Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Estructuras de Datos
Proyecto #2
Ingenieros

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

Auxiliares

- Kevin Mejía
- Ricardo Cutz
- Dennis Masaya

USAC Library

Objetivos

1. Familiarizarse con el lenguaje de programación **Java**
2. Familiarizarse con la lectura y escritura de archivos de **Json**.
3. Comprender y desarrollar distintas estructuras de datos no lineales como lo son tablas hash, árboles B, árboles AVL y los conceptos relacionados a blockchain
4. Definir algoritmos de búsqueda y recorrido y eliminación.
5. Familiarizarse con la herramienta **Graphviz** para la generación de reportes de estructuras gráficos.

Definición del problema

Para los estudiantes de la facultad de ingeniería es importante contar con material de lectura para el desarrollo de las carreras. Con la finalidad de ayudar a los estudiantes con esta necesidad se ha pensado en la posibilidad de crear una biblioteca virtual que permita el compartir material entre los estudiantes de la facultad.

Descripción

Según la idea de poder crear una biblioteca virtual para los estudiantes, se necesita desarrollar una aplicación de escritorio que pueda ejecutarse independientemente del sistema operativo de la pc, por lo que se propone crear una aplicación de escritorio en el lenguaje de programación JAVA para el proyecto.

Implementación de la Solución

La aplicación contará con los siguientes módulos:

- Carga Masiva de datos: permitirá ingresar los datos de la biblioteca de libros que posee cada estudiante y otros datos necesarios para compartir.
- Biblioteca: Este módulo permite la interacción de los estudiantes con las distintas funcionalidades de la biblioteca virtual.
- Reportes: Este módulo permite visualizar los reportes de las estructuras utilizadas para la biblioteca

Carga Masiva de Datos

La carga masiva permitirá ingresar los datos iniciales a la aplicación, esto incluyendo los libros y los usuarios que existen en la aplicación.

UI de la app

El diseño de la interfaz gráfica de la aplicación queda a discreción del estudiante con la única observación que debe de poder realizarse lo siguiente:

- Visualizar reportes de las estructuras
- Navegación en catálogo de libros por categorías
- Búsqueda de los libros existentes en la biblioteca
- Menú de configuraciones (IP y puerto del servidor socket)
- Operaciones sobre Usuarios
- Inicio de Sesión
- Operaciones sobre los Libros

Biblioteca

La biblioteca virtual tiene como objetivo principal unificar todas las bibliotecas de la facultad. En donde cada estudiante contará con su propia biblioteca y al momento de ser cargada al sistema será una parte de la biblioteca virtual, permitiendo con esto el acceso a cada uno de los estudiantes hacia la misma. La biblioteca será multiusuario. La biblioteca permitirá al usuario navegar por los libros que se tienen registrados, permitiendo realizar distintas acciones, tales como cargar, dar de baja, entre otras acciones.

Las bibliotecas utilizan un formato JSON de forma estandarizada para la comunicación entre las mismas. La biblioteca virtual contará con un servidor al cual los estudiantes podrán acceder para realizar realizando peticiones y obtener de esta forma la información del resto de bibliotecas.

Cada estudiante será encargado de desarrollar su interfaz que permita a los usuarios navegar por las distintas bibliotecas y realizar búsquedas para encontrar un libro, mostrando únicamente la información relevante y siendo más detallado si se desea ver su información.

Ya que cada estudiante tiene su propia biblioteca, este cuenta con permisos únicamente de los libros registrados en esa biblioteca, de lo contrario tendrá que solicitar permiso al respectivo propietario para las acciones que desee-

Cada vez que se realiza una acción en las bibliotecas, esta será reflejada en la biblioteca virtual, con el cuidado de mantener la información íntegra y actualizada.

Usuarios de bibliotecas

Los usuarios de la biblioteca serán almacenados utilizando una tabla de dispersión. Los usuarios deben ser cargados a la aplicación utilizando un archivo JSON de entrada. Este archivo JSON contendrá la definición de todos los usuarios que deben existir en la aplicación que harán uso de la biblioteca. Ya que los usuarios serán estudiantes de la Universidad de San Carlos, los datos que se almacenarán de un usuario serán los siguientes:

- Número de Carnet (entero)
- Nombre (cadena)
- Apellido (cadena)
- Carrera (cadena)
- password (cadena)

NOTA: Para almacenar los password se deberá utilizar el hash MD5.

