# Biblioteca Digital Multilíngue: Arquitetura JAMstack com Automação de Traduções

Alexandre Borges Alves Spindola[1]*; Ariel da Silva Dias[2]

[1] Bacharel em Psicologia. Desenvolvedor Web Full Stack. Rua Thimoteo Rodrigues, 575 – Vila Victor Issler; 99.020-320. Passo Fundo, Rio Grande do Sul, Brasil
[2] 2 Universidade de São Paulo. Mestre em Ciência da Computação e Matemática Computacional. São Paulo, SP, Brasil

*autor correspondente: alexandre.spindola11@gmail.com

## Biblioteca Digital Multilíngue: Arquitetura JAMstack com Automação de Traduções

**Resumo**

O presente trabalho apresenta o desenvolvimento de uma biblioteca digital multilingue, utilizando a arquitetura JAMstack. Esta é composta por uma combinação do Astro com template Starlight para documentação, Strapi como CMS headless e Cloudflare Pages para implementação automática. A solução visa centralizar a geração de conteúdos e as traduções, proporcionando uma interface moderna e intuitiva, com funcionalidades nativas, tais como a alternância entre temas claros e escuros, pesquisa integrada, suporte para 28 idiomas, componentes reutilizáveis e fluxos de trabalho de automatização através do n8n. O protótipo atual já demonstra resultados significativos nas métricas de desempenho e carregamento quase instantâneo em diferentes dispositivos e navegadores.
**Palavras-chave:** JAMstack; Astro; Strapi; Automação de Traduções; n8n.

## Multilingual Digital Library: JAMstack Architecture with Translation Automation

**Abstract**

This paper presents the development of a multilingual digital library using the JAMstack architecture. This is made up of a combination of Astro with the Starlight template for documentation, Strapi as a headless CMS and Cloudflare Pages for automatic deployment. The solution aims to centralize content generation and translations, providing a modern and intuitive interface, with native features such as switching between light and dark themes, integrated search, support for 28 languages, reusable components and automation workflows through n8n. The current prototype already shows significant results in performance metrics and almost instant loading on different devices and browsers.
**Keywords:** JAMstack; Astro; Strapi; Translation Automation; n8n.

### Introdução

O desenvolvimento web moderno passa por transformações significativas com arquiteturas como a JAMstack, que redefinem parâmetros de performance, segurança e escalabilidade. Como evidenciam Tramullas (2020), Nguyen (2022) e Vepsäläinen et al. (2023), essa arquitetura prioriza a geração prévia de conteúdo estático (Static Site Generation, SSG), integração ágil de APIs e distribuição via Content Delivery Network

(CDN). Essas características não apenas aceleram o carregamento de páginas, mas também eliminam a dependência de servidores tradicionais, reduzindo vulnerabilidades e custos operacionais.

Nesse cenário, o framework Astro emerge como uma ferramenta estratégica, combinando agnosticismo tecnológico com otimização de performance. Conforme Hassan (2024), sua capacidade de integrar componentes de frameworks como React, Vue e Svelte — além de sua própria linguagem de componentes — o torna ideal para projetos colaborativos, onde a diversidade de ferramentas é comum. A experiência do desenvolvedor (DX) também é um diferencial: Pääkkänen (2023) destaca a configuração simplificada e a automação de tarefas como fatores que aceleram o ciclo de desenvolvimento. Complementando essa estrutura, o template Starlight agrega funcionalidades críticas para projetos globais, como suporte a 28 idiomas, alternância de temas claro/escuro e busca integrada, garantindo acessibilidade e adaptabilidade a diferentes contextos culturais.

A demanda por soluções multilíngues eficientes, no entanto, impõe desafios complexos, especialmente na gestão de traduções em escala. Kamaluddin et al. (2024) e Telaumbanua et al. (2024) apontam que serviços de tradução automatizada, como o DeepL, têm alcançado resultados notáveis ao preservar nuances contextuais, reduzindo a necessidade de intervenção humana. Para coordenar esses processos, ferramentas de automação como o n8n — destacadas por Pasunuri (2025) — tornam-se essenciais, permitindo a orquestração de fluxos de trabalho entre APIs, CMS e serviços de tradução, garantindo sincronia e consistência em múltiplos idiomas.

