Presentación de resultados

Funcionalidades

Las funcionalidades de administración son las siguientes:

- Registrar una nueva cuenta en el servidor:
  Mediante la librería de slixmpp, se pide el usuario y contraseña los cuales son enviados a través de la librería para registrar el usuario en el dominio solicitado

- Iniciar sesión con una cuenta:
  Con credenciales existentes se inicia sesión mostrando todas las funcionalidades de comunicación que solicitaba el chat

- Cerrar sesión con una cuenta
  Cuando ya no se desea seguir utilizando la aplicación se cierra sesión con la cuenta

- Eliminar la cuenta del servidor
  Mediante credenciales validas se elimina el usuario del dominio al que se inscribió

Las funcionalidades de comunicación son las siguientes:

- Mostrar todos los contactos y sus estados:
  Todos los usuarios amigos se listarán mostrando su disponibilidad y su mensaje de estado

- Agregar un usuario a los contactos
  Dado un usuario se enviará una solicitud de suscripción

- Mostrar detalles de contacto de un usuario
  Mediante un usuario ingresado se hace una solicitud para consultar la disponibilidad y mensaje de estado del usuario

- Comunicación 1 a 1 con cualquier contacto/usuario

Se lista los contactos agregados para mayor comodidad y evitar tipos en los nombes. Se selecciona a la persona que se le desea enviar el mensaje. Luego se ingresa el mensaje y se envía al receptor

- Participar en conversaciones grupales
  Se despliega un menú para poder: Crear, Unirse invitar o enviar un mensaje a un grupo. El usuario deberá seleccionar que opción desea hacer en este submenú de grupo

- Definir mensajes de presencia
  Permite ingresar un mensaje de estado y cambiar la disponibilidad de la cuenta

- Enviar / recibir notificaciones
  Esta no es una opción, es una funcionalidad general. Se muestran notificaciones cuando:
  - Se ha agregado a un amigo
  - Mensaje individual
  - Mensaje grupal
  - Invitación de grupo
  - Conexión de contactos
  - Desconexión de contactos

- Enviar / recibir archivos
  Se solicita el nombre del archivo con su extensión, luego se codifica en base 64 para poder enviarlo como un mensaje. Este se le creo un formato especial para poder decodificarlo a pesar de ser enviado como mensaje. El formato utilizado fue: file|extensión|base64

Dificultades y lecciones aprendidas

Una de las mayores dificultades fue la programación concurrente, mis conocimientos y capacidades se encontraban en la programación paralela la cual es muy diferente a la concurrencia en Python. Tuve que leer e investigar un poco sobre como funcionaba la concurrencia para poder llevar a cabo mi proyecto. No me arrepiento para nada haberlo hecho en Python ya que tuve que salir de mi zona de confort para aprender esta técnica de programación. Entre las cosas que aprendí fue sobre como manejar funciones asíncronas, como tratar liberar y manejar la “memoria” o el manejador de tareas asíncronas. Además aprendí sobre las instrucciones asíncronas como son los aprints y ainputs .

Por alguna razón que desconozco algunas de las funcionalidades que tiene slixmpp no me funcionaron como deberían. Slixmpp permite crear handlers personalizado para manejar ciertos eventos como el programador desea. Entre esos handlers se encuentra recibir mensajes, suscripciones, invitaciones a grupos etc. Para poder hacer uso de estos handlers es necesario crear y definir una función asíncrona que deberá realizar las acciones deseadas en base a la presencia que reciba. El problema que tuve fue que estas funciones que creaba operaban lo que se les programo respecto a la parte del protoloc XMPP pero al momento de colocar prints asíncronos para mostrar las notificaciones, estos no se miraban reflejados en la CLI. Tras leer documentación y consultar con compañeros, todo estaba correcto y debería de funcionar, dado que los prints de las notificacioes no funcionaban se procedió a implementar una solución.

Este proceso de arreglar las notificaciones fue el principal proceso que me ayudo a solidificar mis conocimientos de programación asíncrona. Para las solicitudes de suscripción cree una tarea asíncrona que consulta el roaster del usuario para compararlo con una versión anterior, de esta manera fue posible identificar si se agrego a un nuevo contacto y mostrar la notificación. Para las conexiones y desconexiones de los contactos se utilizó una implementación muy similar. Mediante otra tarea asíncrona se esta consultando el roaster y se guardo si el usuario esta desconectado o no, luego esta tarea consulta nuevamente para ver si hay un cambio en la disponibilidad. Si se encentra un cambio se muestra una notificación de que el contacto esta disponible o desconectado, no se muestra cambios de estado como no disponible, ausente etc.

Una de las más importantes lecciones aprendidas fue que dada la naturaleza del protocolo XMPP existen ya muy pocas librerías que continúan con soporte o que siguen siendo útiles. Esta falta de librerías útiles es una situación general entre todos los lenguajes, no solo en Python. La mayor dificultad para realizar el proyecto no fue utilizar el protocolo, fue hacer funcionar la librería.