v7.n12.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

PLATAFORMA DE NOTÍCIAS COM CATEGORIZAÇÃO ATRAVÉS DE INTELIGÊNCIA ARTIFICIAL

NEWS PLATFORM WITH ARTIFICIAL INTELLIGENCE-BASED CATEGORIZATION

PLATAFORMA DE NOTICIAS CON CATEGORIZACIÓN MEDIANTE INTELIGENCIA ARTIFICIAL

Lucas Firmino Cajé[1], André Luiz da Silva[2]

https://doi.org/10.47820/recima21.v7i12.9874

PUBLICADO: 12/2026

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma para leitura de notícias das áreas econômica e política, na qual o conteúdo é categorizado por inteligência artificial. A proposta reúne notícias de diferentes fontes, coletadas por *feeds* RSS e pela NewsAPI, em um ambiente de leitura sem algoritmos de recomendação baseados no comportamento do usuário. A categorização é realizada pelo modelo Llama 3.2, enquanto o modelo nomic-embed-text gera representações vetoriais (*embeddings*) para identificar notícias semanticamente relacionadas. Ambos são executados por meio do Ollama em uma infraestrutura conteinerizada com Docker. Os *embeddings* são armazenados no ChromaDB e utilizados para recuperar as duas notícias com maior proximidade semântica em relação à notícia visualizada. A plataforma foi desenvolvida com Next.js no *front-end* e Node.js no *back-end*. Em uma amostra de 100 notícias, a categoria atribuída pelo modelo coincidiu com a de um avaliador humano em 77% dos casos; o modelo não atribuiu a categoria “outros” a nenhuma notícia e classificou parte das notícias de política como economia. A implementação indica a viabilidade de reunir categorização automática e busca semântica em uma mesma aplicação. Entretanto, a diversidade de fontes não foi controlada nem medida: além de três portais coletados por RSS, a plataforma recebe notícias da NewsAPI, cujo conjunto de veículos varia a cada consulta. Não foi realizada avaliação quantitativa da busca semântica, e os resultados não permitem afirmar melhora na diversidade informacional.

PALAVRAS-CHAVE: Categorização. Engenharia de *prompt*. Inteligência artificial. *Embeddings*. Busca semântica.

ABSTRACT

This paper presents the development of a news platform focused on economics and politics, in which content is categorized using artificial intelligence. The proposal brings together news from different sources, collected through RSS feeds and NewsAPI, in a reading environment without recommendation algorithms based on user behavior. Categorization is performed by the Llama 3.2 model, while nomic-embed-text generates vector representations (embeddings) to identify semantically related news. Both models run through Ollama in an infrastructure containerized with Docker. The embeddings are stored in ChromaDB and used to retrieve the two news items with the highest semantic similarity to the item being viewed.

[1] Graduando em Sistemas de Informação pela Universidade de Araraquara.
[2] Graduado em Engenharia de Computação pela UNIARA, especialista em Licenciatura em Informática pela FATEC, MBA em Gerenciamento de Projetos pela UNIARA, mestre e doutor em Educação Escolar pela Unesp e especialista em TI.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

The platform was developed with Next.js on the front end and Node.js on the back end. In a sample of 100 news items, the category assigned by the model matched that of a human rater in 77% of cases; the model never assigned the "others" category and classified part of the politics news as economics. The implementation indicates the feasibility of combining automatic categorization and semantic search in a single application. However, source diversity was neither controlled nor measured: besides three portals collected via RSS, the platform receives news from NewsAPI, whose set of outlets varies with each query. No quantitative evaluation of the semantic search was conducted, and the results do not allow claims of improvement in information diversity.

KEYWORDS: *Categorization. Prompt engineering. Artificial intelligence. Embeddings. Semantic search.*

RESUMEN
Este trabajo presenta el desarrollo de una plataforma para la lectura de noticias económicas y políticas, en la que el contenido se categoriza mediante inteligencia artificial. La propuesta reúne noticias de distintas fuentes, recopiladas mediante feeds RSS y NewsAPI, en un entorno de lectura sin algoritmos de recomendación basados en el comportamiento del usuario. La categorización se realiza con el modelo Llama 3.2, mientras que nomic-embed-text genera representaciones vectoriales (embeddings) para identificar noticias semánticamente relacionadas. Ambos modelos se ejecutan mediante Ollama en una infraestructura con Docker. Los embeddings se almacenan en ChromaDB y permiten recuperar las dos noticias con mayor proximidad semántica respecto de la noticia consultada. La plataforma utiliza Next.js en el front-end y Node.js en el back-end. En una muestra de 100 noticias, la categoría asignada por el modelo coincidió con la de un evaluador humano en el 77% de los casos; el modelo no asignó la categoría "otros" a ninguna noticia y clasificó parte de las noticias de política como economía. La implementación señala la viabilidad de combinar categorización automática y búsqueda semántica en una aplicación. Sin embargo, la diversidad de fuentes no fue controlada ni medida: además de tres portales recopilados por RSS, la plataforma recibe noticias de NewsAPI, cuyo conjunto de medios varía en cada consulta. No se realizó una evaluación cuantitativa de la búsqueda semántica, y los resultados no permiten afirmar una mejora en la diversidad informativa.

PALABRAS CLAVE: *Categorización. Ingeniería de prompts. Inteligencia artificial. Embeddings. Búsqueda semántica.*

1. INTRODUÇÃO

Com a evolução dos meios de comunicação, especialmente com o avanço da internet e das redes sociais, a disseminação de notícias tornou-se mais rápida e acessível. Segundo o Digital News Report 2026, mais da metade dos brasileiros utiliza redes sociais para se informar semanalmente (CARRO, 2026). Embora plataformas digitais de grande alcance, como Instagram e TikTok, permitam o acesso a conteúdos informativos, elas também são amplamente utilizadas para entretenimento e consumo rápido, não sendo necessariamente concebidas para promover a leitura aprofundada de notícias.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Além disso, essas plataformas utilizam algoritmos de recomendação baseados no comportamento do usuário, priorizando conteúdos alinhados às suas preferências e ao histórico de navegação. Esse funcionamento pode limitar o contato com diferentes perspectivas e com visões divergentes sobre um mesmo tema, especialmente em áreas sensíveis como economia e política.

