

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos “A”
Vacaciones Junio 2020
Catedrático: Ing. Luis Espino
Tutor Académico: Susel Retana



PROYECTO # 2

Objetivos

Objetivo General

- Aplicar los conocimientos del curso de Estructuras de Datos en la creación de soluciones de software.

Objetivos Específicos

- Aplicar los conocimientos adquiridos sobre estructuras de datos lineales y no lineales, árboles, grafos, blockchain, y tablas dispersas.
- Aplicar los conocimientos adquiridos sobre memoria dinámica y apuntadores en el lenguaje de programación Java.
- Utilizar la herramienta graphviz para la generación de reportes gráficos.

Llega Rapidito

Descripción General

La empresa Llega Rapidito, durante varios años ha prestado su servicio de transporte con cobertura en todo el país.

Los directores de la compañía han decidido realizar un cambio de sistema para el control de vehículos, conductores, clientes y viajes, ya que el sistema actual cuenta con varios fallos y últimamente algunos procesos trabajan de forma lenta, esto debido a la gran cantidad de registros que maneja la empresa.

Por lo que se le ha contratado a usted para el desarrollo de una nueva solución, esta contará con una aplicación que estará implementada con estructuras de datos en memoria, esto con el propósito de agilizar los procesos utilizados para el control de la información.

Aplicación

Aplicación desarrollada en lenguaje de programación Java de manera gráfica. Por medio de esta aplicación se realizará el control de toda la información de interés para la empresa.

Control de Información

El sistema debe manejar registros de las siguientes entidades:

- Clientes
- Vehículos
- Conductores
- Viajes
- Rutas

Para cada entidad se deben realizar las siguientes operaciones:

- Agregar
- Modificar (Ingresando llave)
- Eliminar (Ingresando llave)
- Mostrar Información (Ingresando llave)
- Mostrar Estructura de Datos (Graphviz)

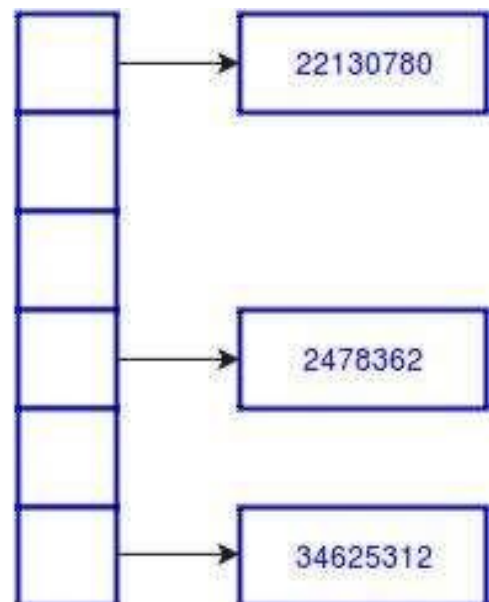
Cientes

La empresa desea tener registro de sus clientes, para la aplicación de descuentos y promociones, para los clientes se maneja la siguiente información:

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

El control de clientes, estos se almacenarán en una tabla hash, utilizando como llave el número de DPI.

- El tamaño inicial de la tabla hash será 37
($M = 37$)
- La función de dispersión a utilizar es la siguiente:
 $h(llv) = llv \bmod M$
- El porcentaje máximo de ocupación será del 75%
- La política para la resolución de colisiones será la doble dispersión por medio de la función:
 $s(llv, i) = (llv \bmod 7 + 1) * i.$



Carga Masiva de Clientes:

La aplicación debe contar con la opción para realizar la carga masiva de clientes por medio de un archivo de texto con el siguiente formato:

DPI, Nombres, Apellidos, Género, Fecha Nacimiento, Teléfono, Dirección;

Ejemplo:

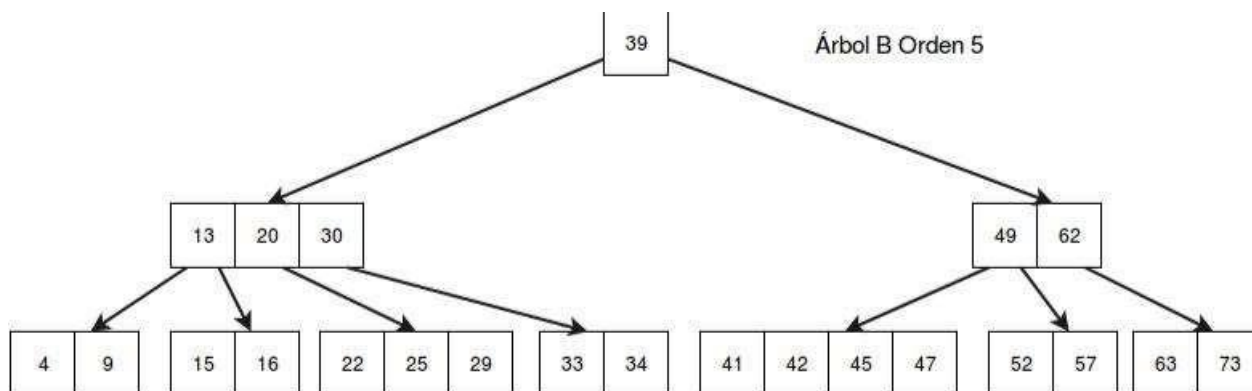
```
221374732231, José Carlos, López Morales, Masculino, 17/08/1982, 54789832, 19 Av. 17-01 Z.6 Ciudad de Guatemala;  
342123984521, Gabriela, Pérez Sandoval, Femenino, 09/09/1984, 45621892, 20 Calle 19-78 Z.7 Ciudad de Guatemala;  
465321232212, Carlos Alejandro, Flores Gómez, Masculino, 27/04/1978, 54678976, 4ta Av. 12-08 Z.5 Ciudad de Guatemala;
```

Vehículos

Para los vehículos se utilizará la siguiente información:

- Placa
- Marca
- Modelo
- Año
- Color
- Precio (en quetzales)
- Tipo de Transmisión (Automática o Mecánica).

Los vehículos se almacenarán en un árbol B de orden 5, utilizando como llave la placa de cada vehículo.

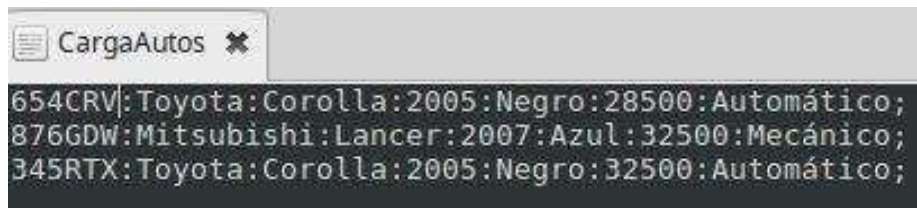


Carga Masiva de Vehículos

La aplicación debe contar con la opción para realizar la carga masiva de vehículos por medio de un archivo de texto con el siguiente formato:

Placa : Marca : Modelo : Año : Color : Precio : Transmisión;

Ejemplo:



```
CargaAutos
654CRV:Toyota:Corolla:2005:Negro:28500:Automático;
876GDW:Mitsubishi:Lancer:2007:Azul:32500:Mecánico;
345RTX:Toyota:Corolla:2005:Negro:32500:Automático;
```

Conductores

Para cada conductor de taxi se manejará la siguiente información:

- DPI
- Nombres
- Apellidos
- Tipo de Licencia (A, B, C)
- Género
- Teléfono
- Dirección

Los conductores se almacenarán en una lista doblemente enlazada circular ordenada por número de DPI.

Carga Masiva de Conductores:

La aplicación debe contar con la opción para realizar la carga masiva de clientes por medio de un archivo de texto con el siguiente formato:

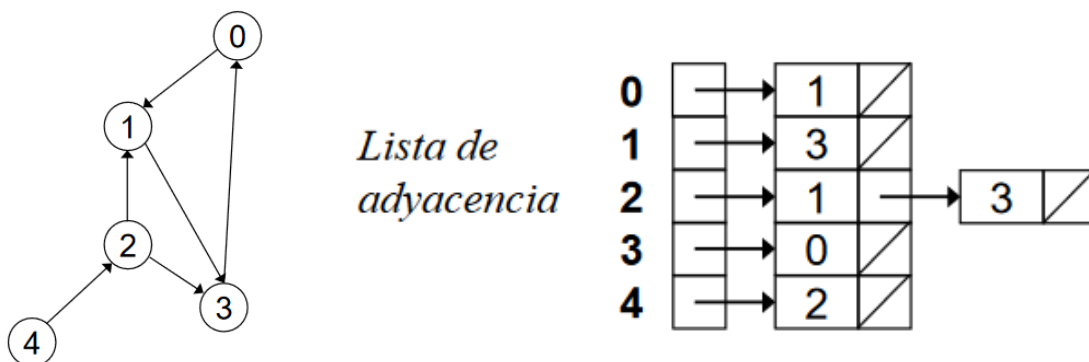
DPI%Nombres%Apellidos%Licencia%Género%Fecha Nacimiento%Teléfono% Dirección;

