INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PICOS
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ARTUR MARTINS SILVA

UMA FERRAMENTA BASEADA EM RETRIEVAL-AUGMENTED GENERATION PARA EXTRAÇÃO DE INSIGHTS EM COMENTÁRIOS DE VÍDEOS DO YOUTUBE

PICOS, PIAUÍ
2026

ARTUR MARTINS SILVA

UMA FERRAMENTA BASEADA EM RETRIEVAL-AUGMENTED GENERATION PARA EXTRAÇÃO DE INSIGHTS EM COMENTÁRIOS DE VÍDEOS DO YOUTUBE

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador: Prof. Dr. Rogério Figueredo de Sousa

PICOS, PIAUÍ
2026

ARTUR MARTINS SILVA

UMA FERRAMENTA BASEADA EM RETRIEVAL-AUGMENTED GENERATION PARA EXTRAÇÃO DE INSIGHTS EM COMENTÁRIOS DE VÍDEOS DO YOUTUBE

Trabalho de Conclusão de Curso (Relatório Técnico de Software) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Aprovado em 12/08/2026.

BANCA EXAMINADORA:

Prof. Dr. Rogério Figueredo de Sousa (Orientador)
Instituto Federal do Piauí (IFPI)

Prof. Me. Anthony Irlan Marques Luz
Instituto Federal do Piauí (IFPI)

Prof. Me. Manoel Messias Pereira Medeiros
Instituto Federal do Piauí (IFPI)

PICOS, PIAUÍ
2026

Dedico este trabalho aos meus pais, por todo amor, apoio e incentivo ao longo desta caminhada; ao meu irmão, pela parceria e por estar sempre ao meu lado; e a toda a minha família, pelo apoio, pela confiança e por contribuírem de maneira significativa para a realização desta conquista.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por ter me concedido força, sabedoria e serenidade ao longo desta caminhada, especialmente nos momentos em que as dificuldades pareciam maiores do que eu poderia enfrentar.

À minha mãe, Marciela, e ao meu pai, Avelino, por todo o amor, cuidado, apoio e incentivo que sempre me ofereceram. Por acreditarem em mim, por me darem forças nos momentos difíceis e por estarem presentes em cada etapa da minha formação. Grande parte do que alcancei até aqui também é resultado de tudo o que fizeram e continuam fazendo por mim.

Ao meu irmão, Murillo, pelo apoio, pela motivação e, sobretudo, pelas brincadeiras e momentos de descontração que tornaram essa trajetória mais leve.

Às minhas tias, Maria do Rosário e Waldênia, pelo acolhimento, cuidado e incentivo ao longo dessa trajetória. Agradeço especialmente por todo o apoio aos meus estudos e por terem contribuído de forma tão importante para a minha formação e para o caminho que me trouxe até o IFPI.

Às minhas avós, Edite e Valdete, pelo carinho, pelo cuidado e pelo apoio que sempre me ofereceram.

De maneira especial, aos meus amigos Kauê e Luis, que estão comigo desde o ensino médio e fizeram parte de tantos momentos importantes da minha vida. Sou muito grato por ter compartilhado com vocês essa caminhada, entre desafios, conquistas e tantos bons momentos.

À Ana Lívia, um presente especial que Deus colocou em meu caminho. Obrigado por todo o carinho, companheirismo e por estar ao meu lado.

Aos meus amigos do grupo Mantinhas, Christian, Rafael, João Pedro, Letícia, Emanuelle, Dário, Gabriel, Àlex e Thávyne, pela amizade, pelas conversas, pelos momentos compartilhados e pela leveza que trouxeram aos dias mais difíceis.

Aos amigos que o curso me proporcionou, Matheus, Vinícius, João Guilherme, Mateus Vital, Ezequiel, Caique, Juan, Misael, Gleison, Mourato, Casemiro, Widiney e Kássio, agradeço pela convivência, pelas trocas de conhecimento, pelas ajudas ao longo das disciplinas e por todos os momentos que fizeram parte desses anos de graduação.

Ao meu orientador, Prof. Dr. Rogério Figueredo de Sousa, pela orientação atenta, pelas críticas construtivas, pela disponibilidade e pelas contribuições fundamentais ao desenvolvimento deste trabalho.

Aos membros da banca examinadora, Prof. Me. Anthony Irlan Marques Luz e Prof. Me. Manoel Messias Pereira Medeiros, pela disponibilidade em avaliar este trabalho e pelas contribuições oferecidas para o seu aprimoramento.

“O coração do homem planeja o seu caminho,
mas o Senhor lhe dirige os passos.”
Provérbios 16:9

RESUMO

Este trabalho apresenta o desenvolvimento do CommentLens, uma extensão de navegador que emprega a arquitetura *Retrieval-Augmented Generation* (RAG) para auxiliar o usuário na compreensão das opiniões expressas nos comentários de vídeos do YouTube. O problema abordado é a sobrecarga informacional decorrente da quantidade de comentários não estruturados associados a vídeos populares, cuja leitura manual se torna pouco viável em grandes volumes. Como solução, a ferramenta coleta os comentários públicos de um vídeo por meio da API oficial da plataforma, ranqueia os comentários coletados de acordo com a estratégia de recuperação empregada e utiliza os itens selecionados como contexto para que um modelo de linguagem de grande porte gere uma resposta concisa, acompanhada dos comentários indicados como suas fontes. O sistema implementa duas estratégias de recuperação (uma léxica e outra semântica, baseada em *embeddings* e similaridade de cosseno) e as executa de forma pareada sobre a mesma entrada, registrando as interações em base relacional para posterior comparação. A comunicação com os serviços externos é intermediada por um *backend* que preserva as credenciais de acesso. Como resultado, obteve-se uma ferramenta funcional, com testes automatizados sobre a lógica central do *backend*, além da infraestrutura necessária à análise comparativa posterior entre as estratégias de recuperação. A avaliação empírica da qualidade das respostas constitui trabalho futuro.

Palavras-chave: Geração Aumentada por Recuperação; Modelos de Linguagem de Grande Porte; Análise de comentários; YouTube; Extensão de navegador.

ABSTRACT

This work presents the development of CommentLens, a browser extension that employs the Retrieval-Augmented Generation (RAG) architecture to help users understand the opinions expressed in the comments of YouTube videos. The problem addressed is the information overload caused by the number of unstructured comments associated with popular videos, whose manual reading becomes hardly feasible at large volumes. As a solution, the tool collects a video's public comments through the platform's official API, ranks the collected comments according to the retrieval strategy employed, and uses the selected items as context for a large language model to generate a concise answer, accompanied by the comments indicated as its sources. The system implements two retrieval strategies (a lexical one and a semantic one, based on embeddings and cosine similarity) and runs them in a paired fashion over the same input, recording the interactions in a relational database for later comparison. Communication with external services is mediated by a backend that protects the access credentials. As a result, a functional tool was obtained, with automated tests covering the core logic of the backend, along with the infrastructure required for a later comparative analysis between the retrieval strategies. The empirical evaluation of answer quality constitutes future work.

Keywords: Retrieval-Augmented Generation; Large Language Models; Comment analysis; YouTube; Browser extension.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IFPI	Instituto Federal de Educação, Ciência e Tecnologia do Piauí
LLM	*Large Language Model*
MMTEB	*Massive Multilingual Text Embedding Benchmark*
ORM	*Object-Relational Mapping*
PLN	Processamento de Linguagem Natural
QA	*Question Answering*
RAG	*Retrieval-Augmented Generation*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

O ecossistema digital contemporâneo é marcado pelo crescimento acelerado de conteúdo gerado por usuários, que consolidou plataformas como o YouTube em grandes repositórios de informação, opinião e influência. Os comentários associados aos vídeos constituem um volume expressivo de dados opinativos em formato textual, cuja análise automatizada é estratégica para compreender percepções, avaliações e atitudes do público nos mais diversos domínios (Musleh *et al.*, 2023). Entretanto, a própria diversidade linguística e a informalidade típicas desses comentários tornam sua exploração manual pouco viável, sobretudo em vídeos de grande engajamento, nos quais centenas ou milhares de mensagens se acumulam de forma redundante, contraditória ou ambígua (Pokharel; Bhatta, 2021).