Diante desse cenário, o problema que orienta este trabalho é: como desenvolver uma plataforma que organize notícias de diferentes fontes e facilite a consulta de conteúdos relacionados, sem personalizar o *feed* com base no comportamento do usuário? Para responder a essa questão, propõe-se o desenvolvimento de uma plataforma de notícias que utiliza inteligência artificial para categorizar e organizar conteúdos de diferentes fontes, oferecendo ao usuário uma experiência de leitura simples e direta. A proposta busca disponibilizar notícias das áreas econômica e política e permitir que o usuário consulte diferentes fontes de forma autônoma. As notícias são coletadas de três portais por RSS e de veículos variados por meio da NewsAPI, cujo conjunto não é controlado pelo projeto; portanto, ainda não é possível afirmar que o conjunto de fontes representa de maneira ampla diferentes posicionamentos ideológicos.

A inteligência artificial é utilizada neste projeto como componente estrutural da plataforma. Em vez de construir manualmente regras de classificação para cada notícia, o sistema utiliza um modelo de linguagem para identificar a seção temática, os assuntos, as entidades, as referências e o sentimento geral do conteúdo. Além disso, utiliza *embeddings* para identificar notícias com conteúdo semanticamente relacionado. Essas funções organizam a navegação, mas não garantem, por si só, a neutralidade das fontes ou a diversidade de perspectivas.

O objetivo geral deste projeto é desenvolver uma plataforma que organize notícias de diferentes fontes e facilite o acesso autônomo a conteúdos relacionados sobre um mesmo tema, sem personalizar o *feed* com base no comportamento do usuário. Especificamente, busca-se:

- Aplicar inteligência artificial para categorização automática de notícias.
- Agregar conteúdo de múltiplas fontes jornalísticas em um único ambiente de leitura.
- Identificar relações semânticas entre os conteúdos.
- Oferecer uma interface de consumo de notícias simples e focada, sem algoritmos de recomendação baseados em comportamento.

A relevância deste trabalho está na contribuição para um consumo mais crítico e autônomo de informações. Ao reunir notícias em um ambiente estruturado e permitir a consulta

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

de conteúdos semanticamente relacionados, a plataforma oferece recursos para que o usuário explore diferentes matérias. A capacidade de ampliar o contato com pontos de vista distintos, contudo, depende também da diversidade das fontes incorporadas, que não foi controlada nem medida nesta versão.

2. REFERENCIAL TEÓRICO

A grande quantidade de notícias disponíveis na internet torna relevante o desenvolvimento de mecanismos que facilitem sua coleta e organização. Os agregadores de notícias reúnem publicações de diferentes fontes em um ambiente centralizado, enquanto os *feeds* RSS (*Really Simple Syndication*) permitem distribuir atualizações em um formato estruturado, que pode ser lido automaticamente por aplicações. Dessa forma, o RSS pode servir como mecanismo de coleta para sistemas agregadores, facilitando o acompanhamento de novas publicações (ALMEIDA, 2026).

Entretanto, reunir notícias não significa necessariamente organizá-las por assunto ou identificar conteúdos relacionados. O Processamento de Linguagem Natural (PLN) oferece técnicas para analisar textos, categorizar informações e identificar elementos relevantes (STRYKER; HOLDSWORTH, 2024). Magalhães e Souza (2019) exploram essa possibilidade ao investigar técnicas de aprendizado de máquina e agrupamento automático de notícias, destacando a influência do pré-processamento e das características textuais sobre os resultados.

Entre as tecnologias relacionadas ao PLN estão os Grandes Modelos de Linguagem (*Large Language Models* - LLMs), treinados com grandes volumes de dados textuais para identificar padrões linguísticos e gerar respostas a partir de instruções. Durante a inferência, o modelo processa o *prompt* e produz uma resposta progressivamente, permitindo executar tarefas como classificação e extração de informações (STRYKER, 2021). No projeto, essa capacidade é utilizada para analisar o título e a descrição das notícias, gerando categorias temáticas, sentimentos, tópicos, entidades e referências.

Para essa finalidade, o projeto utiliza um modelo da família Llama, desenvolvida pela Meta, em um ambiente próprio por meio do *Ollama*. Essa abordagem permite realizar inferências sem depender de uma API externa com cobrança por requisição ou por *tokens* processados. O processamento local, contudo, continua dependendo dos recursos computacionais disponíveis e das condições de licença do modelo utilizado (META, 2024; OLLAMA, s.d.).

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Complementarmente, o sistema utiliza *embeddings* para representar o conteúdo textual em vetores numéricos e armazená-los em uma base vetorial, possibilitando recuperar notícias semanticamente semelhantes. Assim, a proposta integra diferentes etapas: a agregação reúne publicações, o LLM auxilia na categorização e extração de informações, e a busca semântica facilita a recuperação de conteúdos relacionados. Essas funcionalidades contribuem para a organização das notícias sem garantir, por si só, diversidade de perspectivas ou ausência de erros nas classificações.

3. METODOLOGIA

O desenvolvimento da plataforma foi organizado em duas partes principais e interdependentes: o servidor responsável pelo processamento e gerenciamento dos dados, denominado *back-end*, e a interface de interação com o usuário, denominada *front-end*. Ambas as partes foram desenvolvidas utilizando JavaScript. O processo seguiu uma abordagem de desenvolvimento iterativo e incremental, com ciclos de planejamento, implementação e testes.

A escolha por uma única linguagem para ambas as partes do sistema buscou reduzir a complexidade do desenvolvimento e facilitar a manutenção do projeto. O ecossistema JavaScript oferece ferramentas que atendem às necessidades do sistema, desde a coleta de dados até a apresentação das informações ao usuário.

Trata-se de um trabalho de caráter exploratório, apoiado na documentação das ferramentas utilizadas, em pesquisas sobre os temas relacionados e em referências a trabalhos semelhantes. A avaliação combinou a observação do funcionamento da plataforma, quanto à coerência temática, ao enquadramento da categoria e ao preenchimento dos campos, com métricas quantitativas de concordância entre a categoria atribuída pelo modelo e a atribuída por um avaliador humano na categorização das seções.

