



Bumasoft 2.0

15.05.2021



Sean Quintero Ordoñez
A00359408



Paola Andrea Osorio Holguín
A00359822



Bryan Alexander Guapacha Florez
A00359388



John Kennedy Landazuri Sandoval
A00347355



ICESI University

Contexto del problema


Bumasoft es un software que se ha desarrollado pensando en el manejo de las unidades residenciales con parqueaderos comunitarios, el software consiste en poder brindar facilidad al personal que esté en el área encargada de manejar todo lo relacionado con los apartamentos, parqueaderos, propietarios, residentes y visitantes de la unidad residencial.



Enunciado del Proyecto

Un grupo de ingenieros de sistemas que viven en unidades residenciales, observaron la gran necesidad que tienen las porterías de sus conjuntos para llevar a cabo los registros de los apartamentos, propietarios, residentes y visitantes que ingresan y salen de la unidad, por lo cual decidieron desarrollar un software que permita el registro del conjunto residencial, con su respectivo, nombre, dirección, teléfono, NIT, nombre del administrador, identificación de administrador, cantidad de apartamentos, cantidad de torres, cantidad de apartamentos por piso, cantidad de pisos por torres y cantidad de parqueaderos, adicionalmente, debe poder registrar apartamentos pidiendo la matrícula inmobiliaria, número de apartamento, torre, identificación del propietario, estado (habitado ó sin habitar), (en caso de que esté habitado, se debe saber la cantidad de habitantes) y tipo de residente (propietario o arrendatario), El programa también debe permitir el registro de propietarios, donde se debe ingresar nombre completo, número de identificación, tipo de documento (CC o CE), número de contacto y correo electrónico, así mismo debe poder registrar a los residentes; cada residente tendrá un nombre, un número de cédula, tipo de documento (CC, CE ó TI), número de apartamento, torre, número de teléfono y en caso de que tenga vehículo, se le debe pedir la cantidad de vehículos, tipo de vehículo (carro ó moto), marca, modelo, cilindraje, color, placa, tipo de carrocería(automóvil o camioneta), después del registro de vehículos, el programa debe llevar la información del ingreso y la salida de cada vehículo mediante la placa y por último debe poder registrar a los visitantes, ingresando nombre completo, número de cédula, tipo de identificación, número de apartamento, numero de torre y fecha con hora de ingreso actual.

Por otra parte, el programa debe permitir buscar un residente por número de identificación, los datos de un vehículo por número de placa, un propietario por número de identificación, un apartamento por número y torre, adicionalmente se



debe poder consultar cuantos parqueaderos quedan disponibles y cuantos parqueaderos están ocupados, asimismo se deben poder guardar toda su información a través de la serialización de sus objetos en archivos, esta serialización debe ser visible para el usuario, es decir que cuando se actualice algún tipo de información, esta también se actualizará en los archivos serializados, se debe poder generar archivos csv de los residentes con sus respectivos vehículos y de los apartamentos con sus respectivos propietarios, se debe poder leer y escribir en un archivo de texto plano la información de los apartamentos y los respectivos residentes.

Adicional a esto, el programa debe estar en la capacidad de indicar a los visitantes que llegan a pie, el camino más corto para dirigirse a la torre donde se encuentra el apartamento a visitar. También debe estar en la capacidad de mostrar a los residentes que llegan en vehículo, el estacionamiento más cercano a la torre donde se encuentra su apartamento; con el fin de que a la hora de salir del conjunto residencial, desplazarse hacia su auto sea más cómodo.

Para finalizar, el sistema debe permitir actualizar el nombre de la unidad, teléfono, y datos del administrador, también se debe poder actualizar los datos del propietario y residentes con su número de identificación, de los vehículos con su número de placa y de los apartamentos con su número de apartamento y torre (no se debe poder cambiar la matrícula inmobiliaria, numero de apartamento ó torre).

Fase 1: IDENTIFICACIÓN DEL PROBLEMA

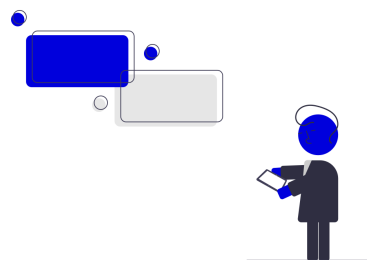


Definición del problema

Se necesita un programa que pueda brindar facilidad al personal que esté en el área encargada de manejar todo lo relacionado con los apartamentos, parqueaderos, propietarios, residentes y visitantes de la unidad residencial.