Essa sobrecarga informacional é particularmente evidente em vídeos de *review* de produtos, em que o espectador frequentemente recorre aos comentários para confirmar impressões e decidir uma compra, mas se depara com um córpus extenso e não estruturado. O problema, contudo, não se restringe a esse gênero: qualquer vídeo popular, como tutoriais, notícias, análises e conteúdos educacionais, reúne opiniões dispersas cuja leitura integral é impraticável para o usuário comum. Revisões sistemáticas sobre análise de sentimentos em comentários do YouTube destacam justamente que o grande volume de dados, aliado ao ruído textual e à ausência de padronização linguística, impõe desafios metodológicos significativos e reforça a necessidade de abordagens automatizadas robustas (Singh; Thomas, 2025).

A maior parte das soluções existentes concentra-se em tarefas tradicionais de classificação e análise de sentimentos, que enfrentam limitações persistentes no tratamento de textos ruidosos e não estruturados (Singh; Thomas, 2025; Xin *et al.*, 2025). Nesse cenário, a arquitetura *Retrieval-Augmented Generation* (RAG) surge como um paradigma promissor: ao combinar mecanismos de recuperação de informação com modelos de linguagem de grande porte (LLMs), fundamenta as respostas geradas em evidências efetivamente recuperadas do córpus, reduzindo inconsistências factuais e aumentando a rastreabilidade, buscando mitigar problemas como alucinação e desatualização de conhecimento inerentes ao uso isolado de LLMs (Gao *et al.*, 2023; Fan *et al.*, 2024). Trabalhos como o modelo HIRO demonstram empiricamente que a integração entre recuperação estruturada e geração textual produz sínteses mais fiéis e fundamentadas do que abordagens puramente generativas (Hosking *et al.*, 2024).

Apesar desses avanços, a aplicação da arquitetura RAG especificamente à análise de comentários do YouTube, com foco em interações de *Question Answering* (QA) em linguagem natural, ainda é pouco explorada. Diante dessa lacuna, este relatório apresenta o desenvolvimento de uma extensão de navegador que implementa

uma arquitetura RAG para extração de *insights* a partir dos comentários de vídeos do YouTube. A ferramenta coleta os comentários públicos de um vídeo, ranqueia o conjunto coletado segundo a estratégia de recuperação empregada e utiliza os comentários selecionados como contexto para um LLM gerar uma resposta concisa. A interface também apresenta os comentários indicados como fontes da resposta, permitindo ao usuário consultar as evidências utilizadas durante o processamento. Com isso, a ferramenta busca reduzir o esforço necessário para consultar grandes volumes de opiniões, permitindo que o usuário obtenha respostas rápidas sem precisar ler manualmente todo o conteúdo.

Como parte do escopo, este trabalho não se limita a implementar uma única forma de recuperação, mas adota duas estratégias distintas: uma léxica, baseada na correspondência de termos entre a pergunta e os comentários, e outra semântica, baseada na proximidade de significado calculada a partir de representações vetoriais. Ambas são executadas sobre o mesmo conjunto de comentários, e as interações resultantes são registradas de forma vinculada, constituindo a base para uma análise comparativa posterior entre as duas abordagens no contexto específico dos comentários do YouTube.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver uma ferramenta baseada em RAG, na forma de uma extensão de navegador, para extração de *insights* em comentários de vídeos do YouTube, integrando estratégias de recuperação léxica e semântica de informação, de modo a auxiliar o usuário na compreensão das opiniões expressas nos comentários por meio de perguntas em linguagem natural.

1.1.2 Objetivos Específicos

a) Coletar os comentários públicos de vídeos do YouTube por meio da API oficial da plataforma;

b) Implementar um *backend* intermediário que gerencie o acesso aos serviços externos e proteja as credenciais;

c) Implementar duas estratégias de recuperação de comentários relevantes a uma pergunta: uma léxica e outra semântica, esta baseada em *embeddings* e similaridade de cosseno;

d) Integrar os componentes de recuperação e geração, de modo a responder, em linguagem natural, a perguntas sobre o conteúdo dos comentários;

e) Registrar as interações de forma pareada e estruturada, constituindo a infraestrutura para a comparação entre as duas estratégias;

f) Desenvolver uma interface que permita formular perguntas e visualizar as respostas junto aos comentários indicados como suas fontes;

g) Verificar o funcionamento dos componentes do *backend* por meio de testes unitários.

2 TECNOLOGIAS ENVOLVIDAS

A solução é composta por dois módulos independentes: uma extensão de navegador, executada no lado do cliente, e um *backend serverless*, responsável pela lógica de recuperação de informação e geração de respostas. As tecnologias empregadas em cada módulo, bem como as justificativas para sua escolha, são descritas a seguir.

2.1 MODELOS DE LINGUAGEM DE GRANDE PORTE

Os LLMs são modelos computacionais treinados sobre grandes volumes de dados textuais com o objetivo de aprender padrões e relações presentes na linguagem. A partir desse treinamento, esses modelos podem realizar diferentes tarefas de Processamento de Linguagem Natural (PLN), como geração de texto, tradução, sumarização e resposta a perguntas. Modelos de grande escala também demonstraram capacidade de realizar diferentes tarefas a partir de instruções ou exemplos fornecidos no próprio contexto de entrada, sem a necessidade de treinamento específico para cada interação (Brown *et al.*, 2020).

Grande parte dos LLMs atuais está relacionada à arquitetura Transformer, introduzida por Vaswani *et al.* (2017). Essa arquitetura utiliza mecanismos de atenção para estabelecer relações entre diferentes elementos de uma sequência, permitindo que o modelo considere o contexto no processamento do texto. Diferentemente das arquiteturas recorrentes predominantes anteriormente, o Transformer foi projetado com base em mecanismos de atenção, favorecendo o processamento paralelo das sequências durante o treinamento (Vaswani *et al.*, 2017).

Apesar de sua capacidade de geração de linguagem, o conhecimento utilizado por um LLM encontra-se associado aos parâmetros aprendidos durante seu treinamento. Isso impõe limitações quando a tarefa exige acesso a informações externas específicas ou atualizadas. Além disso, modelos de linguagem podem produzir informações não sustentadas por evidências disponíveis, fenômeno frequentemente associado às chamadas alucinações. Essas limitações são particularmente relevantes em aplicações que exigem respostas fundamentadas em um conjunto específico de informações e constituem uma das motivações para a utilização de mecanismos de recuperação em conjunto com modelos generativos (Gao *et al.*, 2023).

2.2 GERAÇÃO AUMENTADA POR RECUPERAÇÃO

A arquitetura RAG combina mecanismos de recuperação de informação com modelos generativos. Em vez de produzir uma resposta utilizando apenas o conhe-

cimento representado nos parâmetros do modelo, uma aplicação baseada em RAG recupera informações de uma fonte externa e as fornece como contexto para a etapa de geração. A abordagem foi apresentada por Lewis *et al.* (2020) combinando a memória paramétrica de um modelo de linguagem com uma memória não paramétrica externa acessada por um mecanismo de recuperação.

