Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Desenvolvimento de uma Aplicação Servidora para Coleta Automática de Dados para o Aplicativo de Monitoramento de Crimes BHSafezone em Belo Horizonte

Gustavo Silva da Fonseca

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Alexandre Magno de Sousa

Abril, 2025
João Monlevade–MG

Gustavo Silva da Fonseca

Desenvolvimento de uma Aplicação Servidora para Coleta Automática de Dados para o Aplicativo de Monitoramento de Crimes BHSafezone em Belo Horizonte

Orientador: Alexandre Magno de Sousa

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Abril de 2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Gustavo Silva Fonseca

Desenvolvimento de uma Aplicação Servidora para Coleta Automática de Dados para o Aplicativo de Monitoramento de Crimes BHSafezone em Belo Horizonte

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em 11 de Abril de 2025.

Membros da banca

Doutor - Professor Alexandre Magno de Sousa - Orientador - Universidade Federal de Ouro Preto
Doutor - Professor Carlos Henrique Gomes Ferreira - Universidade Federal de Ouro Preto
Doutora - Professora Helen de Cássia Sousa da Costa Lima - Universidade Federal de Ouro Preto

Professor Alexandre Magno de Sousa, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 02/05/2025.

Documento assinado eletronicamente por **Alexandre Magno de Sousa**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 07/05/2025, às 15:50, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0906143** e o código CRC **E1E0AA20**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado, primeiramente, a Deus, por me dar força e sabedoria para enfrentar os desafios ao longo desta jornada. Aos meus pais, pelo amor, apoio incondicional e incentivo constante, que foram fundamentais para a realização deste projeto.

Agradecimentos

Agradeço primeiramente ao meu orientador, Alexandre Magno, pelo apoio, paciência e orientação ao longo do desenvolvimento deste trabalho, e por não me deixar desistir do TCC nos momentos mais difíceis.

Aos meus amigos e colegas, que de diversas formas contribuíram com sugestões, discussões e apoio durante todo o processo.

A minha família, pelo incentivo e suporte incondicional, essenciais para que eu pudesse concluir esta jornada.

Por fim, agradeço às instituições e fontes de dados que possibilitaram a realização deste estudo, fornecendo informações essenciais para a construção do sistema.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, meus sinceros agradecimentos.

"Science without religion is lame, religion without science is blind."

— Albert Einstein (1879 – 1955)

Resumo

Este trabalho apresenta o desenvolvimento de um sistema de monitoramento de crimes em Belo Horizonte, baseado na coleta automatizada de mensagens de grupos do WhatsApp. Para isso, foi implementado um framework modular utilizando NodeJS com o NestJS, permitindo a integração com bancos de dados MongoDB e Firestore para armazenamento e sincronização eficiente das informações. O sistema é composto por diferentes serviços que realizam desde a filtragem inicial das mensagens até a estruturação e organização dos dados para inserção no banco. A identificação de crimes ocorre em duas etapas: primeiramente, um serviço de filtragem desenvolvido em NodeJS analisa as mensagens e detecta possíveis crimes com base em um dicionário de palavras-chave; em seguida, um modelo de Machine Learning, implementado em Python, realiza a classificação detalhada, garantindo maior precisão na categorização das ocorrências. Para possibilitar análises detalhadas, os dados são organizados hierarquicamente no Firestore, estruturados por regiões, bairros e períodos de tempo, permitindo consultas flexíveis e facilitando o acompanhamento da criminalidade. Além disso, a solução adota uma abordagem de automação para garantir que as mensagens sejam processadas continuamente, sem necessidade de intervenção manual. A arquitetura desenvolvida proporciona escalabilidade e adaptabilidade, tornando o sistema versátil para possíveis expansões e aplicações em outros contextos de monitoramento de dados.

Palavras-chave: Framework. Automação. NestJS. MongoDB. Firestore. Integração. Coleta Automática. Monitoramento de Crimes. WhatsApp.

Abstract

This work presents the development of a crime monitoring system in Belo Horizonte based on the automatic collection of messages from WhatsApp groups. For this purpose, a modular framework was implemented using Node.js with the NestJS framework, integrated with MongoDB and Firestore for efficient data storage and synchronization. The system consists of several services that are responsible for filtering, structuring and organizing the data before it is inserted into the database. Crimes are identified in two steps: (1) first, a filtering service developed in NestJS analyzes the messages and identifies possible crimes based on a predefined dictionary of keywords; then (2) a Machine Learning model implemented in Python classifies the messages in detail, ensuring greater accuracy in the categorization of crimes. To enable detailed analysis, the data in Firestore is organized hierarchically and broken down by region, neighborhood and time period, allowing for flexible queries and facilitating crime monitoring. The system also takes an automated approach to ensure continuous processing of messages without manual intervention. The developed architecture offers scalability and adaptability, making the system versatile for potential extensions and applications in other data monitoring contexts.

Keywords: Framework. Automation. NestJS. MongoDB. Firestore. Integration. Automatic Data Collection. Crime Monitoring. WhatsApp.

Lista de ilustrações

Lista de abreviaturas e siglas

API Application Programming Interface

URL Uniform Resource Locator

LGPD Lei Geral de Proteção de Dados

PLN Processamento de linguagem natural

SVM Support Vector Machine

TF-IDF Term Frequency–Inverse Document Frequency

REM Reconhecimento de Entidades Mencionadas

BH Belo Horizonte

Sumário

1 Introdução

O crescente aumento da criminalidade em grandes centros urbanos exige a adoção de tecnologias para o monitoramento e análise de padrões de segurança. Diversos estudos têm explorado a utilização de mídias sociais e outras fontes abertas como ferramentas de coleta de dados para auxiliar na análise de comportamentos criminosos e no fornecimento de informações em tempo real para autoridades de segurança pública. Silva e Stabile (2016) destacam o uso de plataformas digitais para o monitoramento de segurança, abordando técnicas de análise de dados extraídos de mídias sociais como uma forma eficaz de detectar e mapear atividades suspeitas em tempo real. Complementando essa abordagem, Alves, Ribeiro e Rodrigues (2020) propõem o uso de métricas urbanas e técnicas de aprendizado estatístico para prever crimes, integrando dados de múltiplas fontes, incluindo mídias sociais, para melhorar a precisão das previsões.

A extração automatizada de informações de fontes abertas é uma área amplamente explorada, como abordado por Silva, Duarte e Ugulino (2022), que apresentam técnicas para a extração de estatísticas de segurança pública a partir de dados de mídias digitais. Tais abordagens têm sido fundamentais para melhorar a tomada de decisões no planejamento de ações preventivas e reativas. Complementarmente, Kumar e Singh (2022) enfatizam a importância de algoritmos de deep learning para a análise de grandes volumes de dados em tempo real, destacando sua aplicação na detecção de padrões criminais em plataformas como Twitter e WhatsApp.

Teles e Silva (2021) investigam a coleta de dados utilizando técnicas de web scraping, uma prática fundamental para a obtenção de informações de diversas fontes digitais. Essas abordagens são essenciais para a criação de bancos de dados atualizados, os quais são posteriormente utilizados para análise de padrões e eventos de segurança pública. A utilização dessas técnicas de coleta automatizada foi igualmente explorada por Chen e Liu (2023), que demonstram como o web scraping pode ser aplicado de maneira eficiente para o monitoramento de plataformas sociais e sites de notícias, com o objetivo de detectar incidentes relacionados à criminalidade em grandes cidades.

Por fim, Zandavalle (2016) analisa a opinião pública e como sua interpretação pode influenciar a percepção da segurança em diferentes regiões. A análise de sentimentos tem sido amplamente aplicada para compreender a opinião das pessoas sobre temas como violência e segurança pública, com contribuições importantes de Ribeiro e Almeida (2023) no campo da análise de dados textuais para a interpretação de tendências sociais. As ferramentas de análise de opinião pública, como as propostas por Lima e Costa (2023), permitem uma melhor compreensão dos sentimentos da população e podem ser utilizadas

para aprimorar os sistemas de monitoramento de segurança.

Dessa forma, o presente trabalho busca integrar as metodologias mencionadas, propondo uma solução que combine a coleta de dados em tempo real, a análise de sentimentos e o uso de técnicas avançadas de Machine Learning para a classificação de informações, com o objetivo de fornecer uma ferramenta eficiente no monitoramento de crimes em Belo Horizonte. A integração de novas fontes de dados, como grupos do WhatsApp, Telegram e sites de notícias locais, aliada a técnicas de Processamento de linguagem natural (PLN) e aprendizado de máquina, permitirá a criação de um sistema robusto e atualizado, capaz de fornecer insights valiosos para a segurança pública.

1.1 Motivação e Justificativa

A segurança pública é um dos maiores desafios enfrentados por cidades de grande porte, como Belo Horizonte. O aumento dos índices de criminalidade e a crescente demanda por soluções eficazes para o monitoramento de crimes exigem a utilização de tecnologias avançadas. Nesse contexto, a aplicação de metodologias para o processamento de dados em tempo real provenientes de fontes abertas, como redes sociais, mensageiros instantâneos e sites de notícias, surge como uma solução inovadora. A motivação principal deste trabalho é contribuir para o desenvolvimento de um sistema inteligente de monitoramento e mapeamento de crimes, utilizando dados extraídos de plataformas digitais para detectar, classificar e disseminar informações sobre incidentes de segurança.

Diversos estudos indicam que o uso de mídias sociais tem se mostrado uma ferramenta poderosa para o monitoramento de atividades criminosas. Silva e Stabile (2016) destacam que as redes sociais oferecem uma grande quantidade de dados em tempo real, que podem ser analisados para identificar padrões de comportamento relacionados à criminalidade. A extração automatizada de informações, como proposto por Silva, Duarte e Ugulino (2022), permite não só a coleta de dados, mas também a geração de estatísticas e relatórios de segurança que podem subsidiar ações mais eficientes das autoridades responsáveis. Complementando essa abordagem, Alves, Ribeiro e Rodrigues (2020) propõem o uso de métricas urbanas e técnicas de aprendizado estatístico para prever crimes, integrando dados de múltiplas fontes, incluindo mídias sociais, para melhorar a precisão das previsões. Assim, a utilização dessas tecnologias no contexto de Belo Horizonte é especialmente relevante, considerando a crescente preocupação com a segurança pública e a necessidade de aprimorar as estratégias de prevenção e combate ao crime.

Além disso, a aplicação de técnicas de web scraping, abordada por Teles e Silva (2021), e o uso de análise de sentimentos, conforme descrito por Zandavalle (2016), permite uma compreensão mais profunda da percepção pública em relação à segurança e violência na cidade. Ao analisar mensagens de grupos de WhatsApp, Telegram e notícias de mídia,

é possível identificar rapidamente áreas de risco e fornecer informações críticas em tempo hábil para as autoridades locais. A motivação central deste trabalho é utilizar essas metodologias para criar um sistema ágil e eficiente que possa fornecer dados atualizados sobre ocorrências de crimes, alimentando um banco de dados que permita a visualização em tempo real e o mapeamento de incidentes.

A justificativa para o desenvolvimento deste projeto está no fato de que, apesar dos avanços na utilização de tecnologias para a segurança pública, ainda há uma lacuna significativa no uso de dados extraídos de plataformas digitais, especialmente para o monitoramento de crimes em tempo real em cidades específicas como Belo Horizonte. A proposta deste trabalho visa preencher essa lacuna, oferecendo uma solução que combine as melhores práticas de coleta e análise de dados com o poder de técnicas de Machine Learning para a classificação de informações relacionadas à criminalidade. Com isso, pretende-se não apenas melhorar a eficácia do monitoramento de crimes, mas também fornecer ferramentas que ajudem na tomada de decisões mais assertivas por parte das autoridades públicas, promovendo um ambiente urbano mais seguro para a população. Estudos recentes, como o de Chen e Liu (2023), demonstram a eficácia de técnicas de aprendizado de máquina para prever hotspots de criminalidade, enquanto Ribeiro e Almeida (2023) destacam a importância da análise de sentimentos em redes sociais para a detecção de eventos de segurança pública. Essas abordagens reforçam a relevância e a viabilidade do presente trabalho.

1.2 Definição do Problema

A crescente onda de violência e criminalidade nas grandes cidades brasileiras, como Belo Horizonte, tem gerado uma pressão cada vez maior sobre as autoridades de segurança pública para o desenvolvimento de soluções mais eficientes no monitoramento de atividades criminosas e no planejamento de estratégias de prevenção. Tradicionalmente, os sistemas de monitoramento de crimes dependem de informações provenientes de registros policiais, câmeras de segurança e outros meios convencionais. Contudo, esses sistemas têm limitações, como o tempo de resposta e a falta de dados em tempo real, que podem comprometer a eficácia das ações de segurança (ALVES; RIBEIRO; RODRIGUES, 2020).

Nos últimos anos, o uso de plataformas digitais, como redes sociais e mensageiros instantâneos, tem se mostrado uma fonte promissora de dados em tempo real sobre a percepção pública e a ocorrência de crimes. No entanto, a coleta, o processamento e a análise desses dados de forma eficiente ainda representam um grande desafio. As informações compartilhadas nessas plataformas, como posts em redes sociais e mensagens em grupos de WhatsApp e Telegram, podem fornecer detalhes sobre atividades criminosas ou regiões de risco, mas são difíceis de serem interpretadas de maneira rápida e confiável sem o uso

de tecnologias avançadas de análise de dados (SILVA; DUARTE; UGULINO, 2022).

A principal dificuldade reside na falta de ferramentas eficazes para realizar a extração automatizada de dados, a identificação de conteúdos relevantes, a classificação precisa de informações e a atualização em tempo real de bancos de dados de segurança. Além disso, a necessidade de distinguir entre informações verdadeiras e falsas (fake news) também representa um desafio significativo, pois o volume de dados disponíveis nessas plataformas é massivo e muitas vezes contém conteúdo irrelevante ou impreciso (RIBEIRO; ALMEIDA, 2023).

Portanto, o problema central deste trabalho é como integrar e analisar dados de fontes digitais, como redes sociais, WhatsApp e Telegram, para fornecer informações relevantes sobre crimes em tempo real em Belo Horizonte. A solução proposta busca superar as limitações dos métodos tradicionais de monitoramento, utilizando técnicas de web scraping, análise de sentimentos e Machine Learning para detectar e classificar mensagens relacionadas a crimes. Isso permitirá que as autoridades de segurança pública tenham acesso a dados mais precisos e rápidos, contribuindo para ações mais eficazes no combate à criminalidade e na prevenção de novos incidentes (CHEN; LIU, 2023).

Estudos recentes, como o de Kumar e Singh (2022), demonstram que a aplicação de técnicas de deep learning pode melhorar significativamente a precisão na detecção de padrões criminais em grandes volumes de dados. Além disso, Lima e Costa (2023) destacam a importância da integração de dados heterogêneos para o monitoramento de segurança urbana, reforçando a relevância da abordagem proposta neste trabalho.

1.3 Objetivos Gerais e Específicos

O objetivo geral deste trabalho é desenvolver um sistema automático para monitoramento e mapeamento de crimes em Belo Horizonte, utilizando dados provenientes de grupos de WhatsApp, com o objetivo de identificar, classificar e fornecer informações atuais sobre ocorrências criminosas, contribuindo para a melhoria das estratégias de segurança pública. Para alcançar o objetivo geral deste trabalho, os seguintes objetivos específicos foram definidos:

1. Implementar um sistema de coleta de dados automatizada: utilizar técnicas de *web scraping* para extrair informações de fontes digitais, como mensagens de grupos WhatsApp; garantir que a coleta de dados seja contínua e em tempo real, permitindo a atualização constante do banco de dados;

2. Desenvolver um mecanismo de filtragem de conteúdo: criar um serviço que utilize um dicionário de palavras-chave relacionadas a crimes para filtrar mensagens relevantes;

implementar uma solução eficiente para identificar automaticamente menções a tipos de crimes, como furtos, roubos, homicídios e outros incidentes de segurança;

3. Aplicar técnicas de classificação de texto para distinguir entre mensagens verdadeiras sobre crimes e mensagens irrelevantes ou falsas: utilizar o classificador originalmente implementado para o trabalho de conclusão de curso *Criação de um Aplicativo para Mapeamento da Criminalidade da Cidade de Belo Horizonte por meio de Atividade Crowdsourcing no Twitter* Patrício (2023), adaptando-o para processar mensagens coletadas do WhatsApp;

4. Adaptar o método de extração de localização utilizado no trabalho de conclusão de curso de Guilherme Silva Patrício Patrício (2023), ajustando-o para processar as mensagens coletadas do WhatsApp e associar a localização às ocorrências classificadas como crimes;

5. Integrar os dados coletados e classificados com um sistema de mapeamento de crimes: integrar a uma interface visual para o mapeamento de crimes em tempo real, permitindo que as autoridades de segurança pública monitorem facilmente as ocorrências e identifiquem áreas de risco; garantir que as informações estejam disponíveis de forma intuitiva, com filtros por tipo de crime e localização;

6. Validar a precisão e eficácia do sistema: realizar testes e comparações entre os dados coletados pelo sistema e as informações oficiais, avaliando a acurácia na identificação de crimes e a relevância das informações fornecidas e ajustar o sistema conforme necessário, garantindo a melhoria contínua da coleta de dados, filtragem e classificação.

1.4 Resultados e Contribuições

Este trabalho resultou no desenvolvimento de uma aplicação servidora altamente eficiente, projetada para coletar, classificar e armazenar mensagens de grupos do WhatsApp de forma totalmente automatizada. A solução proposta fornece dados essenciais para um sistema de monitoramento de crimes em Belo Horizonte, oferecendo uma alternativa inovadora às abordagens anteriores que dependiam da extração manual ou de fontes de dados menos dinâmicas. O sistema desenvolvido garante um fluxo contínuo de informações, permitindo atualizações em tempo real e proporcionando uma visão mais precisa e oportuna dos eventos relacionados à segurança pública.

Uma das principais contribuições deste projeto foi a reestruturação do framework, o que o tornou mais modular, flexível e independente de intervenções manuais. Esse aprimoramento permitiu a automação completa de todas as etapas do processo — desde a coleta das mensagens até a classificação dos crimes e o armazenamento dos dados. Com

essa autonomia, o sistema se tornou mais escalável, robusto e confiável, estabelecendo uma base sólida para futuras expansões e integrações, sem a necessidade de ajustes ou execuções manuais constantes.

Outro avanço significativo foi a adaptação do código de classificação, que se mostrou altamente versátil ao se ajustar facilmente a diferentes contextos. Esse aprimoramento permitiu que o processo de classificação, inicialmente projetado para dados de redes sociais como o X (Twitter), fosse adaptado para os grupos de WhatsApp, que apresentam características distintas, como um volume de dados menor e menos estruturado. A flexibilidade do código foi fundamental para garantir a precisão e a relevância na classificação das mensagens, demonstrando sua eficácia ao lidar com diferentes fontes de informação.

Além disso, a integração do banco de dados Firestore foi um ponto crucial para o sucesso da solução. A utilização do Firestore permitiu não só o armazenamento eficiente dos dados, mas também a recuperação rápida e contínua das informações em tempo real. Essa integração garantiu que o aplicativo BHSafezone (PATRíCIO, 2023) tivesse acesso a dados sempre atualizados, refletindo as ocorrências mais recentes e proporcionando uma visão mais dinâmica e precisa da criminalidade na cidade.

