Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y sistemas
Estructuras de Datos

Ingenieros:

- Ing. Luis Espino
- Ing. Edgar Ornelis
- Ing. Álvaro Hernández

Auxiliares:

- Kevin Martinez
- Carlos Castro
- José Montenegro

Proyecto Fase 3

Social Structure

Implementación de estructuras no lineales

ÍNDICE

OBJETIVOS

Objetivo general

- Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de soluciones de software.

Objetivos específicos

- Comprender y desarrollar distintos tipos de estructuras no lineales.
- Implementar una aplicación de escritorio utilizando el lenguaje de programación C++.
- Utilizar la herramienta Graphviz para graficar estructuras de datos no lineales.
- Utilizar los conceptos generales de la tecnología blockchain.
- Implementar algoritmos que requieren la utilización de grafos en el desarrollo de la aplicación. .

DESCRIPCIÓN

Con las nuevas funcionalidades integradas y con la interfaz de usuario implementada en la fase 2, muchos usuarios han quedado satisfechos y más usuarios se han sumado a utilizar la aplicación. Debido a la popularidad de la plataforma y en un mundo digital cada vez más expuesto a riesgos, proteger la privacidad y la integridad de la información del usuario es fundamental. Para ello se le solicita implementar una serie de tecnologías que serán clave para el manejo de la seguridad dentro de la aplicación.

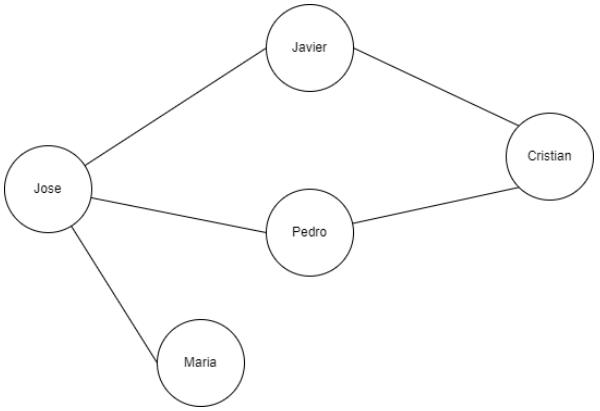
La seguridad en las aplicaciones sociales es esencial no solo para proteger los datos personales, sino también para generar confianza entre los usuarios. Los ciberataques, la filtración de datos y la manipulación de la información son riesgos reales que pueden afectar gravemente la experiencia de los usuarios y la reputación de la plataforma.

Sin embargo, el usuario final rara vez se da cuenta del esfuerzo técnico detrás de la protección de sus datos. A pesar de esto, la implementación de tecnologías avanzadas como la criptografía y el blockchain garantiza que la información personal esté protegida, incluso en situaciones donde las amenazas no son evidentes para el usuario. Esto asegura que, aunque el usuario interactúe de forma sencilla y fluida con la red social, sus datos siempre estarán resguardados contra posibles vulnerabilidades.

Modificaciones

Relaciones de Amistad

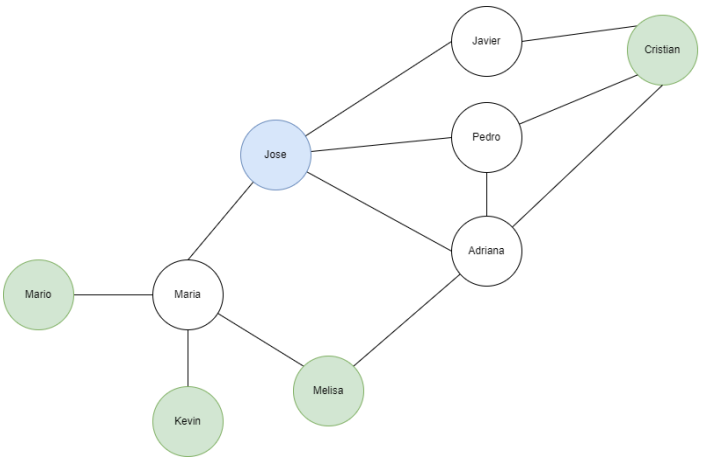
Anteriormente las relaciones se gestionaban por medio de la matriz dispersa, pero se tiene el problema que cada usuario registrado se duplica en la matriz, almacenando al usuario en la fila y también en la columna. Por lo tanto para evitar esa duplicación y desperdicio de memoria se **reemplazará por un grafo no dirigido, representado por una lista de adyacencia**, siendo estos altamente escalables y permiten manejar redes con millones de usuarios y relaciones.



