

INSTITUTO TECNOLÓGICO DE SANTO DOMINGO

ÁREA DE INGENIERÍAS



**APLICACIÓN DE MENSAJERÍA INSTANTÁNEA DE SESIÓN ÚNICA
CON MÚLTIPLES NIVELES DE CIFRADO**

Proyecto de creación que para obtener el título de:

INGENIERO EN CIBERSEGURIDAD

Presentan:

Guillermo Jose Brito Genao - 1074936

Mark Alexander Benítez Kernogo - 1086207

Santo Domingo, República Dominicana

Enero de 2022

ÍNDICE

| | |
|------------------------------------|----|
| RESUMEN (ABSTRACT) | 4 |
| CAPÍTULO I | 5 |
| TÍTULO DEL PROYECTO | 5 |
| DESCRIPCIÓN DEL PROBLEMA | 5 |
| DESCRIPCIÓN DE LA IDEA INCIAL..... | 5 |
| OBJETIVO GENERAL..... | 5 |
| OBJETIVOS ESPECÍFICOS..... | 5 |
| PALABRAS CLAVES | 6 |
| CAPÍTULO 2..... | 8 |
| MARCO TEÓRICO..... | 8 |
| CAPÍTULO 3..... | 10 |
| MARCO METODOLÓGICO..... | 10 |
| CAPITULO 4..... | 11 |
| DESCRIPCIÓN DEL PROYECTO..... | 11 |
| PLAN DE TRABAJO | 12 |
| HITOS Y ENTREGABLES | 13 |
| SUPUESTOS O ASUNCIONES | 13 |
| FUERA DE ALCANCE | 13 |
| RESTRICCIONES DEL PROYECTO | 13 |



| | |
|--|----|
| RIESGOS DEL PROYECTO | 14 |
| PLANIFICACIÓN DE RR.HH..... | 14 |
| CAPITULO 5..... | 15 |
| CONCLUSIONES Y RECOMENDACIONES..... | 15 |
| REFERENCIAS..... | 16 |
| ANEXOS | 18 |
| MANUAL DE USUARIO | 18 |
| SUPUESTOS | 18 |
| PASO 1. INSTALAR LAS DEPENDENCIAS NECESARIAS | 18 |
| PASO 2. EJECUTAR EL SERVIDOR..... | 19 |
| PASO 3. EJECUTAR EL CLIENTE..... | 21 |
| VERSIÓN CONSOLA | 21 |
| VERSIÓN CON INTERFAZ GRAFICA | 23 |
| ENLACE AL REPOSITORIO DEL CÓDIGO FUENTE..... | 26 |



RESUMEN (ABSTRACT)

Hoy en día la mayoría de la población utiliza aplicaciones de chat “populares” desarrolladas por grandes compañías tecnológicas y confían a ciegas en que sus mensajes están siendo protegidos y que su privacidad es respetada. Sin embargo, para nadie es extraño el hecho de que dichas compañías de redes sociales se han visto envueltas numerosas veces en fuertes controversias sobre la manipulación indebida de los mensajes privados de sus clientes.

Mediante la realización de este proyecto de creación buscamos solventar la necesidad de proteger la privacidad en las conversaciones vía chat. Para ello, desarrollamos nuestro propio software de mensajería instantánea utilizando el lenguaje de programación Python. Nuestro software cuenta con diferentes tecnologías de encriptación y hashes que garantizan la privacidad, integridad y confidencialidad de los mensajes.

Con el desarrollo de este proyecto, se entrega al público un aplicativo de código abierto y libre costo pues solo utiliza tecnologías Open Source. Además, al ser de código abierto, cualquier persona puede revisar el código, entender cómo funciona y proponer mejoras para el futuro. En el siguiente documento se presenta el proyecto con más detalle y en los anexos se puede encontrar el manual de uso del software para los usuarios interesados.

CAPÍTULO I

TÍTULO DEL PROYECTO

‘Aplicación de mensajería instantánea de sesión única con múltiples niveles de cifrado’

DESCRIPCIÓN DEL PROBLEMA

Buscamos solventar la necesidad de proteger la privacidad en las conversaciones vía chat. Últimamente las grandes compañías de mensajería instantánea se han visto envueltas en controversias sobre la manipulación indebida de los mensajes privados de sus clientes. Por ejemplo, muchos usuarios tienen la sospecha de que sus mensajes privados son monitoreados y analizados para fines publicitarios. Lo cual indica que hay una violación de la privacidad en la comunicación.

DESCRIPCIÓN DE LA IDEA INICIAL

Se propone desarrollar un aplicativo que integre las tecnologías de criptografía más usadas en la actualidad. La idea consiste en la creación de un chat privado de sesión única, el cual cuente con métodos de autenticación, implementación de hashes para verificar la integridad de los mensajes, además de incorporar otros elementos de la criptografía moderna, como cifrado de flujo y en bloque, así como algoritmos simétricos y asimétricos e intercambio de llaves de forma segura.

OBJETIVO GENERAL

Desarrollar una aplicación de mensajería instantánea que garantice la integridad, la confidencialidad y sobre todo la privacidad de la información de los usuarios.

OBJETIVOS ESPECÍFICOS

- Utilizar los algoritmos de encriptación más avanzados de la criptografía moderna.
- Desarrollar una interfaz amigable para el usuario.
- Proporcionar los medios que garanticen la integridad de la información en el intercambio de los mensajes.
- Asegurar la correcta destrucción de los mensajes una vez terminada la sesión.



PALABRAS CLAVES

Python: Python es un lenguaje de alto nivel que es ampliamente utilizado para desarrollar aplicaciones web o de escritorio. Es multiplataforma de código abierto, brindando sencillez y amplias posibilidades que facilitan el trabajo con inteligencia artificial, big data, machine learning y data science (Python, 2021).

Cliente: Es un ordenador o software que accede a un servidor y recupera servicios especiales o datos de él. Este tiene como función estandarizar las solicitudes, transmitirlos al servidor y procesar los datos obtenidos (RyteWiki, 2021).

Servidor: Es un dispositivo que almacena, distribuye y suministra información. El servidor ofrece la información demandada por el cliente siempre y cuando el cliente esté autorizado. Los servidores funcionan basándose en el modelo "cliente-servidor" (Carrera L, 2021).

Websocket: Es una conexión persistente entre un cliente y un servidor. Esta tecnología provee un canal de comunicación bidireccional, full-duplex, que opera sobre el protocolo HTTP en un socket TCP/IP (Sookocheff K, 2019).

Unix Time: Es un sistema para la descripción de instantes de tiempo: se define como la cantidad de segundos transcurridos desde la medianoche UTC del 1 de enero de 1970, sin contar segundos intercalares. Es universalmente usado no solo en sistemas operativos tipo-Uinx, sino también en muchos otros sistemas computacionales (Wikipedia, 2021).

Criptografía: Es el estudio de técnicas de comunicación seguras que permiten que solo el remitente y el destinatario previsto de un mensaje vea su contenido (Kaspersky, 2021).

Dentro de la criptografía existen categorías, las cuales se definen a continuación:

Criptografía Simétrica: El cifrado simétrico es un tipo de cifrado en el que solo se utiliza una clave (una clave secreta) para cifrar y descifrar información electrónica. Las entidades que se comunican mediante cifrado simétrico deben intercambiar la clave para que pueda utilizarse en el proceso de descifrado (López A, 2021).

Criptografía Asimétrica: La criptografía asimétrica, también conocida como criptografía de clave pública, es un proceso que utiliza un par de claves relacionadas, una clave pública y una privada, para cifrar y descifrar un mensaje y protegerlo del acceso o uso no autorizado. Una clave

pública es una clave criptográfica que puede ser utilizada por cualquier persona para cifrar un mensaje de modo que solo pueda ser descifrado por el destinatario previsto con su clave privada. Una clave privada, también conocida como clave secreta, se comparte solo con el iniciador de la clave (López A, 2021).

Dentro de la criptografía simétrica, existen dos grandes grupos de algoritmos:

Algoritmos de cifrado por bloques: Aplican su transformación sobre bloques de datos de longitud fija (Salesa J. S., 2016).

Algoritmos de cifrado de flujo: Combinan el texto plano con un flujo de clave que se genera con ayuda de la clave de cifrado donde se aplicara bit por bit sobre el texto plano (Salesa J. S., 2016).



CAPÍTULO 2

MARCO TEÓRICO

Para iniciar las bases del proyecto, se inició mirando los aspectos teóricos de la conectividad vía internet, el cual se basa en la conformación de sockets, los cuales son un punto final que funciona estableciendo un enlace de comunicación de red bidireccional entre el extremo del servidor y el programa receptor del cliente (SpeedCheck, 2021).

Un socket de red está conformado por la IP y el puerto por el cual se realizará la conexión, a nivel de programación esto se puede ver con la estructura [IP]:[Puerto], un ejemplo del mismo sería 192.168.0.1:8080, el cual representaría un enlace entre la dirección IP y un puerto. Para realizar una conexión entre un cliente y un servidor es necesario que ambos participantes posean un socket, para poder compartirlo para que el resto sepa con quien establecerá la comunicación. Dentro del lenguaje de programación Python, el cual es el utilizado para el presente proyecto, existe la librería denominada <<socket>> la cual construye el objeto requerido que podrá ser manipulado por el lenguaje para entablar una conversación bidireccional.

Los sockets únicamente entablan la línea de comunicación entre dispositivos, pero dicha comunicación se envía en texto plano, específicamente en el estándar Unicode, el cual es un sistema de codificación de caracteres diseñados para soportar el intercambio, procesamiento y visualización universal de textos escritos en diversos idiomas, reemplazando el estándar ASCII (De Arrilucea J. P., 2016). Pero estos estándares no proveen ningún tipo de encriptado para poder mantener la confidencialidad del mensaje al poder ser visualizado con claridad por parte de un tercero. Esta violación a la confidencialidad del mensaje corrompe la triada CID original (Confidencialidad, Integridad y Disponibilidad) (Sociales R, 2021).

Es por ello por lo que el proyecto implementa el cifrado simétrico como forma de mantener la privacidad de la información transmitida a través del Internet, esto se implementa mediante la librería de python <<cryptography.fernet>>, la cual genera la clave simétrica a partir de un número pseudoaleatorio generado mediante el uso del Unix Time (Wikipedia, 2021). Implementando el cifrado simétrico se provee una capa inicial de seguridad a la transmisión del mensaje, pero esto

no evita que la clave simétrica pueda ser capturada por un tercero, provocando que el objetivo del proyecto no se realice de manera satisfactoria.

Para distribuir la llave simétrica de manera segura, se realiza el procedimiento de intercambio con la implementación de la criptografía asimétrica, la cual, como se explicó en el glosario de términos, utiliza dos llaves, una pública y privada, el proceso del intercambio se efectúa de la siguiente manera:

1. Se genera la llave simétrica.
2. Se generan las llaves pública y privada, mediante el uso de la librería <<Crypto.PublicKey>> con el uso del algoritmo RSA.
3. Se encripta la llave simétrica con la llave privada.
4. Se distribuye la llave simétrica encriptada mediante el medio de comunicación.
5. Se comparte la llave pública para poder desencriptar la llave enviada anteriormente.
6. A partir de este momento, todo mensaje que se transmitirá por los sockets se encriptará con la llave simétrica.

Este proceso de intercambio no posee un único objetivo de asegurar la correcta distribución de la llave simétrica, sino que también, las estadísticas han demostrado que el uso de llaves simétricas para el encriptado y desencriptado de la información se realiza a una mayor velocidad que mediante el uso de llave asimétrica, lo que contribuye a una tasa mayor de respuesta ante las solicitudes de un cliente (López A, 2021).

CAPÍTULO 3

MARCO METODOLÓGICO

Debido a que el proyecto consiste en el desarrollo de un aplicativo, utilizaremos una de las metodologías tradicionales del desarrollo de software: La metodología incremental. En esta metodología de desarrollo de software se va construyendo el producto final de manera progresiva. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada. El software se puede empezar a utilizar incluso antes de que se complete totalmente y, en general, es mucho más flexible que las demás metodologías (Santander Universidades, 2021).

Para la planificación y la organización de las tareas utilizaremos la metodología Kanban, el cual es un marco popular utilizado para implementar el desarrollo de software ágil y DevOps. Requiere comunicación en tiempo real de la capacidad y total transparencia del trabajo. Los elementos de trabajo se representan visualmente en un tablero Kanban, lo que permite a los miembros del equipo ver el estado de cada trabajo en cualquier momento (Atlassian, n.d). Este consiste en un tablero con columnas, las cuales representan una fase en el procedimiento de desarrollo, esto con la finalidad de poder visualizar de manera fácil y rápida cada una de las actividades necesarias y saber cuál es su estado dentro del proyecto.

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación (García B. J., 2020).

Entre las ventajas que puede proporcionar un modelo de este tipo encontramos las siguientes:

- Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.
- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.



- Es más fácil probar y depurar en una iteración más pequeña.
- Es más fácil gestionar riesgos.
- Cada iteración es un hito gestionado fácilmente.

CAPITULO 4

DESCRIPCIÓN DEL PROYECTO

El software constara de dos partes: cliente y servidor.

El servidor utilizara sockets para manejar las conexiones con los diferentes clientes. Para ello se empleará el módulo Sockets de Python. Con ayuda de la librería "threading" el servidor creara un hilo independiente por cada cliente para separar el procesamiento de los datos. Luego el servidor deberá recibir por parte del usuario una clave secreta que servirá como clave de acceso, es decir, cuando nuevos clientes deseen unirse al chat deberán conocer cuál es la clave de acceso. Si no conocen la clave de acceso, los clientes no podrán acceder al chat.

Esta clave de acceso será usada posteriormente para encriptar una llave privada utilizando el algoritmo en flujo RC4. Dicha llave privada es parte del par de llaves RSA (una publica y otra privada) que el servidor generara cuando se inicialice. Este par de llaves es utilizado como una técnica de criptografía asimétrica para el intercambio seguro de otras llaves.

También el servidor generara una llave simétrica utilizada por el algoritmo AES para encriptar los mensajes que los clientes escriban en el chat. Esta llave simétrica será encriptada con la llave publica previamente generada. De esta manera, la llave simétrica que fue encriptada con la llave publica solo podrá ser desencryptada con la llave privada que a su vez esta encriptada con el algoritmo RC4.

El servidor ejecutará una función llamada "recieve ()" la cual se encargará de escuchar constantemente posibles conexiones de clientes nuevos. También el servidor mantendrá una lista de clientes activos que ira actualizando a medida que los clientes se vayan agregando y saliendo del chat.

Por otro lado, el servidor aceptara una serie de comandos que permitirán al administrador verificar ciertos parámetros como la cantidad de clientes en el servidor, lista de las conexiones activas, mostrar la clave de acceso RC4 y apagar el servidor.

El cliente le pedirá al usuario un username y la clave de acceso. El cliente procederá a calcular un hash de la clave que el usuario ingreso y se la enviará al servidor junto con el nombre de usuario. Luego de que el servidor valide que el usuario ingreso la clave de acceso correcta entonces ambos iniciaran el proceso del intercambio de las llaves. El cliente recibirá primero la llave privada que estará encriptada con la clave de acceso y luego recibirá la clave simétrica que estará encriptada con la llave pública. Mas adelante el cliente procederá a desencriptar las llaves para su uso posterior.

El cliente simultáneamente estará ejecutando dos funciones diferentes en dos hilos separados. Una de ellas se encargará de recibir los mensajes de los demás usuarios del chat mientras que la otra función se encargará de enviar los mensajes que el usuario quiera enviar a los demás miembros del chat.

PLAN DE TRABAJO

| Plan de Trabajo | | | | | | | | Línea Temporal - Semanas | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------|--|---|-------------------|----------|----------|----------------|----------------|--------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| Etapas | Act.# | Nombre de la Tarea | Duración (Semana) | Comienzo | Fin | Predecesora | Responsable | Completado % | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | |
| Proyecto Final 1 | Fase 1 | Plan de Proyecto | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1 | Presentar Ideas Iniciales | 2 | 10/08/21 | 24/08/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 2 | Formular Idea Oficial Seleccionada | 2 | 24/08/21 | 07/09/21 | 1 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 3 | Crear Plan de Trabajo | 2 | 07/09/21 | 21/09/21 | 2 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Plan de Proyecto Completado | 0 | 21/09/21 | 21/09/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Fase 2 | Preparacion Herramientas de Colaboración | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 4 | Crear Repositorio en GitHub | 1 | 14/09/21 | 21/09/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 5 | Crear Board en Trello | 1 | 14/09/21 | 21/09/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Herramientas de Colaboracion Listas | 0 | 21/09/21 | 21/09/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Fase 3 | Análisis de requerimientos de la aplicación | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 6 | Levantar Requisitos del Servidor | 1 | 21/09/21 | 28/09/21 | 5 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 7 | Levantar Requisitos del Cliente | 1 | 21/09/21 | 28/09/21 | 5 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 8 | Investigar Librerías a Utilizar | 1 | 21/09/21 | 28/09/21 | 6,7 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Listado de los requerimientos Completo | 0 | 28/09/21 | 28/09/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| Fase 4 | Programacion Cliente - Servidor | | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | Programar la Aplicacion del Servidor | 3 | 28/09/21 | 12/10/21 | 6,7,8 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | Programar la Aplicacion del Cliente | 3 | 28/09/21 | 12/10/21 | 6,7,8 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | Probar de la aplicación | 3 | 28/09/21 | 12/10/21 | 9,10 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | Corrección de errores | 3 | 28/09/21 | 12/10/21 | 11 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Hito | Aplicaciones del Cliente-Servidor Completo | 0 | 12/10/21 | 12/10/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Proyecto Final 2 | Fase 5 | Desarrollo de la Interfaz Grafica | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 13 | Diseñar la Interfaz Gráfica del Chat | 2 | 1/11/21 | 15/11/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 14 | Integrar el Cliente con la Interfaz Grafica | 2 | 15/11/21 | 29/11/21 | 9,10,13 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | Probar de la aplicación | 1 | 29/11/21 | 6/12/21 | 14 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 16 | Corrección de errores | 1 | 29/11/21 | 6/12/21 | 15 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Aplicación con Interfaz Gráfica | 0 | 6/12/21 | 6/12/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Fase 6 | Correcciones Finales | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 17 | Aplicar Correcciones Finales | 2 | 6/12/21 | 20/12/21 | 9,10,14 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 18 | Crear los Archivos Ejecutables Finales | 1 | 13/12/21 | 20/12/21 | 17 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Ejecutables Finales de la Aplicación | 0 | 20/12/21 | 20/12/21 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Fase 7 | Creación Manual de Uso del Software | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 19 | Crear Manual de Uso para Usuario Final | 2 | 3/01/22 | 17/01/22 | 17 | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Hito | Manual de Uso Completo | 0 | 17/01/22 | 17/01/22 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Fase 8 | Presentacion Final del Proyecto | | | | | | Guillermo/Mark | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | Preparar Presentacion PPT del Proyecto | 1 | 17/01/22 | 22/01/22 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Hito | Presentación Final | 0 | 22/01/22 | 22/01/22 | | Guillermo/Mark | 100% | | | | | | | | | | | | | | | | | | | | | | | | | | |

[Ver Plan de trabajo Completo en como archivo Adjunto]

HITOS Y ENTREGABLES

| Hito | Entregable | Fecha de entrega |
|---|---------------------------------------|------------------|
| Plan de proyecto completado | Plan de proyecto | 21/09/2021 |
| Herramientas de colaboración listas | Link del repositorio del proyecto | 21/09/2021 |
| Listado de los requerimientos completo | Requerimientos de la aplicación | 28/09/2021 |
| Aplicaciones Cliente y Servidor completas | Aplicación del cliente y del servidor | 12/10/2021 |
| Interfaz gráfica del cliente completa | Aplicación con interfaz gráfica | 06/12/2021 |
| Correcciones finales del proyecto | Ejecutables finales de la aplicación | 20/12/2021 |
| Manual de uso completado | Manual de uso del usuario | 10/01/2021 |

SUPUESTOS O ASUNCIONES

Se asume que la entrega final es una aplicación de escritorio desarrollada en Python que cuenta con dos partes: una parte que sería el servidor el cual sería una aplicación de consola y otra aplicación con interfaz gráfica que hará el papel de cliente.

La aplicación será desarrollada en el lenguaje de programación Python, por lo tanto, se espera que esta pueda correr en cualquier sistema operativo que tenga Python instalado.

Se espera que el usuario final pueda contar con un manual de uso adecuado para la utilización del software.

FUERA DE ALCANCE

1. Esta fuera del alcance entregar cualquier tipo de aplicación que no sea de escritorio/consola.

RESTRICCIONES DEL PROYECTO

El proyecto por su carácter educativo, sin fines de lucro, presenta las siguientes restricciones:

1. El proyecto se limitará a usar herramientas Open Source en caso de requerirlas, debido a que se espera que su coste sea de 0 pesos dominicanos.

RIESGOS DEL PROYECTO

Entre los riesgos que se pueden presentar a lo largo de la gestión son los siguientes:

1. El ordenador de los miembros del proyecto presente algún tipo de falla a lo largo de la vida del proyecto.
2. Alguno de los colaboradores presente una situación de salud que no le permita continuar con el proyecto por un periodo de tiempo.
3. La creación del proyecto sea muy compleja para el desarrollo con un equipo limitado de personas.

PLANIFICACIÓN DE RR.HH.

| Recursos | Rol o roles por ejecutar dentro del proyecto |
|-----------------|--|
| Guillermo Brito | Investigador, Analista, Diseñador, Programador, Redactor |
| Mark Benítez | Investigador, Analista, Diseñador, Programador, Redactor |

| Rol | Responsabilidades | Actividades del Plan de trabajo asignadas |
|--------------|--|---|
| Investigador | Investigar marco teórico y las tecnologías/librerías/herramientas necesarias para la realización del proyecto. | 1,2,3,4,5 |
| Analista | Levantar correctamente los requerimientos de la aplicación. | 6,7,8 |
| Diseñador | Diseñar la interfaz gráfica del cliente. | 13 |
| Programador | Programar la aplicación del cliente y del servidor en Python. | 9,10,11,12 |
| Redactor | Redactar el manual de uso del software para los usuarios finales. | 19,20 |

CAPITULO 5

CONCLUSIONES Y RECOMENDACIONES

A modo de conclusión se puede decir con certeza que el objetivo general del proyecto fue alcanzado exitosamente. Se logró desarrollar un software de mensajería instantánea que garantiza la integridad, la confidencialidad y sobre todo la privacidad de los mensajes intercambiados por los usuarios finales. Además, se utilizaron los algoritmos de encriptación más avanzados de la criptografía moderna y se crearon dos versiones del aplicativo: una versión por consola y otra con una interfaz gráfica amigable.

Importante destacar que los diferentes usuarios podrán usar la versión que más les guste, ya sea por consola o con interfaz gráfica, el servidor soportará ambas versiones de manera simultánea sin inconvenientes.

Cabe resaltar el hecho de que con este proyecto se entrega al público un aplicativo de código abierto y libre costo. Es decir, al ser de código abierto, cualquier persona puede revisar el código, entender cómo funciona y sugerir mejoras para el futuro. El software puede ser descargado directamente desde GitHub utilizando el siguiente enlace:

<https://github.com/Banano-Cry/Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption->

Recomendamos encarecidamente a los usuarios finales leer detenidamente el Manual de Uso del software para que puedan utilizarlo sin problemas y de manera correcta. Dicho manual puede ser encontrado en los anexos de este documento y en el link de GitHub.

REFERENCIAS

Python. (2021, October 18). Python.Org. Retrieved October 18, 2021, from <https://www.python.org/about/>

Atlassian. (n.d.). *What is kanban?* Retrieved October 18, 2021, from <https://www.atlassian.com/agile/kanban>

Brush, K., Rosencrance, L., & Cobb, M. (2021, September 27). *asymmetric cryptography (public key cryptography)*. SearchSecurity. <https://searchsecurity.techtarget.com/definition/asymmetric-cryptography>

Carrera, L. (2021, February 26). *¿Qué es un servidor, cómo funciona y qué tipos hay?* TIC Portal. Retrieved October 18, 2021, from <https://www.ticportal.es/glosario-tic/servidores>

Colaboradores de Wikipedia. (2021, August 24). *Tiempo Unix*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Tiempo_Unix

De Arrilucea, J. P. (2016, September 16). *¿Qué es UTF-8?* El visualista. Retrieved October 18, 2021, from <https://www.elvisualista.com/2016/09/16/que-es-utf-8/>

García, B. J. (2020). *Fundamentos de los Requisitos de Software (Gestión del Ciclo de Vida del Software n° 1) (Spanish Edition)* (1st ed.). Unknown.

Kaspersky. (2021, January 13). *Cryptography Definition*. Www.Kaspersky.Com. Retrieved October 18, 2021, from <https://www.kaspersky.com/resource-center/definitions/what-is-cryptography>

López, A. (2021, April 4). *Todo sobre criptografía: Algoritmos de clave simétrica y asimétrica*. RedesZone. <https://www.redeszone.net/tutoriales/seguridad/criptografia-algoritmos-clave-simetrica-asimetrica/>

¿Qué es un Cliente? - Ryte Digital Marketing Wiki. (n.d.). RyteWiki. Retrieved October 18, 2021, from <https://es.ryte.com/wiki/Cliente>

¿Qué es un socket? (n.d.). SpeedCheck. Retrieved October 18, 2021, from <https://www.speedcheck.org/es/wiki/socket/>

Salesa, J. S. (2016, November 10). *Criptografía y seguridad*. Adictos al trabajo. Retrieved October 18, 2021, from <https://www.adictosaltrabajo.com/2016/11/10/criptografia-y-seguridad/>

Santander Universidades. (2021, August 31). *Metodologías de desarrollo software / Blog*. Becas Santander. Retrieved October 18, 2021, from <https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html>

Sociales, R. (2021, February 23). *Conceptos básicos de Ciberseguridad: Triada CIA*. UNIAT. Retrieved October 18, 2021, from <https://www.uniat.edu.mx/conceptos-basicos-de-ciberseguridad/>

Sookocheff, K. (2019, April 4). *How Do Websockets Work?* Kevin Sookocheff. Retrieved October 18, 2021, from <https://sookocheff.com/post/networking/how-do-websockets-work/>

Turner, D. S. P. M. (n.d.). *Symmetric Key Encryption - why, where and how it's used in banking*. Cryptomathic. Retrieved October 18, 2021, from <https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking>

ANEXOS

MANUAL DE USUARIO

SUPUESTOS

Se asume que el usuario final tiene instalado en su sistema operativo Python 3.7 o cualquier versión compatible que le permita instalar todas las dependencias mencionadas más adelante.

La versión de python que se utilizó para desarrollar este proyecto Python 3.7.

PASO 1. INSTALAR LAS DEPENDENCIAS NECESARIAS

Primero debemos instalar las dependencias necesarias para que el programa funcione de manera correcta. Para ello abriremos la terminal y utilizaremos la línea de comando “python -m pip install...” para instalar cada una las siguientes librerías: [Ejemplo, figura 1]

- **Numpy**

```
python -m pip install numpy
```

- **Cryptography**

```
python -m pip install cryptography
```

- **Crypto**

```
python -m pip install pycryptodome
```

- **Pyngrok**

```
python -m pip install pyngrok
```

- **PySimpleGui**

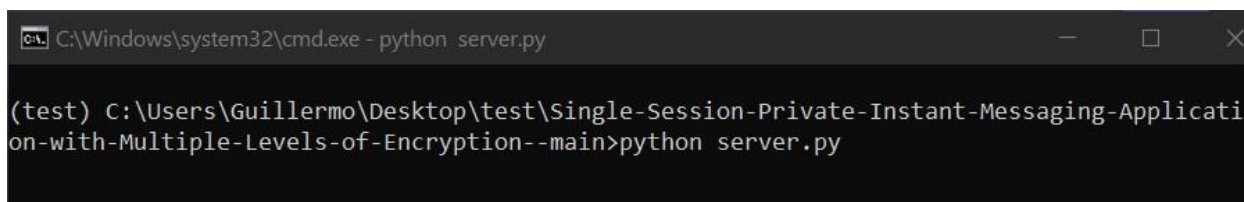
```
python -m pip install PySimpleGUI
```

A screenshot of a Windows command prompt terminal window. The prompt is '(test) C:\Users\Guillermo\Desktop>'. The user has entered 'python -m pip install numpy'. The output shows 'Collecting numpy', followed by a progress bar for downloading 'numpy-1.21.5-cp37-cp37m-win_amd64.whl (14.0 MB)' at '14.0 MB 3.3 MB/s'. It then says 'Installing collected packages: numpy' and 'Successfully installed numpy-1.21.5'. A yellow warning message follows: 'WARNING: You are using pip version 20.1.1; however, version 21.3.1 is available. You should consider upgrading via the 'C:\Users\Guillermo\Desktop\test\Scripts\python.exe -m pip install --upgrade pip' command.' The prompt returns to '(test) C:\Users\Guillermo\Desktop>'.

```
(test) C:\Users\Guillermo\Desktop>python -m pip install numpy
Collecting numpy
  Downloading numpy-1.21.5-cp37-cp37m-win_amd64.whl (14.0 MB)
    |████████████████████| 14.0 MB 3.3 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.21.5
WARNING: You are using pip version 20.1.1; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\Guillermo\Desktop\test\Scripts\python.exe
-m pip install --upgrade pip' command.
(test) C:\Users\Guillermo\Desktop>
```

PASO 2. EJECUTAR EL SERVIDOR

Luego de haber instalado todas las dependencias antes mencionadas, el siguiente paso es iniciar el servidor. El servidor debe ser ejecutado en un solo dispositivo vía línea de comando “python server.py” como se muestra en la siguiente imagen. Luego, los demás clientes que quieran agregarse al chat podrán hacerlo.



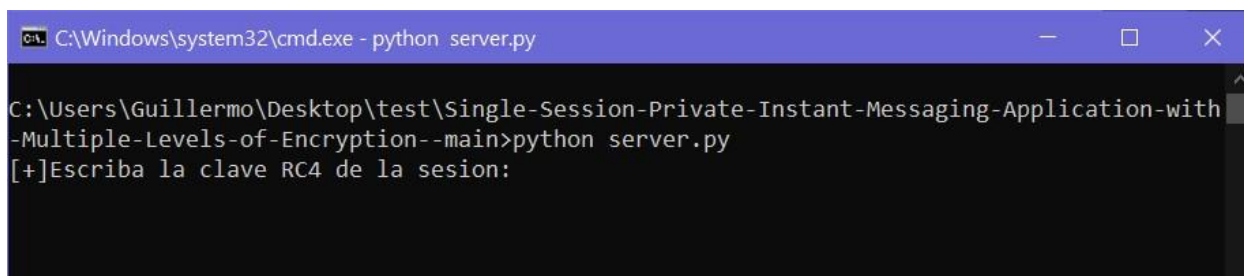
```
C:\Windows\system32\cmd.exe - python server.py

(test) C:\Users\Guillermo\Desktop\test\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main>python server.py
```

Una vez iniciado el servidor, este pedirá que se ingrese una nueva clave. **OJO: Esa clave deberá ser conocida por todos los clientes que quieran participar e integrarse en el chat.**

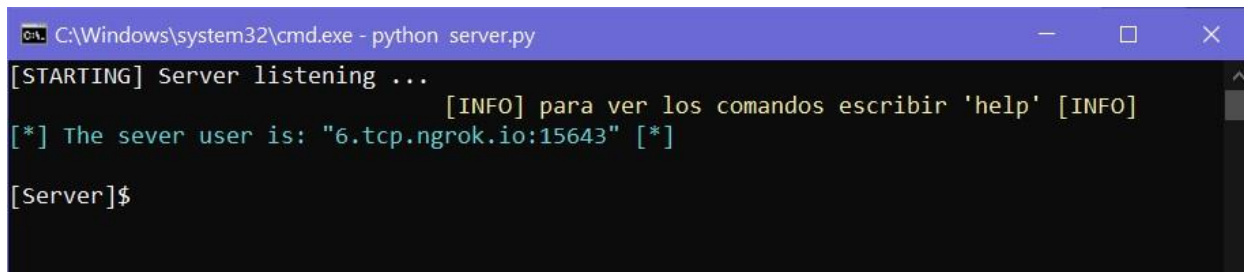
Después de esto, el servidor mostrara un link mediante el cual los nuevos clientes podrán unirse al servidor. Los todos los clientes que quieran integrarse al chat deberán poseer dos cosas:

1. El Link del servidor
2. La clave del servidor



```
C:\Windows\system32\cmd.exe - python server.py

C:\Users\Guillermo\Desktop\test\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main>python server.py
[+]Escriba la clave RC4 de la sesion:
```



```
C:\Windows\system32\cmd.exe - python server.py

[STARTING] Server listening ...
[INFO] para ver los comandos escribir 'help' [INFO]
[*] The sever user is: "6.tcp.ngrok.io:15643" [*]

[Server]$
```

En este ejemplo, el link que se ha generado automáticamente para el servidor es: 6.tcp.ngrok.io:15643. El link no es estático, por lo tanto, cada vez que se inicie el servidor, se generara un nuevo link.

El cliente deberá tener el link completo para poder ingresar al chat, es decir, incluyendo el número de puerto al final: **6.tcp.ngrok.io:15643**

El servidor cuenta con el comando “help” donde le indica al usuario las diferentes funciones que este puede realizar dentro del servidor.

```
C:\Windows\system32\cmd.exe - python server.py
[STARTING] Server listening ...
[INFO] para ver los comandos escribir 'help' [INFO]
[*] The sever user is: "6.tcp.ngrok.io:15643" [*]

[Server]$ help

[*]Lista de comandos del servidor[*]
[1]exit --> Salir del servidor
[2]count --> Cantidad de usuarios en el servidor
[3]list --> Lista a los usuarios en el servidor
[4]key --> Muestra la clave RC4 temporal

[Server]$
```

```
C:\Windows\system32\cmd.exe - python server.py
[STARTING] Server listening ...
[INFO] para ver los comandos escribir 'help' [INFO]
[*] The sever user is: "6.tcp.ngrok.io:15643" [*]

[Server]$ help

[*]Lista de comandos del servidor[*]
[1]exit --> Salir del servidor
[2]count --> Cantidad de usuarios en el servidor
[3]list --> Lista a los usuarios en el servidor
[4]key --> Muestra la clave RC4 temporal

[Server]$ count
[*] Los usuarios conectados actualmente son: [*]

[Server]$ list
[*] Hay 0 usuarios en el servidor [*]

[Server]$ key
[*] La clave RC4 actual es 'hola1234' [*]

[Server]$
```

Importante:

La persona encargada de ejecutar el servidor es la que decidirá que clave se usará para ingresar a su chat. En caso de que no recuerde que clave puso, puede ejecutar el comando “key” para ver la clave y luego compartirla con aquellas personas autorizadas a ingresar al chat.

PASO 3. EJECUTAR EL CLIENTE

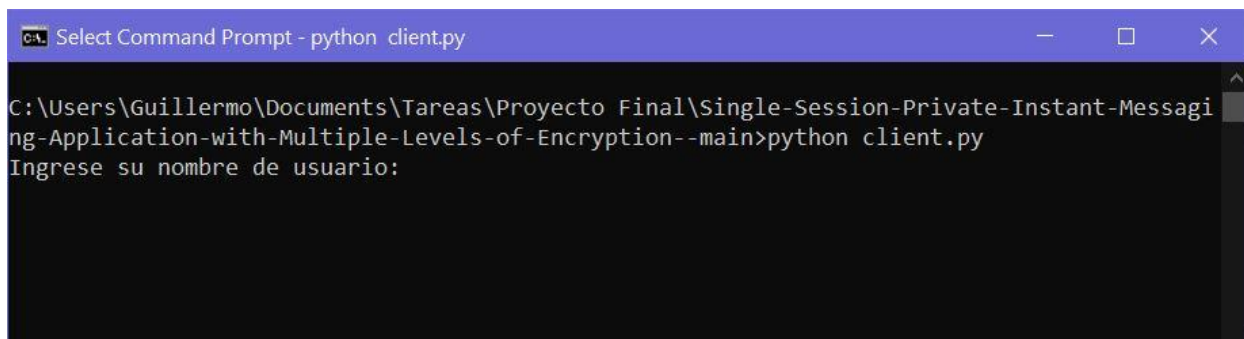
El usuario tiene dos opciones para ejecutar el cliente y conectarse al servidor:

1. Utilizando la versión de consola
2. Utilizando la versión con interfaz grafica

VERSIÓN CONSOLA

Para ejecutar la versión consola del cliente, el usuario deberá abrir CMD y dirigirse a la carpeta donde descargó el código del programa y ejecutar el siguiente comando:

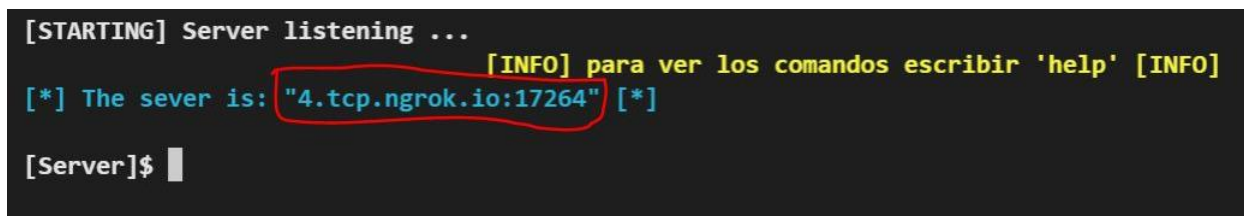
Python client.py



```

C:\Users\Guillermo\Documents\Tareas\Proyecto Final\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main>python client.py
Ingrese su nombre de usuario:
  
```

El programa pedirá al cliente que escriba su nombre de usuario que será utilizado en el chat. Luego le pedirá que ingrese el link del servidor al cual desea conectarse y por último le pedirá que ingrese la clave. La siguiente imagen muestra el link del servidor que está siendo utilizado en este ejemplo:



```

[STARTING] Server listening ...
[INFO] para ver los comandos escribir 'help' [INFO]
[*] The sever is: 4.tcp.ngrok.io:17264 [*]
[Server]$
  
```

El link deberá ser compartido con los usuarios que quieran ingresar al chat. En nuestro ejemplo, serán Juan y Pepe.

```
PS C:\Users\Guillermo\Documents\Tareas\Proyecto Final\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main> python client.py
Ingrese su nombre de usuario: Juan
Ingrese la dirección entregada por el servidor: 4.tcp.ngrok.io:17264
```

```
PS C:\Users\Guillermo\Documents\Tareas\Proyecto Final\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main> python client.py
Ingrese su nombre de usuario: Juan
Ingrese la dirección entregada por el servidor: 4.tcp.ngrok.io:17264
Ingrese la clave acordada entre los clientes: hola
```

```
*****
*                               *
*                               *
*                               *
*                               *
*                               *
*****
```

```
Ejemplo          acion, los mensajes que envíe el usuario se pond
Ejemplo
Los mensajes que reciba de otro cliente se pondrán en morado:
Ejemplo
```

```
[INFO] para ver los comandos escribir '/help' [INFO]
```

```
> Llave simétrica desencriptada exitosamente!
b'Juan' ha entrado al chat!
Connected to the server.
```

```
*****
*                               *
*                               *
*                               *
*                               *
*                               *
*****
```

```
Para facilidad en la visualización, los mensajes que envíe el usuario se pondrán de color amarillo:
```

```
Ejemplo
Los mensajes que reciba de otro cliente se pondrán en morado:
Ejemplo
```

```
[INFO] para ver los comandos escribir '/help' [INFO]
```

```
> Llave simétrica desencriptada exitosamente!
b'Juan' ha entrado al chat!
Connected to the server.
```

```
> Hola soy juan
> [Juan]: Hola soy juan
```



```
PS C:\Users\Guillermo\Documents\Tareas\Proyecto Final\Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption--main> python client.py
Ingrese su nombre de usuario: pepe
Ingrese la dirección entregada por el servidor: 4.tcp.ngrok.io:17264
Ingrese la clave acordada entre los clientes: hola

*****
*                                     *
*                               DISCLAIMER                               *
*                                     *
*****

Para facilidad en la visualizacion, los mensajes que envíe el usuario se pondran de color amarillo:
Ejemplo
Los mensajes que reciba de otro cliente se pondran en morado:
Ejemplo

[INFO] para ver los comandos escribir '/help' [INFO]
> Llave simetrica descriptada exitosamente!
b'pepe' ha entrado al chat!
Connected to the server.
```

```
> Llave simetrica descriptada exitosamente!
b'pepe' ha entrado al chat!
Connected to the server.
```

```
> hola soy pepe!
> [pepe]: hola soy pepe!
█
```

```
[INFO] para ver los comandos escribir '/help' [INFO]
> Llave simetrica descriptada exitosamente!
b'pepe' ha entrado al chat!
Connected to the server.

> hola soy pepe!
> [pepe]: hola soy pepe!
[Juan]: Hola pepe
█
```

Como se puede apreciar en el ejemplo anterior, Juan y Pepe, cada uno por separado inició la aplicación de cliente por consola e ingresaron al chat utilizando el link del servidor y la clave. Los mensajes enviados por el usuario se mostrarán de color amarillo y los mensajes que reciba serán de color morado.

VERSIÓN CON INTERFAZ GRAFICA

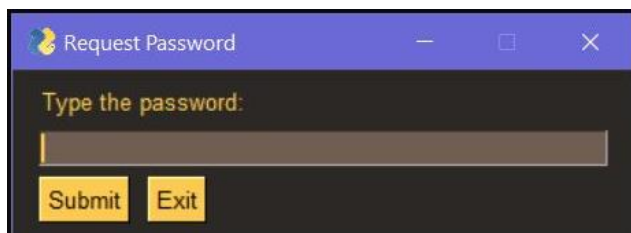
Para iniciar el cliente con interfaz gráfica, el usuario puede dar doble click al archivo “clientGUI.py” o también tiene la opción de ejecutar el comando “python clientGUI.py” vía CMD.

De igual manera que la versión de consola, se le pedirá al cliente que ingrese su nombre de usuario, el link del servidor y la clave para poder entrar al chat de manera exitosa.

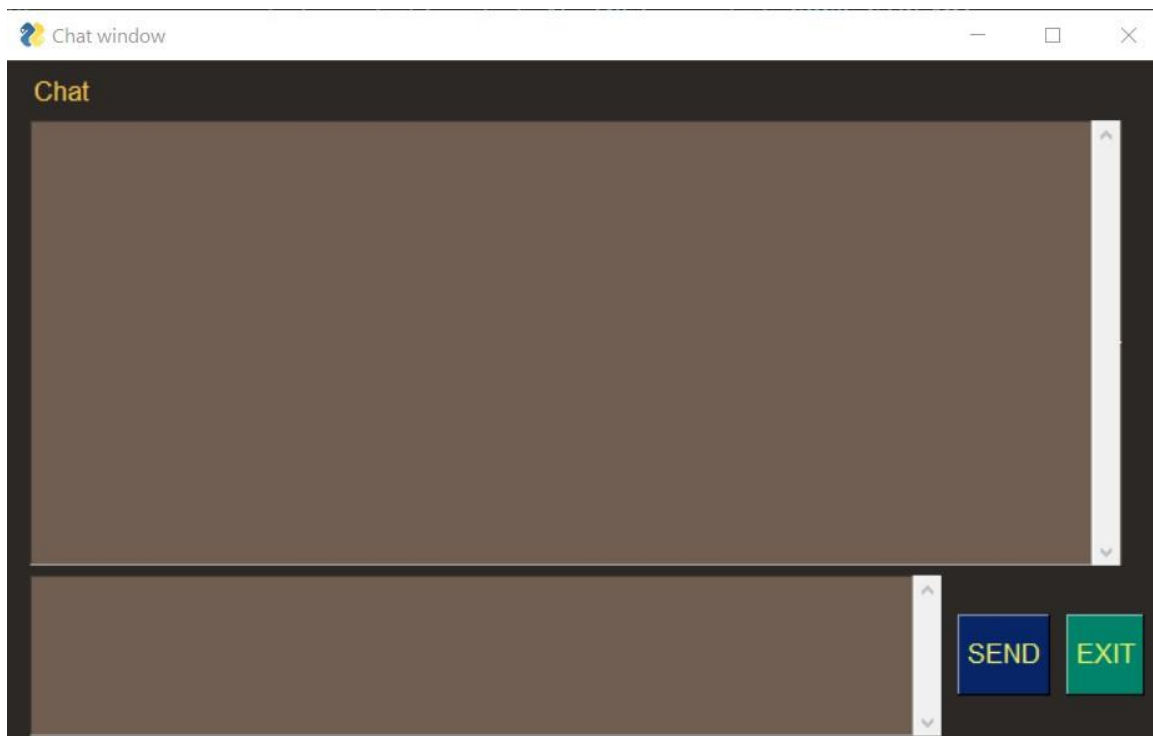
Ver ejemplo a continuación.



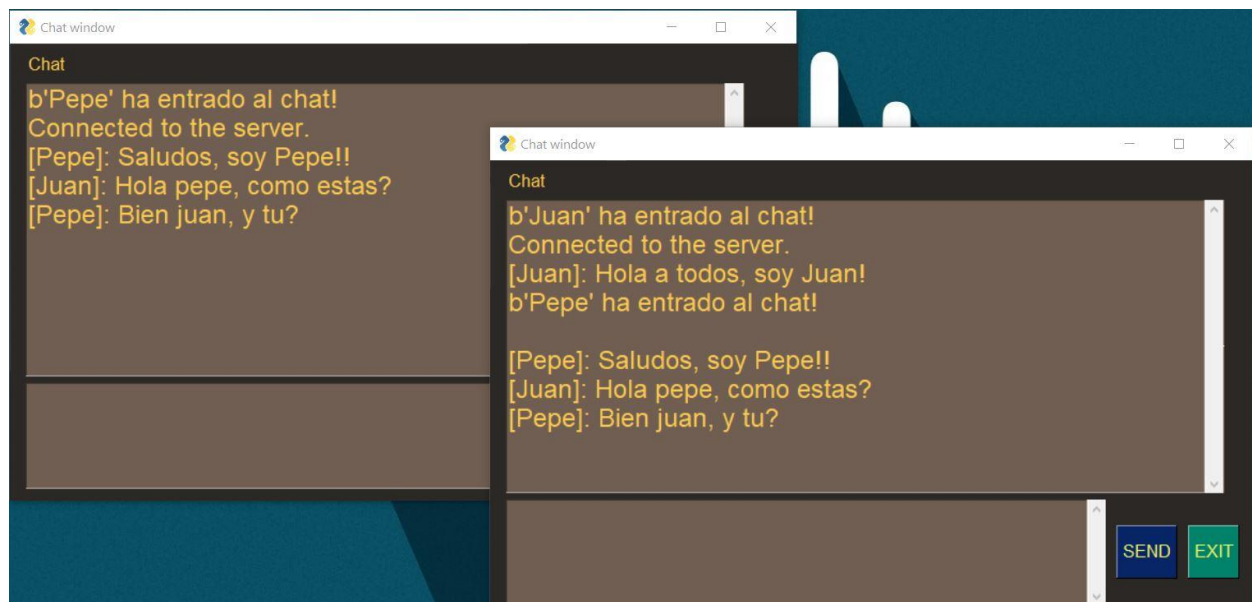
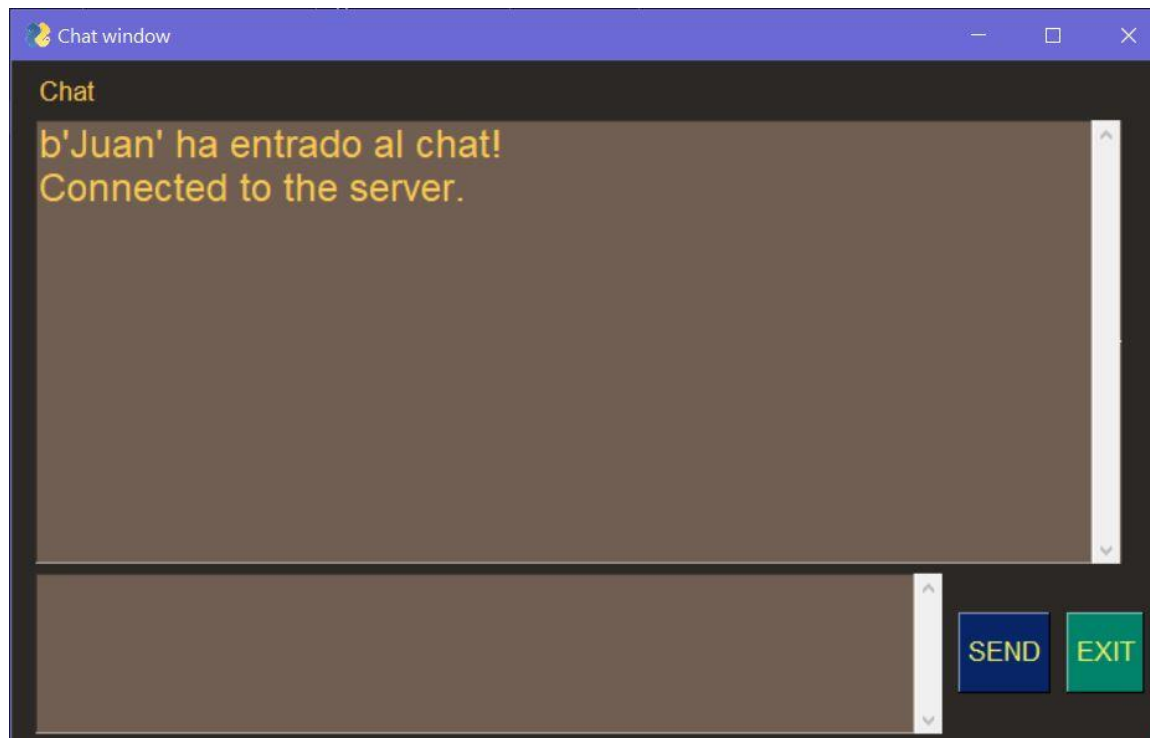
A screenshot of a window titled "SignIn" with a blue header bar. The window has a dark background. It contains two text input fields with yellow labels: "Write your username:" and "Write the address provided by the server:". Below the first input field is a yellow "Submit" button and a yellow "Exit" button. At the bottom of the window is a third, empty text input field.



A screenshot of a window titled "Request Password" with a blue header bar. The window has a dark background. It contains a single text input field with a yellow label: "Type the password:". Below the input field are two yellow buttons: "Submit" and "Exit".



A screenshot of a window titled "Chat window" with a blue header bar. The window has a dark background. It features a large text area for chat messages with a vertical scrollbar on the right. At the bottom of the window is a text input field for sending messages, also with a vertical scrollbar on the right. To the right of the input field are two buttons: a blue "SEND" button and a green "EXIT" button.



Como se puede apreciar, los clientes pueden intercambiar mensajes utilizando el botón de SEND y para salir del chat usaran el botón EXIT.

Nota: Los diferentes clientes podrán usar la versión que más les guste, ya sea por consola o con interfaz gráfica. El servidor soportara ambas versiones de manera simultánea sin inconvenientes.



ENLACE AL REPOSITORIO DEL CÓDIGO FUENTE

- <https://github.com/Banano-Cry/Single-Session-Private-Instant-Messaging-Application-with-Multiple-Levels-of-Encryption->