Na camada de gerenciamento de conteúdo, o CMS headless Strapi oferece uma base estruturada para organizar dados em tabelas relacionadas (authors, books, categories e chapters), facilitando a manutenção hierárquica de informações. Como observam Sobri et al. (2022) e Rodriguez-Martinez et al. (2024), essa abordagem não apenas centraliza a produção de conteúdo, mas também simplifica sua adaptação para diferentes localizações. O deploy automatizado no Heroku completa esse ecossistema, assegurando que atualizações locais sejam rapidamente replicadas em produção, enquanto integrações com DeepL e n8n automatizam a tradução e a distribuição de conteúdo, minimizando atrasos e erros manuais.

Diante desse contexto, este estudo propõe a criação de uma biblioteca digital multilíngue baseada em Astro e Strapi, combinando as vantagens da JAMstack com automação inteligente. O objetivo é estabelecer uma infraestrutura escalável para centralização de conteúdo e traduções, alinhando-se às exigências contemporâneas por soluções digitais ágeis, inclusivas e tecnologicamente robustas. Ao integrar desempenho,

flexibilidade e automação, o projeto busca não apenas atender demandas imediatas, mas também oferecer um modelo adaptável a futuras expansões

**Metodologia**

A implementação da biblioteca digital multilíngue segue uma metodologia que combina princípios da arquitetura JAMstack com automação de fluxos e gestão modular de conteúdo. Cada etapa foi desenhada para garantir escalabilidade, facilidade de manutenção e alto desempenho, priorizando a integração eficiente entre ferramentas modernas.

O CMS Strapi foi configurado como núcleo central de gerenciamento de dados, organizando informações em entidades relacionadas para estruturar a lógica do projeto. Quatro tabelas principais compõem a base: Authors, que armazena dados como nome e biografia dos autores; Books, responsável por metadados como título, descrição e vinculação ao autor correspondente; Categories, destinada à classificação temática ou por gênero literário; e Chapters, que associa o conteúdo textual aos livros, garantindo uma hierarquia clara. Para sincronizar ambientes de desenvolvimento e produção, o deploy automatizado no Heroku foi implementado, atualizando a instância em produção automaticamente após cada alteração local, o que reduz inconsistências e acelera a iteração.

Na camada de frontend, o framework Astro foi adotado para aproveitar sua capacidade nativa de Geração de Sites Estáticos (SSG), garantindo carregamento rápido e segurança. O template Starlight complementou essa estrutura, oferecendo funcionalidades críticas como suporte a 28 idiomas, alternância entre temas claro/escuro e sistema de busca integrado. Para garantir um design coeso e responsivo, o Tailwind CSS, que já é integrado por padrão no template Starlight, foi utilizado para padronizar o estilo visual em todas as seções da aplicação, assegurando consistência e modernidade na interface. O código-fonte, disponível publicamente no GitHub (https://github.com/alexandrespindola/gnosisapp), reflete a transparência e replicabilidade do projeto.

Um script em TypeScript, denominado generateChapters.ts, foi desenvolvido para automatizar a sincronização entre o CMS e o frontend. Este script realiza uma consulta à API do Strapi, obtém os dados dos capítulos e os converte em arquivos MDX — formato que o Astro processa com eficiência, permitindo a inserção de componentes interativos diretamente no conteúdo. Esse processo não apenas assegura coerência entre as camadas, mas também elimina etapas manuais, otimizando o fluxo de trabalho.

O deploy contínuo no Cloudflare Pages foi configurado para construir e publicar o frontend automaticamente a cada atualização no repositório GitHub. Essa integração

permite que o conteúdo seja distribuído globalmente via CDN, garantindo acesso rápido e seguro independentemente da localização do usuário. A aplicação, disponível em https://gnosisapp.pages.dev, reflete alterações instantaneamente a cada no build, demonstrando a eficácia da abordagem JAMstack na entrega de conteúdo estático.

