Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructuras de Datos
Sección A
Ing. Edgar René Ornélyz
Aux. Esvin Armando González Monzón

Help Manager

Proyecto 2 - Administrador de albergues y centros de acopio

Objetivo general

Con la realización de este proyecto se busca que el estudiante utilice sus conocimientos de programación para diseñar e implementar estructuras de datos propias para propósitos específicos creando una solución de software con una arquitectura inspirada en la típica arquitectura cliente-servidor.

Objetivos específicos

Durante el desarrollo de este proyecto se requiere que el estudiante:

- Comprenda el funcionamiento e implemente las siguientes estructuras de datos
  - Árbol B
  - Lista como Diccionario de datos
  - Grafo dirigido
  - Tabla Hash
- Diseñe e implemente una arquitectura cliente-servidor para la comunicación de dos aplicaciones
- Diseñe e Implemente un sistema de cifrado y descifrado de textos por medio del algoritmo DES
- Aplique buenas prácticas de programación
- Construya un sistema sólido y estable

Descripción general

En el mes de septiembre de 2017, en transcurso de 12 días (7/sept - 19/sept) México fue afectado por dos terremotos de 8.2 y 7.1 grados en la escala de Richter. Edificios completos se desplomaron, se sufrieron muchas pérdidas, tanto materiales como humanas, el país quedó gravemente lastimado, pero en las catástrofes como éstas es donde sale lo mejor de los pueblos latinoamericanos. Donaciones, comedores gratuitos al aire libre, rescatistas buscando vida entre los escombros, familias compartiendo sus hogares con desconocidos y mil acciones altruistas más.

Entre tantas personas intentando ayudar, se llega a tener una saturación de mano de obra, sin mencionar a ciertas personas que siempre buscan sacar provecho de la desgracia ajena. Para evitar problemas adicionales a los ya inherentes de la tragedia cómo: determinar quienes aún no tienen un lugar donde pasar la noche o quienes aún no reciben una despensa para su sustento temporal o la gestión de equipo para rescate, entre muchas otras responsabilidades. Es necesario establecer una forma de administrar, de manera transparente y eficiente, todos los recursos para potenciar el impacto de la ayuda recolectada. Esta labor se podría confiar a papeles y tinta, pero sería tardado, ineficiente y por no mencionar poco transparente; es en este punto donde los sistemas de la información juegan un papel trascendental.

Como una iniciativa de la Facultad de Ingeniería, para acelerar la organización de las personas como respuesta de cualquier desastre natural se le ha solicitado a usted como futuro Ingeniero en Ciencias y Sistemas el diseño y la implementación de una solución de software para cumplir los requerimientos planteados en este documento. Dado que este proyecto se desarrolla para el curso de Estructuras de Datos se ha prescindido del uso de una base de datos, por lo que la correcta implementación de las distintas estructuras de datos es fundamental para el éxito del proyecto.

Desde un punto de vista muy general la solución de software debe cumplir con ciertas funciones, en donde cada una de ellas tendrá vinculada una estructura de datos cuyo comportamiento se tratará a detalle en las secciones subsiguientes de este documento, las funcionalidades son:

- Administrar la información de centros de acopio y albergues (grafo dirigido)
- Administrar los recursos de cada centro de acopio (tabla hash)

- Administrar el registro de personas en cada albergue (diccionario)
- Administrar las donaciones económicas recibidas (árbol B)
- Asegurar la integridad y seguridad de la información (cifrado/descifrado)
- Garantizar la conectividad de las distintas aplicaciones (REST, SOAP, sockets u otro)

Arquitectura

La arquitectura de la aplicación se compondrá de un servidor que mantendrá centralizada, protegida e íntegra toda la información y una aplicación cliente que consultará la información para crear, modificar o eliminar valores de manera local para luego persistir todos esos cambios en el servidor.

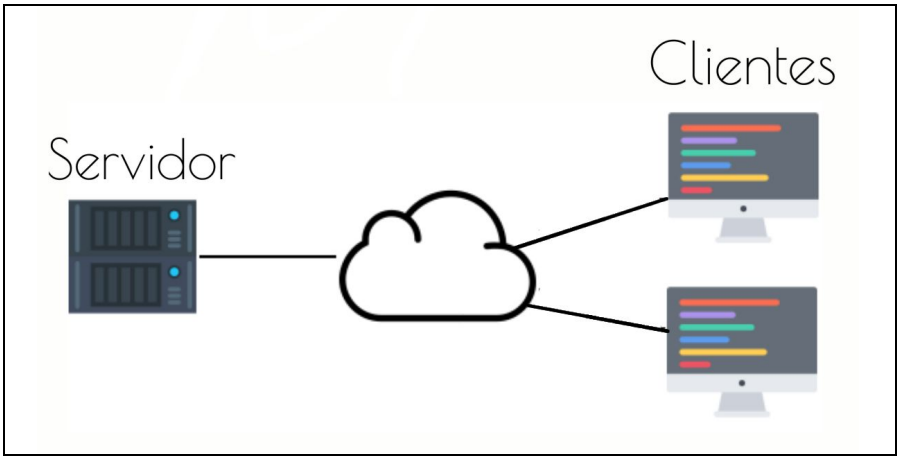

Figura 1: Diagrama más simple de la arquitectura a implementar

Servidor

El servidor será una aplicación web desarrollada en Java. Su función principal será administrar la información de forma centralizada y atender las peticiones de los clientes que se conectarán a él por medio de peticiones HTTP. En el servidor existirán dos estructuras de datos, un grafo dirigido para almacenar la información física tanto de los albergues como de los centros de acopio y un árbol B* que servirá para llevar un registro de las donaciones económicas.