De forma geral, um fluxo RAG pode ser compreendido a partir de três elementos: recuperação, aumento do contexto e geração. Diante de uma consulta, o componente de recuperação busca na fonte de dados os conteúdos considerados pertinentes; os conteúdos recuperados são então incorporados ao contexto fornecido ao modelo de linguagem; por fim, o modelo utiliza esse contexto na geração da resposta. Dessa maneira, o conhecimento externo pode ser consultado durante a execução da aplicação sem precisar ser incorporado aos parâmetros do modelo (Lewis *et al.*, 2020; Gao *et al.*, 2023).

A utilização de informações recuperadas permite fundamentar a geração em uma fonte de dados específica e pode contribuir para reduzir problemas associados ao conhecimento exclusivamente paramétrico dos LLMs, como informações desatualizadas e respostas sem suporte em evidências externas. Entretanto, o emprego de RAG não elimina automaticamente esses problemas, uma vez que o resultado também depende da qualidade da recuperação e da forma como o modelo utiliza o contexto fornecido (Gao *et al.*, 2023).

No CommentLens, a fonte externa de informação é constituída pelos comentários coletados do vídeo do YouTube. A pergunta formulada pelo usuário é utilizada para recuperar um subconjunto desses comentários, que é posteriormente fornecido como contexto ao modelo de linguagem responsável pela geração da resposta. Dessa forma, a arquitetura RAG estabelece a ligação entre a etapa de recuperação dos comentários e a etapa generativa da ferramenta.

2.3 MODELOS DE LINGUAGEM E RECUPERAÇÃO DE INFORMAÇÃO

A geração das respostas em linguagem natural é realizada por meio da API da Groq, utilizada como serviço de inferência para acesso ao modelo de linguagem (Groq, 2026a) e disponível, durante o desenvolvimento, em modalidade gratuita compatível com o volume de requisições deste trabalho. O modelo primário utilizado para a geração é o Llama 3.3 70B Instruct, desenvolvido pela Meta e acessado pelo identificador *llama-3.3-70b-versatile*. Trata-se de um modelo de linguagem com 70 bilhões de parâmetros, ajustado para seguimento de instruções e com suporte multilíngue que inclui o português (Meta, 2024). Em caso de indisponibilidade por limite de requisições, o sistema recorre automaticamente ao Qwen3-32B, da família Qwen, modelo denso com 32 bilhões de parâmetros (Yang *et al.*, 2025), disponibilizado pela mesma plataforma sob o identificador *qwen/qwen3-32b* (Groq, 2026b). Para a recuperação semântica, a

pergunta e os comentários são convertidos em representações vetoriais (*embeddings*) por meio do modelo *gemini-embedding-001*, do Google, e a relevância é obtida pela similaridade de cosseno entre esses vetores. Registra-se que a recuperação semântica opera por busca exaustiva em memória (os vetores são gerados sob demanda a cada requisição e descartados em seguida), não havendo banco de dados vetorial ou índice persistente.

O modelo *gemini-embedding-001* foi adotado por ser um modelo de *embeddings* de texto aplicável, entre outras tarefas, à busca semântica e à recuperação de documentos, atividades compatíveis com o mecanismo de recuperação implementado neste trabalho. O modelo permite ainda especificar o tipo de tarefa associado a cada entrada, possibilitando distinguir a pergunta (*RETRIEVAL_QUERY*) dos textos candidatos à recuperação (*RETRIEVAL_DOCUMENT*). Segundo a documentação do provedor, essas configurações são destinadas, respectivamente, às consultas e aos documentos em cenários de recuperação de informação (Google, 2026c).

A escolha também considera a capacidade multilíngue do modelo. Lee *et al.* (2025) avaliaram o Gemini Embedding no *Massive Multilingual Text Embedding Benchmark* (MMTEB), que reúne mais de 100 tarefas em mais de 250 idiomas, e reportaram desempenho de estado da arte nos *benchmarks* multilíngues avaliados. Esses resultados fornecem evidências da capacidade multilíngue do modelo e sustentam sua aplicação em um sistema que processa conteúdo textual em português. Entretanto, não foi realizada neste trabalho uma avaliação específica de seu desempenho para o português brasileiro. Dessa forma, sua adoção não pressupõe superioridade nesse idioma em relação a outros modelos de *embeddings*, aspecto que pode ser investigado em avaliações futuras.

2.4 EXTENSÃO DE NAVEGADOR

Extensões de navegador são aplicações que acrescentam funcionalidades ao navegador e podem utilizar APIs específicas para interagir com recursos relacionados à navegação. No Chrome, seu comportamento e suas permissões são definidos por um arquivo de manifesto, enquanto as APIs de extensões disponibilizam recursos para integração com abas, armazenamento e outros elementos do navegador (Google, 2026a).

A escolha de uma extensão de navegador para o CommentLens decorre da forma de interação proposta pela ferramenta. Como a análise ocorre sobre comentários associados ao vídeo que o usuário está visualizando no YouTube, a extensão permite disponibilizar a funcionalidade no próprio contexto de navegação, sem exigir que o usuário acesse uma aplicação separada e informe manualmente o endereço ou o identificador do vídeo. A ferramenta identifica o vídeo em exibição, coleta seus comentários mediante ação do usuário e mantém a interface de consulta acessível

pelo navegador. A escolha desse formato também possibilitou distribuir a ferramenta por meio da Chrome Web Store, permitindo sua instalação diretamente pelo navegador e mantendo a interação integrada ao contexto de uso do YouTube.

A camada cliente foi desenvolvida com HTML, CSS e JavaScript, utilizando o Manifest V3 adotado pelo Chrome. No CommentLens, a arquitetura da extensão é organizada em três contextos compatíveis com esse modelo: um *content script*, que detecta a página de vídeo e extrai seu identificador, inclusive após a navegação interna do YouTube; um *service worker*, único componente que realiza comunicação de rede; e um *popup*, que concentra a interface com o usuário. A documentação do Chrome descreve esses mecanismos como componentes disponíveis para a construção de extensões, incluindo *content scripts* para execução no contexto das páginas e *extension service workers* para o processamento de eventos em segundo plano (Google, 2026a). A extensão é responsável pela interação com o usuário, identificação do vídeo, armazenamento local dos comentários coletados e comunicação com o *backend*. As operações que dependem de credenciais de serviços externos permanecem no servidor, evitando que essas chaves sejam incorporadas ao código distribuído ao navegador.

2.5 BACKEND E COMPUTAÇÃO SERVERLESS

O *backend* corresponde à camada responsável pelo processamento que não é executado diretamente na extensão. No CommentLens, essa camada intermedeia o acesso aos serviços externos utilizados pela aplicação, incluindo a YouTube Data API v3, o serviço de *embeddings*, o modelo de linguagem e o banco de dados. Essa separação também permite manter as credenciais desses serviços fora do código executado no navegador.

A implementação utiliza TypeScript sobre Node.js e foi estruturada segundo um modelo de computação *serverless*. Nesse modelo, as funções responsáveis pelo processamento são executadas pela infraestrutura da plataforma sob demanda, sem que o desenvolvedor precise administrar diretamente um servidor permanentemente ativo. Essa abordagem é compatível com o comportamento do CommentLens, cujo *backend* é acionado em operações específicas, como a coleta dos comentários e o processamento das perguntas. O uso de TypeScript em modo estrito amplia as verificações estáticas de tipos realizadas durante o desenvolvimento, permitindo que determinados problemas sejam identificados antes da execução do programa (Microsoft, 2026).

