

UNIVERSIDADE PAULISTA - UNIP

GABRIELE CRISTINE DA SILVA CASARI

GUILHEME AKIO OCHI

HENRIQUE ALONSO

MATHEUS VINÍCIUS BRASIL MORAES

MATEUS GARCIA SANTOS

THIAGO DE PAULA SOUZA

Desenvolvimento de Uma Ferramenta Para Comunicação em Rede

SOROCABA

2022

GABRIELE CRISTINE DA SILVA CASARI

GUILHEME AKIO OCHI

HENRIQUE ALONSO

MATHEUS VINÍCIUS BRASIL MORAES

MATEUS GARCIA SANTOS

THIAGO DE PAULA SOUZA

Desenvolvimento de Uma Ferramenta Para Comunicação em Rede

Atividade prática supervisionada e apresentada ao curso Ciência da Computação, para fins de conhecimento na área.

Orientador: Mestre Aparecido Antonio Donizeti Correia Leite

SOROCABA

2022

Dedicamos este trabalho às pessoas ao nosso redor que nos apoiam a cada passo, sejam amigos, familiares, companheiros e também ao corpo docente presente em todo o período letivo cursado, auxiliando em nossa jornada como alunos e futuros profissionais da área.

AGRADECIMENTOS

Nossos pais e parentes mais próximos são os primeiros, já que nos trouxeram ao mundo, nos mostraram os primeiros passos e nos permitiram dar os próximos sozinhos.

Aos professores que estiveram envolvidos neste projeto, Waldir Silva, Cesar Xavier, Aparecido Leite e Luciano de Carvalho por terem feito o seu melhor para passar todo o conhecimento, incentivando a sermos profissionais melhores.

E a todas as pessoas que têm passado pelas mesmas dificuldades que o grupo, em momentos adversos como estes.

RESUMO

Comunicação em rede refere-se à transferência de dados por meio eletrônico, seguindo essa proposta, o objetivo deste projeto é integrar a comunicação em rede à ecologia consciente, de forma em que uma interface facilite o diálogo entre um grupo de pessoas, sejam funcionários de uma empresa, moradores de um bairro etc., e pessoas especializadas em segurança ambiental. Com uma interface simples e funcional, os usuários podem realizar denúncias/avisos de ocorrências. Desenvolvido em linguagem C#, utilizando o preceito de POO (Programação Orientada a Objetos, paradigma da programação, onde o objeto é uma junção de estados e comportamentos), para a conexão em rede são utilizados protocolos de segurança TCP/IP e uma API (conjunto de padrões que formam uma interface e que permitem que plataformas sejam criadas de maneira simples) que fornece elementos resultantes de ramificações do socket de Berkeley.

Palavras-chave: Rede; Interface; Sustentabilidade.

ABSTRACT

Network communication refers to the transfer of data by electronic means, following this proposal, the objective of this project is to integrate network communication to conscious ecology, in a way that an interface facilitates dialogue between a group of people, whether employees of employees a company, residents of a neighborhood, etc., and people specialized in environmental safety. With a simple and functional interface, users can make complaints/notices of occurrences. Developed#, using the TCP/IP configuration precept/programming paradigm, where the object is a language of states and behavior, using TCP/IP security protocols/network protocols and an API (set of standards that form an interface and that allow that platforms are simply created elements) that provide results from ramifications of the Berkeley socket.

Keywords: Network; Interface; Sustainability

SUMÁRIO

1	INTRODUÇÃO	10
2	COMUNICAÇÃO EM REDE	12
2.1	Protocolos de comunicação	13
2.1.1	Protocolo TCP/IP	15
2.2	Sockets de Berkeley	16
2.3	Comunicação e Meio Ambiente	18
3	PLANO DE DESENVOLVIMENTO	20
3.1	Cronograma	20
3.2	Público	21
3.3	Linguagem	23
3.4	Design	25
4	DESENVOLVIMENTO	26
4.1	Front-End	26
4.2	Back-End	31
5	CONCLUSÃO	34
	REFERÊNCIAS	36
	ANEXO A — IMAGEM LOGO UTILIZADA	38

LISTA DE ILUSTRAÇÕES

Esquema 1 — Modelo Genérico de Comunicação	12
Esquema 2 — Sockets nas camadas	17
Tabela 1 — System Calls de Berkeley Sockets	17
Imagem 1 — Quadro Kanban Jira	20
Imagem 2 — Continuação Deploy	21
Imagem 3 — Logo do Projeto	27
Imagem 4 — FormServer	27
Imagem 5 — FormServer Conectado	28
Imagem 6 — FormLogin	28
Imagem 7 — FormAnalista	29
Imagem 8 — FormAnalista desconectado	30
Imagem 9 — FormDenunciante	30
Imagem 10 — FormDenunciante desconectado	31

LISTA DE ABREVIATURAS E SIGLAS

API	Applications Programming Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment.
IMAP	Internet Access Protocol
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
PNG	Portable Network Graphics
POP3	Post Office Protocol 3
SCTP	Stream Control Transmission Protocol
SFTP	Simple File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SSH	Secure Socket Shell
SSL	Secure Sockets Layer
TCC	Trabalho de Conclusão de Curso
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WWW	World Wide Web

LISTA DE SÍMBOLOS

@	Arrouba
®	Marca registrada
±	Número tanto positivo quanto negativo

1 INTRODUÇÃO

Segundo Oxford Languages, comunicação é “ação de transmitir uma mensagem e, eventualmente, receber outra mensagem como resposta”. Existem elementos essenciais que a formam, os principais são mensagem, transmissor e receptor.

No ramo da computação, o termo rede refere-se a uma comunicação eletrônica, as redes possibilitam a troca de informações entre cidades, países e continentes, o envio, recebimento de e-mails, visualização de vídeos na televisão ou receber áudios no celular só são possíveis pela existência de redes e não é diferente: necessita de dispositivos que realizam o papel de transmissores e receptores, nesse caso computadores ou periféricos conectados entre si por um nó, diversos nós compõem uma malha de rede.

Para assegurar que toda a mensagem será corretamente transmitida, com segurança e compreendida pelo receptor são necessários dispositivos e estruturas de *hardware* e *software*: cabeamento estruturado, *hosts*, roteadores, *switches*, *hubs*, protocolos de comunicação, entre outros.

Para realizar grande parte das conexões de rede, é necessária uma estrutura física composta por cabos, conectores e condutas, também chamado de cabeamento estruturado

Qualquer computador ou máquina conectados a uma rede são considerados *hosts*, a quem são atribuídos IPs (*Internet Protocol* ou protocolo de rede) e nomes, computadores, celulares, modems e servidores são exemplos.

Roteadores são equipamentos físicos pelos quais flui o tráfego dos dados da rede, o tráfego é sustentado por pacotes e camadas.

Um *hub* é um repetidor que promove um ponto de conexão física entre os equipamentos. *Switches* são uma evolução do *hub*, com funções de pontes, roteadores e hardware especiais que lhe conferem baixo custo e alta eficiência.

E por fim, para que toda conexão seja realizada, regras são estabelecidas, elas definem, através de normas e padrões, como o contato será realizado.

No trabalho que se segue será apresentada uma forma de integração da Comunicação em rede com os preceitos da Ecologia, como o desenvolvimento de uma ferramenta de troca de mensagens entre usuários e analistas, com a intenção de serem realizadas denúncias/avisos sobre ocorrências que afetem o meio-ambiente.

Objetivo do Trabalho: O objetivo deste projeto é, inicialmente, promover o estudo das ferramentas aqui utilizadas, linguagem C#, Sockets de Berkeley,

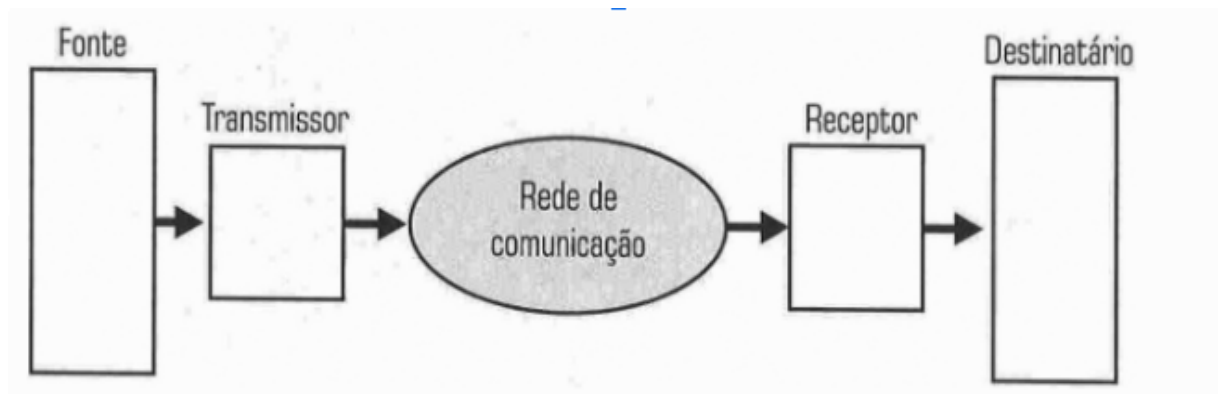
Protocolos de Comunicação em rede, entre outros conteúdos do âmbito, tanto para os leitores quanto para o grupo que o produziu. A partir dele, os participantes chegaram ao objetivo específico de desenvolver um software de comunicação em rede para facilitar o diálogo entre testemunhas de um acidente ambiental e autoridades dentro de um ambiente profissional, a partir de máquinas computadorizadas espalhadas pelo chão de fábrica, onde os funcionários terão acesso à interface.

2 COMUNICAÇÃO EM REDE

Como visto previamente, para que ocorra a comunicação são necessários alguns componentes básicos:

- Mensagem/Informação (o dado que irá ser transmitido, podendo conter textos com caracteres especiais ou não, números, áudio, vídeo, figuras ou qualquer junção desses elementos).
- Emissor ou Transmissor (sendo a pessoa ou dispositivo/máquina que envia o dado, celulares, computadores, câmeras de vídeo, entre outros dispositivos)
- Receptor (sendo a pessoa ou dispositivo/máquina que recebe o dado, dispositivos semelhantes ao emissor.)
- Meio (É a forma ou caminho onde origina-se o dado, também chamado de canal de comunicação.)
- Protocolo (conjunto de regras que estabelecem uma comunicação de dados. Representa um padrão para que os dispositivos se comuniquem.) (vide tópico 2.1)

Esquema 1 — Modelo Genérico de Comunicação



Fonte: Dantas (p. 8)

As informações possuem uma representação específica no momento da transmissão, o que chamamos de sinal, que pode ter dois formatos, analógico e digital e qualquer informação pode ser transmitida por eles.

Analógico: É um sinal variável que gera um formato de onda contínua por meio de propagação mecânica, é considerado obsoleto por ter qualidade inferior aos outros, oscila e trafega por uma frequência muito grande.

Digital: Um sinal variável que em conjunto de valores discretos é transformado em linguagem de máquina.

As formas de comunicação entre dispositivos pode acontecer de três maneiras distintas: sendo *Simplex*, ou *Half-duplex*, ou *Full-duplex*:

A forma mais simples é o *simplex*, onde a mensagem é transmitida de forma em que o receptor não responde, ocorre como uma via de mão única, somente um dos dispositivos é capaz de se transmitir informação/dados, logo o outro só é capaz de receber, como exemplo os *mainframes*, monitores, rádios e televisões.

No *Half-duplex*, cada dispositivo ou estação pode transmitir e também receber, porém não simultaneamente. Então quando um dispositivo está transmitindo o outro está recebendo, nesta transmissão toda capacidade do meio ou canal é dada ao dispositivo que estiver transmitindo no momento, como exemplos os Walkie Talkies.

No modo *Full-duplex*, os dispositivos transmitem e recebem simultaneamente, seus sinais são em direções opostas e compartilham determinada capacidade do canal ou link, como exemplos os telefones.

A qualidade da transmissão de dados depende de alguns aspectos:

- Entrega (*delivery*): Determinado sistema solicita entrega de dados ao destino e devem ser recebidos somente por ele.
- Confiabilidade: Deve garantir que a entrega de dados corrompidos ou dados modificados seja notada e corrigida.
- Tempo de Atraso: Os dados do sistema devem ser entregues em um tempo pré determinado e finito, pois dados que são entregues tardiamente poderão ser considerados poucos úteis. No caso de transmissão de vídeos, fotos, multimídia em geral, especifica-se que eles devem ser entregues de forma imediata.

Rede Local ou LAN (Local Area Network) é formada por um conjunto de computadores que pertence a mesma instituição, conectados entre si por uma rede, através de cabos ou links sem fio, em uma área geográfica definida, geralmente através de uma mesma tecnologia, com o objetivo de compartilhar recursos, dados ou serviços.

Uma conexão remota permite que um usuário, a partir da sua rede local, conectada a internet, conecte-se a rede local de outra pessoa, em locais remotos, podendo realizar o envio de arquivos, vídeos e outros dados.

2.1 Protocolos de comunicação

Os Protocolos podem ser definidos como um compilado de regras/padrões/normas que possibilitam a comunicação de dispositivos que devem estar conectados a uma rede.

A internet e seus protocolos funcionam como uma linguagem universal entre

os dispositivos ou computadores, que são conhecidas por todos os tipos de máquina de forma equitativa.

Os protocolos são separados de acordo com o serviço que oferecem, e em que camada de profundidade da internet operam. As principais divisões por camada são:

- Camada de Aplicação: WWW (*World Wide Web*) para navegação em Web, FTP (*File Transfer Protocol*), possui duas conexões TCP/IP (*Transmission Control Protocol/Internet Protocol*) pelas quais são feitas transferências de arquivos, SSH (*Secure Socket Shell*) protocolo de camada para segurança em trocas de arquivos entre cliente e servidor de internet, utilizando determinada criptografia;
- Camada de Transporte: SCTP (*Stream Control Transmission Protocol*), TCP e UDP (*User Datagram Protocol*), faz o transporte e organização de arquivos recebidos da Camada de Aplicação, transformando-os em pacotes menores, que logo serão enviados à rede;
- Camada de Rede: IP (IPv4 e IPv6), os arquivos da camada anterior são obtidos e anexados ao IP da máquina que envia e recebe os dados;
- Camada de Estrutura Física: *Ethernet* e Dispositivo Modem. É a camada que recebe e envia arquivos para a Web.

Alguns dos protocolos são:

- Protocolo TCP/IP é a combinação de dois protocolos: TCP, Protocolo de Controle de Transmissão e IP, Protocolo de Internet, formam o recebimento e a base de envio de informações (Dados) por toda a internet.
- Protocolo HTTP (*Hypertext Transfer Protocol*): Protocolo de Transferência de Hipertexto que consiste em uma conexão entre cliente (navegador utilizado para acessar a internet) e servidor (máquina em que está hospedado o site na rede). O cliente(navegador) solicita um pedido de acesso a uma página enquanto o servidor requisita uma resposta de permissão ao acesso. é o protocolo mais comum, usado para navegação em sites da internet
- Protocolo HTTPS (*Hypertext Transfer Protocol Secure*): Semelhante ao HTTP, porém conta com uma camada de proteção extra, chamada de SSL (*Secure Sockets Layer*) e pode ser verificada nos navegadores com a letra “S” a mais na URL (*Uniform Resource Locator*). Geralmente utilizado em sites de banco, onde pagamentos são realizados, lojas, órgãos de segurança e governamentais. Exige proteção de dados e possui certificado digital de criptografia.
- Protocolo DHCP (*Dynamic Host Configuration Protocol*): Possibilita a troca

de endereços IP automaticamente entre computadores, sem necessidade de configurações manuais. No momento da obtenção do IP, a máquina fica inviável para determinado uso. Caso a conexão seja desligada ou interrompida, o endereço volta a funcionar. Possui três diferentes formas de funcionamento:

- Protocolo FTP: Permite a troca de arquivos entre dois computadores de forma direta, onde um deles pode acessar as pastas do outro. Considerado inseguro pelos padrões atuais, navegadores modernos estão deixando de oferecer suporte.
- Protocolo SFTP (*Simple File Transfer Protocol*): Forma mais elaborada do protocolo FTP com mais uma camada de proteção de arquivos que se transferem. Quando seu protocolo é solicitado ele vem com a tecnologia SSH para fornecer proteção e autenticação da sua conexão do servidor e cliente.
- Protocolo SSH: Protocolo de Bloqueio de Segurança é um tipo de protocolo feito para segurança específica de troca de arquivos entre servidor e cliente, usando criptografia pelo SSH, o usuário que utiliza a internet consegue estabelecer um sistema com proteção para seu site sem afetar seu desempenho.
- Protocolo POP3 (*Post Office Protocol 3*): POP3 opera de forma semelhante a uma caixa-postal, estabelece o recebimento e armazenamento de mensagens do servidor de e-mail. De forma em que o cliente que faz a conexão se autentica ao servidor e pode ler e acessar os diferentes tipos de dados lá arquivados.
- Protocolo SMTP (*Simple Mail Transfer Protocol*): Diferente do POP3 seu protocolo é orientado para envio de e-mails (mensagens eletrônicas). A mensagem sai de uma máquina, e após ter destinatários determinados, a mensagem é autenticada e então enviada ao servidor. São recebidas e codificadas pelo protocolo POP3. É limitado já que envia somente textos, para outros tipos de arquivos são necessárias extensões para convertê-los.
- Protocolo IMAP (*Internet Access Protocol*): Direcionado ao recebimento e envio de e-mails, diferente dos anteriores, este protocolo possibilita que o usuário acesse e gerencie seus arquivos e também mensagens diretas pelo próprio servidor, logo, o cliente não precisa esperar para que chegue até sua própria máquina.

2.1.1 Protocolo TCP/IP

Criado em 1969, nos Estados Unidos, a partir de pesquisas da ARPANET,

rede de comunicação militar com objetivo de estabelecer conexão com computadores de diversos pontos do País, tornando a troca de dados mais rápida.

É a sigla para dois protocolos misturados: TCP (Protocolo de Controle de Transmissão) e IP (Protocolo de Internet).

É um conjunto de protocolos, e esse conjunto é dividido em 4 camadas, responsáveis por tarefas distintas:

- Aplicação: usado para enviar e receber dados de outros aplicativos pela internet. Esta camada contém os protocolos HTTP(para navegar na internet), FTP (transferência de arquivos) e SMTP(para email).
- Transporte: Responsável por receber os dados da camada anterior e transformá-los em pacote compactos.
- Malha: Os arquivos compactados na camada de transporte são recebidos e anexados ao endereço IP do dispositivo que envia e recebe os dados. Então são enviados pela internet.
- Interface: Esta é a camada usada para enviar ou receber arquivos na internet. Os protocolos nela utilizados variam e dependem do tipo de rede em que está sendo utilizado, o mais comum é Ethernet.

Começou no ano de 1969 perante algumas pesquisas militares da organização/departamento ARPANET

Com o intuito de ser utilizado para fins militares, sua ideia era oferecer uma troca rápida de mensagens entre computadores, pensado também em caso de um desastre nuclear para garantir uma conexão viável entre as máquinas.

