UMAG Universidad de Magallanes | Facultad de Ingeniería

LoopWeb : Una red social en tu terminal

Constanza Araya[†], **Rodolfo Cifuentes**[†], **Bruno Martinez**[†], **Milton Hernández**[†] **y Guliana Ruiz**[†]

[†]*Universidad de Magallanes*

Este informe fue compilado el 14 de diciembre de 2024

Resumen

LoopWeb es una simulación en terminal de una red social, creada con el propósito de desarrollar habilidades en el manejo de volúmenes de datos mediante el uso de estructuras como Tablas hash, Listas enlazadas simples y Grafos.

Keywords: *LoopWeb , Simulación, Estructuras de datos, Tablas hash, Lista enlazada, Grafo*

Índice

1. Introducción

LoopWeb es un proyecto diseñado para emular una red social enfocada en la música, desarrollado como una herramienta de aprendizaje práctico en el ámbito de estructuras de datos. Este sistema tiene como objetivo principal implementar y combinar estructuras como grafos, tablas hash y listas enlazadas logrando una solución funcional y adaptativa.

Mediante este desarrollo, se busca consolidar los conocimientos adquiridos a través de la creación de un entorno digital que imita las dinámicas de interacción y conexión propias de las redes sociales modernas.

Entre las funcionalidades implementadas, destacan la gestión de usuarios, gustos y comentarios y la recomendación de conexiones mediante la priorización de contenidos, demostrando la aplicabilidad de conceptos teóricos en un entorno práctico y estructurado.

2. Objetivos

El proyecto LoopWeb se desarrolló con los siguientes objetivos específicos:

1. Diseñar e implementar una simulación funcional de una red social que permita la gestión de perfiles de usuarios, la creación de conexiones entre ellos y la publicación de contenido.
2. Integrar estructuras de datos complejas, como grafos y tablas hash, en un sistema cohesionado y escalable.
3. Implementar algoritmos eficientes para optimizar las operaciones del sistema, tales como la recomendación de amigos basándose en afinidades o el ordenamiento de listas para una impresión ordenada por pantalla.

3. Planificación del Proyecto

La planificación del proyecto se llevó a cabo en varias etapas. En primer lugar, se definieron los objetivos generales del sistema y se exploraron las ideas fundamentales que darán forma al proyecto.

Centralmente se tienen tres ramas principales que se pueden observar en la figura 1.

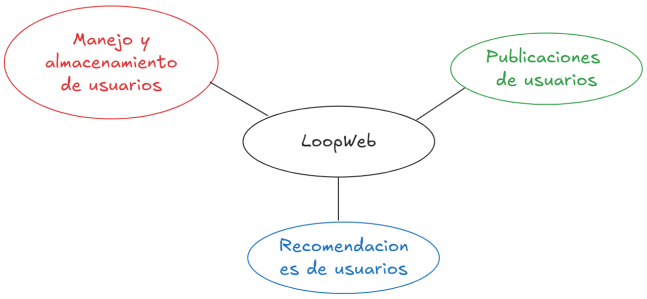

Figura 1. Ejes principales de LoopWeb

A continuación, se presentan los elementos clave que fueron considerados durante esta etapa de planificación:

1. **Creación de perfiles de usuario:** Se busca implementar un sistema básico de perfil de usuario que permite a cada persona en la red social tener una representación digital que contiene:
    - Nombre de usuario
    - Edad del usuario
    - Nacionalidad del usuario
    - Géneros musicales que le gustan al usuario
    - Bandas o artistas que le gustan al usuario (tratadas solo como bandas dentro del programa)
    - Comentarios que ha realizado el usuario
    - Amistades del usuario con otros perfiles de la red
2. **Establecimiento de conexiones entre usuarios:** Continuando con el punto anterior se busca desarrollar una funcionalidad que permita a los usuarios establecer relaciones de amistad, simulando un comportamiento cercano al de una red social.
3. **Publicación y visualización de publicaciones:** Los usuarios podrán crear publicaciones y ver aquellas que estén relacionadas con sus gustos musicales, lo que "fomentará la interacción" dentro de la red social.
4. **Exploración y recomendación de usuarios:** Este programa permitirá a los usuarios explorar otros perfiles recomendados para ellos (recomendaciones que se basarán en la cercanía de edad y los gustos musicales), lo anterior permitirá promover una experiencia personalizada para cada usuario.
5. **Realización de búsquedas eficientes:** Se implementarán algoritmos de búsqueda basados en grafos para permitir la localización eficiente de usuarios dentro de la red social, optimizando la experiencia de navegación.
6. **Eficiencia y Escalabilidad:** El sistema será diseñado utilizando estructuras de datos como tablas hash y listas enlazadas, lo que garantiza un rendimiento óptimo incluso con grandes volúmenes de datos.
7. **Opción Administrativa:** Se plantea una opción de administración que permitirá gestionar de manera sencilla los usuarios, gustos y sus publicaciones, facilitando el control dentro de la red social así como la creación de nuevos usuarios.

Para una correcta realización del proyecto se crean tres diagramas de flujo que juntos describen el comportamiento esperado de la aplicación.

En la figura 2 se muestra el diagrama de flujo principal de LoopWeb , que se divide en dos partes: un modo de administración (figura 3) y otro para la interacción de un usuario en particular (figura 4).

