UMAG Universidad de Magallanes | Facultad de Ingeniería

Informe de Simulación de una Red Social en Lenguaje C

Miguel Loaiza[†], **Ignacio Contreras**[†], **Felipe Paillacar**[†], **Johann Fink**[†] y **Benjamín Sanhueza**[†]

[†] *Universidad de Magallanes*

Este manuscrito fue compilado el: 15 de diciembre de 2024

Resumen

Este proyecto consiste en simular una red social con las características similares a una plataforma digital, en donde se implementaron funcionalidades tales como: Crear perfiles de usuarios y realizar publicaciones, como también poder recomendar usuarios basándose en afinidades o similitud de estos.
Al ejecutar el proyecto, este permite generar usuarios de forma aleatoria, asignando un identificador, nombre, género, edad, personalidad y hobbies. Luego de ello realiza recomendaciones para ellos, tales como mejores coincidencias siendo esta compatibilidad por edad, compatibilidad alta por mismo grupo y hobbies en común.
Otro de los aspectos relevantes a destacar es la entrega de un reporte con el grafo de conexiones entre ellos como también informar aquel usuario con más amigos en la red social simulado en el proyecto.
Finalmente queremos destacar como creatividad es la incorporación de diagramas de flujos para la claridad de la ejecución de los procesos como también la generación de un gráfico visual de conexiones entre usuarios dando una visión más practica de la conexión entre ellos.

Keywords: *Estructura de Datos, Simulación, Red Social, Jaccard, Dijkstra, Grafo, Arista, Vértice*

Índice

1 Contexto **3**

2 Introducción **3**

3 Objetivos **3**

3.1 Objetivos Generales . . . 3

3.2 Objetivos Específicos . . . 3

4 Datos de Entrada **4**

4.1 female_usernames.txt . . . 4

4.2 male_usernames.txt . . . 4

4.3 hobbies.txt . . . 4

4.4 personalities.txt . . . 4

4.5 post_templates.txt . . . 4

5 Estructuras Utilizadas **5**

5.1 Grafos . . . 5

5.2 Listas Enlazadas . . . 5

6 Algoritmos Utilizados **6**

6.1 Índice de Jaccard . . . 6

6.2 Dijkstra . . . 6

6.3 Quicksort . . . 6

7 Desarrollo del Proyecto **7**

7.1 Inicio y Funciones del Historial . . . 7

7.2 Generación de Usuarios Aleatorios . . . 8

7.3 Recomendación, Conexiones y Representación del Grafo . . . 9

7.4 Publicaciones, Usuario con más Amigos y Gráfico . . . 10

8 Directorios **11**

9 Salida del Programa **12**

10 Dificultades y Soluciones **16**

10.1 Asignación de Hobbies a los Usuarios 16

10.2 Historial 16

10.3 Algoritmo de búsqueda 16

10.4 Algoritmo de Dijkstra 16

11 Conclusión **17**

12 Contacto y Recursos **17**

1. Contexto

En la última década, las redes sociales han transformado la forma en que las personas interactúan, se comunican y comparten información. Plataformas como Twitter o Threads han revolucionado la conectividad entre personas, creando espacios virtuales donde millones de usuarios se relacionan en tiempo real.
Desde la perspectiva de la informática, estas plataformas presentan un desafío, debido a la complejidad de los algoritmos de optimización y la implementación de estructuras complejas que los componen. Comprender el funcionamiento interno de las redes sociales se basa en entender conceptos claves como los grafos, ya que estos permiten modelar relaciones entre los usuarios.
La construcción y mantenimiento de estas plataformas requieren conocimientos avanzados en informática, entre los que se encuentran el diseño de algoritmos eficientes y la gestión de sistemas a gran escala.

2. Introducción

Este proyecto tiene como objetivo desarrollar un programa que simule múltiples funcionalidades de una red social, diseñado para emular de manera simplificada el comportamiento de plataformas actuales como Blueskye, Truth Social entre otras.
El simulador generará perfiles de usuarios de manera aleatoria, asignándoles características como hobbies, edad y un tipo de personalidad. En base a esta información, el sistema evaluará las afinidades y similitudes entre los usuarios para generar recomendaciones de conexión entre estos (amistad).
El proceso de recomendación se realizará en base a criterios como la utilización del índice de Jaccard, el cual mide la similitud entre los conjuntos de datos asociados a los usuarios. Aquellos usuarios con mayores niveles de afinidad serán sugeridos como posibles amigos. Una vez realizadas las recomendaciones, el simulador determinará qué usuarios establecen dichas conexiones. Estas amistades serán representadas gráficamente mediante las aristas que unen los vértices de un grafo, simbolizando las relaciones.
Además de abordar los aspectos técnicos, este proyecto busca demostrar cómo las conexiones sociales pueden modelarse utilizando algoritmos y estructuras, ofreciendo de esta manera un enfoque práctico para comprender la dinámica de las redes sociales.

3. Objetivos

El presente proyecto tiene como finalidad consolidar los conocimientos adquiridos durante el curso de estructuras de datos mediante el desarrollo de un sistema práctico que simula una red social.

3.1. Objetivos Generales

Diseñar e implementar un simulador de red social en lenguaje C que utilice diversas estructuras de datos vistas anteriormente, promoviendo su comprensión y aplicación práctica.

3.2. Objetivos Específicos

- Implementar y aplicar estructuras de datos como grafos y listas enlazadas.
  Desarrollar las estructuras de datos necesarias para representar las relaciones entre los usuarios y manejar las listas de atributos.

- Generar perfiles de usuarios de manera aleatoria con atributos como hobbies, personalidad y edad.
  Crear un modulo que cree perfiles con atributos asignados aleatoriamente, simulando la variedad de usuarios en una red real.

- Calcular afinidades entre usuarios mediante el índice de Jaccard para recomendar posibles conexiones.
  Crear una función que mida la similitud entre diversos usuarios en base a sus atributos.

- Establecer conexiones de amistad entre usuarios, representadas como aristas en un grafo.
  Representar las relaciones entre usuarios como aristas de un grafo, indicando las conexiones entre los nodos.

- Simular la publicación de contenido por parte de los usuarios dentro de la red social.
  Implementar un sistema que permita a los usuarios publicar mensajes o contenido en la red social.

- Utilizar el algoritmo de Dijkstra para realizar búsquedas eficientes en la red social.
  Adaptar el algoritmo de Dijkstra para encontrar relaciones relevantes dentro del grafo que representa la relación de los usuarios.

- Diseñar un sistema de almacenamiento que permita guardar y recuperar datos de usuarios anteriormente creados.
  Implementar un sistema que permita que los datos de los usuarios sean persistentes y se pueden cargar posteriormente.

4. Datos de Entrada

Este proyecto considera dos tipos principales de entradas: los archivos de texto, que proporcionan datos esenciales para la simulación, y el número de usuarios a generar, especificado como un parámetro al ejecutar el programa. El límite máximo de usuarios que se pueden crear es de 50. Los archivos de entrada son necesarios para crear los perfiles de usuario de manera aleatoria, y sus contenidos están organizados de la siguiente forma:

4.1. female_usernames.txt

Este archivo contiene una lista de 100 nombres femeninos generados de forma aleatoria. Cuando se genera un usuario con género femenino, se selecciona uno de estos nombres al azar para asignárselo, garantizando así la diversidad de los nombres.

