Transferencia de archivos con chequeo de integridad con clave pública.

Santiago Jose Barraza Sinning
Luisa Fernanda Castaño Pino
Juan Eduardo Yustes Nieto

Cali, valle del cauca
Universidad ICESI
Ciberseguridad
2025

Índice:

Introducción

Este proyecto consiste en la implementación de una aplicación de transferencia segura basada en el modelo cliente-servidor, desarrollada en Java utilizando JavaFX para la interfaz gráfica. El objetivo principal fue establecer un canal de comunicación segura entre un cliente y un servidor mediante el uso de claves públicas y privadas.

Desarrollo del proyecto

Proceso de desarrollo

El desarrollo del proyecto se realizó en varias etapas, siguiendo una lógica progresiva desde la planificación, el desarrollo de la base con las funciones clave, hasta la ejecución y pruebas de la funcionalidad, resumido en:

1. **Diseño del modelo cliente-servidor:**
   Se planteó una arquitectura básica en la cual el servidor estaría encargado de recibir y procesar el archivo enviado por el cliente, y este último de enviar el archivo cifrado, inicialmente sin interfaz, solo probando la funcionalidad base de enviar archivos, se utilizó Java como lenguaje de programación, además implementando sockets para establecer la comunicación entre ambos extremos.
2. **Generación de claves públicas y privadas RSA y clave aleatoria AES:**
   Para asegurar la confidencialidad de los datos, se utilizó cifrado asimétrico RSA. Se creó una clase dedicada para generar las claves públicas y privadas:

   CryptoUtils:

   ```
   public static KeyPair generateRSAKeyPair() throws Exception {
       KeyPairGenerator keyGen = KeyPairGenerator.getInstance(algorithm:"RSA");
       keyGen.initialize(keysize:2048);
       return keyGen.generateKeyPair();
   }
   ```

   ServerApp:

   ```
   callback.onProgress(message:"🔐 Generando par de claves RSA-2048...", progress:0.2);
   java.security.KeyPair keyPair = crypto.CryptoUtils.generateRSAKeyPair();
   java.security.PublicKey publicKey = keyPair.getPublic();
   java.security.PrivateKey privateKey = keyPair.getPrivate();
   ```

   y se almacenaron en archivos para que pudieran ser cargadas por el cliente y el servidor:

```
crypto.CryptoUtils.saveKey(path:"server_files/public_key_server.pem", publicKey);
crypto.CryptoUtils.saveKey(path:"server_files/private_key_server.pem", privateKey);
callback.onProgress(message:"🔑 Par de claves RSA generado y guardado", progress:0.3);
```

Esto permitió que el mensaje enviado por el cliente solo pudiera ser leído por el servidor correspondiente.

Además de esto, se generó una clave aleatoria de 256 bits AES, para enviarla cifrada al servidor con la clave pública creada anteriormente del servidor, para que solo este último sea el que pueda descifrarla:

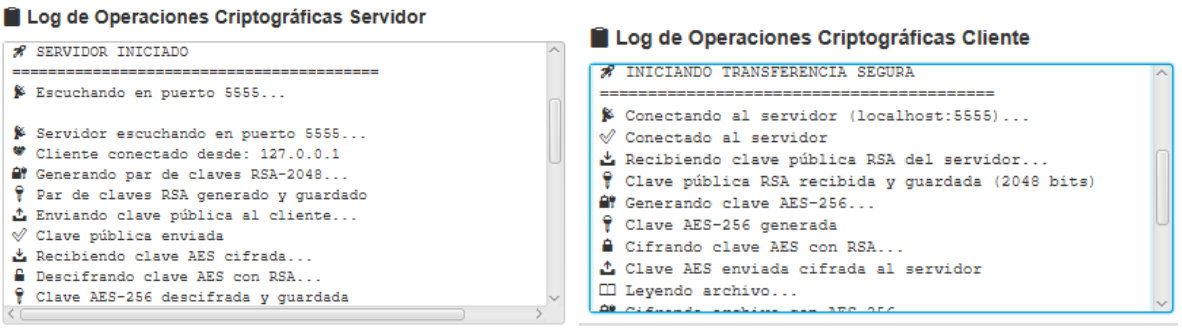
CryptoUtils:

```
public static SecretKey generateAESKey() throws Exception {
    KeyGenerator keyGen = KeyGenerator.getInstance("AES");
    keyGen.init(256);
    return keyGen.generateKey();
}
```

