**UNIVERSIDAD DE INGENIERIA Y TECNOLOGIA**

**ALGORITMOS Y ESTRUCTURA DE DATOS**

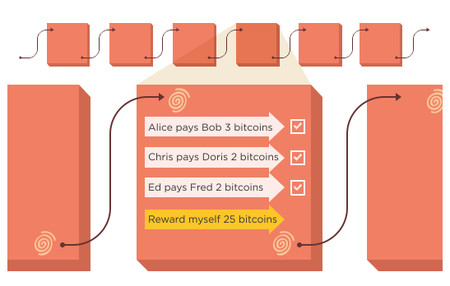
**PROYECTO**

# Consideraciones generales

* Debe usar sus propias estructuras de datos implementadas en el curso: listas, array circular, bst, hash, etc.
* Debe construir las estructuras de datos de la manera más desacoplada posible. Hacer uso de plantillas (templates), de tal forma que pueda utilizarlos dentro de su aplicación sin necesidad de modificar el código interno de las estructuras, sólo instanciar objetos de estas para utilizarlos o extender la funcionalidad de una estructura mediante herencia.
* Las estructuras pueden utilizar lambdas para implementar los criterios de comparación.
* El proyecto debe hacer uso de archivos para guardar y cargar datos desde disco duro.
* La interfaz gráfica puede ser por consola o GUI (Qt Project, Visual Studio, WxWidgets)
* En el caso de evidenciar plagio con otros grupos, se asigna una calificación de cero (0) a todos los integrantes del grupo.
* La calificación en la exposición es individual por cada integrante de grupo, por lo que es mandatorio que todos conozcan el trabajo.
* La exposición se realizará con cámara activa, los que tuvieran algún impedimento (salud y/o otros motivos) indicarlo por correo antes de la exposición.
* La aplicación debe funcionar correctamente para que se considere el 100% de la calificación.

# Enunciado del Trabajo Final

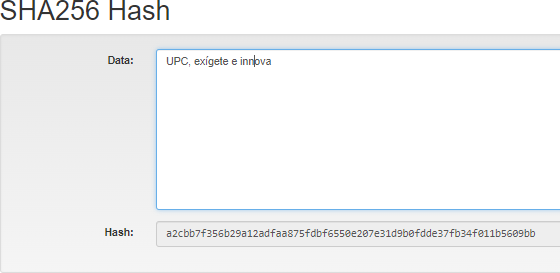
El proyecto consiste en desarrollar una aplicación transaccional de interacción de datos teniendo como base de seguridad una estructura de datos de cadena de bloques (Blockchains en un solo host).



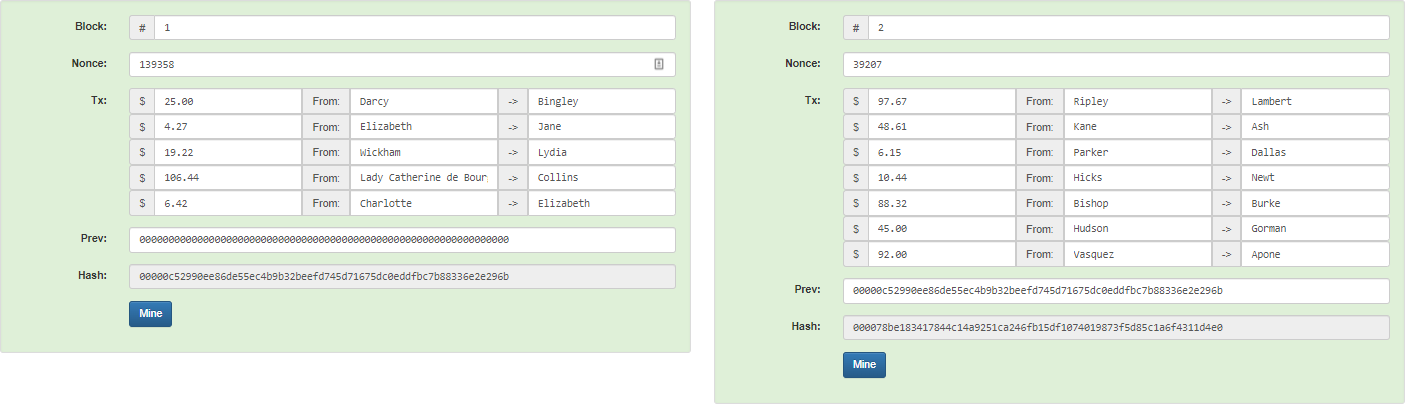
https://www.youtube.com/watch?v=C5NZnD12yjg

# 2.1 Cadena de bloques (Blockchains)

* La estructura de datos del blockchains es una lista de bloques de registros ordenados cronológicamente.
* Cada bloque es identificable por un código hash, generado mediante un algoritmo de hash criptográfico a partir de toda la información que contiene el bloque. De esta manera, el código hash también servirá para verificar la integridad del bloque.



* Debe mantenerse las propiedades del hash: indexación eficiente en el rendimiento de las búsquedas, libre de colisiones y estable.
* Cada bloque hace referencia a un bloque anterior en el campo "Prev", también conocido como bloque padre.