4.2. male_usernames.txt

De manera similar al archivo de nombres femeninos, este archivo contiene 100 nombres masculinos generados aleatoriamente. Estos nombres se utilizan para asignar identidades a los usuarios masculinos creados durante la simulación.

4.3. hobbies.txt

Este archivo incluye una lista de los 10 hobbies más populares entre las personas. Durante la creación de un perfil de usuario, se seleccionan uno o más hobbies al azar, asegurando que cada usuario tenga un conjunto único de intereses. Aunque la cantidad y los tipos de hobbies son aleatorios, no se permiten duplicados dentro del mismo perfil.

4.4. personalities.txt

Basado en el Indicador de Tipos de Myers-Briggs (MBTI), este archivo contiene los 16 tipos de personalidad clasificados en cuatro grupos principales: Analistas, Diplomáticos, Centinelas y Exploradores. Cada personalidad tiene características únicas y es asignada aleatoriamente a los usuarios. Este enfoque permite representar de manera más realista la diversidad en las personalidades humanas.

4.5. post_templates.txt

Este archivo incluye una colección de plantillas de publicaciones que simulan el contenido generado por los usuarios. Durante la ejecución del programa, un usuario seleccionado al azar elige una de estas plantillas para simular la creación de una publicación en la red social, enriqueciendo la dinámica del sistema.

En conjunto, estos archivos no solo proporcionan los datos necesarios para la generación de perfiles, sino que también permiten simular comportamientos y características que emulan las interacciones sociales reales de manera simplificada.

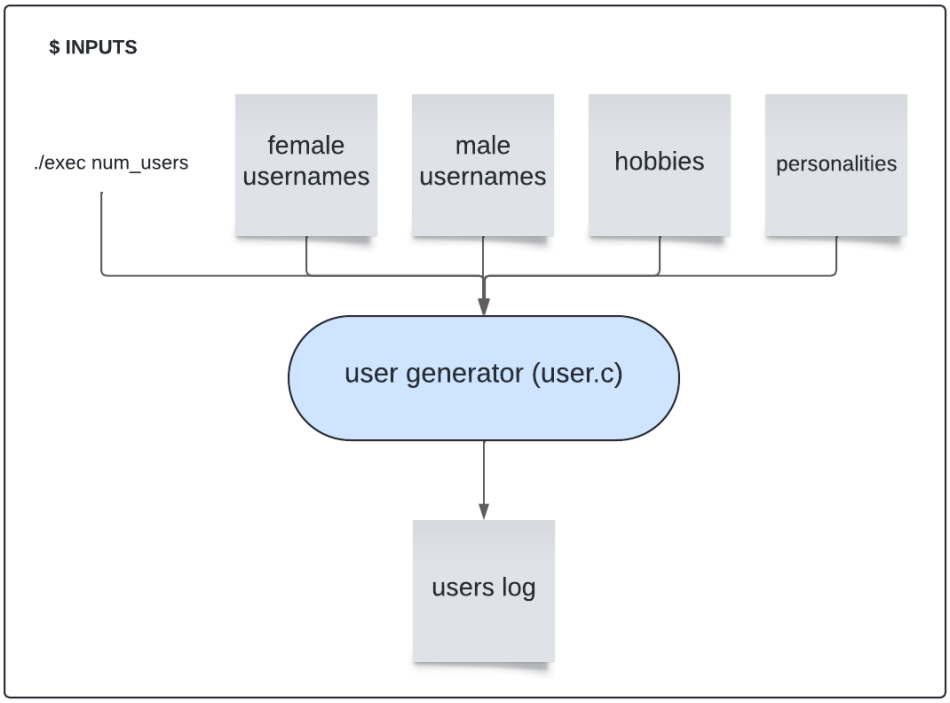

Figura 1. Textos de entrada y Cantidad de usuarios.

5. Estructuras Utilizadas

Las estructuras de datos son fundamentales en la informática, ya que permiten organizar y manipular la información de manera eficiente. La elección adecuada de una estructura depende del problema que se desea resolver y de las operaciones que se necesiten realizar. En este proyecto, se utilizaron estructuras de datos para representar relaciones, gestionar información dinámica y optimizar recursos.
En este apartado se describen las estructuras de datos implementadas en el proyecto, junto con una explicación general sobre su propósito y una breve justificación de su uso. Estas estructuras fueron seleccionadas cuidadosamente para cumplir con los requerimientos del sistema y optimizar el desempeño del simulador.

5.1. Grafos

Esta estructura fue seleccionada porque refleja de manera directa y eficiente el concepto de una red social. Permite modelar las interacciones entre usuarios y realizar un análisis sobre la conectividad, como la búsqueda de rutas cortas o la identificación de grupos conectados. Además, los grafos ofrecen flexibilidad para representar tanto redes densas como dispersas, adaptándose a diferentes escenarios.

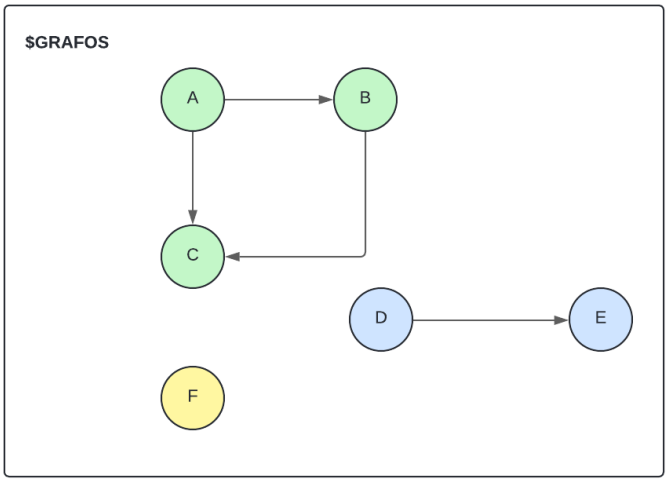

Figura 2. Ejemplo de los Grafos

5.2. Listas Enlazadas

El uso de listas enlazadas permite un crecimiento dinámico, adaptándose a la cantidad de publicaciones que se generan durante la simulación. Esta estructura es ideal en este caso porque optimiza el uso de memoria, ya que solo se reserva el espacio necesario en cada momento. Además, facilita la eliminación de publicaciones antiguas o innecesarias, asegurando un manejo eficiente de los recursos.

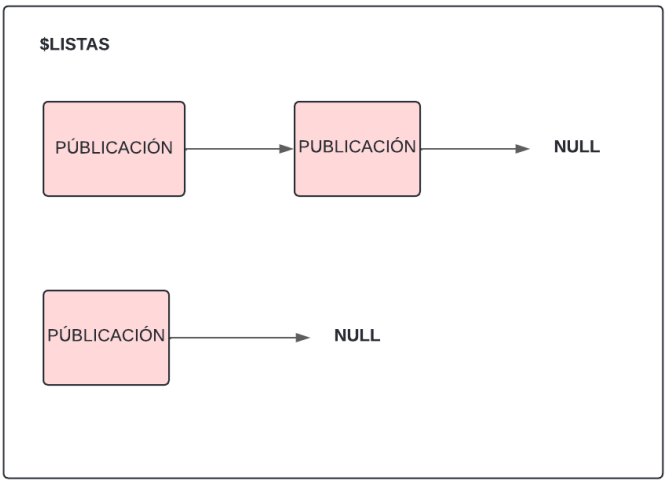

Figura 3. Ejemplo de las Listas

6. Algoritmos Utilizados

Los algoritmos son el núcleo de cualquier solución informática, ya que dictan cómo se procesan los datos para obtener los resultados deseados. La elección del algoritmo adecuado depende de la naturaleza del problema.
En este apartado se describen los algoritmos implementados en el proyecto, junto con una breve explicación de su propósito y una justificación para su selección. Los algoritmos fueron elegidos para resolver problemas específicos dentro del sistema, optimizando tanto la precisión como la eficiencia en cada caso.

