Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Septiembre-Diciembre 2017

Proyecto III: Simulación de mensajería de texto

1. Introducción

Hoy en día son muy populares los servicios de mensajería por Internet como Whatsapp [1] y Telegram [2]. El objetivo de este proyecto es el de hacer un simulador de servicio de mensajería.

2. Especificación de la aplicación

La especificación de la aplicación se hará por medio de los Tipos Abstractos de Datos (TADs) y el programa cliente que se encarga de la interacción entre el usuario y la aplicación.

2.1. TAD Usuario

El TAD Usuario describe la representación de un usuario de la aplicación de ALGOGRAM.

2.1.1. Atributos

Un usuario de la aplicación posee los siguientes atributos, dados con sus tipos de datos.

String nombre: Nombre del usuario. Este será el identificador del usuario.

String password: Contraseña del usuario.

String telefono: Indica el teléfono del usuario.

Lista de Usuarios contactos: Lista enlazada de elementos de tipo `Usuario` en donde los usuarios están ordenados por orden ascendente según el orden lexicográfico de los nombres de los usuarios.

Condiciones sobre los atributos:

- El atributo teléfono debe estar formado por caracteres entre 0 y 9.
- Los atributos *nombre*, *password*, y $telefono$ no podrán ser Nulo.
- Todos los usuarios agregados a la lista *contactos* deben de ser usuarios registrados en la aplicación.
- Los usuarios es la lista *contactos*, deben estar ordenados en orden lexicográfico con respecto al nombre del usuario.

[1] `https://www.whatsapp.com/`
[2] `https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here`

2.1.2. Operaciones

A continuación se presentan las operaciones del TAD.

Crear usuario: Constructor del TAD, crea un nuevo usuario. Los argumentos *nombre*, *password*, y *telefono* no podrán ser Nulos.

crearUsuario(String *nombre*, String *password*, String *telefono*, Lista *contactos*)

Agregar contacto: Se agrega un usuario a la lista *contactos*. Si el usuario no se encuentra en la lista lo agrega y retorna True, en caso contrario retorna False y no se modifica nada en la lista *contactos*. El usuario a agregar deberá haberse registrado en la aplicación. El usuario es agregado en la lista en la posición que le corresponde para mantener el orden de la lista.

agregarContacto(Usuario u) $\Rightarrow$ Bool

Eliminar contacto: Dado un nombre n entonces si algún usuario de la lista contacto tiene como nombre a n, entonces se elimina el usuario de la lista *contactos* y se retorna True, en caso contrario la lista *contactos* permanece igual y se retorna False.

eliminarContacto(String n) $\Rightarrow$ Bool

Mostrar contactos: Se muestra por salida estándar todos los usuarios de la lista *contactos* ordenados por orden alfabético del nombre del contacto, en forma de pares nombre y teléfono.

mostrarContactos(void)

2.2. TAD Registro de Usuarios

TAD que guarda todos los usuarios registrados en la aplicación de ALGOGRAM.

2.2.1. Representación

Está representado con una tabla de hash estática, basada en método de encadenamiento, que contendrá elementos de tipo `Usuario`.

2.2.2. Atributos

int n: Número de usuarios registrados.

Tabla de Hash tablaRU: Tabla de hash en donde se almacenan los usuarios registrados. Debe ser implementada como una tabla de hash de tamaño fijo basada en el método de encadenamiento. Las claves de los elementos serán los identificadores del usuario (el nombre del usuario) y el dato o valor a almacenar serán usuarios.

Condiciones sobre los atributos:

- El valor n debe coincidir con el número de elementos en *tablaRU*.
- Las claves de *tablaRU* corresponden a los identificadores del dato de tipo `Usuario`, que está asociado con la clave.