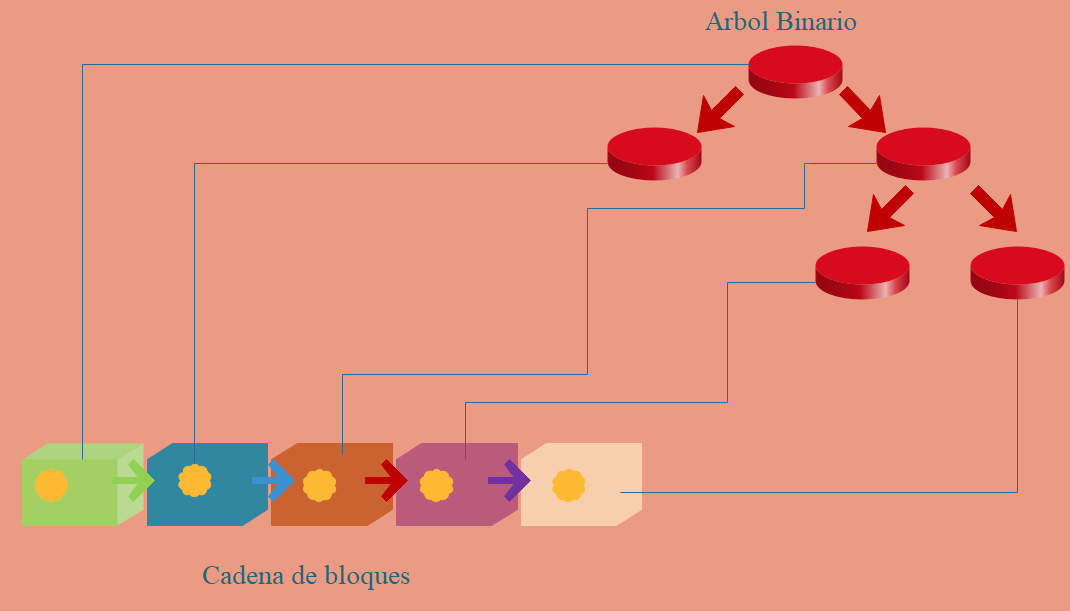


Fuente: https://andersbrownworth.com/blockchain/

* La identidad propia del bloque hijo cambia si la identidad del padre cambia.
  + Si por alguna razón el contenido del bloque padre se modifica, el código hash del padre también cambia.
  + El cambio de hash del padre requiere una alteración en el puntero "Prev" del hijo.
  + Esto, a su vez, hace que el hash del hijo mute, lo que requiere un cambio en el puntero del nieto, que a su vez altera al nieto y así sucesivamente.
* Este efecto en cascada garantiza que, una vez que un bloque tiene muchas generaciones que le suceden, no pueda cambiarse sin **forzar consecuentemente un recálculo** de todos los bloques posteriores (**proof of work**).
  + Por ende, la existencia de una larga cadena de bloques fortalece la seguridad que sea inmutable.
* Se debe investigar e implementar alguna técnica de “**proof of work”.**

# 2.1 Aplicación

Como caso aplicativo, se requiere la implementación de un sistema transaccional que permita a un usuario de sistema registrar operaciones de manera segura (blockchain) y luego realizar búsquedas de manera eficiente usando diversas técnicas de indexación y búsqueda (bst, hash, heap, etc).



**Requerimientos de la aplicación:**

* La aplicación debe soportar registro de usuarios (login).
* Carga de datos: la aplicación debe cargar los datos transaccionales desde archivos de texto, desde una ruta determinada. Puede trabajar con [algún dataset de Kaggle](https://www.kaggle.com/datasets/apoorvwatsky/bank-transaction-data) o generar datos ficticios. Ejemplos de registros transaccionales:
  + Retiros de dinero: cliente, lugar, monto, fecha
  + Transferencias bancarias: emisor, receptor, monto, fecha
  + Registro de ventas: cliente, vendedor, monto, fecha
* La aplicación deberá permitir el ingreso de nuevas transacciones.
* Por cada criterio de búsqueda, se debe indexar los bloques en un árbol binario o un hash-table de acuerdo al tipo de filtrado requerido:
  + Igual a X
  + Entre X y Y
  + Inicia con
  + Está contenido en
  + No está contenido en
* Realizar cálculos en las transacciones por usuario. Por ejemplo: cálculo del monto total recaudado por el usuario X, la cantidad de clientes registrados por el usuario X.
* La aplicación permite la persistencia de datos en memoria secundaria (archivos binarios).

# Entrega

* El trabajo se realizará de manera grupal. Se debe asignar las tareas y hacer seguimiento a su cumplimiento.
  + Utilizar [Project boards de GitHub](https://docs.github.com/en/issues/organizing-your-work-with-project-boards).
* Elaborar un pequeño informe con la siguiente estructura:
  + Introducción
  + Descripción del caso de estudio (en base al dataset elegido) planteado por el grupo
  + Diagrama de clases de las entidades principales.
  + Definición de cada una de las estructuras de datos utilizada
  + Complejidad en notación Big O de los principales métodos de las estructuras de datos.
  + Conclusiones
  + Referencias bibliográficas
* Anexar al informe la lista de actividades concluidas por cada integrante del grupo.
* El informe se subirá al aula virtual, así como el enlace del repositorio Github asignado para grupo.

# Presentación

Para obtener la calificación completa indicada en la rúbrica, cada estudiante debe participar de la presentación del proyecto y responder las preguntas planteadas, además que la aplicación debe funcionar a un 100%. La exposición tendrá una duración máxima de 10 minutos en donde deben participar todos los integrantes, luego tendrán una ronda de preguntas.

Presentación en PowerPoint conteniendo no más de 10 slides:

* Descripción del problema y objetivos
* Elaboración de la solución (gráficos)
* Conclusiones