2.2 Sockets de Berkeley

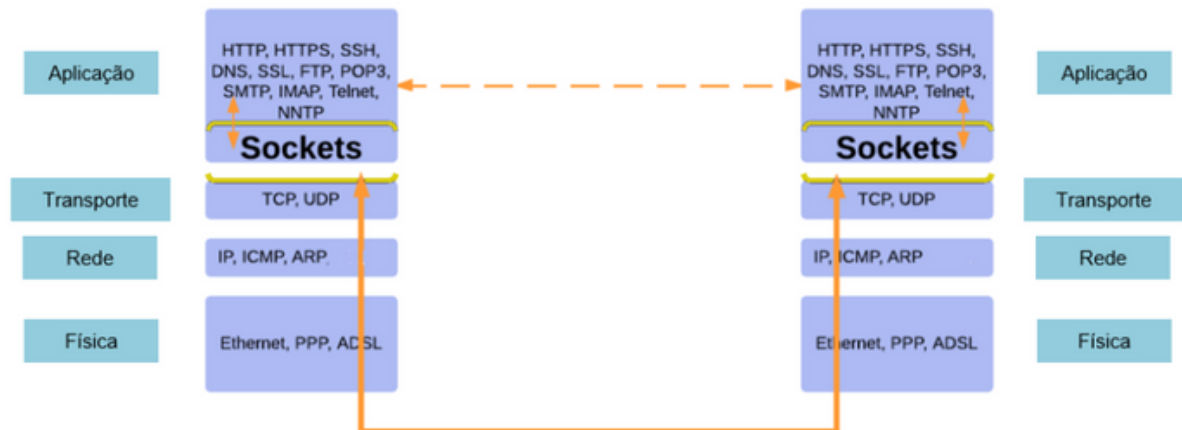
Sockets são utilizados como forma de permitir comunicação bidirecional (origem e destino) entre dois processos (Inter-process communication) na mesma máquina (Sockets Unix) ou rede (Sockets TCP/IP). Parece que a comunicação está acontecendo diretamente entre os processos, porém, está passando pelas camadas da rede. A abstração fornecida por eles é chamada de comunicação lógica.

Na web, sockets são representados por ip:port, por exemplo: 127.0.0.1:4477 (IPv4).

Todos os clientes devem saber o socket do servidor que deseja se comunicar, porém o servidor só conhecerá o socket do cliente quando ele se conectar, portanto, a conexão no modelo cliente-servidor é sempre iniciada pelo cliente iniciado.

Considerando o modelo OSI, os sockets são alocados entre a camada de aplicação e transporte, como no exemplo a seguir:

Esquema 2 — Sockets nas camadas



Fonte: Adaptado de Pantuza (2017)

No cotidiano utilizam-se frequentemente sockets em navegadores, para acessar páginas e solicitar conexão entre servidores (através do protocolo de aplicação SSH).

O socket de Berkeley é uma API genérica para programação de protocolos de comunicação, originado com 4.2BSD UNIX, lançado em 1983, suas system-calls (chamadas de sistema) são padrão em sistemas operativos Unix, utilizada também em outras plataformas.

Como foi projetado para fornecer suporte a diversos protocolos, deve também oferecer suporte a diferentes formatos de dados, como endereços, que são armazenados em uma estrutura genérica especial.

Os system-calls utilizados no socket de Berkeley são:

Tabela 1 — System Calls de Berkeley Sockets

Primitivas	Significado
SOCKET	Cria um ponto final de comunicação
BIND	Anexa um endereço local a um Socket
LISTEN	Mostra a disposição de aceite de conexões
ACCEPT	Bloqueia o processo até que chegue um pedido de conexão
CONNECT	Anexa um endereço local a um Socket, usando o endereço de destino
SEND	Envia alguns dados pela conexão
RECEIVE	Recebe alguns dados da conexão
CLOSE	Encerra a conexão

Fonte: Adaptado de Teamques10 (2017)

Pelo servidor:

- Iniciador do servidor executa as primitivas SOCKET, BIND & LISTEN;
- Primitiva LISTEN aloca fila para vários clientes simultâneos;
- Em seguida, utiliza ACCEPT para suspender o servidor até chegada de solicitação;
- Quando a solicitação do cliente chega: ACCEPT retorna;
- Inicia um novo socket (thread ou processo) com as mesmas propriedades do original, isso trata a requisição, o servidor continua aguardando o socket original;
- Se uma nova solicitação chega durante a geração de uma thread, ela será enfileirada;
- Se a fila estiver cheia é recusada.

Pelo cliente:

- Usa primitivas SOCKET para criar;
- Em seguida, utiliza CONNECT para iniciar o processo de conexão;
- Quando há retorno do anterior, o soquete está aberto;
- Ambos os lados podem agora RECEBER e ENVIAR;
- A conexão não é liberada até que ambos os lados sejam FECHADOS;
- Normalmente o cliente fecha e o servidor reconhece.

Para obter acesso aos protocolos de comunicação, abre-se um socket: utilizando "socket", que envolve parâmetros de diversos protocolos que permite, devolvendo um descritor preparado para as operações subsequentes.

Para ser possível o envio e recebimento de dados através de um descritor aberto, é necessário definir a porta que o descritor utilizará, utilizando a System-call Bind:

```
int bind(int sock, struct sockaddr *myAddress, int addrLen);
```

O parâmetro sock é o descritor retornado anteriormente pela função socket, struct sockaddr *myAddress aponta para a estrutura que contém o endereço e int addrLen é o parâmetro do tamanho da estrutura.

2.3 Comunicação e Meio Ambiente

A palavra sustentabilidade vem do latim "sustentare" e tem como significado defender, sustentar, favorecer, cuidar, apoiar e conservar. Refere-se à busca pelo equilíbrio entre a disponibilidade dos recursos naturais e a exploração deles por parte da sociedade.

O conceito de tripé da sustentabilidade, criado em 1994 pelo empresário britânico John Elkington, conclui que as organizações precisam se responsabilizar

por três pilares: Econômico, Ambiental e Social.

Econômico: A sustentabilidade econômica é como uma organização se preocupa com o setor financeiro e o patrimônio, um exemplo de prática sustentável econômica está no investimento de parte dos lucros em equipamentos e materiais novos garantindo qualidade e eficiência em suas produções, outra prática é a transparência nos dados empresariais, como lucros e perdas.

Ambiental: O âmbito ambiental refere-se a como uma instituição pode usufruir de bens naturais sem agredir o meio ambiente, práticas como o descarte correto de lixo, matérias primas adquiridas em fornecedores que seguem normas ambientais, diminuir ou acabar com o consumo de recursos naturais e a emissão de poluentes.

Social: O pilar social trata de como a organização impacta a sociedade em que está inserida, tanto internamente: preocupando-se com a saúde mental e física de seus funcionários, fornecendo remuneração digna, equipamentos de segurança e tempo de descanso quanto externamente: analisando como o serviço impacta a sociedade como um todo, promovendo projetos sociais e conscientização.

3 PLANO DE DESENVOLVIMENTO

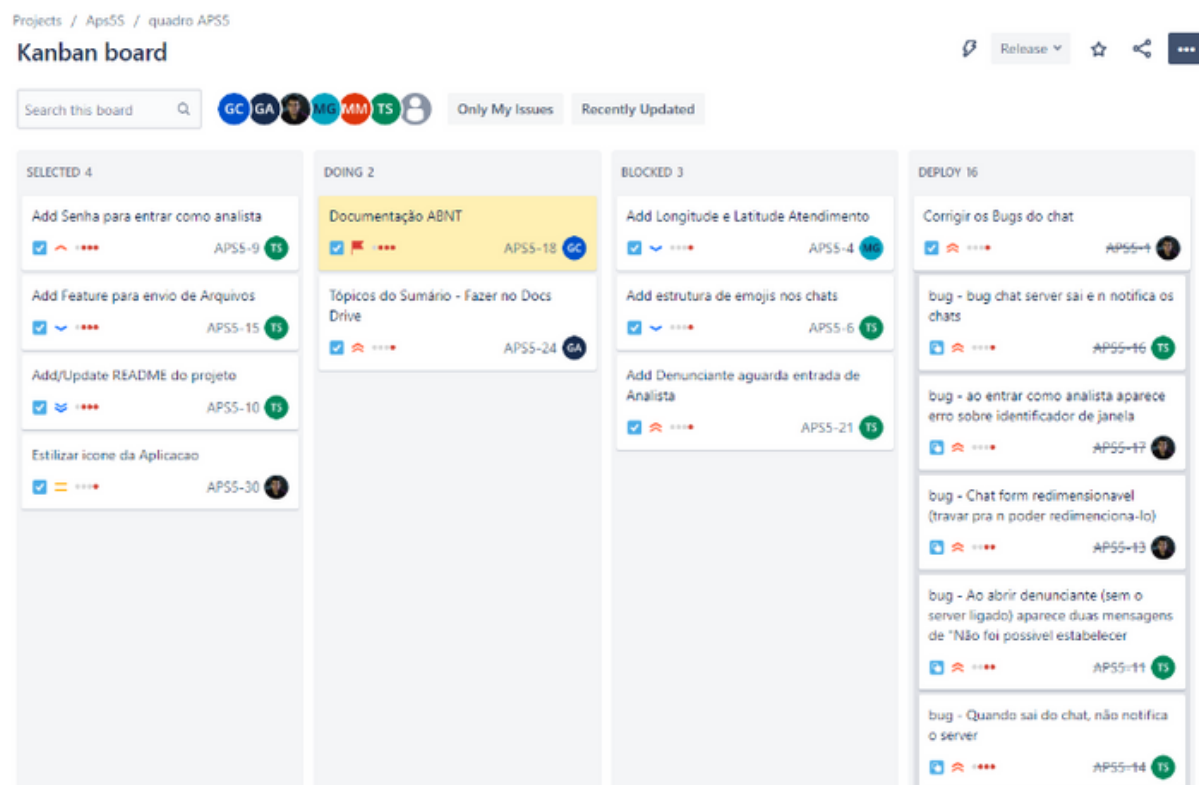
Através de brainstorms e reuniões, o grupo definiu o projeto como a produção de um software de denúncia tendo como cenário um chão de fábrica ou local de serviço onde o uso de celulares é restrito, seja por ausência de sinal ou para aumentar a produtividade dos funcionários (evitando distrações causadas por ele).

Cada setor da empresa deve ter uma máquina com a interface do software acessíveis a todos, onde solicitarão um token para a comunicação com o setor responsável pela segurança ambiental. Há também a interface que será utilizada pelos analistas onde será feito o atendimento das denúncias/avisos, os responsáveis farão o encaminhamento para o setor capaz de resolver o incidente ou prestarão atendimento, encaminhando o usuário a realizar o procedimento necessário.

3.1 Cronograma

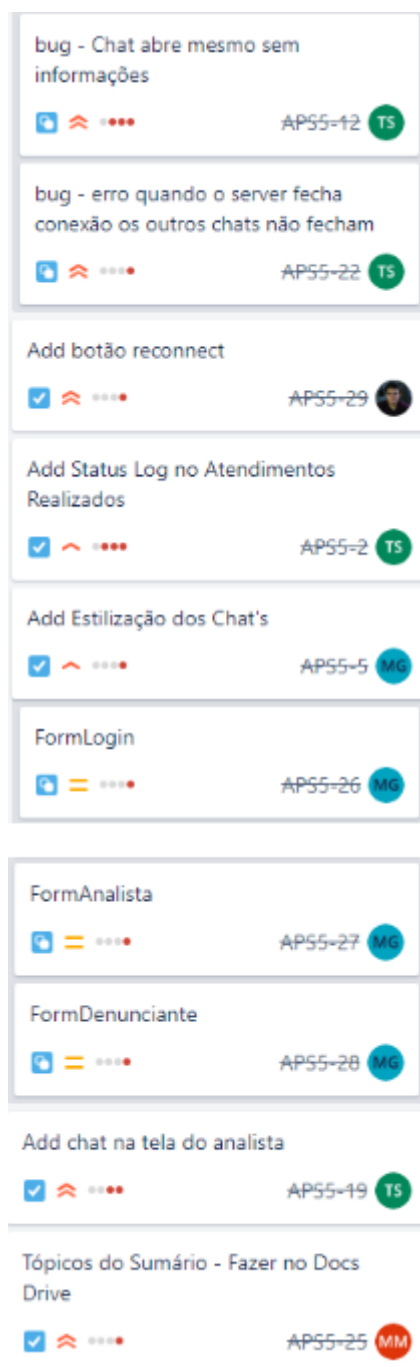
Kanban board é um método de organização de projetos que simula a posição de um painel de controle com cartões que se movem horizontalmente entre as seções divididas por estágios.

Imagem 1 — Quadro Kanban Jira



Fonte: Os autores (2022)

Imagem 2 — Continuação Deploy



Fonte: Os autores (2022)

3.2 Público

Existem leis vigentes no Brasil para reduzir ou regulamentar os impactos ambientais dentro de uma instituição:

- Lei Federal 9.605 de 12 de fevereiro de 1998 – Lei de crimes ambientais:

Estabelece penalidades criminais e administrativas para atos e atividades lesivas ao meio ambiente, responsabiliza criminalmente grandes corporações por danos que seus negócios possam causar à natureza.

- Lei 12.305 de 02 de agosto de 2010 – Política Nacional de Resíduos Sólidos: Tem como objetivo a redução do descarte de resíduos por meio de tratamento e reaproveitamento, determina que os rejeitos devem ser descartados de forma adequada, sem afetar o meio ambiente, uma responsabilidade compartilhada entre governo, empresas e sociedade. Na prática, todos os resíduos devem ser descartados antes da disposição final, e quem descumprir a lei está sujeito a penas passivas, incluindo prisão.
- Lei 6.938 de 31 de agosto de 1981 – Política Nacional do Meio Ambiente: O objetivo desta lei é regular as diversas atividades que envolvem o meio ambiente, para que a qualidade do meio ambiente seja preservada, melhorada e restaurada, beneficiando assim a vida da população e proporcionando condições para o desenvolvimento social e econômico da população.
- Lei 12.651 de 25 de maio de 2012 – Novo Código Florestal: Suprime o Código Florestal de 1965 e estabelece normas gerais sobre locais e maneiras que a vegetação pode ser explorada. Os proprietários são obrigados a manter o meio ambiente natural, mantendo espaços protegidos em propriedade privada, divididos em Área de Preservação Permanente (APP) e Reserva Legal (RL).
- Lei 9.433 de 08 de janeiro de 1997 – Política Nacional de Recursos Hídricos: Define a água como um recurso natural finito, com valor econômico. Também criou um sistema nacional de coleta, processamento, armazenamento e recuperação de informações sobre os recursos hídricos e seus fatores. Reduz o custo para as corporações e visa garantir o abastecimento de água para as presentes e futuras gerações, utilizando os recursos hídricos de forma coerente e integrada.
- Instrução Normativa IBAMA 06 de 15 de março de 2013: Regulamenta o Cadastro Técnico Federal de Atividades Potencialmente Poluidoras e Utilizadoras de Recursos Ambientais – CTF/APP. Fornece uma lista de atividades potencialmente poluidoras.
- Resolução CONAMA 237 de 19 de dezembro de 1997: Aborda o licenciamento ambiental, autoridade federal, estadual e municipal, lista atividades que requerem licenciamento, estudos ambientais, estudos de impacto ambiental e relatórios de
- Lei 7.347 de 24 de julho de 1985: Lei da Ação Civil Pública - O Ministério

da Fazenda do Brasil trata dos danos ao meio ambiente, ao consumidor e ao patrimônio artístico, turístico ou paisagístico por parte de ações civis públicas ao consumidor e ao patrimônio artístico, turístico ou paisagístico.

- Lei 11.284 de 02 de março de 2006: Trata do manejo florestal público com o objetivo de produção sustentável e estabelece os princípios do manejo florestal público, tanto para o manejo florestal quanto para o manejo sustentável.
- Lei 6.766 de 19 de dezembro de 1979 – Parcelamento do Solo Urbano: Desautoriza loteamentos urbanos em áreas de preservação ecológica, onde a poluição pode gerar perigo à saúde e em terrenos alagadiços, estabelece normas
- Lei 9.985 de 18 de julho de 2000 – Sistema Nacional de Unidades de Conservação da Natureza: Tem como objetivo a conservação de variedades de espécies biológicas e dos recursos genéticos, a preservação e restauração da diversidade de ecossistemas naturais e a promoção do desenvolvimento sustentável a partir dos recursos naturais.

Por isso, a comunicação interna em uma companhia, entre funcionários e setores é crucial para o conhecimento de incidentes ambientais pelos setores responsáveis, para que então, ações possam ser tomadas, evitando consequências negativas posteriores.

3.3 Linguagem

A linguagem utilizada no projeto foi C#, linguagem de programação de alto nível e de tipagem forte, desenvolvida pela Microsoft, começou a ser criada no final do século 20 (1999), por Anders Hejlsberg e sua equipe. Foi inicialmente nomeada de Cool, somente anos mais tarde passou a ser chamada de C#, é semelhante ao C++ e Java, utiliza a premissa de Orientação a Objetos, C# é multiplataforma isso significa que é possível a criação de aplicativos para desktop, aplicativos mobile, aplicativos para jogos eletrônicos e também para aplicações com conexão via web.

Com a linguagem C# e a IDE Visual Studio 2017 é possível resolver problemas reais, segundo Andrew Stellman e Jennifer Greene (Use a Cabeça C#, 2008) a IDE faz muito e o C# permite controle do que é feito, permite que os registros sejam mantidos, facilita a edição do código, mantém registro sobre os gráficos, áudios, ícones e outros recursos e gerenciar e interage com o banco de dados.

A linguagem utiliza o conceito de Sockets de Berkeley pela API .NET, um framework (conjunto de abstração unindo códigos específicos para o desenvolvimento de um software).

De acordo com a página de documentação .NET da Microsoft, o socket utilizado no projeto tem como referencia o System.Net.Socket e tem como função fornecer uma implementação gerenciada pelos próprios soquetes do Windows (Winsock - arquivos que permitem às aplicações do Windows se conectar à internet e à outras máquinas/computadores). O System.Net.Sockets e sua classe estabelecem um conjunto de métodos para conexão à rede. A classe possibilita a transferência de dados síncrona e assíncrona utilizando quaisquer protocolos de comunicação listados na ProtocolType.

Outras bibliotecas utilizadas no projeto é a System que é utilizada para acessar a biblioteca do Sistema, o System é o namespace raiz para tipos fundamentais do .NET. O System inclui classes que representam os tipos de dados base usados pelos aplicativos: Object (raiz da hierarquia de herança), Byte, Char, Array, Int32e String, entre outros, e correspondem aos tipos de dados primitivos que o C# utiliza

Além dos tipos de dados base, o System contém mais de 100 classes, que vão desde classes que identificam exceções à classes que lidam com conceitos de runtime: coletor de lixo e domínios de aplicativo. Logo o System também contém várias bibliotecas/namespaces de segundo nível.

