ESQUEMA DEL INFORME

INTEGRANTES

13 de diciembre de 2024

1. INDICE

[implementar indice]

2. INTRODUCCION

[aca iria la introduccion]

3. OBJETIVOS

[Breve descripcion de lo que fue solicitado como proyecto]

4. PLANIFICACION DEL PROYECTO

[describir paso a paso que se buscaba hacer, ideas descartadas, por que fueorn descartadas, cual era nuestro objetivo de un usuario, tanto en interfaz como experiencia dentro de loopweb]

5. LOOP WEB

['vender' loopweb]

6. INVESTIGACIONES PREVIAS

[aca iria la investigacion sobre las redes sociales y la investigacion sobre algoritmos de similitud]

6.1. Analisis de las redes sociales

El proyecto **LoopWeb** trata sobre la creacion de una **red social**, previamente se necesita realizar una investigacion extensiva sobre las distintas redes sociales que existen actualmente, principalmente se necesita saber que hace que **Bluesky** sea tan especial como para ser el referente maximo para este proyecto.

7. Bluesky

Es una plataforma de redes sociales descentralizada que se fundó como una iniciativa de investigación como parte de Twitter en 2019. Inicialmente, el sitio era solo por invitacion, pero se abrio al publico en febrero. En otras palabras **Bluesky** se podria denominar como el nuevo **Twitter**.

¿Que lo diferencia de otras redes sociales?

Bluesky es especial porque combina las ventajas de la descentralizacion (propiedad de datos, flexibiidad, diversidad de comunidades) con la facilidad de uso de las plataformas centralizadas, ofreciendo a los usuarios un control nunca antes visto sobre sus cuenta. ”Bluesky es una red social abierta que ofrece a los creadores independencia de las plataformas, a los desarrolladores libertad de creación y a los usuarios la posibilidad de elegir su experiencia”, afirma la plataforma en su página social.

- **Caracteristicas**
  - Feed algorítmico: Permite a sus usuarios personalizar el contenido que desean ver, es un tipo de visualizacion de contenido en el que las publicaciones se ordenan segun la relevancia y el interes para el usuario, en lugar de la cronologia.
  - Moderacion del contenido: Cuenta con multiples opciones para que los usuarios puedan sentirse comodos utilizando esta red social. Desde silenciar hashtags y palabras hasta bloqueo de cuentas.
  - Limite de caracteres: Los post son mas cortos, cuenta con un limite de 300 caracteres, y pueden incluir fotos y videos.
- **Funcionalidades**: Permite a los usuarios compartir mensajes breves, interactuar con otros, consumir contenido y participar en conversaciones globales.
- **Algoritmo**: Su característica más disruptiva son los feeds personalizados, que permiten a los usuarios construir su propia experiencia algorítmica. La magia de Bluesky reside en la posibilidad de gestionar feeds especificos basados en intereses, palabras clave o listas de usuario, lo cual se gestiona en el buscador de la red social.

¿Que podriamos sacar de Bluesky?

A nivel personal creo que 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, siempre aparecera el contenido que mas le guste al usuatrio, evitando ver post que generen disgusto. Se puede crear un propio feed que permite un mejor viaje a travez de la red social, eligiendo distintos contenidos para visualizar, no como otras redes sociales que basan el algoritmo en base a likes e interacciones.

8. Spotify

Spotify es una plataforma de musica, podcasts y videos digitales que da acceso a millones de canciones.

¿Como funcionan las recomendaciones?

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 seleccion editorial, como una playlist de musica creada por editores musicales.Otras recomendaciones se adaptan al gusto unico de cada oyente, como una playlist personalizada.

- Seleccion editorial: Los editores utilizan datos estadisticos y una comprension de las tendencias culturales para ubicar el contenido donde tiene mas probabilidades de ser relevante.
- Recomendaciones personalizadas: A medida que el usuario interactua en spotify, ya sea con acciones como buscar, escuchar, saltar o guardar en la biblioteca influye en la interpretacion de los gustos.A esto se le llama ”perfil de gustos”lo cual da al algorito una indicacion de lo que le gusta al usuario.
- Informacion compartida: Las recomendaciones tambien se basan en la ubicacion(no precisa) del usuario, idioma, edad, seguidos.

9. Facebook

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.

¿Como recomienda amigos?

