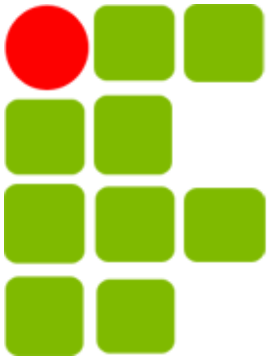
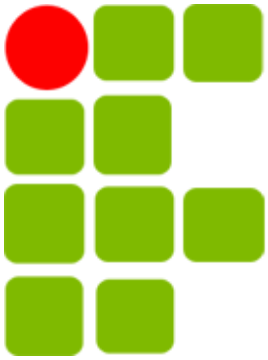


Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

Desenvolvimento de um Simulador de Redes com Análise de Configuração Assistida por Inteligência Artificial

Gustavo Ponciano Barbosa da Silva
Orientador: Prof. Danyllo Wagner Albuquerque, DSc.

Campina Grande-PB
2025



Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Tecnologia em Telemática

Desenvolvimento de um Simulador de Redes com Análise de Configuração Assistida por Inteligência Artificial

Gustavo Ponciano Barbosa da Silva

Trabalho de Conclusão de Curso apresentado ao Curso de Tecnologia em Telemática do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como requisito parcial para obtenção do título de Bacharel em Telemática.

Orientador: Prof. Danyllo Wagner Albuquerque, DSc.

Campina Grande-PB
2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa – CRB 15/479

S586a Silva, Gustavo Ponciano Barbosa da
Desenvolvimento de um simulador de redes com análise de configuração assistida por inteligência artificial / Gustavo Ponciano Barbosa da Silva. - Campina Grande, 2025.
73 f..: il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia em Telemática) - Instituto Federal da Paraíba, 2023.
Orientador: Prof. Dr. Danyllo Wagner Albuquerque

1. Telemática. 2. Redes de computadores. 3. Inteligência artificial. 4. Sistemas Web. I. Albuquerque, Danylo Wagner. II Título.

CDU 004.7/8

Desenvolvimento de um Simulador de Redes com Análise de Configuração Assistida por Inteligência Artificial

Gustavo Ponciano Barbosa da Silva

Prof. Danyllo Wagner Albuquerque, DSc.
Orientador

Prof. Marcelo Portela Sousa, DSc.
Membro da Banca

Prof. Victor Emanuel Farias da Costa Borges, MSc.
Membro da Banca

Campina Grande-PB
2025

Agradecimentos

A realização deste trabalho não seria possível sem o apoio e a contribuição de várias pessoas, às quais expresso minha sincera gratidão. Em primeiro lugar, agradeço a Deus, que me concedeu oportunidades, força de vontade e coragem para superar todas as dificuldades.

À minha família, especialmente aos meus pais, pelo apoio incondicional e pela compreensão ao longo desta jornada. Ao meu orientador, Prof. Danyllo Wagner Albuquerque, DSc., pela orientação dedicada, pelo compartilhamento de seu conhecimento e por estar sempre disposto a auxiliar durante todas as fases deste projeto.

Agradeço também a todos os professores do curso de Tecnologia em Telemática, cujos ensinamentos foram fundamentais para que eu pudesse alcançar a conclusão desta etapa. Quero destacar especialmente os professores Iana Daya e Ewerton Castro, que foram verdadeiros mentores e, mesmo sem perceberem, tornaram-se fontes valiosas de encorajamento em momentos desafiadores.

Aos meus amigos e colegas de curso, pelo apoio, incentivo e companheirismo ao longo dessa longa jornada, que, graças a eles, se tornou muito mais leve. A todos, o meu mais sincero obrigado.

Por fim, não poderia deixar de agradecer ao Instituto Federal da Paraíba (IFPB), uma instituição respeitada, que não só proporcionou uma educação de qualidade, mas também ofereceu suporte fundamental ao longo da minha jornada acadêmica. Agradeço ao IFPB pela oportunidade de estudar em um ambiente que valoriza o conhecimento, o crescimento pessoal e o desenvolvimento profissional.

Resumo

Contextualização: O ensino de redes de computadores enfrenta barreiras significativas devido à curva de aprendizado acentuada das interfaces de linha de comando (CLI) e à exigência de *hardware* robusto para a execução de simuladores tradicionais, o que limita o acesso via dispositivos móveis. **Objetivo:** Este trabalho apresenta o desenvolvimento de uma plataforma *web* de simulação que democratiza o acesso ao ensino prático, oferecendo uma interface gráfica simplificada sincronizada com um emulador de CLI. **Metodologia:** Adotando a *Design Science Research* (DSR), o artefato foi construído utilizando *Next.js* e *React Flow*, implementado sob uma arquitetura de "Fonte Única da Verdade"e implantado em ambiente de produção (VPS Hostinger). A inovação central reside na integração com o modelo de linguagem Google Gemini para a análise semântica automática de configurações. **Resultados:** A validação funcional em cenário real confirmou a estabilidade da aplicação e a consistência de dados entre as interfaces gráfica e textual. A avaliação técnica registrou uma latência média de 31 segundos para a análise assistida, intervalo compensado pela capacidade do sistema de diagnosticar erros lógicos complexos, como rotas inalcançáveis e bloqueios indevidos de ACL, e de fornecer explicações pedagógicas detalhadas que simuladores convencionais não oferecem. **Conclusão:** A ferramenta atua como um andaime cognitivo (*scaffolding*), transformando falhas de configuração em oportunidades de aprendizado guiado e promovendo a inclusão digital no ensino técnico.

Palavras-chave: Simulador de redes, Inteligência Artificial, LLM, Ensino de redes, Acessibilidade Digital.

Abstract

Context: Computer networks education faces significant barriers due to the steep learning curve of Command Line Interfaces (CLI) and the robust hardware requirements of traditional simulators, which often hinder access via mobile devices. **Objective:** This work presents the development of a web-based simulation platform that democratizes access to practical training, offering a simplified graphical interface synchronized with a CLI emulator. **Methodology:** Adopting Design Science Research (DSR), the artifact was built using Next.js and React Flow, employing a *Single Source of Truth* architecture, and deployed in a production environment (Hostinger VPS). The core innovation lies in integrating the Google Gemini Large Language Model for automated semantic configuration analysis. **Results:** Validation in a real-world scenario confirmed the application's stability and data consistency between graphical and textual interfaces. Technical evaluation recorded an average latency of 31 seconds for the assisted analysis, a delay offset by the system's ability to diagnose complex logical errors, such as unreachable routes and ACL conflicts, and provide detailed pedagogical explanations unavailable in standard simulators. **Conclusion:** The tool acts as a cognitive scaffold, transforming configuration errors into guided learning opportunities and fostering digital inclusion in technical education.

Keywords: Network simulator, Artificial Intelligence, LLM, Network education, Digital Accessibility.

Lista de Siglas e Abreviaturas

ACL	*Access Control List* (Lista de Controle de Acesso)
API	*Application Programming Interface*
BFF	*Backend-for-Frontend*
BGP	*Border Gateway Protocol*
CLI	*Command Line Interface* (Interface de Linha de Comando)
CORS	*Cross-Origin Resource Sharing*
CSS	*Cascading Style Sheets*
DSR	*Design Science Research*
EIGRP	*Enhanced Interior Gateway Routing Protocol*
EVE-NG	*Emulated Virtual Environment - Next Generation*
GNS3	*Graphical Network Simulator 3*
GUI	*Graphical User Interface* (Interface Gráfica do Usuário)
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IFPB	Instituto Federal da Paraíba
IOS	*Internetwork Operating System* (Sistema Operacional da Cisco)
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
LAN	*Local Area Network* (Rede Local)
LLM	*Large Language Model* (Modelo de Linguagem de Larga Escala)
MAC	*Media Access Control*
NAT	*Network Address Translation*
NOS	*Network Operating System*
OSI	*Open Systems Interconnection*
OSPF	*Open Shortest Path First*
PWA	*Progressive Web App*
RF	Requisitos Funcionais
RIP	*Routing Information Protocol*
RNF	Requisitos Não Funcionais
RTT	*Round-Trip Time*
SLM	*Small Language Models*
SSH	*Secure Shell*
SSG	*Static Site Generation*
SSR	*Server-Side Rendering*
STP	*Spanning Tree Protocol*
TCC	Trabalho de Conclusão de Curso
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
UDL	*Universal Design for Learning*
UI	*User Interface* (Interface de Usuário)
VLAN	*Virtual Local Area Network*
VPS	*Virtual Private Server*
VRP	*Versatile Routing Platform* (Sistema Operacional da Huawei)

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O avanço da tecnologia da informação e comunicação tem intensificado a necessidade de profissionais capacitados em redes de computadores. Nesse contexto, a prática de configuração de dispositivos de rede torna-se fundamental para a formação de estudantes e profissionais, pois possibilita a compreensão efetiva do funcionamento de protocolos, topologias e mecanismos de comunicação (Tanenbaum and Wetherall, 2011).

Contudo, o aprendizado inicial muitas vezes é prejudicado pela complexidade inerente ao uso de interfaces de linha de comando (do inglês *Command Line Interface (CLI)*), que requerem conhecimento prévio de sintaxe e de comandos específicos. Ferramentas como o Cisco Packet Tracer (Cisco Systems, 2024) e o GNS3 (GNS3 Community, 2024) têm desempenhado um papel essencial no ensino de redes, oferecendo ambientes de simulação próximos à realidade prática. Contudo, apesar de sua reconhecida eficácia, essas ferramentas apresentam uma curva de aprendizado elevada, especialmente para iniciantes, o que pode dificultar a assimilação dos conceitos fundamentais e desmotivar estudantes que ainda não têm familiaridade adequada com os dispositivos e comandos de rede.

Outro aspecto importante está relacionado à acessibilidade digital. O ensino de redes tradicionalmente requer o uso de equipamentos físicos, como roteadores e *switches*, cujo custo elevado dificulta a adoção em larga escala. Embora *software*s de simulação surjam como alternativa, nem todos os estudantes dispõem de computadores pessoais capazes de executar ferramentas como Packet Tracer, GNS3 ou EVE-NG (EVE-NG, 2024). Dados institucionais do IFPB de 2020 reforçam essa limitação: cerca de 29,96% dos estudantes dependem exclusivamente de dispositivos móveis, como *smartphones* ou *tablets*, para acessar a internet, enquanto apenas 70,04% utilizam computadores (IFPB, 2020). Em muitos casos, os *smartphones* representam o único recurso computacional disponível, o que compromete o aprendizado, já que simuladores de redes demandam instalação local e recursos de *hardware* que ultrapassam a capacidade desses dispositivos (Cardoso, 2024; Asadi et al., 2024b). Esse cenário reforça a necessidade de soluções baseadas em navegadores, compatíveis com diferentes plataformas.

Diante desse contexto, este trabalho propõe o desenvolvimento de uma plataforma *web* interativa para a simulação de cenários de redes de computadores, buscando oferecer uma alternativa mais acessível e intuitiva. A proposta contempla a criação de uma interface gráfica simplificada, que permita a configuração de dispositivos de rede sem a necessidade de domínio completo da CLI. A plataforma também manterá a opção de configuração via CLI, não como foco principal, mas como um recurso complementar para configurações específicas ou por preferência do usuário. Adicionalmente, será possível exportar as confi-

gurações geradas, tanto de dispositivos individuais quanto do cenário de rede construído como um todo.

Como diferencial, a solução integrará um modelo de linguagem de larga escala (do inglês, *Large Language Model* (LLM)), que atuará como um sistema consultivo, oferecendo análise das configurações realizadas, apontando erros comuns, sugerindo melhorias e explicando conceitos técnicos relacionados ao cenário criado (Brown et al., 2020; Touvron et al., 2023). Dessa forma, a plataforma não se limita ao caráter de simulador, assumindo também um papel pedagógico, capaz de auxiliar estudantes e profissionais em processo de formação.

1.1 Problemática

Apesar da ampla disponibilidade de simuladores, como o *Cisco Packet Tracer* (Cisco Systems, 2024), o GNS3 (GNS3 Community, 2024), o EVE-NG (EVE-NG, 2024) e o ns-3 (Chan et al., 2015), ainda persistem desafios relevantes no ensino de redes por meio dessas ferramentas. Esses desafios evidenciam a necessidade de soluções alternativas, mais acessíveis e alinhadas às demandas pedagógicas atuais.

Ferramentas tradicionais exigem que o estudante tenha familiaridade com a sintaxe de CLI e comandos específicos, o que constitui uma barreira significativa para iniciantes. Estudos mostram que, embora ferramentas como *Packet Tracer* auxiliem na visualização e na compreensão de conceitos, muitos estudantes relatam dificuldades iniciais relacionadas ao uso da linguagem técnica e à abstração exigida (bin Abdul Rashid et al., 2020; Mwansa et al., 2024). De modo semelhante, análises pedagógicas apontam que a maioria dos simuladores carece de interfaces verdadeiramente amigáveis, o que compromete a experiência do usuário e pode gerar desmotivação (Makasiranondh et al., 2010).

Outro problema está associado à exigência de recursos computacionais elevados. Simuladores como GNS3 e EVE-NG demandam máquinas potentes, o que inviabiliza seu uso por estudantes que dispõem apenas de dispositivos modestos. Tal cenário impõe uma barreira de entrada severa, dado que as soluções convencionais carecem da portabilidade necessária para atender a estudantes que dependem exclusivamente de plataformas móveis. Revisões recentes confirmam essa barreira, destacando a escassez de soluções acessíveis em plataformas móveis ou em navegadores (Asadi et al., 2024a).

A maior parte dos simuladores limita-se a executar topologias e comandos, sem oferecer mecanismos de *feedback* ou de análise pedagógica. Dessa forma, o estudante configura roteadores e *switches* sem avaliação imediata de erros, inconsistências ou boas práticas. Trabalhos recentes têm apontado a importância de integrar agentes inteligentes e modelos de linguagem como apoio ao ensino, oferecendo tutoria automatizada, explicações de conceitos e orientação personalizada (Brown et al., 2020; Touvron et al., 2023; Zhao et al., 2025). Entretanto, até o momento, há uma lacuna significativa na aplicação dessas tecnologias no ensino de redes de computadores.

Por fim, em contextos híbridos, é comum a necessidade de combinar equipamentos reais com simuladores. Embora existam propostas de integração entre dispositivos físicos e ambientes virtuais, como no caso do GNS3, tais soluções apresentam configuração complexa, infraestrutura custosa e não contemplam mecanismos de análise automatizada ou suporte inteligente (Chan et al., 2015).

Em síntese, a problemática enfrentada no ensino de redes envolve: (i) a curva de aprendizado elevada imposta pelas ferramentas atuais, (ii) a falta de acessibilidade em plataformas móveis ou de baixo custo, (iii) a ausência de *feedback* pedagógico integrado e (iv) a dificuldade em integrar ambientes virtuais e físicos de forma eficiente. Esses aspectos justificam o desenvolvimento de uma solução inovadora, que incorpore recursos de acessibilidade e suporte inteligente por meio de LLMs.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver uma plataforma *web* para simulação de cenários de redes de computadores, que permita configuração simplificada de dispositivos, exportação de arquivos de configuração e análise automática assistida por modelos de linguagem de larga escala (LLMs).

Para alcançar esse objetivo, definem-se os seguintes objetivos específicos:

- Analisar os principais desafios enfrentados no uso de simuladores de redes existentes, considerando aspectos de curva de aprendizado, acessibilidade e suporte pedagógico;
- Estabelecer requisitos para uma plataforma acessível e pedagógica, que integre interface gráfica simplificada e análise automatizada por LLMs;
- Projetar e implementar uma plataforma web a partir de requisitos funcionais e não-funcionais bem definidos;
- Validar o funcionamento da plataforma em cenários práticos de configuração de dispositivos, incluindo a exportação de arquivos e a análise de erros;
- Avaliar a usabilidade e eficácia da plataforma em comparação a soluções já existentes, como o *Cisco Packet Tracer*;
- Verificar o potencial pedagógico da solução em ambientes de ensino-aprendizagem em redes de computadores.

1.3 Metodologia

Para conduzir o desenvolvimento deste trabalho, adotou-se a abordagem de *Design Science Research* (DSR), proposta por Peffers *et al.* (Peffers et al., 2007), amplamente empregada em investigações que envolvem a concepção, construção e avaliação de artefatos tecnológicos. Essa metodologia foi escolhida porque este trabalho envolve a concepção e entrega de um artefato tecnológico, o que se alinha diretamente ao propósito da DSR. Como a abordagem é estruturada especificamente para orientar o desenvolvimento, a implementação e a avaliação de soluções inovadoras, ela oferece um enquadramento metodológico adequado para guiar a criação do protótipo proposto e assegurar seu rigor científico.

A DSR organiza o processo em etapas cíclicas e iterativas, permitindo refinar o artefato conforme os requisitos evoluem e novas evidências surgem durante a validação. As etapas consideradas neste trabalho são:

1. **Identificação do problema e motivação:** consiste em compreender as barreiras enfrentadas por estudantes e professores no uso de simuladores de redes (curva de aprendizado elevada, custo de equipamentos, falta de acessibilidade e ausência de *feedback* pedagógico);

2. **Definição dos objetivos da solução:** a partir da análise do problema, estabelecem-se requisitos para a plataforma proposta, contemplando acessibilidade, interface simplificada, exportação de configurações e suporte inteligente por LLMs;

3. **Design e desenvolvimento:** envolve a concepção arquitetural do sistema e a implementação incremental da plataforma por meio de prototipação evolutiva, com ênfase na validação progressiva das funcionalidades, no refinamento contínuo da interface e na adequação às necessidades dos usuários;

4. **Demonstração:** execução de cenários de uso representativos (ex: configuração de dispositivos, exportação de arquivos, inspeção de inconsistências) para evidenciar o funcionamento da solução;

5. **Avaliação:** realização de testes de usabilidade e comparações preliminares com ferramentas consolidadas (ex: *Cisco Packet Tracer*), além de estudos de caso exploratórios em contextos educacionais;

6. **Comunicação:** registro e disseminação dos resultados obtidos neste trabalho, incluindo as limitações identificadas e os próximos passos para a evolução da solução.

O caráter iterativo da DSR é essencial: os resultados obtidos nas etapas de Demonstração e Avaliação realimentam o processo, gerando ajustes nos objetivos e no *design*, o que fortalece a robustez e a utilidade do artefato.

Para reforçar o alinhamento entre os objetivos específicos definidos na Seção 1.2 e o processo metodológico, apresenta-se a Tabela 1.1, que evidencia como cada objetivo é abordado pelas etapas da DSR.

1.4 Relevância e Justificativa

A relevância deste trabalho decorre de três dimensões convergentes — pedagógica, tecnológica e científico — que se manifestam no ensino de redes de computadores. Do ponto de vista **pedagógico**, persiste uma barreira de entrada para estudantes iniciantes, decorrente da dependência de interfaces em linha de comando (CLI) e da necessidade de dominar sintaxes e comandos específicos logo nas primeiras experiências práticas. Ao propor uma plataforma com interface gráfica simplificada, capaz de orientar a configuração de dispositivos sem eliminar o acesso à CLI, este projeto contribui para reduzir a carga cognitiva inicial e favorecer a progressão do aprendizado por etapas.