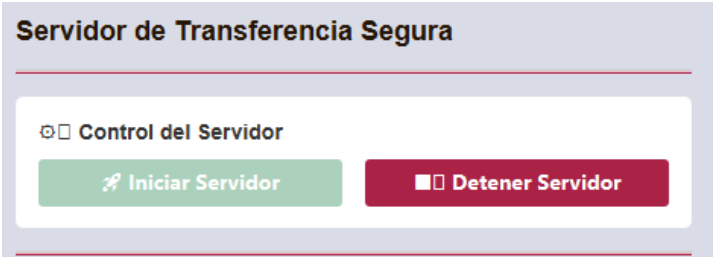
ClientApp:

```
callback.onProgress(message:"🔐 Generando clave AES-256...", progress:0.5);
javax.crypto.SecretKey aesKey = crypto.CryptoUtils.generateAESKey();
byte[] aesBytes = aesKey.getEncoded();
crypto.CryptoUtils.saveToFile(path:"client_files/aes_key.hex", aesBytes);
callback.onProgress(message:"🔑 Clave AES-256 generada", progress:0.55);
```

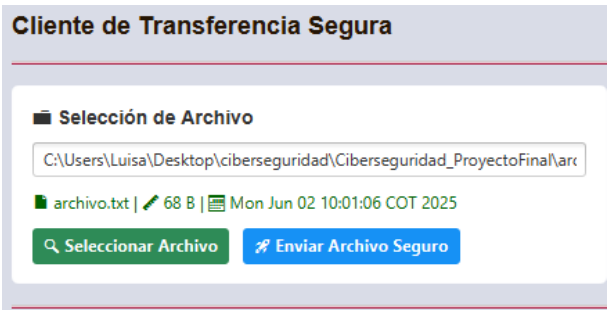
3. **Implementación de la lógica de envío y recepción:**
   Entre las funciones mencionadas, de recibir y procesar el archivo enviado por el cliente, en general la lógica implementada es que el servidor espera a la conexión, se conecta el cliente, desde el servidor se generan un par de claves RSA, se le envía la pública al cliente y el cliente genera una clave AES y se la manda al servidor cifrada con la clave pública, de esta manera se garantiza que el servidor pueda decodificarla con su clave privada, y así mismo el cliente envía al servidor un archivo cifrado con la clave AES, como ya el servidor puede decodificarla, el archivo se guarda correctamente y se calcula el hash por el cliente se le manda al servidor, que también calcula el has y así verificar la igualdad en estos dos hashes que indicarían que contienen el mismo mensaje, lo que quiere decir que se transfirió correctamente el archivo de cliente a servidor. A continuación, un ejemplo de los logs de la interfaz creada:



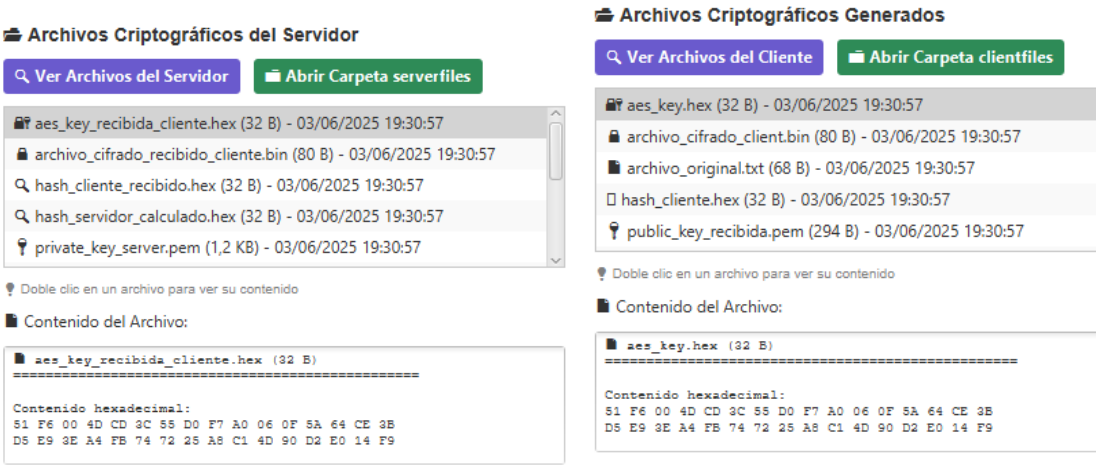
4. **Construcción de la interfaz gráfica con JavaFX:**
   Se diseñó una interfaz visual utilizando JavaFX para facilitar la interacción con el programa, creándose un botón para poder iniciar el servidor y también para poder detener el servidor:



un área de texto tipo consola donde se muestran los logs de lo que ocurre, cuando se crean claves, cuando se envían, etc, además botones para buscar el archivo a enviar, botón para enviar archivo:



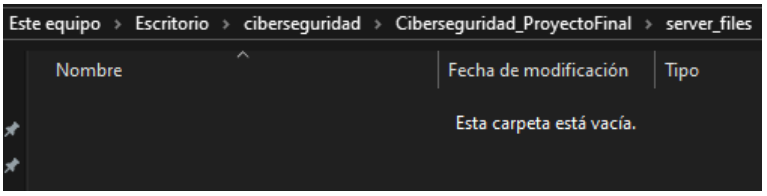
hay una sección para ver los archivos guardados en las carpetas correspondientes, es decir la carpeta para cliente y carpeta para servidor, así mismo una sección para poder ver el contenido y asegurarse desde la interfaz que se hayan transferido correctamente los archivos:

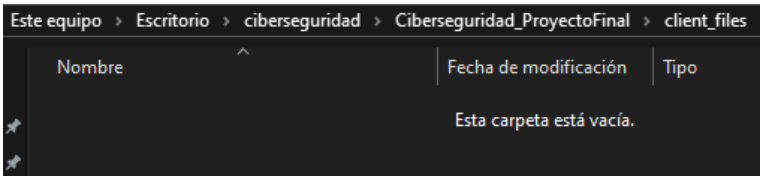


Se utilizó diseño en VBox y HBox para organizar los elementos, y se aplicaron estilos básicos como colores, tamaño de fuente, y espaciado.

5. **Pruebas y ajustes finales:**
   Se realizaron pruebas de comunicación entre cliente y servidor, para verificar que el cifrado y descifrado funcionaban correctamente, así como la transferencia de archivos como el seleccionado por el cliente como por las claves. También se ajustó el diseño visual para que la interfaz fuera más clara y funcional:

   Inicialmente se puede verificar que las carpetas están vacías:





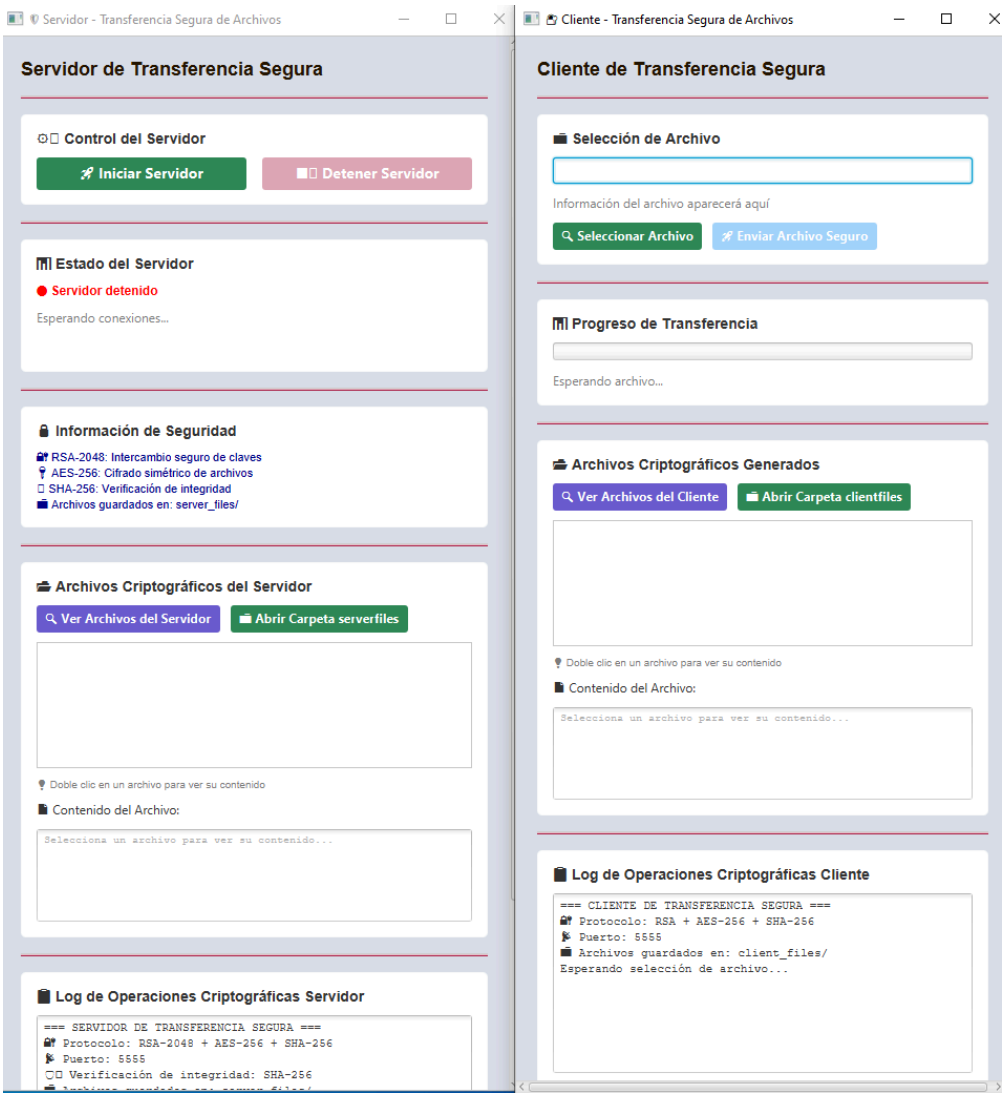
   Se inicia el servidor:

```
PS C:\Users\Luisa\Desktop\ciberseguridad\Ciberseguridad_ProyectoFinal> mvn clean javafx:run -Pserver
[INFO] Scanning for projects...
[INFO]
[INFO] ------< com.proyecto.ciberseguridad:Ciberseguridad_ProyectoFinal >------
[INFO] Building Ciberseguridad_ProyectoFinal 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[WARNING] 6 problems were encountered while building the effective model for org.openjfx:javafx-controls:jar:
21.0.1 during dependency collection step for project (use -X to see details)
```