Para demonstrar os resultados, coletou-se da base de dados uma amostra de 100 notícias, já categorizadas pelo modelo utilizado. A amostra corresponde a cerca de 47% das notícias processadas e categorizadas pela plataforma (até a data da coleta, de um total de 211 notícias) e concentra-se em assuntos de economia e de política relacionados à eleição presidencial no Brasil. As notícias foram apresentadas a um único avaliador (um dos autores), que realizou a mesma tarefa do modelo, atribuindo a cada notícia uma das categorias economia, política ou outros (sem acesso às respostas do modelo). Foi classificada como “outros” a notícia cujo assunto não se enquadra de forma clara em economia nem em política. As categorias atribuídas pelo avaliador foram utilizadas como gabarito.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.1. Estrutura

O *back-end* da plataforma foi desenvolvido com Node.js, utilizando o *framework* Express para a criação das rotas e o gerenciamento das requisições HTTP (NODE.JS, s.d.; EXPRESS, s.d.). A arquitetura interna foi organizada em camadas: *Routes*, que define as rotas de acesso; *Controller*, que recebe as requisições e prepara as respostas da API; *Service*, que concentra as regras de negócio; *Repository*, que realiza a comunicação com o banco de dados; e *Factory*, que instancia e compõe objetos das entidades do sistema. Há também arquivos auxiliares para tarefas específicas. Essa organização busca manter responsabilidades separadas e reduzir o impacto de alterações entre partes do sistema.

A separação de responsabilidades facilita a manutenção e cria uma base para futuras evoluções da plataforma. Embora essa organização favoreça a inclusão de novas funcionalidades, ela não significa, por si só, que a aplicação tenha escalabilidade horizontal implementada; isso dependeria de configuração e infraestrutura adicionais.

O *front-end* foi desenvolvido com Next.js, *framework* baseado em React que oferece recursos de renderização no servidor, otimização de desempenho e roteamento de páginas de forma integrada (NEXT.JS, s.d.; REACT, s.d.). A interface foi projetada com foco no uso em dispositivos móveis, adotando uma abordagem *mobile-first* no desenvolvimento dos componentes visuais.

Para o armazenamento das notícias, a plataforma utiliza o banco de dados NoSQL MongoDB Atlas (MONGODB, s.d.). A escolha se justifica pela natureza dos dados: as notícias possuem estrutura semelhante, mas podem apresentar variações nos campos conforme a fonte. Para os *embeddings* gerados pelo modelo nomic-embed-text, é utilizado o ChromaDB, banco de dados vetorial que permite consultas por similaridade semântica (CHROMA, s.d.).

A conteinerização da aplicação é realizada por meio do Docker, que empacota o *back-end*, o *front-end*, o ambiente de execução dos modelos por meio do Ollama e o ChromaDB em contêineres separados. Essa abordagem ajuda a manter um ambiente consistente entre máquinas e facilita a configuração dos componentes por meio do arquivo docker-compose (DOCKER, s.d.; DOCKER, 2026). A separação em contêineres permite organizar os serviços de forma independente, mas não comprova, isoladamente, escalabilidade horizontal automática.

3.2. Comunicação entre servidor e interface

A comunicação entre as partes da plataforma segue uma arquitetura em camadas. O *front-end*, desenvolvido em Next.js, não realiza chamadas diretas ao servidor Node.js. Em vez

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

disso, utiliza rotas internas de API fornecidas pelo próprio Next.js, localizadas no diretório de rotas da aplicação. Essas rotas atuam como uma camada intermediária entre a interface e o servidor, centralizando o gerenciamento das requisições externas e permitindo que configurações sensíveis, como endereços de servidor e variáveis de ambiente, permaneçam no lado do servidor e não sejam expostas ao navegador do usuário.

As rotas internas do Next.js realizam as chamadas ao servidor Node.js utilizando o protocolo HTTP. O servidor, por sua vez, expõe uma API RESTful desenvolvida com Express, que segue os princípios da arquitetura REST e utiliza os métodos HTTP padrão: GET para consulta de dados, POST para criação de novos registros, PUT para atualização e DELETE para remoção. As respostas são retornadas no formato JSON, garantindo uma estrutura padronizada e de fácil consumo por qualquer cliente que implemente o protocolo HTTP (FIELDING, 2000).

Esse modelo de comunicação em duas etapas (interface para rota interna Next.js, e rota interna para servidor Node.js) oferece maior controle sobre o fluxo de dados da aplicação. Caso o endereço ou a porta do servidor sejam alterados, a mudança é feita apenas na configuração da rota interna, sem impacto nos componentes visuais da interface. Além disso, essa arquitetura facilita a implementação de mecanismos de autenticação e validação de requisições de forma centralizada, antes que os dados cheguem ao servidor principal.

Para o gerenciamento das credenciais e configurações de ambiente, a plataforma utiliza variáveis de ambiente separadas para cada parte do sistema, garantindo que informações como chaves de API, endereços de banco de dados e segredos de autenticação não sejam incluídas diretamente no código-fonte da aplicação.

3.3. Coleta das notícias e extração de informações

A coleta de notícias utiliza a API pública NewsAPI e o protocolo RSS. A NewsAPI permite consultar notícias por meio de requisições HTTP com parâmetros, como assunto, idioma e país, e retorna respostas em JSON (NEWSAPI, s.d.). Na plataforma, a consulta é configurada para retornar notícias do Brasil sobre economia e política em língua portuguesa. O conjunto de veículos retornado varia a cada consulta e não é controlado pelo projeto: o serviço não informa de forma específica quais fontes são utilizadas na busca, e por isso não é possível afirmar quais veículos compõem esse conjunto.

O RSS é um formato de distribuição de informações na web geralmente estruturado em XML. A extração dos *feeds* públicos dos portais é realizada com o auxílio da biblioteca rss-parser do Node.js, que realiza a leitura e o mapeamento dos campos de cada *feed* para um objeto

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

JavaScript (RSS-PARSER, s.d.). As fontes coletadas por RSS foram g1.globo.com, valor.com.br e infomoney.com.br. A quantidade de portais foi delimitada para o desenvolvimento desta versão e para demonstrar o funcionamento da proposta. A arquitetura da plataforma permite integrar novas fontes, desde que seus dados sejam tratados e adaptados à estrutura utilizada pelo sistema. A versão apresentada também recebe veículos variáveis da NewsAPI e, portanto, não deve ser entendida como uma amostra suficiente para representar a diversidade do jornalismo ou de posições ideológicas.