Gestión de localidades (Grafo dirigido)

Esta estructura será implementada como una lista de nodos en donde cada nodo tendrá como atributo una lista que representará el conjunto de aristas que salen del nodo, cada arista tendrá un nodo destino y un multiplicador. El multiplicador será un número decimal entre 1 y 5 (incluídos los extremos) que servirá como un *handicap* para determinar el peso de cada arista según la siguiente ecuación:

$$P = D * m$$

Ecuación 1: Determina el peso de una arista

Donde:
P = peso del camino o arista
D = distancia entre nodos*
m = multiplicador

*calculada en base a latitud y longitud

La información que se requiere para cada albergue o centro de acopio se presenta en el siguiente diagrama de clases (en el diagrama no se muestran los métodos de cada clase, solamente los atributos a implementar). La primera letra del código representa si se trata de un albergue o un centro de acopio.

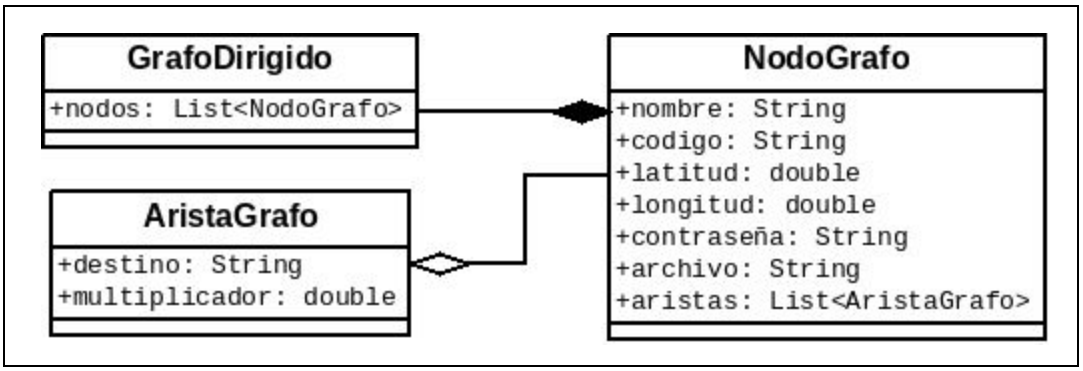

Figura 2: Diagrama de clases de los componentes del grafo dirigido

Cada nodo del grafo contiene información física de la instalación, como el nombre, la posición (como latitud y longitud) un código, una contraseña (para cuestiones de seguridad) y el nombre de un archivo json que contendrá la información particular de cada instalación. El archivo json contendrá información diferente según sea un albergue o un centro de acopio.

Sobre el grafo de albergues y centros de acopio (que de ahora en adelante serán llamados simplemente localidades) se pueden realizar las siguientes operaciones: Creación, Modificación, Eliminación y Definición de aristas. Para los albergues el código iniciará con la letra A, como por ejemplo A001, A1023; mientras que los códigos para los centros de acopio iniciarán con la letra C, p.e. C001, C2030.

Gestión de donaciones (Árbol B*)

Esta estructura tendrá la responsabilidad de gestionar las donaciones económicas de las diferentes personas que deseen realizar aportes para el beneficio de los damnificados. De cada donación interesa saber el documento de identificación del donante, sus nombres y apellidos, el número de voucher, el monto y la fecha de la donación. Cada una de estas donaciones se almacenará utilizando como llave el código de identificación del donante, si una misma persona hace varias donaciones éstas se almacenarán en una lista simplemente enlazada ordenada por la fecha de la donación.

