Universidade Católica do Salvador
Bacharelado em Engenharia de Software

Ana Carolina Armentano
Ellen Sampaio Cerqueira
Lavínia Énora Santana Cunha
Luma Abram Cecilia
Maria Eduarda Pamponet Ramalho

Análise de Comentários de Lives no YouTube: Extração de Métricas em Tempo Real com Apoio de Modelos de Linguagem (LLM)

Salvador
2025

Ana Carolina Armentano
Ellen Sampaio Cerqueira
Lavínia Énora Santana Cunha
Luma Abram Cecilia
Maria Eduarda Pamponet Ramalho

Análise de Comentários de Lives no YouTube: Extração de Métricas em Tempo Real com Apoio de Modelos de Linguagem (LLM)

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software. Orientador: Prof. Angela Peixoto

Universidade Católica do Salvador

Salvador
2025

Ana Carolina Armentano
Ellen Sampaio Cerqueira
Lavínia Énora Santana Cunha
Luma Abram Cecilia
Maria Eduarda Pamponet Ramalho

Análise de Comentários de Lives no YouTube: Extração de Métricas em Tempo Real com Apoio de Modelos de Linguagem (LLM)

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como requisito parcial para a obtenção do título de Engenheiro de Software.

Salvador, 11 de maio de 2026
Banca Examinadora:

Prof. Angela Peixoto
Universidade Católica do Salvador
Orientador

Prof. Glaucya Boechat
Universidade Católica do Salvador

Prof. Cristiana Bispo
Universidade Católica do Salvador

Agradecimentos

Eu, Ana Carolina Armentano e Silva, agradeço aos meus pais, Mariângela e Luis Augusto, pelo incentivo constante à minha educação, me proporcionando as bases para chegar até aqui. Um agradecimento especial à minha esposa, Ingryd. Sua compreensão, paciência e amor irrestritos foram fundamentais para essa conquista. Obrigada por ser meu porto seguro e por dividir a vida e os sonhos comigo. Por fim, agradeço às minhas colegas e à nossa orientadora. Que possamos, enquanto mulheres, seguir ocupando espaços relevantes na Engenharia de Software com excelência e protagonismo.

Eu, Ellen Sampaio Cerqueira, agradeço a Deus pela vida, agradeço profundamente aos meus pais Diani Cristina e Geraldo Cerqueira. Se hoje estou aqui, é graças ao suporte e ao amor de vocês. Agradeço ao meu namorado, meu parceiro de vida e de estudos, obrigada por me apoiar e sempre estar presente quando precisei. Estendo meus agradecimentos aos seus pais, pelo carinho e acolhimento. Por fim, agradeço imensamente às minhas colegas, pela parceria nos trabalhos e pela cumplicidade que construímos. Obrigada por fazerem parte da minha história.

Eu, Lavínia Énora Santana Cunha, agradeço primeiramente a Deus, por abençoar e iluminar cada passo da minha trajetória. Aos meus pais, Elizane e Renato, por tanto amor, orientação e por serem minha base, sem eles nada disso seria possível. À minha tia Elysete e prima Luma, pela morada e pelo acolhimento durante este período importante da minha vida. Ao meu companheiro, Anderson Santana, pelo apoio, paciência e companhia ao longo deste processo. Agradeço também às minhas colegas e professoras de TCC, pela parceria e colaboração na realização deste trabalho. Obrigada a todos por contribuírem para que esta etapa fosse concluída com sucesso.

Eu, Luma Abram Cecilia, agradeço primeiramente aos meus pais, Suzanne e Luis, que são a minha base e me proporcionaram todo o suporte necessário para esta conquista. Aos meus irmãos, que me incentivam e apoiam a cada dia, e aos meus avós, que, em saudosa memória, são o meu gás e a maior motivação para continuar. De coração, agradeço também a João e sua família pelo acolhimento, carinho e apoio fundamental nos dias difíceis. Por fim, o meu muito obrigada às minhas colegas de TCC e à nossa orientadora: foi incrível e enriquecedor construir este projeto ao lado de mulheres tão inspiradoras.

Eu, Maria Eduarda Pamponet Ramalho, gostaria de agradecer, acima de tudo, aos meus pais, Maisa e Carlos, que sempre priorizaram a minha educação e me incentivaram a crescer, me dedicar e buscar a minha melhor versão. Sou imensamente grata por todo amor, apoio e por me proporcionarem as ferramentas necessárias para chegar

até aqui. Agradeço também às mulheres que trabalharam ao meu lado: nossa orientadora Angela Peixoto, por sua dedicação e solicitude, fundamentais para a realização deste projeto, assim como às minhas colegas. Mulheres excepcionais e brilhantes, que fortalecem a certeza de que nós, mulheres, pertencemos à engenharia de software e à tecnologia, não apenas ocupando espaço, mas transformando e liderando.

”My comprehension can only be an infinitesimal fraction of all I want to understand.”
(Ada Lovelace)

Resumo

O crescimento exponencial das transmissões ao vivo no *YouTube* resultou em um volume massivo de interações, impondo desafios significativos para a moderação e a análise de engajamento em tempo real. Este trabalho apresenta o *Live Insights*, uma plataforma web de inteligência de dados projetada para monitorar, processar e extrair *insights* estratégicos de comentários via integração com *Large Language Models* (LLMs). O sistema automatiza a classificação de mensagens segundo critérios de reações e tipologia da interação, transformando fluxos de texto não estruturados em métricas visuais. O objetivo do presente trabalho é descrever a arquitetura de software e a metodologia empregada, validando a eficácia dos LLMs como suporte à gestão de comunidades. Os resultados obtidos demonstram a capacidade do sistema em processar contextos informais com baixa latência, gerando métricas precisas sobre a distribuição de reações, picos de engajamento e identificação de conteúdo ofensivo. Por fim, são apontadas as limitações da solução, especificamente quanto à dependência de *Application Programming Interfaces* (APIs) externas e aos custos operacionais associados à escalabilidade em cenários de alto volume de dados.

Palavras-Chave: *Large Language Models* (LLMs), Análise de Sentimento, Inteligência de Dados, YouTube, Monitoramento em Tempo Real

Abstract

The exponential growth of live broadcasts on YouTube has resulted in a massive volume of interactions, posing significant challenges for moderation and real-time engagement analysis. This study presents Live Insights, a web-based data intelligence platform designed to monitor, process, and extract strategic insights from comments via integration with *Large Language Models* (LLMs). The system automates message classification according to sentiment and interaction type criteria, transforming unstructured text flows into visual metrics. The objective of this work is to describe the software architecture and methodology employed, validating the efficacy of LLMs as support for community management. The obtained results demonstrate the system's capacity to process informal contexts with low latency, generating precise metrics regarding reaction distribution, engagement peaks, and identification of offensive content. Finally, the solution's limitations are pointed out, specifically regarding the reliance on external *Application Programming Interfaces* (APIs) and operational costs associated with scalability in high-data-volume scenarios.

Keywords: *Large Language Models* (LLMs), Sentiment Analysis, Data Intelligence, YouTube, Real-Time Monitoring

Lista de figuras

Lista de Siglas e Abreviaturas

API	*Application Programming Interface*
APIs	*Application Programming Interfaces*
BCrypt	*Blowfish Crypt*
BES	Bacharelado em Engenharia de Software
CORS	*Cross-Origin Resource Sharing*
CRM	*Customer Relationship Management*
CRUD	*Create, Read, Update, Delete*
CSRF	*Cross-Site Request Forgery*
CSS	*Cascading Style Sheets*
DDD	*Domain Driven Design*
DER	Diagrama de Entidade Relacionamento
DTO	*Data Transfer Object*
GDPR	*General Data Protection Regulation*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IA	Inteligência Artificial
JDBC	*Java Database Connectivity*
JPA	*Java Persistence API*
JSON	*JavaScript Object Notation*
JS	*JavaScript*
JWT	*JSON Web Tokens*
JVM	Máquina Virtual Java
LGPD	Lei Geral de Proteção de Dados
LLM	*Large Language Model*
LLMs	*Large Language Models*
NLP	*Natural Language Processing*
PLN	Processamento de Linguagem Natural
RA	*Research Answer*
REST	*Representational State Transfer*
RESTful	*Representational State Transfer (Architectural Style)*

RF	Requisito Funcional
RNF	Requisito Não Funcional
RQ	*Research Question*
SGBD	Sistema Gerenciador de Banco de Dados
SGBDOR	Sistema Gerenciador de Banco de Dados Objeto-Relacional
SOLID	*Single Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Dependency Inversion*
SPA	*Single Page Application*
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
WORA	*Write Once, Run Anywhere*
YAML	*YAML Ain't Markup Language*

Sumário

1 Introdução

As *lives* no *YouTube* geram um volume massivo de comentários o que exige ferramentas avançadas de análise. Nesse contexto, o *Live Insights* surge como uma plataforma *web* de inteligência de dados projetada para monitorar e extrair *insights* em tempo real. O sistema oferece aos criadores de conteúdo e moderadores uma interface que exibe os comentários e suas classificações, além de gerar um gráfico dinâmico que correlaciona o tipo de interação com a minutagem da transmissão.

A funcionalidade central do sistema reside na integração com Modelos de Linguagem de Grande Escala (*Large Language Models* (LLMs)). Após a captura dos dados via *Application Programming Interface* (API), cada comentário é classificado automaticamente segundo dois critérios: reação (negativo, neutro, positivo) e tipo de interação (pergunta, crítica, ofensivo etc.). Essa análise padronizada garante a extração de métricas detalhadas (HAN et al., 2024), transformando o fluxo de texto em dados estratégicos.

O desenvolvimento do *Live Insights* demonstra a aplicação prática dos LLMs na gestão de comunidades, posicionando-se como uma ferramenta agregadora e estratégica para criadores de conteúdo. O sistema fornece *dashboards* e classificações detalhadas sobre o engajamento e a audiência, impactando diretamente o posicionamento e a tomada de decisão dos *streamers* no *YouTube*. Dessa forma, o presente trabalho descreve a arquitetura de *software* e a metodologia empregada para a extração e visualização dessas métricas, validando a eficácia dos LLMs como suporte essencial para análises em tempo real e para o crescimento estratégico no ecossistema de conteúdo digital.

1.1 Aplicabilidade e Motivação

O objetivo central deste trabalho é o desenvolvimento do *Live Insights*, uma plataforma de inteligência de dados projetada para simplificar a apresentação e o consumo das informações geradas pelos comentários em *lives* do *YouTube*. O diferencial do sistema reside no foco na aplicabilidade em tempo real e na usabilidade para o gerenciamento de comunidades e estratégias de conteúdo. Tradicionalmente, a análise dessas interações exige a revisão manual de volumes massivos de texto ou o uso de ferramentas complexas de *Natural Language Processing* (NLP). Essa abordagem torna-se insustentável no ambiente de *live streaming*, em que o volume e a alta velocidade dos comentários em tempo real comprometem a eficácia da análise humana e dos métodos tradicionais (CHAN; THEIN, 2018).

O *Live Insights* demonstra, portanto, a aplicação prática de LLMs para transformar o fluxo bruto de texto em métricas estratégicas acessíveis.

A motivação primordial para a criação do *Live Insights* é democratizar o acesso a *insights* avançados, oferecendo suporte estratégico em tempo real. O sistema realiza a classificação automática de reações e tipos de interação, permitindo a extração de tendências de engajamento e a correlação de picos de interação com momentos específicos da transmissão. Além de otimizar as métricas para a tomada de decisão, a plataforma aprimora a moderação e o gerenciamento dessas interações.

Em resumo, o sistema eleva a qualidade da gestão de comunidades digitais, fornecendo uma ferramenta essencial para o crescimento e a profissionalização no ecossistema de conteúdo do *YouTube*.

1.2 Objetivo Geral

O objetivo geral deste trabalho é desenvolver e validar a eficácia do *Live Insights*, uma plataforma *web* de inteligência de dados. A plataforma emprega LLMs para a analisar reações e a extrair métricas em tempo real a partir de comentários de *lives* no *YouTube*. Com isso, busca-se oferecer suporte estratégico e visual à tomada de decisão e a moderação assistida no ecossistema de conteúdo digital.

1.3 Objetivos Específicos

- Desenvolver o sistema de classificação em tempo real: Projetar e implementar a arquitetura de *software* para a captura de dados e a integração de LLMs, permitindo a classificação automática e padronizada de reações e tipos de interações dos comentários de *lives* em tempo real.
- Criar a plataforma de visualização estratégica: Desenvolver uma plataforma *web* que exiba as métricas extraídas de forma intuitiva, incluindo um gráfico dinâmico para a correlação temporal do engajamento e o armazenamento dos dados brutos para auditoria.
- Validar a aplicação para suporte à decisão e moderação: Demonstrar a eficácia do *Live Insights* como ferramenta essencial para a tomada de decisão e moderação assistida, buscando promover um ambiente *online* mais responsável e transparente, sem comprometer a liberdade de expressão.

1.4 Estrutura do Trabalho

O presente trabalho de conclusão de curso está organizado em seis capítulos. Este primeiro capítulo apresentou a aplicabilidade, motivação, objetivos e contribuições associadas a esta pesquisa. O capítulo 2 aborda a fundamentação teórica, essencial para o embasamento e a compreensão do projeto. No capítulo 3, é detalhada a metodologia utilizada para o desenvolvimento da pesquisa. O capítulo 4 apresenta o contexto da pesquisa e o desenvolvimento da ferramenta *Live Insights*. Em seguida, o capítulo 5 traz as demonstrações e os resultados da aplicação. Por fim, o Capítulo 6 apresenta as conclusões e as sugestões de trabalhos futuros envolvendo a temática e o sistema desenvolvido.

2 Fundamentação Teórica

O presente capítulo tem como objetivo estabelecer a fundamentação teórica necessária para o desenvolvimento desta pesquisa. Para isso, são apresentadas as principais contribuições da literatura científica relacionadas à análise de reações, à extração de dados em tempo real e à aplicação de *Large Language Models* (LLMs). Além disso, são detalhadas as tecnologias e os conceitos que viabilizaram a construção do estudo.

2.1 Youtube e Live de transmissões ao vivo

Criado em 2005, o *YouTube*[1] consolidou-se como uma rede social em virtude das múltiplas possibilidades de interação, engajamento e estabelecimento de vínculos entre os usuários. Originalmente concebido como uma plataforma de compartilhamento de vídeos, o *YouTube* configura-se também como um recurso capaz de integrar, em uma única estrutura, som, cor, imagem e movimento (RODRIGUES; BRENNAND, 2022).

Conforme destacam Silva et al. (2023), ao longo dos anos, a plataforma ampliou suas funcionalidades ao incorporar transmissões ao vivo por meio de *live streamings*, possibilitando a interação em tempo real entre criadores de conteúdo e seu público. Essa dinâmica reforça o papel do *YouTube* como um ambiente comunicacional dinâmico, caracterizado pela troca constante de informações, experiências e percepções.

Durante essas transmissões, o público participa ativamente, expressando emoções e opiniões que variam entre reações positivas e negativas. Esse engajamento, marcado pelo grau de responsividade e sensibilidade da audiência, constitui um elemento central para a análise do comportamento dos espectadores.

A mensuração desse nível de engajamento mostra-se essencial para *streamers* e moderadores, pois permite identificar como os espectadores reagem ao conteúdo em tempo real e, a partir disso, ajustar estratégias voltadas à manutenção ou ao aumento da interação (LEITE, 2016).

Dessa forma, é possível concluir que o *YouTube* consolidou-se como um espaço comunicacional complexo, no qual a interação entre criadores e espectadores ultrapassa o simples ato de compartilhar vídeos, configurando-se como um processo contínuo de construção de sentidos.

A evolução da plataforma, especialmente com a introdução das transmissões ao vivo, ampliou as possibilidades de mediação e participação, reforçando seu papel como

[1] https://youtube.com.br

ambiente de sociabilidade e engajamento digital. Nesse contexto, o estudo do comportamento do público e da mensuração do engajamento torna-se essencial para compreender as dinâmicas comunicacionais contemporâneas, nas quais a interação em tempo real e a resposta emocional da audiência assumem papel central na consolidação das relações mediadas pela tecnologia.

2.2 Redes Sociais

As redes sociais constituem um dos fenômenos mais marcantes da sociedade contemporânea, estando profundamente integradas ao cotidiano e transformando a forma como indivíduos se relacionam, consomem informações e constroem identidades. Conforme a jornalista Raquel Recuero, essas plataformas digitais possibilitam não apenas a socialização e o entretenimento, mas também a circulação massiva de dados, opiniões e reações em tempo real (RECUERO, 2009).

De acordo com o estudo realizado por Xu, Chang e Jayne (2022), denominado de *A systematic review of social media-based sentiment analysis*, as redes sociais têm se consolidado como espaços privilegiados para compreender comportamentos humanos, já que concentram interações espontâneas de milhões de usuários ao redor do mundo. O artigo destaca que essas plataformas oferecem um campo fértil para a aplicação de técnicas de análise de reações, possibilitando investigar tendências, padrões de opinião e percepções coletivas em contextos diversos, como política, saúde pública, economia e mobilização social.

Além disso, o estudo ressalta que a crescente relevância das redes sociais está diretamente relacionada à sua capacidade de difundir conteúdos em larga escala e em alta velocidade. Essa característica, embora amplie o alcance da informação, também traz desafios significativos, como a propagação de desinformação, discursos de ódio e impactos sobre a saúde mental dos usuários.

Nesse contexto, a análise de reações em comentários de *lives* no *YouTube*, apoiada por *Large Language Models* (LLMs), mostra-se uma abordagem promissora para compreender em tempo real as percepções e emoções dos usuários. Essa perspectiva possibilita não apenas monitorar tendências e reações imediatas, mas também extrair métricas que auxiliem na interpretação crítica da dinâmica das interações digitais e na proposição de soluções mais assertivas para os desafios que emergem nesse ambiente.

