Enunciado do Projeto: Sistema de Chat com Deteção de Dados Pessoais (GDPR) para Multiusuários em Python

Duração: 2+ Semanas

Objetivo:

Desenvolver um sistema de chat multiusuários utilizando Python, com funcionalidades de deteção de dados pessoais (em conformidade com o GDPR) e engenharia social. O sistema permitirá que vários utilizadores se conectem a um servidor e interajam uns com os outros em tempo real, com a capacidade de identificar e tratar mensagens que contenham dados pessoais sensíveis, como e-mails, números de telefone, etc.

Requisitos Funcionais:

Servidor:

- O servidor deve ser capaz de aceitar múltiplas conexões simultâneas de clientes utilizando *threading*.
- O servidor precisa manter uma lista de todos os clientes conectados e garantir que as mensagens sejam repassadas para todos os outros utilizadores.
- O servidor deve ser capaz de detetar dados pessoais nas mensagens enviadas pelos clientes (como e-mails, telefones, etc.), alertando ou bloqueando a comunicação quando necessário.
- O servidor precisa enviar alertas aos clientes caso dados pessoais sejam encontrados nas suas mensagens.

Cliente:

- O cliente deve ser capaz de se conectar ao servidor e enviar mensagens de texto.
- O cliente deve exibir as mensagens recebidas do servidor e permitir ao utilizador enviar mensagens.
- O cliente deve permitir que o utilizador digite mensagens, enviando-as para o servidor.
- O cliente deve alertar o utilizador se uma mensagem enviada for bloqueada devido à deteção de dados pessoais sensíveis.
- O cliente deve permitir o envio de comandos especiais, como "exit", para se desconectar do servidor.

Deteção de Dados Pessoais (GDPR):

- O sistema deve ser capaz de detetar e alertar para a presença de dados pessoais em mensagens, como:
    - E-mails
    - Números de telefone
    - Endereços IP
    - Nomes completos

    - Datas de nascimento
    - Cartões de crédito (se necessário)
- Se dados pessoais forem identificados, a mensagem deve ser bloqueada ou um alerta deve ser enviado ao utilizador.
- O sistema deve garantir que dados pessoais não sejam armazenados ou processados sem a devida autorização.

Engenharia Social:

- O servidor pode registar tentativas suspeitas de engenharia social, onde padrões comuns de manipulação ou informações pessoais excessivas são detetadas nas mensagens e guardadas.

Multiusuários:

- O sistema deve permitir que múltiplos utilizadores se conectem ao servidor simultaneamente.
- O servidor deve ser capaz de gerenciar *threading*, garantindo que cada utilizador tenha uma conexão separada para enviar e receber mensagens em tempo real, sem bloquear os outros.

Interface Simples de Texto CLI, (Interface Gráfica é opcional):

- O cliente e servidor terão uma interface de linha de comando simples para interações via terminal.
- O servidor deve exibir *logs* sobre novos clientes conectados ou desconectados e sobre o bloqueio de mensagens que contenham dados pessoais.
- O cliente deve exibir mensagens recebidas e alertas quando houver violação de privacidade.

Requisitos Técnicos:

Tecnologias e Bibliotecas:

- **Python 3**: Linguagem principal para desenvolvimento.
- **Módulo socket**: Para a comunicação entre o cliente e o servidor.
- **Threading**: Para suportar múltiplos utilizadores simultaneamente.
- **Expressões regulares (regex)**: Para a deteção de dados pessoais nas mensagens.
- **Bibliotecas de *logging***: Para registar eventos de conexão, desconexão e tentativas de violação de privacidade.

Funcionamento do Sistema:

- **Servidor**: Um único servidor central que aguarda conexões de clientes. Ele aceita múltiplas conexões, e quando um cliente envia uma mensagem, o servidor a repassa para todos os outros clientes conectados e ou em comunicação direta entre dois utilizadores ou grupo de utilizadores, com a verificação de dados pessoais.
- **Cliente**: A interface no terminal para permitir que os utilizadores enviem e recebam mensagens. A comunicação é feita em tempo real.

Cronograma de Desenvolvimento (2 Semanas):

Semana 1: Planejamento, Estruturação e Desenvolvimento Inicial

Objetivo: Planejar o sistema, estruturar o código e desenvolver a comunicação básica entre cliente e servidor, além de implementar a deteção de dados pessoais nas mensagens.