Algumas outras bibliotecas utilizadas são:

- using System.Net;
- using System.Net.Sockets;
- using System.Threading;
- using System.IO;
- using System.Collections.Generic;
- using System.ComponentModel;
- using System.Data;
- using System.Drawing;
- using System.Linq;
- using System.Text;
- using System.Threading.Tasks;
- using System.Windows.Forms;

As classes de biblioteca system.net, a biblioteca principal, fornecem uma interface de programação simples para o uso de protocolos de redes as principais classes utilizadas no projeto são o IPAddress que contém o endereço de um computador em uma rede IP, TcpClient que estabelece conexões do cliente de serviços de rede TCP, método simples para conectar, enviar e receber dados (informações) no modo de bloqueio síncrono. Criado com TCP ProtocolType, deve requisitar solicitações para conexão de entrada. Podendo se conectar ao cliente de

duas maneiras e Dns que possibilita a funcionalidade da resolução de domínio simples, é estática, onde se recuperam dados de um host DNS (Sistema de Nomes de Domínio da Internet).

3.4 Design

A interface do projeto possui diversos componentes visuais, os principais são:

- Label: Possibilita adicionar legendas para outros elementos em seu projeto;
- Panel Control: Controle de contêiner para hospedar um grupo de controles filhos semelhantes;
- TextBox: Classe que permite exibir ou editar texto não formatado em um formulário;
- PictureBox: Recurso usado para exibir gráficos de um arquivo bitmap, metafile, icon, JPG, GIF ou PNG, em tempo de design ou em tempo de execução;
- Button: Conjunto de comandos pelos quais os botões de texto, botões contidos, botões de ação flutuante e botões de ícone são construídos com base no componente ButtonBase.

4 DESENVOLVIMENTO

Após a idealização do projeto foi dado início ao desenvolvimento, que por sua vez teve dois estágios separados pela equipe, front-end e back-end, sendo o primeiro para principalmente a parte visual do projeto contendo os elementos forms da aplicação, já no back-end é utilizado os sockets Berkeley para estabelecer conexão entre os chats.

Para o desenvolvimento e depuração do projeto foi utilizado apenas uma maquina teste afim de prover de forma concisa o objetivo do projeto, sendo a conexão com o servidor local por um IP estático dentro de uma variável designada para tal. A máquina utilizada no projeto final é composta de um processador AMD A4-3305M APU com placa de vídeo integrada AMD® Radeon(tm) HD 4680G, 1.90 GHz, 6GB de Memória RAM, um HDD de 500GB, Sistema Operacional Windows 7, 64 bits.

Foi utilizada a IDE Visual Studio 2017, que é um editor de código-fonte e um ambiente de desenvolvimento integrado da Microsoft para desenvolvimento de software com foco em desenvolvimento para .Net Framework e às linguagens C#, F#, Visual Basic, C e C++.

No projeto há 4 Forms, FormLogin para que o usuario consiga se conectar a o seu chat especifico o selecionando entre analista e denunciante juntamente com seu nome, estado e cidade. Em sequencia há o FormChat que é o chat do denunciante, o FormAnalista que é o responsavel por receber o chamado e conversar com o denunciante que está conectado. Por fim o FormServer que permite que o analista e o denunciante consigam se conversar entre si.

4.1 Front-End

A logo escolhida para representar o projeto foi a de um becker com uma planta em seu interior, vide imagem 3 e o nome para a aplicação é ChatFlor.

Imagem 3 — Logo do Projeto



Fonte: Freepik

No código da tela apresentada na Imagem 4 há um border radius nos botões e em alguns panels, para isso foi adicionado uma classe chamada `RJButton` e `RJPanel`, com as mesmas funcionalidades do `button` e do `panel` porém com border radius.

Como mostra na Imagem 4, o `FormServer` tem `RJButton`, `labels`, `listBox`, `pictureBox` e o `textBox`. O `label` é utilizado para desenhar na tela, `IP:` e `Porta:`, já o `listBox` é para armazenar o status log do servidor, `PictureBox` armazena a imagem antes descrita e o `textBox` serve para o usuário inserir um texto para que futuramente seja utilizado na lógica do back-end.

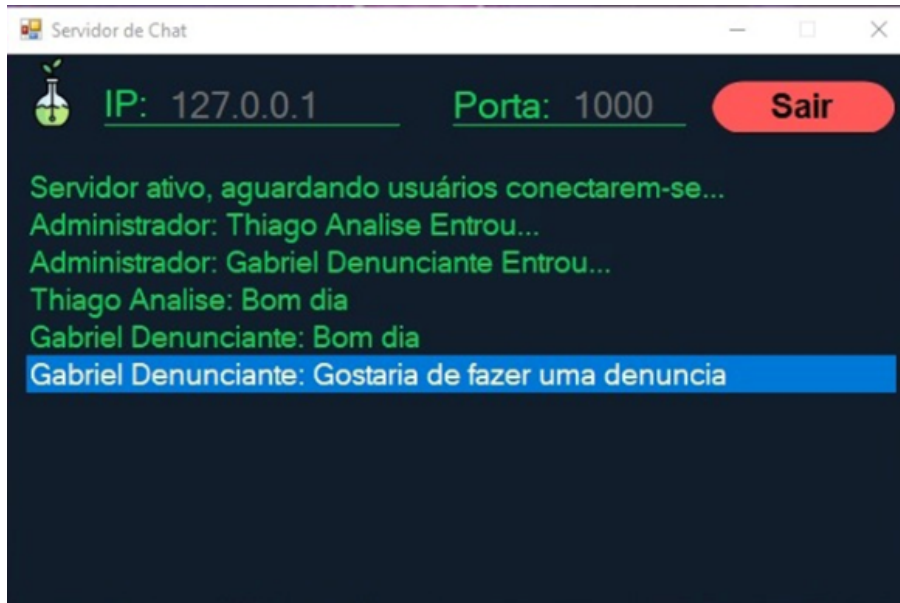
Imagem 4 — FormServer



Fonte: Os autores (2022)

A seguir na Imagem 5 mostra como é o `FormServer` quando está ativo e com outros chats abertos.

Imagem 5 — FormServer Conectado



Fonte: Os autores (2022)

Na Imagem 6, o FormLogin, é possível observar que existe o logo do projeto em picturebox junto com o título chatFlor, uma label, abaixo deles, os campos de preenchimento de dados, sendo eles Nome, Estado, Cidade e um radio-button para alternarem entre analista e denúncia e um RJButton para entrar.

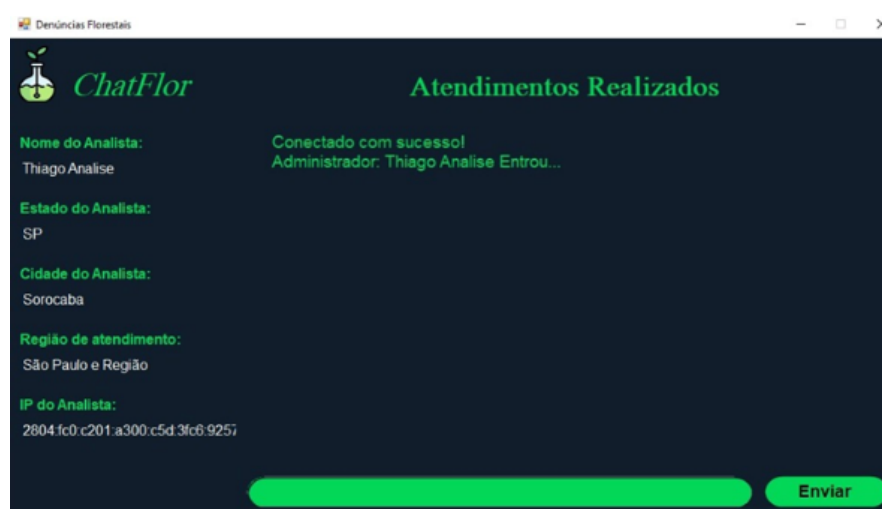
Imagem 6 — FormLogin



Fonte: Os autores (2022)

Ao preencher todas as informações exigidas no login e selecionado o radio-button “Sou um Analista” é aberto outro form com um panel para as informações do analista como seu nome, estado, cidade e o IP, junto também com o IP do analista, já no centro tem um label no canto superior escrito Atendimentos Realizados e em baixo o chat com as mensagens verdes, na parte inferior há um JPanel onde o analista consegue escrever sua mensagem e ao lado há um JButton para enviar a mensagem, como assim mostra a Imagem 7.

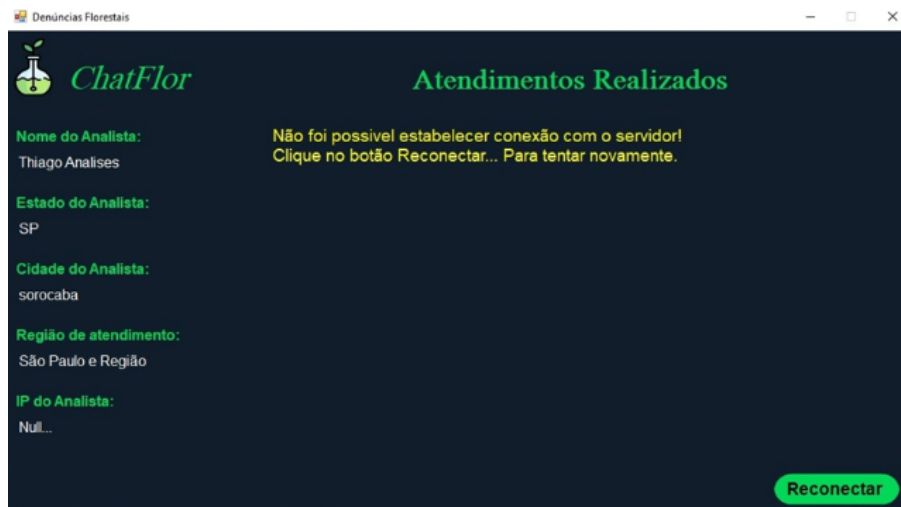
Imagem 7 — FormAnalista



Fonte: Os autores (2022)

Na Imagem 8 um aviso é mostrado quando o analista está desconectado: “Não foi possível estabelecer conexão com o servidor! Clique no botão Reconectar... Para tentar novamente.”, as mensagens do chat ficam amarelas, o botão de enviar muda para “Reconectar” e campo de digitar o texto sumir o IP do analista fica “Null...”.

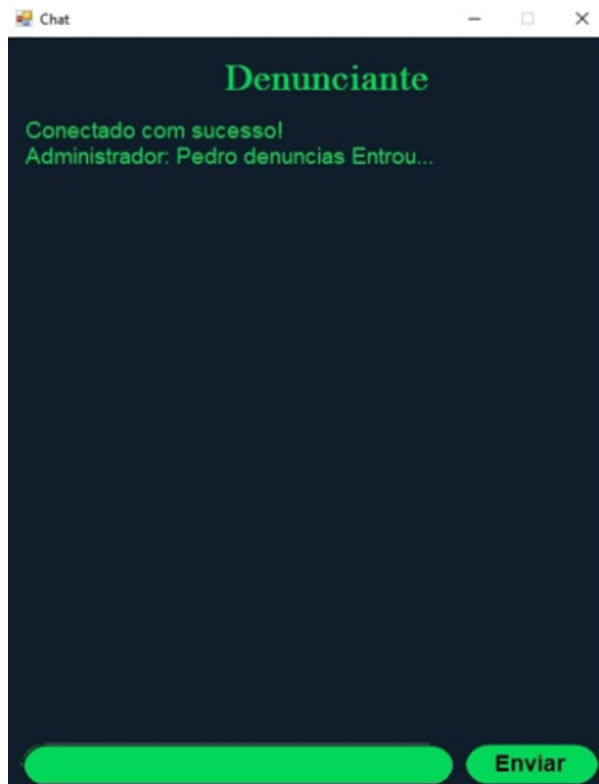
Imagem 8 — FormAnalista desconectado



Fonte: Os autores (2022)

No FormDenunciante uma label é mostrada para identificação do form embaixo do chat e no canto inferior o textBox dentro de um RJPanel para inserir um texto e ao lado um RJButton para enviar o texto, assim como mostra na Imagem 9.

Imagem 9 — FormDenunciante

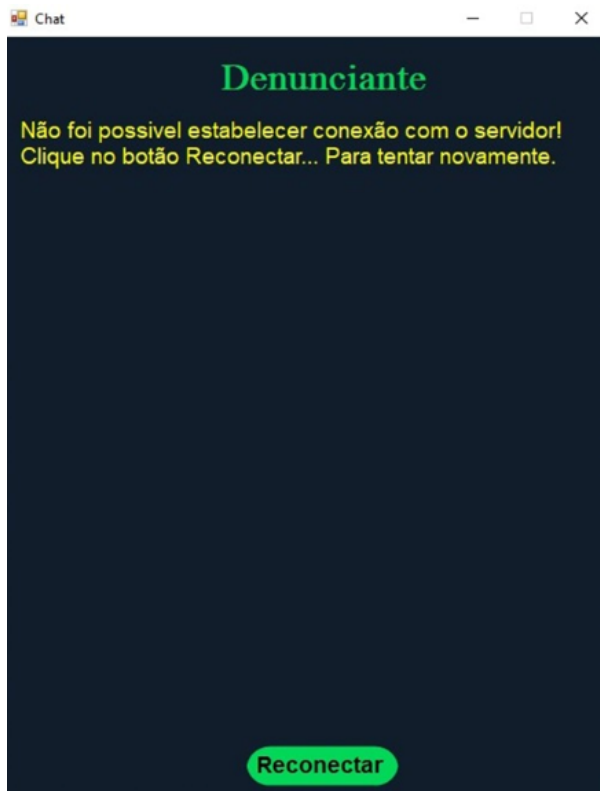


Fonte: Os autores (2022)

Na Imagem 10 é apresentada uma mensagem o denunciante desconectado

que aparece a mensagem “Não foi possível estabelecer conexão com o servidor! Clique no botão Reconectar... Para tentar novamente.” e as mensagens do chat ficam em amarelo e o botão enviar muda de posição para o centro inferior e o texto muda para “Reconectar” além de que o textBox some.

Imagem 10 — FormDenunciante desconectado



Fonte: Os autores (2022)

4.2 Back-End

O FormServer comporta a lógica de comunicação entre chats utilizando um servidor em comum, nos forms FormServer, Analista e Denunciante foram utilizados métodos das classes IPAddress e TcpClient das bibliotecas System.Net e System.Net.Sockets, utilizados a fim de estabelecer conexão entre chats.

No FormServer há um *delegate* para atualizar o status do servidor via eventos, em seguida uma variável booleana define se o servidor está conectado, ao selecionar o botão "Conectar" ele tenta uma conexão com o IP e a porta informados pelo usuário com o método IniciaAtendimento() da classe Servidor.cs, antes disso é passado ao servidor o método mainServidor_StatusChaged como parâmetro de um evento StatusChangedEventHandler que está sendo declarado em Servidor.cs. dentro do método IniciaAtendimento() ele começa a “escutar as conexões exteriores”

com base no IP e porta anteriormente informados e então com uma variável booleana chamada `ServRodando` para decidir se o server está rodando ou não é passado o valor `true`, então é iniciado uma thread com o método `MantemAtendimento` que irá manter a conexão enquanto a variável booleana `ServRodando` for verdadeira e dentro desse método `MantemAtendimento` ele vai executar a instrução de aceitar uma conexão pendente e passar como parâmetro para uma nova classe denominada `Conexao.cs`.

No construtor da classe `Conexao.cs` é enviado o parâmetro passado para uma variável da classe chamada `tcpCliente` em seguida uma nova thread é iniciada com o método `AceitaCliente()`, que lê o nome do atendente ou denunciante. Após as verificações, se o processo for completo com sucesso, é enviada a string "1" para o `TcpCliente`, onde o `FormAnalista` ou `FormDenunciante` que já tenham uma thread iniciada, para receber essa mensagem e se as verificações estiverem corretas os `form chats` atualizaram seus respectivos logs. Em seguida é passado como parâmetro o `tcpCliente` e seu nome para o `Servidor.cs` que irá adicioná-los em um hashtable de usuários e conexões cujo limite são de 30 usuários, e então ele executa o método `EnviaMensagemAdmin()` com o parâmetro hashtable de conexão cujo index é o `tcpUsuario` e uma mensagem que é "Entrou...". Nesse método, primeiro ele atualiza o log de todos os chats com a mensagem "Administrador:" mais a mensagem de Entrou... ou Saiu... que foi passada como parâmetro desse método, em seguida um é criado array de clientes TCPs do tamanho do número de clientes existentes e copia os objetos `TCPCliente` no array, por último ele vai percorrer esse array e tentar enviar a mensagem para o servidor se não conseguir ele remove o usuário.

Após tudo isso ele vai tentar enviar a mensagem para os outros chats, para isso é criado uma variável chamada `strResposta` dentro da condicional do `while` que lerá a mensagem do `TcpCliente`, e no bloco `while` se tudo estiver de acordo ele vai executar o método `EnviaMensagem()` da classe `Servidor.cs` com os parâmetros do nome do usuário mais a `strResposta`, que é muito parecida com a `EnviaMensagemAdmin()`, tendo como diferenças apenas o parâmetro passado pelo `StatusChangedEventArgs` sendo eles a o nome do usuário e a mensagem enviada, no for o que muda é a mensagem enviada para o `TcpCliente` que agora contém o nome do usuário mais a mensagem.

O back-end tanto do `FormDenunciante` e do `FormAnalista` são muito parecidos. Ambos tem o método `InicializaConexao()` que é chamado no construtor e faz a conexão `TcpServidor` com o endereço ip e a porta passados dentro das variáveis locais, em seguida há o envio do nome do usuário para o servidor e uma thread para receber a mensagem do servidor, que armazena na variável local

ConResposta, como já antes dito o servidor envia “1” para conectado e 0 para não conectado então com base nisso é feito uma simples verificação onde no qual caso seja “1” ele atualiza o chat para “Conectado com sucesso!”. Caso contrário ele mostra no chat “Não conectado:” mais o motivo e por fim ele fecha as conexões existentes. Enquanto o chat estiver conectado ele tenta atualizar o log com o que está no TcpCliente.

No FechaConexao é passado um parâmetro opcional onde no qual o método tem como objetivo finalizar todas as conexões existentes, além de sumir com o campo de texto, mudar a cor do chat e alterar o que está escrito no botão para “Reconectar”.

Supondo que o denunciante ou analista está desconectado do ChatServer, o botão de enviar irá ser alterado para “Reconectar” e quando pressionado caso o texto do botão seja reconectar ele tentar novamente estabelecer uma conexão com o servidor com o método InicializarConexao(). Mas caso já esteja conectado ao servidor tanto o botão tanto o campo de texto irá voltar ao normal e o texto do chat irá voltar a coloração normal. O método EnviarMensagem() após uma rápida verificação da mensagem digitado no campo txtMensagem ele envia a mensagem para o TcpCliente que por sua vez consegue ser lido no FormServer. Quando a aplicação FormAnalista e FormDenunciate é fechada existem dois eventos que disparam tanto os fechamentos das conexões quanto o Application.Exit(), para ter certeza de que as conexões foram fechadas corretamente.

A diferença entre o FormAnalista e o FormDenunciante pelo ponto de vista do back-end é que o usuário consegue ver o ip do seu computador.

5 CONCLUSÃO

De acordo com Lamarck (1744-1829), o meio cria necessidades que induzem a mudança de hábitos, essa teoria pode ser aplicada em diversos âmbitos, inclusive no desenvolvimento de novas tecnologias, a partir da necessidade de comunicação entre territórios diferentes nos Estados Unidos, nasceu a comunicação em Rede, para fins militares, assim como, diante de situações presenciadas em Chão de Fábrica, nasceu a necessidade de comunicação direta entre setores.