2.3 Saúde Mental

A saúde mental, no contexto da sociedade digital, é um tema de crescente urgência. Redes sociais como o *YouTube*, atuam não apenas como plataformas de entre-

tenimento e informação, mas também como espaços de intensa interação que podem exercer impactos psicológicos relevantes sobre os indivíduos. Esse fenômeno se insere no processo de midiatização, onde a sociedade se torna progressivamente dependente da mídia e de suas lógicas, influenciando as interações humanas e a cultura (HJARVARD, 2014).

O trabalho ”Redes Sociais Virtuais e Saúde Mental: Os impactos da exposição e dos ataques de ódio”, de Ester Aline Silvério Moreira (MOREIRA, 2022), oferece uma perspectiva valiosa ao investigar como a exposição a discursos e ataques de ódio online afeta a saúde mental, a autoimagem e a autoestima. A pesquisa destaca que a toxicidade dos comentários e a exposição a julgamentos negativos podem gerar sentimentos como medo, tristeza e impotência e diferentes reações.

Dessa forma, a análise de reações em comentários de *lives* no *YouTube* torna-se uma ferramenta crucial para compreender, em tempo real, a atmosfera emocional de um ambiente digital, contribuindo para a identificação de padrões de comportamento e interações que podem influenciar os usuários.

2.4 Inteligência Artificial

A Inteligência Artificial (IA) é a ciência e a engenharia de criar sistemas que podem simular a inteligência humana. De acordo com Kaplan e Haenlein, a IA é a capacidade de um sistema de ”interpretar corretamente dados externos, aprender com esses dados e usar esse aprendizado para alcançar objetivos e tarefas específicas por meio de uma adaptação flexível”(KAPLAN; HAENLEIN, 2019).

Dentro desse escopo, a análise de reações se insere como uma aplicação de IA Analítica, uma das três categorias de sistemas de IA propostas pelos autores. Esse tipo de abordagem utiliza técnicas de aprendizado de máquina (*machine learning*) para analisar dados históricos, identificar padrões e gerar previsões. Assim, mostra-se particularmente adequada para a interpretação de grandes volumes de comentários produzidos em plataformas como o *YouTube*, permitindo a detecção de tendências, reações do público e comportamentos recorrentes (CHOWDHARY, 2019).

A incorporação de *Large Language Models* (LLMs) amplia ainda mais essas possibilidades, uma vez que tais modelos apresentam elevada capacidade de compreender, interpretar e classificar textos de maneira automatizada e eficiente. Dessa forma, seu uso alinha-se ao potencial da IA de processar grandes quantidades de informação, oferecendo análises mais precisas e contribuindo para a construção de sistemas capazes de apoiar decisões com base em dados complexos e dinâmicos.

Segundo Victor Sobreira (2025), essa transformação reflete não apenas mudanças técnicas, mas também uma ampliação do escopo de aplicação da IA, que hoje abrange áreas como a pesquisa histórica, possibilitando a análise automatizada de

grandes conjuntos de documentos, a identificação de padrões temporais e o suporte à interpretação de contextos históricos complexos (SOBREIRA, 2025).

Contudo, a evolução da Inteligência Artificial (IA), impulsionada por avanços em aprendizado de máquina e modelos de linguagem, tem transformado a forma como dados são processados e interpretados em diversos contextos.

A integração de técnicas analíticas e generativas permite não apenas a automatização de tarefas, mas também a ampliação das capacidades cognitivas dos sistemas, possibilitando análises mais profundas e decisões mais informadas. Contudo, o uso dessas tecnologias exige reflexão crítica sobre seus limites, desafios éticos e impactos sociais, destacando a importância de abordagens interdisciplinares para o desenvolvimento e aplicação responsável da IA (DUQUE-PEREIRA; MOURA et al., 2023).

2.5 *Large Language Models* (LLMs)

Segundo Han et al. (2024), em seu estudo "*A Review of Large Language Models*: Fundamental Architectures...", os *Large Language Models* (LLMs) são sistemas de inteligência artificial treinados com extensos volumes de dados textuais e capazes de compreender, gerar e interagir com a linguagem humana. Esses modelos representam um avanço significativo no campo do Processamento de Linguagem Natural (PLN), pois permitem análises mais contextuais e interpretações semânticas mais precisas (MINAEE et al., 2024).

Tais modelos tornaram-se ferramentas centrais para diversas tarefas de PLN, incluindo a análise de reações. A arquitetura que constitui a base desses sistemas, possibilita uma compreensão aprofundada da linguagem informal e dinâmica presente em ambientes digitais, aspecto essencial para interpretar a expressividade e as particularidades do discurso humano (MINAEE et al., 2024).

A complexidade e o grande volume de dados encontrados em plataformas online são tratados de maneira eficaz pelos LLMs. Sua capacidade de processar vastas quantidades de texto e identificar padrões complexos torna esses modelos especialmente valiosos para a extração de métricas e informações em diferentes domínios de pesquisa (HAN et al., 2024).

Assim, a adaptabilidade e a robustez dos LLMs os consolidam como uma das tecnologias mais promissoras para o processamento de dados em tempo real. Sua habilidade de captar nuances da linguagem espontânea e dinâmica como a observada em comentários de *lives* é fundamental para o desenvolvimento deste trabalho, possibilitando a extração de métricas de reação com precisão e velocidade adequadas para uma análise eficiente (MINAEE et al., 2024).

2.5.1 Processamento de Linguagem Natural (PLN) e relação com LLMs

O Processamento de Linguagem Natural (PLN) é uma subárea da Inteligência Artificial dedicada ao desenvolvimento de métodos computacionais capazes de analisar, interpretar e gerar linguagem humana. Na prática, o PLN engloba tarefas como classificação de textos, análise de sentimentos, identificação de intenção (*intent classification*), sumarização e extração de informações, dentre outras (JURAFSKY; MARTIN, 2009).

Segundo Han et al. (2024), em seu estudo "*A Review of Large Language Models*: *Fundamental Architectures*...", os *Large Language Models* (LLMs) são sistemas de inteligência artificial treinados com extensos volumes de dados textuais e capazes de compreender, gerar e interagir com a linguagem humana. Esses modelos representam um avanço significativo no campo do PLN, pois permitem análises mais contextuais e interpretações semânticas mais precisas (MINAEE et al., 2024).

Tais modelos tornaram-se ferramentas centrais para diversas tarefas de PLN, incluindo a análise de reações e a identificação do propósito comunicativo do usuário. Diferentemente de abordagens clássicas baseadas exclusivamente em palavras-chave, os LLMs tendem a realizar a interpretação de forma contextual, considerando o sentido da frase como um todo (incluindo negação, ironia, gírias e dependências entre termos), o que é especialmente relevante em comentários informais de ambientes digitais (MINAEE et al., 2024).

2.5.1.1 Análise de Sentimentos: palavras isoladas vs. interpretação contextual (LLMs)

A Análise de Sentimentos busca identificar a polaridade afetiva expressa em um texto (por exemplo: negativo, neutro, positivo). Em abordagens clássicas, essa tarefa pode ser feita por meio de dicionários léxicos (listas de palavras com pontuações) ou por modelos supervisionados tradicionais. Apesar de úteis, métodos léxicos tendem a falhar quando o significado depende de contexto, como em casos de negação ("não foi ruim"), ironia, gírias, memes e referências ao que está acontecendo na live.

No Live Insights, a análise é realizada por meio de uma LLM, que tende a produzir resultados mais adequados justamente por executar a interpretação de forma contextual, considerando a frase como um todo e o uso real da linguagem em ambientes digitais. Assim, a classificação de reação não se limita à presença de termos isolados, mas busca refletir a intenção comunicativa e o sentido global do comentário.

2.5.2 Redes *Transformer* e mecanismo de atenção (*Attention*)

A arquitetura que constitui a base de grande parte dos LLMs modernos é a rede *Transformer*, proposta para melhorar a modelagem de linguagem ao permitir processamento eficiente e captura de dependências de longo alcance no texto (VASWANI et al., 2017). O principal componente do *Transformer* é o mecanismo de atenção (*attention*), que permite ao modelo atribuir diferentes pesos a palavras e trechos do texto ao construir uma representação interna, destacando os elementos mais relevantes para o significado.

Em particular, a *self-attention* (autoatenção) permite que o modelo relacione termos entre si dentro do próprio comentário, favorecendo a compreensão de estruturas linguísticas complexas e do contexto semântico geral (VASWANI et al., 2017). Dessa forma, os LLMs conseguem interpretar a linguagem espontânea e dinâmica presente em chats de *lives*, aspecto essencial para tarefas como classificação textual e extração de métricas (MINAEE et al., 2024).

A complexidade e o grande volume de dados encontrados em plataformas *online* são tratados de maneira eficaz pelos LLMs. Sua capacidade de processar vastas quantidades de texto e identificar padrões complexos torna esses modelos especialmente valiosos para a extração de métricas e informações em diferentes domínios de pesquisa (HAN et al., 2024).

Assim, a adaptabilidade e a robustez dos LLMs os consolidam como uma das tecnologias mais promissoras para o processamento de dados em tempo real. Sua habilidade de captar nuances da linguagem espontânea e dinâmica como a observada em comentários de *lives* é fundamental para o desenvolvimento deste trabalho, possibilitando a extração de métricas de reação com precisão e velocidade adequadas para uma análise eficiente (MINAEE et al., 2024).

2.6 Java

Criada em 1995 pela equipe de James Gosling na *Sun Microsystems*, o Java é uma linguagem de programação versátil e robusta,sendo amplamente adotada em diferentes contextos computacionais.

Segundo (MARTINEZ; REMEGIO; LINCOPINIS, 2023), o Java foi projetado sob a filosofia "*Write Once, Run Anywhere*" (WORA), que permite que o mesmo código seja executado em diversos sistemas operacionais sem necessidade de reescrita, garantindo portabilidade. Sua arquitetura orientada a objetos, sintaxe similar à do C++ e tipagem forte a tornam uma ferramenta confiável e de alto desempenho, ideal para aplicações empresariais, sistemas distribuídos e, mais recentemente, para a análise de grandes volumes de dados.

A vasta comunidade e um rico ecossistema de *frameworks*, como *Spring* e *Hibernate*, consolidam Java como uma escolha poderosa para o desenvolvimento de soluções corporativas e *web*. Essa maturidade e eficiência do ecossistema tornam-no particularmente relevante para a integração com modelos de *Machine Learning* e LLMs . A capacidade da linguagem de processar grandes volumes de dados de maneira escalável é essencial para projetos que buscam extrair métricas de emoções em tempo real, como em comentários de *lives* no *YouTube*.

Além disso, a Máquina Virtual Java (JVM), que executa o código compilado em *bytecode*, garante a compatibilidade e a alta taxa de transferência (*throughput*) necessária para o processamento contínuo de dados, sendo sua otimização crítica para ambientes que exigem desempenho robusto e alta escalabilidade (MANCHANA, 2015).

De acordo com o índice TIOBE (TIOBE, 2025), o Java mantém-se entre as linguagens de programação mais populares do mundo, ocupando frequentemente posições de destaque entre as cinco primeiras. Essa expressiva popularidade reflete sua combinação de robustez, escalabilidade e portabilidade, características que consolidam sua presença em aplicações corporativas e acadêmicas.

No contexto da análise de comentários em *lives*, a eficiência do Java no processamento de grandes volumes de dados em alta velocidade demonstra alinhamento com as demandas de extração e interpretação de emoções em tempo real, possibilitando a geração de métricas relevantes para a compreensão da dinâmica das interações digitais.

2.7 PostgreSQL

O *PostgreSQL* é um Sistema Gerenciador de Banco de Dados Objeto-Relacional (SGBDOR) de código aberto, projetado para oferecer uma arquitetura robusta, segura e extensível. Sua implementação utiliza a linguagem *Structured Query Language* (SQL) como principal interface para a manipulação e consulta de dados. Desenvolvido a partir do projeto *POSTGRES*, da Universidade da Califórnia em *Berkeley*, sua origem remonta ao ano de 1986 (POSTGRE, 2025).

O *PostgreSQL* se destaca não apenas por sua robustez e adesão a padrões, mas também por sua flexibilidade e extensibilidade, que o tornam uma plataforma ideal para a gestão de dados de alta complexidade. A sua capacidade de processar dados geoespaciais e de documentos *JavaScript Object Notation* (JSON), por exemplo, o posiciona como uma ferramenta versátil.

Essa mesma versatilidade é crucial em cenários de pesquisa, onde a arquitetura do Sistema Gerenciador de Banco de Dados (SGBD) deve ser capaz de gerenciar e persistir fluxos de informações não convencionais, como dados estruturados e semiestruturados extraídos em tempo real de plataformas online. Ele garante a integri-

dade e a consistência das métricas obtidas, elementos fundamentais para qualquer estudo rigoroso e para a operação confiável de sistemas, mostrando-se competitivo mesmo em ambientes de alta demanda (TRUSKOWSKI; KLEWEK; SKUBLEWSKA-PASZKOWSKA, 2020).

2.8 React

Em 2013, o *React.js*, uma biblioteca *JavaScript* desenvolvida por engenheiros do *Facebook (Meta)*, foi lançado para revolucionar o desenvolvimento de interfaces de usuário.

Conforme destacado em obras como *Introduction to React, de Cory Gackenheimer* (GACKENHEIMER, 2015), a principal vantagem da ferramenta é a sua arquitetura baseada em componentes. Essa arquitetura, que enfatiza a modularidade e a reusabilidade dos elementos (RAJALA, 2024), permite que a interface seja decomposta em módulos pequenos e reutilizáveis, tornando cada parte do código isolada e fácil de gerenciar.

Esse modelo de desenvolvimento não apenas simplifica o processo de criação, mas também otimiza a manutenção de interfaces complexas, o que é crucial para projetos em larga escala. Além disso, a reatividade do *React.js* garante que a interface responda de forma eficiente às mudanças nos dados, atualizando apenas os componentes necessários e proporcionando uma experiência de usuário fluida e de alto desempenho.

2.9 Trabalhos relacionados

Este capítulo revisa os principais estudos e sistemas focados na análise de comentários em plataformas online, com ênfase nas metodologias e soluções propostas pela literatura. Serão abordadas pesquisas que investigam a extração de métricas de reação de textos gerados por usuários, explorando tanto abordagens tradicionais quanto o uso de LLMs. A partir da análise desses estudos, surgiram as motivações para a presente pesquisa.

Na pesquisa *Sentiment Analysis on YouTube: A Brief Survey, Asghar et al*. (2015) conduziram uma revisão das principais técnicas para a análise de comentários do *YouTube*. O estudo ressalta os desafios da linguagem informal, que contém gírias e abreviações, e discute as principais abordagens para essa tarefa: aquelas baseadas em dicionários léxicos, que usam listas de palavras com pontuação de reação e as baseadas em aprendizado de máquina, que treinam modelos a partir de dados rotulados. O estudo aponta que as abordagens de aprendizado de máquina são mais promissoras para superar as limitações da linguagem informal do *YouTube* (ASGHAR et al., 2015).

A relevância da análise em tempo real e a capacidade dos LLMs de processar dados sensíveis são amplamente reconhecidas, estendendo-se até o campo da saúde pública. O estudo *'Sentiment Analysis Using a Large Language Model–Based Approach to Detect Opioids Mixed With Other Substances Via Social Media'* (AHMAD; BATYRSHIN; SIDOROV, 2025) é um exemplo crucial. O objetivo central dessa pesquisa é utilizar a análise de interações em mídias sociais para obter informações sobre padrões de uso de opioides, aproveitando o *Large Language Model* (LLM) (GPT-3.5 Turbo) para processar informações autodeclaradas em comentários do *YouTube*. O estudo analisou dados anotados manualmente em categorias multiclasse de risco e bem-estar, demonstrando que o uso dessa tecnologia aliado à análise de texto pode se transformar em um instrumento de vigilância e intervenção em saúde de forma menos invasiva.

A análise de reações também é relevante em domínios técnicos com linguagem específica, o que demanda adaptação metodológica. A dissertação de Doutorado de Gláucya Boechat (BOECHAT et al., 2024), defendida na Universidade Federal da Bahia, investiga a análise de reações em *issues* no *GitHub*. O trabalho de Boechat enfatiza a importância de considerar aspectos subjetivos nas interações, mesmo em ambientes técnicos, e a necessidade de adaptação lexical para a análise, utilizando ferramentas como o *SentiStrength-SE* ajustado à Engenharia de *Software*. Essa pesquisa reforça o desafio da ambiguidade do texto gerado por usuário e a relevância de associar reação a métricas de comportamento (como a reabertura de *issues*), um ponto análogo ao objetivo desta pesquisa de extrair métricas em tempo real.

As principais características metodológicas e desafios da literatura demonstram a necessidade de superar a linguagem informal e ambígua do texto gerado por usuários em plataformas *online*, ao mesmo tempo que validam a eficácia da classificação multiclasse por meio de LLMs e a importância de associar reação a métricas de comportamento estratégico.

O sistema *Live Insights* se posiciona, assim, como um projeto que integra e aplica esses avanços diretamente no domínio específico dos comentários de *lives* do *YouTube*. A solução apresentada amplia o escopo da análise ao classificar e visualizar diversos tipos de interações (reação e tipo) em tempo real. Isso a estabelece como uma ferramenta de inteligência de dados dedicada ao gerenciamento de comunidades e ao suporte à tomada de decisão estratégica dos *streamers*, elevando a qualidade e a profissionalização do ecossistema de conteúdo digital.

3 Metodologia

Este capítulo apresenta a metodologia adotada para o desenvolvimento do presente trabalho e a construção do sistema proposto. A pesquisa foi conduzida com base em uma abordagem aplicada, voltada à criação de uma solução tecnológica capaz de realizar a análise de reações em comentários de transmissões ao vivo na plataforma *YouTube*, em tempo real.

A partir dessa proposta, foram definidas as etapas de levantamento de requisitos, modelagem, implementação e validação do sistema, com foco na integração entre técnicas de PLN e o uso de LLMs. O sistema foi desenvolvido para coletar, processar e interpretar automaticamente os comentários das *lives*, extraindo métricas que possibilitam compreender a percepção do público e gerar *insights* relevantes sobre o comportamento da audiência.