Cada vértice(nodo) representa un usuario y las aristas representan las relaciones de amistad entre usuarios. Por lo tanto en la imagen anterior Jose es amigo de Maria, Pedro y Javier, mientras que Pedro tiene de amigo a Jose y Cristian.

Sugerencias de Amistad

Se agrega la funcionalidad de recomendar amigos, donde se deberá de explorar las distancias en el grafo, buscando usuarios que están a una distancia de dos saltos (es decir, amigos de amigos). Donde se deben priorizar las sugerencias, cuantos más amigos en común tenga un usuario con otro, mayor será la probabilidad de que esa sugerencia sea relevante. Para eso se debe ponderar las sugerencias en base a la cantidad de amigos que tengan en común, para mostrar el listado de usuarios sugeridos empezando con los de mayor ponderación.



En la imagen, los amigos sugeridos de Jose en orden ascendente serían los siguientes:

- Cristian - 3 amigos en común
- Melisa - 2 amigos en común
- Kevin - 1 amigo en común
- Mario - 1 amigo en común

Compresión y descompresión de la información

Dentro de la aplicación se podrán escribir archivos con información específica, estos archivos deben ser comprimidos y descomprimidos utilizando el **método de Huffman**.

La información que se deberá almacenar será una copia de seguridad de las estructuras que se encuentren involucradas en el almacenamiento de la información correspondiente a los usuarios y sus amigos, la información a comprimir serían:

- Usuarios
- Amigos
- Solicitudes recibidas
- Solicitudes enviadas

Cada vez que un usuario cierre sesión, se debe de comprimir la información mencionada anteriormente, al abrir la aplicación se debe descomprimir y cargar la información a las estructuras. Los archivos deberán tener la extensión **.edd**.

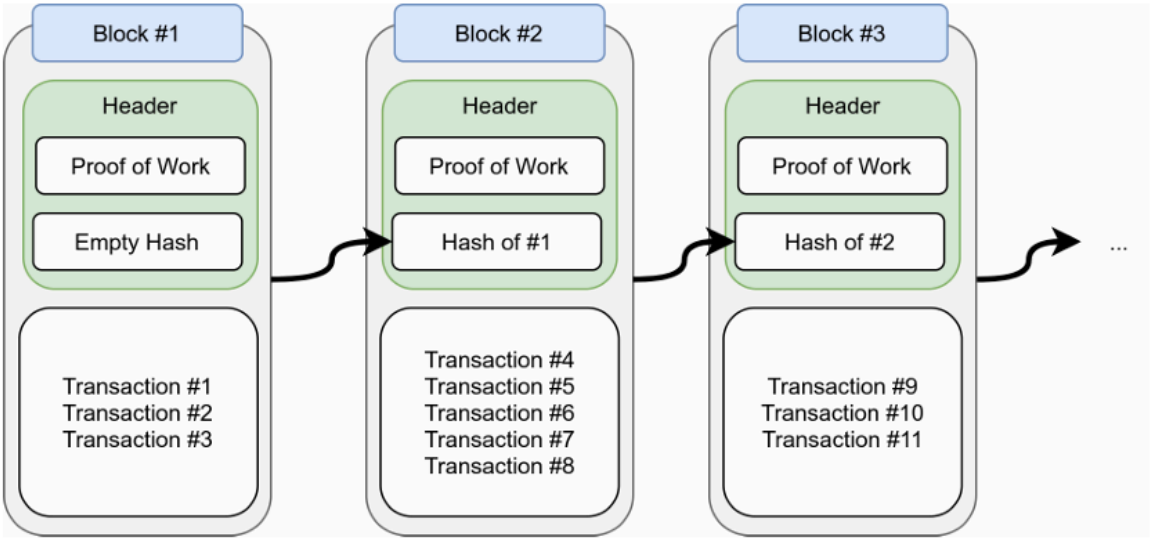
Usuario

Para esta fase se requiere agregar la funcionalidad de **modificar perfil**, donde cada usuario podrá modificar su nombre, apellidos, fecha de nacimiento o contraseña.

