Desenvolvimento de um Sistema de Gerenciamento Remoto para Redução de Inatividade de Servidores Windows Server

Vinícius Fardo Riboldi[1], Hugo André Klauck[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Farroupilha

Farroupilha – RS – Brasil

vinicius.riboldi@hotmail.com, hugo.klauck@farroupilha.ifrs.edu.br

Resumo: *No contexto dos desafios crescentes no setor de TI, como o monitoramento contínuo de múltiplos servidores e a necessidade de respostas rápidas a incidentes em ambientes corporativos, agravados pelo aumento do trabalho remoto e híbrido, este trabalho apresenta o desenvolvimento de um sistema de gerenciamento remoto de servidores, com foco em coleta de dados em tempo real, visualização por dashboards interativos e uma interface acessível via app mobile e web para ações corretivas de servidores Windows. A proposta aborda as demandas por mobilidade, eficiência e segurança em ambientes de TI, permitindo respostas ágeis a incidentes e redução de tempo de inatividade. Tecnologias como Python, Grafana e frameworks para frontend são empregadas para garantir escalabilidade e flexibilidade. O sistema inclui autenticação com JWT e SQLite, timeout de inatividade e compatibilidade com múltiplos servidores.*

Palavras-chave: *Gerenciamento Remoto, Monitoramento de Servidores, Ações Corretivas*.

Abstract: *In the context of growing challenges in the IT sector, such as continuous monitoring of multiple servers and the need for rapid incident response in corporate environments, exacerbated by the increase in remote and hybrid work, this work presents the development of a remote server management system, focusing on real-time data collection, visualization through interactive dashboards, and an accessible interface via mobile and web app for corrective actions on Windows servers. The proposal addresses the demands for mobility, efficiency, and security in IT environments, enabling agile incident response and reduced downtime. Technologies such as Python, Grafana, and frontend frameworks are employed to ensure scalability and flexibility. The system includes authentication with JWT and SQLite, inactivity timeout, and compatibility with multiple servers.*

Keywords: *Remote Management, Server Monitoring, Corrective Actions*

1. Introdução

O setor de TI (Tecnologia da Informação), especialmente em ambientes corporativos com múltiplos servidores, enfrenta desafios crescentes relacionados ao monitoramento contínuo e resposta rápida a incidentes. Com o aumento do trabalho remoto e híbrido, ferramentas que permitam gerenciamento acessível via *web* se tornam essenciais. (Ozer *et al.*, 2020), (Ilag, 2021)

Este projeto propõe um sistema integrado que combina coleta de métricas em tempo real, visualização via dashboards e execução de ações corretivas, utilizando tecnologias como WinRM (*Windows Remote Management*) para comunicação remota em ambientes Windows. O sistema foi desenvolvido com *backend* em Flask para API (*Application Programming Interface*) e ações, *frontend* em React para interface *web*, incluindo autenticação segura, gerenciamento de sessões com *timeout* e hospedagem no IIS (*Internet Information Services*) para acesso externo. Além disso, para ampliar a acessibilidade e atender às necessidades de mobilidade, o projeto inclui uma versão *mobile* desenvolvida em Flutter.

2. Justificativa

A escolha deste tema fundamenta-se na experiência prática vivenciada no cotidiano de um gestor de redes, onde o aumento do tempo de *downtime* (inatividade) e as ineficiências operacionais configuram obstáculos críticos à manutenção de infraestruturas de TI robustas, gerando perdas financeiras significativas e impacto negativo na produtividade corporativa (Melo et al., 2017).

O sistema proposto surge como uma possível solução para mitigar essas questões, reduzindo o *downtime* por meio de monitoramento remoto, respostas ágeis a incidentes e ações corretivas nativas, especialmente em cenários corporativos híbridos e de trabalho remoto que se tornaram predominantes (Ditarso *et al.*, 2008).

Além disso, o desenvolvimento da ferramenta atende à necessidade de superar limitações recorrentes em soluções existentes, como dependências complexas de configuração que frequentemente resultam em erros humanos e indisponibilidade prolongada (Vanbrabant & Joosen, 2011), oferecendo uma abordagem mais simples, mais acessível e totalmente adaptada à realidade de ambientes Windows híbridos.

3. Objetivos

3.1 Objetivo geral

Desenvolver um sistema de gerenciamento remoto de servidores Windows Server para monitoramento e execução de ações corretivas.

3.2 Objetivos Específicos

- Realizar o levantamento de requisitos para a utilização do sistema.
- Coletar e exibir métricas de servidores em tempo real.

- Implementar o gerenciamento de ações remotas.
- Integrar interface *web* e *mobile* com suporte a multi-servidores e autenticação segura.
- Garantir compatibilidade com múltiplos servidores e autenticação segura com JWT e SQLite.
- Implementar *timeout* de inatividade para sessões de usuário.
- Realizar testes para validar a robustez e escalabilidade do sistema.

4. Referencial teórico

No contexto de ambientes corporativos com servidores Windows, as ferramentas de monitoramento remoto são cruciais para garantir a eficiência operacional em cenários híbridos e remotos (Alhamazani et al., 2014).

O monitoramento de servidores é essencial para a detecção precoce de problemas, otimização de recursos e manutenção da disponibilidade em ambientes corporativos, permitindo que as organizações identifiquem anomalias em tempo real, reduzam o tempo de inatividade e garantam a conformidade com padrões de segurança em infraestruturas (Ayoob, Khalil e Aksoy, 2024). Essa prática não apenas melhora a qualidade do serviço ao monitorar métricas como uso de CPU (*central processing unit*), memória e rede, mas também suporta decisões informadas para escalabilidade, especialmente em cenários de nuvem híbrida onde falhas não detectadas podem resultar em perdas significativas (Anerousis *et al.*, 2005).

O gerenciamento de servidores, por sua vez, é vital para o controle eficiente de recursos, a implementação de políticas de erro e a alocação dinâmica em sistemas distribuídos, assegurando a resiliência contra falhas e a otimização do desempenho geral da infraestrutura (Vijayasekaran, Hitaishi, e Harshith, 2025). Essa abordagem facilita a recuperação de erros, o balanceamento de cargas e a manutenção proativa, reduzindo custos operacionais e melhorando a escalabilidade em ambientes de data centers, onde estratégias de gerenciamento distribuído são cruciais para lidar com heterogeneidade e demandas variáveis (Majumdar, 2024).

Dentre as soluções de destaque nesse cenário, o Grafana (Figura 1) se sobressai como uma plataforma *open-source* em sua versão básica, com opções pagas para edições c*loud* e e*nterprise* que oferecem recursos avançados como suporte dedicado e integrações premium, facilitando a visualização de métricas por meio de *dashboards* interativos, alertas configuráveis e integração com fontes de dados como Prometheus ou InfluxDB. Sua configuração é considerada intuitiva e amigável, com uma interface gráfica que simplifica a criação de painéis personalizados, tornando-o ideal para cenários de observação em ambientes de TI complexos, como análise de métricas em tempo real. (Mehdi *et al.*, 2023)