Para el almacenamiento de los usuarios en la tabla de dispersión se debe de tomar en cuenta lo siguiente:

- **Función de Dispersión:** La función de dispersión a utilizar será la siguiente **F(carnet)= carnet mod size.** La tabla de dispersión tendrán un tamaño máximo de 45 espacios.
- **Resolución de Colisiones**: se utilizará el método de dispersión abierta, por lo que si ocurre una colisión en una casilla, este se debe de colocar en la lista de valores que almacena la casilla (lista simplemente enlazada)

Para cada uno de los usuarios se puede realizar lo siguiente:

- Editar información del usuario (Editar su propia información)
- Agregar usuario nuevo (manualmente, registro de un usuario nuevo)
- Eliminar usuario (un usuario puede eliminar su cuenta únicamente si se encuentra logueado).

La aplicación debe de contar con la funcionalidad de inicio de sesión, en caso que el usuario desee entrar con credenciales que no son correctas la aplicación debe de mostrar el mensaje de credenciales incorrectas. El ejemplo del archivo de entrada pueden encontrarlo en el siguiente enlace: usuarios.json

Libros

Los libros tanto digitales como físicos son esenciales en una bibliotecas, siendo esta una biblioteca virtual la búsqueda y tiempo de respuesta es esencial, por ello serán almacenadas las categorías de los libros en un **Árbol AVL**, donde cada nodo poseerá un **Árbol B de orden 5,** que contendrá los libros correspondientes a dicha categoría.

La forma de identificar un libro es esencial, se utilizará el código ISBN de cada libro como criterio de inserción en el Árbol B, al ser este un código universal único se deberá verificar que no exista otro libro con el mismo código a nivel de la biblioteca virtual, es decir no únicamente en la biblioteca local (de cada estudiante)..

La información que se necesita guardar de cada libro es la siguiente:

- ISBN
- Título
- Autor
- Editorial
- Año
- Edición
- Categoría
- Idioma
- Carnet del usuario que agregó el libro a la biblioteca.

Los libros serán cargados a cada biblioteca mediante un archivo JSON de entrada. Este archivo contendrá la definición de los libros que el usuario posee en su biblioteca. Estos libros al ser parte de la biblioteca de cada estudiante, deberán ser registrados a nivel de la biblioteca virtual, por lo que se deberá realizar los registros necesarios en el Árbol AVL, correspondiente a las categorías para luego poder insertar en el árbol B, los libros de su biblioteca tomando en cuenta que no pueden existir libros repetidos a nivel de la biblioteca virtual.

Acciones sobre los libros

Cargar archivo JSON

Desde la sección de libros, se permitirá al estudiante realizar la carga masiva de libros mediante un archivo JSON, se puede encontrar un ejemplo de este archivo: libros.json.

Crear libro

A veces es más fácil realizar la carga de un libro de forma individual y no generar un archivo JSON para realizar esta tarea, por lo tanto la aplicación deberá contar con una sección que le permitirá crear libros con los datos necesarios.

Dar de baja

El usuario propietario de la biblioteca tiene los permisos de dar de baja a aquellos libros que son de su propiedad, es decir los que se encuentran dentro de su biblioteca. Para ello se deberá contar con una sección que permita eliminar los libros de la siguiente forma:

- Ingresando el código ISBN, al ingresar el código se pedirá la confirmación del usuario para proceder a realizar la eliminación.
- Muchas veces no se conoce el código ISBN exacto de un libro, por lo cual es necesario realizar la eliminación por medio de una búsqueda por nombre, es decir: se deberá ingresar una parte del nombre o el nombre completo del libro y serán mostrados aquellos libros que contengan esa parte del nombre o coincidan totalmente, permitiendo elegir de esta forma al usuario qué libro desea dar de baja.

Al eliminar un libro se da de baja tanto en la biblioteca del estudiante propietario como de la biblioteca virtual, por lo que se le solicita al estudiante realizar la eliminación de forma correcta y solicitar una razón del porque se dio de baja el libro.

Visualizar la información del libro

Un libro puede tener más de una edición, ser de diferentes editoriales y un estudiante puede desear uno en específico, entonces se deberá poder ver la información de forma detallada.

Acciones sobre Categorías

Crear una Categoría

Esta operación permite la creación de nuevas categorías en el biblioteca, esto implica con cada nueva categoría creada se debe agregar un nodo al **Árbol AVL.** Es necesario recordar que cada uno de los nodos del árbol AVL tienen una referencia a un árbol B de libros.

Eliminar Categoría

Esta operación permite la eliminación de categorías existentes en la biblioteca, al momento de eliminar una categoría se debe de eliminar completamente los libros asociados a esa categoría. Para eliminar la categoría sólo debe de indicarse el nombre de la categoría a eliminar. Solo el usuario que creó la categoría tienen los permisos para poder eliminarla, es decir que un usuario no creador de la categoría no puede eliminarla.

Blockchain

La aplicación va a funcionar utilizando los conceptos de blockchain. Esto se realizará utilizando sockets para la comunicación de cada uno de los nodos de la red. Para la simulación de los nodos en la red, se debe de poder ejecutar varias instancias de la aplicación, cada una representando un nodo de la red. Todos los nodos deben de comunicarse entre sí.

Estructuras a utilizar en el Blockchain

- **Nodos:** Esta estructura será una lista simplemente enlazada que contiene la información de cada uno de los nodos de la red. Cada nodo llevará la información de IP.
- **Blockchain:** una lista doblemente enlazada de bloques que se crearán por cada nodo de la red

Nodos de la Red

Cada nodo de la red será representado por una instancia de la aplicación que se esté ejecutando en ese momento. Cada nodo podrá ejecutar operaciones sobre la biblioteca para actualizar la información de cada una de ellas.

Al iniciar cada instancia de la aplicación (nodo) este debe de agregarse a la red de nodos existentes. En caso sea el primer nodo simplemente no hace nada. En caso el nodo ya hubiera generado bloques, estos deben estar almacenados en una carpeta que genere la aplicación, donde los bloques estén almacenados en formato json, esto se hará con la finalidad de que al momento de que la aplicación vuelva a iniciarse pueda leer los bloques existentes que tenía en su cadena anteriormente para no perder la información de la biblioteca.

Prueba de Trabajo

Es el proceso por el cual se encuentra un hash que cumpla con la condición de tener un prefijo de cuatro ceros. Para ello se debe de iterar un entero denominado **NONCE** hasta encontrar un hash válido para el bloque.

Longest Chain Rule

La regla de la cadena más larga se aplicará al momento de que dos o más bloques sean creados al mismo tiempo. Esta regla establece que la red va a reconocer como válida la cadena con el mayor trabajo invertido, esta cadena es normalmente la cadena más larga.

Operaciones de Blockchain

- **Registrar Nodo:** Cuando un nodo nuevo se une a la red, este debe de enviar su IP para que sea agregado por los nodos que conforman la red. Esto se refiere a que cada nodo que va a registrar al nodo nuevo debe de agregar a este en su listado de nodos con la

información necesaria para comunicarse con el. Se debe indicar la IP de otro nodo para poder registrar el nodo actual a la red.

- **Eliminar Nodo:** Al momento de que un nodo ya no se encuentre en la red (se va a cerrar la aplicación) este envía la información necesaria al listado de nodos de la red para notificar su retiro. Los demás nodos de la red deben de eliminarlo de su lista de nodos. La cadena de bloques que tenía hasta el momento debe de almacenarse en formato json en la carpeta generada por la aplicación.
- **Sync:** Esta operación consiste en que un nodo de la red solicita a otro nodo el último bloque válido de su cadena para validarlo. Si en caso se encuentra que existen nodos predecesores que aún no están en su cadena, debe solicitarlos hasta tener la cadena completa. En caso el nodo tenga una cadena previa, debe de compararla con la cadena recibida de otros nodos para realizar la sincronización de los nodos en la cadena para que estos sean agregados a la actual.
- **Nuevo Bloque:** Al momento que un nodo genera un nuevo bloque este debe de ser enviado a cada nodo de la red (nodos registrados en su lista). Los demás nodos deben de recibir el bloque nuevo, validarlo (prueba de trabajo) y agregarlo a su cadena. En el caso que dos o más bloques sean creados al mismo tiempo, cada nodo de la red debe realizar la prueba de trabajo de los bloques y agregarlos a su cadena, se tomará la cadena correcta como la cadena del nodo más larga (longest chain) ya que ha tomado más esfuerzo de computación para agregar el bloque.

Bloque

El bloque se define de la siguiente manera:

- **INDEX:** representa el número de bloque el bloque génesis tendrá valor de index 0 , los bloques posteriores deberán tener valores 1, 2, 3, 4 **...** etc.
- **TIMESTAMP (fecha y hora de creación):** Es la fecha y hora exacta en la que se creó el bloque**.** Debe de tener el siguiente formato: (DD-MM-YY-::HH:MM:SS).
- **DATA:** Son todos los datos de las creación, ediciones y eliminaciones de usuarios, categorías o libros.
- **NONCE:** Será el número entero que se debe iterar de uno en uno hasta encontrar un hash válido, es decir un hash que inicie con cuatro ceros.
- **PREVIOUSHASH:** Es el hash del bloque previo, este es necesario para validar que la cadena de bloques no esté corrupta. En caso del bloque génesis, el hash anterior debe de ser 0000.
- **HASH (bloque actual):** El hash que protege que la data no se ha comprometido, el hash deberá generarse aplicando la función SHA256 a las propiedades: INDEX, TIMESTAMP, PREVIOUSHASH, DATA y NONCE **todas estas propiedades como cadenas concatenadas sin espacios en blanco ni saltos de línea**. Es decir SHA256(INDEX+TIMESTAMP+PREVIOUSHASH+DATA+NONCE). Para considerar el hash como válido este debe de tener un prefijo de cuatro ceros. Es decir que un hash valido sería el siguiente: 000002b12041cb5a7bac8ec90f86b654af6b1ac8bfc5ed08092e217235df0229

Definición de Bloques

Cada uno de los bloques tendrá la siguiente estructura:

```
{
    "INDEX": 0,
    "TIMESTAMP": "05-04-20::10:45:22",
    "NONCE": 2849
    "DATA": [
        // ARREGLO DE OPERACIONES
    ],
    "PERVIOUSHASH": "0000",
    "HASH": "00002e5d76809d3f4ee13af11d5ddfc2b06dfb69716893fb6d93399eb63ca28a2e2e"
}
```

La sección de **DATA** se colocarán los datos de las operaciones que se ejecutaron en el nodo que creó el bloque. El usuario de la aplicación decidirá en qué momento creará el bloque, la data que llevará el bloque será un listado de las operaciones realizadas hasta el momento de generar el bloque (crear usuario, crear libro, eliminar categoría, etc)

Creación de un usuario

La operación que representará la creación de un usuario, ya sea manualmente o por carga masiva. De ser por carga masiva se debe de crear una operación por cada usuario que viene en el archivo de carga. La definición será la siguiente:

```
"DATA": [
    {
        "CREAR_USUARIO": [
                {
                    "Carnet": 201503476,
                    "Nombre": "Ricardo",
                    "Apellido": "Cutz",
                    "Carrera": "Ingenieria en Sistemas",
                    "Password": "123456"
                }
        ]
    }
]
```

La operación se llamará **CREAR_USUARIO** que almacenará la definición de un usuario creado y se agregara a la estructura correspondiente.

Edición de un Usuario

La operación que representará la edición de un usuario será la siguiente:


```
"DATA": [
    {
        "EDITAR_USUARIO": [
                {
                    "Carnet": 201503476,
                    "Nombre": "Ricardo",
                    "Apellido": "Cutz",
                    "Carrera": "Ingenieria en Sistemas",
                    "Password": "123456"
                }
        ]
    }
]
```


La operación se llamará **EDITAR_USUARIO** que almacenará la definición de un solo usuario editado y este se actualizará a la estructura correspondiente.

Crear un Libro

La operación que representará la creación de un libro manualmente o por medio de carga masiva. De ser por carga masiva se debe de crear una operación por cada libro que viene en el archivo de carga. La definición será la siguiente:


```
"DATA": [
    {
        "CREAR_LIBRO": [
          {
            "ISBN": 10761,
            "Año": 1999,
            "Idioma": "Aleman",
            "Titulo": "Harry Potter and the chamber of secrets, v 2",
            "Editorial": "20th Century Fox",
            "Autor": "ESCOBAR WOLF, WIELAND JOSÉ",
            "Edicion": 1,
            "Categoria": "Sagas"
          }
        ]
    }
]
```


La operación se llamará **CREAR_LIBRO** que almacenará la definición de un solo usuario creado y se agregara a la estructura correspondiente.

Eliminar un Libro

La operación que representará la eliminación de un libro será la siguiente:

```
"DATA": [
    {
        "ELIMINAR_LIBRO": [
          {
            "ISBN": 10761,
            "Titulo": "Harry Potter and the chamber of secrets, v 2",
            "Categoria": "Sagas"
          }
        ]
    }
]
```

La operación se llamará **ELIMINAR_LIBRO** que almacenará los datos necesarios para eliminar el libro correspondiente y actualizar la estructura

.Crear una Categoría

La operación que representaŕa la creación de una categoría será la siguiente:

```
"DATA": [
    {
        "CREAR_CATEGORIA": [
          {
            "NOMBRE": "Novelas"
          }
        ]
    }
]
```

La operación se llamará **CREAR_CATEGORIA** que almacenará los datos necesarios para crear la categoría correspondiente y agregarla a la estructura.

Eliminar una Categoría

La operación que representaŕa la eliminación de una categoría será la siguiente:

La operación se llamará **ELIMINAR_CATEGORIA** que almacenará los datos necesarios para eliminar la categoría correspondiente y removerla de la estructura.

UN BLOQUE PUEDE CONTENER UNA O MÁS OPERACIONES REALIZADAS SOBRE LA BILBIOTECA Y CADA UNA DE ESTAS SE AGREGARAN EN LA SECCIÓN DATA DEL BLOQUE. EL BLOQUE SERÁ CREADO AL MOMENTO DE QUE EL USUARIO DESEE GUARDAR EL ESTADO DE SU BIBLIOTECA ACTUAL. CADA BLOQUE SE DEBE GUARDAR EN UNA CARPETA CREADA POR LA APLICACIÓN EN FORMATO JSON.

Ejemplo:

Este sería un ejemplo de las operaciones almacenadas que serán enviadas en el bloque por medio de la sección de **DATA.**

NOTA: los tres puntos representan los datos que cada operación debe de llevar, los mismo ya mencionados anteriormente.

```
"DATA": [
    {
        "ELIMNAR_CATEGORIA": [
          ...
        ]
    },
    {
        "CREAR_CATEGORIA": [
          ...
        ]
    },
    {
        "CREAR_LIBRO": [
          ...
        ]
    },
    {
        "ELIMINAR_LIBRO": [
          ...
        ]
    }
]
```

Reportes

El módulo de reportes debe de permitir ver los estados de las estructuras en cualquier momento durante la ejecución de la aplicación. Los reportes tendrán validez siempre y cuando estos sean elaborados utilizando la herramienta de Graphviz. Los reportes deben de mostrarse y actualizarse dentro de la aplicación. Los reportes que deben de mostrarse son los siguientes:

- Árbol AVL de categorías: el reporte debe mostrar el nombre de la categoría y el total de libros asociados a esa categoría
- Árbol B de cada una de las categoría: es decir que podrá buscarse una categoría específica y mostrar el Árbol B asociado a esa categoría, este reporte debe de mostrar el nombre del libro y su ISBN.
- Tabla de Dispersión de usuarios: el reporte debe mostrar todos los usuarios que existen registrados
- Recorrido preorden del AVL
- Recorrido inorden del AVL
- Recorrido postorden del AVL
- Lista simplemente enlazada que representa el listado de nodos de la red
- Blockchain que representa el log de las operaciones realizadas en la biblioteca

Restricciones

Las estructuras deben de ser desarrolladas por los estudiantes sin el uso de ninguna librería o estructura predefinida en el lenguaje a utilizar.

Observaciones

1. Lenguaje a utilizar: **JAVA**
2. Debe entregar el JAR ejecutable ya que desde ese mismo ser realizará la calificación
3. Herramienta para desarrollo de reportes gráficos: **Graphviz.**
4. La aplicación debe de ser capaz de generar y abrir con un visor de imágenes predeterminado las imágenes generadas con Graphviz.
5. En classroom debe realizar la entrega del ejecutable (JAR) con el que se calificará, de lo contrario no tendrá validez su entrega y tendrá nota de 0.
6. La entrega se realizará por medio de: **Github y Classroom**, cada estudiante deberá crear un repositorio con el nombre: **EDD_1S2020_PY2_#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
   Kevin Mejía: https://github.com/kevin140414
7. 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 asignará en su classroom correspondiente.
8. Fecha y hora de entrega: **Domingo 16 de Mayo, a las 23:59 horas**.
9. **Copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.**