

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Estructuras de Datos
Proyecto #1
Ingenieros

- Ing. Luis Espino
- Ing. Jesús Guzmán
- Ing. Álvaro Hernández

Auxiliares

- Dennis Masaya
- Ricardo Cutz
- Antonio Hernández



BLOCKCHAIN

Objetivos

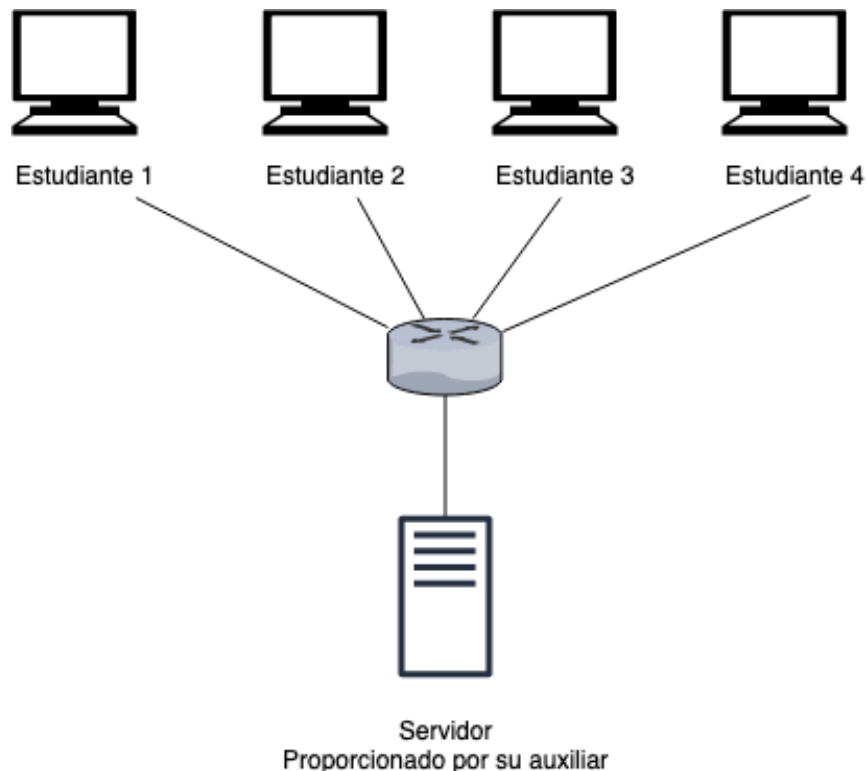
1. Familiarizarse con el lenguaje de programación **Python**.
2. Familiarizarse con la transmisión de información por medio de **Sockets**.
3. Familiarizarse con la traducción de información a formato **JSON**.
4. Comprender y desarrollar distintas estructuras de datos lineales como lo son listas enlazadas dobles.
5. Comprender y desarrollar distintas estructuras de datos no lineales como lo son arboles de autobalanceo (AVL).
6. Definir algoritmos de búsqueda y recorrido.
7. Familiarizarse con la herramienta **Graphviz** para la generación de reportes de estructuras gráficos.
8. Familiarizarse con el manejo y escritura de archivos .csv.

Definición del problema

Debido a la reciente problemática de cyber-ataques en los servidores de la facultad de Ingeniería de la Universidad de San Carlos de Guatemala, se ha discutido distintas soluciones que alivien dicho problema, y se ha llegado al consenso que debido al diseño de la red actual seria prudente implementar un sistema de tipo blockchain para llevar el control de las personas asignadas a cada curso, de esta manera aliviando los problemas de seguridad con los principios que blockchain nos provee.

Descripción

Según los requerimientos antes mencionados se deberá de desarrollar un programa de tipo Cliente el cual será un nodo de la red de computadoras que contendrán el Blockchain. Según el diagrama que se presenta a continuación se muestra el modo de funcionamiento de la aplicación, El Estudiante debe de programar su programa “Cliente”, y varios estudiantes se conectaran mediante un dispositivo de red al programa “Servidor”, proporcionado por su auxiliar, esto permitirá que cualquier estudiante pueda agregar un bloque al blockchain y este se debe de redistribuir a los demás estudiantes conectados.

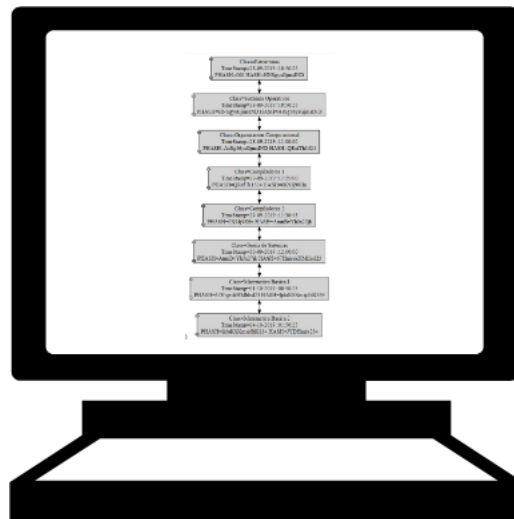


Pasos para añadir un bloque al blockchain

1. Un estudiante debe de crear un bloque por medio de un archivo .csv proporcionado por su auxiliar.
2. El programa “Cliente” del estudiante debe de crear el bloque, y enviarlo al servidor.
3. El Servidor debe de analizar el bloque y reenviarlo a todos los demás clientes conectados.
4. Los clientes deben de responder con la cadena “true”, si el bloque es correcto.
5. Un bloque es correcto si el método SHA-256(); que se ejecuta y el atributo HASH son iguales.
6. En este punto todos los clientes deben de tener el bloque creado sin embargo no debe de añadirse aun al blockchain.
7. Finalmente cuando el servidor recibe el consenso de todos los clientes conectados responde con una cadena “true”, y finalmente todos los clientes conectados pueden añadir el bloque nuevo.

Blockchain

El blockchain será una lista enlazada doble de nodos, este blockchain lo deben de tener todas las computadoras conectadas a la red de nodos.



Estudiante 1

Block

Un block (bloque), se definirá de la siguiente manera:

1. INDEX (INDICE)

Es un numero que representa el numero de bloque, el bloque génesis tendra como valor de index 0, los bloques creados posteriormente deberán de tener valores 1,2,3... etc.

2. TIMESTAMP (FECHA Y HORA DE CREACION)

Es la fecha y hora exacta en que se creo el bloque, la misma deberá de ir en el siguiente formato: (DD-MM-YY--:HH:MM:SS).

3. CLASS (CLASE)

El nombre de la clase que representa el bloque.

4. DATA (INFORMACIÓN)

En este caso será un árbol AVL el cual deberá de ser implementado por el estudiante mediante su conocimiento en estructuras de datos, para guardar dicho árbol en formato de texto se debe de recorrer el árbol de tal forma que se forme una cadena en formato json(se debe de implementar el recorrido en preorden) y de esta manera enviarlo por medio de este atributo. El ejemplo de cómo se debería quedar la cadena generada del árbol se detalla más adelante en el enunciado.

5. PREVIOUSHASH (HASH DEL BLOQUE PREVIO)

Es el hash del bloque previo, este sirve para validar que la cadena Blockchain no esté corrupta, en el caso del bloque génesis, el hash anterior debera de ser '0000'.

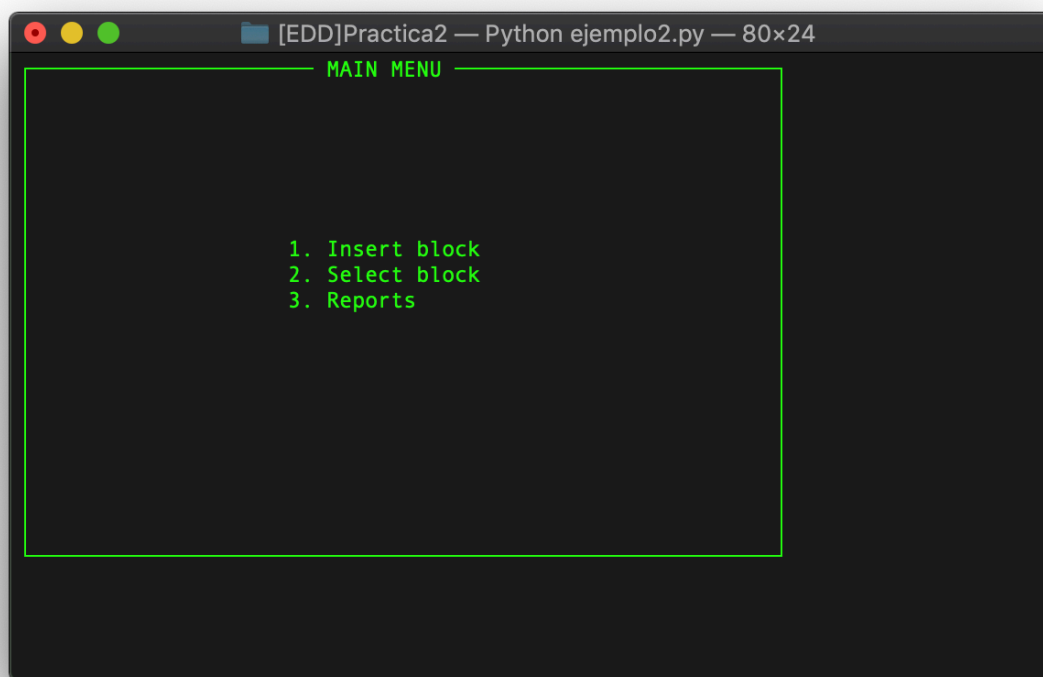
6. HASH (HASH DEL BLOQUE ACTUAL)

Es el hash que protege que la información del bloque actual no haya sido corrompida, el hash deberá de ser una función **SHA-256(Index+Timestamp+Class+Data+PreviousHash)**.

Menú Principal

Al comenzar a correr la aplicación se debe de tener un menú principal con las siguientes opciones:

1. Insert block (Insertar Bloque)
2. Select block (Seleccionar Bloque)
3. Reports (Reportes)



1. INSERT BLOCK (*INSERTAR BLOQUE*)

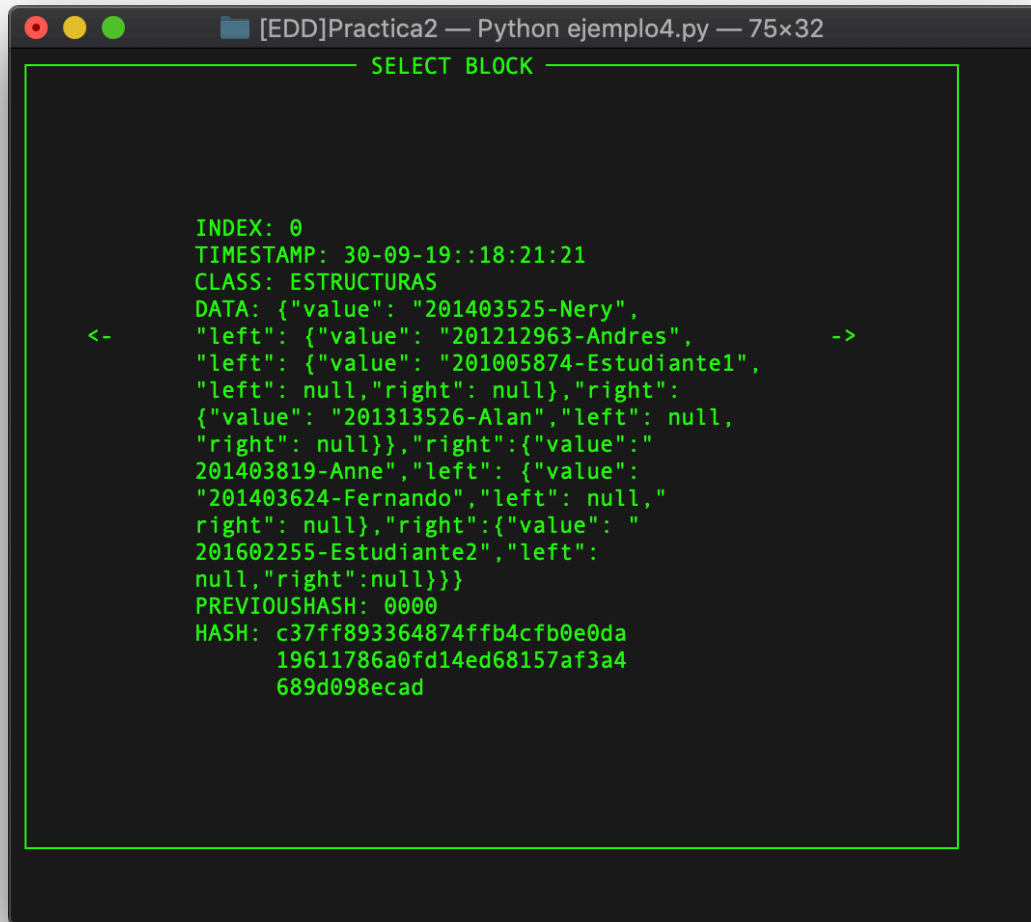
Para insertar un bloque el programa debe de solicitar el nombre del archivo correspondiente, este será un archivo de tipo .csv el cual se encontrara en una carpeta con el nombre "bloques", en la ruta principal del proyecto, el archivo .csv tendra el siguiente formato.

class	Estructuras de Datos	
	<pre> {"value": "201403525-Nery","left": {"value": "201212963- Andres","left": {"value": "201005874-Estudiante1","left": null,"right": null},"right": { "value": "201313526-Alan", "left": null, "right": null } } , "right": { "value": "201403819- Anne", "left": { "value": "201403624-Fernando", "left": null, "right": null } , "right": { "value": "201602255- Estudiante2", "left": null, "right": null } } } </pre>	
data		

- class representa el atributo “Class”, del bloque.
- data representa el árbol avl de estudiantes que será guardado en formato json en el atributo “Data” del bloque.
- Todos los demás atributos del bloque se deberán de generar en el momento de ingresar un archivo (Index es autoincremental, Timestamp es la fecha y hora actual, El previousHash se obtiene del bloque anterior, y el Hash se obtiene de todos los atributos del bloque actual)

2. SELECT BLOCK (SELECCIONAR BLOQUE)

Se debera de implementar una opcion de tipo “Carrousel”, debido a que el blockchain es una lista enlazada doble, se podra visualizar uno por uno cada uno de los bloques que la conforman, una vez se seleccione un bloque se podran generar reportes en base al bloque seleccionado. (NOTA: En el cado del atributo “DATA” únicamente se deben de mostrar los primeros 50 caracteres para evitar que un arbol de tamaño grande deforme el texto o el programa.)



The image shows a terminal window titled "[EDD]Practica2 — Python ejemplo4.py — 75x32". The terminal output is a JSON object representing a block, enclosed in a green rectangular border. The JSON object has the following fields: INDEX (0), TIMESTAMP (30-09-19::18:21:21), CLASS (ESTRUCTURAS), DATA (a nested JSON object), PREVIOUSHASH (0000), and HASH (a 32-character hexadecimal string). The DATA field contains a complex nested structure with 'value', 'left', and 'right' keys, representing a tree or graph structure. The terminal text is as follows:

```
SELECT BLOCK

INDEX: 0
TIMESTAMP: 30-09-19::18:21:21
CLASS: ESTRUCTURAS
DATA: {"value": "201403525-Nery",
      "left": {"value": "201212963-Andres",
               "left": {"value": "201005874-Estudiante1",
                        "left": null, "right": null}, "right": {"value": "201313526-Alan", "left": null, "right": null}}, "right": {"value": "201403819-Anne", "left": {"value": "201403624-Fernando", "left": null, "right": null}, "right": {"value": "201602255-Estudiante2", "left": null, "right": null}}}
PREVIOUSHASH: 0000
HASH: c37ff893364874ffb4cfb0e0da
      19611786a0fd14ed68157af3a4
      689d098ecad
```

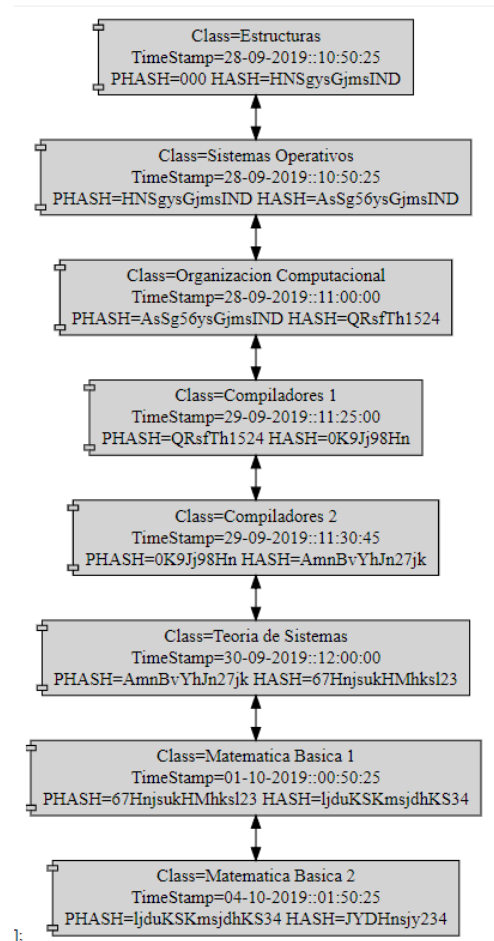
3. REPORTS (*REPORTES*)

Se debe de implementar una sección de reportes, esta sección contendrá un submenú el cual permitirá generar reportes de las estructuras utilizadas en el proyecto, **TODOS** los reportes deberán de ser implementados por medio de la herramienta **GRAPHVIZ**. Para que cada una de las estructuras tenga validez es necesario tener el reporte correspondiente.

BLOCKCHAIN REPORT

Al crear el blockchain el siguiente reporte debe desplegar cada uno de los bloques que conforman la cadena. Este reporte debe de irse actualizando al momento de que cada uno de los bloques se va creado. El reporte debe de mostrar de forma ordenada los nodos existentes, mostrando

atributos: **Class**, **TimeStamp**, **PSHASH** y **HASH**. A continuación se le presenta una sugerencia del reporte de los bloques creados:



TREE REPORTS

Previamente seleccionado cualquiera de los bloques almacenados en la estructura de blockchain.

Se deberá desplegar un menú el cual permitirá

- Visualizar árbol.
- Mostrar recorridos.

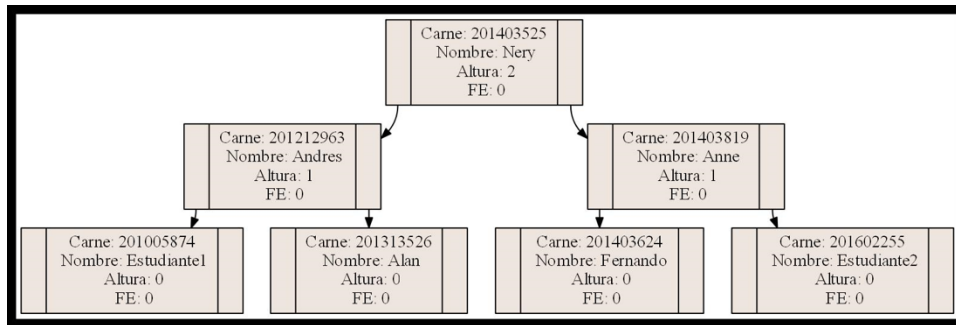
Visualizar Árbol: Se generara una imagen que tendrá la representación gráfica de la estructura del árbol seleccionado, cada nodo grafico del árbol mostrara SOLAMENTE la siguiente información:

- Clave/carnet
- Nombre
- Factor de equilibrio

- Nivel

Estas representaciones se mostraran abriendo la imagen del árbol en la computadora, automáticamente.

Ejemplo.



Mostrar Recorridos:

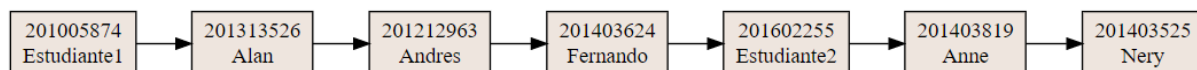
Se debe implementar la representación de los recorridos los cuales son:

- Preorden
- Posorden
- Inorden

Estos recorridos se deberán de desplegar por medio de una imagen de graphviz en forma de lista enlazada simple e imprimir el mismo en consola.

Ejemplo.

Recorrido en Posorden



Impresión en Consola

```

Consola de salida
INICIO-> 201005874-Estudiante1 -> 201313526->Alan-> 201212963-Andres -> 201403624-Fernando -> 201602255-Estudiante2 -> 201403819-Anne -> 201403525-Nery ->FIN
  
```

Es necesario resaltar que para optar a la calificación de cada una de las estructuras el reporte es necesario.

EJEMPLO de archivo json

Utilizando la imagen de ejemplo del reporte del árbol la cadena generada debería de quedar de la siguiente manera.

```
{
  "value": "201403525-Nery",
  "left": {
    "value": "201212963-Andres",
    "left": {
      "value": "201005874-Estudiente1",
      "left": null,
      "right": null
    },
    "right": {
      "value": "201313526-Alan",
      "left": null,
      "right": null
    }
  },
  "right": {
    "value": "201403819-Anne",
    "left": {
      "value": "201403624-Fernando",
      "left": null,
      "right": null
    },
    "right": {
      "value": "201602255-Estudiente2",
      "left": null,
      "right": null
    }
  }
}
```

Detallando que no existen más nodos en un subarbol con la palabra null.

Se debe usar la librería de python “json” para parsear este tipo de cadena y así generar partiendo de esta misma el árbol AVL cuando se reciba por medio de un bloque entrante.

Restricciones

Las estructuras deben de ser desarrolladas por los estudiantes sin el uso de ninguna librería o estructura predefinida en el lenguaje a utilizar Los reportes son esenciales para verificar si se trabajaron correctamente las estructuras solicitadas, Si no se tiene el reporte de alguna estructura se anularán los puntos que tengan relación tanto al reporte como a la estructura en cuestión.

Penalizaciones

1. Falta de seguimiento de desarrollo continuo por medio Github tendrá una penalización del 10% (mínimo 2 commits por semana).
2. Falta de seguimiento de instrucciones conforme al método de entrega (nombre del repositorio) tendrá una penalización del 5%.
3. Falta de puntualidad conforme a la entrega tendrá una penalización de la siguiente manera:
 - a. 0-6 horas – 20%.
 - b. 6-12 horas – 40%
 - c. 12-18 horas 60%
 - d. 18-24 horas – 80%.
 - e. Pasados 24 horas tendrá una nota de 0 y no se calificará.

Observaciones

1. Lenguaje a utilizar: **Python**
2. Herramienta para desarrollo de reportes gráficos: **Graphviz**.
3. La aplicación debe de ser capaz de generar y abrir con un visor de imágenes predeterminado las imágenes generadas con Graphviz.
4. La aplicación debe de ser capaz de conectarse al servidor proporcionado por los auxiliares para tener derecho a calificación.
5. La entrega se realizará por medio de: **Github**, cada estudiante deberá crear un repositorio con el nombre: **EDD_2S2019_P2_#carnet**, y agregar a su auxiliar correspondiente como colaborador del mismo, para poder analizar su progreso y finalmente a partir del mismo repositorio realizar la calificación correspondiente.
Dennis Masaya: <https://github.com/Dennis201503413>
Ricardo Cutz: <https://github.com/ricardcutzh>
Antonio Hernández: <https://github.com/fernando29hernandez>
6. Además de tener a su auxiliar como colaborador del repositorio para tener un control y orden de las personas que entreguen deberán de colocar el Link de su repositorio en la Tarea que cada auxiliar asignara en su classroom correspondiente.
7. Fecha y hora de entrega: **Jueves 17 de octubre, a las 23:59 horas**.
8. **Copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.**