3.1 Questões de Pesquisa

A partir da análise crítica dos estudos apresentados na literatura, observa-se um conjunto de contribuições que fundamentam a proposta deste trabalho e evidenciam a relevância da análise de reações em ambientes digitais. As pesquisas revisadas demonstram diferentes abordagens, metodologias e desafios, revelando lacunas importantes relacionadas à linguagem informal, à necessidade de processamento em tempo real e à interpretação contextual das interações em plataformas como o *YouTube*.

Com base nesse panorama, este trabalho busca avançar na discussão ao integrar técnicas contemporâneas, como o uso de LLMs, para a extração e visualização de métricas estratégicas durante transmissões ao vivo. As reflexões derivadas da revisão teórica forneceram suporte direto para a formulação das questões norteadoras desta pesquisa, que serão retomadas e respondidas na discussão final, apresentada na Seção 6. Nesse sentido, os trabalhos relacionados, apresentados na Seção 2, não apenas contextualizam a temática, mas também orientam as escolhas metodológicas e justificam a necessidade de uma solução como o *Live Insights*, contribuindo de forma decisiva para a consolidação dos objetivos deste estudo.

Research Question (RQ)1 Como os LLMs podem ser aplicados para realizar a análise de reações em comentários de transmissões ao vivo no *YouTube*, permitindo a extração de métricas em tempo real?

RQ2 De que forma a integração entre LLMs e sistemas de monitoramento em tempo real pode contribuir para a geração de métricas precisas sobre o engajamento e

a percepção do público?

RQ3 Como a visualização das métricas de reação pode apoiar criadores de conteúdo e gestores na tomada de decisões baseada em dados?

3.2 Processo de Pesquisa

O processo de pesquisa adotado neste trabalho foi estruturado de forma sistemática, visando garantir rigor metodológico e coerência entre os objetivos, os procedimentos técnicos e a construção dos resultados. A trajetória investigativa foi organizada em etapas sequenciais que envolveram revisão da literatura, definição das perguntas de pesquisa, escolha das abordagens metodológicas, desenvolvimento do sistema proposto e avaliação de sua eficácia no contexto das transmissões ao vivo do *YouTube*.

Do ponto de vista técnico, foram aplicadas metodologias de Engenharia de *Software* para a construção do sistema *Live Insights*, contemplando especificação de requisitos, modelagem da arquitetura, desenvolvimento e validação. Para a última etapa do processo, foram conduzidos testes com transmissões de *lives*, possibilitando verificar o desempenho da ferramenta na captura dos dados, na classificação automática e na geração dos gráficos em tempo real. Esse procedimento permitiu observar o comportamento do sistema em condições próximas ao uso real, fornecendo evidências de confiabilidade e eficiência.

Ao final dessas etapas, definiu-se a estratégia de classificação adotada no sistema, incluindo a modelagem do problema como uma classificação multiclasse em dois eixos (reação e tipo de interação). Essa definição foi necessária para garantir consistência na extração das métricas em tempo real e para suportar a geração de *insights* durante as transmissões ao vivo.

3.3 Classificação multiclasse e categorias adotadas no *Live Insights*

Neste trabalho, a classificação automática dos comentários é tratada como uma classificação multiclasse, pois cada comentário é associado a uma classe dentre várias classes possíveis. No Live Insights, a classificação ocorre em duas dimensões complementares, cada uma sendo um problema multiclasse independente:

(1) Reação (polaridade / análise de sentimentos):

Negativo

Neutro

Positivo

(2) Tipo de interação (classificação de intenção e natureza do comentário):

Pergunta
Elogio
Crítica
Sugestão
Meme/Piada
Reclamação
Reação emocional
Ofensivo/Preconceituoso

Essa separação em dois eixos permite diferenciar, por exemplo, um comentário negativo que é uma crítica construtiva de um comentário negativo que é ofensivo, o que é relevante tanto para métricas de engajamento quanto para moderação.

3.3.1 Combinação de técnicas clássicas de PLN com LLMs na classificação

A abordagem adotada pode ser entendida como híbrida, combinando elementos clássicos do Processamento de Linguagem Natural (PLN) com o uso de *Large Language Models* (LLMs). Em um fluxo típico, técnicas tradicionais ajudam a organizar e estruturar o dado textual antes e depois da inferência, enquanto a *Large Language Model* (LLM) executa a etapa principal de interpretação semântica contextual. Exemplos de contribuições clássicas que se integram ao uso de LLMs incluem: padronização/-normalização textual, tratamento de caracteres e ruídos comuns em *chats* (como repetições, excesso de pontuação, *emojis* e abreviações), organização por lotes (*batching*) para eficiência operacional e validação do formato de saída para garantir consistência dos resultados armazenados.

Com isso, as técnicas tradicionais atuam como suporte de qualidade e eficiência do *pipeline*, e a LLM é utilizada para realizar a classificação multiclasse com base no sentido completo da mensagem, indo além de heurísticas simples.

3.3.2 Classificação de intenção e interação (*Intent Classification*)

A classificação de intenção (*Intent Classification*) é uma tarefa de PLN que consiste em identificar o propósito do usuário ao emitir uma mensagem, isto é, “o que ele está tentando fazer” comunicativamente. No contexto de transmissões ao vivo, isso é particularmente relevante porque comentários podem ter naturezas muito diferentes (perguntar algo, elogiar, criticar, reclamar, fazer piada, expressar emoção ou atacar alguém).

No *Live Insights*, o eixo “tipo de interação” corresponde diretamente a uma forma de *Intent Classification*, pois categoriza o comentário segundo sua finalidade comunicativa (Pergunta, Elogio, Crítica, Sugestão etc.). Essa dimensão complementa a

polaridade (reação) e melhora a qualidade dos *insights*, permitindo análises como: volume de perguntas em momentos específicos, pico de reclamações, crescimento de memes/piadas, ou detecção direcionada de conteúdo ofensivo.

4 Desenvolvimento

O presente capítulo aborda as etapas de desenvolvimento e implementação do sistema de coleta e análise de dados em tempo real em transmissões ao vivo do *YouTube*, utilizando LLMs, nomeado como *Live Insights*.

Serão apresentados desde o processo do planejamento inicial até a integração dos módulos de captura, tratamento e interpretação das informações obtidas durante as *lives*. Portanto, os tópicos a seguir abordarão a definição dos requisitos funcionais e não funcionais, a arquitetura do sistema, as etapas de implementação e a demonstração do funcionamento do *software* de acordo da solução apontada.

4.1 Introdução

Diante do objetivo de compreender o comportamento dos espectadores em *lives*, surgiu a ideia do sistema de monitoramento em tempo real de transmissões ao vivo no *YouTube*, com foco em performance e análise de comentários. Essa proposta nasce da necessidade de transformar o grande volume de mensagens trocadas em tempo real em informações estruturadas e significativas, capazes de compreender o comportamento da audiência.

Partindo dessa necessidade, o sistema foi desenvolvido com o propósito de captar periodicamente os comentários publicados durante uma *live* diretamente da API do *YouTube* enquanto a transmissão está ativa e encaminhá-los a um LLM, que realiza a avaliação e classificação automática de cada mensagem.

Essa classificação ocorre segundo dois eixos principais: reação (negativo, neutro ou positivo) e tipo de interação (como perguntas, elogios, críticas, sugestões, reações emocionais ou mensagens ofensivas). Com base nessa análise, o sistema permite a filtragem de comentários sensíveis, facilitando a identificação de conteúdos negativos ou ofensivos/preconceituosos.

Ao término da transmissão ao vivo, as informações processadas são reunidas em um relatório analítico que permite avaliar a receptividade geral da audiência, além de exibir a *Uniform Resource Locator* (URL) do perfil do autor do comentário, fornecendo aos responsáveis pela *live* ferramentas para agir de forma rápida e eficiente; pois a partir dessas informações, é possível realizar denúncias na própria plataforma do *YouTube* ou, quando necessário, adotar medidas legais.

Dessa forma, o objetivo principal do sistema é disponibilizar uma ferramenta capaz de interpretar e categorizar automaticamente os comentários de transmissões ao vivo, promovendo uma análise precisa e em tempo real, fornecendo uma interface amigá-

vel e poderosa para produtores de conteúdo, moderadores e equipes de *marketing* analisarem o impacto de suas *lives*, identificarem padrões de engajamento e tomarem decisões rápidas frente a interações negativas ou ofensivas de maneira eficiente.

4.2 Quadro Kanban

A fim de aprimorar o gerenciamento deste projeto, optou-se pela implementação de um quadro *Kanban* como mecanismo de monitoramento e organização das atividades e seus respectivos *status*.

O *Kanban* é uma metodologia de origem japonesa que tem como princípio a visualização do fluxo de trabalho e a limitação de tarefas em andamento, permitindo identificar gargalos e otimizar processos de forma contínua. Segundo (ANDERSON, 2010), o método *Kanban* baseia-se em uma abordagem evolutiva de gestão, que busca aprimorar gradualmente a eficiência e a previsibilidade das entregas, por meio da transparência e da melhoria contínua dos fluxos de trabalho.

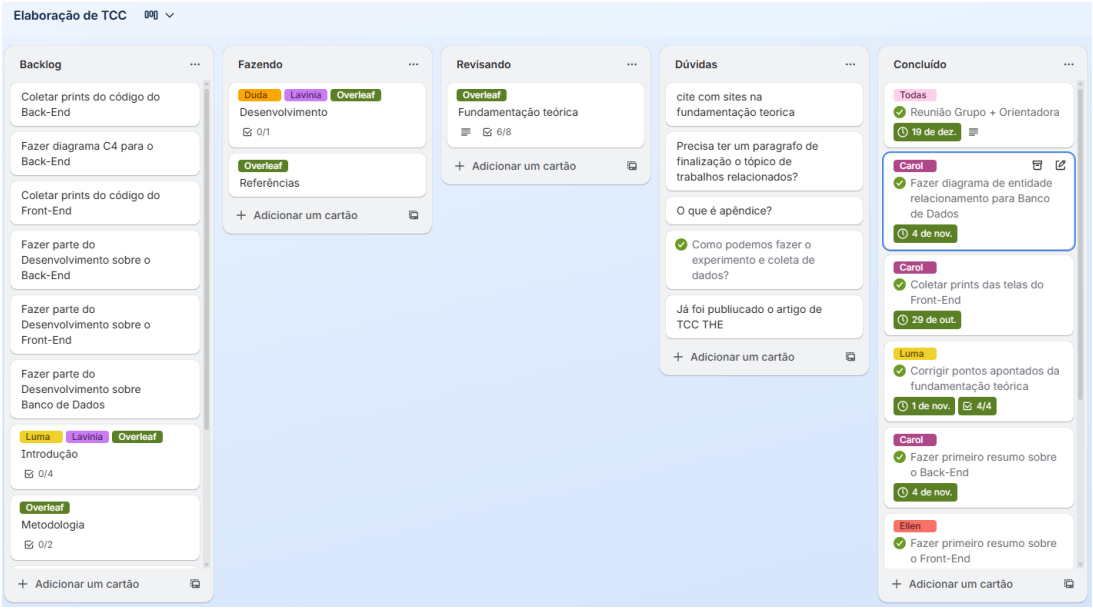
Para a aplicação dessa metodologia, utilizou-se a plataforma *Trello*[1], que permite a criação de quadros digitais compostos por colunas que representam as etapas do processo: *Backlog*, Fazendo, Revisando e Concluído. Essa estrutura facilita a visualização do fluxo de trabalho, possibilita o acompanhamento em tempo real das atividades e promove uma distribuição equilibrada das responsabilidades entre os membros da equipe.

De acordo com (SILVA; ANASTáCIO, 2018), a aplicação do *Kanban* auxilia no controle eficaz de processos e contribui para o aumento da produtividade e da qualidade dos resultados, características que se alinham diretamente às necessidades deste projeto.

A Figura 1 apresenta o quadro utilizado para desenvolvimento do presente trabalho.

[1] https://trello.com/

Figura 1 – Quadro *Kanban*



Autoria própria

Dessa forma, a utilização do *Trello* como ferramenta de apoio ao método *Kanban* mostrou-se fundamental para assegurar a organização, a eficiência e a transparência ao longo do processo de desenvolvimento do sistema.

4.3 Funcionalidades

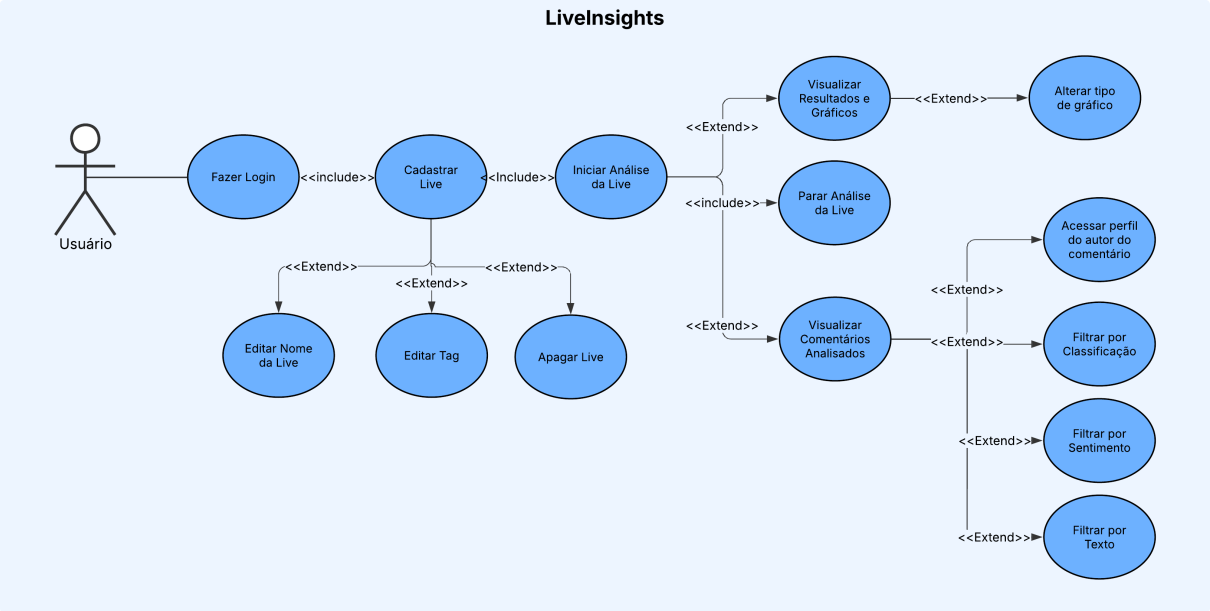
O projeto foi desenvolvido com uma abordagem voltada para a criação de um sistema inteligente e eficiente, capaz de realizar análises complexas de forma automatizada. O foco principal concentrou-se em assegurar a consistência e a confiabilidade dos resultados obtidos, aplicando princípios de engenharia de *software* orientados à qualidade e à precisão das operações.

Em vez de expandir o escopo com funcionalidades secundárias, optou-se por priorizar o desempenho e a robustez das funções essenciais. Essa decisão permitiu maior controle sobre os aspectos técnicos e favoreceu a otimização das etapas de coleta, classificação e geração de relatórios.

Como resultado, cada componente do sistema foi projetado para desempenhar seu papel com eficiência, sem comprometer a estabilidade, a escalabilidade ou a usabilidade da aplicação.

A Figura 2 representa um conjunto de ações realizadas pelo sistema *Live Insights*.

Figura 2 – Diagrama de *Casos de Uso*



Autoria própria

A seguir, são apresentados os requisitos funcionais e não funcionais definidos para o sistema.

Requisitos Funcionais

- Requisito Funcional (RF) 01: O sistema deve conectar-se periodicamente à API do *YouTube* para coletar os comentários publicados durante as transmissões ao vivo.
- RF02: O sistema deve iniciar automaticamente o monitoramento de comentários e estatísticas assim que a *live* estiver ativa, permitindo também pausar e reiniciar a análise.
- RF03: Cada comentário coletado deve ser encaminhado a um LLM para análise semântica e classificação, utilizando a *YouTube Data API v3*.
- RF04: Os dados coletados e analisados devem ser armazenados de forma persistente no banco de dados *PostgreSQL*, por meio dos repositórios *User, Live, Tag* e *Comments*.
- RF05: Armazenamento do texto original, reação, tipo, autor, URL do perfil e *timestamp*
- RF06: Após o encerramento da *live*, o sistema deve gerar um relatório consolidado contendo métricas de engajamento, distribuição de reações e exemplos de interações.

- RF07: Fornecimento da URL do perfil do autor, facilitando denúncias e medidas legais.
- RF08: Visualização de filtros por reação, tipo e busca textual.
- RF09:O sistema deve possuir um serviço de *Check Activity*, responsável por identificar automaticamente o início e o término de transmissões.
- RF10: O cadastro de transmissões ao vivo do *YouTube* será feito partir do identificador (ID) da *live*.

Requisitos Não Funcionais

- Requisito Não Funcional (RNF) 01: Processar até 30 comentários por requisição à LLM
- RNF02: Classificação com latência inferior a 5 segundos por lote.
- RNF03: Suportar múltiplas *lives* simultâneas, com escalabilidade horizontal.
- RNF04: Chaves de API armazenadas em arquivo *.env*, autenticação obrigatória, política de *Cross-Origin Resource Sharing* (CORS) e proteção de dados sensíveis.
- RNF05: Arquitetura desacoplada que permita uso de modelos como *LLaMA*, *Gemini* e *OpenAI*.
- RNF06: O *frontend* em *React.js* deve ser acessível e responsivo.
- RNF07: A API *Representational State Transfer* (REST) deve estar documentada em *Swagger UI* acessível em tempo de execução.

4.4 *Frontend*

Para o desenvolvimento do *frontend* do *Live Insights* foi utilizado o *framework React.js*. A escolha dessa tecnologia se deve à sua arquitetura baseada em componentes, que permite a construção de interfaces dinâmicas, modulares e de fácil manutenção.