Identificación de síntomas

- Las unidades residenciales requieren la implementación de una herramienta para el manejo de información de gran tamaño que permita ingresar, eliminar o modificar



información y realizar consultas de cada apartamento.

- Encontrar la forma más eficiente para relacionar a un arrendatario con un propietario (Dijkstra).
 - La solución debe brindar una forma rápida y eficiente de búsqueda.
 - La solución debe permitir la búsqueda por torre y número de apartamento (DFS).
 - La solución debe poder leer y escribir datos por interfaz gráfica y manejar el almacenamiento por archivos.csv.
-



FASE 2

RECOPILACIÓN DE LA INFORMACIÓN NECESARIA y ESPECIFICACIÓN DE REQUERIMIENTOS

Requerimientos Funcionales


RF1: Ingresar datos de la Unidad Residencial

El programa debe permitir ingresar por única vez los datos del conjunto residencial, solicitando el nombre de la unidad, dirección, teléfono, NIT, nombre del administrador, identificación de administrador y número de contacto, cantidad de apartamentos, número de torres, cantidad de torres, cantidad de apartamentos por piso, cantidad de pisos por torres y cantidad de parqueadero. Después de ingresar esta información, se debe mostrar un mensaje donde le informe al usuario, que los datos se han registrado de manera exitosa.

RF2: Registrar Propietarios

El software debe permitir registrar los datos de los propietarios de los apartamentos de la unidad residencial, se debe ingresar nombre completo, número de identificación, tipo de documento (CC o CE), número de contacto y correo electrónico. Posterior a ello debe informar al usuario que el registro se realizó de manera exitosa.

RF3: Registrar Apartamentos



El sistema debe poder registrar los apartamentos del conjunto residencial, este debe ingresar el número de la matrícula inmobiliaria, número de apartamento, torre, identificación del propietario, estado (habitado ó sin habitar), (en caso de que esté habitado, se debe saber la cantidad de habitantes) y tipo de residente (propietario o arrendatario). por consiguiente, este programa debe generar un aviso al usuario de que el registro se realizó de manera exitosa.

RF4: Registrar Residentes

El programa registra a los residentes que habitan en cada apartamento del conjunto residencial; se debe ingresar la información de cada habitante, como los nombres, los números de identificación, tipo de documento (CC, CE ó TI), número de apartamento, torre y números de teléfonos. finalmente el sistema deberá informar al usuario que el registro se realizó exitosamente.

RF5: Registrar Vehículos:

Si el residente requiere el registro de un vehículo, el sistema debe permitir ingresar los datos del mismo, solicitando la siguiente información: la cantidad de vehículos, el tipo de vehículo(carro o moto), marca, modelo, cilindraje, color, placa, tipo de carrocería(automóvil o camioneta). Posterior a esto, el programa deberá manifestar al usuario que el registro fue exitoso.

RF6: Registrar Visitantes:

El sistema debe permitir registrar a los visitantes que asistan a la unidad residencial, se ingresaran los datos de los mismos, como el nombre completo, número de cédula, tipo de identificación, número de apartamento, número de torre y fecha con hora de ingreso actual. Después, el programa deberá enunciar al usuario el éxito del registro.


RF7: Buscar Apartamento

El programa debe permitir buscar un apartamento con su número de apartamento y torre, en consecuencia de ello, se debe mostrar toda la información que se encuentre registrada en el mismo.

RF8: Buscar Residente

El programa debe permitir buscar un residente con su número de identificación, en consecuencia de ello, se debe mostrar toda la información que pertenezca al residente asociado al número de identificación ingresada por el usuario.

RF9: Buscar Vehículo



El programa debe permitir buscar los datos de un vehículo por número de placa. después se debe arrojar la información asociada a este vehículo.

RF10: Buscar Propietario

El programa debe permitir buscar un propietario con su número de identificación, en consecuencia de ello, se debe mostrar toda la información que pertenezca al propietario asociado al número de identificación ingresada por el usuario.