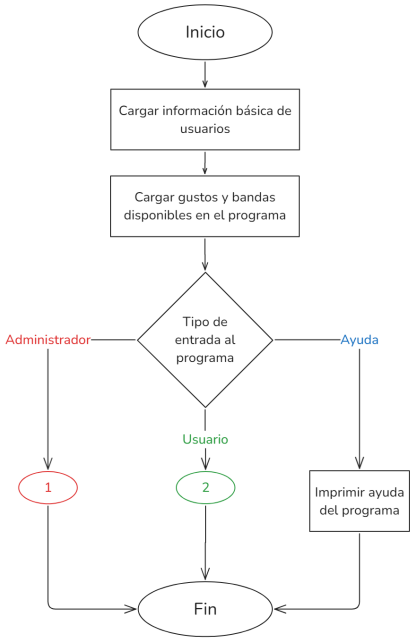

Figura 2. Diagrama de flujo prinicpal de LoopWeb

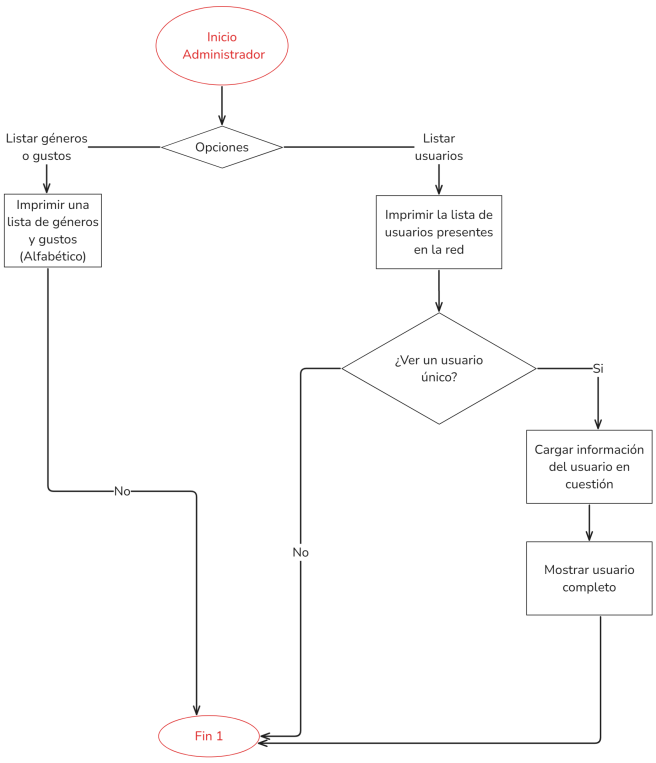

Figura 3. Flujo de un administrador de LoopWeb

4. Investigaciones previas

4.1. Investigación de algoritmos de similitud

Para mejorar la experiencia del usuario en LoopWeb , se analizaron diferentes algoritmos de similitud utilizados en redes sociales. Estos algoritmos permiten establecer conexiones y recomendaciones basadas en intereses y afinidades entre usuarios.

1. **Similaridad de Jaccard**: Este algoritmo utiliza la teoría de conjuntos, usando la unión e intersección, si los conjuntos tienen todas sus componentes similares, su índice de similitud será 1, en cambio si no hay ningún componente similar el índice será 0. Este índice se calcula:

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

2. **Similaridad del Coseno**: Mide la semejanza entre dos vectores calculando el coseno del ángulo entre estos:

$$\text{Similitud}(U,V) = \frac{\vec{U} \circ \vec{V}}{|\vec{U}| \cdot |\vec{V}|}$$

3. **Similaridad de Pearson**: Este algoritmo mide la relación **lineal** entre dos variables, que tan estrechos se alinean los puntos a lo largo de una línea recta en una gráfica de dispersión. El coeficiente varía de -1 a $+1$, donde -1 indica una correlación negativa perfecta, $+1$ representa una correlación positiva perfecta, y 0 sugiere una relación lineal entre las variables.

$$r = \frac{\frac{\sum(x_i \cdot y_i)}{N} - (\overline{x} \cdot \overline{y})}{\sigma_x \cdot \sigma_y}$$

