INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DE PERNAMBUCO

Campus Recife

Departamento Acadêmico de Cursos Superiores

Tecnologia em Análise e Desenvolvimento de Sistemas

WESLEY MATHEUS VAUTHIER

TROCASENHA: Um sistema online para manutenção automática e eficiente de senhas

Recife

2023

WESLEY MATHEUS VAUTHIER

TROCASENHA: Um sistema online para manutenção automática e eficiente de senhas

Trabalho de conclusão de curso apresentado ao Departamento Acadêmico de Cursos Superiores do Instituto Federal de Ciência e Tecnologia de Pernambuco, como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Paulo Abadie Guedes

Recife

2023

V381t
2023 Vauthier, Wesley Matheus.
Trocasenha: um sistema online para manutenção automática e eficiente de senhas / Wesley Matheus Vauthier. --- Recife: O autor, 2023.
52f. il. Color.

TCC (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Pernambuco, Recife, 2023.

Inclui Referências
Orientador: Professor M.e Paulo Abadie Guedes.

1. Desenvolvimento de sistemas. 2. Service Desk. 3.Gerenciamento de senha. 4. Linguagens de programação. I. Título. II. Guedes, Paulo Abadie (orientador). III. Instituto Federal de Pernambuco.

CDD 003 (22 ed.)

Catalogação na fonte: Emmely Silva CRB4/1876

Trabalho de Conclusão de Curso apresentado pelo estudante Wesley Matheus Vauthier à coordenação de Análise e Desenvolvimento de Sistemas, do Instituto Federal de Pernambuco, sob o título de “TROCASENHA: UM SISTEMA ONLINE PARA MANUTENÇÃO AUTOMÁTICA E EFICIENTE DE SENHAS”, orientado pelo Prof. Ms. Paulo Abadie Guedes e aprovado pela banca examinadora formada pelos professores:

Recife, 08 de Maio de 2023.

Prof. Ms. Paulo Abadie Guedes

CTADS/DACS/IFPE

Prof. Dr. Henrique Correia Torres Santos.

CTADS/DACS/IFPE

Prof. Ms. Hilson Gomes Vilar de Andrade

DACT/IFPE

Aluno: Wesley Matheus Vauthier

Dedico este trabalho ao meu Deus,

minha esposa Rute e meu filho Benjamin

AGRADECIMENTOS

Expresso minha profunda gratidão a Deus, que constantemente me presenteia com sua Graça Divina, permitindo-me superar os desafios que surgem em meu caminho, mesmo sem merecer.

Também sou grato a todas as pessoas que contribuíram de forma direta ou indireta para a minha formação, especialmente à minha família, minha esposa Rute e meu filho Benjamim, que sempre acreditaram no poder transformador do conhecimento e no meu potencial. Seu apoio e encorajamento têm sido inestimáveis ao longo da minha jornada.

RESUMO

O TrocaSenha é um sistema desenvolvido em parceria com a Stefanini Group que permite aos colaboradores realizar a troca, reset e desbloqueio de suas senhas de forma simples e segura, sem a necessidade de entrar em contato com o *Service Desk* por telefone. Essa plataforma é uma solução eficiente para atender a grande demanda de chamados recebidos no *Service Desk*, que muitas vezes são relacionados a problemas com senhas. Com o TrocaSenha, os usuários podem acessar o sistema de qualquer dispositivo com acesso à internet, tornando-o bastante acessível. Além disso, a plataforma oferece uma experiência rápida e segura para os usuários, permitindo que eles resolvam seus problemas de senha sem precisar esperar pelo atendimento telefônico do *Service Desk*. O sistema TrocaSenha é uma ferramenta importante para otimizar os processos de gerenciamento de senhas na empresa Stefanini Group.

Palavras-chave: Sistema; Trocar Senha; Acesso; Segurança; Arquitetura; Autenticação; Troca; Reset; Desbloqueio; Chamado;

ABSTRACT

TrocaSenha is a system developed in partner with Stefanini Group that allows employees to change, reset, and unlock their passwords in a simple and secure way, without the need to contact the service desk by phone. This platform is an efficient solution to meet the high demand for calls received in the Service Desk, which are often related to password issues. With TrocaSenha, users can access the system from any device with internet access, making it very accessible. Additionally, the platform provides a fast and secure experience for users, allowing them to solve their password issues without waiting for Service Desk phone support. The TrocaSenha system is an important tool to optimize password management processes in Stefanini Group.

Palavras-chave: System; Change password; Access; Security; Architecture; Change; Reset; Unlock; Request;

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O objetivo deste trabalho é apresentar uma descrição detalhada do processo de desenvolvimento do sistema TrocaSenha, cujo propósito principal é permitir a troca, reset e desbloqueio de senha na empresa Stefanini. Além disso, será apresentado um estudo sobre os resultados da implementação do sistema, incluindo uma análise da redução da quantidade de chamados na equipe de atendimento e a melhoria na velocidade e fluidez do processo de gerenciamento de senha.

A Stefanini é uma empresa global de tecnologia fundada em 1987, com sede no Brasil e presença em mais de 40 países. A empresa oferece serviços de consultoria, desenvolvimento de *software*, terceirização de processos de negócios (BPO) e soluções de tecnologia para diversas indústrias, incluindo finanças, telecomunicações, energia e varejo. A Stefanini é conhecida por sua experiência em tecnologias emergentes, como inteligência artificial, automação de processos robóticos, *blockchain* e nuvem.

A empresa é uma das principais companhias de tecnologia do Brasil, com um forte histórico de crescimento e inovação. Seus serviços são projetados para ajudar os clientes a se adaptarem às mudanças do mercado e às novas tecnologias, com soluções personalizadas que melhoram a eficiência e reduzem custos. Além disso, a Stefanini investe em pesquisa e desenvolvimento, colaborando com *startups* e universidades para criar soluções inovadoras e disruptivas que agregam valor aos negócios de seus clientes.

A Stefanini identificou a necessidade de um sistema de reset, desbloqueio e troca de senha para atender a um grande volume de chamados recebidos no *Service Desk*. A empresa buscava uma solução mais eficiente e menos dispendiosa para gerenciar esses chamados, que poderiam ser automatizados e permitir que os próprios usuários realizassem essas tarefas.

Com um sistema automatizado, a empresa poderia reduzir significativamente o número de chamados abertos e desafogar a equipe de atendimento, permitindo que eles se dedicassem a outras tarefas mais complexas e de maior valor agregado.

Além disso, a empresa também poderia reduzir seus custos, eliminando a necessidade de atendentes dedicados apenas a atender esses chamados.

O novo sistema também pode melhorar a experiência do usuário, permitindo que ele resolva seus problemas mais rapidamente, sem a necessidade de esperar pelo atendimento via telefone.

1.1 Objetivo Geral

Descrever o processo de desenvolvimento do sistema TrocaSenha, sua implementação e análise dos resultados na empresa Stefanini.

1.2 Objetivos específicos

- Analisar as necessidades da empresa em relação ao gerenciamento de senhas;
- Definir as funcionalidades e requisitos do sistema TrocaSenha;
- Desenvolver o sistema com base nas melhores práticas de segurança da informação;
- Testar o sistema para garantir sua eficiência e segurança;
- Implantar o sistema na empresa Stefanini e realizar treinamentos para os colaboradores;
- Analisar os resultados da implementação do sistema, incluindo a redução da quantidade de chamados no *Service Desk* e a melhoria na velocidade e fluidez do processo de gerenciamento de senha;
- Avaliar a satisfação dos usuários com o sistema e identificar possíveis melhorias a serem implementadas.

2 FUNDAMENTAÇÃO TEÓRICA

Para entender melhor este trabalho, é importante ter uma compreensão básica das principais tecnologias, ferramentas e linguagens de programação que serão mencionadas.

2.1 Linguagens de Programação

Neste tópico, serão apresentadas as linguagens de programação utilizadas no desenvolvimento do sistema TrocaSenha, descrevendo as suas principais características e funcionalidades. Serão discutidos aspectos como a facilidade de uso, a flexibilidade, a escalabilidade e a performance de cada linguagem, de forma a compreender as razões pelas quais foram escolhidas para o projeto.

2.1.1 Javascript

O Javascript é uma linguagem de programação de alto nível que é usada principalmente para criar interatividade em páginas *web*. Ele permite que os desenvolvedores criem *scripts* que podem ser executados no lado do cliente, no navegador *web*, para manipular a página e interagir com o usuário. (SILVA, J. A., 2019)

A versatilidade e a popularidade do Javascript fazem com que ele seja uma das principais linguagens de programação utilizadas no desenvolvimento *web* atualmente, tendo uma forte presença em diversos aspectos do desenvolvimento, desde o *frontend* até o *backend* das aplicações (Macedo et al., 2019).

2.2 Outras linguagens

2.2.1 HTML

Segundo Menezes (2019), HTML, ou *Hypertext Markup Language*, é uma linguagem de marcação usada para criar documentos *web*. Sendo usada para

estruturar e formatar conteúdo da web, como textos, imagens, vídeos, formulários e muito mais. De acordo com Silva e Wirth (2018), o HTML é composto por elementos que possuem tags de abertura e fechamento, e essas tags são utilizadas para informar ao navegador como o conteúdo deve ser exibido. Além disso, o HTML permite a utilização de atributos nos elementos para definir características específicas, como cor, tamanho, entre outras. Conforme destacado por Zeldman (2010), o HTML passou por diversas evoluções ao longo dos anos, sendo a mais recente o HTML5, que trouxe novos elementos semânticos, melhorias na acessibilidade e suporte a recursos multimídia.

2.2.2 CSS

CSS, ou *Cascading Style Sheets*, é uma linguagem de folha de estilo usada para controlar a aparência e o *layout* de documentos *web*. Ele permite que os desenvolvedores definam estilos para elementos HTML, como cores, fontes, tamanhos, posicionamento e outras formatações, tornando a apresentação da página mais atraente e consistente. O CSS é geralmente separado do HTML e incluído em um arquivo externo, o que facilita a manutenção e o reuso de estilos em várias páginas da *web*. (Flôres, 2019)

De acordo com Terzic (2019), "o CSS permite criar estilos diferentes para diferentes elementos HTML na página, permitindo um alto nível de controle sobre a aparência visual do conteúdo da página". Além disso, o CSS também permite que o desenvolvedor crie estilos para um grupo de elementos que compartilham as mesmas características. Isso é feito usando seletores de classe e ID (identificador), que permitem criar estilos para elementos específicos com base em sua classe ou ID.

Outra característica importante do CSS é a capacidade de criar *layouts* complexos usando posicionamento relativo, absoluto e fixo. Como observado por Shah (2019), "o CSS permite que os desenvolvedores posicionem elementos na página com precisão milimétrica usando propriedades de posicionamento como *position, top, left, right e bottom*". Isso permite que o desenvolvedor crie *layouts* complexos e responsivos que se adaptem a diferentes tamanhos de tela.

O CSS também permite criar efeitos visuais, como animações e transições, para tornar as páginas da web mais atraentes e dinâmicas. De acordo com Akyildiz e Tugrul (2018), "o CSS fornece várias propriedades que podem ser usadas para criar efeitos visuais, como gradientes, sombras e bordas arredondadas, além de suportar animações e transições que podem ser usadas para criar efeitos de movimento na página".

2.3 Frameworks

Neste tópico, serão apresentadas as principais frameworks utilizadas no desenvolvimento do projeto em questão. As frameworks são ferramentas que permitem agilizar o processo de desenvolvimento de software, fornecendo recursos e funcionalidades prontas para uso. É importante destacar que a escolha das *frameworks* utilizadas foi feita com base em critérios como desempenho, segurança, facilidade de uso e compatibilidade com as linguagens de programação utilizadas no projeto. Dessa forma, as *frameworks* aqui listadas foram consideradas as mais adequadas para atender às necessidades específicas do TrocaSenha.

2.3.1 Bootstrap

Bootstrap é uma biblioteca de código aberto desenvolvida pela equipe do Twitter para desenvolvimento de sites e aplicativos web responsivos e móveis (MIRANDA, 2021). Ele inclui um conjunto de estilos CSS pré-definidos e componentes JavaScript para criar uma interface de usuário moderna e amigável. O Bootstrap facilita a criação de designs responsivos que se adaptam automaticamente a diferentes tamanhos de tela e dispositivos, sem a necessidade de escrever muito código personalizado. Ele também oferece recursos como *grids*, tipografia, formulários, botões, navegação e muito mais, permitindo que os desenvolvedores criem rapidamente páginas web profissionais e consistentes. O Bootstrap é uma das bibliotecas de frontend mais populares e amplamente usadas para desenvolvimento *web*.

2.3.2 NodeJS

Node.js é uma plataforma baseada em JavaScript que permite o desenvolvimento de aplicações *server-side* (SOUSA; CAMPOS, 2018). Uma de suas principais características é a utilização da linguagem JavaScript tanto no backend quanto no frontend (SILVA; MARQUES; DE FREITAS, 2018).

O modelo de execução do Node.js é baseado em eventos (GOMES; FERREIRA, 2017). Isso significa que o servidor é capaz de lidar com um grande número de conexões simultâneas sem bloquear a execução de outras operações. O Node.js é eficiente e adequado para construir aplicativos em tempo real. (Node.js, s.d.). Além disso, a plataforma utiliza um modelo de entrada e saída não-bloqueante (non-blocking I/O), o que aumenta a eficiência no processamento de requisições (LOPES; LEAL, 2017).

Uma das bibliotecas mais utilizadas no Node.js é o Express.js, um *framework* minimalista que permite o desenvolvimento de aplicações *web* de forma rápida e simples (VIANA; MONTEIRO, 2019). Outras bibliotecas importantes incluem o Socket.IO, para desenvolvimento de aplicações em tempo real, e o Mongoose, para integração com bancos de dados MongoDB (CORREA; CASSIANO, 2020).

O Node.js tem se tornado cada vez mais popular entre desenvolvedores devido à sua eficiência no processamento de requisições, facilidade de desenvolvimento e flexibilidade de uso em diversas aplicações (MARTINS; JÚNIOR; ALMEIDA, 2019). Além disso, sua comunidade ativa e extensa disponibiliza uma grande quantidade de bibliotecas e ferramentas que auxiliam no desenvolvimento de aplicações *web* (CARNEIRO; COSTA, 2018).

2.3.3 Angular

Segundo De Almeida (2018), Angular é uma plataforma de desenvolvimento web em JavaScript, mantida pelo Google, que permite criar aplicativos *web* dinâmicos e escaláveis. De acordo com Chen et al. (2019), o uso de *frameworks* de frontend como o Angular permite uma maior modularidade do código, o que facilita a manutenção e a escalabilidade das aplicações.

Além disso, o Angular utiliza uma arquitetura baseada em componentes, na qual cada componente representa uma parte da interface gráfica da aplicação e é responsável por seu próprio estado. Segundo Khurana e Jain (2018), essa abordagem facilita a construção de aplicações complexas e a reutilização de código em diferentes partes do projeto.

O uso de *templates* é outra característica importante do Angular, permitindo a definição de elementos visuais e a interação com o usuário. De acordo com Sharifzadeh e Hosseini (2018), o uso de templates no Angular permite a separação clara entre a lógica da aplicação e a sua apresentação visual, tornando a manutenção do código mais fácil.

Outra funcionalidade relevante do Angular é a injeção de dependências, que permite a criação de serviços que podem ser compartilhados por diferentes componentes da aplicação. Segundo Mijalković et al. (2017), essa abordagem promove uma maior modularidade e reutilização de código.

Por fim, é importante destacar que o Angular é uma ferramenta em constante evolução, com a comunidade de desenvolvedores sempre buscando melhorias e novas funcionalidades. De acordo com Oliveira et al. (2018), a comunidade ativa e engajada do Angular é um dos fatores que contribuem para a sua popularidade e sucesso no desenvolvimento de aplicações *web* modernas.

2.4 Ferramentas

Neste tópico, serão apresentadas as ferramentas de desenvolvimento utilizadas no projeto em questão. Serão também destacados os principais aspectos que influenciaram na escolha dessas ferramentas e como elas contribuíram para a eficiência e eficácia do processo de desenvolvimento do sistema.

2.4.1 PM2

PM2 é um gerenciador de processos de produção para aplicativos Node.js. Ele é usado para gerenciar a execução de vários processos Node.js em um único

servidor e inclui recursos como balanceamento de carga, monitoramento de recursos e monitoramento de falhas de processo. O PM2 permite que os desenvolvedores gerenciem facilmente aplicativos Node.js em ambiente de produção, garantindo que eles estejam em execução sem interrupções e que possam lidar com picos de tráfego. Ele também oferece recursos de recuperação automática em caso de falha do processo, o que ajuda a manter a disponibilidade dos aplicativos. (MATTIASSICH, 2020)

De acordo com Taibi et al. (2019), o PM2 é uma solução de gerenciamento de processos para aplicativos Node.js, que tem como objetivo garantir alta disponibilidade e escalabilidade. Segundo Peng et al. (2019), o PM2 é uma das soluções mais populares para gerenciamento de processos Node.js em produção. O PM2 oferece recursos avançados, como monitoramento de processos, gerenciamento de *cluster*, balanceamento de carga e recuperação automática de falhas. O PM2 também fornece um conjunto de ferramentas de linha de comando, que facilitam a gestão de aplicativos Node.js em ambientes de produção.

Outra vantagem do PM2 é a sua integração com outros sistemas de gerenciamento de aplicativos. De acordo com Jankovic (2019), o PM2 é capaz de se integrar facilmente com essas plataformas, permitindo que os aplicativos Node.js sejam gerenciados de maneira mais eficiente em ambientes de contêineres.

2.4.2 VSCode

Segundo Kettering et al. (2017), o VSCode é um editor de código-fonte de plataforma cruzada desenvolvido pela Microsoft, que possui uma ampla variedade de recursos, como autocompletar, realce de sintaxe, integração com Git, terminal integrado e extensões personalizadas. Ele é usado para escrever e depurar código em várias linguagens de programação, incluindo JavaScript, Python, C++, Java, entre outras. O VSCode possui uma ampla variedade de recursos, como autocompletar, realce de sintaxe, integração com Git, terminal integrado e extensões personalizadas, que podem ser baixadas e instaladas a partir da sua loja de extensões.

2.4.3 Windows

Windows é um sistema operacional desenvolvido pela Microsoft, usado em computadores pessoais e em alguns dispositivos móveis. (MICROSOFT, 2015) Segundo Chellappan et al. (2018), o Windows é amplamente utilizado devido à sua ampla compatibilidade com software e hardware de terceiros, facilidade de uso e suporte robusto ao usuário. Além disso, o Windows possui uma ampla gama de recursos de segurança, incluindo o *firewall* do Windows, o Windows Defender e o BitLocker.

Uma das principais razões pelas quais o Windows é tão popular é a sua capacidade de suportar uma ampla gama de aplicativos de software. De acordo com Alnajjar e Obeidat (2021), o Windows suporta milhões de aplicativos, desde jogos até software de produtividade.

2.4.4 Linux

Linux é um sistema operacional de código aberto baseado no kernel Linux, desenvolvido por uma comunidade de programadores em todo o mundo. (Stallman, 2002, p. 53). De acordo com Vignoli e De La Calleja (2019), o Linux tem se destacado no mercado de servidores, sendo amplamente utilizado por empresas que precisam de soluções robustas e escaláveis. O sistema operacional oferece diversas vantagens para o uso em servidores, como a estabilidade, segurança e a possibilidade de personalização. Além disso, o Linux possui uma grande variedade de ferramentas e aplicativos de código aberto, o que permite que empresas possam economizar com licenças de *software*.

2.4.5 Active Directory

O Active Directory é um serviço de diretório desenvolvido pela Microsoft que fornece gerenciamento centralizado de recursos e políticas de segurança em um ambiente de rede. Ele é responsável por autenticar e autorizar usuários e computadores em uma rede, fornecer serviços de diretório para a organização e gerenciar o acesso a recursos de rede, como arquivos, impressoras e aplicativos.

(MICROSOFT) De acordo com o estudo de Villano et al. (2020), o uso do Active Directory é essencial para a segurança da rede em organizações de todos os tamanhos, desde pequenas empresas até grandes corporações.

2.5 Outros conceitos

2.5.1 VPN

VPN (*Virtual Private Network*) é uma tecnologia que permite conectar dispositivos a uma rede privada por meio de uma conexão pública, como a internet. Ele é amplamente utilizado por empresas para fornecer acesso remoto seguro aos seus funcionários e por indivíduos para proteger sua privacidade e segurança on-line. (FILHO, 2018) Segundo Lederer et. al (2016), VPN é uma solução importante para empresas e organizações que necessitam de acesso remoto a recursos da rede interna, como servidores e sistemas, sem comprometer a segurança dos dados.

2.5.2 MVC

O padrão MVC (*Model-View-Controller*) é uma arquitetura de *software* que separa a aplicação em três componentes principais: o modelo, a visão e o controlador. O modelo representa os dados e a lógica de negócios, a visão é responsável pela apresentação dos dados ao usuário e o controlador é responsável por gerenciar as interações do usuário com o modelo e a visão. Esse padrão promove a separação de responsabilidades e permite que os desenvolvedores trabalhem em paralelo em diferentes partes da aplicação. (Pressman, 2016)

O *Model* representa a camada responsável pela manipulação dos dados da aplicação. Segundo Krasner e Pope, "o modelo contém os dados de aplicação e a lógica de manipulação desses dados" (1988, p. 3). Já a *View* é a camada responsável pela apresentação dos dados ao usuário. Ela pode ser representada por um conjunto de páginas HTML, que são renderizadas a partir dos dados obtidos do *Model*. Por fim, o *Controller* é a camada responsável pelo fluxo de controle da

aplicação. Ele recebe as requisições do usuário e decide qual ação tomar, a partir das informações obtidas do *Model* e da *View*.

O padrão MVC tem como objetivo principal separar as preocupações do código em diferentes camadas, reduzindo o acoplamento entre elas e tornando o sistema mais modular. Segundo a definição de Gamma et al., "O padrão Model-View-Controller (MVC) define uma maneira de separar uma aplicação em três partes distintas, cada uma com responsabilidades específicas. O padrão ajuda a controlar a complexidade da aplicação, facilita a manutenção e permite o desenvolvimento de forma independente" (1995, p. 21).

2.5.3 Office 365

O Office 365 é uma suíte de aplicativos de produtividade em nuvem desenvolvida pela Microsoft. Ele inclui aplicativos como Word, Excel, PowerPoint, Outlook, OneNote, SharePoint, Teams e muito mais, além de oferecer recursos de armazenamento em nuvem e colaboração em tempo real. Com o Office 365, os usuários podem acessar e trabalhar em seus documentos e arquivos de qualquer lugar, em qualquer dispositivo e colaborar em tempo real com outras pessoas. (MICROSOFT)

Segundo Li et al. (2019), o Office 365 tem sido amplamente adotado por empresas em todo o mundo devido às suas vantagens em termos de custo e flexibilidade. De acordo com Garg e Kaur (2019), o Office 365 também tem sido elogiado por sua capacidade de facilitar a colaboração e a comunicação entre equipes distribuídas geograficamente. A plataforma inclui ferramentas como o Microsoft Teams, que permite a comunicação em tempo real entre os membros da equipe, bem como recursos de compartilhamento de arquivos e colaboração em documentos.

2.5.4 Criptografia

De acordo com Ferguson e Schneier (2003), criptografia é a ciência de transformar informações em um formato ilegível para indivíduos não autorizados, por meio do uso de algoritmos e chaves criptográficas. A criptografia tem como objetivo

garantir a confidencialidade, integridade e autenticidade de dados em trânsito ou armazenados em dispositivos. Existem vários tipos de algoritmos criptográficos, incluindo simétricos e assimétricos, cada um com suas próprias vantagens e desvantagens.

Segundo Singh e Sankaranarayanan (2019), a criptografia é essencial para a segurança da informação e deve ser utilizada em todos os processos que envolvam informações sensíveis. Já Barua et al. (2020) destacam a importância da utilização de técnicas de criptografia avançadas, como a criptografia de chave pública, para proteger informações sensíveis de ataques de criptoanálise.

Uma das técnicas de criptografia mais utilizadas atualmente é o AES (*Advanced Encryption Standard*), um algoritmo de criptografia simétrica que utiliza a mesma chave para cifrar e decifrar informações. O AES-256-CBC-HMAC-SHA1 é uma variante desse algoritmo, que utiliza um tamanho de chave de 256 bits, um modo de operação CBC (*Cipher Block Chaining*) para cifrar blocos de dados e uma função HMAC-SHA1 (*Hashed Message Authentication Code* com SHA-1) para garantir a autenticidade e integridade das informações criptografadas.

Estudos recentes mostram que o AES-256-CBC-HMAC-SHA1 é uma técnica robusta e segura para proteger informações sensíveis. Um artigo de ZHANG et al. (2021) afirma que essa técnica pode garantir a confidencialidade, autenticidade e integridade das informações criptografadas, mesmo em ambientes com alto risco de ataques cibernéticos. Outro estudo realizado por WANG et al. (2020) também concluiu que o AES-256-CBC-HMAC-SHA1 é um algoritmo de criptografia seguro e eficiente também para proteger informações em dispositivos IoT (*Internet of Things*).

2.5.5 API

“Uma API é um conjunto de rotinas, protocolos e ferramentas que são usados para criar aplicativos de software, permitindo que aplicativos se comuniquem com outros aplicativos ou serviços." (FOWLER, 2017, p. 84) As APIs podem ser usadas para realizar uma variedade de tarefas, como obter dados de um banco de dados, interagir com um serviço da *web*, enviar emails e muito mais. Segundo Chang et al. (2018), as APIs têm se tornado cada vez mais importantes na indústria de software, permitindo a integração de diferentes sistemas, plataformas e dispositivos.

2.5.6 Frontend

O Frontend, por sua vez, é a camada de apresentação, visível ao usuário final. É aí que a interface com o usuário é construída e onde se concentra a maior parte do trabalho de um web designer. (KALINKE, 2014, p. 16) Segundo Abbas et al. (2021), o Frontend tem um papel fundamental na experiência do usuário, já que é através dele que o usuário interage com o aplicativo. Por isso, é importante que a interface seja intuitiva, responsiva e agradável visualmente.

2.5.7 Backend

O backend é a parte do sistema que não é visível para o usuário final, mas que executa as ações necessárias para que a aplicação funcione corretamente. É responsável por processar as solicitações do usuário, acessar o banco de dados, gerar e enviar respostas ao frontend, e outras tarefas relacionadas ao processamento de dados. (SILVA, 2019, p. 126)

2.5.8 HTTP

O HTTP (Hypertext Transfer Protocol) é um protocolo de comunicação utilizado para transferência de dados na *World Wide Web* (WWW). Ele foi criado em 1990 por Tim Berners-Lee e tem sido amplamente utilizado desde então. O HTTP é um protocolo cliente-servidor, onde o cliente faz uma requisição e o servidor envia uma resposta.

Segundo Valverde e Teles (2015), o HTTP é um protocolo *stateless*, ou seja, ele não guarda informações sobre as requisições anteriores. Cada requisição é tratada de forma independente. Isso significa que se uma aplicação precisa manter informações entre requisições, ela precisa utilizar mecanismos adicionais para armazenar o estado da aplicação.

Segundo Fielding et al. (1999), o HTTP não foi projetado com segurança em mente. Ele foi criado para permitir a transferência de dados de forma simples e

eficiente. Como resultado, ele não possui mecanismos de segurança integrados e pode ser vulnerável a ataques.

No entanto, o HTTP tem evoluído ao longo dos anos e hoje existem mecanismos adicionais que podem ser utilizados para torná-lo mais seguro, como o HTTPS (HTTP Secure) e o HTTP/2. O HTTPS é uma extensão do HTTP que utiliza criptografia para proteger os dados durante a transferência. Já o HTTP/2 é uma nova versão do protocolo que foi projetada para ser mais eficiente e mais segura do que o HTTP/1.

2.5.9 Token de Aplicação

Os *tokens* de aplicação são um tipo de *token* utilizado em sistemas de autenticação para fornecer acesso seguro a recursos específicos da aplicação. Ao contrário de *tokens* de acesso, que fornecem acesso a toda a API de um serviço, os tokens de aplicação permitem o acesso somente às partes específicas da aplicação para a qual foram autorizados. Esses *tokens* são frequentemente usados em aplicações baseadas em API para permitir que aplicativos clientes autentiquem-se diretamente com um servidor de API, sem a necessidade de intermediação de um usuário.

Um estudo realizado por Vu et al. (2020) avaliou a segurança de diferentes mecanismos de autenticação e autorização, incluindo OAuth e OpenID Connect, em ambientes de nuvem. O estudo concluiu que o uso desses mecanismos pode melhorar significativamente a segurança de sistemas distribuídos, mas é importante configurá-los corretamente e monitorá-los continuamente para garantir sua eficácia.

Outro estudo realizado por Chaudhry e Zafar (2020) analisou a segurança dos *tokens* JWTs em aplicações baseadas em API, identificando várias vulnerabilidades comuns que podem ser exploradas por atacantes mal-intencionados. Os autores destacaram a importância de implementar medidas de segurança adequadas, como validação de *token* e autenticação de dois fatores, para mitigar essas vulnerabilidades e garantir a integridade e a confidencialidade dos dados do usuário.

Em conclusão, os *tokens* de aplicação são uma técnica importante para garantir a segurança e a privacidade de dados em aplicações baseadas em API. O

uso de mecanismos de autenticação e autorização bem configurados, como OAuth e OpenID Connect, e a implementação adequada de medidas de segurança, como validação de *token* e autenticação de dois fatores, são essenciais para garantir a eficácia e a segurança dos *tokens* de aplicação.

3 METODOLOGIA

Nessa seção, será abordada a metodologia utilizada para alcançar os objetivos geral e específicos do trabalho. Serão descritas as etapas e procedimentos adotados para o desenvolvimento do projeto TrocaSenha, incluindo a definição dos requisitos, o planejamento, o *design* da solução, a implementação do sistema, os testes realizados e a avaliação dos resultados. O objetivo é fornecer uma visão clara e detalhada do processo metodológico utilizado para a realização deste trabalho.

3.1 Segurança

No início do projeto, o solicitante enfatizou a necessidade de um sistema altamente seguro. O sistema anterior, que foi desenvolvido por outra empresa, apresentou uma série de problemas e falhas de segurança críticas resultando em reprovação. Como resultado, a exigência de segurança para o novo sistema foi ainda maior.

Tal exigência é justificada pelo fato de que o TrocaSenha seria o responsável pela troca de senhas do Active Directory, que afeta diversos sistemas, incluindo o NatCorp RH (sistema de folha de pagamento), o email, o Office 365, o Stefanini Atende (abertura de chamados) e a VPN utilizada para conexão com os clientes. Portanto, o mau funcionamento do sistema pode comprometer o acesso a informações confidenciais e causar impactos significativos na empresa.

Por isso, foi necessário realizar um estudo detalhado das técnicas de proteção e criptografia, para garantir a segurança dos dados. Foram aplicadas diversas medidas, incluindo o uso de algoritmos de criptografia avançados e a implementação de protocolos de autenticação de dois fatores. Todo o processo foi cuidadosamente desenvolvido para garantir a integridade e a confidencialidade das informações da empresa.

Devido à política de segurança da empresa, não é permitido divulgar informações sensíveis ou confidenciais que possam comprometer a integridade do sistema ou colocar em risco a segurança dos dados e informações. Dessa forma, não serão expostas informações que possam ser consideradas sigilosas.

3.1.1 Autenticação em dois fatores

Para acessar o sistema de troca de senha, é necessário passar pela autenticação em dois fatores, utilizando email ou SMS. Após o *login*, o sistema envia um código exclusivo que possui duração de 2 minutos. Esse código é enviado para o celular ou email pessoal do colaborador, garantindo que apenas ele possa acessar a plataforma e realizar as tarefas de troca, reset e desbloqueio de senhas.

Caso o colaborador não consiga trocar a senha dentro do tempo estipulado, o token e a sessão são destruídos e o usuário redirecionado para a tela inicial do sistema, como medida de segurança. Isso ocorre para garantir que não haja possibilidade de alguém mal intencionado utilizar um *token* expirado ou tentar utilizar a sessão já encerrada para fins ilícitos.

3.1.2 Criptografia

O sistema possui uma forte criptografia que protege as informações do usuário. Quando o usuário efetua login, o sistema gera um *token* de sessão que contém informações criptografadas daquele acesso, com duração de 10 minutos, sendo gerado um novo *token* a cada iteração com a API.

Por exemplo: quando se faz a solicitação de recebimento de *token* para autenticação de dois fatores, o frontend envia um *token* para o backend, que o descriptografa, valida e depois retorna um novo contendo as informações do usuário e o passo em que ele está neste momento. O backend valida o usuário requisitante do token, captura o IP (endereço eletrônico) de origem e o navegador da requisição, e se forem diferentes do que está criptografado no *token*, a sessão é destruída. Isso impede ataques cibernéticos que tentam copiar ou aproveitar *tokens* de sessões anteriores.

O processo de troca de senha no sistema segue uma sequência de possíveis passos que envolvem a inserção do CPF e data de nascimento do usuário, a autenticação em dois fatores, a escolha entre troca, reset e desbloqueio de senha, a autenticação de dados pessoais, uma segunda autenticação em dois fatores, a inserção de uma nova senha e a confirmação da nova senha. A cada passo, o sistema retorna um novo *token*, que é gerado com base nas informações do usuário,

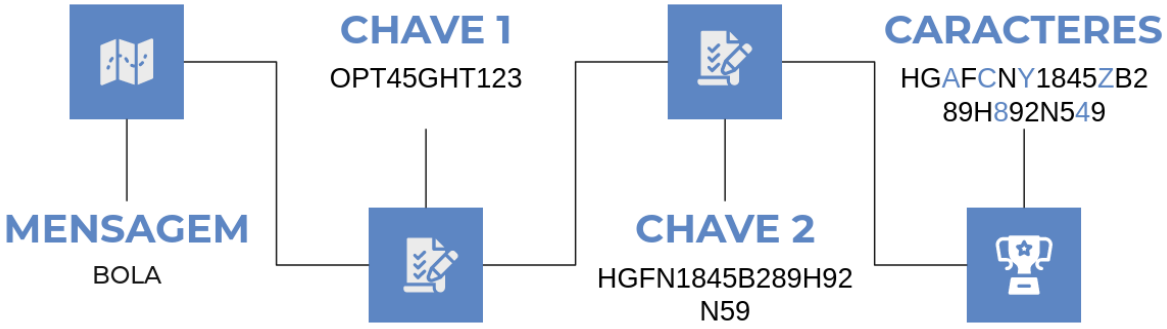
do seu navegador, e no passo atual da solicitação. O número de gerações de *tokens* pode variar de 3 a 7, dependendo da quantidade de passos necessários para concluir a solicitação.

A cada interação realizada pelo frontend com o backend, o sistema é projetado para retornar um *token* de segurança distinto, o qual é gerado por meio do par de chaves de criptografia aleatória. Isso garante maior segurança no processo de autenticação e autorização de acesso ao sistema, uma vez que a utilização de uma chave de criptografia aleatória dificulta a ação de possíveis invasores. Além disso, a constante renovação da chave torna ainda mais difícil a tarefa de decodificar o *token* e acessar o sistema de forma não autorizada. O uso destes *tokens* de aplicação é fundamental para manter a integridade dos dados, prevenir possíveis ataques e evitar o acesso não autorizado a informações confidenciais, tornando o sistema mais robusto e confiável para os usuários.

Caso um usuário tente fazer a troca de senha fora do portal, pulando uma etapa do processo, isso seria considerado uma ação suspeita, já que não há nenhuma API de comunicação externa ao portal. Desta forma, caso haja uma tentativa de trocar a senha sem que o usuário tenha realizado algum dos passos prévios necessários, o sistema invalida o *token* do possível atacante e registra as informações de origem do acesso em um banco de dados, para posterior análise pelo time de segurança da informação. Dessa forma, é possível identificar possíveis tentativas de invasão e tomar medidas para prevenir ataques futuros.

O sistema adota uma abordagem de segurança forte ao usar a criptografia AES-256-CBC-HMAC-SHA1 com quatro chaves distintas. Duas dessas chaves são a pública e a privada da aplicação, enquanto as outras duas são públicas e privadas aleatórias criadas para cada sessão. Essas chaves seguem um padrão pré-determinado para impedir que qualquer chave seja utilizada, e dificulta capturá-las todas de uma só vez, o que reforça ainda mais a segurança do sistema.

Figura 1 - Padrão de criptografia utilizado



Fonte: o autor (2023)

Para criptografar uma mensagem, a aplicação usa a chave fixa, seguida pela chave específica e randômica. Adicionalmente, a aplicação insere aleatoriamente 25 caracteres randômicos no meio da mensagem criptografada gerada. Essa técnica torna a utilização de algoritmos de força bruta para descriptografar a mensagem virtualmente impossível, garantindo ainda mais segurança aos dados sensíveis do sistema.

3.1.3 Outras medidas de segurança

O TrocaSenha é um sistema desenvolvido com foco em segurança, e por isso foi projetado para lidar com diferentes tipos de ataques cibernéticos que podem afetar uma aplicação Node.js. Dentre os principais tipos de ataques que podem ser realizados e que foram tentados pela equipe de testes, é possível citar:

1. XSS (*Cross-Site Scripting*): Cross-site scripting (XSS) é uma vulnerabilidade de segurança comum na *web*. A vulnerabilidade ocorre quando um atacante consegue injetar código malicioso em uma página da web visualizada por outros usuários. Isso pode permitir que o atacante execute ações em nome do usuário, roube informações confidenciais, como senhas e *cookies* de autenticação, ou redirecione o usuário para um site malicioso. (OWASP).
2. SQL Injection: A SQL Injection (Injeção de SQL) é uma vulnerabilidade de segurança em aplicações *web* que permite que um invasor execute comandos maliciosos no banco de dados da aplicação. Isso pode levar a

vazamento de dados, manipulação de informações e até mesmo comprometimento completo do sistema. (OWASP)

3. Injeção de código: A injeção de código ocorre quando o aplicativo não valida adequadamente as entradas fornecidas pelos usuários e permite que um invasor injete código malicioso que é executado pela aplicação. Esses ataques são possíveis em praticamente qualquer aplicação que use entradas de usuário, como campos de pesquisa, formulários de login, filtros e outras entradas de dados. A injeção de código pode levar a uma série de problemas de segurança, como roubo de dados, acesso não autorizado, interrupção de serviços e comprometimento de sistemas. (OWASP). De acordo com Alam et al. (2019), a injeção de código é uma das principais vulnerabilidades de segurança em aplicações web e pode ser explorada para diversos fins, como o roubo de informações confidenciais e o comprometimento do sistema.
4. DDoS (*Distributed Denial of Service*): SANDHU e MEHTA (2017) definem DDoS como um tipo de ataque cibernético que visa tornar um serviço indisponível, sobrecarregando-o com uma quantidade massiva de tráfego de rede. Esse tráfego é gerado por um grande número de dispositivos infectados controlados pelo atacante, que enviam solicitações de conexão simultâneas ao servidor alvo. Como resultado, o servidor fica sobrecarregado e não consegue responder a todas as solicitações, fazendo com que o serviço fique indisponível para os usuários legítimos.

Por meio de todas essas medidas de segurança, o TrocaSenha se torna um sistema altamente resistente e preparado para lidar com possíveis ataques cibernéticos, garantindo a proteção das informações de seus usuários.

A Stefanini Rafael é uma empresa de tecnologia que oferece soluções de segurança cibernética, inteligência artificial, defesa e tecnologias aeroespaciais. Fundada em 2014 como uma *joint venture* entre a brasileira Stefanini e a israelense Rafael Advanced Defense Systems, a empresa combina a expertise em tecnologia e inovação das duas empresas para oferecer soluções inovadoras e integradas que atendem às necessidades de segurança e defesa de governos e empresas em todo o mundo. Com sede em São Paulo, a Stefanini Rafael conta com uma equipe de especialistas altamente qualificados e uma ampla rede de parceiros estratégicos que

garantem a excelência em seus serviços e soluções, e ela foi a empresa responsável pela realização dos testes de segurança no sistema TrocaSenha.

A equipe designada seguiu um protocolo interno e confidencial da própria empresa. Embora não seja possível divulgar detalhes específicos dos testes realizados, sabe-se que eles são rigorosos e abrangem diversas técnicas e cenários para garantir a segurança do sistema. Os testes foram conduzidos com o objetivo de identificar vulnerabilidades e falhas de segurança que possam ser exploradas por atacantes externos ou internos.

3.2 Projeto e Estrutura do Sistema

O TrocaSenha é um sistema criado pela Stefanini Group com o objetivo de gerenciar as senhas dos colaboradores da organização, permitindo que as mesmas sejam trocadas, resetadas e desbloqueadas de forma segura e eficiente.

O desenvolvimento do sistema seguiu um planejamento inicial que previa a criação de três projetos distintos, cada um com funcionalidades específicas que se complementam. A equipe responsável pelo desenvolvimento contou com prazos curtos e uma equipe reduzida, o que levou ao uso de metodologias de desenvolvimento adaptadas para a realidade do projeto.

Foram realizadas reuniões diárias com o requerente, seguindo o fluxo de desenvolvimento evolutivo para organizar as tarefas e garantir o cumprimento dos objetivos propostos. Ao longo do processo, foram levantados e implementados diversos requisitos para garantir a segurança e eficiência do sistema, bem como atender às necessidades específicas da empresa. Após o término do desenvolvimento, foram realizados testes para garantir a segurança e eficiência do projeto em produção.

3.2.1 Estrutura do Projeto

A arquitetura do projeto do TrocaSenha foi planejada para ter três projetos distintos, que são o *web*, API e *server*, cada um com funcionalidades específicas que são cruciais para o funcionamento adequado do sistema.

3.2.1.1 Projeto Web

O projeto Web, desenvolvido em Angular, é um dos três projetos que compõem a arquitetura do sistema. Sua principal responsabilidade é fornecer a interface do usuário final, permitindo que ele interaja com o sistema de maneira intuitiva e amigável. Para isso, o projeto apresenta separações de módulos por páginas da aplicação, com cada módulo contendo os componentes, diretivas e serviços específicos para aquela página.

Além disso, o projeto Web também é responsável por exibir mensagens de feedback ao usuário, coletar informações inseridas por ele e preparar os dados para serem enviados ao projeto API. Para garantir a reutilização de código, o projeto conta com serviços genéricos que podem ser utilizados em toda a aplicação, independentemente da página em que o usuário estiver navegando.

A aplicação possui duas páginas principais: Home e Login. Além disso, ela é composta por três módulos distintos: HTTP, Autenticação e Utilitários. O módulo HTTP é responsável por lidar com as requisições e respostas do protocolo HTTP. O módulo de Autenticação tem como objetivo garantir a segurança e a autenticação dos usuários na aplicação. Já o módulo de Utilitários é responsável por disponibilizar diversas funcionalidades e ferramentas úteis para o desenvolvimento da aplicação.

3.2.1.2 Projeto API

O projeto API é responsável por receber as informações e requisições do frontend. Ele foi desenvolvido em NodeJS e tem como principal objetivo garantir a segurança dos dados do usuário. Para isso, o projeto realiza as validações necessárias, criptografa as informações e envia tudo ao projeto Server, onde as operações de senha serão executadas no Active Directory da empresa.

A arquitetura do projeto API inclui uma variação do padrão de desenvolvimento MVC, que permite separar as responsabilidades em rotas, controladores e modelos. Além disso, o projeto API conta com uma seção dedicada a módulos genéricos, que podem ser utilizados em diversas partes do sistema.

A aplicação é composta por três camadas: Route, Controller e Model. Cada uma dessas camadas tem suas responsabilidades específicas no processo de

desenvolvimento da aplicação. Além disso, a aplicação conta com diversos módulos, tais como Active Directory, chamadas API, criptografia de variáveis e token, comunicação com banco de dados, envio de Email e envio de SMS.

3.2.1.3 Projeto Server

O Server funciona como uma ponte entre o projeto API e o Active Directory da empresa, responsável por executar as operações de senha. Ele recebe as informações criptografadas do projeto API, as descriptografa utilizando os dois pares de chaves e executa a operação necessária no Active Directory.

Além disso, o Server conta com mecanismos de segurança adicionais, como a verificação de autorização do usuário para executar a operação solicitada e a destruição do token de sessão, após a conclusão da operação. Essas medidas garantem que somente usuários autorizados possam executar operações de senha e que informações sensíveis não fiquem armazenadas no servidor após o término da sessão.

3.2.2 Fluxo do Usuário

Para acessar o sistema TrocaSenha, o usuário deve digitar a URL https://trocasenha.stefanini.com no navegador, e para utilizá-lo, o usuário deve seguir os seguintes passos:

1. Inserir o CPF e a data de nascimento;
    a. Caso o usuário possua dois domínios, o sistema irá solicitar que ele escolha em qual deles deseja realizar a troca.
2. Realizar autenticação em dois fatores por meio de SMS ou e-mail;
    a. O sistema exibe informações pessoais incompletas e solicita que o usuário escolha entre receber um código por e-mail ou SMS.
    b. Após o usuário escolher uma opção, o sistema envia o *token* correspondente e então, aguarda por até 2 minutos para que o usuário insira o código recebido. Caso a validação não seja feita neste tempo, o sistema retorna para a tela inicial e invalida aquele *token* de sessão.

3. O usuário pode escolher entre três opções: trocar, resetar ou desbloquear a senha.
    a. Para realizar a opção de "Troca de Senha", o usuário deve inserir sua senha atual e a nova senha desejada nos campos específicos, além de confirmar a nova senha digitando-a novamente.
    b. Ao optar pela opção "Reset de Senha", o usuário deve inserir informações pessoais adicionais para autenticação de dois fatores, como o número do celular e endereço de e-mail cadastrados na empresa. Após preencher esses dados, um *token* será enviado para este SMS ou e-mail, para confirmar a identidade do usuário. Ao contrário da primeira autenticação, em que apenas uma parte da informação é exibida, na segunda autenticação o usuário deve inserir o telefone ou e-mail cadastrado no banco de dados e, em seguida, o sistema realiza novamente a autenticação de dois fatores. Após o recebimento do *token*, o usuário pode inseri-lo na interface do sistema e criar uma nova senha. Essa opção é indicada quando o usuário não lembra a senha, ou a mesma expirou e ele não possui a senha antiga para realizar a troca.
    c. Por fim, a opção "Desbloqueio" é destinada apenas para casos específicos em que a senha esteja bloqueada, sendo necessário apenas selecionar a opção para que a senha seja desbloqueada. Uma segunda autenticação em dois fatores não é necessária para a função de desbloqueio, uma vez que ela não representa um grande risco de segurança ao usuário final caso seja feita indevidamente.
4. Fim do processo

3.2.3 Interface do Sistema

A seguir, estão disponíveis as telas do projeto neste documento.

Figura 2 - Tela de Login

Fonte: o autor (2023)

A Figura 2 contém a tela de entrada do sistema, que requer do usuário insira seu CPF e data de nascimento. A opção "não sou brasileiro" tem a finalidade de remover a formatação do campo CPF. A solicitação foi feita pelo cliente, pois há usuários do sistema em outros países que não utilizam a mesma formatação de documento adotada no Brasil.

Figura 3 - Tela de Autenticação em dois fatores

Fonte: o autor (2023)

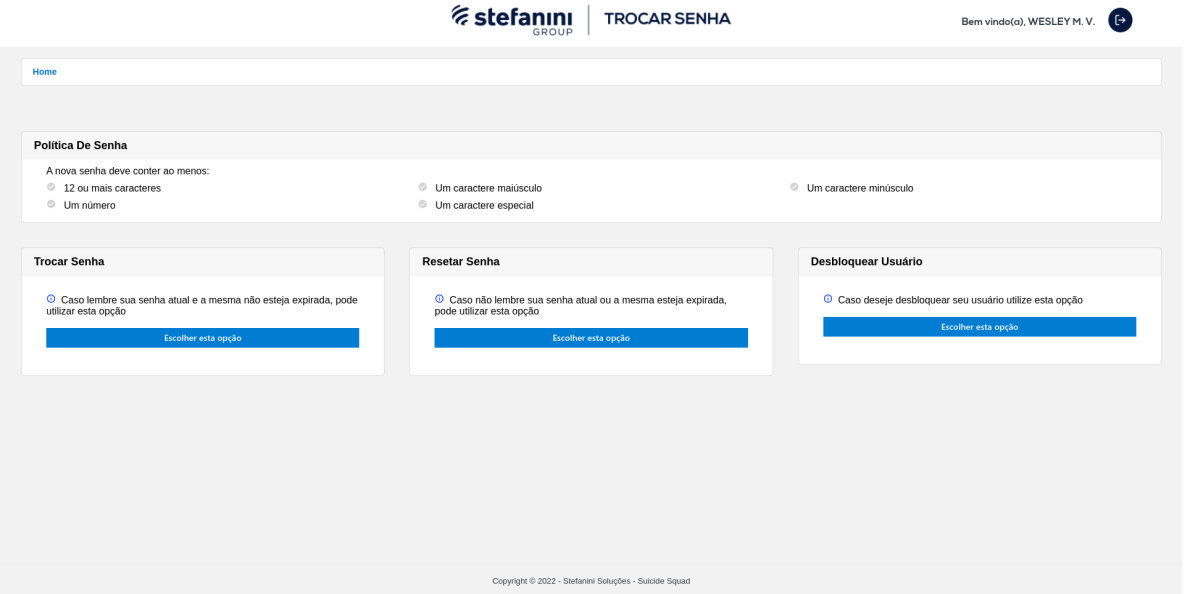
Nesta tela, o usuário pode escolher a forma como deseja receber o token de autenticação de dois fatores, conforme mostra a Figura 3.

Figura 4 - Token de autenticação em dois fatores

Fonte: o autor (2023)

A Figura 4, mostra a tela em que o usuário deve inserir o *token* de autenticação de dois fatores.

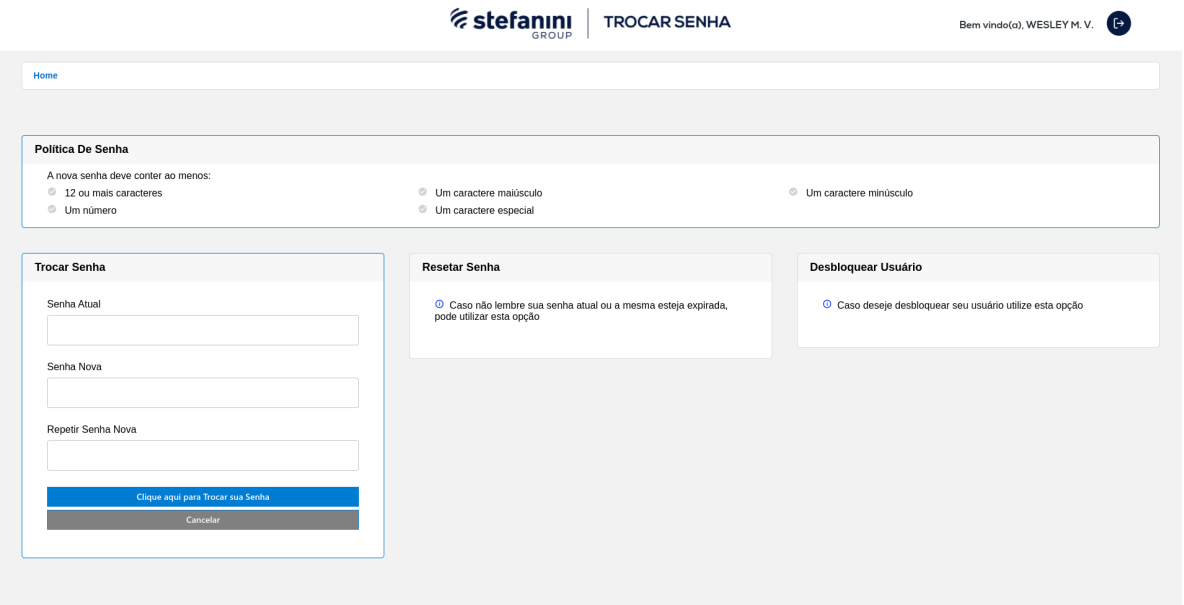
Figura 5 - Tela principal do sistema



Fonte: o autor (2023)

A tela principal do sistema é apresentada na Figura 5, nela o usuário pode optar por trocar, desbloquear ou redefinir sua senha.

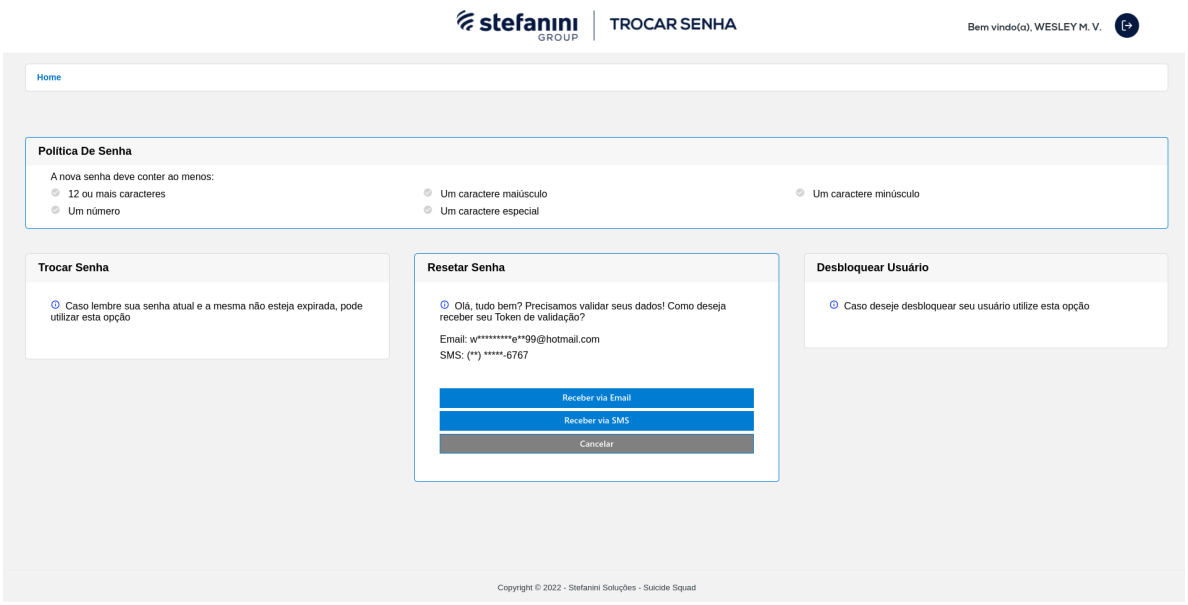
Figura 6 - Opção de troca de senha



Fonte: o autor (2023)

A Figura 6 mostra a tela disponibilizada para o usuário quando ele escolhe a opção de troca de senha.

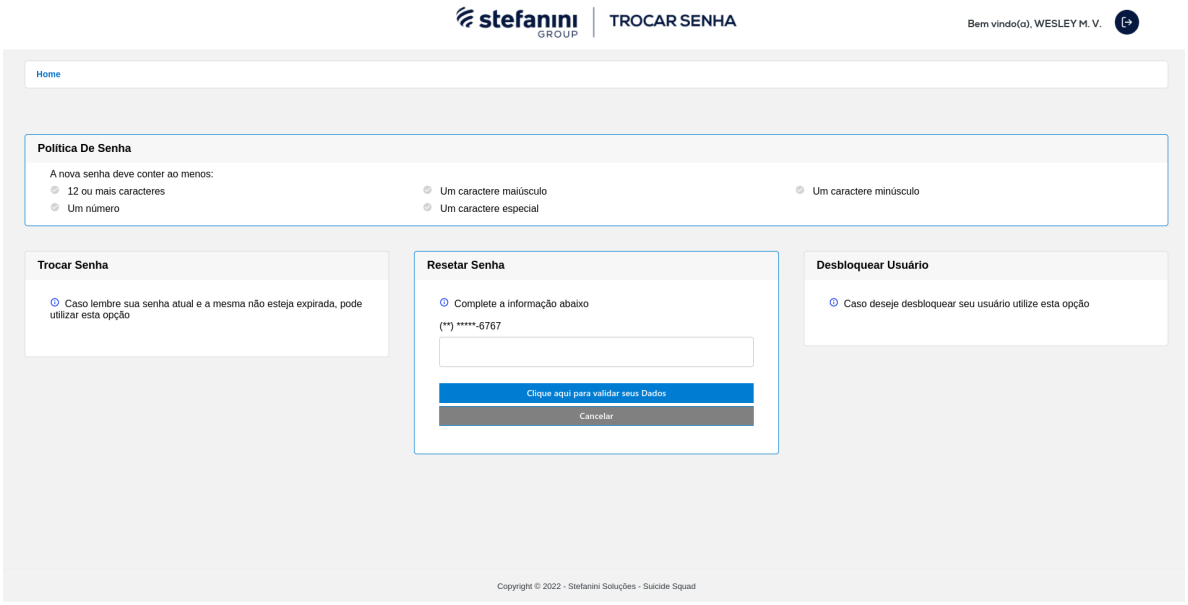
Figura 7 - Opção de reset de senha



Fonte: o autor (2023)

Se o usuário escolher a opção de reset, será necessário realizar a segunda autenticação de dois fatores, como mostrado na Figura 7.

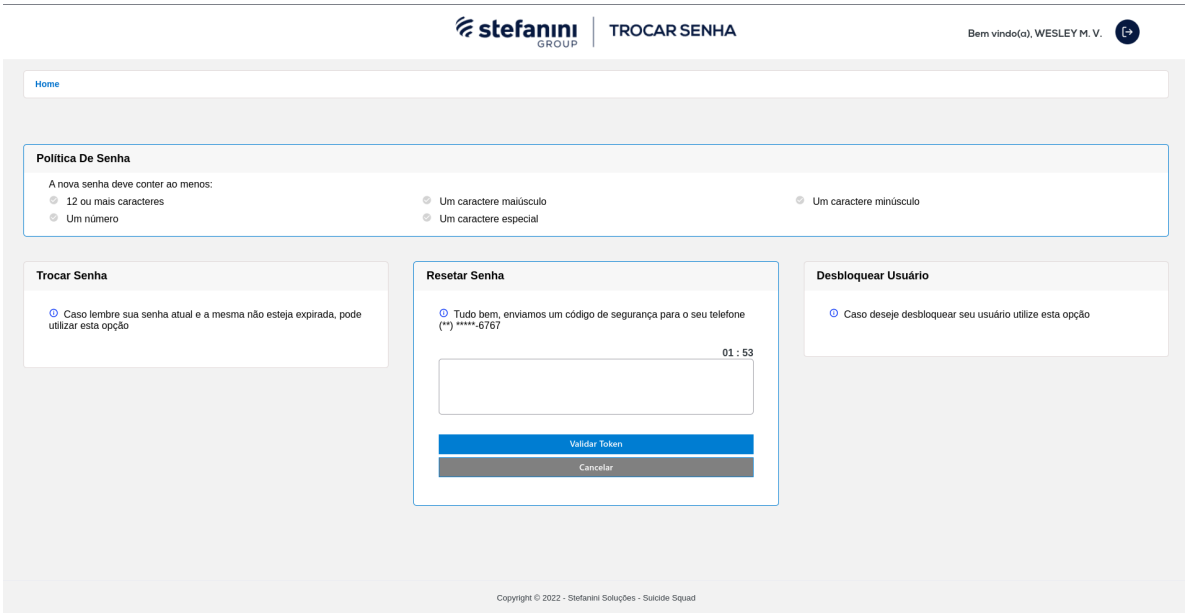
Figura 8 - Autenticação em dois fatores do reset de senha



Fonte: o autor (2023)

Nesta autenticação, é necessário preencher as informações pessoais escolhidas previamente como forma de autenticação extra. A Figura 8 mostra esta verificação. Caso um atacante consiga realizar uma interceptação na primeira autenticação e receber o *token* no seu aparelho celular ou e-mail, ele precisará inserir a informação pessoal no campo para receber o segundo *token*.

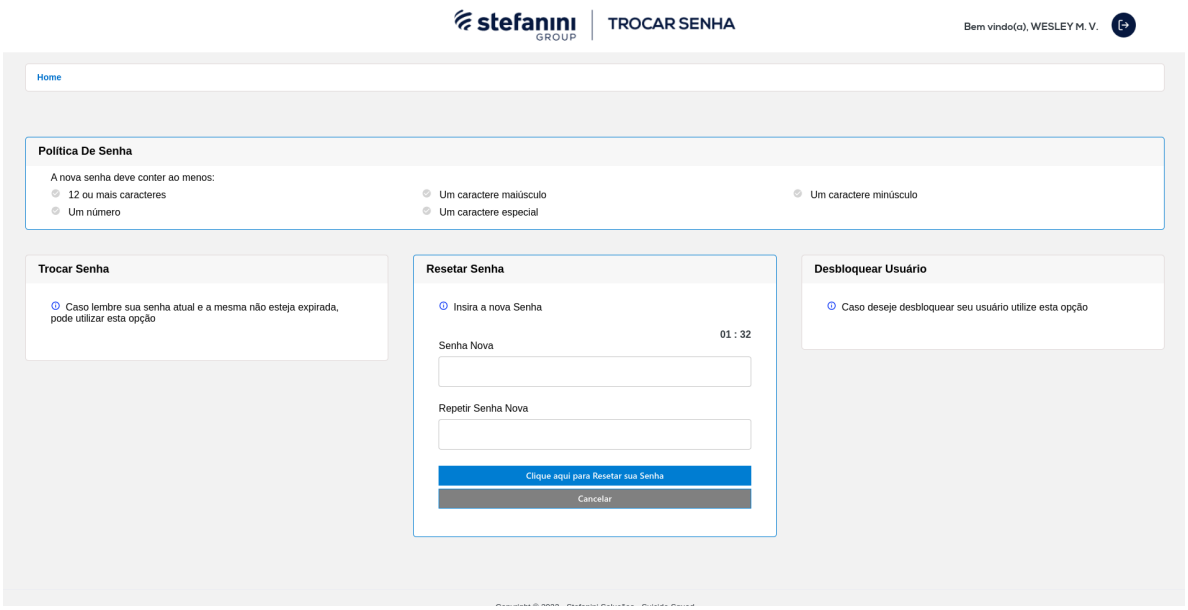
Figura 9 - Token do reset de senha



Fonte: o autor (2023)

Após completar a informação pessoal na autenticação de dois fatores, o usuário recebe um *token* que deve ser inserido no campo correspondente, como mostrado na Figura 9.

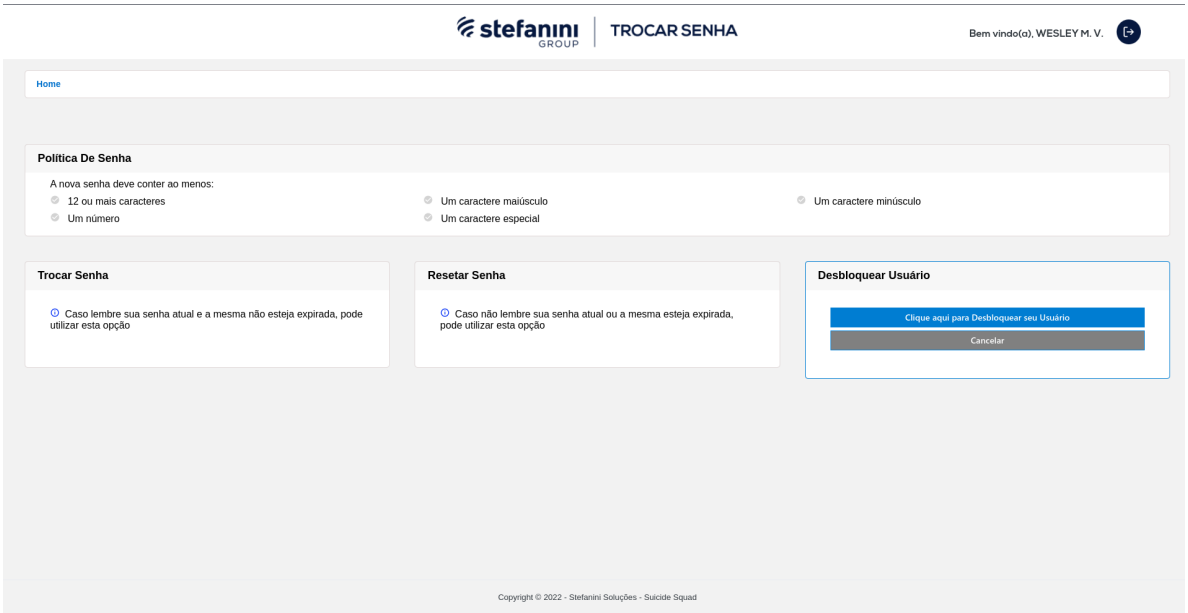
Figura 10 - Inserção de nova senha para reset



Fonte: o autor (2023)

Após passar pela autenticação, o usuário pode inserir sua nova senha, conforme exibido na Figura 10.

Figura 11 - Opção de desbloqueio de senha



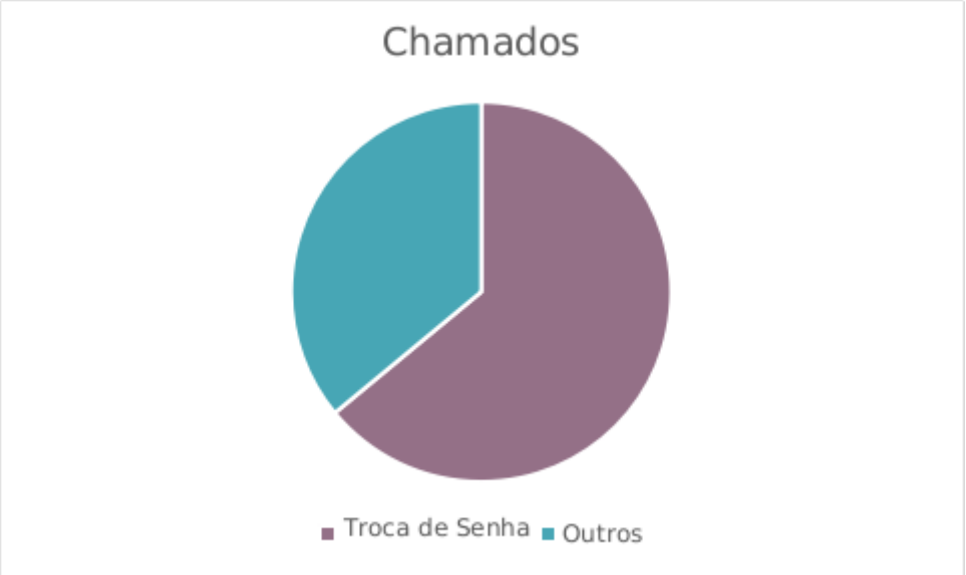
Fonte: o autor (2023)

A Figura 11 mostra a tela disponibilizada para o usuário quando ele escolhe a opção de desbloqueio de senha.

4 RESULTADOS E ANÁLISE

Este projeto teve um impacto muito positivo para a empresa. Antes de sua implementação, a equipe de helpdesk era composta por 100 funcionários em tempo integral, e dela foram analisados 2174 chamados num período de 30 dias.

Figura 12 - Opção de desbloqueio de senha



Fonte: o autor (2023)

A figura 12 mostra os chamados analisados, cerca de 1300 eram relacionados à troca e reset de senha, representando 64% do total de chamados. Dentre esses 1300, 910 chamados foram abertos por usuários que ainda não conheciam o portal de TrocaSenha, o que corresponde a cerca de 70%. Após a implantação do sistema, houve uma redução significativa de 70% nesse número, ou seja, a quantidade de chamados desse tipo caiu para aproximadamente 390. Isso significa uma economia significativa de tempo e recursos para a empresa, além de uma melhoria no atendimento ao usuário, que pode realizar a troca de senha de forma mais rápida e fácil através do portal.

A gerência do Service Desk percebeu que, além da redução de chamados, o portal de troca de senha também permitiu a otimização do tempo de trabalho dos atendentes. Antes, cada chamado de troca ou reset de senha demorava cerca de 15 minutos para ser resolvido, considerando a espera pelo usuário fornecer as

informações necessárias e a execução do procedimento. Com o portal, o processo pode ser concluído em menos de 5 minutos, pois o próprio usuário realiza a troca de senha.

Devido à automatização do TrocaSenha, houve uma redução na quantidade de membros da equipe de atendimento do *Service Desk*. Inicialmente, a equipe foi reduzida de 100 para 80 integrantes, sendo estes 20 realocados para outras atividades. A equipe de suporte optou por não reduzir a equipe ainda mais neste momento, uma vez que o projeto ainda está em fase inicial. À medida que os colaboradores usam e se acostumam com o sistema, a tendência é que a demanda por chamados via telefone diminua ainda mais.

Além disso, a implementação do portal trouxe benefícios em relação à segurança das informações. Antes, quando o *helpdesk* realizava a troca ou reset de senha, os dados do usuário eram expostos a uma terceira pessoa, o que aumentava o risco de vazamento de informações sensíveis. Com o portal, o próprio usuário é o responsável pela troca de senha, garantindo a privacidade e segurança das informações.

Antes da implementação do portal de troca de senha, o processo de troca ou reset de senha para o usuário final da Stefanini era bastante demorado e burocrático. O colaborador precisava ligar para o *helpdesk*, passar várias informações, como CPF e data de nascimento, e esperar na fila de atendimento, o que podia levar cerca de 5 a 10 minutos para ser atendido. Além disso, após a troca de senha, a replicação nos Active Directories da empresa levava em média 20 minutos, o que significava que o usuário precisava aguardar ainda mais tempo para ter acesso aos sistemas da empresa.

No entanto, com a implementação do portal de troca de senha, o colaborador Stefanini ganhou em praticidade e rapidez. Agora, ele pode realizar a troca de senha pelo portal quantas vezes desejar, sem precisar ligar para o helpdesk e passar por todo o processo anterior. Além disso, a replicação da senha nos ADs é feita imediatamente, o que significa que o acesso aos sistemas é liberado rapidamente. De acordo com um estudo de 200 casos, apenas 21 usuários levaram mais de 5 minutos para fazer a troca de senha pelo TrocaSenha, o que comprova a eficácia e rapidez do novo sistema.

A Tabela 1 apresenta os resultados obtidos antes e depois da implementação do projeto, demonstrando, de maneira resumida, algumas das melhorias obtidas com o projeto.

Tabela 1 - Resultados obtidos em porcentagem

	ANTES	DEPOIS	%
Redução de Chamados para Troca de Senha	1300	390	-70%
Redução de Funcionários de HelpDesk	100	80	-20%
Tempo para atendimento (minutos)	10	5	-50%
Tempo para replicação nos AD's (minutos)	20	0	-

Fonte: o autor (2023)

A Stefanini Rafael é uma subsidiária da Stefanini que tem como responsabilidade garantir a segurança dos sistemas desenvolvidos e lançados pela empresa. Todos os projetos passam por uma análise rigorosa da equipe da Stefanini Rafael, antes de serem lançados.

No caso do projeto TrocaSenha, ele foi submetido à equipe de segurança e foi aprovado sem nenhuma ressalva, o que mostra que o sistema foi desenvolvido de forma segura e eficiente. É importante ressaltar que a segurança da informação é uma preocupação constante da Stefanini. Seus projetos são desenvolvidos com o objetivo de garantir a privacidade e a integridade das informações dos seus clientes e colaboradores.

5 CONSIDERAÇÕES

O desenvolvimento do projeto TrocaSenha possibilitou a criação de uma solução eficiente para atender a grande demanda de chamados relacionados a problemas com senhas na empresa Stefanini Group. Com a implementação do sistema, os usuários podem realizar a troca, reset e desbloqueio de suas senhas de forma simples e segura, sem a necessidade de entrar em contato com o *service desk* por telefone. Isso resultou em uma redução significativa na quantidade de chamados recebidos pela equipe de atendimento e em uma melhoria na velocidade e fluidez do processo de gerenciamento de senha.

Durante o processo de desenvolvimento, foram utilizadas diversas tecnologias, incluindo linguagens de programação, *frameworks* e ferramentas de desenvolvimento. A arquitetura do sistema foi baseada em uma variação do padrão de desenvolvimento MVC, permitindo uma separação clara de responsabilidades entre as camadas de *Route, Controller e Model*. Além disso, diversos módulos foram desenvolvidos para garantir a segurança das informações, incluindo criptografia de variáveis e *tokens*, comunicação com BD e envio de email e SMS.

Os objetivos geral e específicos do projeto foram alcançados com sucesso, sendo possível implementar um sistema eficiente que atende às necessidades da empresa Stefanini Group.

Concluímos que o TrocaSenha é uma ferramenta importante para otimizar os processos de gerenciamento de senhas e melhorar a experiência do usuário na empresa Stefanini Group. O desenvolvimento desse projeto nos permitiu adquirir conhecimentos práticos e teóricos sobre tecnologias de desenvolvimento de sistemas e metodologias de trabalho em equipe, além de nos proporcionar uma visão ampla sobre as necessidades de uma empresa no que se refere ao gerenciamento de senhas.

Sobre as perspectivas de futuro, a arquitetura do sistema TrocaSenha está servindo também como modelo para outros 10 projetos desenvolvidos dentro da Stefanini, incluindo projetos novos e em refatorações de sistemas antigos. Isso mostra a eficiência da solução e como ela pode ser aplicada em diferentes

contextos, permitindo que outros sistemas também possam ser beneficiados com a sua arquitetura bem estruturada

REFERÊNCIAS

ABBAS, A. et al. A Comparative Analysis of Front-End Frameworks for Web Development. **IEEE Access**, v. 9, p. 34108-34122, 2021.

AKYILDIZ, İlker; TUGRUL, Özge. **CSS3 Animations and Transitions: An Overview**. International Journal of Advanced Computer Science and Applications, v. 9, n. 3, p. 289-292, 2018.

Alam, M., Parizi, R. M., & Duan, X. (2019). A comprehensive survey of web application security. **Journal of Network and Computer Applications**, 125, 90-114.

ALMEIDA, L. De. **Angular: Uma abordagem prática**. Casa do Código, 2018.

ALNAJJAR, A.; OBEIDAT, R. **Performance Evaluation of Multiple Platforms for Open Source Software Development**. In: Proceedings of the 12th International Conference on Information and Communication Systems, p. 35-40, 2021.

BARUA, S. et al. A comprehensive review of cryptographic techniques for security enhancement in e-commerce. **Journal of Ambient Intelligence and Humanized Computing**, v. 11, n. 6, p. 2413-2429, 2020.

CARNEIRO, F. L. M.; COSTA, L. B. B. Node.js: Uma visão geral sobre a plataforma. In: XXVIII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2018, Maceió. **Anais**[...] Porto Alegre: SBC, 2018.

CHANG, S. et al. **The rise of APIs**. IEEE Software, v. 35, n. 3, p. 20-26, 2018.

CHAUDHRY, S. A., & Zafar, A. (2020). Security vulnerabilities in JSON Web Tokens. **Journal of Ambient Intelligence and Humanized Computing,** 11(11), 5277-5290.

CHEN, Y.; CAI, Y.; ZHU, Z. **Angular: A holistic evaluation of web application framework**. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2019. p. 718-722.

CHELLAPPAN, C.; VALLA, M.; DUBEY, A.; GUPTA, A.; VIJAYAN, J. **Windows 10 Security Assessment**. In: Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence, p. 141-146, 2018.

CORREA, R. M.; CASSIANO, R. Node.js: Um estudo sobre a plataforma e a biblioteca Mongoose. In: CONGRESSO BRASILEIRO DE SISTEMAS COMPUTACIONAIS, 2020, Campina Grande. **Anais**[...] Porto Alegre: SBC, 2020.

FIELDING, R.; GETTYS, J.; MOGUL, J.; FRYSTYK, H.; MASINTER, L.; LEACH, P.; BERNERS-LEE, T. **Hypertext Transfer Protocol** - HTTP/1.1. IETF, 1999.

FILHO, T. A. S. **A segurança da informação em redes de computadores. 1. ed.** Rio de Janeiro: Brasport, 2018.

FERGUSON, Niels; SCHNEIER, Bruce. **Practical Cryptography**. John Wiley & Sons, 2003.

FLORES, M. (2019). **CSS: o que é e como funciona**. Instituto de Gestão e Tecnologia da Informação. Disponível em: http://www.igti.com.br/blog/css-o-que-e-e-como-funciona/. Acesso em 05 de maio de 2023.

FOWLER, Martin. "**APIs: A Shortcut to the Future**". IEEE Software, vol. 34, no. 1, Jan.-Feb. 2017, pp. 84-87. DOI: 10.1109/MS.2017.26.

GAMMA, E. et al. **Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos**. Porto Alegre: Bookman, 1995.

GARG, M.; KAUR, H. A study of Microsoft Office 365 for collaborative learning. **International Journal of Emerging Technologies in Learning,** v. 14, n. 18, p. 99-112, 2019.

GOMES, A. B.; FERREIRA, G. C. Node.js: Uma análise sobre a plataforma e sua arquitetura. In: CONGRESSO BRASILEIRO DE SISTEMAS COMPUTACIONAIS, 2017, Uberlândia. **Anais**[...] Porto Alegre: SBC, 2017.

JANKOVIC, N. **Node.js Application Deployment on Kubernetes**. ProQuest LLC, 2019.

KALINKE, Thiago. **Front-End para Web Designers**. Novatec Editora, 2014.

KETTERING, D. et al. Visual Studio Code: A Source Code Editor for the 21st Century. In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). Raleigh, NC, USA, 2017. p. 245-246.

KHURANA, S.; JAIN, A. AngularJS - A new generation web framework. International **Journal of Advanced Research in Computer Science,** v. 9, n. 2, p. 30-35, mar./abr. 2018.

KRASNER, G. E.; POPE, S. T. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. **Journal of Object-Oriented Programming**, v. 1, n. 3, p. 26-49, 1988.

LEDERER, S.; MÜLLER, M.; KELLERER, W. **VPN as a Service: A Framework for Secure VPNaaS Provisioning**. In: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2016. p. 269-274.

LI, D. et al. A comparison of cloud-based email services: A case study of the adoption of Office 365 in the US and China. **Telematics and Informatics**, v. 38, p. 108-122, 2019.

LOPES, M. L.; LEAL, J. R. C. Desenvolvimento web com Node.js e MongoDB. In: ENCONTRO NACIONAL DE TECNOLOGIA DA INFORMAÇÃO, 2017, Curitiba. **Anais**[...] Curitiba: SENAI, 2017.

MACEDO, P. V. et al. A comparison of JavaScript frameworks and libraries for web application development. **Journal of Software Engineering Research and Development**, v. 7, n. 1, p. 1-21, 2019.

MARTINS, L. M.; JÚNIOR, J. A.; ALMEIDA, M. A. **Análise da eficiência do processamento de requisições HTTP em servidores web utilizando Node.js**. In: CONGRESSO

MATIASICH, Peter. **PM2: The Node.js Production Process Manager**. 2020. Disponível em: https://www.sitepoint.com/pm2-node-js-production-process-manager/. Acesso em: 06 mai. 2023.

MENEZES, N. **HTML: A linguagem de marcação de hipertexto**. Rio de Janeiro: Alta Books, 2019.

MICROSOFT. **Active Directory**. Disponível em: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview. Acesso em: 05 maio 2023.

MICROSOFT. **Office 365**. Disponível em: https://www.microsoft.com/pt-br/microsoft-365/compare-all-microsoft-365-products. Acesso em: 09 maio 2023.

MICROSOFT. **Windows 10**. Redmond: Microsoft, 2015.

MIJALKOVIĆ, S. et al. **Analysis of the AngularJS framework for building web applications**. In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017. p. 769-773.

MIRANDA, E. **Bootstrap**. Disponível em: https://www.devmedia.com.br/bootstrap/. Acesso em: 05 mai. 2023.

NODE.JS. **Node.JS**. Disponível em: https://nodejs.org/. Acesso em: 05 maio 2023.

OLIVEIRA, F. et al. **Comparative Analysis of AngularJS and ReactJS Frameworks**. In: 2018 IEEE 28th International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE). IEEE, 2018. p. 1-6.

OWASP. **Cross-Site Scripting (XSS)**. Disponível em: https://owasp.org/www-community/attacks/xss/. Acesso em: 05 mai. 2023.

OWASP. **Injection Attacks**. Disponível em: https://owasp.org/www-community/attacks/Injection. Acesso em: 05 maio 2023.

OWASP. **SQL Injection**. Disponível em: https://owasp.org/www-community/attacks/SQL_Injection. Acesso em: 05 maio 2023.

Peng, G., Li, S., Li, S., Li, Z., & Li, X. (2019). **Research on Node.js Performance Optimization Technology**. DEStech Transactions on Computer Science and Engineering, (iccie).

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. Porto Alegre: AMGH, 2016.

SANDHU, R., MEHTA, S. **DDoS attacks and mitigation techniques: a review**. In: 2017 International Conference on Computing, Communication and Automation (ICCCA). IEEE, 2017. p. 1195-1200.

SHAH, Krunal. **CSS Position Property**. Medium, 2019. Disponível em: https://medium.com/@krunalshah1992/css-position-property-94b6300a84c9 . Acesso em: 24 abr. 2023.

SHARIFZADEH, M.; HOSSEINI, S. M. **AngularJS vs. ReactJS: A Comparative Study**. In: 2018 7th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS). IEEE, 2018. p. 237-240.

SILVA, João. **JavaScript: uma linguagem de programação para interatividade em páginas web**. São Paulo: Editora Abril, 2019.

Silva, R., & Wirth, F. (2018). **HTML5 e CSS3: Domine a web do futuro**. Novatec Editora.

SINGH, S.; SANKARANARAYANAN, N. Cryptography and Network Security: A Review. **Journal of Computer Science and Technology**, v. 34, n. 6, p. 1257.

SINGH, B.; KUMAR, A.; AGARWAL, N. et al. **A Comparative Study of HMAC using SHA1, SHA256 and SHA512**. In: 2019 5th International Conference on Computing Communication and Automation (ICCCA). Greater Noida, India, 2019. p. 1-6.

Taibi, D., Abdellatif, T., & Orsini, F. (2019). Exploring the Performance of Node.js-based Microservices Architectures. **Procedia Computer Science**, 159, 1307-1316.

TERZIC, Sead. **Mastering CSS: A Comprehensive Guide**. Packt Publishing Ltd, 2019.

VALVERDE, A.; TELES, V. **Desenvolvimento Web com HTML, CSS e JavaScript**. 2. ed. São Paulo: Novatec Editora, 2015.

VIGNOLI, I.; DE LA CALLEJA, J. L. Linux-based server operating systems: a comparative study. **Journal of Systems and Software**, v. 157, p. 1-12, 2019.

Villano, U., Pietro, R. D., & Sgaglione, L. (2020). Cybersecurity in Active Directory domains: A survey. **Computers & Security**, 92, 101740.

Vu, T. A., Vo, Q. B., Huynh, T. V., & Nguyen, T. T. (2020). Security analysis of authentication and authorization protocols in cloud environments. International **Journal of Advanced Computer Science and Applications**, 11(7), 55-61.

WANG, J.; SUN, Y.; ZHANG, Y. et al. **An efficient security scheme for internet of things based on AES-256-CBC-HMAC-SHA1 algorithm**. In: 2020 IEEE 2nd International Conference on Computer Communication and the Internet (ICCCI). Chengdu, China, 2020. p. 193-197.

Zeldman, J. (2010). **Designing with Web Standards (3rd Edition)**. New Riders Publishing.

ZHANG, D.; ZHANG, X.; YU, B. et al. **A New Method of Information Security Based on AES-256-CBC-HMAC-SHA1 Encryption**. In: 2021 2nd International Conference on Advanced Information Technologies and Applications (ICAITA). Fuzhou, China, 2021. p. 301-306.