Las sugerencias de amistad se basan en factores como:

- Amigos en comun.
- Ciudad, escuela, trabajo.
- Pertenecer a un mismo grupo en Facebook.
- Etiquetas en publicaciones, fotos.
- Contactos telefonicos vinculados.

10. X

X es un servicio que permite que los grupos de amigos, familiares y compañeros de trabajo se comuniquen y estén en contacto a través de mensajes rápidos y frecuentes.

¿Como funciona el algoritmo?

X basa sus recomendaciones en:

- Repost(publicaciones compartidas).
- Respuestas(comentarios).
- Suscripcion: Los usuarios que poseen una suscripcion activa aparecen mas en el feed recomendado.
- Tendencias: Para garantizar que la red social sea la mejor forma de conocer las ultimas noticias y novedades.
- Likes en publicaciones.

11. Youtube

Es un sitio web dedicado a compartir videos, presenta una variedad de clips, programas de television, videos musicales, gameplays, videoblogs.

¿Como funcionan sus recomendaciones?

- Encuestas: Se envían millones de encuestas al mes (aunque es probable que a un mismo usuario solo le salgan dos o tres) para pedir opiniones sobre los vídeos de la plataforma.
- Intereses: Se tiene en cuenta cuándo los usuarios hacen clic en ”No me interesa.[en] los vídeos.
- Interacciones: Se tiene en cuenta el número de likes, dislikes y las veces que se ha compartido un vídeo.

12. TikTok

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.

¿Como funciona el algoritmo de TikTok?

El algoritmo de TikTok es una fórmula patentada que determina qué videos de TikTok 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 y la configuración del dispositivo y la cuenta.

- Interacciones del usuario:
    - Likes, comentarios, favoritos, seguidos.
    - Videos marcados como no me interesa.
    - Videos denunciados.
- Informacion de videos:
    - Sonidos, efectos.
    - Hashtags, temas de actualidad.
- Configuracion del dispositivo y de la cuenta:
    - Preferencia linguistica.
    - Configuracion del pais.
    - Tipo de dispositivo movil.

13. Instagram

Instagram es una red social principalmente visual, donde un usuario puede publicar fotos, videos e historias de corta duracion.

Algoritmo de Instagram

No hay un solo algoritmo que decide lo que se ve, ya que cada parte de la app (el feed, la sección .Explorarz los Reels) tiene su propio sistema de clasificación según cómo es usado.

- Contenido de la publicacion: El algoritmo tiene en cuenta el tipo de contenido, como fotos, videos o historias, asi como hashtags utilizados.
- Recomendaciones personalizadas: El sistema se basa en el comportamiento promedio del usuario. Analiza de forma automatizada interacciones como likes, comentarios y publicaciones guardadas. Por lo tanto, las cuentas con las que más interactúa serán priorizadas en su feed.

Actualmente, el algoritmo de Instagram da prioridad a las publicaciones más recientes.

14. ¿Que podemos aplicar de estas redes en el proyecto LoopWeb?

Muchas de estas redes sociales tienen algoritmos, caracteristicas importantes, por lo tanto se deberian aplicar en **LoopWeb**, principalmente son:

- Feed personalizado de Bluesky
  Es muy importante la opcion que ofrece esta red social, en la que el usuario puede personalizar el feed a su gusto, ya que entrega una navegacion mas amigable con cada usuario.
- Personas que quizas conozcas de Facebook e Instagram
  Al tener una red social es muy importante las recomendaciones de amigos que se le hacen al usuario.Y esto es algo que Facebook logra satisfactoriamente, ya que enfoca sus recomendaciones en amigos de amigos, edad, ciudad, grupos, lo cual aumenta la posibilidad de que la sugerencia de amistad pueda ser del agrado del usuario.
- Limite de caracteres de Bluesky y X
  Tener un limite de caracteres es necesario para que no se haga tediosa la navegacion por la red social.
- Bloquear, eliminar y agregar usuarios
  Tanto agregar como bloquear y eliminar usuarios es algo fundamental en una red social, para poder mantener el circulo de amigos limpio.

15. Teoria de los grafos en analisis de redes sociales

Imagina las redes sociales como una gigante telaraña donde cada usuarios representa un punto de conexión. La teoría de grafos nos ayuda a entender esta telaraña ¿Cómo? Representando cada usuario como un punto (o nodo) y cada conexión entre usuarios como una línea (o arista). Esto nos permite ver quién está conectado con quién y cómo interactúan en la red.
La teoria de los grafos permite relacionar usuarios que comparten gustos, intereses.