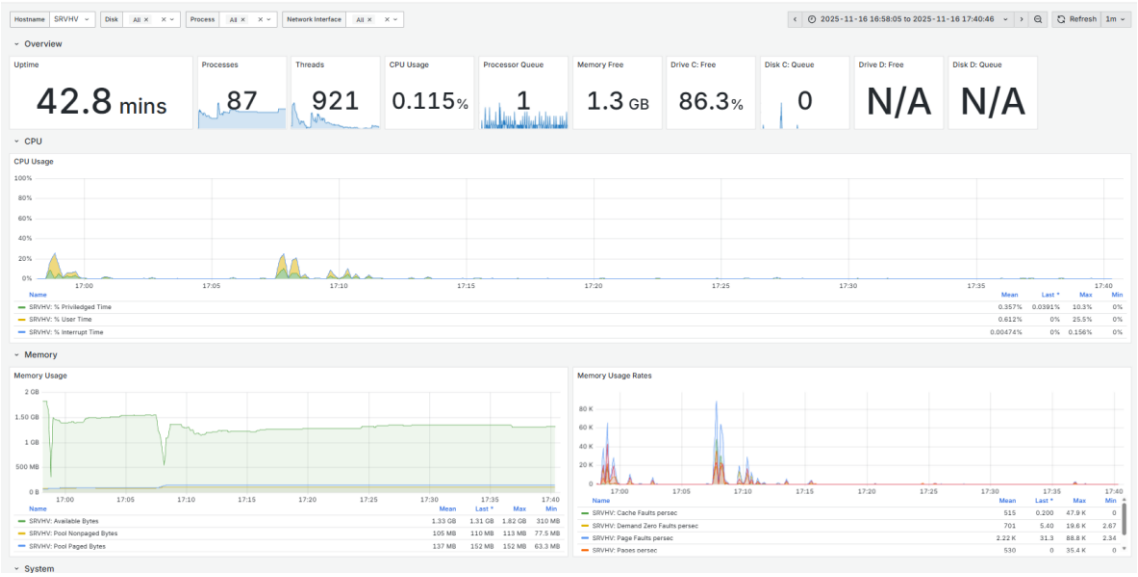

Figura 1. Tela de monitoramento de servidor do Grafana

Outro sistema que se destaca é o Zabbix (Figura 2), uma ferramenta *open-source,* com opções pagas para suporte profissional ou versões em nuvem que escalam para grandes ambientes. Embora sua configuração possa ser mais complexa em implantações extensas, devido à necessidade de ajustes manuais em *templates* e agentes, ele oferece integrações prontas que facilitam o setup inicial, sendo particularmente adequado para monitoramento escalável em infraestruturas empresariais, como redes distribuídas e servidores Windows, onde a detecção proativa de falhas e a gestão unificada de hardware e software promovem eficiência operacional em cenários de alta demanda. (Liefting e Baekel, 2021).

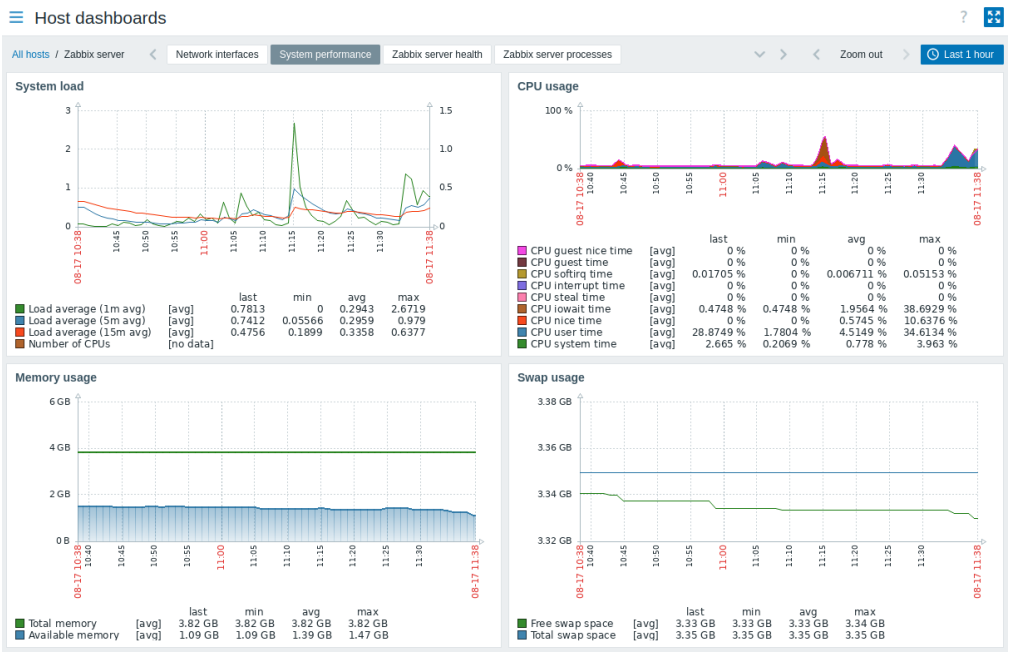

Figura 2. Tela de monitoramento de servidor do Zabbix

O uso de Flask como *framework* de *backend* é justificado por sua natureza leve e flexível, que facilita o desenvolvimento rápido de aplicações web escaláveis, especialmente em projetos de menor escala, onde a simplicidade acelera a prototipagem e a manutenção sem comprometer a segurança, como demonstrado em *frameworks* de teste de segurança para aplicações web (Rohit, *et al.*, 2023). Além disso, Flask promove eficiência no desenvolvimento ao fornecer ferramentas essenciais para criação de APIs (*Application Programming Interface*) RESTful, reduzindo a complexidade e permitindo integrações ágeis com serviços remotos (Vyshnavi e Malik, 2019).

No *frontend*, React é adotado por sua arquitetura baseada em componentes reutilizáveis, que melhora a performance em interfaces dinâmicas com atualizações frequentes, tornando-o ideal para aplicações de monitoramento que exigem renderizações eficientes e responsivas, como em plataformas de trading ou negócios que demandam interfaces intuitivas (Diniz-Junior *et al.*, 2022). Essa abordagem acelera o desenvolvimento e a manutenção, facilitando a criação de UIs (*User Interface*) complexas sem recarregamentos de página, o que é essencial para experiências de usuário em tempo real (Dwivedi, Kshamta e Joshi, 2022).

Para a versão *mobile*, Flutter se destaca por sua capacidade de desenvolvimento *cross-platform* a partir de um único código, o que reduz custos e tempo de desenvolvimento em comparação com apps nativos separados para iOS e Android, garantindo consistência visual e de desempenho em múltiplos dispositivos. Esse *framework* se destaca pela portabilidade de código e estabilidade, permitindo aplicações híbridas de alta performance em ambientes móveis diversificados, com benefícios em termos de manutenção e escalabilidade (Faiz, Kusumo e Alibasa, 2022).

A coleta de métricas via WinRM sem agentes é fundamental por oferecer um método seguro e *agentless* de gerenciamento remoto, baseado no protocolo *WS-Management* da Microsoft*,* que evita a instalação de software adicional nos servidores, reduzindo riscos de vulnerabilidades e simplificando a implantação em ambientes Windows híbridos (Masek *et al.*, 2018). Essa abordagem é mais segura que métodos tradicionais como NetBIOS (*Network Basic Input/Output System*) ou WMI (*Windows Management Instrumentation*), promovendo comunicação criptografada e eficiente para monitoramento em escala (Maduranga, 2016).

As ações administrativas e o prompt PowerShell interativo são integrados para permitir automação avançada e gerenciamento remoto, o que economiza tempo, minimiza erros humanos e melhora a eficiência em tarefas como configuração e *troubleshooting* de servidores, especialmente em infraestruturas distribuídas. PowerShell, como ferramenta de *remoting*, facilita a execução de comandos em múltiplos servidores simultaneamente, sendo uma escolha viável para administração escalável e consistente (Mohapatra, 2020).

A autenticação via JWT (*JSON Web Token*) é priorizada por sua natureza *stateless*, que reduz chamadas ao servidor e melhora a performance em autenticações subsequentes, enquanto assegura robustez contra ameaças comuns em serviços *web*, como em sistemas baseados em nuvem. Combinada com *roles* (*admin* e *viewer*), ela reforça o controle de acesso baseado em permissões, simplificando a autorização e elevando a segurança em aplicações distribuídas (Ahmed e Mahmood, 2019).

5. Metodologia

Nesta seção, detalham-se os passos adotados para o desenvolvimento do sistema de monitoramento remoto de servidores, desde a identificação das necessidades até a validação final. A metodologia adotada foi qualitativa, com ênfase na compreensão profunda das experiências e percepções dos usuários em ambientes de TI corporativos, permitindo uma análise interpretativa das demandas e *feedbacks*. Essa abordagem, inspirada em práticas ágeis, promoveu iterações flexíveis para ajustes contínuos, priorizando dados não numéricos como relatos de profissionais e avaliações subjetivas de usabilidade.