4.4.1 Componentização

A arquitetura do *frontend* adota o princípio de componentização, fundamental no ecossistema *React*. A interface foi decomposta em unidades lógicas independentes e reutilizáveis, permitindo o isolamento de responsabilidades. Isso significa que elementos visuais recorrentes, como botões, formulários de entrada e cartões de exibição de

dados, são desenvolvidos uma única vez e instanciados em diferentes partes da aplicação. Essa abordagem não apenas acelera o desenvolvimento, mas também facilita a manutenção e a escalabilidade do código, garantindo consistência visual e comportamental em todo o sistema *Live Insights*.

4.4.2 Gerenciamento de Autenticação

O controle de acesso e a persistência da sessão do usuário são gerenciados através da *Context API* do *React.js*, especificamente por meio do componente *AuthContext.js*. Este contexto atua como um provedor de estado global, envolvendo a aplicação (via `AuthProvider` no `App.js`) e disponibilizando as informações do usuário autenticado para qualquer componente na árvore de renderização.

Os componentes `Login.js` e `Register.js` consomem este contexto para interagir com a API. Ao receber um *token JSON Web Tokens (JWT)* válido do *backend*, o contexto o armazena (localmente ou em *cookies* seguros) e atualiza o estado da aplicação, liberando o acesso às rotas protegidas sem a necessidade de múltiplos pedidos de *login*.

4.4.3 Rotas da Aplicação

A navegação no sistema foi implementada seguindo o conceito de *Single Page Application* (SPA), utilizando a biblioteca *React Router*. No componente `App.js`, as rotas são definidas declarativamente, separando as áreas públicas das privadas. O roteamento no lado do cliente permite que a transição entre as telas, como do *dashboard* para os detalhes de uma *live*, ocorra instantaneamente, sem recarregar a página inteira, proporcionando uma experiência de usuário fluida e semelhante a aplicativos nativos.

4.4.3.1 *Dashboard de Lives*

O componente `LiveDashboard.jsx` atua como a tela principal para o usuário autenticado. Ele é responsável por listar todas as transmissões cadastradas e monitoradas, apresentando um resumo visual do estado atual de cada *live* (ativa ou encerrada). Nesta interface, o usuário tem acesso rápido a funcionalidades de gerenciamento, como o cadastro de novas URLs de transmissão para monitoramento e a visualização prévia de métricas consolidadas.

4.4.3.2 Detalhes da *Live*

O componente `LiveDetails.jsx` representa o núcleo analítico da aplicação. Ao selecionar uma transmissão, o usuário é direcionado para esta interface, que consome

os dados processados pelo LLM. A tela é projetada para exibir a análise de reações e a categorização dos comentários. É aqui que os dados brutos são transformados em *insights*, permitindo ao moderador filtrar mensagens por categorias (ex.: ofensivas, perguntas) e visualizar o perfil dos autores, cumprindo o objetivo de fornecer ferramentas para tomada de decisão rápida.

4.4.3.3 *Header* Compartilhado

O componente `HeaderTop.jsx` atua como um elemento estrutural persistente no topo da aplicação, garantindo a identidade visual e o acesso às funções de sistema. Diferente de um cabeçalho estático, este componente implementa uma lógica de renderização condicional baseada na rota ativa.

Ao detectar o contexto de navegação do usuário, o componente adapta as opções disponíveis: na tela principal (*Dashboard*), é exibido o botão de ação ”Nova *Live*”para cadastro rápido; já na tela de análise detalhada, a interface é simplificada, ocultando esta opção e priorizando a navegação de retorno ao painel principal e o encerramento da sessão (*Logout*). Essa estratégia foca a atenção do usuário nas ações pertinentes a cada etapa do fluxo de trabalho, evitando sobrecarga cognitiva.

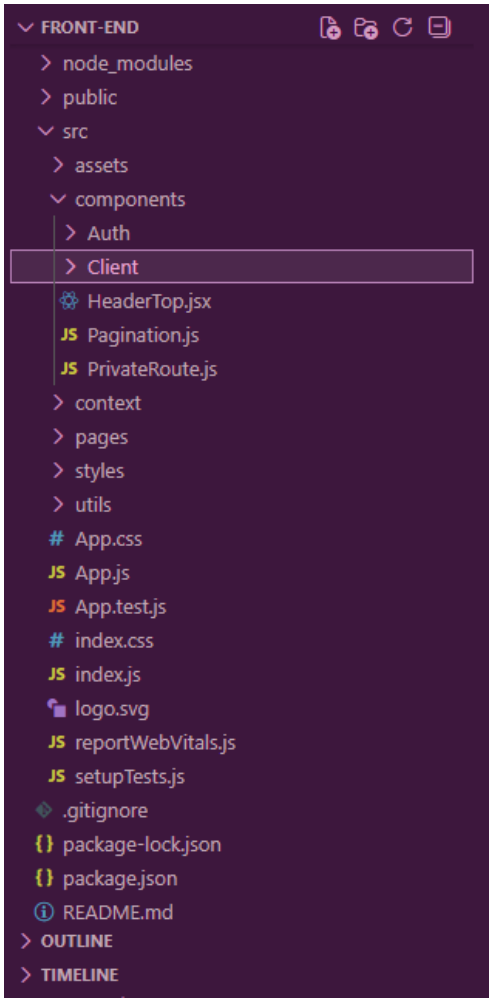
4.4.3.4 Organização do Projeto

No projeto desenvolvido, o diretório `src`, conforme apresentado na Figura 3 foi estruturada em diretórios funcionais com o objetivo de organizar de forma modular os elementos da aplicação, promovendo a reutilização de código e facilitando a manutenção. A arquitetura separa a lógica visual, o gerenciamento de estado e as funções auxiliares.

As principais pastas que compõem a estrutura são:

- ***assets***: Armazena recursos estáticos globais, como a logo para formar a identidade visual do projeto;
- ***components***: Abriga os elementos de interface reutilizáveis (botões, cabeçalhos, formulários) e componentes de lógica de proteção de rotas;
- ***context***: Contém a lógica de gerenciamento de estado global via *Context API*, isolando a regra de negócio de autenticação;
- ***pages***: Responsável por agrupar as ”visões”principais que são renderizadas pelas rotas do navegador;
- ***styles***: Centraliza arquivos de estilização e *Cascading Style Sheets* (CSS) globais para manter a consistência visual;

Figura 3 – Organização do *Frontend*



Autoria própria

- ***utils***: Contém funções auxiliares e lógica pura (*helper functions*) que podem ser invocadas por diferentes partes do sistema.

O diretório ***components***, por ser o núcleo da interface, foi subdividida em diretórios temáticos (`Auth` e `Client`) e contém arquivos essenciais para o funcionamento do sistema:

- `Auth/`: Agrupa os componentes exclusivos de autenticação, como formulários de *login* e registro;
- `Client/`: Contém os componentes da área administrativa, responsáveis pela exibição do *dashboard* e detalhes das transmissões;
- `HeaderTop.jsx`: O componente de cabeçalho adaptativo, que exibe navegação condicional baseada no estado do usuário;
- `Pagination.js`: Componente utilitário responsável pela navegação paginada em listas extensas de dados (como *logs* ou comentários);

- *PrivateRoute*.js: Um componente de ordem superior (*Higher-Order Component*) que atua como guardião, impedindo o acesso não autorizado a rotas privadas.

Essa organização reflete boas práticas de engenharia de *software* para aplicações *React.js*, garantindo que cada diretório tenha uma responsabilidade única e bem definida.

4.5 Integração e comunicação com a API

A integração entre o *frontend* e o *backend* é realizada através de requisições na modalidade *HyperText Transfer Protocol (HTTP) Representational State Transfer (Architectural Style) (RESTful)*. O sistema implementa autenticação baseada em *tokens JWT* para validar as sessões dos usuários e controlar o acesso aos recursos protegidos.

O sistema também implementa tratamento de erros padronizado, retornando códigos HTTP apropriados e mensagens descritivas para auxiliar no *debugging* e melhorar a experiência do usuário.

4.5.1 Segurança

A classe `config/SecurityConfig.java` implementa a configuração de segurança da aplicação utilizando *Spring Security*, definindo políticas de autenticação e autorização.

O método *filterChain*() configura a cadeia de filtros de segurança, permitindo acesso público aos *endpoints* de registro e *login* (*/auth/register* e */auth/login*) e à documentação *Swagger*, enquanto exige autenticação para todas as demais rotas. A proteção *Cross-Site Request Forgery* (CSRF) é desabilitada, prática comum em APIs RESTful *stateless* que utilizam autenticação baseada em *tokens*.

A codificação de senhas é realizada via *BCryptPasswordEncoder*, através do algoritmo *BCrypt*, que gera *hashes* seguros com *salt*[2] automático. O *AuthenticationManager* utiliza um *DaoAuthenticationProvider* configurado com o serviço customizado de detalhes de usuário (*UserDetailsServiceImpl*) e o codificador de senhas, possibilitando a validação de credenciais contra os dados armazenados no banco de dados.

[2] Uma sequência aleatória de caracteres adicionada a uma senha antes de ser transformada em *hash* e armazenada em um banco de dados.

4.5.2 Política de *CORS*

O método `corsConfigurationSource()` da classe `config/SecurityConfig.java` define uma política de CORS que permite requisições de origens específicas em ambiente de desenvolvimento (*localhost* nas portas `3000`, `5500` e `5501`), suportando métodos HTTP essenciais (`GET, POST, PUT, DELETE, OPTIONS`) e permitindo credenciais.

4.5.3 Documentação da API

A documentação da API é gerada automaticamente utilizando *OpenAPI* (*Swagger*), proporcionando uma *interface* interativa onde desenvolvedores podem explorar os *endpoints* disponíveis, visualizar *schemas* de requisição e resposta, e até mesmo testar as chamadas diretamente pelo navegador.

Cada *endpoint* é documentado com descrição detalhada, parâmetros obrigatórios e opcionais, tipos de dados esperados, códigos de resposta possíveis e exemplos de uso. A documentação também inclui informações sobre autenticação e versionamento da API. Esta abordagem facilita a integração por parte de desenvolvedores *frontend*, reduzindo erros de implementação e servindo como referência viva, já que se mantém sincronizada com o código.

4.6 *Backend*

O *backend* da aplicação é desenvolvido em Java com *Spring Boot*[3], seguindo uma arquitetura em camadas que separa responsabilidades entre controladores, serviços e repositórios.

A escolha do *Spring Boot* se deve à sua maturidade, robustez e ecossistema abrangente, oferecendo recursos nativos para injeção de dependências, gerenciamento de transações e integração com diferentes tecnologias.

O gerenciamento de dependências e *build* da aplicação é realizado através do *Maven*, que automatiza o processo de compilação, empacotamento e resolução de bibliotecas externas.

O sistema implementa interceptadores para *logging*, tratamento de erros, validação de dados e autenticação. A arquitetura modular facilita a manutenção, testabilidade e escalabilidade da aplicação, permitindo que novos recursos sejam adicionados sem impactar componentes existentes.

[3] *framework open source* baseado no ecossistema *Spring*.

4.6.1 Gestão de Credenciais e Variáveis

A gestão adequada de credenciais e configurações é fundamental para a segurança e manutenibilidade de aplicações modernas.

No projeto, foi adotada uma abordagem que separa as configurações sensíveis do código-fonte, utilizando variáveis de ambiente e arquivos de configuração específicos.

4.6.1.1 Arquivo de Variáveis de Ambiente

Para gerenciar credenciais sensíveis como chaves de API e configurações que podem variar entre ambientes (desenvolvimento, homologação, produção), foi criado um arquivo *.env* na raiz do projeto. Este arquivo contém as seguintes variáveis:

- *YOUTUBE_API_KEY*: Chave de autenticação para acesso à API do *YouTube Data* v3, necessária para recuperar informações de vídeos e comentários das transmissões ao vivo;
- *LLM_PROVIDER*: Define qual provedor de LLM (*Large Language Model*) será utilizado para análise dos comentários. No projeto, utiliza-se o valor `GROQ`, mas a arquitetura permite alternar entre diferentes provedores (*OpenAI*, *Gemini*, *Groq*) sem modificar o código;
- *LLM_API_KEY*: Chave de autenticação do provedor de LLM selecionado, necessária para realizar requisições à API de processamento de linguagem natural;
- *LLM_BATCH_SIZE*: Define o tamanho do lote de comentários que serão processados em cada requisição à LLM. O valor padrão de 30 foi definido para otimizar o equilíbrio entre custo de API, tempo de resposta e limite de *tokens*;
- *LLM_PROMPT*: Contém o *prompt* de sistema completo que instrui a LLM sobre como classificar os comentários. Este *prompt* define as categorias de reação (Negativo=0, Neutro=1, Positivo=2) e tipo de interação (Pergunta=3, Elogio=4, Crítica=5, Sugestão=6, Meme/Piada=7, Reclamação=8, Reação emocional=9, Ofensivo/Preconceituoso=10), além de especificar o formato exato da resposta esperada.

A utilização do arquivo `.env` oferece diversas vantagens:

1. As credenciais não são versionadas no repositório *Git*, mantendo-as seguras;
2. Diferentes desenvolvedores podem usar suas próprias chaves de API sem conflitos;
3. A transição entre ambientes é facilitada, bastando alterar o arquivo *.env* correspondente;

4. Configurações podem ser modificadas sem necessidade de recompilar o código.

4.6.1.2 Carregamento de Variáveis no Código

Para carregar as variáveis de ambiente definidas no arquivo `.env`, foi utilizada a biblioteca `dotenv-java`, que permite acessar essas configurações de forma programática. O carregamento é realizado através da classe *Dotenv*, como demonstrado na implementação da classe *LLMService*:

```
Dotenv dotenv = Dotenv.load();
this.llmProvider = dotenv.get("LLM_PROVIDER", "GROQ").toUpperCase();
this.batchSize = Integer.parseInt(dotenv.get("LLM_BATCH_SIZE", "30"));
```

O método *Dotenv.load()* realiza a leitura do arquivo `.env` localizado na raiz do projeto. Em seguida, o método *get()* é utilizado para recuperar valores específicos, aceitando dois parâmetros: o nome da variável e um valor padrão caso a variável não esteja definida.

No contexto deste trabalho, o provedor padrão é o *Groq*, pois ele oferece uma utilização gratuita com limites generosos por organização. Para o modelo empregado por padrão (*llama-3.3-70b-versatile*), os limites vigentes são de 30 requisições por minuto, 1 000 requisições por dia, 12 000 *tokens* por minuto e 100 000 *tokens* por dia, conforme documentação disponível (GROQ, 2025).

Na classe *GroqService*, o carregamento das variáveis de ambiente é realizado de forma semelhante ao processo adotado em outras partes do sistema. Inicialmente, a instrução *Dotenv.load()* lê o arquivo `.env` localizado na raiz do projeto e disponibiliza todas as variáveis definidas nele. Por sua vez, os atributos *API_KEY* e *SYSTEM_PROMPT* são inicializados por meio do método `get()`, que recupera os valores correspondentes às chaves *LLM_API_KEY* e *LLM_PROMPT*.

Como esses atributos são declarados como `final`, tornam-se imutáveis após a construção da instância da classe, o que garante previsibilidade e segurança em cenários concorrentes. Além disso, a chave de API carregada nesse ponto será utilizada posteriormente para compor as requisições HTTP enviadas ao provedor *Groq*:

```
private final Dotenv dotenv = Dotenv.load();
private final String API_KEY = dotenv.get("LLM_API_KEY");
private final String SYSTEM_PROMPT = dotenv.get("LLM_PROMPT", "");
```

Aqui, as variáveis *LLM_API_KEY* e *LLM_PROMPT* são carregadas como atributos finais da classe, tornando-as imutáveis após a inicialização e garantindo *thread-safety*.

Nesse contexto, *thread-safety* significa que, como esses valores não podem ser alterados depois de definidos, múltiplas *threads* podem acessar a mesma instância

da classe simultaneamente sem risco de condições de corrida (*race conditions*) ou inconsistências de estado.

Em outras palavras, o uso de atributos `final` impede modificações concorrentes e assegura que todas as *threads* visualizem exatamente o mesmo valor, de forma segura e previsível. A chave de API, por sua vez, é utilizada posteriormente na construção das requisições HTTP para o *endpoint* da *Groq*.

