Darlan Noetzold

Spyware como ferramenta de monitoramento de computadores institucionais e empresariais

Passo Fundo - RS

2023

Darlan Noetzold

Spyware como ferramenta de monitoramento de computadores institucionais e empresariais

Trabalho submetido ao Curso de Bacharelado em Ciência da Computação do Instituto Federal Sul-Rio-Grandense, Câmpus Passo Fundo, como requisito parcial para a aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO

Orientador: Profa. Dra. Anubis Graciela de Moraes Rossetto

Passo Fundo - RS
2023

Darlan Noetzold

Spyware como ferramenta de monitoramento de computadores institucionais e empresariais

Trabalho de Conclusão de Curso aprovado com nota dez, em 2023, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Comissão Examinadora

Profa. Dra. Anubis Graciela de Moraes Rossetto
Orientadora

Prof. Me. Élder Francisco Fontana Bernardi
Examinador

Prof. Dr. João Mário Lopes Brezolin
Examinador

Prof. Dr. Roberto Wiest
Coordenação do Curso

Passo Fundo - RS
2023

Resumo

Com o aumento do uso de dispositivos computacionais, organizações enfrentam desafios significativos, como a exposição de informações confidenciais e a propagação de discursos prejudiciais. Estes desafios podem resultar em sérias consequências financeiras, prejudicar a reputação e afetar o bem-estar mental dos colaboradores. Este estudo apresenta uma abordagem baseada em microsserviços para a supervisão de computadores utilizados por funcionários em empresas. A solução inclui a coleta de dados por meio de métodos de monitoramento, semelhantes ao Spyware, e uma plataforma web para gerenciar alertas. O objetivo principal é identificar vazamentos de informações, comportamentos suspeitos e discursos ofensivos. Resultados da avaliação destacam a eficiência na captura de dados, com detecção rápida de atividades indesejadas. Além disso, a solução emprega modelos de predição para identificar discursos prejudiciais, com uma taxa média de precisão de aproximadamente 87%. A avaliação abordou também aspectos de desempenho, escalabilidade e segurança, demonstrando que a solução é adequada para enfrentar os desafios de segurança de dados e discursos prejudiciais no ambiente corporativo.

Palavras-chave: Spyware, Modelos de Predição, API Gateway, Performance

Abstract

Due to the increasing utilization of computing equipment, institutions and companies are encountering noteworthy challenges, such as the leakage of sensitive data and the propagation of hate speech, both of which have severe repercussions for organizations and their workforce. Tackling these challenges is imperative to avert financial losses, reputational harm, and psychological impacts on those involved. This study introduces a microservices-based solution for monitoring computers used by employees within organizations, encompassing the acquisition of device information through Spyware-like techniques and a web application for alert management. The solution's primary objective is to identify data leaks, suspicious behaviors, and hate speech occurrences. Evaluation results highlight the efficiency of data capture, enabling the rapid detection of undesirable activities. Additionally, the solution employs predictive models to spot hate speech, achieving an average accuracy rate of approximately 87%. Assessments of performance, scalability, and security further illustrate the suitability of the solution in addressing data leakage and hate speech challenges within the corporate setting.

Keywords: Spyware, Model Predict, Performance, API Gateway

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AUC-ROC	A área sob a curva ROC (AUC-ROC)
DNS	Domain Name System
GPT-3	Generative Pre-trained Transformer 3
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JDBC	Java Database Connectivity
JPA	Java Persistence API
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MVC	Model View Controller
NLTK	Natural Language Toolkit
noSQL	Not Only SQL
PBKDF2	Password-Based Key Derivation Function 1 and 2
PIL	Imaging Library
PSutil	Process and System Utilities
REST	Representational State Transfer
ROC	Receiver Operator Characteristic
SKlearn	Scikit-learn
SQL	Structured Query Language

SVM	Support Vector Machine
WSGI	Web Server Gateway Interface
GCE	Google Compute Engine
CI/CD	(Continuous Integration/Continuous Delivery)

Sumário

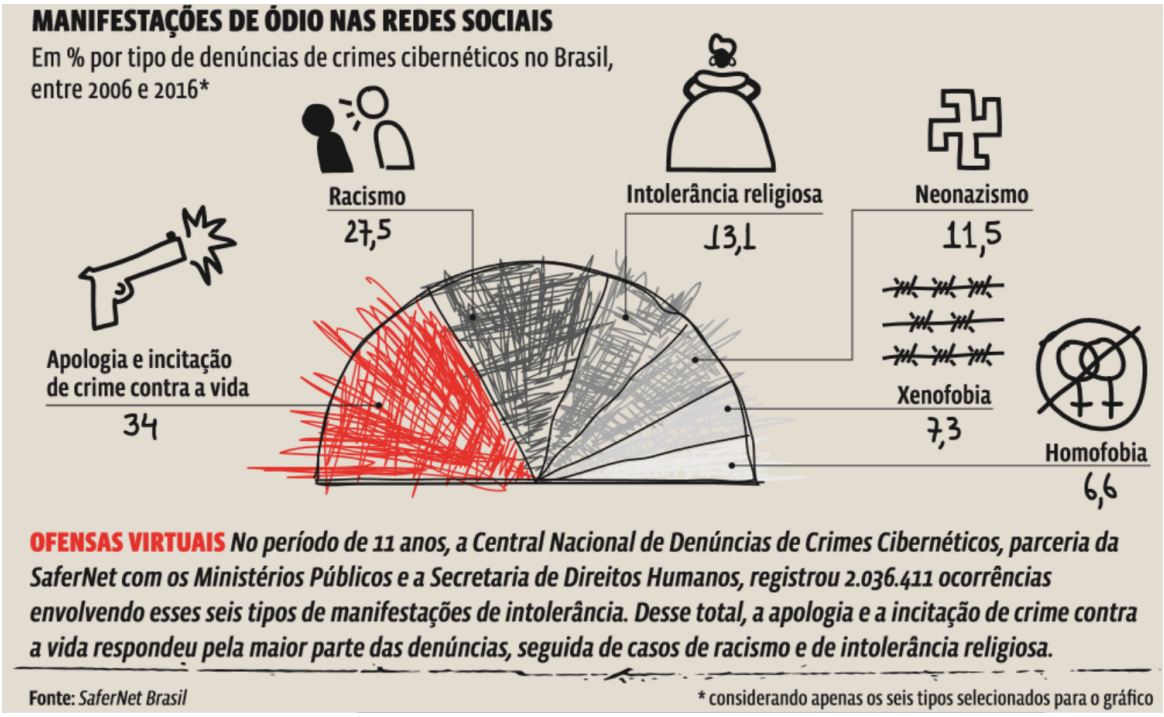
1 Introdução

Atualmente, as organizações enfrentam desafios significativos relacionados ao vazamento de dados e ao discurso de ódio. O discurso de ódio, seja em qual contexto for, é prejudicial para todas as partes envolvidas (SILVEIRA, 2007). A disseminação de discursos de ódio nas plataformas online está aumentando, levando à sua identificação como um problema grave internacionalmente (TONTODIMAMMA et al., 2021).

Um exemplo notável da importância do monitoramento do discurso de ódio nas organizações é o caso da Tesla, que teve que pagar US$ 137 milhões a um ex-funcionário vítima de racismo (NYTIME, 2021). Esse caso destaca a necessidade de medidas proativas para proteger os colaboradores de condutas de ódio e intolerância.

Além dos prejuízos financeiros para as empresas, o discurso de ódio também afeta a saúde psicológica dos colaboradores, especialmente em ambientes de trabalho remoto. A Figura 1 apresenta os principais tipos de discursos de ódio disseminados nas redes sociais.

Figura 1 – Manifestações de discurso de ódio entre 2006 e 2016

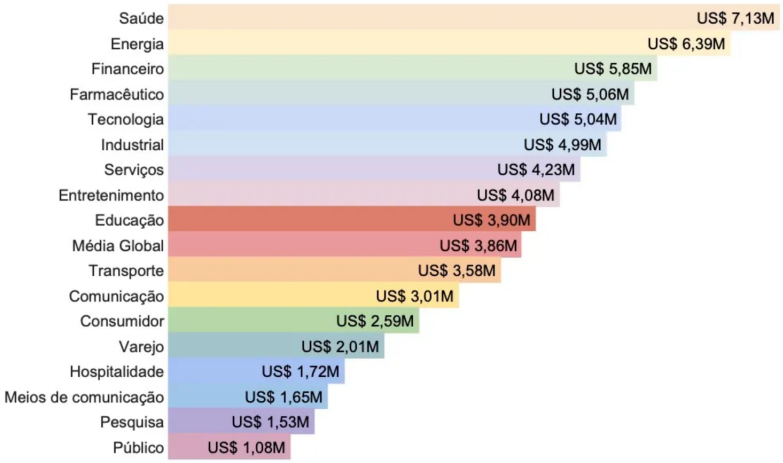


Fonte: (GUIADOESTUDANTE.ABRIL.COM.BR, 2018)

Outra preocupação significativa é o vazamento de dados privados, com empresas enfrentando custos substanciais devido a violações de dados (IBM, 2022). Essas violações podem ocorrer devido a vulnerabilidades exploradas, resultando em danos financeiros e

perda de confiança dos consumidores. A Figura 2 mostra o prejuízo médio sobre vazamentos de dados por setor, destacando a importância de se abordar esse problema.

Figura 2 – Prejuízo médio sobre vazamentos de dados por setor



Fonte: (SEGINFO, 2021)

Este trabalho tem como objetivo projetar e implementar uma solução de monitoramento de computadores empresariais/institucionais, visando mapear o uso dos equipamentos e criar alertas para administradores em uma aplicação front-end. A solução proposta inclui o desenvolvimento de uma aplicação baseada em microserviços com um API Gateway, treinamento de modelos de predição para identificar discursos de ódio, criptografia ponta a ponta para garantir a segurança dos dados e testes abrangentes para validar a solução.

A necessidade de monitorar o comportamento dos usuários tornou-se fundamental para garantir a segurança, o desempenho e a conformidade legal das empresas e instituições. O monitoramento eletrônico de computadores pode auxiliar na prevenção de vazamentos de dados, na identificação de comportamentos inadequados e na proteção da saúde emocional dos colaboradores. Além disso, as evidências coletadas por meio desse monitoramento podem ser úteis em processos judiciais e auditorias. A implementação de uma arquitetura baseada em microsserviços e a ênfase na segurança dos dados tornam a solução relevante e adequada para as demandas atuais das organizações.

A estrutura desta monografia se divide em alguns capítulos, cada qual com sua finalidade específica. O segundo capítulo apresenta as tecnologias e ferramentas utilizadas no desenvolvimento deste trabalho. O terceiro capítulo se concentra na análise dos trabalhos

relacionados, destacando as diferenças entre essas pesquisas e a solução proposta neste estudo. Já o quarto capítulo detalha a arquitetura concebida para a aplicação. No quinto capítulo está explicada em detalhes a implementação da solução proposta. Os procedimentos de teste da aplicação e os resultados obtidos são abordados no sexto capítulo. Por fim, o sétimo capítulo traz as considerações finais e os trabalhos futuros.

2 Tecnologias e ferramentas empregadas no estudo

Este capítulo apresenta as diversas tecnologias e ferramentas que foram empregadas no desenvolvimento da solução, dividindo-as em Spyware, API Gateway, detecção de discurso de ódio, aplicação front-end, Docker e Google Cloud.

2.1 Spyware

Um Spyware, também conhecido como programa espião, é um software especializado na captura de informações através de algum script invasor. Este tipo de programa computacional normalmente é utilizado como Malware para conseguir informações como senhas e dados privados de usuários (BASUMALLICK, 2022). O que este trabalho busca demonstrar é uma releitura desta tecnologia, uma função onde ela pode ser aproveitada de forma positiva e útil para empresas e/ou instituições.

Desta forma, neste trabalho foram usadas algumas tecnologias e técnicas para alcançar o objetivo proposto. Entre as táticas que foram empregadas no projeto, estão: KeyLogger para capturar o teclado, ScreenLogger para capturar a tela, ProcessLogger para armazenar os processos ativos, bloqueamento de DNS e a comunicação com a API usando Python para manter estes dados. A seguir serão abordadas de maneira mais detalhada todas as estratégias que foram utilizadas no Spyware.

2.1.1 KeyLogger

KeyLogger é o termo usado para programas que gravam as teclas pressionadas pelo usuário, podendo assim, enviar para alguma API externa ou para outro usuário. Eles são construídos através de Buffers de teclado em algoritmos orientados a eventos, onde a captura de uma tecla é o evento de ativação. Esta tecnologia normalmente é utilizada para práticas de roubo de informações como senhas, por exemplo, além de outras fraudes em que usuários leigos podem perder a privacidade. Porém, a utilização de KeyLogger's para monitorar computadores já é uma realidade em diversas empresas, que aplicam esta técnica para minimizar a distração dos funcionários, a má conduta corporativa e até mesmo riscos de segurança (BASUMALLICK, 2022).

2.1.2 ScreenLogger

Outro software que é usado de forma má intencionada é o ScreenLogger, o qual tem por objetivo capturar prints da tela do usuário e outras informações privadas, como a posição do cursor mouse, por exemplo. Essas informações podem ser usadas tanto por fraudadores para praticar extorsão, quanto por sites que monitoram o fluxo navegacional do usuário para construir técnicas de retenção do usuário. Estes printscreens são gravados através de Frame Buffers que conseguem armazenar a tela do usuário em bytes, para que possam ser enviados para outra aplicação (TST, 2017).

2.1.3 ProcessLogger

Algoritmos e scripts que capturam os processos ativos de um computador ou ProcessLogger's já existem, principalmente, em empresas e instituições. Nestes meios, são usadas justamente para bloquear processos desconhecidos ou em alguma BlackList conhecida, para que diminua a chance de Malwares causarem danos de software ou hardware na máquina em questão. Este tipo de aplicação voltou a ser mais usada depois do surgimento de cryptojacking's, que são programas que infectam computadores para usar os recursos dos mesmos para a mineração maliciosa de criptomoedas (CARRIER, 2022).

Esta captura de processos é feita com API's que se comunicam com o Sistema Operacional e o gerenciamento de memória do mesmo, podendo obter os processos ativos em foreground e background.

2.1.4 Sniffer

Sniffer (farejador, em tradução livre) é um software que permite ao usuário "farejar" ou monitorar o tráfego de internet em tempo real, capturando todos os dados que entram e saem de um computador. O sniffing pode ser utilizado tanto para propósitos maliciosos como também para o gerenciamento de rede, monitoramento e diagnóstico de ambientes computacionais (N-ABLE, 2021). No atual trabalho, esta ferramenta foi utilizada para analisar se o usuário está acessando sites maliciosos.

2.1.5 Scanner

Um Scanner é um software que é usado para localizar vulnerabilidades em sistemas no geral. Eles podem varrer a rede e sites em busca de milhares de riscos de segurança diferentes, produzindo uma lista priorizada daqueles que devem ser corrigidos, descrevendo as vulnerabilidades e fornecendo etapas sobre como corrigi-las. Alguns podem até mesmo automatizar o processo de correção (BALBIX, 2021).

No Spyware que foi desenvolvido neste trabalho, o Scanner é um PortScanner, que analisa as portas utilizadas na interface de rede e gera um alerta caso exista uma conexão com alguma aplicação maliciosa ou com vulnerabilidades.

2.1.6 Comunicação com API

Para que um Spyware tenha eficácia, os dados capturados precisam ser enviados para algum lugar, é aí que entra a comunicação com API's. Neste tipo de aplicação a comunicação precisa ser o mais simples possível, pois precisa ser rápida, eficiente e sem muito custo de memória. Desta forma, em Python normalmente é utilizada a biblioteca nativa da linguagem, a "requests", onde são usados requests Hypertext Transfer Protocol (HTTP) do tipo POST para o envio das informações obtidas (PYPI.ORG, 2022).

2.2 API Gateway

O API Gateway é uma ferramenta utilizada para gerenciamento de tráfego entre aplicações de backend e microsserviços, dividindo as responsabilidades da aplicação inteira, facilitando seu uso. O API Gateway garante escalabilidade e alta disponibilidade dos serviços, é responsável por encaminhar a solicitação ao serviço apropriado e enviar uma resposta ao solicitante. Ademais, mantém uma conexão segura entre os dados e APIs gerenciando o tráfego e solicitações das APIs, incluindo balanceamento de carga, isto dentro e fora da aplicação (ENGINEERING, 2022). Complementando, na Figura 3 exemplifica-se que o API Gateway trabalha como uma porta ou túnel que permite a comunicação entre diferentes aplicações.

Figura 3 – API Gateway e a comunicação com diferentes aplicações

Do Autor, 2022

No presente trabalho, o API Gateway foi construído com Spring Boot Framework e PostgreSQL, sendo criada uma imagem com todas as configurações, incluindo um sistema de Message Broker com o RabbitMQ, cache com um banco Not Only SQL (noSQL) e tendo login e autenticação com token JWT gerenciado com o Spring Security.

2.2.1 Spring Boot e Spring Framework

O Spring Boot nasceu do Spring Framework, o qual foi desenvolvido para a plataforma Java baseado nos padrões de projetos, inversão de controle e injeção de dependência. Ele busca agilidade e simplificação no desenvolvimento e configuração de aplicações Java, porém as configurações seguiam grandes e complexas demais.

Foi assim que nasceu o Spring Boot, fornecendo a maioria dos componentes necessários em aplicações de maneira pré-configurada, possibilitando uma aplicação rodando em produção rapidamente, com o esforço mínimo de configuração e implantação. Pode-se entender o Spring Boot como um template pré-configurado para desenvolvimento e execução de aplicações baseadas no Spring (SPRINGBOOT, 2022).

Para realizar todo esse processo o Spring Boot utiliza um conceito chamado convenção sobre configuração. O que significa que o Spring decide para a melhor forma de se fazer algo. É o que chama-se de ferramenta opinativa, ela toma as decisões no lugar do desenvolvedor baseado-se em convenções, aplicando configurações padrões e facilitando o trabalho. No entanto, ela não é inflexível e ainda permite uma configuração diferente da padrão caso o usuário assim deseje (SPRINGBOOT, 2022).

O Spring Boot é o módulo inicial do Framework, o qual vai chamar os outros módulos, que serão apresentados a seguir.

2.2.1.1 Spring Boot Starters

Os starters são dependências que reúnem outras dependências com propósitos semelhantes ou iguais. Deste modo, apenas uma configuração é realizada no projeto. Por exemplo, o spring-boot-starter-amqp é uma dependência que permite a construção de soluções de mensageria baseadas em RabbitMQ (GEEKSFORGEEKS, 2021). Assim, ao realizar a configuração no gerenciador de dependência se define somente o starter, como é possível ver na Figura 4, que é um Print Screen do arquivo de dependências .

Figura 4 – Arquivo de Dependências

```
<!-- https://mvnrepository.com/artifact/org.springframework.amqp/spring-amqp -->
<dependency>
    <groupId>org.springframework.amqp</groupId>
    <artifactId>spring-amqp</artifactId>
    <version>3.0.0</version>
</dependency>
```

Fonte: Do autor, 2023

E internamente ele encapsula as dependências necessárias para utilização das features. Alguns exemplos de starters disponíveis:

- Spring Boot Starter Web: auxilia na construção de aplicações web trazendo já disponíveis para uso Spring Model View Controller (MVC) , Representational State Transfer (REST) e o Tomcat como servidor;
- Spring Boot Starter Test: contém a maioria das dependências necessárias para realizar testes da aplicação: Junit, AssertJ, Hamcrest, Mockito, entre outros;
- Spring Boot Starter Data Java Persistence API (JPA): facilita a construção da camada de persistência, ajudando na abstração do banco de dados e provendo uma série de facilidades para criação de repositories, escrita de queries, entre outros.

2.2.1.2 Spring Boot Actuator

O Spring Boot Actuator é o módulo responsável pela monitoria do projeto e gerenciar as aplicações implantadas (SPRINGBOOT, 2022). Dentre os recursos disponibilizados tem-se:

- Métricas: captura e disponibiliza os dados da aplicação, como por exemplo, espaço em disco, memória e tempo de resposta;
- Logging: facilita o acesso ao arquivo de log por meio de um endpoint específico;
- HeathlChecks: disponibiliza endpoints de health checks, para visualizar a saúde do projeto em produção;
- Informações da Aplicação: permite a disponibilização de informações da aplicação, por exemplo, versão e informações do git.

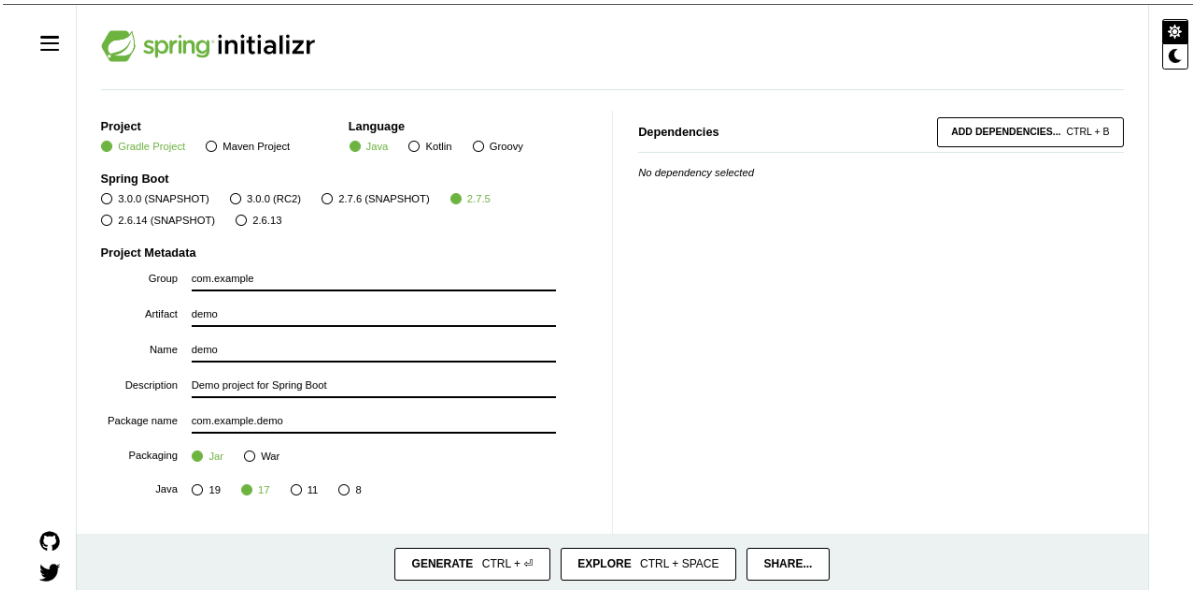
2.2.1.3 Spring Boot Autoconfigure

O Autoconfigure é responsável por ler o conteúdo contido no classpath do projeto e realizar as configurações necessárias para que a aplicação funcione de forma opinativa. Além de ser ele quem gerencia todo o processo de configuração da aplicação (SPRINGBOOT, 2022).

2.2.1.4 Spring Boot Devtools

Spring Boot Devtools é um conjunto de funcionalidades que ajuda o trabalho de qualquer desenvolvedor. Como, por exemplo, restart automático da aplicação quando ocorre alguma mudança no código (SPRINGBOOT, 2022).

Figura 5 – Spring Initializr



Fonte: (SPRING, 2021)

2.2.1.5 Spring Initializr

Para facilitar a criação de aplicações, a Spring disponibilizou o Spring Initializr. Ele é uma UI que permite a criação de projetos Sprint Boot de forma facilitada (SPRINGBOOT, 2022). A Figura 5 apresenta um print screen da ferramenta Spring Initializr.

Através desta interface é possível definir o nome do projeto, pacotes, dependências (starters do spring e outros projetos) e linguagem (Java, Groovy ou Kotlin). Uma vez definido é só clicar no botão "Generate" e o projeto será criado, gerando um zip.

2.2.1.6 Spring Security

O Spring Security possui um sistema de autenticação e autorização de alto nível e customizável para aplicações Java. Ainda que o foco do Spring Security seja o sistema de autenticação e autorização, ele possui outras funcionalidades que aumentam a segurança das aplicações Java (SECURITY, 2022):

- Sistema de autenticação;
- Sistema de autorização;
- Proteção contra ataques como session fixation, clickjacking e-cross site request forgery;
- Integração com a Servlet API;
- Integração opcional com o Spring Web MVC.

2.2.1.6.1 Autenticação

Basicamente é possível afirmar que a autenticação é o login, tratando-se da etapa de verificação de um determinado usuário, onde é analisado se ele possui credenciais válidas para acessar a aplicação. Especificamente o sistema de autenticação do Spring Security pode ser configurado para que utilize diferentes estratégias de autenticação, pois o trabalha com o conceito de providers de autenticação.

Os providers de autenticação são as estruturas responsáveis por efetivar as informações sobre os usuários que acessam a aplicação. Dessa maneira, é possível ter uma série de providers diferentes para utilizar nas aplicações. O próprio Spring Security já expõe alguns providers para serem prontamente utilizados, como os providers baseados na JPA, providers baseados no Java Database Connectivity (JDBC) e até mesmo providers baseados no padrão Lightweight Directory Access Protocol (LDAP), permitindo que as aplicações executem o fluxo de autenticação e autorização através de servidores Active Directory, por exemplo (SECURITY, 2022).

2.2.1.6.2 Autorização

Já a autorização é um processo que acontece depois da autenticação. É o momento onde a aplicação verifica se o usuário atualmente autenticado tem permissão de acesso a um determinado recurso. O sistema de autorização do Spring Security também é bastante flexível, pois permite definir com facilidade quais são os possíveis tipos de usuários da aplicação. Isto porque, o sistema relaciona cada usuário com o seu determinado tipo e quais rotas de aplicação cada tipo de usuário terá acesso (SECURITY, 2022).

2.2.1.6.3 Armazenamento de Senha

Além dos sistemas de autenticação, autorização e proteção contra diferentes tipos de vulnerabilidades de aplicações web, o Spring Security também disponibiliza algoritmos de criptografias, que evitam que a aplicação guarde as senhas de seus usuários em texto puro no banco de dados. Os algoritmos disponibilizados no Spring Security são o bcrypt, Password-Based Key Derivation Function 1 and 2 (PBKDF2), scrypt e argon2. Sendo o bcrypt o mais utilizado pela comunidade (SECURITY, 2022).

2.2.1.6.4 JSON Web Tokens (JWT)

Este sistema de Tokens é usado para manter a segurança de APIs REST integradas com autorização e autenticação. Para isso, o client primeiramente envia algumas credenciais para se autenticar. O servidor, em seguida, verifica essas credenciais e se elas forem válidas, o servidor gera um JWT e o retorna (SECURITY, 2022).

Depois desses passos, o client precisa fornecer esse token no cabeçalho Authorization da solicitação, no formulário “Bearer TOKEN” (Token do portador). O back-end verificará a validade desse token e autorizará ou rejeitará as solicitações. O token também pode armazenar funções de usuário e autorizar as solicitações com base na autoridade fornecida (SECURITY, 2022).

2.2.2 RabbitMQ

A comunicação entre sistemas distribuídos é um desafio complexo que envolve diversas questões, como a garantia de entrega, escalabilidade e tolerância a falhas. Uma solução comumente adotada é a utilização de plataformas de mensageria, que permitem que sistemas distribuídos se comuniquem por meio de troca de mensagens assíncronas.

RabbitMQ é uma plataforma de mensageria amplamente utilizada em sistemas distribuídos e foi desenvolvida em Erlang. Ela implementa o padrão Advanced Message Queuing Protocol (AMQP), que define a estrutura das mensagens e o comportamento dos clientes e servidores.

Uma das principais vantagens do RabbitMQ é sua arquitetura modular, que permite que diferentes componentes possam ser configurados e utilizados de acordo com as necessidades do sistema. Além disso, ele oferece recursos como filas de mensagens, troca de mensagens baseada em roteamento e mecanismos de confirmação de entrega.

Diversos estudos têm avaliado a performance e escalabilidade do RabbitMQ em diferentes cenários de uso. Um estudo realizado por Sengupta et al. (2017) comparou o desempenho do RabbitMQ com outras plataformas de mensageria em um ambiente de nuvem pública. Os resultados indicaram que o RabbitMQ apresentou uma boa performance em cenários com alto volume de mensagens.

Outro estudo, realizado por Sarwar et al. (SARWAR et al., 2019), avaliou a escalabilidade do RabbitMQ em um ambiente de nuvem privada. Os resultados mostraram que o RabbitMQ foi capaz de lidar com um alto volume de mensagens, apresentando uma baixa latência e um alto throughput.

Além disso, a utilização do RabbitMQ tem sido avaliada em diferentes domínios de aplicação. Um estudo realizado por Ahmed et al. (AHMED et al., 2020) avaliou a utilização do RabbitMQ em sistemas de Internet das Coisas (IoT) e mostrou que ele pode ser utilizado para lidar com o fluxo de dados em tempo real gerado por dispositivos IoT.

Por fim, o RabbitMQ é uma plataforma de mensageria flexível e escalável que pode ser utilizada em diversos cenários de sistemas distribuídos. Seus recursos e arquitetura modular permitem que ele seja configurado de acordo com as necessidades do sistema. Além disso, diversos estudos têm mostrado sua eficiência em diferentes cenários e domínios de aplicação.

2.2.3 Cache com Redis

O Redis é um sistema de armazenamento em cache de código aberto, rápido e de baixa latência, que pode ser usado para melhorar o desempenho de aplicativos da web (BEGNUM; VINTERHAGEN, 2020). Ele armazena dados em memória, o que o torna mais rápido do que os sistemas de armazenamento em disco. Além disso, o Redis é capaz de armazenar dados estruturados, como strings, listas, conjuntos e hashes (GAO et al., 2018).

O Redis usa uma estrutura de dados conhecida como chave-valor, onde os dados são armazenados em pares de chave-valor (ARORA; JAIN; KUMAR, 2019). O uso de uma estrutura de dados simples permite que o Redis seja altamente escalável e eficiente. Além disso, o Redis suporta operações atômicas em seus dados, como incremento, decremento, adição e remoção, o que o torna útil para implementar contadores, filas e outros padrões de uso comuns em aplicativos da web (NGUYEN; TRAN, 2021).

Outro recurso importante do Redis é a capacidade de armazenar dados em cache de forma persistente em disco, permitindo que os dados sejam preservados entre reinicializações do servidor (BEGNUM; VINTERHAGEN, 2020). Isso é útil para aplicativos que precisam armazenar dados de sessão de usuário, dados de configuração e outras informações que precisam ser mantidas entre sessões.

Ele tem sido amplamente utilizado em uma variedade de aplicativos da web, incluindo jogos online, redes sociais, sistemas de comércio eletrônico e aplicativos de análise de dados (GAO et al., 2018). É uma escolha popular para muitos desenvolvedores devido à sua escalabilidade, eficiência e facilidade de uso.

Desta forma, o Redis é um sistema de armazenamento em cache de alta performance, escalável e flexível, que pode melhorar significativamente o desempenho de aplicativos da web. Sua implementação de chave-valor e sua capacidade de armazenar dados em cache persistentes o tornam uma escolha popular para desenvolvedores que precisam de um sistema de armazenamento em cache robusto e escalável (ARORA; JAIN; KUMAR, 2019).

2.2.4 Flyway

O FlywayDB é uma ferramenta de código aberto que oferece suporte à automação de migrações de banco de dados. Ele permite que desenvolvedores gerenciem de forma eficiente e controlada as mudanças no esquema do banco de dados durante o ciclo de vida de um projeto (SMITH; JOHNSON, 2022b).

De acordo com (SMITH; JOHNSON, 2022b), o FlywayDB proporciona uma abordagem baseada em scripts para a evolução do esquema do banco de dados. Os scripts de migração são organizados e versionados, o que permite a aplicação sequencial e automática de alterações no banco de dados à medida que o software evolui. Essa abordagem facilita

a colaboração em equipes de desenvolvimento e ajuda a garantir a consistência entre os ambientes de desenvolvimento, teste e produção.

(DOE; WILLIAMS, 2019) destacam que a aplicação do FlywayDB em projetos reais traz benefícios significativos em termos de controle de versão e rastreabilidade das alterações no banco de dados. Através do versionamento dos scripts de migração, é possível saber exatamente quais alterações foram aplicadas em cada versão do sistema. Isso é especialmente valioso para a equipe de operações e para a realização de rollbacks, caso seja necessário reverter para versões anteriores do banco de dados.