5.1 Definição de requisitos

Os requisitos foram estabelecidos a partir de uma análise qualitativa inicial das demandas em ambientes de TI, utilizando métodos como entrevistas com profissionais do setor para capturar percepções sobre funcionalidade, usabilidade e segurança. Foram identificados elementos essenciais, como autenticação de usuários, suporte a múltiplos servidores, coleta de métricas em tempo real e execução de comandos remotos. Essa fase envolveu a interpretação de narrativas coletadas para validar as necessidades, resultando em um conjunto de requisitos que abrangem tanto a versão *web* quanto a *mobile*, garantindo acessibilidade em diferentes dispositivos e uma compreensão holística das expectativas dos usuários.

5.1.1 Requisitos funcionais

Para o sistema proposto de gerenciamento remoto de servidores Windows, foram elicitados os requisitos listados abaixo. Esses requisitos foram avaliados e categorizados em duas principais classes: requisitos funcionais (RF), que abrangem as ações e funcionalidades específicas que o sistema deve executar e requisitos não funcionais (RNF), que incluem restrições e qualidades como desempenho, segurança e usabilidade.

RF1: Autenticação de usuários - O sistema deve permitir login com credenciais (*username* e *password*), gerando um *token* JWT para sessões seguras, com suporte a *roles* de *'admin'* (acesso total) e *'viewer'* (apenas visualização).

RF2: Gerenciamento de Sessões - Implementar *timeout* automático por inatividade (30 minutos), com detecção de ações do usuário para resetar o contador, tanto na parte *web* quanto na parte *mobile*.

RF3: Listagem de servidores - Exibir uma lista de servidores com *status*, filtrável por disponibilidade e navegação para detalhes individuais.

RF4: Monitoramento de métricas - Coletar e exibir métricas em tempo real (CPU, memória, processos, serviços, usuários conectados) via *dashboards* integrados ao Grafana, atualizando a cada 15 segundos.

RF5: Visualização de dashboards - Permitir a exibição de painéis gráficos para métricas, acessíveis na *web* e *mobile*, com adaptação para telas menores.

RF6: Execução de ações remotas - Para usuários *'admin'*, possibilitar comandos como finalizar processos, iniciar/parar/reiniciar serviços, desconectar usuários, reiniciar ou desligar servidores, com confirmação prévia.

RF7: Suporte *mobile* - A versão *mobile* deve replicar as funcionalidades principais, incluindo login, listagem de servidores, visualização de *dashboards* e execução de ações, com interface responsiva a rotações de tela.

RF8: *Logout* - Permitir saída manual da sessão, removendo *token* e dados locais.

5.1.2 Requisitos não funcionais

RNF1: Segurança - Utilizar criptografia em comunicações (WinRM), autenticação JWT com expiração e restrições por *role* para evitar acessos indevidos.

RNF2: Usabilidade - Fornecer interfaces intuitivas e responsivas, com navegação simples e fluida tanto na versão *web* quanto *mobile*, incluindo suporte a rotação de tela no Flutter para otimizar a usabilidade em dispositivos portáteis.

RNF3: Desempenho - Atualizar métricas a cada 15 segundos sem sobrecarga, com *timeouts* de 10 segundos em requisições API.

RNF4: Escalabilidade - Suportar múltiplos servidores simultâneos.

RNF5: Compatibilidade - Funcionar em navegadores modernos (*web*) e dispositivos Android/iOS (*mobile*). Integração exclusiva com servidores Windows Server de 2016 até 2025 via WinRM.

RNF6: Confiabilidade - Implementar tratamentos de erros e logs para depuração, garantindo recuperação automática em falhas de conexão.

5.2 Seleção de tecnologias

A escolha das tecnologias baseou-se em uma avaliação qualitativa de critérios como escalabilidade, compatibilidade com ambientes Windows e facilidade de integração, por meio de análises comparativas e revisão de relatos de usuários em comunidades *open-source*. Priorizaram-se ferramentas de código aberto para reduzir custos e facilitar a manutenção. Essa seleção foi crucial para alinhar o *backend*, *frontend* e a versão *mobile*, assegurando um fluxo de dados eficiente e uma experiência de usuário consistente, com base em uma interpretação subjetiva das vantagens percebidas em cenários reais de TI.

A seleção de tecnologias foi guiada pela necessidade de criar um sistema robusto e acessível, capaz de atender tanto a ambientes *web* quanto *mobile*. Cada ferramenta foi escolhida com base em sua capacidade de integração e suporte a funcionalidades específicas.

5.2.1 Linguagem de programação: A linguagem de programação Python foi selecionada para o *backend* devido à sua sintaxe clara e ao vasto ecossistema de bibliotecas, facilitando tarefas como processamento de dados e comunicação remota. Sua versatilidade permitiu uma implementação ágil, especialmente em scripts para automação de ações em servidores.

5.2.2 *Framework* para *Backend*: O Flask, um *framework* leve para Python, foi utilizado para construir a API, pois permite flexibilidade na criação de *endpoints* para ações remotas. Essa simplicidade acelerou o desenvolvimento sem comprometer a escalabilidade.

5.2.3 Banco de dados: O SQLite foi selecionado como banco de dados por sua leveza e simplicidade, permitindo armazenar dados de configuração sem a necessidade de um servidor dedicado, o que facilita a implantação em ambientes variados.

5.2.4 Dashboard e coleta de métricas: O Grafana foi integrado para a visualização interativa de métricas, permitindo a criação de painéis personalizados que facilitam a análise de dados, incluindo o uso de alertas para notificações em tempo real sobre anomalias detectadas. O Telegraf atua como agente de coleta, capturando métricas de servidores Windows de forma eficiente e enviando-as para armazenamento e visualização no Grafana. Sua configuração flexível garante compatibilidade com o ambiente monitorado e a integração com o Grafana assegura um fluxo contínuo de dados para exibição em tempo real.

5.2.5 Frontend: O React foi empregado para a interface *web* para proporcionar uma experiência dinâmica e responsiva. Já o Flutter foi escolhido para a versão *mobile*, permitindo o desenvolvimento *cross-platform* com suporte a funções nativas, como rotação de tela, garantindo usabilidade em dispositivos móveis.

5.2.6 Comunicação remota: O WinRM foi adotado para a execução segura de comandos em servidores Windows, utilizando protocolos criptografados para proteger as interações remotas e evitar vulnerabilidades.

5.3 Modelagem do sistema

A modelagem do sistema foi realizada por meio de diagramas de casos de uso e fluxos de dados, com ênfase na interpretação de interações entre usuários e servidores, destacando narrativas de cenários reais de monitoramento e ações corretivas. Utilizaram-se ferramentas como UML (*Unified Modeling Language*) para mapear essas interações, permitindo uma análise de potenciais gargalos e otimizando a arquitetura para suportar múltiplos servidores de forma segura. Essa etapa incorporou *feedback* qualitativo inicial para refinar as representações, garantindo que o modelo refletisse as percepções e necessidades identificadas na fase de requisitos. O modelo gerado pode ser observado na Figura 3.

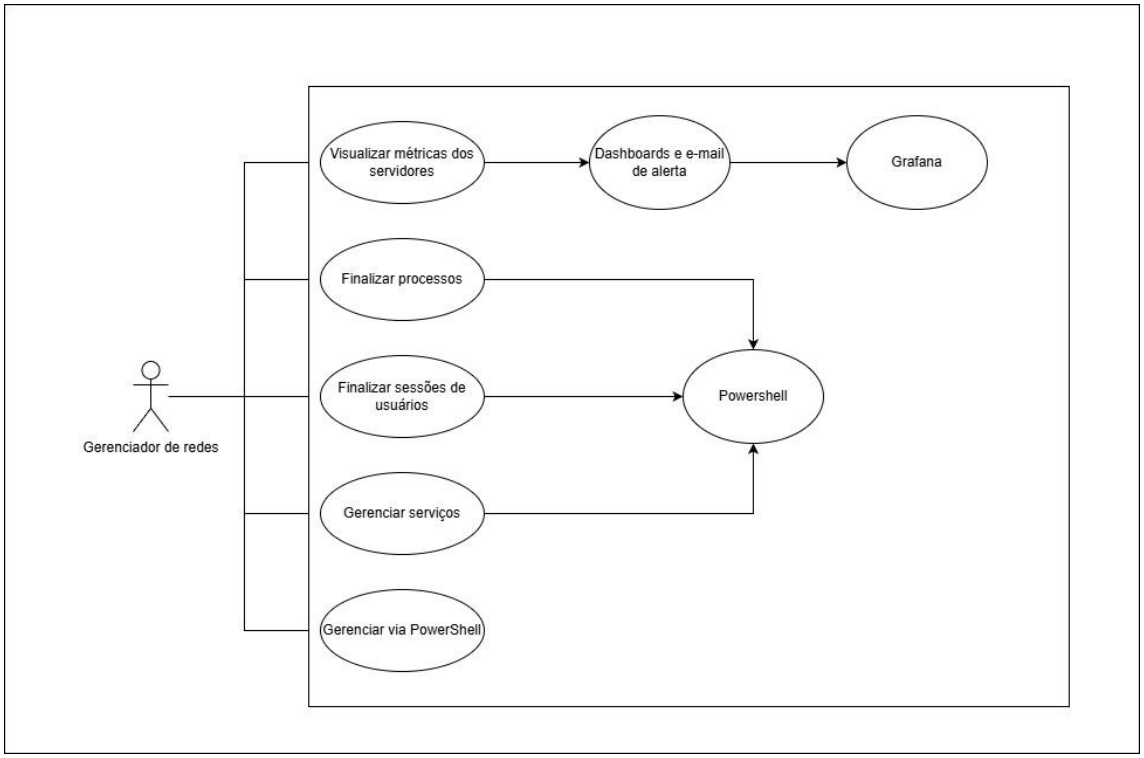

Figura 3. Diagrama de Caso de Uso do Sistema de Monitoramento e Gerenciamento

Fonte: Autoria própria

5.4 Implementação

A implementação ocorreu de forma modular e iterativa, começando pela configuração da coleta de métricas e prosseguindo para o desenvolvimento do *backend* responsável pelas ações remotas. Em paralelo, o *frontend web* e a aplicação *mobile* foram construídos para exibir dashboards interativos e permitir comandos intuitivos, com avaliações qualitativas intercaladas para capturar percepções sobre a usabilidade durante o processo. Testes preliminares foram integrados, utilizando análise de *feedback* para identificar e corrigir inconsistências precocemente.

5.5 Testes e validação

Os testes foram planejados em níveis variados, incluindo unitários para validar funções isoladas, integrados para verificar a comunicação entre módulos e de carga para simular cenários de alto tráfego. Complementando essa abordagem, incorporaram-se testes qualitativos de usabilidade, como sessões de observação com usuários para coletar relatos sobre a experiência intuitiva e a efetividade das ações corretivas. Essa validação abrangente garantiu a robustez do sistema, com foco especial na compatibilidade *mobile*, como adaptação a rotação de tela e desempenho em redes variáveis, interpretando os dados qualitativos para refinamentos finais e confirmação da adequação às demandas identificadas.

6. Desenvolvimento

O desenvolvimento do sistema seguiu uma abordagem estruturada, dividida em etapas que permitiram uma progressão lógica e iterativa. Cada fase incorporou testes iniciais para refinar o produto final.

O ambiente de testes foi configurado em um computador físico executando Windows 11 Pro, com o Hyper-V instalado para hospedar máquinas virtuais dedicadas. Essas incluíram instâncias para o Grafana, o sistema principal (*backend* e *frontend*), servidores clientes simulados executando Windows Server 2016, 2019, 2022 e 2025, como mostrado ilustrado na Figura 4. Além disso, foram configuradas regras de conexões remotas na rede e realizados testes para simular cenários reais de operação híbrida, proporcionando uma base sólida que facilitou o avanço das etapas subsequentes de implementação e validação.

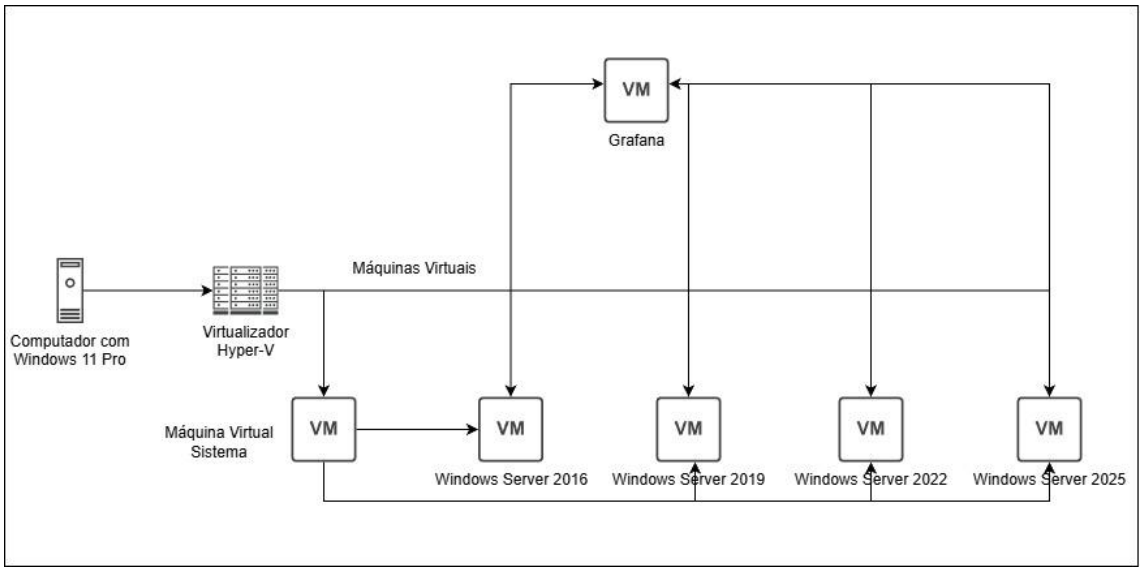

Figura 4. Modelo de virtualização adotado

Fonte: Autoria própria

A coleta e o armazenamento de métricas foram implementados primeiro, configurando agentes para capturar dados como uso de CPU e memória, com integração ao banco para persistência e análise posterior.

O *backend* foi desenvolvido para suportar comandos remotos, como a finalização de processos, com mecanismos de autenticação para garantir operações seguras em múltiplos servidores.

As dashboards foram criadas utilizando ferramentas de visualização, as quais incluem painéis que exibem métricas por meio de gráficos intuitivos, facilitando a identificação de padrões e anomalias. Dentre as diversas opções de alertas disponíveis, optou-se pelo envio via e-mail, no qual o valor considerado anormal é definido manualmente pelo usuário, com base no campo específico a ser monitorado.

A interface *web* e a aplicação *mobile* foram construídas em paralelo, com foco em responsividade e usabilidade. A versão *mobile*, em particular, incorpora recursos nativos para uma experiência fluida em dispositivos portáteis.

Testes foram conduzidos para validar a funcionalidade, desempenho e segurança, incluindo simulações de falhas para assegurar a resiliência do sistema em condições adversas.

A implantação envolveu a configuração para ambientes reais, com hospedagem do *backend* e ajustes para acesso externo, garantindo disponibilidade e escalabilidade.

O sistema desenvolvido oferece uma interface intuitiva para o gerenciamento remoto de servidores, iniciando pela tela de listagem de servidores, que exibe o *status* de cada um (*online* ou *offline*), como ilustrado na Figura 5. Essa tela é complementada por elementos de navegação essenciais, como um botão *Home* para retorno à página inicial, um botão de filtragem por *status* para facilitar a busca rápida e um botão de logout para encerrar a sessão de forma segura.

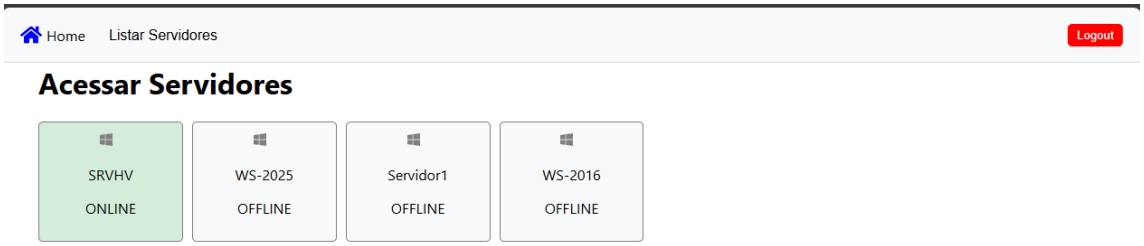

Figura 5. Tela de listagem de servidores

Fonte: Autoria própria

Na tela principal de gerenciamento remoto, os usuários podem visualizar dados integrados do Grafana, proporcionando uma visão em tempo real do desempenho do servidor. Essa interface centraliza opções de ação, permitindo acessar funcionalidades como visualização de processos em execução, verificação de usuários conectados, consulta de serviços instalados, acesso ao PowerShell para comandos avançados, além de comandos para reiniciar ou desligar o servidor como visto na Figura 6, promovendo uma gestão integrada.

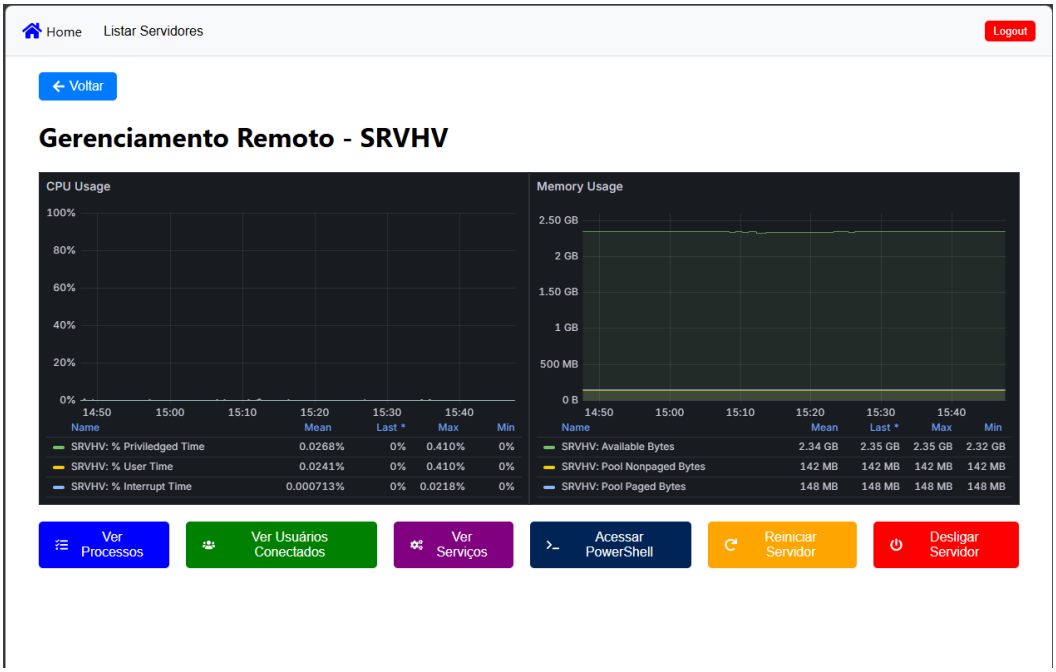

Figura 6. Tela de gerenciamento do servidor.

Fonte: Autoria própria

Para análises mais detalhadas, a tela de processos (Figura 7) apresenta gráficos de uso de CPU e memória, acompanhados de uma tabela listando os processos ativos, o que auxilia na identificação daqueles com maior consumo de recursos. Os usuários podem finalizá-los diretamente via botão dedicado, otimizando o desempenho do sistema. Similarmente, a tela de usuários conectados exibe os mesmos gráficos, com uma tabela detalhando ID de sessão, estado (ativo ou inativo), hora de login e opções para desconectar usuários, facilitando o controle de acessos e a manutenção da segurança.

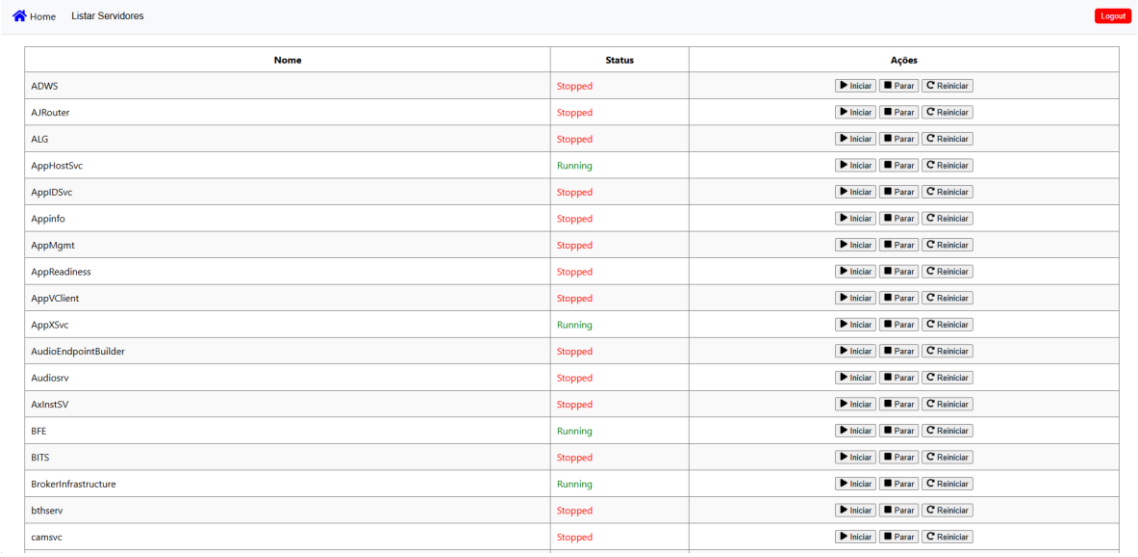

Figura 7. Tela de gerenciamento de serviços

Fonte: Autoria própria

A tela de gerenciamento de serviços lista todos os serviços instalados no servidor, informando nome e *status* de cada um, com botões para iniciar, parar ou reiniciar operações, permitindo intervenções rápidas e precisas. Já a tela de gerenciamento via PowerShell como podemos visualizar na Figura 8 fornece uma interface de linha de comando interativa, permitindo a execução de comandos em uma sessão remota do servidor, o que abrange praticamente todas as tarefas de administração, desde configurações até *troubleshooting* avançado.

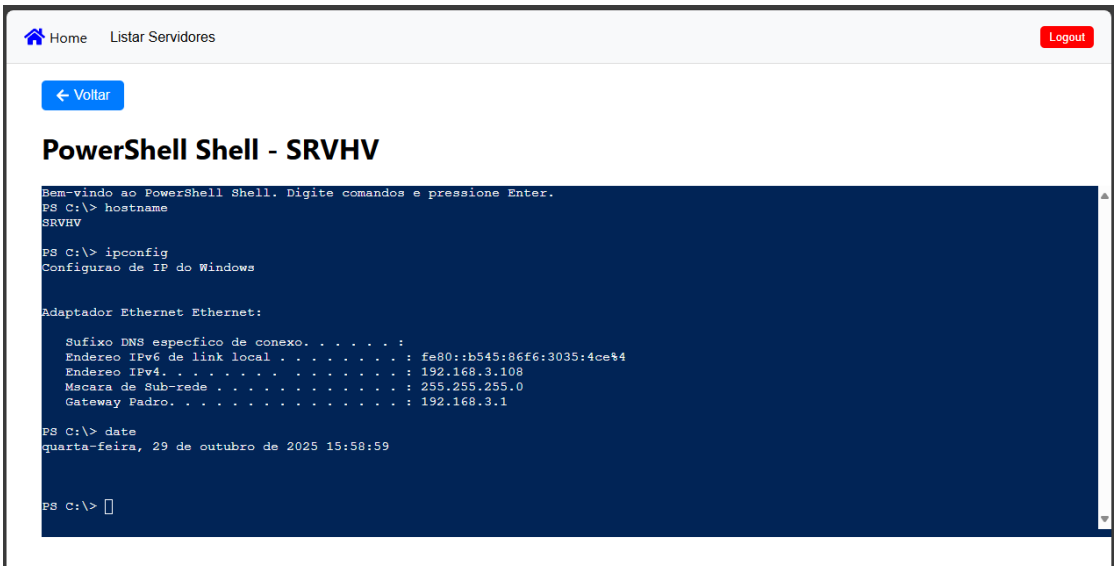

Figura 8. Tela de gerenciamento do powershell.

Fonte: Autoria própria

Por fim, as telas dedicadas ao reinício do servidor e ao desligamento do sistema priorizam a simplicidade, apresentando apenas uma interface de confirmação com opções para prosseguir ou cancelar a ação como observado na Figura 9, minimizando riscos de operações acidentais e assegurando um processo controlado e seguro.

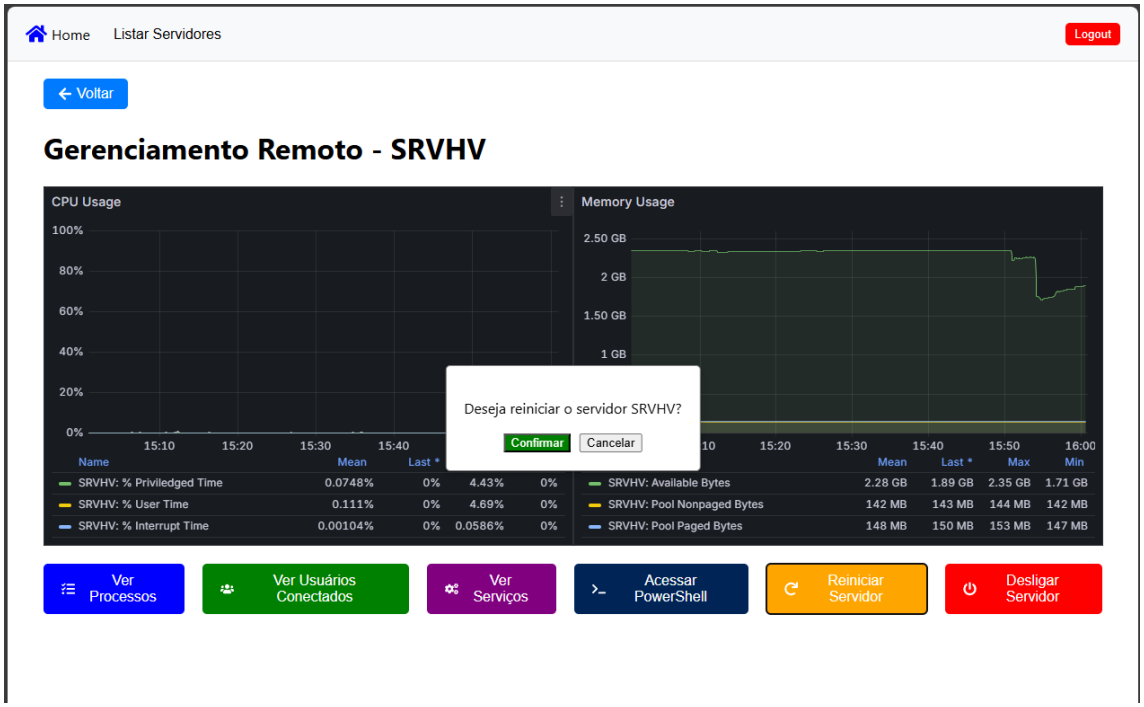

Figura 9. Tela de "Reiniciar Servidor" com campo de confirmação.

Fonte: Autoria própria

As figuras apresentadas foram capturadas na aplicação *web*, no entanto, a versão *mobile* segue o mesmo *layout* como podemos ver na Figura 10, preservando os princípios fundamentais de usabilidade e funcionalidade. Devido ao desenvolvimento individual e independente do React utilizado no *frontend web*, a interface *mobile* apresenta leves diferenças estéticas, as quais não comprometem o funcionamento da ferramenta, garantindo uma experiência consistente e alinhada com a versão *web*.

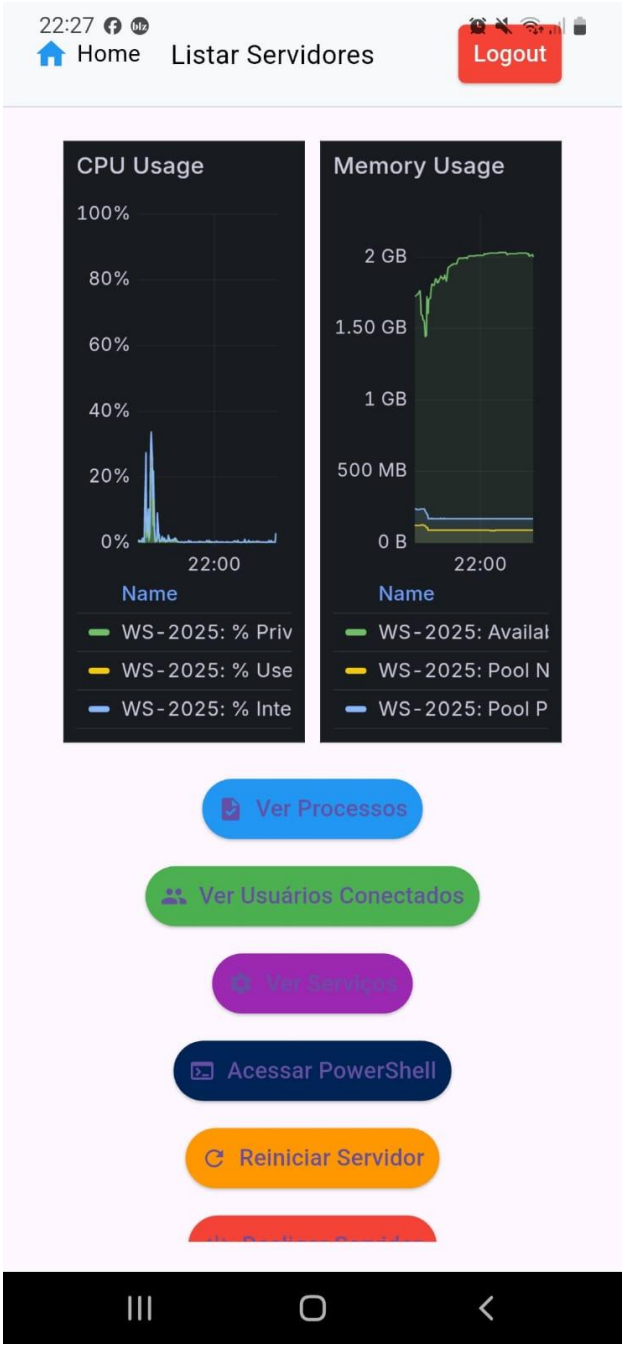

Figura 10. Tela de gerenciamento de servidor na versão *mobile*.

Fonte: Autoria própria

O aplicativo tem como foco atuar quando alguma métrica monitorada no Grafana apresenta status anormal, sinalizando a necessidade de intervenção no servidor para resolução do problema. Para tanto, foram configurados alertas diretamente no Grafana, com o objetivo de enviar e-mails notificando o usuário sempre que uma métrica se torne anormal. Como exemplo temos a Figura 11, onde o alerta foi definido para disparar quando o servidor monitorado apresentar menos de 2 GB de memória disponível,

incorporando layouts específicos para os e-mails: um *template* dedicado ao alerta de incidente, com detalhes sobre a anomalia detectada, e outro para a resolução como ilustrado na Figura 12, confirmando a normalização da métrica.

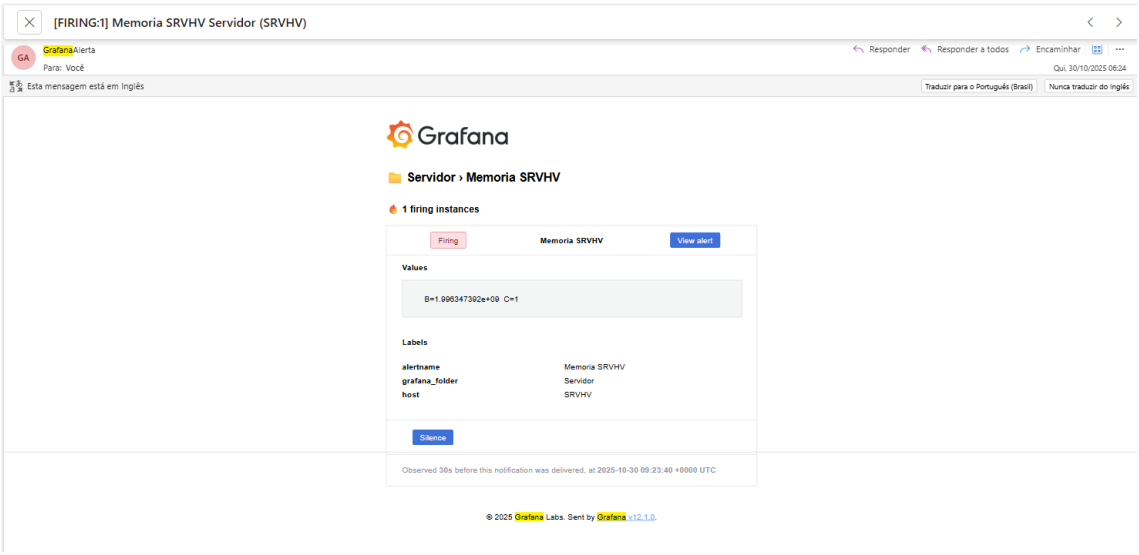

Figura 11. Mensagem de alerta recebida por e-mail.

Fonte: Autoria própria

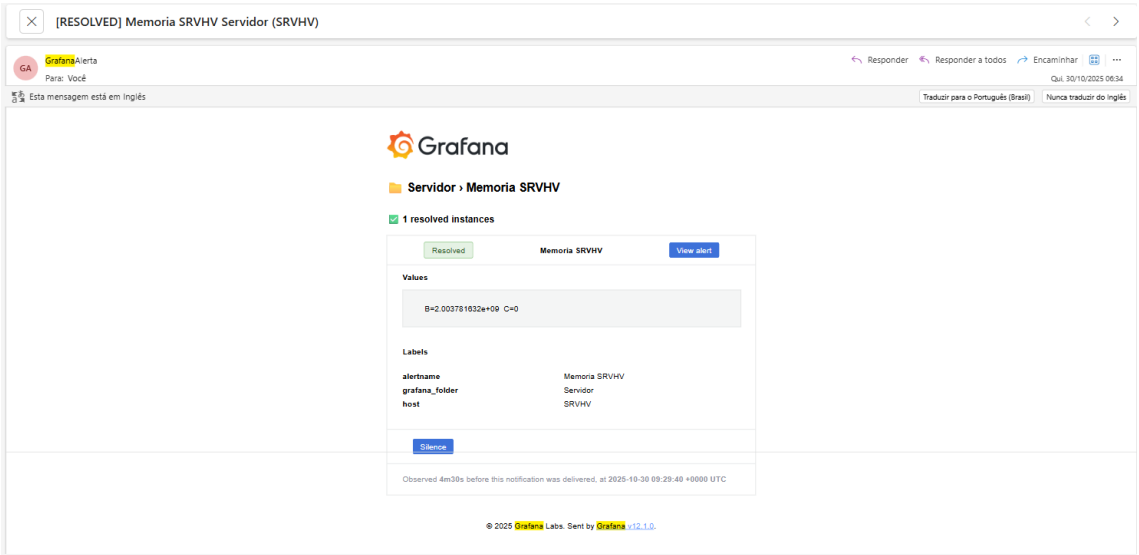

Figura 12. Mensagem de status “Resolvido” do Grafana.

Fonte: Autoria própria

7. Conclusão

No contexto de ambientes corporativos com servidores Windows, a viabilidade do aplicativo proposto para gerenciamento remoto pode ser justificável em cenários operacionais, destacando melhorias em eficiência, usabilidade e redução de complexidade procedimental.

Em um cenário hipotético onde o administrador de redes precisa agir perante um alerta, ele enfrenta uma sequência multifásica para responder ao incidente: inicializar um *notebook* (assumindo portabilidade imediata), estabelecer conexão via VPN (*Virtual Private Network*) para acesso à rede local e subsequentemente, interagir com o servidor afetado, o que introduz atrasos inerentes, dependência de *hardware* específico e potenciais interrupções em mobilidade. Em contraste, o aplicativo *mobile* permite uma resposta simplificada: desbloquear o dispositivo móvel, autenticar-se e acessar o aplicativo que fornece métricas e ações corretivas em tempo real, promovendo agilidade operacional, minimização de dependências externas e maior resiliência em ambientes híbridos ou remotos.

Essa transição qualitativa não apenas otimiza o fluxo de trabalho, reduzindo erros humanos associados a configurações complexas, mas também a efetividade geral do gerenciamento, alinhando-se a princípios de *design* centrado no usuário em aplicações móveis para infraestrutura de TI. Assim, a adoção do aplicativo pode ser uma alternativa viável ao mitigar barreiras tradicionais, fomentando uma administração mais intuitiva e acessível, conforme evidenciado em estudos sobre avaliações qualitativas de ferramentas mobile em contextos semelhantes, pavimentando o caminho para futuras expansões em automação e integração com tecnologias emergentes (Palumbo, 2020).

Para confirmar ainda mais a relevância da aplicação proposta, realiza-se uma comparação com ferramentas semelhantes, destacando seus diferenciais em termos de simplicidade, integração nativa e foco em ambientes Windows híbridos.

O Grafana destaca-se como uma plataforma *open-source* para visualização de métricas em tempo real com dashboards e alertas, mas não coleta dados nativamente, dependendo de integrações como Telegraf e apresenta limitações em controle de acesso baseado em roles na versão gratuita, o que pode comprometer a segurança em equipes distribuídas. O sistema proposto supera essas restrições ao incorporar coleta direta via WinRM (status de servidores, processos, serviços e usuários) e ações corretivas como reinício ou finalização de processos, além de um shell interativo, funcionalidades ausentes no Grafana (Hernández, 2022).

O Zabbix oferece monitoramento escalável com agentes para Windows, coletando métricas como CPU e memória, mas exige instalação de agentes e configuração extensa para automações simples, o que aumenta a complexidade em implantações iniciais. Em contraste, o sistema desenvolvido opera sem agentes via WinRM, incluindo ações interativas como *logoff* de usuários e gerenciamento de serviços, complementadas por um shell PowerShell nativo, função esta não disponível diretamente no Zabbix (Etruti, et al., 2018).

O sistema inova com WinRM nativo para automação PowerShell sem RDP (*Remote Desktop Protocol*), combinando monitoramento, ações corretivas e shell. Sua autenticação granular e Flutter para mobilidade atendem tendências de trabalho remoto

em 2025, oferecendo uma alternativa acessível e ágil para equipes menores, sem as complexidades *enterprise*, pavimentando o caminho para futuras expansões em automação e integração com tecnologias emergentes. (Ilag, 2021)

Essa aplicação permite que os usuários monitorem servidores, visualizem dashboards e executem ações diretamente de dispositivos móveis, como *smartphones* e *tablets*. A integração entre as versões *web* e *mobile* garante uma experiência unificada, facilitando o gerenciamento em movimento sem comprometer a segurança ou o desempenho.

8. Trabalhos Futuros

Com base nos resultados obtidos e nas limitações identificadas no sistema atual, propõem-se expansões futuras para aprimorar sua funcionalidade, segurança e abrangência. Inicialmente, planeja-se a implementação de um módulo de cadastro de usuários com delimitação granular de permissões, permitindo a criação de perfis personalizados além dos *roles* existentes ("admin" e "viewer"), o que facilitaria a adaptação a equipes maiores e hierarquias organizacionais complexas, promovendo maior controle de acesso e conformidade com padrões de governança de TI.

Adicionalmente, a inclusão de um sistema de auditoria integrado é prevista, registrando detalhes como identidade do usuário logado, horário de login, ações executadas (tais como reinícios de servidores, finalizações de processos ou comandos no shell PowerShell) e alterações realizadas, o que contribuiria para rastreabilidade, detecção de anomalias e cumprimento de requisitos regulatórios, elevando o nível de responsabilidade e segurança em ambientes corporativos.

Por fim, pretende-se estender o suporte ao monitoramento de servidores Linux, integrando protocolos como SSH para coleta de métricas e ações administrativas, permitindo uma gestão unificada em infraestruturas híbridas que combinem ambientes Windows e Linux, alinhando-se às tendências de heterogeneidade em data centers modernos e ampliando a aplicabilidade do sistema em cenários diversificados. Essas melhorias pavimentarão o caminho para uma evolução contínua da aplicação, tornando-a mais robusta e adaptável às demandas futuras do gerenciamento remoto de servidores.

Referências

Ahmed, S. e Mahmood, Q. (2019) "An authentication based scheme for applications using JSON web token", em *2019 22nd International Multitopic Conference (INMIC)*. *2019 22nd International Multitopic Conference (INMIC)*, Islamabad, Pakistan: IEEE, p. 1–6. Disponível em: https://doi.org/10.1109/INMIC48123.2019.9022766.

Alhamazani, K. *et al.* (2014) "CLAMS: Cross-layer Multi-cloud Application Monitoring-as-a-Service Framework", em *2014 IEEE International Conference on Services Computing*. *2014 IEEE International Conference on Services Computing (SCC)*,

Anchorage, AK, USA: IEEE, p. 283–290. Disponível em: https://doi.org/10.1109/SCC.2014.45.

Anerousis, N. *et al.* (2005) "Health monitoring and control for application server environments", em *2005 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005. IM 2005.*, Nice, France: IEEE, p. 75–88. Disponível em: https://doi.org/10.1109/INM.2005.1440772.

Ayoob, A., Khalil, G. e Aksoy, M. (2024) "The Importance of Cloud Monitoring Services with A Comprehensive Review of Essential Tools", em *2024 IEEE Asia-Pacific Conference on Geoscience, Electronics and Remote Sensing Technology (AGERS)*. *2024 IEEE Asia-Pacific Conference on Geoscience, Electronics and Remote Sensing Technology (AGERS)*, Manado, Indonesia: IEEE, p. 118–126. Disponível em: https://doi.org/10.1109/AGERS65212.2024.10932924.

Dwivedi, P., Kshamta e Joshi, A. (2022) "ReactJS For Trading Applications", em *2022 International Conference on Cyber Resilience (ICCR)*. *2022 International Conference on Cyber Resilience (ICCR)*, Dubai, United Arab Emirates: IEEE, p. 01–07. Disponível em: https://doi.org/10.1109/ICCR56254.2022.9995932.

Faiz, M.A., Kusumo, D.S. e Alibasa, M.J. (2022) "Flutter Framework Code Portability Measurement on Multiplatform Applications with ISO 9126", em *2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT)*. *2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT)*, Bandung, Indonesia: IEEE, p. 36–40. Disponível em: https://doi.org/10.1109/ICoSEIT55604.2022.10030045.

Hitesh Mohapatra (2020) "Automation and Configuration Management Windows PowerShell". Disponível em: https://doi.org/10.13140/RG.2.2.35853.90085.

Ilag, B.N. (2021) "Tools and Technology for Effective Remote Work", *International Journal of Computer Applications*, 174(21), p. 13–16. Disponível em: https://doi.org/10.5120/ijca2021921109.

Juan Hernández, R. (2022) *Building IoT Visualizations using Grafana: Power up your IoT projects and monitor with Prometheus, LibreNMS, and Elasticsearch*. Packt Publishing.

Maduranga, K.A.M. (2016) "Investigate Windows Management Instrumentation (WMI)Attacks in Windows Operating Systems".

Majumdar, S. (2024) *Resource Management on Distributed Systems: Principles and Techniques*. 1º ed. Wiley. Disponível em: https://doi.org/10.1002/9781119912965.

Ozer, M. *et al.* (2020) "Cloud Incident Response: Challenges and Opportunities", em *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. *2020 International Conference on Computational Science and*

Computational Intelligence (CSCI), Las Vegas, NV, USA: IEEE, p. 49–54. Disponível em: https://doi.org/10.1109/CSCI51800.2020.00015.

Palumbo, F. (2020) "CLOUD AND MOBILE INFRASTRUCTURE MONITORING FOR LATENCY AND BANDWIDTH SENSITIVE APPLICATIONS".

Petruti, C.-M. *et al.* (2018) "Automatic Management Solution in Cloud Using NtopNG and Zabbix", em *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Cluj-Napoca: IEEE, p. 1–6. Disponível em: https://doi.org/10.1109/ROEDUNET.2018.8514142.

Rohit, R. *et al.* (2023) "Web Application Security Testing Framework using Flask", em *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India: IEEE, p. 1646–1652. Disponível em: https://doi.org/10.1109/ICAAIC56838.2023.10140422.

Vijayasekaran, G., K, H.H. e M, H.C. (2025) "Server Error Control in Distributed Systems: A Comprehensive Review", em *2025 Third International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. *2025 Third International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, Trichy, India: IEEE, p. 805–815. Disponível em: https://doi.org/10.1109/ICAISS61471.2025.11041977.

Ditarso, P. *et al.* (2008) "On the planning of a hybrid IT infrastructure", em *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*. *NOMS 2008 - 2008 IEEE Network Operations and Management Symposium*, Salvador, Bahia, Brazil: IEEE, p. 496–503. Disponível em: https://doi.org/10.1109/NOMS.2008.4575173.

Hidayat, T. *et al.* (2025) "Reinforcement Learning-Driven Hybrid Precopy/Postcopy VM Migration for Energy-Efficient Data Centers", *IEEE Access*, 13, p. 169521–169533. Disponível em: https://doi.org/10.1109/ACCESS.2025.3613235.

Melo, C. *et al.* (2017) "Synchronization server infrastructure: A relationship between system downtime and deployment cost", em *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. *2017 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Banff, AB: IEEE, p. 1250–1255. Disponível em: https://doi.org/10.1109/SMC.2017.8122784.

Palshikar, G.K. *et al.* (2012) "Streamlining Service Levels for IT Infrastructure Support", em *2012 IEEE 12th International Conference on Data Mining Workshops*. *2012 IEEE 12th International Conference on Data Mining Workshops*, Brussels, Belgium: IEEE, p. 309–316. Disponível em: https://doi.org/10.1109/ICDMW.2012.118.

Brabant, Bart Van and Wouter Joosen. “Integrated management of network and security devices in IT infrastructures.” *2011 7th International Conference on Network and Service Management* (2011): 1-5.

Vyshnavi, V.R. e Malik, A. (2019) “Efficient Way of Web Development Using Python and Flask”, 6(2).

Zabbix. (2025). Documentação. Disponível em: https://www.zabbix.com/br/manuals. Acesso em: 09 novembro 2025.

Grafana Labs. (2025). Technical documentation. Disponível em: https://grafana.com/docs. Acesso em: 09 novembro 2025.

Nagios. (2025). Nagios Documentation. Disponível em: https://www.nagios.org/documentation/. Acesso em: 09 novembro 2025.

Paessler. (2025). PRTG Manual. Disponível em: https://www.paessler.com/manuals/prtg. Acesso em: 09 novembro 2025.

Prometheus. (2025). Overview. Disponível em: https://prometheus.io/docs/introduction/overview/. Acesso em: 09 novembro 2025.