A função de agendamento executa o processo de coleta, limpeza, validação e categorização a cada três horas, atualizando o conteúdo automaticamente, sem intervenção manual.

3.4. Preparação das informações e validações

Por se tratar de um formato de distribuição de conteúdo genérico, o RSS pode apresentar em seus campos informações irrelevantes para o contexto da plataforma, como chamadas para redes sociais, créditos de imagem e seções de links relacionados misturados ao texto da descrição. Para tratar esse problema, foram desenvolvidas funções de sanitização específicas para cada fonte RSS, capazes de identificar padrões de conteúdo irrelevante e descartá-los, preservando apenas as informações jornalísticas relevantes. Os dados da NewsAPI, por já serem estruturados e uniformes, não passam por sanitização: apenas são extraídos os campos utilizados pela plataforma, a saber, título, descrição, endereço da notícia, imagem, data de publicação e nome do veículo de origem (armazenado no campo author). A inclusão de novas fontes exige um processo equivalente de tratamento e padronização, adaptado à estrutura dos dados de origem, para que as informações possam ser processadas e armazenadas corretamente pela plataforma.

Após a limpeza dos dados, a plataforma verifica se a notícia já foi armazenada. Para cada notícia, seja de origem RSS ou NewsAPI, é gerado um identificador por meio do algoritmo de *hash* SHA-256 aplicado ao *slug* do título. O resultado possui 64 caracteres e funciona como identificador derivado do título, conforme o algoritmo SHA-256 (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 2015). Antes de gravar uma notícia, o sistema consulta esse identificador no banco de dados para verificar se ela já foi armazenada, evitando o reprocessamento de títulos já registrados. Essa verificação vale para as duas origens, o que evita o armazenamento duplicado de uma matéria coletada pelos dois caminhos. Esse identificador é diferente do campo _id gerado pelo MongoDB. Como o *hash* depende do *slug* do título, alterações no título podem impedir que matérias relacionadas

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

sejam reconhecidas como duplicadas. A Figura 1 ilustra a estrutura das informações de uma notícia dentro da plataforma. Essa organização é resultado dos processos de limpeza, extração de dados relevantes e persistência no banco de dados.

Figura 1. Demonstração do padrão de registro da notícia na plataforma



```
{
    "_id": {
        "$oid": "6a11c342d71"
    },
    "slug": "aaa-a-aaa-aa-aaa-aaaa",
    "author": "fonte.com",
    "title": "",
    "hash": "0747ed79767fa2242c59c9da462f",
    "description": "",
    "url": "/",
    "urlToImage": "",
    "publishedAt": {
        "$date": "2026-01-24T17:23:41.000Z"
    },
    "tags": {
        "sesao": "aaaa",
        "assuntos": [
            "a",
            "a"
        ],
        "entidades": [
            "a",
            "a",
            "a",
            "a a"
        ],
        "referencias": [
            "a",
            "a"
        ],
        "sentimento": "a"
    },
    "__v": 0
}
```


Fonte: elaboração dos autores.

3.5. Categorização e preenchimento das informações

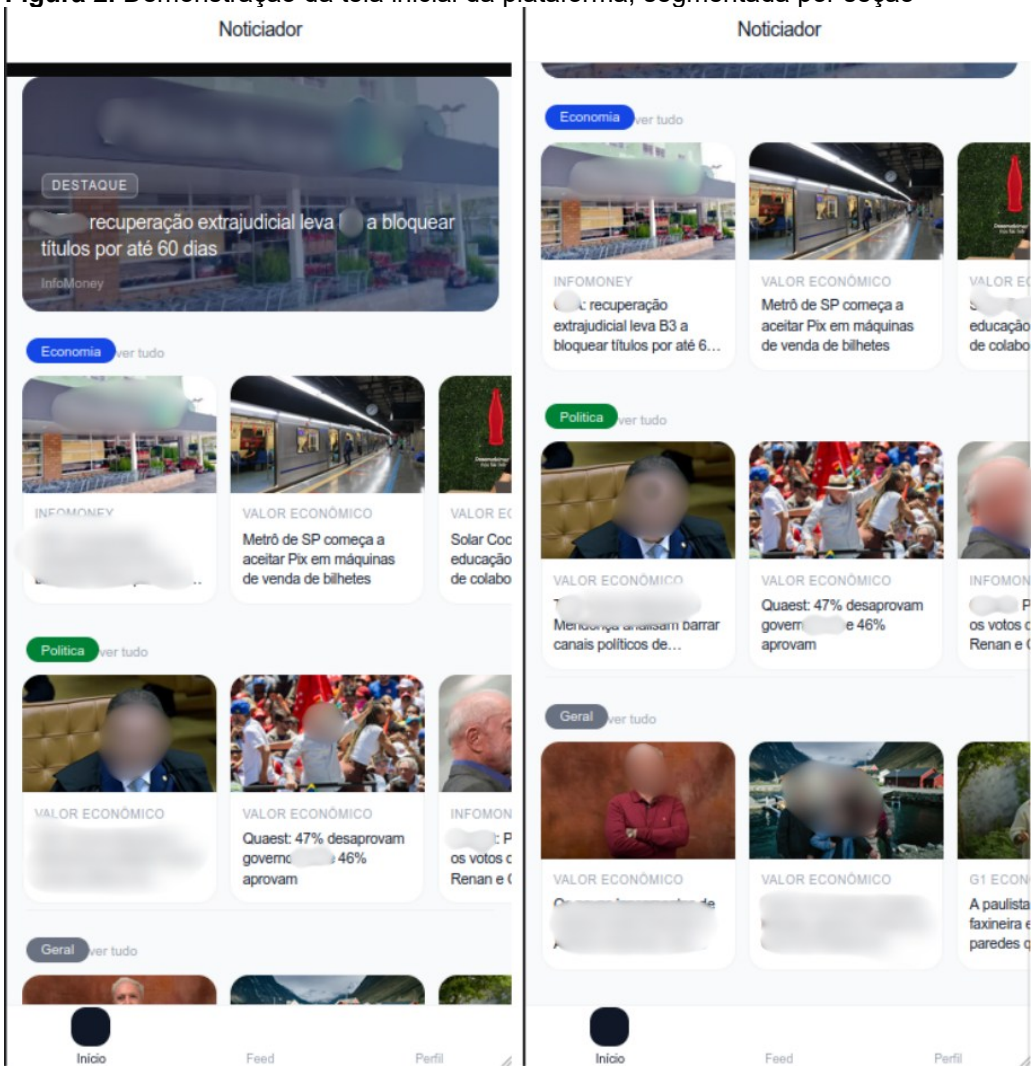
A categorização das notícias é realizada pelo modelo de linguagem Llama 3.2, executado em ambiente próprio por meio da ferramenta Ollama (META, 2024; OLLAMA, s.d.). Para cada notícia validada, o sistema envia ao modelo o título e a descrição do conteúdo, juntamente com um conjunto de instruções estruturadas, denominado *prompt,* que orienta o modelo a retornar um objeto JSON com campos predefinidos: seção, assuntos, entidades, referências e sentimento.