4.6.1.3 Papel da Engenharia de *Prompt* no processo de classificação

A Engenharia de *Prompt* é um componente essencial quando se utiliza uma *Large Language Model* (LLM) como classificadora, pois o desempenho e a consistência dos resultados dependem diretamente de como a tarefa é especificada ao modelo. No *Live Insights*, o `LLM_PROMPT` desempenha três funções centrais:

Definir as categorias de classificação: explicita ao modelo quais classes existem em cada eixo (reação e tipo de interação) e como interpretá-las;

Padronizar as respostas do modelo: impõe um formato de saída controlado (por exemplo, códigos/estrutura fixa), facilitando o *parsing*, armazenamento e visualização;

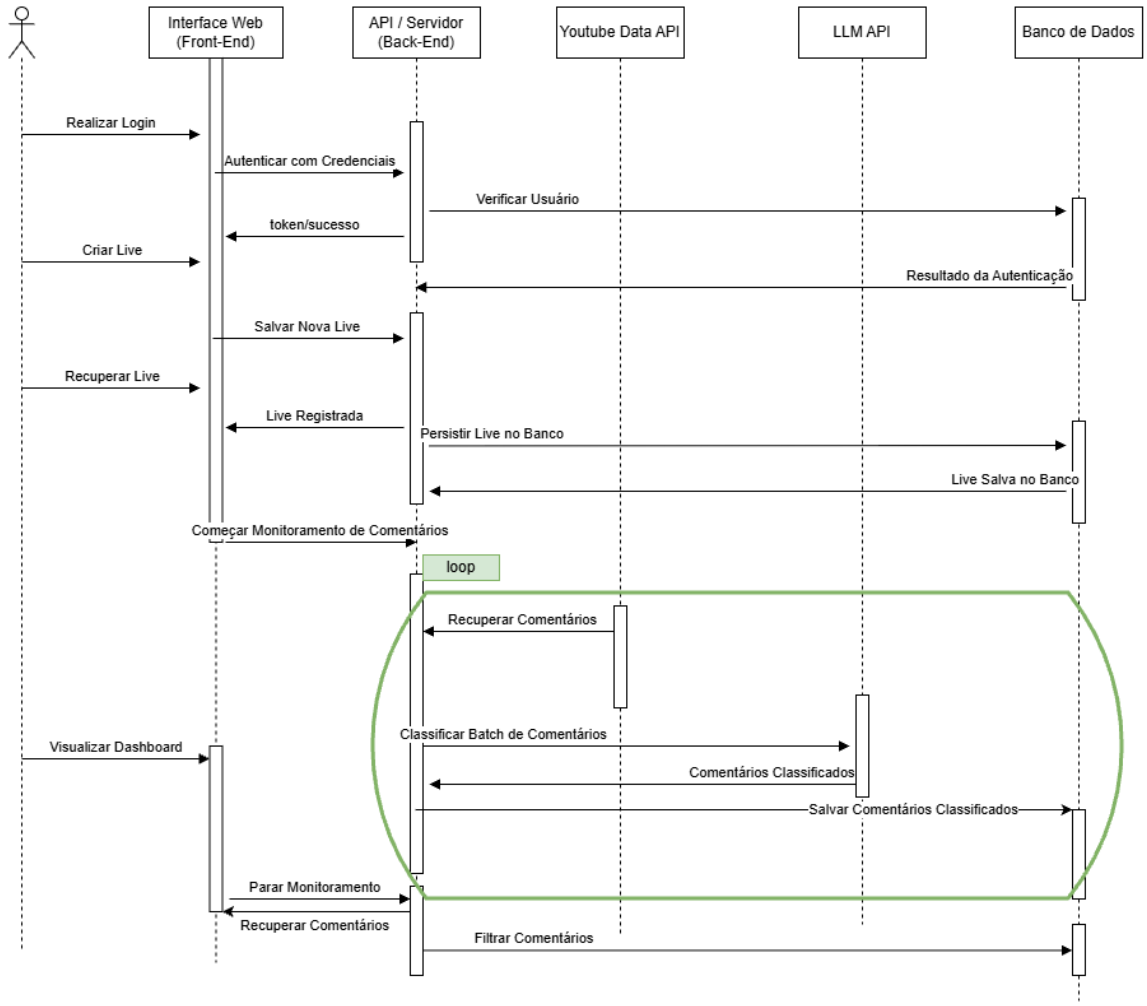
Garantir consistência nos resultados: reduz variações de resposta entre comentários semelhantes e aumenta a reprodutibilidade do sistema, especialmente quando os comentários são processados em lotes.

Dessa forma, o *prompt* não é apenas “texto auxiliar”, mas parte do próprio mecanismo de controle de qualidade do pipeline de classificação.

4.6.2 Captura e classificação de comentários

O diagrama de sequência apresentado na Figura 4 descreve o fluxo de interação entre os principais componentes do sistema durante o processo de autenticação, cadastro e monitoramento de uma transmissão ao vivo:

Figura 4 – Diagrama de *Sequência*



Autoria própria

O ator *Usuário* interage inicialmente com a *Interface Web* (*Frontend*), que se comunica com a API / Servidor (*Backend*) para realizar a autenticação por meio da chamada ao método `autenticar(credenciais)`.

Em seguida, o usuário registra uma nova *live*, cujo identificador é persistido no banco de dados. Ao selecionar uma transmissão, o sistema inicia o processo de monitoramento, que consiste em um ciclo contínuo de busca e análise de comentários. O *Backend* obtém os comentários em tempo real por meio da *YouTube Data API* e envia os lotes coletados para a LLM API, responsável por classificá-los segundo critérios pré-definidos (por exemplo, reações, relevância ou tópicos). Os resultados dessas classificações são então armazenados no banco de dados.

Por fim, o usuário pode visualizar os dados agregados no painel (*Dashboard*) ou consultar comentários específicos aplicando filtros. Esse fluxo demonstra a integração entre múltiplas camadas e serviços externos, assegurando uma análise contínua e automatizada das interações durante transmissões ao vivo.

Esse diagrama evidencia a sequência lógica e temporal das trocas de mensagens entre os componentes, demonstrando como o sistema integra diferentes serviços, como a *YouTube Data API*, a LLM API e o banco de dados, para garantir uma experiência contínua e automatizada de monitoramento e análise de transmissões.

4.6.3 Tratamento de Erros

O tratamento global de exceções da aplicação é centralizado em uma classe anotada com `@ControllerAdvice`(*GlobalExceptionHandler*.java), o que permite ao *Spring* interceptar exceções lançadas pelos controladores e tratá-las de maneira padronizada. Dessa forma, evita-se a repetição de blocos *try-catch* ao longo do código e garante-se que a API retorne respostas consistentes em caso de falhas.

A classe define dois manipuladores principais de exceções. O primeiro, anotado com `@`*ExceptionHandler*(Exception.class), captura qualquer exceção genérica não tratada e retorna uma resposta com o status 500 (*Internal Server Error*). O segundo manipulador, específico para *EntityNotFoundException*, é acionado quando uma entidade requisitada não é encontrada no banco de dados, retornando ao cliente um status 404 (*Not Found*).

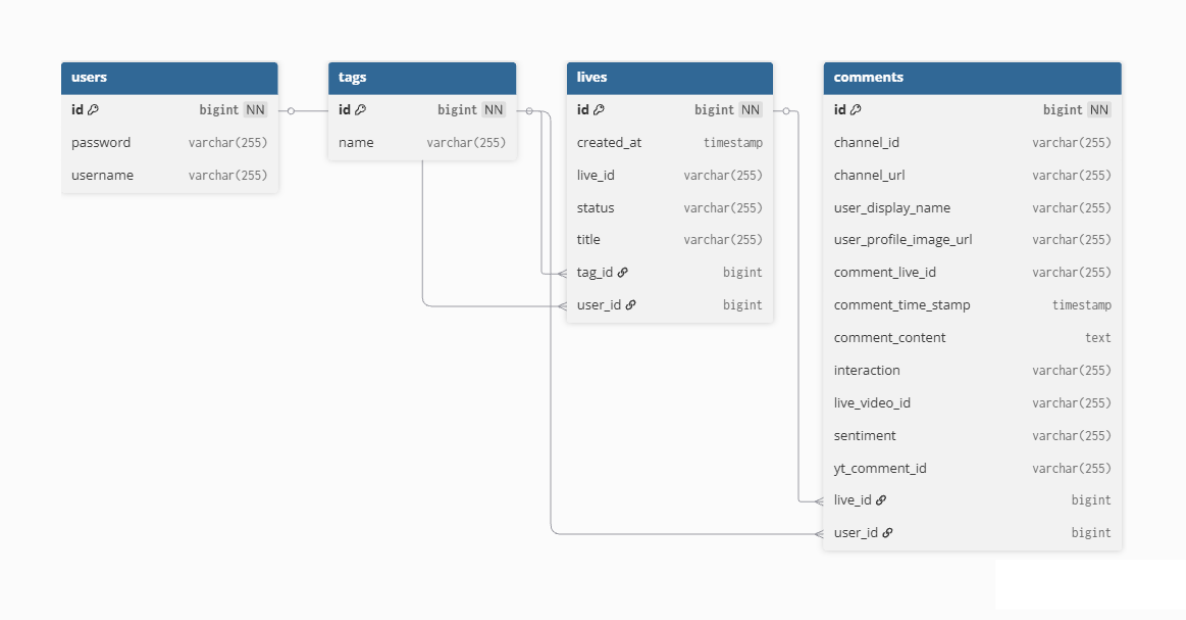
Ambos os manipuladores delegam a construção da resposta ao método que encapsula a mensagem de erro em um objeto *ErrorResponse*, juntamente com o código HTTP correspondente (*buildErrorResponse*()). Antes de gerar a resposta, a exceção é registrada por meio do método *logException*(), que imprime o *stack trace* no *console*, facilitando o diagnóstico durante o desenvolvimento.

Esse mecanismo centralizado garante que a API responda de forma previsível e padronizada frente a diferentes tipos de falhas, além de simplificar a manutenção do código e melhorar a rastreabilidade dos erros.

4.7 Banco de Dados

A camada de repositório do *Backend* está diretamente relacionada ao diagrama da Figura 5, pois implementa o acesso e a manipulação dos dados persistidos nas tabelas correspondentes às entidades do modelo:

Figura 5 – Diagrama de *Entidade Relacionamento*



Autoria própria

Cada repositório mapeia uma entidade do Diagrama de Entidade Relacionamento (DER), como *User, Live, Tag* e *CommentsInfo*, permitindo realizar operações de consulta, inserção, atualização e exclusão de registros de forma abstrata, por meio do uso do *Spring Data Java Persistence API (JPA)*.

Assim, interfaces que herdam do `JpaRepository` interagem diretamente com o banco de dados `PostgreSQL`, definido na configuração da aplicação, refletindo os relacionamentos e chaves estrangeiras especificados no DER.

Essa integração garante consistência entre o modelo conceitual e a camada de persistência, assegurando que as relações entre usuários, transmissões, comentários e *tags* sejam mantidas conforme o desenho lógico do sistema.

No modelo JPA, essas relações são mapeadas por meio de anotações específicas, como `@OneToMany`, `@ManyToOne` e `@ManyToMany`, que correspondem aos relacionamentos 1:N, N:1 e N:N presentes no DER. As chaves estrangeiras definidas no banco de dados *PostgreSQL* garantem a integridade referencial entre as tabelas, permitindo que a aplicação realize consultas e atualizações complexas respeitando as regras de negócio estabelecidas pelo modelo.

4.7.1 Configuração do Banco de Dados

Além das variáveis de ambiente, a aplicação utiliza o arquivo `application.yml` para configurar o acesso ao banco de dados e outras propriedades do *Spring Boot*. A centralização das configurações nesse arquivo oferece vantagens significativas: separação clara entre código e configuração, facilidade de manutenção, suporte a diferen-

tes perfis de execução através de arquivos adicionais como *application-dev.yml* e a*pplication-prod.yml*, e integração natural com o ecossistema *Spring Boot*, que automatiza grande parte da configuração de infraestrutura necessária para a persistência de dados.

Dentre as configurações estabelecidas, esse arquivo centraliza a especificação referente à conexão com o *PostgreSQL* e ao comportamento do *Hibernate/JPA*, da seguinte forma:

```
spring:
  datasource:
    driverClassName: org.postgresql.Driver
    url: jdbc:postgresql://localhost:5432/live-insights
    username: postgres
    password: postgres
  jpa:
    hibernate.ddl-auto: update
    show-sql: true
    format-sql: true
```

A seção `spring.datasource` contém os parâmetros essenciais para estabelecer a conexão com o banco de dados. A propriedade *driverClassName* especifica o driver *Java Database Connectivity* (JDBC) do *PostgreSQL* (*org.postgresql.Driver*), responsável por intermediar a comunicação entre a aplicação Java e o sistema gerenciador de banco de dados.

A propriedade *url* define a *string* de conexão JDBC no formato, indicando que o banco está sendo executado localmente na porta padrão do *PostgreSQL* (i.e., `5432`), e *live-insights* é o nome do banco de dados utilizado pelo sistema. As credenciais de acesso são configuradas através das propriedades *username* e *password*. No exemplo apresentado, ambas utilizam o valor padrão *postgres*, adequado para ambientes de desenvolvimento local.

Para ambientes de produção, recomenda-se fortemente a utilização de variáveis de ambiente, permitindo que as credenciais sejam externalizadas e mantidas fora do controle de versão, aumentando a segurança da aplicação.

A seção `spring.jpa` configura o comportamento da JPA e do *Hibernate*. A propriedade *hibernate.ddl-auto: update* é particularmente importante, pois instrui o *Hibernate* a atualizar automaticamente o esquema do banco de dados sempre que a aplicação é iniciada. Este modo compara as entidades JPA definidas no código com a estrutura atual das tabelas no banco e aplica as alterações necessárias, como criação de novas tabelas, adição de colunas ou modificação de tipos de dados.

A propriedade `show-sql: true` habilita a exibição dos comandos SQL gerados pelo *Hibernate* no console da aplicação. Esta funcionalidade é valiosa durante o desenvolvimento e depuração, pois permite aos desenvolvedores visualizar exatamente quais operações estão sendo executadas no banco de dados, facilitando a identificação de consultas ineficientes ou comportamentos inesperados.

Complementando esta funcionalidade, a propriedade `format-sql: true` instrui o *Hibernate* a formatar os comandos SQL de maneira legível, com indentação e quebras de linha apropriadas, tornando a análise das consultas ainda mais simples.

Frise-se que, durante o desenvolvimento, essas configurações são extremamente convenientes, eliminando a necessidade de executar *scripts* SQL manualmente a cada mudança no modelo de dados; entretanto, não são recomendadas para utilização em produção, uma vez que a formatação e a exibição detalhada das consultas podem introduzir sobrecarga desnecessária e potencialmente expor informações sensíveis nos *logs*.

4.8 Arquitetura

A arquitetura do sistema segue princípios de DDD e *Clean Code* (Arquitetura Limpa), promovendo separação de responsabilidades e independência de *frameworks* externos. O sistema é dividido nas seguintes camadas:

1. *Model*

   Representa o domínio da aplicação, com entidades que refletem conceitos do negócio. Seguindo DDD, essas classes encapsulam regras de negócio e comportamentos essenciais, garantindo que alterações em detalhes de implementação não afetem a lógica central.

2. *Controller*

   Funciona como porta de entrada da aplicação, recebendo requisições HTTP, delegando o processamento para os serviços e retornando respostas ao cliente. Mantém a lógica de negócio isolada da infraestrutura de comunicação.

3. *Service*

   Aplica princípios *SOLID*[4] é amplamente utilizado no *design* de *software*, especialmente inversão de dependência (depende de abstrações, não de implementações) e segregação de *interfaces*. É responsável pela execução das ações do sistema, orquestrando a lógica de negócio e chamando dependências externas.

[4] Acrônimo para cinco princípios da orientação a objetos: *Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation* e *Dependency Inversion*.

4. *Repository*

   Abstrai o acesso a dados utilizando *JpaRepository*, permitindo operações *Create, Read, Update, Delete (CRUD)* sem implementação explícita. Além disso, o sistema usa *JpaAuditing*, ativado por meio de *@EnableJpaAuditing* na classe *LiveInsightsApplication*.java, para rastrear alterações em entidades automaticamente.

5. *Data Transfer Objects (DTOs)*

   São objetos de transferência de dados utilizados em *requests* e *responses*, desacoplados das entidades de domínio, garantindo segurança e clareza na comunicação entre cliente e servidor.

Essa estrutura garante que mudanças em detalhes de implementação não afetem as regras de negócio centrais. Padrões de design como *Repository* e *Dependency Injection* promovem desacoplamento, facilitando escalabilidade e manutenção da arquitetura a longo prazo.

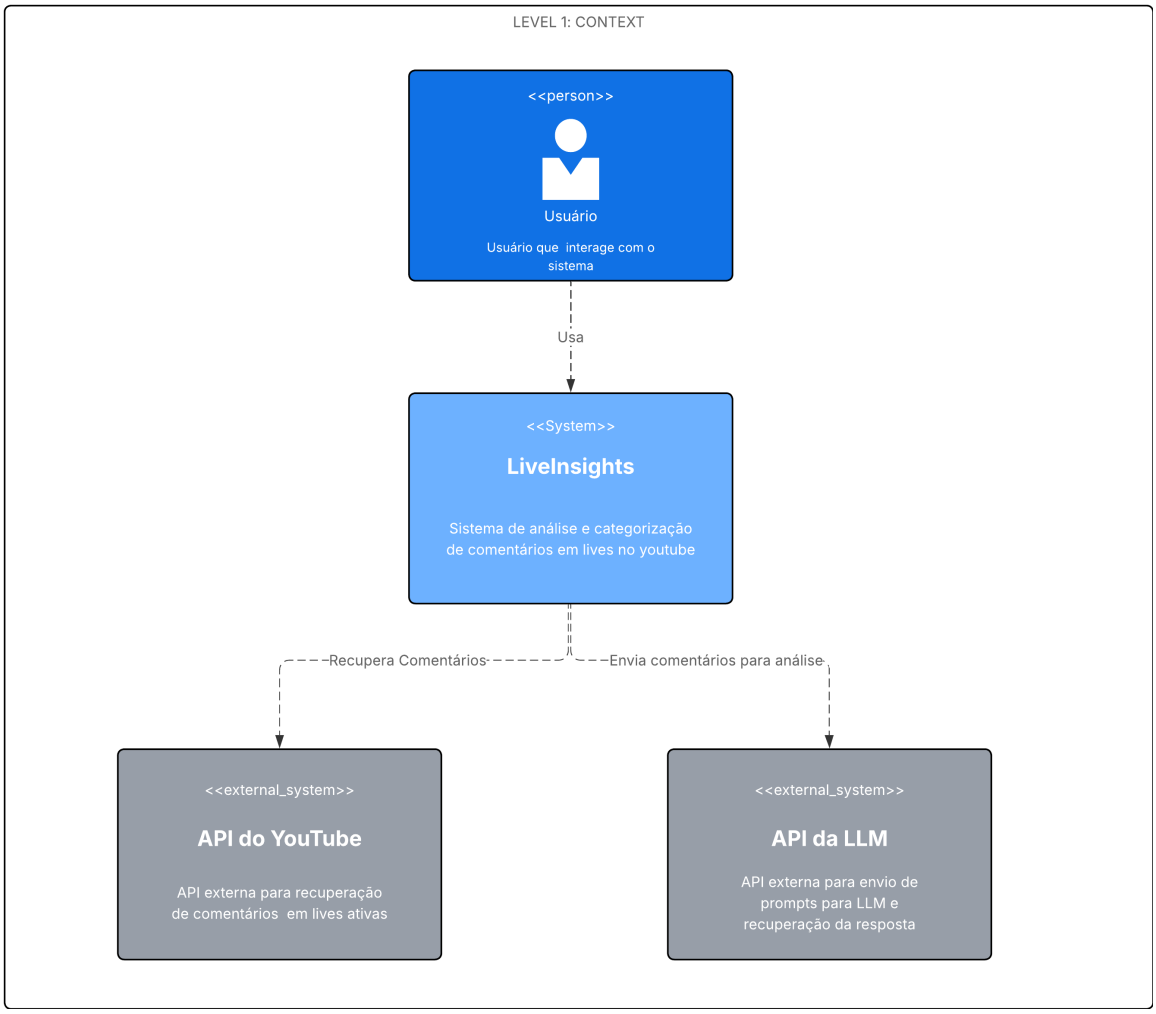
4.8.1 Diagramas C4

Para complementar a documentação da arquitetura, foi utilizado o modelo C4[5], proposto por Simon Brown (BROWN, 2018). Tendo como objetivo reduzir a carga cognitiva e criar um vocabulário visual simples e compreensível. O modelo permite descrever o sistema em quatro níveis de abstração: Contexto, *Containers*, Componentes e Código.