Seguridad

Blockchain

Esta es una estructura de bloques, los cuales se comportan como una lista enlazada simple, en donde cada uno de los nodos(bloques) se ingresan por delante.



Prueba de trabajo: Es el proceso por el cual se encuentra un hash que cumpla con la condición de tener un prefijo de n ceros, donde n corresponde a un número entero, para el caso de la aplicación este poseerá un valor de 4. Para esto se debe iterar un número entero (NONCE) el cual iniciará desde cero y se incrementará en 1 hasta encontrar un hash válido para el bloque.

Generación de nuevos bloques: Un nuevo bloque se deberá generar cada vez que se cree una publicación o un comentario, almacenando en el campo DATA todas las publicaciones existentes con sus comentarios.

Bloque
El bloque se compone de distintas partes que contienen información acerca del mismo, estas partes son:

- **INDEX**: Representa el número del bloque que se estará insertando en la cadena, este campo es incrementado en uno para cada bloque que se inserta por lo cual, naturalmente, el primer bloque tendrá un índice de cero y los demás tendrán índices con valores sucesivos (1,2,3,etc.).
- **TIMESTAMP**:Tendrá la fecha y hora en la cual se creó el bloque, poseerá el siguiente formato: (DD-MM-YY-::HH:MM:SS).
- **DATA:**Este campo poseerá toda la información de las publicaciones(lista doble) y comentarios.

- **NONCE**: Será el número entero que se debe iterar de uno en uno hasta encontrar un hash que cumpla con la prueba de trabajo, es decir, que contenga el prefijo con la cantidad de 0s designada (cuatro).
- **PREVIOUSHASH**: Corresponde al hash del bloque anterior a cada uno de los bloques. Esto actúa como un apuntador que indica si los datos en algún punto de la cadena de bloques fueron corrompidos, en el caso del primer bloque no se posee ningún bloque previo por lo cual el valor de este apuntador será “0000”.
- **HASH**: El hash que identifica al bloque actual y protege la data que este almacena de ser corrompida. Este hash se genera aplicando la función SHA256 a las propiedades: INDEX, TIMESTAMP, NONCE, ROOTMERKLE y PREVIOUSHASH. Todas deben ser cadenas concatenadas sin espacios en blanco ni saltos de línea.

SHA256(INDEX + TIMESTAMP+ NONCE + ROOTMERKLE+ PREVIOUSHASH)

Estructura de un bloque(json)


```
{
  "INDEX": 0,
  "TIMESTAMP": "21-09-2024::11:30:29",
  "NONCE": 114706,
  "DATA":[{
      "correo": "maria@gmail.com",
      "contenido": "Holaaaaaa",
      "fecha": "19/08/2024",
      "hora": "10:00",
      "comentarios":[{
        "correo": "pedro@gmail.com",
        "comentario": "Como te va",
        "fecha": "20/08/2024",
        "hora": "11:00"
      },{
        "correo": "pedro@gmail.com",
        "comentario": "Como te va",
        "fecha": "20/08/2024",
        "hora": "11:00"
      }]
    },
    {
      "correo": "jose@gmail.com",
      "contenido": "Tengo hambre, que recomiendan?",
      "fecha": "19/08/2024",
      "hora": "10:00",
      "comentarios":[]
  }],
  "PREVIOUSHASH": "0000",
  "HASH": "00003d3b856ea2d3ffdd32bd820e7448acf217974b752b469b4bdce206767c7e"
}
```

Almacenamiento de bloques

Los bloques deben ser almacenados en archivos JSON, para que estos puedan ser leídos al iniciar la aplicación y poder tener **persistencia de la información** de las publicaciones. Para esto se deberá de tener una carpeta en la raíz de su proyecto que llevará por nombre blockchain. En esta carpeta se deberá almacenar todos los bloques que se generen durante la ejecución de la aplicación. Estos bloques deberán ser leídos al momento de iniciar la aplicación.

Árbol de Merkle

Un árbol de Merkle es una estructura de datos en forma de árbol binario en el que cada nodo hoja contiene un valor hash de un bloque de datos, y cada nodo intermedio contiene el hash combinado de sus nodos hijos. Al final, el nodo raíz del árbol (conocido como Merkle Root) representa un resumen criptográfico de todo el conjunto de datos.