O campo seção classifica a notícia em uma das três categorias temáticas da plataforma: economia, política ou outros. A categoria “outros” é utilizada quando o modelo não associa a notícia a nenhuma das duas categorias principais. O *front-end* organiza o conteúdo de acordo com essa classificação, incluindo a categoria “outros” entre as opções de navegação. Os campos assuntos e entidades identificam, respectivamente, os temas gerais e as pessoas, empresas ou órgãos mencionados. O campo referências registra siglas e termos técnicos, enquanto o campo sentimento classifica o tom geral da notícia como positivo,

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

negativo ou neutro em relação ao assunto abordado. A Figura 2 demonstra como as seções são visualizadas na plataforma. As notícias são agrupadas de acordo com suas respectivas categorias, permitindo que o usuário filtre e visualize apenas o conteúdo referente à seção selecionada.

Figura 2. Demonstração da tela inicial da plataforma, segmentada por seção



Fonte: elaboração dos autores.

3.6. Modelo de LLM e prompt de comando

A escolha por modelos executados em ambiente próprio, por meio do Ollama em contêiner Docker, foi motivada principalmente por custo, controle do ambiente e reprodutibilidade durante o desenvolvimento. O uso de modelos hospedados por serviços externos poderia gerar custos por volume de requisições. A execução em ambiente próprio

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

também permite controlar as versões utilizadas, embora exija recursos computacionais locais e manutenção da infraestrutura.

O refinamento do *prompt* foi conduzido de forma iterativa ao longo do desenvolvimento. Versões iniciais apresentaram problemas como campos vazios, geração de siglas inexistentes e classificações temáticas incorretas. A substituição do modelo qwen2.5:3b pelo llama3.2:3b (ALIBABA CLOUD, 2024; META, 2024), combinada com instruções mais específicas sobre cada campo e exemplos do formato esperado, pareceu melhorar a consistência durante os testes de desenvolvimento. Como não foi realizada uma avaliação quantitativa controlada, essa observação não deve ser interpretada como comprovação de melhora estatisticamente significativa. Também foi implementada uma função de validação que verifica os campos retornados antes do armazenamento e aplica valores de *fallback* quando um campo obrigatório não é retornado no formato esperado.

Para o desenvolvimento do *prompt* de categorização das notícias, foi utilizada a técnica de engenharia de *prompt*, que visa otimizar a eficiência e a precisão das respostas geradas pelo modelo. Essa técnica propõe que a solicitação feita ao modelo de inteligência artificial contenha elementos bem definidos, como a instrução propriamente dita, o contexto da tarefa, os dados de entrada e o formato esperado de saída. Em algumas abordagens, elementos adicionais como cenário ou persona são incorporados para direcionar ainda mais o comportamento do modelo. Esses elementos têm como objetivo orientar e delimitar a geração de respostas, evitando que o modelo produza conteúdo fora do escopo esperado.

O *prompt* apresentado na Figura 3 é utilizado para categorizar as notícias e preencher seus metadados. Sua estrutura apresenta uma persona, a tarefa e o formato de saída esperado, as informações de entrada, o título e descrição, as definições dos campos seção e sentimento, a ordem de preenchimento e as regras para os campos obrigatórios. Essas instruções procuram delimitar a resposta do modelo e orientar a classificação.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 3. Demonstração da estrutura do prompt utilizado para categorizar as notícias

```
content: `
                Você é um editor de seção de um jornal de noticias brasileiras.

                Sua função é categorizar todas as noticias que
                passarem por você seguindo os parametros a baixo.
                Faça um analise do titulo da noticas e sua descrição
                e então retorne um JSON o do exemplo.

                Título: "${titulo}"
                Descrição: "${descricao}"

                Classifique "sesao" usando estas definições:
                - "economia": mercado financeiro, empresas, inflação, juros, PIB, dólar, bolsa, bancos, imposto comércio
                - "politica": governo, congresso, eleições, partidos, ministérios, leis, presidência, seguran   pública
                - "outros": apenas se não for claramente economia nem política

                Classifique "sentimento" como uma avaliação ao que esta sendo dito:
                - "positivo": A noticias esta falando positivamente sobre o tema
                - "negativo": A noticias esta falando negativamente sobre o tema
                - "neutro": A noticias esta apenas informando sobre o tema

                Retorne APENAS este JSON preenchido, sem texto adicional:
                {
                    "sesao": "economia",
                    "assuntos": ["tema1", "tema2"],
                    "entidades": ["entidade1"],
                    "referencias": [],
                    "sentimento": "neutro"
                }

                Regras:
                - "sesao": OBRIGATÓRIO — "economia", "politica" ou "outros"
                - "assuntos": OBRIGATÓRIO — temas específicos da notícia, mínimo 2,
                evite palavras genériccomo "atuação" ou "situação"
                - "entidades": OBRIGATÓRIO — pessoas, empresas, países ou órgãos citados, mínimo 1
                - "referencias": OPCIONAL — somente siglas ou termos técnicos REAIS que aparecem na notícia
                (ex: SELIC, PIB, STF). Se não houver, retorne array vazio []
                - "sentimento": OBRIGATÓRIO — "positivo", "negativo" ou "neutro" `
        }],
```

Fonte: elaboração dos autores.