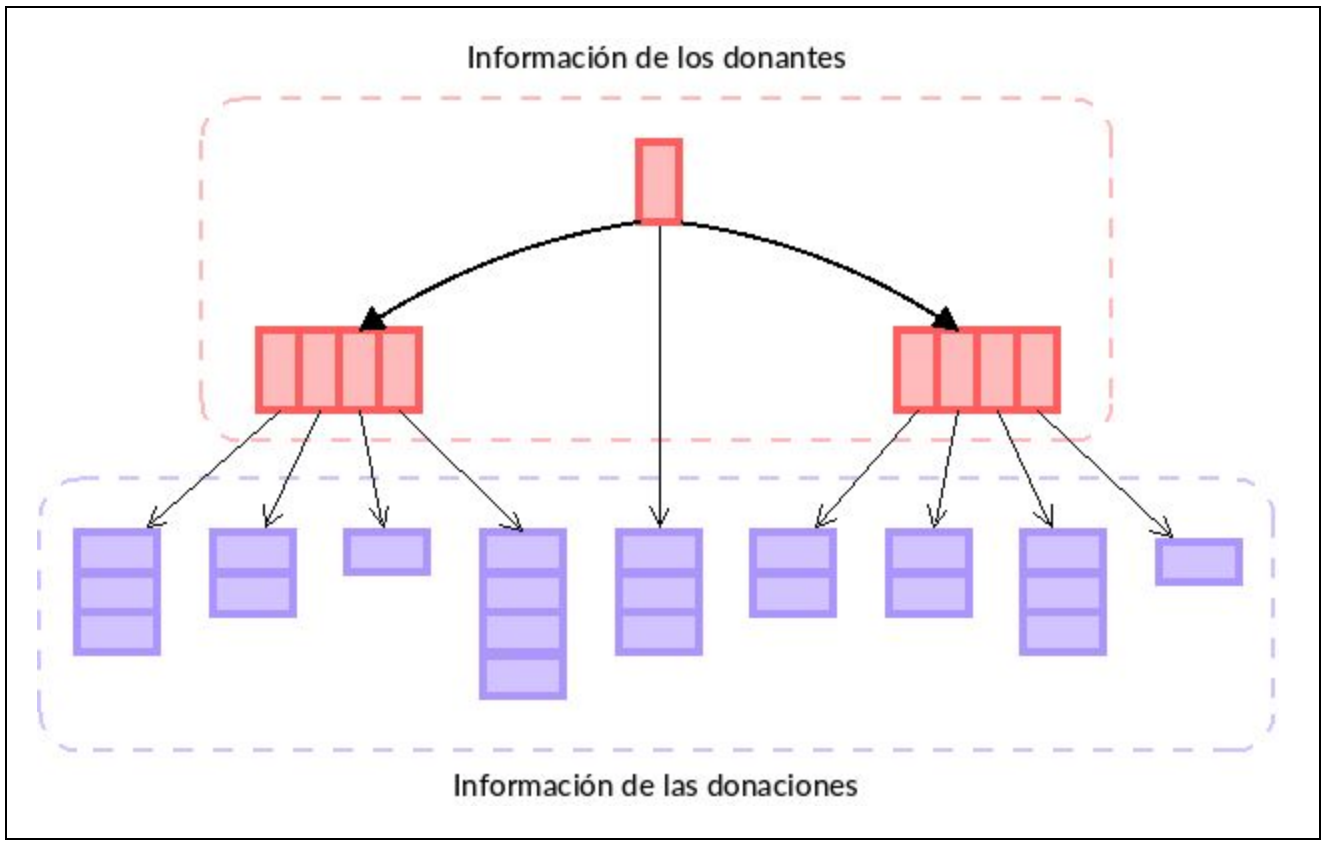

Figura 3: Árbol B* para el control de donantes y donaciones

Persistencia de la información (Encriptación)

El servidor podrá persistir la información de las donaciones y la información física de las localidades por medio de archivos csv, un archivo para almacenar la información de las donaciones y un archivo para la información de las localidades utilizando el siguiente formato.

Localidades:

```
codigo,nombre,password,latitud,longitud,archivo(,destino,multiplicador)*
codigo,nombre,password,latitud,longitud,archivo(,destino,multiplicador)*
```

Donaciones:

```
documento,nombres,apellidos,fecha,monto
documento,nombres,apellidos,fecha,monto
```

En estos archivos cada fila representa una localidad o una donación según el archivo que se consulte. El contenido de estos archivos deberá estar cifrado por algún algoritmo ideado por el estudiante, opciones

como DES o el cifrado César pueden ser fácilmente implementadas en cualquier lenguaje de programación. Se debe tener en cuenta que el algoritmo elegido debe ser de doble vía, es decir que se debe poder tanto cifrar como descifrar la información. El proceso de cifrado y descifrado debe ejecutarse de manera automática al detener o arrancar el servidor respectivamente.

Al arrancar el servidor se tiene que leer los archivos de donaciones y localidades, descifrar su contenido, cargar la información de las donaciones en el árbol B* y la información de las localidades en el grafo dirigido. Al momento de detener el servidor se deberá de tomar la información que se encuentre tanto en el grafo como en el árbol B* y escribirla como texto según el formato CSV indicado anteriormente, luego cifrar ese texto y escribirlo en el archivo que corresponda.