Grafos aplicados a redes sociales

Cuando hablamos de grafos aplicados a redes sociales lo más común es que se usen para “detectar comunidades”. Gracias a los algoritmos podemos ver características, atributos y relaciones que coinciden dentro de un grupo. Cuando se analizan los subgrafos, podremos ver lo vértices que están más relacionados entre sí, y además cómo se relacionan con el resto de vértices.

16. BUSQUEDA DE EFICIENCIA

[investigacion de implementacion de json en c,investigacion sobre bubblesort a mergesort]

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).

¿Por que son importantes?

Dado que a menudo pueden reducir la complejidad de un problema, los **algoritmos de ordenación** son muy importantes en informática. Estos algoritmos tienen aplicaciones directas en algoritmos de búsqueda, algoritmos de bases de datos, algoritmos de estructura de datos y muchos más.

Algunos algoritmos de ordenacion

1. **Selection sort**: Selection Sort o Ordenamiento por Selección, busca el elemento más pequeño en el conjunto de elementos y lo coloca en la posición correcta. Luego, busca el siguiente elemento más pequeño y lo coloca en la siguiente posición correcta. Repite este proceso hasta que todos los elementos estén ordenados.

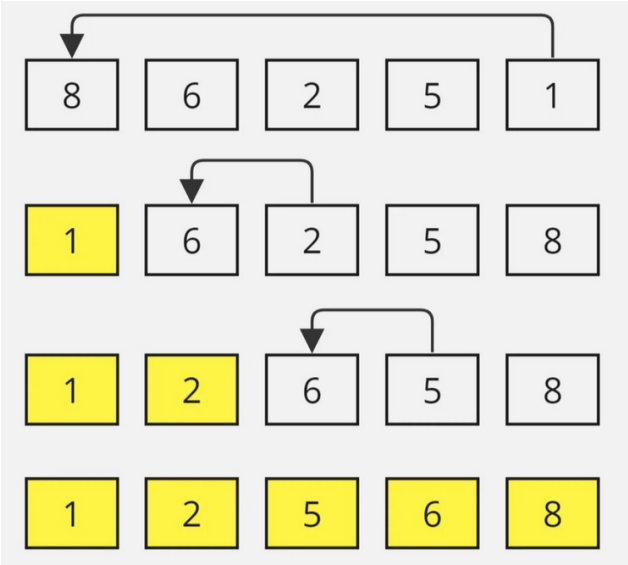

Figura 1: Selection Sort

2. **Bubble sort**: Bubble sort o Ordenamiento de Burbuja, compara pares de elementos adyacentes y los intercambia si están en el orden incorrecto. Repite este proceso hasta que todos los elementos estén ordenados.

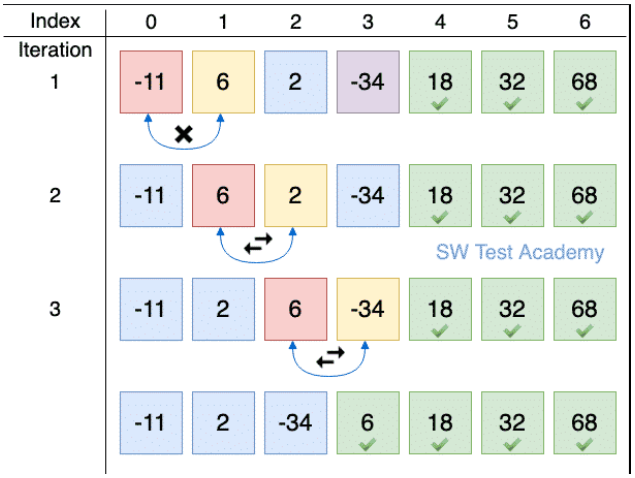

Figura 2: Bubble Sort

3. **Insertion sort**: Insertion Sort o Ordenamiento por Inserción, divide el conjunto de elementos en una parte ordenada y otra desordenada. Toma un elemento de la parte desordenada y lo inserta en la posición correcta en la parte ordenada. Repite este proceso hasta que todos los elementos estén ordenados.