2.2.3. Operaciones

Crear tabla: Constructor del TAD, crea un nuevo registro en el cual *tablaRU* es de tamaño n.

crearTabla(int n)

Agregar usuario: Se agrega un elemento u de tipo `Usuario` en la tabla *tablaRU*. La clave va a ser el nombre del usuario u. Si la clave del elemento a agregar se encuentra en la tabla de usuarios, se retorna True. Por el contrario, si no logra agregarlo dado a que el usuario ya se encuentra registrado se retorna False.

agregarUsuario(Usuario u) $\Rightarrow$ Bool

Eliminar Usuario: Dado un nombre n, si algún elemento en *tablaRU* tiene una clave igual n, entonces se elimina al usuario asociado con la clave n y se retorna True. En caso contrario, la *tablaRU* queda igual y se retorna False.

eliminarUsuario(String n) $\Rightarrow$ Bool

Cargar usuarios desde archivo: Dado el String del nombre de un archivo, se deberá poder cargar los usuarios desde un archivo. Estos deberán agregarse a la tabla *tablaRU*. En caso en que se puedan cargar todos los usuarios correctamente, la aplicación devolverá True, y en caso contrario False.

cargarUsuarios(String *archivo*) $\Rightarrow$ Bool

El archivo con los usuarios a cargar tendrá el siguiente formato:

```
usuario_1[TAB]password_1[TAB]telefono_1
usuario_2[TAB]password_2[TAB]telefono_2
usuario_3[TAB]password_3[TAB]telefono_3
...
usuario_n[TAB]password_n[TAB]telefono_n
```

Mostrar usuarios registrados: Se muestra por salida estándar todos los usuarios registrados.

mostrarRegistro(void)

2.3. TAD Chat

Es tipo de datos cuya funcionalidad consiste almacenar los mensajes que se han enviado dos usuarios, en cualquier espacio de tiempo.

2.3.1. Atributos

String id: identificador de un Chat o conversación entre dos usuarios.

Pila de Strings mensajes: Contiene los mensajes que se han enviado entre dos usuarios.

Condiciones sobre los atributos:

- El *id* no puede ser nulo.
- El *id* es la combinación de dos nombres n_1 y n_2 que corresponden a dos usuarios u_1 y u_2, que deben estar registrados en TAD Registro de Usuarios. El String *id* tiene la forma n_1-n_2, donde $n_1 < n_2$ según el orden lexicográfico.
- Cada uno de los Strings contenidos en la pila mensajes debe comenzar de la siguiente manera **"n1: "** o **"n2: "** donde n1 y n2 son los nombres de los usuarios a los que corresponde esta conversación.

2.3.2. Representación

La pila *mensajes* debe estar implementada como una lista enlazada simple de Strings, donde cada String corresponde a mensaje enviado de un usuario a otro.

2.3.3. Operaciones

Crear Chat: Constructor del TAD. Recibe como entrada a dos usuarios. Con los nombres de los usuarios entonces se crea el identificador del Chat *id* que describió anteriormente.

crearChat(Usuario u_1, Usuario u_2)

Agregar mensaje al Chat: Recibe como entrada un `Usuario` u y String m, que indica que el usuario u manda el mensaje m al Chat. El efecto de esta operación es que se agrega el String $u.nombre$+": "+m a la pila de mensajes. Esta operación es un procedimiento que no retorna ningún valor.

agregarMensaje(Usuario u, String m)

2.3.4. TAD Conversaciones

El TAD que tiene como objetivo guardar todas los Chats o conversaciones entre todos los pares de usuarios.

2.3.5. Atributos

int n: Número de usuarios registrados.

Tabla de Hash tablaC: Estructura en donde se almacenan los Chats. Debe ser implementada como una tabla de hash de tamaño fijo basada en el método de encadenamiento. Las claves de los elementos serán identificadores *id* de elementos tipo `Chat` y el dato o valor a almacenar serán elementos de tipo `Chat`.

Condiciones sobre los atributos:

- El valor n debe coincidir con el número de elementos en *tablaC*.
- Las claves de tablaC corresponden a los identificadores de los datos de tipo Chat, que están asociados con la clave.