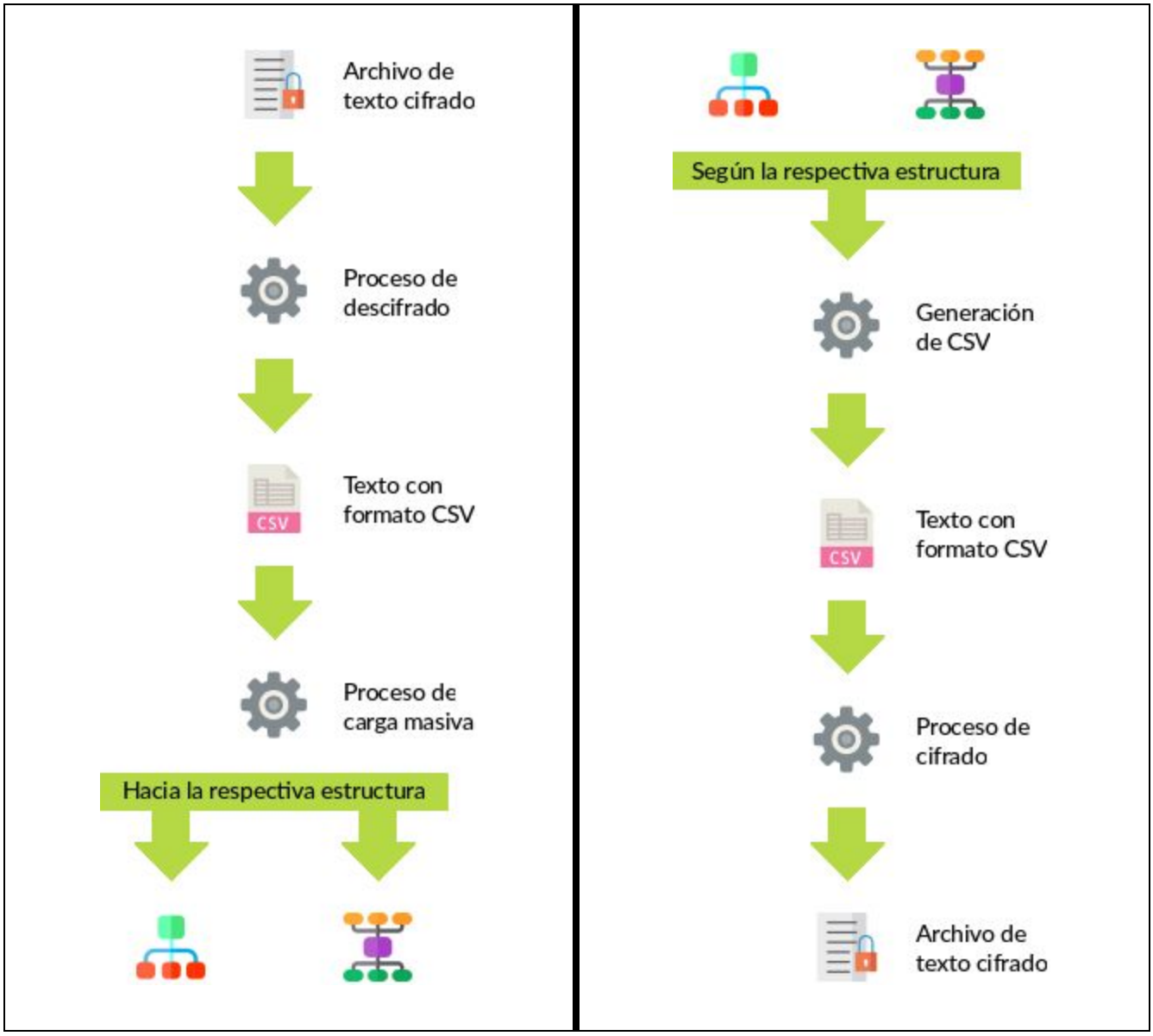

Figura 4: Explicación del proceso de arranque (izquierda) y paro (derecha) del servidor.

Recapitulando, hasta el momento se ha definido que el servidor almacenará dos estructuras de datos, una de ellas será un árbol B* para el manejo de donaciones, mientras que la más elaborada será un grafo dirigido que almacenará la información física de las distintas localidades. En el siguiente apartado se describen las funciones del API que se dispondrá para las aplicaciones cliente.

API para clientes

El servidor colocará a disposición de sus clientes un conjunto de funciones por medio de distintas peticiones de red, este conjunto de peticiones recibe el nombre de API (Application Programming Interface), esta API permitirá consultar la información de las localidades, guardar cambios sobre el contenido de las localidades y registrar donaciones. Las funciones que componen el API a desarrollar se listan a continuación.

`getLocalidad(String codigo, String password)`
Esta función buscará en el grafo dirigido un nodo que tenga el código y password enviados como paŕametros y devolverá el contenido del archivo JSON asociado a dicho nodo.

`freeLocalidad(String codigo, String password)`
Esta función buscará en el grafo dirigido un nodo que tenga el código y password enviados como paŕametros y marcará dicha localidad como libre y se perderán los cambios realizados de manera local en el cliente.

`saveLocalidad(String codigo, String password, String json)`
Esta función identifica al nodo con el código y password enviados como los primeros dos parámetros y reemplazará el contenido del archivo JSON asociado al nodo por el texto enviado en el tercer parámetro, esta función devolverá un mensaje de éxito o error según haya sido la ejecución de la misma.

`makeDonacion(String usuario, String fecha, Double monto)`
Esta función registrará una nueva donación en el árbol B* con los datos enviados como parámetros y devolverá un mensaje de éxito o error según haya sido la ejecución de la función.

`getDonaciones(String usuario)`
Esta función buscará todas las donaciones realizada por el usuario identificado por el código enviado como parámetro y devolverá la lista de donaciones (fecha y monto) realizadas como una lista en formato JSON.

`getGrafo()`
Esta función regresará la información de todas las localidades con el siguiente formato JSON. Además de este ejemplo de sintaxis, en este archivo se muestra un ejemplo de cómo quedaría el JSON de un grafo completo con 5 nodos y varias aristas entre estos.

```
[
	{
		"nombre":"LA ESPERANZA",
		"codigo":"A001",
		"lat":14.6555144,
		"lng":-90.5736757,
		"aristas":	[
			{ "destino":"A002","multiplicador":1.5 },
			... Más aristas separadas por coma
		]
	},
	... Más nodos separados por coma
]
```

Figura 5: Ejemplo de la sintaxis para la transferencia del grafo de localidades

En el servidor existen varios módulos, los mismo se listan a continuación y los mismos se acompañan con una imagen que detalla la arquitectura interna del servidor.