A visão de contexto do sistema está apresentada na Figura 6, e tem como foco as relações e dependências externas, evidenciando quais agentes humanos, ou outros sistemas, interagem com o *software* e com que propósito.

[5] O Modelo C4 é um método de visualização da arquitetura de software. Divide-se em quatro níveis: (1) Contexto, que mostra o sistema e seus atores externos; (2) Contêineres, que descreve as aplicações e serviços que compõem o sistema; (3) Componentes, que detalha a estrutura interna de cada contêiner; e (4) Código, que apresenta a implementação em nível de classes.

Figura 6 – Diagrama de Contexto do Sistema



Autoria própria

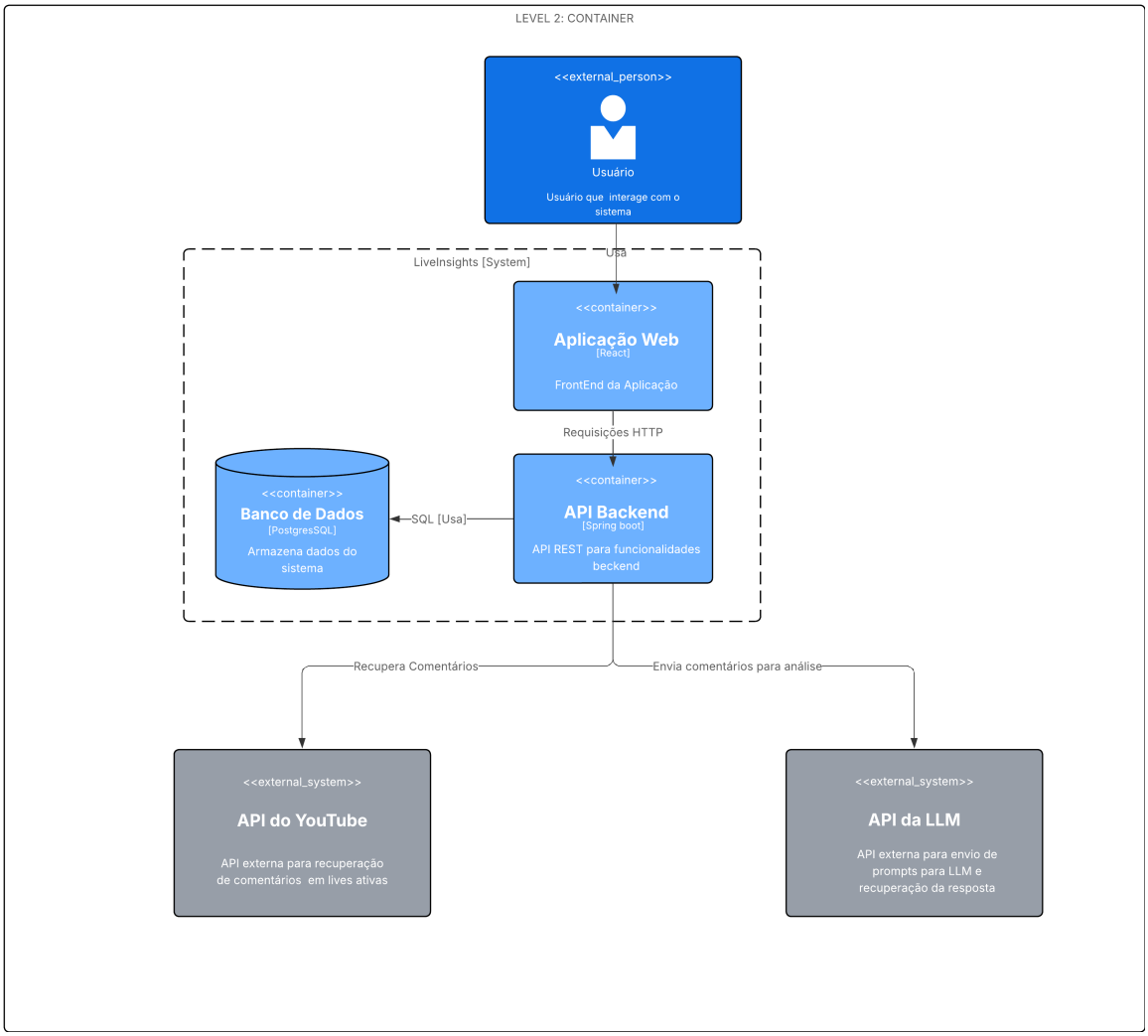
Pode-se observar, além da interação do usuário com o *LiveInsights*, a comunicação do mesmo com dois sistemas externos distintos:

API do *YouTube*: Responsável pela coleta e envio das informações referentes às *Lives*.

API da LLM: Atua recebendo os comentários coletados para análise e retornando suas classificações.

Seguindo para os *containers*, representados na Figura 7, é possível identificar a divisão das principais aplicações e serviços do sistema, e como eles conversam entre si.

Figura 7 – Diagrama de *Containers* do Sistema



Autoria própria

Analisando os *containers* representados no diagrama é possível identificar:

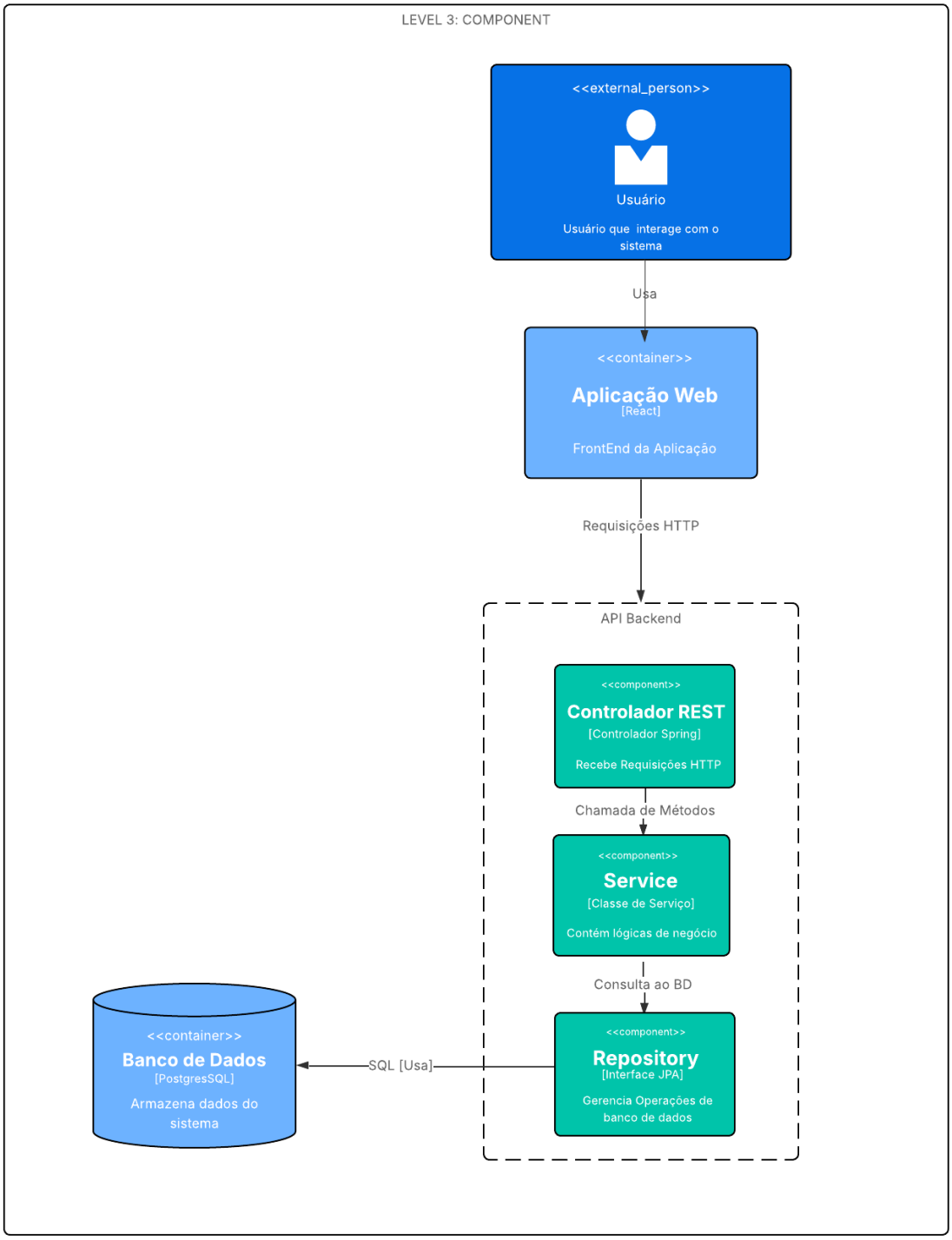
Aplicação *Web*: Interface que conversa diretamente com o usuário, obtendo as informações necessárias e realizando os comandos solicitados através de chamadas aos *endpoints* do *container* subsequente.

API *Backend*: Encarregado de receber requisições, executar regras de negócio, orquestrar fluxos, aplicar validações, consultar ou persistir dados e devolver respostas estruturadas. Atua como ”cérebro”do sistema, concentrando a lógica central que não pertence à interface ou ao armazenamento.

Banco de Dados: Representando a camada de persistência, é responsável por armazenar os dados de maneira durável, garantindo a integridade dos dados, permitindo consultas estruturadas e provendo os registros requisitados pela API sempre que necessário ler ou gravar informações.

O detalhamento interno do *container* ”API *Backend*”, descrevendo sua decomposição modular e lógica é apresentado no diagrama de componentes da Figura 8:

Figura 8 – Diagrama de Componentes do Sistema



Autoria própria

Neste nível, o *container* da API é decomposto em três responsabilidades centrais:

Controlador REST: Tem como função receber as requisições HTTP executadas pelo *frontend*, interpretar parâmetros da requisição, acionar os fluxos apropriados e devolver as respostas ao solicitante

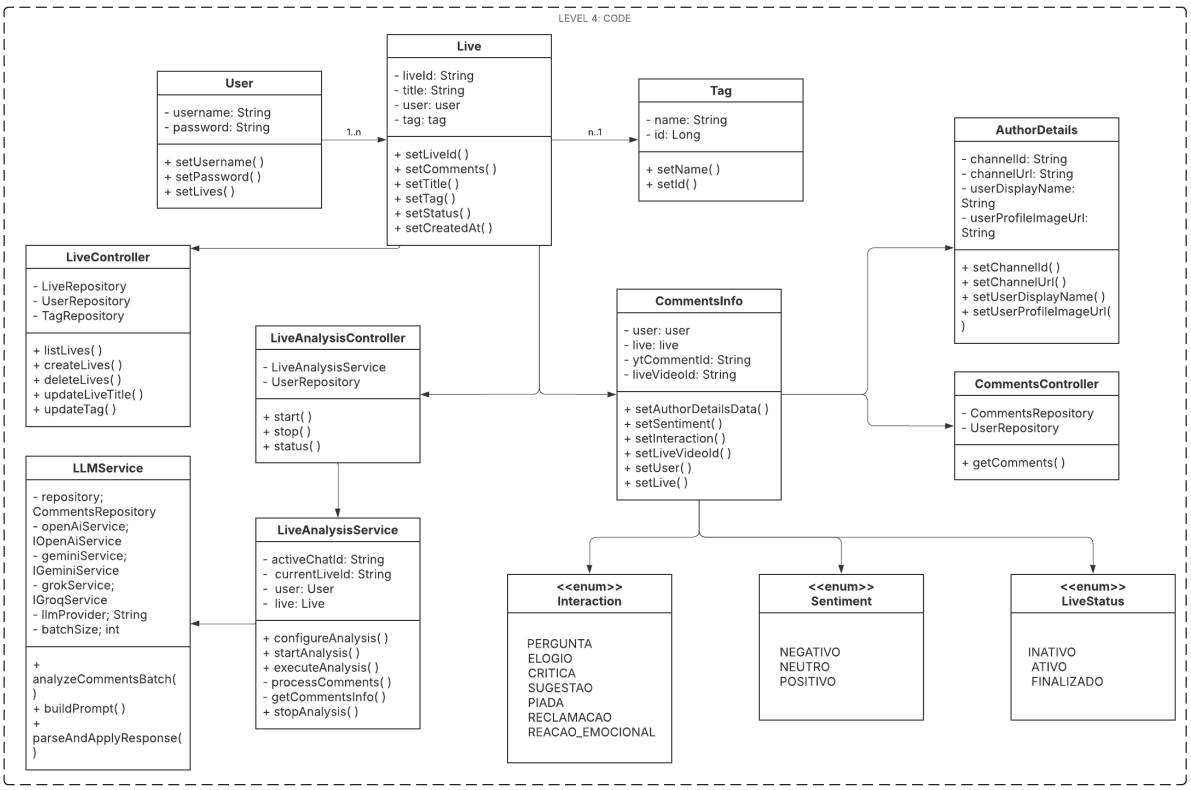
Service: É o componente que concentra as regras de domínio e processamento

principal do sistema. Uma vez acionado pelo Controlador, o *Service* executa as operações essenciais, determina o comportamento adequado para cada fluxo, aplica validações e realiza transformações nos dados sempre que necessário, além de interagir com o *Repository* para recuperar ou persistir informações no banco de dados.

Repository: Retratando a camada de acesso a dados, é o componente encarregado por interagir diretamente com o banco de dados, concentrando as operações de consulta e persistência. Esse elemento atua como o ponto oficial de comunicação com a base de dados, garantindo isolamento dessa responsabilidade e fornecendo os métodos necessários para que as demais camadas do sistema acessem ou armazenem informações de forma segura, organizada e conforme os contratos da aplicação.

Para finalizar o modelo, o último nível é representado na Figura 9:

Figura 9 – Diagrama de Código / Classes do Sistema



Autoria própria

O nível de código liga a modelagem arquitetural à implementação efetiva do sistema, descrevendo a estrutura interna pro meio de representações das classes, funções, *interfaces* e outros artefatos adotados pelo projeto.

4.8.2 *Domain Driven Design* (DDD)

O DDD é uma abordagem de desenvolvimento de *software* introduzida por Evans (2016), que visa lidar com a complexidade inerente a sistemas corporativos focando

no ”coração”da aplicação: o domínio do negócio. Segundo o autor, o sucesso de um *software* não depende apenas da qualidade do código ou da infraestrutura, mas, primordialmente, de o modelo computacional refletir com precisão a realidade do problema que se propõe a resolver.

Um dos pilares centrais dessa metodologia é o conceito de Linguagem Ubíqua (*Ubiquitous Language*). Evans (2016) defende que desenvolvedores e especialistas de negócio (os *domain experts*) devem cultivar uma linguagem comum, rigorosa e compartilhada, livre de ambiguidades. Esse vocabulário deve permear todas as esferas do projeto, desde as reuniões de planejamento até o código-fonte final, evitando a necessidade de traduções constantes entre termos técnicos e funcionais.

Para estruturar essa complexidade, o DDD propõe padrões estratégicos, como os Contextos Delimitados (*Bounded Contexts*), que definem limites lógicos onde um determinado modelo é válido, e padrões táticos, que fornecem os blocos de construção para a implementação, como Entidades, Agregados, Objetos de Valor e Repositórios. Essa arquitetura em camadas isola o domínio, protegendo as regras de negócio de dependências tecnológicas externas.

No contexto específico do projeto *Live Insights*, a aplicação dos princípios do DDD foi fundamental para gerenciar a complexidade inerente à interpretação de linguagem natural. O domínio da aplicação não reside apenas na coleta técnica de dados, mas na capacidade de transformar texto não estruturado em métricas de engajamento qualificadas.

Nesse cenário, a adoção da Linguagem Ubíqua permitiu alinhar o vocabulário técnico das classes, nomeadas explicitamente como *Live*, *Interaction* e *Comment*, por exemplo, com as necessidades reais dos moderadores e criadores de conteúdo, garantindo que o código reflita fielmente o problema de negócio resolvido, conforme preconiza a metodologia.

Sob a perspectiva tática, o sistema utilizou o conceito de Agregados para manter a consistência dos dados. A entidade $Live$ atua como uma raiz de agregado, centralizando o ciclo de vida do monitoramento e garantindo que os comentários processados pertençam inequivocamente a uma sessão de transmissão válida.

Essa separação estratégica assegura que o núcleo da aplicação permaneça agnóstico em relação às tecnologias externas. Dessa forma, as complexas interações com a API do *YouTube* e os serviços de inferência de IA foram isoladas em camadas de adaptação (Infraestrutura), permitindo que o modelo de domínio permaneça puro e focado exclusivamente na lógica de análise e geração de *insights*, facilitando a testabilidade e a evolução sustentável do *software*.