A Vercel foi adotada como plataforma de implantação por oferecer suporte à execução das funções *serverless* utilizadas pelo *backend*. A documentação da Vercel descreve suas *Functions* como um mecanismo para executar código no lado do servidor sem gerenciamento direto da infraestrutura de servidores (Vercel, 2026). A escolha permite manter a camada de processamento separada da extensão e disponibilizar

endpoints HTTP para a comunicação entre as duas partes da aplicação. No projeto, essa arquitetura também possibilita centralizar no *backend* as chamadas que exigem credenciais, evitando sua exposição no cliente.

2.6 PERSISTÊNCIA DE DADOS

A persistência das interações do CommentLens utiliza um banco de dados relacional PostgreSQL hospedado no Neon. O armazenamento é necessário para manter os resultados produzidos pelas duas estratégias de recuperação associados a uma mesma interação, juntamente com informações utilizadas em análises posteriores, como o método empregado, o modelo de linguagem utilizado e os comentários selecionados e indicados como fontes.

O Neon fornece PostgreSQL em uma infraestrutura voltada a ambientes *serverless* e disponibiliza um *driver* próprio para comunicação a partir desse tipo de aplicação (Neon, 2026). Sua adoção é compatível com a arquitetura do *backend* do CommentLens, baseada em funções de curta duração, e permite utilizar um banco relacional PostgreSQL sem manter um servidor de banco de dados administrado diretamente pelo projeto. Essa característica, somada à disponibilidade de um plano compatível com o volume previsto para este trabalho, motivou sua escolha.

O acesso ao banco é realizado por meio do Drizzle ORM, uma biblioteca para TypeScript que permite representar o esquema relacional no código e construir operações de banco com suporte à tipagem (Drizzle Team, 2026). No projeto, o Drizzle também é utilizado para definir o esquema e gerar as migrações responsáveis por versionar alterações na estrutura do banco. Essa escolha mantém a definição das entidades próxima ao código TypeScript do *backend* e reduz a necessidade de manter separadamente representações distintas do esquema da aplicação.

2.7 TESTES AUTOMATIZADOS

Testes automatizados permitem executar de forma repetível verificações sobre o comportamento esperado de componentes do software. Diferentemente de uma verificação manual realizada apenas em um momento específico, os casos automatizados podem ser novamente executados após alterações no código, auxiliando na identificação de regressões e na verificação de comportamentos previamente definidos.

No *backend* do CommentLens, os testes automatizados foram implementados com o Vitest. O *framework* oferece suporte direto a TypeScript e disponibiliza recursos necessários a testes unitários, como asserções e criação de *mocks* (Vitest, 2026). O uso de *mocks* é particularmente útil neste projeto por permitir testar componentes que dependem de serviços externos sem realizar chamadas reais às APIs durante cada execução da suíte.

A escolha do Vitest considerou sua integração com o ecossistema TypeScript utilizado pelo *backend* e sua execução local simplificada. A descrição dos componentes efetivamente cobertos, da quantidade de casos implementados e dos resultados da suíte é apresentada na seção 4.2.

3 MODELAGEM DO PROJETO

3.1 LEVANTAMENTO DE REQUISITOS

Os requisitos do sistema foram definidos a partir dos objetivos e do escopo estabelecidos para o trabalho, considerando as funcionalidades necessárias para atender à proposta do CommentLens. Essa definição ocorreu antes da implementação e foi discutida com o orientador durante o desenvolvimento do projeto. Não foi realizada uma etapa formal de elicitação com usuários externos, como entrevistas ou questionários. Ao longo do desenvolvimento, os requisitos foram revisados para manter sua correspondência com o escopo definido e com as funcionalidades efetivamente implementadas.

Foram elencados os seguintes requisitos funcionais apresentados no Quadro 1.

Além das funcionalidades descritas, foram identificados os requisitos não funcionais apresentados no Quadro 2, agrupados de acordo com a característica do sistema à qual se relacionam.

3.2 ARQUITETURA DO SISTEMA

O sistema organiza-se em três camadas, representadas na Figura 1. A camada de interface corresponde à extensão do navegador, responsável pela interação com o usuário, pela identificação do vídeo em exibição e pelo acionamento das operações de coleta e consulta. A camada de aplicação é o *backend serverless*, responsável por receber as requisições da extensão, realizar a comunicação com os serviços externos e orquestrar as etapas de recuperação e geração, mantendo as credenciais de acesso fora do cliente. A camada de inteligência reúne os mecanismos de recuperação de informação e o modelo de linguagem que produz a resposta.

Quadro 1 – Requisitos funcionais do CommentLens.

Código	Requisito	Descrição
RF01	Detecção do vídeo	Detectar automaticamente se a aba ativa exibe um vídeo do YouTube e extrair seu identificador, reagindo também à navegação interna da plataforma, que não recarrega a página.
RF02	Coleta de comentários	Coletar, sob comando do usuário, os comentários de topo do vídeo, ordenados por relevância, realizando a paginação automaticamente e informando quando o limite de coleta é atingido.
RF03	Armazenamento local	Armazenar localmente os comentários coletados e restaurá-los ao reabrir a interface no mesmo vídeo.
RF04	Formulação de perguntas	Permitir que o usuário formule perguntas em linguagem natural, em português, sobre os comentários, com limite de 500 caracteres, contador visual e atalho de teclado para envio.
RF05	Recuperação de comentários	Selecionar até 30 comentários como contexto para o modelo por meio das estratégias de recuperação léxica e semântica.
RF06	Execução pareada	Executar as duas estratégias sobre a mesma entrada e registrar seus resultados de forma vinculada, permitindo sua comparação posterior.
RF07	Geração de respostas	Gerar uma resposta em português, de duas a quatro frases, utilizando os comentários selecionados como contexto.
RF08	Apresentação das fontes	Identificar e apresentar ao usuário os comentários referenciados pelo modelo como fontes da resposta.
RF09	Modelo secundário	Recorrer automaticamente a um modelo de linguagem secundário quando o modelo primário estiver indisponível por limite de requisições.
RF10	Registro das interações	Registrar cada interação em base relacional para análise posterior.
RF11	Comunicação de estado	Comunicar ao usuário o estado das operações e eventuais falhas.

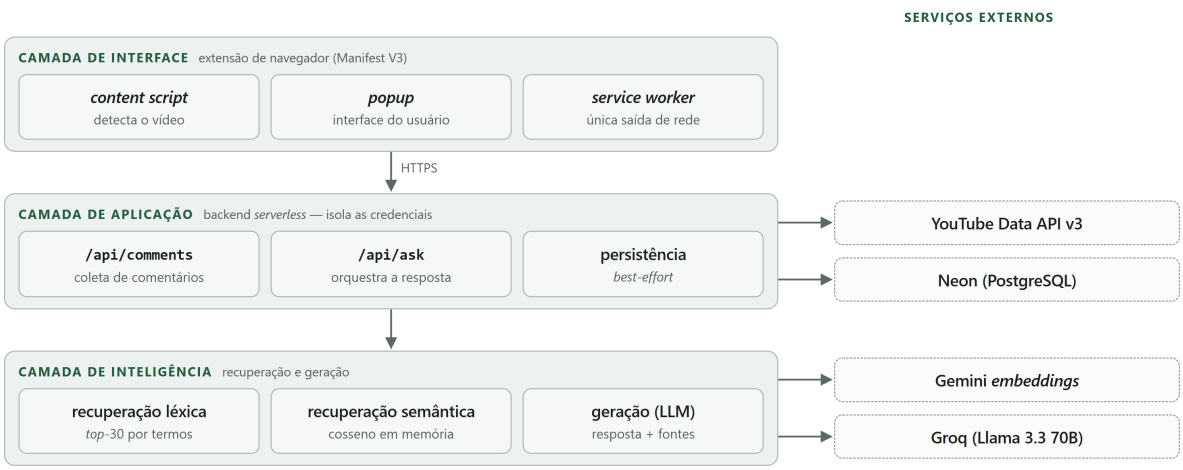
Fonte: Elaborado pelo autor (2026).