Dessa forma, o presente trabalho contribui significativamente para a área de monitoramento de crimes, propondo uma solução inovadora, automatizada e adaptável a diferentes fontes de dados. O sistema desenvolvido oferece uma base sólida para futuras pesquisas e aprimoramentos, ampliando o impacto positivo do uso de tecnologias na segurança pública e no monitoramento de ocorrências criminosas.

1.5 Estrutura da Monografia

A monografia está estruturada de forma a apresentar de maneira clara e objetiva o desenvolvimento e os resultados obtidos com o sistema de monitoramento de crimes em Belo Horizonte, utilizando dados de grupos públicos de WhatsApp. A estrutura está organizada da seguinte forma:

O próximo capítulo apresenta a Fundamentação Teórica, discutindo os principais conceitos e estudos relacionados ao monitoramento de crimes, ao uso de redes sociais para coleta de dados, ao Processamento de Linguagem Natural (PLN) e às técnicas de aprendizado de máquina aplicadas à análise de texto. Essa revisão serve como base teórica para a construção da metodologia e para contextualizar a relevância do tema. Em seguida, a Metodologia é descrita detalhadamente, abordando os métodos de coleta de dados, o pré-processamento das mensagens, a aplicação do filtro de palavras-chave e a classificação das mensagens por meio de técnicas de aprendizado de máquina. Também são apresentadas as ferramentas e tecnologias utilizadas, além das questões éticas e legais envolvidas no uso dos dados. Na sequência, os Resultados e Contribuições são expostos, analisando a

precisão do modelo de classificação, a integração com o sistema de mapeamento de crimes e a validação dos dados. As contribuições do estudo para a segurança pública e para o PLN são enfatizadas. Por fim, o capítulo de Conclusões e Trabalhos Futuros sintetiza os principais resultados, destacando as limitações da pesquisa e sugerindo possíveis direções para aprimoramentos, como melhorias no sistema de classificação, expansão para outras plataformas de redes sociais e integração com sistemas de segurança pública em larga escala.

2 Revisão da Literatura

Este capítulo apresenta uma revisão da literatura sobre o uso de redes sociais e aplicativos de mensagens como fontes de dados para monitoramento de eventos do mundo real. Além disso, discute sistemas e aplicações semelhantes, bem como trabalhos correlatos que fundamentam este estudo.

2.1 Monitoramento e Previsão de Eventos do Mundo Real em Mídias Sociais Online

O monitoramento de redes sociais tem se consolidado como uma ferramenta indispensável para a detecção, análise e compreensão de eventos do mundo real, abrangendo desde crises humanitárias até desastres naturais e ocorrências criminais Silva e Stabile (2016). A popularização dessas plataformas e o grande volume de dados gerados diariamente pelos usuários possibilitam a extração de informações valiosas em tempo real, permitindo identificar padrões, tendências e até prever certos tipos de eventos Teles e Silva (2021). Essa capacidade de análise em larga escala tem despertado o interesse de pesquisadores e autoridades em diversas áreas, incluindo segurança pública, gestão de crises e políticas urbanas. Estudos recentes indicam que, ao utilizar técnicas de mineração de dados e processamento de linguagem natural, é possível transformar postagens e interações em redes sociais em fontes de informação relevantes para a tomada de decisões em situações emergenciais Wang e Taylor (2020). No contexto de desastres naturais, por exemplo, sistemas baseados na coleta e análise de dados dessas plataformas já foram empregados para monitorar o impacto de terremotos, enchentes e incêndios florestais, auxiliando no direcionamento de recursos e na comunicação com a população afetada Chen, Liu e Zhang (2021). Além disso, a análise automatizada de grandes volumes de postagens tem sido utilizada para avaliar o sentimento público em relação a eventos críticos, permitindo uma resposta mais ágil e eficiente por parte das autoridades e da sociedade civil.

Aplicativos de comunicação instantânea, como WhatsApp e Telegram, também desempenham um papel significativo na disseminação de informações sobre eventos emergentes, muitas vezes funcionando como fontes primárias de relatos em tempo real. Diferentemente das redes sociais abertas, essas plataformas possibilitam a comunicação direta entre usuários por meio de mensagens privadas ou em grupos fechados, o que contribui para uma rápida propagação de informações, especialmente em contextos de crise Silva, Duarte e Ugulino (2022). No entanto, essa característica também apresenta desafios significativos para a coleta, análise e verificação dos dados compartilhados, uma vez que o conteúdo

trafegado nesses aplicativos não é indexado publicamente e muitas mensagens podem ser imprecisas, exageradas ou até falsas. A falta de transparência e a dificuldade de acesso a essas informações tornam mais complexa a utilização desses dados para monitoramento e tomada de decisão.

Pesquisas indicam que a combinação de múltiplas fontes de dados – incluindo redes sociais abertas, notícias verificadas e informações extraídas de grupos privados – pode aumentar a confiabilidade na detecção de eventos Zandavalle (2016). Essa abordagem integrada permite mitigar os riscos associados à desinformação e melhorar a precisão das análises. Além disso, técnicas como a análise de sentimentos e a detecção de padrões linguísticos vêm sendo amplamente utilizadas para compreender a percepção pública sobre questões de segurança e criminalidade, auxiliando na identificação de áreas com maior incidência de ocorrências e na formulação de políticas públicas mais eficazes Ribeiro e Almeida (2023). Dessa forma, o estudo e a implementação de sistemas capazes de coletar e processar informações provenientes de aplicativos de mensagens instantâneas tornam-se cada vez mais relevantes para o monitoramento de eventos críticos e para a segurança pública.

2.2 Ferramentas Semelhantes

Diferentes aplicações já foram desenvolvidas com o objetivo de monitorar e mapear ocorrências criminais, utilizando dados de redes sociais, sistemas colaborativos e registros oficiais. Essas ferramentas desempenham um papel importante na segurança pública, permitindo que a população tenha acesso a informações relevantes sobre crimes em suas localidades. Entre os exemplos mais significativos, destacam-se:

O *Radar do Roubo* Oliveira e Lopes (2022) é uma plataforma colaborativa que permite que vítimas de roubos e furtos registrem e compartilhem a localização dos crimes, disponibilizando um mapa interativo em tempo real. De forma semelhante, o portal *Onde Fui Roubado* Ondefuiroubado.com.br (2023) oferece um espaço onde usuários podem relatar crimes e visualizar estatísticas de segurança, fornecendo uma visão detalhada sobre as áreas mais afetadas. O *CrimeSpotter* Advertising (2023), por sua vez, permite a visualização de crimes em diversas regiões e alerta os usuários sobre áreas potencialmente perigosas. No entanto, essas soluções dependem majoritariamente das informações fornecidas pelos usuários, o que pode resultar em dados incompletos ou imprecisos.

Além dessas iniciativas, algumas ferramentas utilizam registros criminais de órgãos oficiais para mapear a criminalidade. O *E-Roubo Celular SP* AkumaEX (2020), por exemplo, exibe um mapa com os locais de roubos de celulares na cidade de São Paulo, com base nos boletins de ocorrência da Secretaria da Segurança Pública. No entanto, a atualização mensal dos dados pode comprometer a identificação de padrões em tempo real.

De forma similar, o *Crime Map* LTD (2023) utiliza registros policiais para exibir um mapa interativo da criminalidade, possibilitando o acompanhamento da quantidade e dos tipos de crimes ocorridos em determinada região. Entretanto, a dependência de informações oficiais pode impactar a frequência e a abrangência das atualizações.

Outras abordagens incluem o uso de redes sociais e participação colaborativa para o monitoramento da criminalidade. O *Fogo Cruzado* Cruzado (2022) é um exemplo relevante nesse contexto, pois coleta e analisa informações sobre tiroteios enviadas por usuários, utilizando geolocalização para mapear os incidentes em tempo real. Já o *Citizen* Citizen (2021) é amplamente utilizado nos Estados Unidos e fornece alertas instantâneos sobre incidentes, extraindo informações de redes sociais e chamadas de emergência. Além disso, permite que os próprios usuários reportem eventos, criando uma rede colaborativa de monitoramento. O *SpotCrime* SpotCrime (2021), por sua vez, combina dados reportados por usuários e agências de segurança pública para exibir um mapa interativo, oferecendo alertas personalizados de acordo com a localização do usuário.

Esses aplicativos demonstram a viabilidade de soluções que combinam participação social e análise automatizada para o monitoramento da criminalidade. No entanto, a maioria depende de dados estruturados de fontes oficiais ou relatos diretos dos usuários, o que pode limitar a precisão e a atualização em tempo real das informações. Nesse contexto, este trabalho propõe uma abordagem alternativa, baseada na extração automatizada de dados de redes sociais e aplicativos de mensagens, ampliando as possibilidades de análise e fornecendo um monitoramento mais dinâmico e abrangente da criminalidade.

2.3 Trabalhos Relacionados

Diversos estudos exploram métodos para extração e análise de dados relacionados à segurança pública. Silva e Stabile (2016) propõem um sistema de monitoramento de mídias sociais para detecção de eventos criminais com base em padrões linguísticos e geolocalização, demonstrando a eficácia da análise de texto em redes sociais para identificar ocorrências de crimes. O sistema proposto utiliza técnicas básicas de PLN para identificar palavras-chave e termos recorrentes associados a crimes, sendo capaz de indicar, com base em termos específicos, possíveis ocorrências em determinadas regiões. No entanto, a coleta dos dados ainda exigia um certo grau de intervenção manual ou scripts pouco escaláveis.

No estudo de Silva, Duarte e Ugulino (2022), é apresentado um modelo para extração automatizada de estatísticas de segurança pública a partir de fontes informais, como redes sociais e fóruns comunitários, destacando-se a importância da integração de múltiplas fontes de dados para aumentar a precisão das análises. O trabalho faz uso de algoritmos de mineração de texto e um pipeline automatizado simples, focando mais na extração de métricas do que na coleta contínua dos dados. Ainda assim, a proposta carece

de uma estrutura robusta de automação da coleta em tempo real e de escalabilidade para múltiplos canais simultâneos.

Teles e Silva (2021) investigam o uso de técnicas de web scraping para coleta de dados de segurança pública em fontes não estruturadas, propondo uma abordagem eficiente para a extração de informações relevantes de plataformas digitais. O estudo utiliza ferramentas como Python, BeautifulSoup e Selenium para acessar sites de delegacias e fóruns comunitários. Apesar da eficiência do método, a abordagem está limitada a páginas web e exige ajustes constantes para manter a coleta funcional frente a mudanças nas estruturas das páginas-alvo.

Já Zandavalle (2016) analisa o impacto da opinião pública nas redes sociais na percepção da criminalidade, ressaltando a importância da análise de sentimentos para compreender as tendências de segurança em áreas urbanas. O trabalho aplica técnicas de PLN e análise de sentimentos com o objetivo de mapear a percepção da população sobre a criminalidade, mas não atua diretamente na coleta automatizada de dados, utilizando dados já disponíveis.

Outros estudos exploram aplicações semelhantes em diferentes contextos. Wang e Taylor (2020) investigam o uso de redes sociais para monitoramento de desastres naturais, evidenciando a eficácia de técnicas de PLN para a detecção de eventos emergentes. Utilizando ferramentas como o Twitter API e bibliotecas como NLTK, o estudo alcança bons resultados na detecção rápida de eventos, mas também depende de uma infraestrutura de coleta limitada a uma única plataforma.

Da mesma forma, Chen, Liu e Zhang (2021) analisam a utilização de dados de redes sociais para o monitoramento de crises de saúde pública, como a pandemia de COVID-19, demonstrando a viabilidade da análise de texto para identificar tendências e comportamentos sociais. O estudo utiliza aprendizado de máquina supervisionado para classificar as mensagens, mas também se baseia em conjuntos de dados previamente obtidos, sem foco em coleta contínua.

Por fim, Ribeiro e Almeida (2023) propõem um sistema de análise de sentimentos em redes sociais para detecção de eventos de segurança pública, utilizando técnicas de aprendizado de máquina para classificar mensagens relacionadas a crimes. Embora o sistema apresente bons resultados de acurácia na classificação de sentimentos e temas, ele depende da existência de uma base de dados previamente estruturada, e não oferece uma solução integrada para a coleta, filtragem e armazenamento dos dados.

Em contraste com esses trabalhos, o presente projeto propõe um sistema completo e altamente automatizado de coleta de dados em tempo real, com foco principal em mensagens oriundas de grupos do WhatsApp, uma fonte ainda pouco explorada academicamente. A aplicação servidora foi desenvolvida utilizando NestJS, com armazenamento em bancos

como MongoDB e Firestore, permitindo flexibilidade, escalabilidade e sincronização com sistemas externos, como um aplicativo de mapeamento de crimes.

Diferente das abordagens anteriores, que geralmente dependem de plataformas abertas (como Twitter, fóruns ou páginas públicas), o sistema desenvolvido neste trabalho é capaz de coletar mensagens automaticamente, em tempo real, mesmo de ambientes sem APIs públicas, como o WhatsApp. Essa característica representa um avanço importante, especialmente na automação da coleta, ao permitir um monitoramento contínuo e descentralizado de relatos sobre crimes, melhorando a atualização da base de dados e reduzindo significativamente a necessidade de intervenção manual.

2.4 Considerações Finais

Este capítulo apresentou um panorama teórico sobre a importância do monitoramento de eventos em redes sociais e aplicativos de mensagens, bem como exemplos de sistemas existentes e trabalhos acadêmicos que embasam esta pesquisa. A revisão de literatura demonstra a relevância do tema e destaca a necessidade de desenvolver ferramentas mais eficazes para coleta, filtragem e análise automatizada dessas informações. Nos próximos capítulos, serão detalhadas a metodologia e as soluções propostas para enfrentar esses desafios.

3 Metodologia e Desenvolvimento

O objetivo deste trabalho é o desenvolvimento de um sistema para monitoramento de crimes em Belo Horizonte, utilizando dados provenientes de fontes como WhatsApp. Inicialmente, a proposta envolvia a coleta de tweets geolocalizados de usuários da cidade para mapear a ocorrência de crimes. Contudo, após mudanças na plataforma do X/Twitter, não foi possível dar continuidade ao desenvolvimento de coletas na plataforma, sendo assim necessitando uma nova fonte de coleta.

Diante desse cenário, o foco do projeto foi ampliado, incluindo fontes de dados alternativas, como grupos de WhatsApp. Foi desenvolvido um sistema que automatiza a coleta, processamento e classificação de dados relacionados a crimes, utilizando técnicas de filtragem e Machine Learning para identificar e classificar ocorrências criminais. A arquitetura do sistema foi reestruturada para garantir que ele operasse de forma autônoma, sem a necessidade de intervenção manual para rodar os scripts.

Neste capítulo, são apresentadas as etapas de desenvolvimento do sistema, abrangendo o estudo das fases de implementação, a reestruturação do framework, a otimização do código e a integração das fontes de dados. Além disso, é detalhado o fluxo das mensagens, descrevendo cada etapa pelo qual uma mensagem passa dentro do sistema. Também são abordados os processos de busca e identificação de grupos no WhatsApp, o processo da coleta de dados em tempo real, filtro por palavras relacionadas e o processamento das mensagens para a classificação de crimes. Por fim, discute-se a integração do sistema de monitoramento com a base de dados e a visualização das informações em tempo real, concluindo a descrição do desenvolvimento do sistema.

A metodologia adotada neste trabalho será estruturada para o desenvolvimento de um sistema de monitoramento de crimes em Belo Horizonte, utilizando exclusivamente dados extraídos de grupos públicos de WhatsApp. O processo será dividido em várias etapas, que incluem desde o estudos de cada etapa até a validação do sistema, com foco na identificação de mensagens relacionadas a crimes e sua classificação.

3.1 Tecnologias utilizadas

A seguir são descritas as ferramentas e tecnologias utilizadas no desenvolvimento do aplicativo proposto por este trabalho e suas respectivas finalidades de uso:

1. Git[1]: utilizado para manter o registro das alterações feitas no código-fonte do

[1] <https://git-scm.com/doc>.

aplicativo.

2. GitHub[2]: utilizado para o controle de versões e hospedagem do código-fonte do projeto;

3. MongoDB[3]: utilizado para armazenar as mensagens coletadas dos grupos e os demais arquivos gerados a partir dos filtros, classificação e análises;

4. Linguagem Python[4]: utilizado para a classificação das mensagens coletadas e para a busca da localização mencionada nos textos;

5. Node.js[5]: utilizado como ambiente de execução para o desenvolvimento do backend do sistema;

6. NestJS[6]: utilizado como framework para a construção da API do sistema, permitindo uma estrutura modular e escalável;

7. Cloud Firestore[7]: utilizado para armazenar os dados de criminalidade os quais alimentam as informações exibidas no aplicativo.

Para a execução do serviço de coleta e processamento dos dados, foi utilizado um notebook Dell Inspiron 14, com as seguintes especificações técnicas:

- Processador: Intel Core i5 de 5ª geração;
- Memória RAM: 8GB;
- Placa de Vídeo: NVIDIA GeForce 620 dedicada;
- Sistema Operacional: Zorin OS.

3.2 Estudo das Etapas

O estudo das etapas envolveu uma análise detalhada sobre como o sistema seria desenvolvido de forma eficiente e autônoma. Inicialmente, foi conduzido um levantamento aprofundado sobre possíveis fontes de dados que pudessem fornecer informações relevantes em tempo real, com mínima ou nenhuma necessidade de intervenção manual. Entre as alternativas exploradas, foram considerados grupos públicos no Telegram, sites de jornais de notícias e a antiga API do Twitter. No entanto, todas essas opções apresentaram

[2] <https://docs.github.com/pt>.
[3] <https://www.mongodb.com/docs/>.
[4] <https://www.python.org/doc/>.
[5] <https://nodejs.org/en/docs/>.
[6] <https://docs.nestjs.com/>.
[7] <https://firebase.google.com/docs/firestore?hl=pt-br>.

limitações técnicas, econômicas ou estruturais que inviabilizaram sua utilização no escopo deste projeto.

A busca por grupos no Telegram, por exemplo, mostrou-se pouco frutífera. Apesar da plataforma permitir acesso mais aberto aos dados por meio de bots e bibliotecas específicas, a dificuldade esteve em localizar grupos que tivessem, de forma contínua e consistente, o compartilhamento de informações relacionadas à criminalidade. A maioria dos grupos encontrados estava inativa, ou possuía conteúdos não relacionados ao tema, o que comprometeria a qualidade e a efetividade da coleta automatizada.

Também foi cogitada a coleta de dados diretamente em sites de jornais locais. No entanto, essa abordagem apresentou diversos obstáculos técnicos. As páginas dos portais de notícia geralmente não seguem um padrão único e estão em constante reformulação, dificultando a construção de um scraper estável. Tecnologias como JavaScript dinâmico, lazy loading, e alterações frequentes no layout dos sites tornaram o processo frágil e de difícil manutenção. Além disso, o conteúdo relevante está muitas vezes fragmentado entre diferentes páginas ou embutido em elementos difíceis de rastrear automaticamente, o que exigiria grande esforço técnico e constante atualização do código de coleta.

A API do Twitter, que já foi uma fonte amplamente utilizada para coleta de dados sociais, tornou-se inviável após a mudança de política da empresa, que passou a cobrar valores elevados pelo acesso. Atualmente, o plano mais acessível (Basic) custa US$ 100 por mês, limitando severamente o número de requisições e dificultando a construção de um sistema contínuo e confiável de coleta. Para uma aplicação com o objetivo de auxiliar a população na prevenção de crimes e contribuir com a segurança pública, esse custo é incompatível com a proposta de acessibilidade e utilidade social.