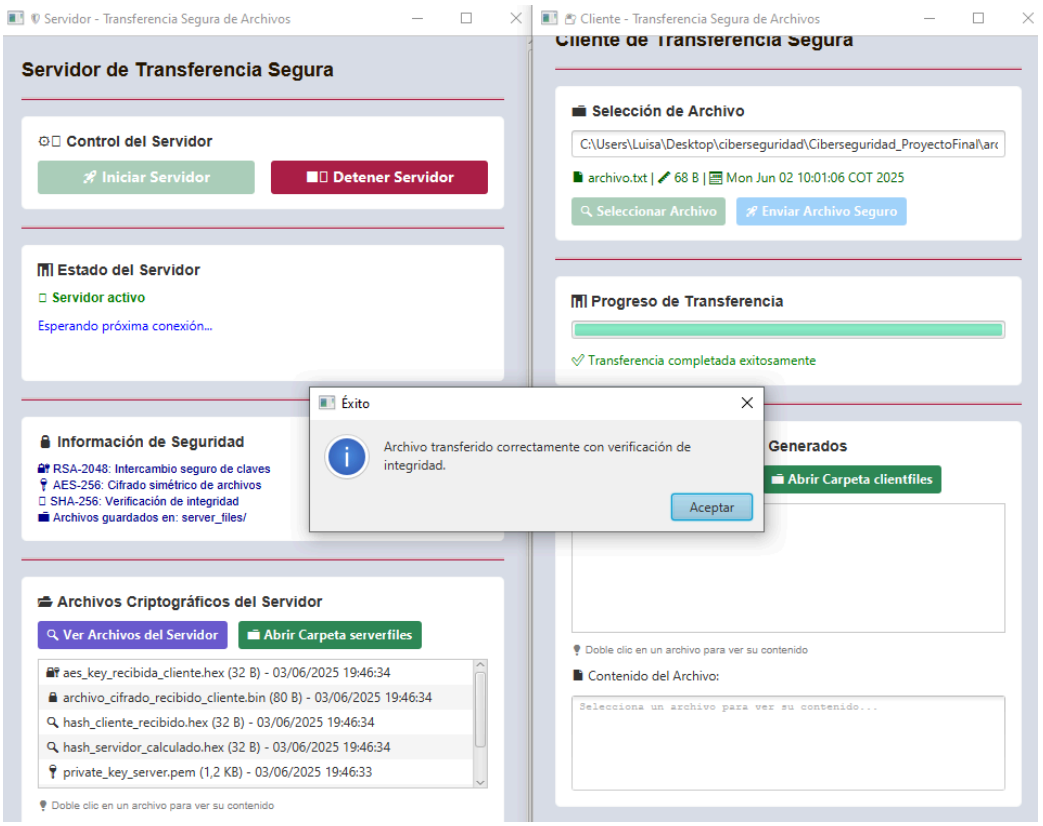
   Y en otra terminal se inicia el cliente:

```
PS C:\Users\Luisa\Desktop\ciberseguridad\Ciberseguridad_ProyectoFinal> mvn clean javafx:run -Pclient
[INFO] Scanning for projects...
[INFO]
[INFO] ------< com.proyecto.ciberseguridad:Ciberseguridad_ProyectoFinal >------
[INFO] Building Ciberseguridad_ProyectoFinal 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[WARNING] 6 problems were encountered while building the effective model for org.openjfx:javafx-controls:jar:
21.0.1 during dependency collection step for project (use -X to see details)
[INFO]
[INFO] --- clean:3.2.0:clean (default-clean) @ Ciberseguridad_ProyectoFinal ---
```