- Módulo de inicio y finalización del servidor
    - Cifrado/descifrado de archivos (únicamente a la información con formato CSV)
    - Carga masiva, tanto para el grafo dirigido como para el árbol B*
- Módulo de administración de las localidades
    - Creación, modificación y eliminación de nodos
    - Asignación de aristas
    - Creación de JSON para transferencia a los clientes
- Módulo del API
    - Inicio de sesión por localidad (código y password para obtener JSON de una localidad)
    - Modificación de archivos JSON por localidad
    - Creación y consulta de donaciones
- Módulo de reportes
    - Generación de imagen representativa del grafo
    - Generación de imagen representativa del árbol B*

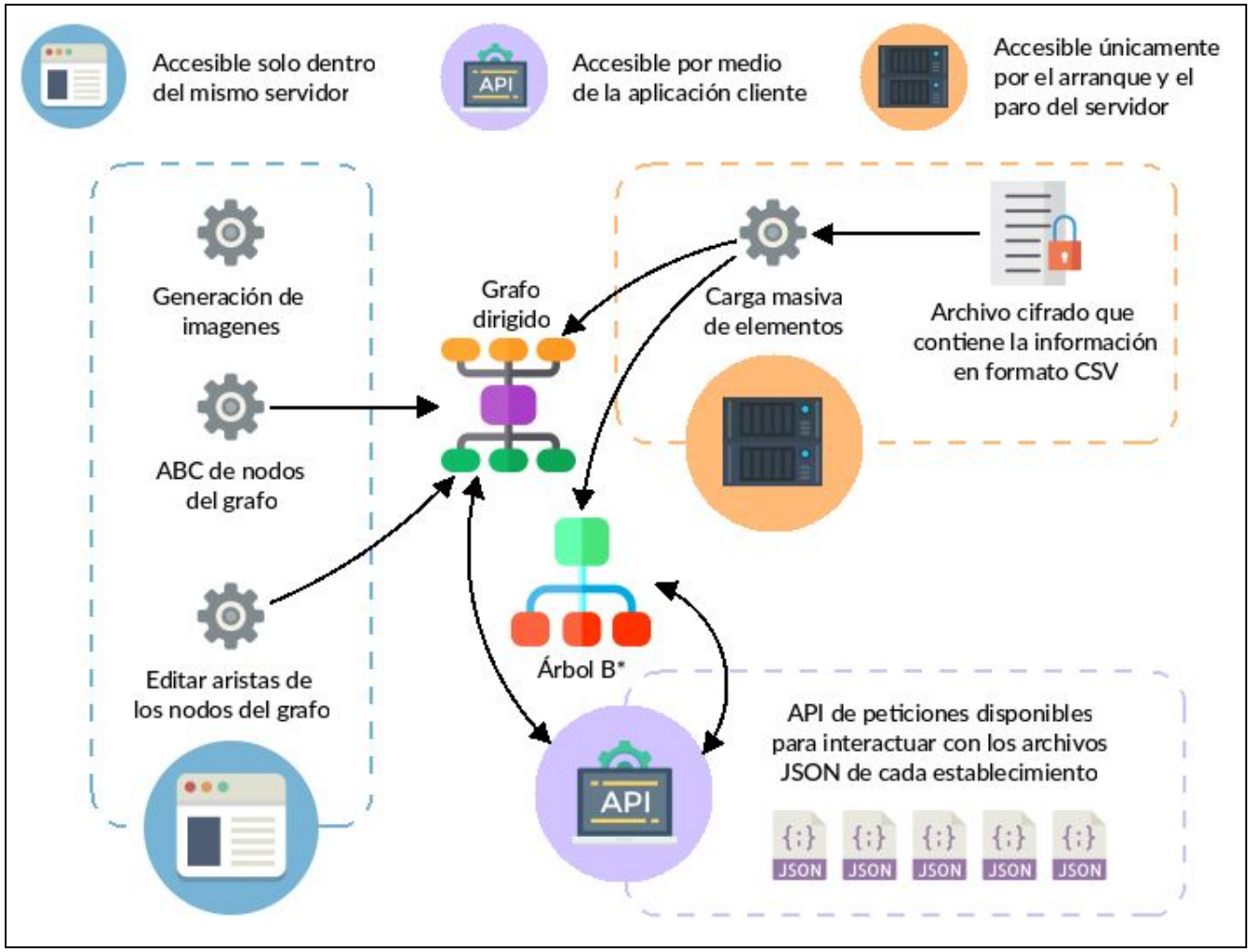

Figura 6: Funcionamiento interno del servidor, las funciones en celeste serán desarrolladas por el usuario dentro del mismo servidor, mientras que la sección en naranja contiene operaciones automáticas del servidor y por último la sección púrpura engloba la gestión de las peticiones disponibles a través del API para los clientes.

Cliente

Después de haber descrito los componentes y comportamiento del servidor para la solución de software propuesta, ahora es turno de hablar de la aplicación cliente. La función principal de esta aplicación cliente será la administración de recursos en los centros de acopio y el registro de personas en los albergues, para esto el flujo de la aplicación será como se ejemplifica en la siguiente imagen.