Diante dessas dificuldades, optou-se pelo uso do WhatsApp como principal fonte de dados. Essa escolha foi motivada pelo fato de que, em diversos contextos brasileiros, o WhatsApp é uma das ferramentas de comunicação mais utilizadas pela população, inclusive para o compartilhamento de ocorrências em tempo real, como crimes, acidentes e situações de risco. Estudos recentes, como o de Pires et al. (2023), apontam o WhatsApp como um canal fundamental para o monitoramento de eventos do mundo real, sobretudo em comunidades que utilizam grupos públicos para alertas e mobilização social.

Além disso, o uso de grupos do WhatsApp apresenta vantagens técnicas importantes: os dados coletados são geralmente estruturados em mensagens de texto curtas e diretas, muitas vezes acompanhadas de localização, imagens ou relatos imediatos de testemunhas. Essa característica torna o processo de análise e classificação mais eficiente e com maior potencial de utilidade para o sistema proposto.

Portanto, a escolha do WhatsApp não foi arbitrária, mas sim resultado de uma análise comparativa entre diversas alternativas, levando em consideração aspectos técnicos,

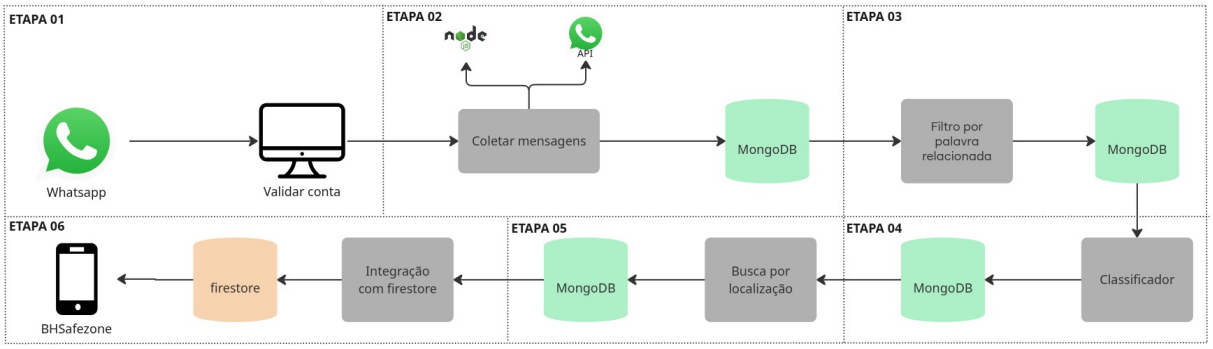


Figura 1 – Desenho do framework do sistema para monitoramento de crimes em Belo Horizonte.

sociais e econômicos. A implementação do sistema de coleta sobre essa plataforma representa um avanço significativo em relação a outros trabalhos da área, que em sua maioria ainda dependem de fontes abertas ou dados previamente disponíveis, não explorando o potencial da coleta automatizada e contínua em ambientes fechados como o WhatsApp.

3.3 Reestruturação do Framework

A reestruturação do framework foi uma etapa essencial para garantir que o sistema pudesse operar de maneira autônoma, eliminando a necessidade de intervenção manual para a execução dos scripts. Inicialmente, o sistema possuía um fluxo de trabalho mais rígido, exigindo a execução manual de certas etapas para coleta e processamento dos dados. Com o estudo detalhado das etapas envolvidas, foi possível redesenhar sua arquitetura para otimizar a integração das fontes de dados externas, permitindo que o processamento e a classificação das informações ocorressem de forma contínua e automatizada.

A nova estrutura do framework foi projetada para refletir a interação entre os diferentes módulos do sistema. Como ilustrado na Figura 1, o fluxo de dados inicia-se com a coleta das mensagens em tempo real, seguida pelo processamento dessas informações, que inclui a filtragem de palavras-chave relacionadas a crimes e a classificação automática dos dados. Em seguida, as informações estruturadas são armazenadas no banco de dados e integradas ao sistema de monitoramento, garantindo que os usuários tenham acesso a dados atualizados e confiáveis.

A modularidade do novo framework também trouxe benefícios significativos para a escalabilidade do sistema. Com essa abordagem, tornou-se possível adicionar novas fontes de dados sem comprometer a estrutura existente, além de permitir ajustes e melhorias em componentes específicos conforme necessário. Essa flexibilidade é fundamental para garantir a evolução contínua do sistema, permitindo que ele se adapte a novas demandas e tecnologias sem a necessidade de reestruturações drásticas.

3.4 Busca e Identificação de Grupos no WhatsApp

A busca e identificação dos grupos no WhatsApp foi uma etapa essencial para a coleta de dados relacionados a crimes em Belo Horizonte. O processo de busca foi realizado em duas frentes principais: a primeira envolveu a pesquisa de grupos no Google, enquanto a segunda consistiu em uma busca na plataforma X/Twitter.

Inicialmente, foi realizada uma busca no Google por grupos de notícias sobre Belo Horizonte, com foco naqueles que pudessem fornecer informações relevantes sobre ocorrências de crimes na cidade. A partir dessa pesquisa, foram identificados os grupos *BHAZap* e *Por Dentro de Minas*, que se mostraram promissores devido ao seu conteúdo relacionado a acontecimentos da cidade, incluindo notícias sobre segurança e criminalidade. Para garantir a qualidade e a confiabilidade das informações, foi adotado o critério de selecionar apenas grupos em que o administrador fosse o único responsável por enviar mensagens, evitando a dispersão de informações irrelevantes.

A segunda parte da busca envolveu a utilização do X/Twitter, onde foram pesquisados grupos de notícias sobre Belo Horizonte, com o intuito de encontrar comunidades ativas na discussão de questões relacionadas à segurança pública. Nessa busca, foram identificados os grupos *DeFato*, *Agito Mais*, *JCO* e *JCA*, que também apresentavam discussões relevantes sobre os temas de interesse. Assim como na busca no Google, os grupos selecionados seguiam o critério de permitir apenas o envio de mensagens pelo administrador, o que contribuiu para a filtragem de conteúdos de alta qualidade.

Além disso, todos os grupos selecionados possuíam sites associados a suas atividades e eram fontes confiáveis de notícias, garantindo que as informações extraídas para o monitoramento de crimes fossem precisas e atualizadas. Esses sites estavam relacionados a veículos de comunicação reconhecidos e respeitados, o que aumentou a credibilidade do conteúdo dos grupos.

Após a identificação e validação dos grupos, iniciou-se a coleta de dados dessas comunidades, com foco nas mensagens relacionadas a crimes, ocorrências policiais e outros temas ligados à segurança pública. Essa estratégia de busca permitiu a inclusão de uma variedade de fontes de dados, ampliando a abrangência do sistema de monitoramento e aumentando a quantidade de informações disponíveis para o processamento e análise.

A busca e identificação de grupos foi, portanto, um passo fundamental para garantir a diversidade e a relevância dos dados coletados, além de assegurar que o sistema fosse alimentado com informações confiáveis e atualizadas sobre o contexto de criminalidade em Belo Horizonte.

3.5 Fluxo das Mensagens no Sistema

O fluxo das mensagens no sistema segue uma sequência estruturada de etapas, garantindo que os dados coletados sejam processados, analisados e armazenados corretamente. A metodologia adotada permite a automação do processo, desde a captura das mensagens até a disponibilização das informações no aplicativo para consulta pelos usuários. A Figura 2 ilustra esse processo de forma detalhada.

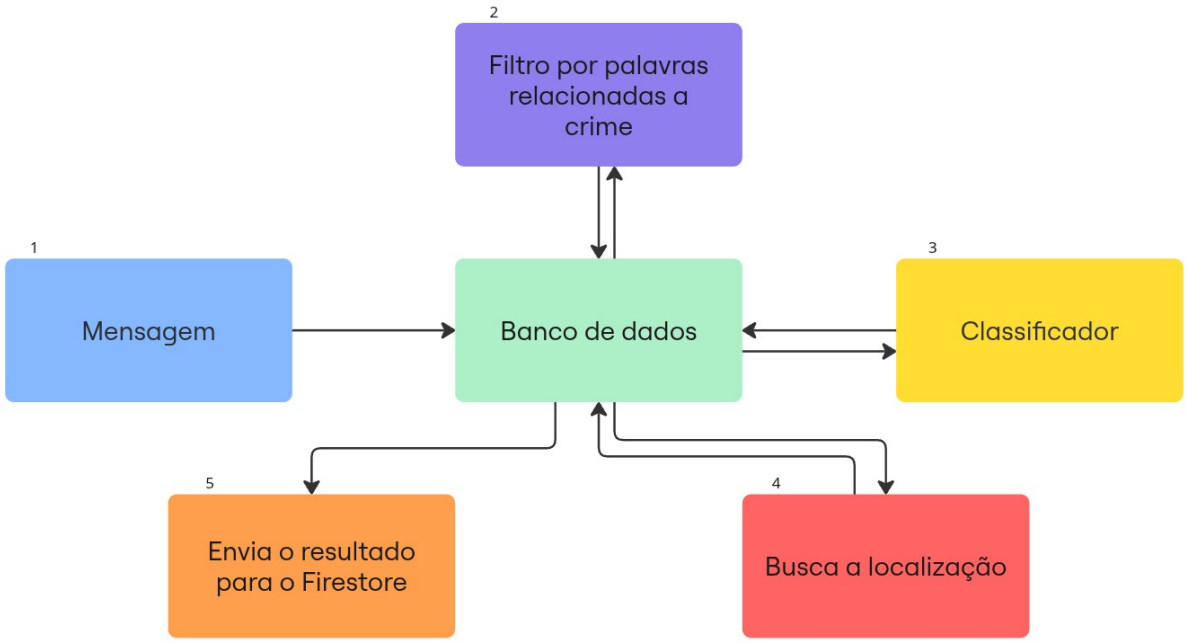


Figura 2 – Fluxo de processamento das mensagens no sistema.

Após a coleta e armazenamento da mensagem no banco de dados, ela passa por mais quatro processos até atingir o objetivo final de visualização no aplicativo BHSafezone (PATRíCIO, 2023). A seguir, cada etapa será detalhada:

1. **Filtro por palavras relacionadas a crime**: A primeira etapa consiste na filtragem por palavras relacionadas a crimes. Para isso, é utilizado um dicionário de crimes construído com dados da Secretaria de Estado de Justiça e Segurança Pública de Minas Gerais. Conforme ilustrado na Figura 3, quando uma palavra relacionada a crime é encontrada no texto, ela é armazenada no banco de dados, no campo `crime`, e o processo segue para a próxima etapa. Caso nenhuma palavra seja identificada, a propriedade `is_crime` no banco de dados é definida como `0` e a mensagem é classificada com `classified = 1`.

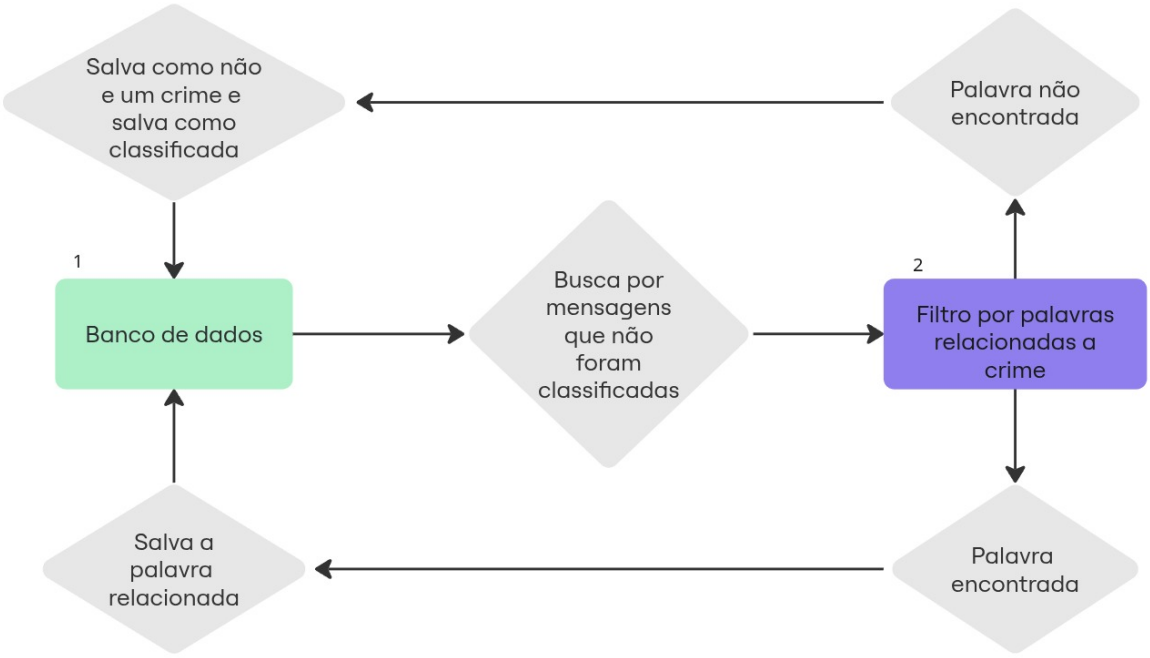


Figura 3 – Filtro por palavras relacionadas a crimes.

2. **Classificador**: A próxima etapa envolve o uso do classificador. Inicialmente, o classificador é treinado utilizando uma base de dados manualmente criada, onde as mensagens são classificadas como `0` (não é crime) ou `1` (é crime), com base em um arquivo CSV. Após o treinamento, o classificador realiza a busca no banco de dados por mensagens que ainda não foram classificadas, ou seja, aquelas com `classified = 0`. O classificador então analisa as mensagens e, após determinar o resultado, os dados são atualizados no banco de dados. Se necessário, o campo `is_crime` é ajustado para `0` ou `1`, conforme o resultado da classificação. O fluxo dessa etapa pode ser visualizado na Figura 4.

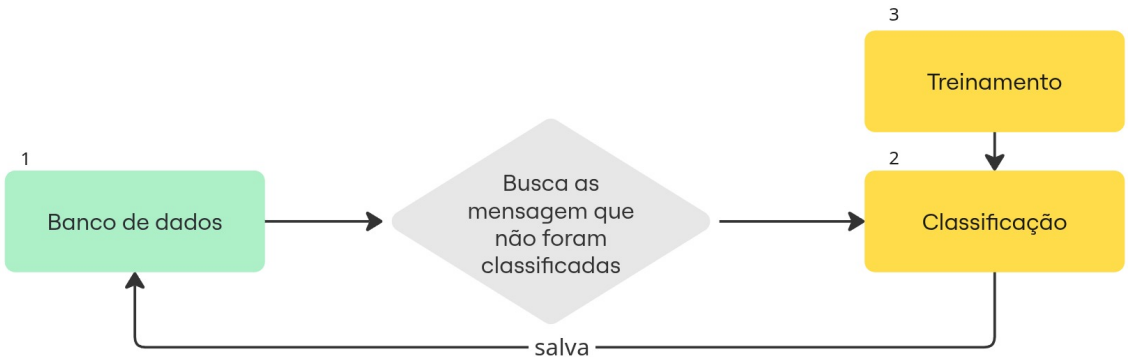


Figura 4 – Classificador de crime.

3. **Busca pela localização**: A terceira etapa do processo consiste na busca pela localização do crime. Inicialmente, o sistema realiza a busca da localização diretamente

no corpo da mensagem. Quando as informações são encontradas, elas são salvas no banco de dados nos campos `region` e `bairro`, conforme ilustrado na Figura 5. Caso a localização não seja encontrada no texto, o sistema verifica a presença de alguma URL. Se uma URL for identificada, é realizada uma requisição e, em seguida, uma busca na resposta da requisição para extrair a localização.

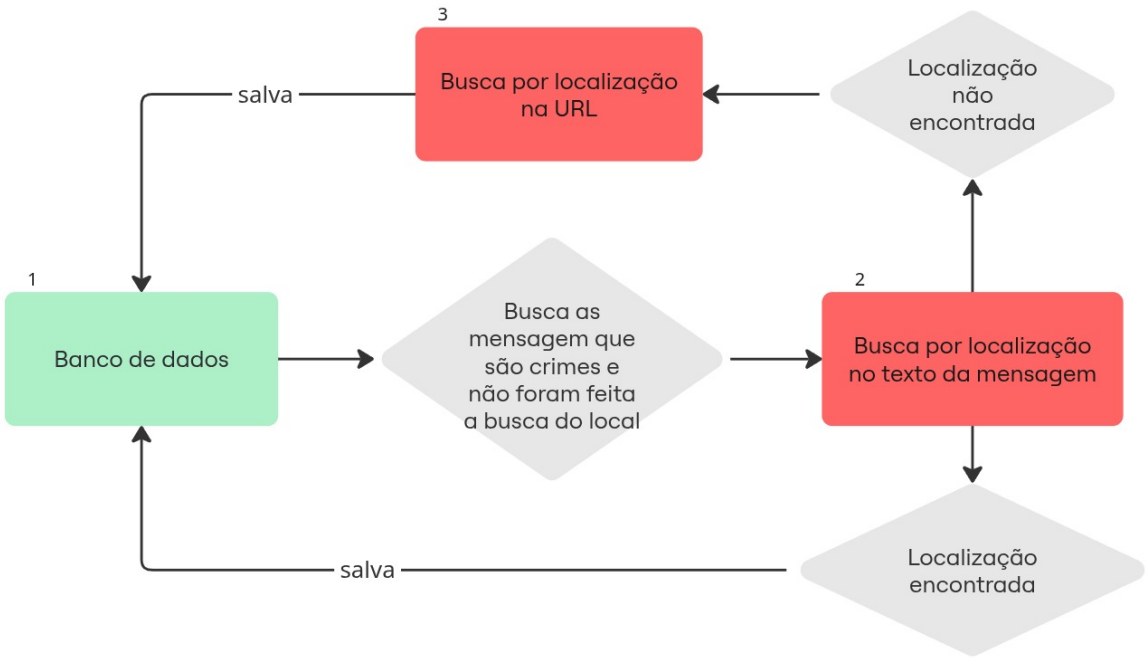


Figura 5 – Busca pela localização do crime.

4. **Envio do resultado para o Firestore**: A última etapa consiste no envio das mensagens classificadas como crime para o banco de dados Firestore, desde que ainda não tenham sido integradas. Nesse processo, os dados são organizados conforme seus respectivos filtros, incluindo região, bairro, região por período, geral e geral por período, como ilustrado na Figura 6. Após o envio, a mensagem é atualizada no MongoDB, alterando o campo `integrated` para 1, indicando que foi corretamente processada e integrada ao Firestore.