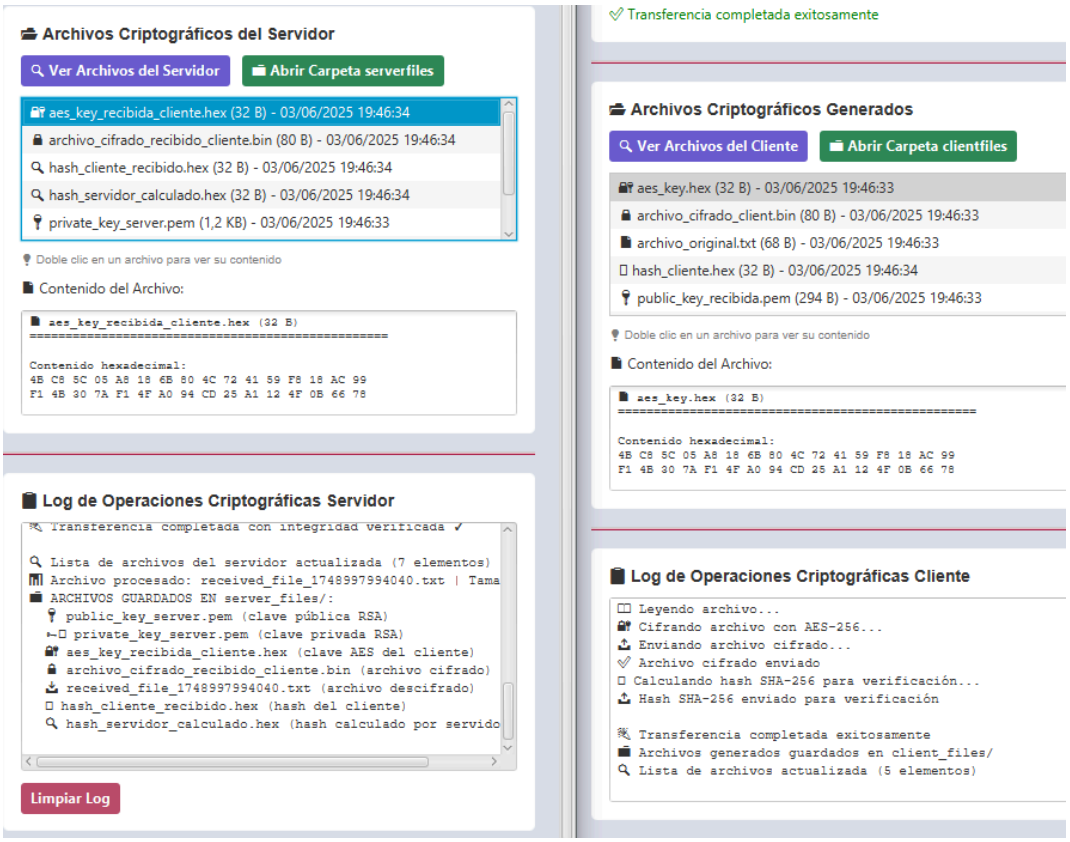
Y de acuerdo a esto, aparecen las interfaces:



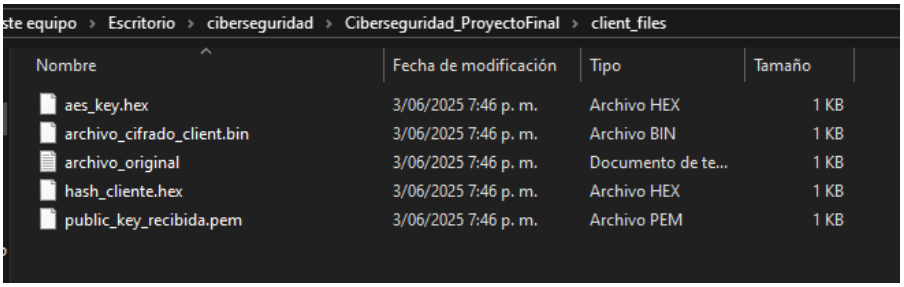
Se inicia el servidor, se selecciona el archivo y se transfiere:

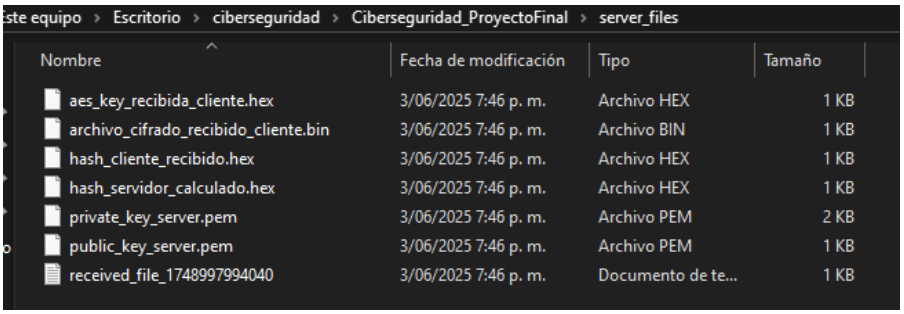


Se comprueba la transferencia:



Y que así mismo, ahora las respectivas carpetas cuentan con los archivos correctamente generados, enviados y recibido:





Uso de herramientas de IA

Durante el desarrollo del proyecto, se emplearon herramientas de inteligencia artificial como apoyo puntual para resolver dudas técnicas específicas y obtener sugerencias de diseño. Estas herramientas se utilizaron como acompañamiento para fortalecer el desarrollo y no como generadores directos del código final y la solución del sistema.