Por lo tanto el **administrador** tendrá la opción de generar el árbol del último bloque agregado en el blockchain, donde cada publicación con sus comentarios del campo DATA del bloque será un nodo hijo del árbol de Merkle. Tomando como referencia la imagen de *Estructura de un bloque*, el árbol tendría 2 nodos hoja(uno para Maria y otro para José).

Ejemplo:
hash = funcionHash(Data)

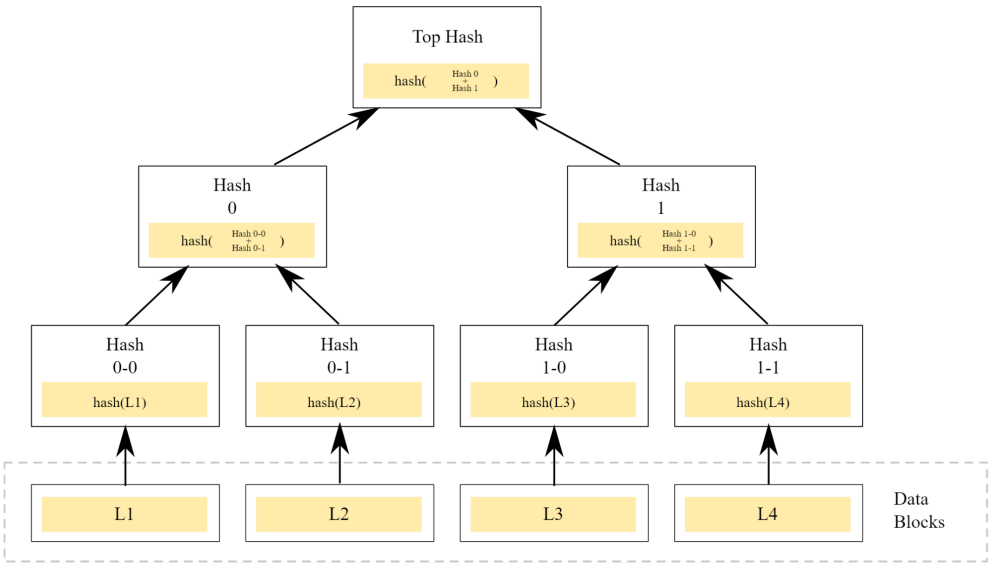
- **Hash**:Será el identificador correspondiente a la data que se almacenará en los nodos hoja del árbol de merkle.
- **FunciónHash**: Es el método utilizado para encriptar la información que servirá de entrada para la función, para esto se utilizará Sha256.
- **Data**: Es la publicación con sus comentarios.

  Ejemplos de nodos hoja

  hash = sha256(DATA)
  hash = 39b923082a194166d8d92989116bd

  Ejemplos de nodos internos

  hash = sha256(“39b923082a194166d8d92989116bd”+
  “aa17d5fdf24761b54ad640691146656095b”)
  hash = 113b685b12dbfa76c04bec7759a24ba66dd