2.3.6. Representación

Las conversaciones están representadas *tablaC* como una tabla de hash estática, basada en el método de encadenamiento, que contendrá elementos de tipo `Chat`.

2.3.7. Operaciones

Crear tabla: Constructor del TAD, crea una nueva tabla *tablaC*, la cual es de tamaño n.

crearTabla(int n)

Agregar Conversación: Recibe como entrada un Chat c y su objetivo es agregar el Chat c a la tabla de conversaciones. Si la clave $c.id$ no existe en *tablaC* se agrega y el método retorna True. En caso de que clave $c.id$ si exista en *tablaC* se sustituye el Chat asociado a la clave $c.id$ por el nuevo Chat c y se retorna False.

agregarConversacion(Chat c)

Buscar Conversación: Retorna el Chat con clave *id*. El Chat corresponde la conversaciones entre dos usuarios u_1 y u_1, por lo que el *id* debe ser un String de la forma $u_1.nombre$-$u_2.nombre$. En caso de que el *id* no se encuentre en la *tablaC*, entonces se retorna Nulo.

buscarConversacion(String id) $\Rightarrow$ Chat

2.4. Módulo Cliente

El objetivo del cliente proveer la infraestructura por medio de la cual un usuario puede tener acceso a la aplicación. El módulo cliente será llamado `cliente.py` y usará las estructuras antes mencionadas para lograr realizar una aplicación funcional.

En el cliente proveerá al usuario de la característica, de poder insertar caritas en sus mensajes de texto. Para poder realizar esto, se debe implementar en el cliente las opciones de cargar caritas y mostrar caritas. A continuación las descripción de estos dos procedimientos.

Cargar Caritas: Dado el nombre de un archivo, se deberá cargar una serie de caritas que estarán disponibles.

cargarCaritas(String *archivo*)

El archivo a cargar tiene el siguiente formato:

```
carita_1=numero[TAB]carita_2=numero[TAB]carita_3=numero...carita_n=numero
```

Un ejemplo de un archivo sería

```
:)=2 :D=1 :C=3
```

En caso de que ya existan un paquete de caritas al momento de cargar las caritas, las caritas del nuevo paquete sustituyen a las ya existente.

Mostrar Caritas: Muestra la lista de caritas cargadas por el usuario. Se mostrarán sin el número del final.

mostrarCaritas(void)

Un ejemplo de la salida de este procedemiento:

```
:), x), :D , xD , :C , xC
```

Cuando se ejecuta el cliente lo primero que la aplicación mostrará por pantalla será el siguiente menú:

1. Registrarse
2. Iniciar sesión
3. Cargar usuarios
4. Eliminar usuario
5. Cargar caritas

A continuación se explican las opciones del menú.

1. Registrarse: Si el usuario desea registrarse el programa deberá pedirle sus datos y una vez creado un usuario, deberá regresar al menú antes mencionado. Si un usuario ya está registrado, entonces deberá aparecer una notificación la cual indique que el usuario se encuentra registrado.
2. Iniciar sesión: En el caso que el usuario quisiera iniciar sesión el programa deberá pedirle al usuario el nombre y contraseña y verificar si efectivamente se encuentra en el registro de usuarios. Si no se encuentra en el registro de usuarios deberá arrojar un error y devolverse al menú anterior. En caso de que si lo encuentra deberá aparecer el menú del usuario, el cual se explicará próximamente.
3. Cargar usuarios: Se deberá pedir al usuario el nombre del archivo que contiene los usuarios que desea agregar al registro de usuarios.

4. Eliminar usuario: Se deberá pedir al usuario el nombre del usuario que desea eliminar. Si el usuario se encuentra registrado, entonces debe eliminarlo del registro de usuarios. En caso contrario, se indica un mensaje de error al usuario.

5. Cargar caritas: La aplicación podrá cargar un paquete de caritas por medio de un archivo. Se le solicita el nombre del archivo con las caritas, luego las mismas se cargan en la aplicación. Las caritas cargadas estarán disponibles a todos los usuarios registrados.