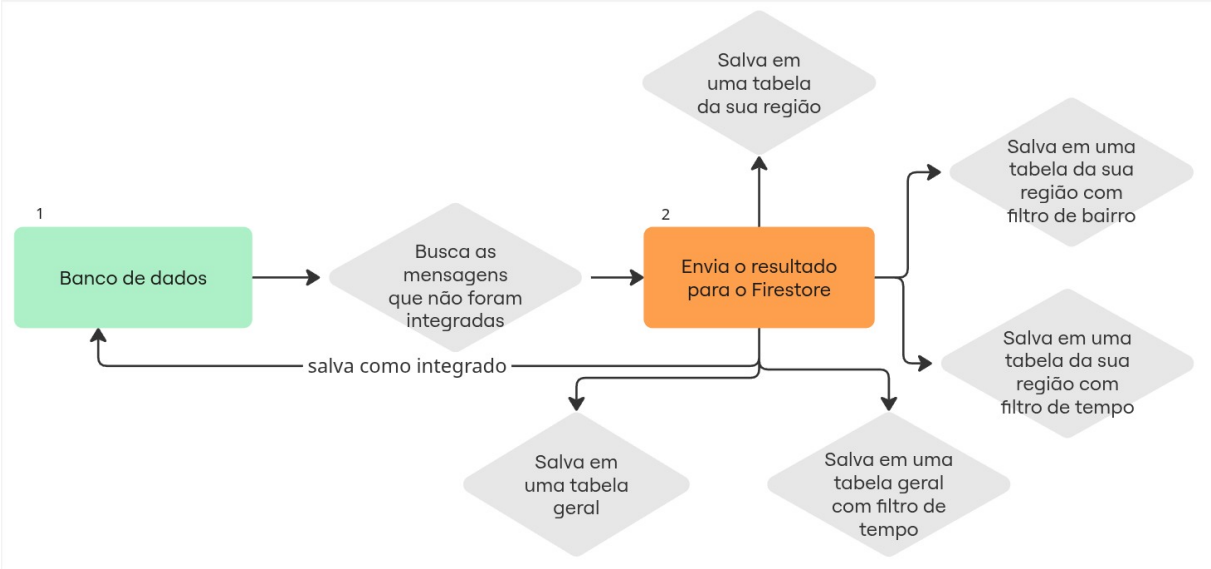


Figura 6 – Envio do resultado para o Firestore.

3.6 Coleta de Dados

A coleta de mensagens do WhatsApp é realizada por meio da biblioteca `whatsapp-web.js`[8], que corresponde à **ETAPA 2** do processo representado na Figura 1. Essa ferramenta permite a extração automatizada de mensagens compartilhadas em grupos públicos. Cada grupo possui um identificador único (ID), que varia de acordo com o usuário. A obtenção das mensagens pode ser feita individualmente para cada grupo, garantindo que o sistema consiga segmentar os dados de forma estruturada e eficiente.

Embora o serviço fornecido pela `whatsapp-web.js` ainda esteja em desenvolvimento e apresente algumas limitações, ele atende perfeitamente ao propósito de capturar mensagens de grupos do WhatsApp. O processo de obtenção das mensagens ocorre por meio da função *getChatById* (linha 46 do código 6.1), que localiza o grupo específico a partir do seu ID. Em seguida, a função *fetchMessages* (linha 48 do código 6.1) recupera as mensagens mais recentes compartilhadas no grupo. Esse fluxo é essencial para garantir que o sistema sempre colete dados atualizados sobre ocorrências mencionadas nos grupos monitorados.

Cada grupo do WhatsApp possui um serviço de coleta específico — como ilustrado no Código 6.2 —, sendo exemplos os grupos *BHAZap*, *Por Dentro de Minas*, *DeFato*, *Agito Mais*, *JCO* e *JCA*, cada um com um identificador único. Para cada serviço, é realizada a busca pelas últimas mensagens recebidas, garantindo que não haja duplicações com base no conteúdo textual. Uma vez obtidas, as mensagens são armazenadas no banco de dados MongoDB, juntamente com metadados relevantes para análises futuras. Esses metadados incluem informações como a origem da mensagem (grupo de onde foi extraída) e

[8] Repositório oficial da biblioteca `whatsapp-web.js`: <https://wwebjs.dev/>

a propriedade `classified`, inicialmente definida como `0`, conforme mostrado no Código 6.2. Esse campo indica que a mensagem ainda não passou pelo processo de classificação, o qual será realizado nas etapas posteriores do fluxo de processamento.

Os dados coletados e armazenados no banco MongoDB seguem a estrutura descrita na Tabela 1.

Campo	**Descrição**
`_id`	Identificador único gerado pelo MongoDB
`title`	Título da mensagem do grupo
`description`	Descrição da mensagem do grupo
`body`	Corpo da mensagem extraída do grupo
`origin`	Grupo do WhatsApp de onde a mensagem foi coletada
`classified`	Indica se a mensagem foi classificada (‘0‘ para não classificada, ‘1‘ para classificada)
`is_crime`	Indica se a mensagem foi identificada como crime (‘0‘ para não, ‘1‘ para sim)
`crime`	Palavra-chave relacionada ao crime identificado na mensagem
`found_location`	Indica se a localização foi identificada (‘0‘ para não, ‘1‘ para sim)
`region`	Região da localização do crime mencionado na mensagem
`bairro`	Bairro associado à localização do crime
`integrated`	Indica se a mensagem foi integrada ao Firestore (‘0‘ para não, ‘1‘ para sim)

Tabela 1 – Estrutura dos dados armazenados no MongoDB.

Esse serviço é executado automaticamente todos os dias às 10 horas, garantindo a coleta contínua das mensagens sem necessidade de intervenção manual. Cada grupo monitorado é acionado por controladores específicos, que seguem uma lógica semelhante para garantir a extração eficiente das informações. Como apresentado no código 6.3, a coleta de mensagens é a **ETAPA 02** do fluxo do sistema e desempenha um papel fundamental na obtenção de dados relevantes para o monitoramento da criminalidade em Belo Horizonte.

3.7 Filtragem de Palavras Relacionadas à Crimes

A filtragem de palavras-chave, correspondente à **ETAPA 3** do processo representado na Figura 1, desempenha um papel essencial na triagem e organização das mensagens, sendo um dos componentes fundamentais para o bom desempenho do sistema. Utilizando um dicionário de palavras associadas a crimes, como “roubo”, “furto”, “homicídio”, entre outras, o sistema é capaz de identificar atividades criminosas nas mensagens. Esse processo assegura que apenas as mensagens que realmente contêm termos relacionados a crimes

avancem para as etapas subsequentes de processamento e classificação, aumentando a precisão do sistema.

A Tabela 2 apresenta o dicionário de crimes utilizado no sistema, que lista os termos mais comuns associados a cada tipo de crime. Esses termos são fundamentais para a busca de correspondências nas mensagens coletadas. Quando uma correspondência é identificada, a mensagem é então associada ao crime correspondente.

Tabela 2 – Dicionário de crimes e termos comuns relacionados. Fonte: Patrício (2023).

Crime	Termos Comuns
Roubo	roubo, roubado, roubada
Furto	furto, furtado, furtada, batedor de carteira, trombadinha
Assalto	assalto, assaltada, assaltado, arrastão
Feminicídio	feminicídio
Estupro	estupro, estuprado, estuprada, violentado, violentada, violência sexual, abuso sexual, abusada, abusado, importunação sexual
Extorsão	extorsão, coagido, coagir
Lesão Corporal	lesão corporal, briga, confronto, agressão, agredida, agredido, confusão, violência doméstica
Sequestro	sequestro, perseguição
Homicídio	homicídio, assassinado, assassinada
Tráfico de Drogas	tráfico de drogas, tráfico, drogas, maconha, entorpecente, cocaína, LSD, ecstasy, heroína
Tentativa de Homicídio	tentativa de homicídio, baleado, baleada
Depredação	depredação, pichações, pichação, vandalismo, vandalização
Incêndio	incêndio, incendiar, incendiou, incendiaram

O sistema foi projetado para ser altamente eficiente, permitindo o processamento de grandes volumes de dados sem prejudicar o desempenho. A flexibilidade do serviço também garante que novas palavras ou expressões associadas a crimes possam ser facilmente integradas ao sistema, mantendo-o sempre atualizado e adaptável. Além disso, a filtragem dos dados organizou as informações, facilitando a interpretação das mensagens e a extração de insights valiosos.

Em resumo, a filtragem de palavras-chave foi um passo crucial para o desenvolvimento do sistema, assegurando que apenas as mensagens mais relevantes fossem selecionadas para uma análise mais aprofundada.

O código 6.4 implementa o serviço responsável pela filtragem.

3.8 Classificação de Crimes

Neste projeto, utilizou-se um modelo de classificação binária baseado em técnicas de Machine Learning, com o objetivo de identificar automaticamente se uma mensagem coletada de grupos do WhatsApp se refere ou não a um crime. O algoritmo escolhido foi o Support Vector Machine (SVM), amplamente reconhecido por sua robustez em tarefas de classificação e por sua capacidade de encontrar um hiperplano ótimo que separa os dados em diferentes categorias com a maior margem possível. No contexto deste trabalho, as categorias de interesse são: **crime** e **não crime**.

O modelo adotado é uma adaptação direta do classificador previamente desenvolvido por Patrício (2023) em outro projeto, já em produção e com desempenho validado. Tal reaproveitamento garante maior confiabilidade à aplicação e economia de tempo, uma vez que os principais componentes do pipeline de classificação – como o `CountVectorizer`[9], o `TfidfTransformer`[10] e o próprio classificador (`clf`) – já haviam sido treinados e salvos utilizando a biblioteca `joblib`, possibilitando sua reutilização direta no novo fluxo da aplicação principal. O código-fonte responsável pelo carregamento do treinamento encontra-se no Apêndice deste trabalho, na Listagem 6.5.

O processo de treinamento do modelo seguiu uma abordagem supervisionada, utilizando um conjunto de dados composto por mensagens reais coletadas de grupos do X/Twitter, previamente rotuladas como `crime = 1` ou `não crime = 0`. Para garantir a qualidade do treinamento e a capacidade de generalização do modelo, a base foi dividida em três subconjuntos: treinamento, validação e teste, como ilustrado na Figura 7.

[9] <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>.

[10] <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html>.

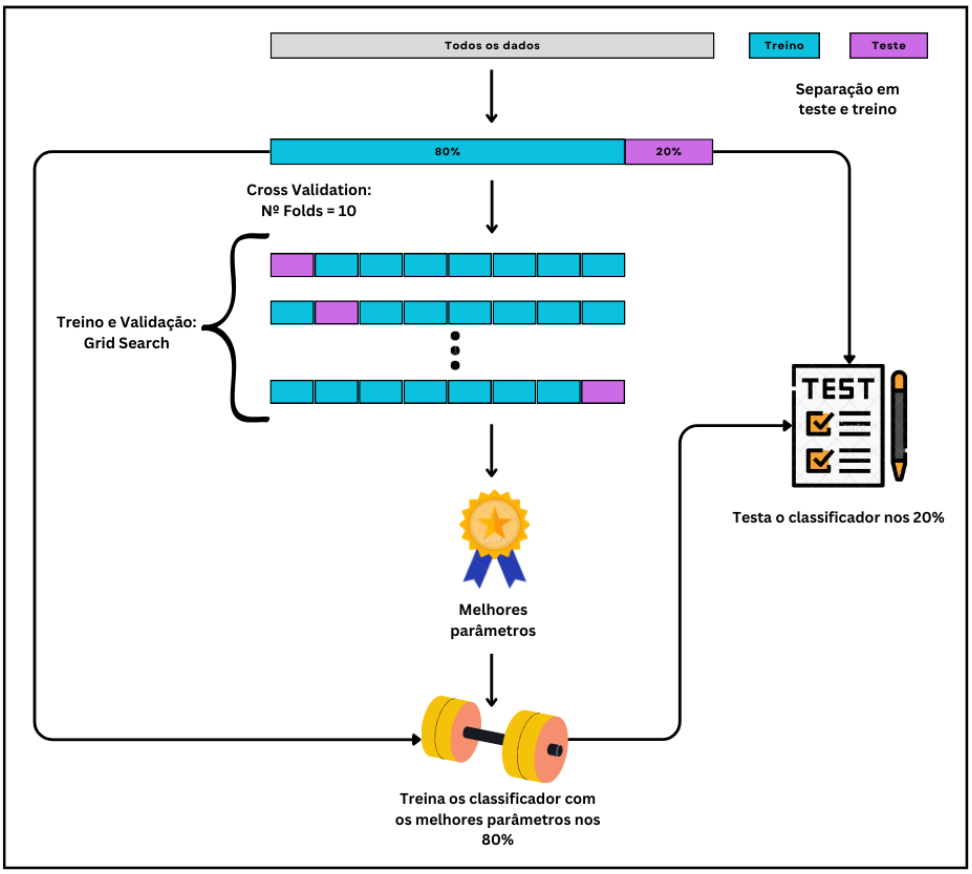


Figura 7 – Representação do processo de divisão do dataset em treino, teste e validação. Fonte: Patrício (2023).

A divisão da base em subconjuntos segue práticas comuns no treinamento de modelos de aprendizado supervisionado. O conjunto de treinamento é utilizado para ajustar os parâmetros do modelo, enquanto o conjunto de validação permite monitorar o desempenho durante o processo e realizar ajustes que evitem o fenômeno conhecido como *overfitting*. Por fim, o conjunto de teste serve como avaliação final da capacidade de generalização do modelo, ou seja, sua habilidade de classificar corretamente dados que ele nunca viu anteriormente.

Durante o treinamento, foi necessário lidar com o problema de classes desbalanceadas, uma vez que mensagens não relacionadas a crimes tendem a ser majoritárias. Para mitigar esse viés, aplicaram-se técnicas simples de balanceamento, como a subamostragem da classe majoritária e o uso de ponderação de classes no treinamento do SVM.

Além disso, os dados passaram por um rigoroso processo de pré-processamento, com etapas como remoção de pontuações e acentuações, conversão para caixa baixa, remoção de *stopwords* e lematização. Após essa limpeza, os textos foram transformados em representações numéricas por meio do `CountVectorizer`, que gera uma matriz de frequência de palavras, seguido pelo `TfidfTransformer`, que ajusta essa frequência com base na importância das palavras em todo o corpus TF-IDF

Para garantir a robustez do modelo, foram realizados experimentos com diferentes sementes de aleatoriedade, validação cruzada e ajuste de hiperparâmetros com o `GridSearchCV`. Após a escolha da configuração ideal, o modelo foi treinado com todo o conjunto de dados e os seguintes artefatos foram salvos:

- **`modelo_classificador.pkl`**: arquivo contendo o classificador treinado (SVM);
- **`count_vectorizer.pkl`**: objeto responsável por transformar texto em vetores de contagem;
- **`tfidf_transformer.pkl`**: transformador que aplica a técnica TF-IDF às contagens.

Esses arquivos foram carregados na aplicação principal para permitir a classificação automática das mensagens em tempo real.

Após a avaliação do modelo e sua aprovação para uso em produção, foi realizada sua integração com a base de dados MongoDB, mais especificamente com a coleção denominada `whatsapps`, responsável por armazenar todas as mensagens coletadas de grupos públicos. Cada nova mensagem inserida na base possui, por padrão, o campo `classified` definido como 0, indicando que ainda não passou pelo processo de classificação. O sistema responsável por essa etapa, correspondente à **ETAPA 4** da Figura 1, implementa um fluxo automatizado composto pelas seguintes fases:

1. O sistema consulta todas as mensagens da coleção `whatsapps` cujo campo `classified` é igual a 0;
2. Cada mensagem é submetida ao pipeline de pré-processamento, vetorização e transformação TF-IDF, de forma idêntica à realizada durante o treinamento;
3. O classificador SVM analisa a mensagem e determina se ela está relacionada a um crime ou não;
4. O resultado da classificação é salvo na própria coleção, com as seguintes alterações:
    - O campo `classified` é alterado para 1, marcando a mensagem como processada;
    - O campo `is_crime` é definido como 1 para mensagens classificadas como crime, e 0 caso contrário;
    - Para mensagens consideradas crimes, o campo `found_location` é inserido e recebe o valor 0, indicando que a extração de local ainda não foi realizada e deverá ocorrer na próxima etapa do fluxo.

Esse processo é executado periodicamente, de forma automatizada, garantindo que a base de dados esteja sempre atualizada com as classificações mais recentes, permitindo a continuidade das etapas seguintes, como a extração de localizações e a geração de alertas.

O funcionamento do classificador pode ser resumido da seguinte forma: ao detectar uma nova mensagem com `classified = 0`, o sistema carrega os modelos salvos previamente (vetorizador, transformador TF-IDF e o classificador SVM), aplica os mesmos passos de pré-processamento utilizados durante o treinamento, e em seguida realiza a predição.

Se a mensagem for considerada relacionada a crime, o sistema a marca como tal, e já a prepara para o módulo de extração de localização. Todo o processo foi desenhado para garantir consistência, escalabilidade e performance, sendo capaz de processar grandes volumes de mensagens com baixa latência, e com acurácia compatível com os padrões esperados.

O código-fonte responsável pelo carregamento do classificador e pela interação com a base de dados encontra-se no Apêndice deste trabalho, na Listagem 6.6. Nessa listagem, são apresentados todos os passos realizados pelo script, incluindo o carregamento dos modelos, a busca de mensagens no banco, o processo de classificação e a atualização dos documentos no MongoDB.

Esse processo representa um avanço significativo na automação da análise de informações provenientes de redes sociais e grupos de mensagens, permitindo que a aplicação detecte padrões de comportamento criminoso em tempo real, de forma autônoma e confiável.

3.9 Processo de Busca da Localização

Depois da classificação da mensagem do WhatsApp como crime, é necessário identificar o local (e.g., bairro ou região da cidade) onde o ocorrido aconteceu, conforme representado na **ETAPA 5** da Figura 1. Para isso, foi feito um filtro composto de 3 etapas:

- Identificar se existe algum local presente no texto da publicação;
- Identificar se existem links para outras páginas no texto da publicação. Caso positivo, identificar se existem locais presentes na página do endereço direcionado pelo link;
- Identificar o bairro e a região dos locais encontrados.

A primeira etapa é necessária para confirmar se realmente há um local presente no texto da mensagem e qual a identificação desse local. Ao indicar a localização do crime na publicação, muitas vezes é utilizado o nome popular do local ou apenas o nome

de uma rua, avenida, praça ou viaduto. Caso o filtro fosse construído apenas com base nos nomes dos bairros, além de deixar algumas publicações de fora, haveria o risco de a coleta não ser precisa. Para resolver esse problema, foi utilizada a biblioteca de software para processamento avançado de linguagem natural, `spaCy`[11]. Entre as funcionalidades do `spaCy`, está o Reconhecimento de Entidades Mencionadas (REM). O REM é uma supertarefa da extração de informações e visa localizar e classificar elementos do texto em categorias pré-definidas, como nomes de pessoas, organizações, lugares, datas e outras classes Carvalho (2012). Dessa forma, o spaCy foi usado para identificar entidades de localização presentes no texto das mensagens.

A segunda etapa tem como objetivo identificar links externos presentes no texto da publicação. Isso é relevante, pois algumas publicações consistem apenas em uma manchete para a notícia completa, que pode ser encontrada no link. Para identificar corretamente o local da ocorrência do crime, foi necessário detectar as mensagens que continham links e capturar o texto da notícia presente na página de destino. Essa coleta foi realizada por meio de *web scraping* no site de notícias para o qual o link apontava. Após obter o texto do WhatsApp, o REM foi utilizado novamente para identificar entidades de localização presentes nas páginas de notícias acessadas.

Com os locais identificados, a terceira etapa consistiu em determinar a região e o bairro dos locais encontrados. Por exemplo, ao identificar o local "Praça da Liberdade" entre as entidades, não é possível determinar onde exatamente essa praça está localizada na cidade. Para isso, foi utilizada a API Geocoder da Prefeitura de Belo Horizonte[12]. A API Geocoder é um serviço de endereços no formato de Web Service REST que possibilita consultas a endereços na base da prefeitura da cidade de BH, retornando um objeto JSON que contém, entre outros atributos, as coordenadas geográficas do endereço pesquisado. Essa API permite diferentes tipos de consultas, e a consulta utilizada neste trabalho foi por endereço textual. Com isso, ao realizar as consultas na API através do texto das entidades de localização identificadas, foi possível determinar o bairro e a região correspondentes. Em seguida, essas localizações foram salvas no banco de dados junto com suas respectivas publicações.