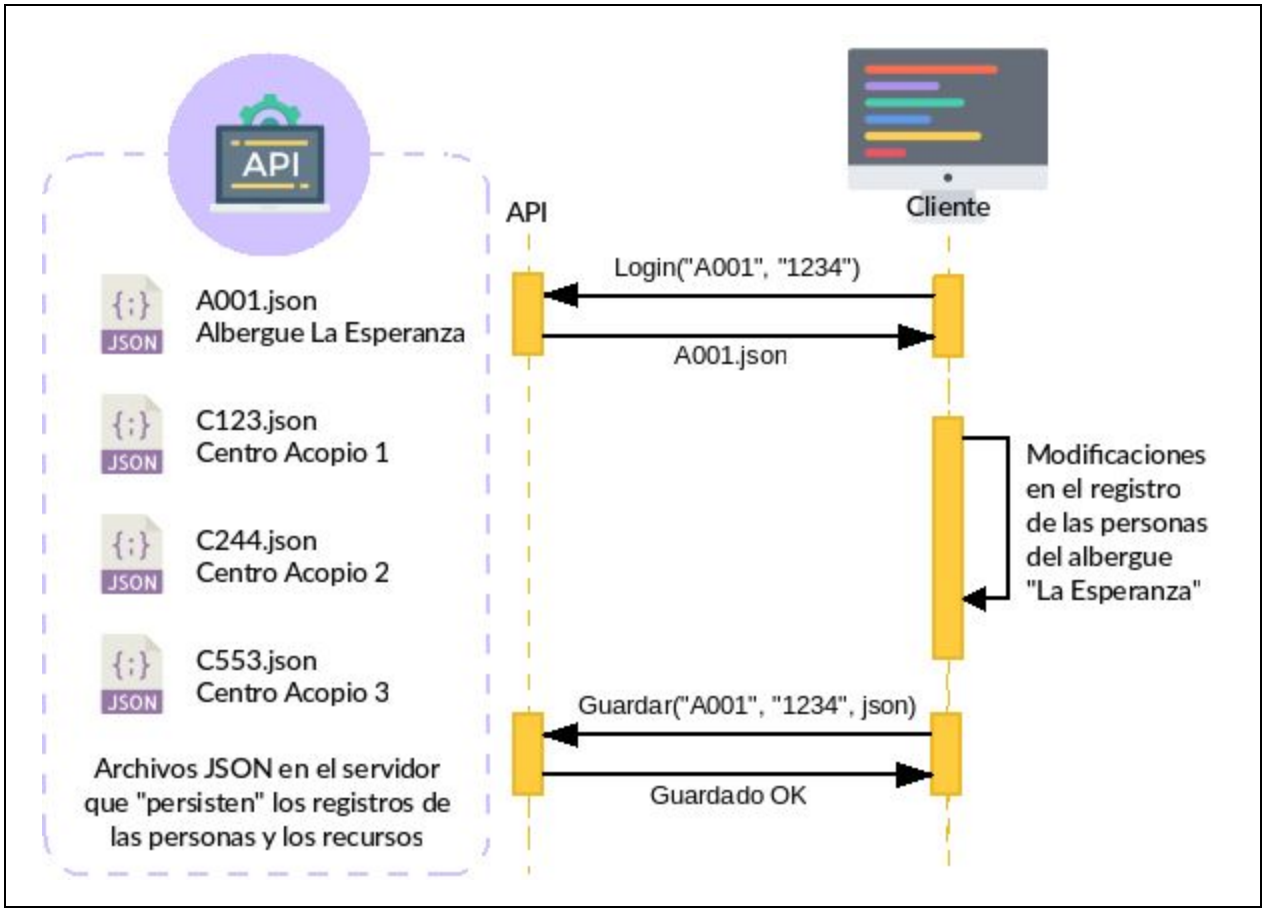

Figura 7: Diagrama de actividades de la aplicación cliente, los flujos alternativos deben ser controlados por medio de alertas que informen si ha ocurrido algún problema para que el usuario sepa qué está sucediendo en todo momento.

La aplicación cliente será una aplicación de escritorio desarrollada en C++, esta aplicación deberá implementar una manera de comunicarse con el servidor, puede ser por medio de sockets o peticiones http. De forma general el cliente contará con los siguientes módulos:

- Inicio de sesión
- Módulo de gestión de albergues (por medio de un Diccionario de Datos)
- Módulo de gestión de recursos por centro de acopio (por medio de una Tabla Hash)
- Módulo de donaciones (consumir los servicios de donaciones)
- Módulo de mapas

Inicio de sesión

Esto en sí no es un módulo, es más bien parte de los módulos de gestión de albergues y centros de acopio, para poder mantener la información íntegra, el servidor tendrá que llevar un control del estado de los archivos JSON para evitar

que se sobreescriba contenido o se pierdan actualizaciones ya que la forma de trabajar sobre el contenido de las localidades (albergues o centros de acopio) será el siguiente.

