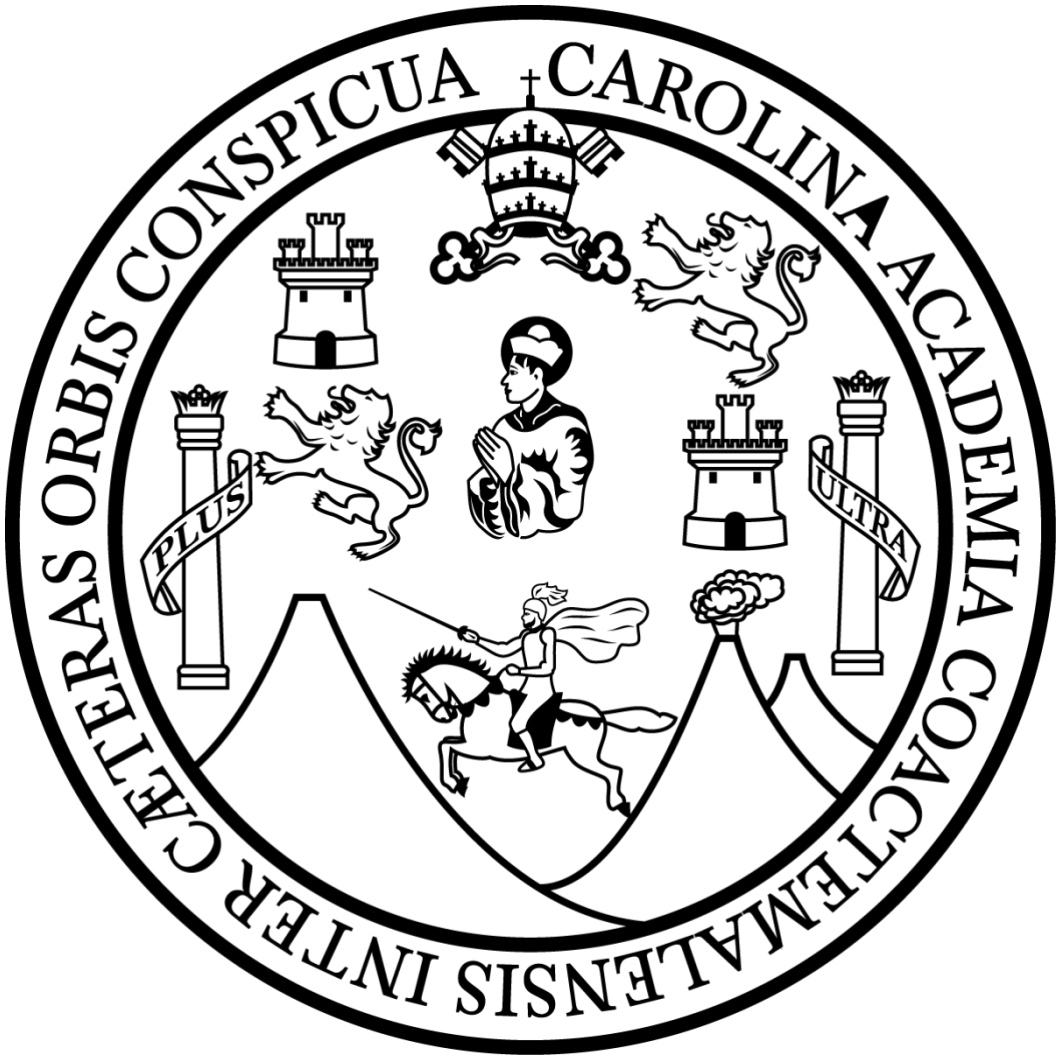

MANUAL TÉCNICO

PROYECTO 2

Creado por: Dayana Alejandra Reyes Rodríguez

Objetivos y alcances del sistema

Objetivo

Crear una aplicación web capaz de almacenar información para el uso de una plataforma de peliculas

Alcance

Utilizar estructuras de datos lineales y no lineales para el almacenamiento de información

Requerimientos del programa

Windows 10 (8u51 y superiores)

RAM: 128 MB

Espacio en disco: 124 MB para JRE;

Procesador: Mínimo Pentium 2 a 266 MHz

Exploradores: Internet Explorer 9 y superior, Firefox

Resolución de la pantalla: 1024 x 728

Lógica del Programa

El programa fue realizado con el lenguaje de programación JavaScript, y utilizando HTML y CSS para el diseño de la interfaz

El programa cuenta con las siguientes estructuras:

- Lista Enlazada Simple
- Árbol B
- Árbol AVL
- Árbol de merkle
- Blockchain
- Tabla hash

Lista Enlazada Simple

se utiliza para guardar Clientes, los nodos cuentan con los siguientes atributos

- dpi
- nombre
- usuario
- correo
- contra
- telefono

cuenta con los siguientes métodos:

AgregarUusuario: Agrega nodos de clientes al final de la lista asegurándose de que los usernames no sean repetidos para poder ingresarlos

Graficar: Crea el archivo.dot para la gráfica de los clientes pasando el dot a la librería d3

Árbol B:

Este Árbol B guarda información de los actores, los nodos se guardan de acuerdo con su dni comparando cual es mas grande o mas pequeño. Los atributos de sus nodos son:

- dni
- nombre
- correo
- descripción
- izquierda
- derecha

y del árbol es raíz y contador. Para llenar y graficar el árbol se utiliza la recursividad, sus métodos son:

AgregarArbol:Manda información para un nuevo nodo al método Insertar

Insertar: Utiliza la recursividad para analizar en donde debe de ir el nodo deprendiendo de su nombre

GraficarArbol: Escribe el .dot para graficar el Árbol

EscribirNodos: Escribe los nosdos para el .dot

EscribirEnlaces: Enlaza los nos para el .dot

PreOrder: manda la raíz para el método RecorrerPreOrder y lo guarda en una variable

RecorrerPreOrder: Recorre el árbol en preorden guardando en una variable el recorrido

InOrder: manda la raíz para el método RecorrerInOrder y lo guarda en una variable

RecorrerInOrder: Recorre el árbol en inorder guardando en una variable el recorrido

PostOrder: manda la raíz para el método RecorrerPostOrder y lo guarda en una variable

RecorrerPostOrder: Recorre el árbol en postorder guardando en una variable el recorrido

Árbol AVL:

Este Árbol AVL guarda información de las películas, los nodos se guardan de acuerdo con su id comparando cual es mas grande o mas pequeño. Los atributos de sus nodos son:

- id
- pelicula
- descripcion
- star
- precio
- paginas
- categoría
- izquierda
- derecha
- altura

y del árbol es raíz y contador. Para llenar y graficar el árbol se utiliza la recursividad, sus métodos son:

AgregarNodoAVL:Manda información para un nuevo nodo al método InsertarAVL

InsertarAVL: Utiliza la recursividad para analizar en donde debe de ir el nodo deprendiendo de su nombre y manda a llamar sus rotaciones dependiendo de la altura del nodo

Altura: devuelve la altura del nodo y si esta vacio regresa -1

Maxima: devuelve la altura del nodo mas grande entre 2 nodos

SRR: hace una rotación simple a la derecha

SRL: hace una rotación simple a la izquierda

DRR: hace una rotación doble a la derecha

DRL: hace una rotación doble a la izquierda

GraficarAVL: Escribe el .dot para graficar el Árbol

EscribirNodos: Escribe los nosdos para el .dot

EscribirEnlaces: Enlaza los nos para el .dot

Tabla hash:

Esta tabla hash guarda información sobre las categorías de las peliculas, el nodo de la tabla tiene atributos:

- idcategorias
- company

La lista hash tiene los atributos

- primero
- tam

Y cuenta con los métodos:

AgregarLista: agrega un nodo hash a la lista

Vacia: devuelve si la lista esta vacia

La tabla hash cuenta con los siguientes atributos:

- ocupación
- tamanio
- tabla

En el constructor se agrega a tabla una lista hash dependiendo de tamanio, cuenta con los métodos:

Insertar: se inserta un valor en la lista hash en la tabla, la posición de la tabla depende de hacer una operación la cual es idcategoria modulo tamaño de la tabla

Rehash: calcula la ocupación de la tabla y si es mayor del 75% reinicia la tabla y haciendo la tabla de un tamaño de los índices ocupados *5

GraficasTabla: genera el dot para graficar la tabla

Árbol Merkle:

Guarda información del alquiler de las peliculas. Contiene:

NodoData

HashNodo

ArbolMerkle

El nodo data contiene los atributos

- cliente
- película

HashNodo contiene los atributos

- hash
- izquierda
- derecha
- cuenta

ArbolMerkle contiene los atributos:

- top
- size
- datablock
- contador

Y tiene los métodos:

Add: agrega nodos NodoData a datablock y aumenta su contador

Creararbol: crea la raíz con hash 0 y manda a crear el árbol a árbol con hash 0

Árbol: crea todos los nodos del árbol dependiendo del tamaño de exponente,

Hashvalores: con el árbol ya creado empieza a crear hash para cada uno, haciendo un recorrido postorden para poder ir de abajo hasta arriba, para los nodos hoja crea un hash con la información sha256(cliente – película), para los nodos padres crea un hash con la información sha256(hash izquierda + hash derecha)

Auth: verifica de que tamaño debe de ser el árbol para que sea par y manda a crear el árbol y a hashearlo

Graficar: escribe el .dot para graficarlo

Escribirgraf: escribe los nodos y enlaces para el .dot

Blockchain:

Contiene:

BloqueBC

NodoBC

BC

BloqueBC contiene sus atributos:

- id,
- tiempo,
- data,
- nonce,
- previoshash,
- rootmerkle,
- hash

NodoBC contiene los atributos:

- bloque
- siguiente
- anterior

BC contiene los atributos

- Primero
- Ultimo

Y sus métodos:

CrearBloque: obtiene el día, el mes, el año, hora, minuto,segundo

Obtiene el hash anterior, crea un nuevo hash con la información de: id,fecha,data,rootmerkle,previoshash, y crear un nodo BloqueBC y lo manda Agregar

Agregar: funciona como una lista, agrega el nodo BloqueBC mandado y lo guarda de ultimo como NodoBC

Vacio: devuelve si primero es null

Graficar: crea el .dot para graficar

Se creo una variable la cual llama a setInterval para que cada 300 segundos se cree un nuevo bloque

Para Graficar las estructuras se utilizó la Librería Graphviz, escribiendo el .dot de las estructuras y enviándolas, haciendo uso de la librería d3 la cual ayuda a JavaScript a mostrar las gráficas en el HTML

Para hacer el hash se utilizó la encriptación sha256 agregada en una clase llamada sha256.js y exportada como script al documento original, esta clase fue creada por el ingeniero Luis Espino