Quadro 2 – Requisitos não funcionais do CommentLens.

Código	Categoria	Requisito não funcional
RNF01	Segurança	Manter as chaves de acesso aos serviços externos exclusivamente no servidor, sem exposição no código da extensão.
RNF02	Segurança	Exibir o conteúdo dos comentários como texto puro, sem interpretá-lo como HTML.
RNF03	Privacidade	Não armazenar no banco de dados informações de identificação do autor dos comentários, como nome, identificador do canal ou imagem de perfil.
RNF04	Desempenho	Operar as funções do *backend* sob os limites de execução de 30 segundos e 256 MB definidos para o ambiente utilizado.
RNF05	Desempenho	Executar as duas estratégias de recuperação em paralelo.
RNF06	Desempenho	Processar a geração dos *embeddings* dos comentários em lotes.
RNF07	Uso de recursos	Limitar a 30 a quantidade de comentários utilizada como contexto em cada estratégia de recuperação e limitar a saída do modelo de linguagem a 1024 *tokens*.
RNF08	Custo	Utilizar os serviços externos dentro dos planos gratuitos adotados no desenvolvimento do projeto.
RNF09	Confiabilidade	Realizar a persistência das interações em regime *best-effort*, de modo que uma falha de gravação não impeça o envio da resposta ao usuário.
RNF10	Idioma	Disponibilizar a interface e as respostas geradas em português.

Fonte: Elaborado pelo autor (2026).

Figura 1 – Arquitetura em três camadas do CommentLens.

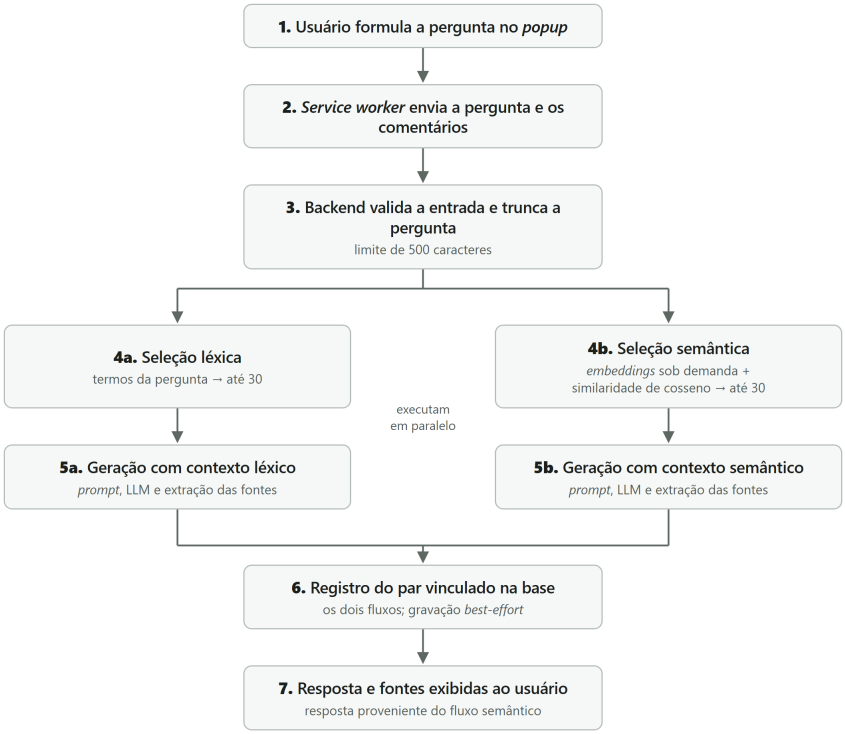


Fonte: Elaborado pelo autor (2026).

Nota: a recuperação semântica não utiliza banco de dados vetorial nem índice persistente: os *embeddings* são gerados sob demanda e descartados ao fim da requisição.

O funcionamento ocorre da forma sintetizada na Figura 2: ao acessar uma página de vídeo do YouTube, a extensão identifica automaticamente o vídeo em exibição por meio de seu identificador. Quando o usuário solicita a análise, a extensão requisita ao *backend* a coleta dos comentários públicos. Formulada a pergunta, a extensão a envia, junto ao conjunto de comentários, ao *backend*, que ranqueia o conjunto de comentários coletados segundo a estratégia de recuperação empregada, monta o *prompt*, submete-o ao modelo de linguagem, interpreta a resposta e os comentários por ela citados, registra a interação e devolve à extensão a resposta acompanhada de suas fontes, que são exibidas ao usuário.

Figura 2 – Fluxo de processamento de uma pergunta.



Fonte: Elaborado pelo autor (2026).

Nota: as vias 4a–5a e 4b–5b são executadas separadamente a partir da mesma pergunta e do mesmo conjunto de comentários coletados. Cada via gera sua própria resposta, sem combinação dos contextos em um único *prompt*. A interface apresenta ao usuário a resposta produzida pela via semântica, enquanto os resultados de ambas são registrados para comparação posterior.

A coleta é realizada por meio do recurso *commentThreads* da YouTube Data API v3, considerando os comentários principais das *threads* e utilizando o parâmetro *order=relevance*. Cada requisição solicita até 100 itens, e a aplicação percorre no máximo cinco páginas, resultando em um conjunto de até 500 comentários principais por vídeo. Quando a quantidade disponível é inferior a esse limite, todos os itens retor-

nados pela paginação são considerados. O limite de 500 foi definido como parâmetro de escopo da implementação e não decorre de uma avaliação experimental destinada a determinar uma quantidade ótima de comentários para a recuperação.

Após a coleta, o fluxo é dividido entre as duas estratégias de recuperação representadas pelas etapas 4a e 4b da Figura 2: léxica e semântica, respectivamente. Cada estratégia processa a mesma pergunta e o mesmo conjunto de comentários coletados e seleciona, de forma independente, até 30 itens para compor seu próprio contexto. Em seguida, as etapas 5a e 5b correspondem à geração de uma resposta a partir do contexto produzido por cada estratégia. Os dois contextos não são combinados em um único *prompt*. A resposta proveniente da via semântica é apresentada ao usuário, enquanto os resultados das duas vias são registrados de forma vinculada no banco de dados para possibilitar análise posterior.

Na estratégia de recuperação léxica, a pergunta do usuário é convertida para letras minúsculas e dividida em termos a partir dos espaços, sendo considerados apenas aqueles com mais de três caracteres. Para cada comentário, também convertido para letras minúsculas, é calculada uma pontuação correspondente ao número de ocorrências desses termos como *substrings* em seu texto. Os comentários são ordenados de forma decrescente pela pontuação obtida, utilizando-se o número de curtidas como critério de desempate. Caso a pergunta não produza termos elegíveis ou nenhum deles seja encontrado nos comentários, o método utiliza como alternativa os comentários com maior número de curtidas. A estratégia não realiza normalização de acentos, remoção de pontuação ou tratamento de palavras vazias.

No contexto deste trabalho, a abordagem léxica funciona como uma estratégia de referência baseada na ocorrência textual dos termos da pergunta, fornecendo um contraponto à recuperação semântica, que considera a proximidade de significado por meio de representações vetoriais. A implementação léxica não depende da geração de *embeddings* durante a etapa de seleção.

Na estratégia de recuperação semântica, a pergunta e os comentários coletados são convertidos em representações vetoriais pelo modelo *gemini-embedding-001*. A pergunta é processada como uma consulta de recuperação (*RETRIEVAL_QUERY*), enquanto os comentários são processados como documentos (*RETRIEVAL_DOCUMENT*), em lotes de até 100 itens. Para cada comentário, calcula-se a similaridade de cosseno entre seu vetor e o vetor da pergunta. Os resultados são ordenados de forma decrescente pela similaridade obtida, utilizando-se o número de curtidas como critério de desempate. Não é empregado um limiar mínimo de similaridade, e o cálculo é realizado em memória, sem utilização de banco ou índice vetorial.

