Sistema de Conversação

Projeto A3 - Chat Seguro

100% Criptografado

EQUIPE:

Antonio Rildo

Daniel Henrique

Kleberson Galdino

José Brito

Guilherme

DOCENTE:

Prof. Alberlan

Tecnologias Fundamentais

Protocolo TCP

Utilizamos o protocolo **TCP** para garantir estabilidade.

- **Sem perdas:** Mensagens chegam íntegras.
- **Ordenado:** O texto aparece na ordem certa.
- **Conexão contínua:** O chat não cai entre mensagens.

Sockets

A tecnologia base da comunicação.

- **Porta de Entrada:** Onde os dados entram e saem.
- **Endereçamento:** Usa IP e Porta (ex: 5454).
- Permite a troca real de dados pela rede.

Conversas Simultâneas

Como funciona o Paralelismo?

Para que vários usuários conversem ao mesmo tempo, o sistema usa processos paralelos.

- **Múltiplos Usuários:** O servidor atende a todos simultaneamente.
- **Sem Travamentos:** Você pode digitar enquanto recebe mensagens.
- Alta eficiência no processamento de dados.

Segurança e Motivação

Problema e Solução

A criptografia foi adicionada para evitar que as mensagens fossem lidas por pessoas não autorizadas.

Problema anterior: Tudo era enviado em texto comum. Se alguém interceptasse a rede (ou o próprio servidor), conseguiria ler todo o conteúdo.

Objetivo: Garantir privacidade total no chat.

Antes vs. Depois

- **Antes:** O servidor via e processava texto legível ("Olá!").
- **Agora:** O servidor só transporta "lixo" criptografado ("gAAAAAB...").
- **Resultado:** Segurança ponta a ponta.

🔑 Chave Simétrica (Symmetric Key)

O sistema usa o modelo de **Chave Simétrica**.

- **Uma chave para tudo:** A mesma chave (arquivo `key.key`) é usada para **trancar** (criptografar) e **destrancar** (descriptografar).
- **Segurança Física:** O arquivo da chave fica *apenas* com os

Algoritmo Fernet

Utilizamos a implementação **Fernet** da biblioteca `cryptography` do Python.

```
from cryptography.fernet import Fernet # 1. Carrega a chave key = open('key.key', 'rb').read() f = Fernet(key) # 2.
```

O Caminho da Mensagem Segura

1. Emissor

Digita "Oi". O computador usa a chave para embaralhar: `gAAAA...`

2. Servidor

Recebe `gAAAA...`. Não entende nada, apenas repassa para os outros.

3. Receptor

Recebe `gAAAA...`. Usa sua chave para transformar de volta em "Oi".

✓ Isso garante a integridade e confidencialidade dos dados.

Funcionalidades do Sistema

Registrar Clientes

Armazena conexões e nomes em listas seguras (`clients` e `nomes`) para controle total de acesso.

Transmissão Geral

Sistema de **Broadcast**: Mensagem enviada por um usuário chega instantaneamente para todos os outros.

Envio e Recebimento

Comunicação bidirecional fluida usando funções de rede otimizadas (`send` / `recv`).

Controle de Acesso

Gerencia automaticamente entrada e saída. Detecta desconexões e remove usuários inativos.

Servidor: Inicialização

Configuração Base

O servidor prepara o ambiente seguro para conexões.

- **Socket TCP:** Garante a integridade.
- **Bind:** Vincula ao endereço local e porta.
- **Listas:** Prepara o registro dos participantes.

```
import threading import socket # Configuração do Pro
```

Servidor: Sistema de Broadcast

```
def transmissao(msg, remetente=None): for client in
```

Distribuição de Mensagens

A função responsável por espalhar a conversa.

- Recebe a mensagem criptografada.
- Envia para todos na lista de `clients`.
- **Anti-Eco:** Não envia a mensagem de volta para quem escreveu.

Gerenciamento de Usuários

Controle de Fluxo

Monitora cada usuário individualmente.

- **Escuta Ativa:** Aguarda mensagens a todo momento.
- **Comando "TT":** Detecta se o usuário quis sair.
- **Segurança:** Remove conexões instáveis ou encerradas.

```
def tratar(client): while True: try: msg = client.recv(1024) if m
```

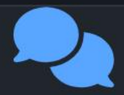
Fluxo de Funcionamento

1. Servidor

Inicia e aguarda conexões seguras.

2. Conexão

Cliente entra e registra seu nome.



3. Chat

Mensagens são trocadas simultaneamente.

4. Segurança

Dados criptografados e saída segura.

Projeto Concluído

Desenvolvemos um **Sistema de Conversação** completo, estável e seguro.

A aplicação de **Sockets TCP** e **Criptografia** garante uma comunicação eficiente e protegida para todos os usuários.

Referências Bibliográficas

Canal NeuralNine

Fonte principal de inspiração para a arquitetura do chat e implementação dos sockets em Python.

Documentação Oficial Python

docs.python.org - Bibliotecas `socket` e `threading`.

Obrigado!