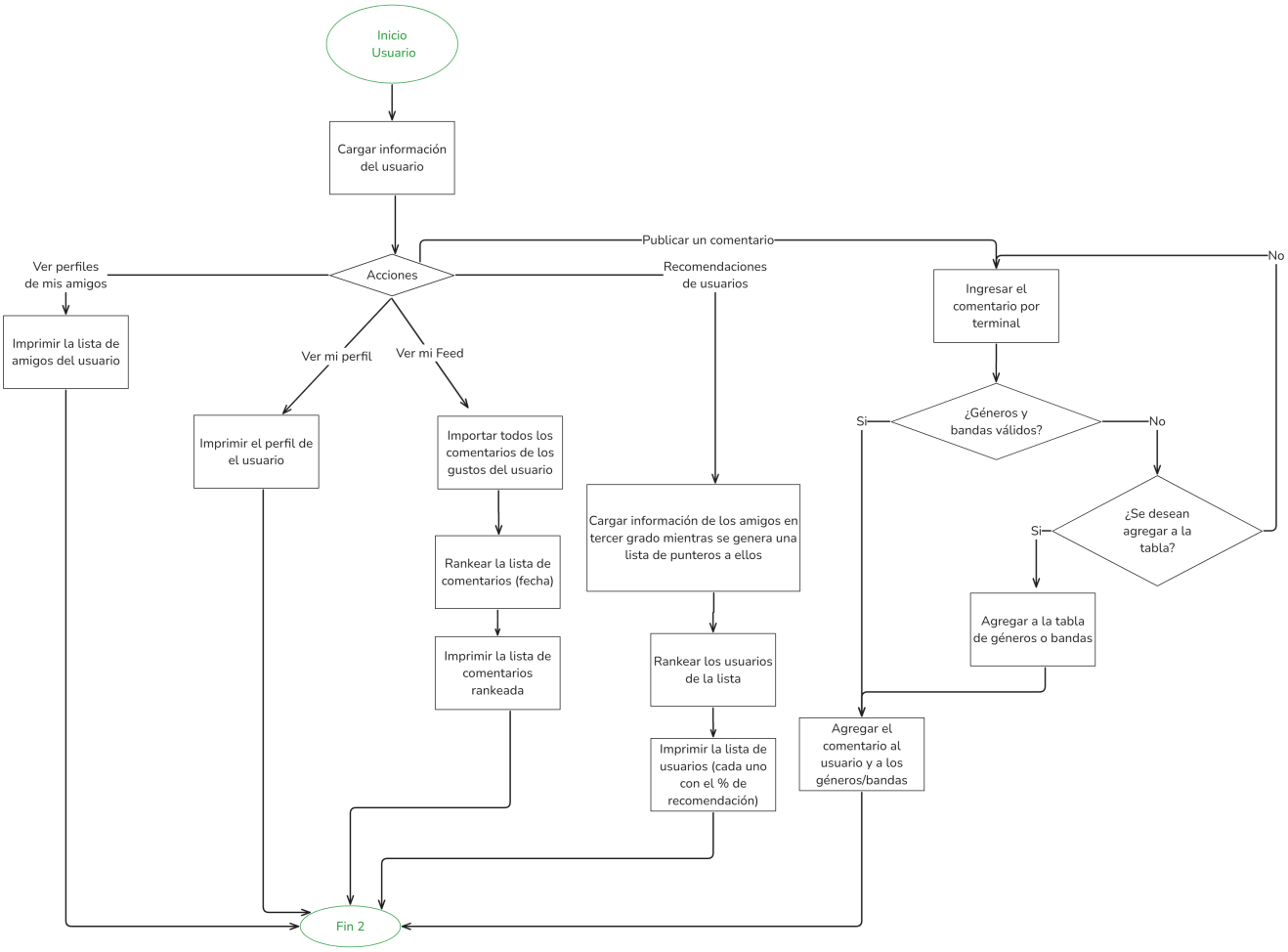

Figura 4. Flujo de un usuario de LoopWeb

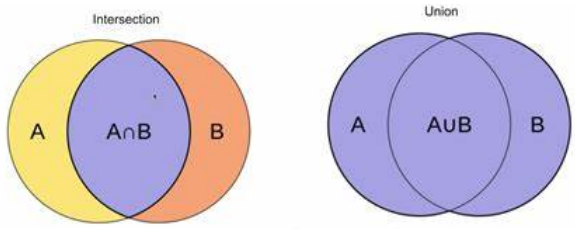

Figura 5. Índice de Jaccard

4. **Coeficiente de Dice**: También conocido como índice de Sørensen-Dice, es una herramienta estadística que está destinada a ser aplicada en datos para ver la presencia o ausencia de valores. Una puntuación de 0 significa que no hay superposición entre los dos conjuntos, mientras que una puntuación de 1 significa que son idénticos.

$$\text{Similaridad } (A, B) = 2\frac{|A \cap B|}{|A| + |B|}$$

5. **Vecinos comunes**: Es una técnica simple que mide la similitud entre nodos (o usuarios) en función de los vecinos compartidos, este identifica los vecinos de cada nodo, luego calcula la intersección de vecinos compartidos y usa metricas de conteo para determinar la similitud.
6. **Índice de Adamic-Adar**: Es una métrica basada en vecinos comunes pero se enfoca mas en los nodos pocos comunes, es más sofisticado que el simple conteo de vecinos comunes porque considera la rareza de los nodos comunes.

4.2. Analisis de las redes sociales

Para crear una simulación de una red social es necesario conocer el funcionamiento de otras redes sociales comunes en nuestros tiempos, es por esta razón que se realizó un análisis de las redes sociales más populares y aquello que pueden aportar a nuestra experiencia en LoopWeb .

1. **Bluesky**: Es una plataforma de redes sociales descentralizada que se fundó como una iniciativa de investigación como parte de Twitter en 2019. **Bluesky** es especial porque combina las ventajas de la descentralización (propiedad de datos, flexibilidad, diversidad de comunidades) con la facilidad de uso de las plataformas centralizadas. Alguna de sus características más destacables son:
   - Permite a sus usuarios personalizar el contenido que desean ver, es un tipo de visualización de contenido en el que las publicaciones se ordenan según la relevancia y el interés para el usuario, en lugar de la cronología.
   - Los post son mas cortos, cuenta con un limite de 300 caracteres, y pueden incluir fotos y videos.

   Uno de los puntos mas fuertes de Bluesky es el feed, ya que permite tener una experiencia mas personalizada al navegar por la red social, se puede crear un propio feed que permite un mejor viaje a través de la red social, eligiendo distintos contenidos para visualizar.
2. **Spotify**: Es una plataforma de música y podcasts digitales que da acceso a millones de canciones. Los creadores de esta red social saben que no hay dos oyentes iguales, por lo que la experiencia para cada usuario y muchas de las recomendaciones, son personalizadas. Algunas recomendaciones se basan en la selección editorial, como una playlist de música creada por editores musicales. Otras recomendaciones se adaptan al gusto único de cada oyente, como una playlist personalizada.
3. **Facebook**: Es la principal red social que existe en el mundo. Una red de vínculos virtuales, cuyo principal objetivo es dar un soporte para producir y compartir contenidos.
   Las sugerencias de amistad se basan en factores como: **amigos en común**, por medio de ciudad, escuela o trabajo, pertenecer a un mismo grupo, etiquetas en publicaciones, contactos vinculados. Sin embargo las recomendaciones también se basan en la ubicación (no precisa) del usuario, su idioma, su edad y sus perfiles seguidos.
4. **Youtube**: Es un sitio web dedicado a compartir videos, presenta una variedad de clips, programas de television, videos musicales, gameplays, videoblogs. Las recomendaciones de contenido de esta aplicación se centran en: Encuestas, Intereses e interacciones.