3.7. Acesso à plataforma

O acesso às notícias é livre, sem necessidade de cadastro ou fornecimento de informações pessoais. Para salvar notícias e acessá-las posteriormente, o usuário pode realizar um registro com nome, e-mail e, opcionalmente, número de telefone, salvos no banco de dados MongoDB. Os dados fornecidos serão utilizados exclusivamente para autenticação na plataforma e para vincular as notícias à conta do usuário. Não é necessária a criação de senha: após o cadastro, um código de cinco dígitos é enviado ao e-mail informado, e sua validação libera o acesso à conta. O código de acesso é armazenado junto às informações do usuário e possui validade de 30 minutos; após esse tempo, é necessário refazer o procedimento de login, solicitando outro código. A sessão é mantida no dispositivo por meio de um *token* JWT (JSON Web *Token*), salvo em *localStorage* (INTERNET ENGINEERING TASK FORCE, 2015; MOZILLA DEVELOPER NETWORK, 2026). O *token*

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

tem validade de um mês após a sua geração; ao expirar, o usuário deverá realizar o login novamente para visualizar suas informações e notícias salvas. O tratamento desses dados deve observar a Lei Geral de Proteção de Dados Pessoais (BRASIL, 2018).

A plataforma ordena as notícias pela data de publicação e registra os identificadores das matérias já visualizadas para evitar repetições no *feed*. Esse registro tem a finalidade de reduzir a exibição repetida de conteúdo; não é utilizado para personalizar as notícias com base nas preferências ou no comportamento do usuário. O histórico é limitado aos cem últimos identificadores de notícias visualizadas e é armazenado localmente no dispositivo do usuário. A cada solicitação de novas notícias, o *front-end* envia os identificadores registrados ao servidor, que os exclui dos resultados. Como o limite é de cem identificadores, notícias antigas podem voltar a aparecer depois que seus identificadores deixam de fazer parte do conjunto mantido. Portanto, o sistema prioriza notícias recentes e reduz repetições, mas não garante que todo conteúdo exibido seja inédito.

3.8. Embeddings e busca semântica

A plataforma implementa um mecanismo de recuperação de notícias relacionadas com base em similaridade semântica, utilizando *embeddings* textuais. Um *embedding* é uma representação vetorial de um texto em um espaço matemático de alta dimensão, no qual textos com significados semelhantes tendem a ocupar posições próximas. Essa abordagem permite identificar relações entre notícias sem depender apenas da correspondência literal de palavras.

Para gerar os *embeddings*, é utilizado o modelo nomic-embed-text, executado por meio do Ollama (NOMIC AI, s.d.; OLLAMA, s.d.). No processamento de cada notícia, o sistema concatena o título e a descrição, gera um vetor e o armazena no ChromaDB. O banco vetorial é executado em um contêiner Docker independente, com volume persistente para manter os dados após a reinicialização do contêiner.

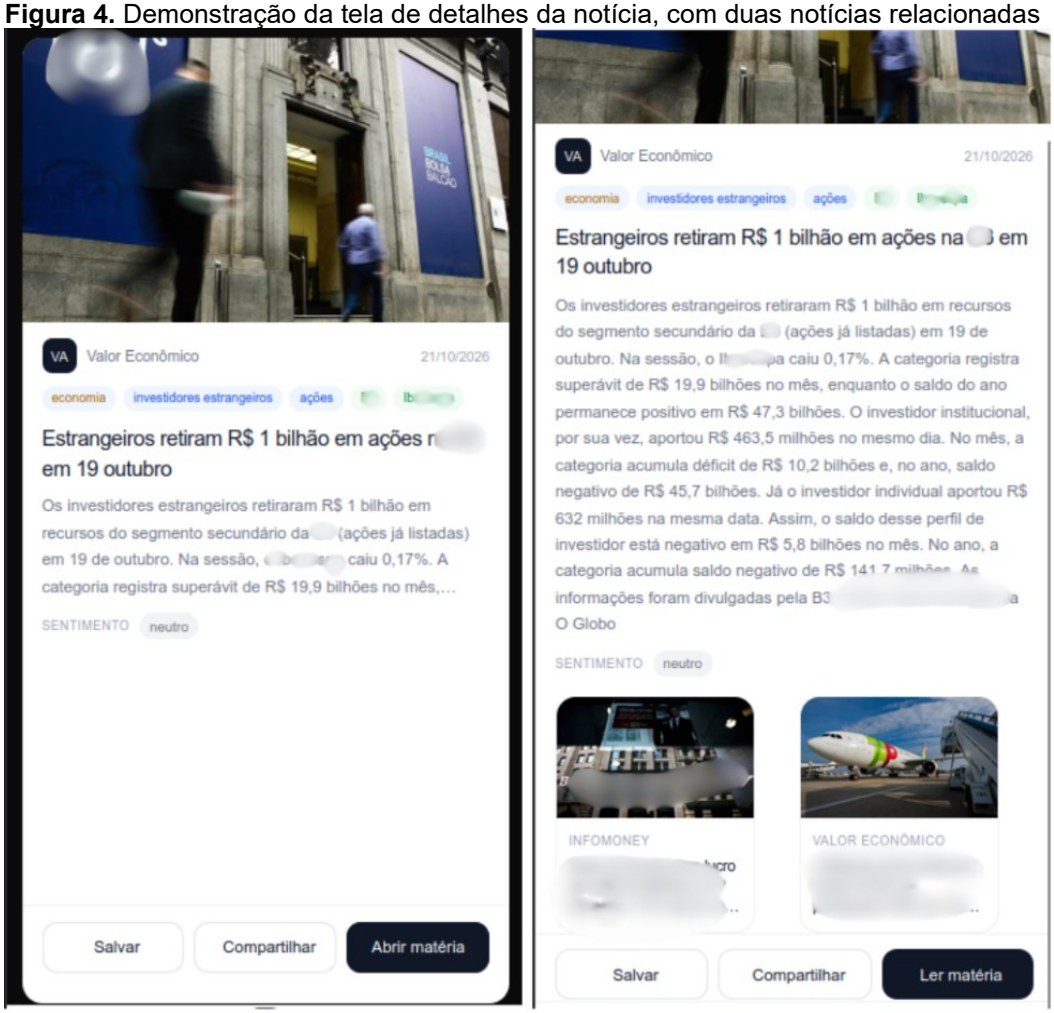
Quando o usuário acessa a tela de detalhe de uma notícia, o sistema utiliza o identificador da notícia para encontrá-la no ChromaDB e acessar seu vetor de *embedding*. A consulta de proximidade utiliza esse vetor para recuperar os identificadores das notícias com maior proximidade semântica no espaço vetorial. Como a consulta parte do vetor da própria notícia, o item mais próximo é ela mesma. Para corrigir isso, a consulta busca três notícias e aplica um filtro que remove aquela cujo identificador é igual ao da notícia principal, restando as duas mais próximas. Esses identificadores são usados para buscar os documentos