En concreto, se utilizaron dos herramientas principales:

- **ChatGPT:**
  Esta IA fue utilizada especialmente para comprender e implementar funciones relacionadas con la generación y uso de claves públicas y privadas en Java (RSA y AES). Se buscó orientación sobre cómo emplear clases que se generaron en nuestro CryotoUtils.Java, como KeyPairGenerator, KeyGenerator, y cómo aplicar el cifrado y descifrado de mensajes Cipher

  Prompts utilizados:
  ¿Cómo se generan claves públicas y privadas en Java usando RSA?

  ¿Cómo se genera una clave AES y se utiliza para cifrar y descifrar datos en Java?

  ¿Cómo se guardan las claves generadas?

- **Claude:**
  Se utilizó para obtener sugerencias de diseño de la interfaz gráfica y estructura general de la comunicación cliente-servidor. Claude en general nos ayudó a visualizar la organización de botones, áreas de texto, eventos y otros elementos dentro del

entorno JavaFX, mejorando la usabilidad y fluidez de la aplicación.

Prompts utilizado:
¿Cómo puedo hacer una interfaz de cliente y servidor en JavaFX con un área de texto y botones?

¿Cómo puedo cambiar los colores usados en los botones?

¿Cómo puedo personalizar el color usado alrededor de los recuadros?

Las respuestas de ambas herramientas nos sirvieron como base de consulta para aclarar conceptos, validar enfoques y adaptar el código de forma personalizada a los requerimientos específicos del proyecto. Las soluciones no fueron copiadas directamente, sino adaptadas y ajustadas críticamente según nuestras necesidades, de esta manera tratando de hacer un uso responsable de la inteligencia artificial para permitir una mejora en la comprensión técnica y en el diseño de la solución.

Dificultades

Durante el desarrollo del proyecto, surgieron varios retos técnicos y conceptuales que requieren investigación adicional, pruebas y ajustes. A continuación, se detallan las principales dificultades encontradas:

- **Sincronización entre cliente y servidor para asegurar una correcta comunicación:**
  Uno de los mayores retos fue asegurar que la comunicación entre el cliente y el servidor fuese fluida, en general la gestión de flujos de entrada y salida (Input/Output Streams), junto con el envío de archivos cifrados, fue un poco una dificultad.

- **Alineación y diseño visual en JavaFX (centrado de elementos, estilos, etc.):**
  Diseñar una interfaz que no solo fuera funcional, sino también clara, supuso una dificultad adicional. JavaFX ofrece múltiples opciones de diseño (como VBox, HBox, BorderPane, etc.), y lograr el centrado adecuado de elementos, aplicar estilos personalizados, y responder correctamente a eventos del usuario tomó tiempo y pruebas constantes.

- **Manejo de errores y validaciones al momento de cifrar/descifrar mensajes:**
  En el proceso de cifrado y descifrado, pequeños errores como el uso de claves incorrectas, formatos de codificación mal aplicados o flujos de datos interrumpidos, generaban fallos que no siempre eran evidentes, por eso se recurrió a implementar validaciones y manejo de excepciones.

Conclusiones

El desarrollo de este proyecto nos permitió aplicar y reforzar conocimientos fundamentales vistos durante algunas clases de la materia, conceptos como RSA y AES y a través de este trabajo, comprendimos mejor cómo se generan, comparten y utilizan las claves públicas, privadas y simétricas, y cómo estos mecanismos permiten garantizar una comunicación segura entre dos puntos. La construcción de este proyecto nos ayudó a visualizar de forma concreta cómo se pueden aplicar estos métodos criptográficos en situaciones reales.

Por otro lado, el uso de JavaFX nos permitió diseñar una interfaz gráfica funcional, que aunque presenta retos en cuanto al diseño, logró cumplir con los objetivos planteados. En general, esta experiencia no solo fortaleció nuestras habilidades técnicas, sino que también nos permitió conectar los contenidos teóricos de clase con soluciones reales.

Link del proyecto:

https://github.com/luisapino/Ciberseguridad_ProyectoFinal