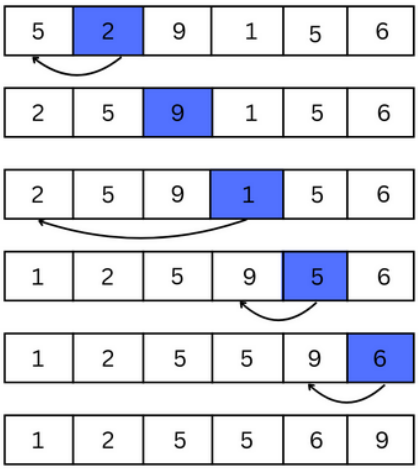

Figura 3: Insertion Sort

4. **Merge sort**: 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 una estrategia de divide y vencerás.

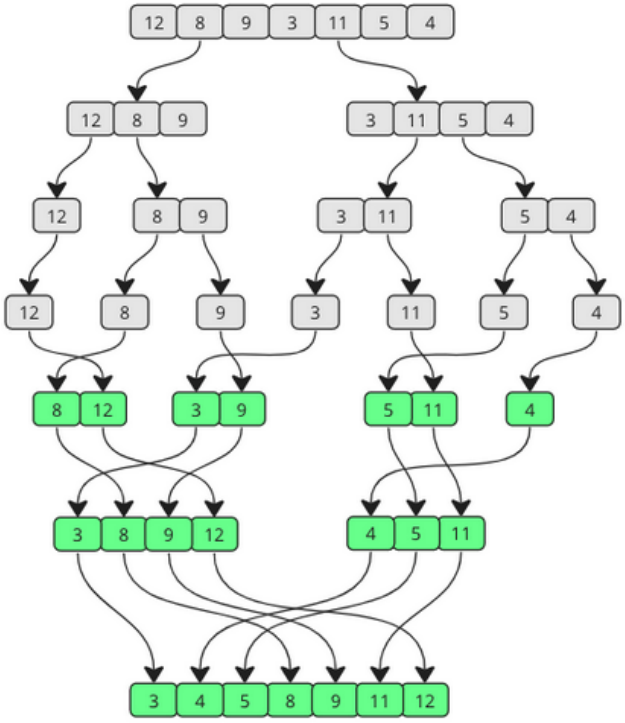

Figura 4: Merge Sort

5. **Heap sort**: Heap Sort o Ordenamiento por Montículos, construye un montículo a partir de los elementos y luego extrae sucesivamente el elemento máximo (o mínimo) del montículo, reajustando el montículo después de cada extracción. El resultado final es un conjunto de elementos ordenados.

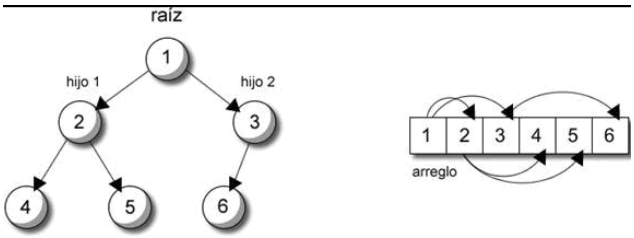

Figura 5: Heap Sort

6. **Quick sort**: Quick Sort o Ordenamiento Rápido, elige un elemento llamado ”pivotez divide el conjunto en dos subconjuntos, uno con elementos menores que el pivote y otro con elementos mayores. Luego, aplica el mismo proceso de forma recursiva en cada uno de los subconjuntos. Este algoritmo también utiliza la estrategia de divide y vencerás.

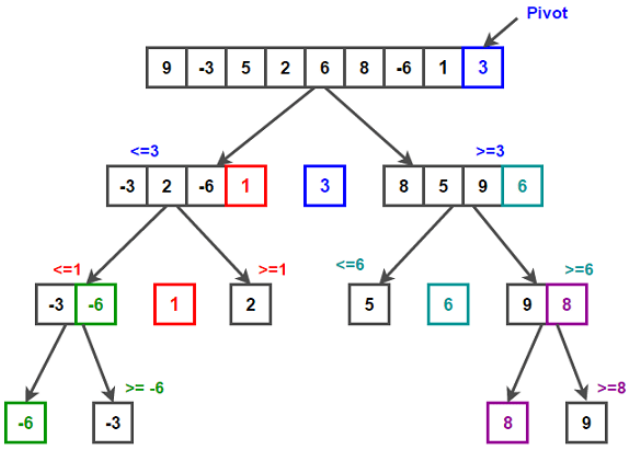

Figura 6: Quick Sort

17. JSON

JSON sus siglas en inglés son por JavaScript Object Notation. Se trata de un formato para guardar e intercambiar información que cualquier persona pueda leer. Los archivos json contienen solo texto y usan la extensión .json.