6.1. Índice de Jaccard

El índice de Jaccard es una métrica que mide la similitud entre dos conjuntos, calculando la proporción entre la intersección y la unión de dichos conjuntos. En este proyecto se utiliza para determinar el grado de compatibilidad de los usuarios, en base a los atributos de cada uno (hobbies, personalidades, edad, etc).
Este algoritmo fue seleccionado por su simplicidad y efectividad para comparar conjuntos. Es útil en el contexto de una red social, donde se busca identificar a los usuarios con intereses comunes de manera rápida y clara.
La fórmula matemática para calcular el índice de Jaccard es la siguiente:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Donde:

- A y B son conjuntos que representan los atributos comparados (por ejemplo, hobbies).
- $|A \cap B|$ es el número de elementos comunes entre A y B.
- $|A \cup B|$ es el número total de elementos únicos entre A y B.

6.2. Dijkstra

El algoritmo de Dijkstra se utiliza para encontrar las rutas más cortas entre los nodos de un grafo. En este proyecto, se emplea para mostrar la representación del grafo de adyacencia y determinar las conexiones más cercanas entre los usuarios.
Dijkstra es un algoritmo eficiente y reconocido para resolver problemas de caminos más cortos. Fue elegido por su capacidad para trabajar con grafos y su relevancia en el análisis de redes sociales, donde es fundamental para identificar usuarios cercanos y explorar relaciones.
La ecuación utilizada en el proceso del algoritmo de Dijkstra es:

$$d[v] = min(d[v], d[u] + w(u,v))$$

Donde:

- $d[v]$ es la distancia más corta conocida al nodo v.
- $d[u]$ es la distancia más corta conocida al nodo u.
- $w(u,v)$ es el peso de la arista entre u y v.

6.3. Quicksort

Quicksort es un algoritmo de ordenamiento eficiente basado en la técnica de "divide y vencerás". Se utiliza para organizar los resultados de similitud entre usuarios en orden descendente, priorizando las relaciones más relevantes.
Se seleccionó Quicksort debido a su veloz rendimiento, el cual es superior al de otros algoritmos de ordenamiento $O(n \log n)$ en la mayoría de casos. La implementación es relativamente sencilla y es adecuada para ordenar grandes conjuntos de datos rápidamente. En el algoritmo Quicksort, se define como:

$$A_{\text{izq}} = \{x \in A \mid x \leq p\}, \quad A_{\text{der}} = \{x \in A \mid x > p\}$$

Donde:

- A_{izq} contiene los elementos menores o iguales al pivote p.
- A_{der} contiene los elementos mayores al pivote p.

7. Desarrollo del Proyecto

En este apartado se detalla el desarrollo del proyecto, abordando los pasos más importantes para la implementación de la red social.

7.1. Inicio y Funciones del Historial

El programa comienza solicitando al usuario la cantidad de perfiles que desea generar. Para manejar los argumentos de entrada y evitar errores en la ejecución, se utiliza la función getopt, la que valida los parámetros ingresados por la terminal. Si el número de usuarios especificado supera el límite definido por los desarrolladores (50 usuarios) o es menor al mínimo admitido (1 usuario), el programa detiene su ejecución y muestra un mensaje de error. En caso contrario, continúa con la carga de los archivos de texto de entrada, los que contienen la información necesaria para la generación aleatoria de usuarios.

Una vez cargados los archivos, el programa verifica si existe un historial previo mediante la función log_check(). Esta función es fundamental para determinar si existe un archivo de historial previo. La función comienza con la lectura del primer carácter del archivo y crea una variable para almacenar un valor entero, si el archivo está vacío, el dato almacenado pasa a ser 0 (historial vacío), si el valor es 1, significa que el historial contiene datos y por ende llama a la función user_count_from_log() para determinar la cantidad de usuarios previamente registrados y poder continuar con la numeración de forma correcta. Posteriormente, se llama a la función log_input(), función utilizada para guardar la información de usuarios ya existentes en el historial.

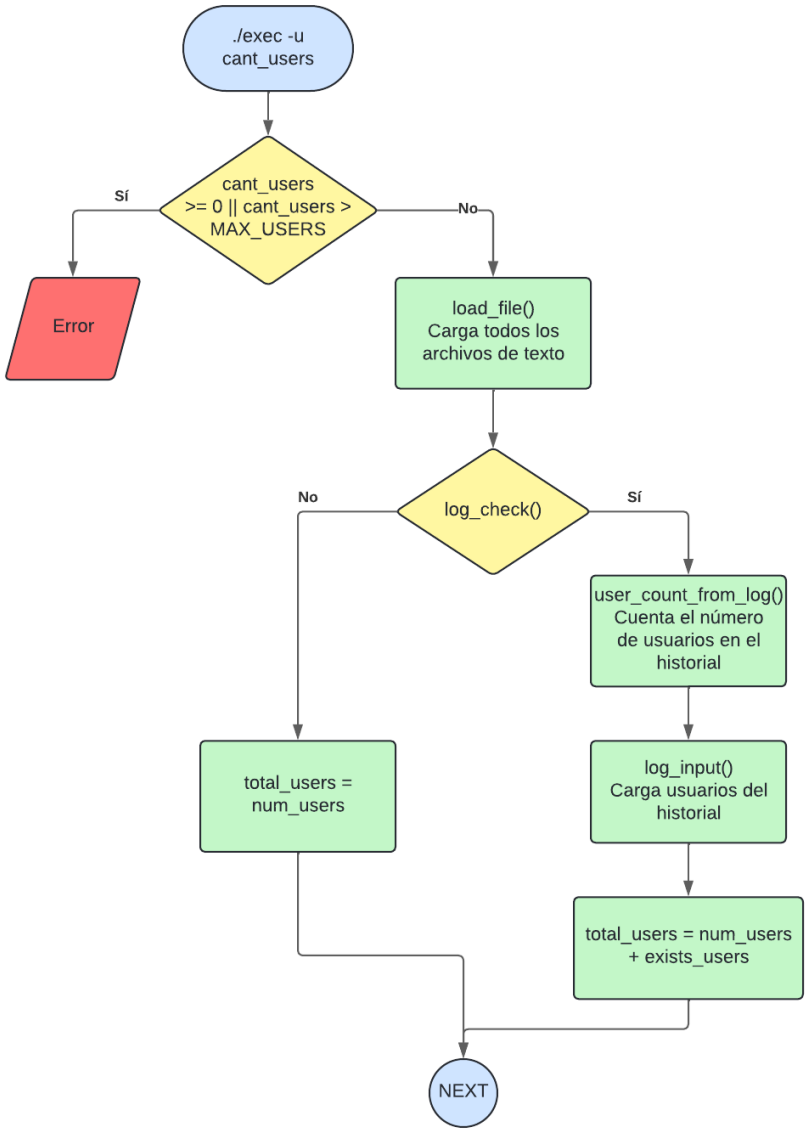

Figura 4. Diagrama del inicio y funcionamiento del historial

7.2. Generación de Usuarios Aleatorios

Tras completar la primera etapa del programa, se realiza una verificación crucial: comprobar si la cantidad de usuarios solicitada excede el límite establecido por los desarrolladores (50 usuarios). Si este límite es superado, el programa detiene su ejecución y muestra un mensaje de error por consola, informando al usuario sobre la infracción. Este control asegura que el sistema opere dentro de los parámetros definidos, evitando sobrecargas o comportamientos inesperados. Por el contrario, si el número de usuarios está dentro del rango permitido, el programa continúa con la ejecución normal y llama a la función generate_random_users().