1. Se ingresa al módulo que se desea gestionar.
2. Se ingresa el código y contraseña de la localidad a gestionar.
3. Se hace la petición de "login" *getLocalidad*.
    a. El servidor busca el nodo que coincida con las credenciales.
    b. Si no encuentra ningún nodo, retorna un mensaje de error "Credenciales incorrectas". Regresa al inciso 2
    c. Se encuentra el nodo que representa la localidad solicitada.
4. El servidor verifica que nadie más haya pedido el acceso a la localidad en cuestión.
    a. Si alguien ya tiene el acceso, el servidor debe responder con un mensaje de error "Localidad ocupada". Regresa al inciso 2
    b. Si nadie tiene "ocupada" la localidad solicitada se le responde al cliente con el contenido del archivo JSON de la localidad que ha solicitado para su gestión y se marca la localidad como ocupada.
5. Se realizan las manipulaciones que se deseen sobre la estructura que representa el archivo JSON y luego se decide terminar con el trabajo.
    a. Si usuario decida guardar sus cambios locales hace una petición *saveLocalidad*. Regresa al inciso 1
    b. Si el usuario decide que sus cambios locales no están bien, los descarta y libera la localidad haciendo una petición *freeLocalidad*. Regresa al inciso 1

Gestión de albergues (Diccionario de datos)

Este módulo trabajará un diccionario de datos (estructura de pares de clave-valor) para llevar el control de las personas que están habitando un albergue, el diccionario podrá almacenar el registro de personas individuales (utilizando como clave el número de su documento de identificación) o familias enteras, utilizando como llave el documento de identificación del encargado de la familia. Cada nodo del diccionario de datos almacenará un árbol binario de búsqueda para los miembros de una familia o un puntero para una persona individual. De cada persona se desea conocer su número de documento de identificación, sus nombres y apellidos, su edad y su género, si los integrantes son niños su identificador será su registro de nacimiento.

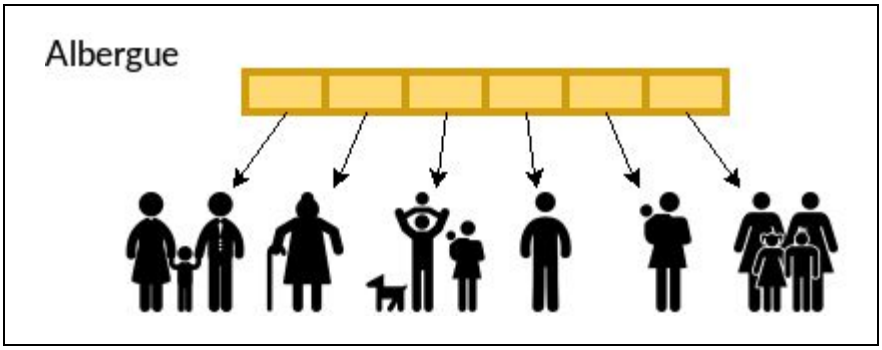

Figura 8: Diccionario de datos para llevar el registro de las personas alojadas en el albergue

Sobre el diccionario de datos se pueden realizar las siguientes operaciones:

- Registro de una familia
- Registro de una persona individual
- Modificación de registros
- Eliminación de registros
- Lista en pantalla de todas las personas registradas
- Reporte gráfico del diccionario de datos, utilizar de referencia la figura 8.

Gestión de centros de acopio (Tabla Hash)

Este módulo trabajará una tabla hash. De cada producto interesa conocer, su categoría (comida, medicina, ropa, utensilios, aseo, herramienta, etc), su nombre y las unidades que se han recibido como donación. En la tabla se calculará el hash de cada recurso sobre su categoría y las colisiones se resolverán por medio del nombre del producto, si un producto se encuentra "repetido", es decir, su categoría y nombre ya existe, se sumarán las unidades del nuevo donativo a las unidades que ya están dentro del registro de la tabla hash.