Devido ao alto volume de requisições, a API da prefeitura restringiu o número de consultas disponíveis. Para resolver esse problema, foi criado um dicionário em Python contendo o nome de todos os bairros da cidade, juntamente com sua respectiva região. Dessa forma, todas as pesquisas em que apenas o nome do bairro era identificado foram feitas nesse dicionário, enquanto as pesquisas com outros tipos de localização (ruas, avenidas, praças, viadutos) foram realizadas através da API Geocoder. Os algoritmos contendo as 3 etapas do filtro de localização estão disponíveis no Apêndice, na Listagem 6.7.

[11] <https://spacy.io/>.
[12] <http://geocoder.pbh.gov.br/geocoder/>.

Após a aplicação do filtro, dos **687** textos classificados como crimes, **379** mensagens apresentaram regiões desconhecidas. Desse total, **146** mensagens eram de fora da região metropolitana de Belo Horizonte, e as **233** mensagens restantes não foram localizadas em bairros ou regiões dentro de Belo Horizonte. A distribuição dos dados é a seguinte:

Das 687 mensagens classificadas como crimes:

a) 379 mensagens com região desconhecida: 146 mensagens fora da região metropolitana de BH; 233 mensagens de BH, mas sem local identificado;

b) 308 mensagens com localização identificada em BH: 120 mensagens indicaram bairros e regiões específicas de Belo Horizonte; 188 mensagens estavam dentro da região metropolitana de BH, mas sem especificação de bairro ou região exata.

Esses dados serão utilizados para a visualização de crimes por região e para a exibição do total de crimes no aplicativo deste trabalho.

3.10 Integração dos Dados

A integração com o Firestore no sistema desenvolvido tem como objetivo armazenar e organizar os dados coletados das mensagens do WhatsApp, garantindo que a informação seja estruturada de forma eficiente para consultas e análises futuras. Essa funcionalidade corresponde à **ETAPA 6** do processo representado na Figura 1. O serviço responsável pela conexão, apresentado no código 6.9 do Apêndice, inicializa a conexão ao banco de dados utilizando as credenciais fornecidas no arquivo de configuração e disponibiliza métodos para inserção e atualização dos registros, garantindo que os dados sejam armazenados corretamente.

Os crimes são registrados no Firestore de maneira hierárquica, sendo organizados em coleções separadas por região e por um agrupamento geral. Essa estrutura possibilita consultas específicas e globais sobre a criminalidade. Dentro de cada coleção de região ou da coleção geral, os crimes são armazenados como documentos, onde cada um representa um tipo específico de crime e contém a quantidade de ocorrências registradas. Para manter a consistência dos dados e evitar duplicações, a solução adota uma abordagem de atualização ou inserção condicional, verificando se um documento correspondente já existe. Caso exista, a contagem da ocorrência é incrementada; se não, um novo documento é criado.

Além dessa categorização por crimes, os registros também são organizados por período de tempo. Em cada região e na coleção geral, há um documento específico denominado `time_series`, que contém uma subcoleção onde os dados são estruturados por mês e ano. Cada documento dentro dessa subcoleção representa um período e armazena

as quantidades de crimes ocorridos nele, permitindo acompanhar a evolução das ocorrências ao longo do tempo.

No caso das regiões, há uma subdivisão adicional por bairros, garantindo uma granularidade maior na análise dos dados. Dentro de cada região, existe um documento denominado "bairros", que lista os bairros registrados e, para cada um deles, há uma subcoleção que segue a mesma estrutura utilizada no nível regional. Cada bairro tem seus crimes armazenados como documentos individuais, com o nome do crime e a quantidade correspondente.

O serviço responsável pela estruturação dos dados para inserção no Firestore, apresentado no código 6.8 do Apêndice, é encarregado de organizar e categorizar as mensagens coletadas antes de enviá-las ao banco de dados. Ele recupera todas as mensagens do banco de dados local que ainda não foram processadas e as insere no Firestore de maneira estruturada. Durante esse processo, as mensagens são analisadas e distribuídas corretamente entre as coleções regionais, de bairros e séries temporais. Após a conclusão da integração de cada registro, a base local é atualizada para indicar que a mensagem já foi processada, evitando duplicações futuras.

Essa organização garante que as consultas possam ser realizadas em diferentes níveis, possibilitando a obtenção de informações detalhadas ou amplas conforme a necessidade. A estrutura hierárquica por regiões e bairros permite análises localizadas, enquanto a categorização por tempo facilita o acompanhamento da evolução da criminalidade. Além disso, a coleção geral proporciona uma visão consolidada dos registros, permitindo uma análise mais abrangente dos dados coletados.

3.11 Considerações Finais

Será dada especial atenção às questões éticas e legais relacionadas à coleta de dados. O sistema de coleta será configurado para respeitar as diretrizes da LGPD, garantindo a privacidade dos usuários e o uso responsável das informações. Além disso, será garantido que a coleta ocorra apenas em **grupos públicos do WhatsApp**, sem violar a privacidade dos participantes ou acessar grupos privados sem o consentimento adequado.

O desenvolvimento do sistema de monitoramento de crimes em Belo Horizonte, utilizando dados provenientes de grupos de WhatsApp, representou uma evolução significativa em relação à ideia inicial, que se baseava na coleta de tweets geolocalizados. A adaptação para uma nova fonte de dados, aliada ao uso de técnicas de Machine Learning e automação, permitiu a construção de uma plataforma robusta e eficiente para a análise de ocorrências criminosas na cidade.

A escolha do WhatsApp como principal fonte de dados se deu por sua ampla

adoção entre os brasileiros e sua relevância como meio de comunicação local, especialmente em comunidades onde as informações sobre ocorrências cotidianas são frequentemente compartilhadas de forma rápida e direta por meio de grupos públicos. Essa característica o torna uma ferramenta estratégica para a coleta de dados relacionados à criminalidade em tempo real.

Durante o processo de desenvolvimento, foram realizadas diversas melhorias no sistema, incluindo a reestruturação do framework, otimização do código e integração contínua com o banco de dados. A utilização do NestJS e MongoDB, junto ao Firestore para o armazenamento em tempo real, garantiu uma base sólida e escalável para o sistema. A automação da coleta e classificação das mensagens foi outro ponto crucial, pois permitiu a atualização constante da base de dados sem a necessidade de intervenção manual.

Embora o sistema tenha sido projetado para oferecer informações em tempo real sobre a criminalidade, um dos desafios encontrados foi a variabilidade e a qualidade das informações disponíveis nos grupos de WhatsApp. A filtragem e classificação das mensagens, mesmo com o uso de técnicas avançadas de Machine Learning, dependem da qualidade e da precisão das informações fornecidas pelos usuários.

Com a conclusão deste projeto, o sistema se apresenta como uma ferramenta importante para o monitoramento de crimes, com o potencial de fornecer dados em tempo real para a população e autoridades. No entanto, o aprimoramento contínuo do modelo de aprendizado e a expansão para novas fontes de dados podem tornar a solução ainda mais robusta e eficiente.

Em suma, este projeto contribui para o avanço das soluções tecnológicas no campo da segurança pública, trazendo inovações no uso de dados para monitoramento e mapeamento de crimes, com um impacto potencial no aumento da segurança e na redução da criminalidade em Belo Horizonte.

4 Resultados

Este capítulo apresenta os resultados alcançados com a implementação e execução do sistema de monitoramento de crimes, conforme os passos descritos no Capítulo 3. O objetivo do sistema foi fornecer uma ferramenta eficiente para coletar, processar e analisar mensagens de grupos de WhatsApp relacionadas à segurança pública, contribuindo para o monitoramento e a prevenção da criminalidade em Belo Horizonte.

A Seção 4.1 descreve a coleta e o armazenamento das mensagens, explicando o processo automatizado de extração das mensagens de grupos relevantes e sua organização inicial. A Seção 4.2 detalha as adaptações realizadas no código do classificador e do módulo de localização, que aprimoraram a precisão na categorização dos crimes e na identificação de regiões mencionadas nas mensagens. Em seguida, a Seção 4.3 discute a integração automática com o Firestore, garantindo que os dados sejam armazenados de maneira estruturada e continuamente atualizados. Finalmente, a Seção 4.4 apresenta a utilização dos dados processados no aplicativo BHSafeZone (PATRíCIO, 2023), que oferece aos usuários informações em tempo real sobre a criminalidade na cidade, promovendo uma maior segurança pública e uma comunicação mais eficiente entre a população e as autoridades.

Ao longo deste capítulo, serão detalhadas as etapas do processo e discutidos os principais resultados alcançados, destacando a eficácia do sistema desenvolvido na coleta, organização e visualização das informações de segurança pública.

4.1 Coleta e Armazenamento de Mensagens

A coleta de dados foi uma das etapas iniciais e fundamentais para o sucesso do sistema de monitoramento de crimes. Para realizar essa coleta de forma automatizada, foi utilizada a biblioteca `whatsapp-web.js`, que possibilita a interação com a versão web do WhatsApp. Essa abordagem foi escolhida por sua flexibilidade e capacidade de operação em tempo real, sem a necessidade de acessar a API oficial do WhatsApp, o que aumenta a liberdade e a escalabilidade da solução.

A coleta foi realizada em grupos específicos de WhatsApp que discutem questões relacionadas à segurança pública e eventos criminosos em Belo Horizonte. O sistema foi configurado para extrair as mensagens desses grupos de maneira contínua e automatizada, o que garantiu a atualização constante dos dados coletados. Durante o período de coleta, foram obtidas um total de **4313 mensagens**. Essas mensagens foram analisadas quanto à sua relevância para o sistema, buscando identificar informações sobre ocorrências criminosas,

denúncias de segurança, e outros dados pertinentes ao monitoramento de crimes na cidade.

O processo de coleta foi estruturado de forma a garantir a integridade dos dados. Para isso, mecanismos foram implementados para evitar a duplicação de mensagens, garantindo que cada ocorrência fosse registrada uma única vez. Além disso, o sistema assegurou que todas as mensagens fossem armazenadas com informações associadas como remetente, data e hora de envio, links compartilhados e anexos, quando presentes. Esses dados adicionais são valiosos para a contextualização das mensagens e para futuras análises.

A flexibilidade da abordagem de coleta em tempo real foi um dos principais pontos positivos dessa etapa. Com a coleta contínua, foi possível garantir que o sistema estivesse sempre alimentado com dados atualizados, proporcionando informações relevantes e atuais para a análise de criminalidade. Essa integração em tempo real também permitiu uma maior agilidade no processamento e no armazenamento dos dados no banco de dados, assegurando que o sistema estivesse sempre pronto para fornecer informações de monitoramento de crimes de forma imediata.

Em resumo, a coleta de dados foi realizada de forma eficiente e contínua, garantindo que um volume significativo de mensagens fosse extraído e processado. A abordagem adotada foi flexível o suficiente para se adaptar à dinâmica dos grupos de WhatsApp, assegurando que as informações sobre segurança pública e ocorrências criminosas fossem sempre atualizadas, e formando a base de dados necessária para as etapas seguintes do sistema.

4.2 Adaptação dos Códigos Classificador e Busca de Localização

A adaptação do classificador de crimes e do sistema de busca por localização foi um passo essencial na transição do monitoramento de mensagens do `Twitter` para o `WhatsApp`. Essa mudança possibilitou a coleta e análise de um volume significativo de dados, totalizando **4313 mensagens** coletadas, das quais **687** foram classificadas como crimes.

Os resultados apresentados nesta seção referem-se ao modelo desenvolvido por Patrício (2023), que foi salvo e reutilizado neste trabalho na classificação de crimes e atualmente está em produção. O modelo foi adaptado e incorporado com sucesso ao novo sistema, garantindo que a classificação das mensagens do `WhatsApp` ocorra de maneira automatizada e contínua, sem necessidade de intervenção manual.

O desempenho do classificador foi avaliado por meio de métricas estatísticas, apresentando um desvio-padrão de **0,0049**. O intervalo de confiança foi calculado entre **0,9528** e **0,9665**, garantindo uma margem de erro reduzida. A acurácia do modelo atingiu **96%**, enquanto a métrica F1-score foi de **0,9649**, evidenciando a eficiência na classificação

das mensagens como crimes.

Além da adaptação do classificador, o sistema de busca por localização também foi ajustado para processar as mensagens coletadas e identificar padrões geográficos. Das **687 mensagens** classificadas como crimes, **379** apresentavam uma região desconhecida. Dentro desse grupo, **146** estavam fora da região metropolitana de Belo Horizonte, enquanto **233** eram de Belo Horizonte, mas sem um local específico identificado. Em contrapartida, **308 mensagens** tiveram uma localização determinada dentro de Belo Horizonte, sendo que **120** apontaram bairros e regiões específicas, enquanto **188** estavam dentro da região metropolitana, mas sem especificação detalhada.

Esses resultados demonstram a eficácia das adaptações realizadas tanto no classificador quanto no sistema de busca por localização, permitindo uma identificação mais precisa dos crimes reportados e de suas respectivas localizações dentro do contexto do WhatsApp.

4.3 Integração com o Firestore

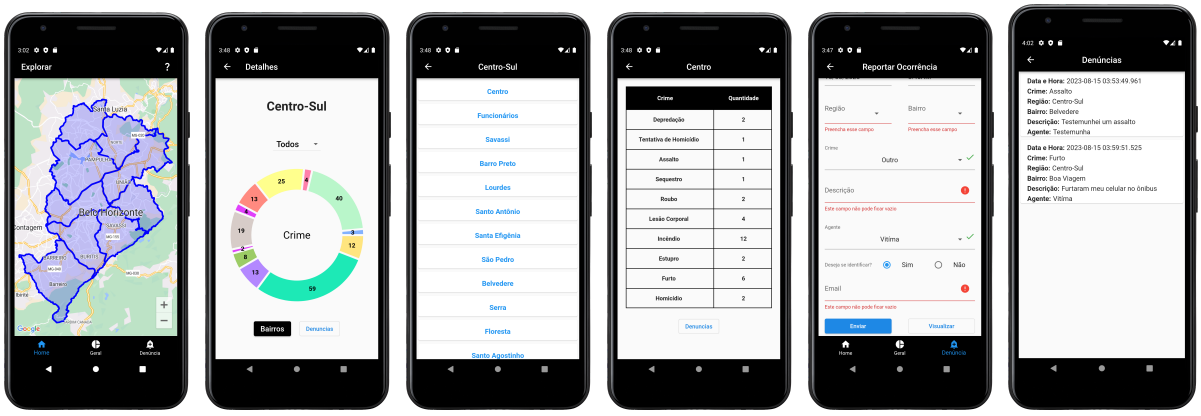
A integração com o Firestore foi um avanço significativo para a automação do sistema, permitindo a atualização automática das informações classificadas e garantindo que os dados estejam sempre acessíveis em tempo real. Esse desenvolvimento foi crucial para otimizar o fluxo de informações, eliminando a necessidade de inserção manual e reduzindo o tempo de processamento das mensagens coletadas.

A estrutura do Firestore, previamente definida, foi fundamental para essa integração. Como o banco de dados foi projetado de forma a utilizar as próprias tabelas como filtros, a organização e recuperação dos dados tornaram-se mais eficientes. Dessa forma, cada nova mensagem classificada como crime é automaticamente armazenada com suas respectivas informações detalhadas, facilitando consultas e análises posteriores.

Essa automação fortaleceu a consistência e confiabilidade do sistema, permitindo que a base de dados se mantenha constantemente atualizada e pronta para alimentar o aplicativo de monitoramento de crimes em Belo Horizonte.

4.4 Visualização no BHSafezone

A integração do sistema com o aplicativo BHSafezone (PATRíCIO, 2023) foi concluída com sucesso, permitindo que os usuários tenham acesso, em tempo real, às ocorrências registradas na cidade. A sincronização entre os dados processados e a interface do aplicativo foi viabilizada através da migração das informações do banco de dados MongoDB para o Firestore. Esse processo garante que as informações mais recentes estejam sempre disponíveis para consulta.



(a) Tela principal. (b) Região. (c) Bairros lista. (d)Bairro: tabela. (e)Denúncias *form.* (f)Denúncias *card.*

Figura 8 – Interfaces do aplicativo. Fonte: Patrício (2023)

A adaptação do sistema para se integrar com o Firestore foi necessária para garantir uma sincronização eficiente e automática dos dados entre a aplicação servidora e o aplicativo BHSafezone (PATRíCIO, 2023). O Firestore foi utilizado devido à sua compatibilidade com a estrutura do aplicativo e sua capacidade de fornecer atualizações em tempo real. Assim, as mensagens classificadas como crimes, juntamente com suas respectivas localizações e categorias, são armazenadas no Firestore após passarem por todas as etapas de processamento e validação. Dessa forma, o aplicativo exibe apenas dados estruturados e relevantes aos usuários.

O BHSafezone (PATRíCIO, 2023) disponibiliza as ocorrências por meio de um mapa interativo, no qual cada crime identificado é exibido de acordo com sua localização extraída das mensagens. Os usuários podem filtrar os registros por tipo de crime e período de tempo, tornando a experiência de navegação mais personalizada e eficiente. Além disso, a plataforma oferece estatísticas detalhadas sobre a criminalidade na região, auxiliando tanto a população quanto as autoridades no monitoramento da segurança pública.

A Figura 8 apresenta algumas telas do aplicativo. A Tela Principal (Figura 8a) oferece um panorama geral das funcionalidades do sistema, permitindo a visualização da distribuição geográfica das ocorrências. A tela de Região (Figura 8b) permite visualizar os números das ocorrências de acordo com o filtro desejado. A listagem de bairros (Figura 8c) possibilita que o usuário filtre as informações pelo bairro de interesse, enquanto a tela de detalhes do bairro (Figura 8d) apresenta informações específicas sobre os crimes na área selecionada. Já a tela de Denúncias (Figura 8e) permite o envio de novas ocorrências pelos usuários, e a tela de Denúncias Detalhadas (Figura 8f) fornece um histórico das submissões realizadas.

Essa etapa da integração garante que as informações coletadas sejam disponibilizadas de forma intuitiva e eficiente, consolidando o BHSafezone (PATRíCIO, 2023) como uma ferramenta essencial para o monitoramento da criminalidade em Belo Horizonte.

4.5 Considerações Finais

A principal contribuição deste trabalho foi a automação completa do processo de coleta, classificação e armazenamento das mensagens, além da reestruturação do framework para que todas as etapas fossem executadas de maneira eficiente e sem necessidade de intervenção manual. Essas melhorias permitiram que o sistema fosse capaz de operar de forma autônoma, garantindo uma coleta contínua de dados e uma atualização em tempo real das informações sobre criminalidade em Belo Horizonte.

Um dos desafios enfrentados foi a adaptação da metodologia para lidar com a natureza dos dados provenientes do WhatsApp. Diferente do X/Twitter, onde há uma grande quantidade de contas oficiais, postagens frequentes e informações estruturadas, os grupos de WhatsApp apresentam fluxos de mensagens menos regulares e um volume menor de dados relevantes para a análise da criminalidade. Essa diferença exigiu a implementação de estratégias mais robustas para garantir que as informações extraídas fossem pertinentes e pudessem ser classificadas corretamente.

Além disso, a reestruturação do framework foi essencial para garantir que o sistema fosse escalável e pudesse ser aprimorado no futuro sem a necessidade de grandes alterações na base de código. A modularização das funções e a otimização do processamento permitiram que novas funcionalidades fossem integradas com mais facilidade, além de garantir maior estabilidade ao sistema como um todo.

