Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y sistemas
Estructuras de Datos

Ingenieros:

- Ing. Luis Espino
- Ing. Edgar Ornelis
- Ing. Álvaro Hernández

Auxiliares:

- Kevin Martinez
- Carlos Castro
- José Montenegro

Proyecto Fase 3

Pixel Print Studio

Implementación de estructuras no lineales

ÍNDICE

Objetivos **3**
Objetivo general 3
Objetivos específicos 3
Descripción General **4**
Estructura **5**
Técnicos 5
Sucursales 6
Rutas 6
Cálculo de ruta más óptima 8
Almacenamiento de Cierre de Servicio 8
Generar Arbol de Merkle 8
Blockchain 10
Aplicación **12**
Observaciones **14**
Entregables **15**

Objetivos

Objetivo general

- Aplicar los conocimientos del curso Estructuras de Datos en el desarrollo de soluciones de software.

Objetivos específicos

- Aplicar los conocimientos adquiridos sobre estructuras de datos lineales y no lineales como tablas hash, grafos y demás temas del curso como encriptación.
- Implementar una aplicación en consola utilizando el lenguaje de programación Fortran.
- Familiarizarse con la lectura y escritura de archivos de JSON.
- Utilizar la herramienta Graphviz para graficar estructuras de datos no lineales.
- Definir e implementar algoritmos de búsqueda, recorrido y eliminación en estructuras de datos.
- Utilizar los conceptos generales de la tecnología blockchain.
- Implementar algoritmos que requieren la utilización de grafos en el desarrollo de la aplicación.

Descripción General

Pixel Print Studio, una empresa especializada en servicios de impresión, ha expandido su presencia a lo largo del país mediante la apertura de múltiples sucursales. Esta expansión ha generado la necesidad de realizar mantenimiento y reparaciones periódicas en todas las tiendas, asegurando así el óptimo funcionamiento de cada una de ellas. Para abordar esta necesidad, se propone el desarrollo de un Sistema de Gestión de Mantenimiento.

El Sistema de Gestión de Mantenimiento tiene como objetivo principal coordinar y registrar las actividades de mantenimiento y reparación realizadas en cada sucursal de Pixel Print Studio. Para lograr esto, el sistema mantendrá un registro detallado de los técnicos asignados a cada tarea de mantenimiento, así como los costos asociados con cada intervención. Dada la naturaleza sensible de los datos involucrados, se pondrá especial énfasis en garantizar la seguridad y confiabilidad de la información almacenada en el sistema.

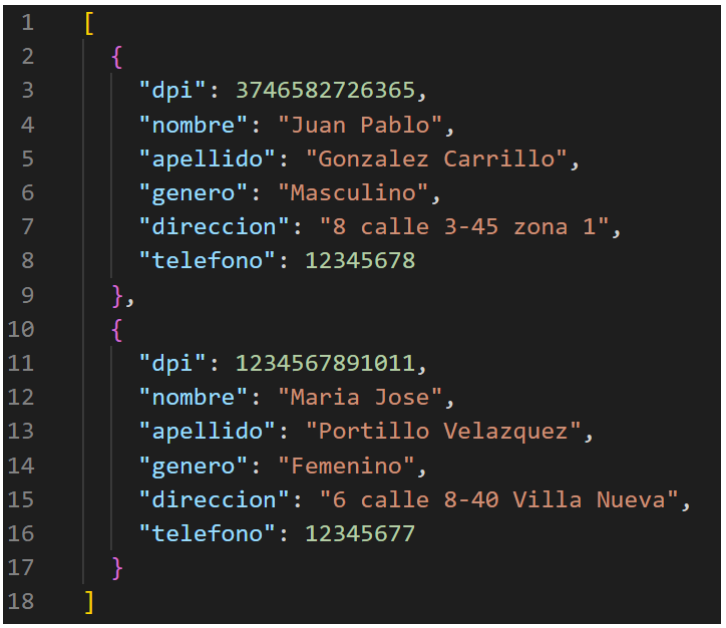
Estructura

Técnicos

Los técnicos son los encargados de ir a las sucursales a reparar o dar mantenimiento a las impresoras. Estos tendrán los siguientes datos:

- DPI
- Nombres
- Apellidos
- Género
- Dirección
- Teléfono

El archivo de entrada debe de ser cargado al ingresar a una sucursal desde la aplicación, por lo tanto cada sucursal podrá tener de 0 a muchos técnicos. El archivo de entrada tendrá el siguiente formato:



```json
[
  {
    "dpi": 3746582726365,
    "nombre": "Juan Pablo",
    "apellido": "Gonzalez Carrillo",
    "genero": "Masculino",
    "direccion": "8 calle 3-45 zona 1",
    "telefono": 12345678
  },
  {
    "dpi": 1234567891011,
    "nombre": "Maria Jose",
    "apellido": "Portillo Velazquez",
    "genero": "Femenino",
    "direccion": "6 calle 8-40 Villa Nueva",
    "telefono": 12345677
  }
]
```

Los técnicos deben de almacenarse en una **tabla hash**, donde se utiliza como llave el número de DPI. La tabla tendrá las siguientes características:

- La función de dispersión será:

  $$h(llv) = llv \bmod M$$

  Donde:
  - h(llv) = Posición a insertar
  - llv = Número de DPI
  - M = Tamaño de la tabla
- La Política para la resolución de colisiones será la doble dispersión por medio de la función:

  $$s(llv, i) = (llv \bmod 7 + 1) * i$$

  Donde:
  - s(llv, i) = Posición a insertar

    - llv = Número de DPI
    - i = Contador de colisiones
- El tamaño inicial de la tabla hash será de 7, cuando la tabla hash llegue a un 70% de uso, se tendrá que hacer rehash de la tabla.

Sucursales

Cada sucursal tendrá cierto número de impresoras de las cuales los técnicos revisarán, en la aplicación debe de haber una opción para cargar los técnicos pero no es obligatorio hacer la carga de los técnicos a todas las sucursales que existan, por lo tanto habrán sucursales con o sin técnicos. La información de cada sucursal será la siguiente:

- ID
- Departamento
- Dirección
- Impresoras para mantenimiento
- Contraseña

```json
[
  {
    "id": 1,
    "departamento": "Guatemala",
    "direccion": "8 calle 3-45 zona 1",
    "password": "S1_20$23"
  },
  {
    "id": 2,
    "departamento": "Huehuetenango",
    "direccion": "3 avenida 5-45 zona 2",
    "password": "TKM2_20$24"
  }
]
```

Al brindarle mantenimiento a las impresoras y solucionar los fallos que estas puedan tener, se espera que la empresa genere ingresos extras de Q100.00 por impresora ya que estarán nuevamente en condiciones de ser usadas para impresiones solicitadas por clientes, estos datos se deberán tomar en cuenta para poder generar los grafos más adelante. Las sucursales se almacenarán en un árbol (elección del estudiante ABB, AVL o B).