A globalização tornou a necessidade de inovação veloz, a extensa disposição de informações implica no imediatismo da comunicação, por isso ferramentas como essa tem se tornado essenciais no dia-a-dia de todos, e na velocidade em que são criadas, são também ultrapassadas por novas

Por outro lado, à medida que a população se desenvolve, os recursos naturais são utilizados de forma desenfreada para suprir o consumismo, portanto, a conscientização e cuidado com os recursos ainda disponíveis é de extrema importância e uma preocupação de todos, por isso, unindo esses dois tópicos, foi desenvolvido o ChatFlor que teve como objetivo principal a denúncia de incidentes ambientais, troca de mensagens com caracteres especiais como: "#, *, &, \$", com a comunicação estabelecida o Usuário/Funcionário conhecido como Denunciante relata o problema envolvendo questões ambientais para o Analista que grava o problema e propõe uma solução.

Para este projeto o estudo de apostilas, livros, vídeo aulas e outras atividades complementares foi essencial, com o conhecimento adquirido notou-se que a comunicação e seus meios se tornam uma ferramenta valiosa para resolver problemas cotidianos.

Comunicação pode ser compreendida como a troca de pensamentos, ideias ou sentimentos entre pessoas, mas só é efetiva quando a mensagem é entendida pelo receptor, por isso, a linguagem deve se adaptar de acordo com o âmbito em que se aplica, com isso, o desenvolvimento de uma ferramenta para comunicação em rede torna-se não somente sobre o código e uma interface e sim sobre os desafios da comunicação em um país que ocupa o 60º lugar em educação (OCDE, 2017).

As empresas são alicerce essencial, agem diretamente no impacto da sociedade e no progresso econômico de um país, portanto, devem ser as primeiras a se preocuparem com quesitos sociais e ambientais, já que grande parte da poluição e danos aos ecossistemas que contribuem para o efeito estufa se deve à produção de empresas.

REFERÊNCIAS

adegeo. **Visão geral de TextBox**. Microsoft. 2022. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/desktop/wpf/controls/textbox-overview?view=netframeworkdesktop-4.8>. Acesso em: 8 abr. 2022.

BURROWS, Chris . **C# 4.0** : Novos recursos do C# no .NET Framework 4. Microsoft. 2015. Disponível em: <https://docs.microsoft.com/pt-br/archive/msdn-magazine/2010/july/csharp-4-0-new-csharp-features-in-the-net-framework-4>. Acesso em: 21 abr. 2022.

C#: Dicionário do Programador. Código Fonte TV. Código Fonte TV. Petrópolis, 2020 (8min). Disponível em: <https://www.youtube.com/watch?v=NXVQasys0B8>. Acesso em: 8 abr. 2022.

CHAND, Mahesh. **Panel In C#**. C#Corner. 2018. Disponível em: <https://www.c-sharpcorner.com/UploadFile/mahesh/panel-in-C-Sharp/>. Acesso em: 22 abr. 2022.

DANTAS, Mário. **Tecnologias de redes de comunicação e computadores**, f. 164. 328 p.

DEVMEDIA. **A evolução da linguagem de programação C#**. DevMedia. Disponível em: [https://www.devmedia.com.br/a-evolucao-da-linguagem-de-programacao-csharp/28639#:~:text=O%20desenvolvimento%20do%20C%23%20\(pronuncia,uma%20nova%20linguagem%20de%20programa%C3%A7%C3%A3o](https://www.devmedia.com.br/a-evolucao-da-linguagem-de-programacao-csharp/28639#:~:text=O%20desenvolvimento%20do%20C%23%20(pronuncia,uma%20nova%20linguagem%20de%20programa%C3%A7%C3%A3o). Acesso em: 21 abr. 2022.

DEVMEDIA. **Linguagem C#**: Introdução ao C#. DevMedia. Disponível em: <https://www.devmedia.com.br/guia/linguagem-csharp/38152>. Acesso em: 8 abr. 2022.

FREEPIK. **Biotechnology**. Flaticon. Disponível em: https://www.flaticon.es/icono-gratis/biotecnologia_6679303. Acesso em: 11 abr. 2022.

L, Andrei. **Protocolos de Rede**: O Que São, Como Funcionam e Tipos de Protocolos de Internet. weblink. 2020. Disponível em: <https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/>. Acesso em: 2 abr. 2022.

LIMA, Eugenio . **C# E .Net**: Guia Do Desenvolvedor. 1 ed. Rio de Janeiro: Campus, 2002. 358 p.

MICROSOFT. **Dns Classe**. .Net. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/api/system.net.dns?view=net-6.0>. Acesso em: 22 abr. 2022.

MICROSOFT. **IPAddress Classe**. .Net. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/api/system.net.ipaddress?view=net-6.0>. Acesso em: 1 abr. 2022.

MICROSOFT. **TcpClient Classe**. .Net. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/api/system.net.sockets.tcpclient?view=net-6.0>. Acesso em: 15 abr. 2022.

MICROSOFT. **Visão geral da biblioteca de classes do .NET**. Microsoft. 2022. Disponível em: . Acesso em: 8 abr. 2022.

NASCIMENTO, Anderson . **O que é API?**. CanalTech. 2014. Disponível em: <https://canaltech.com.br/software/o-que-e-api/#:~:text=API%C3%A9%20um%20conjunto%20de,Interface%20de%20Programas%C3%A7%C3%A3o%20de%20Aplicativos%22>. Acesso em: 15 abr. 2022.

NET SUPORT. **REDE DE COMPUTADORES: DEFINIÇÃO E TIPOS**. CRT-RJ. Rio de Janeiro, 2021. Disponível em: <https://www.crtrj.gov.br/rede-de-computadores-definicao-e-tipos/>. Acesso em: 17 abr. 2022.

PANTUZA, Gustavo . **O QUE SÃO E COMO FUNCIONAM OS SOCKETS**: Aprenda como os sockets funcionam em baixo nível e como ele é fundamental para os sistemas distribuídos. 2017. Disponível em: <https://blog.pantuza.com/artigos/o-que-sao-e-como-funcionam-os-sockets>. Acesso em: 21 abr. 2022.

REIS, Tiago. **BOLSA DE VALORES**: Tripé da sustentabilidade: entenda o que é e qual a sua importância. Suno. 2021. Disponível em: <https://www.suno.com.br/artigos/tripe-da-sustentabilidade/>. Acesso em: 15 abr. 2022.

teamques10. **Short Note**: Berkeley Socket. India, 2017. Disponível em: <https://www.ques10.com/p/3854/short-note-berkeley-socket/>. Acesso em: 21 abr. 2022.

ANEXO A — IMAGEM LOGO UTILIZADA

ANEXO A — imagem logo utilizada



ANEXO B – Projeto Chat Server – ChatServer

ANEXO BA – Código da Classe Conexao.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
namespace ChatServer
{
    // Esta classe trata as conexões, serão tantas quanto as instâncias dos
    usuários conectados
    class Conexao
    {
        TcpClient tcpCliente;
        // A thread que ira enviar a informação para o cliente
        private Thread thrSender;
        private StreamReader srReceptor;
        private StreamWriter swEnviador;
```

```

private string usuarioAtual;
private string strResposta;
// O construtor da classe que toma a conexão TCP
public Conexao(TcpClient tcpCon)
{
    tcpCliente = tcpCon;
    // A thread que aceita o cliente e espera a mensagem
    thrSender = new Thread(AceitaCliente);
    thrSender.IsBackground = true;
    // A thread chama o método AceitaCliente()
    thrSender.Start();
}
private void FechaConexao()
{
    // Fecha os objetos abertos
    tcpCliente.Close();
    srReceptor.Close();
    swEnviador.Close();
}
// Ocorre quando um novo cliente é aceito
private void AceitaCliente()
{
    srReceptor = new StreamReader(tcpCliente.GetStream());
    swEnviador = new StreamWriter(tcpCliente.GetStream());
    // Lê a informação da conta do cliente
    usuarioAtual = srReceptor.ReadLine();
    // temos uma resposta do cliente
    if (usuarioAtual != "")
    {
        // Armazena o nome do usuário na hash table
        if (Servidor.htUsuarios.Contains(usuarioAtual))
        {
            // 0 => significa não conectado
            swEnviador.WriteLine("0|Este nome de usuário já existe.");
            swEnviador.Flush();
            FechaConexao();
            return;
        }
    }
}

```

```

else if (usuarioAtual == "Administrator")
{
    // 0 => não conectado
    swEnviador.WriteLine("0|Este nome de usuário é reservado.");
    swEnviador.Flush();
    FechaConexao();
    return;
}
else
{
    // 1 => conectou com sucesso
    swEnviador.WriteLine("1");
    swEnviador.Flush();
    // Inclui o usuário na hash table e inicia a escuta de suas
mensagens
    Servidor.IncluiUsuario(tcpCliente, usuarioAtual);
}
}
else
{
    FechaConexao();
    return;
}
try
{
    // Continua aguardando por uma mensagem do usuário
    while ((strResposta = srReceptor.ReadLine()) != "")
    {
        // Se for inválido remove-o
        if (strResposta == null)
        {
            Servidor.RemoveUsuario(tcpCliente);
        }
        else
        {
            // envia a mensagem para todos os outros usuários
            Servidor.EnviaMensagem(usuarioAtual, strResposta);
        }
    }
}

```



```

    }
    catch
    {
        // Se houve um problema com este usuário desconecta-o
        Servidor.RemoveUsuario(tcpCliente);
    }
}
}
}

```

ANEXO BB – Código da Classe Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace ChatServer
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FormServer());
        }
    }
}

```

ANEXO BC – Código da Classe RJButton.cs

```

using System;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;
namespace prjAps
{

```

```

public class RJButton : Button
{
    //Fields
    private int borderSize = 0;
    private int borderRadius = 40;
    private Color borderColor = Color.PaleVioletRed;
    public RJButton()
    {
        this.FlatStyle = FlatStyle.Flat;
        this.FlatAppearance.BorderSize = 0;
        this.Size = new Size(150, 40);
        this.BackColor = Color.MediumSlateBlue;
        this.ForeColor = Color.White;
    }
    //Methods
    private GraphicsPath GetFigurePath(RectangleF rect, float radius)
    {
        GraphicsPath path = new GraphicsPath();
        path.StartFigure();
        path.AddArc(rect.X, rect.Y, radius, radius, 180, 90);
        path.AddArc(rect.Width - radius, rect.Y, radius, radius, 270, 90);
        path.AddArc(rect.Width - radius, rect.Height - radius, radius, radius, 0,
90);
        path.AddArc(rect.X, rect.Height - radius, radius, radius, 90, 90);
        path.CloseFigure();
        return path;
    }
    protected override void OnPaint(PaintEventArgs pevent)
    {
        base.OnPaint(pevent);
        pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
        RectangleF rectSurface = new RectangleF(0, 0, this.Width, this.Height);
        RectangleF rectBorder = new RectangleF(1, 1, this.Width - 0.8F,
this.Height - 1);
        if (borderRadius > 2) // Rounded panel
        {
            using (GraphicsPath pathSurface = GetFigurePath(rectSurface,
borderRadius))

```

```

        using (GraphicsPath pathBorder = GetFigurePath(rectBorder,
borderRadius - 1F))
        using (Pen penSurface = new Pen(this.Parent.BackColor, 2))
        using (Pen penBorder = new Pen(borderColor, borderSize))
        {
            penBorder.Alignment = PenAlignment.Inset;
            //button surface
            this.Region = new Region(pathSurface);
            // Draw surface border for HD result
            pevent.Graphics.DrawPath(penSurface, pathSurface);
            // panel border
            if (borderSize >= 1)
                // draw control border
                pevent.Graphics.DrawPath(penBorder, pathBorder);
        }
    }
    else // Normal button
    {
        // button surface
        this.Region = new Region(rectSurface);
        // button border
        if (borderSize >= 1)
        {
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                pevent.Graphics.DrawRectangle(penBorder, 0, 0, this.Width - 1,
this.Height - 1);
            }
        }
    }
}
protected override void OnHandleCreated(EventArgs e)
{
    base.OnHandleCreated(e);
    this.Parent.BackColorChanged += new
EventHandler(Container_BackColorChanged);
}

```

```

private void Container_BackColorChanged(object sender, EventArgs e)
{
    if (this.DesignMode)
        this.Invalidate();
    }
}

```

ANEXO BD – Código da Classe StatusChangedEventArgs.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ChatServer
{
    // Trata os argumentos para o evento StatusChanged
    public class StatusChangedEventArgs : EventArgs
    {
        // Estamos interessados na mensagem descrevendo o evento
        private string EventMsg;
        // Propriedade para retornar e definir um mensagem do evento
        public string EventMessage
        {
            get { return EventMsg; }
            set { EventMsg = value; }
        }
        // Construtor para definir a mensagem do evento
        public StatusChangedEventArgs(string strEventMsg)
        {
            EventMsg = strEventMsg;
        }
    }
}

```

ANEXO BE – Código da Classe Servidor.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;

```

```

using System.Net;
using System.Net.Sockets;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
namespace ChatServer
{
    // Este delegate é necessário para especificar os parametros que estamos
    // passando com o nosso evento
    public delegate void StatusChangedEventHandler(object sender,
    StatusChangedEventArgs e);
    class Servidor
    {
        // Esta hash table armazena os usuários e as conexões
        // (acessado/consultado por usuário)
        public static Hashtable htUsuarios = new Hashtable(30); // 30 usuarios é o
        // limite definido

        // Esta hash table armazena os usuários e as conexões
        // (acessada/consultada por conexão)
        public static Hashtable htConexoes = new Hashtable(30); // 30 usuários é o
        // limite definido

        // armazena o endereço IP passado
        private IPAddress enderecolP;
        private int portaHost;
        private TcpClient tcpCliente;

        // O evento e o seu argumento irá notificar o formulário quando um usuário
        // se conecta, desconecta, envia uma mensagem, etc
        public static event StatusChangedEventHandler StatusChanged;
        private static StatusChangedEventArgs e;

        // O construtor define o endereço IP para aquele retornado pela
        // instanciação do objeto
        public Servidor(IPAddress endereco, int porta)
        {
            enderecolP = endereco;
            portaHost = porta;
        }

        // A thread que ira tratar o escutador de conexões

```

```

private Thread thrListener;
// O objeto TCP object que escuta as conexões
private TcpListener tlsCliente;
// Ira dizer ao laço while para manter a monitoração das conexões
bool ServRodando = false;
// Inclui o usuário nas tabelas hash
public static void IncluiUsuario(TcpClient tcpUsuario, string strUsername)
{
    // Primeiro inclui o nome e conexão associada para ambas as hash
tables
    Servidor.htUsuarios.Add(strUsername, tcpUsuario);
    Servidor.htConexoes.Add(tcpUsuario, strUsername);
    // Informa a nova conexão para todos os usuário e para o formulário do
servidor
    EnviaMensagemAdmin(htConexoes[tcpUsuario] + " Entrou...");
}
// Remove o usuário das tabelas (hash tables)
public static void RemoveUsuario(TcpClient tcpUsuario)
{
    // Se o usuário existir
    if (htConexoes[tcpUsuario] != null)
    {
        // Primeiro mostra a informação e informa os outros usuários sobre a
conexão
        EnviaMensagemAdmin(htConexoes[tcpUsuario] + " Saiu...");
        // Remove o usuário da hash table
        Servidor.htUsuarios.Remove(Servidor.htConexoes[tcpUsuario]);
        Servidor.htConexoes.Remove(tcpUsuario);
    }
}

// Este evento é chamado quando queremos disparar o evento
StatusChanged
public static void OnStatusChanged(StatusChangedEventArgs e)
{
    StatusChangedEventHandler statusHandler = StatusChanged;
    if (statusHandler != null)
    {
        // invoca o delegate
        statusHandler(null, e);
    }
}

```

```

    }
}
// Envia mensagens administrativas
public static void EnviaMensagemAdmin(string Mensagem)
{
    StreamWriter swSenderSender;
    // Exibe primeiro na aplicação
    e = new StatusChangedEventArgs($"Administrador: {Mensagem}");
    OnStatusChanged(e);
    // Cria um array de clientes TCPs do tamanho do numero de clientes
    existentes
    TcpClient[] tcpClientes = new TcpClient[Servidor.htUsuarios.Count];
    // Copia os objetos TcpClient no array
    Servidor.htUsuarios.Values.CopyTo(tcpClientes, 0);
    // Percorre a lista de clientes TCP
    for (int i = 0; i < tcpClientes.Length; i++)
    {
        // Tenta enviar uma mensagem para cada cliente
        try
        {
            // Se a mensagem estiver em branco ou a conexão for nula sai...
            if (Mensagem.Trim() == "" || tcpClientes[i] == null)
            {
                continue;
            }
            // Envia a mensagem para o usuário atual no laço
            swSenderSender = new StreamWriter(tcpClientes[i].GetStream());
            swSenderSender.WriteLine($"Administrador: {Mensagem}\n");
            swSenderSender.Flush();
            swSenderSender = null;
        }
        catch
        {
            // Se houver um problema , o usuário não existe , então remove-o
            RemoveUsuario(tcpClientes[i]);
        }
    }
}
// Envia mensagens de um usuário para todos os outros

```

```

public static void EnviaMensagem(string Origem, string Mensagem)
{
    StreamWriter swSenderSender;
    // Primeiro exibe a mensagem na aplicação
    e = new StatusChangedEventArgs($"{Origem}: {Mensagem}");
    OnStatusChanged(e);
    // Cria um array de clientes TCPs do tamanho do numero de clientes
    existentes
    TcpClient[] tcpClientes = new TcpClient[Servidor.htUsuarios.Count];
    // Copia os objetos TcpClient no array
    Servidor.htUsuarios.Values.CopyTo(tcpClientes, 0);
    // Percorre a lista de clientes TCP
    for (int i = 0; i < tcpClientes.Length; i++)
    {
        // Tenta enviar uma mensagem para cada cliente
        try
        {
            // Se a mensagem estiver em branco ou a conexão for nula sai...
            if (Mensagem.Trim() == "" || tcpClientes[i] == null)
            {
                continue;
            }
            // Envia a mensagem para o usuário atual no laço
            swSenderSender = new StreamWriter(tcpClientes[i].GetStream());
            swSenderSender.WriteLine($"{Origem}: {Mensagem}");
            swSenderSender.Flush();
            swSenderSender = null;
        }
        catch // Se houver um problema , o usuário não existe , então remove-
        o
        {
            RemoveUsuario(tcpClientes[i]);
        }
    }
}

public void IniciaAtendimento()
{
    try

```



```

    {
        IPAddress ipaLocal = enderecoIP;
        int portaLocal = portaHost;
        tlsCliente = new TcpListener(ipaLocal, portaLocal);
        // Escuta as conexões
        tlsCliente.Start();
        // O laço While verifica se o servidor esta rodando antes de checar as
conexões
        ServRodando = true;
        // Inicia uma nova tread que hospeda o listener
        thrListener = new Thread(MantemAtendimento);
        thrListener.IsBackground = true;
        thrListener.Start();
    }
    catch (Exception ex)
    {
    }
}
private void MantemAtendimento()
{
    // Enquanto o servidor estiver rodando
    while (ServRodando)
    {
        // Aceita uma conexão pendente
        tcpCliente = tlsCliente.AcceptTcpClient();
        // Cria uma nova instância da conexão
        Conexao newConnection = new Conexao(tcpCliente);
    }
}
}
}