No plano **tecnológico**, a proposta incorpora recursos de análise automática baseados em LLMs. Esses modelos podem oferecer *feedback* imediato sobre inconsistências, apontar boas práticas e explicar os conceitos subjacentes às configurações realizadas. Tal mecanismo adiciona uma camada de suporte (*scaffolding*) ao ambiente de simulação, aproximando-o de um tutor inteligente e ampliando o potencial formativo da solução. Diferentemente de simuladores que se restringem à execução de topologias e comandos, a

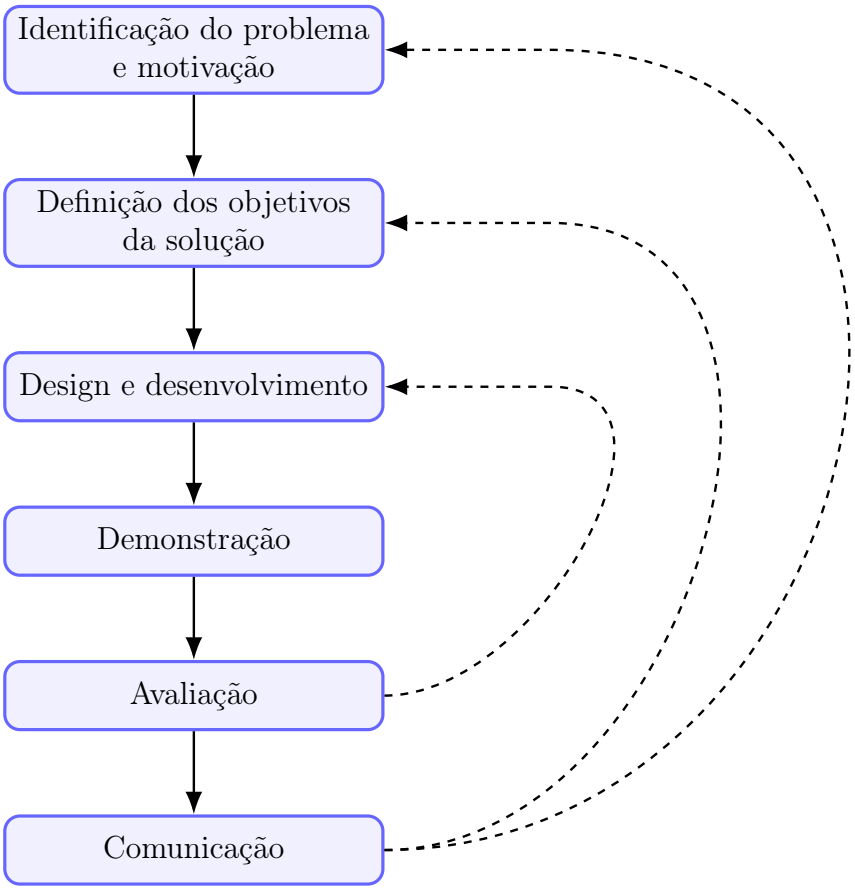


Figura 1.1: Etapas e iterações na DSR.

Tabela 1.1: Alinhamento entre objetivos específicos e etapas da DSR

Objetivo	**Etapas DSR relacionadas**
OE1: analisar desafios em simuladores existentes	Identificação do problema; Avaliação (diagnóstico inicial)
OE2: estabelecer requisitos da solução	Definição dos objetivos da solução
OE3; projetar e implementar a plataforma	Design e desenvolvimento; Demonstração
OE4: validar funcionamento do protótipo em cenários práticos	Demonstração; Avaliação
OE5: avaliar usabilidade e eficácia em comparação a simuladores existentes	Avaliação
OE6: verificar potencial pedagógico da solução	Avaliação; Comunicação

integração com LLMs busca transformar o ambiente prático em um espaço de orientação e reflexão, promovendo autonomia e compreensão conceitual.

Sob a ótica da acessibilidade e inclusão digital, há um descompasso entre os requisitos

computacionais de simuladores tradicionais e a realidade de parte do público-alvo, que frequentemente dispõe apenas de dispositivos modestos ou de acesso predominantemente móvel. Ao privilegiar uma solução *web-based*, responsiva e *cross-platform*, o trabalho atende ao princípio de ampliar o alcance do ensino prático, permitindo atividades fora de laboratórios dedicados e favorecendo contextos de aprendizagem híbridos e remotos. Essa opção tecnológica também reduz custos de infraestrutura e manutenção, beneficiando instituições de ensino com restrições orçamentárias.

Adicionalmente, a contribuição **científica** e profissional advém do enquadramento metodológico em DSR. Ao conceber, desenvolver, demonstrar e avaliar um artefato orientado a um problema real do ensino de redes, o estudo oferece evidências sobre: (i) requisitos de usabilidade e eficácia para ambientes de simulação acessíveis; (ii) utilidade prática de mecanismos de análise assistida por LLMs no diagnóstico de erros e na explicação de conceitos; e (iii) diretrizes de projeto para integrar, de forma ética e transparente, IA generativa a contextos educacionais. Tais resultados — ainda que parciais durante a execução do trabalho — são potencialmente transferíveis para outras disciplinas técnicas que combinam prática de laboratório, operação de ferramentas e necessidade de *feedback* imediato.

1.5 Estrutura do Trabalho

Esta pesquisa foi organizada de modo a refletir, de forma rastreável, as etapas do processo de DSR adotado. Em cada capítulo, indicamos explicitamente as etapas correspondentes — usando exatamente as denominações: **Identificação do problema e motivação**, **Definição dos objetivos da solução**, **Design e desenvolvimento**, **Demonstração**, **Avaliação** e **Comunicação**.

Capítulo 1 — Introdução

Alinha-se a: **Identificação do problema e motivação** & **Definição dos objetivos da solução**.

Apresenta o contexto, a problemática, os objetivos (gerais e específicos), a metodologia adotada (DSR), a relevância/justificativa e esta própria estrutura do trabalho.

Capítulo 2 — Fundamentação Teórica

Alinha-se principalmente a: **Definição dos objetivos da solução**.

Sistematiza o conhecimento que embasa os requisitos do artefato (simuladores de redes, usabilidade/acessibilidade, LLMs), servindo de suporte direto à definição dos objetivos e às decisões de projeto.

Capítulo 3 — Metodologia

Alinha-se transversalmente às etapas da DSR, com ênfase em: **Definição dos objetivos da solução** e preparação para **Design e desenvolvimento**, **Demonstração** e **Avaliação**.

Detalha o processo DSR, bem como os procedimentos, instrumentos e critérios a serem utilizados nas fases subsequentes.

Capítulo 4 — Design e Desenvolvimento do Artefato

Alinha-se a: **Design e desenvolvimento**.

Descreve requisitos, arquitetura, requisitos, arquitetura, implementação da interface e integração com LLMs

Capítulo 5 — Demonstração

Alinha-se a: **Demonstração**.

Apresenta cenários representativos de uso (configuração de dispositivos, exportação de *scripts*, inspeção de inconsistências) que evidenciam o funcionamento do artefato.

Capítulo 6 — Avaliação

Alinha-se a: **Avaliação**.

Reporta procedimentos e resultados parciais de testes de usabilidade, comparação inicial com simuladores (ex: *Packet Tracer*/GNS3) e análise do potencial pedagógico.

Capítulo 7 — Conclusões e Trabalhos Futuros

Alinha-se a: **Comunicação**.

Sintetiza contribuições, limitações e lições aprendidas, além de apontar desdobramentos futuros e diretrizes para a evolução do artefato.

Materiais complementares (Apêndices/Anexos) incluem instrumentos, roteiros e detalhes técnicos que reforçam a reprodutibilidade e a transparência do estudo.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta a base conceitual que sustenta o desenvolvimento do artefato proposto. São discutidos: (i) fundamentos de redes de computadores e tarefas típicas de configuração; (ii) classes de ferramentas de simulação e suas limitações; (iii) papéis e limites de interfaces gráficas no ensino de redes; (iv) acessibilidade e execução em dispositivos móveis; e (v) capacidades e riscos de modelos de linguagem de larga escala (LLMs) aplicados ao domínio. Ao final, sintetizam-se os objetivos de solução que emergem desta análise.

2.1 Redes de Computadores e Dispositivos de Rede

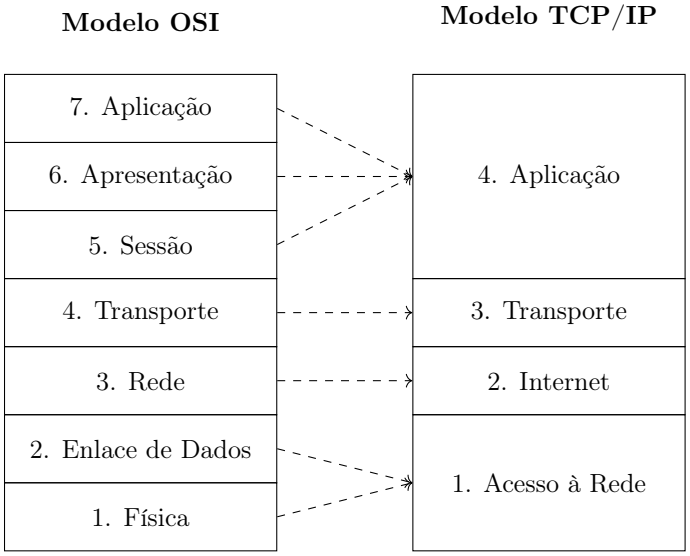
As redes de computadores são sistemas complexos que permitem a comunicação entre dispositivos computacionais. A sua arquitetura é frequentemente descrita por meio de modelos de camadas, que organizam as diversas funções e protocolos necessários à transmissão de dados (Kurose and Ross, 2016). Os modelos mais conhecidos são o OSI (*Open Systems Interconnection*) e o TCP/IP (*Transmission Control Protocol/Internet Protocol*). O modelo TCP/IP, sendo a base da internet moderna, é o mais utilizado na prática e divide a arquitetura em quatro camadas principais, conforme ilustrado na Figura 2.1.

A formação em redes requer a compreensão de como dispositivos intermediários, como *switches* e roteadores, operam em diferentes camadas para viabilizar a comunicação (Tanenbaum and Wetherall, 2011):

- ***Switches***: Operam primariamente na Camada de Enlace (Camada 2 do modelo OSI). Eles conectam dispositivos dentro de uma mesma rede local (LAN) e encaminham pacotes de dados (*frames*) com base nos endereços físicos (MAC *addresses*) dos dispositivos de destino.
- **Roteadores**: Atuam na Camada de Rede (Camada 3). Sua principal função é interconectar diferentes redes e determinar o melhor caminho para que os pacotes de dados (*packets*) alcancem seus destinos, utilizando para isso endereços lógicos (endereços IP).

A configuração desses dispositivos é uma tarefa central para administradores de redes. Em contextos educacionais, as atividades práticas recorrentes envolvem um conjunto de conceitos que, quando mal aplicados, geram erros comuns. A Tabela 2.1 detalha algumas dessas tarefas e os erros frequentemente associados a elas.

Figura 2.1: Comparativo entre os modelos de camadas OSI e TCP/IP



Fonte: Elaborada pelo autor, com base em (Tanenbaum and Wetherall, 2011).

Tabela 2.1: Tarefas de Configuração, Conceitos e Erros Comuns para Iniciantes

Tarefa de Configuração	**Conceitos Envolvidos**	**Erros Comuns de Iniciantes**
Endereçamento IP	Sub-redes, máscaras de rede, *gateway* padrão.	Máscaras de sub-rede incorretas; endereços IP duplicados; *gateway* fora da sub-rede local.
Criação de VLANs	Domínios de *broadcast*, portas de acesso (*access*), portas de tronco (*trunk*), encapsulamento 802.1Q.	Esquecer de configurar o modo tronco em enlaces entre switches; atribuir uma porta a uma VLAN inexistente.
Roteamento Estático/Dinâmico	Tabela de roteamento, métricas, protocolos (RIP, OSPF), redes diretamente conectadas.	Rotas estáticas com próximo salto (next-hop) inalcançável; redes não anunciadas em protocolos de roteamento dinâmico.
Listas de Controle de Acesso (ACLs)	Ordem de processamento (de cima para baixo), máscaras de *wildcard*, negação implícita.	Ordem incorreta das regras (uma regra mais específica após outra mais genérica); aplicação da ACL na interface ou na direção errada.

A identificação e correção desses erros são cruciais para o processo de aprendizagem, justificando a necessidade de ferramentas que não apenas simulem o ambiente, mas também ofereçam suporte e *feedback* pedagógico.

2.2 Ferramentas de Simulação de Redes

As ferramentas de simulação e emulação são essenciais para o ensino prático de redes, pois oferecem uma alternativa segura e de baixo custo aos laboratórios que dispõem de equipamentos físicos. Elas podem ser classificadas em um espectro que vai desde a simulação de protocolos até a emulação de sistemas operacionais de rede (NOS - *Network Operating System*) completos. As principais ferramentas utilizadas no contexto educacional são:

- **Simuladores (ex: *Cisco Packet Tracer*)**: São ambientes de *software* que simulam o comportamento de dispositivos e protocolos de rede. O *Packet Tracer*, desenvolvido pela Cisco, é amplamente utilizado por sua interface visual intuitiva e por simular um subconjunto dos comandos do sistema operacional Cisco IOS. É ideal para iniciantes, mas apresenta limitações em termos de fidelidade e de suporte a funcionalidades avançadas ou a dispositivos de outros fabricantes (Cisco Systems, 2024; Mwansa et al., 2024).

- **Emuladores (ex: GNS3, EVE-NG)**: Estas ferramentas permitem executar imagens reais de sistemas operacionais de rede (como o Cisco IOS e o Juniper JunOS) em um ambiente virtualizado. GNS3 e EVE-NG oferecem alto grau de realismo e flexibilidade, sendo preferidos por profissionais e estudantes avançados (GNS3 Community, 2024; EVE-NG, 2024). Contudo, exigem mais recursos computacionais e podem apresentar uma curva de aprendizado mais acentuada devido à necessidade de obter e configurar as imagens dos sistemas.

A Tabela 2.2 apresenta um comparativo entre essas ferramentas, destacando suas principais características e sua adequação a diferentes públicos-alvo.

Apesar de sua utilidade, essas ferramentas consolidadas apresentam lacunas importantes: (i) a dependência de interfaces de linha de comando (CLI) impõe uma barreira inicial aos novatos (bin Abdul Rashid et al., 2020); (ii) a exigência de recursos de *hardware* limita o acesso para muitos estudantes; e (iii) a ausência de um mecanismo de *feedback* inteligente, que guie o aprendizado ao apontar erros e explicar conceitos, é uma carência notável (Makasiranondh et al., 2010).

2.3 Usabilidade e Acessibilidade no Ensino de Redes

A complexidade da sintaxe da CLI pode desviar o foco do estudante do aprendizado conceitual para a memorização de comandos. Interfaces Gráficas de Usuário (GUI) bem projetadas podem mitigar esse problema ao reduzir a carga cognitiva extrínseca, permitindo que o aluno se concentre nos objetivos da tarefa (o "quê") em vez dos detalhes da implementação (o "como") (Sweller, 1988).

No ensino de redes, uma GUI pode atuar como um mecanismo de tutoria, oferecendo suporte estruturado que é gradualmente removido à medida que o aprendiz adquire autonomia. Entre as estratégias mais eficazes estão a topologia visual interativa, que permite ao aluno construir a rede de forma intuitiva por meio de ações como "arrastar e soltar", facilitando a compreensão da estrutura física e lógica; e os formulários guiados (*wizards*), que simplificam tarefas complexas, como a configuração de ACLs ou NAT, ao coletar parâmetros por meio de campos estruturados que também validam as entradas.

Tabela 2.2: Comparativo entre Ferramentas de Simulação e Emulação de Redes

Critério	**Cisco Packet Tracer**	**GNS3**	**EVE-NG**
Tipo	Simulador	Emulador	Emulador
Realismo	Médio (comandos simulados)	Alto (imagens reais de NOS)	Alto (imagens reais de NOS)
Suporte a Vendedores	Apenas Cisco	Multi-vendedor (Cisco, Juniper, etc.)	Multi-vendedor (amplo suporte)
Recursos Computacionais	Baixo a Médio	Alto	Alto
Curva de Aprendizado	Baixa (ideal para iniciantes)	Alta (requer configuração de imagens)	Média a Alta
Feedback Pedagógico	Limitado (verificação de conectividade)	Nenhum (foco em emulação fiel)	Nenhum (foco em emulação fiel)
Acessibilidade	Instalação local; versão mobile limitada.	Instalação local; sem suporte móvel.	Instalação via servidor/VM; acesso via navegador.

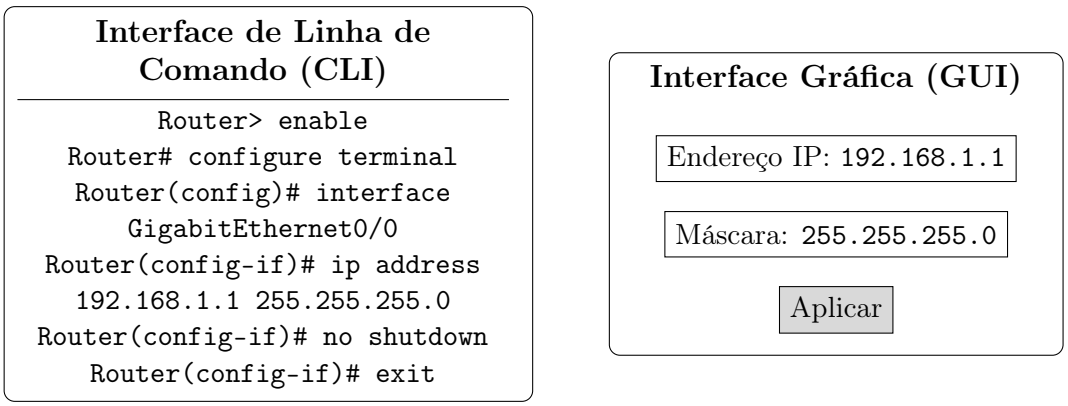


Figura 2.2: Contraste entre a Abordagem via CLI e uma GUI Simplificada para configuração de uma interface.

A Figura 2.2 ilustra a diferença conceitual entre configurar um dispositivo via CLI e realizar o mesmo procedimento por meio de uma interface gráfica simplificada.

Contudo, é fundamental que a GUI não se torne uma "caixa-preta"(Nielsen, 1994). Para evitar que o estudante aprenda apenas a "clicar em botões", a ferramenta deve sempre exibir os comandos CLI correspondentes à ação na GUI. Essa abordagem dual permite uma transição suave, em que o aluno primeiro utiliza a GUI para entender o processo e depois passa a inspecionar e, eventualmente, a escrever os comandos diretamente.