Para viabilizar as traduções multilíngues, está em fase de planejamento a automação via n8n. O processo envolverá o monitoramento de alterações no Strapi, o envio automático de textos ao serviço DeepL para tradução e a sincronização das versões localizadas com os arquivos MDX existentes. Essa estratégia visa reduzir a intervenção manual, mantendo a consistência do conteúdo em diferentes idiomas sem comprometer a agilidade.

Por fim, a escalabilidade dos builds está sendo priorizada para lidar com o crescimento progressivo de livros e capítulos. Pretende-se implementar pipelines de CI/CD via GitHub Actions, utilizando cache para armazenar resultados de builds anteriores e avaliando a adoção de builds incrementais. Com essa abordagem, apenas os arquivos modificados seriam recompilados, reduzindo tempo de processamento e consumo de recursos, o que é crucial para sustentar a eficiência do projeto em larga escala.

**Resultados Preliminares**

O protótipo em desenvolvimento integra componentes e processos que priorizam desempenho, velocidade de carregamento e escalabilidade estrutural. Na camada de gestão de conteúdo, o CMS headless Strapi foi configurado como núcleo centralizador, organizando dados em tabelas especializadas para autores (authors), livros (books), categorias (categories) e capítulos (chapters). Essa modelagem hierárquica não apenas estrutura as relações entre entidades de forma clara, mas também oferece flexibilidade para expansões futuras, simplificando a manutenção e adaptação do sistema à medida que o acervo cresce.

No frontend, a combinação do framework Astro com o template Starlight otimiza a geração estática de páginas (SSG), aproveitando recursos nativos como suporte a múltiplos idiomas, alternância entre temas claro/escuro e funcionalidade de busca integrada. Para conectar dinamicamente o conteúdo do Strapi à interface, um script em TypeScript (generateChapters.ts) foi implementado: ele consulta a API do CMS, recupera os dados dos capítulos e os converte em arquivos MDX. Essa abordagem garante que o conteúdo seja renderizado estaticamente pelo Astro, preservando a velocidade de carregamento enquanto permite a incorporação de elementos interativos diretamente nos textos.

A entrega global do aplicativo é gerenciada pelo Cloudflare Pages, que combina CDN com pipelines de deploy contínuo. Toda alteração no repositório GitHub dispara automaticamente um novo build, validando e publicando as atualizações no ambiente

provisório (https://gnosisapp.pages.dev) em minutos. Esse fluxo não apenas acelera a disponibilização de novas versões, mas também assegura segurança e distribuição eficiente via rede global de servidores. Testes preliminares em dispositivos desktop e móveis confirmaram uma experiência de usuário fluida, respaldada por métricas do Google PageSpeed Insights: desempenho de 98/100, SEO e práticas recomendadas em 100/100, e acessibilidade em 91/100. Esses resultados evidenciam não apenas a otimização técnica, mas também a eficácia da arquitetura JAMstack aliada a um CMS headless e pipelines automatizados.

Embora oportunidades de refinamento persistam — como a busca por pontuações máximas em todos os critérios —, o protótipo já demonstra como a integração estratégica de tecnologias modernas pode entregar aplicações web rápidas, acessíveis e preparadas para escalar de forma sustentável. A combinação entre renderização estática, gestão de conteúdo modular e automação de fluxos posiciona o projeto como um modelo replicável para soluções digitais que demandem eficiência e adaptabilidade.

## Considerações Finais

Os resultados preliminares evidenciam que a integração entre Astro, Strapi e Cloudflare Pages oferece uma base técnica robusta para a biblioteca digital multilíngue proposta. O Astro, aliado ao template Starlight, mostrou-se eficaz na criação de um site estático com desempenho excepcional — rápido, responsivo e funcional —, enquanto o Strapi atuou como núcleo centralizador para gestão modular e escalável de conteúdo. Complementando esse ecossistema, o deploy automatizado no Cloudflare Pages garantiu atualizações contínuas e distribuição global via CDN, resultando em uma experiência de usuário consistente e acessível. Esses elementos não apenas validam a viabilidade técnica do projeto, mas também destacam sua capacidade de atender demandas críticas de performance e organização de conteúdo em escala.