La función generate_random_users() es responsable de crear el número de usuarios solicitado de manera completamente aleatoria. Para lograrlo, esta función se apoya de otras dos funciones: generate_random_hobbies() y generate_random_personalities(). Además, realiza pasos previos para establecer atributos básicos de los usuarios, como el género, la edad y el nombre.

El género se elige de forma aleatoria, siendo posible que el usuario sea masculino o femenino, luego se asigna un valor aleatorio mayor o igual a 18 años, ya que este es el límite inferior definido para los usuarios en el sistema. Dependiendo del género asignado, se selecciona un archivo de texto diferente (male_usernames.txt o female_usernames.txt). A partir de este archivo, se elige un nombre aleatorio para cada usuario.

La primera función llamada es generate_random_hobbies(). Su objetivo es asignar un conjunto de hobbies (pasatiempos) al usuario (escogidos del archivo de texto hobbies.txt), se determina aleatoriamente la cantidad de hobbies para el usuario, asegurando que sea menor o igual al máximo permitido. Los hobbies son seleccionados de manera aleatoria y no se pueden repetir dentro de un mismo usuario.

La siguiente función en ser llamada es generate_random_personalities(), que se encarga de seleccionar la personalidad del usuario, para esto, se utiliza un archivo de texto que contiene una lista de 16 personalidades predefinidas y se selecciona aleatoriamente una personalidad para asignarla al usuario.

Al completar estos pasos, cada usuario cuenta con una serie de atributos básicos generados de manera aleatoria: género, edad, nombre, hobbies y personalidad. Estos atributos son almacenados en estructuras predefinidas que representan a los usuarios en el programa, permitiendo su posterior utilización en las operaciones de la red social, como la creación de conexiones y el cálculo de similitudes.

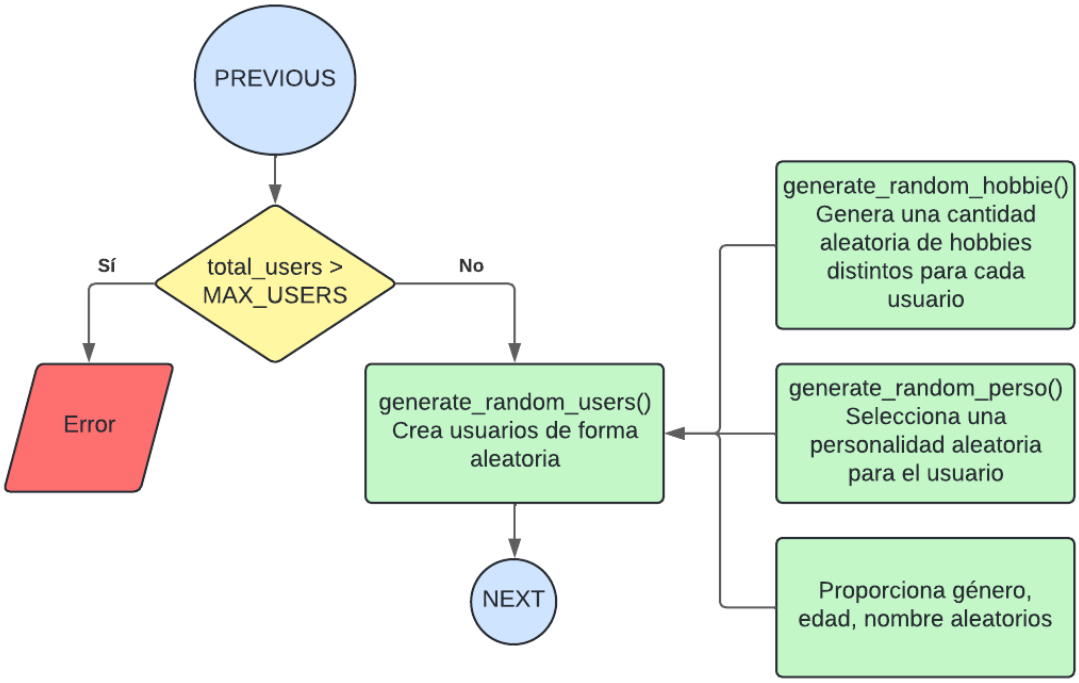

Figura 5. Diagrama del funcionamiento de la generación de usuarios aleatoria

7.3. Recomendación, Conexiones y Representación del Grafo

Después de la creación aleatoria de los usuarios, el programa procede a generar recomendaciones de amistad, establecer conexiones entre usuarios y construir un grafo que represente visualmente la red social en la terminal.

El primer paso es la ejecución de la función recommend_users(), que se encarga de identificar las mejores tres recomendaciones de amistad para cada usuario. Dentro de esta función se llama a otra llamada calculate_jaccard_similarity(), esta función utiliza el índice de similitud de Jaccard para comparar los hobbies de cada par de usuarios. El índice se calcula como el cociente entre el tamaño de la intersección y el tamaño de la unión de los conjuntos de hobbies de los dos usuarios. Esto genera un valor inicial que refleja qué tan similares son los usuarios en base a sus intereses.

Una vez calculado el índice de similitud inicial, este se ajusta considerando otros factores como la edad y la personalidad, de la siguiente manera:

Si las personalidades de los usuarios pertenecen al mismo grupo, el índice de similitud se incrementa.

Si existe una diferencia significativa en la edad entre los usuarios, el índice se modifica en consecuencia, ya que una mayor diferencia puede reducir la afinidad potencial.

El resultado final es un valor entre 0 y 1, donde un valor cercano a 1 indica una mayor similitud entre los usuarios. Una vez obtenidos los valores de similitud, se ordenan las recomendaciones para cada usuario en orden descendente utilizando la función quicksort(), asegurando que los tres usuarios con mayor similitud sean sugeridos como posibles amistades.

Con las recomendaciones calculadas, se ejecuta la función create_connections(), encargada de establecer conexiones de amistad entre los usuarios. Para cada par de usuarios, se compara su índice de similitud con un umbral predefinido de 0.3. Si el índice es igual o superior a este valor, se considera que los usuarios tienen suficiente compatibilidad para formar una amistad, y se crea una conexión en el grafo.