Após o ranqueamento, cada estratégia seleciona até 30 comentários para compor seu respectivo contexto. O mesmo limite é adotado nos dois métodos, restringindo a quantidade máxima de comentários disponibilizada à etapa de geração e,

consequentemente, o volume de texto inserido no *prompt*. O valor 30 foi definido como parâmetro de implementação do protótipo, sem que tenha sido realizada uma análise experimental para determinar uma quantidade ótima.

A execução pareada permite preservar, para uma mesma pergunta e um mesmo conjunto de comentários coletados, os resultados produzidos pelas duas estratégias. Essa estrutura possibilita uma comparação posterior entre os métodos; entretanto, tal avaliação não foi realizada neste trabalho e, portanto, não é possível concluir que uma das estratégias produza respostas de maior qualidade que a outra.

3.3 DIAGRAMA DE ENTIDADES-RELACIONAMENTOS

O modelo de dados, apresentado na Figura 3, é composto por duas entidades. A entidade **interacoes** representa cada pergunta processada por um método e reúne:

- identificador;
- um agrupador que vincula o par de uma mesma comparação (léxico e semântico);
- o identificador do vídeo;
- a pergunta;
- a resposta;
- o método de recuperação empregado;
- o modelo de linguagem que efetivamente respondeu;
- o total de comentários recebidos;
- a latência da etapa de seleção;
- a data e hora do registro.

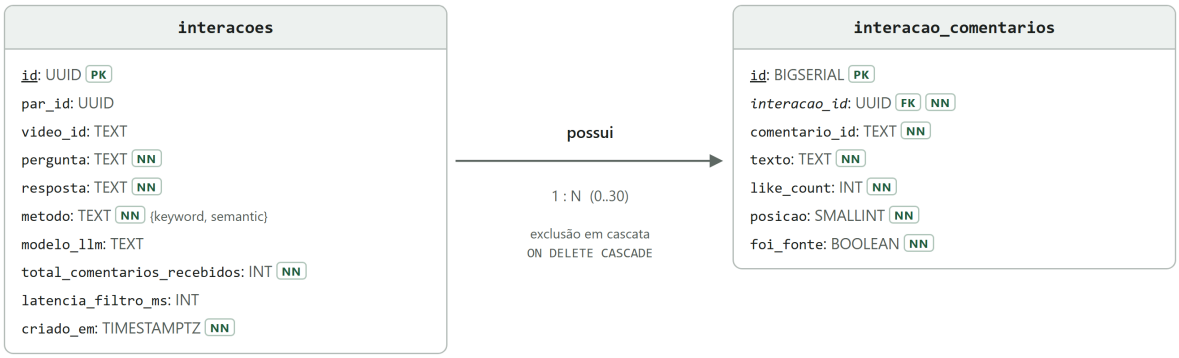
A entidade **interacao_comentarios** representa cada comentário filtrado e reúne:

- identificador;
- referência à interação à qual pertence;
- identificador e texto do comentário;
- contagem de curtidas;
- posição no *ranking* de relevância;

- um indicador de se aquele comentário foi citado como fonte na resposta.

Cada interação relaciona-se a até 30 comentários filtrados, em relacionamento de um-para-muitos com exclusão em cascata. Duas interações que compartilham o mesmo agrupador formam um par de comparação. O modelo não contempla entidades de usuário, sessão ou vídeo canônico: o identificador do vídeo é um atributo simples, e não há deduplicação de comentários entre interações distintas.

Figura 3 – Diagrama de entidades-relacionamentos do CommentLens.



Fonte: Elaborado pelo autor (2026).

Nota: PK indica chave primária; FK, chave estrangeira; e NN, *not null*. Duas linhas de `interacoes` com o mesmo `par_id` formam um par de comparação.

4 SOFTWARE

4.1 IMPLANTAÇÃO

O sistema é composto pela extensão de navegador e por um *backend* implantado na plataforma Vercel. O *backend* disponibiliza as funções *serverless* responsáveis pelas operações que dependem de serviços externos: uma para a coleta de comentários e outra para o processamento das perguntas, cada qual configurada com 256 MB de memória e tempo máximo de execução de 30 segundos. A extensão se comunica com essas funções por meio de requisições HTTPS, mantendo no *backend* as credenciais necessárias ao acesso aos serviços utilizados pela aplicação.

A execução das funções depende de quatro variáveis de ambiente, mantidas fora do código-fonte: as chaves de acesso à Groq, ao serviço de *embeddings* do Google e à YouTube Data API v3, e a URL de conexão com o banco de dados Neon. O esquema do banco é criado e versionado por migrações geradas pelo ORM Drizzle, aplicadas ao Neon.

A extensão encontra-se publicada na Chrome Web Store, o que permite sua instalação diretamente pelo navegador sem a necessidade de carregamento manual dos arquivos do projeto (Google, 2026b). A versão publicada é distribuída em português e disponibiliza ao usuário a interface do CommentLens para análise dos comentários de vídeos do YouTube. A publicação também estabelece um canal oficial de distribuição e atualização da extensão. Dessa forma, a versão distribuída utiliza o *backend* implantado na Vercel para executar as operações que não são realizadas localmente pela extensão.

4.2 TESTES

A suíte automatizada, construída com o *framework* Vitest, contempla diferentes componentes do *backend*, conforme sintetizado no Quadro 3. Ao todo, são 51 testes distribuídos em seis arquivos, todos aprovados na última execução realizada.

Os testes verificam comportamentos dos componentes do *backend* com dependências externas substituídas por *mocks* nos casos em que essas dependências estão presentes. Dessa forma, não constituem uma avaliação da qualidade das respostas produzidas pelo modelo de linguagem, da qualidade dos *embeddings* reais ou da relevância dos comentários recuperados em um conjunto de dados real. Esses aspectos demandam avaliação empírica específica, não realizada nesta etapa do trabalho.

Registra-se que os testes implementados são de natureza unitária, não havendo, neste ciclo, testes automatizados de integração ou de ponta a ponta. O funcionamento integrado da extensão com o *backend* foi verificado manualmente durante o desenvolvimento.

Quadro 3 – Síntese dos testes unitários automatizados do *backend*.

Componente testado	**Arquivo**	**Testes**	**Principais comportamentos verificados**
Recuperação de comentários	`retrieval.test.ts`	10	Recuperação léxica e semântica, ordenação por similaridade, limite de resultados, *fallback* léxico e seleção da estratégia.
Embeddings e similaridade	`embeddings.test.ts`	11	Similaridade de cosseno, divisão em lotes, geração de *embeddings* e propagação de erros.
Endpoint de consulta	`ask.test.ts`	8	Resposta HTTP, persistência, execução pareada e tratamento de falhas parciais ou totais.
Interpretação da saída do modelo	`llm.test.ts`	7	Extração das fontes, validação dos índices e separação entre resposta e marcação de fontes.
Persistência	`persistence.test.ts`	11	Construção e gravação dos registros, associação das fontes, metadados e vínculo entre interações.
Coleta de comentários	`youtube.test.ts`	4	Mapeamento dos comentários, paginação, limite de páginas e tratamento de erros da API.
Total	**6 arquivos**	**51**	

Fonte: Elaborado pelo autor (2026).