2.3.1 Acessibilidade Digital e Mobilidade

A transformação digital na educação expôs uma divisão digital significativa. Muitos estudantes não possuem computadores pessoais potentes, sendo o *smartphone* seu principal ou único meio de acesso à internet e a recursos educacionais (Cardoso, 2024). Ferramentas de simulação tradicionais, que exigem instalação local e elevado consumo de recursos computacionais, acabam por excluir estudantes que dispõem apenas de dispositivos móveis ou de computadores de baixo desempenho (Asadi et al., 2024b,a).

Uma solução baseada na web (*web-based*) e com design responsivo, que se adapta a diferentes tamanhos de tela, é uma abordagem mais inclusiva e alinhada aos princípios do *Universal Design for Learning* (UDL) (Rose and Meyer, 2012). Tal solução oferece as seguintes vantagens:

- **Acessibilidade**: Pode ser acessada de qualquer dispositivo com navegador, incluindo *smartphones*, *tablets* e computadores de baixo custo.
- **Zero Instalação**: Remove a barreira da instalação e configuração de *software* complexo.
- **Flexibilidade**: Permite a aprendizagem em qualquer lugar e a qualquer hora, fora dos limites do laboratório de informática.

2.4 Modelos de Linguagem de Larga Escala (LLMs)

Os Modelos de Linguagem de Larga Escala (LLMs) são uma categoria de inteligência artificial treinada em vastos conjuntos de dados de texto, tornando-os capazes de gerar, traduzir e analisar linguagem humana com notável fluidez. Em contextos educacionais, essa capacidade permite que eles funcionem como tutores inteligentes, oferecendo o tipo de *feedback* personalizado e explicações sob demanda que faltam nas ferramentas tradicionais (Brown et al., 2020; Zhao et al., 2025). A aplicação deles ao ensino de redes é particularmente promissora porque, ao serem treinados em grandes volumes de documentação técnica e código-fonte, esses modelos adquirem proficiência em linguagens formais. Visto que a Interface de Linha de Comando (CLI) de equipamentos de rede possui uma sintaxe rigorosa e uma lógica estruturada, os LLMs a processam de forma análoga a uma linguagem de programação.

Essa capacidade de compreender a CLI permite que um LLM integrado a um simulador (como o GPT (Brown et al., 2020) ou o LLaMA (Touvron et al., 2023)) vá além da simples validação de sintaxe e execute tarefas pedagógicas complexas:

- **Analisar Configurações Semânticas**: Identificar erros lógicos que são sintaticamente válidos, mas funcionalmente incorretos, como os erros de sub-rede ou VLAN listados na Tabela 2.1.
- **Sugerir Boas Práticas**: Recomendar otimizações de segurança ou desempenho que um simulador tradicional, focado apenas na conectividade, ignoraria.
- **Explicar Conceitos**: Fornecer explicações contextuais sobre por que uma configuração está incorreta ou qual o propósito de um protocolo, preenchendo o "vazio" de *feedback* que muitas vezes frustra o estudante iniciante.

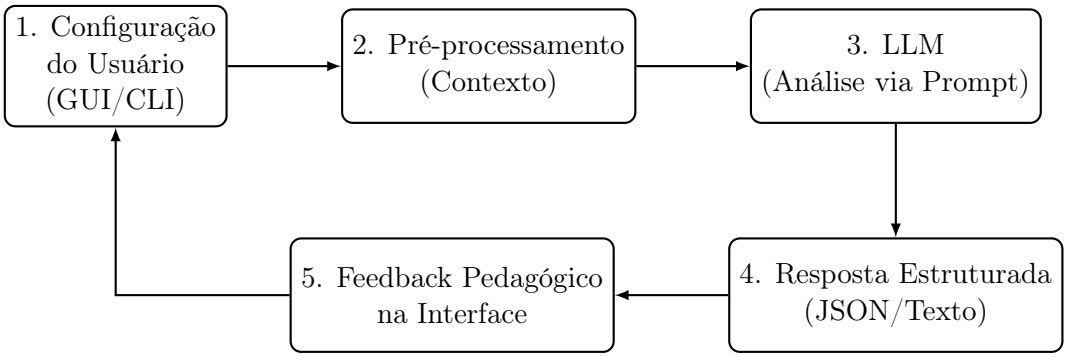


Figura 2.3: Fluxo de Análise de Configuração: O LLM recebe o estado da rede e retorna diagnósticos pedagógicos.

A Figura 2.3 apresenta um fluxo de trabalho conceitual para essa integração, onde o modelo de IA atua como uma camada de análise pedagógica.

Contudo, o uso de LLMs em um papel de tutoria apresenta riscos significativos que devem ser gerenciados. O principal é a alucinação, a capacidade do modelo de gerar informações incorretas, mas plausíveis, o que é especialmente perigoso ao sugerir comandos de configuração errados (Zhao et al., 2025). Além disso, a privacidade dos dados de configuração enviados para análise, bem como o custo e a latência de resposta de APIs comerciais, são desafios de implementação prática. Para mitigar esses riscos, estratégias como verificações determinísticas prévias (*linters*) e, crucialmente, o uso de **Engenharia de *Prompt*** para guiar e restringir as respostas do modelo são essenciais. Por fim, o sistema deve sempre submeter as sugestões à aprovação do usuário, mantendo-o como o agente central do aprendizado.

2.5 Considerações Finais do Capítulo

Com base nas discussões acima, a solução a ser desenvolvida deverá atender aos seguintes objetivos (funcionais e não-funcionais):

1. **Configuração acessível e progressiva**: Prover interface gráfica baseada em formulários/assistentes para as tarefas introdutórias (endereçamento, VLANs, roteamento, ACLs), com exibição do *diff* de comandos e opção de alternância GUI/CLI.

2. **Exportação e interoperabilidade**: Permitir exportar configurações por dispositivo e por cenário completo, em formatos compatíveis com equipamentos reais ou com outras ferramentas didáticas.

3. **Análise automática assistida**: Integrar um módulo de verificação que aponte problemas comuns (endereçamento inválido, rotas inconsistentes, VLAN não propagada, ordem de ACLs, STP) e ofereça sugestões *explicadas*, com salvaguardas contra comandos destrutivos.

4. **Acessibilidade e alcance**: Disponibilizar a aplicação como *web app* responsiva (*mobile-first*), com baixo consumo de recursos e suporte a uso intermitente (PWA), ampliando o acesso fora de laboratórios dedicados.

5. **Transparência e segurança**: Registrar e exibir o raciocínio verificável (*por que* uma regra é inválida), permitir revisão/edição das recomendações antes de aplicá-las e adotar políticas claras de privacidade quando houver uso de LLMs.

6. **Avaliabilidade pedagógica**: Instrumentar o protótipo para coleta ética de métricas de usabilidade e de utilidade do *feedback*, possibilitando comparações iniciais com simuladores de referência.

Esses objetivos consolidam requisitos diretamente acionáveis para o capítulo de projeto e implementação do artefato, mantendo o foco na redução da curva de aprendizagem, na democratização do acesso e no suporte pedagógico imediato durante a prática de configuração de redes.

2.5.1 Síntese e Encaminhamentos

Este capítulo apresentou a fundamentação teórica que sustenta a proposta da solução, discutindo os princípios de redes de computadores, as principais ferramentas de simulação e emulação, o papel das interfaces gráficas no processo de aprendizagem, os desafios de acessibilidade em dispositivos móveis e as potencialidades e riscos do uso de LLMs no ensino. A análise evidenciou limitações relevantes nas abordagens atuais, especialmente quanto à elevada curva de aprendizado, à dependência de infraestrutura computacional robusta e à ausência de mecanismos de *feedback* pedagógico inteligente.

A partir dessas lacunas, foram consolidados objetivos funcionais e não funcionais diretamente acionáveis para o desenvolvimento da solução proposta, com foco na acessibilidade, na transparência das configurações, na análise automática assistida e na avaliabilidade pedagógica do sistema. Assim, este capítulo estabelece o alicerce conceitual que orienta todas as decisões de projeto do artefato.

No próximo capítulo, apresenta-se a Metodologia, na qual são detalhados o método de pesquisa adotado, as etapas de desenvolvimento do protótipo, as estratégias de validação e os instrumentos de coleta e de análise de dados que permitirão avaliar, de forma sistemática, a efetividade da solução proposta.

Capítulo 3

Metodologia

Este capítulo descreve a abordagem metodológica adotada para a concepção, desenvolvimento e avaliação do artefato proposto. Inicialmente, apresenta-se a opção pela DSR como eixo central da pesquisa, justificando sua adequação ao problema investigado. Em seguida, detalha-se o processo de DSR conforme o modelo de Peffers et al. (2007), explicitando as etapas, os entregáveis e as evidências associadas. São também descritos os procedimentos de coleta e análise de dados, os critérios de avaliação do artefato, bem como os instrumentos e tecnologias empregados ao longo do estudo. Com isso, estabelece-se um roteiro sistemático que orienta as etapas subsequentes de design, demonstração e avaliação da solução.

3.1 Abordagem Geral

Dada a natureza deste trabalho, que visa à concepção e validação de um artefato tecnológico para resolver um problema educacional específico, adotou-se a *Design Science Research* (DSR). Segundo Peffers et al. (2007), a DSR é o método indicado quando o objetivo da pesquisa é criar inovações que definam ideias, práticas ou produtos técnicos por meio dos quais a análise, o projeto e a implementação de sistemas de informação podem ser realizados de forma eficaz.

Diferentemente de metodologias puramente descritivas, a DSR é prescritiva e orientada à solução de problemas. Ela permite que o desenvolvimento do simulador não seja apenas uma tarefa de engenharia de *software*, mas um processo de investigação científica em que cada decisão de design (interface, uso de LLMs, arquitetura) é tomada para atender a requisitos fundamentais de aprendizagem e acessibilidade (Hevner et al., 2004).

3.2 Processo do DSR

O processo aqui adotado segue as etapas propostas pelo modelo de processo de Peffers et al. (2007). Cada etapa gera entregáveis e evidências específicas, conforme detalhado nas subseções a seguir e ilustrado na Figura 3.1.

3.2.1 Identificação do problema e motivação

O ensino de redes de computadores, embora fundamental para a formação em tecnologia, apresenta desafios significativos. A configuração de dispositivos como roteadores

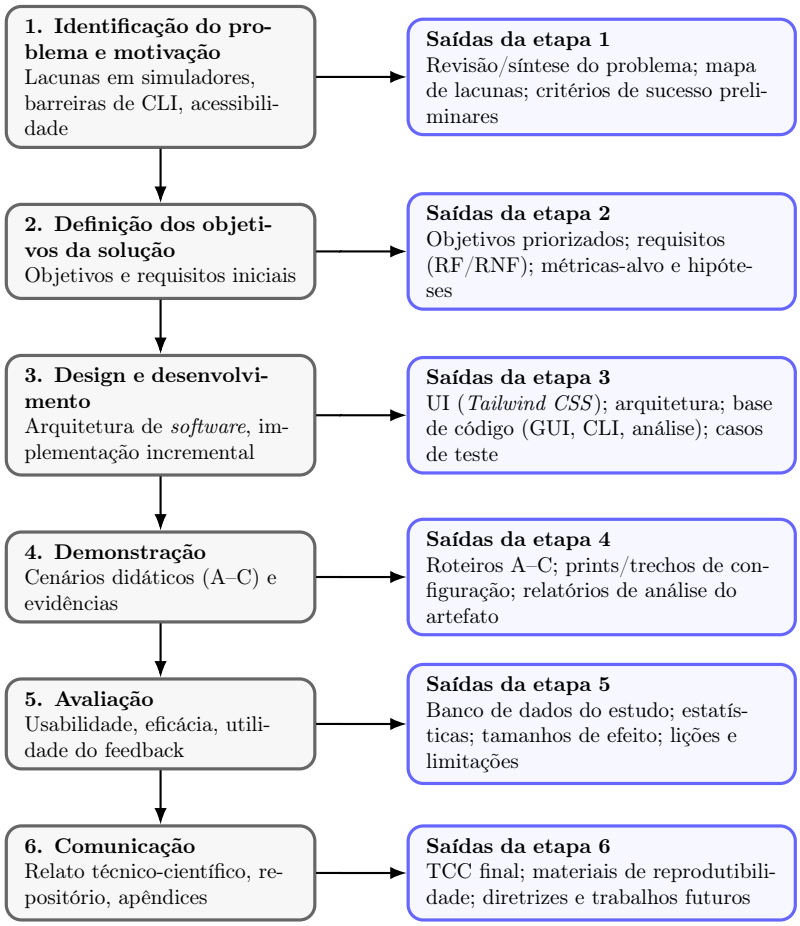


Figura 3.1: Etapas do DSR e principais saídas/entregáveis de cada etapa.

e *switches* exige a memorização de uma vasta gama de comandos (CLI), cuja sintaxe é muitas vezes complexa e pouco intuitiva para novatos (Tanenbaum and Wetherall, 2011).

Para diagnosticar a percepção desses desafios no contexto local do IFPB, foi realizada uma coleta de dados (instrumento detalhado no Apêndice A) com um grupo focal de 8 estudantes do Curso Superior de Tecnologia em Telemática. Conforme ilustrado na Figura 3.2, a amostra é composta majoritariamente (62,5%) por alunos do 5º ou 6º períodos, indicando que os participantes já possuem contato prévio com a disciplina de redes e vivência prática com as ferramentas atuais.

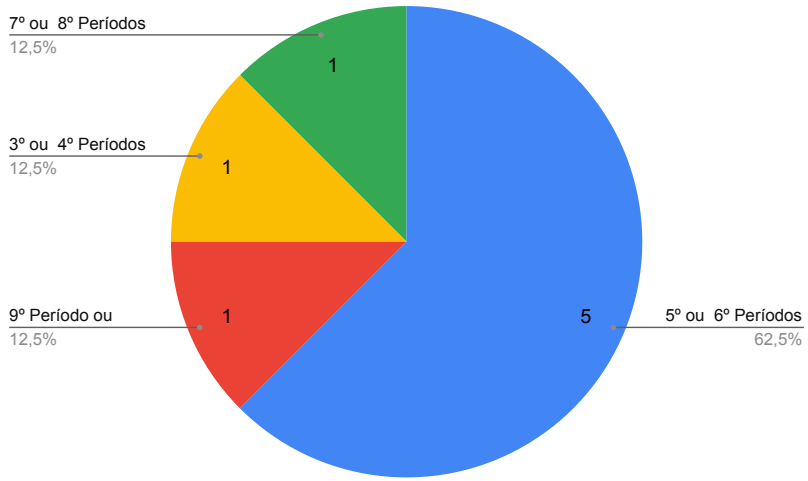


Figura 3.2: Distribuição dos participantes por período letivo (n=8).
Fonte: Dados da pesquisa (2025).

Barreiras no Aprendizado e Limitações dos Simuladores Atuais

Os resultados confirmam que, mesmo para alunos com nível médio a alto de familiaridade (Figura 3.3), a experiência com simuladores tradicionais é marcada por atritos.

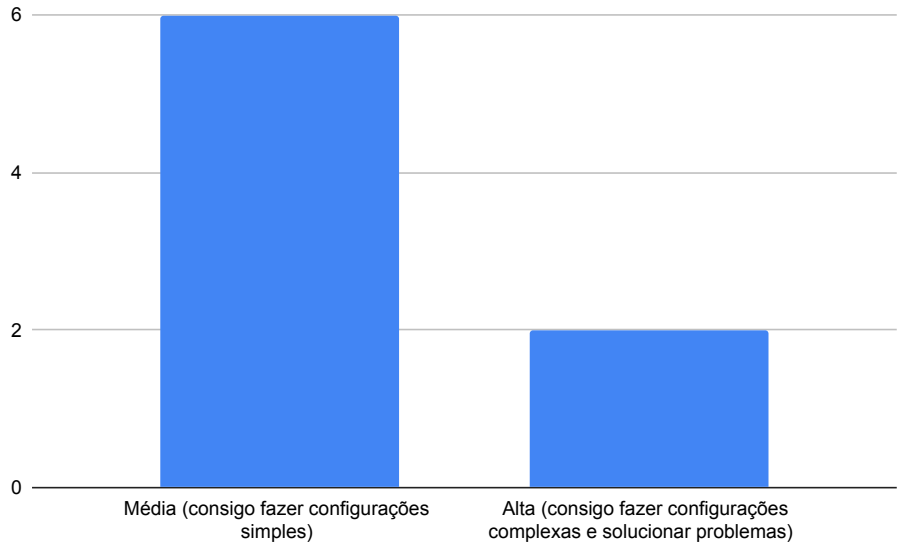


Figura 3.3: Autodeclaração de nível de familiaridade com configuração via CLI.
Fonte: Dados da pesquisa (2025).

Ao serem questionados sobre aspectos específicos da usabilidade e didática (Figura 3.4), houve consenso de que a interface de linha de comando não é intuitiva para iniciantes e que o processo de configuração é repetitivo. O dado mais crítico, no entanto, refere-se à falta de orientação: a maioria dos respondentes concorda que falta *feedback* detalhado quando uma configuração falha.

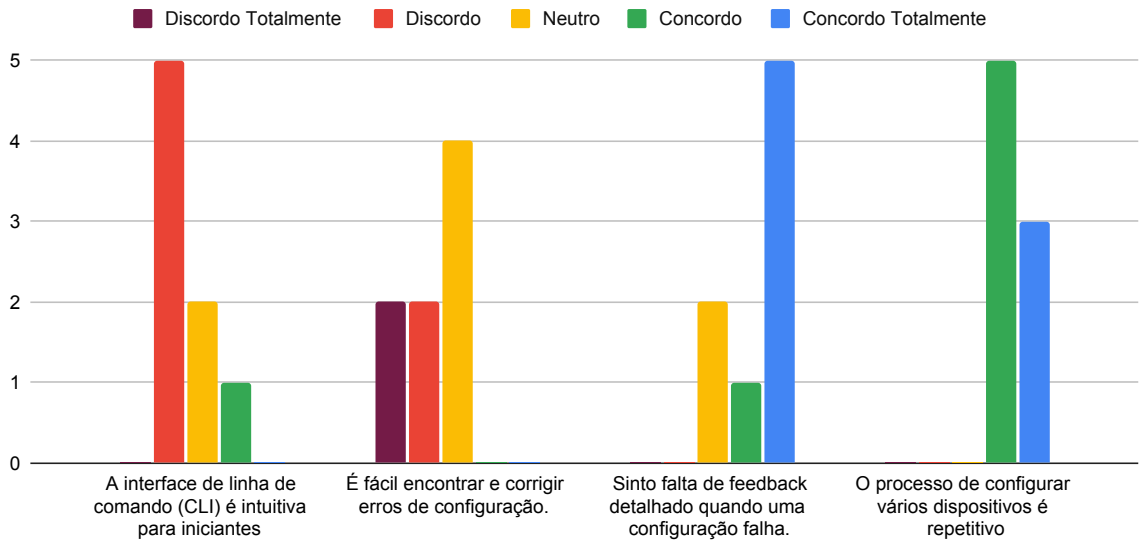


Figura 3.4: Percepção dos estudantes quanto às dificuldades e limitações dos simuladores atuais.

Fonte: Dados da pesquisa (2025).

