📘 Documento Técnico — Servidor Python (server.py)

Autor: [Pedro Rodrigues]

Data: [13/10/2025]

Versão: 1.0

# 1. Objetivo do Sistema

Este script implementa um servidor híbrido WebSocket + HTTP que permite a troca de mensagens em tempo real entre clientes e um painel administrativo, além de expor APIs REST para gerenciamento de usuários, setores e mensagens.

# 2. Estrutura Geral do Arquivo

O arquivo está dividido em cinco blocos principais: configurações globais, manipulador WebSocket, funções auxiliares, inicialização do servidor e execução do script.

# 3. Configurações e Recursos

Define portas de conexão, diretórios temporários, e caminho do banco de dados SQLite usado para persistência de mensagens, usuários e setores.

# 4. Manipulação de Conexões WebSocket

A função handler gerencia conexões de clientes, recebimento e envio de mensagens e desconexões, garantindo controle por ID e registro de logs em tempo real.

# 5. Envio de Mensagens

A função broadcast\_message envia mensagens JSON (texto ou imagem) para todos os clientes conectados ou para um setor específico.

# 6. Servidor HTTP (API REST)

Implementado com aiohttp, o servidor expõe rotas administrativas e de controle, permitindo listar usuários, setores, enviar mensagens e realizar cadastros.

# 7. Inicialização e Execução

A função main cria os servidores WebSocket e HTTP, inicializa o banco de dados e ativa o modo terminal para o administrador enviar mensagens diretamente.

# 8. Execução e Encerramento

O bloco principal garante execução direta via terminal, tratamento de interrupções (Ctrl+C) e captura de erros inesperados.

# 9. Resumo de Funcionamento

O servidor atua como hub central de comunicação entre clientes (via WebSocket) e o painel administrativo (via HTTP), com armazenamento em SQLite.

# 10. Requisitos Técnicos

Python 3.7+, bibliotecas asyncio, websockets, aiohttp, sqlite3. Compatível com Linux e Windows, exige portas TCP abertas.

# 11. Considerações de Segurança

Recomenda-se uso de proxy reverso, autenticação em rotas /admin, sanitização de dados e HTTPS/TLS para segurança em produção.

# 12. Possíveis Extensões

Suporte a JWT, logs estruturados, painel web unificado e métricas em tempo real.

# 13. Conclusão

O server.py funciona como núcleo de comunicação em tempo real entre usuários e painel, integrando WebSocket, HTTP e banco SQLite em um único sistema.