5. **TikTok**: Permite crear, editar y subir vídeos propios o de terceros inicialmente con una duración máxima de un minuto a los que se les incluyen fondos musicales o sonidos. El algoritmo de TikTok es una fórmula patentada que determina qué videos se muestran a cada usuario. Los principales factores que influyen en el algoritmo de TikTok son las interacciones de los usuarios, la información de los videos, la configuración del dispositivo y la cuenta.
6. **Instagram**: Es una red social principalmente visual, donde un usuario puede publicar fotos, videos e historias de corta duracion. En Instagram, no hay un solo algoritmo que decide lo que se ve, ya que cada parte de la app (el feed, la sección 'Explorar' y los Reels) tiene su propio sistema de clasificación según cómo es usado. Actualmente, el algoritmo de Instagram da prioridad a las **publicaciones más recientes**.

¿Que se aplicó de estas aplicaciones a LoopWeb ?

Muchas de estas redes sociales tienen algoritmos y características importantes, algunas de las ideas que tomó LoopWeb de estas aplicaciones son:

- Feed personalizado (como Bluesky)
- Personas que quizás conozcas (como Facebook e Instagram)
- Limite de caracteres (como Bluesky)
- Gestión y enlaces entre usuarios

5. Estructuras de datos implementadas

Dentro de LoopWeb se utilizaron en esencia dos estructuras de datos: Tablas hash y Listas enlazadas simples. Ambas tienen sus propias características y usos específicos, y se han diseñado para cumplir con diferentes objetivos propuestos para la aplicación.

5.1. Tablas Hash

El uso de esta estructura de datos se debe a su eficiencia a la hora de almacenar información. Esta estructura fue utilizada para almacenar los **Usuarios**, **Bandas**, **Generos** y **Comentarios** que existen dentro de la "base de datos" de LoopWeb .

Las tablas hash también son fáciles de crear y eliminar en forma recursiva y los datos que se guardan en ellas pueden ser de cualquier tipo, lo que facilita la gestión de información de los usuarios o bandas quienes no tienen necesariamente un Identificador numérico.

5.2. Listas Enlazadas Simples

Esta estructura es el 'Engranaje central' de este programa, escogida por su gran flexibilidad en cuanto a espacio.

El principal uso de esta estructura de datos se encuentra en la creación de "Enlaces a las tablas hash" anteriores. Esto permite el almacenamiento de información que se encuentra en las tablas sin necesidad de duplicar la información presente en las mismas.

5.3. Grafos

El almacenamiento de usuarios es la caracteristica fundamental de LoopWeb , esencial para su correcto funcionamiento.

Para "dar vida" a esta estructura de datos dentro del programa no se creó una estructura de datos especifica sino que se usó una abstracción de los conceptos de grafos mediante las tablas hash.

Los algoritmos para grafos fueron utilizados para recorrer las conexiones de los usuarios y poder recomendar amigos entre amigos de amigos para ello se usó el algoritmo **BFS** para recorrer el grafo de usuarios.

¿Qué otras estructuras de datos podrían haber sido implementadas?

El uso de Listas enlazadas simples permite la facilidad de implementación de todas las funciones propias de LoopWeb , sin embargo estructuras como las Listas Doblemente Enlazadas o las Colas podrían dar más eficiencia a ciertas operaciones.

6. En busca de la eficiencia

6.1. Algoritmos de ordenamiento

Los **algoritmos de ordenación** son un conjunto de instrucciones que toman un arreglo o lista como entrada y organizan los elementos en un orden particular. Las ordenaciones suelen ser numéricas o una forma de orden alfabético (o lexicográfico), y pueden ser en orden ascendente (AZ, 0—9) o descendente (ZA, 9—0).

Estos algoritmos son fundamentales dado que permiten reducir la tarea de ordenar datos a una tarea más simple y eficiente. Estos algoritmos tienen aplicaciones directas en algoritmos de búsqueda, algoritmos de bases de datos, algoritmos de estructura de datos y muchos más.

Para la creación de este programa se requiere la ordenación de diferentes listas enlazadas (Listas de usuarios, de bandas, de géneros y comentarios) bajo criterios tanto numéricos como lexicográficos.

Inicialmente se planteó el uso del algoritmo Bubble Sort o **Ordenamiento por intercambio** el cuál, compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto. Repite este proceso hasta que todos los elementos estén ordenados. Este algoritmo es fácil de implementar pero su eficiencia es bastante baja, con una complejidad temporal de $O(n^2)$. Dado que su rendimiento empeora conforme crece el tamaño de la lista, se consideró que no era adecuado para este proyecto, donde se manejan conjuntos de datos más grandes.

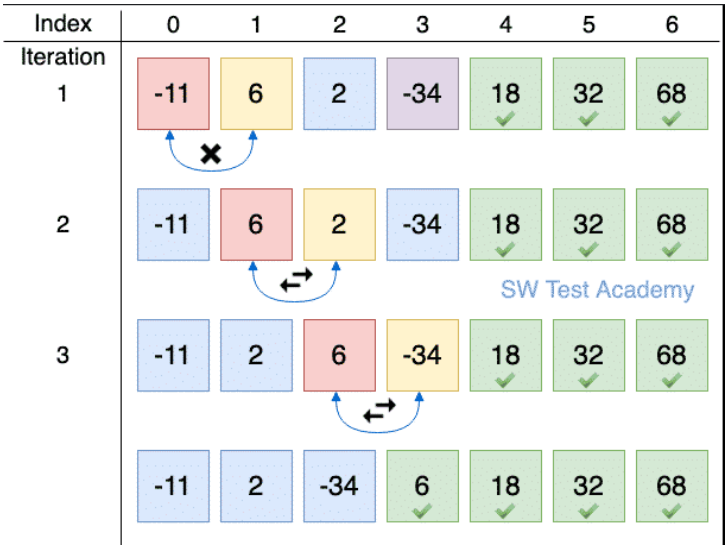

Figura 6. Bubble Sort

Por otro lado el algoritmo Merge Sort o **Ordenamiento por Mezcla** divide el conjunto de elementos en subconjuntos más pequeños, los ordena por separado y luego los fusiona para obtener un conjunto ordenado más grande. Este algoritmo utiliza la estrategia de “divide y vencerás”, lo que le da una eficiencia mucho mayor que Bubble Sort, con una complejidad temporal de $O(n \log n)$. Dado que el proyecto implicaba manejar (posiblemente) grandes volúmenes de datos, se optó por Merge Sort debido a su mejor comportamiento en términos de rendimiento.

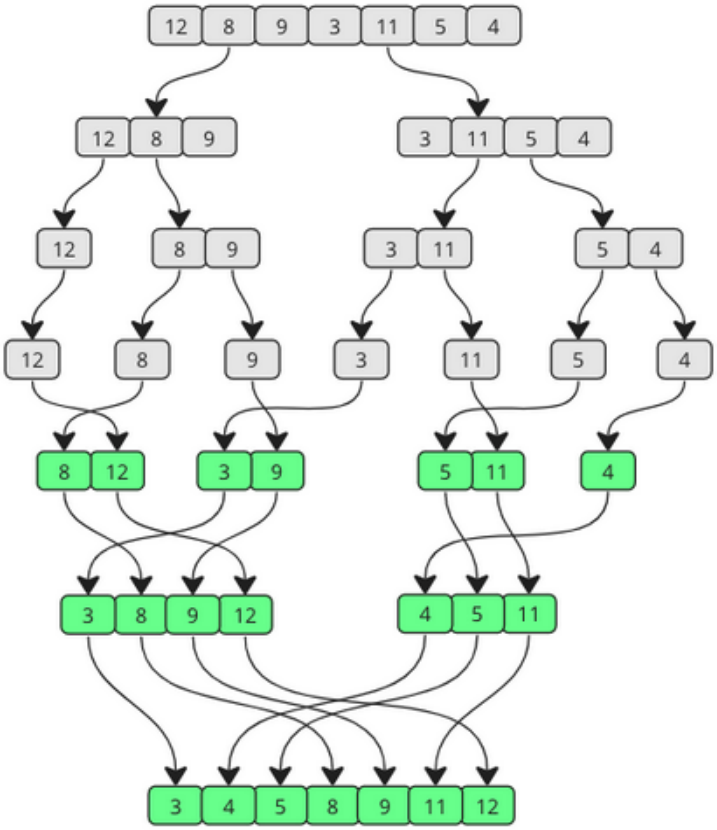

Figura 7. Merge Sort

6.2. JSON

JSON es un formato de intercambio de datos utilizados para almacenar e intercambiar información estructurada de manera legible. Es ampliamente usado para la transferencia de datos entre servidores y clientes.

¿Para qué se utiliza un archivo JSON?

JSON almacena información en forma de pares clave-valor y es utilizado principalmente para transferir datos estructurados. Un archivo JSON puede contener diferentes tipos de datos, como cadenas de texto, números, objetos, arreglos, valores booleanos y null, como se muestra en la figura 8

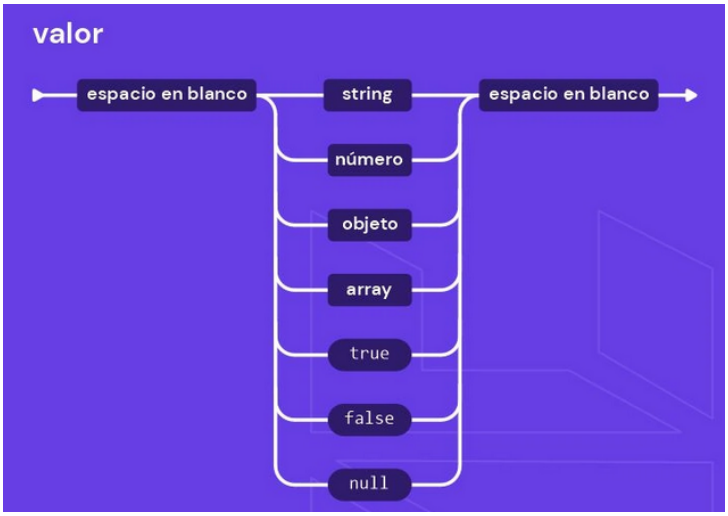

Figura 8. Diagrama json

En el estracto de código 1 se muestra un ejemplo de un archivo JSON que contiene información sobre un usuario perteneciente a LoopWeb

```
{
  "userName": "Alice",
  "age": 18,
  "nationality": "Spain",
  "genres": [ "rock", "pop" ],
  "artists": [ "rihanna", "ladyGaga" ],
  "description": "Amante de los festivales de musica.",
  "friends": [ "Bob", "Carol", "Mallory" ]
}
```

Código 1. Ejemplo de .json

¿Cómo utilizar JSON en C?

Para trabajar con archivos .json en C, es necesario utilizar librerías externas. Algunas opciones son:

1. Jansson
2. Json-c
3. cjson

En este caso, se usará la librería Jansson.

Jansson es una librería de C para trabajar con datos JSON, simple, sin dependencias externas y con documentación completa.

Algunas de las funcionalidades más importantes de Jansson son:

1. **json_error_t**: Tipo de dato para almacenar errores durante la lectura de un archivo JSON.

```
        json_error_t error;

```

Código 2. json_error_t

2. **json_t**: Tupo de dato para unobjeto JSON, siempre a través de un puntero.

```
        json_t *materias_json = json_object_get(alumnos_json,"materias");

```

Código 3. json_t

3. **json_loadf()**: Carga el contenido de un archivo JSON en una variable de tipo json_t.

```
        json_t *json = json_loadf(archivo,0,&error);

```

Código 4. json_loadf

4. **json_object_get()**: Accede a los valores de un objeto JSON.

```
        json_object_value(alumnos_json,"nombre");

```

Código 5. json_object_get

5. **json_string_value()**: Extrae el valor de una cadena de texto en JSON.

```
        json_string_value(json_object_value(alumnos_json,"nombre"));

```

Código 6. json_string_value

6. **json_integer_value()**: Extrae el valor de un entero de un objeto JSON.

```
        json_integer_value(json_object_value(alumnos_json,"edad"));

```

Código 7. json_integer_value

La utilización de arhcivos JSON para el almacenamiento de la información de este programa es crucial para que la información sea sencilla de almacenar y eficiente de leer, permitiendo no cargar toda la información del programa en cada acción sino cargar **solo lo necesario** en cada momento.

7. Algunos algoritmos y funciones destacables

A continuación se explican algunas de las funciones más interesantes que se construyeron durante el desarrollo del programa.

7.1. Funcion GetOPT

getOPT es una función parte de las librerías estándar de C que permite una cómoda interacción con la línea de comandos facilitando el uso de banderas de opciones.

Esta función recibe como argumento un string de texto y devuelve un entero que representa el código de acceso con el que se quiere ingresar.

A pesar de que esta funcionalidad que es de por sí bastante útil se optó por el uso de la función **getoptLong()** para poder utilizar banderas más extensas. Así dentro de LoopWeb se pueden ocupar las opciones `./build/loopweb.out -h`, `./build/loopweb.out -a`, `./build/loopweb.out -u <nombre>`, `./build/loopweb.out --help`, `./build/loopweb.out --administrador` ó `./build/loopweb.out --user <nombre>`.

7.2. Implementacion de mergeSort para Listas enlazadas

El algoritmo de ordenamiento de merge sort es utilizado principalmente sobre arreglos de datos, pero también puede ser utilizado sobre listas enlazadas. En el presente proyecto se implementó una triada de funciones que juntas permiten implementar merge sort sobre listas enlazadas (A continuación se muestra un ejemplo con listas de enlaces a géneros, aunque se puede implementar con cualquier lista enlazada simple):

1. **split_genreLinkList**: Esta función divide una lista enlazada en dos mitades, parte importante y esencial para el merge sort.

```c
/**
* @brief Divide una lista enlazada en dos mitades.
* @param source Puntero al nodo inicial de la lista a dividir.
* @param frontRef Referencia a la primera mitad de la lista.
* @param backRef Referencia a la segunda mitad de la lista.
*/
void split_genreLinkList(GenreLinkPosition source, GenreLinkPosition* frontRef, GenreLinkPosition*
    backRef)
{
    GenreLinkPosition slow, fast;
    slow = source;
    fast = source->next;
    while (fast != NULL) {
        fast = fast->next;
        if (fast != NULL) {
            slow = slow->next;
            fast = fast->next;
        }
    }
    *frontRef = source;
    *backRef = slow->next;
    slow->next = NULL;
}
```

Código 8. split_genreLinkList

2. **merge_genreLinkLists:** Esta función fusiona dos listas enlazadas ordenadas en una sola lista ordenada.

```c
/**
* @brief Fusiona dos listas enlazadas ordenadas en una sola lista ordenada
* @param a Puntero a la primera lista ordenada
* @param b Puntero a la segunda lista ordenada
* @return Puntero al nodo inicial de la lista fusionada ordenada
*/
GenreLinkPosition merge_genreLinkLists(GenreLinkPosition a, GenreLinkPosition b)
{
    if (a == NULL) return b;
    if (b == NULL) return a;
    GenreLinkPosition result;
    if (strcmp(a->genre, b->genre) <= 0) {
        result = a;
        result->next = merge_genreLinkLists(a->next, b);
    } else {
        result = b;
        result->next = merge_genreLinkLists(a, b->next);
    }
    return result;
}
```

Código 9. fusion de listas

3. **mergerSort_genreLinkList**: Esta función controla el algoritmo como tal llamando **recursivamente** a las funciones anteriores y a sí misma.

```c
/**
* @brief Ordena una lista enlazada utilizando el algoritmo merge sort.
* @param headRef Referencia al puntero del nodo inicial de la lista a ordenar.
*/
void mergeSort_genreLinkList(GenreLinkPosition* headRef)
{
    GenreLinkPosition head = *headRef;
    GenreLinkPosition a, b;
    if ((head == NULL) || (head->next == NULL)) {
        return;  /**<si la lista esta vacia o tiene un solo elemento, no hay que ordenar */
    }
    split_genreLinkList(head, &a, &b);
    mergeSort_genreLinkList(&a);  /**<ordena la primera mitad */
    mergeSort_genreLinkList(&b);  /**<ordena la segunda mitad */
    *headRef = merge_genreLinkLists(a, b);  /**<fusiona las dos mitades ordenadas */
}
```

Código 10. mergerSort_genreLinkList

Estas funciones fueron implementadas y ligeramente modificadas al interior del código para ajustarse a cada uno de los casos de uso necesarios[1].

[1] Es importante notar que todas las funciones hacen uso de punteros a los nodos de la lista enlazada y que además la función principal debe recibir como parámetro un puntero al **Primer** nodo de la lista enlazada y NO al centinela de la misma

7.3. Algoritmo BFS para recorrido de un grafo

El algoritmo BFS (algoritmo de búsqueda en profundidad) se encarga de recorrer los usuarios conectados por "nivel" y es utilizado para generar las recomendaciones de amigos, esta información se guarda en una lista de usuarios visitados, esto se hace para evitar repetir usuarios que ya se han visitado anteriormente.

Las recomendaciones dentro del programa se restringen a los amigos en tercer grado (Los amigos de amigos de amigos) para evitar una lista muy larga de recomendaciones.

```
BFS(Grafo G, Nodo inicial S):
Crear una cola vacia Q
Crear un conjunto vacio VISITADOS
Crear un diccionario NIVELES, donde cada nodo tiene un nivel asociado

Encolar S en Q
Marcar S como visitado anhadiendolo a VISITADOS
Asignar nivel 1 a S en NIVELES

Mientras Q no este vacia:
    Nodo actual <- desencolar Q
    Procesar Nodo actual (EJ: Almacenarlo en una lista de recomendaciones)

    Para cada vecino V de Nodo actual en G:
        Si V no esta en VISITADOS:
            Encolar V en Q
            Marcar V como visitado anhadiendolo a VISITADOS
            Asignar nivel (NIVELES[Nodo actual] + 1) a V en NIVELES
```

Código 11. Pseudocodigo del algoritmo BFS

8. Apariencia de loopweb

La función `print_loopweb` tiene como objetivo procesar y resaltar publicaciones desde una cadena de caracteres. Las palabras que comienzan con `@` (bandas o artistas) se imprimen en verde, mientras que las que comienzan con `#` (géneros) se imprimen en rojo.

Esta función divide el texto en palabras, procesa cada una (carácter por carácter) y aplica los colores correspondientes a los géneros o bandas, asegurando que caracteres mal ubicados o repetidos no interfieran con el formato.

Un aspecto clave del funcionamiento es el uso de `strtok()` para dividir el texto en palabras y el ciclo que recorre cada carácter para identificar y resaltar correctamente los géneros y bandas. La línea `printf(ANSI_COLOR_GREEN);` es esencial, ya que aplica el color verde a las palabras que comienzan con `@`, permitiendo que se resalten correctamente en la salida (análogamente para los géneros).

El formato resultante permite identificar fácilmente los géneros (`@`) y bandas (`#`) en las publicaciones, mejorando la visualización del contenido en LoopWe♭ . Esta funcionalidad contribuye a resaltar elementos clave de manera clara y eficiente, y es útil para visualizar perfiles o contenido dinámico de los usuarios.

9. Experiencia del Usuario

En LoopWe♭ , la experiencia del usuario ha sido diseñada para ser dinámica, llamativa y altamente interactiva, esto debido a su característica de ser una aplicación no estática (donde todas las acciones son realizadas por el usuario y no existe generación automática o programada de contenido).

Para poder llevar a cabo este dinamismo se hizo uso de los colores proporcionados por el código ANSI, que permite representar colores en la pantalla.

10. Conlusiones

LoopWeb ha sido un proyecto que ha supuesto un gran reto en diversos sentidos para quienes lo desarrollaron, esto debido a su complejidad conceptual y la necesidad de llevar a cabo una correcta abstracción del problema para poder implementarlo en un entorno no tan gráfico como es el terminal.

La investigación fue un punto clave para el correcto desarrollo del programa, ya que permitió un correcto enfoque en aquellos puntos que se consideraron podrían ser de interés para un producto final, como la implementación de algoritmos de recomendación o las restricciones de longitud de las publicaciones.

El trabajo en equipo fue también un punto clave, logrando unir esfuerzos de diferentes profesores para conseguir una documentación completa y un código funcional y eficiente.

Referencias

[1] D. E. Knuth. «The Art of Computer Programming, Volume 3: Sorting and Searching. Addison-Wesley.» (nov. de 1998), dirección: https://www.oreilly.com/library/view/art-of-computer/9780321635792/title.xhtml (visitado 29-11-2024).

[2] ISO/IEC. «Programming Languages - C (ISO/IEC 9899:2011)». (dic. de 2011), dirección: https://www.iso.org/standard/57853.html (visitado 01-12-2024).

[3] B. Stroustrup. «The C Programming Language: Special Edition for Beginners. Addison-Wesley.» (jun. de 2013), dirección: https://www.amazon.com/C-Programming-Language-Special-Beginners/dp/B00K6FI0QM (visitado 01-12-2024).

[4] Gerardo Acevedo Arzola. «Merge Sort en lista enlazada». (nov. de 2018), dirección: https://pacoluisger.wixsite.com/estructurasdatosc/post/merge-sort (visitado 04-12-2024).

[5] BettaTech. «video explicativo sobre los grafos y algoritmos de recorrido». (sep. de 2019), dirección: https://www.youtube.com/watch?v=23pdz9VtIBo&t=1s (visitado 26-09-2019).

[6] (IJIET) International Journal of Innovations in Engineering and Technology (IJIET). «Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm». (2020), dirección: https://www.semanticscholar.org/paper/International-Journal-of-Innovations-in-Engineering/8575e8beef47bd2880c92f54a749f933db983e56 (visitado 2020).

[7] Eragroup. «Ejemplos del coeficiente de correlación de Pearson». (nov. de 2020), dirección: https://eragroup.eu/ejemplos-del-coeficiente-de-correlacion-de-pearson/ (visitado 17-11-2020).

[8] Daniel J Bell. «Dice similarity coefficient». (ago. de 2021), dirección: https://radiopaedia.org/articles/dice-similarity-coefficient (visitado 02-08-2021).

[9] programacioniiuno. «Algoritmo MergeSort». (abr. de 2021), dirección: https://sites.google.com/site/programacioniiuno/temario/unidad-6---anlisis-de-algoritmos/algoritmo-mergesort (visitado 04-12-2024).

[10] techiedelight. «Ordenar una lista doblemente enlazada usando la ordenación por combinación». (oct. de 2021), dirección: https://www.techiedelight.com/es/sort-doubly-linked-list-merge-sort/ (visitado 04-12-2024).

[11] D. A. «¿Qué es JSON?» (Ene. de 2023), dirección: https://www.hostinger.es/tutoriales/que-es-json (visitado 27-11-2024).

[12] Grapheverywhere. «Algoritmos de similiridad». (ene. de 2023), dirección: https://www.grapheverywhere.com/algoritmos-de-similaridad/#:~:text=Los%20algoritmos%20de%20similaridad%20o%20algoritmos%20de%20similitud%2C,o%20compatibilidades%20que%20existen%20entre%20conjuntos%20de%20datos. (visitado 01-01-2023).

[13] Jansson. «API reference». (jun. de 2023), dirección: https://jansson.readthedocs.io/en/latest/apiref.html#encoding (visitado 27-11-2024).

[14] lathashreeharisha. «Dice Coefficient! What is it?» (Feb. de 2023), dirección: https://medium.com/@lathashreeh/dice-coefficient-what-is-it-ff090ec97bda (visitado 18-02-2023).

[15] Daniel Ixbalanque. «¿Qué es el algoritmo de Merge Sort ?» (Abr. de 2024), dirección: https://asimov.cloud/blog/programacion-5/que-es-el-algoritmo-de-merge-sort-270 (visitado 04-12-2024).

[16] Daniel Jurafsky y James H. Martin. «Speech and Language Processing(3rd ed.)» (ago. de 2024), dirección: https://web.stanford.edu/~jurafsky/slp3/6.pdf (visitado 20-08-2024).

[17] Faster capital. «Similitud de datos revelando conexiones con el coeficiente de Pearson». (jun. de 2024), dirección: https://fastercapital.com/es/contenido/Similitud-de-datos--revelando-conexiones-con-el-coeficiente-de-Pearson.html (visitado 04-06-2024).

[18] Jose Canuman. «DataStructure.pdf». (dic. de 2024), dirección: https://github.com/jcanuman/Apunte_EDatos/blob/master/DataStructure.pdf (visitado 07-12-2024).

[19] Miguel Jesús Torres Ruiz. «Analítica y Visualización de Datos - Similitud DICE o Coeficiente de Sørensen y Similitud de Jaccard». (mar. de 2024), dirección: https://www.youtube.com/watch?v=7wVgYmvvyCM (visitado 18-03-2024).

[20] Sandra Navarro. «¿Cómo funciona el algoritmo de vecinos más próximos». (oct. de 2024), dirección: https://keepcoding.io/blog/algoritmo-de-vecinos-mas-proximos/ (visitado 14-10-2024).

[21] Sandra Navarro. «Similitud entre vectores o cosine similarity». (abr. de 2024), dirección: https://keepcoding.io/blog/similitud-entre-vectores-o-cosine-similarity/ (visitado 12-04-2021).

[22] SW Team. «Introducción al Algoritmo de Ordenación Merge Sort». (mayo de 2024), dirección: https://www.swhosting.com/es/blog/introduccion-al-algoritmo-de-ordenacion-merge-sort (visitado 04-12-2024).

[23] Adobe. «Archivos de texto: diferentes formatos de texto y usos más comunes». (), dirección: https://www.adobe.com/es/acrobat/resources/document-files/text-files.html#qu%C3%A9-es-un-archivo-de-texto (visitado 27-11-2024).

[24] C. Bravo. «Así funciona el algoritmo de X, el nuevo Twitter». (), dirección: https://metricool.com/es/al-descubierto-el-algoritmo-de-x/ (visitado 24-11-2024).

[25] J. P. Colomé. «Por qué Bluesky es diferente: la libertad de elegir contenidos frente a la dictadura del algoritmo». (), dirección: https://elpais.com/tecnologia/2024-11-22/por-que-bluesky-es-diferente-la-libertad-de-elegir-contenidos-frente-a-la-dictadura-del-algoritmo.html (visitado 24-11-2024).

[26] Facebook. «"Personas que quizá conozcas" en Facebook». (), dirección: https://facebook.com/help/163810437015615/?locale=es_LA&_rdc=1&_rdr (visitado 24-11-2024).

[27] Y. Fernández. «Guía de inicio a Bluesky: qué es, cómo funciona, qué son los feeds y cómo configurarlo». (), dirección: https://www.xataka.com/basics/guia-inicio-a-bluesky-que-como-funciona-que-feeds-como-configurarlo (visitado 24-11-2024).

[28] ITCHI.PRO. «Comprensión de la similitud del coseno y su aplicación». (), dirección: https://ichi.pro/es/comprension-de-la-similitud-del-coseno-y-su-aplicacion-278464028813674.

[29] Linear Algebra for Data Science. «Adamic-Adar Index)». (), dirección: https://library.fiveable.me/key-terms/linear-algebra-for-data-science/adamic-adar-index.

[30] Luis Benites. «Índice Jaccard / Coeficiente de similitud». (), dirección: https://statologos.com/indice-jaccard/.

[31] Paula Rochina. «El análisis de redes sociales mediante la teoría de grafos». (), dirección: https://www.inesem.es/revistadigital/informatica-y-tics/teoria-grafos/ (visitado 24-11-2024).

[32] Spotify. «Comprender las recomendaciones en Spotify». (), dirección: https://www.spotify.com/mx/safetyandprivacy/understanding-recommendations (visitado 24-11-2024).