```

ANEXO BF – Código da Classe FormServer.Designer.cs

```

namespace ChatServer
{
    partial class FormServer
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>

```

```

private System.ComponentModel.IContainer components = null;
/// <summary>
/// Clean up any resources being used.
/// </summary>
    /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(FormServer));
    this.txtIP = new System.Windows.Forms.TextBox();
    this.listaLog = new System.Windows.Forms.ListBox();
    this.btnStartServer = new prjAps.RJButton();
    this.pictureBox1 = new System.Windows.Forms.PictureBox();
    this.numPorta = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.label2 = new System.Windows.Forms.Label();
    this.panel1 = new System.Windows.Forms.Panel();
    this.panel2 = new System.Windows.Forms.Panel();

    ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
    this.SuspendLayout();
    //
    // txtIP
    //

```

```

        this.txtIP.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.txtIP.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.txtIP.Font = new System.Drawing.Font("Microsoft Sans Serif", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.txtIP.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(5))),
((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.txtIP.Location = new System.Drawing.Point(105, 18);
        this.txtIP.Name = "txtIP";
        this.txtIP.Size = new System.Drawing.Size(145, 28);
        this.txtIP.TabIndex = 0;
        this.txtIP.Text = "127.0.0.1";
        //
        // listaLog
        //
        this.listaLog.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.listaLog.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.listaLog.Font = new System.Drawing.Font("Microsoft Sans Serif",
14F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));
        this.listaLog.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.listaLog.FormattingEnabled = true;
        this.listaLog.HorizontalScrollbar = true;
        this.listaLog.ItemHeight = 24;
        this.listaLog.Location = new System.Drawing.Point(13, 73);
        this.listaLog.Name = "listaLog";
        this.listaLog.Size = new System.Drawing.Size(559, 264);
        this.listaLog.TabIndex = 4;
        //
        // btnStartServer
        //
        this.btnStartServer.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.btnStartServer.FlatAppearance.BorderSize = 0;
        this.btnStartServer.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
        this.btnStartServer.Font = new System.Drawing.Font("Arial", 16F,
System.Drawing.FontStyle.Bold);

```

```

this.btnStartServer.ForeColor = System.Drawing.Color.Black;
this.btnStartServer.Location = new System.Drawing.Point(452, 16);
this.btnStartServer.Name = "btnStartServer";
this.btnStartServer.Size = new System.Drawing.Size(120, 35);
this.btnStartServer.TabIndex = 4;
this.btnStartServer.Text = "Iniciar";
this.btnStartServer.UseVisualStyleBackColor = false;
this.btnStartServer.Click += new
System.EventHandler(this.btnStartServer_Click);
//
// pictureBox1
//
this.pictureBox1.Image = ((System.Drawing.Image)
(resources.GetObject("pictureBox1.Image")));
this.pictureBox1.Location = new System.Drawing.Point(7, -1);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(45, 52);
this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.Zoom;
this.pictureBox1.TabIndex = 32;
this.pictureBox1.TabStop = false;
//
// numPorta
//
this.numPorta.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
this.numPorta.BorderStyle = System.Windows.Forms.BorderStyle.None;
this.numPorta.Font = new System.Drawing.Font("Microsoft Sans Serif",
18F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));
this.numPorta.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
this.numPorta.Location = new System.Drawing.Point(364, 18);
this.numPorta.Name = "numPorta";
this.numPorta.Size = new System.Drawing.Size(58, 28);
this.numPorta.TabIndex = 33;
this.numPorta.Text = "1000";
//

```

```

// label1
//
this.label1.AutoSize = true;
this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label1.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
this.label1.Location = new System.Drawing.Point(58, 17);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(41, 29);
this.label1.TabIndex = 34;
this.label1.Text = "IP:";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 18F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label2.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
this.label2.Location = new System.Drawing.Point(282, 18);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(76, 29);
this.label2.TabIndex = 35;
this.label2.Text = "Porta:";
//
// panel1
//
this.panel1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
this.panel1.Location = new System.Drawing.Point(63, 45);
this.panel1.Name = "panel1";
this.panel1.Size = new System.Drawing.Size(190, 1);
this.panel1.TabIndex = 36;
//
// panel2
//
this.panel2.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));

```

```

this.panel2.Location = new System.Drawing.Point(287, 45);
this.panel2.Name = "panel2";
this.panel2.Size = new System.Drawing.Size(150, 1);
this.panel2.TabIndex = 37;
//
// FormServer
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(19)))),
((int)(((byte)(30)))), ((int)(((byte)(45)))));
this.ClientSize = new System.Drawing.Size(584, 355);
this.Controls.Add(this.panel2);
this.Controls.Add(this.panel1);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.numPorta);
this.Controls.Add(this.pictureBox1);
this.Controls.Add(this.btnStartServer);
this.Controls.Add(this.listaLog);
this.Controls.Add(this.txtIP);

this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
this.MaximizeBox = false;
this.Name = "FormServer";

this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
this.Text = "Servidor de Chat";
        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();
    }
#endregion
private System.Windows.Forms.TextBox txtIP;
private System.Windows.Forms.ListBox listaLog;
private prjAps.RJButton btnStartServer;
private System.Windows.Forms.PictureBox pictureBox1;

```

```

        private System.Windows.Forms.TextBox numPorta;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Panel panel1;
        private System.Windows.Forms.Panel panel2;
    }
}

```

ANEXO BG – Código da Classe FormServer.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace ChatServer
{
    public partial class FormServer : Form
    {
        private delegate void AtualizaStatusCallback(string strMensagem);
        bool conectado = false;
        public FormServer()
        {
            InitializeComponent();
        }
        private void btnStartServer_Click(object sender, EventArgs e)
        {
            if (conectado)
            {
                Application.Exit();
                return;
            }
            if (txtIP.Text == string.Empty)
            {
                MessageBox.Show("Informe o endereço IP.");
                txtIP.Focus();
            }
        }
    }
}

```

```

        return;
    }
    try
    {
        IPAddress enderecolP = IPAddress.Parse(txtIP.Text);
        int portaHost = int.Parse(numPorta.Text);
        Servidor mainServidor = new Servidor(enderecolP, portaHost);
        // Vincula o evento StatusChanged a atualização do formulário
        (mainServidor_StatusChanged)
        Servidor.StatusChanged += new
        StatusChangedEventHandler(mainServidor_StatusChanged);
        // Inicia o atendimento das conexões
        mainServidor.IniciaAtendimento();
        // Mostra que nos iniciamos o atendimento para conexões
        listaLog.Items.Add("Servidor ativo, aguardando usuários conectarem-
se...");

        listaLog.SetSelected(listaLog.Items.Count - 1, true);
    }
    catch (Exception ex)
    {
        listaLog.Items.Add("Erro de conexão : " + ex.Message);
        listaLog.SetSelected(listaLog.Items.Count - 1, true);
        return;
    }
    conectado = true;
    txtIP.Enabled = false;
    numPorta.Enabled = false;
    btnStartServer.BackColor = Color.FromArgb(255, 88, 88);
    btnStartServer.Text = "Sair";
}

public void mainServidor_StatusChanged(object sender,
StatusChangedEventArgs e)
{
    // Chama o método que atualiza o formulário
    this.Invoke(new AtualizaStatusCallback(this.AtualizaStatus), new object[]
{ e.EventMessage });
}
private void AtualizaStatus(string strMensagem)

```



```

    {
        // Atualiza o logo com mensagens
        listaLog.Items.Add(strMensagem);
        listaLog.SetSelected(listaLog.Items.Count - 1, true);
    }
}

```

ANEXO C – Projeto prjAps – prjAps

ANEXO CA – Código da Classe Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace prjAps
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new FormLogin());
        }
    }
}

```

ANEXO CB – Código da Classe RJButton.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

```

```

namespace prjAps
{
    public class RJButton : Button
    {
        //Fields
        private int borderSize = 0;
        private int borderRadius = 40;
        private Color borderColor = Color.PaleVioletRed;
        public RJButton()
        {
            this.FlatStyle = FlatStyle.Flat;
            this.FlatAppearance.BorderSize = 0;
            this.Size = new Size(150, 40);
            this.BackColor = Color.MediumSlateBlue;
            this.ForeColor = Color.White;
        }
        //Methods
        private GraphicsPath GetFigurePath(RectangleF rect, float radius)
        {
            GraphicsPath path = new GraphicsPath();
            path.StartFigure();
            path.AddArc(rect.X, rect.Y, radius, radius, 180, 90);
            path.AddArc(rect.Width-radius, rect.Y, radius, radius, 270, 90);
            path.AddArc(rect.Width-radius, rect.Height-radius, radius, radius, 0, 90);
            path.AddArc(rect.X, rect.Height-radius, radius, radius, 90, 90);
            path.CloseFigure();
            return path;
        }
        protected override void OnPaint(PaintEventArgs pevent)
        {
            base.OnPaint(pevent);
            pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
            RectangleF rectSurface = new RectangleF(0, 0, this.Width, this.Height);
            RectangleF rectBorder = new RectangleF(1, 1, this.Width - 0.8F,
this.Height - 1);
            if (borderRadius > 2) // Rounded panel
            {
                using (GraphicsPath pathSurface = GetFigurePath(rectSurface,

```

```

borderRadius))
        using (GraphicsPath pathBorder = GetFigurePath(rectBorder,
borderRadius - 1F))
            using (Pen penSurface = new Pen(this.Parent.BackColor, 2))
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                //button surface
                this.Region = new Region(pathSurface);
                // Draw surface border for HD result
                pevent.Graphics.DrawPath(penSurface, pathSurface);
                // panel border
                if (borderSize >= 1)
                    // draw control border
                    pevent.Graphics.DrawPath(penBorder, pathBorder);
            }
        }
    else // Normal button
    {
        // button surface
        this.Region = new Region(rectSurface);
        // button border
        if (borderSize >= 1)
        {
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                pevent.Graphics.DrawRectangle(penBorder, 0, 0, this.Width - 1,
this.Height - 1);
            }
        }
    }
}
protected override void OnHandleCreated(EventArgs e)
{
    base.OnHandleCreated(e);
    this.Parent.BackColorChanged += new
EventHandler(Container_BackColorChanged);
}

```

```

private void Container_BackColorChanged(object sender, EventArgs e)
{
    if (this.DesignMode)
        this.Invalidate();
    }
}
}

```

ANEXO CC – Código da Classe RJPanel.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;
namespace prjAps
{
    public class RJPanel : Panel
    {
        //Fields
        private int borderSize = 0;
        private int borderRadius = 40;
        private Color borderColor = Color.PaleVioletRed;
        public RJPanel()
        {
            this.BorderStyle = BorderStyle.FixedSingle;
            this.Size = new Size(150, 40);
            this.BackColor = Color.MediumSlateBlue;
            this.ForeColor = Color.White;
        }
        //Methods
        private GraphicsPath GetFigurePath(RectangleF rect, float radius)
        {
            GraphicsPath path = new GraphicsPath();
            path.StartFigure();
            path.AddArc(rect.X, rect.Y, radius, radius, 180, 90);
            path.AddArc(rect.Width - radius, rect.Y, radius, radius, 270, 90);
        }
    }
}

```

```

        path.AddArc(rect.Width - radius, rect.Height - radius, radius, radius, 0,
90);

        path.AddArc(rect.X, rect.Height - radius, radius, radius, 90, 90);
        path.CloseFigure();
        return path;
    }
    protected override void OnPaint(PaintEventArgs pevent)
    {
        base.OnPaint(pevent);
        pevent.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
        RectangleF rectSurface = new RectangleF(0, 0, this.Width, this.Height);
        RectangleF rectBorder = new RectangleF(1, 1, this.Width - 0.8F,
this.Height - 1);
        if (borderRadius > 2) // Rounded panel
        {
            using (GraphicsPath pathSurface = GetFigurePath(rectSurface,
borderRadius))
            using (GraphicsPath pathBorder = GetFigurePath(rectBorder,
borderRadius - 1F))
            using (Pen penSurface = new Pen(this.Parent.BackColor, 2))
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                //button surface
                this.Region = new Region(pathSurface);
                // Draw surface border for HD result
                pevent.Graphics.DrawPath(penSurface, pathSurface);
                // panel border
                if (borderSize >= 1)
                // draw control border
                pevent.Graphics.DrawPath(penBorder, pathBorder);
            }
        }
        else // Normal button
        {
            // button surface
            this.Region = new Region(rectSurface);
            // button border
            if (borderSize >= 1)

```

```

        {
            using (Pen penBorder = new Pen(borderColor, borderSize))
            {
                penBorder.Alignment = PenAlignment.Inset;
                pevent.Graphics.DrawRectangle(penBorder, 0, 0, this.Width - 1,
this.Height - 1);
            }
        }
    }
}
protected override void OnHandleCreated(EventArgs e)
{
    base.OnHandleCreated(e);
    this.Parent.BackColorChanged += new
EventHandler(Container_BackColorChanged);
}
private void Container_BackColorChanged(object sender, EventArgs e)
{
    if (this.DesignMode)
        this.Invalidate();
}
}
}

```

ANEXO CD – Código da Classe FormLogin.Designer.cs

```

namespace prjAps
{
    partial class FormLogin
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)

```

```

{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(FormLogin));
    this.txf_cidade = new System.Windows.Forms.TextBox();
    this.txf_estado = new System.Windows.Forms.TextBox();
    this.panel7 = new System.Windows.Forms.Panel();
    this.panel5 = new System.Windows.Forms.Panel();
    this.panel1 = new System.Windows.Forms.Panel();
    this.labelTitulo = new System.Windows.Forms.Label();
    this.pictureBox1 = new System.Windows.Forms.PictureBox();
    this.ratioBtn_denuncia = new System.Windows.Forms.RadioButton();
    this.ratioBtn_analista = new System.Windows.Forms.RadioButton();
    this.txf_nome = new System.Windows.Forms.TextBox();
    this.label_cidade = new System.Windows.Forms.Label();
    this.label_estado = new System.Windows.Forms.Label();
    this.label_nome = new System.Windows.Forms.Label();
    this.btnEntrar = new prjAps.RJButton();
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
    this.SuspendLayout();
    //
    // txf_cidade
    //
    this.txf_cidade.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
    this.txf_cidade.BorderStyle = System.Windows.Forms.BorderStyle.None;

```

```

        this.txf_cidade.Font = new System.Drawing.Font("Microsoft Sans Serif",
11.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));

        this.txf_cidade.ForeColor = System.Drawing.Color.White;
        this.txf_cidade.Location = new System.Drawing.Point(126, 302);
        this.txf_cidade.Name = "txf_cidade";
        this.txf_cidade.Size = new System.Drawing.Size(177, 17);
        this.txf_cidade.TabIndex = 3;
        //
        // txf_estado
        //
        this.txf_estado.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.txf_estado.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.txf_estado.Font = new System.Drawing.Font("Microsoft Sans Serif",
11.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));

        this.txf_estado.ForeColor = System.Drawing.Color.White;
        this.txf_estado.Location = new System.Drawing.Point(124, 234);
        this.txf_estado.Name = "txf_estado";
        this.txf_estado.Size = new System.Drawing.Size(179, 17);
        this.txf_estado.TabIndex = 2;
        //
        // panel7
        //
        this.panel7.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.panel7.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.panel7.Location = new System.Drawing.Point(53, 324);
        this.panel7.Name = "panel7";
        this.panel7.Size = new System.Drawing.Size(280, 1);
        this.panel7.TabIndex = 27;
        //
        // panel5
        //
        this.panel5.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.panel5.Location = new System.Drawing.Point(52, 256);

```



```

this.panel5.Name = "panel5";
this.panel5.Size = new System.Drawing.Size(280, 1);
this.panel5.TabIndex = 26;
//
// panel1
//
        this.panel1.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.panel1.Location = new System.Drawing.Point(52, 192);
        this.panel1.Name = "panel1";
        this.panel1.Size = new System.Drawing.Size(280, 1);
        this.panel1.TabIndex = 25;
//
// labelTitulo
//
        this.labelTitulo.AutoSize = true;
        this.labelTitulo.Font = new System.Drawing.Font("Mongolian Baiti", 36F,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.labelTitulo.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.labelTitulo.Location = new System.Drawing.Point(118, 62);
        this.labelTitulo.Name = "labelTitulo";
        this.labelTitulo.Size = new System.Drawing.Size(192, 50);
        this.labelTitulo.TabIndex = 24;
        this.labelTitulo.Text = "ChatFlor";
//
// pictureBox1
//
        this.pictureBox1.Image = ((System.Drawing.Image)
(resources.GetObject("pictureBox1.Image")));
        this.pictureBox1.Location = new System.Drawing.Point(38, 28);
        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new System.Drawing.Size(100, 84);
        this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
        this.pictureBox1.TabIndex = 30;
        this.pictureBox1.TabStop = false;
//

```

```

// ratioBtn_denuncia
//
this.ratioBtn_denuncia.AutoSize = true;
this.ratioBtn_denuncia.Font = new System.Drawing.Font("Arial", 10.25F);
this.ratioBtn_denuncia.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)
(88))))));
this.ratioBtn_denuncia.Location = new System.Drawing.Point(204, 383);
this.ratioBtn_denuncia.Name = "ratioBtn_denuncia";
this.ratioBtn_denuncia.Size = new System.Drawing.Size(126, 20);
this.ratioBtn_denuncia.TabIndex = 5;
this.ratioBtn_denuncia.TabStop = true;
this.ratioBtn_denuncia.Text = "Fazer Denúncia";
this.ratioBtn_denuncia.UseVisualStyleBackColor = true;
//
// ratioBtn_analista
//
this.ratioBtn_analista.AutoSize = true;
this.ratioBtn_analista.Font = new System.Drawing.Font("Arial", 10.25F);
this.ratioBtn_analista.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
this.ratioBtn_analista.Location = new System.Drawing.Point(53, 383);
this.ratioBtn_analista.Name = "ratioBtn_analista";
this.ratioBtn_analista.Size = new System.Drawing.Size(128, 20);
this.ratioBtn_analista.TabIndex = 4;
this.ratioBtn_analista.TabStop = true;
this.ratioBtn_analista.Text = "Sou um Analista";
this.ratioBtn_analista.UseVisualStyleBackColor = true;
//
// txf_nome
//
this.txf_nome.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45))))));
this.txf_nome.BorderStyle = System.Windows.Forms.BorderStyle.None;
this.txf_nome.Font = new System.Drawing.Font("Microsoft Sans Serif",
11.25F, System.Drawing.FontStyle.Bold);
this.txf_nome.ForeColor = System.Drawing.Color.White;
this.txf_nome.Location = new System.Drawing.Point(125, 170);
this.txf_nome.Name = "txf_nome";