Tarefas:

Planejamento e Estruturação do Projeto (Dia 1):

- Definir a arquitetura geral do sistema: cliente, servidor e deteção de dados pessoais.
- Planear o fluxo do chat, incluindo a interação entre cliente e servidor, e como a verificação de dados pessoais será realizada.
- Definir os tipos de dados pessoais a serem detetados nas mensagens e como o sistema vai bloquear ou alertar os utilizadores.

Implementação do Servidor (Dia 2-4):

- Criar o servidor de chat básico utilizando o módulo *socket*.
- Implementar a aceitação de múltiplas conexões de clientes com *threading*.
- Criar uma função para enviar mensagens recebidas de um cliente para todos os outros.
- Implementar a deteção de dados pessoais em mensagens utilizando expressões regulares.
- Implementar o bloqueio de mensagens ou alertas caso dados pessoais sejam encontrados.
- Adicionar *logs* no servidor para monitorar clientes conectados e desconectados.

Implementação do Cliente (Dia 5-7):

- Criar a interface do cliente com a capacidade de se conectar ao servidor e enviar mensagens.
- Adicionar a capacidade de exibir mensagens recebidas do servidor.
- Implementar a função de envio de mensagens e comandos especiais (como "exit").
- Implementar alertas para o utilizador caso a mensagem enviada seja bloqueada devido à deteção de dados pessoais.

Semana 2: Funcionalidades Adicionais, Testes e Finalização

Objetivo: Implementar funcionalidades adicionais, realizar testes de desempenho e usabilidade, e documentar o projeto.

Tarefas:

Gerenciamento de Conexões e Desconexões (Dia 8-10):

- Adicionar a funcionalidade de registar quando um cliente se desconectar.
- Implementar a lógica para notificar os outros utilizadores sobre a entrada e saída de clientes.

- Garantir que mensagens enviadas antes da desconexão sejam processadas corretamente.

Melhorias no Cliente (Dia 10-12):

- Melhorar a interface do cliente, permitindo ao utilizador digitar várias mensagens sem sair do loop de envio.
- Implementar tratamento de erros de desconexão e falha de comunicação.
- Ajustar o sistema de alertas e bloqueios de mensagens para garantir um feedback rápido e claro ao utilizador.

Testes e Ajustes Finais (Dia 12-14):

- Realizar testes de *stress*, conectando múltiplos clientes simultâneos (5-10 clientes) e verificando o desempenho do servidor.
- Testar a deteção de dados pessoais e garantir que o sistema está funcionando corretamente em condições de uso real.
- Corrigir bugs e otimizar o código.

Documentação e Finalização (Dia 13-14):

- Escrever o arquivo **README.md**, explicando como configurar e executar o servidor e o cliente.
- Documentar a arquitetura do código e as principais funcionalidades de deteção de dados pessoais.
- Disponibilizar o código em um repositório GitHub num repositório privado e apenas partilhado comigo e elemento do grupo.

Resumo do Cronograma de Desenvolvimento (2 Semanas):

Semana	Objetivo Principal	Tarefas Principais
Semana 1	Planejamento e Implementação do Servidor e Cliente	- Planear o projeto e o fluxo de mensagens - Implementar o servidor básico com deteção de dados pessoais - Criar o cliente com envio e recebimento de mensagens
Semana 2	Funcionalidades Adicionais, Testes e Documentação	- Implementar gerenciamento de conexões/desconexões - Testar e corrigir bugs - Documentar o projeto e publicar

Critérios de Avaliação:

1. **Funcionalidade do Chat:**
    - O servidor deve ser capaz de gerir múltiplos clientes e enviar as mensagens para todos os outros conectados bem como mensagens direcionadas.
    - O cliente deve ser capaz de enviar e receber mensagens corretamente.
2. **Deteção de Dados Pessoais (GDPR):**

    - O servidor deve detectar dados pessoais nas mensagens e guardar quando necessário.
3. **Gerenciamento de Conexões:**
    - O servidor deve ser capaz de notificar os clientes quando um novo utilizador se conecta ou se desconecta.

4. **Multithreading e Desempenho:**
    - O sistema deve ser capaz de gerir múltiplos clientes simultaneamente sem comprometer o desempenho.
5. **Interface e Usabilidade:**
    - A interface do cliente deve ser clara e o sistema deve ser fácil de usar.
6. **Código Limpo e Organizado:**
    - O código deve ser bem estruturado, modularizado e documentado.

Entrega ao final:

- Servidor de chat funcional, aceitando múltiplos clientes e retransmitindo mensagens.
- Deteção de dados pessoais implementada e funcionando corretamente.
- Cliente de chat básico, enviando e recebendo mensagens.

- Documentação completa (**README.md**) explicando a implementação no GITHUB