¿Para qué se utiliza un archivo JSON?

JSON es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente. Un archivo json puede contener diversos datos, como se muestra en la **Figura 1.**

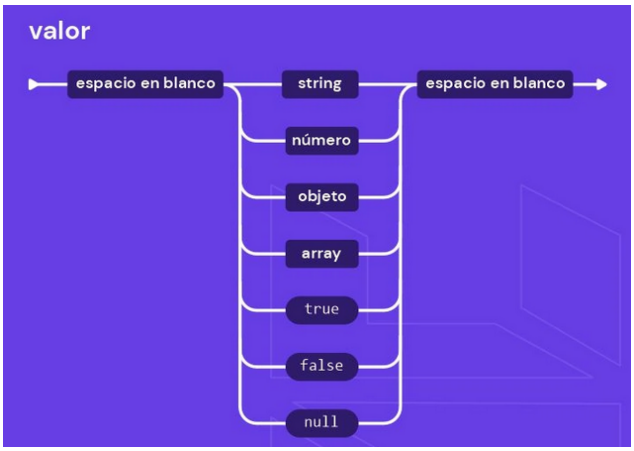

Figura 7: Diagrama json

Tipos de datos que se pueden almacenar en JSON:

- Strings: conjunto de caracteres almacenado como objeto.
- Numero: se utilizan para datos numericos (edad,peso,etc).
- Objeto: estructura de datos que permite almacenar un conjunto de pares clave-valor. Es una manera de organizar y representar datos relacionados entre sí de forma estructurada.
- Array: colección de datos del mismo tipo. Sirve para manejar un número “n” de elementos en común.
- Booleanos: valores que son true o false.
- Null: ausencia de un valor o dato.

Json aplicado en codigo

```
[
    {
        "nombre": "Juan Perez",
        "curso": "septimo basico",
        "edad":"12",
        "contacto":{
            "correo": "juanperez@gmail.com",
            "movil": "+569 12345678"
        },
        "asignaturas": ["matematicas","lenguaje","fisica"],
        "asistencia":false,
        "deportes":NULL
    }
]
```

Listing 1: Ejemplo de .json

Este .json contiene informacion sobre un objeto, en este caso es un estudiante, el cual almacena: nombre, curso, edad, contacto, asignaturas cursadas, asistencia diaria, deportes a los cuales asiste. Tambien se podria utilizar un archivo .txt, pero tiene algunas desventajas ante un archivo json.

Ventajas de utilizar un archivo json

- Estructura bien definida: json tiene una estructura estandar, basada en objetos y arreglos.
- Compatibilidad universal: ampliamente utilizado, soportado en la mayoria de lenguajes de programacion.
- Facil de entender y modificar: existen muchas bibliotecas que permiten leer y escribir json.

Desventajas de utilizar un archivo de texto

- Falta de estructuras: es un archivo de texto plano.
- Sin soporte directo para datos complejos: no puede manejar directamente arreglos, booleanos, etc.
- Mas propenso a errores: al no tener una estructura definida es mas probable cometer errores al guardar, agregar o eliminar datos.

¿Como utilizar json en c?

Al estar trabajando en lenguaje c, no es posible aplicar directamente archivos .json, por lo tanto se debe encontrar la forma de poder leer .json en c. Para hacer esta tarea posible existen librerias externas las cuales se deben instalar, algunas de estas son:

1. Jansson
2. Json-c
3. cjson

En este caso se trabajara con la libreria Jansson.

18. Libreria Jansson

Jansson es una libreria de C para decodificar y manipular datos JSON.Se caracteriza por ser simple e intuitivo, sin dependencias de otras librerias y posee una documentacion completa.

Funciones de Jansson

Posee diversas funciones, en este caso no se utilizaran todas. Las funciones mas importantes que se utilizaran en este codigo son:

1. **json_error_t**: 'declara' una especie de 'variable' que almacena informacion sobre cualquier error de analisis que pudiese ocurrir al leer el archivo JSON, necesario al trabajar con json_loadf y otras funciones.

```
        json_error_t error;

```

Listing 2: json_error_t

2. **json_t**: se utiliza para 'declarar' un json, siempre a travez de un puntero.

