CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Carlos Daniel Silva de Sousa

IPWebCalc:
Calculadora IPv4 Web

Palmas – TO

2023

Carlos Daniel Silva de Sousa

IPWebCalc: Calculadora IPv4 Web

Monografia elaborada e apresentada na disciplina de Projeto Tecnológico como requisito parcial para obtenção do título de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas.

Orientador: Prof. M.e Madianita Bogo Marioti.

Palmas – TO

2023

Carlos Daniel Silva de Sousa

IPWebCalc: Calculadora IPv4 Web

Monografia elaborada e apresentada na disciplina de Projeto Tecnológico como requisito parcial para obtenção do título de bacharel em Engenharia de Software pelo Centro Universitário Luterano de Palmas.

Orientador: Prof. M.e Madianita Bogo Marioti.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Madianita Bogo Marioti

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Profº. Esp. Fábio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

_____

Profº. Esp. Fernanda Pereira Gomes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2023

RESUMO

Silva de Sousa, Carlos Daniel. **IPWebCalc: Calculadora IPv4 Web**. 2023. Projeto Tecnológico (Graduação) – Engenharia de Software, Centro Universitário Luterano de Palmas, Palmas/TO, 2023[1].

O aprendizado e administração de endereçamento IPv4 e sub-redes pode ser algo complexo, devido à grande quantidade de detalhes envolvidos nos cálculos das informações. O endereçamento IPv4 envolve a utilização de uma série de técnicas para gerenciar e identificar os dispositivos conectados a uma rede, sendo preciso entender como realizar a divisão de endereços em sub-redes e identificar as informações relacionadas. Ao analisar algumas calculadoras de informações de redes e sub-redes disponíveis na web, notou-se que há uma apresentação de informaçoẽs pouco intuitiva, com interfaces confusas e que não oferecem todos os cálculos básicos, informações que geram ainda mais dificuldade para os acadêmicos e para os profissionais a compreender e resolver suas dúvidas. Por tanto, neste projeto foi desenvolvido uma calculadora IPv4, que calcula e retorna em um único ambiente as informações de rede e sub-redes que são abordados em conteúdos de redes de computadores, que retorna resultados completos, organizados para o auxílio de acadêmicos e profissionais da área.

Palavras-chave: Redes. IPv4. Sub-Redes.

1

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Internet é uma rede de computadores mundial pública, que conecta milhões de dispositivos computacionais pelo mundo (KUROSE; ROSS, 2005). Como afirma Soares (1995), redes de computadores são "um conjunto de módulos processadores capazes de trocar informações e compartilhar recursos, interligados por um sistema de comunicação". A comunicação entre dispositivos é essencial na era em que vivemos. Para que possamos trocar informações entre computadores, tablets, *smartphones* e outros dispositivos, é necessário uma forma de conexão que permita essa interação. Nesse sentido, é utilizada a arquitetura TCP/IP, que é um conjunto de regras que garante a transmissão de dados de forma eficiente e segura na internet.

A arquitetura TCP/IP, que forma um modelo padrão de comunicação entre dispositivos que estabelece a troca de informações dá origem ao destino. O nome TCP/IP vem dos seus principais protocolos, o TCP e o IP (KUROSE; ROSS, 2005). O TCP (*Transmission Control Protocol* - Protocolo de Controle de Transmissão) é um protocolo orientado a conexão, que garante a entrega fim-a-fim de dados e recursos, fornecendo uma transmissão confiável entre as aplicações. O protocolo IP (*Internet Protocol*) é responsável pelo endereçamento e roteamento (definição do caminho) das mensagens entre os dispositivos.

Segundo Silveira (1995), para duas máquinas se comunicarem utilizando o protocolo TCP/IP, cada uma destas máquinas precisa ter um endereço IP diferente, pois é através do endereço IP que é possível identificar uma determinada máquina. Portanto, o endereço IP é o identificador que permite que as máquinas sejam reconhecidas em sua rede local ou em outras redes, de forma que qualquer dispositivo conectado na Internet possa ser localizado. O endereço IPv4, versão do IP em uso na maioria das redes que compõem a Internet, é um conjunto de 32 bits, dividido em 4 octetos, onde cada octeto pode ir numa escala de 0 a 255, possibilitando 4294967296 endereços diferentes (MUNIZ *et al.*, 2017). De acordo com PAMPLONA, o endereço IP foi projetado de forma a identificar a rede e o *host* (dispositivo), em que a parte que identifica a rede representa a rede a qual o dispositivo está conectado, enquanto a parte que identifica o *host* representa o próprio dispositivo naquela rede.

O endereçamento IPv4 possui uma divisão em classes, sendo elas A, B, C, D e E, no entanto, são usadas apenas A, B e C (FRANCISCATTO; CRISTO; PERLIN, 2014). A classe A tem como padrão informar seu endereço de rede no primeiro octeto, que pode ir de 1 à 126, disponibilizando os outros octetos para *host*. Já a classe B que vai de 128 a 191 e possui dois octetos para identificar a rede e apenas dois octetos para identificar o *host*. A classe C é a que tem menor disponibilidade para *host*, já que seus três primeiros octetos identificam a rede,

deixando apenas um para *host*. Contudo, a classificação se mostrou ineficiente, visto que com o uso de blocos pré definidos, gerava um grande desperdício de conexões (*host*) (Kurose, Ross, 2002). Para sanar essa falha surgiu a divisão de sub-redes, possibilitando um melhor aproveitamento dos endereços e oferecendo um maior desempenho e mais segurança.

Na divisão de sub-redes um endereço de rede é sub-dividido em duas ou mais sub-redes distintas. Por exemplo, um endereço de classe B como 128.10.0.0 possui os dois últimos octetos disponíveis para conexão (128.10.conexão.conexão). Ao dividir o endereço em sub-redes: 128.10.sub-rede.conexão; o campo destinado à conexão é subdividido, e o primeiro octeto é utilizado como identificador de sub-rede, permitindo um melhor aproveitamento do endereço.

Logo, a divisão de sub-redes apresenta algumas vantagens, como: desempenho, pois ao realizar a divisão o tráfego se torna menor e diminui o fluxo de dados; e segurança, pois, com a divisão os dados não trafegam no meio físico de outras sub-redes, assim, evitando a interceptação de informações. No entanto, para que haja uma divisão correta de uma faixa de um endereço IP, é necessário identificar se o atual endereço já possui alguma divisão.

Para a identificação de um endereço IP, é utilizada como base a máscara, que mostra qual parte do endereço IP é destinado à rede e qual parte do endereço é destinado aos *hosts*. Ela é formada por uma sequência contínua de 4 octetos de bits 1 e 0, onde os bits 1 identificam a parte correspondente à rede e os bits 0 identificam a parte correspondente a *host*. Por exemplo, em um endereço “192.168.100.0”, com máscara “255.255.255.0”, os primeiros 3 octetos (192.168.100) destinam a rede, enquanto o último octeto (0) restante destina ao *host*.

Dessa forma, percebeu-se que realizar cálculos de informações sobre sub-redes como, por exemplo, máscara, endereço de rede e endereço de *host*, torna-se algo complicado. No entanto, é indispensável que administradores e estudantes de redes tenham o domínio sobre estes cálculos. Logo, realizar cálculos relacionados a redes e sub-redes é uma habilidade essencial para os administradores de redes, pois permite que eles planejem e gerenciem melhor a rede, evitando conflitos de endereço e garantindo a segurança da rede.

O conhecimento de sub-redes permite aos administradores dividir uma rede grande em várias sub-redes menores, o que ajuda a reduzir o tráfego de rede e a melhorar o desempenho. Além disso, ao entender como as sub-redes funcionam, os administradores podem gerenciá-las para controlar o acesso à rede e garantir a privacidade e a integridade dos dados.

Por outro lado, o conhecimento sobre os cálculos de sub-redes é fundamental não apenas aos administradores de redes, mas também aos estudantes de redes. Segundo as diretrizes do MEC(2012), Redes de Computadores é uma disciplina obrigatória ou

recomendada para cursos da área de computação, como Engenharia de Software, Ciências da Computação e Sistemas de Informações. Portanto, percebe-se que a disciplina de redes de computadores é fundamental para a área de computação.

Portanto, verificou-se que a realização de cálculos relacionados às informações de sub-redes, como máscara, endereço de rede e endereço de *host*, é um aspecto crucial para a formação acadêmica de estudantes de redes. Diante disso, é importante que os alunos adquiram habilidade nesse tipo de cálculo, tendo em vista que se trata de um conhecimento imprescindível tanto para a aprendizagem quanto para a administração de redes. Dessa maneira, chegou-se a conclusão que uma calculadora IP web que oferece esses tipos de serviços seria interessante para auxiliar tanto na aprendizagem quanto na administração de redes.

Na Internet, foram encontradas várias calculadoras de IP que buscam auxiliar no cálculo de sub-redes, que oferecem funcionalidades onde o usuário consegue realizar diversos cálculos como, por exemplo, ao inserir apenas o endereço IP no formato CIDR, a calculadora consegue identificar a máscara, o tipo de classe e o *broadcast,* além de os apresentar esses dados em binário. Porém, não foi encontrada uma que apresentou as informações de acordo com a abordagem da disciplina de Redes de Computadores, apresentando muitas informações, o que poderia confundir os alunos, e outras não apresentavam todas as informações necessárias em uma só tela.

Diante disso, decidiu-se criar uma calculadora personalizada, que fosse adequada ao conteúdo e à metodologia de trabalho da disciplina. Uma calculadora baseada em materiais didáticos, como o livro "Redes de Computadores" de Tanenbaum, amplamente reconhecido na área, e conteúdos disponibilizados pelo CEULP (Centro Universitário Luterano de Palmas). Essa calculadora teria como objetivo apresentar, em uma única interface, todos os cálculos relacionados a endereçamentos IPv4, de forma clara e concisa, superando as limitações das calculadoras existentes.

Uma ferramenta que oferece uma experiência amigável e intuitiva e que realiza cálculos a partir de diferentes entradas do usuário, como, por exemplo, encontrar o endereço de rede e de *broadcast* através do IP de um *host* e máscara; calcular a máscara a partir da quantidade de endereços etc. Com linguagem simples e direta, para que o usuário tenha uma maior flexibilidade em atender suas demandas, sejam profissionais ou acadêmicas, apresentando uma interface que o permite utilizá-la intuitivamente.

Portanto, o objetivo desse projeto foi desenvolver uma calculadora IPv4 que oferece cálculos relacionados ao endereçamento IPv4 de forma organizada e didática, seguindo os

padrões adotados na disciplina do CEULP/ULBRA. Essa calculadora tem como finalidade facilitar a compreensão e auxiliar alunos e profissionais a compreender os conceitos de endereçamento de redes e sub-redes, ao realizar cálculos de informações de endereçamento de maneira automatizada e organizada. Assim, espera-se que ela auxilie os professores a ensinar de uma forma dinâmica em sala de aula, realizando atividades de reforço com o uso da calculadora.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos fundamentais das redes de computadores, destacando o protocolo IP e suas regras. Serão apresentadas as principais características e funcionalidades do protocolo IP, bem como os conceitos e regras essenciais relacionados ao endereçamento IP.

2.1 Protocolo IP

Criado por volta de 1970 pela ARPANET (*Advanced Research Projects Agency Network*), o protocolo IP(*Internet Protocol)* é um protocolo de comunicação utilizado para conectar vários dispositivos em uma rede. O IP é o principal protocolo da camada de redes e possui algumas funcionalidades, como afirma Muniz *et al.* (2017), “o protocolo IP foi definido na RFC 791 para prover duas funções básicas: a fragmentação, possibilitando o envio de pacotes maiores que o limite de tráfego estabelecido num enlace, com a divisão deles em partes menores; e o endereçamento, que permite identificar o destino e a origem dos pacotes a partir dos endereços armazenados no cabeçalho do protocolo.”, portanto o protocolo IP possui algumas funcionalidade, sendo suas principais:

- **Endereçamento:** através de um esquema de endereçamento único para os dispositivos, o protocolo IP permite que cada máquina seja identificada e localizada;
- **Fragmentação:** ip pode fragmentar grandes pacotes em pacotes menores, permitindo assim sua transmissão em redes com diferentes tamanhos de MTU (Unidade Máxima de Transmissão);
- **Roteamento:** ao dividir os dados em pacotes, o protocolo ip atribui informações de cabeçalho, como por exemplo, o endereço de origem e o endereço final, permitindo que os pacotes enviados cheguem ao destino correto.

Atualmente, o protocolo IP está em sua versão 6 (IPv6), no entanto, a versão mais utilizada nos dispositivos é o IPv4. Como afirma SMETANA (2012), “a especificação do IPv4 foi publicada em setembro de 1981, sob o RFC 791, com o auxílio do Information Sciences Institute – *University of Southern California* (Instituto de Ciências da Informação da Universidade do Sul da Califórnia)". O IPv4 utiliza endereços com 32 bits, possibilitando mais de 4 bilhões de endereços onde sua arquitetura é formada por duas partes, uma que identifica a rede e outra que identifica o *host*.

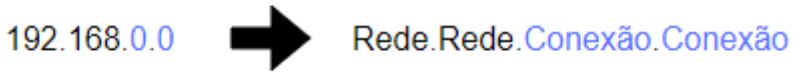
2.2 Endereçamento IPv4

Segundo Lima Junior (2017), o protocolo IP em sua versão 4 é composto por endereços com 32 bits divididos em 4 octetos separados por pontos, onde cada octeto ao ser convertido para decimal pode ir numa escala de 0 a 255, como, por exemplo, 192.168.42.10. Ao realizar o cálculo 2^{32}, considerando que os endereços possuem 32 bits, essa arquitetura tem como resultado um total de 4294967296 endereços únicos possíveis para identificar dispositivos na rede.

> Os 32 bits de endereçamento do IPv4 estão separados em duas partes, sendo que a primeira informa o endereço de rede e a segunda, o endereço de *host*. A representação do endereço IPv4 é feita através da chamada notação decimal pontuada. Nela, cada um dos quatro bytes do endereço IPv4 é representado pelo seu valor decimal, separados por um ".". (SMETANA, 2012).

portanto, a faixa de endereço é constituída por duas partes, a parte da rede, que identifica a rede em que o dispositivo está conectado, e a parte do *host*, que identifica o próprio dispositivo na rede, como mostra a Figura 1.

Figura 1. Endereço IP



Fonte: Autor

Por sua vez, as partes da identificação do endereço IP são determinadas através da divisão do endereço em classes. Essa divisão foi feita para ajudar a definir como os endereços seriam atribuídos e quais tamanho de redes suportariam. A divisão em classes foi feita em três principais classes: A, B e C. A classe A é destinada a redes muito grandes, a classe B para redes médias e a classe C para redes pequenas. Cada classe tem sua própria faixa de endereços IP, garantindo que os endereços IP sejam atribuídos de maneira eficiente e organizada. Essa divisão é ilustrada na figura 2.

Figura 2. Identificação da rede e conexão das classes

Classe A	Rede	Conexão	Conexão	Conexão
Classe B	Rede	Rede	Conexão	Conexão
Classe C	Rede	Rede	Rede	Conexão

Fonte: Autor

A divisão por classes não apenas identifica as partes no endereço, mas também flexibiliza o roteamento, possibilitando uma melhor distribuição de IPs para diferentes tamanhos de redes, conforme afirma Braga (2011), “A estratégia de divisão dos endereços em classes tinha como objetivo tornar a distribuição de endereços mais flexível, podendo criar redes de tamanhos diferentes”, como pode ser observado a seguir:

- **Classe A:** destinada a organizações que possuem poucas redes e muitas conexões (*host*), esta classe o endereço é caracterizado por seus primeiros 8 bits, que identificam a rede e podem ir de 0 à 126, os 24 bits restantes ficam destinados a *host*. Exemplo de endereços de classe A: “17.0.0.1”, “126.128.0.1”, "10.10.10.10".
- **Classe B:** na classe B a quantidade do número de redes e de conexões (*host*) são iguais, nesta classe o endereço é caracterizado por seus primeiros 16 bits, que identificam a rede e seus primeiros 8 bits podem ir de 128 a 191, os 16 bits restantes ficam destinados a *host*. Exemplo de endereços de classe B: “172.16.0.1”, “192.168.0.1”, “150.100.0.1”.
- **Classe C:** a classe C é destinada a organizações que possuem poucas conexões (host) e muitas redes, esta classe o endereço é caracterizado por seus primeiros 24 bits, que identificam a rede e seus primeiros 8 bits podem ir de 192 a 223, os 8 bits restantes ficam destinados a *host*. Exemplo de endereços de classe C: “192.0.2.1”, “223.220.0.1”, “198.162.1.1”.
- **Classe D:** faixa de 224 a 239, no entanto, essa classe é destinada a uso de endereços de *multicast,* em outras palavras, ela permite a comunicação de dados em grupo.
- **Classe E:** faixa de 240 a 247, esta classe é destinada a uso futuro.

A figura 3 a seguir apresenta como foi definida a divisão por classes dos endereços IP, ilustrando como foram divididas as faixas de endereços. Por exemplo, na tabela da imagem, pode-se obter as seguintes informações: a classe A começa com o endereço 1.0.0.0 e termina em 126.255.255.255, permitindo a criação de apenas 126 redes, cada uma com 1677214 conexões (*host*) possíveis.

Figura 3. Faixas das Classes

Classes	Endereço Inicial	Endereço Final	Número de Redes	Número de Endereços	Número de Conexõs
A	1.0.0.0	126.255.255.255	126	16.777.216	16.777.214
B	128.0.0.0	191.255.255.255	16.384	65.536	65.534
C	192.0.0.0	223.255.255.255	2097151	256	254
D	224.0.0.0	239.255.255.255	Multicast		
E	240.0.0.0	247.255.255.255	Reservado		

Fonte: Autor

Existem ainda os endereços reservados, que não podem ser endereçados associados a nenhum equipamento:

- **Endereço de Rede**: é o endereço IP utilizado para identificar uma rede específica na qual os dispositivos estão conectados. Ele é definido como o primeiro endereço de uma faixa de endereços IP disponíveis para a rede, e não pode ser associado a nenhum dispositivo individual. Por exemplo, em uma rede com endereço de rede 192.168.0.0, o endereço 192.168.0.1 seria o primeiro endereço disponível para um dispositivo naquela rede, enquanto 192.168.0.0 seria o endereço de rede em si, utilizado para identificar a rede como um todo.
- **Endereço de *Loopback***: todos os endereços que iniciam com 127.x.x.x são reservados a Loopback, este endereço representa a própria máquina local em que ele é usado. É utilizado para testes de *software* que utilizam processos em um mesmo dispositivo, como, por exemplo, sistemas cliente/servidor. Quando um processo tenta se comunicar usando o endereço de Loopback, a mensagem é enviada para o próprio dispositivo e não é encaminhada para a rede externa.
- **Endereço de *Broadcast***: se trata de um endereço de difusão e é sempre o último endereço de uma determinada rede. Quando um pacote de dados é enviado para o endereço de *broadcast*, todos os dispositivos conectados àquela rede recebem o pacote, independentemente do endereço IP de destino.
- **Endereços Privados**: são endereços IP reservados para redes internas privadas, que não são válidos na Internet. Eles permitem que várias máquinas compartilhem o mesmo endereço IP público, que é atribuído a um roteador ou gateway. As faixas de endereços reservadas são:

- Classe A: 10.0.0.0 a 10.255.255.255,
- Classe B: 172.16.0.0 a 172.31.255.255
- Classe C: 192.168.0.0 a 192.168.255.255

A divisão em classes teve como objetivo flexibilizar e endereçamento e possibilitar uma configuração de redes de tamanhos variados. No entanto, de acordo com Kurose e Ross (2002) “a exigência de que a parcela de rede de um endereço IP tenha exatamente um, dois ou três bytes há muito tempo se tornou problemática para acomodar o número crescente de organizações com redes de pequeno e médio porte.”, ou seja, a classificação se mostrou ineficiente, pois com o uso de blocos pré-definidos, muitas vezes ocorria um grande desperdício de endereços de redes e conexões (*host*), como, por exemplo, para endereçar 500 máquinas em uma organização, seria necessário utilizar um bloco de endereço de classe B, com isso acarretaria no desperdício de mais de 65 mil endereços. Assim, para superar essa limitação, foi criado a técnica de divisão de sub-redes, que será abordado a seguir.

2.3 Sub-Redes

Em 1993, com a publicação dos RFCs 1518 e 1519, foi introduzida a técnica de sub-redes, onde foi estabelecido o Classless Inter-Domain Routing (CIDR-Roteamento Inter-Domínio sem Classe) como uma solução para o esgotamento dos endereços IPv4, permitindo o uso de máscaras para criar sub-redes e proporcionando maior flexibilidade no endereçamento IPv4 (SMETANA, 2012). A criação da técnica de sub-redes tem como finalidade buscar um melhor aproveitamento dos endereços disponíveis em uma rede, permitindo ao administrador dividir uma rede em faixas menores de redes internas. Como afirma TANENBAUM (2011), "permitir que uma rede seja dividida em diversas partes para uso interno, como várias redes, mas externamente continue a funcionar como uma única rede. Isso é subdivisão de rede".

Com a adoção da técnica de sub-redes o endereçamento IPv4 ganhou uma maior flexibilidade, deixando de usar blocos pré-definidos e utilizando blocos de acordo com necessidade da organização, gerando menos desperdício. “Sub-redes fornece vários tipos de vantagens, como melhora na segurança, habilidade de gerenciar recursos de redes locais, diminui a necessidade de utilizar todos os endereços IPS, diminui o tamanho das tabelas de roteamento no núcleo dos roteadores, reduz o tráfego de broadcast e ainda melhora a utilização de largura de banda.”(KAMIS; TOPI, 2007).

O uso de sub-redes oferece uma série de vantagens, como: desempenho, pois ao realizar a divisão de um rede em sub-redes é realizada uma redução no tráfego de dados na

rede; e segurança, pois, ao realizar a divisão, é possível ter um melhor controle no tráfego da rede, evitando assim acessos indevidos e interceptação de informações.

Na divisão em sub-redes é possível utilizar bits que antes eram destinados a conexão(*host*) para a definição da rede, dessa forma possibilitando que a rede seja particionada em redes menores. Logo, a divisão em sub-redes têm como finalidade um melhor gerenciamento dos endereços IP disponíveis e a otimização da infraestrutura de rede, permitindo um uso mais eficaz dos endereços disponíveis.

Por exemplo, ao considerar um endereço de classe A como 100.0.0.0, onde o primeiro octeto (100) identifica a rede e os três últimos octetos (0.0.0) identificam a conexão, é possível utilizar os octetos destinados à conexão para definir sub-redes, como 100.*subrede.conexão.conexão*. Dessa forma, uma rede pode ser subdividida em várias outras, permitindo uma melhor utilização dos endereços disponíveis alocando-os de acordo com a necessidade da organização.

O mesmo ocorre com endereços de classe B, como 192.168.0.0, onde os dois primeiros octetos (192.168) identificam a rede e os dois últimos (0.0) a conexão. Ao aplicar a divisão em sub-redes, os octetos destinados à conexão são utilizados para definir sub-redes, como 192.168.*subrede.conexão.* Isso possibilita uma maior flexibilidade sobre os endereços IP disponíveis, permitindo uma melhor organização e divisão da rede.

No caso de endereços de classe C, como 200.100.50.0, onde os três primeiros octetos (200.100.50) identificam a rede e o último (0) a conexão, é possível utilizar a divisão em sub-redes para criar sub-redes mais precisas, como 192.168.0.*subrede+conexão*. Assim como nos casos anteriores, essa técnica permite uma melhor utilização dos endereços disponíveis e uma gestão mais eficiente da infraestrutura de rede, possibilitando a criação de várias sub-redes em uma mesma rede.

Os exemplos demonstram como a técnica de sub-redes utiliza os bits das conexões (*host*) para identificar as sub-redes. Entretanto, essa prática torna difícil distinguir qual faixa de endereço pertence a rede e qual se torna a conexão(*host*). Para que haja uma identificação de maneira correta de um endereço IP é utilizada a máscara de rede.

2.4 Máscara de Rede

A máscara de rede é utilizada para a identificação de um endereço IP, pois ela permite verificar se a rede está dividida, quantos endereços IP há na rede e qual parte do endereço pertence à rede e qual parte pertence à conexão (PENA, 2018). A máscara define a quantidade de bits destinados à rede e à conexão, permitindo a identificação correta da rede e dos *hosts*

que a compõem, como afirma Franciscatto, Cristo e Perlin (2014), “é importante salientar que a máscara de rede determina qual parte do endereço IP é destinado a identificar a rede e qual delas endereçam os hosts”.

Conforme os estudos de Lima Junior (2017), a máscara de rede é constituída por uma sequência contínua de 4 octetos de bits 1 seguida de uma sequência de bits 0, onde os bits 1 identificam a parte correspondente a rede e as sub-redes, e os bits 0 identificam a parte correspondente a conexão(*host*), como pode ser observado em máscaras padrões. Uma máscara de sub-rede padrão é baseada na classe do endereço IP e é usada em redes que ainda não foram subdivididas (Neto, 2010), como observado na Tabela 1.

Tabela 1. Máscaras Padrões

Classes	Máscara em Binário	Máscara em Decimal
A	11111111.00000000.00000000.0000000	255.0.0.0 ou /8
B	11111111.11111111.00000000.0000000	255.255.0.0 ou /16
C	11111111.11111111.11111111.0000000	255.255.255.0 ou /24

Fonte: Autor

As máscaras padrões ilustradas na Tabela 1 demonstraram de forma explícita quais octetos pertencem à rede e quais octetos pertencem à conexão, ou seja, demonstra quais bits estão disponíveis para realizar a divisão de sub-redes.

Por exemplo, em um endereço associado a uma máscara seja de 255.0.0.0, apenas o primeiro octeto é destinado a rede e, consequentemente, o restante é destinado às conexões (*host*). Ao utilizar o endereço IP 12.5.5.3 com máscara 255.0.0.0, significa que a única parte que identifica a rede é o primeiro octeto (12). Portanto, todas as conexões (*host*) dessa rede iniciarão com 12 no primeiro octeto, enquanto os outros três octetos irão variar para identificar cada conexão específica, como, por exemplo, 12.10.10.50.

Em um endereço associado a uma máscara seja de 255.255.0.0, apenas o primeiro octeto e o segundo octeto são destinados a rede e, consequentemente, o restante é destinado às conexões (*host*). Por exemplo, ter o endereço IP 12.5.5.3 com máscara 255.255.0.0, significa que a única parte que identifica a rede é o primeiro e o segundo octeto (12.5). Portanto, todas as conexões (*host*) dessa rede iniciarão com 12.5 no primeiro e segundo octeto, enquanto os

outros dois octetos irão variar para identificar cada conexão específica, como, por exemplo, 12.5.50.60.

Assim, seguindo a lógica dos exemplos anteriores, em um endereço associado a uma máscara seja de 255.255.255.0, os três primeiros octetos são destinados à rede e, consequentemente, o último octeto é destinado às conexões (*host*). Por exemplo, ter o endereço IP 12.5.5.3 com máscara 255.255.255.0, significa que a única parte que identifica a rede é os três primeiros octetos (12.5.5). Portanto, todas as conexões (*host*) dessa rede iniciarão com 12.5.5 nos primeiros octetos, enquanto o último octeto vai variar para identificar cada conexão específica, como, por exemplo, 12.5.5.100.

Como a máscara é uma sequência contínua de bits 1 seguida de bits 0, é importante destacar que, embora as máscaras padrões permitam identificar claramente os octetos que pertencem à rede e à conexão, em algumas redes subdivididas pode ocorrer uma subdivisão parcial de um octeto, onde parte deste octeto vai identificar à rede e a outra parte vai identificar à conexão, como pode ser observado na Tabela 2.

Tabela 2. Máscaras com Subdivisões no Octeto

Máscara em Binário	Máscara em Decimal
10000000	128
11000000	192
11100000	224
11110000	240
11111000	248
11111100	252
11111110	254
11111111	255

Fonte: Autor

Logo, em uma rede com máscara 255.255.255.224 (11111111.11111111.11111111.11100000), os três primeiros bits do último octeto são

utilizados para identificar as sub-redes, enquanto os outros cinco bits são utilizados para identificar os *hosts*. Por isso, em alguns casos, é possível que apenas uma parte de um octeto identifique a rede e outra parte identifique o *host*. Além disso, a tabela 2 apresenta os valores máximos e mínimos de *hosts* e sub-redes possíveis para cada máscara.

Para realizar a subdivisão de redes em sub-redes, é necessário seguir os padrões de máscara estabelecidos na tabela 2, respeitando a necessidade das máscaras serem uma sequência contínua de bits 1 seguida de bits 0. É importante ressaltar que é obrigatório utilizar uma das máscaras pré definidas e não é possível utilizar uma máscara diferente (NETO, 2010). Quando há a necessidade de utilizar um valor diferente de 255 em uma máscara, os primeiros octetos devem receber o valor máximo de 255, seguidos por um dos valores disponíveis na tabela 2, garantindo que a parte da rede seja identificada corretamente e que os bits restantes possam ser usados para subdividir a rede em sub-redes. Assim, é possível definir máscaras válidas, como 255.255.128.0 (11111111.11111111.10000000.00000000) e 255.255.255.248 (11111111.11111111.11111111.11111000).

Cabe ressaltar que ao utilizar uma máscara que possua octetos com valores diferentes dos padrões estabelecidos na tabela 2, a máscara estará incorreta. Por exemplo, máscaras como

Tabela 3. Máscaras Inválidas

Decimal	Binário
255.255.255.120	11111111.11111111.11111111.01111000
255.128.255.0	11111111.10000000.11111111.11111111
240.255.255.255	11110000.11111111.11111111.11111111
128.255.240.255	10000000.11111111.11110000.11111111

Fonte: Autor

não seguem a premissa de ser uma sequência e 1s e outra sequência de 0s não intercalados, portanto, estão em notação incorreta.

Ao utilizar uma máscara que possua octetos com valores que seguem os padrões estabelecidos na tabela 2, a máscara estará correta. Dessa forma, máscaras como

Tabela 4. Máscaras Válidas

Decimal	Binário
255.255.255.128	11111111.11111111.11111111.10000000
255.252.0.0	11111111.11111100.00000000.00000000
255.255.255.254	11110000.11111111.11111111.11111110
255.192.0.0	11111111.11000000.00000000.00000000

Fonte: Autor

são exemplos de máscaras corretas que obedecem a essa premissa da máscara ser uma sequência de bits 1s seguido por bits 0s.

Dessa forma, compreender a relação entre a máscara e o endereço de rede é essencial para uma eficiente gestão da faixa de rede. É a partir dessa associação que se torna possível realizar uma administração mais precisa e adequada das sub-redes e dos *hosts* dentro da rede. Para tornar essa associação mais compreensível, é fundamental entender de forma mais aprofundada como funciona o cálculo de divisão de sub-redes, que será abordado na próxima seção.

2.4.1 Exemplos de Sub-redes

Conforme os estudos de Franciscatto, Cristo e Perlin (2014), cada octeto da máscara tem um valor que representa a quantidade de bits usados para identificar a rede e a quantidade de conexões (*host*) possíveis para aquela rede. Isso significa que cada octeto tem um número máximo de conexões (*host*) possíveis, que é determinado pelo valor do octeto. Por exemplo, uma máscara que segue o padrão 255.255.255.0 indica que os três primeiros octetos (255) definem a rede, e somente o último octeto está disponível para uso como identificador de conexões (*host*). Isso permite que até 256 *hosts* possam ser conectados em uma rede com essa máscara.

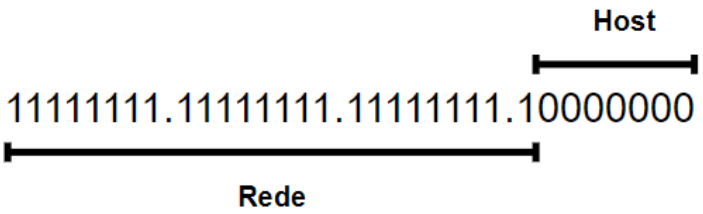
Para calcular a quantidade máxima de endereços possíveis para um octeto da máscara, é preciso utilizar uma fórmula que consiste em elevar 2 ao número de bits 0 disponíveis para conexão. Por exemplo, considerando o endereço associado à máscara 255.255.255.128 (11111111.11111111.11111111.10000000), em que o último octeto tem apenas o primeiro bit definido como 1 e 7 bits 0 disponíveis para conexão, 2^7 é igual a 128, significa que este é o número máximo de endereços possíveis para este octeto.

Logo, para obter o total de endereços possíveis para uma rede, é necessário utilizar a

fórmula ilustrada anteriormente para cada octeto e por fim multiplicar todos os resultados. Então, ao considerarmos um endereço associado a uma máscara 255.242.0.0 (11111111.11111100.00000000.00000000), por exemplo, podemos calcular o número total de endereços possíveis para essa faixa de rede. Para isso, é necessário utilizar a fórmula ilustrada anteriormente para cada octeto e, por fim, multiplicar todos os resultados. Nesse caso, temos $2^2 * 2^8 * 2^8$, o que resulta em 4 * 256 * 256, totalizando 262144 endereços possíveis para essa faixa de rede.

Agora, utilizando um endereço original de classe C 200.199.229.0 com a máscara 255.255.255.128 (1111111.1111111.11111111.10000000 ou /25), é possível observar que os 25 primeiros bits representam a rede, e os 7 bits restante identifica a conexão (*host*), como pode ser observado na figura 4.

Figura 4. Identificação da Máscara



Fonte: Autor

Logo, ao utilizar essa máscara se torna possível utilizar apenas os últimos 7 bits do quarto octeto, resultando em 128 endereços possíveis. Para realizar o cálculo foi preciso pegar apenas o bits destinados à conexão e aplicar a fórmula demonstrada anteriormente: 2^7; que resulta em 128 endereços possíveis.

Então, a faixa de endereços disponíveis para essa rede é de 200.199.229.0 à 200.199.229.127, contabilizando os 128 endereços. Cabe ressaltar que o número de endereços possíveis em uma faixa de rede é diferente da quantidade de endereços que vão ser realmente utilizados, pois deve-se sempre retirar os dois endereços extremos. Trata-se do endereço de rede, que é o primeiro endereço e é utilizado para identificar uma rede específica na qual os dispositivos estão conectados; e o endereço de *Broadcast*, que é o último endereço da rede e é o endereço de difusão. Portanto, a fórmula que define os endereços a serem utilizados é 2^7-2, que resulta em 126 conexões (*host*) livres para utilização.

Agora, utilizando um endereço original de classe B, 192.168.10.0 com a máscara 255.255.254.0 (/23), onde os dois primeiros octetos com todos os bits são definidos como 1s (255), enquanto o terceiro octeto tem apenas o último bit definido como 0 (254), e o restante

como 0. Isso significa que a rede possui 23 bits (16 bits dos dois primeiros octetos e mais 7 do terceiro octeto) e, portanto, aplicando a fórmula 2^1*256 pode ter um total de 512 endereços IP para os *hosts*.

3 METODOLOGIA

Esse projeto é definido como uma pesquisa aplicada, voltado para o contexto do ensino de Redes de Computadores, com o objetivo de desenvolver uma aplicação web que seja possível realizar cálculos de IPs e obter informações relevantes relacionadas aos preceitos e conceitos de redes. O procedimento metodológico é bibliográfico, dado que o trabalho orientou-se a conceitos de Redes de Computadores. Assim, torna-se experimental, visto que se utilizam softwares e técnicas computacionais para o desenvolvimento da ferramenta proposta.

3.1 Materiais

As ferramentas utilizadas para o desenvolvimento da interface e do funcionamento da calculadora IPv4 estão mencionadas e descritas a seguir. Além disso, é apresentado o esboço do projeto, destacando as etapas envolvidas.

3.1.1 Figma

Lançada em 2016 por Dylan Field e Evan Wallace, Figma é uma ferramenta de design colaborativo, que é possível criar interfaces de usuário, protótipos interativos com colaboração em tempo real com outros membros. Oferece uma interface intuitiva e de fácil uso, permitindo que designers e desenvolvedores trabalhem juntos em um ambiente centralizado.

No desenvolvimento deste projeto, o Figma foi utilizado para a criação das imagens que compõem a interface gráfica da calculadora. Utilizado para projetar e aprimorar campos, botões e demais elementos visuais, além de ser utilizado para a formulação das telas do projeto e na organização do fluxo delas.

3.1.2 Javascript

O JavaScript (JS) é uma linguagem de programação interpretada, tipada e orientada a objetos amplamente utilizada no desenvolvimento web (FLANAGAN, 2011). Ela foi desenvolvida pela NetScape em parceria com a Sun Microsystems com o objetivo de adicionar interatividade às páginas HTML (Grillo e Fortes, 2008).

Na aplicação, a linguagem JavaScript foi utilizada para definir as regras lógicas necessárias para o funcionamento dos algoritmos que realizam os cálculos de endereçamento

IP.

3.1.3 Vue.js

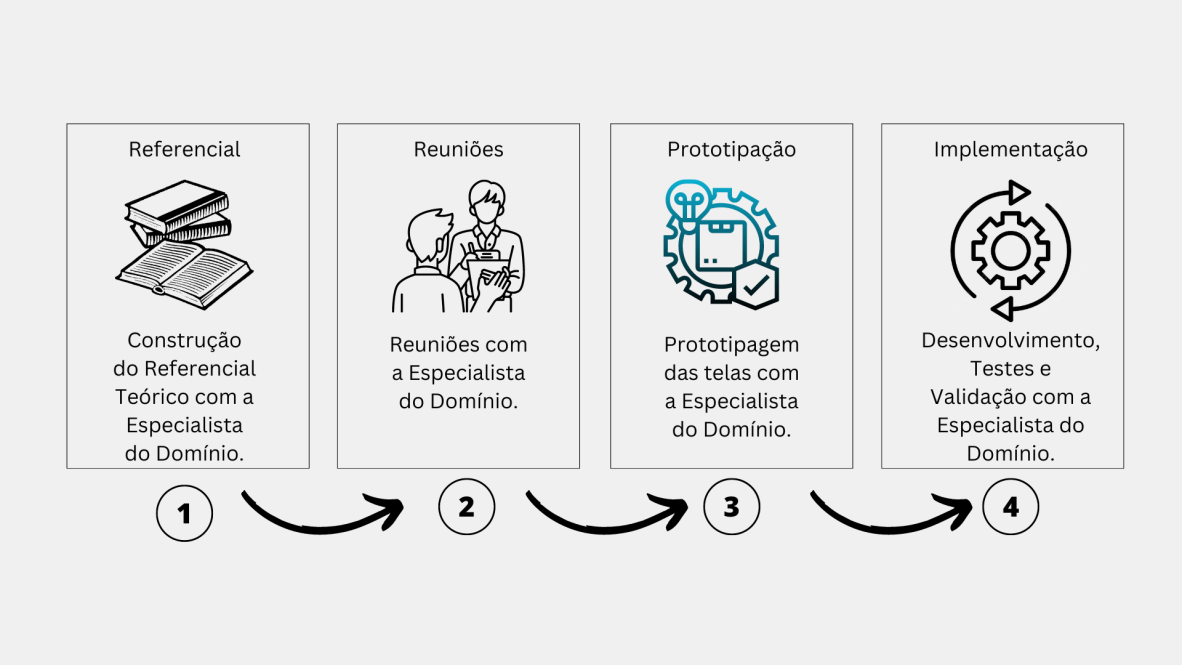
Segundo Adam (2018) o *Vue.js* é um “framework" de código aberto para desenvolvimento de aplicativos do lado do cliente. Ele é flexível e poderoso, e permite criar aplicativos web ricos e interativos com facilidade”. Portanto, o *Vue.js* é amplamente utilizado por possuir uma abordagem simples e intuitiva, além de uma curva de aprendizado suave para os desenvolvedores.

Nesse projeto o *Vue.js* foi utilizado para implementar os métodos de verificação e resolução dos cálculos de endereçamento IP, permitindo que a aplicação realizasse os cálculos de forma eficiente e precisa. Além disso, o *Vue.js* proporcionou uma estrutura flexível e modular para o desenvolvimento das funcionalidades da aplicação, facilitando a manutenção e expansão do código.

3.2 Métodos

Para a criação da aplicação web, a metodologia utilizada seguiu as etapas apresentadas na Figura 5, que foram executadas para o desenvolvimento da aplicação, cumprindo o objetivo do projeto.

Figura 5. Metodologia dos projeto



Fonte: Autor

A Figura 5 ilustra a metodologia adotada durante o processo de desenvolvimento, que resultou na conclusão do trabalho de acordo com os objetivos estabelecidos. Na primeira etapa do projeto, foi conduzido um estudo do referencial teórico com a especialista do domínio, que envolveu uma pesquisa bibliográfica abrangendo os conceitos fundamentais necessários para o desenvolvimento da aplicação. Foram explorados temas como conceitos, protocolos, regras e objetivos relacionados à proposta do projeto.

Na segunda e em todas as etapas do projeto, foram conduzidas entrevistas com um especialista do domínio, com o objetivo de obter informações relevantes e estabelecer os objetivos a serem alcançados no desenvolvimento da aplicação. Essas entrevistas foram realizadas para obter *insights* valiosos e garantir que a aplicação atendesse às necessidades e expectativas dos usuários finais.

Na prototipagem da aplicação, a ferramenta Figma foi utilizada para o processo de design da interface, definindo os tipos de entradas necessárias para que os usuários pudessem realizar os cálculos desejados. Nessa etapa, juntamente com a especialista foi elaborado o

layout e a estrutura visual da aplicação, levando em consideração a usabilidade e a experiência do usuário.

Na quarta etapa, como apresentado na figura 5, foi realizada a implementação da calculadora, utilizando as ferramentas *Javascript* e *Vue.js*. Conforme demonstrado, essa etapa abrangeu o desenvolvimento, os testes e a validação dos requisitos estabelecidos na terceira etapa pela especialista do domínio. Durante a fase de desenvolvimento, um conjunto de cálculos foi selecionado para ser desenvolvido, resultando em um protótipo que seria submetido a diversas etapas de testes, onde seria apresentado à especialista do domínio para validação e possíveis correções ou ajustes.

Durante os testes, foram verificados aspectos como entradas de dados, validação de endereços, resultados dos cálculos e mensagens de erro. Além disso, a especialista do domínio, realizou a verificação de aspectos como usabilidade da aplicação, verificação das informações de entrada e saída e a eficiência dos cálculos. Após a especialista do domínio realizar a aprovação de uma funcionalidade, outra era escolhida para ser desenvolvida, até que todos os requisitos determinados fossem concluídos, garantindo que a calculadora atende aos requisitos e expectativas estabelecidos.

4 RESULTADOS E DISCUSSÃO

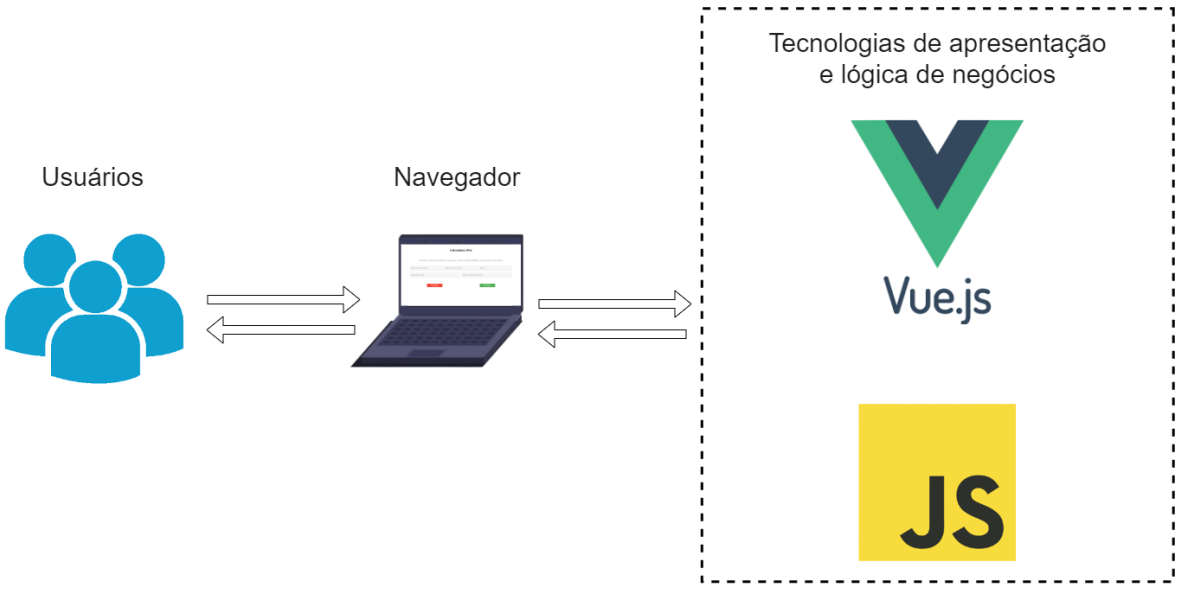
A Calculadora IP desenvolvida oferece uma variedade de cálculos úteis para o aprendizado e gerenciamento de endereçamento IP. Os cálculos disponíveis são: calcular rede, que é realizado a partir do endereços de *host* com a máscara ou a máscara com CIDR; calcula máscara, que é realizado com o endereço de rede com CIDR ou endereço de rede com *broadcast*; calcular *broadcast*, que é realizado através do endereço de *host* com a máscara ou endereço de rede com o CIDR; calcular endereço CIDR, que é possível através do endereço de rede com a máscara ou endereço de rede com o *broadcast*. Esses cálculos são realizados de maneira fácil e intuitiva, permitindo ao usuário inserir os valores necessários e obter os resultados desejados de forma rápida e precisa.

Nas sessões seguintes são apresentadas: a arquitetura do software; a prototipação da aplicação; a estrutura interna da aplicação; a interface de interação com o usuário; exemplos dos cálculos possíveis; e, apresentação de trechos relevantes da aplicação.

4.1 Arquitetura do Software

A seguir é apresentada a representação visual da estrutura da aplicação, destacando as ferramentas e tecnologias utilizadas, bem como o relacionamento entre elas. Essa imagem ilustra a arquitetura e os componentes fundamentais da calculadora, fornecendo uma visão geral do seu funcionamento.

Figura 6. Estrutura da Aplicação



Fonte: Autor

A figura 6 ilustra a arquitetura da aplicação, que segue uma abordagem de aplicativo de página única no *Vue.js*. A arquitetura baseada em aplicativos de página única (Single-Page Application - SPA) é altamente adequada para o desenvolvimento de interfaces dinâmicas e interativas. Nesta arquitetura, as tecnologias utilizadas são responsáveis por renderizar a interface do usuário e exibir os resultados obtidos pela lógica de negócio. O Vue.js não apenas controla toda a página, mas também lida com atualizações de dados em um único arquivo, proporcionando uma apresentação clara e intuitiva de todas as funcionalidades da calculadora para o usuário.

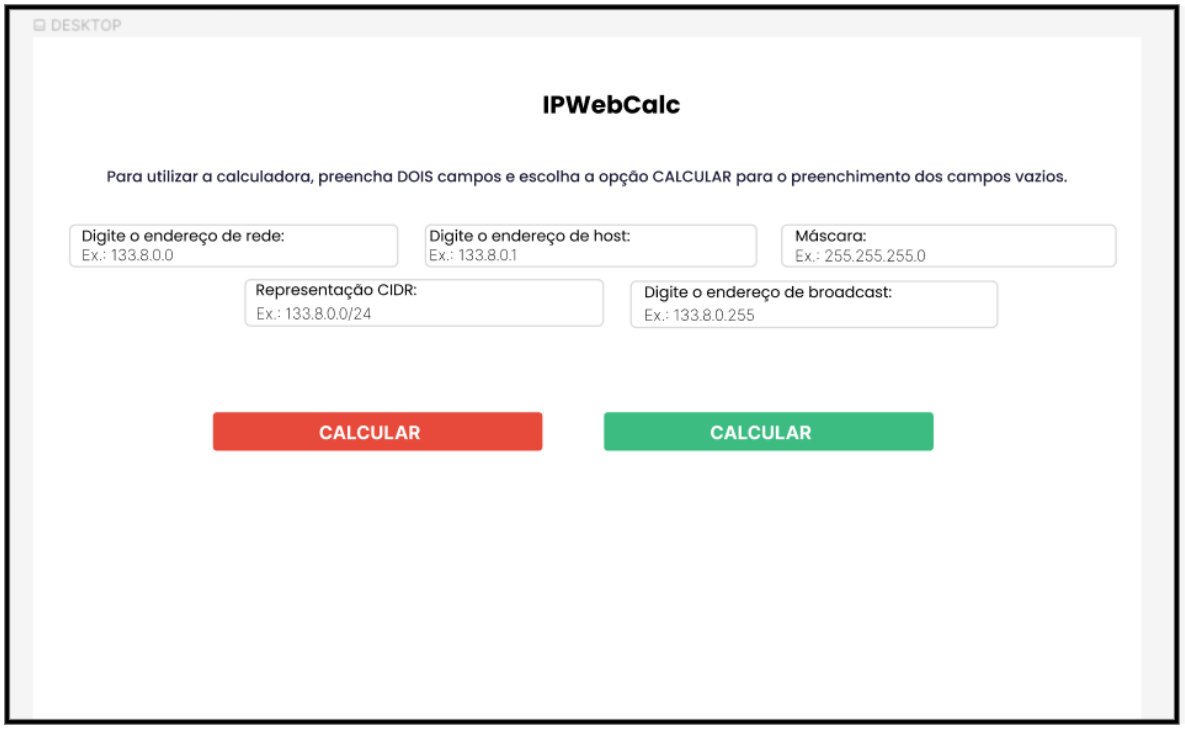
As tecnologias de lógica de negócio são utilizadas no processamento dos dados, onde são definidos os métodos, funções e componentes que realizam os cálculos e manipulam os dados inseridos pelo usuário. Nessa parte do desenvolvimento foi utilizada a linguagem de programação *JavaScript* juntamente com o *framework Vue.js*, para implementação das funções que realizam os cálculos, permitindo a obtenção de dados relevantes sobre endereçamento IP.

A abordagem em camada única só é possível graças à estrutura modular dos que o *Vue.js* fornece, que permite que a arquitetura da aplicação seja composta por componentes reutilizáveis que encapsulam tanto a apresentação quanto a lógica de negócio em um único arquivo com extensão "*.vue"*. Cada arquivo "*.vue"* possui um *template*, que define a estrutura e a aparência visual do elemento, e uma seção de *script*, onde são implementados os métodos, as propriedades e a lógica necessária para o funcionamento do componente, que poderá ser visualizada de uma melhor maneira no tópico a seguir.

4.2 Prototipação da Aplicação

A Figura 7 demonstra como foi realizada a prototipagem da aplicação, onde foi utilizada a ferramenta Figma para o processo de design da interface, definindo os tipos de entradas necessárias.

Figura 7. Protótipo da Interface

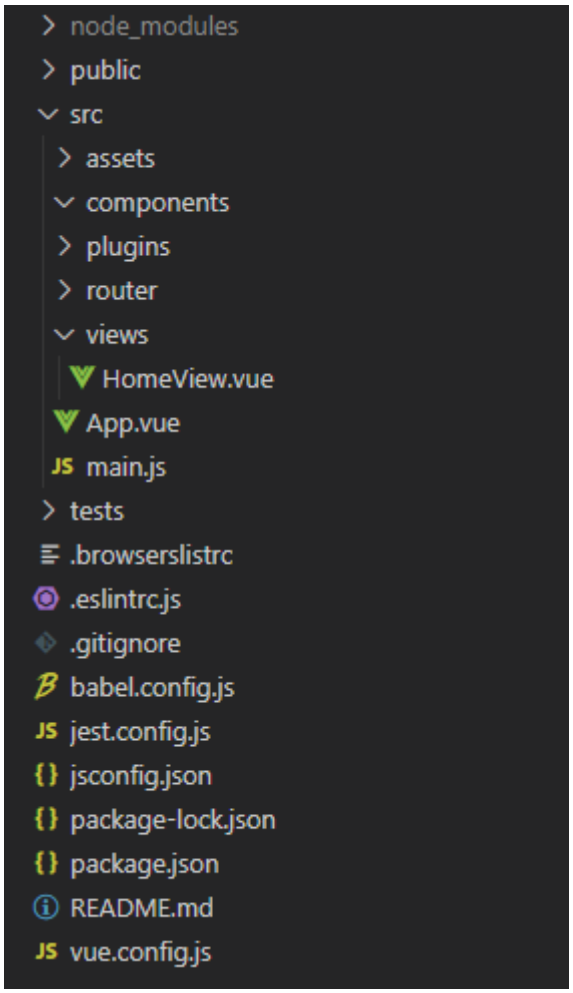


Para realizar a prototipagem da Figura 7, foi trabalhado em conjunto com a especialista a elaboração do layout e a estrutura visual da aplicação, levando em consideração a usabilidade e a experiência do usuário.

4.2 Estrutura Interna da Aplicação

A Figura 8 ilustra a estrutura interna da aplicação, seguindo o padrão definido pelo framework *Vue.js*. Essa estrutura é projetada para organizar e separar os diferentes elementos da calculadora de forma lógica e coerente. As pastas específicas desempenham papéis distintos na construção da calculadora, facilitando sua manutenção e escalabilidade.

Figura 8. Estrutura Interna da Aplicação



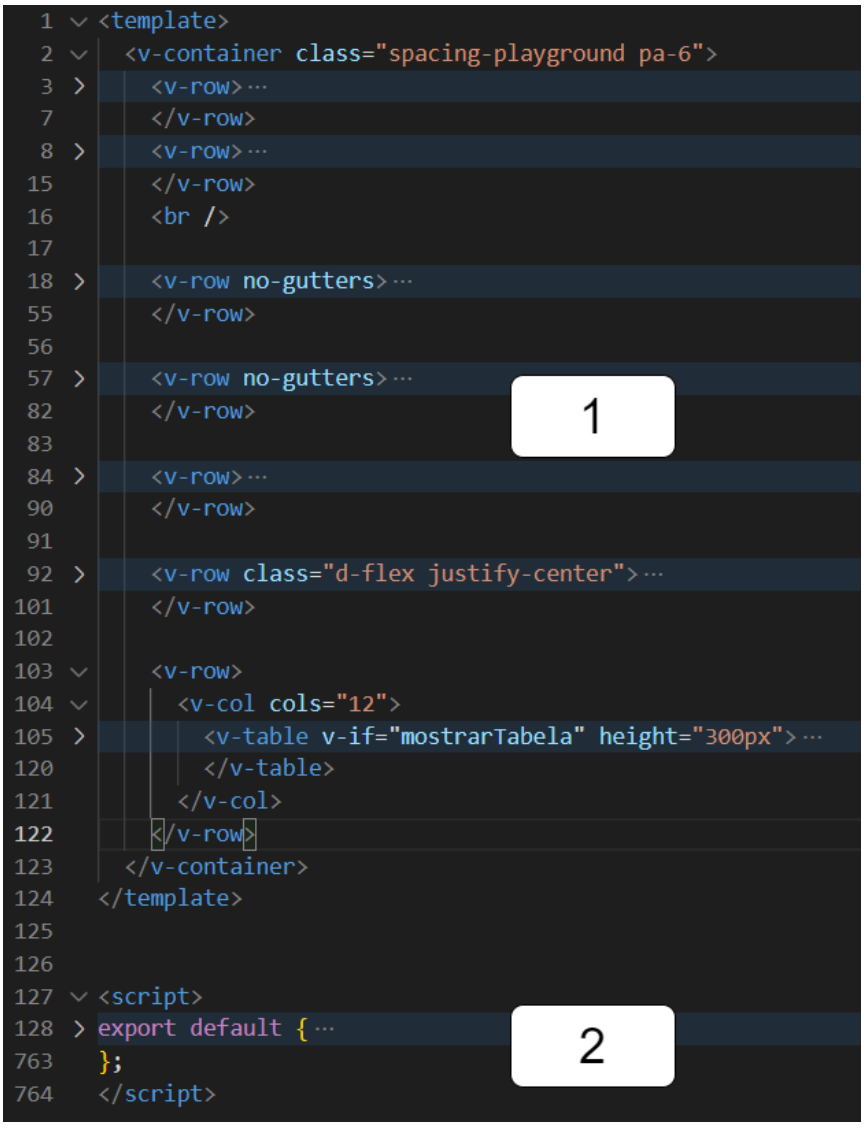
Conforme a Figura 8, a estrutura interna da aplicação é dividida em vários diretórios definidos previamente pelo *framework Vue.js*, onde o principal diretório utilizado é o `src` que contém os arquivos principais do projeto. Com a visualização da imagem, nota-se que a partir do `src` existe uma série de diretórios, onde cada um desempenha uma função na aplicação. A pasta `assets` é utilizada para armazenar recursos estáticos, como imagens, arquivos de estilo e fontes. Já a pasta `components` contém os componentes reutilizáveis da aplicação, organizados em subpastas de acordo com sua função.

A pasta `router` contém os arquivos de configuração do roteamento da aplicação, permitindo a navegação entre as diferentes páginas. No contexto da calculadora, a pasta utilizada foi a pasta `views`, que é responsável por abrigar os componentes principais de visualização da aplicação. Como o projeto consiste em apenas uma página, a pasta contém um

único arquivo chamado `HomeView.vue`, que contém todo o código da calculadora. Essa estrutura interna bem definida facilita a manutenção, a escalabilidade e a colaboração entre os membros da equipe de desenvolvimento.

A figura 9 demonstra a estrutura de como funciona o arquivo `HomeView.vue`. O arquivo é composto por duas tags principais que desempenham papéis distintos na construção da aplicação: a área indicada com o número 1 é a tag `template` e a área indicada com o número 2 é a tag `script`.

Figura 9. Estrutura do Arquivo

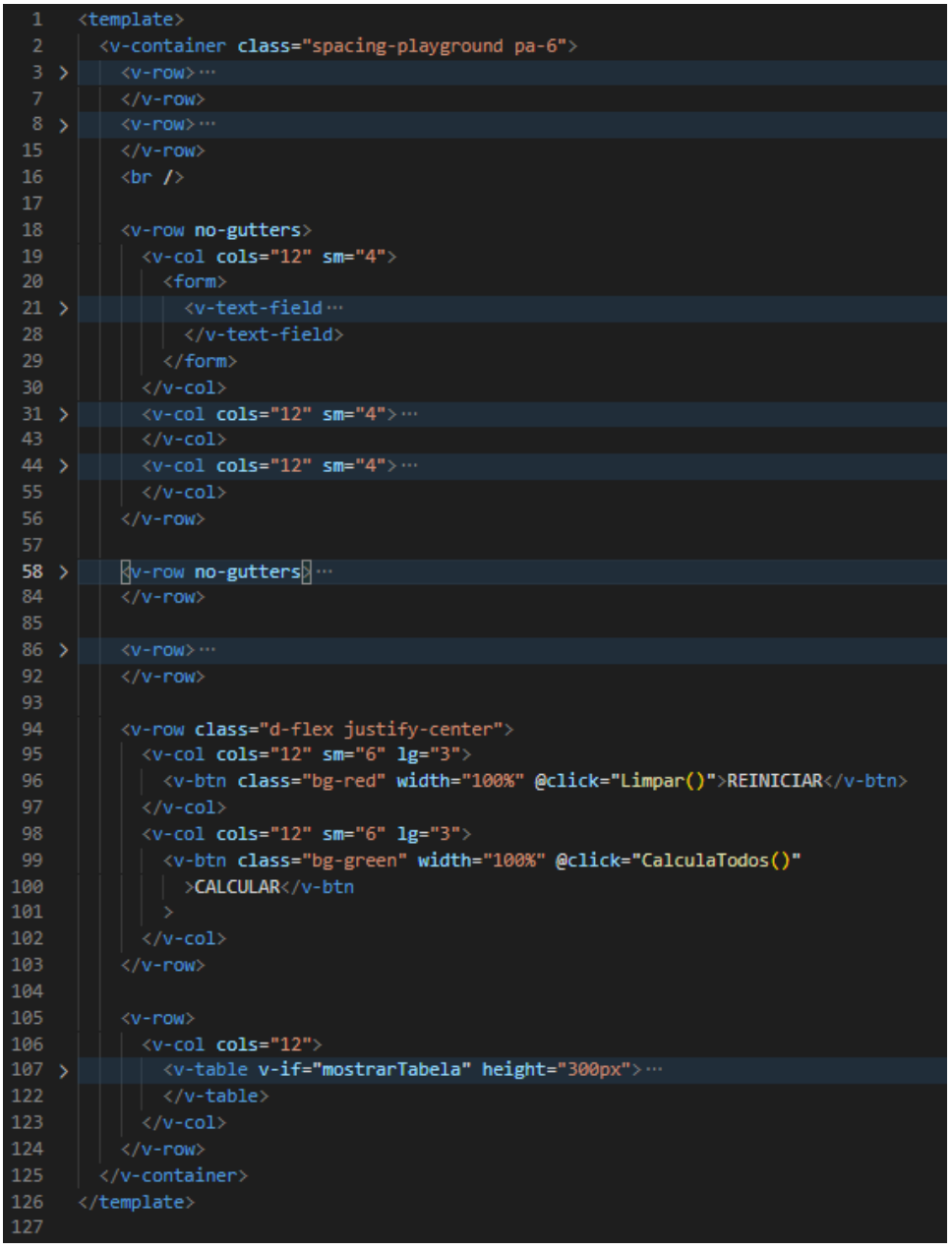


Na próxima seção é fornecida uma explicação mais detalhada sobre cada uma das tags presentes na figura 9, abordando suas funcionalidades e como elas contribuem para a criação da calculadora IP.

4.3 Trechos de Código da Aplicação

Como visto na seção anterior, o arquivo principal da calculadora é constituído por duas tags principais, a tag `template` e a tag `script`. A tag `template` encontrada no campo 1 contém a estrutura que define a parte visual da aplicação, ou seja, os elementos de interface, como campos de entrada, botões e elementos de exibição dos resultados, como pode ser observado na figura 10.

Figura 10. Estrutura do *Template*



```
<template>
  <v-container class="spacing-playground pa-6">
    <v-row>···
    </v-row>
    <v-row>···
    </v-row>
    <br />

    <v-row no-gutters>
      <v-col cols="12" sm="4">
        <form>
          <v-text-field···
          </v-text-field>
        </form>
      </v-col>
      <v-col cols="12" sm="4">···
      </v-col>
      <v-col cols="12" sm="4">···
      </v-col>
    </v-row>

    <v-row no-gutters>···
    </v-row>

    <v-row>···
    </v-row>

    <v-row class="d-flex justify-center">
      <v-col cols="12" sm="6" lg="3">
        <v-btn class="bg-red" width="100%" @click="Limpar()">REINICIAR</v-btn>
      </v-col>
      <v-col cols="12" sm="6" lg="3">
        <v-btn class="bg-green" width="100%" @click="CalculaTodos()"
          >CALCULAR</v-btn
          >
      </v-col>
    </v-row>

    <v-row>
      <v-col cols="12">
        <v-table v-if="mostrarTabela" height="300px">···
        </v-table>
      </v-col>
    </v-row>
  </v-container>
</template>
```

Na figura 10 é apresentada a estrutura do `template` utilizado na aplicação, onde foi aplicada uma padronização visual utilizando a biblioteca *Vuetify*. Essa biblioteca permitiu o *design* de toda a interface, incluindo a criação dos campos de entrada, o estilo dos botões e a exibição da tabela. A utilização do *Vuetify* possibilitou uma aparência visualmente intuitiva e consistente em toda a calculadora IP.

Com o uso da biblioteca *Vuetify* foi possível utilizar uma variedade de tags personalizadas para facilitar a criação da interface. Dentre as tags utilizadas estão o `v-row` e `v-col` apresentadas nas linhas 3 e 4 do código, que permitem organizar a estrutura da aplicação em colunas e linhas. Os campos de entrada e outros elementos da interface são criados utilizando as tags `v-text-field`, `v-btn` e `v-table` presentes nas linhas 21, 96 e 107 respectivamente. Cada um desses elementos possui atributos específicos, como o atributo `v-model`, que está vinculado a propriedades de dados do componente e permite a sincronização bidirecional dos valores entre a interface e a lógica de negócio.

Já a tag `script` na figura 9 no campo 2 é responsável por conter a lógica de negócio da aplicação. Nela são definidos os métodos, funções e variáveis necessários para realizar os cálculos e manipular os dados inseridos pelo usuário. Essa parte do código é escrita em *JavaScript* e segue a sintaxe e as convenções do *Vue.js*, como demonstra a figura 11 a seguir.

Figura 11. Estrutura do *Script*

Ao analisar a figura 11 apresentada, é visível que a linha 131 `name: "HomeView"` no início da tag `script` demonstra que a tag contém um código *JavaScript* para o componente chamado `HomeView`. O `export default` na linha 130, é usado para exportar esse componente como o valor padrão do módulo, permitindo que ele seja importado e usado em outros arquivos.

No componente, na linha 133, há uma propriedade chamada `data` que é uma função que retorna um objeto, esse objeto contém várias propriedades que são usadas para armazenar o estado da aplicação. No caso da calculadora, essas propriedades são utilizadas para controlar o estado dos campos de entrada e outros elementos da interface.

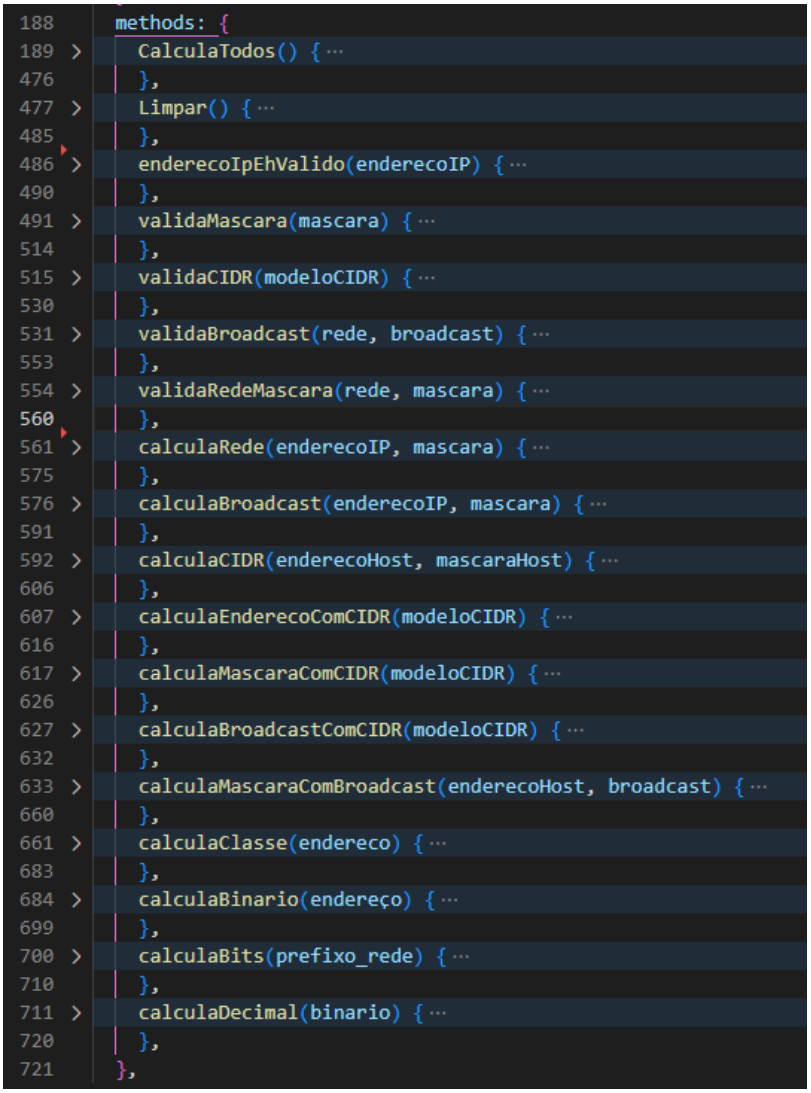
A variável `mostrarTabela` que é do tipo booleana, indica se a tabela de resultados deve ser exibida ou não. Seu valor inicial é definido como "*false*", o que significa que a tabela

não é exibida inicialmente. As variáveis `endereco_ip_rede`, `endereco_ip`, `mascara`, `modelo_CIDR` e `endereco_broadcast` nas linhas 135 a 141 são declaradas vazias e do tipo *strings* para armazenar os valores inseridos pelos usuários nos campos de entrada, e são atualizadas conforme o usuário interage com a calculadora.

Da mesma forma que `mostrarTabela`, a variável `exibirMensagem` também é do tipo booleana e indica se uma mensagem de erro deve ser exibida. Seu valor inicial é definido como `false`, o que significa que nenhuma mensagem é exibida inicialmente. Essa propriedade é utilizada para informar ao usuário caso ele tente efetuar o cálculo com dados incompatíveis ou inválidos. Essas propriedades são atualizadas e manipuladas por métodos dentro do componente para refletir as interações do usuário e atualizar o estado da aplicação.

Os métodos que realizam a manipulação dos dados são contidos na propriedade `methods`, onde é verificado a validade dos dados inseridos pelo usuário e é realizado os cálculos. A estrutura dessa propriedade pode ser visualizada de uma melhor forma a partir da figura 12 a seguir.

Figura 12. Estrutura do *Methods*

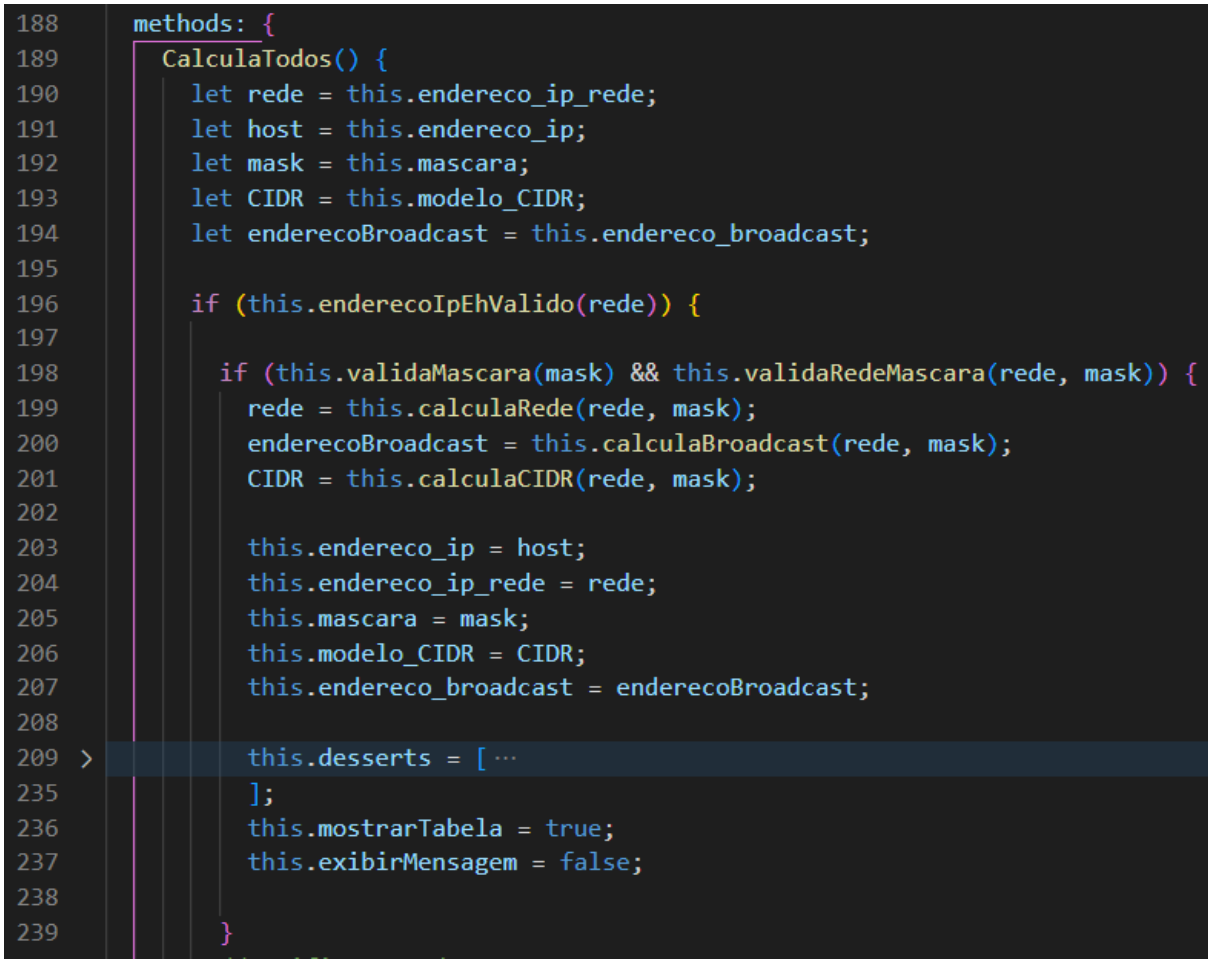


Conforme a ilustração da figura 12, a aplicação foi dividida em uma série de funções, onde cada uma efetua um papel crucial para o funcionamento da aplicação. As funções são separadas de acordo com seu papel, onde pode ser notado que a função principal do programa é a `CalculaTodos()` na linha 189, que tem como objetivo coordenar a chamada das outras funções de acordo com as entradas fornecidas pelo usuário.

Essa função é acionada quando o usuário clica no botão CALCULAR, para que sejam realizados os cálculos referente aos valores relacionados aos endereços IP e informações de rede. Primeiramente, ela verifica se os endereços fornecidos são válidos, garantindo que

estejam de acordo com as regras estabelecidas, conforme demonstra o pequeno trecho de código na figura 13.

Figura 13. Trecho da Função `CalculaTodos()`



Conforme ilustrado na figura 13, após a validação dos endereços, a função prossegue para realizar os cálculos a fim de obter os valores do endereço de rede, endereço de *host*, máscara, endereço de *broadcast* e classe do IP. Após a conclusão desses cálculos, os resultados são atualizados nos campos da interface do usuário. Além disso, uma tabela exibindo os valores em formato binário é apresentada na interface, permitindo uma visualização clara e organizada das informações.

Por outro lado, a função `Limpar()` na linha 477 presente na figura 12, tem como finalidade reiniciar os cálculos, garantindo que todos os campos sejam limpos e que as variáveis retornem aos seus valores iniciais. Ao executar a função `Limpar()`, todos os valores dos campos da interface são apagados e as variáveis são restauradas aos seus valores padrão, preparando o ambiente para uma nova utilização do usuário. Essa funcionalidade contribui para um melhor uso do usuário, permitindo que ele comece novamente do zero, com uma calculadora pronta para receber novos dados e fornecer novos resultados.

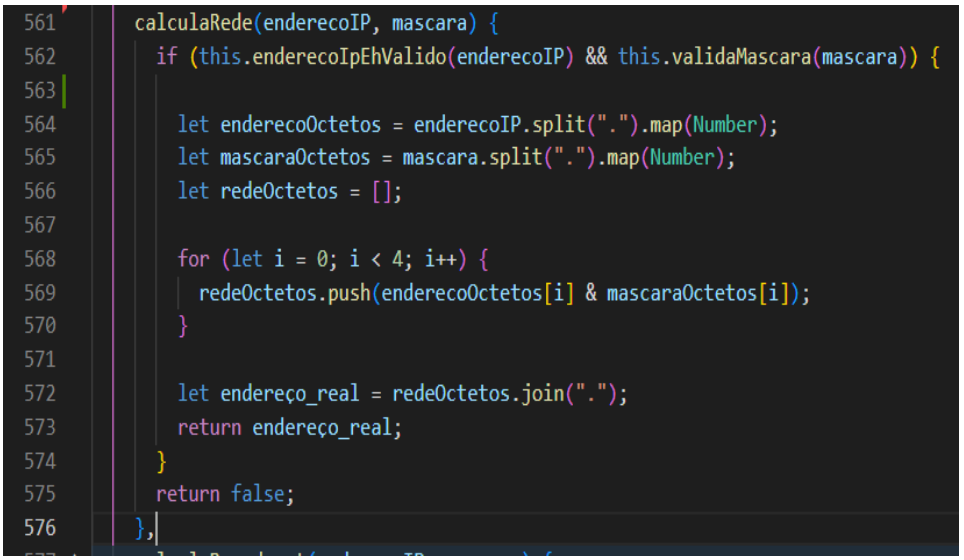
Já o conjunto de funcionalidades `enderecoIpEhValido()`, `validaMascara()`, `validaCIDR()`, `validaBroadcast()`, `validaRedeMascara()` presentes nas linhas 486 a 560 na figura 12, são responsáveis pela validação dos endereços fornecidos pelo usuário. Essas funcionalidades garantem que os dados estejam corretos e em conformidade com as regras de formatação e padrões de rede. Logo, a função `enderecoIpEhValido()` na linha 486 representada na figura 12, verifica se o endereço IP inserido pelo usuário é válido, garantindo que esteja no formato correto e dentro dos limites aceitáveis.

Na linha 491, a função `validaMascara()` é responsável por validar a máscara de sub-rede fornecida, verificando se está no formato adequado e dentro do intervalo permitido. A função `validaCIDR()` contida na linha 515, válida a notação CIDR (*Classless Inter-Domain Routing*), garantindo que o valor fornecido esteja dentro dos parâmetros permitidos. A `validaBroadcast()` na linha 531 verifica se o endereço de *broadcast* fornecido pelo usuário é válido, assegurando que esteja dentro dos limites corretos e seguindo as regras de cálculo de *broadcast*. Por fim, a função `validaRedeMascara()` na linha 556 válida se o endereço de rede e a máscara estão em conformidade, verificando se correspondem adequadamente.

Por fim, a seção de código entre as linhas 561 e 720 contém as funcionalidades: `calculaRede()`, `calculaBroadcast()`, `calculaCIDR()`, `calculaEnderecoComCIDR()`, `calculaMascaraComCIDR()`, `calculaBroadcastComCIDR()`, `calculaMascaraComBroadcast()`, `calculaClasse()`, `calculaBinario()` e `calculaBits()`, `calculaDecimal`. Essas funcionalidades realizam diferentes cálculos com base nas chamadas feitas pela função `CalculaTodos()` e nos dados fornecidos pelo usuário.

Por exemplo, a primeira função contida na linha 561 chamada `calculaRede()`, ela é responsável por realizar o cálculo que determina o valor do endereço de rede. Essa função recebe como parâmetros o endereço de *host* e a máscara de sub-rede. Utilizando esses dados, o cálculo é realizado e o resultado correspondente ao endereço de rede é retornado, como apresentado na figura 14.

Figura 14. Estrutura da Função `calculaRede()`



Na figura 14 é possível observar que a estrutura inicia na linha 562 com uma condição `if` que verifica se o endereço de *host* e a máscara são endereços válidos. Em seguida, os métodos `.split(".")` e `.map(Number)` são utilizados para dividir as strings dos endereços em partes e converter essas partes em números inteiros, atribuindo os resultados às variáveis `enderecoOctetos` e `mascaraOctetos`.

Na linha 568, um *loop* `for` é utilizado para iterar sobre cada octeto do endereço IP e da máscara de rede, realizando a operação *bitwise And* "&" entre o octeto correspondente do endereço IP e o octeto correspondente da máscara de rede. O operador "&" realiza uma operação lógica bit a bit entre os dois valores, resultando em um novo valor cujos bits serão "1" apenas se ambos os bits comparados forem "1". O resultado dessa operação é adicionado à lista `redeOctetos` utilizando o método `.push()`. Após sair do loop, a variável `endereço_real` recebe a lista `redeOctetos` convertida em uma string pelo método `.join(".")`. Caso os endereços inseridos no início da estrutura sejam inválidos, a função

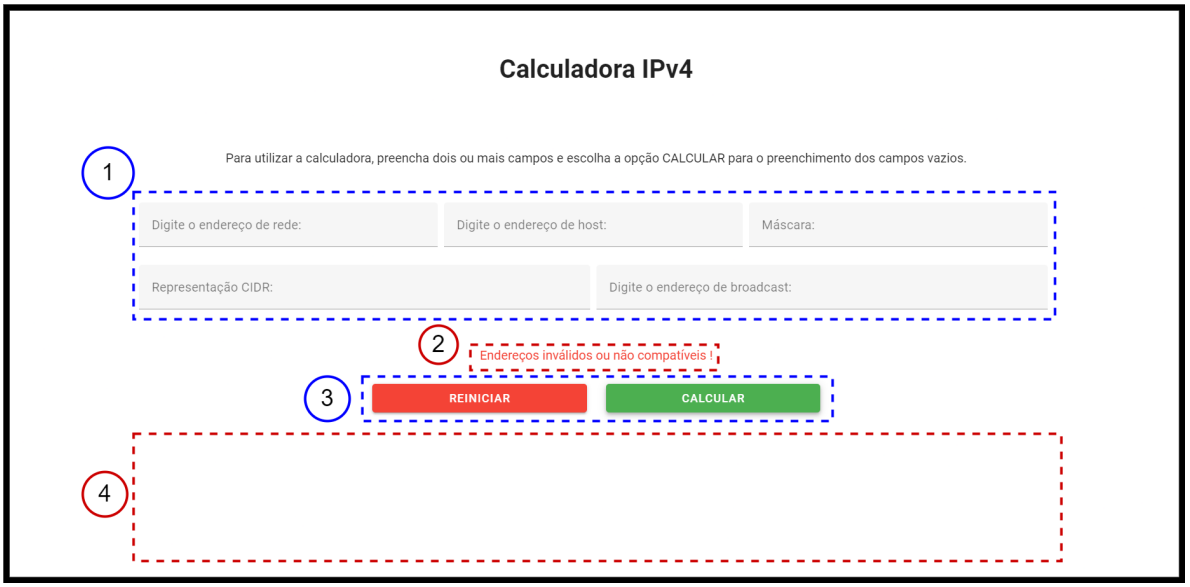
retorna *false*, interrompendo a execução e finalizando a chamada.

Em resumo, o conjunto de funcionalidades apresentadas nesta seção, são responsáveis por realizar todo o trabalho lógico da calculadora, efetuando validações e cálculos para fornecer dados corretos e concisos ao usuário. Na seção a seguir será apresentada a interface que o usuário terá acesso, onde serão apresentados os resultados das seções mencionadas anteriormente.

4.4 Interface da Aplicação

A interface visual da Calculadora IP adota os parâmetros de design estabelecidos durante a fase de prototipagem. A Figura 15 exibe a tela inicial da calculadora, representando o estado inicial antes de qualquer interação do usuário.

Figura 15. Tela Inicial



Na figura 15 é apresentada a tela inicial da calculadora, onde foram enumeradas os componentes da interface:

1. Campos de Entrada: nesta área, estão localizados os campos de entrada nos quais o usuário pode inserir os dados necessários para realizar os cálculos. Esses campos

devem ser preenchidos corretamente com as informações requeridas. A funcionalidade de preencher diferentes endereços permite que, por exemplo, o usuário forneça um endereço IP de *host* e, com base na máscara de rede ou em outro endereço fornecido, a calculadora determina o endereço de rede correspondente e outras informações sobre aquele endereço;

2. Área reservada para mensagem de erro: esta área é específica para exibir mensagens de erro. Quando o usuário insere dados que não estão de acordo com os requisitos necessários para realizar o cálculo, é exibida a mensagem "Endereços inválidos ou não compatíveis!". Essa mensagem serve para alertar o usuário sobre a necessidade de verificar e corrigir os dados inseridos antes de prosseguir com o cálculo;
3. Área reservada aos botões: área em que encontram-se os botões de ação da calculadora. O botão "CALCULAR" é acionado para realizar os cálculos com base nos dados inseridos nos campos de entrada e exibir os resultados na tabela. O botão "REINICIAR" permite limpar todos os campos preenchidos, reiniciando assim a calculadora;
4. Área destinada à tabela: é exibida uma tabela que apresenta os resultados dos cálculos realizados em formato binário. Os dados calculados são organizados de forma clara e legível para facilitar a visualização e compreensão por parte do usuário.

Nas próximas seções, são apresentados exemplos de cálculos e será fornecida uma descrição do seu funcionamento. Isso permitirá observar como a interface se comporta em diferentes situações e como os cálculos são realizados. Ao longo dessas demonstrações, serão exploradas as funcionalidades da calculadora e suas respostas correspondentes, fornecendo uma visão abrangente do seu desempenho e eficácia.

4.5 Demonstrações de Cálculos

Nesta seção, são apresentadas demonstrações da utilização da calculadora, explorando suas funcionalidades. Para iniciar um cálculo, o usuário precisa preencher no mínimo dois campos com os dados que possui, exceto no caso do endereço CIDR, o qual possui informações suficientes para cálculos a partir dele. Os cálculos disponíveis, apresentados nas próximas seções, são: cálculo de endereços de rede, CIDR e *Broadcast* a partir do *host e* da máscara; cálculo de endereços de rede, máscara e *broadcast*, a partir do endereço CIDR; cálculo do CIDR e *broadcast,* a partir do endereço de rede com a máscara; cálculo do CIDR e máscara, a partir do endereço de rede com *broadcast*.

4.5.1 Cálculo de endereços de rede, CIDR e *Broadcast*

O cálculo de endereços de rede, CIDR e *broadcast* é realizado quando o usuário digita a combinação do host com a máscara. A figura 16 mostra o resultado do cálculo indicado.

Figura 16 Exemplo Cálculo com Host e Máscara

Na situação apresentada na figura 16, o usuário preencheu o endereço de *host* “192.168.250.45” e máscara “255.255.0.0”. Após o usuário selecionar o botão “CALCULAR”, foi realizado o cálculo e a apresentação dos endereços de rede, CIDR e *broadcast*, conforme pode ser verificado na imagem. Além disso, a calculadora apresentou uma tabela contendo todos os endereços calculados no formato decimal e binário, permitindo ao usuário visualizar e compreender melhor a estrutura da rede.

No entanto, caso o usuário preencha um dos endereços de forma errada como, por exemplo, uma máscara de rede inválida, com valor fora do intervalo permitido, a calculadora irá apresentar uma mensagem de erro informando que os endereços são inválidos ou não compatíveis, como pode ser demonstrado na figura 17.

Figura 17. Máscara Inválida

Na situação do exemplo da figura 17, o usuário digitou "255.255.256.0" como máscara de rede, onde é notado que o endereço está digitado de forma incorreta, pois em uma máscara válida seus octetos devem seguir uma sequência contínua de 1 seguida de 0, onde em cada octeto o número máximo é 255. No exemplo, a calculadora retorna uma mensagem de erro na tela “Endereços inválidos ou não compatíveis”, impossibilitando a realização do cálculo.

4.5.2 Cálculo de endereços de rede, máscara e *Broadcast*

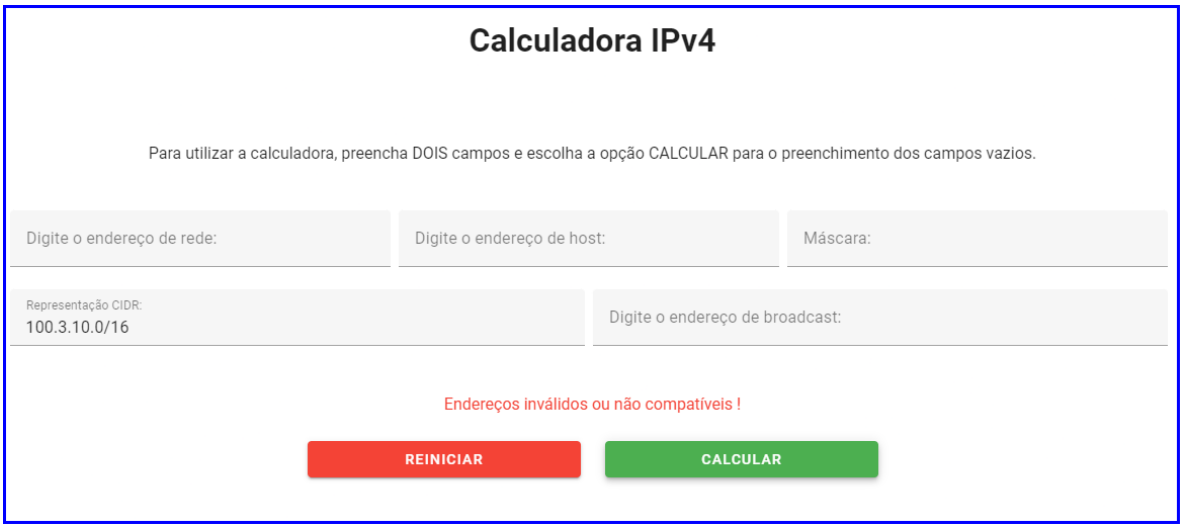
O cálculo de endereços de rede, máscara e *broadcast* pode ser realizado quando o usuário digita somente o endereço CIDR. A figura 18 mostra o resultado do cálculo indicado.

Figura 18. Exemplo de Cálculo com Endereço CIDR

No exemplo apresentado na figura 18, o usuário preencheu o endereço de CIDR "10.0.0.0/16". Após o usuário selecionar o botão "CALCULAR", foi realizado o cálculo e a apresentação dos endereços de rede, máscara e *broadcast* conforme pode ser verificado na imagem. Além disso, a calculadora exibe uma tabela com todos os endereços calculados no formato decimal e binário. É importante ressaltar que o endereço de host não foi apresentado nessa tabela, uma vez que o *host* só pode ser disponibilizado pelo usuário.

No entanto, como abordado na seção anterior, caso o usuário preencha um dos endereços de forma incorreta como, por exemplo, o endereço CIDR, a calculadora irá apresentar uma mensagem de erro informando que os endereços são inválidos ou não compatíveis, como pode ser demonstrado na figura 19.

Figura 19. Endereço CIDR Inválido



Na figura 19, é evidente que o usuário digitou "100.3.10.0/16" como endereço CIDR e, ao selecionar o botão "CALCULAR", foi notificado de que os endereços digitados são inválidos ou incompatíveis. Essa situação ocorre porque as partes que compõem o endereço CIDR não são compatíveis. No exemplo mencionado, a parte referente ao endereço de rede "100.3.10.0" foi fornecido, mas o endereço CIDR especifica uma máscara diferente, "/16", portanto, as duas partes não correspondem. A calculadora identifica que o endereço é inválido ou incompatível e exibe uma mensagem de erro ao usuário, impossibilitando a realização de cálculos.

4.5.3 Cálculo de endereços de CIDR e *Broadcast*

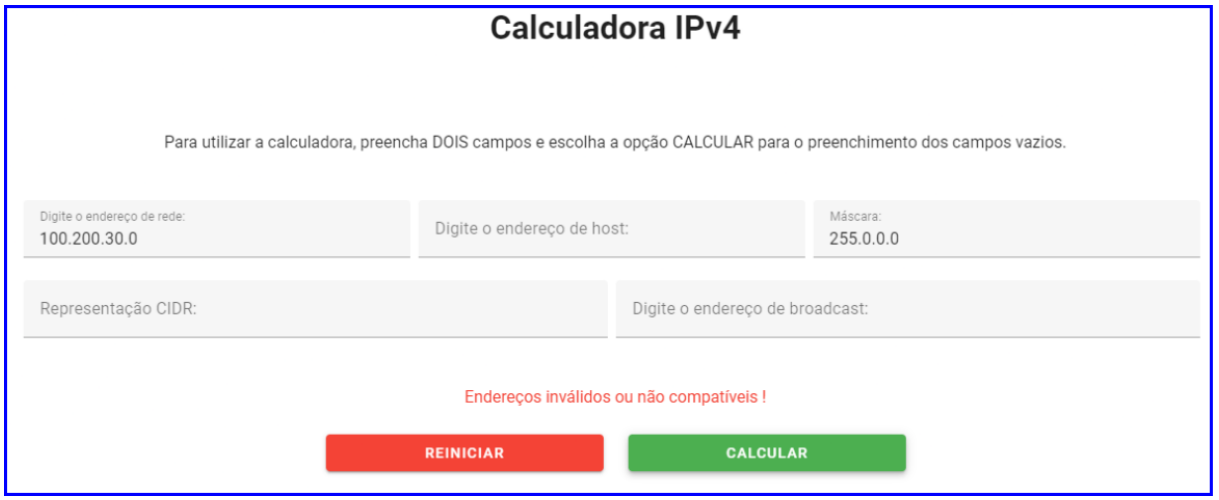
O cálculo de endereços CIDR e *broadcast* pode ser realizado quando o usuário digita a combinação do endereço de rede com a máscara. A figura 20 mostra o resultado do cálculo indicado.

Figura 20. Exemplo Cálculo com Rede e Máscara

No exemplo apresentado na figura 20, o usuário preencheu o endereço de rede "92.120.0.0" e a máscara "255.255.0.0". Após selecionar o botão "CALCULAR", a calculadora realizou o cálculo e apresentou os endereços CIDR e *broadcast*. Além disso, exibiu uma tabela com todos os endereços possíveis dentro da rede, no formato decimal e binário. Da mesma forma que a seção anterior, o endereço de host não foi apresentado nessa tabela, uma vez que o *host* só pode ser disponibilizado pelo usuário.

Como abordado nas seções anteriores, quando o usuário preenche um dos endereços de forma incorreta como, por exemplo, o endereço rede, a calculadora irá apresentar uma mensagem de erro informando que endereços são inválidos ou não compatíveis, como pode ser demonstrado na figura 21.

Figura 21. Endereço de Rede Inválido



No exemplo da figura 21, é ilustrado que o usuário digitou "100.200.30.0" como endereço de rede e como máscara “255.0.0.0” e, ao selecionar o botão "CALCULAR", foi notificado de que os endereços são inválidos ou incompatíveis. Essa situação ocorre porque o endereço de rede não corresponde com a máscara fornecida. No exemplo mencionado, a parte referente ao endereço de rede "100.200.30.0" foi fornecido, mas a máscara indica uma rede diferente, "100.0.0.0", portanto, as duas partes não correspondem. A calculadora identifica que os endereços são inválido ou incompatíveis e exibe uma mensagem de erro ao usuário e impossibilitando a realização de cálculos.

4.5.4 Cálculo de endereços CIDR e Máscara

O cálculo de endereços CIDR e máscara é realizado quando o usuário digita a combinação do endereço de rede com o *broadcast*. A figura 22 mostra o resultado do cálculo indicado.

Figura 22. Exemplo de Cálculo com Rede e Broadcast

No exemplo apresentado na figura 22, o usuário preencheu o endereço de rede "120.230.0.0" e o *broadcast* "255.255.0.0". Após selecionar o botão "CALCULAR", a calculadora realizou o cálculo e apresentou o endereço CIDR e a máscara. Além disso, exibiu uma tabela com todos os endereços possíveis dentro da rede, no formato decimal e binário. De acordo com os outros exemplos demonstrados anteriormente, o endereço de host não foi apresentado nessa tabela, uma vez que o *host* só pode ser disponibilizado pelo usuário.

Assim como nos exemplos anteriores, é crucial que o usuário forneça os endereços corretamente para realizar o cálculo de forma precisa. Caso o usuário digite algum endereço de forma incorreta, a calculadora emitirá uma mensagem de erro informando que os endereços são inválidos ou incompatíveis. Essa verificação é importante para garantir a correta execução dos cálculos. Como ilustrado na figura 21 da seção anterior, a calculadora identifica o erro e notifica o usuário sobre a inconsistência nos endereços fornecidos.

Os exemplos apresentados nas seções anteriores tinham como objetivo ilustrar as funcionalidades e cálculos disponíveis na calculadora de IP. Foram demonstrados diversos

cálculos com diferentes combinações de endereços, destacando a importância das validações realizadas pela calculadora para garantir resultados precisos e confiáveis. Os exemplos serviram para demonstrar a capacidade da calculadora em lidar com uma variedade de situações e auxiliar o usuário no cálculo de informações relacionadas a endereços IP.

5 CONSIDERAÇÕES FINAIS

O objetivo principal deste projeto consistiu no desenvolvimento de uma calculadora IP, capaz de realizar uma série de cálculos relacionados ao endereçamento de redes. Essa calculadora foi criada com o propósito de oferecer recursos que podem auxiliar tanto estudantes como profissionais da área de redes em suas atividades diárias.

Nesta calculadora são ofertados alguns cálculos como, cálculo de rede, *broadcast*, CIDR e máscara. Esses cálculos servem de recursos para os alunos testarem seu entendimento sobre os conteúdos explicados em sala de aula, contribuindo para o aprendizado e aprimoramento dos acadêmicos.

Durante o processo de desenvolvimento, foram encontradas algumas dificuldades, como, por exemplo, a necessidade de validar os endereços IP fornecidos pelo usuário. Essa validação foi uma etapa importante para garantir a precisão dos resultados obtidos pela calculadora. Para superar esse desafio, foi utilizado expressões regulares (regex) para validar os endereços IP inseridos.

A utilização do regex permitiu a definição de padrões específicos para verificar se os endereços digitados pelo usuário estavam de acordo com a estrutura correta de um endereço IP. Isso incluiu a verificação do número de octetos, a faixa de valores permitidos para cada octeto e a presença dos separadores corretos entre os octetos. Dessa forma, foi possível evitar erros e garantir a precisão dos cálculos realizados pela ferramenta.

Dessa forma, a calculadora é disponibilizada para o público, e pode ser acessada através do link https://calculadora-ip-navy.vercel.app/#/, com base nos resultados obtidos, o objetivo deste trabalho foi alcançado, a criação de uma calculadora que oferece uma experiência amigável e intuitiva e que realiza cálculos a partir de diferentes entradas do usuário. Uma ferramenta que pode auxiliar os professores a ensinar de uma forma dinâmica em sala de aula, realizando atividades de reforço com o uso da calculadora.

Como trabalhos futuros, sugere-se a implementação de novas funcionalidades, como o cálculo de sub-redes. Além disso, pode-se considerar a integração da calculadora IP com outras ferramentas e módulos relacionados a área de redes de computadores, visando fornecer uma solução mais completa e abrangente para os usuários.

REFERÊNCIAS

BIRKNER, Mattheus H.. **Projeto de Interconexão de Redes**. São Paulo: Pearson Education do Brasil, 2003.

BRAGA, Fagner Geraldes. **A CONTINUIDADE DA INTERNET PASSA PELO IPv6**. 2011. 53 f. TCC (Graduação) - Curso de Tecnólogo em Processamento de Dados, Faculdade de Tecnologia do Estado de São Paulo, São Paulo, 2011.

FRANCISCATTO, Roberto; CRISTO, Fernando de; PERLIN, Tiago. **Redes de Computadores**. Frederico Westphalen: Colégio Agrícola de Frederico Westphalen, 2014.

FREEMAN, Adam. Understanding Vue.js. 1 Jan. 2018, pp. 29–36, https://doi.org/10.1007/978-1-4842-3805-9_2.

FLANAGAN, David. JavaScript: the definitive guide. O`Reilly Media, Inc.,1005 Gravenstein Highway North, Sebastopol. 2011.

FREITAS JUNIOR, Vanderlei et al. Tecnologias e Redes de Computadores: estudos aplicados. Campus Avançado Sombrio: Vanderlei Freitas Junior, 2015.

GRILLO, Filipe D. N. FORTES, Renata P. D. M. Aprendendo JavaScript. São Carlos, 2008.

IBRAHIM, Issam. **Conjunto de Protocolos TCP/IP e suas Falhas**. 2011. 34 f. Monografia (Especialização) - Curso de Especialização em Teleinformática e Redes de Computadores,

Departamento de Informática, Universidade Tecnológica Federal do Paraná, Curitiba, 2011. Cap. 1.

KAMIS, Arnold; TOPI, Heikki. Network Subnetting: an instance of technical problem solving in kolb's experiential learning cycle. **2007 40Th Annual Hawaii International Conference On System Sciences (Hicss'07)**, [S.L.], p. 196-206, jan. 2007. IEEE. http://dx.doi.org/10.1109/hicss.2007.399

KUROSE, James F.; ROSS, Keith W.. **Redes de Computadores e a Internet**: uma abordagem top-dow. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

KUROSE, James F.; ROSS, Keith W.. **Redes de Computadores e a Internet**: uma abordagem top-dow. 5. ed. São Paulo: Pearson Education do Brasil, 2010.

KUROSE, James F.; ROSS, Keith W.. **Redes de Computadores e a Internet**: uma abordagem top-dow. 3. ed. São Paulo: Pearson Education do Brasil, 2005.

KUROSE, James F.; ROSS, Keith W.. **Redes de computadores e a Internet : uma nova abordagem**. São Paulo: Pearson, 2002. 570 p.

KUROSE, James F.; ROSS, Keith W.. **Redes de Computadores e a Internet**: uma abordagem top-dow. 8. ed. São Paulo: Pearson Education do Brasil, 2021.

LIMA JUNIOR, Luiz Carlos Chaves. PLANEJAMENTO DE REDE DE COMUNICAÇÃO DE DADOS PARA TRANSIÇÃO DE ENDEREÇAMENTO DE IPV4 PARA IPV6. 2017. 63

f. TCC (Graduação) - Curso de Engenharia de Computação, Universidade Estadual do Maranhão, São Luís, 2017.

MUNIZ, Angelo Henrique Alves *et al*. ESTUDO DE CASO TRANSIÇÃO DO PROTOCOLO IPv4 PARA IPv6. **Revista Gestão em Foco**, [*s. l*], v. 9, p. 529-545, 2017.

NETO, Celso Cardoso. A ARQUITETURA TCP/IP E A CONFIGURAÇÃO DAS REDES DE COMPUTADORES POR MEIO DAS MÁSCARAS DE SUB-REDE-TCP/IP ARCHITECTURE AND COMPUTER NETWORK DESIGN. **CADERNOS DE ESTUDOS E PESQUISAS-JOURNAL OF STUDIES AND RESEARCH**, 2010.

OLIVEIRA, Marcelo Hely da Silva. **ALTERNATIVAS PARA A IMPLEMENTAÇÃO DE REDES TCP/IP SOBRE INTERFACES AÉREAS**. 2002. 77 f. Dissertação (Mestrado) - Curso de Mestre em Ciência da Computação, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2002.

PAMPLONA, Edno Gustavo. **TRANSIÇÃO IPV4/IPV6**: técnica de tunelamento. 2014. 42 f. TCC (Graduação) - Curso de Tecnologia em Sistemas de Telecomunicações, Departamento Acadêmico de Eletrônica – Daeln – da Universidade Tecnológica Federal do Paraná – Utfpr, Universidade Tecnológica Federal do Paraná, Curitiba, 2014.

PENA, Salomão Bento Nilo. Guia Prático de Redes de Computadores. Angola: Instituto Superior de Ciências de Educação da Huíla, 2018. 128 p.

RIBEIRO, Thatiane Cristina dos Santos de Carvalho. Fundamentos de redes de computadores. Londrina: Editora e Distribuidora Educacional S.A, 2016.

SILVA, Cassiana Fagundes da. **Arquitetura e Práticas TCP/IP 1 e 2**. Curitiba: Contentus, 2021.

SILVEIRA, Carlos Kelner. Um Esquema para o Gerenciamento do Tráfego de Aplicações em Redes TCP-IP. 1995. 104 f. Dissertação (Mestrado) - Curso de Mestre em Ciência da Computação., Instituto de Matemática, Estatística e Ciência da Computação, Unicamp,, Campinas, 1995.

SMETANA, George Marcel MA. IPv4 e IPv6. 2012.

TANENBAUM, Andrew S.. David Wetherall. **Redes de Computadores.** 5. ed. São Palmas: Pearson Education do Brasil, 2011.

TANENBAUM, Andrew et al. Redes de Computadores. 6. ed. Porto Alegre: Pearson, 2021.