Universidad de San Carlos de Guatemala

Laboratorio de Estructura de Datos

Sección A

**MANUAL TECNICO**

Esdras Eliab Bautista Guerra

202100301

* Clases utilizadas:

Texto

Descripción generada automáticamente

* Lista Enlazada .h y .cpp:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En esta clase es donde se almacena la información del usuario, todos sus datos necesarios para crear el usuario, así mismo se crea como instancia lo que son pila de solicitudes, lista de solicitudes, lista de publicaciones, lista de amigos, matriz de amigos y lista circular de publicaciones, para así podes administrar de una mejora manera cada lista o pila por usuario.

Del lado de la clase de lista enlazada tenemos el buscar nombre por correo que nos servirá para obtener el nodo de cada usuario y así poder usar las instancias anteriores por usuario, así mismo otras validaciones.

Texto

Descripción generada automáticamente

En la lista enlazada.cpp procedemos a desarrollar todos los métodos que creamos anteriormente, podemos ver que inicializamos para cada usuario una pila y las listas como nuevas y las creamos.

Así mismo creamos el destructor y los getters y setters correspondientes.

Y hacemos lo mismo para la lista enlazada:

Texto

Descripción generada automáticamente

* PilaSoli.h y cpp

Texto

Descripción generada automáticamente

Este seria nuestro nodo pila donde almacenaremos únicamente el correo del usuario y tendremos el puntero siguiente.

Texto

Descripción generada automáticamente

Y este seria la pila de solicitudes, donde se Irán almacenando, tenemos los atributos necesarios y los métodos.

Texto

Descripción generada automáticamente

En la parte del .cpp inicializamos todo y creamos los getter y setter del nodo

Y en el caso de la pila le damos forma a los métodos

Texto

Descripción generada automáticamente

El método push, añadira nuestro elemento a la cima de la pila, nuestro método peek devolverá el elemento en la cima de la pila, nuestro elemento pop eliminará nuestro elemento en el tope de la pila.

* Nodo publicación circular:

Texto

Descripción generada automáticamente

Esta lista es una lista circular doblemente enlazada, la cual unirá cada publicación del usuario y de sus amistades para así poder visualizarla y navegar entre dichas publicaciones.

Texto

Descripción generada automáticamente

La lista circular, encargada de agregar nodos y recorrerlos en una lista de nodos.

Texto

Descripción generada automáticamente

Procedemos a inicializar cada atributo y en el constructor le asignamos cada valor, así mismo creamos los getter y setters.

Posteriormente desarrollamos los métodos que usaremos para la lista circular, para agregar los datos, ver la lista, entre otras cosas:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

En ver lista de publicaciones, recorreremos nuestra lista y con ayuda del código ASCII podemos avanzar y retroceder a las publicación con las flechas de nuestro teclado.

* Nodo publicación (lista doblemente enlazada):

Este nodo básicamente nos sirve para almacenar las publicaciones de los usuarios individualmente:

Texto

Descripción generada automáticamente

Agregamos los atributos necesarios y los métodos que son el constructor y los getter y setters.

Texto

Descripción generada automáticamente

Igualmente, para la lista Publicaciones.

En este hacemos lo mismo que en el circular(el anterior) con excepción de que aquí es una lista doblemente enlazada y no circular doblemente enlazada.

* Matriz

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Nuestro nodo matriz tendrá 4 apuntadores, arriba, abajo, izquierda y derecha, así mismo tendrá como atributo el correo emisor y receptor y un atributo booleano.

Creamos los métodos para los constructores y getters y setters.

Para la matriz, tenemos como atributos el nodo raíz, que es donde inicia toda la matriz dispersa, y métodos que nos servirán para la ubicación de los nodos por filas, columnas y así mismo la inserción del nodo donde corresponda.

Texto

Descripción generada automáticamente

Como todo nodo, inicializamos las variables en el constructor, y creamos los getters y setter para poder acceder a los atributos, manteniendo la información encapsulada.

Texto

Descripción generada automáticamente

Para buscar por columnas, empezaremos por la raíz y desde ese punto empezaremos a recorrer las columnas con el apuntador derecha que seria el getDcha y si encontramos el nodo que nos retorne el nodo.

Mismo caso para buscar por filas pero avanzaremos hacia abajo con getAbajo.

Texto

Descripción generada automáticamente

Para insertar o agregar un nodo a nuestra matriz dispersa, necesitamos ubicarlo tanto en filas como en columnas para que este bien posicionado, por ello necesitamos ciertos metodos para dicha ejecución, al nuevo nodo lo tenemos que insertar tanto en filas como en columnas, si no esta aun agregado(si es nullptr) se agrega tanto en fila como en columna, y posteriormente que se agrego en fila y columna, se procede a colocarlo en donde corresponde, que seria con los metoso insertarEnFila e insertarEnCol.

Texto

Descripción generada automáticamente

En estos métodos lo que hacemos es colocar bien los apuntadores a los nodos para que se conecten y por ende se agregue a la matriz dispersa.

Texto

Descripción generada automáticamente

El método verificar amistad, verifica si el usuario ya tiene en su matriz dispersa al otro usuario, para validar que ya son amigos.

Y finalmente el método graficar, el cual nos ayuda a crear la grafica mediante la estructura de graphviz y creando una imagen .png.

* Gestionar Soli:

Texto

Descripción generada automáticamente

Esta clase es importante ya que aquí se gestiona todo lo que son los procesos que puede hacer el usuario individualmente, mediante los métodos que se ven, se crean métodos estáticos debido a que no se requiere instanciar la clase para poder utilizarlos y para que no dependan de otros datos y simplifica la invocación y uso del mismo.

Texto

Descripción generada automáticamente

Tenemos el envio de solicitudes, donde tenemos como parámetro una lista de usuarios, y dos correos, con la lista de usuarios accedemos al nodo de cada usuario con su correo, posteriormente hacemos ciertas validaciones, como una revisión en nuestra matriz dispersa para ver si son amigos o no, una revisada a nuestra pila y lista de solicitudes para ver si no ya se le ha enviado una solicitud de amistad, si paramos todos los parámetros correctamente, se le envía la solicitud de amistad a la persona.

Texto

Descripción generada automáticamente

Mismos parámetros en aceptar solicitud, si se acepta, se agrega el usuario a la lista de amigos, se actualiza la matriz dispersa de amistad y se elimina de la pila de solicitudes y lista el correo de emisor y receptor.

Texto

Descripción generada automáticamente

Lo mismo para rechazar la solicitud, solo que no se agrega a ninguna lista ya que no son amigos.

Texto

Descripción generada automáticamente

Creamos las publicaciones y mandamos a llamar a los métodos que anteriormente explicamos en las clases.

Al igual que para ver publicaciones.

Texto

Descripción generada automáticamente

Para ver las publicaciones de los amigos y del usuario, se agrega cada publicación a lo que es la lista circualr y posteriormente mediante unas validaciones, la mostramos y el usuario puede navegar entre ellas.

Texto

Descripción generada automáticamente

En las gráficas básicamente solo mandamos graficar de cada clase para que se ejecute correctamente.

* Main:

Texto

Descripción generada automáticamente con confianza baja

En esta clase que es la principal, es donde el usuario interacciona mediante la consola con el programa, cada método tiene su función y algunos sus propios menús.