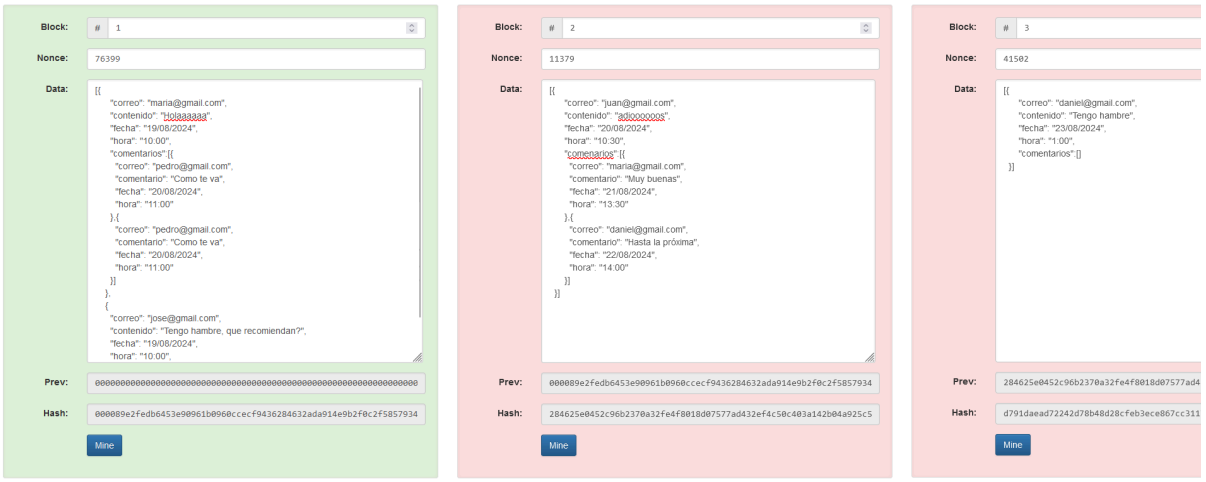
Protección de datos

Desde el inicio de la aplicación se ha almacenado la información sensible de los usuarios (contraseñas) de forma no segura, por lo cual es necesario corregir esta debilidad e implementar buenas prácticas de seguridad para salvaguardar dicha información.

Para lograr este cometido se deben hacer uso de la función hash criptográfica **SHA256** en las contraseñas de los usuarios al momento de realizar una carga masiva, registrar un usuario o cuando el usuario edite su contraseña.

Además de implementar medidas de seguridad en los datos sensibles de los usuarios, se debe asegurar que los datos de los mismos (datos de usuario y amigos) no sean modificados desde el archivo JSON en el cual se están almacenando, es por esto que la Blockchain debe ser capaz de identificar esta manipulación en los datos y debe ser representada en el grafo de la misma.

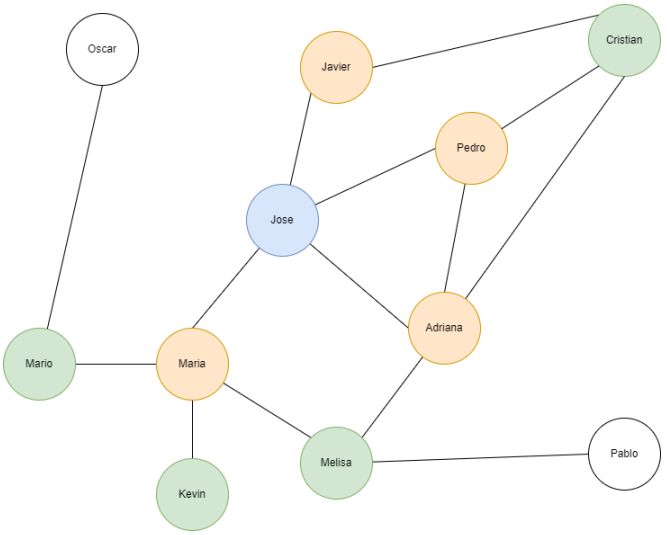
Esta manipulación en los datos (si es que hubiere) se deberá representar coloreando el nodo de la blockchain que fue manipulado de color rojo, si por el contrario no fue manipulado se deberá colorear de verde.



Reportes

Usuario

- Sugerencias de amistad en la interfaz
- Gráfico de grafo de amistades y sugerencias de amistad, diferenciando por colores el usuario, sus amigos y las sugerencias de amistad. Como en la siguiente imagen:



Administrador

- Gráfico de lista de adyacencia de amistades en la aplicación
- Gráfico de grafo que representa las amistades en la aplicación
- Gráfico de Blockchain
- Gráfico de árbol de Merkle de un determinado usuario

Observaciones

- Lenguaje de programación a utilizar: **C++**
- El nombre del usuario administrador será **admin@gmail.com** y su contraseña será **EDD2S2024**.
- Sistema Operativo: Libre
- IDE: Libre
- GUI: Libre
- Herramienta para desarrollo de reportes gráficos: **Graphviz**.
- Durante la calificación se harán preguntas para validar que el estudiante realizó el proyecto, de no responder correctamente anulará la nota obtenida en la o las secciones en la que se aplique tal concepto.
- Cada estudiante deberá utilizar un repositorio de github con el nombre **[EDD]Proyecto_carnet,** ir agrupando cada fase por carpetas dentro del mismo repositorio.
- Apartado de entrega en la plataforma UEDI:
    - Fecha y hora de entrega: **21 de octubre a las 23:59 horas.**
- Es **obligatorio** implementar una interfaz gráfica para tener derecho a calificarse.
- Todas las estructuras mencionadas anteriormente deben ser implementadas por el usuario y no podrá usar ninguna librería de C++.
- Las copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.

Entregables

- Link a repositorio
    - Código fuente
    - Manual de Usuario
    - Manual Técnico