No entanto, para consolidar a aplicação, aprimoramentos estratégicos são essenciais. A implementação da automação de traduções via n8n e DeepL emerge como prioridade, permitindo que o conteúdo no Strapi seja convertido automaticamente nos 28 idiomas suportados pelo Starlight. Essa etapa reduziria drasticamente a intervenção manual, sincronizando versões originais e traduzidas com precisão — um avanço fundamental para materializar a proposta multilíngue e ampliar o alcance global da plataforma.

Paralelamente, a otimização dos processos de build torna-se urgente à medida que o volume de conteúdo cresce. A introdução de estratégias de cache via GitHub Actions,

combinada com builds incrementais que recompilam apenas arquivos alterados, mitigaria gargalos de tempo e recursos. Essas medidas não apenas acelerariam os pipelines de CI/CD, mas também garantiriam que a escalabilidade não comprometa a eficiência operacional, mantendo a agilidade mesmo com milhares de capítulos de livros.

Embora as métricas do Google PageSpeed Insights (98 em desempenho, 100 em SEO e 91 em acessibilidade) já reflitam uma base técnica sólida, refinamentos adicionais podem elevar a excelência do projeto. Ajustes na semântica HTML, otimização de metadados e aprimoramento de contrastes visuais têm potencial para maximizar as pontuações de acessibilidade e SEO, tornando a aplicação não apenas mais inclusiva, mas também mais visível em mecanismos de busca.

A convergência dessas melhorias posicionaria a biblioteca digital como uma solução moderna e adaptável. Ao harmonizar desempenho, automação e acessibilidade, o projeto transcenderia suas metas iniciais, estabelecendo-se não apenas como uma ferramenta de gestão de conteúdo, mas como um modelo replicável para iniciativas que demandem agilidade, inclusividade e escalabilidade sustentável. A jornada de evolução contínua, portanto, não apenas consolida resultados já alcançados, mas abre caminho para inovações que reforçarão o impacto global da plataforma.

**Referências**

Hassan, J. N. (2024). The Effects of Architectural Design Decisions on Framework Adoption: A Comparative Evaluation of Meta-Frameworks in Modern Web Development. Aalto University, School of Science.

Kamaluddin, M. I. et al. (2024). Analysis of DeepL: Breakthroughs in Machine Translation Technology. Journal of English Education Forum, 4(2), 122-126.

Nguyen, T. (2022). Jamstack: A Modern Solution for E-commerce. Vaasan Ammattikorkeakoulu University of Applied Sciences, School of Technology.

Paakkanen, J. (2023). Upcoming JavaScript web frameworks and their techniques. Bachelor's Thesis, Aalto University, School of Science.

Pasunuri, K. K. (2025). Digital Workflow Automation in Telecommunications: A Systems Integration Approach to Operational Excellence. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 11(1), 20-28.

Rodriguez-Martinez, C., Sanchez-Gomez, A., & Fernandez-Lopez, M. (2024). Soluciones de E-Commerce: Integración de Strapi y Next.js para la creación de una API Adaptable y Personalizable. Revista Ibérica de Sistemas e Tecnologias de Informação, (52), 156-169.

Sobri, M. R. et al. (2022). Headless CMS Adoption in Modern Web Development: A Comparative Study. IEEE Conference on Software and Systems.

Telaumbanua, Y. A. et al. (2024). An Analysis of Two Translation Applications: Why is DeepL Translate more accurate than Google Translate? Journal of Artificial Intelligence and Engineering Applications, 4(1), 82-87.

Tramullas, J. (2020). Elaboración de productos de información con JAMstack: del sistema de gestión de contenidos al web estático. Anuario ThinkEPI, 14, e14f05.

Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2023). The Rise of Disappearing Frameworks in Web Development. Lecture Notes in Computer Science.