No que se refere à avaliação da qualidade das respostas, foi implementada a infraestrutura necessária a uma análise comparativa posterior entre as duas estratégias de recuperação: a execução pareada de ambos os métodos sobre a mesma entrada e a persistência das variáveis que permitem a análise (o método empregado, o modelo de linguagem que respondeu, a latência da etapa de seleção, a posição de cada comentário no *ranking* de relevância e a indicação de quais comentários foram citados como fonte). A execução do experimento em si e a mensuração de métricas específicas de sistemas de recuperação e geração (como *faithfulness* e *answer relevance*) não foram realizadas neste ciclo e constituem trabalho futuro, detalhado no capítulo seguinte.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento do CommentLens, uma extensão de navegador baseada na arquitetura RAG para responder, em linguagem natural, a perguntas sobre comentários de vídeos do YouTube. Os objetivos definidos no escopo de desenvolvimento deste trabalho foram atendidos. A ferramenta realiza a coleta de comentários públicos por meio da API oficial da plataforma, implementa estratégias de recuperação léxica e semântica, integra os componentes de recuperação e geração de respostas e apresenta ao usuário os comentários indicados como fontes da resposta. As interações produzidas pelas duas estratégias são registradas de forma pareada em uma base relacional, constituindo a infraestrutura necessária para análises posteriores. O funcionamento dos componentes do *backend* foi verificado por meio de 51 testes unitários automatizados, todos aprovados na última execução realizada. O uso de um *backend* intermediário contribui para proteger as credenciais dos serviços externos.

Além da ferramenta, o trabalho contribui ao implementar e integrar duas estratégias distintas de recuperação (uma léxica, baseada na correspondência de termos, e outra semântica, baseada em *embeddings* e similaridade de cosseno) e ao construir uma infraestrutura de execução e registro pareados, na qual ambas as estratégias são aplicadas à mesma pergunta e ao mesmo conjunto de comentários coletados. Essa infraestrutura permite armazenar os resultados associados a uma mesma interação para possibilitar sua análise posterior.

Algumas limitações devem ser registradas. A avaliação empírica da qualidade das respostas ainda não foi conduzida, de modo que a comparação entre as duas estratégias permanece como possibilidade estruturada, e não como resultado medido. Essa avaliação também depende da coleta de um volume suficiente de interações para constituir uma base experimental que permita analisar e comparar os resultados das duas estratégias.

No âmbito técnico, a recuperação semântica opera por busca exaustiva em memória, recalculando os *embeddings* dos comentários a cada pergunta, o que implica custo computacional adicional. O método léxico, por basear-se em correspondência de *substrings* sem tratamento de pontuação, acentuação ou palavras vazias, é sensível à forma como a pergunta é redigida. Além disso, os *endpoints* do *backend* não implementam, no estágio atual, autenticação nem limitação de requisições. Quanto à injeção indireta de instruções por meio dos comentários, o sistema adota um tratamento parcial, instruindo o modelo a interpretá-los apenas como dados e a ignorar comandos neles contidos. Essa medida, entretanto, não constitui uma defesa robusta, uma vez que o modelo pode desconsiderar a instrução, permanecendo o fortalecimento dessa proteção como trabalho futuro.

Como trabalhos futuros, destacam-se: a ampliação da coleta de interações para constituir uma base experimental, seguida da avaliação da qualidade das respostas, com a definição de um conjunto de perguntas de referência e a aplicação de métricas próprias de sistemas RAG, permitindo a comparação empírica entre as estratégias de recuperação; a adoção de um índice vetorial persistente, que evite o reprocessamento dos *embeddings* a cada requisição; e o fortalecimento da segurança dos *endpoints* e das defesas contra injeção de instruções.

REFERÊNCIAS

BROWN, T. B. *et al.* **Language Models are Few-Shot Learners**. [*S. l.: s. n.*], 2020. ArXiv preprint. Disponível em: https://arxiv.org/abs/2005.14165. Acesso em: 9 ago. 2026.

DRIZZLE TEAM. **Drizzle ORM — Overview**. [*S. l.: s. n.*], 2026. Documentação do Drizzle ORM. Disponível em: https://orm.drizzle.team/docs/overview. Acesso em: 10 ago. 2026.

FAN, W. *et al.* **A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models**. [*S. l.: s. n.*], 2024. ArXiv preprint. Disponível em: https://arxiv.org/abs/2405.06211. Acesso em: 27 jul. 2026.

GAO, Y. *et al.* **Retrieval-Augmented Generation for Large Language Models: A Survey**. [*S. l.: s. n.*], 2023. ArXiv preprint. Disponível em: https://arxiv.org/abs/2312.10997. Acesso em: 27 jul. 2026.

GOOGLE. **Chrome Extensions**. [*S. l.: s. n.*], 2026a. Documentação para desenvolvedores do Chrome. Disponível em: https://developer.chrome.com/docs/extensions. Acesso em: 10 ago. 2026.

GOOGLE. **CommentLens — Chrome Web Store**. [*S. l.: s. n.*], 2026b. Disponível em: https://chromewebstore.google.com/detail/commentlens/jjdkldmihdplhjibhllpnkfmoiipkmgo. Acesso em: 10 ago. 2026.

GOOGLE. **Embeddings**. [*S. l.: s. n.*], 2026c. Documentação da Gemini API. Disponível em: https://ai.google.dev/gemini-api/docs/embeddings. Acesso em: 8 ago. 2026.

GROQ. **Groq API Reference**. [*S. l.: s. n.*], 2026a. Disponível em: https://console.groq.com/docs/api-reference. Acesso em: 10 ago. 2026.

GROQ. **Qwen3 32B**. [*S. l.: s. n.*], 2026b. Documentação de modelos da Groq. Disponível em: https://console.groq.com/docs/model/qwen3-32b. Acesso em: 10 ago. 2026.

HOSKING, T. *et al.* **Hierarchical Indexing for Retrieval-Augmented Opinion Summarization**. [*S. l.: s. n.*], 2024. ArXiv preprint. Disponível em: https://arxiv.org/abs/2403.00435. Acesso em: 27 jul. 2026.

LEE, Jinhyuk; CHEN, Feiyang; DUA, Sahil; CER, Daniel; SHANBHOGUE, Madhuri; NAIM, Iftekhar; HERNÁNDEZ ÁBREGO, Gustavo; LI, Zhe; CHEN, Kaifeng;

SCHECHTER VERA, Henrique *et al.* **Gemini Embedding: Generalizable Embeddings from Gemini**. [*S. l.: s. n.*], 2025. ArXiv preprint. Disponível em: https://arxiv.org/abs/2503.07891. Acesso em: 9 ago. 2026.

LEWIS, P. *et al.* **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**. [*S. l.: s. n.*], 2020. ArXiv preprint. Disponível em: https://arxiv.org/abs/2005.11401. Acesso em: 9 ago. 2026.

META. **Llama 3.3 Model Card**. [*S. l.: s. n.*], 2024. Disponível em: https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md. Acesso em: 10 ago. 2026.

MICROSOFT. **The TypeScript Handbook: The Basics**. [*S. l.: s. n.*], 2026. Documentação do TypeScript. Disponível em: https://www.typescriptlang.org/docs/handbook/2/basic-types.html. Acesso em: 19 ago. 2026.

MUSLEH, D. A. *et al.* Arabic Sentiment Analysis of YouTube Comments: NLP-based Machine Learning Approaches for Content Evaluation. **Big Data and Cognitive Computing**, v. 7, n. 3, p. 127, 2023. Disponível em: https://www.mdpi.com/2504-2289/7/3/127. Acesso em: 27 jul. 2026.

NEON. **Neon Serverless Driver**. [*S. l.: s. n.*], 2026. Documentação do Neon. Disponível em: https://neon.com/docs/serverless/serverless-driver. Acesso em: 10 ago. 2026.