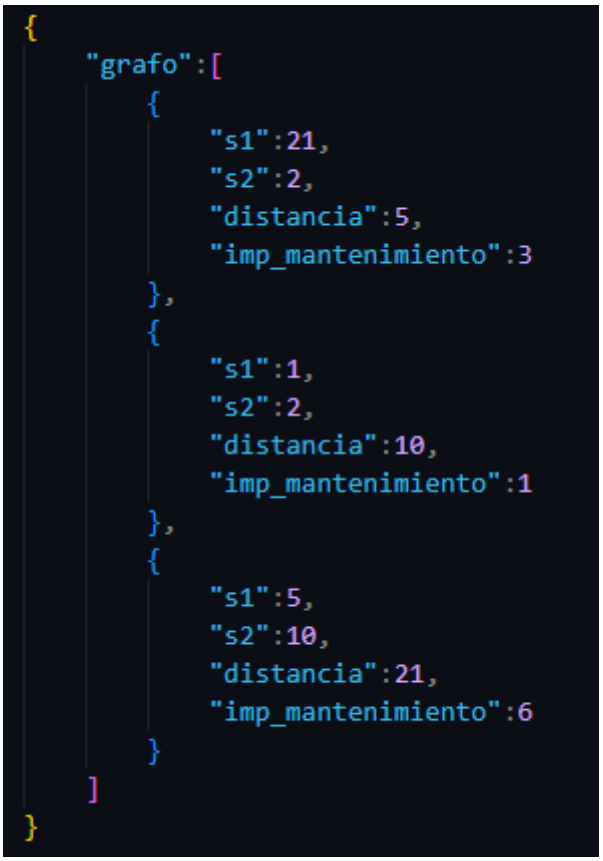
Rutas

Al tener cargadas las sucursales se pueden cargar las rutas, estas rutas contendrán información importante para el uso de los grafos como la sucursal donde inicia y termina una arista, la distancia de la misma y la cantidad de impresoras que se restaurarán al momento de recorrer dicha arista. El costo de la distancia por kilómetro es de Q 80.00, esta

información será necesaria para calcular el costo total que realiza cada técnico en su recorrido. La información de las rutas son las siguientes:

- Sucursal 1
- Sucursal 2
- Distancia en kilómetros
- Cantidad de impresoras a las que se le brindaría mantenimiento si se escogiera ese camino

A continuación se muestra un ejemplo del archivo de entrada.



Cálculo de ruta más óptima

Como se mencionó anteriormente, las aristas de los grafos poseen dos tipos de métricas a utilizar, la distancia que se posee entre dos sucursales conectadas y la cantidad de impresoras a las cuales se les brindaría mantenimiento para recorrer dicha arista. Como parte del sistema de gestión de mantenimiento se le solicita a usted que genere, dado un punto de partida (sucursal x) y un punto de finalización (sucursal y) calcule la ruta que minimice la distancia recorrida (y la cantidad de impresoras a las cuales se les proporcionó mantenimiento al seguir esta ruta) y también que calcule la ruta que maximice la cantidad de impresoras a las cuales se les brinda mantenimiento (y la distancia que se recorrió al seguir esta ruta). Finalmente se deberá contrastar ambas soluciones y brindar, a la persona

que use el sistema, la solución que minimice los gastos totales de la empresa realizando la diferencia entre la ganancia (impresoras atendidas) y la pérdida (distancia recorrida).

Almacenamiento de Cierre de Servicio

Debido a que es necesario tener persistencia en los datos que se almacenan en la aplicacion, para que estos sean confiables, no solo para los usuarios sino también para futuras revisiones de la empresa, es necesario implementar una estructura de datos que pueda mantener la información íntegra, el árbol de Merkle es una estructura de datos que sirve para realizar el guardado de transacciones que se se realizan dentro de las aplicaciones, para lo cual se necesita seguir ciertos pasos.

Para determinar el recorrido que llevará a cabo cada técnico se debe de ingresar a la sucursal con el id (este será el inicio del recorrido), poner el id de la sucursal final que es donde terminará el cierre del serivicio y por último el id del técnico que realizará el servicio.

Generar Arbol de Merkle

Como se mencionó anteriormente, se deberán generar dos rutas distintas (una tomando la cantidad de impresoras a las cuales se les brindará mantenimiento como peso y otro tomando la distancia como peso), después de hacer esto se deberá decidir cuál de las dos rutas fue la que menos costos representa para la empresa. Cada una de estas rutas estará conformada de distintas sucursales que debieron de ser visitadas para llegar desde la sucursal de origen hasta la sucursal de destino, por cada una de estas visitas se deberá hacer una inserción en el árbol de Merkle, la cual contendrá los siguientes datos:

- ID de sucursal de origen
- Dirección de sucursal de origen
- ID de sucursal de destino
- Dirección de sucursal de destino
- Costo total de la arista entre estas dos sucursales

Ejemplo:
hash = funcionHash(Data)
Donde:

- **Hash:** Será el identificador correspondiente a la data que se almacenará en los nodos hoja del árbol de merkle.
- **FunciónHash:** Es el método utilizado para encriptar la información que servirá de entrada para la función, para esto se utilizará Sha256.
- **Data:** Es la información que se usará para poder generar el hash, en esta data se deberá incluir toda la información indicada anteriormente.

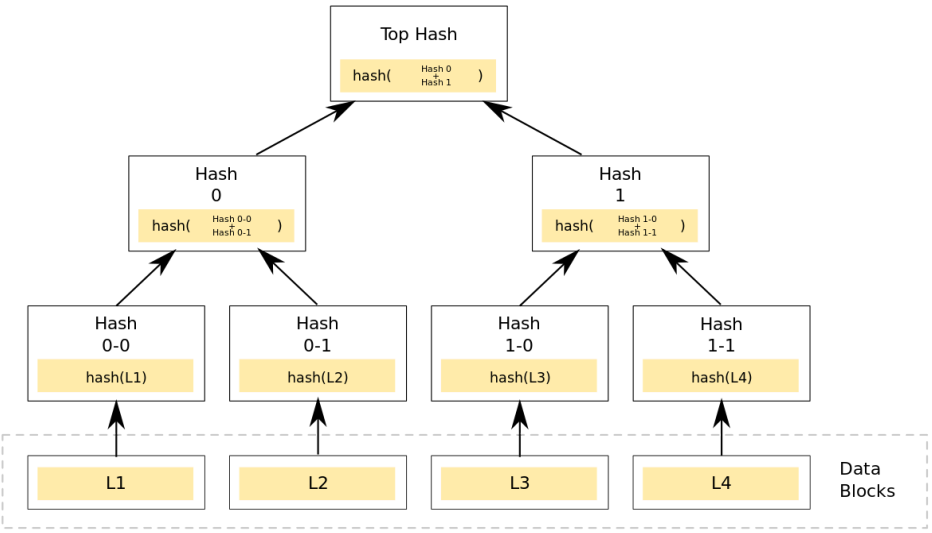
Ejemplos de nodos hoja
hash = sha256(“s1”, “dirección s1”, “s2” …)
hash = 39b923082a194166d8d92989116bd

Ejemplos de nodos internos

Estos nodos se crean tomando como base los hash de los nodos hoja, estos son la entrada de la función para generar el hash de dicho nodo interno.

hash = sha256(“39b923082a194166d8d92989116bd, aa17d5fdf24761b54ad640691146656095b”)
hash = 113b685b12dbfa76c04bec7759a24ba66dd

Ejemplo

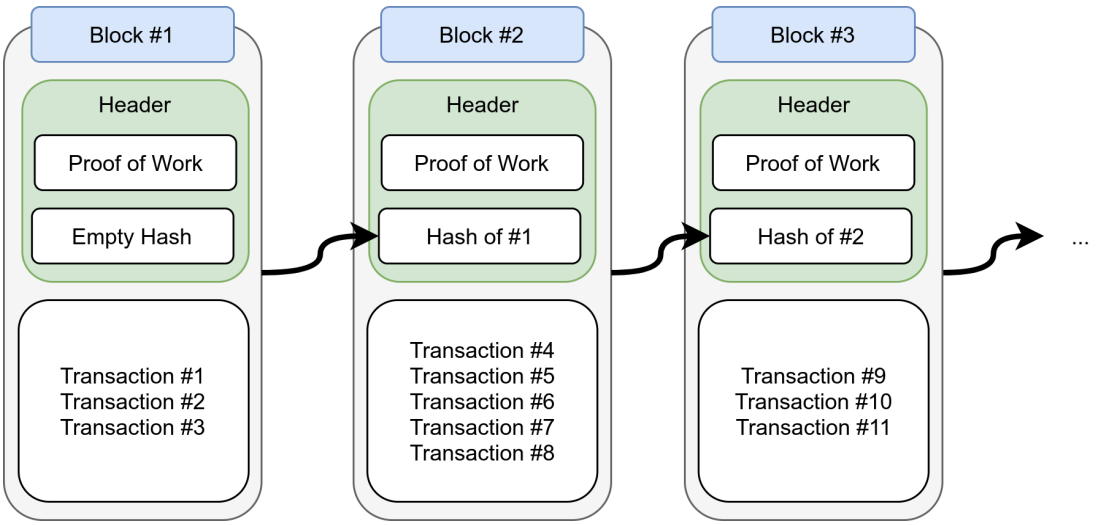


Notas:

- El árbol de Merkle es un árbol que, por concepto, siempre está totalmente lleno (es decir, todos los nodos en el árbol poseen el número máximo de hijos que pueden tener), sin embargo, puede ser que las aristas que se recorran en los caminos que se almacenen no formen una cantidad de nodos hoja con los cuales sea posible llenar completamente el árbol, cuando esto ocurra se debe completar con valores representativos nulos, como el -1.

Blockchain

Esta es una estructura de bloques, los cuales se comportan como una lista enlazada simple, en donde cada uno de los nodos(bloques) se ingresan por delante.



Prueba de trabajo

Es el proceso por el cual se encuentra un hash que cumpla con la condición de tener un prefijo de n ceros, donde n corresponde a un número entero, para el caso de la aplicación este poseerá un valor de 4. Para esto se debe iterar un número entero (NONCE) el cual iniciará desde cero y se incrementará en 1 hasta encontrar un hash válido para el bloque.

Generación de nuevos bloques

Un nuevo bloque se deberá generar cada vez que se encuentre el camino menos costoso de una sucursal de origen a una sucursal de destino. La información de las aristas recorridas (como se explica en la sección “Generar árbol de Merkle”) será almacenada en la sección Data del bloque, esto se explicará a continuación.

Bloque

El bloque se compone de distintas partes que contienen información acerca del mismo, estas partes son:

- **INDEX:** Representa el número del bloque que se estará insertando en la cadena, este campo es incrementado en uno para cada bloque que se inserta por lo cual, naturalmente, el primer bloque tendrá un índice de cero y los demás tendrán índices con valores sucesivos (1,2,3,etc.).
- **TIMESTAMP**: Tendrá la fecha y hora en la cual se creó el bloque, poseerá el siguiente formato: (DD-MM-YY-::HH:MM:SS).
- **DATA**: Este campo poseerá un listado de todas las aristas recorridas a lo largo del camino más óptimo de una sucursal a otra.
- **NONCE:** Será el número entero que se debe iterar de uno en uno hasta encontrar un hash que cumpla con la prueba de trabajo, es decir, que contenga el prefijo con la cantidad de 0s designada (cuatro).
- **PREVIOUS HASH:** Corresponde a, como se puede observar en la imagen, el hash del bloque anterior a cada uno de los bloques. Esto actúa como un apuntador que indica si los datos en algún punto de la cadena de bloques fueron corrompidos, en el caso del primer bloque no se posee ningún bloque previo por lo cual el valor de este apuntador será “0000”.

- **ROOTMERKLE**: Como se mencionó anteriormente, con las aristas de un camino, se deberá hacer un árbol de Merkle, el cual en su raíz tendrá un hash que representará todo el conjunto de estos datos, en este atributo se almacena este hash.
- **HASH:** El hash que identifica al bloque actual y protege la data que este almacena de ser corrompida. Este hash se genera aplicando la función SHA256 a las propiedades: INDEX, TIMESTAMP, PREVIOUSHASH, ROOTMERKLE y NONE. Todas deben ser cadenas concatenadas sin espacios en blanco ni saltos de línea.

Estructura de los bloques

```
{
    "INDEX": 0,
    "TIMESTAMP": "01-04-2024::11:30:29",
    "NONCE":4560,
    "DATA":[
        {
            "sucursal_o": "s1",
            "direccion_o": "5ta av 4 calle zona 18, Guatemala, Guatemala",
            "sucursal_d": "s2",
            "direccion_d": "6ta av 12 calle zona 1, Guatemala, Guatemala",
            "costo": 200
        },
        {
            "sucursal_o": "s2",
            "direccion_o": "6ta av 12 calle zona 1, Guatemala, Guatemala",
            "sucursal_d": "s3",
            "direccion_d": "9na av 5 calle zona 6, Guatemala, Guatemala",
            "costo": 150
        }
    ],
    "PREVIOUSHASH": "0000",
    "ROOTMERKLE": "b55126a39f9b1170a32e6f61e4a694c45235e5ac11c05ecd6ff6395de6a11187",
    "HASH":"13c78c707b010724cd9e1f596b58246a2c829384fc8a8a4b49aa38b3fddfc1c2"
}
```

La información almacenada en DATA representa el camino óptimo de una sucursal de origen a una sucursal destino, por lo tanto, un bloque deberá ser generado después de realizar estos cálculos.

Almacenamiento de bloques

Los bloques deben ser almacenados en archivos JSON, para que estos puedan ser leídos al iniciar la aplicación y poder tener persistencia de las rutas óptimas entre todas las sucursales.

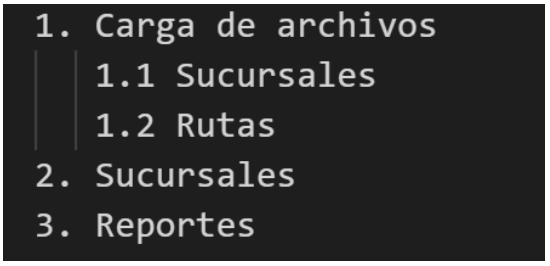
Para esto los alumnos deberán tener una carpeta en la raíz de su proyecto que llevará por nombre **blockchain**. En esta carpeta se deberá almacenar todos los bloques que se generen durante la ejecución de la aplicación.

Estos bloques deberán ser leídos al momento de iniciar la aplicación.

Aplicación

Para poder acceder a la aplicación primero debe loguearse el usuario administrador, por medio del usuario y contraseña.

Una vez logueado tendrá acceso a realizar la carga de sucursales, rutas, ingresar a una sucursal y ver reportes.



Para la opción de sucursal, debe de solicitar el id de la sucursal y la contraseña. Una vez dentro de una sucursal se tendrán las siguientes opciones:

- **Carga de tecnicos**
  Se debe solicitar la ruta para cargar el json con los técnicos
- **Generar recorrido más óptimo**
  Se tiene que solicitar el ID de la sucursal final y el técnico que realizará el trabajo, con estos datos se deben de generar dos grafos, uno con la ruta más corta en base a la distancia y otra maximizando las impresoras arregladas.
  En consola se tiene que mostrar las ganancias generadas en cada uno de los recorridos y recomendar la ruta que genere mayor ganancias.
- **Información Técnico en Específico**
  Se debe de solicitar el id del técnico y se debe de mostrar toda la información del mismo.
- **Listar Tecnicos**
  Al seleccionar esta opción se debe de mostrar en una tabla los técnicos que tiene la sucursal.
- **Generar Reporte**
  Se debe de mostrar el top 5 de técnicos con más trabajos realizados. También los costos y ganancias que genera la sucursal al momento de que se haga un recorrido.

En el área de reportes se debe de mostrar las estructuras (árbol de Merkle y Blockchain), el top 5 de sucursales con más trabajos solicitados y las ganancias totales.

Reportes

Gráficos:

- Grafo de sucursales y sus rutas
- Árbol de Merckle
- BlockChain
- Tabla Hash

Datos de la empresa:

- Top 5 técnicos con más trabajos realizados
- Top 5 sucursales con más trabajos solicitados.
- Ganancias.
- Costos.
- Ganancias Totales.

- Ruta de viaje por trabajo realizado.

Protección de los datos

Para poder mantener segura la información de los clientes se deberá encriptar las contraseñas tanto de las sucursales como del usuario administrador.

Observaciones

- Lenguaje de programación a utilizar: **FORTRAN**
- El nombre del usuario administrador será **EDD1S2024** y su contraseña será **ProyectoFase3**.
- Sistema Operativo: Libre
- IDE: Libre.
- Herramienta para desarrollo de reportes gráficos: Graphviz.
- Las gráficas deben mostrarse dentro de la aplicación, no buscarse en carpetas ajenas.
- Durante la calificación se harán preguntas para validar que el estudiante realizó el proyecto, de no responder correctamente anulará la nota obtenida en la o las secciones en la que se aplique tal concepto.
- Cada estudiante deberá utilizar el repositorio de la fase anterior separando por carpetas cada fase del proyecto, verificar que su auxiliar esté agregado
- como colaborador para poder analizar su progreso.
- Apartado de entrega en la plataforma UEDI: Fecha y hora de entrega: 29 de abril a las 23:59 horas.
- Las copias serán penalizadas con una nota de 0 y castigadas según lo indique el reglamento.

Entregables

- Manual de Usuario
- Manual Técnico
- Link a repositorio
- Código fuente