A integração do Firestore como banco de dados principal também trouxe benefícios significativos, permitindo que os dados fossem acessados em tempo real pelo aplicativo BH-Safezone (PATRíCIO, 2023). Essa estrutura viabilizou a disponibilização das informações de forma organizada e intuitiva, proporcionando aos usuários uma ferramenta eficiente para monitoramento da criminalidade.

Por fim, os resultados alcançados demonstram que, apesar dos desafios inerentes ao monitoramento de crimes por meio de mensagens no WhatsApp, a automação e a reestruturação do sistema foram capazes de superar essas limitações e fornecer um fluxo contínuo e confiável de informações. Com isso, o sistema desenvolvido representa uma importante contribuição para a análise e o mapeamento da segurança pública, podendo ser expandido e aprimorado para atender a novas demandas no futuro.

5 Conclusão e Trabalhos Futuros

Este trabalho apresentou o desenvolvimento de um sistema para coleta, processamento e visualização de dados sobre criminalidade em Belo Horizonte, utilizando mensagens de grupos de WhatsApp como fonte principal. A solução implementada foi estruturada para garantir a extração eficiente das informações, sua classificação automatizada e a disponibilização dos dados em tempo real no aplicativo BHSafezone (PATRíCIO, 2023). Ao longo do desenvolvimento, diversas etapas foram analisadas e aprimoradas, desde a reformulação do framework até a integração dos dados com o Firestore, garantindo maior automação e acessibilidade às informações processadas.

O trabalho teve como objetivo geral o desenvolvimento de uma aplicação servidora capaz de coletar e armazenar dados do WhatsApp como fonte principal para alimentar um sistema de monitoramento da criminalidade em Belo Horizonte. Para atingir esse objetivo, foram definidos objetivos específicos, os quais foram progressivamente alcançados ao longo da implementação do projeto.

O primeiro objetivo foi a implementação de um sistema de coleta de mensagens do WhatsApp. Esse processo foi realizado utilizando a biblioteca *whatsapp-web.js*, que permitiu a captura das mensagens em grupos específicos, como *BHAZap*, *Por Dentro de Minas*, *DeFato*, *Agito Mais*, *JCO* e *JCA*, de forma contínua e automatizada. O segundo objetivo foi a adaptação dos códigos antigos, que eram focados no Twitter, para que pudessem receber dados de uma estrutura diferente. O classificador foi treinado para identificar padrões textuais relacionados a crimes, refinando suas previsões continuamente a partir dos dados mais recentes. Esse classificador, desenvolvido e treinado no contexto deste trabalho, foi implantado como um serviço funcional, sendo utilizado diretamente no processamento das mensagens em produção. Dessa forma, não é mais necessário realizar o treinamento sempre que o modelo for utilizado para classificar novas mensagens.

O terceiro objetivo consistiu na integração da aplicação servidora com o sistema de mapeamento e monitoramento da criminalidade. Após o processamento das mensagens e extração das informações relevantes, os dados estruturados foram armazenados no banco de dados MongoDB e posteriormente enviados ao Firestore. Esse processo garantiu que o aplicativo BHSafezone (PATRíCIO, 2023) pudesse consumir as informações em tempo real, oferecendo aos usuários uma visualização detalhada das ocorrências registradas.

Além da adaptação do classificador, o sistema de busca por localização também foi ajustado para processar as mensagens coletadas e identificar padrões geográficos. Dos **687** registros classificados como crimes, **379** apresentavam uma região desconhecida. Dentro desse grupo, **146 mensagens** estavam fora da região metropolitana de Belo Horizonte,

enquanto **233** eram de Belo Horizonte, mas sem um local específico identificado. Em contrapartida, **308 mensagens** tiveram uma localização determinada dentro de Belo Horizonte, sendo que **120** apontaram bairros e regiões específicas, enquanto **188** estavam dentro da região metropolitana, mas sem especificação detalhada.

Com base nesses resultados, pode-se concluir que o sistema desenvolvido atingiu seus objetivos, demonstrando viabilidade na aplicação de tecnologias de inteligência artificial e processamento de linguagem natural para o monitoramento da criminalidade em Belo Horizonte.

5.1 Contribuições

Este trabalho contribui significativamente para o campo da segurança pública e da análise de dados em tempo real ao desenvolver uma solução inovadora para o monitoramento da criminalidade em Belo Horizonte, utilizando mensagens de grupos de WhatsApp como fonte de dados. A principal contribuição foi a criação de um sistema automatizado capaz de coletar, processar e classificar mensagens de forma eficiente e autônoma, com o objetivo de identificar ocorrências de crimes e gerar informações geográficas sobre essas ocorrências.

Além disso, a adaptação do classificador, treinado para identificar padrões textuais relacionados a crimes, representou uma contribuição importante, pois garantiu a precisão na identificação dos eventos, com uma taxa de acurácia de **96%** e uma pontuação F1 de **0.9649**. A implementação de um sistema de busca por localização, que permitiu identificar as regiões associadas aos crimes, foi uma adição significativa, pois proporcionou uma melhor organização e segmentação dos dados, facilitando a visualização de ocorrências específicas no aplicativo BHSafezone (PATRíCIO, 2023).

A integração com o Firestore foi outro ponto relevante, permitindo que o sistema oferecesse atualizações em tempo real, garantindo que as informações estivessem sempre acessíveis para os usuários do aplicativo. Essa solução não só beneficiou a população, oferecendo uma visualização detalhada das ocorrências, mas também contribuiu para a eficiência no monitoramento da segurança pública em Belo Horizonte.

Vale destacar que, até o momento da realização deste trabalho, não foi identificado nenhum outro aplicativo dedicado exclusivamente ao monitoramento da criminalidade na cidade de Belo Horizonte com base em dados coletados de redes sociais ou mensageiros instantâneos como o WhatsApp. Isso evidencia a contribuição da proposta e o seu potencial de impacto positivo na gestão da segurança urbana local.

Em resumo, as contribuições deste trabalho estão relacionadas à implementação de uma solução prática e eficaz para a coleta e análise de dados sobre criminalidade, utilizando tecnologias avançadas como inteligência artificial, processamento de linguagem natural e

bancos de dados em tempo real. Além disso, a pesquisa forneceu insights sobre a utilização de fontes não convencionais de dados, como as mensagens de grupos de WhatsApp, para auxiliar no monitoramento e na melhoria da segurança pública em áreas urbanas.

5.2 Limitações do Trabalho

Uma das principais limitações está relacionada à dependência da biblioteca `whatsapp-web.js` para a coleta de mensagens. Como essa biblioteca simula a interface web do WhatsApp, eventuais mudanças na plataforma ou restrições impostas pelo WhatsApp podem comprometer a continuidade da coleta de dados, exigindo manutenções frequentes ou a busca por alternativas. A biblioteca utilizada, *whatsapp-web.js*, possui algumas limitações quanto à captação de dados de grupos de WhatsApp que não possuem interação ativa e não havendo uma forma de obter as mensagens de canais, onde havia mais disponibilidade de noticias e jornais. Embora a solução tenha sido eficaz para a coleta contínua de dados em grupos de WhatsApp específicos, há uma dependência da adesão dos grupos à plataforma de coleta e à possibilidade de obter os dados a partir das interações dos usuários nesses grupos. Expandir a coleta para uma maior variedade de grupos ou integrar outras fontes de dados de grupos do WhatsApp poderia melhorar a amplitude da coleta, mas isso exigiria ajustes na estratégia de integração da plataforma.

Apesar dos avanços alcançados, algumas limitações ainda estão presentes no sistema desenvolvido. Uma das principais dificuldades está na obtenção de uma localização precisa a partir das mensagens de WhatsApp, uma vez que nem todas as mensagens incluem informações claras sobre o local do crime. Muitas vezes, as mensagens não especificam o bairro ou a região de maneira direta, o que impacta diretamente na acuracidade dos dados exibidos no mapa interativo do aplicativo BHSafezone (PATRíCIO, 2023). Essa limitação foi particularmente evidente em relação a **379** registros classificados como crimes, dos quais **233** estavam na cidade de Belo Horizonte, mas sem uma localização específica identificada.

A classificação das mensagens também apresenta limitações, pois depende da qualidade dos dados coletados. Mensagens irrelevantes ou com informações imprecisas podem interferir na precisão do sistema. Embora o classificador tenha alcançado uma taxa de acurácia de **96%**, ainda existe a possibilidade de erros na classificação, especialmente em mensagens ambíguas ou mal estruturadas. Além disso, o dicionário de palavras-chave utilizado no filtro de crimes requer manutenção constante, pois a linguagem utilizada nas mensagens pode mudar ao longo do tempo, o que implica a necessidade de atualização periódica das palavras-chave para garantir a eficiência do filtro.

Finalmente, a integração com o Firestore, embora eficiente, ainda apresenta desafios no que diz respeito à sincronização dos dados em tempo real. A otimização dessa integração

é crucial para garantir que o sistema continue a fornecer dados atualizados com rapidez, sem impactar o desempenho do aplicativo. Melhorias no gerenciamento de dados em tempo real e na escalabilidade do sistema serão essenciais para que o aplicativo continue a funcionar de forma eficiente à medida que mais dados sejam coletados.

5.3 Trabalhos Futuros

Com base nas limitações identificadas e nas oportunidades de aprimoramento, diversos pontos podem ser explorados em trabalhos futuros para otimizar o desempenho e a eficiência do sistema desenvolvido.

Uma das principais melhorias a serem consideradas é a implementação de um serviço intermediário entre o aplicativo BHSafezone (PATRíCIO, 2023) e o banco de dados. Atualmente, o aplicativo acessa diretamente os dados armazenados no Firestore, o que, apesar de ser funcional, pode gerar desafios relacionados à segurança, controle de acesso e escalabilidade. A criação de um serviço dedicado permitiria a implementação de autenticação robusta, cache para otimizar as consultas e políticas de acesso mais refinadas. Isso garantiria maior segurança no fornecimento das informações e facilitaria o controle dos dados exibidos, permitindo também a inclusão de filtros mais detalhados e opções de visualizações mais precisas.

Outra área promissora para o futuro é a expansão das fontes de dados utilizadas. Atualmente, a coleta de informações é limitada a grupos do WhatsApp, o que pode restringir a abrangência das ocorrências registradas. A integração com outras plataformas de comunicação, como Telegram, além da incorporação de fontes de dados governamentais, poderia oferecer uma visão mais abrangente e precisa da criminalidade. Isso permitiria ao sistema coletar uma gama mais rica de dados e, consequentemente, aprimorar a análise das ocorrências.

A extração de localização também pode ser aprimorada, utilizando técnicas mais avançadas de PLN e geocodificação. O uso de algoritmos de aprendizado de máquina pode ajudar na identificação automática de locais mencionados nas mensagens, mesmo quando esses endereços estão mal especificados ou contêm ambiguidade. Essa abordagem aumentaria a precisão das informações geográficas, permitindo uma visualização mais fiel e detalhada das ocorrências no aplicativo.

Além disso, é recomendável desenvolver uma estratégia para a atualização automática do modelo de classificação de crimes à medida que o banco de dados é atualizado com novas mensagens. Isso permitiria que o modelo se mantivesse constantemente treinado com os dados mais recentes, aumentando sua capacidade de adaptação a novas formas de linguagem, gírias ou padrões de relato presentes nas mensagens coletadas. Tal abordagem tornaria o sistema mais preciso e resiliente ao longo do tempo, reduzindo a necessidade de

reprocessamentos manuais e intervenções frequentes.

Por fim, a escalabilidade do sistema deve ser uma prioridade. A adoção de uma arquitetura baseada em microsserviços e o uso de tecnologias como filas de mensagens podem otimizar significativamente o processamento e o armazenamento dos dados. Esse tipo de arquitetura facilita a distribuição de cargas de trabalho e torna o sistema mais eficiente na coleta, processamento e disponibilização das informações. Além disso, a introdução de uma infraestrutura mais escalável ajudaria a garantir a performance mesmo à medida que o volume de dados aumentasse.

Essas melhorias permitirão que o sistema evolua para uma solução mais robusta, confiável e eficiente para o monitoramento da criminalidade em Belo Horizonte.

Referências

ADVERTISING, I.-D. *CrimeSpotter*. 2023. <https://play.google.com/store/apps/details?id=com.crimespotter>. Acesso em: 28-07-2023. Citado na página 21.

AKUMAEX. *e-Roubo Celular SP*. 2020. <https://play.google.com/store/apps/details?id=com.akumaex.e_roubo_celular>. Acesso em: 28-07-2023. Citado na página 21.

ALVES, L. G. A.; RIBEIRO, H. V.; RODRIGUES, F. A. Crime prediction through urban metrics and statistical learning. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 540, p. 123–138, 2020. Citado 3 vezes nas páginas 13, 14 e 15.

CARVALHO, W. S. *Reconhecimento de entidades mencionadas em português utilizando aprendizado de máquina*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado na página 40.

CHEN, X.; LIU, Y. Crime hotspot prediction using social media data and machine learning: A case study in new york city. *Computers, Environment and Urban Systems*, Elsevier, v. 99, p. 102–115, 2023. Citado 3 vezes nas páginas 13, 15 e 16.

CHEN, X.; LIU, Y.; ZHANG, Z. Covid-19 and social media: A systematic review of the literature. *Journal of Medical Internet Research*, JMIR Publications, v. 23, n. 4, p. e24682, 2021. Citado 2 vezes nas páginas 20 e 23.

CITIZEN. *Citizen: Safety in real-time*. 2021. Disponível em: <https://www.citizen.com/>. Citado na página 22.

CRUZADO, F. *Fogo Cruzado: Monitoramento de tiroteios*. 2022. Disponível em: <https://www.fogocruzado.org.br/>. Citado na página 22.

KUMAR, S.; SINGH, S. K. Real-time crime detection using social media analytics: A deep learning approach. *Journal of Big Data*, Springer, v. 9, n. 1, p. 1–20, 2022. Citado 2 vezes nas páginas 13 e 16.

LIMA, A.; COSTA, R. Integração de dados heterogêneos para monitoramento de segurança urbana: Um estudo de caso em belo horizonte. In: *Anais do XV Congresso Brasileiro de Sistemas de Informação (CBSI)*. Juiz de Fora - MG: Congresso Brasileiro de Sistemas de Informação (CBSI), 2023. Citado 2 vezes nas páginas 13 e 16.

LTD, L. A. W. *Crime Map*. 2023. <https://play.google.com/store/apps/details?id=com.londonappworks.cvu>. Acesso em: 28-07-2023. Citado na página 22.

OLIVEIRA, I.; LOPES Átila. Um aplicativo móvel para o registro e mapeamento de furtos e roubos em regiões metropolitanas. In: *Anais da X Escola Regional de Computação do Ceará, Maranhão e Piauí*. Porto Alegre, RS, Brasil: SBC, 2022. p. 149–158. Citado na página 21.

ONDEFUIROUBADO.COM.BR. *Onde fui roubado*. 2023. <https://www.ondefuiroubado.com.br>. Acesso em: 28-07-2023. Citado na página 21.

PATRíCIO, G. S. *Criação de um Aplicativo para Mapeamento da Criminalidade da Cidade de Belo Horizonte por meio de Atividade Crowdsourcing no Twitter*. 2023. Citado 16 vezes nas páginas 9, 17, 18, 30, 35, 36, 37, 44, 45, 46, 47, 48, 49, 50, 51 e 52.

PIRES, B. d. A. et al. A importância de grupos de whatsapp na detecção de eventos do mundo real. In: *Anais do Simpósio Brasileiro de Sistemas Colaborativos (SBSC)*. Porto Alegre: SBC, 2023. Citado na página 27.

RIBEIRO, M.; ALMEIDA, V. Análise de sentimentos em redes sociais para detecção de eventos de segurança pública. In: *Anais do XXIV Simpósio Brasileiro de Computação Aplicada à Segurança Pública (SBCASP)*. [S.l.]: Simpósio Brasileiro de Computação, 2023. Citado 5 vezes nas páginas 13, 15, 16, 21 e 23.

SILVA, F. F.; DUARTE, J. C.; UGULINO, W. C. *Automated statistics extraction of public security events reported through microtexts on social networks*. 2022. Citado 5 vezes nas páginas 13, 14, 16, 20 e 22.

SILVA, T.; STABILE, M. *Monitoramento e Pesquisa em Mídias Sociais: Metodologias, Aplicações e Inovações*. [S.l.]: E-book, 2016. Citado 4 vezes nas páginas 13, 14, 20 e 22.

SPOTCRIME. *SpotCrime: Crime mapping and alerts*. 2021. Disponível em: <https://www.spotcrime.com/>. Citado na página 22.

TELES, C. M.; SILVA, L. A. da. *Análise de dados e segurança pública utilizando web scraping e PLN*. 2021. Citado 4 vezes nas páginas 13, 14, 20 e 23.

WANG, Y.; TAYLOR, J. E. Social media for emergency management: A systematic review. *Cities*, Elsevier, v. 106, p. 102–115, 2020. Citado 2 vezes nas páginas 20 e 23.

ZANDAVALLE, A. C. *Monitoramento e Pesquisa em Mídias Sociais*. [S.l.]: Editora Comarte, 2016. Citado 4 vezes nas páginas 13, 14, 21 e 23.

6 Apêndice

Esta capítulo apresenta os principais trechos de código desenvolvidos ao longo do projeto, que foram fundamentais para a coleta, processamento e classificação das mensagens extraídas do WhatsApp. Os códigos incluídos abrangem desde o treinamento e funcionamento do classificador até os serviços responsáveis pela extração de localização, conexão com o banco de dados Firestore e envio dos dados processados. Abaixo, cada componente é detalhado de forma individual, contribuindo para uma compreensão mais completa da estrutura e funcionamento da aplicação.