POKHAREL, R.; BHATTA, D. **Classifying YouTube Comments Based on Sentiment and Type of Sentence**. [*S. l.: s. n.*], 2021. ArXiv preprint. Disponível em: https://arxiv.org/abs/2111.01908. Acesso em: 27 jul. 2026.

SINGH, R. K.; THOMAS, A. A Systematic Literature Review of YouTube Comments Sentiment Analysis: Challenges and Emerging Trends. **ICTACT Journal on Data Science and Machine Learning**, v. 7, n. 1, p. 947–961, 2025. Disponível em: https://ictactjournals.in/ijdsml/ArticleDetails?id=22688. Acesso em: 27 jul. 2026.

VASWANI, A. *et al.* **Attention Is All You Need**. [*S. l.: s. n.*], 2017. ArXiv preprint. Disponível em: https://arxiv.org/abs/1706.03762. Acesso em: 9 ago. 2026.

VERCEL. **Vercel Functions**. [*S. l.: s. n.*], 2026. Documentação da Vercel. Disponível em: https://vercel.com/docs/functions. Acesso em: 19 ago. 2026.

VITEST. **Getting Started**. [*S. l.: s. n.*], 2026. Documentação do Vitest. Disponível em: https://vitest.dev/guide/. Acesso em: 10 ago. 2026.

XIN, Y. *et al.* Improving Topic Modeling Performance on Social Media Through Semantic Relationships Within Biomedical Terminology. **PLOS ONE**, v. 20, n. 2, e0318702, 2025. Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0318702. Acesso em: 27 jul. 2026.

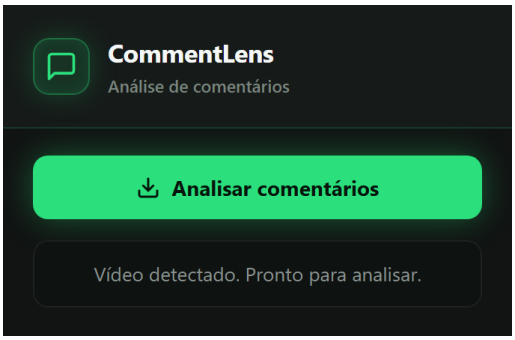
YANG, An; LI, Anfeng; YANG, Baosong; ZHANG, Beichen; HUI, Binyuan; ZHENG, Bo; YU, Bowen *et al.* **Qwen3 Technical Report**. [*S. l.: s. n.*], 2025. ArXiv preprint. Disponível em: https://arxiv.org/abs/2505.09388. Acesso em: 10 ago. 2026.

APÊNDICE A – ESTADOS DA INTERFACE DO COMMENTLENS

A interface da extensão é apresentada em um *popup* de tema escuro e integralmente em português. Ao acessar um vídeo do YouTube, o usuário pode iniciar a coleta dos comentários por meio do botão de análise e acompanhar o andamento da operação pelas mensagens de estado exibidas na própria interface. Após a conclusão da coleta, o campo de pergunta é habilitado, permitindo a formulação de consultas em linguagem natural com limite de 500 caracteres e contador visual.

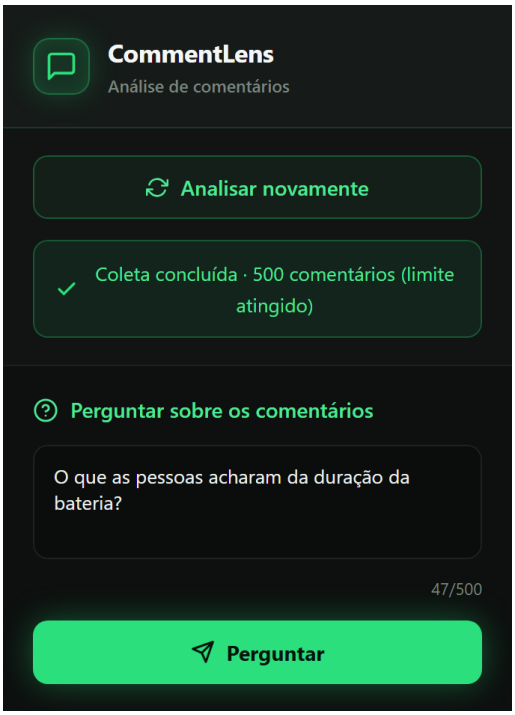
Depois do envio da pergunta, a resposta gerada é apresentada na mesma interface, seguida dos comentários indicados pelo modelo como fontes. Esses comentários são exibidos em cartões separados, permitindo ao usuário consultar o conteúdo textual utilizado como referência na resposta. As Figuras 4, 5 e 6 apresentam, respectivamente, o estado inicial da interface, o estado após a coleta dos comentários e a apresentação da resposta acompanhada dos comentários indicados como fontes.

Figura 4 – Estado inicial do *popup*, antes da coleta.



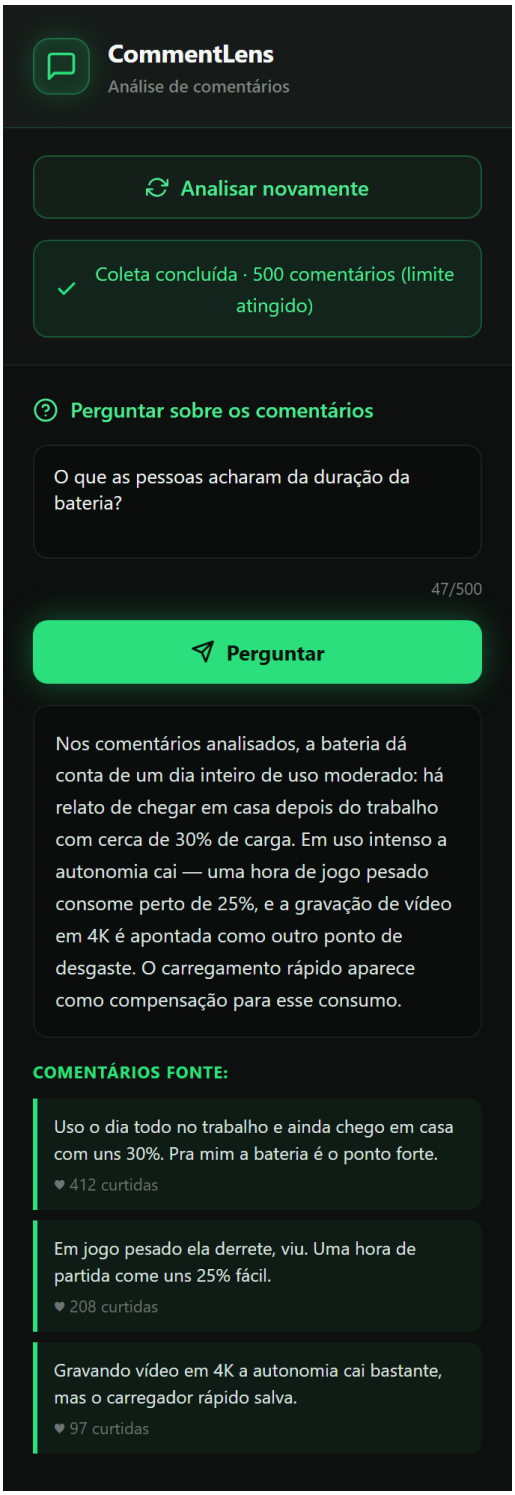
Fonte: Elaborado pelo autor (2026).

Figura 5 – Comentários coletados e campo de pergunta habilitado.



Fonte: Elaborado pelo autor (2026).

Figura 6 – Resposta gerada e comentários indicados como fontes.



Fonte: Elaborado pelo autor (2026).