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

completos no MongoDB, e as notícias relacionadas são apresentadas abaixo do conteúdo principal.

A Figura 4 apresenta a tela de visualização de uma notícia. Na parte superior, exibe-se a imagem principal, seguida da fonte, da data e das informações geradas pelo modelo (seção, assuntos, entidades e referências), da descrição da notícia e, por fim, das duas notícias relacionadas. Os botões permitem salvar a notícia, compartilhá-la e abrir a matéria na fonte original.

Figura 4. Demonstração da tela de detalhes da notícia, com duas notícias relacionadas



Fonte: elaboração dos autores.

A métrica utilizada é a distância L2 (euclidiana), padrão do ChromaDB. Não há garantia de que as duas notícias relacionadas sejam de fontes jornalísticas diferentes. A ampliação do

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

conjunto de fontes pode aumentar as possibilidades de encontrar conteúdos de diferentes veículos, mas a diversidade de perspectivas dependerá da composição efetiva das fontes incorporadas.

4. RESULTADOS E DISCUSSÃO

A implementação reúne coleta e tratamento de notícias, categorização automática por modelo de linguagem, armazenamento de metadados e recuperação de conteúdos semanticamente relacionados. Durante os testes de desenvolvimento, foram observados casos em que o modelo preenchia campos vazios, criava siglas não presentes no conteúdo ou classificava incorretamente notícias ambíguas. A validação dos campos e o uso de valores de *fallback* ajudam a manter a estrutura dos registros, mas não corrigem necessariamente uma classificação semanticamente incorreta.

O mecanismo de busca semântica utiliza os *embeddings* do título e da descrição, armazenados no ChromaDB durante o processamento das notícias, e consulta o banco quando o usuário abre uma notícia. O sistema apresenta as duas notícias mais próximas encontradas, o que permite explorar conteúdos relacionados sem depender exclusivamente de palavras-chave. Como não foi realizada uma avaliação quantitativa dessa funcionalidade, não é possível afirmar uma taxa de acerto.

A plataforma coleta notícias de três portais por RSS: g1, Valor e InfoMoney, sendo dois deles voltados principalmente à economia, e de veículos variados por meio da NewsAPI, cujo conjunto muda a cada consulta e não foi mapeado neste trabalho. A quantidade de portais definidos diretamente no projeto foi delimitada para o desenvolvimento desta versão, e a arquitetura permite a integração de novas fontes mediante o tratamento e a adequação de seus dados. Ainda assim, o arranjo atual não é suficiente para sustentar a promessa de diversidade ampla de perspectivas, pois a composição e o equilíbrio editorial das fontes não foram controlados nem medidos. O mapeamento dos veículos efetivamente armazenados é uma etapa necessária para investigar essa questão. Também são limitações o possível reaparecimento de notícias antigas após o limite de cem identificadores visualizados e a possibilidade de duplicatas quando o título de uma matéria é alterado.

Tomando as categorias do avaliador como gabarito, a categoria atribuída pelo modelo coincidiu com o gabarito em 77 das 100 notícias (77,0%). O modelo classificou 80 notícias como economia, das quais 59 (73,8%) coincidiram com o gabarito, e 19 como política, das quais 18 (94,7%) coincidiram. Considerando as categorias do gabarito, o modelo reconheceu todas as 59

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

notícias de economia, 18 das 25 de política e nenhuma das 16 de “outros”. Como referência, classificar todas as notícias como economia coincidiria com o gabarito em 59 das 100 notícias (59,0%).

A maior divergência está na categoria “outros”, que o modelo não atribuiu a nenhuma notícia. Das 16 notícias classificadas como “outros” no gabarito, o modelo classificou 14 como economia, 1 como política e 1 sem categoria válida. Entre as 25 notícias de política do gabarito, 7 foram classificadas como economia. Esses resultados indicam que, na amostra, o modelo atribuiu economia ou política mesmo a notícias cujo assunto não se enquadra nessas áreas. Isso sugere que o modelo, mesmo tendo disponível a categoria “outros”, tende a classificar as notícias como “economia” ou “política”, o que pode ser ajustado com melhorias no *prompt* de comando.

Os resultados são indicativos: a amostra é pequena (margem de erro de cerca de 8 pontos percentuais), foi rotulada por um único avaliador e concentra-se em um período e em temas específicos.

Para avaliar a categorização com maior precisão, trabalhos futuros devem ampliar a amostra, incluir mais de um avaliador para medir a concordância entre eles, registrar o uso de *fallback* e refinar a definição da categoria “outros” no *prompt*, reavaliando em seguida os resultados.

5. CONSIDERAÇÕES FINAIS

O trabalho desenvolveu uma plataforma que reúne notícias econômicas e políticas, organiza o conteúdo por meio de inteligência artificial e apresenta matérias semanticamente relacionadas. A combinação de Llama 3.2, nomic-embed-text, Ollama e ChromaDB permite integrar categorização e busca semântica, enquanto a arquitetura baseada em Next.js, Node.js, MongoDB e Docker organiza os componentes da aplicação. Em uma amostra de 100 notícias, a categoria atribuída pelo modelo coincidiu com a do avaliador humano em 77% dos casos: o modelo reconheceu todas as notícias de economia do gabarito e 18 das 25 de política, mas não atribuiu a categoria “outros” a nenhuma notícia.