Essa percepção foi corroborada pelas questões qualitativas, onde os relatos (Figura 3.5) apontaram o diagnóstico de erros e a sintaxe complexa como os maiores obstáculos. Um dos participantes resumiu o problema central: *"Quando acontece erro/falha, fica difícil identificar a causa na maioria das vezes"*.

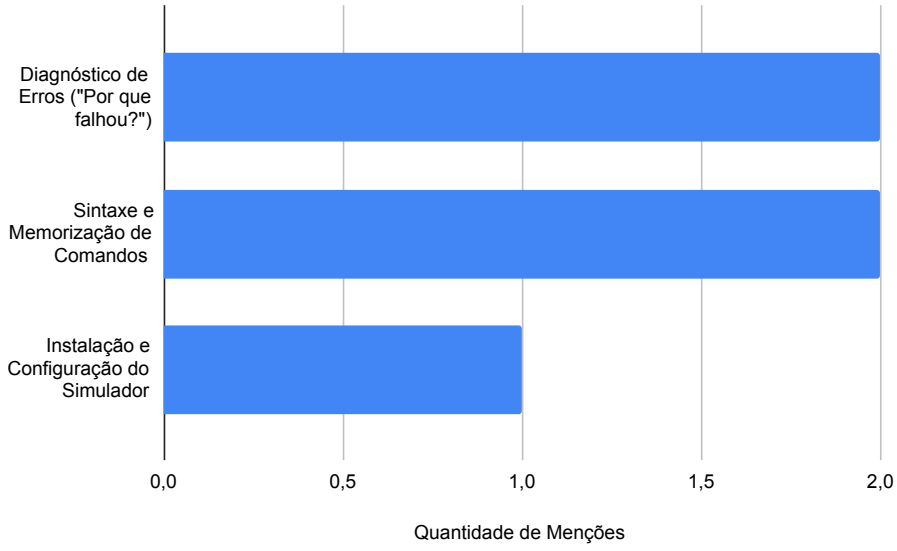


Figura 3.5: Categorização dos desafios relatados nas questões abertas.
Fonte: Dados da pesquisa (2025).

A Demanda por Inteligência Artificial e Suporte

A pesquisa revelou que os estudantes já buscam soluções externas para suprir as lacunas dos simuladores. Conforme a Figura 3.6, 87,5% dos alunos já utilizam ferramentas de IA Generativa (como *ChatGPT*) em seus estudos, seja frequentemente ou ocasionalmente.

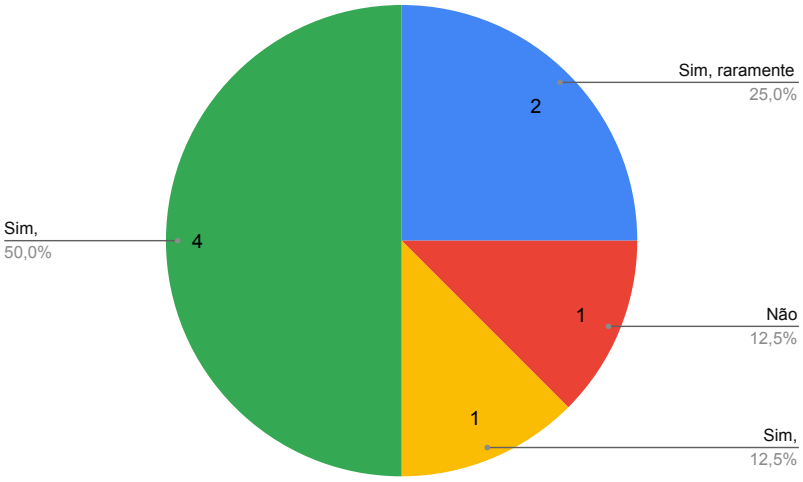


Figura 3.6: Frequência de uso de ferramentas de IA para estudo de redes.
Fonte: Dados da pesquisa (2025).

O objetivo desse uso não é apenas gerar respostas prontas. Como detalhado na Figura 3.7, os alunos buscam a IA principalmente para entender conceitos, seguir guias passo a passo e melhorar a documentação, funções que um simulador estático não oferece.

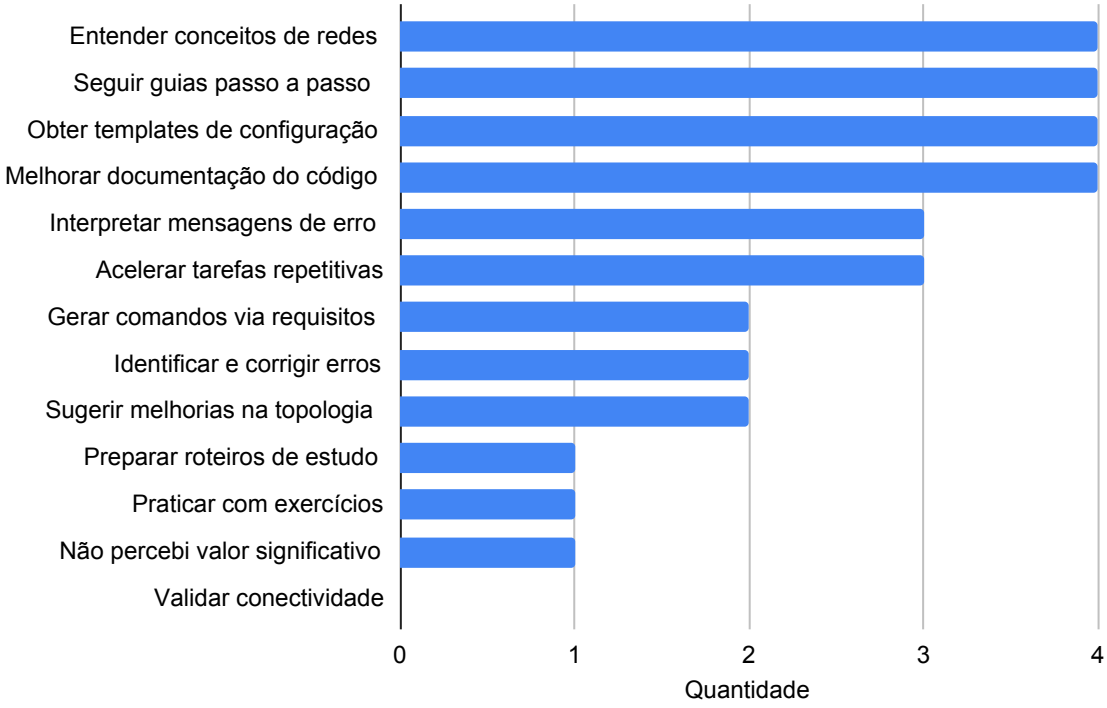


Figura 3.7: Principais benefícios percebidos no uso de IA para o aprendizado de redes.
Fonte: Dados da pesquisa (2025).

Validação da Proposta e Expectativas Futuras

Diante desse cenário, a aceitação de uma nova plataforma que integre esses recursos foi expressiva. A quase totalidade dos participantes (87,5%) acredita que uma solução com suporte inteligente facilitaria o aprendizado (Figura 3.8).

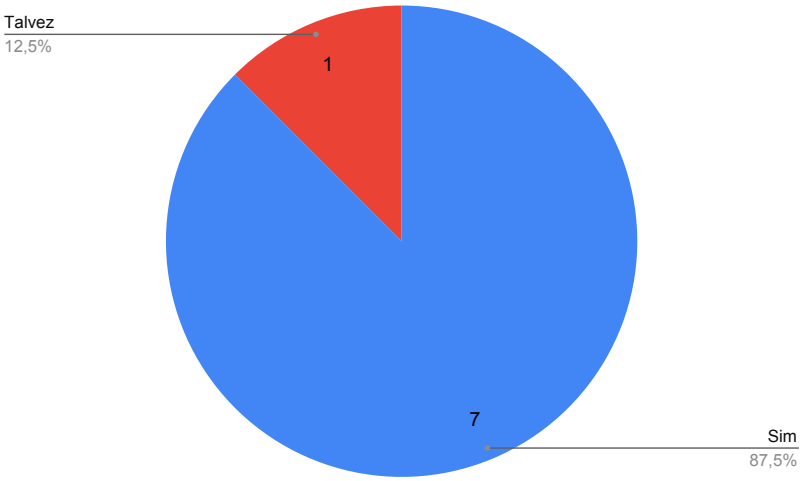


Figura 3.8: Expectativa sobre o impacto de uma plataforma com suporte inteligente no aprendizado.

Fonte: Dados da pesquisa (2025).

Ao elencar as funcionalidades mais desejadas (Figura 3.9), a configuração guiada e a análise de cenário via IA figuraram entre as prioridades.

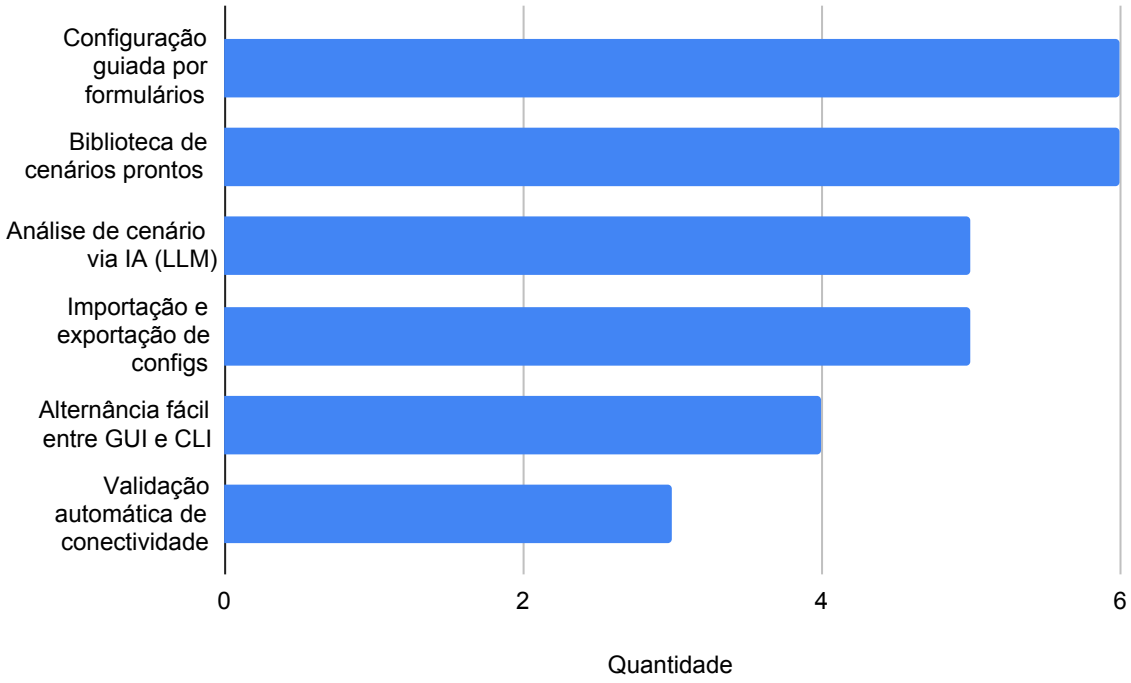


Figura 3.9: Funcionalidades consideradas mais úteis para um novo simulador.
Fonte: Dados da pesquisa (2025).

Por fim, os participantes puderam oferecer sugestões abertas para a evolução da ferramenta. Conforme a Figura 3.10, embora uma parcela não tenha indicado demandas adicionais, houve pedidos específicos por suporte a múltiplos fabricantes (*multi-vendor*) e melhorias na interface.

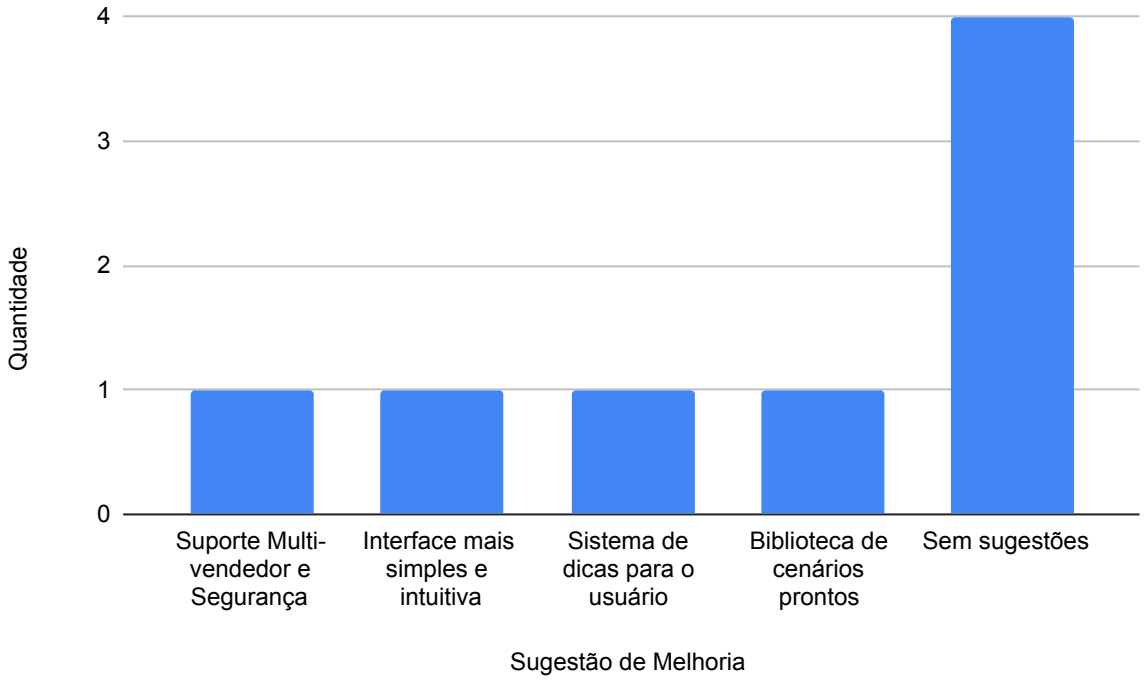


Figura 3.10: Sugestões de melhorias e novas funcionalidades apontadas pelos participantes.

Fonte: Dados da pesquisa (2025).

3.2.2 Definição dos objetivos da solução

Com base nas lacunas identificadas, definem-se os objetivos prioritários do artefato: prover configuração baseada em formulários (*form-based*) com exibição de diferenças em tempo real entre GUI e CLI; permitir exportação por dispositivo e por cenário; implementar análise assistida por LLM com salvaguardas pedagógicas; garantir operação *web-based* responsiva; assegurar transparência e privacidade de dados; e permitir avaliabilidade pedagógica.

Entregáveis: lista priorizada de objetivos e requisitos funcionais e não-funcionais (detalhados no Cap. 4).

3.2.3 Design e desenvolvimento

O artefato foi concebido e implementado de forma incremental, seguindo uma abordagem ágil. O design da interface de usuário (UI) foi realizado de maneira integrada ao desenvolvimento, utilizando uma filosofia de prototipagem rápida diretamente no código.

1. **Tecnologias:** A plataforma foi construída com *Next.js* (baseado em *React*) e a linguagem de programação *TypeScript*, cuja tipagem estática assegura a integridade do código,, garantindo uma arquitetura robusta e manutenível. A estilização foi feita com *Tailwind CSS*, um *framework utility-first* que permitiu a criação de uma interface responsiva e customizada de forma ágil, diretamente na marcação dos componentes.

2. **Desenvolvimento Incremental:** O projeto foi dividido em estágios lógicos para gerenciar a complexidade e permitir validações contínuas:

    - **Estágio 1 (Núcleo Visual):** Implementação da área de trabalho interativa com a biblioteca *React Flow*, permitindo a adição e conexão de dispositivos.
    - **Estágio 2 (Configuração via GUI):** Desenvolvimento dos formulários de configuração para cada dispositivo, integrados ao sistema de gerenciamento de estado centralizado (*React Context*).
    - **Estágio 3 (Simulação de CLI):** Criação do emulador de terminal com parser de comandos, garantindo a sincronia entre as configurações realizadas via CLI e GUI.
    - **Estágio 4 (Integração com IA):** Construção da API para orquestrar a comunicação com o LLM (*Gemini*) e desenvolvimento dos componentes para exibir a análise de forma estruturada ao usuário.

Entregáveis: arquitetura de *software*, interface de usuário responsiva, base de código documentada e casos de teste funcionais.

3.2.4 Demonstração

A demonstração evidenciará o funcionamento do artefato em três cenários didáticos: (A) configuração de VLANs e acesso entre redes; (B) implementação de roteamento dinâmico (RIP/OSPF); e (C) aplicação de listas de controle de acesso (ACLs). Para cada cenário, serão apresentados os passos, artefatos gerados (configurações e exportações) e o relatório de análise/explicações do módulo assistido.

Entregáveis: roteiros, evidências (capturas de tela, trechos de código) e o relatório de demonstração (Cap. 5).

3.2.5 Avaliação

A etapa de avaliação é crítica para verificar se o artefato atinge os objetivos de usabilidade e utilidade pedagógica. Nesta pesquisa, optou-se por uma estratégia de avaliação mista, dividida em validação técnica e inspeção de usabilidade.

A avaliação foi desenhada para responder às seguintes questões de pesquisa:

- **QP1:** O artefato é tecnicamente viável para uso em ambientes com recursos limitados (latência e processamento)?
- **QP2:** A interface proposta atende aos critérios heurísticos de usabilidade estabelecidos para ferramentas educacionais?

Para responder à QP1, foram realizados testes de caixa-preta focados na execução de tarefas críticas (configuração e análise IA) em ambiente de produção real. Para a QP2, adotou-se o método de Avaliação Heurística de Nielsen (1994). Este método consiste na inspeção sistemática da interface para identificar violações de princípios de usabilidade conhecidos, baseando-se nas 10 Heurísticas de Nielsen. Esse conjunto de dez regras foi consolidado pelo autor para sintetizar os princípios mais abrangentes de interação humano-computador, como visibilidade do estado do sistema, consistência e prevenção de erros,

servindo como um guia geral capaz de detectar a maioria das falhas de design antes de testes com usuários.

Diferente de testes com usuários finais em larga escala, que exigem aprovação ética complexa e logística extensa, a inspeção heurística permite identificar a maioria dos problemas graves de usabilidade na fase de desenvolvimento do protótipo, garantindo que o artefato esteja maduro antes de ser introduzido em sala de aula (Rubin and Chisnell, 2008). Os dados coletados (tempos de resposta, taxas de sucesso na tarefa e violações heurísticas) foram analisados para permitir iterações sobre o design do artefato.

Entregáveis: protocolo do estudo, dados da inspeção e relatório de resultados (Cap. 6).

3.2.6 Comunicação

A etapa de Comunicação encerra o ciclo DSR e consiste na disseminação do conhecimento produzido. Neste trabalho, essa etapa não se limita apenas à escrita desta monografia, mas estende-se à disponibilização irrestrita dos artefatos construídos para a comunidade acadêmica, visando a transparência e a continuidade da pesquisa.