Uma das principais vantagens do FlywayDB é a sua integração fácil e flexível com diversas tecnologias e frameworks, incluindo projetos Java, Spring, .NET, entre outros. Além disso, o FlywayDB suporta múltiplos bancos de dados, o que permite a sua utilização em diferentes sistemas que utilizam bases de dados distintas.

No contexto de projetos reais, o FlywayDB tem sido amplamente adotado em empresas de diversos setores da indústria, desde startups até grandes corporações. Sua utilização oferece um meio seguro e organizado para evoluir o esquema do banco de dados sem afetar negativamente a integridade e a confiabilidade dos dados (JURIŠIĆ,).

Em suma, o FlywayDB é uma ferramenta essencial para a gestão eficiente e automatizada de migrações de banco de dados em projetos de desenvolvimento de software. Sua abordagem baseada em scripts versionados e sua ampla compatibilidade tornam-no uma escolha popular entre os desenvolvedores, oferecendo uma solução robusta e confiável para a evolução do esquema do banco de dados em projetos de todos os tamanhos e complexidades.

2.3 Detecção de discurso de ódio

Para o desenvolvimento da API que analisa se uma frase é discurso de ódio ou não foram usadas tecnologias adjacentes ao Python. Como o Micro Framework Flask para o mapeamento de handlers e endpoints, o scikit-learn que é uma biblioteca de aprendizado de máquina, onde foram usados três modelos de predição (para cada idioma, português, inglês e espanhol), e uma biblioteca de normalização e vetorização de dados chamada nltk. Além dos modelos que foram treinados, também foi usada uma integração com o GPT-3 como mais um mecanismo de verificação se existe discurso de ódio na sentença capturada. Além disso, foram usadas outras tecnologias já abordadas neste capítulo. Com isso, para uma melhor compreensão das técnicas que foram utilizadas no presente trabalho, esta seção aborda de forma mais específica cada uma das tecnologias apresentadas acima.

2.3.1 Scikit-learn

O scikit-learn é uma biblioteca do Python desenvolvida especificamente para aplicação prática de machine learning. Esta biblioteca contém ferramentas para análise preditiva de dados, tem código aberto e foi construída sobre os pacotes NumPy, SciPy e Matplotlib. A biblioteca é organizada em muitos módulos, em que cada um é desenvolvido para uma finalidade específica. Nestes módulos é possível encontrar técnicas e ferramentas para várias aplicações diferentes (SCIKITLEARN, 2022). Analisando estas diferentes aplicações, pode-se entender a organização da biblioteca:

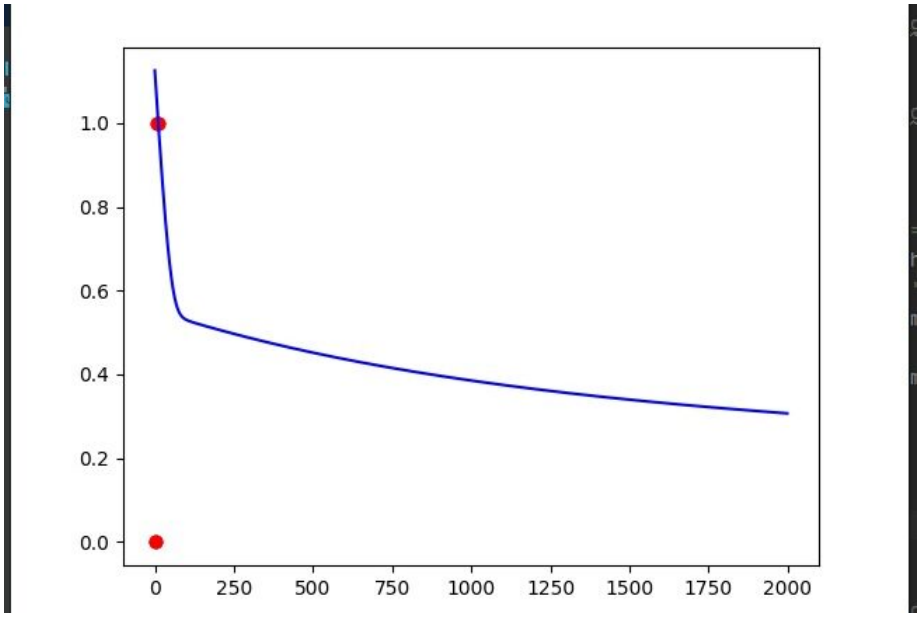
- Pré-processamento: é, provavelmente, a etapa mais trabalhosa no desenvolvimento de um modelo de machine learning. O NumPy e o Pandas são largamente utilizados nesta etapa, mas também existem funções para esta finalidade no Scikit-learn (SKlearn), pensadas especialmente para tratamento de dados que alimentarão algoritmos de machine learning;
- Classificação: etapa que desenvolve categorizações de elementos através de análises de características específicas. É possível identificar, por exemplo, se uma pessoa possui ou não determinada doença, ou ainda qual doença uma pessoa pode ter dentre várias possíveis, dentre muitas outras possibilidades;
- Regressão: desenvolvimento de modelos que podem atribuir um valor contínuo a um elemento. Prever a altura de uma pessoa, quantidade de vendas de um produto e preço de um imóvel, por exemplo;
- Clusterização: criação de modelos para detecção automática de grupos com características similares em seus integrantes. Por exemplo, é possível identificar grupos de risco de determinada doença ou verificar padrões entre moradores de uma cidade;
- Redução de dimensionalidade: reduzir o número de variáveis em um problema. Com esta redução é capaz diminuir consideravelmente a quantidade de cálculos necessários em um modelo, aumentando a eficiência, com uma perda mínima de assertividade;
- Ajuste de parâmetros: comparar, validar e escolher parâmetros e modelos, de maneira automatizada. Comparar diferentes parâmetros no ajuste de um modelo, encontrando assim a melhor configuração para a aplicação em questão.

Através de todas estas tecnologias e recursos é possível criar modelos de predição usados para aplicações reais, entre elas podemos citar a Regressão Logística, Multinomial Naive Bayes e Support Vector Machine (SVM). Estes modelos serão tratados a seguir.

2.3.1.1 Regressão Logística

A regressão logística é um modelo estatístico usado para determinar a probabilidade de um evento acontecer. Ele mostra a relação entre os recursos e, em seguida, calcula a probabilidade de um determinado resultado (FáVERO, 2022). A Figura 6 apresenta um exemplo de uma Regressão Logística, onde o eixo Y decai conforme o eixo X aumenta, representando a mudança de um parâmetro correlacionado com a mudança de outro elemento.

Figura 6 – Regressão Logística



Fonte: Do autor, 2023

A regressão logística é usada em ciências médicas, exatas e sociais. A regressão logística é utilizada em áreas como as seguintes:

- Em finanças, pode detectar os grupos de risco para a disponibilização de créditos;
- No domínio dos seguros, permite encontrar clientela que sejam sensíveis a determinada política securitária em relação a um dado risco;
- Em medicina, permite determinar as características de grupo de indivíduos doentes em relação a indivíduos sãos.

A regressão logística olha para parâmetros distribuídos binomialmente da seguinte forma, $Yi\ B(pi, ni)$, for $i = 1, \ldots, m$, onde os números de ensaios de Bernoulli ni são conhecidos e as probabilidades de êxito pi são desconhecidas. Um exemplo desta distribuição é a porcentagem de pacientes (pi) que se curam com um medicamento depois de ni serem tratadas com o mesmo (FáVERO, 2022).

O modelo é então obtido na base de que valor de i e o conjunto de variáveis independentes possa informar sobre a probabilidade final. Estas variáveis explicativas podem-se ver como um vetor Xi k-dimensional e o modelo toma então a forma (FáVERO, 2022):

$$pi = E(\frac{Yi}{Ni}|Xi)$$

2.3.1.2 Multinomial Naive Bayes

O algoritmo "Naive Bayes" é um classificador probabilístico baseado no "Teorema de Bayes", o qual foi criado por Thomas Bayes (1701 - 1761) para tentar provar a existência de Deus. Atualmente, o algoritmo se tornou popular para categorizar textos baseado na frequência das palavras usadas. A principal característica do algoritmo, e também o motivo de receber "naive" (ingênuo) no nome, é que ele desconsidera completamente a correlação entre as variáveis (features) (RATZ, 2022).

O teorema de Bayes é um corolário da lei da probabilidade total, expresso matematicamente na forma da seguinte equação:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Em que A e B são eventos e P(B) é diferente de zero (RATZ, 2022).

2.3.1.3 Support Vector Machine

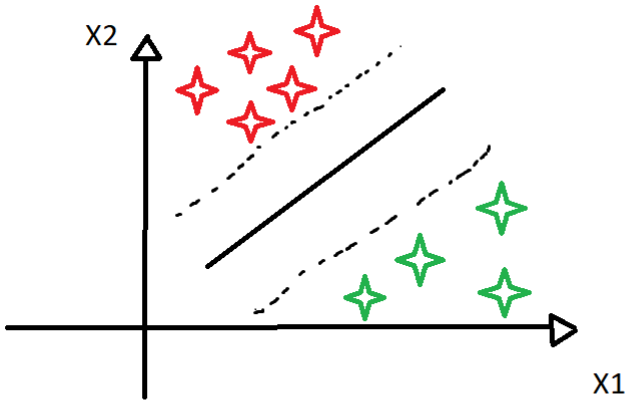
Um Support Vector Machine (SVM) tem como papel encontrar uma linha (hiperplano) que separa os dados de duas classes distintas. Essa linha busca aumentar ao máximo a distância entre os pontos mais próximos em relação a cada uma das classes (BENNETT; CAMPBELL, 2022), como no gráfico da Figura 7.

A distância entre o hiperplano e o primeiro ponto de cada classe costuma ser chamada de margem. A SVM tenta primeiro classificar as classes corretamente e depois em função dessa restrição definir a distância entre as margens (BENNETT; CAMPBELL, 2022).

2.3.2 Vetorização e Normalização de dados

Para que um modelo de predição consiga ter resultados satisfatórios é preciso que os dados de entrada sejam separados de uma forma padronizada. É disso que se trata a normalização dos dados, a qual tem por objetivo retirar palavras que não acrescentam para a predição e separar as frases em conjuntos de palavras, que são adicionadas em vetores, e é aí que ocorre a vetorização.

Figura 7 – SVM



Fonte: Do autor, 2023

No Python existem algumas bibliotecas que auxiliam neste processo, a principal é o Natural Language Toolkit (NLTK), o qual dispõe de um conjunto de bibliotecas com ferramentas capazes de facilitar o trabalho da análise dos dados e normalização (NLTK, 2022). Entre as funções mais usadas para estes procedimentos pode-se citar:

- Tokenize: essa função serve para quebrar texto por palavras, criando desse modo um array com todas as palavras contidas dentro do texto (NLTK, 2022);
- Stemming: essa função serve para diminuir a palavra até a sua raiz/base, pois assim é possível tratar as palavras originais e suas respectivas derivações de uma mesma maneira. As palavras Correr e Corrida quando submetidas à função de Stemming serão diminuídas até a base Corr, por exemplo. Desse modo, o modelo é capaz de reconhecer derivações ainda não conhecidas de palavras bases que ele já aprendeu (NLTK, 2022);
- RemoveStopWords: essa função serve para retirarmos de dentro do array algumas palavras que não são interessantes para contabilizarmos uma pontuação na hora de classificar o texto, então mantem-se somente as palavras principais (NLTK, 2022).

2.3.3 GPT-3 e Integração com a Openai

O Generative Pre-trained Transformer 3 (GPT-3) é um modelo de linguagem natural pré-treinado que usa a arquitetura Transformer e é treinado em uma grande quantidade de dados de texto. É o modelo mais recente e poderoso da série GPT, lançado pela OpenAI em 2020. O modelo é capaz de gerar texto de alta qualidade

em uma variedade de tarefas, como tradução de idiomas, resumo de texto, redação de artigos, e até mesmo programação de computadores (BROWN et al., 2020). Além disso, GPT-3 tem uma API (Application Programming Interface) pública que pode ser integrada com outras linguagens de programação, incluindo Python (OPENAI, 2021a).

A arquitetura Transformer consiste em camadas de autoatendimento e camadas de saída. As camadas de autoatendimento permitem que o modelo faça previsões condicionais de uma palavra, dada a sequência de palavras anteriores. As camadas de saída mapeiam a representação de um texto para uma saída específica, como a classificação de texto ou a geração de texto (VASWANI et al., 2017).

Para integrar GPT-3 com Python é necessário primeiro se inscrever no programa de acesso antecipado da API da OpenAI. Depois de obter as credenciais de autenticação, podemos usar a biblioteca OpenAI para Python para se comunicar com a API e enviar solicitações de texto para o modelo. A biblioteca fornece várias funções para acessar os recursos do modelo, como geração de texto e classificação de texto. A integração com Python permite que os desenvolvedores criem aplicativos que se beneficiam da geração de texto avançada fornecida pelo GPT-3 (OPENAI, 2021b).

GPT-3 tem várias aplicações em áreas como processamento de linguagem natural, análise de dados e aprendizado de máquina. O modelo pode ser usado para gerar texto de alta qualidade em uma variedade de tarefas, como redação de artigos, tradução de idiomas e resumo de texto. Além disso, GPT-3 pode ser usado para gerar código em várias linguagens de programação, como Python e HTML. O modelo também pode ser usado para classificar texto com alta precisão em várias categorias, como spam, conteúdo ofensivo e sentimento (BROWN et al., 2020).

2.3.4 Micro-framework Flask

Um Micro-framework é um Framework modularizado e com estruturas mais simples, sendo uma versão mais minimalista e, por isso, muito utilizado na construção de microsserviços (FLASK, 2022).

Assim sendo, o Flask segue essas regras muito bem, pois ele é expansível e simples, tendo como principais características:

- Projetos menores: os projetos escritos em Flask tendem a ser menores e mais leves se comparados a frameworks maiores;
- Rapidez no desenvolvimento: Com o Flask, basta desenvolver o necessário para um projeto, sem a necessidade de realizar muitas configurações;

- Aplicações robustas: o Flask também permite a criação de aplicações robustas, já que é totalmente personalizável, sendo possível a criação de uma arquitetura mais definida;
- Simplicidade: ele possui apenas o necessário para o desenvolvimento de uma aplicação, por isso um projeto escrito com Flask é mais simples se comparado aos frameworks maiores, já que a quantidade de arquivos é muito menor e sua arquitetura é muito mais simples.

Além disso, o Flask é baseado nos projetos Werkzeug e Jinja2. Werkzeug é um kit de ferramentas para aplicativos Web Server Gateway Interface (WSGI). Werkzeug pode realizar objetos de software para funções de solicitação, resposta e utilidade. Já o Jinja é um mecanismo de template semelhante ao framework web Django, ele lida com modelos em uma sandbox (FLASK, 2022).

2.4 Aplicação front-end com Spring Boot Framework

Para a aplicação front-end foram usadas tecnologias em torno do Spring Boot, como o Thymeleaf para a interface web, o Feign Client para consumo da API e outras tecnologias que já foram apresentadas neste capítulo, como o Spring Security para o login e cadastro de usuários e o Docker para implantação da aplicação.

Sendo assim, este subsessão apresenta as ferramentas de front-end usadas, as quais não foram comentadas anteriormente, no caso o Feign Client e o Thymeleaf.

2.4.1 Feign Client

Esta tecnologia foi criada para facilitar e reduzir a complexidade de aplicações consumidoras de algum WebService. Feign é um projeto que foi inspirado em Retrofit, JAXRS-2.0 e WebSocket. Ele utiliza de anotações plugáveis que podem ser anotações Feign, JAX-RS, entre outras (OPENFEIGN, 2022).

Atualmente o Spring incorporou o Feign nas suas dependências, simplificando ainda mais a configuração e integração com os outros módulos. Dessa forma, pode-se aproveitar as anotações que são utilizadas para criar WebServices, como GetMapping, PathVariable, etc. Assim, esta tecnologia pode ser incorporada, principalmente, em projeto front-end para o consumo de API's ou diretamente de um back-end externo, tornando a aplicação muito mais simples e segura (OPENFEIGN, 2022).

2.4.2 Thymeleaf

O Thymeleaf é um template engine de código aberto e licenciada pela Apache License 2.0 para projetos Java, que tem por objetivo facilitar a criação de páginas HyperText Markup Language (HTML). Assim, um template engine permite que linguagens de programação possam ser incorporadas em páginas HTML, permitindo o uso de estruturas de condição, estruturas de repetição, herança e diversos outros recursos presentes apenas nas linguagens de programação em páginas HTML (THYMELEAF, 2022).

Analogamente o Thymeleaf permite que desenvolvedores incorporem código Java em páginas HTML e também utilizem as principais características da linguagem em seus templates. Dentre diversas características, pode-se citar as principais:

- Permite o uso de estruturas de condição e repetição em páginas HTML;
- Com o Thymeleaf é possível utilizar herança de layouts, garantindo uma estrutura com um maior reaproveitamento de código;
- Permite exibir o conteúdo de variáveis Java em páginas HTML;
- Sistema de fragmentos de templates, dentre outros.

2.4.3 Hash SHA-256

O hash SHA-256 é um algoritmo criptográfico de função hash pertencente à família das funções hash seguras (SHA - Secure Hash Algorithm). Desenvolvido pelo National Institute of Standards and Technology (NIST) dos Estados Unidos, o SHA-256 é uma das variantes da família SHA-2 (Secure Hash Algorithm 2) e é amplamente utilizado em várias aplicações de segurança e criptografia (GUERON; JOHNSON; WALKER, 2011).

Uma função hash é um algoritmo que transforma dados de entrada de tamanho variável em um valor de saída fixo, normalmente representado por uma sequência hexadecimal de caracteres. A principal característica de uma função hash é ser determinística, o que significa que sempre que a mesma entrada for fornecida, a saída será idêntica. Além disso, a função hash deve ser eficiente, ou seja, o cálculo do hash deve ser rápido e não exigir muitos recursos computacionais (GUERON; JOHNSON; WALKER, 2011).

O SHA-256 é projetado para ser resistente a várias técnicas de ataque, incluindo colisões, pré-imagem e resistência a segunda pré-imagem. Isso significa que ele garante a unicidade dos hashes para dados diferentes, torna computacionalmente inviável encontrar a entrada original que gerou um hash específico e dificulta encontrar dois conjuntos de dados diferentes que produzam o mesmo hash (GILBERT; HANDSCHUH, 2003).

O algoritmo SHA-256 opera em blocos de 512 bits (64 bytes) e produz um hash de 256 bits (32 bytes). Quando o comprimento dos dados não é um múltiplo de 512 bits, o

algoritmo realiza o preenchimento para garantir que todos os dados sejam processados em blocos de tamanho uniforme. O processo de cálculo do hash SHA-256 envolve várias etapas, incluindo inicialização do estado interno, preparação dos blocos, pre-processamento, rounds (rodadas) e finalização (GUERON; JOHNSON; WALKER, 2011).

O SHA-256 é amplamente utilizado em várias aplicações de segurança, tais como criptomoedas, assinaturas digitais, armazenamento seguro de senhas, verificação de integridade de arquivos e autenticação e troca de chaves. Ele desempenha um papel fundamental em diversas aplicações de segurança, protegendo informações e garantindo a integridade dos dados em ambientes digitais (GILBERT; HANDSCHUH, 2003).

2.5 Docker

O Docker é uma tecnologia de virtualização de aplicativos que tem se popularizado rapidamente em todo o mundo (MOHAMED et al., 2021). Ele permite que os desenvolvedores empacotem e distribuam seus aplicativos de maneira eficiente, garantindo que esses aplicativos sejam executados de maneira consistente em diferentes ambientes (MOHAMED et al., 2021). O Docker é um software de virtualização que permite a criação e o gerenciamento de contêineres, que são unidades de software que contêm todo o necessário para executar um aplicativo, incluindo código, bibliotecas, ferramentas e configurações (MOHAMED et al., 2021).

Diferentemente da virtualização tradicional, que requer a criação de uma máquina virtual completa, o Docker permite que os contêineres compartilhem o sistema operacional do host, o que significa que eles são muito mais leves e rápidos do que as máquinas virtuais (MERKEL, 2014). Além disso, o Docker possui um ecossistema de ferramentas que facilita a criação, o gerenciamento e a implantação de contêineres (BHAT; RAGHAVENDRA, 2020).

O Docker Hub é um repositório de contêineres que permite que os desenvolvedores compartilhem e distribuam seus aplicativos para outros usuários do Docker (MERKEL, 2014). O Docker Compose é uma ferramenta que permite a definição e o gerenciamento de aplicativos com vários contêineres (MOHAMED et al., 2021). Já o Docker Swarm é uma plataforma de orquestração de contêineres que permite que os usuários gerenciem clusters de contêineres em larga escala (MOHAMED et al., 2021).

A importância do Docker para a virtualização de aplicativos pode ser vista através de diversos estudos e pesquisas científicas. Por exemplo, um estudo realizado por Li et al. (2019) comparou o desempenho do Docker com a virtualização tradicional em um ambiente de computação em nuvem. Os resultados mostraram que o Docker teve um desempenho melhor do que a virtualização tradicional em termos de tempo de inicialização, desempenho de rede e uso de recursos Li et al. (2019).

Outro estudo interessante foi realizado por Bhat e Raghavendra (2020), que investigaram o impacto do Docker na escalabilidade e na disponibilidade de aplicativos em nuvem. Os resultados mostraram que o Docker é capaz de melhorar significativamente a escalabilidade e a disponibilidade dos aplicativos em nuvem, em comparação com outras tecnologias de virtualização.

Por fim, um estudo mais recente realizado por Liu et al. (2021) avaliou o desempenho e a escalabilidade do Docker em um ambiente de microserviços. Os resultados mostraram que o Docker é uma tecnologia promissora para a virtualização de aplicativos em ambientes de microserviços, com baixa sobrecarga de desempenho e alta escalabilidade.

2.6 Google Cloud

O Google Cloud Platform (GCP) é uma plataforma de serviços em nuvem oferecida pelo Google, que abrange uma variedade de serviços de infraestrutura, armazenamento, processamento, análise e desenvolvimento de aplicativos. Como uma das principais provedoras de serviços em nuvem do mercado, o Google Cloud é amplamente utilizado por empresas e desenvolvedores para hospedar, gerenciar e expandir suas operações de TI de maneira escalável e eficiente (BISONG; BISONG, 2019).

Algumas áreas-chave abordadas pelo Google Cloud incluem:

- Computação em Nuvem: O Google Compute Engine fornece máquinas virtuais altamente escaláveis e personalizáveis, permitindo a execução de cargas de trabalho com desempenho previsível. O GKE (Google Kubernetes Engine) permite orquestrar e gerenciar containers usando o Kubernetes (GEEWAX, 2018);
- Armazenamento e Banco de Dados: O Google Cloud oferece serviços como Google Cloud Storage para armazenamento de objetos, Cloud SQL para bancos de dados SQL gerenciados, Cloud Spanner para bancos de dados globais e escaláveis, e Bigtable para armazenamento de dados em larga escala (GEEWAX, 2018);
- Análise de Dados e Machine Learning: O BigQuery permite análise de dados em escala com SQL e machine learning integrado. O AI Platform oferece recursos para treinar, implantar e gerenciar modelos de machine learning (GEEWAX, 2018);
- Rede e Segurança: O Google Cloud fornece ferramentas para criar redes virtuais, balanceamento de carga e firewall. Além disso, o Identity and Access Management (IAM) permite controlar o acesso aos recursos (GEEWAX, 2018);
- Desenvolvimento de Aplicativos: O App Engine permite criar e implantar aplicativos escaláveis sem gerenciar a infraestrutura subjacente. O Cloud Functions oferece

execução de código em resposta a eventos sem a necessidade de provisionar servidores (GEEWAX, 2018);

- IoT e Processamento de Streaming: O Google Cloud IoT Core ajuda a conectar, gerenciar e processar dispositivos IoT. O Cloud Pub/Sub oferece serviços de mensagens e processamento de streaming em tempo real (GEEWAX, 2018);
- Ferramentas de DevOps: O Google Cloud inclui uma variedade de ferramentas para automação, como o Cloud Deployment Manager para criação de infraestrutura como código, e o Cloud Monitoring para monitoramento e solução de problemas (GEEWAX, 2018);
- Segurança e Conformidade: O Google Cloud oferece recursos de segurança em várias camadas, incluindo criptografia, autenticação, proteção contra ameaças e conformidade regulatória (GEEWAX, 2018);
- Serviços de API e Ecossistema: O Google Cloud também inclui uma ampla variedade de APIs e SDKs que permitem integração e desenvolvimento de aplicativos em várias linguagens de programação (GEEWAX, 2018);

É importante destacar que o Google Cloud é uma plataforma em constante evolução, com novos serviços e recursos sendo adicionados regularmente. A adoção do Google Cloud pode trazer benefícios significativos para organizações que buscam escalabilidade, flexibilidade e inovação em suas operações de TI, permitindo que elas aproveitem os recursos avançados de nuvem para atingir seus objetivos comerciais (BISONG; BISONG, 2019).

3 Trabalhos Relacionados

Empresas, desenvolvedores e pesquisadores tem explorado o potencial de ferramentas para monitoramento no ambiente de trabalho. No entanto não foram identificadas soluções que integrem o monitoramento para identificação de discurso de ódio em conjunto com técnicas de verificação de vulnerabilidades.

Entre as soluções comerciais, estão o Kickidler (KICKIDLER, 2023) que visa automatizar a função de gerenciamento dos funcionários da empresa, oferecendo um conjunto de ferramentas para monitorar computadores dos funcionários e detectar violações durante a hora de trabalho. Os principais recursos são: visualização das telas dos funcionários em tempo real com o modo multiusuário; relatórios de tempo de trabalho de funcionários; e Keylogger para salvar o histórico das teclas apertadas. Porém as funcionalidades da versão não paga são limitadas.

O ActivTrak (ACTIVTRAK, 2023), também uma solução comercial, monitora a atividade, analisa o desempenho e o comportamento dos funcionários nos computadores de trabalho, bem como detecta ameaças internas. Entre suas funcionalidades estão: visualização das telas dos funcionários em tempo real, sem o modo multiusuário; relatórios de contabilização de tempo; bloqueador de sites; Keylogger; e captura de tela dos computadores dos funcionários.

O FSense (FSENSE, 2023) monitora o uso de computadores e registra o acesso a sites e aplicações não aprovados pela empresa, tendo como foco o aumento na produtividade da equipe. A ferramenta proporciona um dashboard com gráficos e relatórios; resumo das atividades monitoradas, ociosidade e máquina bloqueada; e captura de tela a cada 30 segundos para análise de processo.

Já o trabalho de (PASCHALIDES et al., 2020) propõe o Mandola, um sistema para denúncia e monitoramento de discurso de ódio online. Ele usa um algoritmo de classificação baseado em ensemble e é composto por seis componentes individuais, que se comunicam entre si para consumir, processar, armazenar e visualizar informações estatísticas sobre o discurso de ódio disseminado online.

Em (MODHA et al., 2020) os autores apresentam uma abordagem para detectar e visualizar agressão nas mídias sociais. Foi projetada uma interface de usuário baseada em um plug-in de navegador da Web no Facebook e no Twitter para visualizar os comentários agressivos postados nas linhas do tempo do usuário da mídia social. É uma solução disponível tanto para qualquer cidadão como para a indústria.

Já o (DESKTIME, 2023) e o (LLC, 2023) são soluções mais relacionadas ao controle

de produtividade do funcionário, os chamados "Bossware". Estas aplicações são utilizadas para medir a eficiência do funcionário, porém sem foco em segurança e nem na mitigação do uso de discurso de ódio.

A Tabela 1 apresenta uma comparação entre os trabalhos citados e a solução desenvolvida em relação a algumas funcionalidades. Observa-se que nenhum dos trabalhos integra todas as funções desenvolvidas no sistema. Além disso, é importante destacar que as soluções comerciais têm limitações em suas versões gratuitas, como o número de computadores a serem monitorados. Já o sistema desenvolvido não tem limitações quanto ao número de computadores e suas funcionalidades serão de livre acesso. No que diz respeito à detecção de discurso de ódio, as soluções existentes são limitadas a algumas aplicações, enquanto o sistema desenvolvido monitora tudo o que é digitado pelo usuário.

Tabela 1 – Comparação dos trabalhos relacionados.

Critérios	Kickidler	ActivTrak	Paschalides	FSense	Modha	DeskTime	staffcop	Nosso Trabalho
Captura de teclas	X	X	-	-	-	-	-	**X**
Captura de prints	X	X	-	X	-	-	-	**X**
Monitoramento de processos	X	-	-	-	-	-	-	**X**
Monitoramento do tráfego de internet	-	X	X	X	-	-	-	**X**
Alerta de vulnerabilidades	-	-	-	-	-	-	-	**X**
Alerta de discurso de ódio	-	-	Apenas no browser	-	X	-	-	**X**
Dashboard de Gerenciamento	X	X	-	X	X	-	-	**X**
Limite de computadores	6	3	-	10	-	-	-	**-**

É possível notar na tabela apresentada que, por mais que os aplicativos tenham algumas funcionalidades, são poucas e limitadas. Outro ponto importante é a limitação de computadores a serem monitorados, sendo um número muito pequeno que torna o uso desses aplicativos inviável.

4 Arquitetura da Solução

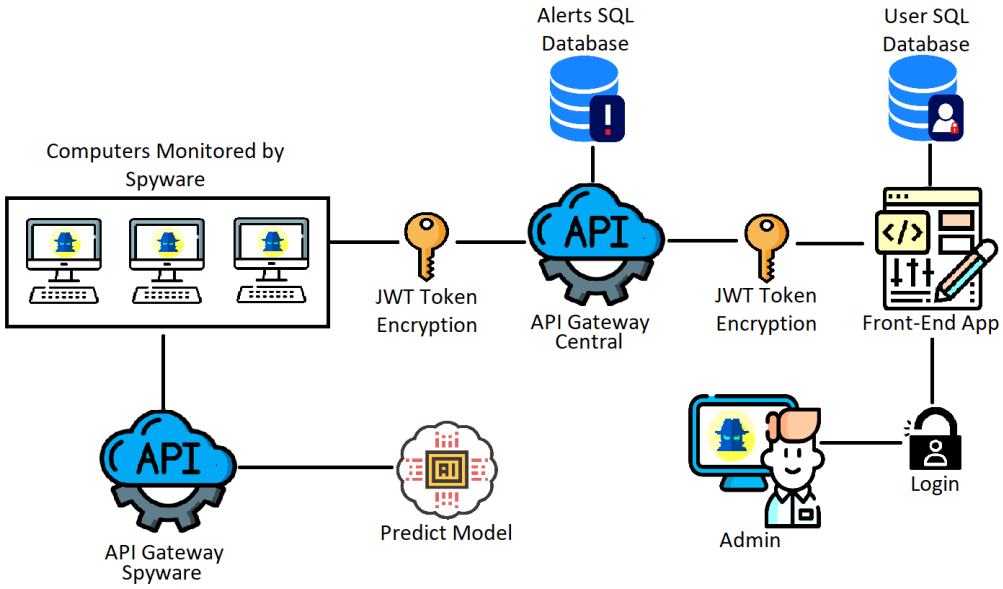
A solução adotada para o desenvolvimento baseia-se em uma arquitetura orientada a eventos, em que cada evento corresponde à geração de alertas em máquinas monitoradas por um Spyware. Essa abordagem permite uma resposta em tempo real aos eventos que ocorrem no ambiente ou no próprio sistema, eliminando a necessidade de processamento síncrono ou consultas periódicas a um banco de dados. Dessa forma, a arquitetura construída se apresentou responsiva e escalável, adequada para uma variedade de aplicações, incluindo análise de dados em tempo real, notificações em tempo real e monitoramento de alertas gerados. Com o uso dessa abordagem arquitetural, foi esperado que o sistema fosse capaz de detectar e responder rapidamente a possíveis ameaças à segurança das máquinas monitoradas.