A proposta oferece uma alternativa de navegação que não utiliza o comportamento do usuário para personalizar o *feed*. Entretanto, não se pode concluir que a plataforma reduz efetivamente o viés informacional, pois isso depende da diversidade das fontes e de uma avaliação específica. A versão descrita combina três portais coletados por RSS com veículos variáveis da NewsAPI, cuja composição não foi mapeada, e a avaliação da categorização contou com um único avaliador e uma amostra pequena. A quantidade de portais definida no projeto

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

não é um limite estrutural: novas fontes podem ser integradas mediante o tratamento e a adequação de seus dados. Como trabalhos futuros, propõe-se ampliar e mapear as fontes coletadas, inclusive as da NewsAPI; ampliar a amostra de avaliação e incluir mais de um avaliador; refinar a definição da categoria “outros”; avaliar métricas de distância alternativas e definir um nível mínimo de proximidade semântica na busca; e melhorar o controle de notícias visualizadas.

REFERÊNCIAS

ALIBABA CLOUD. Qwen: generative AI models. Hangzhou: Alibaba Group, 2024. Disponível em: https://www.alibabacloud.com/en/solutions/generative-ai/qwen. Acesso em: 28 out. 2026.

ALMEIDA, Vanessa. A origem e a importância dos feeds RSS. Revista Prisma, 21 out. 2026. Disponível em: https://revistaprisma.com.br/posts/a-origem-e-a-importancia-dos-feeds-rss/. Acesso em: 29 out. 2026.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Dispõe sobre a proteção de dados pessoais e altera a Lei nº 12.965, de 23 de abril de 2014 (Marco Civil da Internet). Brasília, DF: Presidência da República, [2020]. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 16 nov. 2026.

CARRO, Rodrigo. Brazil. Reuters Institute for the Study of Journalism, 16 jun. 2026. Disponível em: https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2026/brazil. Acesso em: 29 out. 2026.

CHROMA. Chroma: the open-source embedding database. [S. l.: s. n.], [s.d.]. Disponível em: https://www.trychroma.com. Acesso em: 28 out. 2026.

DOCKER. Docker Compose documentation. Docker Docs, 2026. Disponível em: https://docs.docker.com/compose/. Acesso em: 28 out. 2026.

DOCKER. Docker documentation. [S. l.: s. n.], [s.d.]. Disponível em: https://docs.docker.com. Acesso em: 28 out. 2026.

EXPRESS. Express: fast, unopinionated, minimalist web framework for Node.js. [S. l.: s. n.], [s.d.]. Disponível em: https://expressjs.com. Acesso em: 28 out. 2026.

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. Tese (Doctor of Philosophy in Information and Computer Science) – University of California, Irvine, 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.html. Acesso em: 28 out. 2026.

INTERNET ENGINEERING TASK FORCE (IETF). RFC 7519: JSON Web Token (JWT). [S. l.]: IETF, 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 28 out. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

MAGALHÃES, Lúcia Helena de; SOUZA, Renato Rocha. Agrupamento automático de notícias de jornais on-line usando técnicas de Machine Learning para clustering de textos no idioma português. Múltiplos Olhares em Ciência da Informação, v. 9, n. 2, 2019. Disponível em: https://periodicos.ufmg.br/index.php/moci/article/download/19170/16237. Acesso em: 29 out. 2026.

META. Llama 3.2: model cards and prompt formats. Meta AI, 2024. Disponível em: https://dev.meta.ai/llama/docs/model-cards-and-prompt-formats/llama3_2. Acesso em: 28 out. 2026.

MONGODB. MongoDB manual. [S. l.: s. n.], [s.d.]. Disponível em: https://www.mongodb.com/docs/manual. Acesso em: 28 out. 2026.

MOZILLA DEVELOPER NETWORK (MDN). Window.localStorage. MDN Web Docs, 2026. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage. Acesso em: 28 out. 2026.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). FIPS PUB 180-4: Secure Hash Standard (SHS). Gaithersburg: NIST, 2015. Disponível em: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. Acesso em: 28 out. 2026.

NEWSAPI. NewsAPI documentation. [S. l.: s. n.], [s.d.]. Disponível em: https://newsapi.org/docs. Acesso em: 28 out. 2026.

NEXT.JS. Next.js documentation. [S. l.: s. n.], [s.d.]. Disponível em: https://nextjs.org/docs. Acesso em: 28 out. 2026.

NODE.JS. Node.js v23.x documentation. [S. l.: s. n.], [s.d.]. Disponível em: https://nodejs.org/docs/latest-v23.x/api. Acesso em: 28 out. 2026.

NOMIC AI. nomic-embed-text: open-source embedding model. [S. l.: s. n.], [s.d.]. Disponível em: https://ollama.com/library/nomic-embed-text. Acesso em: 28 out. 2026.

OLLAMA. Ollama: run LLMs locally. [S. l.: s. n.], [s.d.]. Disponível em: https://ollama.com. Acesso em: 28 out. 2026.

REACT. React: the library for web and native user interfaces. [S. l.: s. n.], [s.d.]. Disponível em: https://react.dev. Acesso em: 28 out. 2026.

RSS-PARSER. rss-parser: a lightweight RSS parser for Node.js. Versão 3.13.0. npm, [s. d.]. Disponível em: https://www.npmjs.com/package/rss-parser. Acesso em: 23 jul. 2026.

STRYKER, Cole. O que é LLM (grandes modelos de linguagem)? IBM Think, 6 out. 2021. Disponível em: https://www.ibm.com/br-pt/think/topics/large-language-models. Acesso em: 29 out. 2026.

STRYKER, Cole; HOLDSWORTH, Jim. O que é PLN (processamento de linguagem natural)? IBM Think, 11 ago. 2024. Disponível em: https://www.ibm.com/br-pt/think/topics/natural-language-processing. Acesso em: 29 out. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.