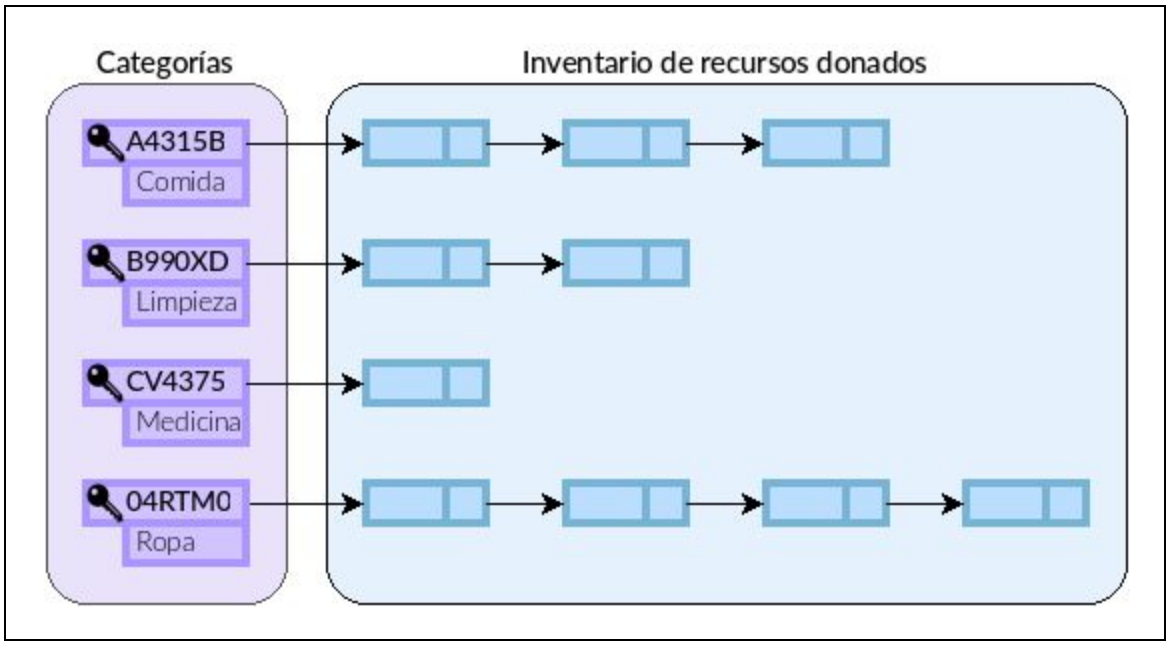

Figura 9: Representación de la tabla hash como estructura para controlar los recursos donados

Los recursos deben ser liberados para su entrega en los lugares necesitados, por lo que se ha agregado la función de entrega de productos en donde se mostrará una tabla con todos los productos existentes y el inventario del que se dispone. Para su entrega, el usuario ingresará en la tabla la cantidad de producto que se entregará, al tener lista la entrega existirá una opción para hacer "efectiva la entrega", esto reducirá de cada registro dentro de la tabla hash la cantidad de unidades que has sido despachadas, si el contador llegase a 0, el registro de la tabla hash tendrá que eliminarse.

Sobre la tabla hash de recursos se pueden realizar las siguientes operaciones:

- Registro de recursos
- Entrega de recursos
- Modificación
- Eliminación
- Lista en pantalla de todos los recursos
- Reporte gráfico de la tabla hash, utilizar de referencia la figura 9.

Control de donaciones

Este módulo se encargará de realizar peticiones para el manejo de las donaciones, para ello no es necesario utilizar el mecanismo de login, cualquiera podrá realizar las siguientes operaciones.

- Crear una nueva donación (se asume que quien registra la donación tiene acceso al voucher de la misma y a la información del donante, pueden haber donantes anónimos, para ellos se utilizará el código de identificación '000').
- Consultar las donaciones hechas por persona según su documento de identificación.

Mapas (Algoritmo de Djkstra)

Para acceder a este módulo no es necesario utilizar el mecanismo de inicio de sesión, este módulo realizará una petición a la función getGrafo y al obtener el JSON se realizará la transformación de éste en un grafo dirigido con caminos ponderados calculando el peso de los caminos según la Ecuación 1.

Vista general del grafo

- El grafo tendrá que ser representado utilizando la librería de mapas de Google dentro de la aplicación cliente. En el mapa se tendrá que utilizar un marcador o color diferente según sea un albergue o un centro de acopio y se dibujarán los caminos entre las localidades como líneas rectas.

Determinación del camino más corto entre dos puntos del grafo

- En la misma pantalla donde se muestre el mapa (fuera del webview que se usará para representar el mapa por supuesto) se tendrán dos comboBox, ellos tendrán almacenados todos los códigos de las localidades y serán los que representan el punto de origen y el destino de un camino. Al presionar sobre el botón de "Hallar el camino más corto" se deberá pintar de un color diferente la ruta que represente el camino más corto entre los puntos elegidos como origen y destino por medio del algoritmo de Djkstra.

Consideraciones finales

Para el desarrollo de este proyecto se deben de tomar en cuenta los siguientes aspectos.

- El proyecto debe ser realizado de manera individual.
- Para el servidor:
  - El lenguaje a utilizar para el servidor será Java.
  - Debe ser una aplicación web.
  - El IDE a utilizar es libre (se recomiendo NetBeans).
  - El servidor a utilizar es libre (se recomienda GlassFish).
- Para la aplicación cliente:
  - El lenguaje a utilizar será C++.
  - Debe ser una aplicación de escritorio.
  - El IDE a utilizar es libre (se recomienda QtCreator).
- La comunicación debe ser por medio de peticiones HTTP.
- La entrega será por medio de Dropbox, como una solicitud de archivo publicada por el auxiliar el día de entrega, no se recibirán proyectos fuera de tiempo o enviados por correo electrónico.
- Se debe entregar para el día 18 de octubre el diagrama de clases del proyecto completo. Esto con el fin de realizar correcciones oportunas sobre la interpretación que el estudiante pueda darle al proyecto.
- Los entregables finales de este proyecto consisten en:
  - Código fuente (sin dependencias rotas de ambos proyectos)
  - Ejecutables (de cada aplicación)
- Para tener derecho a calificación es necesario que la solución creada cuente con:
  - Cifrado/descifrado de archivos CSV
  - Grafo dirigido (tanto en cliente como en el servidor)
  - Tabla Hash
  - Árbol B*
  - Implementación de mapas y el algoritmo del camino más corto
- Todas las estructuras de datos deben ser desarrolladas por el estudiante.
- Fecha de entrega: Miércoles 15 de noviembre de 2017 antes de las 23:59 horas.
- Copias totales o parciales serán penalizadas con nota de 0 puntos y reportadas ante escuela de ciencias y sistemas.