Proyecto → Fase 2

Sistema ERP

Ing. Luis Espino

Josué Pérez

Vacaciones diciembre 2021

Facultad de Ingeniería

Estructuras de Datos

Visión General

Un ERP (siglas de 'Enterprise Resource Planning' o 'Planificación de Recursos Empresariales') es un conjunto de aplicaciones de software integradas, que nos permiten automatizar la mayoría de las prácticas de negocio relacionadas con los aspectos operativos o productivos de nuestra empresa, facilitando y centralizando la información de todas las áreas que la componen: compras, producción, logística, finanzas, recursos humanos, marketing, servicios, proyectos y atención al cliente.

Debido al éxito de la fase anterior, se ha decidido expandir el proyecto incorporando nuevas funcionalidades que son, **Manejo de inventario, Registro de ventas (facturas), seguridad de la información sensible (criptografía) , implementación de Blockchain para la persistencia la información.** Por lo que estas nuevas funcionalidades se agregaran a las ya existentes en la aplicación.

Objetivos

- Que el estudiante se familiarice con el lenguaje de programación JavaScript.
- Que el estudiante sepa envolverse en el ámbito del manejo de la memoria.
- Que el estudiante sea capaz de implementar estructuras de datos complejas en una aplicación funcional.
- Familiarizarse con el uso de Git.
- Que el estudiante se familiarice con el manejo de lectura de archivos.
- Comprender el uso de estructuras de datos compl

Especificaciones

A la fase anterior se le agregaran funcionalidades, haciendo uso de estructuras de datos avanzadas con las cuales poder manejar la información necesaria para cada funcionalidad. El lenguaje utilizado seguirá siendo JavaScript por lo que todas las EDD's deben ser desarrolladas en este lenguaje y la aplicación debe estar publicada en GitHub Pages.

Seguridad

Para la seguridad en esta fase se plantea la posibilidad de encriptar la información más sensible de los usuarios de la aplicación (vendedores), esto para evitar el robo de la información asociada a los servicios que se brindan en el sistema, para poder visualizarla el usuario que trate de hacerlo deberá contar con una llave que le permita el acceso a la misma.

Esto será implementado de la siguiente manera, para la encriptación de contraseñas se debe realizar utilizando "Sha256", esto posibilitará que se encripte las contraseñas, correo y cualquier otra información que considere sensible, de una manera segura.

posible sugerencia puede ser con la librería sjcl para java, pero el uso de las librerías se deja libre para esta parte de la encriptación. La forma de proceder es que será solicitado en el usuario administrador la contraseña maestra para la encriptación y esta contraseña será la utilizada al momento de realizar dicha encriptación, esto puede ser recibido en una entrada de texto en el Frontend.

Inventario

Para esta fase, se agregará el módulo de manejo de inventario, este será manejado atravesó de un Árbol B de orden 5. En esta estructura se deben almacenar los productos que el administrador ingrese por medio de carga masiva o de manera manual.

Los campos utilizados para los productos son:

- Id
- Nombre
- Precio
- Cantidad

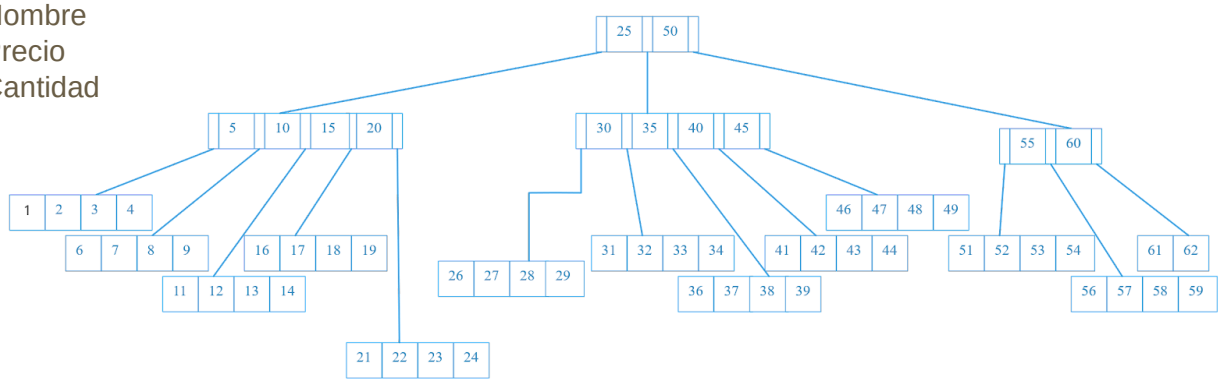


Imagen de referencia.

Registro de ventas

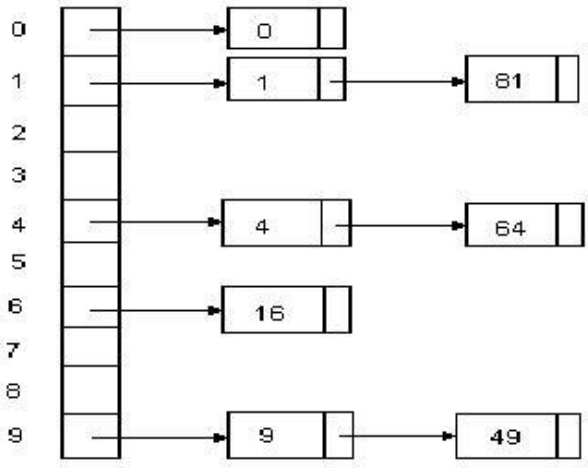
Para el registro de ventas, se utilizará una tabla hash general, donde se almacenarán las ventas de todos los vendedores.
Para esta estructura se utilizará una tabla hash con los siguientes parámetros:

Para calcular las claves, la función hash a utilizar debe ser el **método de división.** El tipo de hash será cerrado y el método para resolver colisiones debe ser **exploración cuadrática.**

El **tamaño inicial** de la tabla hash será de **7** y cuando se llegue a un **50% de uso**, se tendrá que hacer rehash de la tabla, el tamaño debe crecer al **número primo próximo.**

La información que debe almacenar cada nodo de la tabla hash es la siguiente:

- Id_venta (generado automáticamente)
- Nombre vendedor
- Nombre cliente
- Total de venta (calculado de la lista de productos)
- Lista enlazada de productos (productos registrados en el árbol B)



La tabla hash será manejada por hashsing cerrado, las lista mostradas en la imagen son el atributo de los productos de cada venta.

BlockChain

Esta estructura se utilizará para almacenar todas las transacciones de ventas realizadas. Este registro deberá realizarse cada cierto periodo de tiempo, este tiempo lo podrá cambiar el administrador desde la vista de configuración, pero inicialmente el **tiempo será de 5 min.**

Creación de bloque:

Una vez cumplido el tiempo configurado, la aplicación deberá crear un bloque de blockchain, donde se guarden las transacciones realizadas en el periodo de tiempo definido, si no hubieran transacciones de debe crear el bloque "vacío".

Cada bloque deberá almacenar la siguiente información:

- **Índice:** el número correlativo del bloque, el bloque inicial tendrá el valor **0.**
- **Fecha:** Almacena el momento exacto en el que se genera el bloque, debe de tener el siguiente formato (DD-MM-YY::HH:MM::SS).
- **Data:** Deben ser todas las somarizaciones de las transacciones realizadas.
- **Nonce:** Es un número entero que se debe iterar de uno en uno hasta encontrar un hash válido, **por defecto** el número de ceros al inicio del hash debe de ser de **4 ceros (0000) pero este valor debe poder cambiarse desde el panel de administrador**.
- **PreviousHash:** Es el hash del bloque anterior, nos permite validar que la cadena del bloque no esté alterada. Para el primer bloque el hash anterior debe ser 0000.
- **Hash:** Es el hash del bloque actual.

Crear el hash:

Para crear el hash de este bloque se debe hacer uso del algoritmo SHA256, utilizando las propiedades listadas anteriormente. Todos estos bloques deben ir como cadenas concatenadas sin espacios en blanco:

SHA256(Indice + Fecha + PreviousHash + Data + Nonce)

Para encontrar un hash válido para nuestro bloque, debemos tomar en cuenta lo mencionado en la prueba de trabajo.

Prueba de trabajo:

Es un sistema con el fin de dificultar el proceso de generación de hashes válidos, para evitar comportamientos indeseados como ataques o spam. Para esto se debe iterar el campo Nonce hasta encontrar un hash válido para el bloque.

Para que el hash cumpla con la condición de que contenga una cantidad de ceros al inicio, por defecto el número de ceros al inicio del hash debe de ser de 4 ceros (0000) pero este valor debe poder cambiarse desde el panel de administrador.

Guardar bloque:

Luego de terminar el proceso de crear un nuevo bloque, se procede a almacenar el archivo en la carpeta de bloques.

Para generar el bloque el estudiante debe generar un archivo en el cual guardará la información necesaria para que al ser leída, la aplicación siga teniendo los mismos datos que tenía al momento de cerrarse o ser interrumpida.

Estos bloques no son legibles para personas que tienen la intención de vulnerar el sistema o manipular los datos.

Notas:

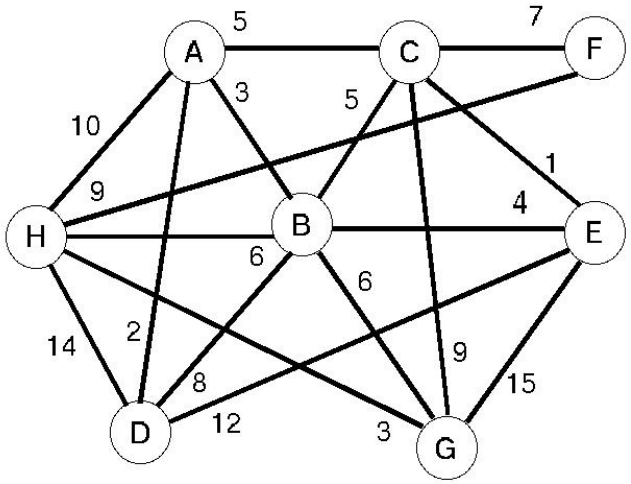
1. Los bloques se deben guardar en una carpeta llamada "**bloques**", de no existir esta carpeta debe de crearse.
2. El nombre de los bloque debe ser manejado por correlativos para evitar que se reemplace los mismos y haya pérdida de información o inconsistencia.
3. Si no hay nuevas transacciones en la estructura, se generará un nuevo bloque pero en el apartado de DATA no tendrá información por lo que se encontrará vacio.
4. Sel usuario administrador debe tener acceso a esta interfaz.

Grafo

Para esta fase, se implementa un modulo de rutas de las bodegas de los proveedores, esto para administrar de mejor manera el recurso de transportes de las empresas.

Este modulo nos permitirá ingresar todas las bodegas de los proveedores con las distancias entre ella y calcular la ruta optima de los camiones de la empresa para hacer un recorrido para recoger productos.

Para esto se utilizara la estructura de datos de Grafo ponderado.



Estructuras

- Árbol B
    - Utilizado para almacenar el inventario de producto
- Tabla hash
    - Utilizado para almacenar las ventas de los usuarios
    - Atributo de lista para almacenar los productos de la venta.
- Grafo
    - Utilizado para almacenar las rutas de los proveedores y definir las rutas optimas.
- BlockChain
    - Utilizado para la consistencia de las transacciones de las ventas.

Carga masiva

La carga de los datos se realizará mediante archivo json. A continuación, se detalla el formato para cada carga.

