

UNEMAT

Universidade do Estado do Mato Grosso

Carlos Alberto Reyes Maldonado

Campus Universitário Dep. Est. Renê Barbour - Barra do Bugres

Felipe F. da Silva, Gabriel L. Ribeiro, Luís Augusto F. Godoi  
{faria.felipe, gabriel.ribeiro1, godoi.luis}@unemat.br

**Chat Privado para Substituição do Ramal**

Barra do Bugres, Mato Grosso

2024

Felipe F. da Silva, Gabriel L. Ribeiro, Luís Augusto F. Godoi  
{faria.felipe, gabriel.ribeiro1, godoi.luis}@unemat.br

### **Chat Privado para Substituição do Ramal**

O Artigo da disciplina de Sistemas Lineares, com o tema de chat privado para substituição do ramal, apresenta um breve resumo sobre o artigo, dando uma introdução de tudo que será abordado ao longo do trabalho, citando trabalhos que inspiraram a sua criação, descrevendo sua arquitetura, quais elementos compõem o aplicativo, como ele foi desenvolvido, quais as dificuldades encontradas ao longo do caminho e concluindo tudo, refletindo sobre todas as informações que serão apresentadas até ali.

Barra do Bugres, Mato Grosso

2024

**Abstract.** *The communication carried out between large companies such as call centers, using devices like extensions, has become quite common in recent years, facilitating the work of many employees, including customer interactions. However, extensions have been experiencing significant problems for some time, showing signs of failures that can hinder all the work done by companies. Based on these drawbacks, the article proposes a substitute for extensions, a communication-focused application that allows not only interaction between employees and clients through messages but also document exchange. Developed in Python, utilizing sockets, and storing crucial data in a database, during its development, numerous challenges were encountered; fortunately, all were resolved, and the application demonstrated gratifying results, fulfilling exactly what was required. Moreover, its simplicity makes the application intuitive, enabling others to learn more about application development and how individual records in the database function. Additionally, its installation is straightforward, requiring only a network connection and the saved code on the machine.*

**Resumo.** *A comunicação realizada entre grandes empresas como a call centers, através de dispositivos como ramais, tem se tornado bem comum nos últimos anos, facilitando o trabalho de diversos funcionários, incluindo a parte de interação com os clientes. Todavia, os ramais vêm apresentando problemas significativos há algum tempo, demonstrando sinais de falhas que podem prejudicar todo o trabalho realizado pelas empresas. Com base nesses inconvenientes, o artigo propõe um substituto para o Ramal, um aplicativo focado em comunicação, que permite não só a interação entre os funcionários e clientes através de mensagens, como também a troca de documentos. Sendo desenvolvido na linguagem Python, utilizando os sockets, e armazenando os dados mais importantes em um banco de dados. Durante o seu desenvolvimento, muitas dificuldades foram sendo encontradas, felizmente, todas foram resolvidas e o aplicativo foi demonstrando resultados foram gratificantes, cumprindo exatamente o que era pedido. Além disso, graças a sua simplicidade, o aplicativo se torna bem intuitivo, permitindo que outras pessoas possam aprender mais sobre o desenvolvimento de aplicativos e como funcionam os registros de uma pessoa no banco de dados, inclusive, sua instalação é bem simples, exigindo apenas uma conexão a uma rede e o código salvo na máquina.*

## Sumário

<b>INTRODUÇÃO.....</b>	
<b>TRABALHOS RELACIONADOS.....</b>	
<b>DESCRIÇÃO DA PROPOSTA.....</b>	
<b>MATERIAIS E MÉTODOS.....</b>	
<b>RESULTADO E DISCUSSÃO.....</b>	
<b>DIFICULDADES ENCONTRADAS.....</b>	
<b>CONCLUSÃO.....</b>	
<b>REFERÊNCIAS.....</b>	

## **1. Introdução**

Não é de hoje que as empresas vêm utilizando tecnologias que melhoram a comunicação entre os funcionários e os clientes, facilitando o desenvolvimento dos seus serviços e os tornando mais eficientes. Entre essas tecnologias, uma das mais conhecidas entre empresas com um grande número de funcionários, como call centers, é o sistema de ramais. O ramal consiste em uma extensão para a comunicação, permitindo que as comunicações entre os colaboradores ocorram de maneira remota, seja para ligações internas ou externas. Todas essas ligações estão conectadas por uma central, que pode ser analógica, digital ou baseada em IP. Essencialmente, o ramal é uma extensão dos sistemas de telefonia, permitindo que todos possam se comunicar, seja através de dispositivos físicos, como telefones, ou digitalmente, conectados ao sistema telefônico da empresa. Sua implementação auxilia os colaboradores a realizarem seu trabalho de maneira mais eficiente, com comunicações mais fluidas e diretas.