RF11: Registro ingreso de Vehículo

El programa debe poder llevar el registro de vehículos que ingresan y salen de la unidad residencial, ingresando el número de placa. Cada que un vehículo ingrese, se debe ocupar un parqueadero y cuando salga se debe actualizar esta información de manera automática para poder dejarlo disponible.

RF12: Consultar Parqueadero

El programa debe permitir la consulta de cuantos parqueaderos quedan disponibles y cuantos parqueaderos están ocupados seleccionando la opción de consultar parqueaderos. En efecto se le debe mostrar al usuario la disponibilidad de los mismos.

RF13: Generar archivos csv de Residentes

El programa debe poder leer y escribir los archivos csv de los residentes con sus respectivos vehículos, a través de la serialización de los mismos. Estos deben estar ordenados por nombre de forma ascendente.

RF14: Generar archivos txt de Apartamentos

El programa debe poder leer y escribir los archivos txt de los apartamentos con sus respectivos propietarios e información, a través de un archivo de texto plano. Estos deben estar ordenados por número de apartamento y torre de forma ascendente.

RF15: Generar archivos txt de residentes

El programa debe poder leer y escribir los archivos txt de los residentes con sus respectivos propietarios e información, a través de un archivo de texto plano. Estos deben estar ordenados por número de identificación de forma ascendente.

RF16: Actualizar Datos Unidad Residencial

El sistema debe permitir actualizar los datos de la unidad residencial, se realiza la modificación de los siguientes datos: nombre de la unidad, teléfono y datos del administrador. en consecuencia, el programa debe generar un aviso al usuario de datos actualizados exitosamente.



RF17: Actualizar Datos de Propietarios

El sistema debe permitir actualizar los datos de los propietarios ingresando el número de documento, después de verificar que el propietario existe, debe permitir actualizar la siguiente información: nombre completo, número de identificación, tipo de documento (CC o CE), número de contacto y correo electrónico. en consecuencia, el programa debe generar un aviso al usuario que los datos han sido actualizados exitosamente.

RF18: Actualizar Datos de Residentes

El sistema debe permitir actualizar los datos de los residentes, registrando el número de identificación de los mismos. en consecuencia, el programa debe generar un aviso al usuario de que los datos fueron actualizados exitosamente.

RF19: Actualizar Datos de Vehículos

El sistema debe permitir actualizar los datos de un vehículo, ingresando la placa, después de verificar que el vehículo existe, debe permitir actualizar la siguiente información: tipo de vehículo (carro o moto), marca, modelo, cilindraje, color, placa, tipo de carrocería (automóvil o camioneta). En consecuencia, el programa debe generar un aviso al usuario que los datos han sido actualizados exitosamente.

RF20: Actualizar Datos de Apartamentos

El sistema debe permitir actualizar los datos del apartamento tales como el estado (habitado ó sin habitar), (en caso de que esté habitado, se debe saber la cantidad de ocupantes) y el habitante que reside en ellos (es decir, si es propietario o arrendatario), se debe modificar con el número de apartamento y torre (no se debe poder cambiar la matrícula inmobiliaria, número de apartamento ó torre). Por consiguiente el programa debe generar un aviso al usuario de que los datos han sido actualizados exitosamente.

RF21: Generar camino más cercano a la torre a visitar.

El sistema debe estar en la capacidad de dar a conocer a los visitantes, el camino más conveniente y de menor trayectoria para desplazarse a la torre en la cual se encuentra el apartamento a visitar. Las entradas son la torre y el apartamento a visitar y como resultado obtenemos el camino de menor trayectoria desde la entrada al conjunto residencial hasta la torre indicada por el visitante.

RF21: Indicar el parqueadero más cercano a la torre donde se reside.

El sistema debe estar en la capacidad de indicar el slot de parqueo más cercano a la torre donde reside un habitante del conjunto residencial. Como entrada

tendríamos el ID del cliente, como salida estaría el slot de parqueo más cercano a su residencia.