Una vez que un usuario unicia una sesión, entonces debe aparece el menú del usuario, el cual posee las siguientes opciones:

1. Agregar contacto
2. Eliminar contacto
3. Mostrar contactos
4. Ver conversación
5. Enviar mensaje
6. Mostrar caritas
7. Mostrar usuarios registrados
8. Cerrar sesión

La explicación de cada opción es la siguiente:

1. Agregar contacto: Se deberá agregar un contacto a la lista de contactos del usuario. Se deberá pedir al usuario nombre del contacto a agregar. Sólo podrán agregar usuarios registrados.

2. Eliminar contacto: Se deberá pedir al usuario el nombre del contacto que desea eliminar y luego lo elimina de su lista de contactos.

3. Mostrar contactos: Se debe imprimir todos los contactos de un usuario por orden alfabético.

4. Ver conversación: El programa deberá pedirle al usuario que inserte el nombre de la persona. Luego deberá aparecer una pantalla donde salgan todos los mensajes o el Chat que el usuario tenga con ese usuario. En caso de no existir el usuario en la lista de contactos del usuario deberá notificarle del error.

5. Enviar Mensaje: Solicita al usuario el nombre de la persona a la que le quiere enviar un mensaje. Esa persona debe debe estar en la lista de contactos del usuario. En caso de que no este, se le indica un mensaje de error al usuario. Luego se le pide al usuario el mensaje a enviar, una vez recibido se agrega ese mensaje al Chat del contacto correspondiente.

6. Mostrar caritas: La aplicación podrá mostrar las caritas que tiene el usuario en su aplicación. Más adelante se explica la implementación de esta función.

7. Mostrar usuarios registrados: La aplicación podrá mostrar todos los usuarios registrados en su aplicación.

8. Cerrar sesión: Si el usuario cierra sesión, deberá salir de la sesión y regresar al menú inicial.

3. Detalles de la implementación

A continuación se indican los archivos obligatorios que debe poseer su aplicación y la función de cada uno de ellos:

`usuario.py`: Este módulo deberá tener implementado el TAD Usuario y las funciones necesarias relacionadas a éste para cumplir los requisitos del proyecto.

`registro_usuarios.py`: Este módulo deberá tener implementado el TAD Registro Usuarios y las funciones necesarias relacionadas a éste para cumplir los requisitos del proyecto.

`chat.py`: Este módulo deberá tener implementado el TAD Chat y las funciones necesarias relacionadas a éste para cumplir los requisitos del proyecto.

`conversaciones.py`: Este módulo deberá tener implementado el TAD Conversaciones y las funciones necesarias relacionadas a éste para cumplir los requisitos del proyecto.

`cliente.py`: Este módulo deberá tener implementar el cliente de la aplicación.

La ejecución del programa será por medio del comando:

$ *python3 cliente.py*

No puede hacer uso de las estructura de datos diccionario, ni lista que vienen con el lenguaje `Python3`. Puede hacer uso del módulo *arrayT* si así lo prefiere. Recuerde que su código debe estar debidamente documentado y debe seguir la guía de estilo de programación indicada en clase.

4. Condiciones de Entrega

Su programa debe poderse ejecutarse en `Python3`, en el sistema operativo Linux. Si un programa no se ejecuta, el equipo tiene cero como nota del proyecto. El trabajo es por equipos de laboratorio. Debe entregar los códigos fuentes de sus programas, en un archivo comprimido llamado Proyecto3-X-Y.tar.gz donde X y Y son los números de carné de los integrantes del grupo. La entrega se realizará por medio del aula virtual antes de la 2:00 pm del sabado 1 de abril de 2017. Cada grupo debe entregar firmada, en el laboratorio de la semana 12, la "Declaración de Autenticidad para Entregas", cuya plantilla se encuentra en la página del curso. Ambos integrantes del equipo deben trabajar en el proyecto. El no cumplimiento de algunos de los requerimientos podrá resultar el rechazo de su entrega.