Apesar de melhorar a eficiência das comunicações, os sistemas de ramais não são isentos de problemas. O sistema pode parar de funcionar repentinamente devido a diversos motivos, que vão desde problemas nas configurações, mais comuns nas versões digitais, até falhas nos dispositivos físicos. Quando ocorre uma falha em um ramal físico, tanto o ramal quanto o telefone conectado podem parar de funcionar, afetando múltiplos telefones se estiverem conectados ao mesmo ramal. Essas falhas podem interromper as funções dos funcionários, prejudicando a eficiência da empresa, como foi o caso da Prefeitura Municipal de Arenópolis, onde falhas no sistema de ramais atrasam processos importantes.

Diante desses problemas, o objetivo deste trabalho consiste no desenvolvimento de um novo sistema de comunicação, utilizando sockets na linguagem Python. Este sistema permitirá uma comunicação fluida e sem o risco de ocorrerem erros repentinos, funcionando como uma espécie de Chat Privado. Este Chat, estará conectado diretamente com o servidor da empresa, diferentemente dos ramais que se limitam apenas ao sistema telefônico. Além de ser um substituto para a comunicação através de mensagens, o sistema também permitirá a troca de arquivos, como documentos e imagens. Estando conectado diretamente aos computadores dos funcionários, quando alguma mensagem, ou arquivo for enviado, o destinatário receberá uma notificação, permitindo que ela tenha uma resposta mais rápida e eficiente.

Para um melhor entendimento de seu funcionamento, o artigo separou de maneira organizada algumas seções que irão esclarecer melhor o desenvolvimento e o funcionamento do aplicativo proposto, como os trabalhos relacionados da seção 2, onde serão apresentados alguns aplicativos e plataformas que inspiraram o desenvolvimento do trabalho, dentre eles o Slack e o Rocket.Chat, ambos focado na troca de mensagens e documentos, a sua descrição da proposta na seção 3, nela será apresentado alguns elementos que compõem o aplicativo e como eles influenciam no seu funcionamento, na seção 4 serão apresentados os seus materiais e métodos, ou seja, será abordada a sua metodologia, depois serão apresentados os resultados e discussões na seção 5, resumindo o desenvolvimento do aplicativo, mostrando quais foram os resultados obtidos com cada parte, logo em seguida na seção 6, dando uma visão geral sobre os problemas que surgiram no decorrer do desenvolvimento do aplicativo e por fim a

sua conclusão na seção 7, relembrando tudo que foi apresentado e o que pode ser tirado disso, além é claro, de finalizar apresentando as referências utilizadas para as pesquisas.

## **2. Trabalhos Relacionados**

Esta seção consiste na descrição dos principais trabalhos que serviram de inspiração para o desenvolvimento do sistema de chat privado, eles tratam de aplicações com seu código fonte disponível para visualização e através deles é possível obter uma maior compreensão sobre o aplicativo proposto.

O Private Ping é um aplicativo capaz de realizar uma troca de mensagens entre seus usuários de maneira segura, seu criador se denomina de Prince Khunt, ele atua como técnico no GDSC PPSU (Google Developer Student Clubs), sendo um profissional certificado em segurança cibernética, com experiência na área de desenvolvimento web e gerenciamento de projetos. Seu aplicativo utiliza a estrutura Django do Python, que permite que as mensagens trocadas sejam criptografadas, ele utiliza outras linguagens em segundo plano, mantendo o foco ainda em Python, que seriam o HTML, o CSS, o JavaScript, o API SubtleCrypto, o canais e o Redis. Sua criptografia utiliza o API SubtleCrypto em JavaScript, garantindo que apenas o usuário seja capaz de descriptografar as mensagens, além disso, ele não armazena as mensagens nos servidores, garantindo a privacidade do usuário.

O Rocket.Chat é uma plataforma desenvolvida para a comunicação, seu código é aberto e foi desenvolvida na linguagem javascript, sendo utilizada por empresas reconhecidas por sua proteção de dados avançados, como a Gluu e a DB Systel. Seu objetivo é permitir que conversas, através de um canal de comunicação, ou de vídeos chamadas, aconteçam em tempo real, em empresas, elas são muito utilizadas por equipes grandes, para que elas possam se comunicar de diferentes lugares, mas também para falar com seus clientes e até mesmo com outras empresas, graças a isso, a produtividade se eleva, melhorando os índices de satisfação dos clientes.

Slack é uma plataforma de comunicação criada especificamente para o meio comercial, foi desenvolvida em 2009 pela empresa Slack Technologie, fundada por Stewart Butterfield com o propósito de ser um meio de comunicação corporativa, assim diversas empresas de vários tamanhos adotam essa plataforma pela sua simplicidade, gerando não só uma alta produtividade como também aumentando a comunicação entre os empregados.

A ferramenta digital Slack é baseada no protocolo IRC no qual, para realizar a comunicação entre os usuários, a plataforma permite a criação de conversas particulares e canais de comunicação, esses canais são grupos de comunicação que podem ser definidos por temas e limitar o acesso a equipes específicas para o grupo. Além disso, nos canais de comunicação é possível enviar qualquer tipo de arquivos ou links, sendo capaz de reproduzir diretamente na conversa, como também, a permissão de conexão de plataformas distintas, facilitando diversos processos e problemas que poderiam vir por conta da diferença nos serviços, por exemplo, aplicativos como o Google Drive.

O Zulip consiste em uma ferramenta de colaboração para equipes que trabalham de diferentes lugares. Assim como os outros, ele também possui seu código aberto, além de ter uma threading exclusiva, combinando o melhor do e-mail e dos chats, para tornar o trabalho remoto, ou home office, mais produtivo e agradável. Ele foi construído por toda uma

comunidade de desenvolvedores de todas as partes do mundo, contendo mais de 74 pessoas contribuindo com mais de 100 commits, graças a isso, ele se tornou o maior e mais rápido projeto de chat de código aberto.

### 3. Descrição da Proposta

A aplicação foi desenvolvida através da linguagem Python, possuindo como característica o socket e com sua programação sendo orientada a objetos. A sua arquitetura pode ser separada entre três classes, as quais são, TSServer, TSClient e TSDataBase. Abordando um pouco o TSServer, ele fica responsável pela comunicação, sendo aquele que permite a comunicação dos clientes entre si.

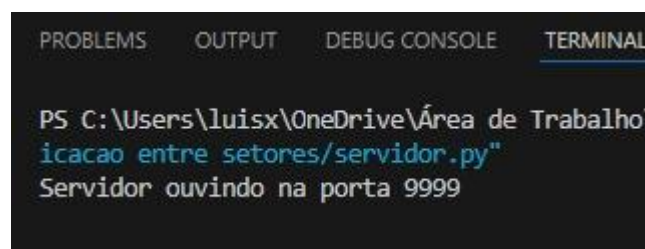
A screenshot of a Windows terminal window. The title bar shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The terminal text shows the command 'PS C:\Users\luisx\OneDrive\Área de Trabalho\icacao entre setores\servidor.py' and the output 'Servidor ouvindo na porta 9999'.

Figura 1. Terminal executando a porta do servidor.

Na Figura 1, é possível observar o código da classe TSServer em execução, com sua porta, que permite a comunicação entre os clientes, sendo aberta. A porta em questão se trata da porta 9999, que consiste em uma porta não atribuída a nenhum serviço padrão específico, em outras palavras, ela é considerada uma porta não registrada, graças a isso ela pode ser utilizada por qualquer aplicação que necessite de uma porta de comunicação, desde que não entre em conflito com outras aplicações ou serviços no mesmo sistema.

Na classe TSClient, é onde ocorre a execução da interface gráfica do sistema de chat privado. É nela que os usuários serão capazes de efetuar e trocar mensagens, ou arquivos. Na Figura 2, veja uma breve demonstração do seu acesso sendo executado.

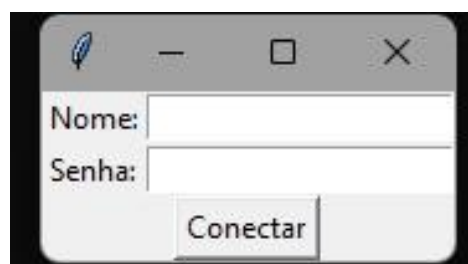


Figura 2. Interface para login.

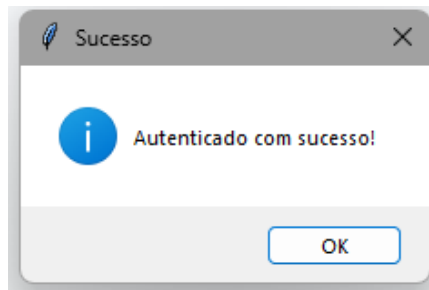


Figura 3. Autenticação sucedida.

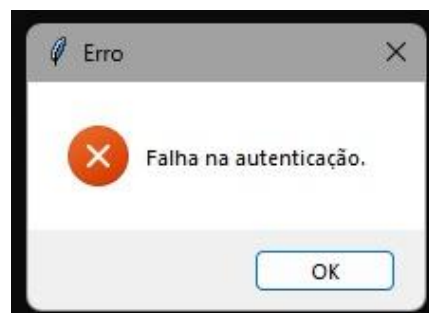


Figura 4. Falha na autenticação.

Na interface, o usuário deve digitar o seu nome, junto do setor de onde ele trabalha, informações essas armazenadas no servidor, como é mostrado na Figura 1. Realizando esse processo, inserindo todas as informações necessárias, caso elas estejam corretas, aparecerá uma mensagem confirmando que a autenticação foi realizada com sucesso, conforme mostra a Figura 3, caso contrário, uma mensagem será exibida para informar a falha que ocorreu na autenticação, como foi demonstrado na Figura 4. Após a inserção dos dados, será exibido uma nova interface, onde deverão ser digitados novamente tanto o nome, quanto o setor do usuário, mas além disso, também será necessário inserir o nome e setor do destinatário, ou seja, da pessoa que irá receber a mensagem. Na imagem abaixo (Figura 5), será mostrado essa nova interface com uma caixa para o remetente digitar a mensagem que desejar, podendo enviá-la através do botão “Enviar Mensagem”, além disso, ela também possui um botão, “Selecionar Arquivo”, que irá permitir que ele selecione um arquivo e o envie ao clicar em um outro botão, “Enviar Arquivo”, para o destinatário.



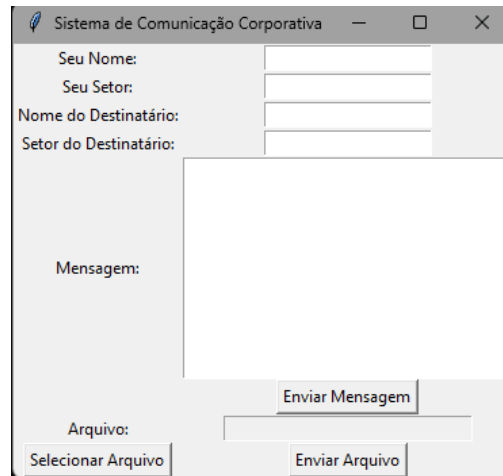


Figura 5. Interface para enviar a mensagem ou arquivo.

A partir do momento em que o remetente envia a mensagem para o destinatário, que deve estar conectado ao sistema, o programa irá abrir uma janela confirmando que o envio da mensagem foi realizado com sucesso, como pode ser visto na Figura 6, do contrário, será exibida uma outra janela que será oposta a anterior, mostrando que o usuário, apesar de estar registrado no banco de dados, ele não está conectado não está conectado, sendo ilustrado na Figura 7, e no caso das informações inseridas não estarem armazenadas no banco de dados, o programa irá mostrar mais uma janela, dessa vez, informando que o usuário não foi encontrado, veja o exemplo na Figura 8.

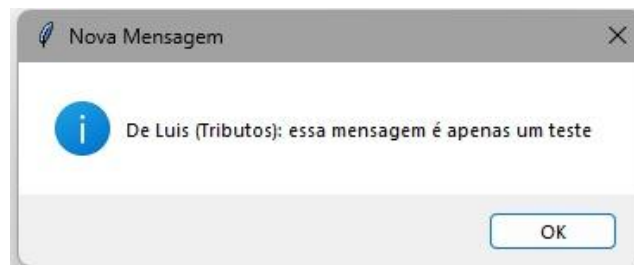


Figura 6. A mensagem foi enviada.

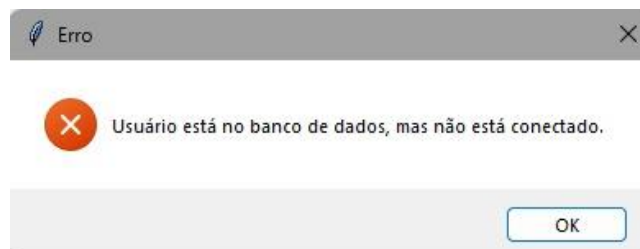


Figura 7. O usuário foi encontrado, entretanto, não está conectado.

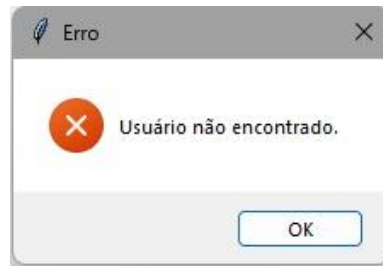


Figura 8. Usuário não encontrado.

Caso tudo seja realizado da maneira correta, selecionando e enviando o arquivo desejado, o destinatário receberá uma notificação informando que o mesmo recebeu um arquivo de algum usuário do banco de dados, e nela estará o nome e setor desse usuário, assim como o nome do arquivo (Figura 9), após clicar em ok, aparecerá outra mensagem, perguntando se ele deseja baixá-lo na máquina (Figura 10).

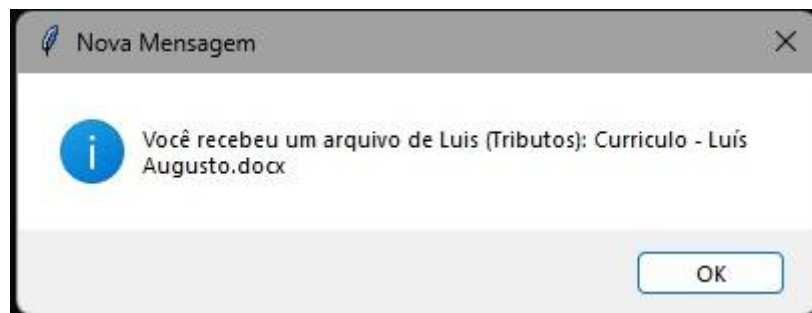


Figura 9. Informações relacionadas ao arquivo recebido.

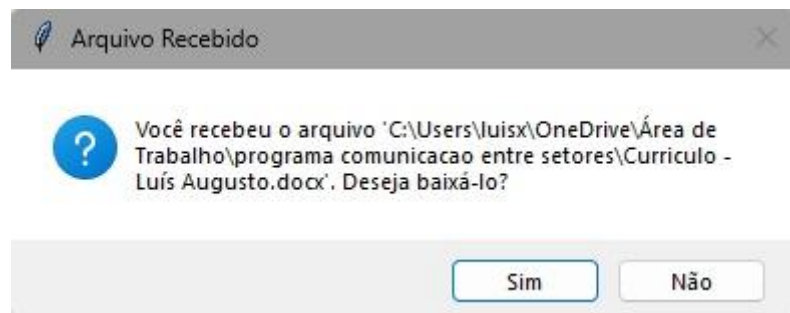


Figura 10. Permissão para download do arquivo.

Na classe TSDataBase, será simulado o banco de dados, será dele que as informações dos usuários e das suas senhas serão coletadas pelo servidor, e como foi exemplificado nas imagens anteriores, ele trabalhará juntamente do TSClient, demonstrando na interface gráfica processos importantes, como a autenticação do usuário e o envio e recebimento das mensagens e dos arquivos, como pode ser visto na Figura 11.

```

{} usuarios.json > ...
1  {
2      "Luis": {"setor": "Tributos", "senha": "senha_luis"},
3      "Gabriel": {"setor": "Prefeito", "senha": "senha_gabriel"},
4      "Felipe": {"setor": "Rh", "senha": "senha_felipe"}

```

Figura 11. Usuários e suas senhas cadastradas no banco de dados.

#### 4. Materiais e Métodos

Durante o desenvolvimento do algoritmo, as bases para o trabalho, ou seja, os materiais utilizados para realizá-lo, foram desde análise de outros trabalhos que foram citados anteriormente, como no caso do Slack e do Rocket.Chat, até vídeos que explicam melhor o funcionamento de sockets e também da comunicação entre usuários no geral. Sendo assim, a metodologia utilizada ao longo do seu desenvolvimento, para garantir que ele fosse executado corretamente, foi realizada através de:

- **Softwares utilizados:** foi utilizada a linguagem de programação python 3.12.3, utilizada no Visual Studio Code, através de um Sistema Operacional Windows 11 Home 64bits.
- **Tipo de socket:** utilizando o socket TCP (Transmission Control Protocol), para realizar a comunicação entre Cliente e Servidor, nele os clientes fazem a autenticação para enviar mensagens e arquivos através desse socket;
- **Hardware para desenvolvimento:** o código foi desenvolvido em um desktop com Intel® Xeon® E5-2620 v3 CPU 2.40GHz, de 16GB e Ram DDR4 (2x8) 3200MHz, além de um AMD Radeon™ RX 580 8GB, e um SSD Sata 240GB + SSD Sata 1TB;
- **Hardware do ambiente de testes:** o ambiente de testes era composto por um Intel® Xeon® E5-2620 v3 CPU 2.40GHz, com 16GB Ram DDR4 (2x8) 3200MHz, um AMD Radeon™ RX 580 8GB, e um SSD Sata 240GB + SSD Sata 1TB;
- **Testes realizados:** os testes foram feitos localmente em uma mesma máquina, que realizava uma simulação de um servidor, o qual estavam conectados diversos clientes, através da porta 9999.

#### 5. Resultado e Discussão

De maneira simples, este tópico consiste em um resumo de todo o desenvolvimento do trabalho proposto, demonstrando como ele foi feito e quais as suas capacidades, ou seja, como ele funciona na prática. Antes de descrever passo a passo, vale mencionar que os testes de troca de mensagem foram feitos em apenas um computador simulando um servidor e a conexão de vários clientes dentro de uma mesma rede. Para dar início ao algoritmo, primeiramente foi necessário escolher as bibliotecas, dentre elas, foram utilizados no código, json, threading, os, mas também alguns módulos, como no caso da tkinter, filedialog, messagebox, porém dentre todas elas, a principal classe utilizada foi o socket, com ele sendo importado pelo algoritmo, garantindo que a comunicação na rede fosse possível. Agora, veja nas imagens abaixo, as bibliotecas que foram utilizadas (Figura 12 e 13):

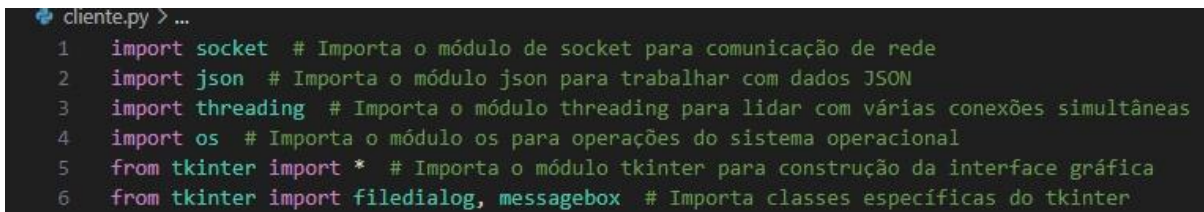


```

servidor.py > tratar_cliente
1  import socket
2  import threading
3  import json
4  import os

```

Figura 12. Bibliotecas utilizadas no servidor.



```

cliente.py > ...
1  import socket # Importa o módulo de socket para comunicação de rede
2  import json # Importa o módulo json para trabalhar com dados JSON
3  import threading # Importa o módulo threading para lidar com várias conexões simultâneas
4  import os # Importa o módulo os para operações do sistema operacional
5  from tkinter import * # Importa o módulo tkinter para construção da interface gráfica
6  from tkinter import filedialog, messagebox # Importa classes específicas do tkinter

```

Figura 13. Bibliotecas utilizadas no cliente.

Ao analisar a figura 13, fica perceptível que ele possui a maior quantidade de módulos e bibliotecas, isso ocorre, pois a interface gráfica foi desenvolvida nele, sendo simples e intuitiva, para que atenda a todos os usuários. Ela permite que o usuário seja capaz de se conectar dentro do chat privado garantindo a comunicação entre os funcionários da empresa, ao executar o servidor e deixá-lo ativo o próximo passo é executar o arquivo cliente.py que é o arquivo responsável pela interface gráfica para que possa acontecer a troca de mensagens e arquivos. Dentro da interface o usuário precisa apenas inserir o próprio nome e setor (remetente) e também o nome e setor de quem deseja enviar a mensagem ou arquivo (destinatário), após isso, basta apenas digitar e enviar a mensagem ou selecionar e enviar um arquivo para que o destinatário possa recebê-los.

## 6. Dificuldades Encontradas

O desenvolvimento de um trabalho sempre irá apresentar problemas, ainda mais um envolvendo uma aplicação, e com este trabalho não foi diferente, durante o seu desenvolvimento problemas foram surgindo, alguns mais simples de se resolver e outros mais complexos que demandam mais tempo, como no caso do desenvolvimento da interface, deveria ser uma parte mais simples, mas por inexperiência na área de design, ela foi uma das mais complicadas, felizmente ela foi concluída, mas como resultado, acabou sendo muito simples e com algumas falhas de alinhamento, por conta disso cada janela que é aberta surge no canto da tela.

Outra dificuldade foi por causa de uma ideia descartada, originalmente o código seria feito através de um dns, no windows server, mas por complicações que estavam atrasando o desenvolvimento, o método utilizado para sua criação foi substituída pelo socket, mas mesmo essa parte também trouxe complicações, por conta da porta 9999, a falta de costume em utilizá-la, contribuiu para mais problemas no decorrer do seu desenvolvimento, mas no fim, após seu funcionamento ser compreendido, através de pesquisas e análises, essa parte logo foi finalizada.

## Conclusão

Neste trabalho, foi apresentado um aplicativo voltado para comunicação, se inspirando em diversos aplicativos e plataformas, com destaque para a plataforma do Slack, da empresa Slack Technologies. O objetivo do aplicativo proposto é substituir o sistema de comunicação conhecido como Ramal, que apresenta problemas como desligamentos e falhas que podem prejudicar a comunicação empresarial. O aplicativo desenvolvido consiste em um Chat Privado que oferece uma alternativa mais estável e segura ao Ramal.

O código desenvolvido proporciona boas contribuições para a comunicação entre funcionários, permitindo a melhor comunicação possível e garantindo a segurança dos usuários, através do registro de informações como nome e setor de trabalho, armazenados no banco de dados. O aplicativo foi desenvolvido em Python, utilizando sockets e programação orientada a objetos. Apesar dos Problemas que surgiram ao longo do seu desenvolvimento, todos foram resolvidos, permitindo que o código funcionasse normalmente, sem nenhum problema grave.

Dessa forma, conclui-se que o aplicativo do Chat Privado, apesar de ainda poder evoluir, atingiu seu objetivo principal de oferecer um sistema de mensagens seguro e sem as falhas do Ramal. Futuras melhorias podem incluir a expansão do banco de dados, sendo capaz de agrupar um maior número de informações, aprimoramento da interface para torná-la mais atrativa, porém, ainda mantendo sua base simples, para que continue sendo intuitiva e a sua capacidade de troca de mensagens com outras redes, permitindo a comunicação entre mais computadores, já que no momento ele ainda se restringe a uma única rede local.

## Referências

KUPPE, Fernanda. O QUE É RAMAL? **VC-X Solutions**, 04 de novembro de 2022. Disponível em: <<https://vcx.solutions/ramal/>>. Acesso em: 31 de maio de 2024.

TUTIDA, Daniel. Melhores opções de chat interno para empresas. **Eunerd**, 10 de agosto de 2021. Disponível em: <<https://encontreunnerd.com.br/blog/chat-interno-para-empresas>>. Acesso em: 31 de maio de 2024.

DIAS, Joilson Alcindo; NASCIMENTO, Maria do Socorro Araújo. COMUNICAÇÃO EMPRESARIAL, A importância da comunicação nas organizações e o advento de novas tecnologias. **RIOS - Revista Científica da Faculdade Sete de Setembro**, Fortaleza, v.10, n.11, 1 de dezembro de 2016. <<https://www.publicacoes.unirios.edu.br/index.php/revistarios/article/view/493/492>> Acesso em: 31 de maio de 2024.

VILVERT, Cassiane. O Papel da Comunicação Interna na Transformação Digital das Empresas. **Transformação Digital**, 17 de Outubro de 2017. Disponível em:

<<https://transformacaodigital.com/recursos-humanos/comunicacao-interna-na-transformacao-digital/>>. Acesso em: 31 de maio de 2024.

BAPTISTA, Renato Dias. A comunicação empresarial e a gestão da mudança. Disponível em:

<<http://connexos.com.br/artigos/baptista-renato-comunicacao-gestao.pdf>>. Acesso em: 31 de maio de 2024.

KHUNT, Prince. PrivatePing. **GitHub**, 10 de março de 2024. Disponível em: <<https://github.com/princekhunt/privateping?tab=readme-ov-file>>. Acesso em: 31 de maio de 2024.

GUBERT, Douglas; TAPIA, Rafael. Rocket.Chat. **GitHub**, 2015. Disponível em: <<https://github.com/RocketChat/Rocket.Chat>>. Acesso em: 31 de maio de 2024.

HEALEY, Harrison. Mattermost. **GitHub**, 2015. Disponível em: <<https://github.com/mattermost/mattermost>>. Acesso em: 31 de maio de 2024.

ABBOTT, Tim. Zulip. **GitHub**, 2015. Disponível em: <<https://github.com/zulip/zulip>>. Acesso em: 31 de maio de 2024.

RUÃO, Teresa; MARINHO, Sandra; SILVA, Sónia. O impacto das tecnologias na comunicação interna das equipes de inovação. 23 de outubro. Disponível em: <[https://abrapcorp2.org.br/site/manager/arq/\(cod2\\_23134\)um\\_trabalho\\_de\\_investigaAoaAAo\\_.pdf](https://abrapcorp2.org.br/site/manager/arq/(cod2_23134)um_trabalho_de_investigaAoaAAo_.pdf)>. Acesso em: 21 de maio de 2024.

MACHADO, Alisson. Criando um Messenger com Sockets em Python: Tutorial Passo a Passo. **Blog 4linux**, 30 de novembro de 2018. Disponível em: <<https://blog.4linux.com.br/socket-em-python/>>. Acesso em: 08 de junho de 2024.

Aprenda Tudo sobre Socket Python e Domine a Programação de Rede. **Awari**, 26 de agosto de 2023. Disponível em: <

Python - Socket Programming. **TutorialsPoint**. Disponível em:  
<[https://www.tutorialspoint.com/python/python\\_socket\\_programming.htm](https://www.tutorialspoint.com/python/python_socket_programming.htm)>. Acesso em: 08 de junho de 2024.

Python - JSON. **TutorialsPoint**. Disponível em:  
<[https://www.tutorialspoint.com/python/python\\_json.htm](https://www.tutorialspoint.com/python/python_json.htm)>. Acesso em: 08 de junho de 2024.

DASR, Felipe. COMO ENVIAR E RECEBER ARQUIVOS COM SOCKET EM PYTHON. **Youtube**, 11 de outubro de 2020. Disponível em:  
<<https://www.youtube.com/watch?v=j4Drn47pc3o>>. Acesso em: 08 de junho de 2024.

DASR, Felipe. MÚLTIPLAS CONEXÕES COM SOCKET EM PYTHON. **Youtube**, 08 de setembro de 2021. Disponível em:  
<<https://www.youtube.com/watch?v=QyJBrS1c1s8>>. Acesso em: 08 de junho de 2024.

MILANEZ, Pini Samuel. Sistema de Chat Simples(TCP/IP) - Python 3.6 - Socket - Thread - Tkinter. **Youtube**, 26 de novembro de 2019. Disponível em:  
<<https://www.youtube.com/watch?v=jAOWsFCSrIY>>. Acesso em: 08 de junho de 2024.

MOREIRA, Esdras. Novas Tecnologias de Comunicação: novos comportamentos. **Transformação Digital**, 5 de março de 2018. Disponível em:  
<<https://transformacaodigital.com/tecnologia/novas-tecnologias-de-comunicacao-e-o-futuro-das-nossas-relacoes/>>. Acesso em: 11 de junho de 2024.

BORGES, Diego. O que é Slack? Como funciona a ferramenta. **TecMundo**, 14 de março de 2021. Disponível em:  
<<https://www.tecmundo.com.br/software/212674-slack-funciona-ferramenta.htm>>. Acesso em: 11 de junho de 2024.