Especificación de limitantes

- Las operaciones de registrar, eliminar, modificar, consultar deben ser eficientes.
- Se debe implementar los algoritmos de grafos como: Recorridos sobre Grafos (BFS, DFS), Caminos de Peso Mínimo (Dijkstra, Floyd-Warshall), Árbol de Recubrimiento Mínimo -MST- (Prim, Kruskal).
- La solución debe ser basada en dos tipos de grafos.
- Se debe usar interfaz gráfica

Definiciones y ejemplos encontrados

Hemos buscado distinta información sobre software que realiza funciones parecidas:

-- **EDIFITO** → Desde el cálculo de los gastos de administración, informes contables, movimientos bancarios, registro de visitas y correspondencia, encuestas, videoconferencias, votaciones, mensajería, seguridad y más!... de manera online y desde un mismo lugar con el software de administración Edifito.



https://www.edifito.co/?gclid=CjwKCAjwIuiEBhBzEiwAO9B_HaOfbb68HwRdgjFh9nOAbFjygWJlJuCXADcNQyW9b9W3QN8Y7RyjRoCNGAQAvD_BwE

-- **CRM** → Tenemos más beneficios que permiten al administrador no destinar tiempo a la implementación, soporte y otras tareas de Kipo. Somos una herramienta, no una carga más. Nos encargamos de toda la implementación, soporte técnico y solución a dudas, nuevas funcionalidades sin costo, protección de Datos Personales, nos enfocamos en la privacidad, escuchamos tus sugerencias.

<https://kipo.app/>


-- **PROPIEDATA** → Optimiza el tiempo y automatiza las actividades diarias, enfocando la atención en el bienestar de los residentes y la copropiedad.

<https://www.propiedata.com/>

Como definiciones tenemos

1. Vértice

Son puntos o nodos con los que están conformados los grafos. Llamaremos grado



de un vértice al número de aristas de las que es extremo. Se le dice vértice “par” o “impar” según sea su grado.

2. Arista

Una arista es una relación entre dos vértices de un grafo.

3. Grafo

Conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

4. Caminos

En Teoría de Grafos, se llama camino a una secuencia de vértices dentro de un grafo tal que exista una arista entre cada vértice y el siguiente. Se dice que dos vértices están conectados si existe un camino que vaya de uno a otro, de lo contrario estarán desconectados. Dos vértices pueden estar conectados por varios caminos. El número de aristas dentro de un camino es su longitud

5. Dijkstra


Algoritmo de Dijkstra. También llamado algoritmo de caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

6. Deep first search

Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

7. Breadth first search

En Ciencias de la Computación, Búsqueda en anchura (en inglés BFS – Breadth First Search) es un algoritmo de búsqueda no informada utilizado para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación, para cada uno



de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

8. Algoritmo de Floyd-Warshall

En informática, el algoritmo de Floyd-Warshall, descrito en 1959 por Bernard Roy, es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución.

9. Árbol de recubrimiento mínimo

Dado un grafo conexo y no dirigido, un árbol recubridor, árbol de cobertura o árbol de expansión de ese grafo es un subgrafo que tiene que ser un árbol y contener todos los vértices del grafo inicial. Cada arista tiene asignado un peso proporcional entre ellos, que es un número representativo de algún objeto, distancia, etc.; y se usa para asignar un peso total al árbol recubridor mínimo computando la suma de todos los pesos de las aristas del árbol en cuestión. Un árbol recubridor mínimo o un árbol de expansión mínimo es un árbol recubridor que pesa menos o igual que todos los otros árboles recubridores. Todo grafo tiene un bosque recubridor mínimo.

10. Algoritmo de Prim

El algoritmo de Prim es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

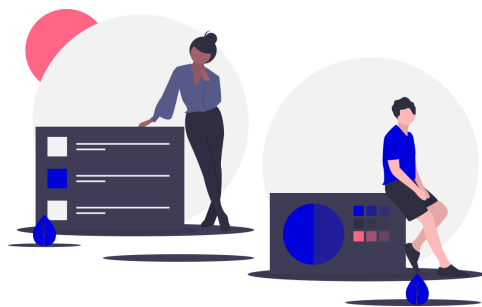
En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo.

11. Algoritmo de Kruskal

El algoritmo de Kruskal es un algoritmo de la teoría de grafos para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa).

12. Grado del vértice

Es el número de aristas que inciden sobre un vértice.