Código 6.1 – Implementação da biblioteca whatsapp-web.js.

```
import { Injectable, OnModuleInit } from '@nestjs/common';
import { Client, LocalAuth } from 'whatsapp-web.js';
import * as qrcode from 'qrcode-terminal';
import * as fs from 'fs';
import * as path from 'path';

@Injectable()
export class WhatsappService implements OnModuleInit {
  private client: Client;

  constructor() {
    deleteFolders();

    this.client = new Client({
      authStrategy: new LocalAuth({
        clientId: 'bhsafezone-service',
      }),
      puppeteer: {
        headless: true,
      },
    });

    this.client.on('qr', (qr) => {
      qrcode.generate(qr, { small: true });
    });

    this.client.on('ready', () => {
      console.log('WhatsApp client is ready!');
    });

    this.client.on('message', (message) => {
      console.log(`Mensagem de ${message.from}: ${message.body}`);
    });
```

```
34
35          this.client.on('error', (error) => {
36            console.error('Erro no cliente:', error);
37          });
38        }
39
40        onModuleInit() {
41          this.client.initialize();
42        }
43
44        async fetchMessage(id: string): Promise<any> {
45          try {
46            const getChatById = await this.client.getChatById(id);
47
48            const messages = await getChatById.fetchMessages({
49              limit: 1000,
50              fromMe: false,
51            });
52
53            return messages;
54          } catch (error) {
55            console.log({ error });
56            return [];
57          }
58        }
59      }
```

Código 6.2 – Exemplo do serviço de coleta de mensagens.

```
1  import { Inject, Injectable } from '@nestjs/common';
2  import { WhatsappService } from '../../../../system/whatsapp/whatsapp.
       service';
3  import { WhatsAppRepository } from '../../repository/whatsapp.repository
       ';
4
5  @Injectable()
6  export class BhazapService {
7    constructor(
8      @Inject(WhatsappService)
9      private whatsappService: WhatsappService,
10
11     @Inject(WhatsAppRepository)
12     private whatsAppRepository: WhatsAppRepository,
13   ) {}
14
15   async execute(): Promise<void> {
16     const data = await this.whatsappService.fetchMessage(
17       '120363372671317259@g.us',
```

```
    );

    for (const item of data) {
      if (item.body !== '') {
        try {
          await this.whatsAppRepository.create({
            ...item,
            origin: 'bhazap',
            classified: 0,
          });
        } catch (error) {
          console.log({ error });
        }
      }
    }
  }
}
```

Código 6.3 – Exemplo de uso do cron job no NestJS.

```
import { Cron } from '@nestjs/schedule';
import { CronEnum } from '../../../../common/enums/cron.enum';
import { BhazapService } from './bhazap.service';

export class BhazapController {
  constructor(private readonly bhazapService: BhazapService) {}

  @Cron(CronEnum.EVERY_DAY_AT_10AM)
  async sendMessage(): Promise<void> {
    await this.bhazapService.execute();
  }
}
```

Código 6.4 – Filtro por palavras-chaves.

```
import { Injectable } from '@nestjs/common';

@Injectable()
export class FilterService {
  private crimeDictionary: Record<string, string> = {
    roubo: 'Roubo',
    roubado: 'Roubo',
    roubada: 'Roubo',
    furto: 'Furto',
    furtado: 'Furto',
    furtada: 'Furto',
    'batedor de carteira': 'Furto',
    trombadinha: 'Furto',
```

```
    assalto: 'Assalto',
    assaltada: 'Assalto',
    assaltado: 'Assalto',
    arrast o: 'Assalto',
    feminic dio: 'Feminic dio',
    estupro: 'Estupro',
    estuprado: 'Estupro',
    estuprada: 'Estupro',
    violentado: 'Estupro',
    violentada: 'Estupro',
    'viol ncia sexual': 'Estupro',
    'abuso sexual': 'Estupro',
    abusada: 'Estupro',
    abusado: 'Estupro',
    'importuna  o sexual': 'Estupro',
    extors o: 'Extors o',
    coagido: 'Extors o',
    coagir: 'Extors o',
    'les o corporal': 'Les o Corporal',
    briga: 'Les o Corporal',
    confronto: 'Les o Corporal',
    agress o: 'Les o Corporal',
    agredida: 'Les o Corporal',
    agredido: 'Les o Corporal',
    confus o: 'Les o Corporal',
    'viol ncia dom stica': 'Les o Corporal',
    sequestro: 'Sequestro',
    persegui  o: 'Sequestro',
    homic dio: 'Homic dio',
    assassinado: 'Homic dio',
    assassinada: 'Homic dio',
    'tr fico de drogas': 'Tr fico de Drogas',
    tr fico: 'Tr fico de Drogas',
    drogas: 'Tr fico de Drogas',
    maconha: 'Tr fico de Drogas',
    entorpecente: 'Tr fico de Drogas',
    coca na: 'Tr fico de Drogas',
    lsd: 'Tr fico de Drogas',
    ecstasy: 'Tr fico de Drogas',
    hero na: 'Tr fico de Drogas',
    'tentativa de homic dio': 'Tentativa de Homic dio',
    baleado: 'Tentativa de Homic dio',
    baleada: 'Tentativa de Homic dio',
    depreda  o: 'Depreda  o',
    picha  es: 'Depreda  o',
    picha  o: 'Depreda  o',
    vandalismo: 'Depreda  o',
```

```
    vandaliza  o: 'Depreda  o',
    inc ndio: 'Inc ndio',
    incendiar: 'Inc ndio',
    incendiou: 'Inc ndio',
    incendiaram: 'Inc ndio',
  };

  containsCrime(text: string): string | undefined {
    const lowerText = text.toLowerCase();
    for (const [term, crime] of Object.entries(this.crimeDictionary)) {
      if (lowerText.includes(term)) {
        return crime;
      }
    }
    return undefined;
  }
}
```

Código 6.5 – Código do Treinamento do Classificador.

```
import pandas as pd
import numpy as np
import random
import math
import re
import unicodedata
import nltk
import joblib
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer,
    TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import f1_score


nltk.download('stopwords')

# Stopwords personalizadas
stop_words = set(stopwords.words('portuguese'))
stop_words.update(['https', 'co', 'leia'])

# Pr -processamento
def remove_accents(text):
    return ''.join(c for c in unicodedata.normalize('NFKD', text) if not
        unicodedata.combining(c))

def clean_text(text):
```

```
27      text = re.sub(r'https?://\S+|www\.\S+', ' ', text)
28      text = re.sub(r'[,.:;!?/<>()[\]{}|\\+\-=%&#@\"\'\*]', ' ', text)
29      text = re.sub(r'\d+', '', text)
30      text = remove_accents(text)
31      text = text.lower()
32      words = [w for w in text.split() if w not in stop_words]
33      return ' '.join(words)
34
35  # Carrega os dados
36  df = pd.read_csv('src/modules/classifier/services/tweetsNoticiaCrime.csv
        ')
37
38  # Copia texto e limpa
39  df['novo_texto'] = df['Tweet'].astype(str).apply(clean_text)
40
41  nao_crime = df[df['Classe'] == 0]
42
43  amostras = nao_crime.sample(n=279, random_state=42, replace = True)
44
45  df = df[df['Classe'] != 0]
46
47  df = pd.concat([df, amostras])
48
49  # Remove pontua  o
50
51  df['novo_texto'] = df['novo_texto'].str.replace('[,.:;!?]+', ' ', regex=
        True).copy()
52
53  # Remove caracteres especiais
54  df['novo_texto'] = df['novo_texto'].str.replace ('[/<>()|\+\-\$%&#@
        \'\"]+', ' ', regex=True).copy()
55
56  # remove Numeros
57  df['novo_texto'] = df['novo_texto'].str.replace('[0-9]+', '', regex=True
        )
58
59  #StopWords
60  stop_words = ['em','sao','ao','de','da','do','para','c','kg','un','ml',
61                'pct','und','das','no','ou','pc','gr','pt','cm','vd','com'
                      ,
62                'sem','gfa','jg','la','1','2','3','4','5','6','7','8','9',
63                '0','a','b','c','d','e','lt','f','g','h','i','j','k','l',
64                'm','n','o','p','q','r','s','t','u','v','x','w','y','z',
65                'ate', 'eramos', 'estao', 'estavamos', 'estiveramos',
66                'estivessemos', 'foramos', 'fossemos', 'ha', 'hao',
67                'houveramos', 'houverao', 'houveriamos', 'houvessemos',
68                'ja', 'nao', 'sera', 'serao', 'seriamos', 'so', 'tambem',
```

```
69                'tera', 'terao', 'teriamos', 'tinhamos', 'tiveramos',
70                'tivessemos', 'voce', 'voces',"https","co","leia"]
71
72 for word in stopwords.words('portuguese'):
73     stop_words.append(word)
74
75 # Balanceamento simples
76 nao_crime = df[df['Classe'] == 0].sample(n=279, random_state=42, replace
       =True)
77 df = pd.concat([df[df['Classe'] != 0], nao_crime])
78
79 # Cria  o da fun  o CountVectorizer
80 cvt = CountVectorizer(strip_accents='ascii', lowercase=True, stop_words=
       stop_words)
81 X_cvt = cvt.fit_transform(df['novo_texto'])
82 tfi = TfidfTransformer()
83 X_tfi = tfi.fit_transform(X_cvt)
84 entrada = X_tfi.toarray()
85 saida = df['Classe']
86
87 # Testes com m ltiplos random_state
88 random_list = []
89 melhores_param = []
90 melhores_scores = []
91
92 for _ in range(5):
93     seed = random.randint(0, 100)
94     random_list.append(seed)
95
96     X_train, _, y_train, _ = train_test_split(entrada, saida, test_size
           =0.2, random_state=seed)
97
98     clf = MultinomialNB()
99     param_grid = {
100         'alpha': [0.1, 0.01, 0.001],
101         'fit_prior': [True, False],
102         'class_prior': [(0.5, 0.5), (0.2, 0.8)]
103     }
104
105     grid = GridSearchCV(clf, param_grid, cv=10, scoring='f1', verbose=0)
106     grid.fit(X_train, y_train)
107
108     melhores_param.append(grid.best_params_)
109     melhores_scores.append(grid.best_score_)
110
111     print(f"Seed: {seed} | Best Params: {grid.best_params_} | F1 Score:
           {grid.best_score_:.4f}")
```

```
112
113 # Resultados finais
114 media = np.mean(melhores_scores)
115 std = np.std(melhores_scores)
116 conf_int = 2.131 * std / math.sqrt(len(melhores_scores))
117
118 print("\nResumo Final:")
119 print("Seeds testadas:", random_list)
120 print("Melhores par metros:", melhores_param)
121 print(f"M dia F1: {media:.4f}")
122 print(f"Intervalo de confian a (95%): [{media - conf_int:.4f}, {media +
        conf_int:.4f}]")
123
124 # Ap s o melhor treinamento
125 melhor_index = np.argmax(melhores_scores)
126 melhor_seed = random_list[melhor_index]
127
128 # Refaz o melhor modelo com a melhor seed
129 X_train, _, y_train, _ = train_test_split(entrada, saida, test_size=0.2,
        random_state=melhor_seed)
130
131 clf_final = MultinomialNB(**melhores_param[melhor_index])
132 clf_final.fit(X_train, y_train)
133
134 # Salva os modelos
135 joblib.dump(clf_final, 'src/modules/classifier/services/
       modelo_classificador.pkl')
136 joblib.dump(cvt, 'src/modules/classifier/services/count_vectorizer.pkl')
137 joblib.dump(tfi, 'src/modules/classifier/services/tfidf_transformer.pkl'
       )
138
139 print("Modelos salvos com sucesso.")
```

Código 6.6 – Código do classificador.

```
1 import joblib
2 import re
3 import unicodedata
4 from nltk.corpus import stopwords
5 import pymongo
6
7 # Carregar o classificador, CountVectorizer e TfidfTransformer
8 clf = joblib.load('src/modules/classifier/services/modelo_classificador.
       pkl')
9 cvt = joblib.load('src/modules/classifier/services/count_vectorizer.pkl'
       )
10 tfi = joblib.load('src/modules/classifier/services/tfidf_transformer.pkl
       ')
```

```
11
12 # Stopwords
13 stop_words = set(stopwords.words('portuguese'))
14 stop_words.update([
15     'em','sao','ao','de','da','do','para','c','kg','un','ml','pct','und'
       ,'das',
16     'no','ou','pc','gr','pt','cm','vd','com','sem','gfa','jg','la','1','
       2','3',
17     '4','5','6','7','8','9','0','a','b','c','d','e','lt','f','g','h','i'
       ,'j','k',
18     'l','m','n','o','p','q','r','s','t','u','v','x','w','y','z','ate', '
       eramos',
19     'estao', 'estavamos', 'estiveramos','estivessemos', 'foramos', '
       fossemos',
20     'ha', 'hao','houveramos', 'houverao', 'houveriamos', 'houvessemos','
       ja',
21     'nao', 'sera', 'serao', 'seriamos', 'so', 'tambem','tera', 'terao',
22     'teriamos', 'tinhamos', 'tiveramos','tivessemos', 'voce', 'voces',
23     "https","co","leia"
24 ])
25
26 def remove_accents(text):
27     text = unicodedata.normalize('NFKD', text)
28     text = ''.join([c for c in text if not unicodedata.combining(c)])
29     return text
30
31 def format_text(text):
32     text = re.sub(r'https?://\S+|www\.\S+', ' ', text)
33     text = re.sub(r'[,.:;!?]+', ' ', text)
34     text = re.sub(r'[/<>()|\+\-\$%&#\*@\'\"]+', ' ', text)
35     text = re.sub(r'[0-9]+', '', text)
36     text = remove_accents(text)
37     text = re.sub(r'\s+', ' ', text).strip().lower()
38
39     return text
40
41 def classifica_crime(text, clf, cvt, tfi):
42     print('text1:', text)
43     text = format_text(text)
44     print('text2:', text)
45
46     X_cvt = cvt.transform([text])
47     X_tfi = tfi.transform(X_cvt)
48
49     print('X_tfi:', X_tfi.toarray())
50     classe = clf.predict(X_tfi)[0]
51     print('classe:', classe)
```

```
52      return classe
53
54  def salva_e_classifica(data, mongo_db, mongo_db_coll, clf, cvt, tfi, **
        mongo_conn_kw):
55      client = pymongo.MongoClient(**mongo_conn_kw)
56      db = client[mongo_db]
57      coll = db[mongo_db_coll]
58
59      for item in data:
60          if any(item.get(key) for key in ['body', 'title', 'description
                ']):
61              texto = f"{item.get('title', '')} {item.get('description', '
                    ')} {item.get('body', '')}"
62              resul = classifica_crime(texto, clf, cvt, tfi)
63
64              if resul == 1:
65                  coll.update_one(
66                      {'_id': item['_id']},
67                      {'$set': {'is_crime': 1, 'classified': 1, '
                            found_location': 0}}
68                  )
69              else:
70                  coll.update_one(
71                      {'_id': item['_id']},
72                      {'$set': {'is_crime': 0, 'classified': 1, '
                            found_location': 0}}
73                  )
74          else:
75              print('Sem dados v lidos para classificar:', item)
76
77  def salva_Tweets_classificados_crime(clf, cvt, tfi):
78      client = pymongo.MongoClient()
79      db = client["bh-safezone"]
80      mycol = db["whatsapps"]
81      data = mycol.find({"classified": {"$ne": 0}})  # Pega s   os n o
            classificados
82
83      t = data.clone()
84      tt = data.clone()
85      c = len(list(t))
86
87      if c >= 1:
88          salva_e_classifica(tt, "bh-safezone", "whatsapps", clf, cvt, tfi
                )
89
90
91  salva_Tweets_classificados_crime(clf, cvt, tfi)
```