Finalmente, se llama a la función display_graph() para construir y mostrar la representación de las conexiones. Además de la representación visual, se implementa el algoritmo de Dijkstra para calcular los caminos más cortos desde un nodo fuente hacia los demás nodos del grafo. Este algoritmo itera sobre todos los nodos para determinar las distancias mínimas y almacena esta información en un arreglo llamado distance.

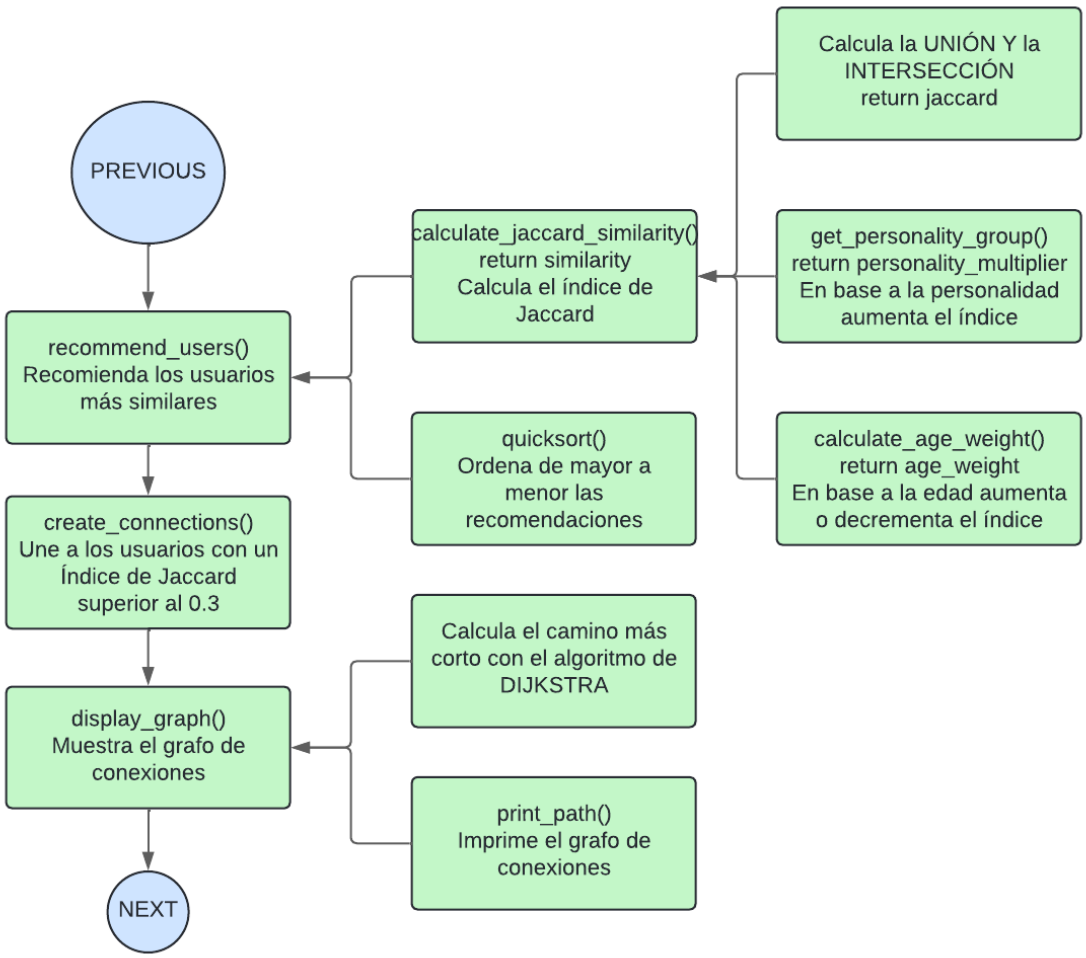

Figura 6. Diagrama del funcionamiento de la recomendación de usuarios, la creación de conexiones y la muestra por terminal del grafo

7.4. Publicaciones, Usuario con más Amigos y Gráfico

Después de la creación del grafo y las conexiones entre los usuarios, el programa avanza hacia la simulación de publicaciones por usuarios aleatorios. Este proceso se realiza mediante las funciones generate_random_post() y display_all_post(). La primera es fundamental para la creación aleatoria de publicaciones y hace uso de dos funciones clave: load_post_templates() y publish_post().

La función load_post_templates() tiene como propósito cargar un archivo de texto que contiene una variedad de plantillas predefinidas de publicaciones. Estas plantillas actúan como base para generar nuevos contenidos mediante la función create_new_post(), que selecciona una plantilla existente de manera aleatoria. Una vez creada la publicación, la función publish_post() se encarga de publicarla, asociándola al usuario correspondiente y dejándola registrada en el sistema para su posterior visualización.

Tras completar la etapa de publicaciones, se llama a la función find_user_with_most_friends(), diseñada para recorrer la lista de usuarios, contar sus conexiones y determinar cuál de ellos tiene la mayor cantidad de amistades. Este resultado se muestra en la terminal como una característica adicional del programa, que aporta una visión interesante sobre la dinámica de las conexiones sociales simuladas.

El programa termina con la generación de una representación gráfica del grafo de conexiones. Para ello, se utiliza una función que genera una imagen en formato EPS, donde cada nodo representa un usuario y cada arista refleja una amistad. Posteriormente, la función transform_eps_to_png() se encarga de convertir el archivo EPS a un formato PNG más accesible, eliminando el archivo EPS original.

Por último, antes de finalizar la ejecución del programa, se liberan cuidadosamente los punteros asociados al grafo y las publicaciones. Esto garantiza una adecuada gestión de memoria, previniendo fugas y asegurando que todos los recursos utilizados durante la simulación sean correctamente liberados. Este paso marca el cierre del programa, dejando como resultado una simulación funcional y bien estructurada de una red social.

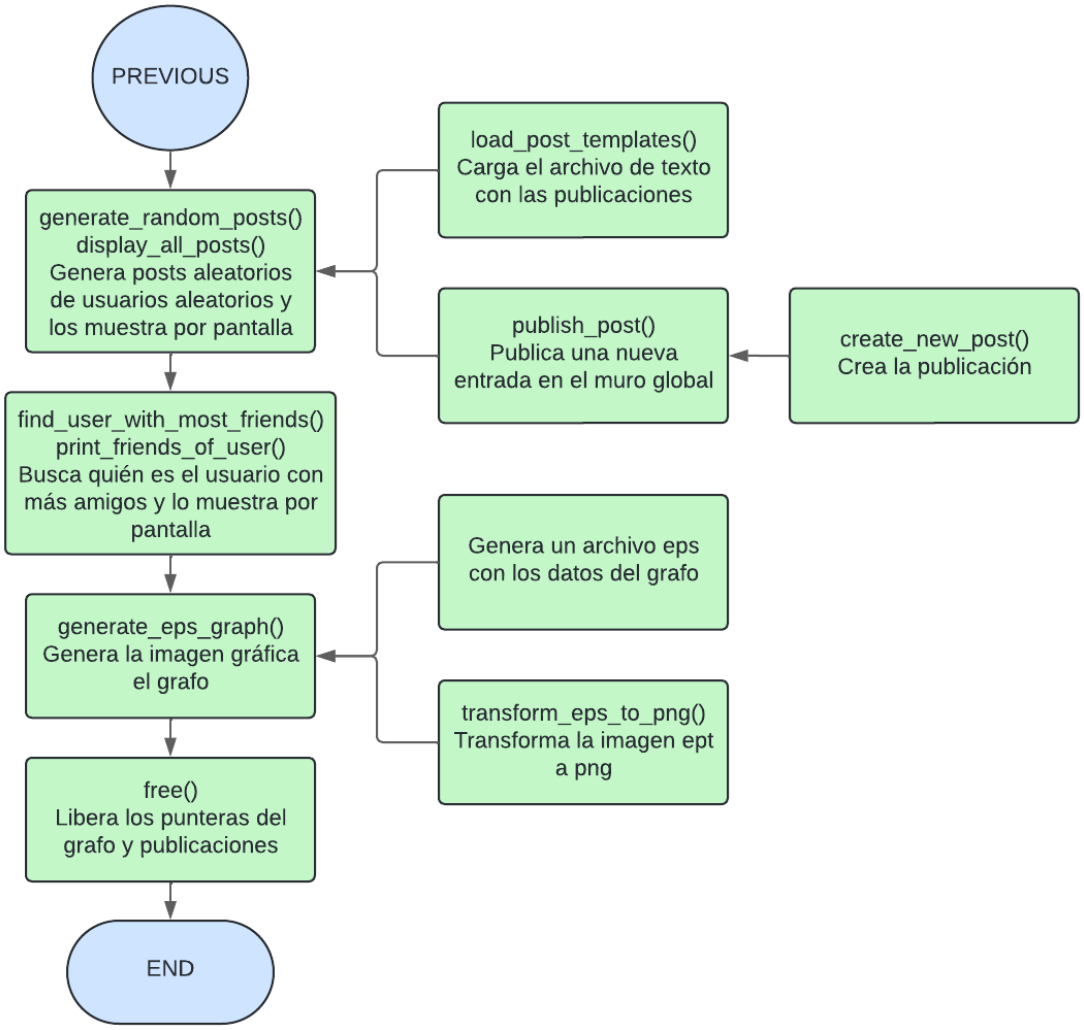

Figura 7. Diagrama del funcionamiento de las publicaciones, la búsqueda del usuario con más amigos y la generación de imagen PNG del grafo de conexiones

8. Directorios

Desde el inicio del desarrollo, se procuró organizar los directorios del proyecto de manera estructurada y eficiente, con el objetivo de mantener una clara separación entre los componentes, facilitar la navegación por el código y agilizar la identificación y resolución de problemas. La distribución final de los directorios es la siguiente:

```
grupo_6
     incs
        graph.h
        log.h
        main.h
        posts.h
        similarity.h
        users.h
     input
        female_usernames.txt
        male_usernames.txt
        hobbies.txt
        personalities.txt
        post_templates.txt
        users_log.txt
     output
         social_network.png
     src
         connections_graph.c
         graphic.c
         main.c
         parameters.c
         post.c
         search.c
         similarity.c
         user_log.c
         user.c
```

9. Salida del Programa

En este proyecto, la salida (output) generada por el sistema puede visualizarse de dos formas principales, proporcionando diferentes perspectivas según las necesidades del usuario.
La primera forma de visualizar los resultados es a través de la terminal, al ejecutar el proyecto. En esta modalidad, se imprime en pantalla un registro detallado de los procesos realizados durante la ejecución.
Esta salida en la terminal es especialmente útil para depuración, análisis técnico o para usuarios interesados en examinar el funcionamiento interno del programa de forma detallada y secuencial.
La segunda forma de visualizar los resultados es mediante una representación gráfica más intuitiva y visual, destinada a facilitar la comprensión de las relaciones entre usuarios en la red social.
Esta representación gráfica es ideal para usuarios que prefieren interpretar los datos de manera visual o para presentaciones en las que sea necesario explicar la estructura y las relaciones de la red social de una forma más accesible y atractiva.

La salida por terminal es la siguiente:

```
Usuarios generados:
Usuario 1:
ID: 1
Nombre: Nuria
G nero: Femenino
Edad: 75
Personalidad: ESTJA
Hobbies:
 - Bailar
 - Peliculas
 - Videojuegos
 - Escribir
 - Leer
 - Viajar
 - Musica
Usuario 2:
ID: 2
Nombre: Arturo
G nero: Masculino
Edad: 37
Personalidad: ISTJA
Hobbies:
 - Bailar
 - Videojuegos
Usuario 3:
ID: 3
Nombre: Fernando
G nero: Masculino
Edad: 18
Personalidad: ESTJA
Hobbies:
 - Videojuegos
 - Deporte
 - Escribir
 - Musica
 - Viajar
Usuario 4:
ID: 4
Nombre: Emilio
G nero: Masculino
Edad: 74
Personalidad: INTPA
Hobbies:
 - Videojuegos
 - Deporte
 - Cocinar
 - Escribir
 - Leer
 - Musica
Usuario 5:
ID: 5
Nombre: Victor
G nero: Masculino
Edad: 77
Personalidad: ISFJA
Hobbies:
 - Peliculas
 - Bailar
```

```
 - Escribir
 - Videojuegos
 - Deporte
 - Leer
```

Código 1. Output del terminal

```
Recomendaciones para los Usuarios:
Usuario 1 (Nuria, 75 a os):
Mejores coincidencias:
1. Nombre: Victor (77 a os)
   - Similitud total: 0.50
   - Compatibilidad por edad: Excelente (diferencia 2 a os)
   - Compatibilidad BAJA por grupos diferentes: Nuria: Centinelas (Conservadores) - Victor: Exploradores (Art sticos)
   - Hobbies en com n: Peliculas, Bailar, Escribir, Videojuegos, Leer
2. Nombre: Emilio (74 a os)
   - Similitud total: 0.36
   - Compatibilidad por edad: Excelente (diferencia 1 a os)
   - Compatibilidad BAJA por grupos diferentes: Nuria: Centinelas (Conservadores) - Emilio: Analistas (Racionales)
   - Hobbies en com n: Videojuegos, Escribir, Leer, Musica
3. Nombre: Fernando (18 a os)
   - Similitud total: 0.12
   - Compatibilidad por edad: Muy baja (diferencia 57 a os)
   - Compatibilidad alta por mismo grupo: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos, Escribir, Musica, Viajar
Usuario 2 (Arturo, 37 a os):
Mejores coincidencias:
1. Nombre: Fernando (18 a os)
   - Similitud total: 0.08
   - Compatibilidad por edad: Baja (diferencia 19 a os)
   - Compatibilidad alta por mismo grupo: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos
2. Nombre: Nuria (75 a os)
   - Similitud total: 0.07
   - Compatibilidad por edad: Muy baja (diferencia 38 a os)
   - Compatibilidad alta por mismo grupo: Centinelas (Conservadores)
   - Hobbies en com n: Bailar, Videojuegos
3. Nombre: Victor (77 a os)
   - Similitud total: 0.05
   - Compatibilidad por edad: Muy baja (diferencia 40 a os)
   - Compatibilidad BAJA por grupos diferentes: Arturo: Centinelas (Conservadores) - Victor: Exploradores (Art sticos)
   - Hobbies en com n: Bailar, Videojuegos
Usuario 3 (Fernando, 18 a os):
Mejores coincidencias:
1. Nombre: Nuria (75 a os)
   - Similitud total: 0.12
   - Compatibilidad por edad: Muy baja (diferencia 57 a os)
   - Compatibilidad alta por mismo grupo: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos, Escribir, Viajar
2. Nombre: Emilio (74 a os)
   - Similitud total: 0.09
   - Compatibilidad por edad: Muy baja (diferencia 56 a os)
   - Compatibilidad BAJA por grupos diferentes: Fernando: Centinelas (Conservadores) - Emilio: Analistas (Racionales)
   - Hobbies en com n: Videojuegos, Deporte, Escribir, Musica
3. Nombre: Arturo (37 a os)
   - Similitud total: 0.08
   - Compatibilidad por edad: Baja (diferencia 19 a os)
   - Compatibilidad alta por mismo grupo: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos
Usuario 4 (Emilio, 74 a os):
Mejores coincidencias:
1. Nombre: Victor (77 a os)
   - Similitud total: 0.40
   - Compatibilidad por edad: Excelente (diferencia 3 a os)
   - Compatibilidad BAJA por grupos diferentes: Emilio: Analistas (Racionales) - Victor: Exploradores (Art sticos)
   - Hobbies en com n: Escribir, Videojuegos, Deporte, Leer
2. Nombre: Nuria (75 a os)
   - Similitud total: 0.36
   - Compatibilidad por edad: Excelente (diferencia 1 a os)
```

```
   - Compatibilidad BAJA por grupos diferentes: Emilio: Analistas (Racionales) - Nuria: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos, Escribir, Leer
3. Nombre: Fernando (18 a os)
   - Similitud total: 0.09
   - Compatibilidad por edad: Muy baja (diferencia 56 a os)
   - Compatibilidad BAJA por grupos diferentes: Emilio: Analistas (Racionales) - Fernando: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos, Deporte, Escribir, Musica
Usuario 5 (Victor, 77 a os):
Mejores coincidencias:
1. Nombre: Nuria (75 a os)
   - Similitud total: 0.50
   - Compatibilidad por edad: Excelente (diferencia 2 a os)
   - Compatibilidad BAJA por grupos diferentes: Victor: Exploradores (Art sticos) - Nuria: Centinelas (Conservadores)
   - Hobbies en com n: Bailar, Peliculas, Videojuegos, Escribir, Leer
2. Nombre: Emilio (74 a os)
   - Similitud total: 0.40
   - Compatibilidad por edad: Excelente (diferencia 3 a os)
   - Compatibilidad BAJA por grupos diferentes: Victor: Exploradores (Art sticos) - Emilio: Analistas (Racionales)
   - Hobbies en com n: Videojuegos, Deporte, Escribir, Leer
3. Nombre: Fernando (18 a os)
   - Similitud total: 0.06
   - Compatibilidad por edad: Muy baja (diferencia 59 a os)
   - Compatibilidad BAJA por grupos diferentes: Victor: Exploradores (Art sticos) - Fernando: Centinelas (Conservadores)
   - Hobbies en com n: Videojuegos, Deporte, Escribir
```

Código 2. Output del terminal

```
Conexiones creadas:
Conectando a los usuarios Nuria y Emilio ( ndice  de Jaccard: 0.36)
Conectando a los usuarios Nuria y Victor ( ndice  de Jaccard: 0.50)
Conectando a los usuarios Emilio y Victor ( ndice  de Jaccard: 0.40)

Grafo de Conexiones:
Nuria     : Distancia: 1 , Camino: Nuria -> Emilio
Arturo    : Distancia: 0 , Camino: Arturo
Fernando  : Distancia: 0 , Camino: Fernando
Emilio    : Distancia: 1 , Camino: Emilio -> Nuria
Victor    : Distancia: 1 , Camino: Victor -> Nuria

Publicaciones:
Publicaci n 3
Usuario: Fernando (ID: 3)
Contenido: Es un excelente  d a para maratonear pel culas , me encanta descubrir nuevas historias.
Fecha: 2024-12-06 07:10:13
Publicaci n 2
Usuario: Nuria (ID: 1)
Contenido: Acabo de terminar un libro incre ble y ya quiero empezar el siguiente
Fecha: 2024-12-07 02:35:19
Publicaci n 1
Usuario: Victor (ID: 5)
Contenido: No puedo evitar moverme cuando suena mi canci n favorita, hoy es un gran  d a para bailar.
Fecha: 2024-12-11 09:14:36

Usuario con  m  s  amigos:
El usuario con  m  s  amigos es: Nuria con 2 amigos.
Amigos de Nuria:
- Victor
- Emilio

La imagen del grafo ha sido guardado en la ruta: ./output/social_network.png.
```

Código 3. Output del terminal

La salida gráfica del proyecto es la siguiente:

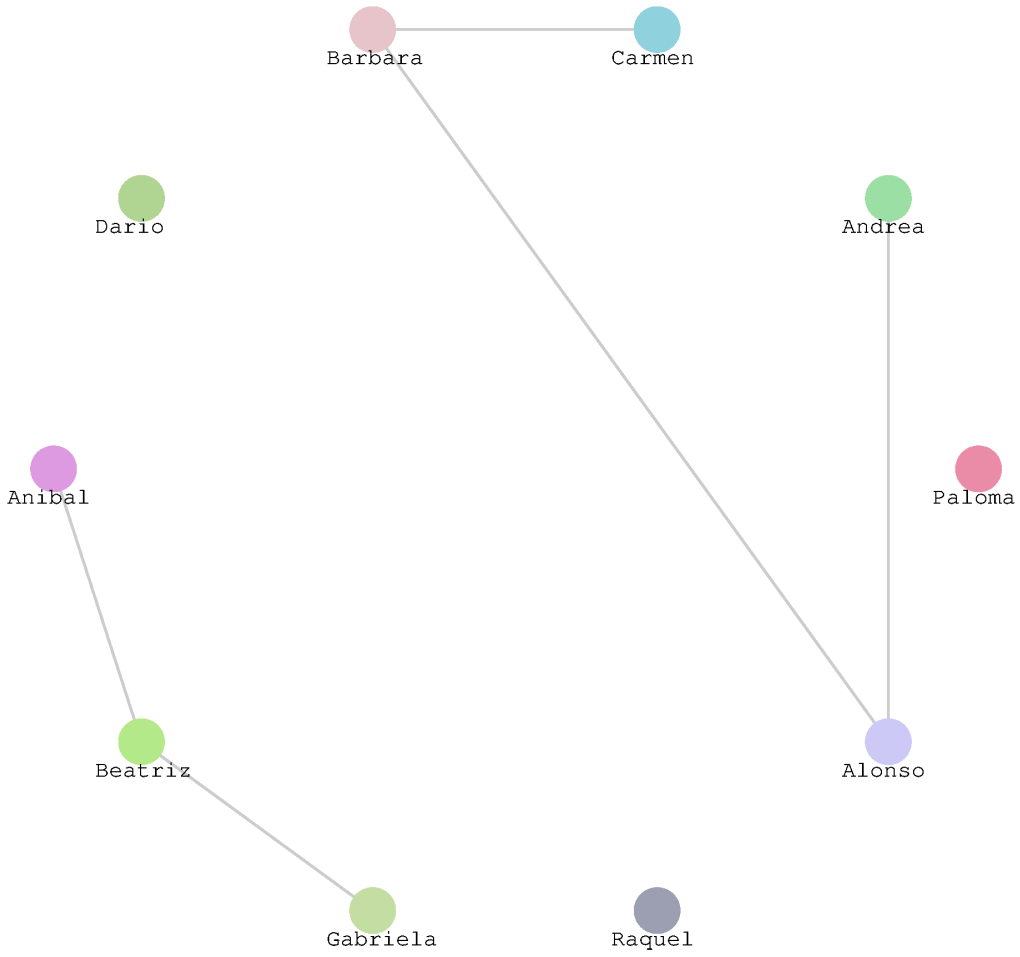

Figura 8. Imagen gráfica del grafo de conexiones

10. Dificultades y Soluciones

10.1. Asignación de Hobbies a los Usuarios

Una de las dificultades enfrentadas al inicio del desarrollo del proyecto fue la implementación de la generación de hobbies aleatorios en los perfiles de usuarios. En las primeras fases del proyecto surgió un problema recurrente el cual era que los hobbies asignados a un usuario se podían repetir dentro de su perfil. Esto era un error crítico, ya que conceptualmente no tendría sentido que un usuario tuviera, por ejemplo, "Deportes"listado dos veces entre sus intereses. Esto afectaba la coherencia de los datos generados y comprometía las conexiones entre los usuarios.

Solución :
Dentro de la función generate_random_hobbies() se introdujo una variable que actuaba como un registro de hobbies ya seleccionados para el usuario actual. Así logrando que los hobbies no se repitieran dentro del perfil de usuario.

10.2. Historial

El desarrollo del historial fue inicialmente planteado en dos partes, la primera parte se encargaría de rescatar la información de los usuarios en un archivo de texto y la segunda parte se encargaría de ingresar la información rescatada del archivo de texto hacia el programa.
Al implementar la primera parte del desarrollo del historial este respondió correctamente, pero al momento de implementar la segunda parte, el programa no guardaba de manera correcta la información de los usuarios, entrando así en conflicto con las secciones fundamentales del programa.

Solución :
Como grupo se realizo una reunión al momento de percatarnos de la problemática, lo que llevo a que analizáramos una pronta solución, llegando a un acuerdo de restaurar el programa a una versión anterior y de igual manera trabajar el desarrollo de ambas partes del historial como si fuera solo una parte, para poder descartar cualquier problema similar antes de implementar.

10.3. Algoritmo de búsqueda

En un principio, la intención era implementar el algoritmo de Dijkstra en el archivo search.c para encontrar al usuario con mas amigos. Esto tiene sentido, ya que Dijkstra es un algoritmo clásico y eficiente para resolver problemas de caminos mínimos en grafos ponderados con pesos no negativos.
Pero nos dimos cuenta que para buscar al usuario con mas amigos, usar Dijkstra no era la mejor opción ya que este algoritmo buscaba el camino mas corto desde un nodo, no la cantidad total de conexiones.

Solución:
La solución fue utilizar un algoritmo de búsqueda lineal para contar la cantidad de conexiones que cada usuario posee.

10.4. Algoritmo de Dijkstra

La principal dificultad encontrada en la implementación del algoritmo de Dijkstra fue que este debía de utilizarse mediante listas de adyacencia, y los ejemplos disponibles de como realizar esta implementación estaban escritos en el lenguaje C++.
Además, el algoritmo Dijkstra calcula los caminos mas cortos, lo que entrega resultados adicionales al necesitado para el programa.

Solución:
El primer problema se soluciono mediante el análisis del flujo del código de los ejemplos y la replicación de su funcionalidad.
El segundo problema se soluciono a través de la implementación de un segmento de código dedicado a identificar el nodo más lejano al nodo fuente e imprimir su camino.

11. Conclusión

El desarrollo del proyecto de simulación de una red social ha permitido integrar diversas técnicas de programación y estructuras de datos para replicar las dinámicas de las plataformas sociales modernas. Desde la creación aleatoria de perfiles de usuario con características aleatorias, hasta la implementación de algoritmos que evalúan la afinidad de los usuarios y generan conexiones entre ellos, el programa es capaz de simular las interacciones fundamentales de una red social.

Una de las características más destacable del proyecto es la implementación del índice de similitud de Jaccard, el que fue modificado para incorporar factores adicionales como compatibilidad según la edad y la personalidad. Este enfoque permite realizar una evaluación más completa de las posibles conexiones. Además, debido al diseño modular en el que esta basado el programa, que posee funciones específicas para tareas como la generación de usuarios, cálculo de similitudes, creación de conexiones y representación gráfica, facilita el mantenimiento y la expansión del programa.

La representación gráfica de la simulación logra mostrar las conexiones entre usuarios en un formato visual de fácil comprensión, además logra aplicar algoritmos avanzados como el de Dijkstra para determinar las conexiones de los usuarios, ilustrando cómo una red social puede ser modelada como un grafo.

La utilización adecuada de la memoria y liberación de recursos no utilizados aseguran que el programa no solo es funcional, sino también es eficiente.

Este proyecto demuestra cómo el lenguaje C puede ser utilizado para realizar aplicaciones complejas y completas si se combina una planificación adecuada, algoritmos eficientes y buenas prácticas de programación.

Finalmente, este proyecto no solo ha cumplido con los objetivos iniciales para este trabajo, sino que también ha entregado experiencia practica en la creación de programas con múltiples personas, representando una base sólida para futuras implementaciones y mejoras.

12. Contacto y Recursos

A continuación, puedes conocer más sobre nuestros trabajos y contactarnos a través de los siguientes medios:

- miloaiza@umag.cl — EhMigueh
- igcontre@umag.cl — Dysnomia9
- fpaillac@umag.cl — FelipePaillacar
- jfink@umag.cl — Johannsss
- bsanhuez@umag.cl — Bisalva

El proyecto completo está alojado en GitHub. Puedes visitarlo en el siguiente enlace: github.com/EhMigueh

Referencias

[1] 16Personalities. «Personalities Types». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://www.16personalities.com/es/descripcion-de-los-tipos.

[2] Z. Bobbitt. «Jaccard Index». Último acceso: 12 de diciembre 2024. (2024), dirección: https://www.statology.org/jaccard-similarity/.

[3] E. Etecé. «Algorithm». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://concepto.de/algoritmo-en-informatica/.

[4] GeekForGeeks. «Dijsktra». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.

[5] D. Ixbalanque. «QuickSort». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://asimov.cloud/blog/programacion-5/que-es-el-algoritmo-quicksort-275#:~:text=El%20algoritmo%20QuickSort%2C%20creado%20por,dividir%20el%20arreglo%20en%20subarreglos..

[6] T. U. Penguin. «Colors». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://www.theurbanpenguin.com/4184-2/.

[7] A. Torres. «Personalities Info». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://psicologiaymente.com/personalidad/tipos-de-personalidad.

[8] TutorialsPoint. «Dijsktra Representation». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://www.tutorialspoint.com/dijkstra-s-algorithm-for-adjacency-list-representation.

[9] Wikipedia. «DataStruct». Último acceso: 12 de diciembre de 2024. (2024), dirección: https://es.wikipedia.org/wiki/Estructura_de_datos.