FASE 3

BÚSQUEDA DE SOLUCIONES CREATIVAS

Para la generación de ideas se realizó una revisión secuencial a conceptos previamente adquiridos relacionados con las necesidades del problema anteriormente expuesto. Partiendo de lo anterior, las posibles soluciones encontradas se exponen a continuación:

Implementación estructura de datos

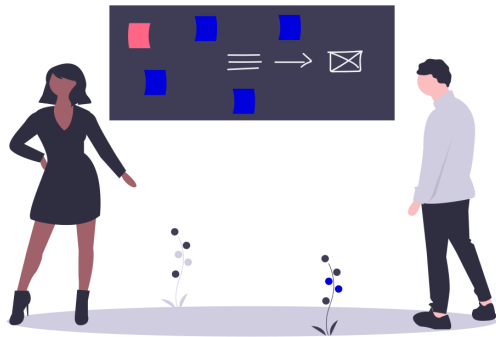
- a. Grafo ponderado
- b. Grafo no ponderado
- c. LinkedList
- d. Árbol n-ario

Tipo de estructura de datos

- a. Grafo dirigido
- b. Grafo no dirigido
- c. Multigrafo dirigido
- d. Multígrafo no dirigido
- e. Ninguno de los anteriores (en caso de que no se elijan los grafos, si no la opción c o d)

Forma de implementación de estructura de datos

- a. Matriz de adyacencias
- b. Lista de adyacencias
- c. Ninguna de las anteriores (en caso de que no se elijan los grafos, si no la opción c o d)



FASE 4

TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES

En esta fase hemos descartado las peores alternativas que no brindan una solución adecuada a los requerimientos, tales como: LinkedList, árbol n-ario, grado dirigido y multigrafo dirigido.

Implementación estructura de datos

Analizando a profundidad lo requerido, las estructuras tales como LinkedList y árbol n-ario no son las más eficientes ni nos permiten resolver los requerimientos que se piden en el enunciado.

Tipo de estructura de datos

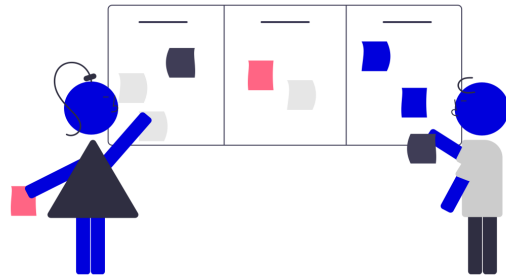
En este caso, no seleccionamos la opción ninguno de las anteriores, dado que la estructura LinkedList y árbol n-ario se descartaron. Se descarta además el grafo dirigido y multigrafo dirigido dado que el tipo de grafo que mejor se acomoda a nuestros requerimientos son los grafos no dirigidos.

Forma de implementación estructura de datos

En este caso, no seleccionamos la opción ninguno de las anteriores dado que la estructura LinkedList y árbol n-ario se descartaron. Sin embargo, la opción a y b siguen siendo muy factibles por lo que no se descarta ninguna de las dos.

FASE 5

EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN



Criterios implementación de estructura de datos

- Permita asignar los tipos de relación entre los seguidores del usuario
- Permita encontrar el camino más conveniente a el usuario para conocer a una persona (esto se realiza por medio de los tipos de relación)

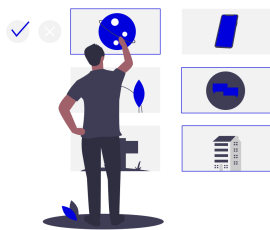
Criterios tipo de estructura de datos

Permita modelar de la mejor manera el programa por medio de las funciones seguir (tanto del usuario hacía del seguidor, como del seguidor hacia el usuario, es decir que ambos pueden seguirse mutuamente)

Forma de implementación de estructuras de datos

No tiene criterios, dado que se decide realizar la implementación de ambos tipos de grafo.

Selección de alternativas de implementación de estructuras de datos



	Criterio a	Criterio b	Total
Alternativa a	5	5	10
Alternativa b	0	0	0

Dado que obtiene el mayor puntaje la alternativa a, descartamos la alternativa b.

Selección de alternativas de tipo de estructuras de datos

	Criterio a	Total
Alternativa b	2	2
Alternativa d	5	5

Dado que obtiene el mayor puntaje la alternativa b, descartamos la alternativa a.

