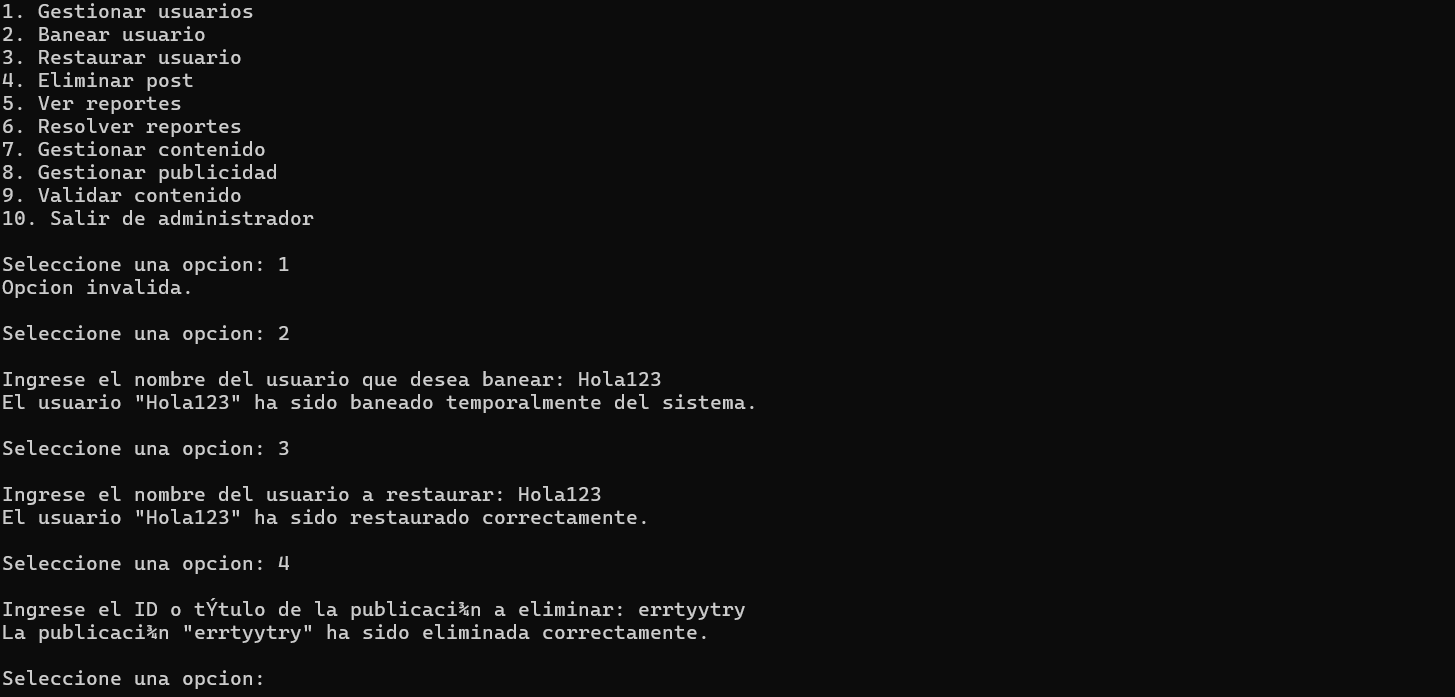


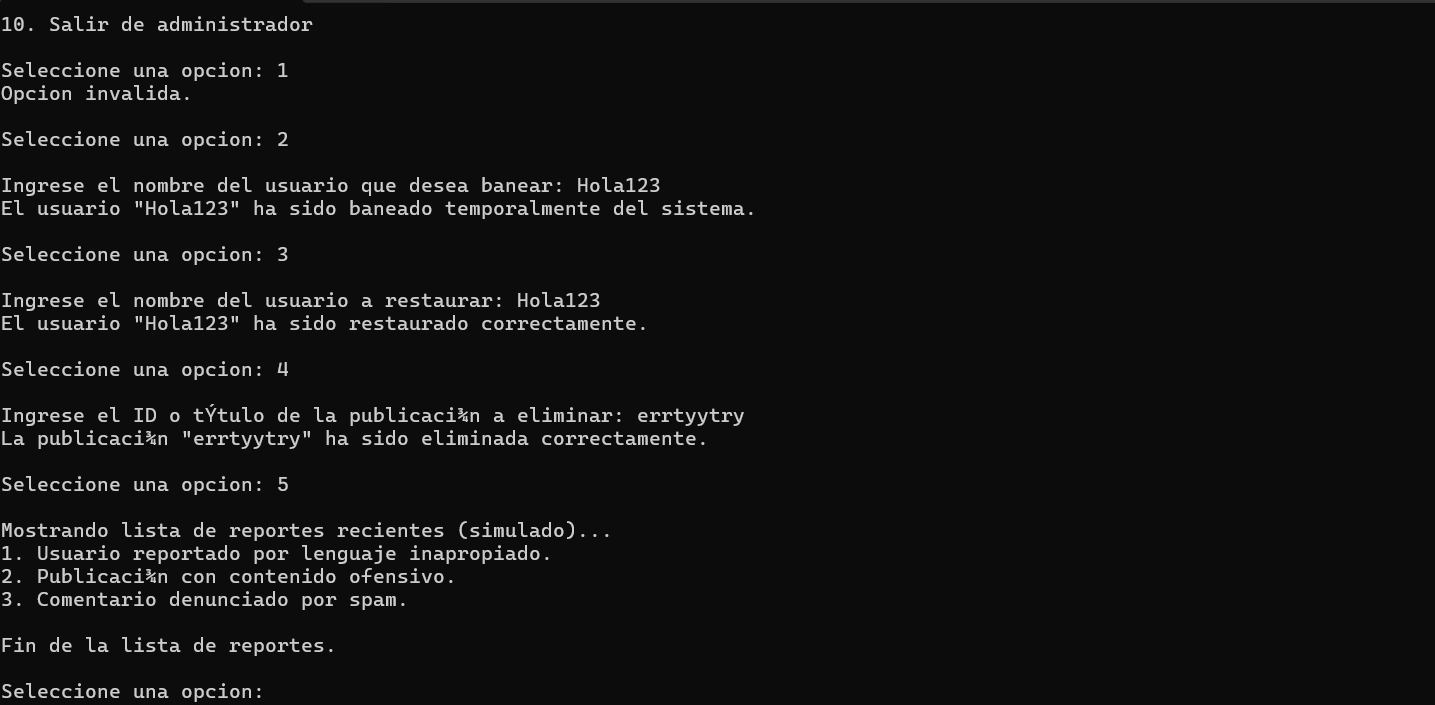
Se utiliza el algoritmo QuickSort porque ofrece un ordenamiento eficiente en promedio con complejidad O(n log n). Es un método in-place que no requiere memoria adicional, ideal para conjuntos de datos grandes. Además, su implementación con plantillas permite reutilizarlo con distintos tipos de datos. Frente a otros métodos como BubbleSort o InsertionSort, QuickSort resulta significativamente más rápido y práctico para la gestión de información dentro del proyecto.

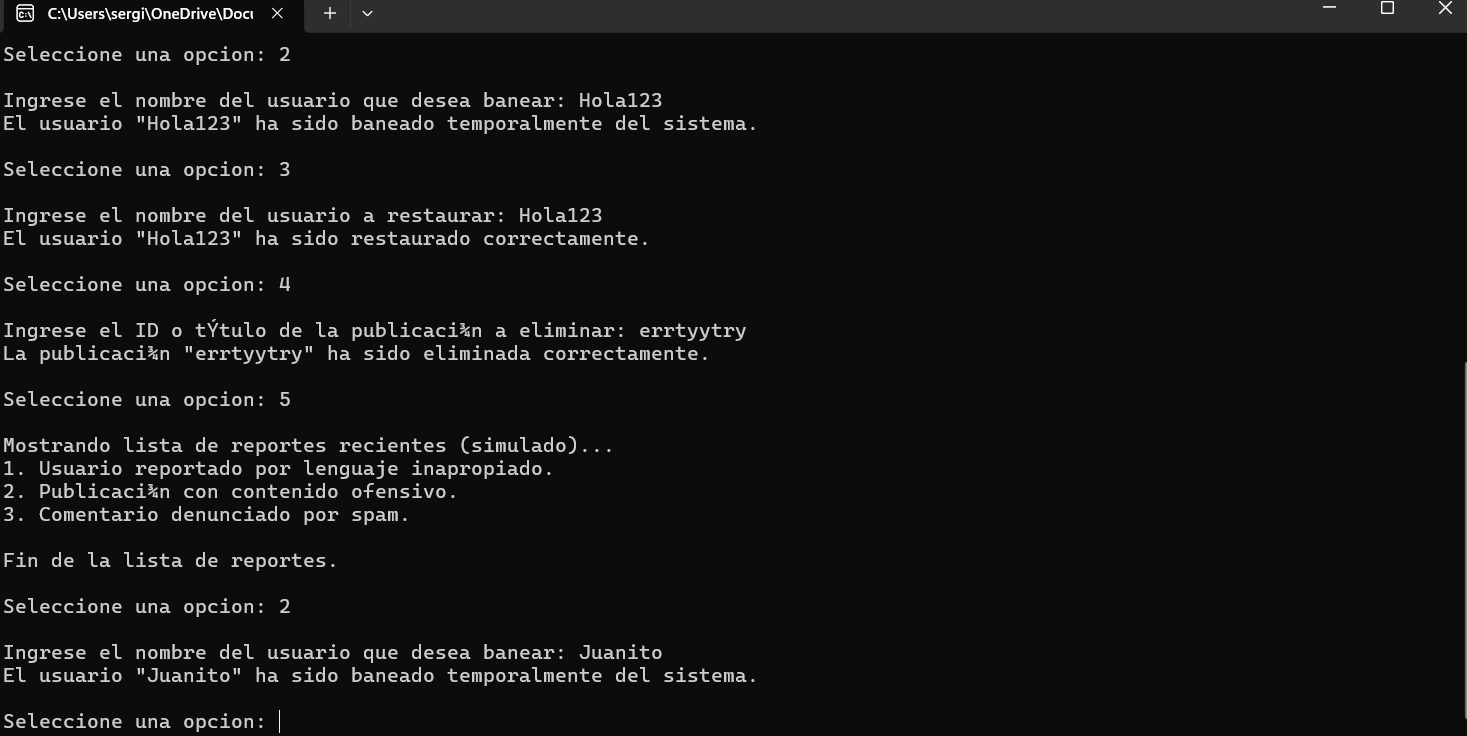


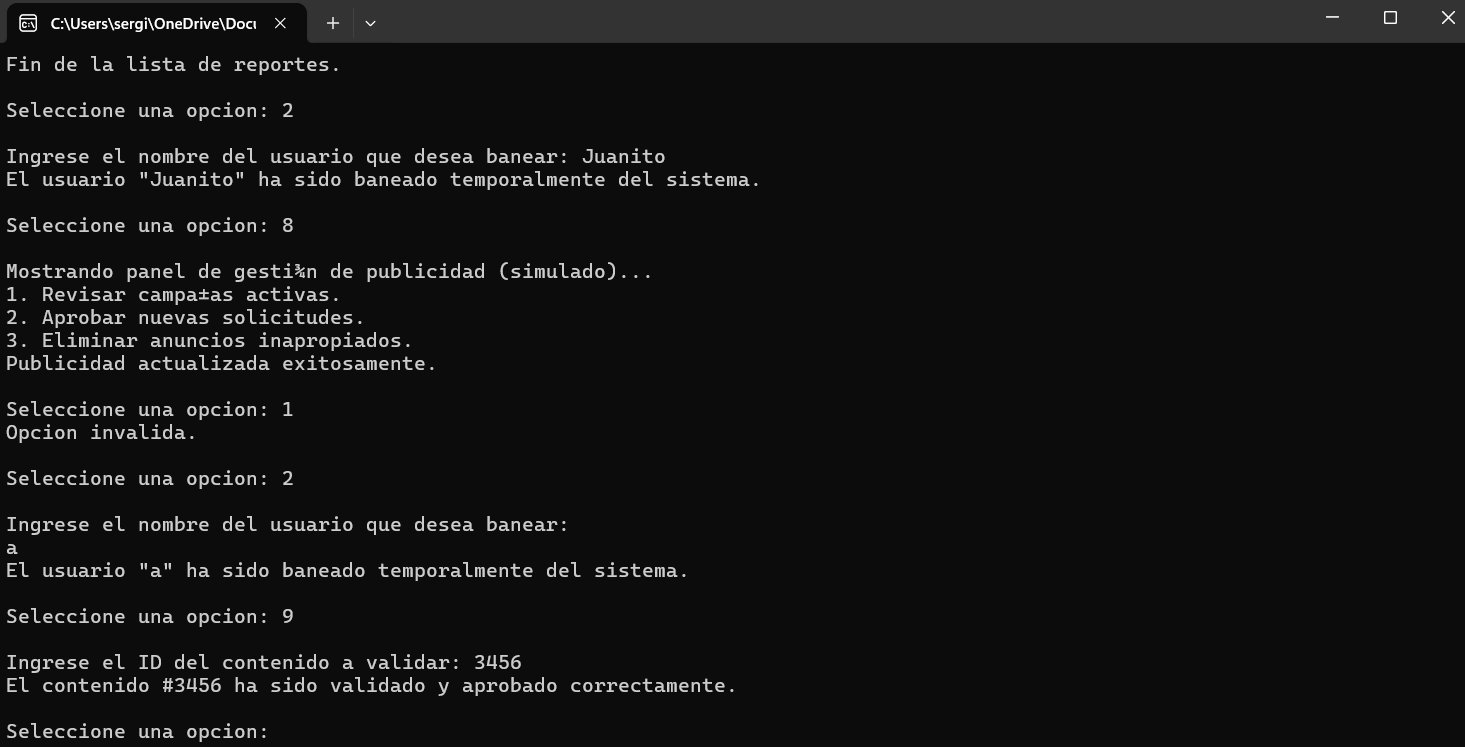


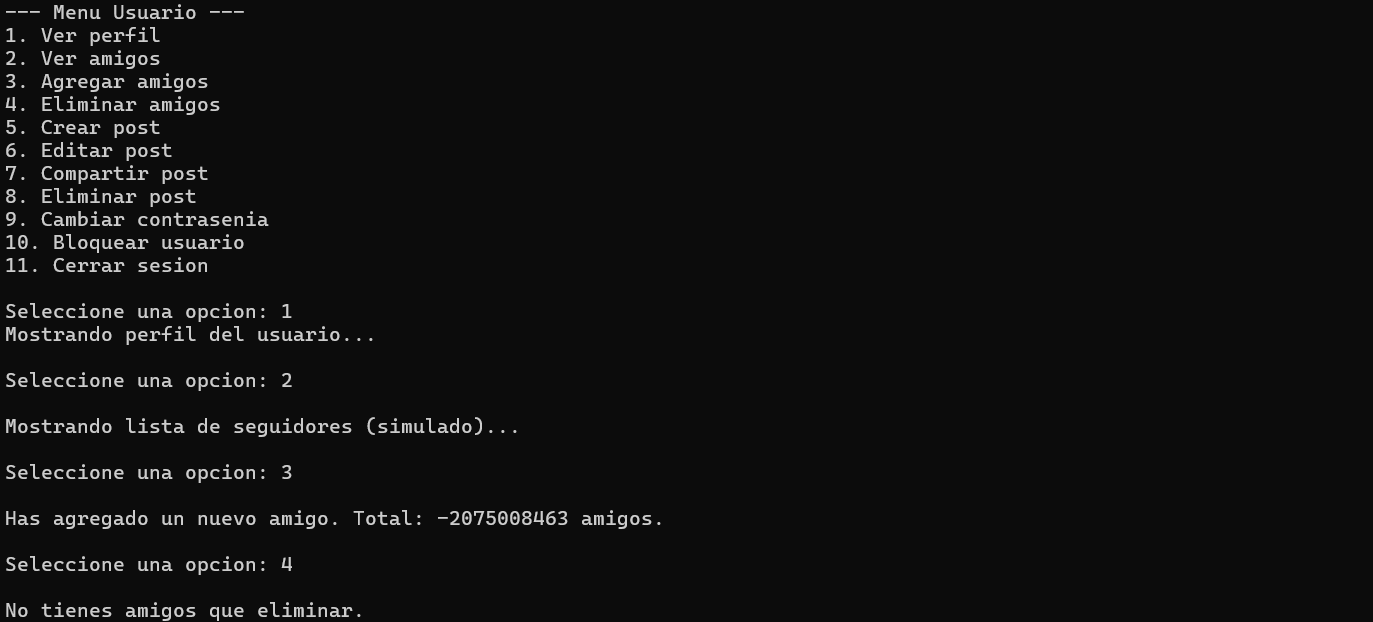










****

****

1. **Uso de archivos de texto y/o binarios**

Adjuntar código fuente, en donde se defina y se evidencie su uso de archivos de texto y/o binario para las principales clases.



1. **Enlace de video**

[**https://youtu.be/fNNyR6AUDZk**](https://youtu.be/fNNyR6AUDZk)

1. **Anexos**

* **Diagrama de clase:** <https://lucid.app/lucidchart/bf129e68-19c4-4ec7-8183-ca04606267f1/edit?viewport_loc=-5340%2C-955%2C4671%2C2005%2C0_0&invitationId=inv_b5619298-de60-4907-ac70-3c74c38dbd1a>

<https://github.com/JFabianSandoval/upc-pre-202502-1acc0182-1378-trabajo-grupo8/>

1. **Referencias Bibliográficas**

Universidad Peruana de Ciencias Aplicadas. (2025). *Guía académica del curso CC182 Algoritmos y Estructura de Datos*. Lima: UPC.

Visual Studio Documentation. (s.f.). *Using C++ in Visual Studio*. Recuperado el 15 de junio de 2025, de<https://learn.microsoft.com/en-us/cpp/?view=msvc-170>