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

```

Listing 3: json_t

3. **json_loadf(FILE *input, unsigned flags, size_t *flags_out);**: encargada de cargar el contenido del archivo JSON en una 'variable' del tipo json_t. Pide tres parametros:

- FILE *imput: archivo .json que se utilizara.
- unsigned flags: banderas con las que cuenta la libreria jansson.
- size_t *flags_out: puntero a una variable (puede ser NULL)

```
1        json_t *json = json_loadf(archivo,0,&error);
2
```

Listing 4: json_loadf

4. **json_object_get(json_t *obj, const char *key)**: se utiliza para acceder a los valores de un objeto json, sus parametros son:
   - json_t *obj: puntero a un objeto json.
   - const char *key: dato que se quiere obtener, por ejemplo 'nombre'.

```
1        json_object_value(alumnos_json,"nombre");
2
```

Listing 5: json_object_get

5. **json_string_value(json_t *json)**: extrae el valor de una cadena de texto, necesita un unico parametro (de donde se quiere obtener el string).

```
1        json_string_value(json_object_value(alumnos_json,"nombre");)
2
```

Listing 6: json_string_value

   En este caso se esta obteniendo la cadena de texto 'nombre' que se encuentra en el json alumnos.

6. **json_interger_value(json_t *json)**: obtiene el valor de un dato de tipo entero, solo pide un unico parametro (de donde se quiere obtener el string).

```
1        json_interger_value(json_object_value(alumnos_json,"edad");)
2
```

Listing 7: json_interger_value

   En este caso se esta obteniendo el entero 'edad' que se encuentra en el json alumnos.

19. ¿Como aplica esto en loopweb?

Los algoritmos de ordenacion son de suma importancia, actualmente se esta utilizando **bubble sort**, pero este algoritmo es conocido por ser lento e ineficiente en la mayoria de los casos, especialmente cuando se enfrenta a conjunto de datos

grandes, por lo tanto en este caso no es recomendable utilizarlo, pero hay otro algoritmo que si puede ser muy efectivo, este algoritmo es **merge sort** el cual es de los mas eficientes algoritmos de busqueda y ordenamiento, es a menudo la mejor opción para ordenar una lista enlazada.

Referencia merge sort en codigo

```
    void merge(int arr[], int l, int m, int r) {
        int i, j, k;
        int n1 = m - l + 1; ///< Numero de elementos en el subarreglo izquierdo.
        int n2 = r - m; ///< Numero de elementos en el subarreglo derecho.
        int L[n1], R[n2]; ///< Subarreglos izquierdo y derecho.

        for (i = 0; i < n1; i++){
            L[i] = arr[l + i]; ///< Copia de datos al subarreglo izquierdo.
        }
        for (j = 0; j < n2; j++){
            R[j] = arr[m + 1 + j]; ///< Copia de datos al subarreglo derecho.
        }
        i = 0; ///< Indice para el subarreglo izquierdo L.
        j = 0; ///< Indice para el subarreglo derecho R.
        k = l; ///< Indice para el arreglo original arr.

        // Fusion de los subarreglos
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i]; ///< Toma el elemento de L si es menor o igual que el de R.
                i++;
            } else {
                arr[k] = R[j]; ///< Toma el elemento de R si es menor que el de L.
                j++;
            }
            k++;
        }
        // Copia los elementos restantes de L si los hay
        while (i < n1) {
            arr[k] = L[i]; ///< Copia el elemento de L al arreglo original.
            i++;
            k++;
        }
        // Copia los elementos restantes de R si los hay
        while (j < n2) {
            arr[k] = R[j]; ///< Copia el elemento de R al arreglo original.
            j++;
            k++;
```