Código 6.7 – Busca pela localização do crime.

```
import pymongo
import re
import spacy
import requests

client = pymongo.MongoClient("localhost",27017)

# Rela   o bairro/regi o

bairro_regiao = {
 'AAR O REIS': 'NORTE',
 'ACABA MUNDO': 'CENTRO-SUL',
 'ACAIACA': 'NORDESTE',
 'ADEMAR MALDONADO': 'BARREIRO',
 .
 .
 .
 'VILA NOVA CACHOEIRINHA I': 'NOROESTE',
 'VILA S O GABRIEL': 'NORDESTE'
}

# Lista de bairros
bairros = ['aar o reis',
 'acaba mundo',
 'acaiaca',
 'ademar maldonado',
 'aeroporto',
 .
 .
 .
 'universit rio',
 'vila de s  ',
 'vila nova cachoeirinha i',
 'vila s o gabriel']


def encontra_Regiao(bairro):
    regiao = bairro_regiao[bairro.upper()]

    return regiao

def regioes():

    reg =     [["CENTRO-SUL","Regi o Centro-Sul","Regi o Centro-Sul de
        Belo Horizonte","Regi o Centro Sul", "Regiao Centro-Sul da
       capital","Regi o Centro-Sul de BH","Centro-Sul","Centro Sul"],
```

```
45                ["LESTE","Regi o Leste","Regi o Leste de Belo Horizonte
                     ","Leste","Regi o Leste da capital","Regi o Leste de
                      BH"],
46                ["NORDESTE","Regi o Nordeste","Regi o Nordeste de Belo
                     Horizonte","Regi o Nordeste da Capital","Regi o
                     Nordeste de BH"],
47                ["NORTE","Regi o Norte","Regi o Norte de Belo Horizonte
                     ","Regi o Norte da Capital","Regi o Norte de BH"],
48                ["VENDA NOVA","Venda Nova","Regi o Venda Nova de Belo
                     Horizonte","Regi o Venda Nova da Capital","Regi o
                     Venda Nova de BH"],
49                ["PAMPULHA","Regiao da Pampulha","Regi o da Pampulha de
                     Belo Horizonte","Regi o da Pampulha de BH","Pampulha"
                     ],
50                ["NOROESTE","Regi o Noroeste","Regi o Noroeste de Belo
                     Horizonte","Regi o Noroeste da Capital","Regi o
                     Noroeste de BH"],
51                ["OESTE","Regi o Oeste","Regi o Oeste de Belo Horizonte
                     ","Regi o Oeste da Capital","Regi o Oeste de BH"],
52                ["BARREIRO","Regi o do Barreiro","Regi o do Barreiro de
                      Belo Horizonte","Regi o do Barreiro da Capital","
                     Regi o do Barreiro de BH","Barreiro","Regi o
                     Barreiro"]]
53
54        return reg;
55
56
57    def avenidas(loc):
58        try:
59            url = f'http://geocoder.pbh.gov.br/geocoder/v2/address?
                 logradouro={loc.replace(" ", "%20")}'
60            headers = {
61                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0
                     Safari/537.36"
62            }
63            r = requests.get(url, headers=headers)
64            print(f"URL solicitada: {url}")  # Debug: Printar a URL
                 solicitada
65
66            if r.status_code == 200:
67                try:
68                    avenida = r.json()
69                    print(f"Resposta JSON: {avenida}")  # Debug: Printar a
                         resposta JSON
70                    if 'endereco' in avenida and len(avenida['endereco']) >
                         0:
```

```
71                      bairro = avenida['endereco'][0].get('bairropopular',
                            'Desconhecido')
72                      regiao = avenida['endereco'][0].get('nomeregional',
                           'Desconhecido')
73
74                      return [bairro, regiao]
75                  else:
76                      print("Endere o  n  o  encontrado")
77                      return "Endere o  n  o  encontrado"
78              except ValueError as e:
79                  print(f"Erro ao decodificar JSON: {e}")
80                  print(f"Resposta do servidor: {r.text}")  # Debug:
                        Printar a resposta bruta do servidor
81                  return "Erro ao decodificar JSON"
82          else:
83              print(f"Erro na solicita  o HTTP: {r.status_code}")
84              return "Erro na solicita  o HTTP"
85      except requests.RequestException as e:
86          print(f"Erro ao fazer a solicita  o HTTP: {e}")
87          return "Erro ao fazer a solicita  o HTTP"
88
89
90  def viadutos(loc):
91      try:
92          # Adicionar cabe alho User-Agent para simular uma requisi  o
                de navegador
93          headers = {
94              'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                    AppleWebKit/537.36 (KHTML, like Gecko) Chrome
                    /91.0.4472.124 Safari/537.36'
95          }
96
97          # Construir a URL com o logradouro
98          url = f'http://geocoder.pbh.gov.br/geocoder/v2/address?
                logradouro={loc}'
99
100         # Fazer a requisi  o para a API
101         r = requests.get(url, headers=headers)
102
103         # Verificar se a resposta foi bem-sucedida
104         if r.status_code == 200:
105             try:
106                 # Tentar decodificar a resposta JSON
107                 viaduto = r.json()
108
109                 # Verificar se h  dados de 'endereco' e se a lista n o
                        est  vazia
```

```
110                 if 'endereco' in viaduto and len(viaduto['endereco']) >
                        0:
111                     for end in viaduto['endereco']:
112                         # Verificar se o tipo de logradouro    'VIADUTO'
113                         if end.get('tipologradouro') == 'VIADUTO':
114                             regiao = end.get('nomeregional', '
                                    Desconhecida')
115                             bairro = end.get('bairropopular', '
                                    Desconhecida')
116                             return [bairro, regiao]
117                 # Caso n o encontre o viaduto ou os dados
118                 return ['Desconhecido', 'Desconhecida', "N o_BH"]
119
120             except ValueError as e:
121                 # Erro ao decodificar JSON
122                 print(f"Erro ao decodificar JSON: {e}")
123                 return ['Desconhecido', 'Desconhecida', "N o_BH"]
124
125         elif r.status_code == 403:
126             # Lidar com o erro 403 (Proibido)
127             print(f"Erro 403: Acesso proibido. Verifique as permiss es
                    ou limites da API.")
128             return ['Desconhecido', 'Desconhecida', "N o_BH"]
129
130         else:
131             # Se a resposta n o for 200 ou 403, exibir erro
132             print(f"Erro na solicita  o HTTP: {r.status_code}")
133             return ['Desconhecido', 'Desconhecida', "N o_BH"]
134
135     except requests.RequestException as e:
136         # Exce  es relacionadas    requisi  o HTTP
137         print(f"Erro ao fazer a solicita  o HTTP: {e}")
138         return ['Desconhecido', 'Desconhecida', "N o_BH"]
139
140
141 def ruas(loc):
142     try:
143         url = f'http://geocoder.pbh.gov.br/geocoder/v2/address?
                logradouro={loc}'
144         headers = {
145             "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0
                    Safari/537.36"
146         }
147         r = requests.get(url, headers=headers)
148
149         if r.status_code == 200:
```

```
150             try:
151                 rua = r.json()
152                 if 'endereco' in rua and len(rua['endereco']) > 0:
153                     bairro = rua['endereco'][0].get('bairropopular', '
                            Desconhecido')
154                     regiao = rua['endereco'][0].get('nomeregional', '
                            Desconhecido')
155
156                     return [bairro, regiao]
157                 else:
158                     return ['Desconhecido', 'Desconhecida', "N o_BH"]
159             except ValueError as e:
160                 print(f"Erro ao decodificar JSON: {e}")
161                 return ['Desconhecido', 'Desconhecida', "N o_BH"]
162         else:
163             print(f"Erro na solicita  o HTTP: {r.status_code}")
164             return ['Desconhecido', 'Desconhecida', "N o_BH"]
165     except requests.RequestException as e:
166         print(f"Erro ao fazer a solicita  o HTTP: {e}")
167         return ['Desconhecido', 'Desconhecida', "N o_BH"]
168
169
170
171 def save_to_mongo_crime(termo, tt, db_name, perfil):
172     # Fun  o exemplo para salvar no MongoDB
173     client = pymongo.MongoClient("localhost", 27017)
174     db = client[db_name]
175     collection = db[perfil]
176     document = {"termo": termo, "tt": tt}
177     collection.insert_one(document)
178
179
180 def pracas(loc):
181     try:
182         # Adicionando o cabe alho User-Agent para simular uma
                requisi  o de navegador
183         headers = {
184             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                    AppleWebKit/537.36 (KHTML, like Gecko) Chrome
                    /91.0.4472.124 Safari/537.36'
185         }
186         url = f'http://geocoder.pbh.gov.br/geocoder/v2/address?
                logradouro={loc}'
187         r = requests.get(url, headers=headers)
188         print(f"URL solicitada 2: {url}")  # Debug: Printar a URL
                solicitada
189
```

```
190         if r.status_code == 200:
191             try:
192                 praca = r.json()
193                 print(f"Resposta JSON: {praca}")  # Debug: Printar a
                        resposta JSON
194                 if 'endereco' in praca and len(praca['endereco']) > 0:
195                     bairro = praca['endereco'][0].get('bairropopular', '
                            Desconhecido')
196                     regiao = praca['endereco'][0].get('nomeregional', '
                            Desconhecido')
197
198                     return [bairro, regiao]
199                 else:
200                     print("Endere o  n  o  encontrado")
201                     return "Endere o  n  o  encontrado"
202             except ValueError as e:
203                 print(f"Erro ao decodificar JSON: {e}")
204                 print(f"Resposta do servidor: {r.text}")  # Debug:
                        Printar a resposta bruta do servidor
205                 return "Erro ao decodificar JSON"
206         else:
207             print(f"Erro na solicita  o HTTP: {r.status_code}")
208             return "Erro na solicita  o HTTP"
209     except requests.RequestException as e:
210         print(f"Erro ao fazer a solicita  o HTTP: {e}")
211         return "Erro ao fazer a solicita  o HTTP"
212
213
214 def salva_tweet_local(bairro,regiao,data, mongo_db, mongo_db_coll, **
        mongo_conn_kw):
215
216     client = pymongo.MongoClient(**mongo_conn_kw)
217
218     db = client[mongo_db]
219
220     coll = db[mongo_db_coll]
221
222     data['regiao'] = regiao.upper()
223     data['bairro'] = bairro.lower()
224     coll.update_one(
225     {'_id': data['_id']},
226     {'$set': {'is_crime': 1,
227               'found_location': 1,
228               'region': regiao.lower(),
229               'bairro': bairro.lower(),
230               'integrated': 0
231               }}
```

```
232     )
233
234 def encontra_local_link(txt,bairros,nlp):
235     doc = nlp(txt)
236     print(doc)
237     for ent in doc.ents:
238
239         if(ent.label_ == 'LOC'):
240
241             loc = str(ent).lower()
242
243             if loc in bairros:
244                 regiao = encontra_Regiao(loc)
245                 bairro = loc
246                 return [bairro,regiao];
247
248             elif("avenida" in loc):
249                 avenida = avenidas(loc[8:])
250                 return avenida;
251
252             elif("rua" in loc):
253                 rua = ruas(loc[4:])
254                 return rua;
255
256             elif("pra a" in loc):
257                 praca = pracas(loc[6:])
258                 print(praca)
259                 return praca;
260
261             elif("viaduto" in loc):
262                 viaduto = viadutos(loc[8:])
263                 return viaduto;
264
265     regi = regioes()
266     for regiao in regi:
267         for x in regiao:
268             if x.lower() in txt.lower():
269                 return ["Desconhecido",regiao[0],regiao[0]];
270
271
272     if "metropolitana" in txt.lower():
273         return ["Desconhecido","Regi o Metropolitana","
                Regi o_Metropolitana"];
274
275     elif "belo horizonte" in txt.lower():
276         return ["Desconhecido","Desconhecida","Geral"];
277
```

```
278     elif "BH" in txt:
279         return ["Desconhecido","Desconhecida","Geral"];
280 
281     elif "#BH" in txt:
282         return ["Desconhecido","Desconhecida","Geral"];
283 
284     elif "capital" in txt.lower():
285         return ["Desconhecido","Desconhecida","Geral"];
286 
287     return ["Desconhecido","Desconhecido","N o_BH"];
288 
289 
290 def encontra_texto_link(link):
291     from bs4 import BeautifulSoup
292     from requests.adapters import HTTPAdapter
293     from requests.packages.urllib3.util.retry import Retry
294 
295     retry_strategy = Retry(
296       total=20,
297       backoff_factor=1
298     )
299 
300 
301     adapter = HTTPAdapter(max_retries=retry_strategy)
302     http = requests.Session()
303     http.mount("https://", adapter)
304     http.mount("http://", adapter)
305 
306 
307     texto = ''
308 
309     html = http.get(link).content
310     soup = BeautifulSoup(html, 'html.parser')
311 
312     print("Texto:", texto)
313 
314     cleantext = BeautifulSoup(texto, "lxml").text
315 
316     caracteres = ["/","<",">","(",")","|",'[',']','*']
317 
318     for c in caracteres:
319         cleantext = cleantext.replace(c, ' ')
320 
321     return cleantext
322 
323 def encontra_local(item,bairros,nlp):
324 
```

```
325     txt = f"{item.get('title', '')} {item.get('description', '')} {item.
            get('body', '')} "
326
327     doc = nlp(txt)
328
329     for ent in doc.ents:
330         if(ent.label_ == 'LOC'):
331             loc = str(ent).lower()
332
333             if loc in bairros:
334                 regiao = encontra_Regiao(loc)
335                 bairro = loc
336                 return [bairro,regiao];
337
338             elif("avenida" in loc):
339                 avenida = avenidas(loc[8:])
340                 return avenida;
341
342             elif("rua" in loc):
343                 rua = ruas(loc[4:])
344                 return rua;
345
346             elif("pra a" in loc):
347                 praca = pracas(loc[6:])
348                 return praca;
349
350             elif("viaduto" in loc):
351                 viaduto = viadutos(loc[8:])
352                 return viaduto;
353
354
355     if(len(item['links']) > 0):
356         link = item['links'][0]['link']
357         textoLink = encontra_texto_link(link)
358         resul = encontra_local_link(textoLink,bairros,nlp)
359         return resul
360
361     regi = regioes()
362     for regiao in regi:
363         for x in regiao:
364             if x.lower() in txt.lower():
365                 return ["Desconhecido",regiao[0],regiao[0]];
366
367
368     if "metropolitana" in txt.lower():
369         return ["Desconhecido","Regi o Metropolitana","
                Regi o_Metropolitana"];
```

```
370
371     elif "belo horizonte" in txt.lower():
372         return ["Desconhecido","Desconhecida","Geral"];
373
374     elif "BH" in txt:
375         return ["Desconhecido","Desconhecida","Geral"];
376
377     elif "#BH" in txt:
378         return ["Desconhecido","Desconhecida","Geral"];
379
380     elif "capital" in txt.lower():
381         return ["Desconhecido","Desconhecida","Geral"];
382
383     return ["Desconhecido","Desconhecido","N o_BH"];
384
385
386 def filtra_local(bairros):
387     import pymongo
388     import re
389     import spacy
390     bairros2 = []
391
392     for bairro in bairros:
393         bairros2.append(bairro.lower())
394
395     nlp = spacy.load('pt_core_news_lg')
396
397     client = pymongo.MongoClient("localhost", 27017)
398
399     db = client["bh-safezone"]
400
401     mycol = db["whatsapps"]
402     data = mycol.find({"is_crime": 1, "found_location": 0})
403     t = data.clone()
404
405     for item in data:
406         lista = encontra_local(item,bairros2,nlp)
407
408         print("Lista:", lista)
409
410         if(len(lista)<3):
411             salva_tweet_local(lista[0],lista[1],item,"bh-safezone","
                whatsapps")
412         else:
413             salva_tweet_local(lista[0],lista[1],item,"bh-safezone","
                whatsapps")
414
```

```
415
416
417 filtra_local(bairros)
```

Código 6.8 – Serviço de conecção com firestore.

```
1  import { Injectable, OnModuleInit } from '@nestjs/common';
2  import * as admin from 'firebase-admin';
3  import * as fs from 'fs';
4
5  @Injectable()
6  export class FirebaseService implements OnModuleInit {
7    private db: FirebaseFirestore.Firestore;
8
9    onModuleInit() {
10     if (!admin.apps.length) {
11       const serviceAccount = JSON.parse(
12         fs.readFileSync('src/firebase-key.json', 'utf8'),
13       );
14
15       admin.initializeApp({
16         credential: admin.credential.cert(serviceAccount),
17       });
18     }
19
20     this.db = admin.firestore();
21   }
22
23   async salveCrime(dados: any): Promise<string> {
24     const crimesRef = this.db.collection('crimes');
25     const docRef = await crimesRef.add(dados);
26     return docRef.id;
27   }
28
29   async upsertCrimeRegion(colecao: string, crime: string, quantidade:
       number) {
30     try {
31       const snapshot = await this.db
32         .collection(colecao)
33         .where('crime', '==', crime)
34         .get();
35
36       if (!snapshot.empty) {
37         snapshot.forEach(async (doc) => {
38           const quantidadeAtual = parseInt(doc.data().quantidade) || 0;
39
40           await doc.ref.update({ quantidade: quantidadeAtual +
               quantidade });
```

```
41          console.log(
42            `Atualizado: ${crime} na cole  o ${colecao}, nova
                 quantidade: ${quantidadeAtual + quantidade}`,
43          );
44        });
45      } else {
46        await this.db.collection(colecao).add({
47          crime: crime,
48          quantidade: quantidade,
49        });
50        console.log(
51          `Novo crime adicionado: ${crime} na cole  o ${colecao} com
                 quantidade: ${quantidade}`,
52        );
53      }
54    } catch (error) {
55      console.log('Cole  o recebida:', colecao);
56
57      console.error('Erro ao atualizar ou criar crime:', error);
58    }
59  }
60
61  async upsertCrimeBairro(colecao: string, crime: string, quantidade:
       number) {
62    try {
63      const snapshot = await this.db
64        .collection(colecao)
65        .where('crime', '==', crime)
66        .get();
67
68      if (!snapshot.empty) {
69        snapshot.forEach(async (doc) => {
70          const quantidadeAtual = parseInt(doc.data().quantidade) || 0;
71
72          await doc.ref.update({ quantidade: quantidadeAtual +
                 quantidade });
73          console.log(
74            `Atualizado: ${crime} na cole  o ${colecao}, nova
                 quantidade: ${quantidadeAtual + quantidade}`,
75          );
76        });
77      } else {
78        await this.db.collection(colecao).add({
79          crime: crime,
80          quantidade: quantidade,
81        });
82        console.log(
```

```
83          `Novo crime adicionado: ${crime} na cole  o ${colecao} com
                quantidade: ${quantidade}`,
84        );
85      }
86    } catch (error) {
87      console.log('Cole  o recebida:', colecao);
88
89      console.error('Erro ao atualizar ou criar crime:', error);
90    }
91  }
92
93  async upsertCrimePorBairro(
94    colecao: string,
95    bairro: string,
96    crime: string,
97    quantidade: number,
98  ) {
99    try {
100     const bairroRef = this.db
101       .collection(colecao)
102       .doc('bairros')
103       .collection(bairro);
104
105     const crimeSnapshot = await bairroRef.where('crime', '==', crime).
          get();
106
107     if (!crimeSnapshot.empty) {
108       crimeSnapshot.forEach(async (doc) => {
109         const dadosCrime = doc.data();
110         const quantidadeAtual = +dadosCrime.quantidade || 0;
111
112         await doc.ref.update({ quantidade: quantidadeAtual +
              quantidade });
113         console.log(
114           `Atualizado: ${crime} no bairro ${bairro}, nova quantidade:
                ${quantidadeAtual + quantidade}`,
115         );
116       });
117     } else {
118       await bairroRef.add({ crime, quantidade });
119       console.log(
120         `Novo crime adicionado: ${crime} no bairro ${bairro} com
              quantidade: ${quantidade}`,
121       );
122
123       const bairrosDocRef = this.db.collection(colecao).doc('bairros')
            ;
```

```
124         const bairrosDocSnapshot = await bairrosDocRef.get();
125         if (bairrosDocSnapshot.exists) {
126           const bairrosData = bairrosDocSnapshot.data()?.bairros || [];
127
128           if (!bairrosData.includes(bairro)) {
129             await bairrosDocRef.update({ bairros: [...bairrosData,
                    bairro] });
130             console.log(
131               `Bairro ${bairro} adicionado ao campo "data" do documento
                      "bairros".`,
132             );
133           }
134         } else {
135           await bairrosDocRef.set({ bairros: [bairro] }, { merge: true
                  });
136           console.log(
137             `Campo "data" criado no documento "bairros" com o bairro ${
                    bairro}.`,
138           );
139         }
140       }
141     } catch (error) {
142       console.log('Cole  o recebida:', colecao);
143       console.log('Bairro recebido:', bairro);
144       console.error('Erro ao atualizar ou criar crime no bairro:', error
              );
145     }
146   }
147
148   async upsertCrimePorTimeSeries(
149     colecao: string,
150     document: string,
151     date: string,
152     crime: string,
153     quantidade: number,
154   ) {
155     try {
156       const bairroRef = this.db
157         .collection(colecao)
158         .doc(document)
159         .collection(`${colecao} ${date}`);
160
161       const crimeSnapshot = await bairroRef.where('crime', '==', crime).
              get();
162
163       if (!crimeSnapshot.empty) {
164         crimeSnapshot.forEach(async (doc) => {
```

```
165           const dadosCrime = doc.data();
166           const quantidadeAtual = +dadosCrime.quantidade || 0;
167
168           await doc.ref.update({ quantidade: quantidadeAtual +
                  quantidade });
169           console.log(
170             `Atualizado: ${crime} no bairro ${colecao}, nova quantidade:
                  ${quantidadeAtual + quantidade}`,
171           );
172         });
173       } else {
174         await bairroRef.add({ crime, quantidade });
175         console.log(
176           `Novo crime adicionado: ${crime} no bairro ${colecao} com
                quantidade: ${quantidade}`,
177         );
178
179         const bairrosDocRef = this.db.collection(colecao).doc(document);
180         const bairrosDocSnapshot = await bairrosDocRef.get();
181         if (bairrosDocSnapshot.exists) {
182           const bairrosData = bairrosDocSnapshot.data()?.data || [];
183
184           if (!bairrosData.includes(date)) {
185             await bairrosDocRef.update({ data: [...bairrosData, date] })
                    ;
186             console.log(
187               `Bairro ${date} adicionado ao campo "data" do documento "
                      bairros".`,
188             );
189           }
190         } else {
191           await bairrosDocRef.set({ data: [date] }, { merge: true });
192           console.log(
193             `Campo "data" criado no documento "bairros" com o bairro ${
                    colecao}.`,
194           );
195         }
196       }
197     } catch (error) {
198       console.log('Cole   o recebida:', colecao);
199       console.log('Bairro recebido:', colecao);
200       console.error('Erro ao atualizar ou criar crime no bairro:', error
              );
201     }
202   }
203 }
```

Código 6.9 – Serviço responsavel pelo envio dos dados para o firestore.

```
import { Inject, Injectable } from '@nestjs/common';
import { WhatsAppRepository } from '../../../whatsApp/repository/
    whatsapp.repository';
import { FirebaseService } from '../conection/firebase.service';
import { RegionEnum } from '../../enum/Region.enum';

@Injectable()
export class IntegratedInFireStoreService {
  constructor(
    @Inject(WhatsAppRepository)
    private whatsAppRepository: WhatsAppRepository,

    @Inject(FirebaseService)
    private firebaseService: FirebaseService,
  ) {}

  async execute(): Promise<any> {
    const data = await this.whatsAppRepository.getByIntegratedFalse();

    for (const item of data) {
      await this.firebaseService.upsertCrimeRegion(
        RegionEnum[item.region.toLocaleUpperCase()],
        item.crime,
        1,
      );

      await this.firebaseService.upsertCrimePorBairro(
        RegionEnum[item.region.toLocaleUpperCase()],
        item.bairro,
        item.crime,
        1,
      );

      await this.firebaseService.upsertCrimePorTimeSeries(
        RegionEnum[item.region.toLocaleUpperCase()],
        'time_series',
        formatarMesAnoNumerico(item.created_at),
        item.crime,
        1,
      );

      await this.firebaseService.upsertCrimeRegion(
        'Geral',
        RegionEnum[item.region.toLocaleUpperCase()],
        1,
      );
```

```
46
47        await this.firebaseService.upsertCrimePorTimeSeries(
48          'Geral',
49          'time_series',
50          formatarMesAnoNumerico(item.created_at),
51          item.crime,
52          1,
53        );
54
55        await this.whatsAppRepository.update(item._id, {
56          integrated: 1,
57        });
58      }
59    }
60  }
61
62  function formatarMesAnoNumerico(data: Date): string {
63    const mes = String(data.getMonth() + 1).padStart(2, '0');
64    const ano = data.getFullYear();
65
66    return `${mes}-${ano}`;
67  }
```