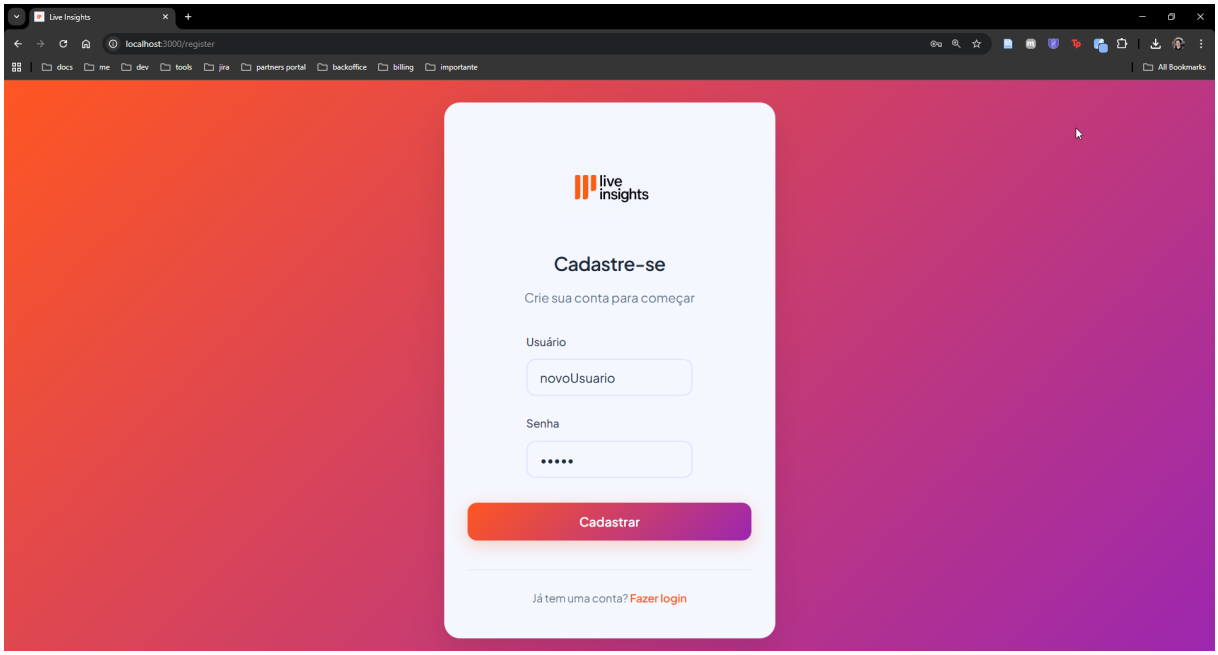
5 Demonstração da Aplicação

Este capítulo apresenta a aplicação desenvolvida em funcionamento, mostrando suas principais funcionalidades, a forma como a *interface* foi organizada e os fluxos de uso que compõem o sistema. A intenção é demonstrar, de maneira objetiva, como os requisitos definidos ao longo do projeto foram implementados e como a aplicação se comporta em situações reais. Com isso, é possível verificar na prática se as escolhas de arquitetura, usabilidade e tecnologias adotadas atenderam às necessidades propostas.

5.1 Acesso à Aplicação

A *interface* do *Frontend* do sistema permite ao usuário realizar diversas operações essenciais para o uso da plataforma. Inicialmente, para criar um novo usuário conforme pode ser visualizado na Figura 10, o sistema oferece uma rota específica acessível em <http://localhost:3000/register> , na qual é possível preencher os dados necessários para o cadastro:

Figura 10 – Cadastro de Usuário

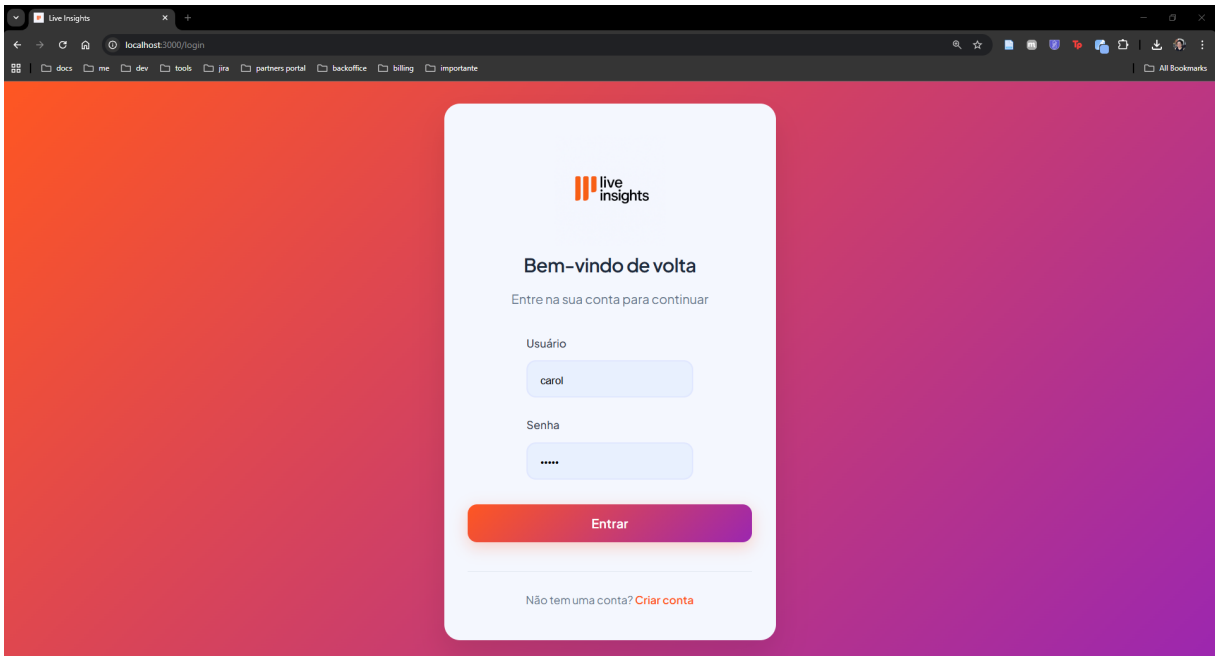


Autoria própria

Após o cadastro, o usuário pode realizar o *login* por meio da rota <http://localhost:3000/login> conforme apresentado na Figura 11, utilizando as credenciais previamente cadastradas. Nessa etapa, a senha enviada é comparada com o *hash* criptografado

que foi salvo no banco de dados para autorizar ou não o acesso daquele usuário ao sistema.

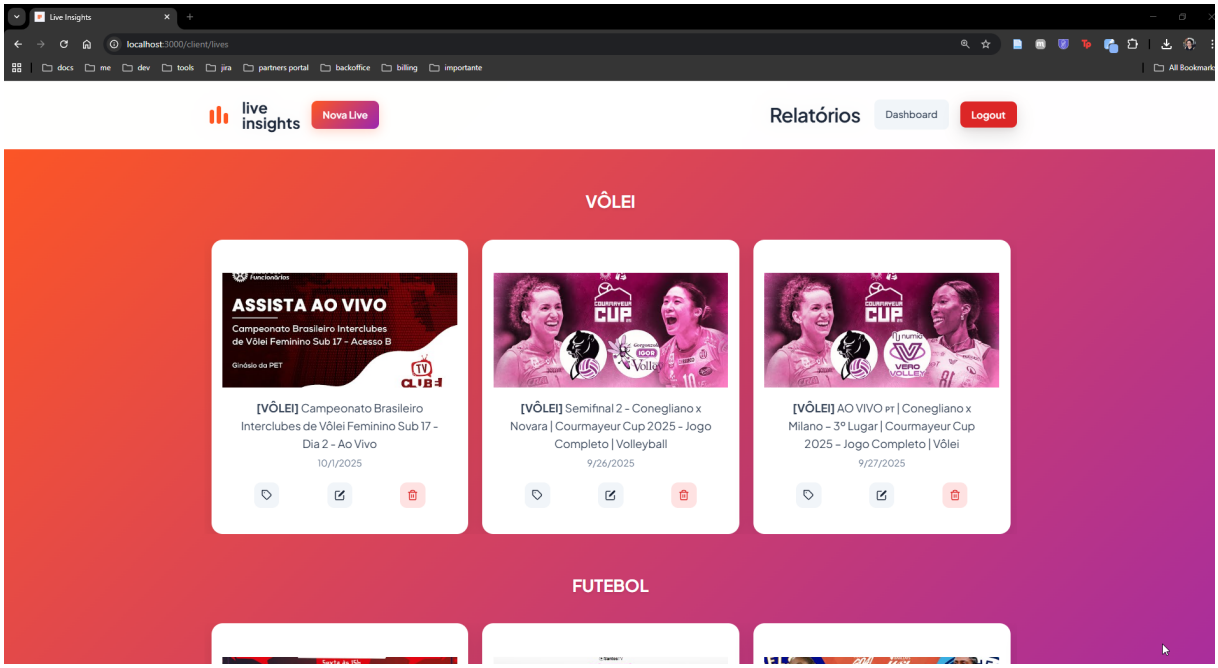
Figura 11 – *Login* de Usuário



Autoria própria

Uma vez autenticado, o usuário é direcionado para a tela inicial, que apresenta um *dashboard* contendo os *cards* das *lives* já cadastradas, disponíveis para análise. A Figura 12 representa os *Dashboards* de *Lives*.

Figura 12 – *Dashboard de Lives*

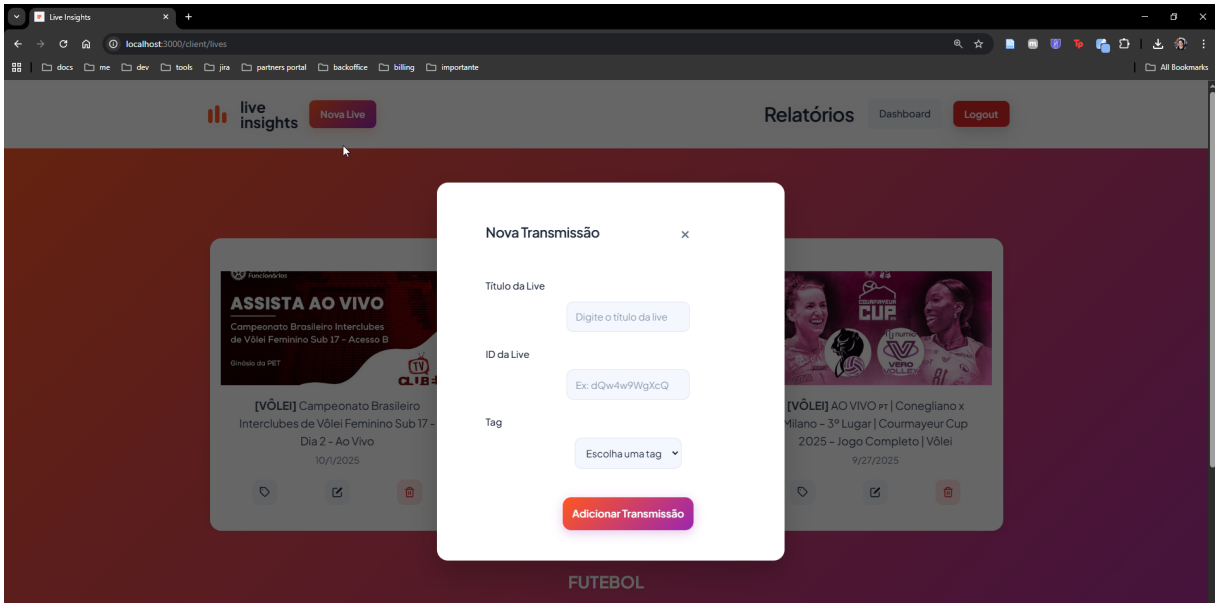


Autoria própria

5.2 Cadastro de *Lives*

Na tela principal, representada na Figura 13, também é possível cadastrar uma nova *live*, ampliando o conjunto de transmissões monitoradas pelo sistema:

Figura 13 – Cadastro de *Lives*



Autoria própria

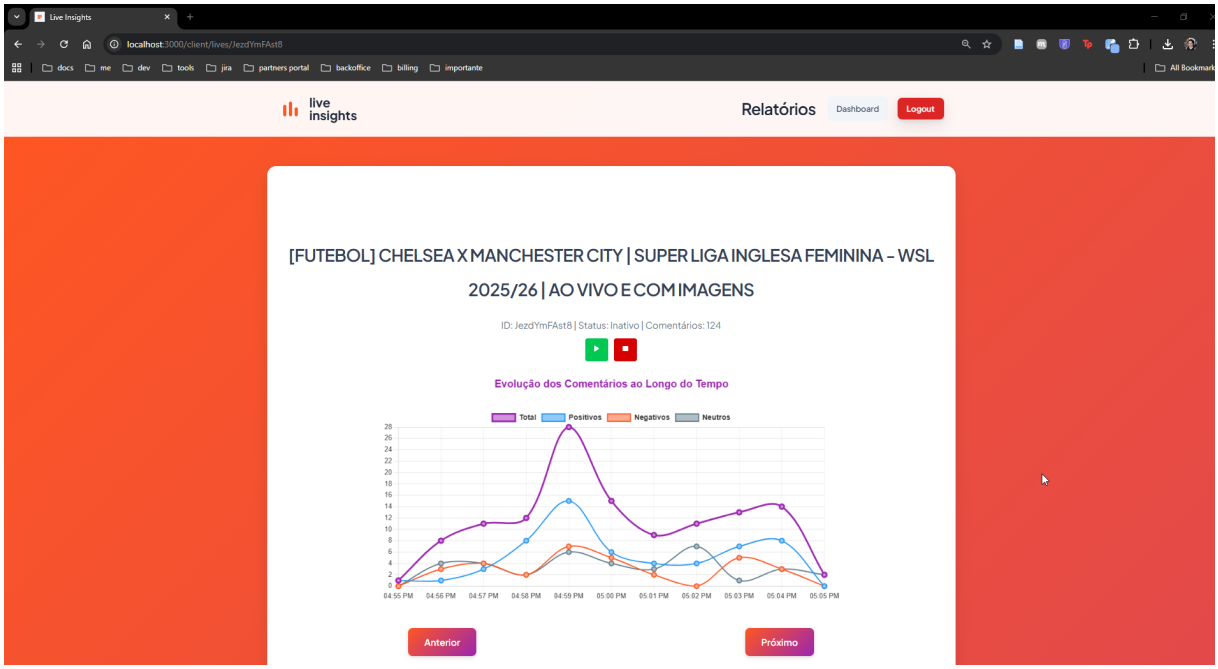
5.3 Acompanhamento de *Live*

Ao selecionar um *card* de *live*, o usuário acessa a análise detalhada daquela transmissão, que inclui resultados gráficos e estatísticos sobre o conteúdo monitorado.

Além das métricas quantitativas de comentários, a análise detalhada incorpora estatísticas de engajamento, que são vitais para medir a interação do público com o conteúdo, o que inclui o volume e a taxa de reações positivas, negativas e neutras no *chat* ao vivo ao longo do tempo.

Tais métricas permitem o melhor entendimento da performance da transmissão, conforme apresentado na Figura 14:

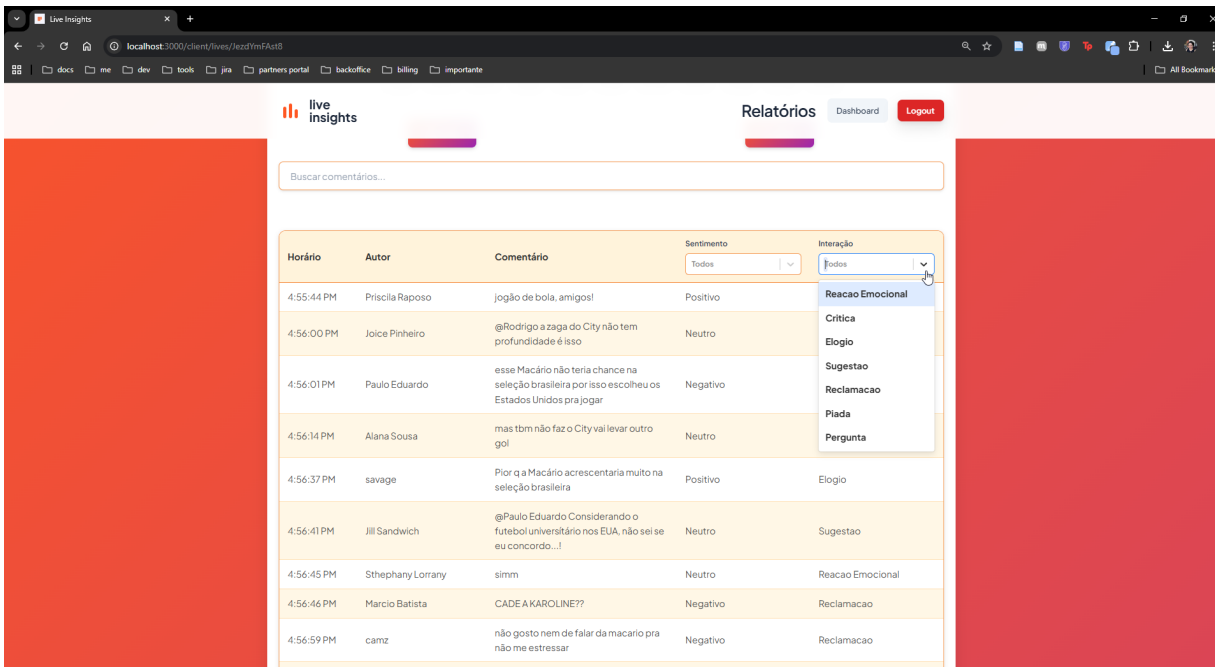
Figura 14 – Acompanhamento de Lives



Autoria própria

Além disso, a plataforma possibilita a visualização e filtragem dos comentários realizados durante as *lives*, com filtros aplicáveis por reação e por interação, permitindo uma análise mais segmentada e detalhada do *feedback* do público. A Figura 15 apresenta uma visualização dos comentários e filtros do sistema:

Figura 15 – Comentários e Filtros



Autoria própria

Essas funcionalidades tornam a experiência do usuário completa e intuitiva, facili-

tando tanto o acompanhamento quanto a gestão das transmissões ao vivo.

A partir da demonstração apresentada, foi possível observar o sistema funcionando de forma completa e compreender como cada parte contribui para o todo. Os testes realizados e os fluxos exibidos mostram que as funcionalidades implementadas atendem ao que foi planejado no projeto. Com isso, reforça-se que as escolhas técnicas e de *design* feitas ao longo do desenvolvimento foram adequadas, permitindo seguir para as considerações finais do trabalho.

6 Conclusões

No presente trabalho, foi proposto o desenvolvimento e a validação do *Live Insights*, uma plataforma de inteligência de dados voltada à análise de reações e à extração de métricas em tempo real a partir de comentários de transmissões ao vivo no *YouTube*.