Rutas

Al iniciar la aplicación, se deberá pedir la carga de un archivo el cuál contendrá las rutas de un país, en el archivo de proporcionará:

- Origen
- Destino
- Tiempo de Ruta

Por lo que se deberá ir realizando un mapa general de todas las rutas, implementado como un grafo que será implementado por el método de listas de adyacencia



Carga de Rutas

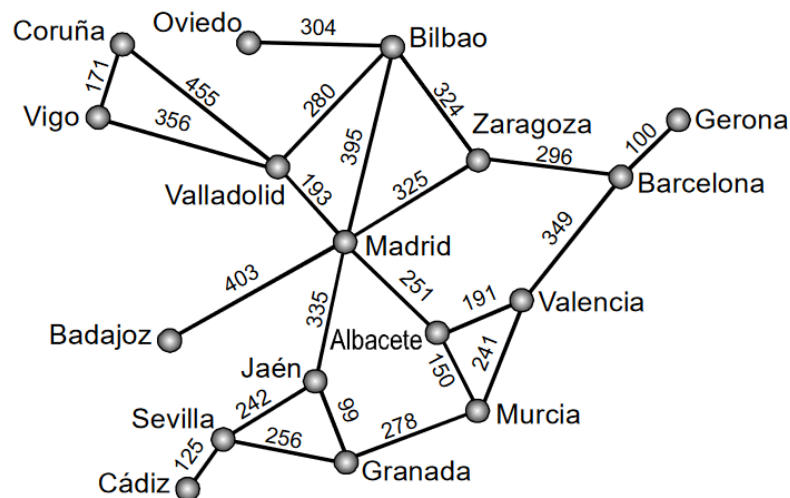
Las rutas se cargarán al iniciar la aplicación, por medio de un archivo de texto con el siguiente formato:

Lugar Origen / Lugar Destino / Tiempo de Ruta %

Cada lugar tiene un nombre único, y el tiempo de ruta será proporcionado en segundos, proporcionados únicamente como números enteros.

Ejemplo:

Oviedo / Bilbao / 304 %
Bilbao / Zaragoza / 324 %
Bilbao / Madrid / 395 %
Bilbao / Valladolid / 280 %
Zaragoza / Barcelona / 296 %
Barcelona / Gerona / 100 %
Zaragoza / Madrid / 325 % ...



Viajes

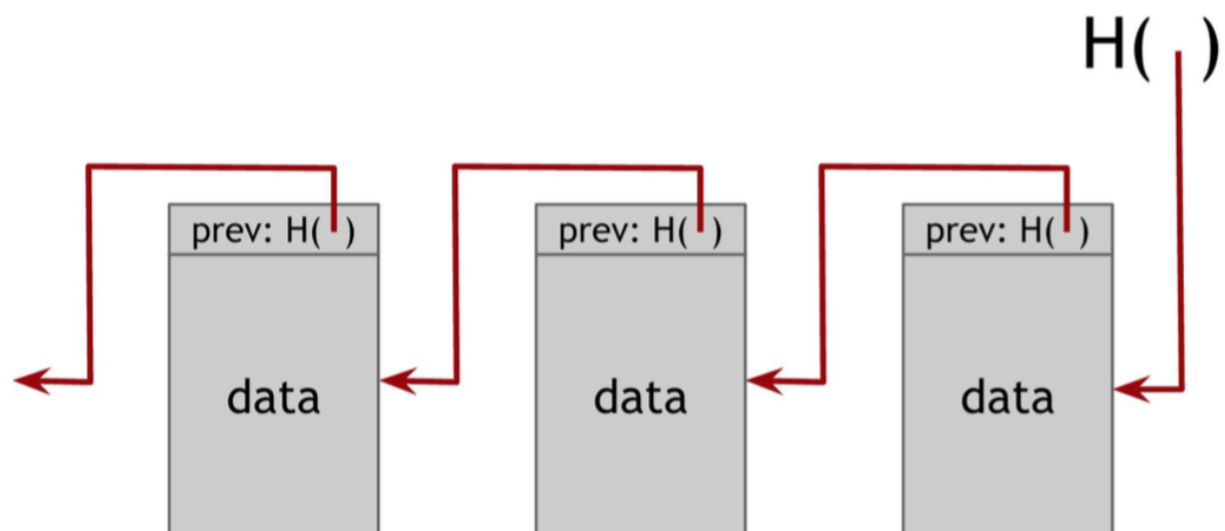
Para cada viaje se tendrá la siguiente información:

- Lugar de Origen
- Lugar Destino
- Fecha y Hora de Inicio
- Cliente (Apuntador a Tabla Hash)
- Conductor (Apuntador a Lista Enlazada)
- Vehículo (Apuntador a árbol B)
- Ruta Tomada. (Apuntador a Lista Simple)
 - o El programa debe ser capaz de determinar el mejor camino por tomar, basándose en el grafo generado con el archivo cargado de rutas, y guardando en una lista simple, la ruta tomada, cada nodo debe tener: Nombre del lugar, Enlace al siguiente lugar, y Tiempo de llegada.



Se llevara el control de los viajes por medio de una estructura Blockchain, utilizando como llave la concatenación de la placa con la fecha y hora del viaje.

Formato llave: PlacaDDMMYYHH:MM (utilizando un formato de 24 horas para la hora del viaje), la llave deberá ser encriptada, por medio de la función de encripta miento hash **MD5** o **SH1**, para su almacenamiento en la estructura.



Reportes

Serán generados en la aplicación por medio de la herramienta graphviz.

- Estructura Completa: Se mostrarán **todas las estructuras de datos** y la relación entre ellas.
- Top Viajes: Top 10 de viajes más largos (por número de destinos).
- Top Clientes: Top 10 de clientes con mayor cantidad de viajes.
- Top Conductores: Top 10 de conductores que han generado mayor cantidad de ingresos para la empresa.
- Top vehículos: Top 10 de vehículos con mayor cantidad de viajes.
- Ruta de un viaje (Ingresando llave)

Nota: Los reportes de tipo TOP deberán mostrarse en la aplicación como texto y a su vez se deberán generar archivos (distintos por reporte) de tipo. edd que podrán ser comprimidos y descomprimidos desde la aplicación por medio del método Huffman.

Flujo de la Aplicación

Básicamente el sistema tendrá el siguiente flujo:

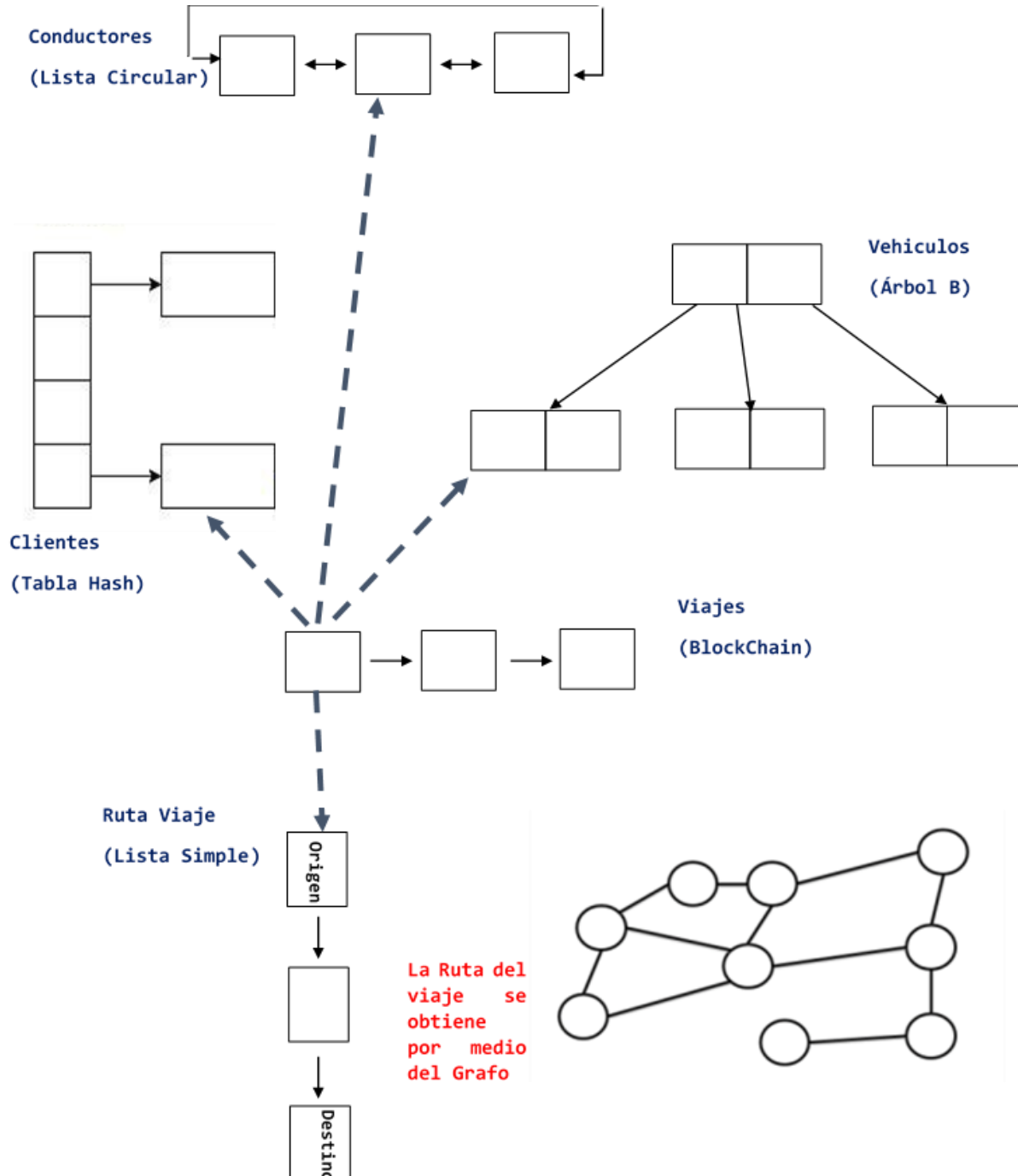
- Al iniciar la aplicación se cargará el archivo de rutas, generando el grafo del cual se obtendrán las rutas mas eficientes para los viajes.
- Se podrán realizar cargas masivas de:
 - o Clientes
 - o Vehículos
 - o Conductores
- Se debe tener un menú por cada entidad (Clientes, Vehículos, Conductores, Viajes)
Cada menú contara con la opción de:
 - Crear
 - Modificar (Ingresando llave)
 - Eliminar (Ingresando llave)
 - Mostrar Información (Ingresando llave)
 - Mostrar Estructura de Datos (Graphviz)

A excepción de los viajes de los cuáles se podrán aplicar únicamente las siguientes operaciones:

- Crear
- Mostrar Estructura de Datos (Graphviz)

- Se podrán generar Reportes.

Ejemplo Estructura Completa



Entregables

- Ejecutable
- Código Fuente
- Manual de Usuario
- Manual Técnico

Todo en un archivo .rar con el nombre [EDD]Proyecto2_<carnet>.rar

Observaciones

- El proyecto se llevará a cabo en parejas, por lo que se deberá trabajar juntamente con GitHub, realizando commits.
- Cada commit deberá llevar la siguiente estructura:

Comentario, # Carne , Nombre

Cada estudiante deberá haber realizado como mínimo 10 commits.

- Lenguaje de Programación: Java
- Sistema Operativo: Elección del estudiante
- La aplicación será compilada y ejecutada al momento de la calificación
- Las estructuras de datos deben ser realizadas por el estudiante.
- Fecha de Entrega y Calificación: **Martes 30 de junio** antes de medianoche.
- Las gráficas deben mostrarse dentro de la aplicación, no buscarse en carpetas ajenas.
- Entrega en plataforma CLASSROOM.
- Copias tendrán nota de 0 puntos y serán reportadas al catedrático y a la escuela de sistemas.