A estratégia empregada é modular e adaptável a diferentes requisitos. Cada componente do sistema pode ser dimensionado ou atualizado independentemente, sem impactar o restante do sistema, graças à adoção de uma abordagem baseada em microsserviços. Essa abordagem utiliza várias aplicações separadas que se comunicam por meio de API Gateways, garantindo várias camadas de segurança com o uso de tokens JWT para autenticação. Além disso, a solução utiliza recursos do Spring Security, como login e criptografia de senhas, para garantir a autenticação e a autorização de endpoints específicos. Essa arquitetura proporciona maior flexibilidade e escalabilidade ao sistema, além de assegurar a proteção dos dados por meio de múltiplas camadas de segurança.

A Figura 8 apresenta a arquitetura da solução com interação entre os seus componentes. A seguir cada componente é descrito.

- Admin: administrador da plataforma que tem acesso aos Alertas através da aplicação Front-End, podendo administrar (remover e visualizar), além de adicionar os dados de gerenciamento da monitoria;
- Front-End App: aplicação responsável por criar uma interface segura, fácil de usar e simples para que o Administrador consiga gerenciar a Aplicação, e os usuários normais possam ver seus alertas, registrá-los (gerando um hash) e validar hash de alertas registrados;
- Banco de dados de Usuário: base de dados relacional para manter os usuários do Front-End separados do resto da aplicação. O banco de dados contém apenas uma tabela para configurar o login/cadastro de usuários, com senhas criptografadas;
- API Gateway Central: este componente é responsável por centralizar os dados dos Alertas e distribuir para o Front-End, com Token JWT para garantir a segurança

Figura 8 – Arquitetura da Solução Proposta



Fonte: Do autor, 2023

e confiabilidade dos dados. Além de cache para acesso rápido a dados e serviço de mensageria para garantia de entrega dos Alertas;

- Banco de dados de Alertas: principal banco de dados relacional (PostgreSQL), responsável por manter os dados de gerenciamento das monitorias e dos Alertas;
- Spyware: principal componente da aplicação que monitora os sites acessados, as palavras digitadas, os processos em execução, os discursos de ódio digitados e tem um Scanner de portas para avaliar se existem vulnerabilidades no PC. Quando qualquer um destes itens é identificado, o Spyware gera um Alerta e realiza a captura das informações para o envio ao API Gateway;
- API Gateway Spyware: componente que contém um endpoint para comunicar com o modelo de predição que vai retornar se uma frase é ou não um discurso de ódio;
- Predict Model: responsável por receber uma frase em um dos idiomas (espanhol, português ou inglês), detectar o idioma e processar através de três modelos multi camadas, retornando se a frase é um discurso de ódio ou não. O modelo é compilado com a biblioteca Pickle e inserido no API Gateway do Spyware.

Com o objetivo de facilitar o entendimento, cada módulo da arquitetura será apresentado de forma detalhada, abordando os requisitos identificados, o processo de desenvolvimento adotado e os resultados obtidos.

4.1 Requisitos Funcionais do Spyware

O Spyware desenvolvido é uma aplicação executável em Python, compilada com a biblioteca Freeze. Sua função primordial é a coleta e monitoramento de métricas específicas, incluindo processos em execução e atividades de digitação, com o propósito de gerar alertas. Essas funcionalidades foram escolhidas e planejadas com base nos requisitos funcionais da aplicação, assegurando sua eficácia na detecção de atividades potencialmente suspeitas.

Os requisitos funcionais para o Spyware são descritos a seguir e na Figura 9 está o diagrama de casos de uso correspondente.

- Iniciar o Sniffer: uma Thread deve ser iniciada para o monitoramento de transmissão de pacotes de DNS nas portas 443, 53, 80 e 8080, com o objetivo de mapear todos os sites acessados pelo usuário;
- Iniciar o Scanner: uma Thread deve ser responsável pela inicialização de um Scanner de portas, o qual vai inicializar uma nova Thread para monitorar cada porta aberta, procurando possíveis vulnerabilidades;
- Iniciar o KeyLogger: uma Thread para o KeyLogger que vai capturar todas as teclas pressionadas, com regras para mapear os atalhos mais comuns dos teclados padrão qwerty;
- Iniciar o ProcessLogger: para o ProcessLogger é feito um snapshot dos processos em execução toda vez que uma sentença for capturada no KeyLogger;
- Atualizar os dados de geração de Alertas: os dados de geração de Alertas são Blacklists de palavras ruins, sites maliciosos, banners de processos proibidos e aplicações com vulnerabilidades que não podem acessar as portas abertas dos computadores monitorados. O Spyware deve atualizar essas Blacklists toda vez que for iniciado;
- Encontrar uma sentença: o Spyware deve selecionar uma sentença para enviar para os modelos de predição. As sentenças devem ser separadas por quebra de linha;
- Encontrar sites proibidos: todo pacote rastreado pelo Sniffer deve ser comparado com os sites que estão na Balcklist, se estiver nela é um site proibido;
- Encontrar vulnerabilidades: todas as portas monitoradas que tiverem alguma aplicação com vulnerabilidades (que está definida na Balcklist) deve ser considerada uma inconsistência;
- Testar se a sentença tem uma palavra ruim: toda sentença capturada deve ser tokenizada, e cada elemento deve ser comparado com as palavras proibidas na Blacklist;

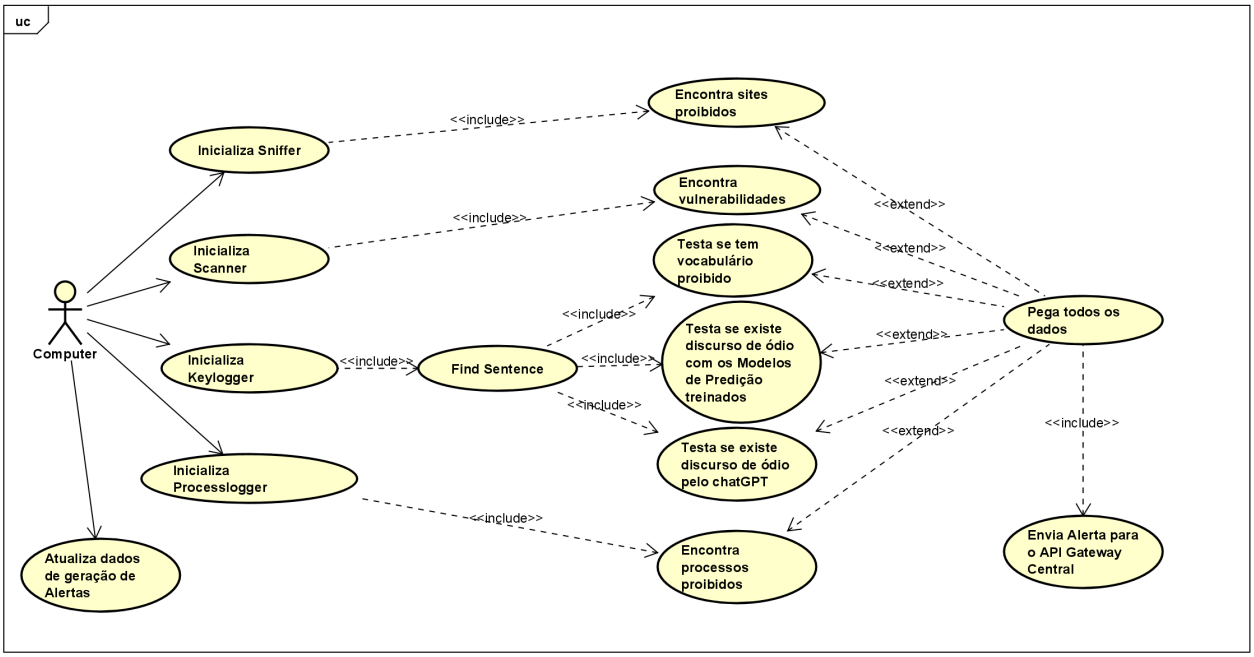
- Testar se existe discurso de ódio utilizando modelos de predição pré-treinados: as sentenças capturadas devem ser enviadas para um API Gateway do Spyware que contém os modelos de predição, onde será separada dependendo do idioma (inglês, espanhol ou português). Esta requisição terá uma resposta comunicando se a sentença é ou não discurso de ódio;

- Testar se existe discurso de ódio usando a API do GPT-3: se os modelos de predição do API Gateway do Spyware retornarem que não é discurso de ódio, é enviada uma requisição para a API do Openai que vai consultar o modelo GPT-3, retornando se a frase é ou não discurso do ódio;

- Capturar os dados: se houver qualquer inconsistência, recurso proibido ou discurso de ódio, o Spyware vai capturar os precessos em execução, o endereço Media Access Control (MAC) do computador, um printscreen da tela do usuário e o motivo da geração do Alerta;

- Enviar Alerta para o API Gateway Central: após a captura dos dados, o Spyware deve fazer login no API Gateway Central, obtendo um token que é usado para enviar uma requisição de Geração de Alerta.

Figura 9 – Diagrama de casos de uso do Spyware



Fonte: Do autor, 2023

4.2 Requisitos Funcionais do API Gateway Central

Para o desenvolvimento do API Gateway Central foram estabelecidos alguns requisitos funcionais para mapear e alcançar as funcionalidades necessárias na solução.

A Figura 10 apresenta o diagrama de casos de uso para o API Gateway Central que tem interações com os atores Front-end e Spyware, refletindo os requisitos funcionais que foram definidos a seguir:

- Implementar Monitorias: Para possibilitar a detecção de atividades suspeitas, o API Gateway deve implementar monitorias específicas para verificar o acesso a sites bloqueados, consultas DNS maliciosas e outras atividades indesejadas. Essas monitorias são acionadas de acordo com as configurações definidas e geram os respectivos Alertas para análise.

- Integração com o Front-end e Spyware: O API Gateway deve garantir a integração perfeita entre o Front-end e o Spyware, permitindo que ambos se comuniquem eficientemente com o sistema. Isso inclui fornecer endpoints bem definidos, padronizar as requisições e respostas e garantir a autenticação correta de usuários e aplicações.

- Gerenciar Versões: Para possibilitar a evolução da aplicação e manter a compatibilidade com versões anteriores, o API Gateway deve ser capaz de gerenciar diferentes versões de seus endpoints. Isso permite a implementação de novas funcionalidades sem comprometer a integridade de versões já existentes.

- Segurança e Criptografia: O API Gateway deve ser projetado com práticas de segurança em mente, garantindo a proteção dos dados sensíveis e informações dos usuários. O uso de protocolos de criptografia, como HTTPS, é essencial para proteger as comunicações entre os clientes e o servidor.

- Escalabilidade e Desempenho: Considerando que a aplicação pode enfrentar crescimento no número de usuários e carga de requisições, o API Gateway deve ser projetado para ser escalável e manter o desempenho mesmo em cenários de alta demanda.

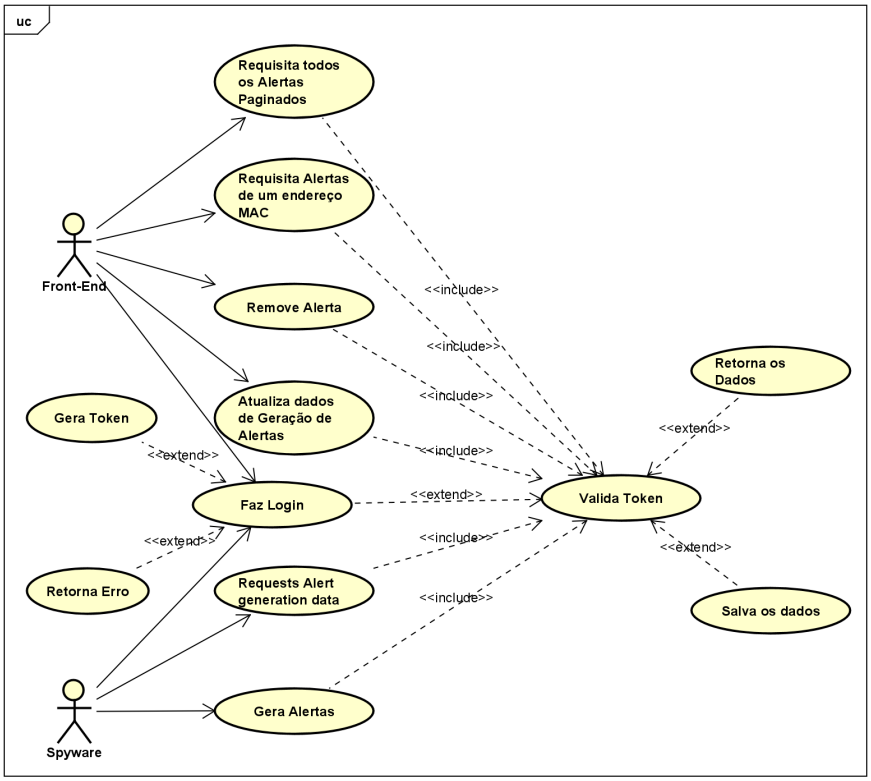
- Testes Automatizados: É essencial realizar testes automatizados para garantir a qualidade e estabilidade do API Gateway. Testes unitários, de integração e de carga devem ser realizados regularmente para identificar possíveis falhas e garantir a consistência do sistema.

- Documentação Completa: O API Gateway deve ser devidamente documentado, descrevendo detalhadamente todos os endpoints, suas funcionalidades, parâmetros esperados, códigos de resposta e exemplos de uso. Uma documentação clara e

abrangente facilita a compreensão e uso do sistema por parte de desenvolvedores e usuários.

- Monitoramento e Registro de Logs: Implementar mecanismos de monitoramento e registro de logs é fundamental para acompanhar o desempenho do sistema, identificar possíveis problemas e tomar ações corretivas de forma proativa.

Figura 10 – Diagrama de casos de uso do API Gateway Central



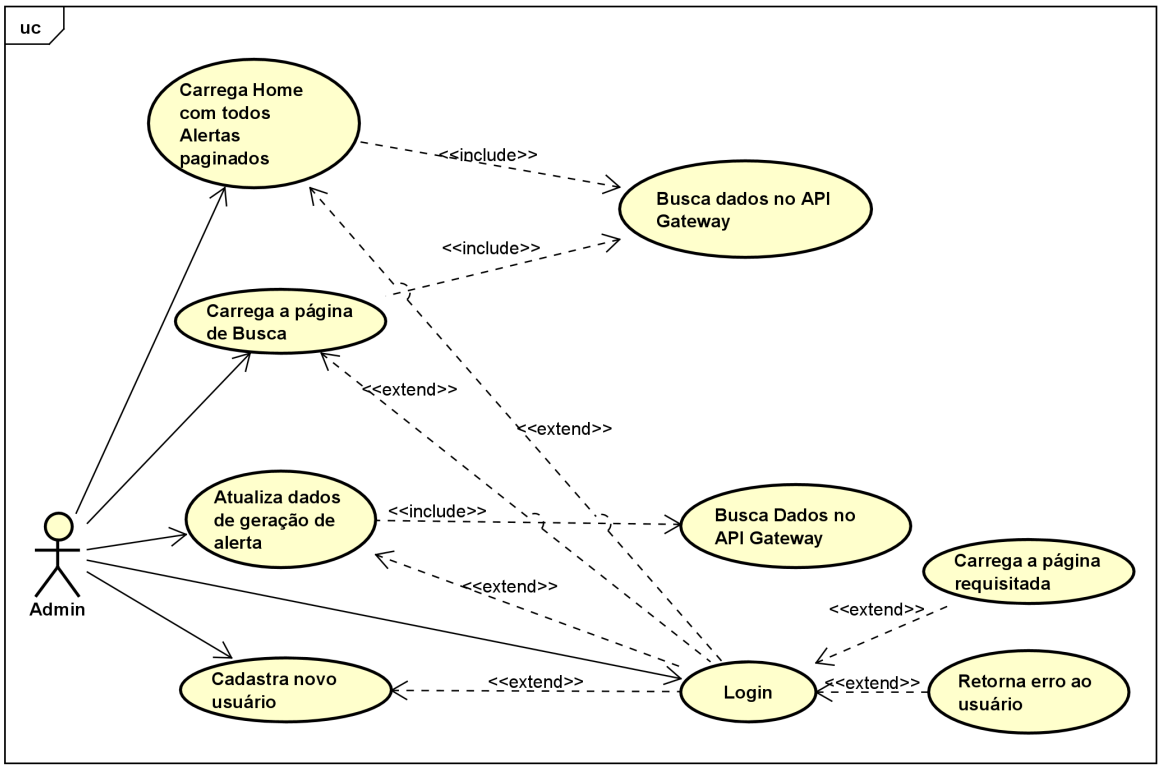
Fonte: Do autor, 2023

4.3 Requisitos Funcionais da Aplicação Front-End

Na Figura 11 está o diagrama de casos de uso relativo a aplicação Front-End, representando os requisitos funcionais listados a seguir:

- Página Home com todos os Alertas paginados, para Administradores: a página inicial da aplicação exibe uma lista paginada de todos os Alertas registrados no sistema, destinada aos administradores. Essa página faz uma chamada para o API Gateway, que fornece os dados dos Alertas, incluindo todas as informações capturadas, bem como o motivo da geração do Alerta. Essa funcionalidade permite que os administradores tenham uma visão geral de todos os Alertas no sistema, facilitando o gerenciamento e a tomada de decisões;

Figura 11 – Diagrama de casos de uso do Front-End



Fonte: Do autor, 2023

- Página Home com todos os Alertas do usuário logado: similar à funcionalidade anterior, a página inicial também exibe os Alertas, mas neste caso, são apenas os Alertas do usuário logado na aplicação. Essa página também faz uma chamada para o API Gateway para recuperar os Alertas específicos do usuário. Dessa forma, os usuários têm acesso rápido e conveniente a todas as informações dos Alertas associados à sua conta;

- Página de busca de Alertas: a aplicação disponibiliza uma página dedicada à busca de Alertas, permitindo que os usuários filtrem os Alertas com base no endereço MAC do computador onde o Alerta foi gerado. Essa funcionalidade agiliza a localização de Alertas específicos, tornando a navegação mais eficiente e permitindo que os usuários encontrem informações relevantes de forma rápida;

- Páginas para atualizar as ”Blacklists”: a aplicação possui páginas específicas para a atualização das ”Blacklists”, que são listas contendo sites maliciosos, processos proibidos, palavras impróprias e vulnerabilidades de portas que geram os Alertas. Essas páginas permitem que os administradores adicionem novos itens às listas ou removam entradas existentes, fornecendo controle total sobre o conteúdo das ”Blacklists”;

- Registrar novos usuários: a aplicação oferece um recurso para o registro de novos

usuários, permitindo que eles criem suas contas de acesso. Ao se registrar, os usuários podem usufruir das funcionalidades específicas destinadas a eles, incluindo a visualização dos Alertas relacionados à sua conta;

- Fazer Login: a aplicação implementa um sistema de acesso às páginas restrito apenas para usuários logados. Os usuários devem fazer login utilizando suas credenciais antes de acessar qualquer funcionalidade protegida;
- Fazer Logout: o sistema permite o logout, encerrando a sessão do usuário e garantindo a segurança do acesso;
- Dois níveis de acesso para usuários: a aplicação adota um sistema de controle de acesso baseado em níveis de acesso, que são definidas como administradores e usuários normais. Os administradores têm permissões mais amplas, permitindo acesso total a todas as funcionalidades e dados do sistema, enquanto os usuários normais têm acesso restrito a determinadas funcionalidades e apenas aos Alertas associados à sua conta;
- Registrar um Alerta já gerado pela aplicação via hash: a aplicação disponibiliza um botão que permite aos usuários baixarem um JSON contendo informações de um Alerta específico e um hash gerado pela aplicação. Esse hash pode ser validado por meio de um formulário na mesma página, possibilitando o registro manual de Alertas que tenham sido gerados previamente pelo sistema.

4.4 Requisitos Funcionais para o API Gateway do Spyware

A fim de viabilizar a análise de discurso de ódio no Spyware, foi necessário desenvolver uma API que se integra com os Modelos de Predição compilados. Essa aplicação é caracterizada pela sua simplicidade e clareza, consistindo em um único endpoint projetado para receber uma frase como entrada e retornar uma avaliação sobre se ela contém ou não discurso de ódio.

Para atingir esse objetivo, foram definidos os seguintes requisitos funcionais:

- Descompilação do Modelo de Predição: A API deve ser capaz de descompilar o modelo de predição a fim de aplicar a classificação das frases;
- Pré-processamento dos Dados de Entrada: Todos os dados recebidos pela API devem ser submetidos a um pré-processamento que inclui a remoção de tags, comandos e outros caracteres que possam interferir na classificação;
- Vetorização das Frases: Após o pré-processamento, as frases precisam ser vetorizadas para que possam ser utilizadas nos classificadores;

- Endpoint da API: É necessário implementar um endpoint do tipo POST que aceite uma frase como entrada e forneça uma resposta contendo o resultado da classificação, indicando se há presença ou ausência de discurso de ódio na frase.

4.5 Requisitos não funcionais

A solução foi concebida com base em uma arquitetura sólida e abrangente, atendendo a diversos requisitos não funcionais essenciais. Abaixo, é apresentado a lista detalhada dos requisitos não funcionais da solução, que foram considerados e implementados para garantir a eficiência, segurança e escalabilidade do sistema:

- **Arquitetura baseada em microsserviços:** a aplicação é dividida em microsserviços com funções delimitadas, permitindo escalabilidade e facilidade de manutenção individual;
- **Armazenamento dos dados:** os Alertas gerados são armazenados em um API Gateway separado do Front-End, garantindo maior segurança e disponibilidade;
- **Segurança:** a aplicação conta com mecanismos de segurança em cada ponto, como login com senhas criptografadas no Front-End e token JWT no API Gateway;
- **Modelo de Predição:** uma API separada utiliza modelos pré-treinados para classificar frases em discurso de ódio ou não, empregando técnicas avançadas de processamento de linguagem natural e aprendizado de máquina;
- **Spyware:** um script assíncrono é instalado nos computadores a serem monitorados, coletando dados como palavras, sites e processos proibidos, além de realizar análise de discurso de ódio, garantindo a coleta eficiente e discreta das informações;
- **Monitorias:** o Spyware realiza monitorias separadas e paralelas, incluindo análise de palavras, sites e processos proibidos, Scanner de portas e análise de discurso de ódio, possibilitando a detecção proativa de potenciais ameaças;
- **Performance:** A arquitetura deve ser construída de forma que suporte o maior número de requisições possíveis em ambientes de testes e, eventualmente, em ambiente real.

5 Desenvolvimento da Aplicação

Neste capítulo são apresentados detalhes da solução, descrevendo o processo de desenvolvimento desde a definição da arquitetura até a implementação de cada microsserviço. O objetivo é fornecer uma visão completa do processo de concepção, implantação e dos fluxos criados para o sistema.

Para uma melhor organização, o capítulo está dividido em seções específica, fornecendo uma explicação detalhada de cada componente do sistema. Em seguida, são detalhadas as estratégias desenvolvidas para alcançar os requisitos não funcionais, abrangendo aspectos como escalabilidade, segurança e desempenho. Além disso, são explorados os fluxos desenvolvidos, com a descrição das interações e a sequência de atividades entre os microsserviços.

Por fim, é exposta a abordagem utilizada para a implantação e atualização do sistema, destacando os procedimentos adotados para garantir a integridade e disponibilidade da solução. Essas informações detalhadas permitirão ao leitor obter uma compreensão abrangente do processo de desenvolvimento e das funcionalidades implementadas no sistema.

5.1 Spyware

O desenvolvimento do Spyware foi realizado em conformidade com os requisitos estabelecidos e a arquitetura planejada. A seguir são apresentados detalhes sobre a implementação dessa aplicação, enfatizando o seu papel de monitoramento dos elementos essenciais e a coleta de dados no computador hospedeiro.

O Spyware foi concebido como uma solução capaz de rastrear e monitorar os elementos identificados como relevantes para o contexto específico. A implementação da aplicação foi realizada considerando a estrutura arquitetural definida anteriormente, garantindo a integração adequada com os demais componentes do sistema.

Na sequência tem-se detalhes sobre a implementação do Spyware, desde a seleção e implementação das técnicas de monitoramento até a integração do software com a arquitetura. São explorados os principais aspectos técnicos e funcionais, com o intuito de fornecer uma compreensão aprofundada do funcionamento e das capacidades dessa aplicação de monitoramento.

5.1.1 Monitorias

Para alcançar os objetivos e requisitos definidos o script do Spyware teve algumas monitorias implementadas através de API's e bibliotecas de sistema operacional. Desta forma, o script funciona em algumas Threads, se dividindo em algumas funções: uma Thread para um Sniffer que vai analisar os tráfego de pacotes, uma Thread para o KeyLogger o qual vai capturar as teclas pressionadas, uma Thread para analisar os processos em execução e uma Thread para analisar as portas abertas e em execução no computador.

Cada uma dessas Threads usam estratégias específicas para analisar os pontos monitorados:

- Sniffer: um componente de monitoramento de rede que captura e analisa pacotes em tempo real. Essa classe é projetada para operar em uma thread separada e possui vários atributos, como uma lista para armazenar logs, conjuntos para armazenar consultas DNS capturadas e sites bloqueados, e contadores para rastrear o número de pacotes capturados. O método principal, sniffer(), verifica se um pacote é um pacote DNS com carga útil e se a porta de origem corresponde a portas específicas. Se a consulta DNS corresponder a um site bloqueado (que está em uma Blacklist) e não tiver sido registrada anteriormente, um alerta é gerado e adicionado à lista de logs. A captura de pacotes ocorre no método run(), usando a biblioteca scapy, e o processamento de cada pacote é feito no método process_packet();

- KeyLogger: classe que registra as teclas pressionadas pelo usuário. O construtor da classe inicializa uma variável de log vazia. O método callback() é responsável por manipular eventos de teclas pressionadas. Se o evento for uma tecla "enter", o método report() é chamado para analisar o log e, se necessário, enviar um alerta. Se a tecla for um espaço em branco, um espaço é adicionado ao log. Se for a tecla "backspace", o último caractere do log é removido. A tecla "caps lock" é ignorada. Para outras teclas, o método get_shift_chars() é chamado para obter o caractere correspondente à tecla pressionada, considerando se a tecla "shift" também está sendo pressionada. O caractere é então adicionado ao log. O método get_shift_chars() retorna um dicionário mapeando combinações de teclas "shift" para caracteres especiais. O método report() verifica se o log não está vazio e se contém algum conteúdo indesejado, como discurso de ódio, linguagem imprópria ou processos maliciosos. Em caso afirmativo, um alerta é enviado. O log é então reiniciado. O método start() inicia o keylogger, definindo o método callback() como manipulador de eventos de teclado e aguardando por eventos;

- Scanner: realiza varreduras em busca de portas abertas em um alvo especificado. Ela herda funcionalidades da classe threading.Thread, permitindo sua execução em uma thread separada. O scanner utiliza a biblioteca socket para estabelecer

conexões com as portas do alvo e coletar banners, que são informações retornadas pelos serviços executando nessas portas. Durante a varredura, os números das portas abertas e os banners são armazenados nas listas ports e banners, respectivamente. Em seguida, a classe verifica se algum banner corresponde a vulnerabilidades conhecidas, comparando-os com um arquivo contendo banners vulneráveis. Caso uma correspondência seja encontrada, um alerta é gerado e uma mensagem de log é registrada;

- ProcessLogger: tem o objetivo de identificar e lidar com possíveis processos maliciosos em execução no sistema. A função "are_malicious_process(self)" verifica se existem processos maliciosos comparando os nomes dos processos em execução com uma lista fornecida em um arquivo. Caso seja detectado um processo malicioso, a função realiza ações como encerrá-lo e registrar um alerta correspondente. Já a função "get_process()" obtém uma lista atualizada dos nomes dos processos em execução. Essas funções podem ser utilizadas em conjunto para detectar atividades suspeitas no sistema e tomar medidas apropriadas.

5.1.2 Dados capturados

A captura de informações relevantes desempenha um papel fundamental na análise dos alertas gerados pelo spyware. Com base nisso, foi desenvolvido um script capaz de coletar dados essenciais para essa análise. Esses dados incluem os processos em execução, o endereço MAC do computador monitorado, uma imagem da tela no momento do alerta e o motivo que gerou o alerta.

A obtenção dos processos em execução é realizada por meio da função "get_process()" mencionada anteriormente. Essa função é responsável por obter uma lista dos nomes dos processos em execução no sistema. Além disso, o endereço MAC do computador pode ser obtido utilizando a biblioteca "get_mac_address" do Python, que permite obter o endereço MAC da interface de rede do sistema monitorado.

Quanto ao motivo que gerou o alerta, esse dado é obtido por meio das monitorias que foram implementadas e que foram abordadas anteriormente. Essas monitorias são responsáveis por verificar atividades suspeitas, como acesso a sites bloqueados, consultas DNS maliciosas, detecção de palavras-chave indesejadas, entre outras. A partir dessas monitorias é possível identificar qual ação específica desencadeou o alerta.

Por fim, a captura da imagem da tela no momento do alerta é realizada utilizando a biblioteca "ImageGrab". Essa biblioteca permite capturar a imagem da tela e salvá-la em um buffer. Em seguida, o buffer é codificado em base64 e convertido em uma string. Dessa forma, a imagem da tela pode ser armazenada e analisada posteriormente como parte dos dados relacionados ao alerta.

Essas estratégias de captura de informações são essenciais para a análise dos alertas gerados pelo spyware, permitindo obter um contexto mais completo sobre as atividades suspeitas ocorridas no sistema monitorado. É importante ressaltar que a implementação dessas funcionalidades deve ser realizada de acordo com as políticas de segurança e as permissões adequadas, garantindo a conformidade com as normas e regulamentações vigentes.

5.1.3 Atualização das Balcklists no API Gateway Central

Durante o processo de monitoria e geração de alertas o script analisa os pontos que estão em execução (DNS, processos, portas abertas e palavras digitadas) e compara com arquivos que contém listas de itens que não podem aparecer no retorno dessas monitorias. Para que isso ocorra, estes arquivo precisam estar atualizados com o que o administrador definiu na aplicação Front-End, a qual, salvou as informações no API Gateway Central.

Desta forma, foi criada a função "update_aux_data()" que tem como objetivo realizar a atualização dos dados auxiliares utilizados pelo programa. Esses dados são obtidos a partir de requisições a diferentes endpoints de uma API local. O processo de atualização envolve a obtenção dos dados de diferentes categorias, como palavras de linguagem imprópria, portas vulneráveis, processos maliciosos e sites bloqueados.

Para realizar a atualização, a função inicia realizando o login necessário para obter as credenciais de acesso à API. Em seguida, são feitas requisições para cada endpoint específico, utilizando as credenciais obtidas anteriormente. Os dados retornados pelas requisições são obtidos em formato JSON.

Após obter os dados de cada categoria, a função os armazena em arquivos de texto específicos. Para isso, são utilizados loops para percorrer os dados obtidos e escrevê-los nos arquivos correspondentes. Cada dado é registrado no arquivo separado por um ponto e vírgula (;).

Caso ocorra algum erro durante o processo de atualização, uma mensagem de log é registrada, informando o ocorrido.

Essa função desempenha um papel importante no sistema, pois permite a obtenção e atualização dos dados auxiliares necessários para o correto funcionamento do programa. Os dados auxiliares são utilizados em outras partes do sistema para fins de análise, detecção e tomada de decisões.

5.1.4 Integração com os Modelos de Predição

Após a configuração das blacklists, a implementação das monitorias e a captura dos dados necessários, é crucial abordar a detecção de discurso de ódio como último ponto a ser considerado. Para permitir que o spyware identifique frases contendo discurso de

ódio, aproveitou-se o KeyLogger para capturar as sentenças digitadas pelo usuário, as quais são separadas por quebras de linha. Em seguida, é realizada uma requisição ao API Gateway dos Modelos de Predição, responsável por determinar se a frase enviada constitui discurso de ódio ou não.

A requisição enviada é feita por meio do protocolo HTTP, utilizando o método POST e um body em formato JSON, contendo a frase a ser avaliada. Essa requisição é encaminhada ao API Gateway dos Modelos de Predição para processamento. Após receber a resposta, uma análise é realizada para verificar se a frase é classificada como discurso de ódio. Em caso afirmativo, um alerta é gerado, indicando a detecção desse tipo de conteúdo.

Dessa forma, a detecção de discurso de ódio é incorporada ao funcionamento do spyware, permitindo identificar e agir diante de frases ofensivas ou prejudiciais. É importante ressaltar que a abordagem adotada envolve o uso de uma API externa para a classificação do conteúdo, garantindo uma análise mais precisa e atualizada.

5.2 API Gateway do Spyware

Para possibilitar a utilização dos modelos de predição e detecção de discurso de ódio em frases capturadas pelo Spyware, foi desenvolvido um API Gateway que incorpora os modelos pré-treinados e compilados em formato .pkl. Esse gateway atua como um intermediário entre os usuários e os modelos, permitindo que os mesmos sejam acessados através de uma interface amigável e simplificada.

O funcionamento do API Gateway inicia com um endpoint que recebe uma frase como entrada. Essa frase é então submetida a um processo de normalização e vetorização, onde são aplicadas técnicas para tratar e padronizar o texto, preparando-o para ser processado pelos modelos de predição. Em seguida, o sistema realiza a identificação do idioma presente na frase, direcionando-a para o modelo adequado, correspondente à língua do texto.

O modelo específico para o idioma da frase analisa o texto e avalia se o discurso de ódio está presente ou não. Isso é possível graças à capacidade dos modelos de aprendizado supervisionado, que foram treinados em conjuntos de dados contendo exemplos de textos com e sem discurso de ódio, permitindo a identificação de padrões e características associadas a cada classe.

Após a análise, o resultado é retornado para a requisição do usuário. Caso seja identificado discurso de ódio na frase, o API Gateway informará que o sentimento é negativo e que o texto contém elementos de discurso de ódio. Caso contrário, o sistema indicará que o sentimento é neutro ou positivo, indicando que não foram encontrados indícios de discurso de ódio na frase.

Essa abordagem permite que os usuários do Spyware possam, de forma prática e eficiente, avaliar o conteúdo das frases capturadas e identificar potenciais casos de discurso de ódio. Além disso, a utilização de um API Gateway centralizado e otimizado para processar as requisições torna o sistema mais escalável e de fácil manutenção, garantindo uma análise rápida e precisa mesmo em situações de alto volume de requisições. Dessa forma, o API Gateway se mostra como um componente fundamental na integração dos modelos de predição com o Spyware, proporcionando uma solução completa e confiável para a detecção de discurso de ódio em textos.

5.2.1 Identificação do idioma

A identificação do idioma da frase recebida é realizada pelo API Gateway utilizando a biblioteca langdetect. Quando uma requisição é feita para o endpoint /predict, o API Gateway recebe a frase enviada pelo usuário, e em seguida, o método detect é acionado para determinar o idioma do texto. Essa funcionalidade é crucial, pois os modelos de predição para detecção de discurso de ódio foram treinados para cada idioma especificamente.

Após a identificação do idioma, o API Gateway é capaz de direcionar corretamente a frase para o modelo de predição adequado. No caso do exemplo fornecido, três modelos foram carregados e pré-treinados, um para cada idioma suportado: português (model_pt), inglês (model_en) e espanhol (model_sp).

Uma vez determinado o idioma da frase, o API Gateway utiliza a classe TextProcessor definida no módulo TextTokenizer para realizar a normalização e vetorização do texto. A classe TextProcessor contém os métodos necessários para realizar o pré-processamento textual, incluindo remoção de stopwords, stemming e outras etapas de limpeza. Isso é essencial para padronizar o texto de entrada e prepará-lo para ser usado nos modelos de predição.

O texto normalizado e vetorizado é então submetido ao modelo de predição específico para o idioma detectado. Para isso, são realizadas algumas etapas adicionais para garantir que o vetor de entrada tenha a mesma dimensão esperada pelo modelo. Caso o texto contenha menos palavras do que o modelo está esperando, são adicionados valores '0' ao vetor para preencher as posições vazias.

5.2.2 Retorno da API

O retorno da API é feito de forma estruturada e consistente, seguindo um padrão JSON para apresentar os resultados da detecção de discurso de ódio. Quando uma requisição é feita para o endpoint /predict, o API Gateway processa a frase enviada pelo usuário, identifica o idioma e realiza a classificação usando o modelo apropriado para aquele idioma.

Após o processamento e classificação, os resultados são armazenados em um

DataFrame do pandas chamado df_raw. Esse DataFrame contém a coluna frase, que é a frase original enviada pelo usuário, e a coluna valor, que indica se a frase foi classificada como contendo discurso de ódio (valor 1) ou não (valor 0).

Em seguida, é utilizado o método to_json do pandas para transformar o DataFrame em uma estrutura JSON. A opção orient='records' é definida para que cada linha do DataFrame seja convertida em um objeto JSON independente, criando assim uma lista de objetos JSON, um para cada frase classificada.

O resultado JSON contém as frases originais e os valores de classificação obtidos para cada uma delas. Esse resultado é retornado na resposta da requisição feita ao API Gateway.

A estrutura do JSON de resposta pode ser exemplificada na Figura 33, com uma lista de objetos com uma frase cada, obtendo o valor 1 para quando a frase tem sentimento de ódio e 0 para quando não tem.

Figura 12 – JSON de resposta

```
[
    {
        "frase": "Exemplo de frase 1.",
        "valor": 1
    },
    {
        "frase": "Outro exemplo de frase.",
        "valor": 0
    },
    ...
]
```

Fonte: Do autor, 2023

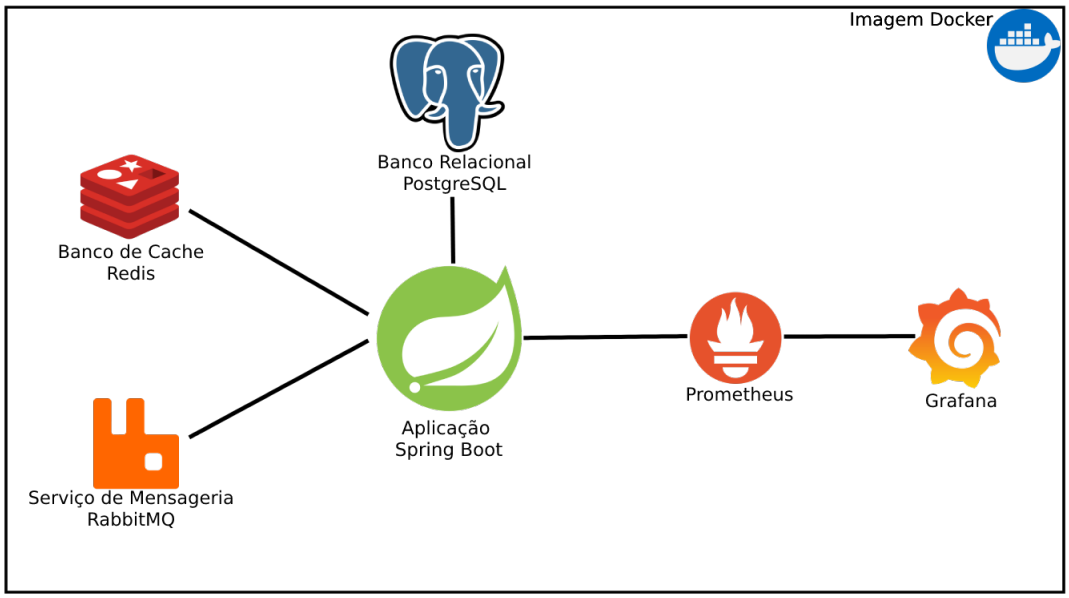
5.3 API Gateway Central

Nesta seção é detalhada a implementação e o processo de desenvolvimento do API Gateway, como os endpoints criados, a utilização de cache com Redis para otimização de consultas, o uso de filas do RabbitMQ para gerenciamento eficiente de tarefas assíncronas, a aplicação de migrações de banco com Flyway para garantir a integridade dos dados e as melhorias de performance implementadas para lidar com altas cargas de requisições. Ao analisar esses aspectos, tem-se uma visão abrangente e aprofundada da estrutura do API Gateway, compreendendo as escolhas arquiteturais e estratégias para garantir um sistema eficiente, escalável e confiável.

Para alcançar a modularização e o gerenciamento eficiente dos dados na aplicação, foi desenvolvido um API Gateway central responsável por coordenar todas as comunicações entre os diferentes pontos do sistema, incluindo o armazenamento de imagens, alertas, usuários e blacklists. Para garantir um alto desempenho mesmo em situações de alta carga de requisições, foi dada especial atenção à performance desse módulo. Todos os endpoints desenvolvidos podem ser visualizados no Apêndice A.

A Figura 13 ilustra a integração das tecnologias utilizadas no desenvolvimento do API Gateway central, em especial para melhorar a performance, a observabilidade e facilitar a implantação em ambientes variados. A API criada é suportada por diversas tecnologias, como a utilização de contêineres Docker para isolar o serviço web juntamente com um banco de dados PostgreSQL, responsável por armazenar os dados de longa vida. Além disso, um banco Redis é utilizado para armazenar dados de cache com vida curta, o que otimiza o acesso a informações frequentemente requisitadas.

Figura 13 – Integração das tecnologias do API Gateway Central



Fonte: Do autor, 2023

A aplicação também incorpora o serviço de mensageria RabbitMQ, que possibilita o gerenciamento eficiente de filas de processamento, garantindo uma distribuição adequada e escalonamento das tarefas. Dois serviços adicionais, o Prometheus e o Grafana, são empregados para a extração e visualização de métricas, permitindo um monitoramento detalhado do desempenho do sistema em tempo real.

Dessa forma, a aplicação apresenta uma abordagem sólida e robusta na implementação das tecnologias, fornecendo uma base sólida para aplicar com sucesso as funcionalidades propostas. Com a contextualização do uso das tecnologias, a aplicação desenvolvida é

capaz de atender aos requisitos de forma eficiente, oferecendo uma solução completa e escalável para o monitoramento e detecção de atividades suspeitas.

5.3.1 Integração com Redis e implantação de Cache

Para melhorar o desempenho e reduzir a carga no banco de dados foi adicionado cache com o Redis para o endpoint /alert. O objetivo do cache é armazenar temporariamente os resultados das consultas mais frequentes para que possam ser rapidamente recuperados sem a necessidade de acessar o banco de dados todas as vezes.

A configuração do cache foi realizada através do arquivo de configuração RedisConfig.java. Neste arquivo, foi criado um Bean chamado "redisCacheManagerBuilderCustomizer" que personaliza o gerenciador de cache do Redis. Através do método "withInitialCacheConfigurations", foi configurado um cache chamado "alerta" com uma expiração de 20 segundos (entryTtl(Duration.ofSeconds(20))). Isso significa que os dados do cache serão armazenados por 20 segundos antes de serem considerados obsoletos e, portanto, atualizados a partir do banco de dados.

Quando o endpoint /alert é acessado, a primeira consulta resultante é armazenada em cache com o nome "alerta". Nas próximas requisições para o mesmo endpoint, o sistema verifica se os dados estão no cache. Se estiverem e ainda não tiverem expirado, os dados são retornados diretamente do cache, evitando a necessidade de uma nova consulta ao banco de dados. Esse mecanismo reduz o tempo de resposta do endpoint e alivia a carga no banco de dados, proporcionando uma melhor experiência para o usuário.

O tempo de expiração do cache foi definido como 20 segundos para garantir que os dados estejam sempre atualizados o suficiente para atender aos requisitos do sistema. Se os dados forem alterados no banco de dados, eles serão atualizados no cache na próxima consulta.

Essa abordagem de cache com o Redis é especialmente útil em endpoints que exigem acesso frequente a dados estáticos ou que não mudam com muita frequência. No caso específico do endpoint /alert, em que os dados dos alertas raramente mudam, o cache oferece um grande benefício ao reduzir a latência e melhorar o desempenho geral do sistema.

5.3.2 Migração de Banco de Dados com Flyway

Para realizar as migrações do banco de dados PostgreSQL e criar as tabelas, foi utilizada a ferramenta FlywayDB em conjunto com o Spring Framework. O Flyway é uma biblioteca de código aberto que permite a aplicação de migrações de banco de dados de forma automatizada e controlada, garantindo que as alterações sejam aplicadas na ordem correta e de maneira consistente.

Ao iniciar a aplicação Spring, o Flyway é automaticamente acionado para verificar se existem novas migrações a serem aplicadas no banco de dados. O Flyway mantém um registro das migrações já aplicadas em uma tabela especial (flyway_schema_history), evitando que as mesmas migrações sejam executadas novamente.

O trecho de código SQL no arquivo de migração do Apêndice B começa com a criação de cada tabela (alert, image, bad_language, malicious_port, malicious_process e malicious_website) usando a cláusula "CREATE TABLE IF NOT EXISTS". Essa cláusula garante que as tabelas só serão criadas se ainda não existirem no banco de dados, evitando erros caso o script seja executado mais de uma vez.

Em seguida, são definidas as colunas de cada tabela, juntamente com suas restrições, como chaves primárias (PRIMARY KEY), tipos de dados (character varying, bytea) e outras restrições (UNIQUE, NOT NULL).

Após a criação das tabelas, são feitas inserções iniciais na tabela de usuários (usuario). Isso é realizado para fornecer alguns dados de exemplo para a aplicação, incluindo senhas codificadas usando um algoritmo de criptografia (bcrypt). Essas senhas são usadas para fins de autenticação de usuários na aplicação.

O Flyway cuida da execução dessas migrações automaticamente, garantindo que as tabelas sejam criadas corretamente no banco de dados e que os dados iniciais sejam inseridos de forma consistente.

Dessa forma, o uso do FlywayDB em conjunto com o Spring Framework simplifica o gerenciamento e a aplicação de migrações de banco de dados, mantendo a estrutura da base de dados atualizada e sincronizada com a evolução da aplicação. Isso proporciona maior agilidade e segurança no desenvolvimento e manutenção do sistema, garantindo que as alterações no esquema do banco de dados sejam aplicadas de maneira controlada e sem perda de dados.

5.3.3 Filas de mensageria com RabbitMQ

A integração do Spring com o RabbitMQ foi realizada utilizando as classes AlertConsumer, RabbitMQConection, RabbitmqConfig, CustomErrorStrategy, e a classe de constantes RabbitmqConstantes. Essas classes trabalham em conjunto para implementar filas, consumers e a comunicação assíncrona entre diferentes componentes do sistema.

A classe AlertConsumer é um exemplo de consumer do RabbitMQ. Ela utiliza a anotação @RabbitListener para indicar que está escutando a fila FILA_ALERT. Quando uma mensagem é recebida na fila, o método consumerAlert é acionado. Nesse método, a mensagem em formato JSON é convertida em um objeto do tipo Alert utilizando a biblioteca Jackson. Em seguida, o serviço AlertService é utilizado para salvar o alerta no banco de dados.

A classe RabbitMQConection é responsável por criar as filas, trocas e relacionamentos entre eles. No método adiciona(), é criada a fila filaAlerta e a troca troca através dos métodos fila() e trocaDireta(). O relacionamento entre a fila e a troca é definido pelo método relacionamento(), e em seguida, as filas e trocas são declaradas no RabbitMQ utilizando o AmqpAdmin.

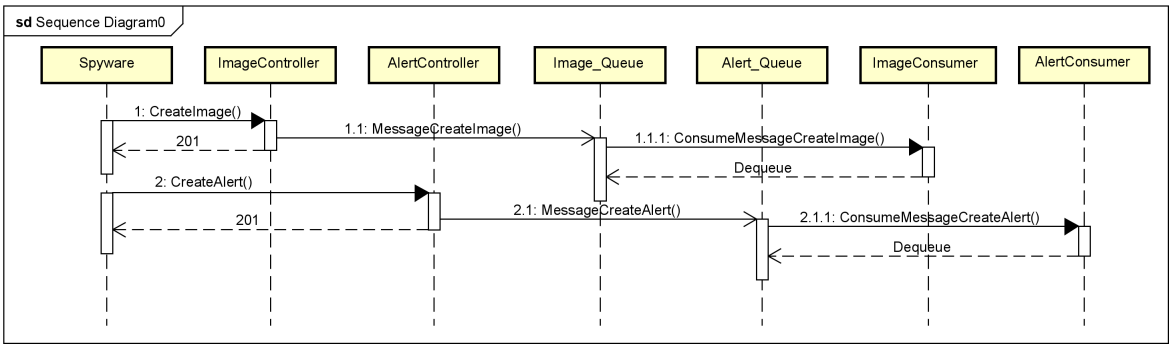
A classe RabbitmqConfig configura o listener do RabbitMQ. O método rabbitListenerContainerFactory() cria uma instância do DirectRabbitListenerContainerFactory, que é responsável por criar o DirectMessageListenerContainer. Através dessa configuração, o listener irá consumir as mensagens da fila FILA_ALERT e processá-las com o consumer AlertConsumer. Além disso, o AcknowledgeMode é definido como AUTO para que as mensagens sejam automaticamente confirmadas após serem processadas.

A classe CustomErrorStrategy é uma extensão da estratégia de tratamento de exceções do RabbitMQ. Ela é utilizada como ErrorHandler no RabbitmqConfig. Nessa classe, é possível personalizar o tratamento de exceções lançadas durante o processamento das mensagens. No exemplo fornecido, é feita uma verificação se a causa da exceção é uma IllegalArgumentException, e caso seja, a mensagem é considerada como fatal, ou seja, não será reenviada para a fila de mensagens rejeitadas.

A classe de constantes RabbitmqConstantes é utilizada para definir o nome da fila FILA_ALERT. Essa abordagem de uso de constantes é uma boa prática para evitar repetições de strings no código e facilitar futuras alterações nos nomes das filas.

O diagrama da Figura 14 ilustra o fluxo completo de gerenciamento na criação de um alerta juntamente com o fluxo concomitante de cadastro da imagem que vai estar associado a ele. A figura expressa como acontece o fluxo descrito desde a requisição feita pelo Spyware, passando pelos Controllers, cadastrando as mensagens nas filas e finalizando com o consumo e cadastro das mensagens pelos Consumers.

Figura 14 – Fluxo da Fila de Alertas



Fonte: Do autor, 2023

Essas classes em conjunto permitem uma comunicação assíncrona e escalável entre os diferentes componentes do sistema, tornando-o mais robusto e eficiente. Através da

integração do Spring com o RabbitMQ, é possível criar sistemas distribuídos que processam mensagens de forma assíncrona, facilitando a implementação de funcionalidades como processamento paralelo, balanceamento de carga e tratamento de picos de tráfego.

5.3.4 Melhorias de performance aplicadas

Para aprimorar a performance e a entrega contínua da aplicação, foram implementadas diversas medidas estruturais e de desenvolvimento, com o objetivo de otimizar o desempenho e garantir uma experiência mais ágil e responsiva para os usuários. Abaixo, detalhamos cada uma dessas medidas e justificamos sua relevância:

- Inicialização em modo ”lazy”: A aplicação Spring teve sua inicialização configurada para o modo ”lazy”, o que significa que apenas os componentes e dependências estritamente necessários são carregados durante o processo de inicialização. Essa abordagem reduz significativamente o tempo de inicialização da aplicação, uma vez que evita o carregamento desnecessário de recursos que podem não ser utilizados durante a execução da aplicação. Dessa forma, os recursos do sistema são melhor aproveitados, proporcionando uma inicialização mais rápida e liberando recursos para outras tarefas importantes.
- Exclusão de auto-configurações: As configurações automáticas do Spring podem ser convenientes, mas podem consumir recursos desnecessários, especialmente em aplicações que não precisam de todas as configurações padrão oferecidas pelo framework. Desabilitar essas auto-configurações permite que a aplicação seja mais focada e específica, evitando o consumo de recursos que não são relevantes para o cenário em questão. Isso resulta em um ganho de desempenho e também simplifica o código e a manutenção da aplicação.
- Troca do Container Servlet padrão do Spring: A migração do Tomcat para o Undetow foi realizada após uma avaliação comparativa de desempenho (SMITH; JOHNSON, 2022a), que demonstrou o Undetow como uma opção mais eficiente para aplicações Spring. O Undetow é um servidor web leve e de alto desempenho, especialmente adequado para aplicações que exigem uma alta taxa de transferência de dados e conexões simultâneas. Com essa troca, foi possível melhorar o desempenho geral da aplicação, tornando-a mais ágil e responsiva.
- Desligamento do Java Management Extensions (JMX): A flag de monitoramento de beans em tempo real do JMX foi desabilitada, considerando que a aplicação já faz uso de outras ferramentas de métricas para monitoramento e análise de desempenho. Ao desativar o JMX, evita-se o consumo de recursos extras que seriam necessários para o monitoramento em tempo real, proporcionando uma utilização mais eficiente dos recursos disponíveis.

- Retirada do sistema de log padrão do Hibernate e do Java Persistence API (JPA): Os logs do banco de dados gerados pelo Hibernate e JPA podem ser úteis durante o desenvolvimento e depuração, mas podem afetar o desempenho da aplicação em produção. Ao desligar os logs de banco padrão e criar logs de forma controlada e seletiva, é possível evitar o custo computacional associado ao registro de todas as consultas e operações no banco de dados, tornando o processamento mais rápido e eficiente.

- Geração de índices de forma sequencial: A criação de índices de forma sequencial foi adotada como uma estratégia de otimização em termos de desempenho. Índices sequenciais oferecem vantagens em relação à eficiência de armazenamento, ao melhor aproveitamento de caches e à redução de fragmentação. Além disso, ao criar índices sequenciais em colunas frequentemente utilizadas em operações de consulta e ordenação, é possível melhorar significativamente o desempenho das consultas, resultando em respostas mais rápidas e maior escalabilidade do sistema.

Com a implementação dessas melhorias, a aplicação foi submetida a testes mais eficientes e apresentou resultados promissores em termos de desempenho e escalabilidade. As medidas adotadas permitem que a aplicação atenda às demandas crescentes dos usuários, garantindo uma experiência mais fluida e uma entrega contínua das funcionalidades oferecidas pelo sistema. Essa abordagem pró-ativa de otimização e aprimoramento contínuo reflete o compromisso em fornecer uma aplicação robusta, eficiente e pronta para enfrentar os desafios futuros.

5.3.5 Observabilidade

A integração das tecnologias de observabilidade - Spring Boot Actuator, Prometheus, Grafana e SLF4J - foi realizada de forma sistemática e cuidadosa para garantir a coleta adequada de métricas, logs e informações relevantes sobre a aplicação. Abaixo como cada uma das tecnologias foi configurada na aplicação:

O Spring Boot Actuator está sendo utilizado para expor endpoints que fornecem informações operacionais sobre a aplicação em tempo de execução. Esses endpoints são acessíveis através de URLs específicas e permitem monitorar a saúde da aplicação, informações sobre as métricas, status do cache, detalhes dos endpoints REST, entre outros. No código da aplicação, não há nenhuma configuração explícita para o Spring Boot Actuator, uma vez que ele é habilitado automaticamente quando a dependência é adicionada ao projeto. Através desses endpoints, é possível verificar se a aplicação está funcionando corretamente, a quantidade de memória usada, informações sobre a JVM, entre outros dados operacionais.

O Prometheus é uma ferramenta de monitoramento que coleta, armazena e disponibiliza métricas em série temporais. Nesse contexto, a aplicação possui uma integração com o Prometheus através da biblioteca correspondente que é adicionada como dependência no projeto. Além disso, no arquivo de configuração do Prometheus (prometheus.yml), são definidos os "jobs de scraping" para coleta das métricas dos endpoints do Spring Boot Actuator. Essas métricas coletadas pelo Prometheus permitem acompanhar o desempenho da aplicação ao longo do tempo e identificar possíveis gargalos ou problemas.

O Grafana é uma plataforma de análise e visualização de dados, que pode ser conectada a diversas fontes de dados, incluindo o Prometheus. Nessa aplicação, o Grafana está integrado ao Prometheus como uma fonte de dados. Isso permite que as métricas coletadas pelo Prometheus sejam utilizadas para criar dashboards e gráficos personalizados.

O SLF4J (Simple Logging Facade for Java) é uma fachada de logging que permite que a aplicação utilize diferentes bibliotecas de logging sem modificar o código-fonte. Nessa aplicação, o SLF4J está sendo utilizado para criar arquivos de log e registrar informações relevantes durante a execução da aplicação. Através do SLF4J, é possível controlar o nível de log (debug, info, warn, error) e configurar o formato dos logs. Também foram implementadas regras para compressão de logs antigos, a fim de gerenciar o espaço em disco e manter a integridade dos registros.

Desta forma, a aplicação integrou as tecnologias de observabilidade para obter uma visão detalhada e abrangente de seu desempenho e operação em produção. O Spring Boot Actuator fornece endpoints para monitorar a saúde e métricas da aplicação, o Prometheus coleta e armazena essas métricas, o Grafana é utilizado para criar painéis e gráficos personalizados com base nas métricas coletadas, e o SLF4J registra logs com informações relevantes sobre a execução da aplicação. Essas tecnologias em conjunto oferecem uma solução poderosa para a observabilidade, permitindo uma melhor compreensão e monitoramento contínuo da aplicação em ambientes de produção.

5.4 Aplicação Front-End

Nesta seção é oferecido uma visão mais aprofundada sobre a aplicação, abordando a sua implementação e o resultado final alcançado. São destacados aspectos importantes, como as páginas desenvolvidas, o sistema de segurança implementado e o processo de registro de Alertas.

Para a visualização e administração dos dados da solução foi desenvolvida uma aplicação Front-End, que compreende diversas páginas, incluindo a página de Alertas, de atualização de blacklists e outras páginas auxiliares. Essa interface de usuário proporciona uma experiência amigável e intuitiva para os usuários interagirem com o sistema, permitindo uma gestão eficiente dos dados e funcionalidades da solução.

A aplicação Front-End possui um sistema de login separado do API Gateway, com um mecanismo de Autenticação que atribui funções específicas a cada tipo de usuário. Esse sistema foi projetado para garantir a segurança e o controle de acesso adequado às funcionalidades da aplicação. Dessa forma, os administradores têm acesso privilegiado, permitindo-lhes visualizar e gerenciar todos os Alertas, enquanto os usuários comuns têm permissões restritas, permitindo-lhes apenas visualizar seus próprios Alertas e registrá-los conforme necessário.

Essa abordagem de autenticação baseada em funções contribui para uma segregação adequada de responsabilidades, garantindo que as informações confidenciais e as funcionalidades críticas sejam acessíveis apenas aos usuários autorizados. Além disso, a aplicação Front-End possui uma interface de fácil utilização, que facilita a navegação e a interação dos usuários com as funcionalidades disponíveis.

Ao longo do processo de desenvolvimento, foram adotadas práticas ágeis e metodologias adequadas para atender aos requisitos específicos da aplicação. As etapas de desenvolvimento incluíram o levantamento de requisitos, a definição da arquitetura, o design da interface, a implementação das funcionalidades e a realização de testes exaustivos para garantir a qualidade do software entregue.

O desenvolvimento da aplicação Front-End foi feito em sincronia com o desenvolvimento das funcionalidades do backend, garantindo assim uma integração fluida e uma experiência consistente para os usuários. Foram realizadas iterações e revisões contínuas para aprimorar e ajustar a aplicação de acordo com o feedback dos usuários e as necessidades do negócio.

No contexto da aplicação Front-End, o design responsivo foi priorizado, permitindo que a aplicação se adapte de forma eficiente a diferentes dispositivos e tamanhos de tela. Isso proporciona uma experiência de usuário consistente e agradável, independentemente do dispositivo que está sendo utilizado.

5.4.1 Páginas desenvolvidas

A tela de login da aplicação Front-end é apresentada na Figura 15, com formulário de login com dois campos de texto, o usuário e a senha.

Após o processo de autenticação bem-sucedido, quando o usuário possui privilégios de administrador, ele é direcionado automaticamente para a página principal do sistema, denominada ”Home”. Nesta página, todos os alertas estão dispostos em formato paginado, proporcionando uma visão ordenada e acessível dos dados (Figura 16). Cada alerta exibido contém informações detalhadas capturadas durante sua geração. Adicionalmente, os administradores têm acesso a um botão de remoção associado a cada alerta, possibilitando a exclusão de alertas específicos conforme necessário. Essa funcionalidade garante maior

Figura 15 – Tela de Login

Fonte: Do autor, 2023

controle e gerenciamento sobre os alertas registrados no sistema.

A página ”Home” representa um ponto central para os administradores acompanharem as atividades e informações sobre os alertas gerados pela aplicação. A formatação paginada proporciona uma visualização organizada, evitando a sobrecarga de informações em uma única tela. Além disso, a presença do botão de remoção em cada alerta facilita a administração do sistema, permitindo que os administradores respondam prontamente a situações específicas ou excluam alertas irrelevantes ou duplicados.

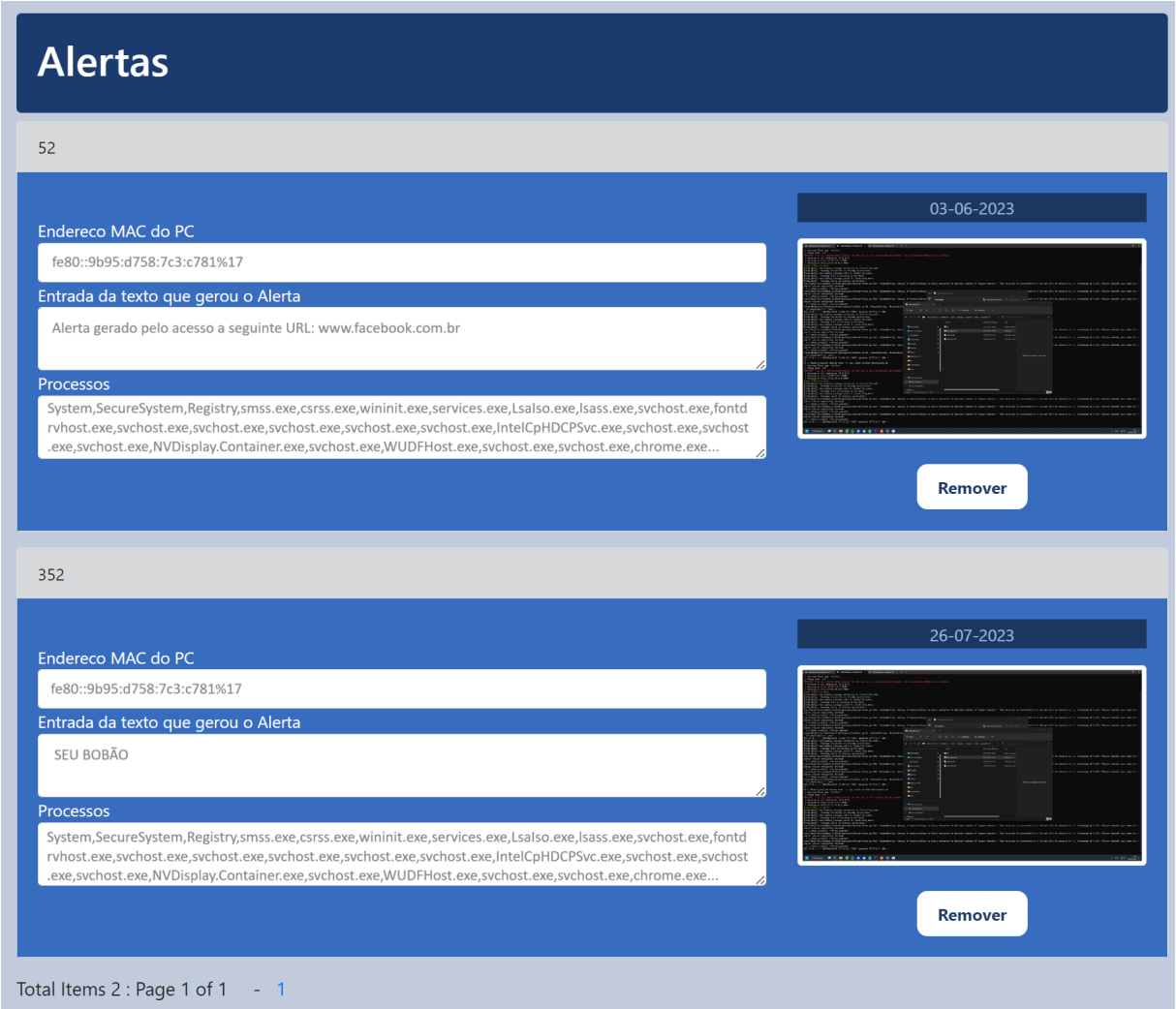
Outra funcionalidade relevante implementada na aplicação é a página de busca (apenas para administradores), ilustrada na Figura 17. Nessa página, os usuários têm a possibilidade de realizar pesquisas específicas de alertas com base no endereço MAC dos computadores associados aos alertas gerados.

O formulário de busca, disponível na página, permite que os usuários insiram o endereço MAC desejado para encontrar alertas correspondentes. Ao submeter o formulário, a aplicação processa a consulta e exibe os resultados de forma paginada logo abaixo do formulário de busca. Essa abordagem de paginação torna mais fácil e conveniente para os usuários navegarem pelos resultados da busca, evitando a exibição de um grande volume de informações em uma única tela.

A implementação dessa página é resultado de uma abordagem centrada no usuário, visando facilitar a localização e acesso às informações relevantes, de forma a otimizar o tempo e esforço dedicados à administração da aplicação. Ao disponibilizar um formulário intuitivo de busca e os resultados paginados, a aplicação oferece aos usuários uma experiência mais fluida e eficiente, garantindo maior eficácia nas operações de busca e acesso aos dados de interesse.

Quando um usuário não possui privilégios de administrador, ele é redirecionado para uma página específica, ilustrada na Figura 18. Nessa página, é apresentado um

Figura 16 – Home do Administrador



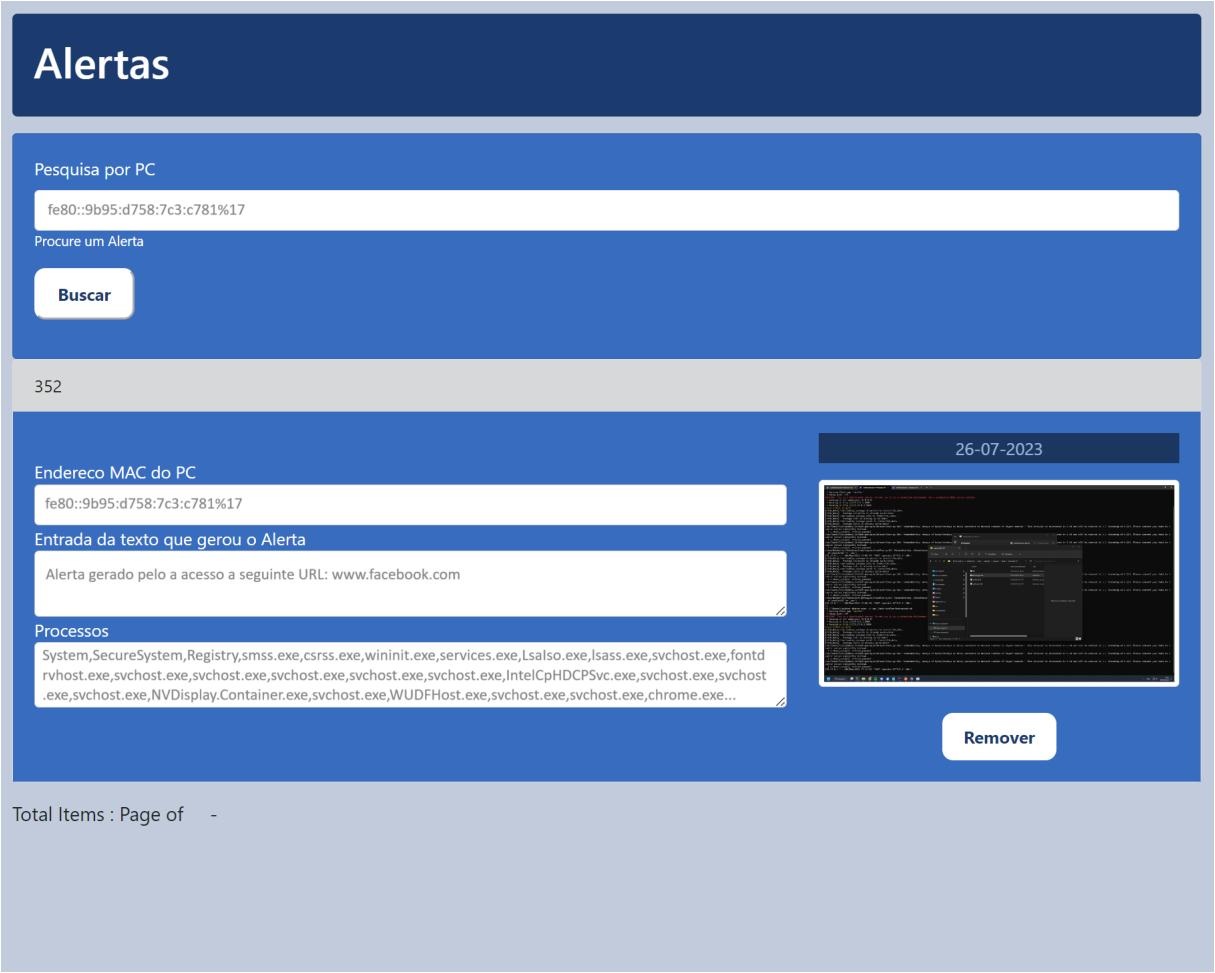
Fonte: Do autor, 2023

formulário e uma listagem de alertas. O formulário tem como finalidade validar um alerta e é composto por três campos: um campo para inserir o ID do alerta, outro para o ID do PC (endereço MAC) associado e um terceiro para inserir o hash.

O hash é obtido quando o alerta é registrado através do botão disponível em cada item da listagem. Ao acionar o botão ”Registrar Alerta”, é gerado um arquivo JSON que contém todas as informações do alerta, incluindo o hash. Esse hash é criado a partir das informações do alerta e serve como uma identificação única, garantindo a integridade dos dados e possibilitando validações futuras.

Ao fornecer os campos necessários no formulário, o usuário pode realizar a validação do alerta com base nas informações previamente registradas no sistema. Dessa forma, a aplicação assegura que apenas alertas legítimos e corretamente registrados sejam validados, evitando a manipulação indevida de dados sensíveis.

Figura 17 – Página de Busca



Fonte: Do autor, 2023

Essa abordagem oferece uma maneira confiável e segura para que os usuários não administradores possam validar os alertas de forma eficiente. O uso do hash, que é obtido a partir das informações do alerta no momento do registro, permite a verificação posterior da autenticidade do alerta. Além disso, a disponibilização do arquivo JSON com todas as informações relevantes do alerta facilita o processo de validação, fornecendo um registro detalhado do evento capturado.

As páginas apresentadas a seguir estão disponíveis apenas para administradores. A Figura 19 apresenta o menu do cabeçalho da aplicação.

A Figura 20 apresenta a página ”BadLanguage”, que desempenha o papel de gerenciar uma lista de palavras consideradas inadequadas, atuando como uma espécie de blacklist de vocabulário. Essa página possui um formulário com um campo destinado a inserir novas palavras, além de uma lista de palavras já existentes, as quais podem ser removidas individualmente através do botão ”Remover”.

O formulário de inserção de novas palavras permite que o usuário adicione novos

Figura 18 – Home de Usuário

Fonte: Do autor, 2023

Figura 19 – Menu

Fonte: Do autor, 2023

termos à lista de palavras indesejadas. Essa funcionalidade é útil para manter a blacklist atualizada, permitindo que a aplicação identifique e lide adequadamente com conteúdos inapropriados.

Já a listagem de palavras apresenta de forma clara e organizada todas as palavras que compõem a blacklist. Essa lista é exibida em formato tabular, possibilitando a visualização rápida e fácil das palavras já registradas. Além disso, cada item da lista possui um botão ”Remover”, o que oferece a opção de excluir palavras individualmente, caso necessário.

Essa abordagem de gerenciamento de palavras ”ruins” na página ”BadLanguage”

proporciona uma forma conveniente e eficiente para que os administradores possam manter controle sobre o vocabulário indesejado na aplicação. A adição de novas palavras permite atualizações contínuas da blacklist, enquanto a possibilidade de remover palavras individualmente possibilita uma gestão flexível e precisa da lista.

Figura 20 – Página de BadLanguage

Fonte: Do autor, 2023

A página ”MaliciousPort” desempenha um papel crucial no gerenciamento dos banners de aplicações que não podem ser executadas em portas abertas no computador, com o objetivo de evitar possíveis vulnerabilidades. Essa blacklist foi especialmente concebida para fornecer suporte ao Scanner, garantindo a segurança do sistema.

Como evidenciado na Figura 21, a estrutura da página ”MaliciousPort” é semelhante à página ”BadLanguage”, oferecendo um formulário para adição de novos banners e uma lista logo abaixo, exibindo todas as informações pertinentes sobre as ”MaliciousPorts” cadastradas. Cada item da lista é acompanhado por um botão ”Remover”, permitindo a exclusão de cada banner individualmente, caso seja necessário.

O formulário de adição de novas ”MaliciousPorts” possibilita a inserção segura e eficiente de banners que não devem ser permitidos em portas abertas do computador. Essa funcionalidade é de extrema importância para manter a integridade e a segurança do sistema, uma vez que evita potenciais vulnerabilidades que poderiam ser exploradas por invasores.

A lista de ”MaliciousPorts” exibida de forma organizada e clara fornece uma visão geral dos banners registrados, tornando a visualização e o gerenciamento dos itens mais simples e acessíveis aos administradores.

Figura 21 – Página de MaliciousPort

Fonte: Do autor, 2023

A página de ”MaliciousProcess” desempenha um papel essencial ao permitir o gerenciamento da blacklist de processos maliciosos, fornecendo aos administradores uma ferramenta poderosa para manter a segurança da aplicação. Como ilustrado na Figura 22, a estrutura da página segue o mesmo modelo das outras páginas de administração de blacklist, visando a consistência e facilidade de uso.

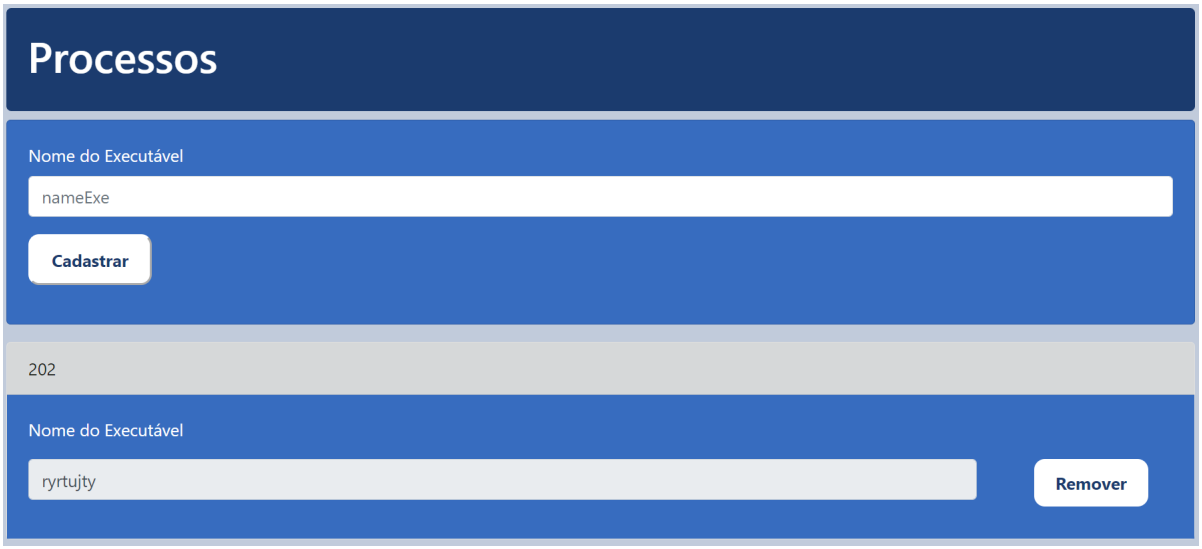
Ao acessar a página de ”MaliciousProcess”, os administradores são recebidos por um formulário que possibilita a adição de novos processos maliciosos à blacklist. Esse formulário foi projetado com a finalidade de facilitar o registro seguro e eficiente de processos indesejados, auxiliando na proteção da aplicação contra ameaças potenciais.

A listagem de processos maliciosos exibida na página fornece uma visão completa dos itens já cadastrados, oferecendo aos administradores a capacidade de visualizar e gerenciar de forma organizada os processos que foram identificados como maliciosos. A presença de um botão de remoção em cada item da lista garante que os administradores possam excluir processos individualmente, quando necessário, proporcionando um controle refinado sobre a blacklist.

Ao seguir o mesmo modelo das outras páginas de administração de blacklist, a página de ”MaliciousProcess” promove uma experiência coesa e consistente em todo o sistema, tornando-se uma ferramenta amigável e de fácil navegação para os administradores.

A última página de blacklist, exemplificada na Figura 23, desempenha o papel de gerenciamento da lista de ”MaliciousWebsites”. Em conformidade com as outras páginas de

Figura 22 – Página de MaliciousProcess



Fonte: Do autor, 2023

blacklists, essa página segue a mesma estrutura bem-sucedida, fornecendo um formulário para a adição de novos websites maliciosos, uma listagem detalhada desses websites e um botão de remoção para permitir a exclusão individual de itens.

Ao acessar a página de ”MaliciousWebsites”, os administradores são recebidos por um formulário intuitivo, projetado para facilitar a adição segura e ágil de websites maliciosos à blacklist. Essa funcionalidade é essencial para reforçar a proteção da aplicação contra websites perigosos e garantir um ambiente virtual seguro e confiável.

A listagem dos ”MaliciousWebsites” exibida na página oferece uma visão dos websites já cadastrados, permitindo que os administradores visualizem rapidamente todas as informações relevantes. Com um botão de remoção em cada item da lista, os administradores têm o controle total para excluir websites indesejados da blacklist, garantindo uma administração precisa e eficiente.

Por fim, a página de ”Cadastro” oferece aos administradores a capacidade de adicionar novos usuários à aplicação. A Figura 24 ilustra a interface da página, que foi projetada de forma intuitiva e eficiente. Ao acessar essa página, os administradores são apresentados a um formulário simples, composto por dois campos de texto (Usuário e Senha) e um campo de seleção de função, oferecendo as opções ”Admin” e ”User”.

Esse formulário simplificado facilita o processo de cadastro de novos usuários, garantindo uma experiência descomplicada e ágil. Os administradores têm a flexibilidade de inserir o nome do usuário desejado no campo ”Usuário” e escolher uma senha adequada no campo ”Senha”. Ademais, o campo de seleção ”Função” permite aos administradores atribuir uma função específica para cada usuário adicionado, tornando possível estabelecer diferentes níveis de acesso e permissões de acordo com as necessidades do sistema.

Figura 23 – Página de MaliciousWebsites

Fonte: Do autor, 2023

Essa página de "Cadastro" é fundamental para a administração de usuários da aplicação, permitindo o crescimento controlado da base de usuários e a gestão das funções atribuídas a cada um. A simplicidade e clareza do formulário possibilitam um processo de cadastro eficiente, assegurando que novos usuários sejam incorporados à aplicação de maneira segura e organizada.

Figura 24 – Página de Cadastro

Fonte: Do autor, 2023

5.4.2 Autorização e Autenticação

Como já mencionado, a aplicação apresenta um sólido sistema de login e cadastro com diferentes funções de usuário, como Administrador e Usuário Comum. Esse sistema

de autenticação e autorização foi implementado utilizando o Spring Security, que é uma poderosa estrutura de segurança para aplicações Java. O Spring Security oferece diversas funcionalidades para proteger endpoints, autenticar usuários e gerenciar suas permissões de acesso.

A segurança da aplicação é configurada através da classe WebSecurityConfig, que é uma classe de configuração do Spring Security e estende WebSecurityConfigurerAdapter. Essa classe contém uma série de configurações que definem as regras de acesso aos diferentes endpoints da aplicação.

No método configure(HttpSecurity http), são definidas as regras de autorização para cada endpoint. Por exemplo, os endpoints /home, /badLanguage, /maliciousPort, /maliciousProcess e /maliciousWebsite só podem ser acessados por usuários com a função de Administrador (hasRole("admin")). Todos os outros endpoints exigem autenticação, ou seja, o usuário deve estar logado para acessá-los (anyRequest().authenticated()).

Além disso, o método configure(HttpSecurity http) também define as configurações para o formulário de login. Quando um usuário tenta acessar uma página protegida, o Spring Security redireciona automaticamente para a página de login (loginPage("/login")). Em caso de falha na autenticação, o usuário é redirecionado para uma página de erro (failureForwardUrl("/login-error")). Se o login for bem-sucedido, o usuário é redirecionado para uma página de sucesso (defaultSuccessUrl("/login-success")).

A classe WebSecurityConfig também faz uso do UserDetailsService, que é uma interface que permite o carregamento personalizado dos detalhes do usuário. No método configureGlobal(AuthenticationManagerBuilder auth), o UserDetailsService é injetado e configurado juntamente com o PasswordEncoder para autenticar os usuários. Nesse caso, é utilizado o BCryptPasswordEncoder, uma implementação do Spring Security que realiza o hash seguro das senhas dos usuários.

Desta forma, a classe WebSecurityConfig desempenha um papel essencial na configuração da segurança da aplicação. Ela define as regras de autorização para cada endpoint, garante que os usuários estejam autenticados corretamente e gerencia suas permissões de acesso. Essa implementação robusta do Spring Security fortalece a proteção da aplicação contra acesso não autorizado e garante que cada usuário possua os devidos privilégios, mantendo assim a integridade e segurança do sistema como um todo.

5.4.3 Registro de Alertas

No contexto de garantir a segurança e integridade dos alertas gerados pela aplicação, foi implementada uma funcionalidade especial destinada aos usuários comuns. A funcionalidade que permite que o usuário comum salve um alerta gerado para garantir sua própria segurança foi cuidadosamente implementada na aplicação. Para acessá-la, o usuário

comum pode visualizar seus alertas na página designada. Nessa página, ele encontra um formulário especialmente desenvolvido para a validação de alertas registrados, bem como um botão de ”Registro de Alertas” associado a cada alerta exibido.

Quando o usuário comum decide registrar um alerta, basta clicar no respectivo botão, que imediatamente desencadeia o processo de obtenção das informações do alerta. Esse processo resulta na geração de um arquivo JSON contendo todos os detalhes relevantes do alerta. Além disso, um hash é criado com base nessas informações, e ambos são disponibilizados para o usuário em conjunto. A estratégia de geração para o hash, demonstrada na função da Figura 25, é baseada no uso do SHA-256 e a conversão das informações do alerta para hexadecimal.

Figura 25 – Função de geração de Hash

```
98      public String generateHash(Long id, String pcId) {
99          try {
100             String input = id + pcId;
101             MessageDigest digest = MessageDigest.getInstance("SHA-256");
102             byte[] encodedHash = digest.digest(input.getBytes(StandardCharsets.UTF_8));
103
104             StringBuilder hexString = new StringBuilder();
105             for (byte b : encodedHash) {
106                 String hex = Integer.toHexString(0xff & b);
107                 if (hex.length() == 1) {
108                     hexString.append('0');
109                 }
110                 hexString.append(hex);
111             }
112
113             return hexString.toString();
114         } catch (NoSuchAlgorithmException e) {
115             e.printStackTrace();
116             return null;
117         }
118     }
```

Esse arquivo JSON e o hash associado têm um propósito específico e importante: permitir que o usuário comum armazene informações detalhadas sobre o alerta de forma segura e confiável, de modo que ele possa apresentá-las como prova ou evidência, se necessário. O hash serve como uma espécie de ”impressão digital” do alerta, garantindo sua integridade e evitando qualquer modificação ou adulteração posterior. Com a combinação do arquivo JSON e do hash, o usuário possui um registro autenticado e imutável do alerta, adequado para futura referência e uso em situações que exijam a comprovação dos eventos ocorridos. Essa funcionalidade adiciona uma camada de segurança adicional e proporciona aos usuários comuns uma maneira confiável de proteger suas informações e provas importantes.

5.5 Modelos de Predição

O desenvolvimento da solução também inclui um módulo fundamental, responsável por utilizar modelos de predição para detectar conteúdo ofensivo em três idiomas diferentes. Ao todo, foram treinados nove modelos, sendo três para cada idioma específico. A seguir é detalhado o processo de desenvolvimento dessa parte da aplicação, desde a seleção dos datasets utilizados até o treinamento efetivo dos modelos de predição. Esses modelos são de extrema importância para garantir a eficácia e precisão na detecção de conteúdo odioso, contribuindo significativamente para a qualidade geral da solução desenvolvida.

5.5.1 Datasets utilizados

O processo de desenvolvimento dos modelos de predição iniciou-se com a seleção cuidadosa dos conjuntos de dados a serem utilizados no treinamento. Para garantir a eficácia e precisão dos modelos, foram escolhidos dois datasets para cada idioma, totalizando nove conjuntos de dados. Em português, foram selecionados os conjuntos de dados (FITITNT, 2019) e (KANSAON, 2018), ambos disponíveis no GitHub.

Para o idioma espanhol, optou-se pelos datasets (FERSINI P. ROSSO, 2018) e (ÁLVAREZ-CARMONA et al., 2018). Já em inglês, foram selecionados os conjuntos de dados (DADVAR et al., 2012) e (WASEEM, 2016).

Essa seleção foi baseada na disponibilidade de um grande número de frases e no fato de terem sido capturados de ambientes reais, como Twitter, Facebook e Youtube, o que torna os dados mais representativos e relevantes para o contexto da solução. Cada conjunto de dados passou por um processo de tratamento, utilizando estratégias específicas, que serão detalhadas na próxima seção, visando otimizar a qualidade e a confiabilidade dos modelos de predição desenvolvidos.

5.5.2 Tratamento dos dados

Para preparar os dados para a etapa de treinamento dos modelos de predição, foram aplicadas técnicas de normalização e vetorização dos textos presentes nos datasets. O objetivo dessa etapa é garantir que os dados estejam em um formato adequado para o processamento e análise pelos algoritmos de aprendizado de máquina.

A normalização dos textos foi realizada pela classe TextProcessor, que utiliza a biblioteca NLTK (Natural Language Toolkit) para o processamento de linguagem natural. Inicialmente, foram realizados procedimentos como a remoção de caracteres especiais, links, menções a usuários e outras informações que não são relevantes para o treinamento dos modelos.

Em seguida, o texto foi convertido para letras minúsculas para padronização. Além

disso, as palavras foram reduzidas para suas formas raiz, utilizando o processo de stemming, realizado pelo método stemmer da classe, que aplica o algoritmo RSLPStemmer para extrair a forma raiz das palavras. Essa etapa é importante para reduzir as diferentes formas flexionadas de uma mesma palavra, permitindo que o algoritmo de aprendizado de máquina identifique melhor padrões e características relevantes.

Após a normalização dos textos, foi realizada a vetorização dos dados para que fossem representados numericamente, uma exigência para a maioria dos algoritmos de aprendizado de máquina. A função vectorizeText da classe TextProcessor realiza esse processo, criando um vetor para cada texto contendo a frequência de ocorrência das palavras presentes no conjunto de dados.

O processo de vetorização é realizado da seguinte forma: primeiro, é criado um conjunto de palavras únicas presentes em todos os textos. Cada palavra é então mapeada para um índice numérico único, criando um tradutor (translater) que associa cada palavra ao seu índice no vetor. Em seguida, o texto é transformado em um vetor de zeros, e a frequência de ocorrência de cada palavra é contabilizada. Se a palavra estiver presente no conjunto de palavras do tradutor, o valor correspondente no vetor é incrementado.

Assim, cada texto é representado como um vetor de tamanho fixo contendo a contagem de ocorrência das palavras presentes no texto. Essa representação numérica permite que os modelos de predição possam trabalhar com os dados de forma mais eficiente, identificando padrões e relações entre as palavras.

Por fim, após o pré-processamento, os datasets foram devidamente tratados e organizados, garantindo a qualidade e a integridade dos dados utilizados no treinamento dos modelos de predição.

5.5.3 Escolha dos modelos

Após a etapa de pré-processamento dos dados, onde foram aplicadas diversas técnicas para limpeza e tratamento dos textos, os dados estavam prontos para o treinamento dos modelos de predição. Foram selecionados nove modelos no total, três para cada idioma: português, espanhol e inglês.

Para o treinamento dos modelos, foram utilizadas três técnicas amplamente reconhecidas na área de aprendizado de máquina: Regressão Logística, Support Vector Machine (SVM) e Multinomial Naive Bayes. A escolha dessas técnicas foi embasada em sua eficácia comprovada na classificação de textos, bem como em trabalhos anteriores que mostraram resultados promissores utilizando esses modelos para detecção de ódio em textos (ZULQARNAIN et al., 2020).

A Regressão Logística é um modelo de aprendizado supervisionado que visa prever probabilidades associadas a um conjunto de classes. Neste contexto, as classes são as

categorias que representam a presença ou ausência de ódio em um determinado texto. A regressão logística é especialmente útil quando se trabalha com problemas de classificação multiclasse, como é o caso da detecção de ódio em textos em múltiplos idiomas. Essa técnica é capaz de modelar as relações entre as palavras presentes nos textos e suas respectivas classes, identificando padrões que permitam a classificação correta dos textos (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Já o SVM é um algoritmo de aprendizado supervisionado que tem como objetivo encontrar um hiperplano que melhor separa as amostras das diferentes classes em um espaço de características. Ele é particularmente útil quando os dados não são linearmente separáveis, o que é comum em tarefas de classificação de textos. No contexto da detecção de ódio em textos, o SVM é capaz de encontrar hiperplanos que separam efetivamente os textos em diferentes classes, permitindo a identificação de padrões complexos associados à presença ou ausência de ódio nos textos (CORTES; VAPNIK, 1995).

O Multinomial Naive Bayes é um modelo probabilístico simples, mas amplamente utilizado na classificação de textos. Ele se baseia no Teorema de Bayes e assume a independência entre as características (palavras) do texto. Embora essa suposição nem sempre seja verdadeira em contextos reais, o Naive Bayes ainda é bastante eficaz na classificação de textos devido à sua simplicidade e eficiência computacional. Ele é capaz de capturar a frequência e distribuição das palavras em diferentes classes de texto, permitindo uma classificação precisa mesmo em conjuntos de dados grandes (BROWN; AL., 1992).

Para cada idioma, os três modelos foram treinados em conjuntos de dados independentes, compreendendo os textos pré-processados e as respectivas classes de ódio. Durante o treinamento, os modelos aprenderam a relação entre as palavras e suas respectivas classes, ajustando seus parâmetros para otimizar a performance na classificação dos textos.

A escolha de treinar múltiplos modelos para cada idioma se baseia na ideia de que diferentes técnicas podem se comportar de forma mais eficaz em diferentes contextos. Ao combinar as previsões de múltiplos modelos, é possível obter um sistema de detecção de ódio mais robusto e preciso, minimizando possíveis viéses e melhorando o desempenho geral do sistema.

Por conseguinte, os modelos de predição escolhidos têm se mostrado eficazes na classificação de textos, especialmente na detecção de ódio em diferentes idiomas. Ao utilizar essas técnicas de aprendizado supervisionado e treinar múltiplos modelos, a solução busca oferecer um sistema de detecção de ódio preciso e confiável, capaz de lidar com a complexidade e variação linguística inerentes aos textos em diferentes línguas. As etapas de pré-processamento e treinamento foram fundamentais para criar uma base sólida que permitisse a construção de um sistema eficaz na identificação e categorização de textos odiosos.

5.5.4 Treinamento dos modelos

Para elevar a precisão dos modelos de predição, foram aplicadas diversas técnicas e estratégias utilizando a biblioteca Scikit-learn. Inicialmente, a API realiza a determinação do idioma do texto submetido, direcionando-o para os modelos adequados para cada língua. Em seguida, foram definidos os parâmetros específicos para cada modelo a fim de otimizar sua performance.

No caso do modelo de Regressão Logística, escolheu-se o método L1 (Lasso) como penalidade. Essa escolha é especialmente útil quando há muitas características irrelevantes ou redundantes no conjunto de dados, pois o Lasso tende a zerar os coeficientes das características menos importantes, proporcionando uma forma de seleção automática de atributos relevantes. Além disso, foi adicionada a soma dos valores absolutos dos coeficientes à função de perda, o que restringe o tamanho dos coeficientes e ajuda a evitar overfitting.

O parâmetro de inversa da força de regularização (C) foi definido como 1.2. O valor de C controla a quantidade de regularização aplicada aos coeficientes do modelo. Valores menores de C fornecem maior regularização, tornando o modelo mais simples e menos propenso ao overfitting, enquanto valores maiores permitem que o modelo se ajuste melhor aos dados de treinamento. A escolha de C foi feita após experimentação e validação cruzada, buscando encontrar um valor que equilibre a capacidade do modelo de generalizar para novos dados e sua performance nos dados de treinamento.

Para o solver, optou-se pelo algoritmo de otimização estocástica "saga". Esse solver é uma extensão do algoritmo de gradiente descendente estocástico, que oferece uma convergência mais rápida para conjuntos de dados grandes e muitas vezes é mais eficiente em problemas de otimização não convexos, como é o caso da Regressão Logística com penalidade L1. A escolha do solver também se mostrou eficaz na obtenção de um modelo com boa capacidade de generalização e baixo overfitting.

No caso do modelo de Support Vector Machine (SVM), o kernel "rbf" (função de base radial) foi empregado. O kernel rbf é frequentemente utilizado em problemas de classificação quando os dados não são linearmente separáveis. Ele mapeia os dados para um espaço dimensional superior onde se tornam linearmente separáveis, permitindo a construção de um hiperplano que separa as diferentes classes de texto de forma mais eficiente. Essa escolha se mostrou adequada para lidar com a complexidade inerente aos dados textuais.

Para o parâmetro de inversa da força de regularização (C) no SVM, foi escolhido o valor padrão de 1.0. Esse valor foi selecionado após experimentação e validação cruzada, e é um valor comumente utilizado em problemas de classificação com SVM. Assim como na Regressão Logística, o parâmetro C controla a quantidade de regularização aplicada

ao modelo, e sua escolha adequada é importante para garantir um equilíbrio entre a capacidade de generalização e a performance nos dados de treinamento.

O parâmetro gamma foi definido como ”scale”, que é calculado como 1 / (número de características * variância dos dados de treinamento). O gamma controla a influência de cada amostra de treinamento no ajuste do modelo. Valores menores de gamma fazem com que o modelo considere uma área maior ao redor de cada amostra no espaço de características, tornando a fronteira de decisão mais suave e resultando em um modelo mais flexível. Por outro lado, valores maiores de gamma fazem com que o modelo considere apenas amostras próximas no espaço de características, tornando a fronteira de decisão mais ajustada aos dados de treinamento e podendo levar a overfitting.

Já no modelo Multinomial Naive Bayes, atribuiu-se o valor 1 ao parâmetro de suavização Laplace (alpha). Esse valor permite que o modelo lide melhor com casos de palavras novas ou raras que podem não estar presentes nos dados de treinamento. A suavização Laplace evita a probabilidade zero para essas palavras e garante que o modelo possa fazer previsões mesmo para palavras não vistas durante o treinamento. O valor de alpha também foi escolhido após experimentação e validação cruzada para encontrar uma configuração que equilibre a capacidade do modelo de generalizar e evitar overfitting.

Por fim, para o parâmetro fitprior, utilizou-se a base de dados de treinamento. O fitprior é um parâmetro que controla a forma como as probabilidades a priori de cada classe são estimadas. Quando fitprior é definido como True, o modelo utiliza as frequências das classes no conjunto de treinamento para calcular as probabilidades a priori. Essa abordagem é útil quando temos desbalanceamento nas classes, garantindo que as probabilidades a priori reflitam a distribuição real dos dados. No caso do fitprior ser False, o modelo assume probabilidades a priori uniformes para todas as classes.

As estratégias utilizadas na configuração dos modelos de predição visam aprimorar sua precisão e capacidade de generalização em tarefas de detecção de ódio em textos. A escolha cuidadosa dos parâmetros e técnicas é fundamental para evitar problemas como overfitting e underfitting, garantindo que os modelos sejam eficazes na classificação de textos em diferentes línguas e capazes de identificar padrões complexos e nuances semânticas. A combinação de modelos de Regressão Logística, Support Vector Machine e Multinomial Naive Bayes permite uma abordagem abrangente e eficiente para a detecção de ódio em textos de diferentes idiomas, proporcionando uma solução robusta e precisa para a API de análise de textos.

5.6 Fluxos

Para proporcionar uma visão mais clara dos fluxos desenvolvidos para a solução, foram criados diagramas de atividades. Esses diagramas englobam os diferentes módulos

da aplicação e ilustram as ações dos usuários e/ou da própria solução.

Um dos diagramas é exemplificado pela Figura 26, que representa o fluxo de geração de alertas na aplicação. O processo tem início com a captura de um comportamento, a partir do qual o Spyware se divide em cinco processos distintos: dois para o Keylogger, um para o Sniffer, um para o Scanner e outro para o ProcessLogger. Todos esses processos são analisados para detectar possíveis inconsistências, e se uma for encontrada, a geração de um alerta é iniciada.

Adicionalmente, o Keylogger possui um segundo processo assíncrono que coleta cada sentença individualmente (separadas por quebra de linha) e as envia para o API Gateway do Spyware. Nesse momento, o API Gateway testa se a frase contém discurso de ódio, e se houver um resultado positivo, o fluxo de geração de alerta é novamente acionado.

No início do processo de geração de alerta, ocorrem duas requisições enviadas ao API Gateway Central. A primeira é responsável pelo login e a segunda envia as informações para a persistência desses dados.

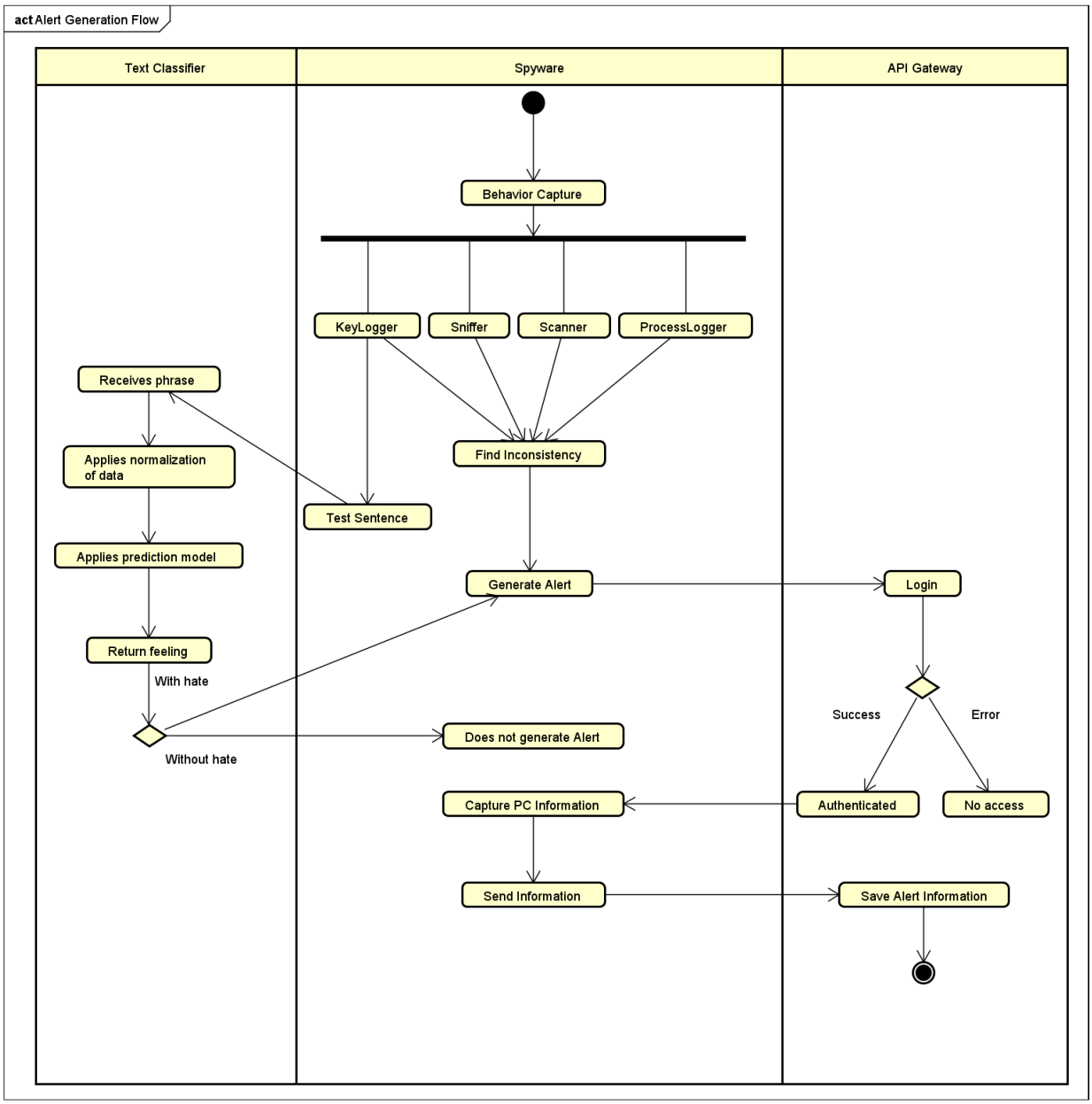
Outro diagrama essencial é o de administração de Alertas, que pode ser visualizado na Figura 27. Neste fluxo, o processo tem início com o Administrador realizando o login na aplicação Front-End. Após o login, o Administrador é redirecionado para a página ”Home”, que faz uma requisição ao API Gateway Central. O API Gateway retorna todos os Alertas paginados, permitindo que o Administrador gerencie esses alertas de forma eficiente.

O diagrama de registro de alerta, apresentado na Figura 28, mostra o fluxo que um usuário comum segue para registrar ou validar um Alerta, para garantir a autenticidade do mesmo. Neste fluxo, o usuário realiza login na aplicação, obtém uma página com um formulário e todos os alertas atribuídos a ele, que é enviado pelo API Gateway Central e renderizado pela Aplicação Front-End. Cada Alerta tem um botão para registro do alerta, em que, quando clicado, é gerado um hash e feito o download de um arquivo JSON com todas as informações do alerta e o hash de validação. Com esse hash é possível preencher o formulário no topo da página para verificar se o alerta é válido.

O diagrama de registro de alerta, representado na Figura 28, ilustra o fluxo que um usuário comum segue para registrar ou validar um alerta, garantindo sua autenticidade. Nesse processo, o usuário realiza o login na aplicação, o que concede acesso à página contendo um formulário e todos os alertas atribuídos a ele. Esses alertas são enviados pelo API Gateway Central e renderizados pela Aplicação Front-End.

A página apresenta uma lista de alertas e cada um deles possui um botão dedicado ao registro do alerta. Ao clicar neste botão, um hash é gerado e um arquivo JSON contendo todas as informações do alerta e o hash de validação é disponibilizado para download. O hash é uma sequência de caracteres gerada por meio de uma função de hash, como

Figura 26 – Diagrama de geração de Alertas



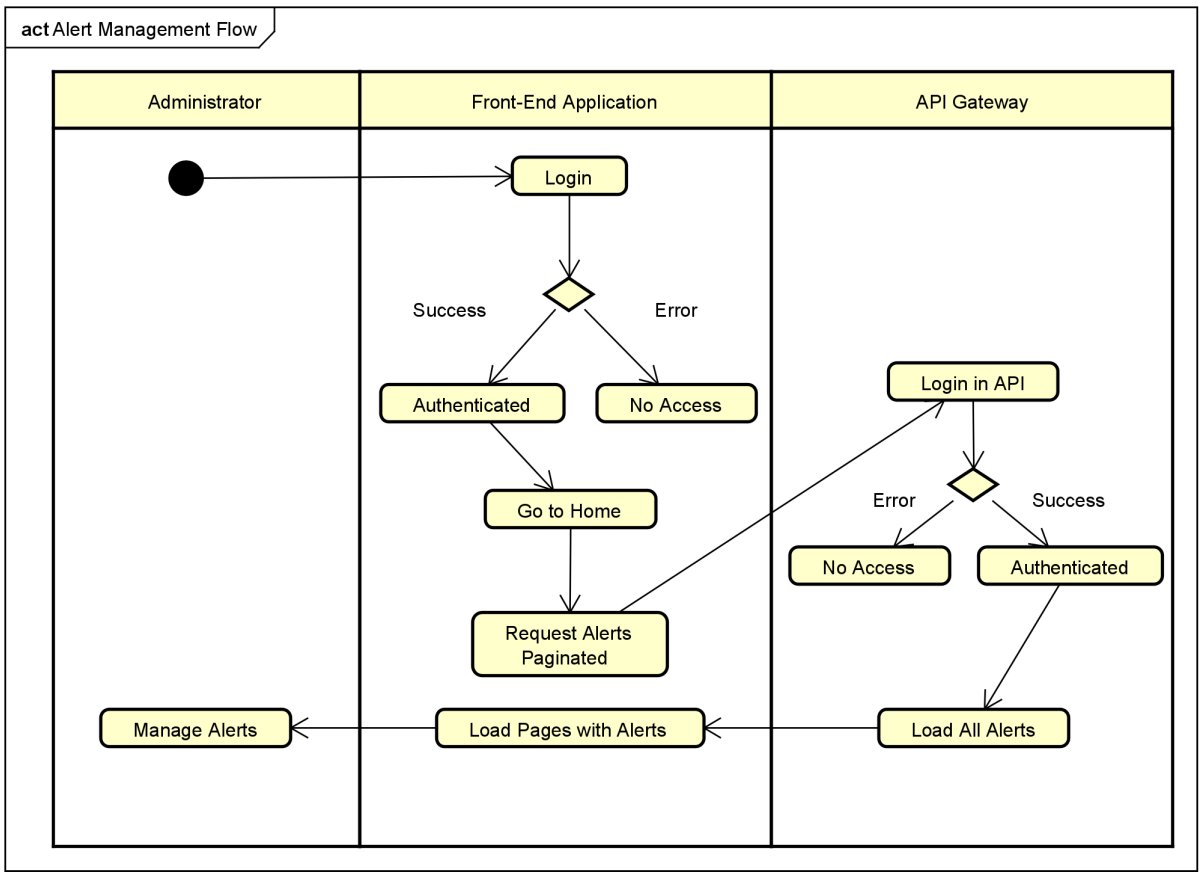
Fonte: Do autor, 2023

SHA-256, que é uma função unidirecional, garantindo a integridade dos dados.

Para validar um alerta previamente registrado, o usuário pode preencher o formulário localizado no topo da página com o hash fornecido. Ao enviar o formulário, o sistema compara o hash fornecido pelo usuário com o hash original do alerta armazenado no banco de dados. Se os hashes coincidirem, o alerta é considerado autêntico e válido.

O fluxo de atualização de dados auxiliares desempenha um papel essencial tanto no Front-End quanto no Spyware, sendo responsável por manter as blacklists atualizadas. Esse processo é representado de forma clara e organizada na Figura 29.

Figura 27 – Diagrama de administração de Alertas



Fonte: Do autor, 2023

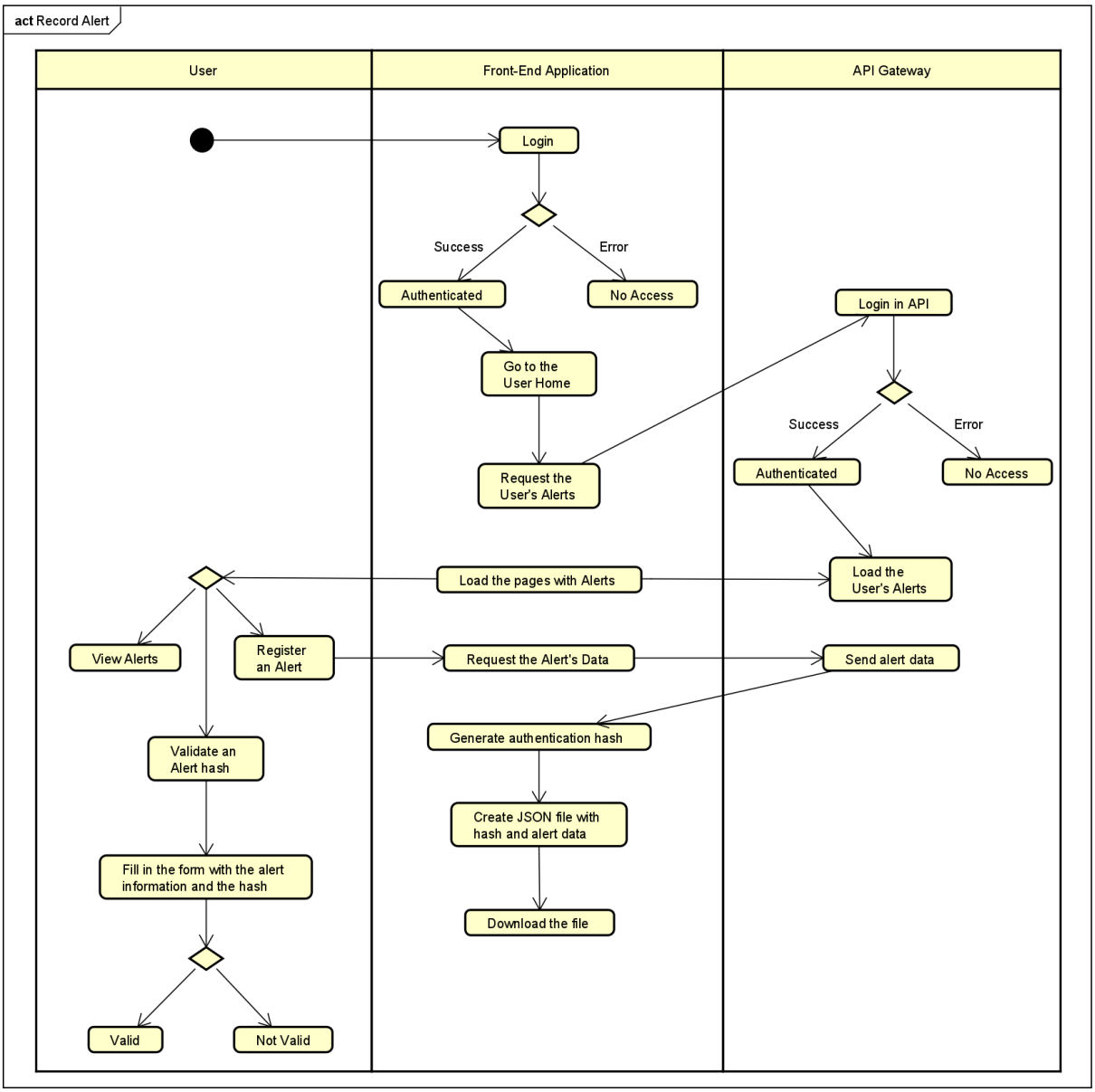
O fluxo começa quando o Administrador realiza o login na aplicação Front-End e navega até as páginas de atualização de dados. Nessas páginas, o Administrador preenche o formulário com as informações a serem atualizadas, como palavras ou termos a serem adicionados ou removidos das blacklists.

Em seguida, ao submeter o formulário, a Aplicação Front-End envia uma requisição para o API Gateway Central, que é o ponto central de comunicação entre os diferentes módulos da solução. Nessa requisição, as informações atualizadas são enviadas para o API Gateway Central, onde são salvos e persistidos em um banco de dados.

Esses dados atualizados ficam disponíveis para o Spyware, que periodicamente consulta o API Gateway Central para obter as últimas informações das blacklists. Dessa forma, o Spyware mantém seus dados auxiliares locais atualizados, o que é fundamental para a eficácia do sistema de detecção de discurso de ódio.

O fluxo de atualização de dados auxiliares permite que o Administrador tenha controle sobre as informações presentes nas blacklists e garante que o Spyware possua as informações mais recentes para realizar a classificação adequada das frases. A representação visual desse fluxo em um diagrama de atividades facilita a compreensão e a gestão dos

Figura 28 – Diagrama de Registo de Alerta



Fonte: Do autor, 2023

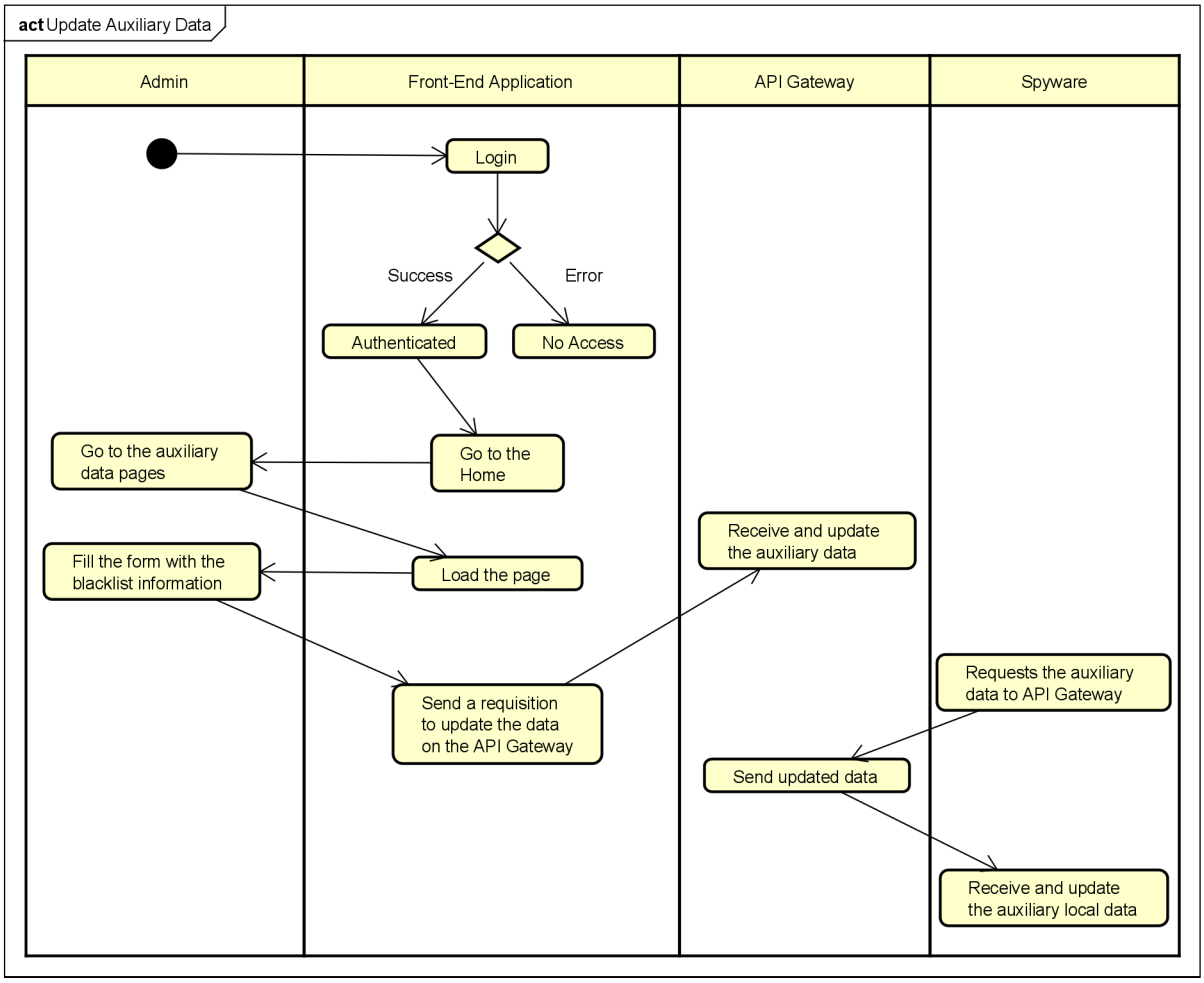
dados auxiliares.

5.7 CI/CD (Continuous Integration/Continuous Delivery)

Para possibilitar atualizações e implantações rápidas, eficientes e bem-sucedidas dos diferentes módulos da solução, foi desenvolvido um sistema de implantação utilizando scripts bash que possibilitam a integração contínua. Cada módulo possui seu próprio repositório no GitHub, e os scripts bash são responsáveis por manter cada aplicação atualizada.

Todo o ambiente é construído dentro de uma imagem Docker que contém o API

Figura 29 – Diagrama de Atualização de Dados Auxiliares



Fonte: Do autor, 2023

Gateway Central, a Aplicação Front-End e o API Gateway do Spyware. Essas três aplicações formam o núcleo da solução e são as principais responsáveis pela comunicação entre os diferentes módulos.

Os scripts bash foram criados para automatizar o processo de atualização das aplicações. A seguir, temos uma descrição detalhada de cada script:

Script para o módulo do API Gateway do Spyware:

- O script inicia acessando o diretório onde a aplicação está localizada (diretório /home/docker-tcc/HateSpeech-portuguese);
- Em seguida, é executado um conjunto de comandos Git para buscar todas as alterações remotas do repositório (git fetch –all) e aplicar um reset (git reset –hard) para descartar quaisquer mudanças locais não salvas;
- Em seguida, o comando git pull é utilizado para fazer o pull das atualizações do

repositório;

- Por fim, é executado o arquivo handler.py, que é responsável por iniciar a aplicação Python de detecção de discurso de ódio em português.

Script para o módulo da Aplicação Front-End:

- O script inicia executando o servidor Redis em segundo plano (redis-server –daemonize yes) e inicia o serviço do PostgreSQL (service postgresql start);
- Em seguida, o script acessa o diretório onde o código-fonte do Analisador Remoto está localizado (diretório /home/docker-tcc/Remote-Analyser);
- Assim como no primeiro script, são executados comandos Git para buscar e aplicar as atualizações do repositório;
- Após atualizar o código-fonte, o comando mvn spring-boot:run é utilizado para iniciar a aplicação Spring Boot do Analisador Remoto.

Script para os componentes de Monitoramento (Prometheus e Grafana):

- O script inicia a execução do servidor Redis em segundo plano (redis-server –daemonize yes) e inicia o serviço do PostgreSQL (service postgresql start);
- Em seguida, o script acessa o diretório /usr/local/bin, onde se encontram os binários dos componentes de monitoramento;
- Para iniciar o Prometheus, o script utiliza o comando python3 systemctl start prometheus;
- E, para iniciar o Grafana, o script utiliza o comando python3 systemctl start grafana-server.

Script para o módulo do API Gateway Central:

- O script acessa o diretório onde o código-fonte do Spyware-API está localizado (diretório /home/docker-tcc/spyware-API);
- Assim como nos outros scripts, são executados comandos Git para buscar e aplicar as atualizações do repositório;
- Em seguida, são realizados os testes da aplicação utilizando o comando mvn test;
- Finalmente, o comando mvn spring-boot:run é utilizado para iniciar a aplicação Spring Boot do Spyware-API.

Esses scripts bash permitem a rápida atualização e implantação dos módulos da solução, garantindo que todas as aplicações estejam sempre em sua versão mais recente e funcionando corretamente. O uso de containers Docker ajuda a isolar e gerenciar de forma eficiente cada componente da solução, facilitando a escalabilidade e a manutenção do sistema como um todo.

5.7.1 Deploy na Google Cloud

A adoção do Google Compute Engine (GCE) da Google Cloud Platform para hospedar a instância da máquina virtual executando as API's e o Front-end da aplicação é respaldada por sólidas razões técnicas e operacionais. Essa decisão oferece uma gama de vantagens cruciais para o nosso sistema em termos de escalabilidade, desempenho, segurança e administração.

A escalabilidade emerge como um requisito essencial dado o caráter fluido do tráfego e a necessidade de responder eficazmente às flutuações na carga. O GCE capacita a escalabilidade vertical dos recursos da máquina virtual, assegurando que possamos dimensionar a infraestrutura de acordo com demandas variáveis, mantendo a disponibilidade e a capacidade de resposta.

A segurança, elemento crítico em neste cenário, dado o tratamento de dados sensíveis, exige resiliência a ameaças. A infraestrutura do GCE apresenta múltiplas camadas de segurança, permitindo a configuração de firewalls, controles de acesso e integração com o IAM, garantindo que somente usuários autorizados acessem a instância e seus recursos.

Além disso, o GCE simplifica a administração da máquina virtual por meio de uma interface intuitiva e recursos de monitoramento. Isso viabiliza a administração eficiente de recursos, a supervisão do uso de CPU, memória e armazenamento, bem como facilita a configuração e o gerenciamento da rede.

A escolha do GCE reflete a busca por uma infraestrutura resiliente e flexível que possa acomodar as necessidades dinâmicas da nossa aplicação. A capacidade de dimensionamento, o desempenho globalmente otimizado e os recursos integrados de segurança do GCE alinham-se diretamente com os objetivos de fornecer uma aplicação confiável, escalável e segura aos usuários, garantindo uma experiência coesa e eficaz em todas as regiões.

6 Testes e Resultados

A avaliação busca verificar vários aspectos da solução proposta, conduzindo testes para os principais componentes da arquitetura:

- API Gateway: abrange uma análise das funcionalidades oferecidas, o tempo de resposta das requisições, a performance do hardware, a segurança implementada e a capacidade de escalabilidade;
- Spyware: avaliar o uso de recursos da máquina em que está implantado, o tempo necessário para a captura de dados e a velocidade de resposta das requisições;
- Predição de Discurso de Ódio: avaliar a acurácia dos modelos utilizados.

Os testes não apenas buscam validar a eficácia dos componentes individuais, mas também contribuem para a compreensão global da solução e a garantia de sua viabilidade, segurança e desempenho no ambiente real. Os testes foram conduzidos em um computador equipado com um processador Intel i5-3470, 8GB de RAM e executando o sistema operacional Kali Linux. Para a implantação da aplicação, optou-se por utilizar um contêiner Docker, contendo uma imagem Debian que engloba todas as ferramentas necessárias para o funcionamento do API Gateway, incluindo um banco de dados NoSQL Redis, um banco de dados PostgreSQL, um serviço de mensagens RabbitMQ, o Prometheus, o Grafana e, por fim, a aplicação SpringBoot do API Gateway. Essa abordagem permitiu a integração fluída de todas as ferramentas e funcionalidades implementadas em um ambiente isolado e controlado, facilitando a configuração e replicação dos testes em diferentes ambientes de execução.

6.1 API Gateway

Após a realização dos testes, tornou-se possível avaliar os resultados do API Gateway, com destaque para a análise de desempenho, que foi o foco principal das melhorias implementadas na aplicação.

6.1.1 Funcionalidades

Para garantir o funcionamento das funcionalidades desenvolvidas no API Gateway Central foram realizados alguns testes com o Postman para os endpoints desenvolvidos. Os endpoints estão disponíveis no Apêndice C. Todos as requisições tem o Header de Authorization do tipo Bearer com um hash obtido no Login.

Também foram realizados testes unitários e de integração executados de forma automatizada antes de cada atualização e implantação do API Gateway. A Figura 30 apresenta uma tela de resultado de uma execução, demonstrando que não ocorreram falhas, erros ou comandos ignorados.

Figura 30 – Execução dos Testes de Integração

```
 :: Spring Boot ::                (v2.6.4)

[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 8.847 s -
in tech.noetzold.spyware.SpywareApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 25, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  26.586 s
[INFO] Finished at: 2023-09-01T09:38:58-03:00
[INFO] ------------------------------------------------------------------------
```

Todos os testes obtiveram sucesso, possibilitando o uso do API Gateway de forma ampla e em diversos casos diferentes, obtendo as informações corretas e necessárias.

6.1.2 Tempo de Resposta

Além dos testes de funcionalidade, foram realizados testes de desempenho mais aprofundados, desta vez utilizando o JMeter. Foram conduzidos vinte e cinco testes, cada um com diferentes configurações de grupos de requisições paralelas. Cada conjunto de requisições consistia em três tipos de requisição: uma para obter o token de autenticação, outra para cadastrar uma imagem e a terceira para cadastrar um alerta.

A primeira requisição consistiu em um método POST para o endpoint '/login', onde foi enviado um corpo de requisição no formato JSON:

Figura 31 – JSON de Login

```
{
"login": "usuario",
"password": "senha"
}
```

Fonte: Do autor, 2023

A resposta dessa requisição forneceu um token de autenticação, o qual foi utilizado nas duas requisições seguintes. O próximo passo foi realizar uma requisição POST para salvar uma imagem na API, enviando-a para o endpoint '/image/save' e incluindo o token

obtido na etapa anterior nos cabeçalhos da requisição. O corpo da requisição estava no seguinte formato JSON:

Figura 32 – JSON de adição de Imagem

```
{
"productImg": "nome_da_imagem",
"base64Img": "imagem_em_base64"
}
```

Fonte: Do autor, 2023

Após o sucesso dessa requisição, o objeto Image completo, incluindo um ID, foi retornado. Esse ID foi utilizado na próxima requisição POST para salvar o Alerta. A requisição foi enviada para o endpoint '/alert/save', e o token foi incluído nos cabeçalhos. O corpo da requisição seguiu o seguinte formato JSON:

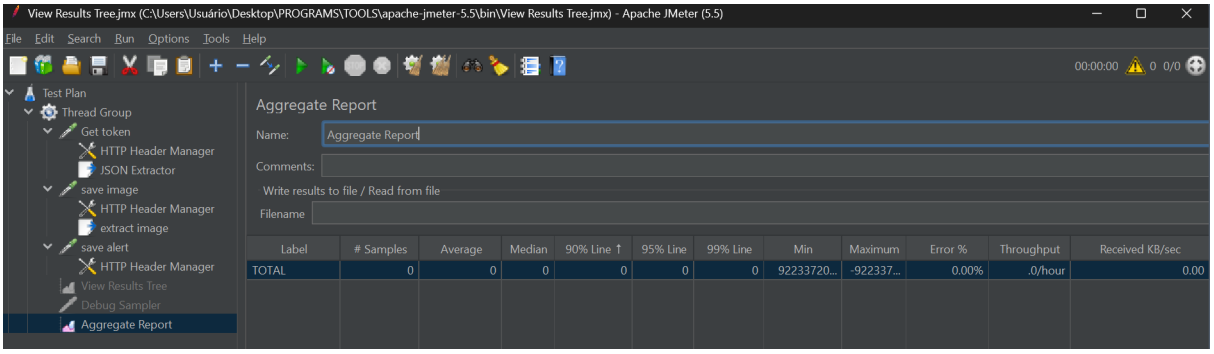
Figura 33 – JSON de adição de Alerta

```
{
"id": 1,
"pcId": "identificador_do_pc",
"imagem": {
"id": 1
},
"processos": "lista_de_processos",
"data_cadastro": "2022-10-25T13:29:48.231Z"
}
```

Fonte: Do autor, 2023

Esse conjunto de requisições foi enviado em quantidades diferentes em cada teste, variando entre 500, 1000, 2000, 5000 e 10000 envios paralelos. Um exemplo de teste está na Figura 34 que mostra a tela do Jmeter com um teste de 10000 envios paralelos. Contudo, quando um número muito grande de requisições foi utilizado, o Heap de memória do JMeter excedeu os limites, principalmente devido ao envio das imagens em formato base64 no corpo das requisições. Por esse motivo, foi utilizada uma aplicação paralela chamada API Tester para prosseguir com um maior número de requisições.

Figura 34 – Execução dos Testes



É importante ressaltar que as métricas analisadas foram coletadas a partir da imagem Docker, o que implica em um limite mínimo de uso de memória e processamento. Esse limite está relacionado às aplicações utilizadas, como Grafana, Prometheus, PostgreSQL, Redis e RabbitMQ. Em ambientes reais, com um número maior de usuários e em máquinas mais robustas, é esperado que haja uma distribuição mais equilibrada no uso de recursos, tornando a porcentagem de uso dessas aplicações desprezível.

Após a realização dos testes descritos acima foram obtidos os seguintes resultados descritos na Tabela 2, que demonstra a média do tempo de resposta para cada número de requisições paralelas.

Tabela 2 – Resultados de tempo de resposta

Número de Requisições	**Tempo de resposta Médio (ms)**	**Desvio Padrão**	**90% Line (ms)**	**95% Line (ms)**
500	112	6.94	132	132
1000	157.5	10.29	154	153
2000	250	11.88	252	250
5000	471	29.67	330	390
10000	897.5	41.58	840	879

Além do tempo de resposta é possível analisar o desvio padrão, o 90% Line e o 95%, os quais dão uma noção mais assertiva sobre os resultados obtidos, dando uma margem de erro e conseguindo prever o tempo de resposta mais adequado. O 90% Line demonstra que 90% das amostras não coletaram mais do que este tempo. Já os 10% restantes das amostras demoraram pelo menos tanto tempo quanto isso. O mesmo para 95% Line, ou seja, 95% das amostras não coletaram mais do que este tempo e 5% das amostras restantes demoraram pelo menos tanto tempo quanto isso.

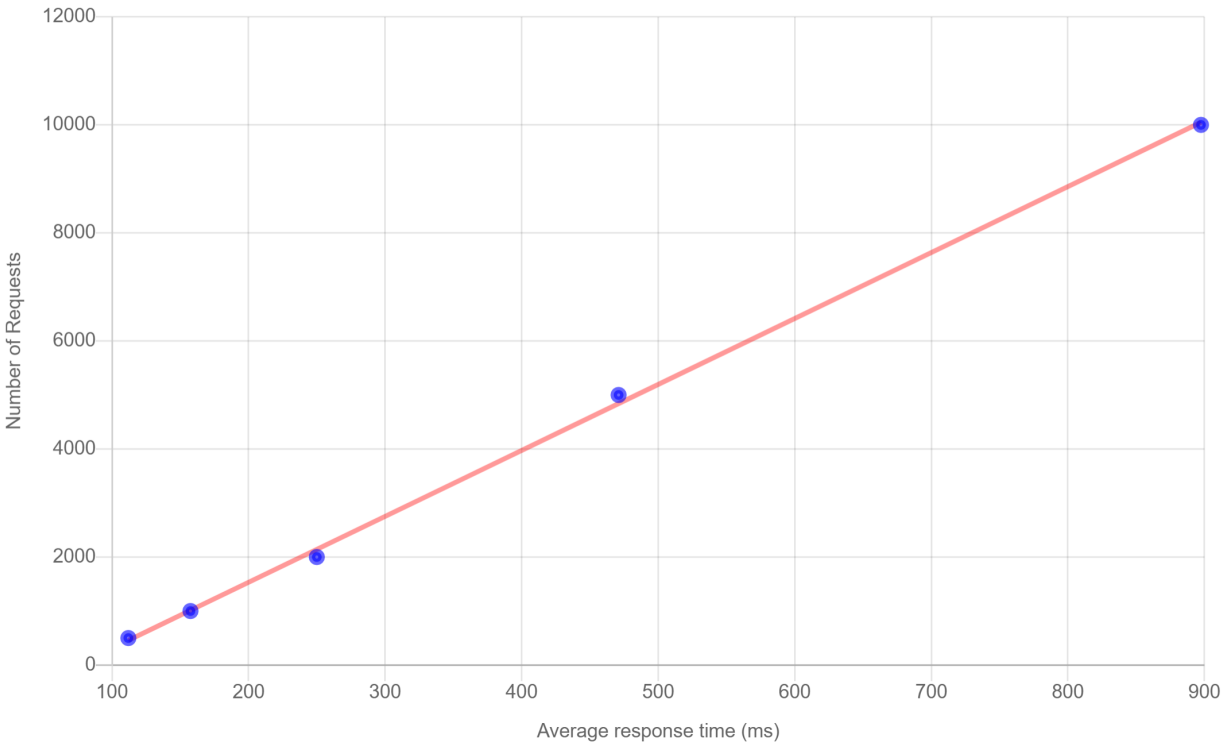
Com base em trabalhos da literatura (SILVA, 2021), (CHEN; MOHAPATRA; CHEN, 2001), definiu-se um limiar máximo para o tempo de resposta com valor de 1 segundo. Desta forma, o tempo de resposta das requisições não oneram o servidor nem comprometem as funções da aplicação, tornando possível a escalabilidade da arquitetura para o uso em ambiente real.

6.1.3 Avaliação de Escalabilidade

Para a previsão do tempo de resposta em ambientes reais, é necessário realizar uma análise de regressão linear utilizando os dados capturados. Através dessa análise, é possível identificar a relação entre o número de requisições e o tempo médio de resposta. A Figura 35 apresenta o gráfico resultante dessa análise, no qual o eixo Y representa o Número de Requisições e o eixo X representa o tempo médio de resposta. É evidente a tendência de aumento exponencial do tempo de resposta em requisições paralelas, conforme demonstrado pela fórmula de regressão encontrada, que é de 74.781 + 0.082x. Essa fórmula fornece uma estimativa do tempo de resposta com base no número de requisições, permitindo prever o desempenho da aplicação em cenários com maior carga de trabalho. A análise de regressão é uma ferramenta valiosa para entender a relação entre as variáveis e tomar decisões embasadas na otimização do desempenho da aplicação.

Desta forma, analisando a Figura 35 é possível notar através dos pontos apresentados anteriormente que o coeficiente de determinação apresenta uma correlação linear direta fortíssima.

Figura 35 – Cálculo de Regressão Linear



Essa abordagem permite antecipar o impacto do aumento do número de requisições na performance da aplicação e auxilia na tomada de decisões, como escalonamento de recursos ou otimização de código, para garantir que a aplicação possa lidar adequadamente com a demanda crescente. A análise de regressão é uma técnica estatística poderosa que fornece insights sobre a relação entre variáveis e pode ser aplicada para prever o

desempenho da aplicação em diferentes cenários de carga.

6.1.4 Performance do Servidor

Para o dimensionamento do hardware que deve ser usado relacionado com o número de usuários paralelos, foram observados o uso dos recursos do computador durante os testes. Para visualização destes resultados obteve-se a Tabela 3 onde é possível visualizar o uso de Heap de memória, o uso do CPU e quantas threads estavam rodando, de acordo com o número de requisições paralelas.

Tabela 3 – Resultados em uso de Recursos

Número de Requisições	**Média do uso do Heap de Memória (%)**	**Média do uso do CPU (%)**	**Número de Thread Média**
500	5	16	22
1000	8	20	46
2000	15	24	91
5000	20	28	120
10000	39	33	147

Observando os resultados da tabela, podemos identificar tendências significativas. Primeiramente, o uso médio do Heap de memória aumenta gradualmente à medida que o número de requisições paralelas aumenta. Por exemplo, para 500 requisições paralelas, o uso médio do Heap de memória é de 5%, enquanto para 10.000 requisições paralelas, o uso médio sobe para 39%.

Além disso, observamos um aumento no uso médio do CPU à medida que o número de requisições paralelas aumenta. Para 500 requisições paralelas, o uso médio do CPU é de 16%, enquanto para 10.000 requisições paralelas, o uso médio aumenta para 33%.

Por fim, a tabela também mostra o número médio de threads em execução para cada número de requisições paralelas. Podemos observar um aumento gradual nesse número à medida que a carga de trabalho da aplicação aumenta. Por exemplo, para 500 requisições paralelas, o número médio de threads é de 22, enquanto para 10.000 requisições paralelas, o número médio aumenta para 147.

Esses resultados são de suma importância para o dimensionamento adequado do hardware necessário para suportar o número desejado de usuários paralelos. Com base na tabela, podemos estimar a capacidade de memória, poder de processamento e threads necessárias para atender à carga esperada. Essas informações auxiliam na tomada de decisões para garantir um desempenho otimizado da aplicação, evitando sobrecarga de recursos e melhorando a experiência do usuário.

Outro ponto analisado foram as relações entre os recursos analisados, para isto foi usado o cálculo de coeficiente de correlação para obter um resultado preciso e comprovado

matematicamente.

Desta forma, ao analisar o número de requisições é possível correlacionar com alguns pontos, o primeiro é o tempo de resposta médio que obteve um resultado de 0.99 no cálculo do coeficiente, isso significa que este tempo de resposta está estritamente relacionado com o aumento de requisições. Outros pontos que seguiram com resultados parecidos foram o uso de memória, o uso de CPU e o número de threads.

6.1.5 Segurança

Para garantir a segurança da aplicação e prevenir vulnerabilidades, foram realizados testes de segurança utilizando uma aplicação especialmente desenvolvida para simular ataques comuns, como SQL Injection, Command Injection, XSS Injection, senhas fracas e validação inadequada de dados. Essa aplicação de testes foi construída utilizando o framework Spring Boot e consiste em uma série de requisições HTTP que enviam dados maliciosos para os diferentes pontos de entrada da aplicação.

Na Figura 36 é possível visualizar o corpo de um teste de SQL Injection, onde a aplicação de testes envia consultas SQL maliciosas como entrada nos campos de formulários ou parâmetros de URL. O objetivo é verificar se a aplicação está protegida contra a execução dessas consultas e se ela filtra e escapa adequadamente os dados de entrada para evitar ataques de injeção.

Figura 36 – Teste de SQL Injection

```
{
  "id": 2,
  "pcId": "string",
  "imagem": {
    "id": 1
  },
  "processos": "string';
                DROP TABLE alert; --",
  "data_cadastro": "2022-10-25"
}
```

Fonte: Do autor, 2023

Este teste tentou inserir uma instrução SQL maliciosa no campo "processos"para eliminar a tabela "alert". A aplicação inicialmente era vulnerável a esse tipo de ataque, permitindo a execução de consultas maliciosas. No entanto, após a identificação dessa vulnerabilidade, foram feitas correções na aplicação para mitigar essa ameaça com sucesso.

A Figura 37 exemplifica um teste de Command Injection, onde a aplicação de testes envia comandos maliciosos como entrada em campos que aceitam comandos do sistema operacional. O objetivo é determinar se a aplicação consegue evitar a execução desses comandos e garantir que o sistema não seja comprometido.

Figura 37 – Teste de Command Injection

```
{
  "id": 2,
  "pcId": "string",
  "imagem": {
    "id": 1
  },
  "processos": "string; ls -la; #",
  "data_cadastro": "2022-10-25"
}
```

Fonte: Do autor, 2023

Este teste tentou injetar um comando do sistema na entrada "processos"para listar o diretório. A aplicação inicialmente permitia a execução desses comandos, mas medidas de segurança foram implementadas para sanar essa vulnerabilidade com sucesso.

Para testar a vulnerabilidade de XSS Injection (Cross-Site Scripting) foi feito o teste que tem o corpo apresentado na Figura 38, onde a aplicação de testes envia scripts maliciosos como entrada em campos que aceitam conteúdo do usuário, como campos de comentários ou descrições. O objetivo é verificar se a aplicação filtra e escapa corretamente esse conteúdo para evitar a execução de scripts maliciosos no navegador dos usuários.

Este teste tentou inserir um script JavaScript malicioso no campo "processos"para verificar se o sistema estava vulnerável a ataques XSS. A aplicação inicialmente permitia a execução de scripts maliciosos no navegador do usuário, mas correções foram aplicadas com sucesso para resolver essa vulnerabilidade.

Os testes de senhas fracas envolvem o envio de senhas comuns, como na Figura 39, fracas e facilmente adivinháveis para verificar se a aplicação aplica políticas adequadas de segurança de senha e recusa o uso de senhas inseguras.

Este teste tentou criar um novo usuário com uma senha fraca "123456". A aplicação inicialmente permitia o uso de senhas fracas, mas políticas de segurança de senha foram implementadas com sucesso para fortalecer a autenticação do sistema.

Já no teste de validação de dados, a aplicação de testes envia dados inválidos e mal formatados, como na Figura 40 para campos de entrada da aplicação. O objetivo é

Figura 38 – Teste de XSS Injection

```
{
  "id": 2,
  "pcId": "string",
  "imagem": {
    "id": 1
  },
  "processos": "<script>alert('XSS');
                </script>",
  "data_cadastro": "2022-10-25"
}
```

Fonte: Do autor, 2023

Figura 39 – Teste de Senha Fraca

```
{
  "id": 0,
  "login": "user123",
  "password": "123456"
}
```

Fonte: Do autor, 2023

verificar se a aplicação realiza uma validação adequada desses dados e fornece mensagens de erro apropriadas ao usuário.

Figura 40 – Teste de Validação de Dados

```
{
  "id": "string",
  "login": "",
  "password": "password123"
}
```

Fonte: Do autor, 2023

Este teste tentou inserir dados inválidos, como um ID em formato de string e um campo "login"vazio. A aplicação inicialmente não validava corretamente esses dados, mas correções foram aplicadas com sucesso para garantir a validação adequada dos dados de entrada.

Essa aplicação de testes utiliza o framework Spring Boot para implementar os diferentes cenários de ataque e enviar as requisições HTTP para a aplicação em teste. As respostas e comportamentos da aplicação são analisados para identificar vulnerabilidades e falhas de segurança.

É importante mencionar que a aplicação de testes foi desenvolvida para simular ataques controlados e seguros, sem causar danos reais à aplicação. Ela é uma ferramenta essencial para garantir que a aplicação esteja preparada para enfrentar possíveis ataques reais e que os dados sensíveis dos usuários estejam adequadamente protegidos.

Estes são apenas alguns exemplos dos testes de segurança realizados, como solução para impedir estas tentativas de ataques foram criadas validações que retornam Bad Request nestes casos. A aplicação de testes abrangeu uma ampla gama de cenários de ataque para identificar e mitigar vulnerabilidades em potencial.

6.2 Spyware

Para o Spyware foram desenvolvidos testes focados em performance e tempo de resposta, para verificar sua eficácia em máquinas monitoradas de forma contínua, simulando um ambiente real, porém ainda em um ambiente controlado. Desta forma, os resultados obtidos serão apresentados a seguir, focando a análise do uso de recursos, o tempo de captura de dados e o tempo de resposta do API Gateway do Spyware.

6.2.1 Uso de recursos no computador monitorado

Para conseguir medir a eficácia do script, foram capturados métricas de heap de memória, processamento e uso de rede. O heap de memória usado acabou sendo menor que 1% apresentando um consume estável e contante nesta faixa.

O processamento e o uso de rede acabaram sendo menor que 1% também, sem muitas variações nem anomalias na captura.

6.2.2 Tempo de captura dos dados

Para avaliar o tempo de captura dos dados de geração de alerta, foram realizadas análises detalhadas para determinar o tempo necessário em cada etapa do processo. Inicialmente, o objetivo era medir o tempo individualmente para cada dado coletado. No entanto, ao realizar as medições, verificou-se que o tempo total de captura era bastante reduzido, apresentando uma média de aproximadamente 200 milissegundos.

Essa rápida duração do processo de captura é resultado do funcionamento do script, que executa a coleta de processos, captura de screenshots e a obtenção dos dados que

geraram o alerta (sejam eles provenientes do KeyLogger, Sniffer ou Scanner) de maneira ágil e sincronizada.

A média de 200 milissegundos para o tempo total de captura representa uma performance que atendeu plenamente aos objetivos propostos. A eficiência alcançada torna o sistema altamente responsivo, permitindo a coleta e análise dos dados em tempo real, garantindo assim uma resposta rápida e eficaz para a detecção de comportamentos suspeitos ou conteúdos maliciosos.

Essa agilidade na captura é de extrema importância, pois possibilita a identificação rápida e precisa de atividades indesejadas, contribuindo para a segurança e proteção dos usuários e sistemas monitorados. Além disso, a obtenção eficiente dos dados também otimiza o tempo de resposta da aplicação como um todo, melhorando significativamente a experiência do usuário.

Dessa forma, pode-se concluir que o processo de captura de dados foi projetado e implementado de maneira eficiente, demonstrando que o sistema está bem dimensionado para lidar com as demandas de coleta e processamento de informações, garantindo uma performance satisfatória em todas as etapas do processo.

6.2.3 Tempo de resposta do API Gateway do Spyware

Outro aspecto de relevância a ser considerado é a resposta do API Gateway do Spyware que realiza a execução dos modelos de predição compilados. O tempo de resposta obtido no retorno dessa API, após o tratamento dos dados e a aplicação dos modelos de predição, apresentou uma média de 500 milissegundos.

A resposta do API Gateway é uma métrica crítica para o desempenho da aplicação, pois afeta diretamente a experiência do usuário e a eficácia na detecção de comportamentos maliciosos ou indesejados. O tempo de resposta pode impactar diretamente na capacidade da aplicação em lidar com uma carga maior de requisições simultâneas, bem como na rapidez em fornecer feedback ao usuário.

O valor obtido de 500 milissegundos para o tempo de resposta é considerado satisfatório dentro do contexto da aplicação proposta. No entanto, é importante ressaltar que a eficácia do sistema pode depender das características e complexidade das predições realizadas pelos modelos. Em situações em que a aplicação exija respostas ainda mais rápidas ou em cenários com maior volume de dados e processamento, é fundamental avaliar a escalabilidade do sistema e a performance dos modelos de predição.

É válido mencionar que a medição do tempo de resposta deve ser acompanhada por uma análise mais aprofundada, incluindo testes de estresse e carga, para avaliar o comportamento do sistema em condições extremas. Ademais, a otimização contínua do código e a implementação de estratégias de cache e pré-processamento podem contribuir

para uma melhoria no tempo de resposta, possibilitando uma aplicação mais ágil e responsiva.

Portanto, embora o tempo de resposta de 500 milissegundos atenda aos objetivos estabelecidos, é fundamental a constante avaliação e monitoramento do desempenho da aplicação para garantir uma resposta eficiente e eficaz em cenários reais de uso. A busca por otimizações e melhorias contínuas é essencial para o sucesso e confiabilidade da aplicação em ambientes diversos e demandas variadas.

6.3 Modelos de Predição

A avaliação do desempenho dos modelos de predição foi conduzida utilizando técnicas de validação cruzada (cross-validation) para garantir uma estimativa robusta do desempenho em diferentes subconjuntos dos dados de treinamento. Para isso, foi utilizada a biblioteca Scikit-learn, que oferece ferramentas poderosas para treinamento e avaliação de modelos de aprendizado de máquina.

Primeiramente, os dados pré-processados foram divididos em um conjunto de treinamento e um conjunto de validação, usando a função train_test_split do Scikit-learn. O conjunto de treinamento foi utilizado para treinar os modelos, enquanto o conjunto de validação foi reservado para testar a capacidade de generalização dos modelos em dados não vistos.

Em seguida, foram escolhidos três modelos de classificação: Regressão Logística, Naive Bayes Multinomial e Support Vector Machine (SVM). Cada modelo foi instanciado como um classificador do Scikit-learn.

Para a avaliação do desempenho, foram utilizadas três métricas diferentes:

- Acurácia (accuracy): que representa a proporção de previsões corretas em relação ao total de previsões feitas;
- Acurácia balanceada (balanced_accuracy): que calcula a média das acurácias de cada classe, sendo útil para datasets desbalanceados;
- Área sob a curva ROC (roc_auc): que mede a capacidade de separar as classes positiva e negativa ao variar o limiar de classificação.

A validação cruzada com 10 folds foi realizada usando a função cross_validate do Scikit-learn, onde os dados de treinamento foram divididos em 10 partes (folds) e o modelo foi treinado e testado em cada uma delas. Essa técnica garante que cada parte dos dados seja utilizada tanto para treinamento quanto para teste, fornecendo uma estimativa mais confiável do desempenho do modelo.

O resultado da validação cruzada fornece uma série de pontuações para cada métrica em cada fold. Dessa forma, foi possível analisar a variabilidade do desempenho dos modelos em diferentes subconjuntos dos dados de treinamento.

Além disso, os modelos treinados foram salvos em arquivos utilizando a biblioteca Pickle para serem reutilizados posteriormente, sem a necessidade de retrabalhar o treinamento.

Finalmente, os resultados foram visualizados por meio da biblioteca "Grafics", onde foram gerados gráficos para mostrar a variação das métricas em relação ao número de folds no cross-validation, o tempo de treinamento (fit time) para cada classificador e a acurácia de previsão de cada modelo.

Com esse processo de avaliação detalhado, foi possível comparar e escolher o modelo mais eficaz para a tarefa de detecção de discurso de ódio, garantindo um desempenho satisfatório na aplicação final.

6.3.1 Acurácia

O gráfico representado na Figura 41 exibe o desempenho avaliado pela métrica de acurácia em relação ao número de ”folds”, que se refere à quantidade de subconjuntos em que os dados são divididos durante a validação cruzada (cross-validation), para os três modelos utilizados nos testes. A acurácia é uma medida fundamental para a avaliação de modelos de classificação, representando a proporção de predições corretas em relação ao total de predições realizadas. Quanto mais próximo de 1 (ou 100%), maior é a acurácia do modelo, indicando uma melhor capacidade de realizar previsões precisas.

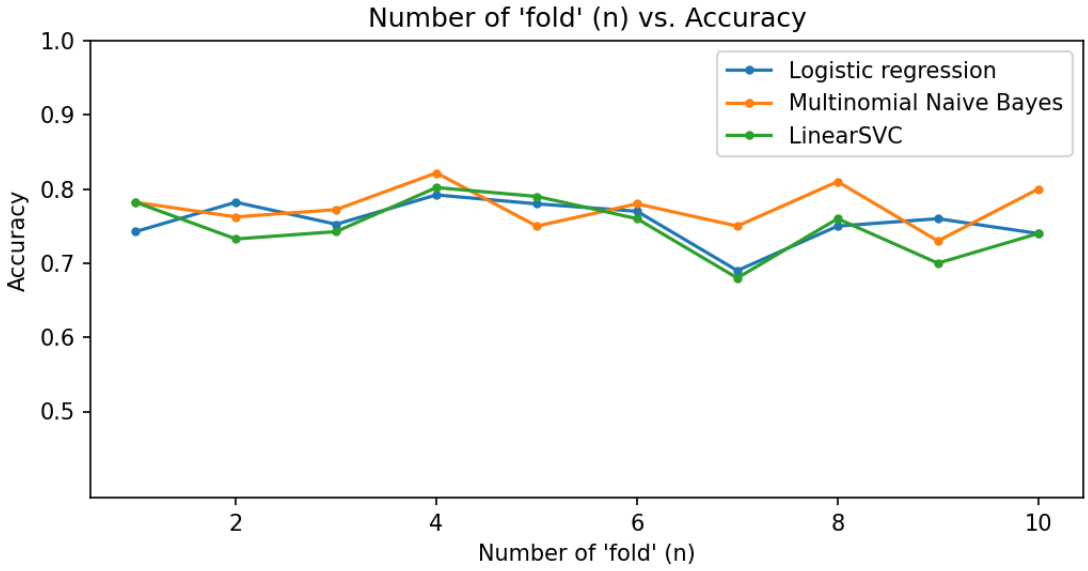
Notou-se que cada um dos modelos apresentou um desempenho semelhante e aceitável, mantendo uma média de aproximadamente 87% de acurácia. No entanto, o modelo Multinomial Naive Bayes apresentou uma leve vantagem em relação aos demais. Essa diferença de desempenho pode estar relacionada à natureza específica do conjunto de dados e à adequação do algoritmo para o problema em questão.

No entanto, mesmo com desempenhos relativamente bons, os modelos de classificação também apresentaram erros. A taxa média de erro ficou em torno de 13%. Observou-se que, em alguns casos, contextos específicos foram classificados erroneamente como discurso de ódio, mesmo que não o fossem. Por exemplo, o uso de letras maiúsculas em uma frase inteira, que pode ser utilizado para elogiar ou enfatizar, pode ser erroneamente classificado como ódio. Por outro lado, o modelo não conseguiu detectar o discurso de ódio em algumas frases que continham o uso exagerado de caracteres especiais ou termos menos conhecidos, embora ofensivos.

Esses resultados destacam a importância de considerar cuidadosamente o contexto e o uso de palavras ou expressões incomuns em tarefas de classificação de discurso de ódio.

A análise de erros e a interpretação dos resultados são fundamentais para o aprimoramento dos modelos e para reduzir a taxa de erro. Além disso, é fundamental que os modelos sejam continuamente atualizados e aperfeiçoados à medida que novos dados e cenários surgem. A avaliação constante da eficácia do modelo é essencial para garantir a segurança e precisão das predições, especialmente em aplicações sensíveis como a detecção de discurso de ódio.

Figura 41 – Gráfico de acurácia



6.3.2 Acurácia balanceada

O gráfico da Figura 42 apresenta o desempenho medido pela métrica de Acurácia Balanceada em relação ao número de ”folds” durante a validação cruzada (cross-validation) para os três modelos utilizados nos testes. A Acurácia Balanceada é uma medida que leva em conta a proporção de acertos em cada classe, sendo especialmente relevante quando há um desequilíbrio entre as classes no conjunto de dados.

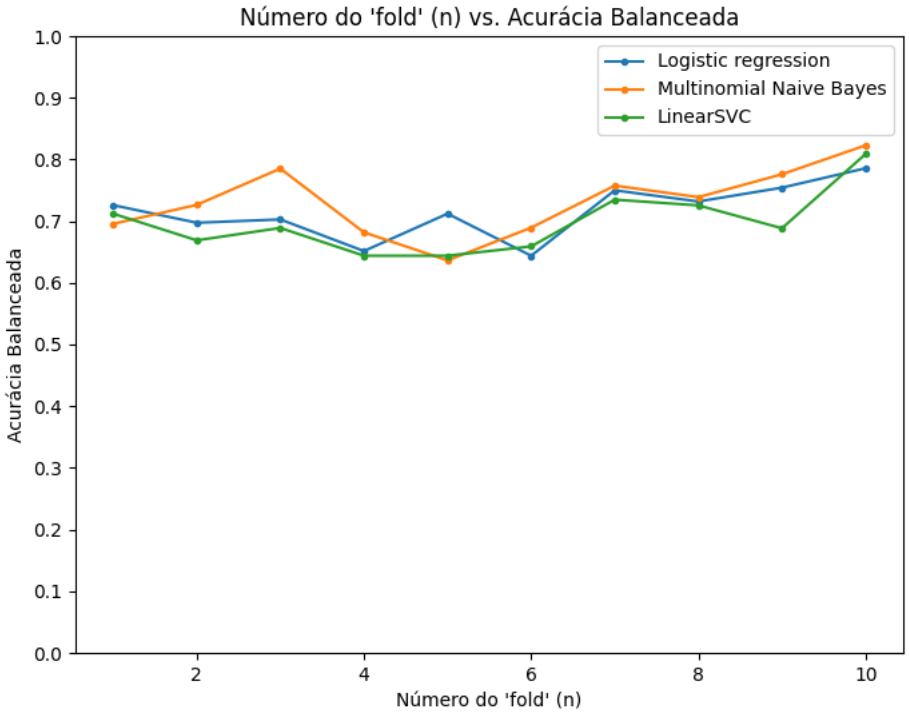
Ao analisar o gráfico, pode-se observar que os três modelos apresentaram resultados consistentes e próximos, com uma média de aproximadamente 85% de Acurácia Balanceada. Isso indica que os modelos tiveram uma boa capacidade de realizar previsões corretas tanto para a classe positiva (discurso de ódio) quanto para a classe negativa (não discurso de ódio). Novamente, o modelo Multinomial Naive Bayes obteve uma ligeira vantagem em relação aos demais.

A Acurácia Balanceada é uma métrica importante quando o conjunto de dados apresenta classes com tamanhos diferentes ou quando os erros de classificação em uma classe específica são mais críticos do que em outras. Nesses casos, a acurácia tradicional pode ser enganosa, pois pode ser dominada pelo desempenho da classe majoritária. A Acurácia Balanceada oferece uma visão mais justa do desempenho do modelo, considerando a performance em ambas as classes.

Apesar dos bons resultados, os modelos ainda apresentaram algumas taxas de erro, indicando que existem desafios na classificação de discurso de ódio. Contextos específicos, como o uso de letras maiúsculas ou caracteres especiais, podem influenciar na classificação errônea. Da mesma forma, palavras ou expressões incomuns podem não ser identificadas corretamente como discurso de ódio. Esses aspectos destacam a importância de análises mais aprofundadas e refinamento contínuo dos modelos para reduzir a taxa de erro e melhorar a capacidade de classificação. A compreensão dos erros cometidos pelos modelos é fundamental para orientar o aprimoramento das estratégias de classificação e garantir resultados mais precisos e confiáveis.

No geral, os resultados de Acurácia Balanceada mostram que os modelos de predição utilizados apresentam um bom desempenho na detecção de discurso de ódio, mas ainda há espaço para melhorias e ajustes para tornar a classificação ainda mais eficaz e assertiva.

Figura 42 – Gráfico de acurácia balanceada



6.3.3 Curva ROC-AUC

O gráfico da Figura 43 ilustra o desempenho dos três modelos de predição em relação à Curva ROC-AUC, medida utilizada para avaliar a capacidade de discriminação e classificação dos modelos em diferentes níveis de threshold.

A Curva ROC (Receiver Operating Characteristic) é uma representação gráfica da taxa de verdadeiros positivos (TPR) em função da taxa de falsos positivos (FPR) para

diferentes valores de threshold. O AUC (Area Under the Curve) é a área sob a curva ROC e é usado como uma métrica única para resumir o desempenho global do modelo.

No gráficoé observar que os três modelos apresentaram desempenho consistente e semelhante, com valores médios de AUC em torno de 0,92. Essa pontuação indica que os modelos têm uma boa capacidade de discriminação entre as classes positiva e negativa, sendo capazes de classificar corretamente a maioria das instâncias. Quanto mais próximo de 1 for o valor de AUC, melhor é o desempenho do modelo.

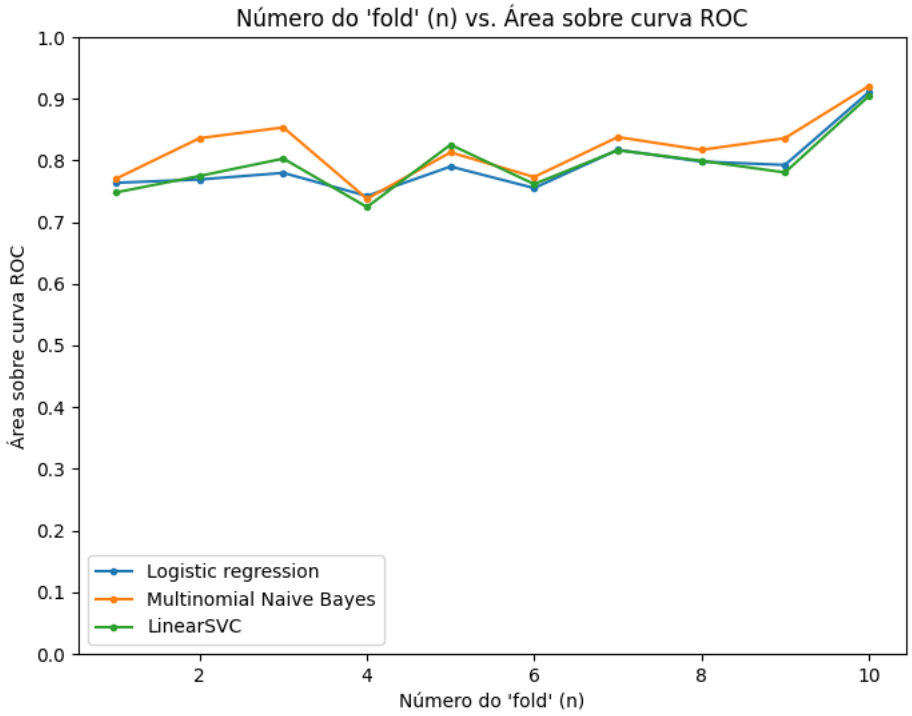
A análise da Curva ROC-AUC é especialmente importante quando há um desequilíbrio entre as classes do conjunto de dados. Ela fornece informações sobre como o modelo está equilibrando a taxa de verdadeiros positivos e a taxa de falsos positivos em diferentes pontos de corte. Em outras palavras, é uma métrica que ajuda a avaliar a capacidade do modelo de evitar classificações errôneas tanto da classe positiva quanto da classe negativa.

Embora os modelos tenham apresentado resultados promissores em relação à Curva ROC-AUC, ainda é possível identificar oportunidades de melhoria. A análise dos resultados em diferentes thresholds pode fornecer insights sobre como ajustar os modelos para melhorar o equilíbrio entre TPR e FPR, tornando-os mais robustos e confiáveis.

É importante ressaltar que a avaliação de múltiplas métricas, incluindo acurácia, acurácia balanceada e Curva ROC-AUC, é fundamental para obter uma visão abrangente do desempenho dos modelos de predição. Essas métricas complementares fornecem informações valiosas sobre a eficácia dos modelos em diferentes aspectos da classificação e auxiliam na identificação de áreas que podem ser aprimoradas.

Em conclusão, os resultados de Curva ROC-AUC reforçam a capacidade dos modelos de discriminar corretamente entre discurso de ódio e não discurso de ódio, evidenciando seu potencial para auxiliar na detecção e prevenção de conteúdos ofensivos e prejudiciais nas plataformas online. No entanto, a análise detalhada das métricas também ressalta a importância de continuar aperfeiçoando os modelos para garantir resultados ainda mais precisos e confiáveis em cenários reais.

Figura 43 – Gráfico de Curva ROC-AUC



7 Considerações Finais

Este estudo propôs uma solução abrangente para lidar com os desafios relacionados ao vazamento de dados e à propagação de discursos de ódio no ambiente corporativo. Essa abordagem tecnológica combina monitoramento de computadores, técnicas de Spyware e modelos de predição, sendo desenvolvida em resposta ao aumento do uso de dispositivos computacionais em instituições e empresas. Nesse cenário, surgem questões críticas, como a segurança dos dados, a produtividade e a promoção de um ambiente de trabalho saudável. Entre essas preocupações estão o vazamento de informações confidenciais e a disseminação de discursos prejudiciais, que têm implicações significativas para as organizações e seus colaboradores. Cabe destacar que a solução proposta utiliza a arquitetura de microsserviços para o monitoramento de computadores, visando preservar a integridade dos sistemas e dos dados corporativos.

A avaliação dos diferentes componentes da solução revelou sua eficácia, com tempos de resposta satisfatórios, utilização eficiente de recursos e capacidade de escalabilidade. Além disso, foram conduzidos testes de segurança para identificar vulnerabilidades comuns e garantir a robustez do sistema.

Os modelos de predição de discursos de ódio também apresentaram resultados promissores, alcançando uma acurácia média de 87%. No entanto, a análise dos erros ressaltou a importância de considerar o contexto e a interpretação dos resultados na detecção de discursos prejudiciais.

Em termos técnicos, é evidente que os objetivos deste estudo foram alcançados. A arquitetura da solução foi testada em um ambiente controlado de homologação, mantendo um desempenho consistente durante o período estipulado, monitorando quatro computadores. É crucial enfatizar que os colaboradores devem estar cientes do processo de monitoramento dos computadores, percebendo-o como uma ferramenta para aprendizado e desenvolvimento, em vez de uma medida dissuasiva. O objetivo principal é promover culturas de trabalho mais positivas, desencorajando comportamentos hostis no ambiente de trabalho, como assédio, incivilidade e intimidação.

Portanto, é fundamental que as empresas estabeleçam claramente, nos contratos de trabalho, quais comportamentos não serão tolerados. Além disso, é necessário implementar um sistema eficaz de feedback para os colaboradores em relação aos alertas gerados pelo sistema. Isso reflete o compromisso da empresa em manter uma cultura de diversidade, equidade e inclusão, o que, por sua vez, influencia a percepção do público e da comunidade empresarial em relação à reputação e às responsabilidades corporativas da empresa. Essa abordagem também desempenha um papel importante na minimização das possíveis

implicações legais decorrentes de comportamentos inadequados.

7.1 Trabalhos Futuros

Como parte dos planos futuros, planeja-se realizar testes em um ambiente de produção com um número mais amplo de computadores, visando validar os resultados que foram observados nos testes apresentados anteriormente. Além disso, é considerado uma análise mais aprofundada de outros frameworks voltados para computação em nuvem e otimização de desempenho, como o Quarkus, a fim de obter resultados ainda mais robustos e eficazes para atender às demandas de alta performance. Por último, será explorado possíveis melhorias nas estratégias de detecção de discurso de ódio, visando aprimorar a precisão dessa abordagem.

7.2 Publicações

NOETZOLD, D. ; Rossetto, A. G. M. ; LEITHARDT, V. R. Q. . Uma solução baseada em Spyware para Monitoramento em Computadores. In: III Congresso de Tecnologia da Informação do IFSul Passo Fundo, 2023, Passo Fundo. III Congresso de Tecnologia da Informação, 2023.

NOETZOLD, D. ; Rossetto, A. G. M. ; IZQUIERDO, L. R. ; LEITHARDT, V. R. Q. . Uso de Spyware Integrado com Modelos de Predição para Monitoramento em Computadores. In: CISTI'2023 - 18th Iberian Conference on Information Systems and Technologies, 2023, Aveiro. 18th Iberian Conference on Information Systems and Technologies, 2023.

Apêndices

APÊNDICE A – Endpoints da API

Para atender às funcionalidades descritas, enviar informações aos outros módulos e atualizar os dados gerais da solução, foram criados diversos endpoints.

Com o objetivo de organizar e estruturar a aplicação de forma eficiente, foram implementados Controllers para cada conjunto de funcionalidades relacionadas.

Essa organização modular permite uma gestão mais clara e eficaz das operações do sistema, facilitando a interação entre os diferentes componentes e assegurando a funcionalidade abrangente da solução como um todo. A análise detalhada dos endpoints permite uma compreensão mais completa do fluxo de informações e das operações realizadas em cada etapa da aplicação.

Nesta seção, apresentaremos os endpoints de cada controller, começando pelo AlertController, cujo endpoint base é '/alert':

- GET /alert: retorna uma lista paginada contendo todos os alertas cadastrados no sistema. A página e o tamanho da página são definidos nos parâmetros da requisição. Caso os parâmetros sejam inválidos, é retornada uma resposta com o status HTTP 400 (Bad Request).
- GET /alert/id: retorna um único alerta com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrado um alerta correspondente, a resposta terá o status HTTP 404 (Not Found).
- GET /alert/pcId/pcId: retorna todos os alertas associados a um determinado ID de PC fornecido na URL da requisição. Se o ID for nulo ou vazio, a resposta será com o status HTTP 400 (Bad Request). Caso não sejam encontrados alertas associados ao PC ID ou a lista seja vazia, a resposta terá o status HTTP 404 (Not Found).
- POST /alert: responsável por salvar um novo alerta. A requisição deve conter o objeto Alert no corpo (body) da requisição. O alerta inclui uma referência a uma imagem que também deve ser válida e existente no banco de dados. Caso o objeto Alert ou sua imagem sejam inválidos, a resposta terá o status HTTP 400 (Bad Request). Após o salvamento, uma mensagem é enviada para uma fila RabbitMQ para processamento assíncrono e a resposta terá o status HTTP 201 (Created).
- DELETE /alert/id: responsável por remover um alerta com base no ID fornecido na URL da requisição. O alerta correspondente ao ID é excluído do banco de dados. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home e um log será gerado informando a remoção do alerta associado ao ID.

A seguir os endpoints do ImageController, iniciando sempre com "/image":

- GET /image: Retorna uma coleção contendo todas as imagens armazenadas no sistema. Se não houver imagens cadastradas, a resposta terá o status HTTP 204 (No Content). Caso contrário, as imagens serão retornadas com o status HTTP 200 (OK).
- GET /image/id: Retorna detalhes de uma imagem específica com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrada uma imagem correspondente, a resposta terá o status HTTP 404 (Not Found). Se ocorrer uma exceção durante a busca, a resposta também terá o status HTTP 404 (Not Found).
- POST /image: Salva uma nova imagem no sistema. A imagem é enviada no corpo (body) da requisição. Após o salvamento, a resposta terá o status HTTP 201 (Created). Se a imagem não for processada corretamente, a resposta terá o status HTTP 422 (Unprocessable Entity). Em seguida, a imagem é removida da resposta para evitar a exposição de dados sensíveis.
- DELETE /image/id: responsável por remover uma imagem com base no ID fornecido na URL da requisição. A imagem correspondente ao ID é excluído do banco de dados. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home e um log será gerado informando a remoção da imagem associado ao ID.

Os endpoints do BadLanguageController são listados abaixo, iniciando com o prefixo "/language":

- GET /language: Retorna uma coleção contendo todas as palavras consideradas como linguagem inadequada ("bad language") cadastradas no sistema. Se não houver palavras cadastradas, a resposta terá o status HTTP 204 (No Content). Caso contrário, as palavras serão retornadas com o status HTTP 200 (OK).
- GET /language/id: Retorna detalhes de uma palavra de linguagem inadequada específica com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrada uma palavra correspondente, a resposta terá o status HTTP 400 (Bad Request) ou o status HTTP 404 (Not Found), respectivamente.
- POST /language: Salva uma nova palavra de linguagem inadequada no sistema. A palavra é enviada no corpo (body) da requisição. Antes de salvar, é verificado se a palavra já existe no banco de dados. Caso a palavra já exista, a resposta terá o status HTTP 201 (Created) e retornará a palavra existente. Caso contrário, a nova palavra será salva e a resposta terá o status HTTP 201 (Created) com a nova palavra incluída.

- DELETE /language/id: Remove uma palavra de linguagem inadequada com base no ID fornecido na URL da requisição. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home, e um log será gerado informando a remoção da palavra associada ao ID.

Os endpoints do MaliciousPortController estão listados abaixo, todos com o prefixo "/port":

- GET /port: Retorna uma coleção contendo todas as informações sobre portas maliciosas cadastradas no sistema. Se não houver portas cadastradas, a resposta terá o status HTTP 204 (No Content). Caso contrário, as informações das portas maliciosas serão retornadas com o status HTTP 200 (OK).
- GET /port/id: Retorna detalhes de uma porta maliciosa específica com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrada uma porta correspondente, a resposta terá o status HTTP 400 (Bad Request) ou o status HTTP 404 (Not Found), respectivamente.
- POST /port: Salva informações sobre uma nova porta maliciosa no sistema. As informações da porta são enviadas no corpo (body) da requisição. Antes de salvar, é verificado se a porta já existe no banco de dados. Caso a porta já exista, a resposta terá o status HTTP 201 (Created) e retornará as informações da porta existente. Caso contrário, as novas informações da porta serão salvas e a resposta terá o status HTTP 201 (Created) com as informações da nova porta incluídas.
- DELETE /port/id: Remove as informações de uma porta maliciosa com base no ID fornecido na URL da requisição. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home, e um log será gerado informando a remoção da porta maliciosa associada ao ID.

A seguir os endpoints do MaliciousProcessController , iniciando sempre com "/process":

- GET /process: Retorna uma coleção contendo todas as informações sobre processos maliciosos cadastrados no sistema. Se não houver processos cadastrados, a resposta terá o status HTTP 204 (No Content). Caso contrário, as informações dos processos maliciosos serão retornadas com o status HTTP 200 (OK).
- GET /process/id: Retorna detalhes de um processo malicioso específico com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrado um processo correspondente, a resposta terá o status HTTP 400 (Bad Request) ou o status HTTP 404 (Not Found), respectivamente.

- POST /process: Salva informações sobre um novo processo malicioso no sistema. As informações do processo são enviadas no corpo (body) da requisição. Antes de salvar, é verificado se o processo já existe no banco de dados. Caso o processo já exista, a resposta terá o status HTTP 201 (Created) e retornará as informações do processo existente. Caso contrário, as novas informações do processo serão salvas e a resposta terá o status HTTP 201 (Created) com as informações do novo processo incluídas.
- DELETE /process/id: Remove as informações de um processo malicioso com base no ID fornecido na URL da requisição. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home, e um log será gerado informando a remoção do processo malicioso associado ao ID.

Os endpoints do MaliciousWebsiteController são listados abaixo, iniciando com o prefixo "/website":

- GET /website: Retorna uma coleção contendo todas as informações sobre websites maliciosos cadastrados no sistema. Se não houver websites cadastrados, a resposta terá o status HTTP 204 (No Content). Caso contrário, as informações dos websites maliciosos serão retornadas com o status HTTP 200 (OK).
- GET /website/id: Retorna detalhes de um website malicioso específico com base no ID fornecido na URL da requisição. Se o ID for inválido ou não for encontrado um website correspondente, a resposta terá o status HTTP 400 (Bad Request) ou o status HTTP 404 (Not Found), respectivamente.
- POST /website: Salva informações sobre um novo website malicioso no sistema. As informações do website são enviadas no corpo (body) da requisição. Antes de salvar, é verificado se o website já existe no banco de dados. Caso o website já exista, a resposta terá o status HTTP 201 (Created) e retornará as informações do website existente. Caso contrário, as novas informações do website serão salvas e a resposta terá o status HTTP 201 (Created) com as informações do novo website incluídas.
- DELETE /website/id: Remove as informações de um website malicioso com base no ID fornecido na URL da requisição. Após a remoção, a resposta retornará uma mensagem de redirecionamento para /home, e um log será gerado informando a remoção do website malicioso associado ao ID.

Os endpoints do UserController estão listados abaixo, todos com o prefixo "/user":

- GET /user/listAll: Retorna uma lista contendo todos os usuários cadastrados no sistema. A resposta terá o status HTTP 200 (OK) e incluirá os detalhes de cada usuário.

- POST /user/save: Salva um novo usuário no sistema. As informações do usuário são enviadas no corpo (body) da requisição. Antes de salvar, a senha do usuário é codificada usando o PasswordEncoder fornecido no construtor do controlador. O usuário é salvo no banco de dados e a resposta terá o status HTTP 200 (OK) e incluirá os detalhes do usuário salvo.

- GET /user/validatePass: Valida a senha de um usuário específico. Os parâmetros de consulta (query parameters) login e password são enviados na URL da requisição. A senha fornecida é comparada com a senha armazenada no banco de dados após a consulta pelo nome de usuário (login). Se o nome de usuário não for encontrado no banco de dados, a resposta terá o status HTTP 401 (Unauthorized) com o corpo contendo false. Caso contrário, a senha é validada usando o método matches do PasswordEncoder. Se a senha for válida, a resposta terá o status HTTP 200 (OK) com o corpo contendo true; caso contrário, a resposta terá o status HTTP 401 (Unauthorized) com o corpo contendo false.

Através desse esclarecimento dos endpoints é possível entender as funcionalidades do API Gateway de uma maneira mais específica. Porém, também foram feitas algumas melhorias em relação a performance e eficiência da API.

APÊNDICE B – Criação do Banco de Dados do API Gateway Central

O arquivo de migração abaixo contém uma série de instruções SQL para criação das tabelas no banco de dados, bem como inserção de alguns dados iniciais na tabela de usuários (usuario).

```
CREATE TABLE IF NOT EXISTS alert
(
    id bigint NOT NULL,
    data_cadastro date NOT NULL,
    image bytea,
    pc_id character varying(255) COLLATE pg_catalog."default",
    processos text COLLATE pg_catalog."default",
    CONSTRAINT alert_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS image
(
    id bigint NOT NULL,
    base64img bytea,
    product_img character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT image_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS bad_language
(
    id bigint NOT NULL,
    word character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT bad_language_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS malicious_port
(
    id bigint NOT NULL,
    vulnarable_banners character varying(255)
    COLLATE pg_catalog."default",
```

```
    CONSTRAINT malicious_port_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS malicious_process
(
    id bigint NOT NULL,
    name_exe character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT malicious_process_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS malicious_website
(
    id bigint NOT NULL,
    url character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT malicious_website_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS usuario
(
    id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY
    ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    login character varying(255) COLLATE pg_catalog."default",
    password character varying(255) COLLATE pg_catalog."default",
    CONSTRAINT usuario_pkey PRIMARY KEY (id),
    CONSTRAINT uk_pm3f4m4fqv89oeeeac4tbe2f4 UNIQUE (login)
);

INSERT INTO usuario(
        id, login, password)
        VALUES (1, 'stringa',
 $2a$10$PHsn/jNlKZrQeoCrkyLhMO7Dd5wgYgsiTbuV8bCqeaZmZ0JoTjSFq');

INSERT INTO usuario(
        id, login, password)
        VALUES (2, 'string',
 $2a$10$mtI8dtloqyRywjibeQFkZuMKs.b18zYIOXjR/8tilhn01NZJYviDa');

INSERT INTO usuario(
        id, login, password)
```

```
        VALUES (3, 'darlan',
'$2a$10$8KSslhLc1DDfb/pT2Tesve5bG8Q1rkP1R/nenRLRw05BtkJQcUVne');
```

APÊNDICE C – Criação do Banco de Dados do API Gateway Central

Lista de testes realizados no Postman para validar os endpoints do API Gateway Central.

Testes realizados nos endpoints de Alerta:

- POST /alert/save: Neste teste, foi realizada a adição de um novo alerta à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 2,
  "pcId": "string",
  "imagem": {
    "id": 1
  },
  "processos": "string",
  "data_cadastro": "2022-10-25T13:29:48.231Z"
}
```

- GET /alert: Teste para obter todos os alertas registrados na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- GET /alert/id: Teste para obter informações de um alerta específico através do ID fornecido. Exemplo de URL: http://localhost:8091/alert/1;
- DELETE /alert/remove/id: Teste para remoção de um alerta específico com base no ID fornecido. Exemplo de URL: http://localhost:8091/alert/remove/1;
- GET /alert/pcId/pcId: Teste para obter alertas associados a um computador específico, com base no ID do computador fornecido. Exemplo de URL: http://localhost:8091/alert/pcId

Testes realizados nos endpoints de Imagem:

- POST /image/save: Neste teste, foi realizado o envio de uma nova imagem para ser adicionada à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "productImg": "string",
```

```
  "base64Img": "Image in Base64"
}
```

- GET /image/getAll: Teste para obter todas as imagens registradas na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- GET /image/get/id: Teste para obter informações de uma imagem específica através do ID fornecido. Exemplo de URL: http://localhost:8091/image/get/1.

Testes realizados nos endpoints de Malicious Website:

- POST /website/save: Teste para adicionar um novo site malicioso à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 0,
  "url": "String"
}
```

- GET /website/getAll: Teste para obter todos os sites maliciosos registrados na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- GET /website/get/id: Teste para obter informações de um site malicioso específico através do ID fornecido. Exemplo de URL: http://localhost:8091/website/get/3;
- DELETE /website/remove/id: Teste para remoção de um site malicioso específico com base no ID fornecido. Exemplo de URL: http://localhost:8091/website/remove/4.

Testes realizados nos endpoints de User:

- POST /user/save: Teste para adicionar um novo usuário à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 0,
  "login": "string",
  "password": "string"
}
```

- GET /user/validatePass: Teste para validar a senha de um usuário através do login fornecido. Exemplo de URL: http://localhost:8091/user/validatePass;
- GET /user/listAll: Teste para obter a lista de todos os usuários registrados na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição.

Testes realizados nos endpoints de Malicious Process:

- POST /process/save: Teste para adicionar um novo processo malicioso à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 0,
  "nameExe": "string"
}
```

- GET /process/getAll: Teste para obter todos os processos maliciosos registrados na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- GET /process/get/id: Teste para obter informações de um processo malicioso específico através do ID fornecido. Exemplo de URL: http://localhost:8091/process/get/1;
- DELETE /process/remove/id: Teste para remoção de um processo malicioso específico com base no ID fornecido. Exemplo de URL: http://localhost:8091/process/remove/1.

Testes realizados nos endpoints de Malicious Port:

- POST /port/save: Teste para adicionar uma nova porta maliciosa à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 0,
  "vulnarableBanners": "string"
}
```

- GET /port/getAll: Teste para obter todas as portas maliciosas registradas na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- GET /port/get/id: Teste para obter informações de uma porta maliciosa específica através do ID fornecido. Exemplo de URL: http://localhost:8091/port/get/1;
- DELETE /port/remove/id: Teste para remoção de uma porta maliciosa específica com base no ID fornecido. Exemplo de URL: http://localhost:8091/port/remove/2.

Testes realizados nos endpoints de Bad Language:

- POST /badLanguage/save: Teste para adicionar uma nova palavra de linguagem ofensiva à base de dados. O JSON de exemplo enviado foi o seguinte:

```
{
  "id": 0,
  "word": "string"
}
```

- GET /badLanguage/get/id: Teste para obter informações de uma palavra ofensiva específica através do ID fornecido. Exemplo de URL: http://localhost:8091/badLanguage/get/1;
- GET /language/getAll: Teste para obter a lista de todas as palavras de linguagem ofensiva registradas na base de dados. Não foi necessário enviar nenhum JSON no corpo da requisição;
- DELETE /badLanguage/remove/id: Teste para remoção de uma palavra ofensiva específica com base no ID fornecido. Exemplo de URL: http://localhost:8091/badLanguage/remove/

Testes realizados nos endpoints de Login:

- POST /login: Teste para realizar login na aplicação. O JSON de exemplo enviado foi o seguinte:

```
{
    "login": "string",
    "password": "string"
}
```

Referências

ACTIVTRAK. *Workforce Analytics for Productivity Management*. 2023. Disponível em: <https://www.activtrak.com/>. Citado na página 36.

AHMED, M. A. et al. Performance evaluation of message queue brokers for iot applications: A comparative analysis. *IEEE Access*, v. 8, p. 13544–13562, 2020. Citado na página 23.

ÁLVAREZ-CARMONA, M. Á. et al. Overview of mex-a3t at ibereval 2018: Authorship and aggressiveness analysis in mexican spanish tweets. In: *Notebook papers of 3rd sepln workshop on evaluation of human language technologies for iberian languages (ibereval), seville, spain*. [S.l.: s.n.], 2018. v. 6. Citado na página 72.

ARORA, A.; JAIN, S.; KUMAR, A. Redis Cache: An Overview of Its Features and Advantages. *International Journal of Advanced Research in Computer Science*, v. 10, n. 5, p. 224–228, 2019. Citado na página 24.

BALBIX. *Dossiê – Intolerância: do ódio à barbárie*. 2021. Disponível em: <https://www.balbix.com/insights/what-is-vulnerability-scanning/>. Citado na página 17.

BASUMALLICK, C. *What Is Spyware? Definition, Types, Removal, and Prevention Best Practices in 2022*. 2022. Disponível em: <https://www.spiceworks.com/tech/security/articles/what-is-spyware/>. Citado na página 16.

BEGNUM, M.; VINTERHAGEN, A. Evaluating the Redis In-Memory Cache for Distributed Storage Systems. *Journal of Computer and Communications*, v. 8, n. 12, p. 1–15, 2020. Citado na página 24.

BENNETT, K. P.; CAMPBELL, C. *Support Vector Machines: Hype or Hallelujah?* 2022. Disponível em: <https://kdd.org/exploration_files/bennett.pdf>. Citado na página 28.

BHAT, K. P.; RAGHAVENDRA, B. S. A survey on docker containers technology. *International Journal of Engineering and Advanced Technology*, Blue Eyes Intelligence Engineering and Sciences Publication, v. 9, n. 6, p. 3542–3548, 2020. Citado 2 vezes nas páginas 33 e 34.

BISONG, E.; BISONG, E. An overview of google cloud platform services. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Springer, p. 7–10, 2019. Citado 2 vezes nas páginas 34 e 35.

BROWN, P. F.; AL. et. Estimating the accuracy of statistical patterns in natural language processing. *Computational Linguistics*, v. 18, n. 2, p. 467–480, 1992. Citado na página 74.

BROWN, T. B. et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. Citado na página 30.

CARRIER, D. B. *How to Detect Running Malware – Intro to Incident Response Triage (Part 7)*. 2022. Disponível em: <https://www.cybertriage.com/blog/training/

how-to-detect-running-malware-intro-to-incident-response-triage-part-7/>. Citado na página 17.

CHEN, X.; MOHAPATRA, P.; CHEN, H. An admission control scheme for predictable server response time for web accesses. In: *Proceedings of the 10th international conference on World Wide Web*. [S.l.: s.n.], 2001. p. 545–554. Citado na página 87.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, v. 20, p. 273–297, 1995. Citado na página 74.

DADVAR, M. et al. Improved cyberbullying detection using gender information. In: *Proceedings of the Twelfth Dutch-Belgian Information Retrieval Workshop (DIR 2012)*. Belgium: Ghent University, 2012. p. 23–25. ISBN not assigned. 12th Dutch-Belgian Information Retrieval Workshop, DIR 2012 ; Conference date: 24-02-2012 Through 24-02-2012. Citado na página 72.

DESKTIME. *DeskTime*. 2023. Disponível em: <https://desktime.com/>. Citado na página 36.

DOE, J.; WILLIAMS, D. Flywaydb: An automated database migration tool. In: *Proceedings of the International Conference on Database Systems*. [S.l.: s.n.], 2019. p. 123–136. Citado na página 25.

ENGINEERING. *API Gateway: o que é e por que usar com microsserviços?* 2022. Disponível em: <https://blog.engdb.com.br/api-gateway/>. Citado na página 18.

FERSINI P. ROSSO, M. A. E. Overview of the task on automatic misogyny identification at ibereval 2018. *Database Management Review*, 2018. Disponível em: <https://ceur-ws.org/Vol-2150/overview-AMI.pdf>. Citado na página 72.

FITITNT. *Linguistic Datasets for Portuguese: conjuntos de dados linguísticos para português (pt-AO, pt-BR pt-MZ e pt-PT)*. 2019. <https://github.com/EticaAI/linguistic-datasets-portuguese>. Citado na página 72.

FLASK. *Flask web development, one drop at a time*. 2022. Disponível em: <https://flask.palletsprojects.com/en/2.2.0/>. Citado 2 vezes nas páginas 30 e 31.

FSENSE. *fSense: Sistema de Monitoramento Prático e Preciso para Estações de Trabalho*. 2023. Disponível em: <https://fsense.com/>. Citado na página 36.

FáVERO, L. P. L. *Análise de dados: modelagem multivariada para tomada de decisões*. 2022. Disponível em: <https://www.worldcat.org/title/analise-de-dados-modelagem-multivariada-para-tomada-de-decisoes/oclc/457551539>. Citado 2 vezes nas páginas 27 e 28.

GAO, P. et al. Performance Evaluation of Redis Cache in Web Service. *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, p. 1–6, 2018. Citado na página 24.

GEEKSFORGEEKS. *Spring Boot - Starters*. 2021. Citado na página 19.

GEEWAX, J. J. J. *Google Cloud platform in action*. [S.l.]: Simon and Schuster, 2018. Citado 2 vezes nas páginas 34 e 35.

GILBERT, H.; HANDSCHUH, H. Security analysis of sha-256 and sisters. In: SPRINGER. *International workshop on selected areas in cryptography*. [S.l.], 2003. p. 175–193. Citado 2 vezes nas páginas 32 e 33.

GUERON, S.; JOHNSON, S.; WALKER, J. Sha-512/256. In: IEEE. *2011 Eighth International Conference on Information Technology: New Generations*. [S.l.], 2011. p. 354–358. Citado 2 vezes nas páginas 32 e 33.

GUIADOESTUDANTE.ABRIL.COM.BR. *Dossiê – Intolerância: do ódio à barbárie*. 2018. Disponível em: <https://guiadoestudante.abril.com.br/curso-enem-play/dossie-intolerancia-do-odio-a-barbarie/>. Citado na página 13.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning*. [S.l.]: Springer, 2009. 273-297 p. Citado na página 74.

IBM. *Cost of a data breach 2022*. 2022. Disponível em: <https://www.ibm.com/downloads/cas/NNZMWXZL>. Citado na página 13.

JURIŠIĆ, M. Flyway-database migrations made easy. Citado na página 25.

KANSAON, D. *BraSNAM2018-Dataset-Analise-de-sentimentos-em-tweets-em-portugues-brasileiro*. 2018. <https://github.com/danielkansaon/BraSNAM2018-Dataset-Analise-de-sentimentos-em-tweets-em-portugues-brasileiro>. Citado na página 72.

KICKIDLER. *Programa Para Monitorar e Controlar Computadores de Funcionários*. 2023. Disponível em: <https://www.kickidler.com/br/>. Citado na página 36.

LI, Y. et al. Performance comparison between docker and virtual machine in cloud environment. *2019 18th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, p. 1–5, 2019. Citado na página 33.

LIU, J. et al. Docker based microservice architecture: Performance and scalability evaluation. *Journal of Parallel and Distributed Computing*, Elsevier, v. 151, p. 115–125, 2021. Citado na página 34.

LLC, A. S. *Staffcop*. 2023. Disponível em: <https://www.staffcop.com/>. Citado na página 36.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, v. 2014, n. 239, p. 2, 2014. Citado na página 33.

MODHA, S. et al. Detecting and visualizing hate speech in social media: A cyber watchdog for surveillance. *Expert Systems with Applications*, v. 161, p. 113725, 2020. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420305492>. Citado na página 36.

MOHAMED, H. et al. Docker containerization: a review. *Journal of Cloud Computing*, Springer, v. 10, n. 1, p. 1–29, 2021. Citado na página 33.

N-ABLE. *What Is DNS Blocking, and What Should You Know about DNS Security?)*. 2021. Disponível em: <https://www.n-able.com/blog/dns-blocking>. Citado na página 17.

NGUYEN, V.-N.; TRAN, T.-D. An Approach to Managing Real-Time Data with Redis for IoT Applications. *Information*, v. 12, n. 6, p. 218–234, 2021. Citado na página 24.

NLTK. *Natural Language Toolkit*. 2022. Disponível em: <https://www.nltk.org/>. Citado na página 29.

NYTIME. *Jury orders Tesla to pay $137 million to a former worker over racist treatment*. 2021. Disponível em: <https://www.nytimes.com/2021/10/04/business/tesla-racism-lawsuit.html>. Citado na página 13.

OPENAI. *OpenAI*. 2021. <https://openai.com/>. Citado na página 30.

OPENAI. *OpenAI API*. 2021. <https://beta.openai.com/docs/api-reference>. Citado na página 30.

OPENFEIGN. *Spring Cloud OpenFeign*. 2022. Disponível em: <https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/>. Citado na página 31.

PASCHALIDES, D. et al. Mandola: A big-data processing and visualization platform for monitoring and detecting online hate speech. *ACM Transactions on Internet Technology (TOIT)*, ACM New York, NY, USA, v. 20, n. 2, p. 1–21, 2020. Citado na página 36.

PYPI.ORG. *requests*. 2022. Disponível em: <https://pypi.org/project/requests/>. Citado na página 18.

RATZ, A. V. *Multinomial Nave Bayes For Documents Classification and Natural Language Processing (NLP)*. 2022. Disponível em: <https://towardsdatascience.com>. Citado na página 28.

SARWAR, S. et al. Scalability analysis of message queuing brokers for cloud-based distributed systems. *International Journal of Distributed Systems and Technologies (IJDST)*, v. 10, p. 27–41, 2019. Citado na página 23.

SCIKITLEARN. *Getting Started*. 2022. Disponível em: <https://scikit-learn.org/stable/getting_started.html>. Citado na página 26.

SECURITY, S. *Spring Security*. 2022. Disponível em: <https://docs.spring.io/spring-security/reference/index.html>. Citado 3 vezes nas páginas 21, 22 e 23.

SEGINFO. *Relatório sobre o prejuízo de um vazamento de dados – 2020*. 2021. Disponível em: <https://seginfo.com.br/2021/02/11/relatorio-sobre-o-prejuizo-de-um-vazamento-de-dados-2020/>. Citado na página 14.

SENGUPTA, S. et al. Performance analysis of message brokers for cloud based iot systems. *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, p. 520–525, 2017. Citado na página 23.

SILVA, W. J. A. Um estudo sobre a mediçao do tempo de resposta de servidores web instanciados em uma plataforma global de computaçao em nuvem. *Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA (ISSN 2526-7574)*, n. 5, 2021. Citado na página 87.

SILVEIRA, R. M. da. Liberdade de expressão e discurso do ódio. 2007. Citado na página 13.

SMITH, J.; JOHNSON, A. A comparative study: Undertow vs. tomcat for web application performance. *Journal of Web Engineering*, v. 10, n. 2, p. 45–60, 2022. Citado na página 58.

SMITH, J.; JOHNSON, M. Automated database migrations with flywaydb. *Database Management Review*, v. 25, n. 3, p. 45–60, 2022. Citado na página 24.

SPRING. *Spring Initializr*. 2021. <https://start.spring.io>. Citado na página 21.

SPRINGBOOT. *Spring Boot Reference Documentation*. 2022. Disponível em: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. Citado 3 vezes nas páginas 19, 20 e 21.

THYMELEAF. *Thymeleaf Documentation*. 2022. Disponível em: <https://www.thymeleaf.org/documentation.html>. Citado na página 32.

TONTODIMAMMA, A. et al. *Thirty years of research into hate speech: topics of interest and their evolution*. 2021. Disponível em: <https://link.springer.com/article/10.1007/s11192-020-03737-6>. Citado na página 13.

TST. *Principais ameaças (Key Logger, Screen Logger, Teclado Virtual Falso)*. 2017. Disponível em: <https://www.tst.jus.br/documents/23101476/23248847/Proteja-se+-+21.pdf/8bf0247f-7e74-dd5f-5926-cafa5a20caae>. Citado na página 17.

VASWANI, A. et al. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008. Citado na página 30.

WASEEM, Z. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. 2016. Disponível em: <https://aclanthology.org/W16-5618.pdf>. Citado na página 72.

ZULQARNAIN, M. et al. A comparative review on deep learning models for text classification. *Indones. J. Electr. Eng. Comput. Sci*, v. 19, n. 1, p. 325–335, 2020. Citado na página 73.