A solução integra técnicas modernas de PLN, utilizando LLMs, com arquitetura escalável baseada em *Spring Boot* e *React.js*, garantindo coleta automatizada, classificação semântica e visualização dinâmica das interações da audiência.

O sistema foi estruturado para atender às demandas de criadores de conteúdo, moderadores e gestores de *marketing* que buscam compreender, em tempo real, a percepção do público durante eventos ao vivo. Por meio da integração com a *YouTube Data API v3* e provedores de serviços de LLMs (como *Groq*, *OpenAI* e *Gemini*), o *Live Insights* demonstrou capacidade de processar grandes volumes de comentários, classificá-los segundo reação e tipo de interação, e apresentar resultados consolidados de forma acessível e acionável.

Mediante os resultados obtidos, é possível responder às questões levantadas:

RQ1: Como os LLMs podem ser aplicados para realizar a análise de reações em comentários de transmissões ao vivo no *YouTube*, permitindo a extração de métricas em tempo real?

***Research Answer* (RA) 1:** Os LLMs demonstraram eficácia na análise de reações em comentários de *lives* ao oferecerem compreensão contextual profunda, superando limitações de abordagens baseadas exclusivamente em léxicos ou modelos estatísticos tradicionais. No *Live Insights*, a aplicação de modelos como o *llama-3.3-70b-versatile* (provedor *Groq*) permitiu classificar comentários segundo duas dimensões principais: reação (negativo, neutro, positivo) e tipo de interação (pergunta, elogio, crítica, sugestão, meme/piada, reclamação, reação emocional, ofensivo/preconceituoso).

A arquitetura adotada possibilitou processamento em lotes de até 30 comentários por requisição, com latência média inferior a 5 segundos, viabilizando análise em tempo real mesmo durante transmissões com alto volume de interações. O uso de *prompts* estruturados, com instruções claras sobre categorias e formato de resposta em JSON, garantiu consistência e previsibilidade nas classificações, reduzindo ambiguidades e facilitando a integração com os demais módulos do sistema.

Além disso, a flexibilidade arquitetural implementada, com abstração da camada de serviço de LLMs, permitiu alternância entre diferentes provedores (*Groq*, *OpenAI*, *Gemini*) sem modificações no código central, demonstrando escalabilidade e adaptabilidade da solução. A validação prática evidenciou que LLMs são capazes de interpretar

nuances da linguagem informal, gírias, emojis e contextos culturais presentes em comentários de *lives*, aspectos frequentemente ignorados por técnicas convencionais de PLN.

RQ2: De que forma a integração entre LLMs e sistemas de monitoramento em tempo real pode contribuir para a geração de métricas precisas sobre o engajamento e a percepção do público?

RA2: A integração entre LLMs e o sistema de monitoramento em tempo real revelou-se fundamental para a geração de métricas precisas e acionáveis. O *Live Insights* implementa um ciclo contínuo de captura, classificação e persistência de dados que opera de forma automatizada durante toda a duração da transmissão. O serviço *Check Activity* monitora o estado das *lives* cadastradas, iniciando e encerrando o monitoramento de forma autônoma, sem intervenção manual.

A coleta periódica de comentários via *YouTube Data API v3*, associada ao processamento imediato por LLMs, possibilitou a geração de métricas em tempo real sobre a distribuição de reações, tipos de interação predominantes, identificação de picos de engajamento e detecção de comentários ofensivos ou preconceituosos. Essa abordagem supera sistemas reativos que dependem de análise posterior, permitindo intervenções tempestivas por parte de moderadores e criadores de conteúdo.

A persistência estruturada dos dados classificados no banco de dados *PostgreSQL*, com rastreamento temporal via JPA *Auditing*, garantiu integridade e rastreabilidade completa das interações. As métricas geradas refletem não apenas agregações quantitativas (percentuais de reação, contagens por tipo), mas também qualitativas, com exemplos representativos de cada categoria e informações sobre autores, facilitando ações de moderação e análise de perfis comportamentais.

A arquitetura desacoplada, baseada em princípios de DDD e padrões como *Repository* e *Service*, assegurou que a lógica de negócio permanecesse independente de detalhes de implementação, favorecendo manutenibilidade e expansibilidade. Testes em transmissões reais demonstraram que o sistema mantém estabilidade e precisão mesmo sob carga elevada, validando a escolha arquitetural e a estratégia de integração.

RQ3: Como a visualização das métricas de reação pode apoiar criadores de conteúdo e gestores na tomada de decisões baseada em dados?

RA3: A visualização das métricas de reação, implementada através de um *dashboard* interativo desenvolvido em *React.js*, provou ser elemento central para apoiar decisões estratégicas baseadas em dados. A interface apresenta de forma consolidada a distribuição de reações (gráficos de barras e de pizza), evolução temporal do engajamento, categorização por tipo de interação e filtros personalizáveis que permitem análises segmentadas.

A disponibilização da URL do perfil do autor de cada comentário, especialmente

aqueles classificados como ofensivos ou preconceituosos, fornece aos responsáveis pela *live* meios imediatos para ação, seja através de denúncias na própria plataforma do *YouTube*, seja mediante adoção de medidas legais quando necessário. Essa funcionalidade atende diretamente à necessidade de ferramentas que promovam ambientes digitais mais seguros e respeitosos.

O relatório analítico atualizada a cada análise realizada na transmissão oferece visão panorâmica do desempenho da *live*, identificando padrões de engajamento, momentos críticos e oportunidades de melhoria. Criadores de conteúdo podem, por exemplo, correlacionar picos de reação negativo com trechos específicos da transmissão, ajustando estratégias futuras. Gestores de *marketing* podem avaliar a receptividade de campanhas, produtos ou mensagens, fundamentando decisões em evidências concretas extraídas do comportamento real da audiência.

A acessibilidade e responsividade da interface, aliadas à capacidade de processar e apresentar informações complexas de forma intuitiva, reduzem a barreira técnica para uso do sistema, democratizando o acesso a análises avançadas de reação. A combinação entre métricas quantitativas e exemplos qualitativos permite interpretações ricas e contextualizadas, superando limitações de *dashboards* puramente numéricos.

Portanto, o *Live Insights* demonstrou viabilidade técnica e potencial prático como ferramenta de inteligência de dados voltada ao monitoramento e análise de transmissões ao vivo. A integração bem-sucedida entre LLMs, sistemas de tempo real e interfaces de visualização confirma a hipótese de que tecnologias emergentes de PLN podem ser aplicadas de forma eficiente em contextos que demandam processamento rápido, precisão analítica e usabilidade.

Os resultados obtidos evidenciam que a análise automatizada de reações em *lives* não apenas é factível, mas também gera valor mensurável para diferentes perfis de usuários. As respostas às questões de pesquisa validam a proposta inicial e consolidam o sistema como contribuição relevante tanto para a comunidade acadêmica quanto para o mercado de produção de conteúdo digital.

Contudo, reconhece-se que limitações subsistem, especialmente quanto à dependência de APIs externas, custos associados ao uso intensivo de LLMs e desafios relacionados à escalabilidade em cenários de múltiplas transmissões simultâneas de altíssimo volume. Tais aspectos configuram oportunidades para aprimoramentos futuros, conforme detalhado na seção subsequente.

6.1 Trabalhos Futuros

A partir dos resultados obtidos e das limitações identificadas ao longo do desenvolvimento deste estudo, emergem diversas oportunidades para aprimoramentos e novas

investigações. Esta seção apresenta propostas de continuação que podem ampliar o alcance, a robustez e a aplicabilidade do *Live Insights*.

Expansão para Outras Plataformas: O sistema foi desenvolvido especificamente para transmissões do *YouTube*, porém a arquitetura modular permite adaptação para outras plataformas de *streaming*, como *Twitch*, *Facebook Live* e *Instagram Live*. Trabalhos futuros podem explorar a integração com APIs dessas plataformas, ampliando o escopo de aplicação e possibilitando análises comparativas entre diferentes canais de distribuição de conteúdo.

Implementação de LLMss Locais: Atualmente, o sistema depende de APIs externas de provedores de LLMs, o que implica custos operacionais e potenciais limitações de taxa de requisições. A implementação de modelos de linguagem executados localmente, como versões otimizadas do *LLaMA* ou *Mistral*, pode reduzir custos, aumentar privacidade dos dados e eliminar dependências externas, favorecendo a sustentabilidade da solução a longo prazo.

Análise de reação Multimodal: Transmissões ao vivo frequentemente incluem elementos visuais e auditivos relevantes para compreensão do contexto. Pesquisas futuras podem explorar a integração de análise de imagens (reconhecimento de expressões faciais, objetos, cenários) e áudio (tom de voz, entonação) com a análise textual de comentários, proporcionando visão mais holística do engajamento e das reações da audiência.

Sistema de Recomendações Inteligentes: Com base nas métricas coletadas e analisadas, é possível desenvolver um módulo de recomendações que sugira ajustes em tempo real aos criadores de conteúdo, como mudança de assunto, intensificação de interação com o público ou pausas estratégicas, visando maximizar engajamento positivo e minimizar reações negativas.

Detecção Avançada de *Toxicidade* e Moderação Automática: Embora o sistema identifique comentários ofensivos ou preconceituosos, trabalhos futuros podem aprofundar essa funcionalidade, implementando níveis granulares de toxicidade, detecção de *hate speech*, *cyberbullying* e outros comportamentos nocivos. Integração com ferramentas de moderação automática pode permitir ações proativas, como ocultação temporária de comentários suspeitos ou alertas imediatos para moderadores.

Análise Longitudinal e Tendências Temporais: O armazenamento persistente de dados históricos possibilita estudos longitudinais sobre evolução do reação ao longo de múltiplas transmissões, identificação de tendências sazonais, correlações entre estratégias de conteúdo e engajamento, e construção de perfis comportamentais de audiências. Ferramentas de *data mining* e aprendizado de máquina podem ser aplicadas para descoberta de padrões não evidentes.

Otimização de Desempenho e Escalabilidade: Para cenários de múltiplas transmissões simultâneas com volumes extremamente elevados de comentários, investiga-

ções sobre otimização de processamento paralelo, uso de filas de mensagens (como *RabbitMQ* ou *Kafka*), *caching* inteligente e arquiteturas de microsserviços podem aprimorar escalabilidade horizontal e reduzir latência.

Validação em Estudos de Caso Diversificados: Embora testes tenham sido conduzidos em transmissões reais, validações mais extensas em diferentes contextos, como *lives* educacionais, eventos corporativos, *gameplays*, *shows* musicais e debates políticos, podem revelar especificidades e demandas particulares de cada nicho, orientando personalizações e melhorias direcionadas.

Interface de Administração Avançada: Desenvolvimento de funcionalidades administrativas como gestão de usuários com diferentes níveis de acesso, configuração dinâmica de *prompts* para LLMs, personalização de categorias de classificação, agendamento de relatórios periódicos e integração com sistemas de *Customer Relationship Management* (CRM) ou ferramentas de *marketing analytics*.

***Compliance* e Privacidade de Dados:** Considerando regulamentações como Lei Geral de Proteção de Dados (LGPD) e *General Data Protection Regulation* (GDPR), trabalhos futuros devem aprofundar aspectos relacionados à anonimização de dados, consentimento de usuários, políticas de retenção e direito ao esquecimento, garantindo conformidade legal e ética no tratamento de informações sensíveis.

Importante destacar que os trabalhos futuros aqui sugeridos visam não apenas evoluir o sistema, mas também incentivar pesquisas que explorem novas abordagens, integrem tecnologias emergentes e respondam a desafios ainda não totalmente resolvidos no contexto das transmissões ao vivo. A continuidade deste estudo pode contribuir significativamente para consolidação de práticas mais inteligentes, seguras e eficazes no gerenciamento de interações digitais em tempo real.

Uso de computação em nuvem (AWS, Azure, entre outras): A implantação do *Live Insights* em provedores de nuvem pode ampliar a disponibilidade e facilitar escalabilidade elástica. Em trabalhos futuros, a solução pode ser containerizada e executada em serviços gerenciados (por exemplo, ambientes com *auto scaling*), além de incorporar observabilidade, balanceamento de carga e práticas de integração contínua e entrega/implantação contínua (CI/CD) para atualizações contínuas.

Processamento em larga escala de comentários (alto volume e múltiplas *lives*): Para cenários de grande volume de dados e múltiplas transmissões simultâneas, trabalhos futuros podem evoluir o pipeline para arquiteturas de processamento distribuído e orientadas a eventos, com filas/*streaming*, paralelização e persistência escalável. Isso permitiria aumentar *throughput*, reduzir latência e manter a robustez do sistema em situações de pico.

Referências

AHMAD, M.; BATYRSHIN, I.; SIDOROV, G. Sentiment analysis using a large language model–based approach to detect opioids mixed with other substances via social media: Method development and validation. *JMIR infodemiology*, JMIR Publications Toronto, Canada, v. 5, p. e70525, 2025. 25

ANDERSON, D. J. *Kanban: successful evolutionary change for your technology business*. [S.l.]: Blue hole press, 2010. 31

ASGHAR, M. Z. et al. Sentiment analysis on youtube: A brief survey. *arXiv preprint arXiv:1511.09142*, 2015. 24

BOECHAT, G. C. et al. Uma investigação sobre análise de sentimentos e categorização de issues reabertas do github. Universidade Federal da Bahia, 2024. 25

BROWN, S. *Software Architecture for Developers: Volume 2 – Visualising Software Architecture with the C4 Model*. [S.l.]: CreateSpace Independent Publishing Platform, 2018. 48

CHAN, W. N.; THEIN, T. A comparative study of machine learning techniques for real-time multi-tier sentiment analysis. In: IEEE. *2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII)*. [S.l.], 2018. p. 90–93. 14

CHOWDHARY, C. L. Artificial intelligence and machine learning: Recent advances and applications (part 1). *Recent Patents on Computer Science*, Bentham Science Publishers, v. 12, n. 1, p. 3–4, 2019. 19

DUQUE-PEREIRA, I. da S.; MOURA, S. A. de et al. Compreendendo a inteligência artificial generativa na perspectiva da língua. SciELO Preprints, 2023. 20

EVANS, E. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston: Addison-Wesley Professional, 2016. ISBN 978-0321125217. 52, 53

GACKENHEIMER, C. *Introduction to React*. [S.l.]: Apress, 2015. 24

GROQ. *Índice TIOBE*. 2025. Acesso em: 01 nov. 2025, às 12:00. Disponível em: <https://console.groq.com/settings/limits>. 41

HAN, S. et al. A review of large language models: Fundamental architectures, key technological evolutions, interdisciplinary technologies integration, optimization and compression techniques, applications, and challenges. *Electronics*, MDPI, v. 13, n. 24, p. 5040, 2024. 14, 20, 21, 22

HJARVARD, S. Midiatização: conceituando a mudança social e cultural. *Matrizes*, Universidade de São Paulo, v. 8, n. 1, p. 21–44, 2014. 19

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2. ed. Upper Saddle River, NJ: Prentice Hall, 2009. ISBN 978-0131873216. 21

KAPLAN, A.; HAENLEIN, M. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business horizons*, Elsevier, v. 62, n. 1, p. 15–25, 2019. 19

LEITE, F. P. A. O exercício da liberdade de expressão nas redes sociais: e o marco civil da internet. *Revista de Direito Brasileira*, v. 13, n. 6, p. 150–166, 2016. 17

MANCHANA, R. Java virtual machine (jvm): Architecture, goals, and tuning options. *International Journal of Scientific Research and Engineering Trends*, v. 1, n. 3, p. 42–52, 2015. 23

MARTINEZ, D. D.; REMEGIO, A.; LINCOPINIS, D. R. A review on java programming language. *ResearchGate. URL: https://www. researchgate. net/publication/371166744_A_Review_on_Java_Programming_Lan-guage (accessed 04.02. 2024)*, 2023. 22

MINAEE, S. et al. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024. 20, 21, 22

MOREIRA, E. A. S. M. Redes sociais virtuais e saúde mental. 2022. 19

POSTGRE. *Postgre*. 2025. Acesso em: 10 set. 2025, às 18:30. Disponível em: <https://portgresql.org>. 23

RAJALA, O. Impact of react component libraries on developer experience-an empirical study on component libraries' styling approaches. 2024. 24

RECUERO, R. Redes sociais na internet, difusão de informação e jornalismo: elementos para discussão. *Metamorfoses jornalísticas*, v. 2, p. 1–269, 2009. 18

RODRIGUES, G. C. F. S.; BRENNAND, E. G. de G. Youtube: rede de interação e formação de colégios invisíveis. *Creativity and Educational Innovation Review*, n. 6, p. 38–52, 2022. 17

SILVA, J. B. d.; ANASTáCIO, F. A. d. M. Método kanban como ferramenta de controle de gestão. *ID on line. Revista de psicologia*, v. 13, n. 43, p. 1018–1027, dez. 2018. Disponível em: <https://idonline.emnuvens.com.br/id/article/view/1575>. 31

SILVA, N. K. A. d. et al. A comunicação digital e o entretenimento no streaming ao vivo: uma análise a partir do canal cazétv. Universidade Federal do Pampa, 2023. 17

SOBREIRA, V. Um panorama da história da inteligência artificial e suas aplicações na pesquisa histórica. *Varia Historia*, SciELO Brasil, v. 41, p. e25035, 2025. 20

TIOBE. *Índice TIOBE*. 2025. Acesso em: 01 nov. 2025, às 17:03. Disponível em: <https://www.tiobe.com/tiobe-index/>. 23

TRUSKOWSKI, W.; KLEWEK, R.; SKUBLEWSKA-PASZKOWSKA, M. Comparison of mysql, mssql, postgresql, oracle databases performance, including virtualization. *Journal of Computer Sciences Institute*, v. 16, p. 279–284, 2020. 24

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2017. v. 30. 22

XU, Q. A.; CHANG, V.; JAYNE, C. A systematic review of social media-based sentiment analysis: Emerging trends and challenges. *Decision Analytics Journal*, Elsevier, v. 3, p. 100073, 2022. 18