Para garantir a reprodutibilidade do estudo, o código-fonte completo da aplicação, incluindo a arquitetura, os *prompts* de sistema utilizados e as instruções de implantação (*deploy*), foi documentado e hospedado em um repositório de acesso aberto na plataforma GitHub[1]. Adicionalmente, a aplicação em versão de produção, utilizada nos testes descritos no Capítulo 5, permanece acessível *online* para demonstração. Os roteiros de teste e as topologias utilizadas na avaliação também foram consolidados nos Apêndices deste trabalho, permitindo que outros pesquisadores repliquem os cenários para validação futura.

Entregáveis: texto final do trabalho, repositório público de código-fonte, aplicação acessível *online* e materiais de reprodutibilidade (apêndices).

3.3 Instrumentos e Tecnologias da Pesquisa

Para a operacionalização das etapas da DSR, foram utilizados instrumentos específicos de coleta de dados e desenvolvimento:

- **Coleta de Dados Preliminar:** Questionário estruturado aplicado via Google Forms, contendo perguntas fechadas em escala Likert de 5 pontos para mensurar a percepção dos estudantes, além de campos abertos para coleta qualitativa de dores e necessidades.

- **Prototipagem e Design de Interface:** Adotou-se a abordagem de *Code-First Prototyping* (Prototipagem em Código). Em vez de utilizar ferramentas de desenho estático intermédias, a interface foi desenvolvida e iterada diretamente no navegador utilizando o *framework Tailwind CSS*. Essa escolha permitiu validar a responsividade e a usabilidade técnica em tempo real, acelerando o ciclo de desenvolvimento.

- **Ambiente de Desenvolvimento:** A construção do artefato utilizou a *stack React/Next.js*. Esta escolha metodológica justifica-se pela necessidade de Componentização (reuso de código) e pela capacidade de Renderização Híbrida, essencial para o desempenho em dispositivos móveis.

[1]Disponível em: `https://github.com/GustavoPBDS/cisco_device_config`

- **Análise Automática:** A integração via API com o modelo *Gemini (Google)* foi instrumentada utilizando *System Prompts* (comandos de sistema) projetados e refinados iterativamente para garantir que a IA atue como um tutor socrático.

3.4 Considerações Finais do Capítulo

Este capítulo apresentou a estrutura metodológica que orienta o desenvolvimento do artefato, ancorada na DSR. Foram descritas as etapas do processo de DSR, seus principais entregáveis e evidências, bem como os instrumentos de coleta de dados e as tecnologias de suporte utilizadas na pesquisa. Também foram estabelecidas as questões de pesquisa e a estratégia de avaliação, delineando como a viabilidade técnica, a usabilidade e a utilidade pedagógica do simulador serão analisadas ao longo do estudo.

Ao definir esse enquadramento metodológico, cria-se um roteiro claro para a condução das etapas seguintes. No próximo capítulo (Cap. 4), são detalhadas a arquitetura, as decisões de projeto e a implementação do artefato, correspondentes à fase de **design e desenvolvimento** da DSR. Em seguida, o Cap. 5 apresenta a **demonstração** do simulador em cenários práticos de configuração de redes, e o Cap. 6 conduz a **avaliação** sistemática do artefato com base nos métodos e métricas definidos neste capítulo. Dessa forma, a metodologia aqui exposta atua como o elo entre a fundamentação teórica e a materialização do protótipo, assegurando rigor e rastreabilidade ao longo de todo o processo de pesquisa.

Capítulo 4

Design e Desenvolvimento do Artefato

Este capítulo descreve o processo de concepção, arquitetura e implementação do artefato tecnológico proposto: um simulador de redes com análise de configuração assistida por Inteligência Artificial. As decisões de projeto aqui detalhadas são diretamente derivadas dos requisitos estabelecidos na fundamentação teórica (Capítulo 2) e alinhadas aos objetivos definidos na introdução (Capítulo 1), seguindo a metodologia de *Design Science Research*.

4.1 Requisitos

A definição dos requisitos do sistema fundamenta-se na síntese das investigações realizadas nos capítulos anteriores. As lacunas mapeadas na **Fundamentação Teórica** (Capítulo 2), especificamente a curva de aprendizado acentuada das interfaces de linha de comando (CLI), a escassez de ferramentas acessíveis em dispositivos móveis e a ausência de *feedback* pedagógico imediato, ditaram o escopo funcional da solução.

Adicionalmente, os requisitos foram derivados e validados com base nos dados empíricos coletados com estudantes (Capítulo 3), os quais evidenciaram limitações na capacidade de depuração autônoma de erros de configuração, bem como a recorrência de tarefas operacionais de natureza repetitiva como um fator de sobrecarga do processo de aprendizagem.

Nesse contexto, os requisitos foram sistematicamente organizados em *Requisitos Funcionais* (RFs), que especificam o comportamento observável do sistema, e *Requisitos Não-Funcionais* (RNFs), que definem atributos de qualidade, com ênfase na redução da carga cognitiva, na usabilidade, na acessibilidade e na ampliação do alcance do artefato.

4.1.1 Requisitos Funcionais

- **RF-01: Criação e Manipulação de Topologias:** O sistema deve permitir que o usuário crie uma topologia de rede adicionando dispositivos (PCs, *Switches*, Roteadores) a uma área de trabalho e conectando-os por meio de cabos virtuais.

- **RF-02: Configuração via Interface Gráfica (GUI):** O sistema deve fornecer formulários intuitivos para a configuração dos parâmetros de cada tipo de dispositivo. Por exemplo, para um switch, deve ser possível configurar *hostname, VLANs, Channel Groups, Spanning Tree* e interfaces de gerenciamento.

- **RF-03: Configuração via Linha de Comando (CLI):** O sistema deve oferecer um emulador de terminal com uma interface de linha de comando (CLI) que simule a sintaxe e os modos de operação de equipamentos Cisco (exec, privilegiado, configuração global, etc.).
- **RF-04: Análise de Configuração por IA:** O sistema deve possuir uma funcionalidade que envia a configuração do cenário atual para um modelo de linguagem de larga escala (LLM), que, por sua vez, deve analisar a configuração, identificar problemas, sugerir melhorias e fornecer um resumo.
- **RF-05: Exportação de Configurações:** O sistema deve permitir ao usuário exportar as configurações de um dispositivo individual para um arquivo de texto, no formato de comandos CLI, garantindo interoperabilidade com equipamentos reais.
- **RF-06: Sincronização entre GUI e CLI:** As alterações realizadas via formulários (GUI) devem refletir no estado do dispositivo acessível via CLI, e vice-versa, garantindo um modelo de dados unificado.

4.1.2 Requisitos Não Funcionais

- **RNF-01: Usabilidade:** A interface deve ser clara, intuitiva e responsiva, minimizando a curva de aprendizado para usuários iniciantes. A ferramenta deve fornecer *feedback* visual para as ações do usuário (ex: notificações de sucesso ou erro).
- **RNF-02: Desempenho:** A aplicação *web* deve ter um tempo de carregamento rápido e as interações na interface (arrastar dispositivos, abrir menus) devem ser fluidas e sem atrasos perceptíveis.
- **RNF-03: Manutenibilidade:** O código-fonte deve ser bem estruturado, utilizando componentes reutilizáveis e tipagem estrita (*TypeScript*) para facilitar a manutenção e a adição de novas funcionalidades no futuro.
- **RNF-04: Acessibilidade:** Por ser uma aplicação baseada em navegador, deve ser acessível a partir de diferentes sistemas operacionais sem a necessidade de instalação, bastando um navegador *web* moderno.

4.2 Visão Geral da Arquitetura

Para atender aos requisitos, optou-se por uma arquitetura de aplicação web moderna, utilizando o *framework*Next.js. A solução foi organizada em duas camadas principais: a interface cliente (*front-end*) e um *backend* mínimo, responsável por orquestrar a análise via LLM, conforme ilustrado na Figura 4.1. A interface principal, que contém o editor de topologias, utiliza a biblioteca **React Flow** para a renderização visual e interativa dos dispositivos e suas conexões. O estado global de todo o cenário é gerenciado de forma centralizada por um contexto global, implementado com a API de Contexto do React, servindo como a fonte única da verdade para todos os componentes.

A escolha do **Next.js** como *framework*principal não se deu apenas por sua popularidade no ecossistema React, mas por duas razões estratégicas. Primeiramente, sua estrutura de "Rotas de API"(*API Routes*) permite a criação de um *backend-for-frontend* (BFF) leve e integrado. Isso é crucial para o **RF-04 (Análise por IA)**, pois centraliza a

lógica de comunicação com a API externa do LLM (Gemini) em um ambiente de servidor controlado, protegendo chaves de API e facilitando o pré-processamento dos *prompts*. Em segundo lugar, a arquitetura Next.js oferece capacidade nativa de renderização no servidor (SSR) e geração estática (SSG), o que viabiliza futuras expansões, como a criação de uma biblioteca pública de cenários de rede pré-renderizados.

Para a renderização da topologia (atendendo ao **RF-01**), a biblioteca **React Flow** foi selecionada em detrimento de alternativas de *canvas* tradicionais por sua natureza declarativa. O React Flow gerencia nativamente interações complexas como zoom, panorâmica (*pan*) e o estado de conexões (arestas), permitindo que o desenvolvimento se concentre na lógica de negócios dos "nós"(os dispositivos de rede) em vez de na complexa geometria de renderização. Por fim, o gerenciamento de estado via **Context API** do React foi uma decisão deliberada para atender ao **RNF-03 (Manutenibilidade)**. Em vez de introduzir bibliotecas externas complexas (como Redux), o Contexto provê a simplicidade necessária para manter a consistência dos dados.

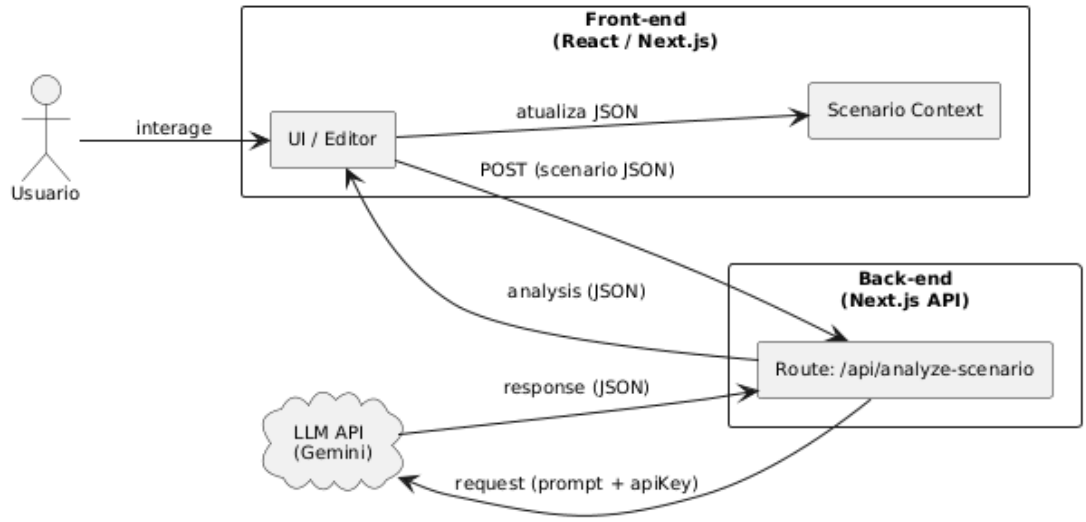


Figura 4.1: Diagrama de arquitetura da aplicação, ilustrando a interação entre front-end, back-end e a API externa do LLM.
Fonte: Elaborado pelo autor, utilizando PlantUML.

4.3 Modelo de Dados e Gerenciamento de Estado

O coração do simulador é um modelo de dados centralizado que representa o estado de toda a topologia, gerenciado pelo contexto *ScenarioContextProvider*. A principal estrutura de dados responsável por modelar cada dispositivo é representada pela entidade *NetworkDeviceData*, ilustrada no diagrama da Figura 4.2.

Essa estrutura foi projetada para ser flexível e extensível, permitindo armazenar atributos essenciais, como tipo, rótulo e portas físicas, bem como informações configuráveis, incluindo endereçamento *IPv4, VLANs, gateways* e parâmetros específicos de cada interface. Ela também contém um mapeamento das conexões entre dispositivos, possibilitando a sincronização entre a representação gráfica e o estado lógico do cenário.

O código completo dessa estrutura, implementado em TypeScript, encontra-se no Apêndice B.1.

```
Structure NetworkDeviceData
    type: deviceType
    label: string
    position: (x, y)
    ports: list of portId
    portsConnected: map(portId → Connection)

    config (optional):
        ipv4 settings
        vlan settings
        spanning-tree settings
        channel-groups
        interface configurations
        routing protocols (OSPF, BGP, static routes)
        ACL rules
        DHCP exclusions
```

Figura 4.2: Representação abstrata da estrutura de dados NetworkDeviceData.

As principais responsabilidades do gerenciamento de estado são:

- **Armazenar os dispositivos** e suas conexões em uma estrutura de dados do tipo `Map`, onde a chave é o ID do nó.
- **Fornecer operações de mutação** para a interface, como adição/remoção de dispositivos (`addPc`, `addRouter`, `removeDevice`), conexão/desconexão de portas (`connectDevices`) e atualização das configurações (`updateDeviceConfig`).
- **Gerenciar alocações DHCP simuladas**, através das funções `requestDhcpLease` e `releaseDhcpLease`, permitindo que PCs obtenham endereçamento IP dinamicamente de um roteador configurado.
- **Armazenar o resultado da análise** retornada pelo LLM na função *storeAnalysisResult*, para que possa ser exibido na interface.

4.3.1 O Desafio da Sincronia (RF-06) e o Padrão "*Single Source of Truth*"

O requisito mais complexo do artefato é o **RF-06**: garantir que uma alteração feita nos formulários da GUI (ex: adicionar uma VLAN) seja instantaneamente refletida no emulador de CLI e vice-versa. A arquitetura de gerenciamento de estado foi projetada especificamente para resolver esse problema, tratando tanto a GUI quanto a CLI como "clientes"do mesmo estado centralizado. O fluxo de dados é unidirecional:

1. **Ação na GUI:** Quando o usuário preenche um formulário (ex: `SwitchForm`) e salva, o componente invoca a função `updateDeviceConfig`, que atualiza o objeto `NetworkDeviceData` no estado global.
2. **Ação na CLI:** Quando o usuário digita um comando (ex: `switchport mode trunk`), o *parser* de comandos não manipula o estado localmente. Ele interpreta a intenção e invoca a mesma função `updateDeviceConfig` do contexto.

Dessa forma, ambos os componentes reagem às mudanças na "fonte única da verdade" (*Single Source of Truth*), garantindo consistência absoluta sem a necessidade de mecanismos complexos de sincronização bilateral.

4.4 Implementação dos Módulos Principais

A implementação do artefato seguiu as decisões de arquitetura, utilizando um conjunto de tecnologias modernas do ecossistema do JavaScript e TypeScript.

4.4.1 Front-end: Editor Visual e Menus de Configuração

O editor visual, componente central da aplicação, foi implementado com a biblioteca React Flow para renderizar nós e arestas interativas. Um tipo de nó personalizado, `CustomNode`, foi criado para exibir os dispositivos de rede de forma adequada. A interface de usuário foi projetadasimultaneamenteforma simultânea, utilizando o *framework Tailwind CSS* para estilizar componentes e criar um layout responsivo diretamente no código. A adição de novos dispositivos à topologia ocorre via "arrastar e soltar"(*drag-and-drop*).

A Figura 4.3 ilustra os principais componentes da interface de usuário desenvolvida.

Figura 4.3: Ilustração dos principais componentes visuais implementados: a área de trabalho para a topologia e o painel lateral para configuração de dispositivos.

A escolha pelo ***Tailwind CSS*** está diretamente alinhada aos requisitos não funcionais. Para o **RNF-04 (Acessibilidade)**, a abordagem *utility-first* permitiu a rápida implementação de um design responsivo (*mobile-first*), crucial para o objetivo pedagógico de ampliar o acesso a dispositivos móveis. Para o **RNF-01 (Usabilidade)**, o *Tailwind* agilizou a prototipação de interfaces limpas e consistentes. Adicionalmente, a decisão de usar ícones gerados por IA (Sora) em vez de pacotes de ícones padrão teve um propósito estético e pedagógico: criar uma identidade visual amigável e menos intimidante para estudantes iniciantes, diferenciando-se dos simuladores corporativos tradicionais.

Quando um dispositivo é selecionado, um menu de contexto abre formulários de configuração específicos para aquele tipo de equipamento. O formulário de *switch*, por exemplo, encapsulado no componente `SwitchForm`, traduz as opções selecionadas pelo usuário para a estrutura de dados `NetworkDeviceData`. Validações essenciais são aplicadas diretamente no formulário antes de invocar a função de atualização do estado.

4.4.2 Mecanismo de Interface de Linha de Comando (CLI)

Para oferecer uma experiência de configuração autêntica, foi implementado um simulador de CLI (Figura 4.4) inspirado no sistema operacional Cisco IOS através do *custom hook* `useCiscoCli`. Este *hook* oferece:

- **Modos de operação hierárquicos:** exec, privilegiado, configuração global, configuração de interface, sub-interface e VLAN.
- **Parser de comandos:** Um analisador sintático simplificado que interpreta comandos relevantes.
- **Recursos de usabilidade:** Autocomplete de comandos com a tecla "Tab"e histórico de comandos.

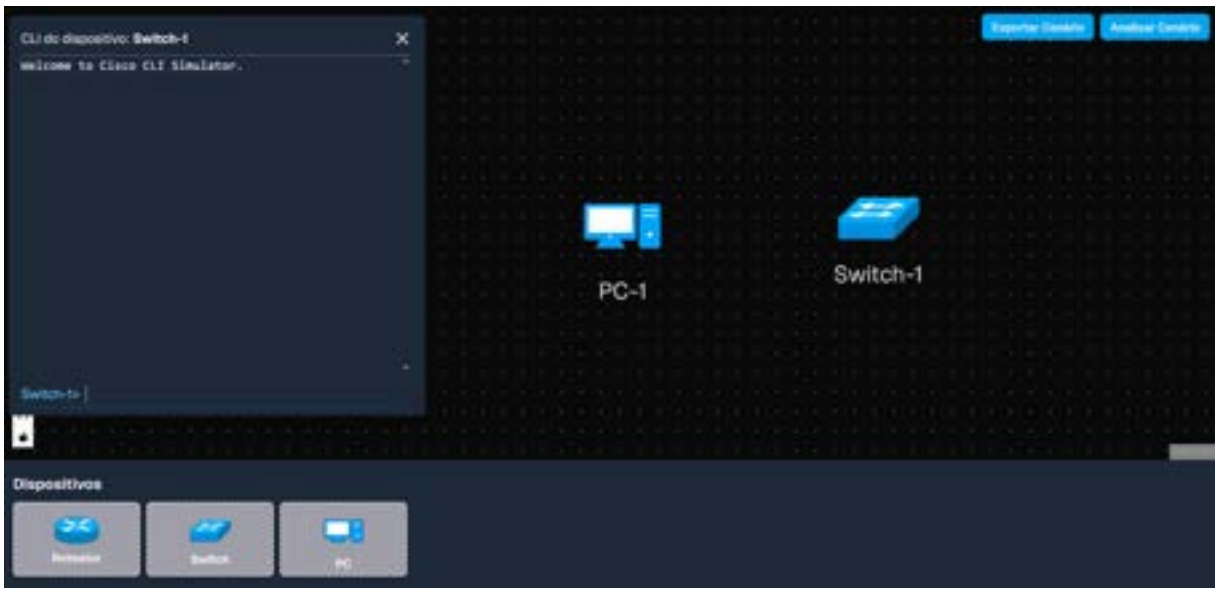


Figura 4.4: Componente de terminal implementado, simulando a interface de linha de comando.

Crucialmente, a CLI não opera de forma isolada. Cada comando que modifica a configuração do dispositivo invoca a função `updateDeviceConfig` do contexto, garantindo a consistência do estado com a GUI.

4.4.3 Módulo de Análise e Exportação

A funcionalidade de análise por IA foi implementada utilizando uma Rota de API do Next.js. O fluxo cliente-servidor funciona da seguinte maneira:

1. A interface do usuário dispara a função `analyzeScenario`, que envia a configuração do cenário atual para a Rota de API.
2. A Rota de API interage com o modelo de linguagem Gemini, enviando um *prompt* estruturado.
3. O *front-end* recebe a resposta JSON, a armazena no estado global e a exibe para o usuário em um componente modal (Figura 4.5).

```
Voce e um consultor especialista em redes Cisco.
Recebera um array de objetos { device, config } onde config e a
    configuracao em CLI do dispositivo.
Sempre analise procurando:
- VLANs criadas mas nao utilizadas
- PCs sem IP ou com IP repetido
- Switch sem ip default-gateway
- Roteador sem interfaces IP ou rotas
- Possiveis loops de spanning-tree
- Falta de redundancia ou ma pratica de configuracao
Responda SEMPRE em JSON, um array de objetos no formato abaixo (um
    objeto por dispositivo):
[
  {
    "device": "nome do dispositivo",
    "problemas": [ "descricao do problema 1", "descricao do problema 2"
    ],
    "melhorias": [ "sugestao 1", "sugestao 2" ],
    "resumo": "Resumo geral do dispositivo"
  }
]
Imprima somente o JSON. Nao escreva texto extra.
Configuracoes para analise:
${JSON.stringify(scenario_configuration)}
```

Listing 4.1: Exemplo de prompt de sistema utilizado para guiar o LLM a gerar uma análise estruturada.

4.4.4 Engenharia de Prompt e Mitigação de Riscos do LLM

A eficácia do módulo de análise (RF-04) depende da capacidade de extrair respostas precisas e estruturadas de um modelo probabilístico. Para tal, aplicou-se um processo de Engenharia de Prompt iterativo com três técnicas principais (visíveis na Listagem 4.1):

- **Definição de Persona (*Persona Scoping*):** A instrução "Você é um consultor especialista..."ativa o contexto de conhecimento relevante no modelo.
- **Delimitação da Tarefa (*Task Scoping*):** A lista explícita de itens a procurar (ex: "PCs sem IP") foca a análise nos erros pedagógicos comuns mapeados na Fundamentação Teórica, evitando digressões.
- **Formatação de Saída (*Output Formatting*):** A restrição rígida ao formato JSON permite que a aplicação processe a resposta programaticamente.s

Para mitigar riscos de alucinações (informações incorretas geradas pela IA), a plataforma adota uma abordagem de *Human-in-the-Loop*: as sugestões não são aplicadas automaticamente. O sistema atua como um mentor que aconselha, cabendo ao estudante avaliar e executar as correções, o que reforça o pensamento crítico.

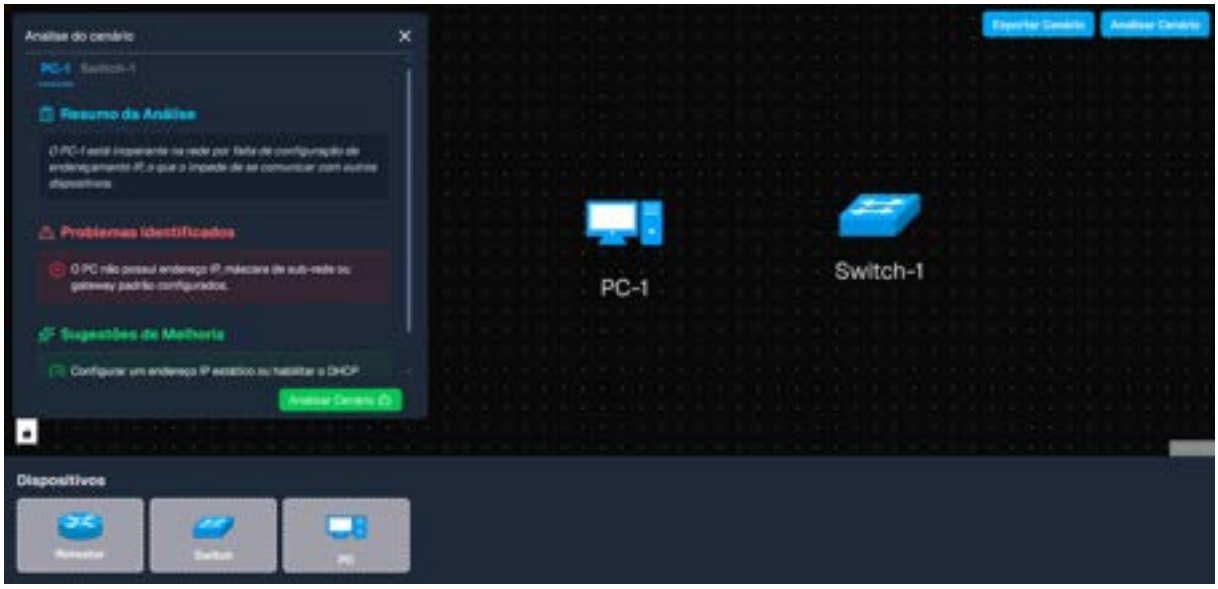


Figura 4.5: Design do componente modal desenvolvido para apresentar ao usuário o *feedback* da análise de configuração.

4.5 Decisões de Projeto e Trade-offs

Durante o desenvolvimento, decisões estratégicas resultaram em *trade-offs* importantes para viabilizar o projeto dentro do escopo de um TCC:

- **Persistência em Memória vs. Escopo:** A decisão de manter o estado apenas em memória (*React State*) em vez de implementar um banco de dados persistente permitiu focar os recursos de desenvolvimento na inovação principal: o módulo de análise por IA. A consequência é que o artefato atual não salva projetos entre sessões, sendo ideal para práticas de laboratório pontuais, mas limitado para projetos de longo prazo.
- **Simulação de Estado vs. Simulação de Pacotes:** Diferentemente de emuladores como o GNS3, que simulam o tráfego real de pacotes (consumindo muitos recursos), este artefato realiza uma *simulação de estado de configuração*. Por exemplo, o sistema calcula logicamente se um PC deve receber um IP via DHCP, mas não simula a troca dos pacotes DORA (*Discover, Offer, Request, Acknowledge*). Isso garante a leveza necessária para rodar no navegador (acessibilidade), mantendo o foco pedagógico na análise da configuração estática, onde reside a maioria dos erros de iniciantes.

4.6 Verificação de Requisitos e Validação

Antes da avaliação com usuários (detalhada no Capítulo 6), uma etapa de verificação técnica foi conduzida para assegurar o cumprimento dos requisitos funcionais, utilizando uma abordagem de Testes Baseados em Cenários (*Scenario-Based Testing*):

- **Verificação de GUI e Exportação (RF-01, RF-02, RF-05):** Topologias de teste foram criadas e configuradas exclusivamente via GUI. Os arquivos gerados

pela função de exportação foram inspecionados manualmente, confirmando que os comandos CLI gerados correspondiam fielmente às configurações visuais.

- **Verificação de Sincronia (RF-03, RF-06):** A consistência do modelo *Single Source of Truth* foi validada através de fluxos cruzados: alterações de *hostname* e VLANs realizadas via comandos CLI foram imediatamente verificadas nos formulários da GUI e vice-versa, confirmando a atualização bidirecional do estado.
- **Validação da Análise IA (RF-04):** Utilizou-se a técnica de Injeção de Falhas (*Fault Injection*). Cenários contendo erros lógicos propositais (como sobreposição de sub-redes e loops de roteamento) foram submetidos ao módulo de análise. O sistema foi capaz de identificar corretamente as falhas e sugerir as correções apropriadas, validando a eficácia da engenharia de prompt aplicada.

4.7 Considerações do Capítulo

Este capítulo apresentou, de forma detalhada, o processo de design e desenvolvimento do artefato proposto, contemplando a especificação dos requisitos funcionais e não funcionais, a definição da arquitetura da solução, o modelo de dados centralizado, os principais módulos implementados e as decisões de projeto adotadas ao longo da construção do simulador. Também foram discutidos os principais *trade-offs* assumidos para viabilizar o artefato dentro do escopo do trabalho, bem como as estratégias de verificação técnica empregadas para assegurar a conformidade com os requisitos.

A apresentação da arquitetura e da implementação evidenciou como os objetivos de acessibilidade, usabilidade, sincronização entre GUI e CLI e análise assistida por IA foram operacionalizados de forma concreta no sistema. Além disso, a verificação por meio de testes baseados em cenários forneceu evidências iniciais da corretude funcional do artefato e da efetividade do módulo de análise automática.

No próximo capítulo (Cap. 5), o artefato desenvolvido é apresentado em operação por meio de cenários didáticos representativos, nos quais são demonstradas, passo a passo, as funcionalidades de configuração, análise por IA e exportação de configurações. Essa demonstração tem como objetivo evidenciar, de maneira prática, como as soluções de projeto descritas neste capítulo se materializam no uso real do simulador.

Capítulo 5

Demonstração

Este capítulo corresponde à etapa de **Demonstração** da metodologia de *Design Science Research*, tendo como finalidade evidenciar a aplicabilidade do artefato desenvolvido em contextos reais de ensino de redes de computadores. Mais do que comprovar a correta execução de comandos ou a simples conectividade técnica, a demonstração busca evidenciar o potencial pedagógico da plataforma como um mecanismo de *tutoria cognitiva* ao longo do processo de aprendizagem.

Ao longo de três cenários didáticos progressivos, é simulada a trajetória de um estudante que, ao incorrer em erros lógicos recorrentes — frequentemente silenciosos em simuladores tradicionais — passa a receber orientação imediata, explicativa e contextualizada por meio do módulo de IA. Dessa forma, a demonstração centra-se na capacidade do sistema de converter falhas de configuração, que usualmente resultariam em frustração e tentativa-e-erro pouco produtivos, em oportunidades estruturadas de aprendizagem supervisionada e reflexiva.

5.1 Cenários Representativos de Uso

Foram selecionados três cenários didáticos representativos, contemplando conceitos fundamentais do ensino de redes de computadores e que, à luz da literatura e dos dados empíricos apresentados no Capítulo 3, constituem fontes recorrentes de dificuldade para estudantes iniciantes. Cada cenário foi intencionalmente configurado com erros lógicos típicos, de modo a avaliar a precisão, a profundidade e a clareza do *feedback* gerado pelo módulo de LLM.

5.1.1 Cenário A: Configuração de VLANs e Roteamento Inter-VLAN

Justificativa Pedagógica: Este cenário é fundamental para ilustrar a segmentação de redes (Camada 2) e a necessidade de um dispositivo de Camada 3 para permitir a comunicação entre segmentos distintos. O erro clássico explorado aqui é o esquecimento da configuração do "tronco" (*trunk*) ou a configuração incorreta de sub-redes, o que impede a comunicação sem fornecer uma causa clara ao aluno.

Execução e Evidência da Funcionalidade:

1. **Construção da Topologia e Configuração GUI:** A topologia com um roteador, um switch e dois PCs foi montada usando a interface de "arrastar e soltar". No

Switch-1, a GUI foi utilizada para criar as VLANs 10 (VENDAS) e 20 (TI). As portas dos PCs foram corretamente associadas a essas VLANs ($Fa0/1$ para VLAN 10, $Fa0/2$ para VLAN 20) através dos formulários visuais.

Figura 5.1: Topologia do Cenário A e configuração de VLANs via GUI no Switch-1.

2. **Introdução de Erros Comuns para Teste:** Para validar a capacidade de análise da IA, dois erros lógicos foram introduzidos:

   - **Erro de Camada 2 (Enlace):** A porta do switch que se conecta ao roteador ($Gi0/1$) foi intencionalmente mantida em seu modo padrão (acesso à VLAN 10), em vez de ser configurada como *trunk* (Figura 5.2). Em um simulador comum, isso resultaria na falha de comunicação da VLAN 20.
   - **Erro de Camada 3 (Rede):** No roteador, configurado via CLI, as sub-interfaces para as VLANs 10 e 20 receberam endereços IP da mesma sub-rede (192.168.10.0/24), causando um conflito de sobreposição de redes (Figura 5.3).

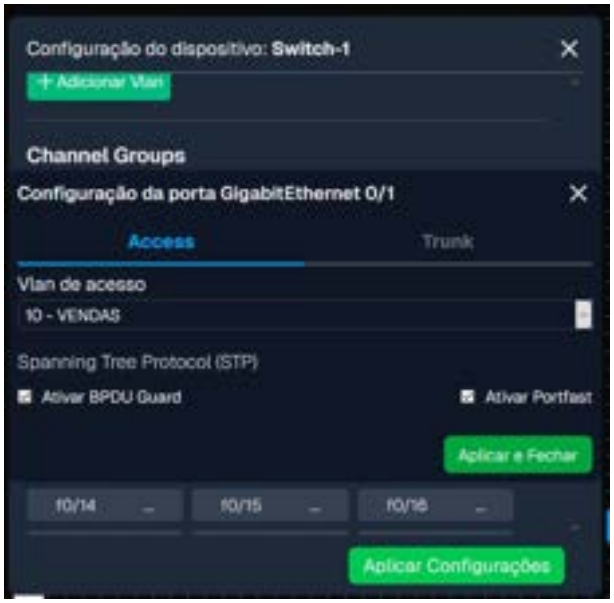


Figura 5.2: Erro proposital (Camada 2): A interface Gi0/1 (conexão com o roteador) foi deixada como modo de acesso.

Figura 5.3: Erro proposital (Camada 3): Atribuição de IPs da mesma sub-rede às sub-interfaces do roteador via CLI.

3. **Análise e Diagnóstico pela IA:** Ao acionar a funcionalidade "Analisar Cenário", a plataforma identificou ambos os erros e forneceu explicações pedagógicas:

    - **No Switch-1:** A IA (Figura 5.4) identificou que a interface deveria ser um *trunk*, explicando que o modo atual bloqueia múltiplas VLANs. As sugestões (Figura 5.5) incluíram também boas práticas de segurança.
    - **No Roteador-1:** O sistema detectou o erro crítico de sobreposição de redes (Figura 5.6), explicando por que isso impede o roteamento e sugerindo a separação das faixas de IP (Figura 5.7).

Figura 5.4: Relatório da IA para o Switch-1, identificando a falta de uma porta trunk.

Figura 5.5: Sugestões de melhoria propostas pela IA para o Switch-1.

Figura 5.6: Relatório da IA para o Roteador-1, identificando o erro crítico de sobreposição de sub-redes.

Figura 5.7: Sugestões de melhoria propostas pela IA para corrigir a sobreposição de redes no roteador.

5.1.2 Cenário B: Diagnóstico de Rota Inalcançável em Roteamento Estático

Contextualização e Desafio Pedagógico: O roteamento estático exige que o estudante compreenda a topologia da rede para definir manualmente o próximo salto (*next-hop*). Um erro recorrente entre iniciantes é configurar um *next-hop* com um endereço IP válido sintaticamente, mas que não reside em nenhuma rede diretamente conectada ao roteador. Em simuladores convencionais, esse erro cria uma "rota de buraco negro": a configuração é aceita pela CLI sem avisos, mas o tráfego é descartado silenciosamente,

dificultando o diagnóstico.

Execução e Evidência da Funcionalidade:

1. **Modelagem:** Foi elaborada uma topologia linear contendo dois roteadores (R1 e R2) interligados por um enlace serial, onde cada roteador atende a uma LAN distinta.

Figura 5.8: Topologia de rede para o cenário de roteamento estático.

2. **Injeção de Falha Lógica:** Após a configuração básica, utilizou-se o terminal simulado do Roteador R1 para inserir uma rota estática destinada à rede do Roteador R2. Propositalmente, definiu-se o *next-hop* como `10.0.0.3` (Figura 5.9). Embora este seja um IP possível na sub-rede, ele não estava atribuído à interface do R2 (que utilizava `10.0.0.2`), tornando a rota inalcançável.

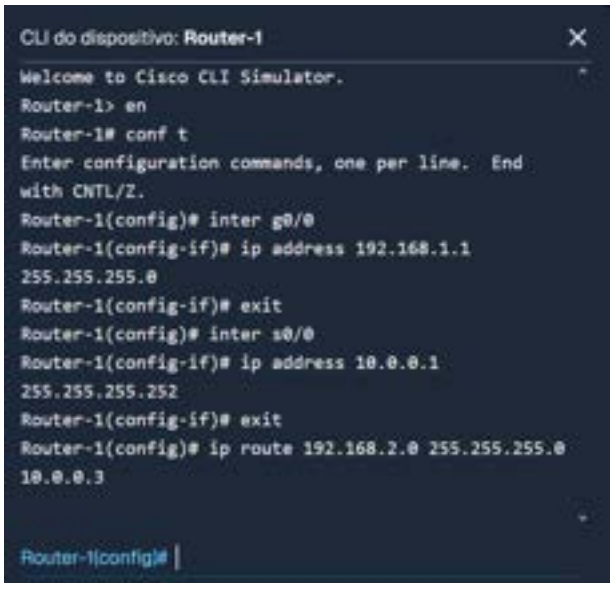


Figura 5.9: Configuração CLI do Roteador R1, mostrando a rota estática com *next-hop* incorreto (10.0.0.3).

3. **Intervenção da IA e Valor Educacional:** Ao solicitar a análise, o sistema ultrapassou a verificação de sintaxe. Conforme ilustrado na Figura 5.10, o módulo inteligente cruzou a tabela de roteamento com o estado das interfaces e detectou a inconsistência. Diferente de uma mensagem de erro genérica, o *feedback* explicou

o conceito de *reachability* (alcançabilidade), guiando o aluno a apontar o *next-hop* para o endereço correto (`10.0.0.2`).

Figura 5.10: Relatório da IA para o Roteador R1, identificando a rota estática com *next-hop* inalcançável.

5.1.3 Cenário C: Análise de Bloqueio Indevido em ACLs

Contextualização e Desafio Pedagógico: As Listas de Controle de Acesso (ACLs) representam um desafio cognitivo elevado. Um equívoco comum é aplicar regras restritivas que, inadvertidamente, bloqueiam o tráfego de gerenciamento necessário. Diagnosticar se um bloqueio ocorre por falha de rota, erro de porta ou uma regra de ACL mal posicionada é uma tarefa complexa para novatos.

Execução e Evidência da Funcionalidade:

1. **Topologia:** A topologia deste cenário é idêntica à apresentada no Cenário A Figura 5.1. Portanto, sua representação gráfica foi omitida para evitar redundância.

2. **Definição de Política:** O objetivo era permitir o acesso do PC-Admin à interface de gerenciamento do roteador, mas negar o acesso dos demais usuários.

3. **Configuração Conflitante:** Via CLI, implementou-se uma ACL restritiva na ordem incorreta, resultando no bloqueio total do acesso de gerenciamento, inclusive para o administrador. Sintaticamente, os comandos foram aceitos pelo roteador, mas a lógica impedia o acesso (Figura 5.11).

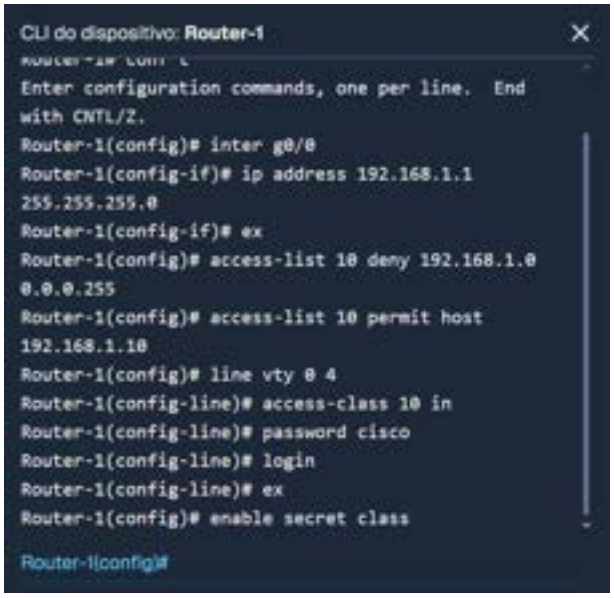


Figura 5.11: Configuração de ACL no Roteador R1 resultando em bloqueio de gerenciamento.

4. **Análise e Diagnóstico pela IA:** A funcionalidade "Analisar Cenário" foi executada no Roteador R1.

Resultados da Demonstração (Cenário C): O sistema demonstrou capacidade de correlacionar as regras de segurança com a conectividade esperada.

- **Identificação do Bloqueio:** O relatório (Figura 5.12) apontou explicitamente que a *access-list 10* estava negando o acesso de gerenciamento para a sub-rede, impedindo a administração do dispositivo.
- **Sugestões de Segurança e Correção:** Além de sugerir a revisão da lógica da ACL para restaurar o acesso (Figura 5.13), a IA aproveitou o contexto para recomendar boas práticas de segurança essenciais que não haviam sido configuradas, como a habilitação do protocolo SSH em substituição ao Telnet e o uso de senhas fortes. Isso demonstra que a ferramenta atua além da correção de erros, promovendo o endurecimento (*hardening*) da segurança da rede.

Figura 5.12: Relatório da IA identificando que a ACL está bloqueando o gerenciamento da rede.

Figura 5.13: Sugestões de melhoria, incluindo revisão da lógica de acesso e adoção de SSH.

5.2 Análise dos Resultados da Demonstração

A execução dos cenários A, B e C fornece evidências empíricas que validam o artefato não apenas como *software* funcional, mas como ferramenta pedagógica. Relacionando os resultados aos Requisitos Funcionais (RF) definidos no Capítulo 4:

- **Interoperabilidade e Sincronia (RF-03 e RF-06):** A demonstração comprovou que o modelo de dados *Single Source of Truth* funciona. Alterações feitas via CLI no Cenário B (rota estática) e Cenário C (ACLs) foram corretamente interpretadas pelo sistema, validando a arquitetura híbrida proposta.
- **Capacidade de Diagnóstico Semântico (RF-04):** Este foi o principal diferencial validado. Nos simuladores tradicionais, os erros apresentados nos cenários B (rota inalcançável) e C (ordem da ACL) resultariam apenas em "falha de ping", obrigando o aluno a revisar linha por linha sem saber o que procurar. O artefato proposto, ao contrário, atuou na camada semântica do problema, diagnosticando a *lógica* por trás do comando e não apenas sua sintaxe.

Essa capacidade de transformar um erro silencioso em um momento de instrução explícita confirma a hipótese de que a integração com LLMs pode reduzir a carga cognitiva durante o processo de *troubleshooting*, permitindo que o aluno foque nos conceitos (ex: "como o roteador escolhe o caminho?") e não apenas na mecânica da ferramenta.

5.3 Considerações Finais do Capítulo

Este capítulo demonstrou, através de casos de uso práticos, que o artefato desenvolvido atende aos seus objetivos de projeto. A ferramenta provou ser capaz de atuar naquilo que Vygotsky (1978) define como a "Zona de Desenvolvimento Proximal": o espaço entre o que o aluno consegue fazer sozinho (configurar comandos básicos) e o que ele não consegue fazer sem ajuda (diagnosticar erros lógicos complexos).

Ao fornecer um *feedback* que explica o porquê do erro e sugere o caminho da correção, o simulador deixa de ser um ambiente passivo de execução para se tornar um agente ativo no processo de ensino-aprendizagem. Com a validação técnica e funcional do protótipo concluída nesta demonstração, o próximo passo (Capítulo 6) será submeter essa proposta à avaliação de usuários reais para quantificar sua eficácia e usabilidade.

Capítulo 6

Avaliação

Este capítulo corresponde à quinta etapa da metodologia de *Design Science Research* (DSR), dedicada à avaliação do artefato desenvolvido. Considerando o estágio atual do projeto e o cronograma de desenvolvimento, optou-se por uma abordagem híbrida de validação técnica e inspeção de usabilidade, composta por: (i) uma verificação funcional em ambiente de produção, conduzida pelo desenvolvedor para validar a robustez da implantação e a execução de tarefas críticas sob condições reais de rede; e (ii) uma avaliação heurística baseada nos princípios de Nielsen, aplicada como inspeção especializada da interface.

Essa estratégia permite gerar evidências sobre o comportamento do sistema quando hospedado em infraestrutura *web* real, a clareza da interface e o potencial pedagógico, mantendo o alinhamento com os requisitos funcionais e não funcionais estabelecidos no Capítulo 4.

6.1 Objetivos de Avaliação

Os objetivos específicos desta etapa são:

- Validar a eficácia do artefato na execução de tarefas de configuração quando acessado remotamente (ambiente de produção);
- Verificar a latência e a estabilidade do módulo de análise assistida por IA em condições reais de tráfego de dados;
- Examinar a clareza, consistência e eficiência da interface gráfica (GUI);
- Analisar a aderência às heurísticas de usabilidade de Nielsen;
- Identificar limitações técnicas decorrentes da hospedagem e pontos de melhoria futura.

6.2 Desenho do Estudo

A avaliação foi estruturada para testar o artefato não apenas como *software* local, mas como um serviço *web* disponível, simulando a experiência real de um estudante acessando a ferramenta fora do laboratório. A arquitetura de validação, ilustrando o fluxo de dados entre o avaliador e a infraestrutura de nuvem, é apresentada na Figura 6.1.

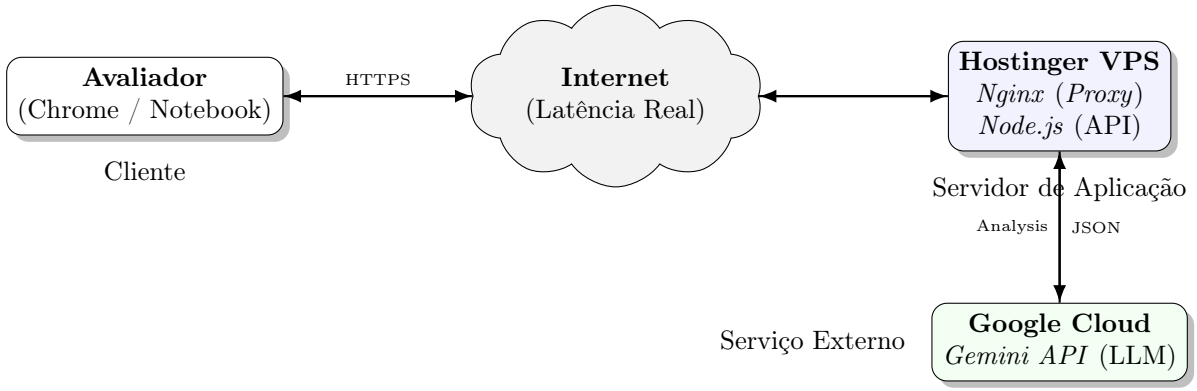


Figura 6.1: Diagrama do ambiente de produção utilizado na verificação técnica, evidenciando os pontos de latência de rede.

6.2.1 Participantes e Procedimento de Verificação

A etapa preliminar consistiu em uma verificação técnica e funcional, conduzida pelo desenvolvedor do sistema. Diferentemente de testes de usabilidade tradicionais com usuários finais, esta etapa focou na garantia de qualidade (*Quality Assurance*) do *deploy*. O avaliador executou roteiros de tarefas predefinidos buscando identificar falhas de carregamento, erros de comunicação com a API (*Timeouts/CORS*) e inconsistências de estado que poderiam surgir no ambiente de produção.

A validação com uma amostragem ampla de estudantes permanece prevista como trabalho futuro, dependendo da estabilização completa desta versão de produção.

6.2.2 Variáveis e Métricas

Foram coletadas métricas quantitativas de desempenho e dados qualitativos de inspeção:

- **Tempo de execução de tarefa (s):** Tempo cronometrado para completar fluxos de configuração;
- **Latência de resposta da IA (s):** Tempo total entre o envio da requisição e a renderização do *feedback* (incluindo *RTT* de rede);
- **Taxa de sucesso técnico:** Capacidade de completar a tarefa sem erros de execução (ex: *crashes* ou falhas de API);
- **Consistência de Estado:** Verificação da sincronia entre a visualização gráfica e os dados internos após ações repetitivas;
- **Conformidade Heurística:** Inspeção visual baseada nos 10 princípios de Nielsen.

6.2.3 Tarefas Avaliadas

O roteiro de verificação cobriu as funcionalidades centrais (*Core Features*):

1. Criação e manipulação de topologia (RF-01);
2. Configuração de VLANs e endereçamento via GUI (RF-02);

3. Configuração avançada via CLI simulada (RF-03);
4. Solicitação de análise de cenário à IA (RF-04);
5. Exportação e persistência de dados (RF-05).

6.2.4 Condições e Controle (Ambiente de Produção)

As avaliações foram realizadas em um computador pessoal equipado com processador [Intel Core i5-9300H], operando a uma frequência de [2.40 GHz], com [16 GB] de memória RAM e sistema operacional [Windows 10 Home 64-bits]. Para a execução dos testes, utilizou-se o navegador *Google Chrome* em sua versão mais recente [versão 120.0]. O artefato avaliado corresponde à versão de produção, implantada em um servidor na nuvem na plataforma Hostinger.

A infraestrutura de produção utilizou o servidor *WEB* Nginx configurado como *proxy* reverso, gerenciando as requisições HTTP/HTTPS e servindo os arquivos estáticos da aplicação React, enquanto a API de *backend* (Node.js) gerenciava a comunicação segura com o LLM. Essa configuração permitiu validar o requisito de acessibilidade *WEB* (RNF-04) em um cenário realista, expondo a aplicação a variáveis como latência de DNS, tempo de negociação SSL e variações de banda de *internet*, diferentemente de testes isolados em *localhost*.

6.3 Análise de Dados

A Tabela 6.1 apresenta os tempos médios registrados durante a execução das tarefas no ambiente hospedado.

Tabela 6.1: Resultados da verificação técnica em ambiente de produção.

Tarefa	**Tempo médio (s)**	**Tentativas até sucesso**
Criar topologia simples	20	1
Configurar VLANs e portas	39	1
Configurar subinterfaces	22	1
Configurar rota estática	33	1
Identificação de erro via IA	31*	1
Exportação de configurações	3	1
Salvamento automático (*localforage*)	Imediato (< 0.1)	N/A

* O tempo médio de 31 segundos reflete a carga computacional da inferência profunda do LLM somada à latência de rede (RTT). Dada a natureza da tarefa, uma revisão pedagógica completa do cenário, este intervalo é aceitável, assemelhando-se ao tempo de consulta a um tutor, desde que o sistema forneça *feedback* visual de progresso para manter o usuário engajado.

6.4 Resultados

6.4.1 Resultados da Verificação Funcional

A implantação na *Hostinger* e o uso do *Nginx* demonstraram estabilidade. Os principais achados técnicos foram:

- **Desempenho da GUI:** O carregamento inicial da aplicação (*First Contentful Paint*) manteve-se rápido devido à otimização dos *bundles* estáticos servidos pelo Nginx.
- **Robustez da Sincronização:** A lógica de *Single Source of Truth* não apresentou falhas de concorrência, mesmo quando múltiplos eventos de clique ocorriam rapidamente.
- **Comportamento da IA:** O tempo médio de resposta de 31 segundos, embora superior a interações de interface convencionais, mostrou-se compatível com a expectativa de uma atividade de "correção" ou "tutoria". O uso de indicadores de carregamento (*loading spinners*) e mensagens de estado provou-se crítico para evitar o abandono da tarefa, mantendo o usuário ciente de que uma análise complexa estava em processamento remoto.
- **Persistência:** O salvamento gerenciado pela biblioteca *localforage* funcionou corretamente, permitindo a recuperação do estado após o recarregamento da página, simulando uma sessão contínua de estudo.

6.4.2 Resultados da Avaliação Heurística

A inspeção baseada nas heurísticas de Nielsen destacou os seguintes pontos fortes e atenções no *design*:

Visibilidade do estado do sistema. A interface fornece *feedback* imediato para ações locais (arrastar, conectar). Para ações assíncronas (análise da IA), o sistema informa explicitamente que o processamento está em curso, mitigando a incerteza causada pela latência da rede.

Consistência e padrões. A aplicação mantém consistência visual entre os formulários e o terminal. A tradução dos comandos da GUI para a CLI segue estritamente a sintaxe Cisco IOS, garantindo que o padrão mental que o aluno está construindo não seja violado.

Prevenção de erros e Tratamento de Falhas. A interface bloqueia entradas inválidas básicas (ex: portas inexistentes). No contexto de produção, o sistema demonstrou resiliência: caso a requisição à API da IA falhe (ex: *timeout* do servidor ou queda de *internet*), uma mensagem de erro amigável é exibida, instruindo o usuário a tentar novamente, em vez de travar a aplicação ("tela branca").

Estética e *design* minimalista. O uso de ícones customizados e a ausência de menus densos reduzem a carga cognitiva, permitindo foco na topologia.

6.5 Ameaças à Validade

Reconhecem-se as limitações desta avaliação:

- **Amostragem:** A ausência de testes massivos com estudantes iniciantes impede a generalização sobre a curva de aprendizado subjetiva.
- **Dependência Externa:** A estabilidade do recurso de IA depende da disponibilidade da API do fornecedor e da conectividade do servidor de hospedagem.
- **Viés do Desenvolvedor:** A verificação funcional, embora técnica, foi realizada por quem conhece os caminhos ideais ("*happy path*") do sistema.

6.6 Considerações Finais do Capítulo

A avaliação demonstrou que o artefato atingiu um nível de maturidade técnica suficiente para operação em ambiente de produção. A implantação na *Hostinger* com *Nginx* validou os requisitos de acessibilidade e desempenho *web*. O sistema mostrou-se capaz de entregar *feedback* pedagógico complexo com latência aceitável, e a interface seguiu princípios de usabilidade que favorecem a prevenção de erros. Esses resultados habilitam o sistema para, em trabalhos futuros, ser submetido a estudos de campo longitudinais em salas de aula.

Capítulo 7

Conclusões e Trabalhos Futuros

Este capítulo encerra o ciclo da metodologia de *Design Science Research* (DSR) aplicado neste trabalho, sintetizando as contribuições do artefato desenvolvido frente à problemática da acessibilidade e complexidade no ensino de redes de computadores. Além disso, discutem-se as limitações identificadas durante as etapas de validação e delineiam-se caminhos para a continuidade da pesquisa e evolução da plataforma.

7.1 Síntese das Contribuições

O objetivo principal deste trabalho foi desenvolver uma plataforma *web* capaz de reduzir a barreira de entrada para estudantes de redes, integrando simulação simplificada e análise assistida por Inteligência Artificial. A execução das etapas de *design*, desenvolvimento e demonstração permitiu alcançar contribuições relevantes tanto no aspecto tecnológico quanto pedagógico.

Primeiramente, o trabalho promove a democratização do acesso ao validar a viabilidade técnica de um simulador operando inteiramente no navegador. Diferentemente de ferramentas tradicionais que exigem *hardware* robusto e instalação local, a implementação responsiva e a arquitetura *mobile-first* atenderam ao requisito de acessibilidade, permitindo que estudantes utilizem dispositivos móveis para realizar configurações complexas. A validação em ambiente de produção, utilizando servidor VPS na Hostinger e *proxy* reverso *Nginx*, confirmou que a solução é estável e suporta latências reais de rede.

No que tange à redução da carga cognitiva, a arquitetura de "fonte única da verdade" (*Single Source of Truth*) provou ser eficaz. Ao sincronizar instantaneamente as ações visuais (GUI) com o emulador de terminal (CLI), o sistema oferece um *scaffolding* cognitivo que permite ao aluno transitar progressivamente da interface visual para a linha de comando, compreendendo a correlação entre ambas sem o atrito comum de ferramentas desconexas.

A inovação central do artefato, contudo, reside no diagnóstico semântico via LLM. A integração com o modelo *Gemini* permitiu que o sistema identificasse erros lógicos, como rotas inalcançáveis ou ACLs mal posicionadas, que simuladores determinísticos geralmente ignoram. Embora a avaliação tenha registrado um tempo médio de resposta de 31 segundos, esse intervalo foi compensado pela profundidade do *feedback* pedagógico, que explica a causa raiz do erro, atuando efetivamente como um tutor virtual.

7.2 Limitações do Estudo

Apesar dos resultados promissores, o desenvolvimento e a avaliação do artefato revelaram limitações técnicas e metodológicas. A restrição mais evidente é a dependência de conectividade. Como verificado nos testes, a funcionalidade de análise depende estritamente de uma conexão ativa com a *internet* e da disponibilidade da API externa. A latência de rede e o tempo de

inferência do modelo introduzem uma espera que, embora aceitável para fins educativos, impede o uso da ferramenta em cenários de desconexão total (*offline*).

Outra limitação refere-se ao escopo da simulação. O artefato realiza uma simulação baseada em estado e configuração, diferindo de emuladores de pacotes em tempo real como o GNS3. Consequentemente, protocolos que dependem de temporização precisa ou inspeção profunda de tráfego não são suportados nesta versão. Adicionalmente, embora a biblioteca *localforage* garanta a persistência local no navegador, a ausência atual de um banco de dados em nuvem impede a sincronização de sessões entre diferentes dispositivos.

Do ponto de vista metodológico, a avaliação concentrou-se na verificação técnica e inspeção de usabilidade por especialistas. A ausência de um estudo de campo longitudinal com uma turma real de alunos limita a generalização dos resultados estatísticos sobre o impacto direto na curva de aprendizado, permanecendo como uma oportunidade para investigações futuras.

7.3 Trabalhos Futuros

A evolução natural desta pesquisa aponta para direções técnicas e pedagógicas. No âmbito técnico, sugere-se a implementação de um sistema de autenticação e banco de dados em nuvem para permitir a portabilidade de cenários entre dispositivos. Também é pertinente investigar o uso de *Small Language Models* (SLM) executados diretamente no navegador via *WebAssembly*, o que eliminaria a dependência de APIs externas e reduziria a latência da análise. A expansão do suporte a protocolos de roteamento dinâmico avançado (como EIGRP e BGP completo) também aumentaria a cobertura curricular da ferramenta. Uma expansão crítica para a empregabilidade dos estudantes é o suporte a ambientes multi-vendedor. Planeja-se adicionar a sintaxe da Huawei (VRP) ao lado da atual Cisco (IOS), permitindo que o simulador prepare os alunos para a realidade heterogênea dos provedores de internet e grandes datacenters.

Na vertente pedagógica, a incorporação de elementos de gamificação, como níveis de dificuldade progressiva e sistemas de conquistas, poderia aumentar o engajamento discente. Além disso, a implementação de funcionalidades colaborativas em tempo real permitiria que grupos de alunos configurassem a mesma topologia simultaneamente. Por fim, recomenda-se a realização de experimentos controlados em sala de aula para quantificar, de forma comparativa, o ganho de aprendizado proporcionado pelo *feedback* assistido por IA em relação aos métodos tradicionais.

7.4 Considerações Finais

O ensino de redes de computadores não precisa ser limitado pela complexidade das ferramentas ou pelo custo do *hardware*. Este trabalho demonstrou que é possível aliar interfaces modernas e Inteligência Artificial para criar um ambiente de aprendizado que é, ao mesmo tempo, tecnicamente rigoroso e pedagogicamente acolhedor. Ao transformar a mensagem de erro em explicação e a configuração repetitiva em prática guiada, o artefato desenvolvido representa um avanço em direção a um ensino técnico mais inclusivo, eficiente e alinhado com a realidade dos estudantes.

Referências Bibliográficas

Shahla Asadi, Jordan Allison, Madhu Khurana, and Mehrbakhsh Nilashi. Simulation-based learning for computer and networking teaching: A systematic literature review and bibliometric analysis. *Education and Information Technologies*, 2024a. doi: 10.1007/s10639-024-12476-7.

Shahla Asadi, Jordan Allison, Madhu Khurana, and Mehrbakhsh Nilashi. Simulation-based learning for computer and networking teaching: A systematic literature review and bibliometric analysis. 2024b. doi: 10.1007/s10639-024-12476-7.

Nazre bin Abdul Rashid, Md. Zahar bin Othman, Rasyid bin Johan, and Salman Firdaus bin Hj. Sidek. Cisco packet tracer simulation as effective pedagogy in computer networking course. *International Journal of Interactive Mobile Technologies (iJIM)*, 13(10), 2020. doi: 10.3991/ijim.v13i10.11283.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Adir Lima Cardoso. Inclusão digital na educação: Desafios e perspectivas na implementação da computação conforme a legislação educacional. 2024. URL `https://lcomp.pelotas.ifsul.edu.br/uploads/tcc/J5KF2iCPpd.pdf`.

Ka Ching Chan et al. Integration of physical equipment and simulators for on-campus and online delivery of practical networking labs, 2015. arXiv preprint arXiv:1510.00482.

Cisco Systems. Cisco packet tracer, 2024. Disponível em: `https://www.netacad.com/courses/packet-tracer`. Acesso em: 09 out. 2025.

EVE-NG. Eve-ng - emulated virtual environment next generation, 2024. Disponível em: `https://www.eve-ng.net/`. Acesso em: 09 out. 2025.

GNS3 Community. Gns3 - graphical network simulator, 2024. Disponível em: `https://www.gns3.com/`. Acesso em: 09 out. 2025.

Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, pages 75–105, 2004.

IFPB. Ministério da Educação, Secretaria de Educação Profissional e Tecnológica, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Comissão Própria de Avaliação. Relatório de Autoavaliação Institucional 2020. 2020. URL `https://www.ifpb.edu.br/cajazeiras/cpa/documentos/relatorio-autoavaliacao-ifpb-2020.pdf`.

James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Pearson Education, 6 edition, 2016.

Woratat Makasiranondh, S. Paul Maj, and David Veal. Pedagogical evaluation of simulation tools usage in network technology education. *World Transactions on Engineering and Technology Education*, 8(3):321–326, 2010.

Gardner Mwansa, Matipa Ricky Ngandu, and Zola Sydney Dasi. Enhancing practical skills in computer networking: Evaluating the unique impact of simulation tools, particularly cisco packet tracer, in resource-constrained higher education settings. *Education Sciences*, 14(10), 2024. doi: 10.3390/educsci14101099.

Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1994.

Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. In *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST)*, pages 83–106. Springer, 2007.

David H. Rose and Anne Meyer. *Universal Design for Learning: Theory and Practice*. CAST Professional Publishing, 2012.

Jeffrey Rubin and Dana Chisnell. *Handbook of usability testing: how to plan, design, and conduct effective tests*. John Wiley & Sons, 2008.

John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12 (2):257–285, 1988.

Andrew S. Tanenbaum and David J. Wetherall. *Redes de Computadores*. Pearson, 5 edition, 2011.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models, 2023. Disponível em: `https://arxiv.org/abs/2302.13971`.

Lev S. Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard University Press, Cambridge, MA, 1978.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL `https://arxiv.org/abs/2303.18223`.

Apêndice A

Roteiros e Instrumentos de Coleta

A.1 Termo de Consentimento e Informações da Pesquisa

Abaixo é apresentado o texto de consentimento exibido aos participantes da coleta de dados (discutida na Seção 3.2.1).

> Você está sendo convidado(a) a participar de uma pesquisa acadêmica realizada no âmbito de um Trabalho de Conclusão de Curso (TCC) do curso de Telemática do Instituto Federal da Paraíba (IFPB) – Campus Campina Grande.
>
> O objetivo deste estudo é compreender como os estudantes percebem as atividades de configuração de redes, de modo a apoiar o desenvolvimento de uma plataforma de simulação que facilite o processo de aprendizagem.
>
> A participação é voluntária e consiste no preenchimento de um questionário com duração aproximada de 5 minutos.
>
> Todas as informações fornecidas serão anônimas e confidenciais, e utilizadas apenas para fins de pesquisa. Não há qualquer tipo de risco, punição, benefício financeiro ou identificação individual. Você poderá interromper sua participação a qualquer momento, sem justificativa.
>
> Os resultados serão apresentados de forma agrupada, sem qualquer menção a nomes ou dados pessoais.
>
> *Ao clicar em "Concordo em participar", você declara estar ciente das informações e autoriza a utilização anônima das respostas para fins acadêmicos.*

A.2 Perguntas do Formulário

A seguir, são listadas as perguntas apresentadas no formulário "Percepções sobre o Processo de Aprendizagem em Redes de Computadores".

1. Qual curso você está cursando?
2. Qual período você está cursando?
3. Você já teve contato com simuladores de redes (como Packet Tracer, GNS3, etc.) em alguma disciplina do curso?
4. Qual seu nível de familiaridade com a configuração de dispositivos de rede (roteadores, switches) usando Linha de Comando (CLI)?

- *Nenhuma (nunca configurei)*
- *Baixa (configurei poucas vezes, com ajuda)*
- *Média (consigo fazer configurações simples)*
- *Alta (consigo fazer configurações complexas e solucionar problemas)*

5. Pensando na sua experiência com simuladores de redes, indique seu nível de concordância com as afirmações abaixo: *(Opções: Discordo Totalmente, Discordo, Neutro, Concordo, Concordo Totalmente)*

    - A interface de linha de comando (CLI) é intuitiva para iniciantes.
    - É fácil encontrar e corrigir erros de configuração.
    - Sinto falta de *feedback* detalhado quando uma configuração falha.
    - O processo de configurar vários dispositivos é repetitivo.

6. Quais são os maiores desafios que você enfrenta ou já enfrentou ao aprender a configurar cenários de redes? *(Campo de texto aberto)*

7. Você já utilizou alguma ferramenta digital ou aplicativo para estudar/configurar cenários de redes (como ChatGPT, Gemini, DeepSeek, etc.)? *(Opções: Sim, frequentemente; Sim, ocasionalmente; Sim, raramente; Não)*

8. Se sim, como essa ferramenta ajudou nos seus estudos? *(Múltipla escolha, ex: "Entender conceitos", "Gerar comandos", "Identificar erros", etc.)*

9. Imagine poder configurar equipamentos de rede com suporte de uma plataforma que oferece dicas e validações em tempo real, além da CLI tradicional. Você acredita que essa abordagem facilitaria o aprendizado para quem está começando? *(Opções: Sim, Não, Talvez)*

10. Das funcionalidades abaixo para um novo simulador, quais você considera mais úteis para o seu aprendizado? *(Múltipla escolha, ex: "Uma I.A. que analisa o cenário", "Configuração guiada por formulários", "Biblioteca de cenários prontos", etc.)*

11. Você tem alguma sugestão ou comentário adicional sobre como uma ferramenta poderia melhorar o ensino de redes de computadores? *(Campo de texto aberto)*

Apêndice B

Detalhes de Implementação e Configurações

Esta seção apresenta a estrutura de dados interna utilizada pelo simulador e um exemplo representativo dos arquivos de configuração exportados.

B.1 Estrutura de Dados (TypeScript)

A listagem abaixo apresenta, de forma resumida, a interface principal que modela os dispositivos no *frontend*. Detalhes repetitivos foram omitidos para clareza.

```
export interface NetworkDeviceData {
    type: 'switch' | 'router' | 'pc';
    label: string;
    position: { x: number; y: number };
    ports: string[];

    portsConnected: Map<string, {
        port: string,
        deviceConnected: string,
        deviceConnectedPort: string
    }>;

    config?: {
        hostname?: string;
        ipv4?: string;
        subnetMask?: string;
        defaultGateway?: string;
)
        vlans?: {
            id: string;
            name: string;
        }[];

        interfaces?: Record<string, {
            isUp?: boolean;
            mode?: 'trunk' | 'access';
            accessVlan?: string;
            ip?: string;
            subnetMask?: string;
            subInterfaces?: Record<string, {
                ip?: string;
```

```
                subnetMask?: string;
            }>
        }>;

        staticRoutes?: { network: string; nextHop: string; }[];
        accessLists?: { id: string; rules: any[] }[];
        ospf?: { processId: string; networks: any[] };
    }
};
```

Listing B.1: Estrutura de dados resumida da entidade NetworkDeviceData.

B.2 Configurações Exportadas (Exemplo Representativo)

Para demonstrar a capacidade do artefato em gerar sintaxe válida para equipamentos reais (Cisco IOS), apresentam-se abaixo os arquivos de configuração extraídos do Cenário A.

Nota: As configurações exportadas dos Cenários B e C foram omitidas desta seção por seguirem a mesma estrutura lógica e de formatação apresentada no exemplo abaixo, evitando assim a redundância de informações.

B.2.1 Configuração do Switch-1 (Cenário A)

```
hostname S1
spanning-tree mode pvst
!
vlan 10
 name VENDAS
!
vlan 20
 name TI
!
interface GigabitEthernet 0/1
 switchport mode access
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet 0/2
!
interface FastEthernet 0/1
 switchport mode access
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface FastEthernet 0/2
 switchport mode access
 switchport access vlan 20
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
...interfaces sem configuracao
```

Listing B.2: Script exportado do Switch-1 contendo configuração de VLANs e erro de trunk.

B.2.2 Configuração do Roteador-1 (Cenário A)

```
hostname R1
!
ip dhcp pool LAN-POOL-VLAN-10
  network 192.168.10.0 255.255.255.0
  default-router 192.168.10.1
!
ip dhcp pool LAN-POOL-VLAN-20
  network 192.168.10.0 255.255.255.0
  default-router 192.168.10.2
!
interface GigabitEthernet 0/0
  no ip address
  no shutdown
!
interface GigabitEthernet 0/0.10
  encapsulation dot1Q 10
  ip address 192.168.10.1 255.255.255.0
  no shutdown
!
interface GigabitEthernet 0/0.20
  encapsulation dot1Q 20
  ip address 192.168.10.2 255.255.255.0
  no shutdown
!
```

Listing B.3: Script exportado do Roteador-1 com configuração Router-on-a-Stick.

Apêndice C

Código-Fonte de Diagramas

Esta seção contém o código-fonte PlantUML para o diagrama de arquitetura apresentado na Figura 4.1 (Capítulo 4).

```
@startuml
' Compact layout
skinparam dpi 96
skinparam defaultFontSize 10
skinparam componentStyle rectangle
left to right direction

actor "Usuario" as User

rectangle "Front-end\n(React / Next.js)" as FE {
  [UI / Editor]
  [Scenario Context]
}

rectangle "Back-end\n(Next.js API)" as BE {
  [Route: /api/analyze-scenario]
}

cloud "LLM API\n(Gemini)" as LLM

User --> "UI / Editor" : interage
"UI / Editor" --> "Scenario Context" : atualiza JSON
"UI / Editor" --> "Route: /api/analyze-scenario" : POST (scenario JSON)
"Route: /api/analyze-scenario" --> LLM : request (prompt + apiKey)
LLM --> "Route: /api/analyze-scenario" : response (JSON)
"Route: /api/analyze-scenario" --> "UI / Editor" : analysis (JSON)
@enduml
```

Listing C.1: Código-fonte PlantUML do diagrama de arquitetura.

Apêndice D

Prompts e Exemplos de Interação com LLM

Esta seção detalha a engenharia de prompt utilizada e apresenta um exemplo real de dados brutos trocados entre a aplicação e a API do Google Gemini.

D.1 Prompt de Sistema (System Prompt)

O prompt abaixo é injetado em todas as requisições para garantir que o modelo atue como um especialista e retorne dados estruturados.

```
Voce e um consultor especialista em redes Cisco.
Recebera um array de objetos { device, config } onde config e a
    configuracao em CLI do dispositivo.
Sempre analise procurando:
- VLANs criadas mas nao utilizadas
- PCs sem IP ou com IP repetido
- Switch sem ip default-gateway
- Roteador sem interfaces IP ou rotas
- Possiveis loops de spanning-tree
- Falta de redundancia ou ma pratica de configuracao (seguranca,
    otimizacao)
Responda SEMPRE em JSON, um array de objetos no formato abaixo (um
    objeto por dispositivo):
[
  {
    "device": "nome do dispositivo",
    "problemas": [ "descricao do problema 1" ],
    "melhorias": [ "sugestao 1" ],
    "resumo": "Resumo geral do dispositivo"
  }
]
Imprima somente o JSON. Nao escreva texto extra.
```

Listing D.1: Prompt de sistema utilizado para instruir o LLM.

D.2 Exemplo de Resposta Estruturada (JSON)

O modelo retorna uma análise estruturada que é processada pelo *frontend* para gerar o relatório visual. Abaixo, apresenta-se a resposta bruta recebida durante a análise do Cenário A 5.1.1.

Vale destacar que a resposta identifica corretamente o conflito crítico de endereçamento IP (sobreposição de sub-redes) e a ausência da porta *trunk*, além de sugerir melhorias de segurança não solicitadas explicitamente, como a desativação de portas não utilizadas.


```
[
    {
        "device": "Router-1",
        "problemas": [
            "Conflito de enderecamento IP: Os pools DHCP 'LAN-POOL-VLAN-10' e 'LAN-POOL-VLAN-20' estao configurados para a mesma rede (192.168.10.0/24), o que causara falhas na atribuicao de IPs.",
            "Sub-redes sobrepostas: As sub-interfaces Gi0/0.10 e Gi0/0.20 estao na mesma sub-rede (192.168.10.0/24), uma configuracao invalida que impede o roteamento correto entre as VLANs.",
            "Falta de conectividade externa: O roteador nao possui uma interface WAN ou uma rota padrao configurada, limitando a comunicacao apenas entre as VLANs locais."
        ],
        "melhorias": [
            "Atribuir sub-redes IP distintas para cada VLAN e seus respectivos pools DHCP. Exemplo: 192.168.10.0/24 para VLAN 10 e 192.168.20.0/24 para VLAN 20.",
            "Corrigir os enderecos IP das sub-interfaces para que correspondam as suas novas sub-redes.",
            "Excluir os enderecos IP dos gateways dos pools DHCP para evitar que sejam atribuidos a clientes (usando `ip dhcp excluded-address`).",
            "Configurar uma interface para conexao WAN e adicionar uma rota padrao (`ip route 0.0.0.0 0.0.0.0 <next-hop>`) para permitir acesso a internet ou outras redes."
        ],
        "resumo": "O roteador possui uma falha critica de configuracao com sub-redes IP sobrepostas para as VLANs, o que impedira o funcionamento da rede. A correcao do enderecamento e essencial para a operacao."
    },
    {
        "device": "Switch-1",
        "problemas": [
            "Falta de configuracao de gerenciamento: Nao ha uma interface SVI (VLAN) com endereco IP nem um 'ip default-gateway', o que impede o gerenciamento remoto do dispositivo.",
            "Risco de seguranca e loop: Multiplas portas nao estao configuradas (ex: Gi0/2, Fa0/3-24), permanecendo ativas na VLAN 1 padrao. Isso permite acesso nao autorizado a rede e aumenta o risco de loops de spanning-tree.",
            "Falta de seguranca de acesso: A configuracao nao mostra senhas para as linhas de console ou VTY, deixando o switch vulneravel a acessos nao autorizados.",
            "Falta de conectividade inter-VLAN: Nenhuma porta esta configurada como 'trunk', o que significa que o switch nao pode se comunicar com um roteador ou outro switch para permitir que os dispositivos nas VLANs 10 e 20 acessem outras redes."
        ],
        "melhorias": [
            "Criar uma interface de gerenciamento (SVI), por exemplo, 'interface Vlan99', atribuir um endereco IP e configurar o comando 'ip default-gateway' para permitir o gerenciamento remoto.",
            "Aplicar o comando 'shutdown' em todas as portas que nao estao em uso para mitigar riscos de seguranca e de loop.",
            "Como boa pratica de seguranca, mover todas as portas nao utilizadas para uma VLAN 'blackhole' que nao seja roteada.",
            "Configurar senhas seguras para as linhas console e VTY e para o modo 'enable'.",
            "Definir a(s) porta(s) de uplink (ex: GigabitEthernet0/2) como trunk ('switchport mode trunk') para permitir a passagem de trafego de multiplas VLANs."
        ],
        "resumo": "O switch tem uma configuracao de acesso basica funcional, com boas praticas de PortFast e BPDU Guard nas portas de usuario. No entanto, falha em aspectos criticos de seguranca e gerenciamento, como senhas de acesso, tratamento de portas nao utilizadas e configuracao de um IP para gerenciamento remoto. A ausencia de uma porta trunk o isola do resto da rede."
    }
]
```


Listing D.2: JSON real retornado pela API contendo a análise do Cenário A.