```
39          }
40      }
```

Listing 8: merge()

20. ESTRUCUTURAS DE DATOS IMPLEMENTADAS

[¿Por que decidimos implementar estas estructuras de datos y no otra?]

21. ALGORITMOS DE ORDENAMIENTO

¿Por que implementamos cierto algotirmo de ordenamiento y no otro?

22. FUNCIONAMIENTO DE LOOPWEB

[aca iria algo asi como, comentarios, amigos, etc, explicar un poco como funciona]

Comments

Para la lectura de datos de **LoopWeb** se utilizaron las listas enlazadas para ir guardando de comentario en comentario del archivo en el que se lee(importante saber que la separacion de los comentarios es con una linea blanca entera) en el cual tambien se incluyen los tags de **GenreLikes** y **BandLikes** los cuales se guardan por comentario y van guardando su cantidad como el tag en si mismo

curiosidades de codigo

A continuación se hace una mencion de curiosidades del codigo:

1. Isalnum(): es una funcion que devuelve true si el caracter es alfanumerico, esta funcion se utilizo en un proyecto anterior para crear la tokenizacion del indice invertido
2. Creacion de comentarios: la funcion **appendComments** crea un nodo de comentarios y en este se guardan los tags de géneros y bandas
3. Tags: son guardados en arreglos de char de tamaño **MAXTAGS** y cada uno de ellos tiene un tamaño de **MAXTAGLENGTH**

23. FUNCIONES IMPLEMENTADAS

Breve descripcion de funciones implementadas mas extracto de codigo

Funcion GetOTP

getOTP es una funcion que recibe como argumento un string de texto y devuelve un entero que representa el código de acceso al cual queremos entrar. Para esto se utiliza un bucle while que tiene como condición que el caracter que estamos leyendo sea un número distinto a -1 ya que este seria un caracter que no corresponde con un código de acceso. Principalmente la funcion es **getOTP()** pero esta funcion solo se puede usar para condiciones de una palabra, Como por ejemplo **./main.o -a**,**./main.o -u** pero no **./main.o –administrador** o **./main.o –user**. ya que estas opciones son muy largas para la funcion por lo cual si utilizamos la funcion **getoptLong()** podemos ocupar las opciones **./main.o -a**,**./main.o -u** y **./main.o –administrador**,**./main.o –user**

Importante saber para implementar getoptLong

1. Hacer un archivo **getotp.h** prototipo de esta función dado de esta manera: int getoptlong(int argc, char * const argv[],const char *optstring,const struct option *longopts,int *longindex);
2. **Struct option**, este struct se encarga de almacenar toda la información necesaria para poder usar la funcion **getoptLong()**.
3. Dentro del main: hacer el struct option y definir cuales son las opciones a usar y si requieren argumento o no.
4. Al utilizar **./main.o -administrador** tener en cuenta que tiene que ser doble guión, sino puede causar problemas, pero si se quiere utilizar **./main.o –a** no hay problemas dado que tambien lo toma como administrador, al igual que si se deja la palabra incompleta, por ejemplo **./main.o –admi**

Implementacion de mergeSort_genreLinkList

```
1    /**
2    * @brief Divide una lista enlazada en dos mitades.
3    *
4    * @param source Puntero al nodo inicial de la lista a dividir.
5    * @param frontRef Referencia a la primera mitad de la lista.
6    * @param backRef Referencia a la segunda mitad de la lista.
7    */
8   void split_genreLinkList(GenreLinkPosition source, GenreLinkPosition* frontRef,
     GenreLinkPosition* backRef) {
9       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;
    }
```

Listing 9: split_genreLinkList

```
    /**
 * @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;
}
```

Listing 10: fusion de listas

```
    /**
    * @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 */
    }
```

Listing 11: mergerSort_genreLinkList

Codigo para poder abrir un archivo .json en c

```
    /**
 * @brief Funcion para leer un arreglo de strings .json y almacenarlos en una matriz
 *
 * @param array_json  objeto json del tipo arreglo  que contiene los valores se van a leer
 * @param matriz  donde se acumularan los valores leidos del arreglo
 * @return cantidad de elementos leidos
 */
int read_list_json(json_t *array_json, char matriz[MAX_ELEMENTS][MAX_CADENA]) {
    size_t cantidad = json_array_size(array_json);/**<obtener el numero de elementos en el
    arreglo json*/
    if (cantidad > MAX_ELEMENTS) {/**< verificar que no supere la cantidad permitida*/
        printf("max elementos superado\n");
        cantidad = MAX_ELEMENTS;
    }

    for (size_t i = 0; i < cantidad; i++) {
        const char *valor = json_string_value(json_array_get(array_json, i));  /**< obtener
    el valor del string en el indice i de la matriz*/
        if (!valor) {
            printf("Elemento no valido en indice %zu.\n", i);
            return 0;
        }
```

```
        snprintf(matriz[i], MAX_CADENA, "%s", valor); /**<copia el valor del string donde corresponda en la matriz */
    }
    return (int)cantidad;
}
```

Listing 12: read_list_json

```
    /**
 * @brief Funcion para leer el archivo json, en este caso usuarios.json
 *
 * @param nombre_archivo  archivo .json a leer
 * @param usuarios  usuario de la estructura antes creada en el que se guardaran los datos obtenidos
 * @param total_users  puntero a un entero en el que se guardara el numero total de usuarios leidos en el .json
 * @return int
 */
int read_archive_json(const char *nombre_archivo, User usuarios[], int *total_users) {
    FILE *archivo = fopen(nombre_archivo, "r");  /**< Abre el archivo .json en modo lectura */
    if (!archivo) {
        perror("No se pudo abrir el archivo JSON");
        return 1;
    }
    json_error_t error;     /**< declaracion de "variable" error basada en una funcion de la libreria jansson */
    json_t *json = json_loadf(archivo, 0, &error); /**<  puntero en el cual se guarda el archivo .json*/
    fclose(archivo);

    if (!json) {
        printf("Error al analizar el .json: %s\n", error.text);
        return 1;
    }

    *total_users = (int)json_array_size(json);  /**<  tamano total de usuarios basado en el arreglo json, utilizando una funcion de la libreria jansson*/
    for (int i = 0; i < *total_users; i++) {
        json_t *usuario_json = json_array_get(json, i); /**< Obtiene el objeto json, basada en el usuario[i] */

        /**<  Lectura de campos basicos*/
        const char *name = json_string_value(json_object_get(usuario_json, "nombre"));  /**<  almacena el nombre del usuario[i]*/
        json_int_t age = json_integer_value(json_object_get(usuario_json, "edad"));     /**<
```

```
      almacena la edad del usuario[i]*/
31        const char *nationality = json_string_value(json_object_get(usuario_json, "
    nacionalidad"));  /**< almacena la nacionalidad del usuario[i] */
32        json_t *gustos_json = json_object_get(usuario_json, "gustos");  /**< almacena los
    gustos del usuario[i] */
33        json_t *bands_json = json_object_get(usuario_json, "bandas");   /**< almacena las
    bandas del usuario[i] */
34        json_t *friends_json = json_object_get(usuario_json, "amigos"); /**< almacena los
    amigos del usuario[i] */
35
36        snprintf(usuarios[i].name, sizeof(usuarios[i].name), "%s", name);  /**<  copia el
    nombre a la estructura user*/
37        usuarios[i].age = (int)age;                                        /**<  copia la
    edad a la estructura user*/
38        snprintf(usuarios[i].nationality, sizeof(usuarios[i].nationality), "%s", nationality
    );   /**< copia la nacionalidad a la estructura user */
39
40       /**< Lee y almacena las listas de gustos, bandas, amigos en la estructura user */
41        read_list_json(gustos_json, usuarios[i].gustos);
42        read_list_json(bands_json, usuarios[i].bands);
43        read_list_json(friends_json, usuarios[i].friends);
44    }
45
46    json_decref(json); /**< libera la memoria utilizada por el json, funcion de la libreria
    jansson */
47    return 0;
48 }
```

Listing 13: read_archive_json

```
1  #include <stdio.h>
2  #include "funciones.h"
3
4  int main() {
5      User usuarios[MAX_USERS];
6      int total_users = 0;
7
8      if (read_archive_json("usuarios.json", usuarios, &total_users) == 0) {
9          printf("Se leyeron %d usuarios:\n", total_users);
10
11         /**<  imprimir los datos obtenidos de cada usuario*/
12         for (int i = 0; i < total_users; i++) {
13             printf("ID %d:\n", i + 1);
14             printf("Nombre: %s\n", usuarios[i].name);
15             printf("Edad: %d\n", usuarios[i].age);
16             printf("Nacionalidad: %s\n", usuarios[i].nationality);
```

```
            printf("Gustos:\n");
            for (int j = 0; j < MAX_ELEMENTS && usuarios[i].gustos[j][0]; j++) {
                printf("  - %s\n", usuarios[i].gustos[j]);
            }
            printf("Bandas:\n");
            for (int j = 0; j < MAX_ELEMENTS && usuarios[i].bands[j][0]; j++) {
                printf("  - %s\n", usuarios[i].bands[j]);
            }
            printf("Amigos:\n");
            for (int j = 0; j < MAX_ELEMENTS && usuarios[i].friends[j][0]; j++) {
                printf(" ID - %s\n", usuarios[i].friends[j]);
            }
            printf("\n");
        }
    }
    return 0;
}
```

Listing 14: main.c

24. CONCLUSION