```

```

this.txf_nome.Size = new System.Drawing.Size(189, 17);
this.txf_nome.TabIndex = 1;
//
// label_cidade
//
this.label_cidade.AutoSize = true;
    this.label_cidade.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
    this.label_cidade.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)5))), ((int)(((byte)215))), ((int)(((byte)88))));
    this.label_cidade.Location = new System.Drawing.Point(49, 299);
    this.label_cidade.Name = "label_cidade";
    this.label_cidade.Size = new System.Drawing.Size(71, 22);
    this.label_cidade.TabIndex = 20;
    this.label_cidade.Text = "Cidade";
//
// label_estado
//
this.label_estado.AutoSize = true;
this.label_estado.BackColor = System.Drawing.Color.Transparent;
    this.label_estado.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
    this.label_estado.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)5))), ((int)(((byte)215))), ((int)(((byte)88))));
    this.label_estado.Location = new System.Drawing.Point(48, 231);
    this.label_estado.Name = "label_estado";
    this.label_estado.Size = new System.Drawing.Size(70, 22);
    this.label_estado.TabIndex = 19;
    this.label_estado.Text = "Estado";
//
// label_nome
//
this.label_nome.AutoSize = true;
    this.label_nome.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
    this.label_nome.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)5))), ((int)(((byte)215))), ((int)(((byte)88))));
    this.label_nome.Location = new System.Drawing.Point(48, 167);

```

```

this.label_nome.Name = "label_nome";
this.label_nome.Size = new System.Drawing.Size(61, 22);
this.label_nome.TabIndex = 17;
this.label_nome.Text = "Nome";
//
// btnEntrar
//
this.btnEntrar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
this.btnEntrar.FlatAppearance.BorderSize = 0;
this.btnEntrar.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.btnEntrar.Font = new System.Drawing.Font("Bahnschrift", 14.25F,
System.Drawing.FontStyle.Bold);
this.btnEntrar.ForeColor = System.Drawing.Color.Black;
this.btnEntrar.Location = new System.Drawing.Point(52, 409);
this.btnEntrar.Name = "btnEntrar";
this.btnEntrar.Size = new System.Drawing.Size(281, 40);
this.btnEntrar.TabIndex = 31;
this.btnEntrar.Text = "Entrar";
this.btnEntrar.UseVisualStyleBackColor = false;
this.btnEntrar.Click += new System.EventHandler(this.btnEntrar_Click);
//
// FormLogin
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(19)))),
((int)(((byte)(30)))), ((int)(((byte)(45)))));
this.ClientSize = new System.Drawing.Size(387, 500);
this.Controls.Add(this.btnEntrar);
this.Controls.Add(this.txf_cidade);
this.Controls.Add(this.txf_estado);
this.Controls.Add(this.panel7);
this.Controls.Add(this.panel5);
this.Controls.Add(this.panel1);
this.Controls.Add(this.labelTitulo);
this.Controls.Add(this.pictureBox1);
this.Controls.Add(this.ratioBtn_denuncia);
this.Controls.Add(this.ratioBtn_analista);

```

```

        this.Controls.Add(this.txf_nome);
        this.Controls.Add(this.label_cidade);
        this.Controls.Add(this.label_estado);
        this.Controls.Add(this.label_nome);

        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
        this.MaximizeBox = false;
        this.Name = "FormLogin";
        this.Text = "Login";

        ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();
    }
    private void Phtxbx_GotFocus(object sender, System.EventArgs e)
    {
        throw new System.NotImplementedException();
    }
    #endregion
    private System.Windows.Forms.TextBox txf_cidade;
    private System.Windows.Forms.TextBox txf_estado;
    private System.Windows.Forms.Panel panel7;
    private System.Windows.Forms.Panel panel5;
    private System.Windows.Forms.Panel panel1;
    private System.Windows.Forms.Label labelTitulo;
    private System.Windows.Forms.PictureBox pictureBox1;
    private System.Windows.Forms.RadioButton ratioBtn_denuncia;
    private System.Windows.Forms.RadioButton ratioBtn_analista;
    private System.Windows.Forms.TextBox txf_nome;
    private System.Windows.Forms.Label label_cidade;
    private System.Windows.Forms.Label label_estado;
    private System.Windows.Forms.Label label_nome;
    private prjAps.RJButton btnEntrar;
    }
}
ANEXO CE – Código da Classe FormAnalista.Designer.cs
namespace prjAps
{

```

```

partial class FormAnalista
{
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }
    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        System.Windows.Forms.Panel pnlAtendimentos;
        System.ComponentModel.ComponentResourceManager resources =
new System.ComponentModel.ComponentResourceManager(typeof(FormAnalista));
        this.pnlTxtMensagem = new prjAps.RJPanel();
        this.txtMensagem = new System.Windows.Forms.TextBox();
        this.btnEnviar = new prjAps.RJButton();
        this.atendimentoLog = new System.Windows.Forms.TextBox();
        this.txtRegiao = new System.Windows.Forms.Label();
        this.txtCidade = new System.Windows.Forms.Label();
        this.txtEstado = new System.Windows.Forms.Label();
        this.txtNome = new System.Windows.Forms.Label();
        this.lblIP = new System.Windows.Forms.Label();
        this.lblRegiao = new System.Windows.Forms.Label();
    }
}

```

```

this.lblEstadoAnalista = new System.Windows.Forms.Label();
this.lblCidadeAnalista = new System.Windows.Forms.Label();
this.lblNomeAnalista = new System.Windows.Forms.Label();
this.panel_lblAtendimento = new System.Windows.Forms.Panel();
this.lblAtendimentos = new System.Windows.Forms.Label();
this.pictureBox1 = new System.Windows.Forms.PictureBox();
this.labelTitulo = new System.Windows.Forms.Label();
this.pnlLogo = new System.Windows.Forms.Panel();
this.tblLayoutPnl = new System.Windows.Forms.TableLayoutPanel();
this.pnlNome = new System.Windows.Forms.Panel();
this.pnlIP = new System.Windows.Forms.Panel();
this.txtIP = new System.Windows.Forms.Label();
this.pnlRegiao = new System.Windows.Forms.Panel();
this.pnlCidade = new System.Windows.Forms.Panel();
this.pnlEstado = new System.Windows.Forms.Panel();
pnlAtendimentos = new System.Windows.Forms.Panel();
pnlAtendimentos.SuspendLayout();
this.pnlTxtMensagem.SuspendLayout();
this.panel_lblAtendimento.SuspendLayout();
                                ((System.ComponentModel.ISupportInitialize))
(this.pictureBox1)).BeginInit();
    this.pnlLogo.SuspendLayout();
    this.tblLayoutPnl.SuspendLayout();
    this.pnlNome.SuspendLayout();
    this.pnlIP.SuspendLayout();
    this.pnlRegiao.SuspendLayout();
    this.pnlCidade.SuspendLayout();
    this.pnlEstado.SuspendLayout();
    this.SuspendLayout();
    //
    // pnlAtendimentos
    //
    pnlAtendimentos.Controls.Add(this.pnlTxtMensagem);
    pnlAtendimentos.Controls.Add(this.btnEnviar);
    pnlAtendimentos.Controls.Add(this.atendimentoLog);
    pnlAtendimentos.Location = new System.Drawing.Point(266, 106);
    pnlAtendimentos.Name = "pnlAtendimentos";
    pnlAtendimentos.Size = new System.Drawing.Size(727, 446);

```

```

        pnlAtendimentos.TabIndex = 999;
        //
        // pnlTxtMensagem
        //
        this.pnlTxtMensagem.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)
(88))))));
        this.pnlTxtMensagem.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
        this.pnlTxtMensagem.Controls.Add(this.txtMensagem);
        this.pnlTxtMensagem.ForeColor = System.Drawing.Color.White;
        this.pnlTxtMensagem.Location = new System.Drawing.Point(0, 389);
        this.pnlTxtMensagem.Name = "pnlTxtMensagem";
        this.pnlTxtMensagem.Size = new System.Drawing.Size(570, 35);
        this.pnlTxtMensagem.TabIndex = 11;
        //
        // txtMensagem
        //
        this.txtMensagem.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
        this.txtMensagem.BorderStyle =
System.Windows.Forms.BorderStyle.None;
        this.txtMensagem.Cursor = System.Windows.Forms.Cursors.IBeam;
        this.txtMensagem.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.txtMensagem.ForeColor = System.Drawing.Color.Black;
        this.txtMensagem.Location = new System.Drawing.Point(23, 6);
        this.txtMensagem.Name = "txtMensagem";
        this.txtMensagem.Size = new System.Drawing.Size(524, 22);
        this.txtMensagem.TabIndex = 0;
        this.txtMensagem.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.txtMensagem_KeyPress);
        //
        // btnEnviar
        //
        this.btnEnviar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
        this.btnEnviar.FlatAppearance.BorderSize = 0;
        this.btnEnviar.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

```



```

        this.btnEnviar.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)0));
        this.btnEnviar.ForeColor = System.Drawing.Color.Black;
        this.btnEnviar.Location = new System.Drawing.Point(583, 388);
        this.btnEnviar.Name = "btnEnviar";
        this.btnEnviar.Size = new System.Drawing.Size(141, 36);
        this.btnEnviar.TabIndex = 2;
        this.btnEnviar.Text = "Enviar";
        this.btnEnviar.UseVisualStyleBackColor = false;
        this.btnEnviar.Click += new System.EventHandler(this.btnEnviar_Click);
        //
        // atendimentoLog
        //
        this.atendimentoLog.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.atendimentoLog.BorderStyle =
System.Windows.Forms.BorderStyle.None;
        this.atendimentoLog.Cursor = System.Windows.Forms.Cursors.Default;
        this.atendimentoLog.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
        this.atendimentoLog.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.atendimentoLog.Location = new System.Drawing.Point(24, 0);
        this.atendimentoLog.Multiline = true;
        this.atendimentoLog.Name = "atendimentoLog";
        this.atendimentoLog.ReadOnly = true;
        this.atendimentoLog.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
        this.atendimentoLog.Size = new System.Drawing.Size(721, 371);
        this.atendimentoLog.TabIndex = 99;
        this.atendimentoLog.TabStop = false;
        //
        // txtRegiao
        //
        this.txtRegiao.AutoSize = true;
        this.txtRegiao.BackColor = System.Drawing.Color.Transparent;
        this.txtRegiao.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));

```

```

this.txtRegiao.ForeColor = System.Drawing.Color.White;
this.txtRegiao.Location = new System.Drawing.Point(6, 29);
this.txtRegiao.Name = "txtRegiao";
this.txtRegiao.Size = new System.Drawing.Size(149, 18);
this.txtRegiao.TabIndex = 16;
this.txtRegiao.Text = "São Paulo e Região";
//
// txtCidade
//
this.txtCidade.AutoSize = true;
        this.txtCidade.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
this.txtCidade.ForeColor = System.Drawing.Color.White;
this.txtCidade.Location = new System.Drawing.Point(6, 29);
this.txtCidade.Name = "txtCidade";
this.txtCidade.Size = new System.Drawing.Size(45, 18);
this.txtCidade.TabIndex = 15;
this.txtCidade.Text = "Null...";
//
// txtEstado
//
this.txtEstado.AutoSize = true;
        this.txtEstado.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
this.txtEstado.ForeColor = System.Drawing.Color.White;
this.txtEstado.Location = new System.Drawing.Point(6, 29);
this.txtEstado.Name = "txtEstado";
this.txtEstado.Size = new System.Drawing.Size(45, 18);
this.txtEstado.TabIndex = 14;
this.txtEstado.Text = "Null...";
//
// txtNome
//
this.txtNome.AutoSize = true;
        this.txtNome.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)0));
this.txtNome.ForeColor = System.Drawing.Color.White;
this.txtNome.Location = new System.Drawing.Point(6, 29);
this.txtNome.Name = "txtNome";

```

```

this.txtNome.Size = new System.Drawing.Size(45, 18);
this.txtNome.TabIndex = 123;
this.txtNome.Text = "Null...";
//
// lblIP
//
this.lblIP.AutoSize = true;
this.lblIP.FlatStyle = System.Windows.Forms.FlatStyle.System;
        this.lblIP.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.lblIP.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)(5)))),
((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.lblIP.Location = new System.Drawing.Point(6, 0);
        this.lblIP.Name = "lblIP";
        this.lblIP.Size = new System.Drawing.Size(118, 19);
        this.lblIP.TabIndex = 12;
        this.lblIP.Text = "IP do Analista:";
//
// lblRegiao
//
this.lblRegiao.AutoSize = true;
this.lblRegiao.FlatStyle = System.Windows.Forms.FlatStyle.System;
        this.lblRegiao.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.lblRegiao.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.lblRegiao.Location = new System.Drawing.Point(6, 0);
        this.lblRegiao.Name = "lblRegiao";
        this.lblRegiao.Size = new System.Drawing.Size(191, 19);
        this.lblRegiao.TabIndex = 865;
        this.lblRegiao.Text = "Região de atendimento:";
//
// lblEstadoAnalista
//
this.lblEstadoAnalista.AutoSize = true;
        this.lblEstadoAnalista.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.lblEstadoAnalista.Font = new System.Drawing.Font("Arial", 12F,

```

```

System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.IblEstadoAnalista.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
        this.IblEstadoAnalista.Location = new System.Drawing.Point(6, 0);
        this.IblEstadoAnalista.Name = "IblEstadoAnalista";
        this.IblEstadoAnalista.Size = new System.Drawing.Size(157, 19);
        this.IblEstadoAnalista.TabIndex = 1234;
        this.IblEstadoAnalista.Text = "Estado do Analista:";
        //
        // IblCidadeAnalista
        //
        this.IblCidadeAnalista.AutoSize = true;
                                this.IblCidadeAnalista.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.IblCidadeAnalista.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.IblCidadeAnalista.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
        this.IblCidadeAnalista.Location = new System.Drawing.Point(6, 0);
        this.IblCidadeAnalista.Name = "IblCidadeAnalista";
        this.IblCidadeAnalista.Size = new System.Drawing.Size(157, 19);
        this.IblCidadeAnalista.TabIndex = 86;
        this.IblCidadeAnalista.Text = "Cidade do Analista:";
        //
        // IblNomeAnalista
        //
        this.IblNomeAnalista.AutoSize = true;
                                this.IblNomeAnalista.FlatStyle =
System.Windows.Forms.FlatStyle.System;
        this.IblNomeAnalista.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.IblNomeAnalista.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
        this.IblNomeAnalista.Location = new System.Drawing.Point(6, 0);
        this.IblNomeAnalista.Name = "IblNomeAnalista";
        this.IblNomeAnalista.Size = new System.Drawing.Size(148, 19);
        this.IblNomeAnalista.TabIndex = 123;

```

```

this.IblNomeAnalista.Text = "Nome do Analista:";
//
// panel_IblAtendimento
//
this.panel_IblAtendimento.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)
(45)))));
this.panel_IblAtendimento.Controls.Add(this.IblAtendimentos);
this.panel_IblAtendimento.Location = new System.Drawing.Point(266,
12);
this.panel_IblAtendimento.Name = "panel_IblAtendimento";
this.panel_IblAtendimento.Size = new System.Drawing.Size(727, 85);
this.panel_IblAtendimento.TabIndex = 122;
//
// IblAtendimentos
//
this.IblAtendimentos.AutoSize = true;
this.IblAtendimentos.FlatStyle =
System.Windows.Forms.FlatStyle.System;
this.IblAtendimentos.Font = new System.Drawing.Font("Mongolian Baiti",
24F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));
this.IblAtendimentos.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
this.IblAtendimentos.Location = new System.Drawing.Point(184, 26);
this.IblAtendimentos.Name = "IblAtendimentos";
this.IblAtendimentos.Size = new System.Drawing.Size(364, 34);
this.IblAtendimentos.TabIndex = 17;
this.IblAtendimentos.Text = "Atendimentos Realizados";
//
// pictureBox1
//
this.pictureBox1.Image = ((System.Drawing.Image)
(resources.GetObject("pictureBox1.Image")));
this.pictureBox1.Location = new System.Drawing.Point(-12, 0);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(80, 64);
this.pictureBox1.SizeMode =

```

```

System.Windows.Forms.PictureBoxSizeMode.StretchImage;
    this.pictureBox1.TabIndex = 31;
    this.pictureBox1.TabStop = false;
    this.pictureBox1.Tag = "567";
    //
    // labelTitulo
    //
    this.labelTitulo.AutoSize = true;
    this.labelTitulo.BackColor = System.Drawing.Color.Transparent;
        this.labelTitulo.Font = new System.Drawing.Font("Mongolian Baiti",
27.75F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)
(0)));
        this.labelTitulo.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
    this.labelTitulo.Location = new System.Drawing.Point(59, 24);
    this.labelTitulo.Name = "labelTitulo";
    this.labelTitulo.Size = new System.Drawing.Size(149, 40);
    this.labelTitulo.TabIndex = 32;
    this.labelTitulo.Text = "ChatFlor";
    //
    // pnlLogo
    //
        this.pnlLogo.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
    this.pnlLogo.Controls.Add(this.labelTitulo);
    this.pnlLogo.Controls.Add(this.pictureBox1);
    this.pnlLogo.Location = new System.Drawing.Point(2, 9);
    this.pnlLogo.Name = "pnlLogo";
    this.pnlLogo.Size = new System.Drawing.Size(258, 88);
    this.pnlLogo.TabIndex = 21;
    //
    // tblLayoutPnl
    //
        this.tblLayoutPnl.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.tblLayoutPnl.ColumnCount = 1;
                                this.tblLayoutPnl.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
100F));

```

```

this.tblLayoutPnl.Controls.Add(this.pnlNome, 0, 0);
this.tblLayoutPnl.Controls.Add(this.pnlIP, 0, 4);
this.tblLayoutPnl.Controls.Add(this.pnlRegiao, 0, 3);
this.tblLayoutPnl.Controls.Add(this.pnlCidade, 0, 2);
this.tblLayoutPnl.Controls.Add(this.pnlEstado, 0, 1);
this.tblLayoutPnl.Location = new System.Drawing.Point(2, 106);
this.tblLayoutPnl.Name = "tblLayoutPnl";
this.tblLayoutPnl.RowCount = 6;

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.67032F));

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.66594F));

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.66594F));

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.66594F));

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.66594F));

this.tblLayoutPnl.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent,
16.66594F));

this.tblLayoutPnl.Size = new System.Drawing.Size(255, 446);
this.tblLayoutPnl.TabIndex = 75;
//
// pnlNome
//
this.pnlNome.Controls.Add(this.txtNome);
this.pnlNome.Controls.Add(this.lblNomeAnalista);
this.pnlNome.Dock = System.Windows.Forms.DockStyle.Fill;
this.pnlNome.Location = new System.Drawing.Point(3, 3);
this.pnlNome.Name = "pnlNome";
this.pnlNome.Size = new System.Drawing.Size(249, 68);
this.pnlNome.TabIndex = 11;
//

```

```

// pnlIP
//
        this.pnlIP.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.pnlIP.Controls.Add(this.lblIP);
        this.pnlIP.Controls.Add(this.txtIP);
        this.pnlIP.Dock = System.Windows.Forms.DockStyle.Fill;
        this.pnlIP.Location = new System.Drawing.Point(3, 299);
        this.pnlIP.Name = "pnlIP";
        this.pnlIP.Size = new System.Drawing.Size(249, 68);
        this.pnlIP.TabIndex = 12;
//
// txtIP
//
        this.txtIP.AutoSize = true;
                this.txtIP.Font = new System.Drawing.Font("Arial", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.txtIP.ForeColor = System.Drawing.Color.White;
        this.txtIP.Location = new System.Drawing.Point(6, 29);
        this.txtIP.Name = "txtIP";
        this.txtIP.Size = new System.Drawing.Size(45, 18);
        this.txtIP.TabIndex = 19;
        this.txtIP.Text = "Null...";
//
// pnlRegiao
//
        this.pnlRegiao.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.pnlRegiao.Controls.Add(this.lblRegiao);
        this.pnlRegiao.Controls.Add(this.txtRegiao);
        this.pnlRegiao.Dock = System.Windows.Forms.DockStyle.Fill;
        this.pnlRegiao.Location = new System.Drawing.Point(3, 225);
        this.pnlRegiao.Name = "pnlRegiao";
        this.pnlRegiao.Size = new System.Drawing.Size(249, 68);
        this.pnlRegiao.TabIndex = 12;
//
// pnlCidade
//