- Inventario

```
{
    "productos":[
        {
            "id":111,
            "nombre":"producto1",
            "precio":150.75,
            "cantidad":150
        },
        {
            "id":112,
            "nombre":"producto2",
            "precio":150,
            "cantidad":500
        },
        {
            "id":113,
            "nombre":"producto3",
            "precio":50,
            "cantidad":100
        },
        {
            "id":114,
            "nombre":"producto4",
            "precio":14.90,
            "cantidad":1000
        }
    ]
}
```

- Ventas (facturas)

```
{
    "ventas":[
        {
            "vendedor":"vendedor1",
            "cliente":"cliente1",
            "productos":[
                {
                    "id":1,
                    "cantidad":3
                },
                {
                    "id":10,
                    "cantidad":1
                }
            ]
        },
        {
            "vendedor":"vendedor2",
            "cliente":"cliente2",
            "productos":[
                {
                    "id":5,
                    "cantidad":10
                },
                {
                    "id":10,
                    "cantidad":2
                },
                {
                    "id":20,
                    "cantidad":7
                }
            ]
        }
    ]
}
```

- Rutas

```
{
    "rutas":[
        {
            "id":1,
            "nombre":"bodega1",
            "adyacentes":[
                {
                    "id":3,
                    "nombre":"bodega3",
                    "distancia":15
                },
                {
                    "id":5,
                    "nombre":"bodega5",
                    "distancia":5
                }
            ]
        },
        {
            "id":2,
            "nombre":"bodega2",
            "adyacentes":[
                {
                    "id":3,
                    "nombre":"bodega3",
                    "distancia":5
                },
                {
                    "id":4,
                    "nombre":"bodega4",
                    "distancia":2
                }
            ]
        },
        {
            "id":3,
            "nombre":"bodega3",
            "adyacentes":[
                {
                    "id":1,
                    "nombre":"bodega1",
                    "distancia":15
                },
                {
```

```
                    "id":2,
                    "nombre":"bodega2",
                    "distancia":5
                }
            ]
        }
    ]
}
```

Reportes

El Administrador será el que pueda ver los reportes de manera gráfica, por lo de se deberá poder generar el reporte de cada estructura en cualquier momento.

- **Reporte Inventario,** muestra todos los productos almacenados en el árbol B.
- **Reporte de ventas,** muestra las ventas registradas por todos los vendedores, grafica de la tabla hash.
- **Ventas por vendedor,** muestra las ventas realizadas por un vendedor, este reporte lo podrá ver tanto el administrador como el vendedor.
- **Rutas,** muestra la gráfica el grafo de las rutas registradas, con las aristas ponderadas.
- **Ruta optima,** recibe como parámetros el origen y el destino, y deberá calcular la ruta que sea más corta.
- **Usuarios Encriptados,** muestra el reporte del árbol AVL con la información encriptada.
- **BlockChain,** este reporte estará conformado por los bloques generados y el grafico de la cadena de bloques.

Para todos los reportes mostrar toda la información que se considere necesaria (id, nombre, correo, etc.).

Consideraciones

- **El lenguaje para utilizar será JavaScript.**
- **Se debe utilizar la plataforma de Github Pages.**
- **Las estructuras serán realizadas por el estudiante.**
- **IDE para utilizar es libre (Recomendaciones: Visual Studio Code, Sublime Text).**
- **La entrega será por medio de la plataforma UEDI.**
- **El estudiante debe tener un repositorio privado en github con el nombre EDD_Fase1_#Carné y agregar a su tutor como colaborador al repositorio del proyecto.**
- **Será calificado del último commit realizado antes de la fecha y hora de entrega.**
- **Las copias totales o parciales serán penalizadas con nota de 0 puntos y reportadas ante la escuela de ciencias y sistemas.**
- **Fecha de entrega: 4 de Enero de 2022 antes de las 23:59 horas.**