UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructura de Datos B

Ing. Carlos Alonzo

Aux. Josué David Itzep Salvador

PRACTICA 2

FUNDAMENTOS DE BLOCKCHAIN

DESCRIPCION GENERAL

En la actualidad las tecnologías distribuidas o DLT (Distributed Ledger Technologies), han irrumpido en distintos campos, y se presentan como una de las tecnologías con mayor fuerza en los últimos años, ya que brinda seguridad, elimina intermediarios, y brinda fiabilidad. Se solicita a los estudiantes de Ciencias y Sistemas de la FI-USAC, que desarrollen una aplicación grafica en los lenguajes Java y C/C++, haciendo uso de estructuras de datos, en la cual se registren y se carguen distintas transacciones mediante la carga de archivos a memoria dinámica y almacenaje de los mismos a memoria secundaria, para lo cual se pide que utilicen sockets (TCP/UDP) para la comunicación entre dichas aplicaciones. Así mismo se aplicaran distintos conceptos de *Blockchain* o Cadena de Bloques, como lo son las anteriormente mencionadas DLT, bloques, log de transacciones y referencias entre los bloques de la red para poder crear el antes mencionado *Blockchain.*

SERVIDOR

Se desarrollara en Qt Creator, con las estructuras en lenguaje C puro, y tendrá las siguientes opciones:

- Archivo (Carga de Archivos)
- Recorridos (Pre-orden, Post-Orden, In-Orden, Amplitud)
- Bloque (Crear Bloque)
- Ayuda (Manual técnico, Manual de Usuario, Acerca De)

CARGA DE ARCHIVOS

Se podrán cargar múltiples archivos de texto con extensión '.edd-b' los cuales poseerán información acerca de los estudiantes del curso Estructura de Datos, sección B, que poseerán los siguientes campos:

- Carnet (sin espacios ni guiones)
- Nombre
- Nota (decimal)

Dicho archivo poseerá la misma estructura que un archivo '.csv', ósea una cabecera y delimitado por comas, la estructura se detalla a continuación:

#carnet, nombre apellido, nota

La información será cargada a memoria en un **árbol AVL,** y se utilizara el número de carnet como valor de la llave para cada nodo.

ARBOLES EN MEMORIA

Tanto los árboles que han sido cargados por medio de un archivo como los arboles resultado, serán almacenados en una **lista doblemente enlazada** cada nodo de esta lista poseerá:

- ID (autogenerado e incremental)
- Nombre del árbol (si fue cargado, el nombre será el mismo del archivo, sin la extensión)
- Referencia al árbol
- Fecha y hora de carga o creación

Se deberá mostrar la lista de árboles en memoria de manera gráfica, así como el nodo que se tiene seleccionado, para lo cual en la gráfica deberá mostrar algo distinto en el nodo que se tiene seleccionado actualmente.

SELECCIÓN DE ARBOL

Se debe de poder seleccionar cualquier árbol en memoria, y se mostrara dicho árbol de manera gráfica, cada nodo grafico del árbol mostrara SOLAMENTE la siguiente información:

- Clave/carnet
- Factor de equilibrio
- Nivel

Estas representaciones se mostraran dentro de la GUI del programa, en la lista de árboles se debe de mostrar el nodo seleccionado en ese momento, así mismo se deberá mostrar la información del árbol mencionada a continuación

INFORMACION DEL ARBOL

Se deberán de implementar distintos algoritmos, con los cuales se pueda calcular la información de los distintos árboles que se encuentren en memoria, los datos a calcular son:

1. Nombre del árbol
2. Altura del árbol
3. Nivel del árbol
4. Cantidad total de ramas
5. Cantidad de nodos hoja
6. Cantidad de nodos padre

Esta información debe de ser mostrada dentro de la grafica

RECORRIDOS DEL ARBOL

Se debe de realizar los siguientes recorridos para los distintos árboles:

- Pre-orden
- Post-orden
- In-Orden
- Amplitud/Anchura

Estos recorridos serán mostrados en la consola del IDE, y mostraran solamente los números de carnet, separados por coma, se deberá mostrar únicamente el recorrido solicitado

OPERACIONES EN ARBOL

Se podrá obtener un nodo mediante el ingreso del número de carnet, el cual devolverá toda la información del nodo y llenara los campos de la interfaz, y permitirá realizar las siguientes operaciones sobre el árbol:

- Editar Nodo
- Eliminar Nodo
- Obtener Subárbol (este se mostrara por aparte, puede ser en otra ventana, o por medio de una llamada al S.O., se deberá de poder ver el árbol completo y el subárbol simultáneamente)

AYUDA

Este botón desplegara los manuales técnicos, y de usuario, así como una ventana con la información del estudiante, DPI, nombre, carnet, nombre y sección del curso

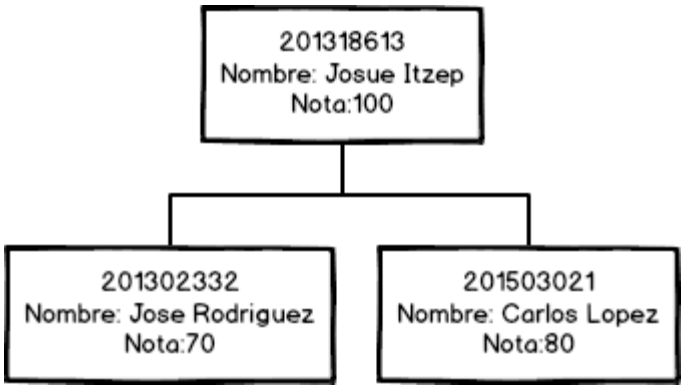
BLOQUE

CREACION DE BLOQUE

Esta opción permitirá seleccionar un árbol de la lista, mediante el ingreso del nombre registrado, el cual servirá para realizar la creación de la llave única dentro del bloque, además será identificada mediante el **número 1 para tipo de operación**

LLAVE UNICA

Se realizara un recorrido In-orden, del árbol que se haya seleccionado, para lo cual se concatenara cada valor que posea un nodo de la siguiente manera:



Se iniciara la cadena con el símbolo "<", y se finalizara con ">", la información que posee un nodo será separada por medio de ",", y la información de distintos nodos será separada mediante "&&". Ej:

<201318613,Josue Itzep,100&&201302332,Jose Rodriguez,70&&201503021,Carlos Lopez,80>

Esta llave única será almacenada en memoria por parte del cliente, por lo que será enviada en la cadena descrita a continuación

INTERACCION CLIENTE

En la opción "Bloque", deberá de existir una opción, en la que se envié al cliente mediante sockets la siguiente información:

- Tipo Operación
- Nombre del árbol
- Llave única
- Recorrido Pre-orden
- Recorrido Post-orden
- Recorrido In-Orden

Con la información descrita anteriormente se enviara una cadena en formato JSON al cliente que tendrá la siguiente estructura:

```
{
  "tipo-operacion":1,
  "nombre":"arbol-1",
  "llave-unica":"<201318613,Josue Itzep,100&&201302332,Jose Rodriguez
    ,70&&201503021,Carlos Lopez,80>",
  "pre-orden":[
    {"carnet":201318613,"Nombre":"Josue Itzep","Nota":100.00},
    {"carnet":201302332,"Nombre":"Jose Rodriguez","Nota":70.00},
    {"carnet":201318613,"Nombre":"Carlos Lopez","Nota":80.00}
  ],
  "post-orden":[
    {"carnet":201302332,"Nombre":"Jose Rodriguez","Nota":70.00},
    {"carnet":201318613,"Nombre":"Carlos Lopez","Nota":80.00},
    {"carnet":201318613,"Nombre":"Josue Itzep","Nota":100.00}
  ],
  "in-orden":[
    {"carnet":201302332,"Nombre":"Jose Rodriguez","Nota":70.00},
    {"carnet":201318613,"Nombre":"Josue Itzep","Nota":100.00},
    {"carnet":201318613,"Nombre":"Carlos Lopez","Nota":80.00}
  ]
}
```


Figura 2. Respuesta Servidor

ENVIO INFORMACION DE BLOQUE

Se deberá de enviar la información al bloque siguiente (Cliente), luego de que este la reciba devolverá un mensaje, por lo que al recibir la confirmación de parte del cliente, deberá de mostrar una notificación.

SOLICITUD DE BLOQUE

El cliente podrá solicitar en cualquier momento la información del árbol de transacciones con el cual se generó el bloque, para lo cual se mostrara una alerta de que se solicita el árbol, dando la opción a aceptar o a rechazar dicha solicitud. Si se acepta se buscara dentro de la lista de aboles el nombre del árbol, y se generara la llave única, y se enviara al cliente.

Esta operación de solicitud vendrá identificada mediante el **tipo de operación 2,** y se responderá de la misma manera que la figura 2.

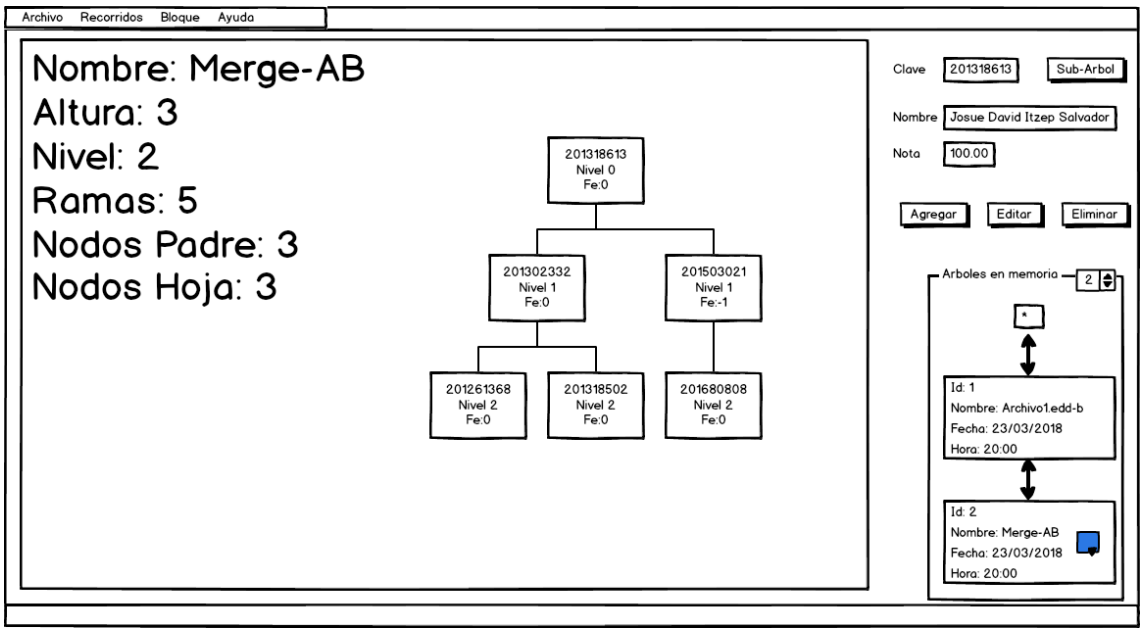


Figura 3. Servidor

CLIENTE

Se desarrollara en Java (Netbeans) con interfaz gráfica, y las estructuras con clases

INTERACCION SERVIDOR

El cliente estará en espera de información enviada por el servidor, dicha información será recibida en formato JSON. En esta se recibirá también información de las transacciones realizadas en forma de recorridos, dichos recorridos servirán para generar el árbol en memoria en esta aplicación. Al recibir la información se deberá de solicitar la información (de ser necesaria), que se describe en la ***Figura 2***

La información será cargada a memoria en un **árbol AVL,** y se utilizara el número de carnet como valor de la llave para cada nodo.

ARBOLES EN MEMORIA

Los arboles serán almacenados en una **lista doblemente enlazada** cada nodo de esta lista poseerá:

- ID (autogenerado e incremental)
- Nombre del árbol (si fue cargado, el nombre será el mismo del archivo)
- Llave unica
- Referencia al árbol
- Fecha y hora de carga o creación

Se deberá mostrar la lista de árboles en memoria de manera gráfica, así como el nodo que se tiene seleccionado, que a su vez será el **bloque previo** para lo cual en la gráfica deberá mostrar algo distinto en el nodo que se tiene seleccionado actualmente.

SOLICITUD DE BLOQUE

Se deberá de poder solicitar en cualquier momento la **llave única** del **bloque previo** a través del nombre del árbol, por lo que se enviara una solicitud al servidor, si este la acepta, buscara en memoria el nombre del árbol solicitado y devolverá la llave única del árbol solicitado. La estructura será enviada en formato JSON de la siguiente manera, y será i**dentificada por medio del tipo de operación 2** de la siguiente manera:

```
{
  "tipo-operacion":2,
  "nombre":"arbol-1",
  "llave-unica":"",
  "pre-orden":[
  ],
  "post-orden":[
  ],
  "in-orden":[
  ]
}
```

Esto será realizado con el fin de detectar cambios en las transacciones, por lo que si el servidor acepta la solicitud devolverá la información con el formato de la figura 2, y deberá de

comparar las cadenas, y de detectarse el cambio deberá de mostrarse una alerta, y listarse los cambios detectados

SELECCIÓN DE ARBOL

Se debe de poder seleccionar cualquier árbol en memoria, y se mostrara dicho árbol de manera gráfica, cada nodo grafico del árbol mostrara SOLAMENTE la siguiente información:

- Clave/carnet

En la lista de árboles se debe de mostrar el nodo seleccionado en ese momento.

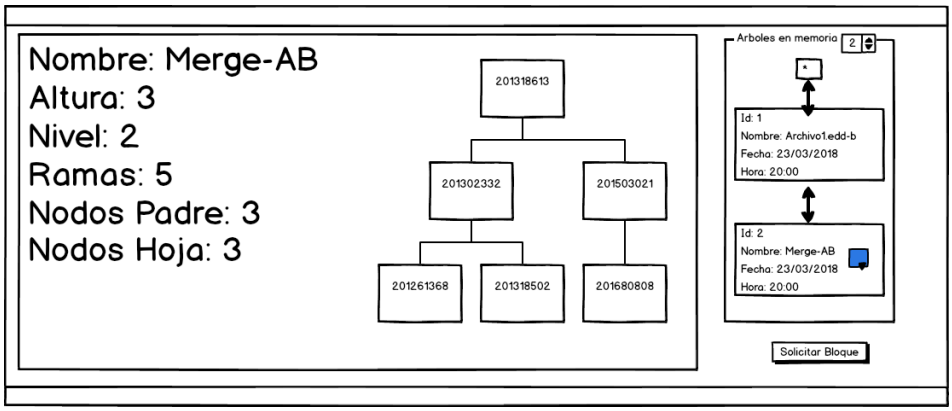


Figura 5. Cliente

Observaciones

- Sistema Operativo: Linux
- IDE's permitidos: QtCreator, Netbeans
- Herramienta de Comunicación: Sockets TCP/UDP
- Las estructuras deben de ser realizadas por el estudiante, deben usar struct y punteros desarrollados en C puro para el servidor (los struct NO deben de ir dentro de clases) y clases para el cliente Java
- Herramienta de Graficas: Graphviz
- La calificación se realizara en grupos de 4 personas para lo cual deberán de ponerse de acuerdo con las personas de su grupo para realizar la conectividad
- Se recomienda el uso de variables globales para poder cambiar el valor de los puertos y de las ip's, necesarias para la conectividad con las personas del grupo el día de la calificación
- Se recomienda la validación de las cadenas JSON mediante el uso del sitio: https://jsoneditoronline.org/
- O cualquier otra similar
- Habrá un punteo para una prueba local (en el mismo computador), y otro asociado al funcionamiento del grupo completo

ENTREGABLES

- Código Fuente
- Ejecutable

- Manual Técnico
- Manual Usuario

REQUERIMIENTOS MINIMOS

- ✓ Servidor-Carga de Múltiples Archivos
- ✓ Servidor-Recorridos en: Pre-Orden, Post-Orden, In-Orden
- ✓ Servidor-Operaciones en Árbol (Editar y Eliminar)
- ✓ Servidor-Bloque (envió y solicitud)
- ✓ Servidor- Lista de Árboles, Selección, Grafica de Arboles e Integración GUI
- ✓ Cliente-Lista de Árboles y Grafica de árboles e Integración GUI
- ✓ Cliente-Recepción y envió de solicitudes
- ✓ Comprobación en mismo ordenador

FECHA Y MODO DE ENTREGA

- Viernes 13 de Abril de 2018 antes de las 23:59, entregas tarde tendrán automáticamente una nota igual a 0
- Se deben de enviar los entregables vía Dropbox, en un comprimido con el nombre: [EDD]Practica2_#carnet (e.g. [EDD]Pratica2_201318613)
- Dropbox: https://www.dropbox.com/request/xFz164DTcnFuswZA5YX1
- Requerimientos Mínimos

Penalizaciones

- Si ocurre un NULL Pointer Exception, la nota obtenida será 0 automáticamente
- Si se cierra la aplicación repentinamente, la nota obtenida será 0 automáticamente
- Las representaciones graficas de las estructuras son esenciales para verificar si se realizó de manera correcta la estructura, por lo que si no se representan de esta manera NO se calificara
- Si NO se realizan las estructuras de la manera en que se indica en la sección Observaciones la nota obtenida será 0 automáticamente.
- Si se detecta el uso de librerías como QStack, QQueue, QList, etc, para C/C++ o el uso de LinkedList, Stack, Queue, o similares para Java, la nota obtenida será 0 automáticamente
- Las representaciones graficas deberán de mostrarse integradas en las GUI, y realizadas utilizando Graphviz, de no ser así NO se calificara

***Copias totales o parciales, tendrán nota de 0 puntos, serán reportadas al catedrático y a la Escuela de Ciencias y Sistemas FI-USAC**