```



```

        this.pnlCidade.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.pnlCidade.Controls.Add(this.lblCidadeAnalista);
        this.pnlCidade.Controls.Add(this.txtCidade);
        this.pnlCidade.Dock = System.Windows.Forms.DockStyle.Fill;
        this.pnlCidade.Location = new System.Drawing.Point(3, 151);
        this.pnlCidade.Name = "pnlCidade";
        this.pnlCidade.Size = new System.Drawing.Size(249, 68);
        this.pnlCidade.TabIndex = 46;
        //
        // pnlEstado
        //
        this.pnlEstado.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.pnlEstado.BackgroundImageLayout =
System.Windows.Forms.ImageLayout.Center;
        this.pnlEstado.Controls.Add(this.lblEstadoAnalista);
        this.pnlEstado.Controls.Add(this.txtEstado);
        this.pnlEstado.Dock = System.Windows.Forms.DockStyle.Fill;
        this.pnlEstado.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.pnlEstado.Location = new System.Drawing.Point(3, 77);
        this.pnlEstado.Name = "pnlEstado";
        this.pnlEstado.Size = new System.Drawing.Size(249, 68);
        this.pnlEstado.TabIndex = 11;
        //
        // FormAnalista
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(19)))),
((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.ClientSize = new System.Drawing.Size(1005, 542);
        this.Controls.Add(this.tblLayoutPnl);
        this.Controls.Add(this.pnlLogo);
        this.Controls.Add(pnlAtendimentos);
        this.Controls.Add(this.panel_lblAtendimento);
        this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;

```

```

this.MaximizeBox = false;
this.Name = "FormAnalista";
this.Text = "Denúncias Florestais";

this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(this.FormAnalista_FormClosed);
    pnlAtendimentos.ResumeLayout(false);
    pnlAtendimentos.PerformLayout();
    this.pnlTxtMensagem.ResumeLayout(false);
    this.pnlTxtMensagem.PerformLayout();
    this.panel_IblAtendimento.ResumeLayout(false);
    this.panel_IblAtendimento.PerformLayout();
    ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
    this.pnlLogo.ResumeLayout(false);
    this.pnlLogo.PerformLayout();
    this.tblLayoutPnl.ResumeLayout(false);
    this.pnlNome.ResumeLayout(false);
    this.pnlNome.PerformLayout();
    this.pnlIP.ResumeLayout(false);
    this.pnlIP.PerformLayout();
    this.pnlRegiao.ResumeLayout(false);
    this.pnlRegiao.PerformLayout();
    this.pnlCidade.ResumeLayout(false);
    this.pnlCidade.PerformLayout();
    this.pnlEstado.ResumeLayout(false);
    this.pnlEstado.PerformLayout();
    this.ResumeLayout(false);
}
#endregion
private System.Windows.Forms.Label lblRegiao;
private System.Windows.Forms.Label lblEstadoAnalista;
private System.Windows.Forms.Label lblCidadeAnalista;
private System.Windows.Forms.Label lblNomeAnalista;
private System.Windows.Forms.Panel panel_IblAtendimento;
private System.Windows.Forms.Label lblIP;
private System.Windows.Forms.Label lblAtendimentos;
private System.Windows.Forms.Label txtNome;
private System.Windows.Forms.Label txtRegiao;

```

```

private System.Windows.Forms.Label txtCidade;
private System.Windows.Forms.Label txtEstado;
private System.Windows.Forms.TextBox atendimentoLog;
private System.Windows.Forms.TextBox txtMensagem;
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.Label labelTitulo;
private System.Windows.Forms.Panel pnlLogo;
private System.Windows.Forms.TableLayoutPanel tblLayoutPnl;
private System.Windows.Forms.Panel pnlIP;
private System.Windows.Forms.Panel pnlRegiao;
private System.Windows.Forms.Panel pnlCidade;
private System.Windows.Forms.Panel pnlEstado;
private System.Windows.Forms.Panel pnlNome;
private RJButton btnEnviar;
private RJPanel pnlTxtMensagem;
private System.Windows.Forms.Label txtIP;
}
}

```

ANEXO CF – Código da Classe FormChat.Designer.cs

```

namespace prjAps
{
    partial class FormChat
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;
        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}

```

```

}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.pnlDenunciante = new System.Windows.Forms.Panel();
    this.lblDenunciante = new System.Windows.Forms.Label();
    this.pnlTxtLog = new System.Windows.Forms.Panel();
    this.txtLog = new System.Windows.Forms.TextBox();
    this.panel3 = new System.Windows.Forms.Panel();
    this.pnlTxtMensagem = new prjAps.RJPanel();
    this.txtMensagem = new System.Windows.Forms.TextBox();
    this.btnEnviar = new prjAps.RJButton();
    this.pnlDenunciante.SuspendLayout();
    this.pnlTxtLog.SuspendLayout();
    this.panel3.SuspendLayout();
    this.pnlTxtMensagem.SuspendLayout();
    this.SuspendLayout();
    //
    // pnlDenunciante
    //
    this.pnlDenunciante.Controls.Add(this.lblDenunciante);
    this.pnlDenunciante.Location = new System.Drawing.Point(13, 13);
    this.pnlDenunciante.Name = "pnlDenunciante";
    this.pnlDenunciante.Size = new System.Drawing.Size(488, 45);
    this.pnlDenunciante.TabIndex = 0;
    //
    // lblDenunciante
    //
    this.lblDenunciante.AutoSize = true;
    this.lblDenunciante.Font = new System.Drawing.Font("Mongolian Baiti",
24F, System.Drawing.FontStyle.Bold);
    this.lblDenunciante.ForeColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88))))));
    this.lblDenunciante.Location = new System.Drawing.Point(166, 5);

```

```

this.lblDenunciante.Name = "lblDenunciante";
this.lblDenunciante.Size = new System.Drawing.Size(187, 34);
this.lblDenunciante.TabIndex = 88;
this.lblDenunciante.Text = "Denunciante";
//
// pnlTxtLog
//
this.pnlTxtLog.Controls.Add(this.txtLog);
this.pnlTxtLog.Location = new System.Drawing.Point(13, 65);
this.pnlTxtLog.Name = "pnlTxtLog";
this.pnlTxtLog.Size = new System.Drawing.Size(488, 531);
this.pnlTxtLog.TabIndex = 887;
//
// txtLog
//
        this.txtLog.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(19)))), ((int)(((byte)(30)))), ((int)(((byte)(45)))));
        this.txtLog.BorderStyle = System.Windows.Forms.BorderStyle.None;
        this.txtLog.Cursor = System.Windows.Forms.Cursors.Default;
        this.txtLog.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.txtLog.ForeColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
        this.txtLog.Location = new System.Drawing.Point(-1, 3);
        this.txtLog.Multiline = true;
        this.txtLog.Name = "txtLog";
        this.txtLog.ReadOnly = true;
        this.txtLog.ScrollBars = System.Windows.Forms.ScrollBars.Both;
        this.txtLog.Size = new System.Drawing.Size(506, 526);
        this.txtLog.TabIndex = 9;
//
// panel3
//
this.panel3.Controls.Add(this.pnlTxtMensagem);
this.panel3.Controls.Add(this.btnEnviar);
this.panel3.Location = new System.Drawing.Point(13, 602);
this.panel3.Name = "panel3";
this.panel3.Size = new System.Drawing.Size(488, 82);
this.panel3.TabIndex = 2;

```

```

//
// pnlTxtMensagem
//
this.pnlTxtMensagem.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)
(88)))));
this.pnlTxtMensagem.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.pnlTxtMensagem.Controls.Add(this.txtMensagem);
this.pnlTxtMensagem.ForeColor = System.Drawing.Color.White;
this.pnlTxtMensagem.Location = new System.Drawing.Point(0, 0);
this.pnlTxtMensagem.Name = "pnlTxtMensagem";
this.pnlTxtMensagem.Size = new System.Drawing.Size(365, 35);
this.pnlTxtMensagem.TabIndex = 12;
//
// txtMensagem
//
this.txtMensagem.BackColor = System.Drawing.Color.FromArgb(((int)
(((byte)(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
this.txtMensagem.BorderStyle =
System.Windows.Forms.BorderStyle.None;
this.txtMensagem.Font = new System.Drawing.Font("Arial", 14.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.txtMensagem.ForeColor = System.Drawing.Color.Black;
this.txtMensagem.Location = new System.Drawing.Point(15, 6);
this.txtMensagem.Name = "txtMensagem";
this.txtMensagem.Size = new System.Drawing.Size(337, 22);
this.txtMensagem.TabIndex = 1;
this.txtMensagem.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.txtMensagem_KeyPress);
//
// btnEnviar
//
this.btnEnviar.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)
(5)))), ((int)(((byte)(215)))), ((int)(((byte)(88)))));
this.btnEnviar.FlatAppearance.BorderSize = 0;
this.btnEnviar.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.btnEnviar.Font = new System.Drawing.Font("Arial", 14.25F,

```

```

System.Drawing.FontStyle.Bold);
    this.btnEnviar.ForeColor = System.Drawing.Color.Black;
    this.btnEnviar.Location = new System.Drawing.Point(374, 0);
    this.btnEnviar.Name = "btnEnviar";
    this.btnEnviar.Size = new System.Drawing.Size(114, 35);
    this.btnEnviar.TabIndex = 2;
    this.btnEnviar.Text = "Enviar";
    this.btnEnviar.UseVisualStyleBackColor = false;
    this.btnEnviar.Click += new System.EventHandler(this.btnEnviar_Click);
    //
    // FormChat
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(19)))),
((int)(((byte)(30)))), ((int)(((byte)(45)))));
    this.ClientSize = new System.Drawing.Size(510, 646);
    this.Controls.Add(this.panel3);
    this.Controls.Add(this.pnlTxtLog);
    this.Controls.Add(this.pnlDenunciante);

    this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
    this.MaximizeBox = false;
    this.Name = "FormChat";
    this.Text = "Chat";

    this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(this.FormChat_FormClosed);
    this.pnlDenunciante.ResumeLayout(false);
    this.pnlDenunciante.PerformLayout();
    this.pnlTxtLog.ResumeLayout(false);
    this.pnlTxtLog.PerformLayout();
    this.panel3.ResumeLayout(false);
    this.pnlTxtMensagem.ResumeLayout(false);
    this.pnlTxtMensagem.PerformLayout();
    this.ResumeLayout(false);
}
#endregion
private System.Windows.Forms.Panel pnlDenunciante;
private System.Windows.Forms.Label lblDenunciante;

```

```

private System.Windows.Forms.Panel pnlTxtLog;
private System.Windows.Forms.Panel panel3;
private System.Windows.Forms.TextBox txtMensagem;
private System.Windows.Forms.TextBox txtLog;
private RJButton btnEnviar;
private RJPanel pnlTxtMensagem;
}
}

```

ANEXO CG – Código da Classe FormLogin.cs

```

using System;
using System.Windows.Forms;
namespace prjAps
{
    public partial class FormLogin : Form
    {
        private string name, state, city;
        private bool analista, denunciante;
        public FormLogin()
        {
            InitializeComponent();
        }
        private void btnEntrar_Click(object sender, EventArgs e)
        {
            analista = radioBtn_analista.Checked;
            denunciante = radioBtn_denuncia.Checked;
            name = txf_nome.Text;
            state = txf_estado.Text;
            city = txf_cidade.Text;
            if (name != ""
                && (analista == true || denunciante == true)
                && state != ""
                && city != "")
            {
                if (analista == true)
                {
                    Cursor = Cursors.WaitCursor;
                    FormAnalista frmAnalista = new FormAnalista(name, state, city);
                    Cursor = Cursors.Default;
                }
            }
        }
    }
}

```



```

        frmAnalista.ShowDialog();
        if (frmAnalista.IsDisposed)
        {
            this.Close();
        }
    }
    else
    {
        Cursor = Cursors.WaitCursor;
        FormChat frmChat = new FormChat(name);
        Cursor = Cursors.Default;
        frmChat.ShowDialog();
        if (frmChat.IsDisposed)
        {
            this.Close();
        }
    }
}
else
{
    MessageBox.Show("Preencha Todos os Campos!");
}
}
private void RemoveText(TextBox txt, object sender, EventArgs e)
{
    if (txf_nome.Text == "Digite seu Nome...")
    {
        txf_nome.Text = "";
    }
}
public void AddText(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txf_nome.Text))
        txf_nome.Text = "Enter text here...";
}
}
}

```

ANEXO CH – Código da Classe FormAnalista.cs
using System;

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
namespace prjAps
{
    public partial class FormAnalista : Form
    {
        #region Variaveis
        private string name;
        private StreamWriter StwEnvia;
        private StreamReader StrRecebe;
        private TcpClient TcpServidor;
        private IPAddress EnderecoIP;
        private int PortaHost;
        //atualiza formulario com a mensagem de outra thread
        private delegate void AtualizaLogCallBack(string strMensagem);
        //Define o formulario para o estado "disconnected" de outra thread
        private delegate void FechaConexaoCallBack(string strMotivo);
        private bool Conectado;
        private Thread MsgThread;
        #endregion
        #region Construtor
        public FormAnalista(string name, string state, string city)
        {
            //atualiza variaveis locais
            this.name = name;
            Application.ApplicationExit += new EventHandler(OnApplicationExit);
            InitializeComponent();
            //carrega dados no form

```

```

txtNome.Text = name;
txtEstado.Text = state;
txtCidade.Text = city;
if (!Conectado)
{
    InicializaConexao();
}
else
{
    FechaConexao("Desconectado a pedido do analista");
}
}
#endregion
#region Conexao com o Servidor / Lógica
// Inicia conexão com o servidor
private void InicializaConexao()
{
    try
    {
        //hard coded
        EnderecoIP = IPAddress.Parse("127.0.0.1");
        PortaHost = 1000;
        //inicia uma nova conexão tcp com o chat server
        TcpServidor = new TcpClient();
        TcpServidor.Connect(EnderecoIP, PortaHost);
        Conectado = true;
        //inicializa componentes
        Point ptTxtOriginal = new Point(23, 6);
        Point ptPnlOriginal = new Point(0, 389);
        txtMensagem.Location = ptTxtOriginal;
        pnlTxtMensagem.Location = ptPnlOriginal;
        txtMensagem.Enabled = true;
        btnEnviar.Enabled = true;
        btnEnviar.ForeColor = Color.Black;
        btnEnviar.Text = "Enviar";
        atendimentoLog.ForeColor = Color.FromArgb(5, 215, 88);
        //IP e ID
        IPAddress[] ip = Dns.GetHostAddresses(Dns.GetHostName());
        txtIP.Text = ip[1].ToString();
    }
    catch { }
}

```

```

        //envia o nome do user para o servidor
        StwEnvia = new StreamWriter(TcpServidor.GetStream());
        StwEnvia.WriteLine(name);
        StwEnvia.Flush();
        // Inicializa a thread para receber mensagens e nova comunicação
        MsgThread = new Thread(new ThreadStart(RecebeMensagem));
        MsgThread.IsBackground = true;
        MsgThread.Start();
    }
    catch
    {
        FechaConexao("Não foi possível estabelecer conexão com o servidor!
        \n\nClique no botão Reconectar... Para tentar novamente.\n\n");
    }
}
// Recebe mensagem do servidor
private void RecebeMensagem()
{
    // Recebe a resposta do servidor
    StrRecebe = new StreamReader(TcpServidor.GetStream());
    string ConResposta = StrRecebe.ReadLine();
    // Se o primeiro caractere da resposta é 1 a conexão foi feita com
    sucesso
    if (ConResposta[0] == '1' && Conectado == true)
    {
        // Atualiza o formulário (do servidor) para informar que está conectado
        try
        {
            // AtualizaLog o formulario com "Conectado com sucesso!"
            this.Invoke(new AtualizaLogCallBack(this.AtualizaLog), new object[]
            { "Conectado com sucesso!" });
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error Message: \n" + ex);
        }
    }
}
else

```

```

{
    // Se o primeiro caractere não for 1 a conexão falhou
    string Motivo = "Não Conectado: ";
    // Extrai o motivo da mensagem resposta. O motivo começa no 3o.
    caractere
    Motivo += ConResposta.Substring(2, ConResposta.Length - 2);
    try
    {
        this.Invoke(new FechaConexaoCallBack(this.FechaConexao), new
object[] { Motivo });
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error Message: \n" + ex);
    }
    // Sai do método
    return;
}

// Enquanto estiver conectado lê as linhas que estão chegando do
servidor
while (Conectado)
{
    // Exibe mensagens no TextBox
    try
    {
        this.Invoke(new AtualizaLogCallBack(this.AtualizaLog), new object[]
{ StrRecebe.ReadLine() });
    }
    catch (Exception ex)
    {
        // Atualiza o formulário com o motivo da falha da conexão
        this.Invoke(new FechaConexaoCallBack(this.FechaConexao), new
object[] { "Não foi possível estabelecer conexão com o servidor! \r\nClique no botão
Reconectar... Para tentar novamente.\r\n" });
        Console.WriteLine("Error Message: \n" + ex);
    }
}
}

// Fecha a conexão com o servidor

```

```

private void FechaConexao(string Motivo = null)
{
    //se não tiver motivo significa que apenas quer fechar a aplicação
    if (Motivo != null)
    {
        atendimentoLog.ForeColor = Color.Yellow;
        atendimentoLog.AppendText($"{Motivo} \r\n");
    }
    //fecha os objetos
    Conectado = false;
    TcpServidor.Close();
    //desabilita e habilita os campos apropriados
    btnEnviar.ForeColor = Color.Black;
    btnEnviar.Text = "Reconectar";
    Point ptTxt = new Point(1, 492);
    pnlTxtMensagem.Location = ptTxt;
    txtMensagem.Location = ptTxt;
    txtMensagem.Enabled = false;
    //n foi iniciado
    if (StwEnvia != null)
    {
        StwEnvia.Close();
        StrRecebe.Close();
    }
}
// Envia mensagem ao servidor
private void EnviaMensagem()
{
    if(Conectado)
    {
        //envia mensagem pro servidor
        if (txtMensagem.Lines.Length >= 1)
        {
            StwEnvia.WriteLine(txtMensagem.Text);
            StwEnvia.Flush();
            txtMensagem.Lines = null;
        }
        txtMensagem.Text = "";
    }
}

```

```

    } else
    {
        FechaConexao();
    }
}
// Atualiza a mensagem no atendimentoLog
private void AtualizaLog(string strMensagem)
{
    atendimentoLog.AppendText($"{strMensagem} \r\n");
}
#endregion
#region Eventos
private void btnEnviar_Click(object sender, EventArgs e)
{
    if (btnEnviar.Text == "Reconectar")
    {
        this.Cursor = Cursors.WaitCursor;
        InicializaConexao();
        this.Cursor = Cursors.Default;
    }
    else
    {
        EnviaMensagem();
    }
}
public void OnApplicationExit(object sender, EventArgs e)
{
    //testar quando estiver conectado!
    if (Conectado)
    {
        FechaConexao();
    }
}
private void FormAnalista_Closing(object sender, CancelEventArgs e)
{
    //testar quando estiver conectado!
    if (Conectado)
    {
        FechaConexao();
    }
}

```

```

    }
}

private void FormAnalista_FormClosed(object sender,
FormClosedEventArgs e)
{
    Application.Exit();
}
private void txtMensagem_KeyPress(object sender, KeyPressEventArgs e)
{
    // Se precionou a tecla enter
    if (e.KeyChar == (char)13)
    {
        EnviaMensagem();
    }
}
}
#endregion
}
}

```

ANEXO CI – Código da Classe FormChat.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Drawing.Drawing2D;
namespace prjAps
{
    public partial class FormChat : Form
    {
        #region Variaveis
        private string name;

```



```

private StreamWriter StwEnvia;
private StreamReader StrRecebe;
private TcpClient TcpServidor;
//definindo o ip e porta global
private IPAddress EnderecoIP;
private int PortaHost;
//atualiza formulario com a mensagem de outra thread
private delegate void AtualizaLogCallBack(string strMensagem);
//Define o formulario para o estado "disconnected" de outra thread
private delegate void FechaConexaoCallBack(string strMotivo);
private bool Conectado;
private Thread MsgThread;
#endregion
#region Construtor
//passo o nome do usuario por aqui
public FormChat(string name)
{
    this.name = name;
    Application.ApplicationExit += new EventHandler(OnApplicationExit);
    InitializeComponent();
    if (!Conectado)
    {
        InicializaConexao();
    }
    else
    {
        FechaConexao("Desconectado a pedido do usuário");
    }
}
#endregion
#region Conexao com o Servidor / Lógica
// Inicia conexão com o servidor
private void InicializaConexao()
{
    try
    {
        //hard coded
        EnderecoIP = IPAddress.Parse("127.0.0.1");
    }
}

```

```

PortaHost = 1000;
//inicia uma nova conexão tcp com o chat server
TcpServidor = new TcpClient();
TcpServidor.Connect(EnderecoIP, PortaHost);
Conectado = true;
//inicializa componentes
Point ptPnlOriginal = new Point(15, 6);
Point ptTxtOriginal = new Point(0, 0);
Point ptBtn = new Point(374, 0);
Size szBtn = new Size(114, 35);
btnEnviar.Location = ptBtn;
btnEnviar.Size = szBtn;
txtMensagem.Location = ptPnlOriginal;
pnlTxtMensagem.Location = ptTxtOriginal;
txtMensagem.Enabled = true;
btnEnviar.Enabled = true;
btnEnviar.ForeColor = Color.Black;
btnEnviar.Text = "Enviar";
txtLog.ForeColor = Color.FromArgb(5, 215, 88);
//envia o nome do user para o servidor
StwEnvia = new StreamWriter(TcpServidor.GetStream());
StwEnvia.WriteLine(name);
StwEnvia.Flush();
MsgThread = new Thread(new ThreadStart(RecebeMensagem));
MsgThread.IsBackground = true;
MsgThread.Start();
}
catch
{
    FechaConexao("Não foi possível estabelecer conexão com o servidor!
\r\nClique no botão Reconectar... Para tentar novamente.\r\n");
}
}
// Recebe mensagem do servidor
private void RecebeMensagem()
{
    // Recebe a resposta do servidor
    StrRecebe = new StreamReader(TcpServidor.GetStream());
    string ConResposta = StrRecebe.ReadLine();

```

```

        // Se o primeiro caractere da resposta é 1 a conexão foi feita com
sucesso
        if (ConResposta[0] == '1' && Conectado == true)
        {
            //atualiza o formulário (do servidor) para informar que está conectado
            try
            {
                this.Invoke(new AtualizaLogCallBack(this.AtualizaLog), new object[]
{ "Conectado com sucesso!" });
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error Message: \n" + ex);
            }
            //this.Invoke(new AtualizaLogCallBack(this.AtualizaLog), new object[] {
"Conectado com sucesso!" });
        }
        else
        {
            // Se o primeiro caractere não for 1 a conexão falhou
            string Motivo = "Não Conectado: ";
            //extraí o motivo da mensagem resposta o motivo começa no 3char
            Motivo += ConResposta.Substring(2, ConResposta.Length - 2);
            // Atualiza o formulário com o motivo da falha da conexão
            try
            {
                this.Invoke(new FechaConexaoCallBack(this.FechaConexao), new
object[] { Motivo });
            }
            catch (Exception ex)
            {
                this.Invoke(new FechaConexaoCallBack(this.FechaConexao), new
object[] { Motivo });
                Console.WriteLine("Error Message: \n" + ex);
            }
            // Sai do metodo
            return;
        }
    }

```

```

servidor
    // Enquanto estiver conectado lê as linhas que estão chegando do
    while (Conectado)
    {
        //exibe mensagem no txtLog
        try
        {
            this.Invoke(new AtualizaLogCallBack(this.AtualizaLog), new object[]
            { StrRecebe.ReadLine() });
        }
        catch (Exception ex)
        {
            // Atualiza o formulário com o motivo da falha da conexão
            this.Invoke(new FechaConexaoCallBack(this.FechaConexao), new
            object[] { "Não foi possível estabelecer conexão com o servidor! \r\nClique no botão
            Reconectar... Para tentar novamente.\r\n" });
            Console.WriteLine("Error Message: \n" + ex);
        }
    }
}
// Fecha a conexão com o servidor
private void FechaConexao(string Motivo = null)
{
    // se não tiver motivo significa que apenas quer fechar a aplicação
    if (Motivo != null)
    {
        txtLog.ForeColor = Color.Yellow;
        txtLog.AppendText($"{Motivo} \r\n");
    }
    //fecha os objetos
    Conectado = false;
    TcpServidor.Close();
    //desabilita e habilita os campos apropriados
    Point ptTxt = new Point(0, 64);
    Point ptBtn = new Point(192, 0);
    Size szBtn = new Size(129, 35);
    txtMensagem.Location = ptTxt;
    pnlTxtMensagem.Location = ptTxt;
    btnEnviar.ForeColor = Color.Black;

```

```

        btnEnviar.Text = "Reconectar";
        btnEnviar.Location = ptBtn;
        btnEnviar.Size = szBtn;
        txtMensagem.Enabled = false;
        //n foi iniciado
        if (StwEnvia != null)
        {
            StwEnvia.Close();
            StrRecebe.Close();
        }
    }
    // Envia mensagem ao servidor
    private void EnviaMensagem()
    {
        //envia mensagem pro servidor
        if (txtMensagem.Lines.Length >= 1)
        {
            StwEnvia.WriteLine(txtMensagem.Text);
            StwEnvia.Flush();
            txtMensagem.Lines = null;
        }
        txtMensagem.Text = "";
    }
    // Atualiza a mensagem no atendimentoLog
    private void AtualizaLog(string strMensagem)
    {
        txtLog.AppendText($"{strMensagem} \r\n");
    }
    #endregion
    #region Eventos
    public void OnApplicationExit(object sender, EventArgs e)
    {
        if (Conectado)
        {
            FechaConexao();
        }
    }
    private void FormChat_Closing(object sender, CancelEventArgs e)

```

```

{
    //testar quando estiver conectado!
    if (Conectado)
    {
        FechaConexao();
    }
}
private void FormChat_FormClosed(object sender, FormClosedEventArgs
e)
{
    Application.Exit();
}
private void btnEnviar_Click(object sender, EventArgs e)
{
    if (btnEnviar.Text == "Reconectar")
    {
        this.Cursor = Cursors.WaitCursor;
        InicializaConexao();
        this.Cursor = Cursors.Default;
    }
    else
    {
        EnviaMensagem();
    }
}
private void txtMensagem_KeyPress(object sender, KeyPressEventArgs e)
{
    // Se precionou a tecla enter
    if (e.KeyChar == (char)13)
    {
        EnviaMensagem();
    }
}
#endregion
}
}

```



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Matheus Vinicius Brasil Moraes

TURMA: CCAP17

RA: F339475

CURSO: Ciência da Computação

CAMPUS: Sorocaba(17)

SEMESTRE: 4º Semestre TURNO: Noturno

CODIGO DA ATIVIDADE: 77B2

SEMESTRE: 4º Semestre

ANO/GRAD: 2022

DATA DA ATIVIDADE	DESCRICAO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUIDAS (1)	ASSINATURA DO PROFESSOR
07/04/2022	Formação do grupo e divisão de Tarefas	4h	MATHEUS VINICIUS BRASIL MORAES		
08/04/2022	Reunião de Brainstorm do projeto	6h	MATHEUS VINICIUS BRASIL MORAES		
09/04/2022	Pesquisas de Fundamentação da ABNT	5h	MATHEUS VINICIUS BRASIL MORAES		
10/04/2022	Pesquisas sobre os Temas propostos	5h	MATHEUS VINICIUS BRASIL MORAES		
11/04/2022	Leitura do Livro "Use a cabeça C#"	5h	MATHEUS VINICIUS BRASIL MORAES		
12/04/2022	Contribuição com Pesquisas Teóricas da ABNT	6h	MATHEUS VINICIUS BRASIL MORAES		
13/04/2022	Iniciando o desenvolvimento da ABNT	3h	MATHEUS VINICIUS BRASIL MORAES		
16/04/2022	Auxiliando nos testes do código do projeto	6h	MATHEUS VINICIUS BRASIL MORAES		
09/05/2022	Leitura do Livro "Algoritmos Teoria e Prática"	3h	MATHEUS VINICIUS BRASIL MORAES		
10/05/2022	Contribuição para o Resumo da ABNT	6h	MATHEUS VINICIUS BRASIL MORAES		
12/05/2022	Contribuição para o Referencial Teórico ABNT	4h	MATHEUS VINICIUS BRASIL MORAES		
13/05/2022	Contribuição para o Desenvolvimento da ABNT	5h	MATHEUS VINICIUS BRASIL MORAES		
15/05/2022	Revisando pesquisas de integrantes e formatação ABNT	6h	MATHEUS VINICIUS BRASIL MORAES		
17/05/2022	Auxílio na finalização do código do projeto	6h	MATHEUS VINICIUS BRASIL MORAES		
18/05/2022	Contribuição com Referências Bibliográficas	6h	MATHEUS VINICIUS BRASIL MORAES		
19/05/2022	Desenvolvimento da Conclusão da ABNT	4h	MATHEUS VINICIUS BRASIL MORAES		

(1) Horas atribuídas de acordo CDM o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUIDAS: 80 Hrs

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Henrique Thomas Correa Alonso da Silva TURMA: CCSP17 RA: N574HD-0
CURSO: Ciência da Computação CAMPUS: Sorocaba (17) SEMESTRE: 2º Semestre TURNO: Nocturno
CÓDIGO DA ATIVIDADE: 7782 SEMESTRE: 5º Semestre ANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
07/04/2022	Formação do grupo e divisão de tarefas	4h	Henrique Thomas Correa Alonso da Silva		
08/04/2022	Reunião de brainstorm do projeto	2h	Henrique Thomas Correa Alonso da Silva		
11/04/2022	Pesquisas sobre sockets Berkeley	6h	Henrique Thomas Correa Alonso da Silva		
13/04/2022	Estudo da linguagem C#	10h	Henrique Thomas Correa Alonso da Silva		
14/04/2022	Leitura do livro Use a Cabeça C# de Andrew Stellman E Jennifer Greene	3h	Henrique Thomas Correa Alonso da Silva		
15/04/2022	Desenvolvimento do Front-end do projeto	5h	Henrique Thomas Correa Alonso da Silva		
17/04/2022	Testes do código front-end do projeto	3h	Henrique Thomas Correa Alonso da Silva		
20/04/2022	Desenvolvimento do Back-end do projeto	15h	Henrique Thomas Correa Alonso da Silva		
25/04/2022	Testes do código back-end do projeto	3h	Henrique Thomas Correa Alonso da Silva		
10/05/2022	Contribuição com Pesquisa Teórica do Software	3h	Henrique Thomas Correa Alonso da Silva		
11/05/2022	Contribuição com Pesquisa Teórica da ABNT	6h	Henrique Thomas Correa Alonso da Silva		
12/05/2022	Contribuição de referências bibliográficas da ABNT	2h	Henrique Thomas Correa Alonso da Silva		
13/05/2022	Testes e ajustes da Interface e finalização do código	8h	Henrique Thomas Correa Alonso da Silva		
18/05/2022	Desenvolvimento do Trabalho Teórico	5h	Henrique Thomas Correa Alonso da Silva		
20/05/2022	Desenvolvimento do Conclusão do Trabalho Teórico	5h	Henrique Thomas Correa Alonso da Silva		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80hrs
AVALIAÇÃO: _____
NOTA: _____ Aprovado ou Reprovado
DATA: ____/____/____
CARIMBO E ASSINATURA DO COORDENADOR DO CURSO _____

**FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS**

NOME: Guilherme akio ochi

TURMA: ccSP17RA: N5989F3CURSO: Ciência da Computação













—CAMPUS: Sorocabá (17)

SEMENTRE: 5º Sementre TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B2

SEMESTRE: 5º Semestre

ANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
07/04/22	Formação do grupo e divisão de tarefas	5			
08/04/22	Reunião de brainstorm do projeto	5			
09/04/22	Pesquisa sobre formatação abnt	7			
11/04/22	Pesquisa sobre o tema proposto	10			
13/04/22	Conversa com integrantes do grupo	5			
14/04/22	Início da escrita da formatação abnt	5			
16/04/22	Pesquisa da parte teórica do código da programação	7			
20/04/22	Pesquisa e comunicação com integrante do grupo	6			
10/05/22	Revisando pesquisas de integrantes e formatação abnt	10			
15/05/22	Formatação de texto para norma abnt	10			
17/05/22	Conversa em grupo para ajustes	5			
20/05/22	Últimos ajustes sobre a formatação abnt	5			

(1) Horas atribuídas de acordo com o regulamento das Atividades Supervisionadas do curso.

TOTAL DE HORAS ATRIBUIDAS: 80

AVALIAÇÃO:

Aprovado ou Reprovado

NOTA: _____

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Mateus Garcia dos Santos

TURMA: cc5q17RA: f208613CURSO: Ciência da Computação

CAMPUS: Sorocaba (17)

SEMESTRE: 5º Semestre TURNO: Noturno

CÓDIGO DA ATIVIDADE: 7782

SEMESTRE: 5º Semestre

ANO GRADE: 2022[illegible]

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 81 _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: / /

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Gabriele Cristine da Silva Casari

TURMA: (CCSP12)

RA: (F0865C0)

CURSO: Ciência da Computação

CAMPUS: Sorocaba (12)

SEMESTRE: 2º Semestre TURNO: Noturno

CÓDIGO DA ATIVIDADE: 77B2

SEMESTRE: 2º Semestre

ANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
07/04/2022	Formação do grupo e divisão de tarefas	4h	<i>Gabriele Casari</i>		
08/04/2022	Reunião de brainstorm do projeto	3h	<i>Gabriele Casari</i>		
11/04/2022	Pesquisas sobre sockets Berkeley	6h	<i>Gabriele Casari</i>		
13/04/2022	Estudo da linguagem C#	6h	<i>Gabriele Casari</i>		
14/04/2022	Leitura do livro Use a Cabeça C# de Andrew Stellman E Jennifer Greene	6h	<i>Gabriele Casari</i>		
15/04/2022	Desenvolvimento do Front-end do projeto	5h	<i>Gabriele Casari</i>		
17/04/2022	Testes do código front-end do projeto	3h	<i>Gabriele Casari</i>		
20/04/2022	Desenvolvimento do Back-end do projeto	6h	<i>Gabriele Casari</i>		
21/04/2022	Estruturação do Documento ABNT	6h	<i>Gabriele Casari</i>		
25/04/2022	Testes do código back-end do projeto	3h	<i>Gabriele Casari</i>		
10/05/2022	Contribuição com Pesquisa Teórica do Software	6h	<i>Gabriele Casari</i>		
11/05/2022	Contribuição com Pesquisa Teórica da ABNT	6h	<i>Gabriele Casari</i>		
12/05/2022	Contribuição de referências bibliográficas da ABNT	4h	<i>Gabriele Casari</i>		
13/05/2022	Testes e ajustes da interface e finalização do código	6h	<i>Gabriele Casari</i>		
18/05/2022	Desenvolvimento do Trabalho Teórico	5h	<i>Gabriele Casari</i>		
20/05/2022	Desenvolvimento de Conclusão do Trabalho Teórico	5h	<i>Gabriele Casari</i>		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80hrs

AValiação: _____

Aprovado ou Reprovado

Nota: _____

Data: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Thiara de Paula SouzaTURMA: CCAP17RA: F331C8CURSO: Ciência da ComputaçãoCAMPUS: Zorocoba (17)SEMESTRE: 5º Semestre TURNO: NocturnoCÓDIGO DA ATIVIDADE: 7782SEMESTRE: 5º SemestreANO GRADE: 2022

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
07/04/2022	Formação do grupo e divisão de tarefas	4h	<u>Thiara P. Souza</u>		
08/04/2022	Reunião de brainstorm do projeto	2h	<u>Thiara P. Souza</u>		
11/04/2022	Pesquisas sobre sockets Berkeley	6h	<u>Thiara P. Souza</u>		
13/04/2022	Estudo da linguagem C#	10h	<u>Thiara P. Souza</u>		
14/04/2022	Leitura do livro Use a Cabeça C# de Andrew Stellman E Jennifer Greene	3h	<u>Thiara P. Souza</u>		
15/04/2022	Desenvolvimento do Front-end do projeto	5h	<u>Thiara P. Souza</u>		
17/04/2022	Testes do código front-end do projeto	3h	<u>Thiara P. Souza</u>		
20/04/2022	Desenvolvimento do Back-end do projeto	15h	<u>Thiara P. Souza</u>		
25/04/2022	Testes do código back-end do projeto	3h	<u>Thiara P. Souza</u>		
10/05/2022	Contribuição com Pesquisa Teórica do Software	3h	<u>Thiara P. Souza</u>		
11/05/2022	Contribuição com Pesquisa Teórica da ABNT	6h	<u>Thiara P. Souza</u>		
12/05/2022	Contribuição de referências bibliográficas da ABNT	2h	<u>Thiara P. Souza</u>		
13/05/2022	Testes e ajustes da interface e finalização do código	8h	<u>Thiara P. Souza</u>		
18/05/2022	Desenvolvimento do Trabalho Técnico	5h	<u>Thiara P. Souza</u>		
20/05/2022	Desenvolvimento de Conclusão do Trabalho Técnico	5h	<u>Thiara P. Souza</u>		
(1) Horas atribuídas de acordo com o regulamento das Atividades Supervisionadas do curso.					
TOTAL DE HORAS ATRIBUÍDAS: <u>80hrs</u>					

AVALIAÇÃO: _____

NOTA: _____

Aprovado ou Reprovado

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO